WEB TECHNOLOGY AND ITS APPLICATIONS

17CS71

Mr. GANESH D R
ASSISTANT PROFESSOR,
DEPT OF CSE, CITECH

WEB TECHNOLOGY AND ITS APPLICATIONS

[As per Choice Based Credit System (CBCS) scheme]

(Effective from the academic year 2017 - 2018)

SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	**10 Hours**
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	**10 Hours**
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	**10 Hours**
Module – 4	
PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays, $_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	**10 Hours**
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	**10 Hours**

HTML 1: Overview

Chapter 2

Section 1 of 6

HTML DEFINED + ITS HISTORY

Brief History of HTML

- ARPANET of the late 1960s
- Jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.

HTML Syntax

What is a markup language?

HTML is defined as a **markup language**.

- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.

- The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor.

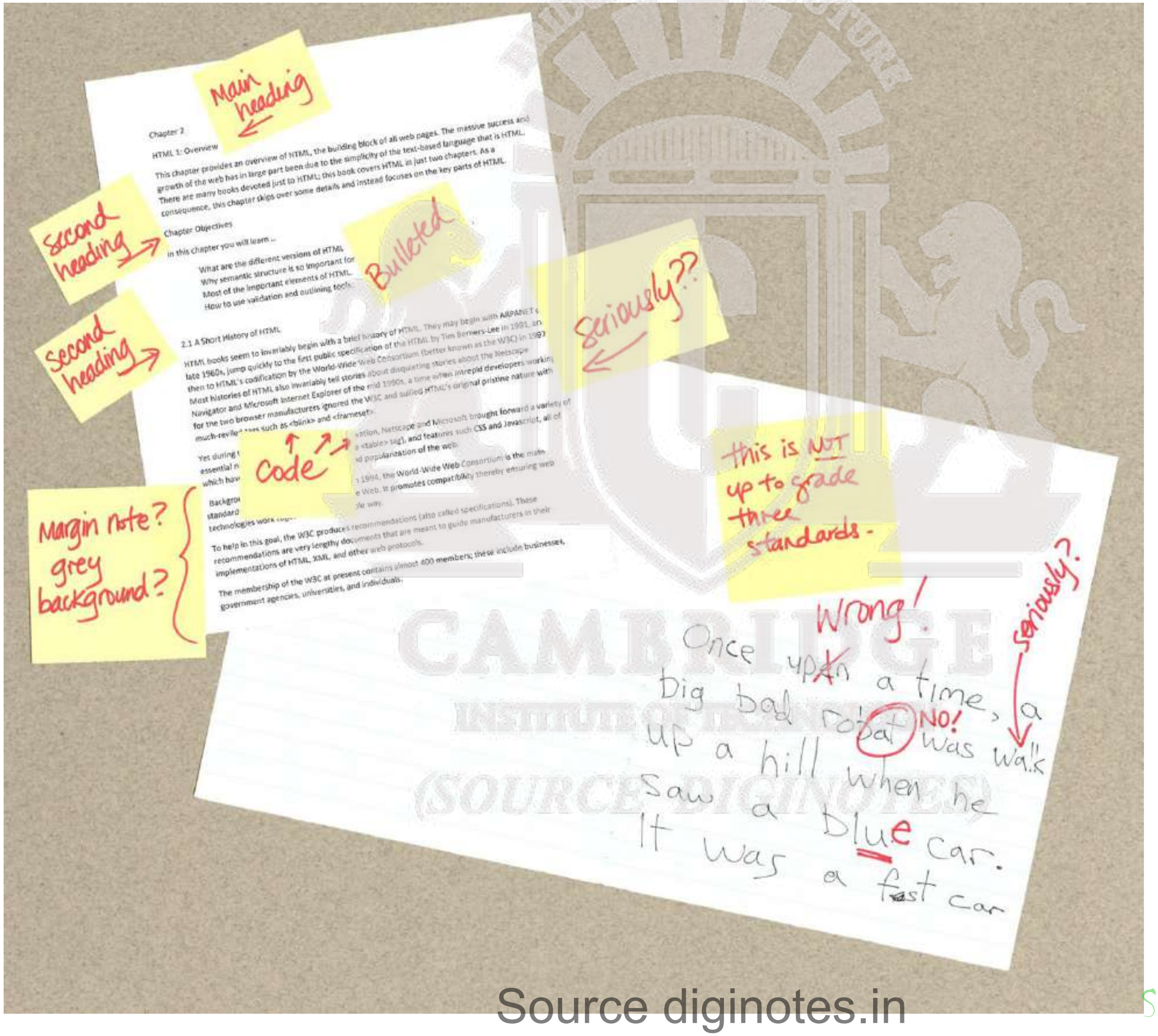
Markup

What is it again?

At its simplest, **markup** is a way to indicate information about the content

- This "information about content" in HTML is implemented via **tags** (aka elements).
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

Sample ad hoc markup



Chapter 2

HTML 1: Overview

This chapter provides an overview of HTML, the building block of all web pages. The massive success and growth of the web has in large part been due to the simplicity of the text-based language that is HTML. There are many books devoted just to HTML; this book covers HTML in just two chapters. As a consequence, this chapter skips over some details and instead focuses on the key parts of HTML.

Chapter Objectives

In this chapter you will learn ...

What are the different versions of HTML
Why semantic structure is so important for
Most of the important elements of HTML
How to use validation and outlining tools

2.1 A Short History of HTML

HTML books seem to invariably begin with a brief history of HTML. They may begin with ARPANET c late 1960s, jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991, an then to HTML's codification by the World-Wide Web Consortium (better known as the W3C) in 1997 Most histories of HTML also invariably tell stories about disquieting stories about the Netscape Navigator and Microsoft Internet Explorer of the mid 1990s, a time when intrepid developers working for the two browser manufactures ignored the W3C and sullied HTML's original pristine nature with much-reviled tags such as <blink> and <frameset>.

Yet during [illegible] vation, Netscape and Microsoft brought forward a variety of essential n[illegible] e <table> tag), and features such CSS and Javascript, all of which hav[illegible] d popularization of the web.

Backgro[illegible] n 1994, the World-Wide Web Consortium is the main standard[illegible] e Web. It promotes compatibility thereby ensuring web technologies work [illegible] le way.

To help in this goal, the W3C produces recommendations (also called specifications). These recommendations are very lengthy documents that are meant to guide manufacturers in their implementations of HTML, XML, and other web protocols.

The membership of the W3C at present contains almost 400 members; these include businesses, government agencies, universities, and individuals.

Once up~~A~~n a time, a
big bad robat was walk
up a hill when he
saw a blue car.
It was a fast car

What is the W3C?

Standards

- The W3C is the main standards organization for the World Wide Web.
- To promotes compatibility the W3C produces **recommendations** (also called **specifications**).
- In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter XML (Extensible Markup Language) syntax rules.

XHTML

Partying like it's 1999

○The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors.

XHTML

You too can be strict

The XML-based syntax rules for XHTML are pretty easy to follow.

The main rules are:

- lowercase tag names,
- attributes always within quotes,
- and all elements must have a closing element (or be self-closing).

XHTML

Two versions

To help web authors, two versions of XHTML were created:

XHTML 1.0 Strict and XHTML 1.0 Transitional.

- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavor of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

Standards Movement

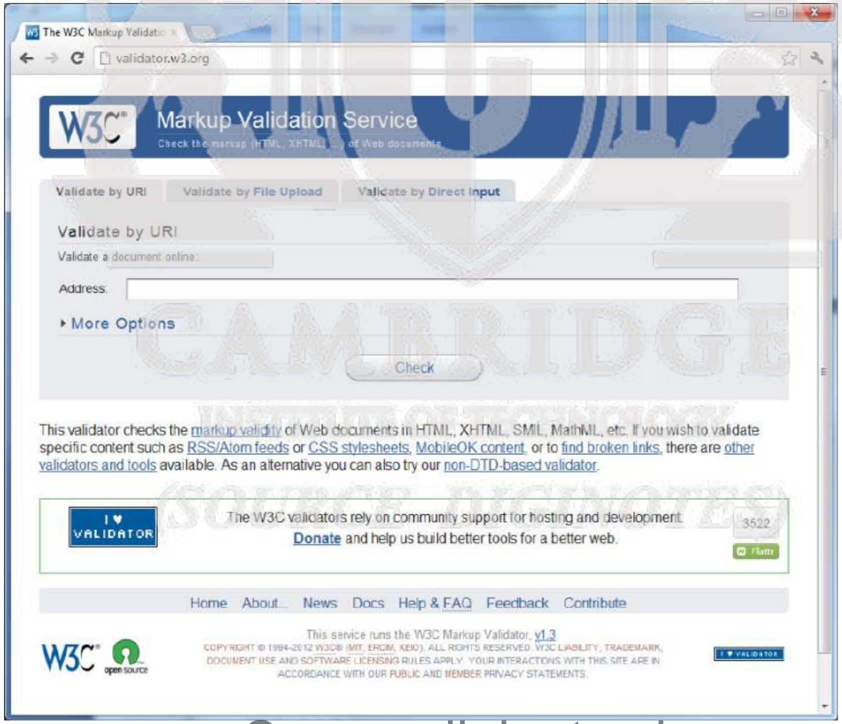
Following a standard is a good thing

- During much of the 2000s, the focus in the professional web development community was on standards: that is, on limiting oneself to the W3C specification for XHTML.

Validators

How to ensure your pages follow a standard

❑A key part of the standards movement in the web development community of the 2000s was the use of **HTML Validators** as a means of verifying that a web page's markup followed the rules for XHTML transitional or strict.



This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

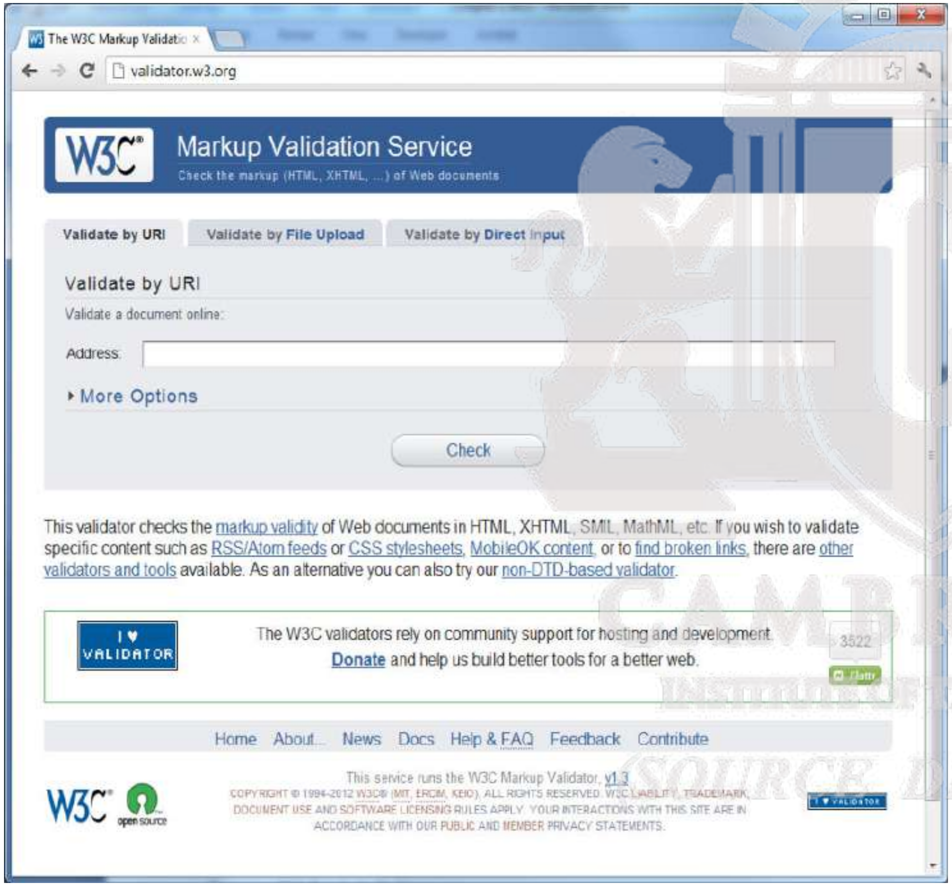
This service runs the W3C Markup Validator, v1.3

COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

Source diginotes.in

How about an example

Only if you have an internet connection



This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators rely on community support for hosting and development. Donate and help us build better tools for a better web.

This service runs the W3C Markup Validator, v1.3

COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

❖Open a web browser to the W3C validator and find a few websites to test.

❖Type the URL into the bar, and you can check if the home page is valid against various standards (or auto-detect)

XHTML 2.0 and WHATWG

Where did it go?

In the mid 2000s, XHTML 2.0 proposed a revolutionary and substantial change to HTML.

- backwards compatibility with HTML and XHTML 1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.

❖ At around the same time, a group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C.

❖ This group was not convinced that the W3C's embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web.

HTML5

The new hotness

By 2009, the W3C stopped work on XHTML 2.0 and instead adopted the work done by WHATWG and named it HTML5.

There are three main aims to HTML5:

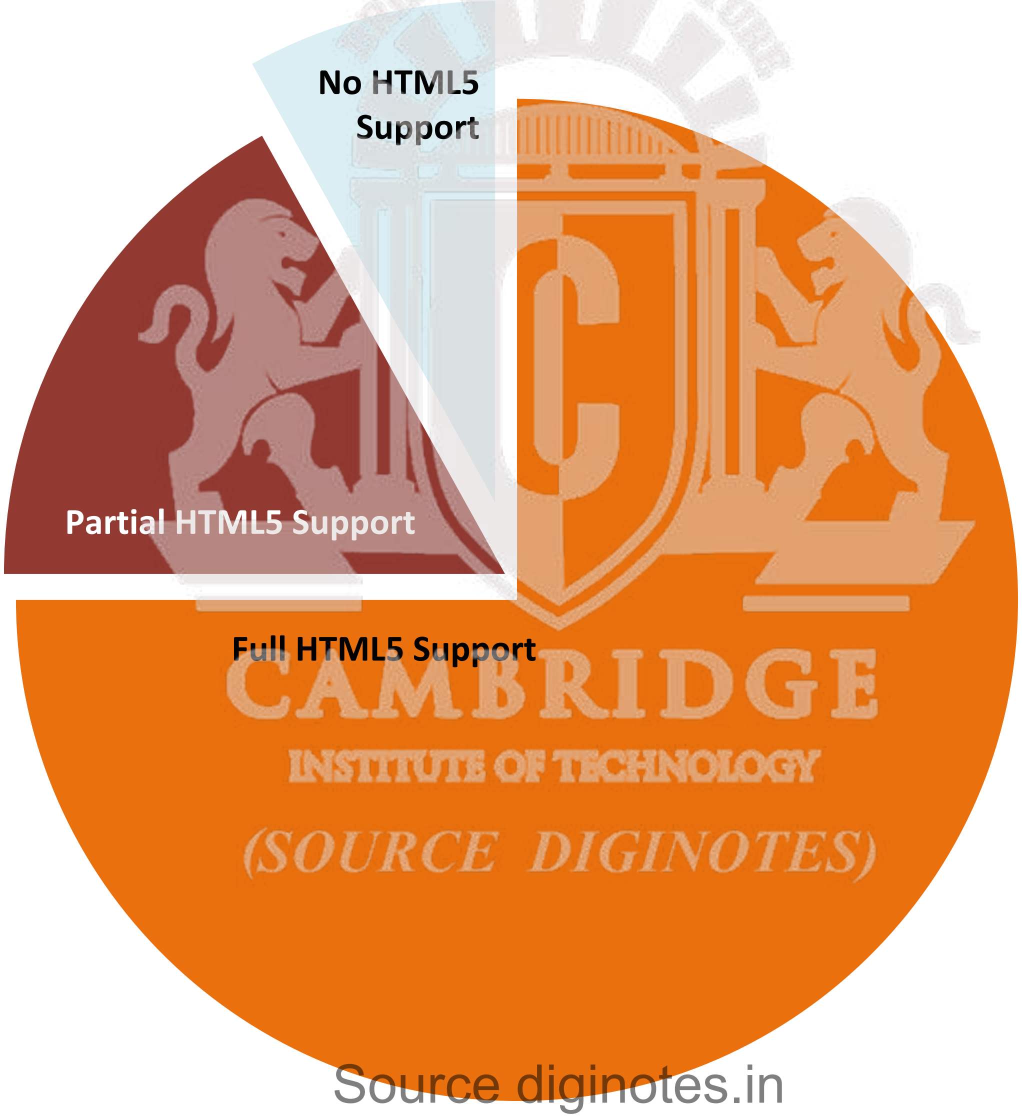
- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via Javascript) for creating rich web applications.
- Be backwards compatible with the existing web.

HTML5

It evolves

- While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.
- Certainly not all browsers and all versions support every feature of HTML5.
- This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and develops over time.
- As such, every browser will support a gradually increasing subset of HTML5 capabilities

HTML5 Support in Browsers



Section 2 of 6

HTML SYNTAX

Elements and Attributes

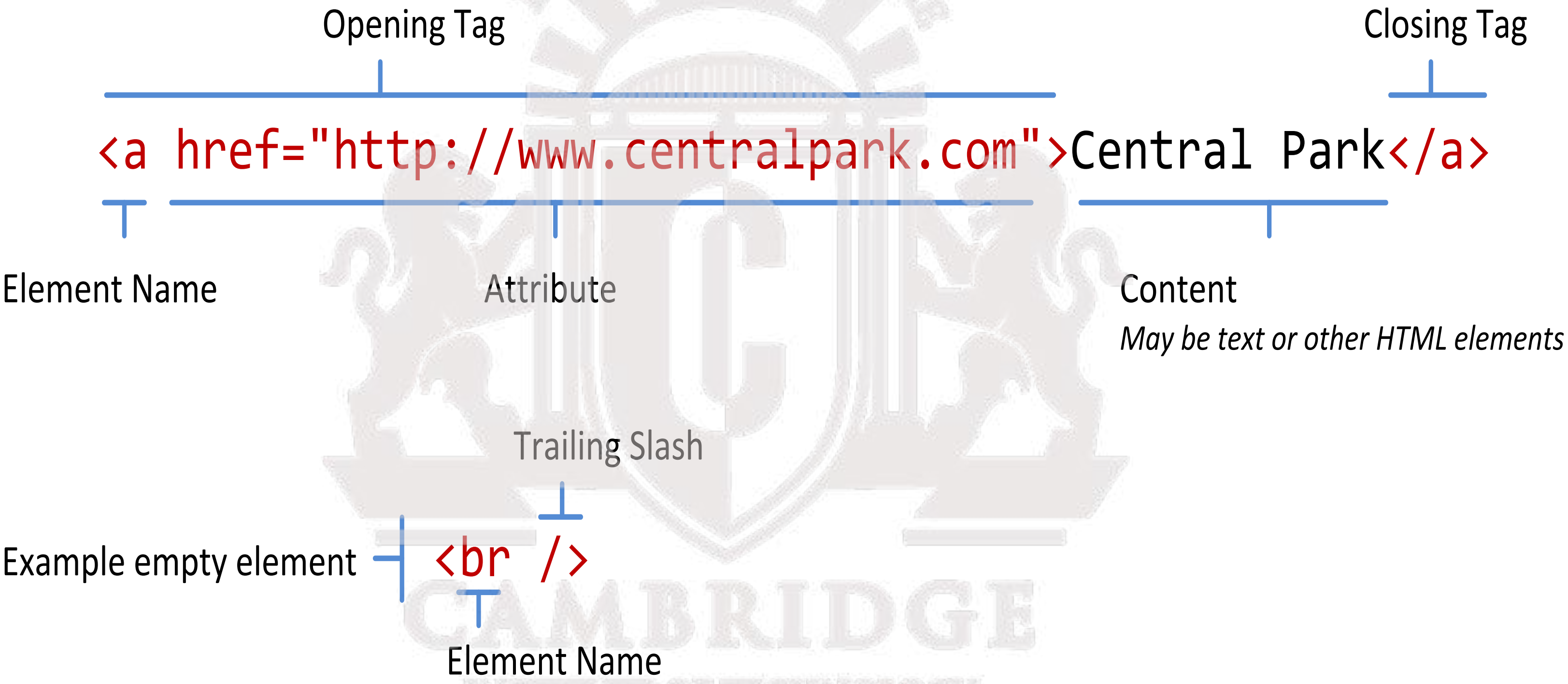
More syntax

- **HTML documents** are composed of textual content and HTML elements.
- An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.
- HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.
- *In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.*

What HTML lets you do

- Insert images using the **<img>** tag
- Create links with the **<a>** tag
- Create lists with the **<ul>**, **<ol>** and **<li>** tags
- Create headings with **<H1>**, **<H2>**, **…**, **<H6>**
- Define metatdata with **<meta>** tag
- And much more…

Elements and Attributes



Opening Tag

Closing Tag

```
<a href="http://www.centralpark.com">Central Park</a>
```

Element Name

Attribute

Content

May be text or other HTML elements

Trailing Slash

Example empty element

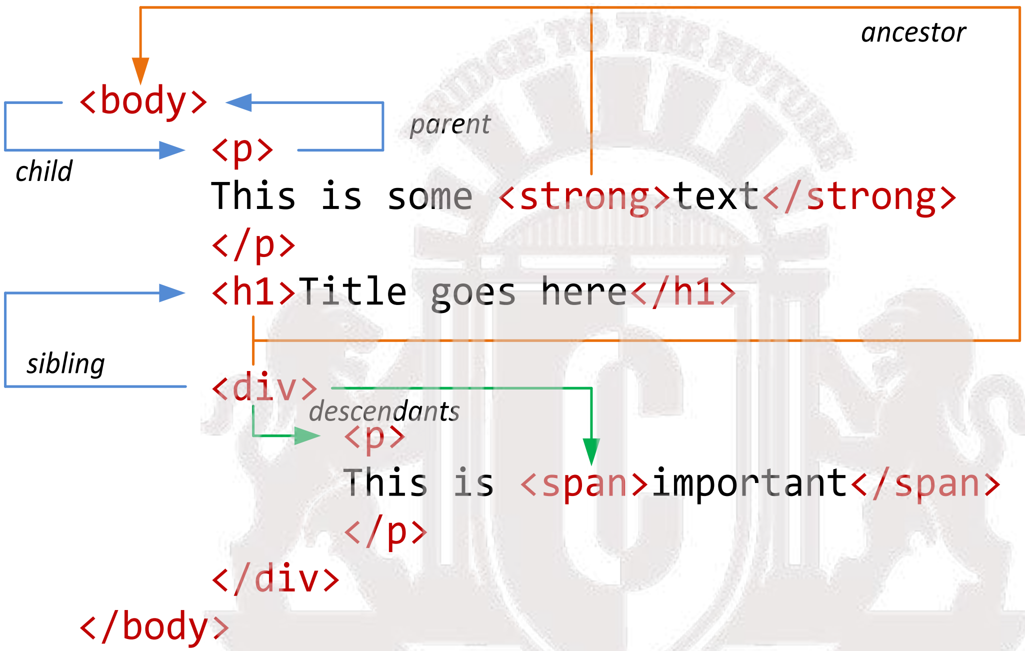
```
<br />
```

Element Name

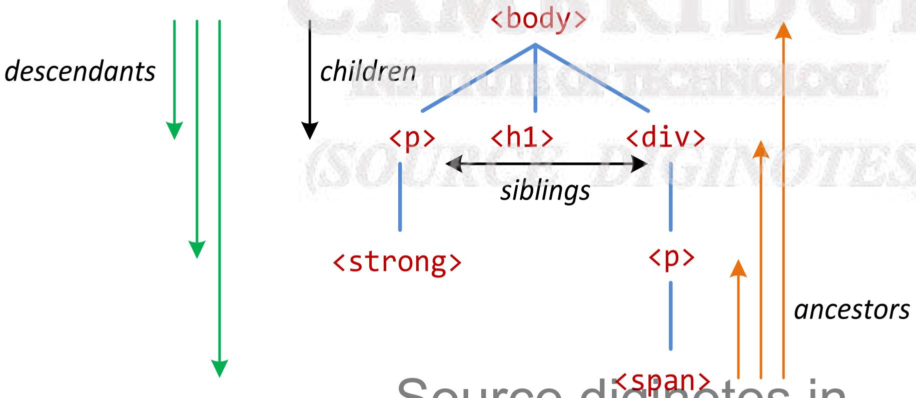
Nesting HTML elements

- ❑ Often an HTML element will contain other HTML elements.
- ❑ In such a case, the container element is said to be a parent of the contained, or child, element.
- ❑ Any elements contained within the child are said to be **descendents** of the parent element; likewise, any given child element, may have a variety of **ancestors**.

Hierarchy of elements



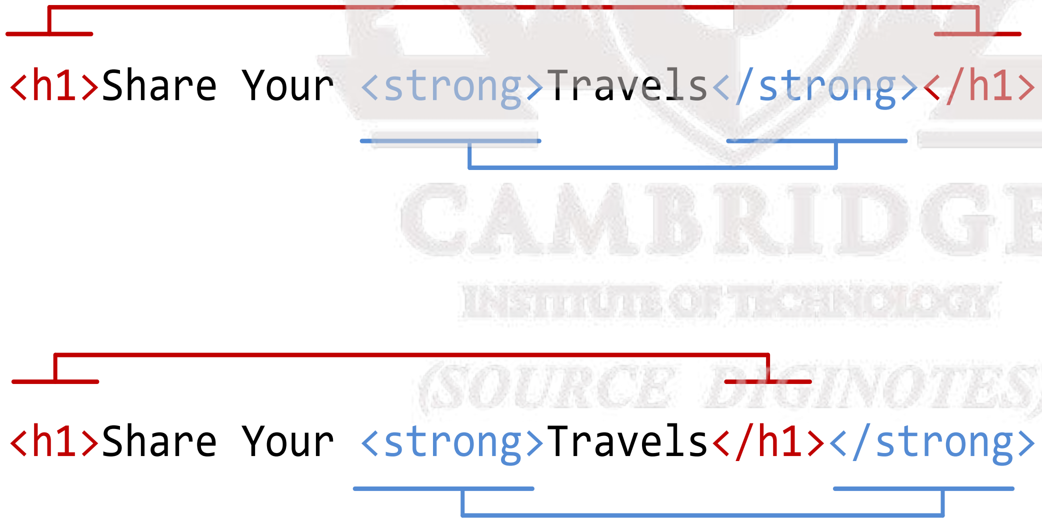
```
<body>
    <p>
    This is some <strong>text</strong>
    </p>
    <h1>Title goes here</h1>

    <div>
        <p>
        This is <span>important</span>
        </p>
    </div>
</body>
```



Nesting HTML elements

- In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

- That is, a child’s ending tag must occur before its parent’s ending tag.



Correct Nesting

```
<h1>Share Your <strong>Travels</strong></h1>
```

```
<h1>Share Your <strong>Travels</h1></strong>
```

Incorrect Nesting

Section 3 of 6

SEMANTIC MARKUP

Semantic Markup

What does it mean?

- ❑ Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.
- ❑ Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).
- ❑ As a consequence, beginning HTML authors are often counseled to create **semantic HTML** documents.
- ❑ That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Structure

- Structure is a vital way of communicating information in paper and electronic documents.
- All of the tags that we will examine in this presentation are used to describe the basic structural information in a document, such as articles, headings, lists, paragraphs, links, images, navigation, footers, and so on.

Semantic Markup

Its advantages

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

- **Maintainability**. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.
- **Faster**. Semantic web pages are typically quicker to author and faster to download.
- **Accessibility**. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.
- **Search engine optimization**. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

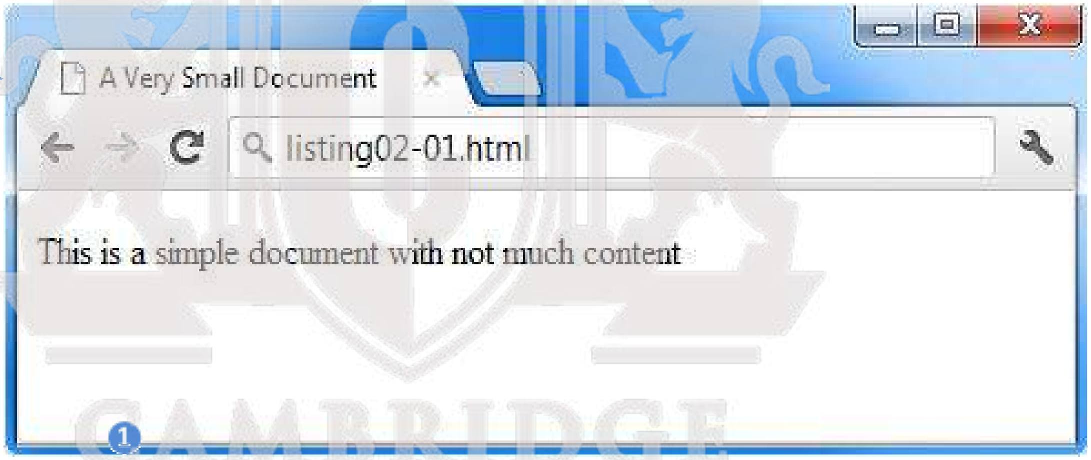
Section 4 of 6

STRUCTURE OF HTML

Simplest HTML document

1

```
<!DOCTYPE html>
<title>A Very Small Document</title>
<p>This is a simple document with not much content</p>
```



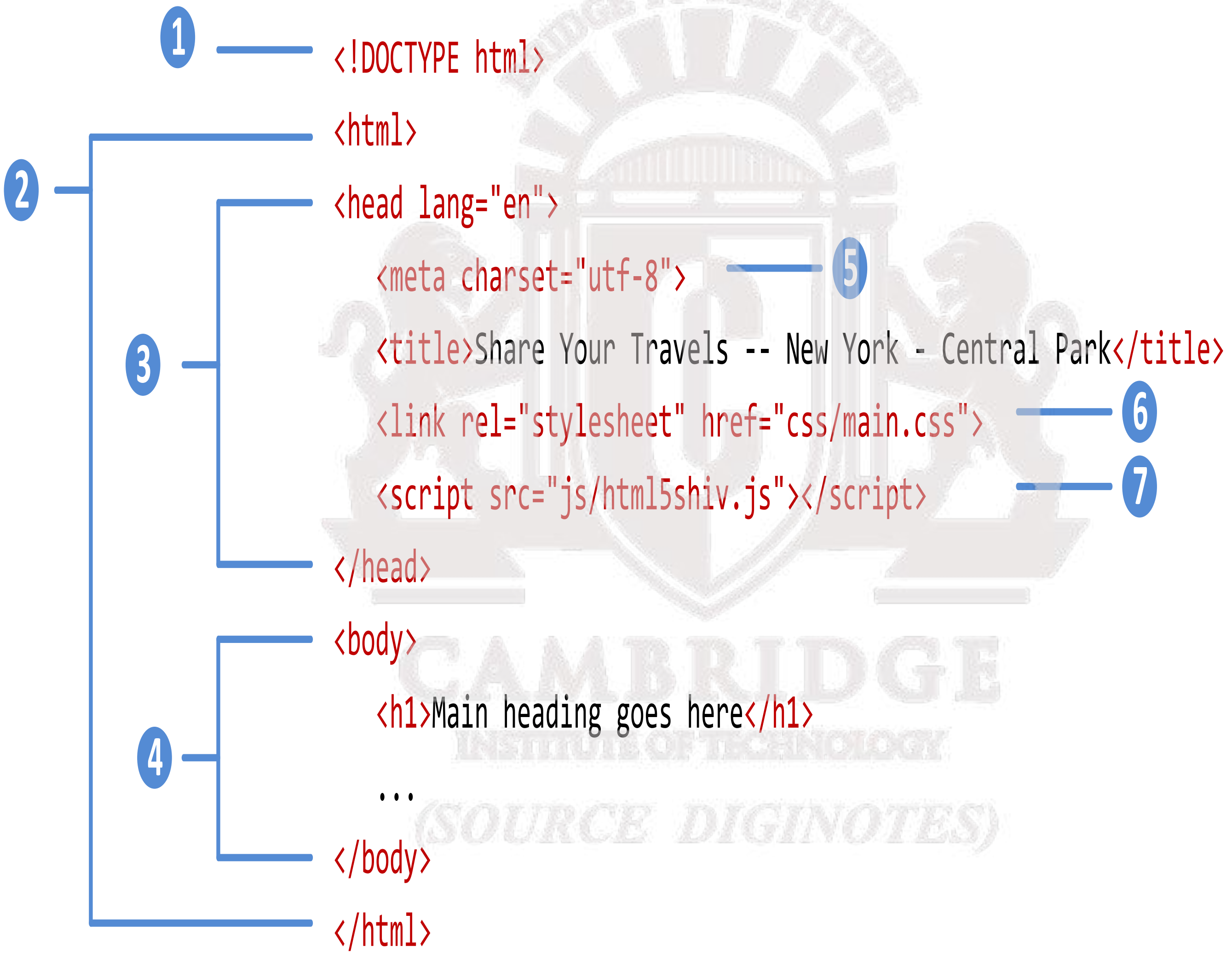
A Very Small Document

listing02-01.html

This is a simple document with not much content

➢The <title> element (Item) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

A more complete document



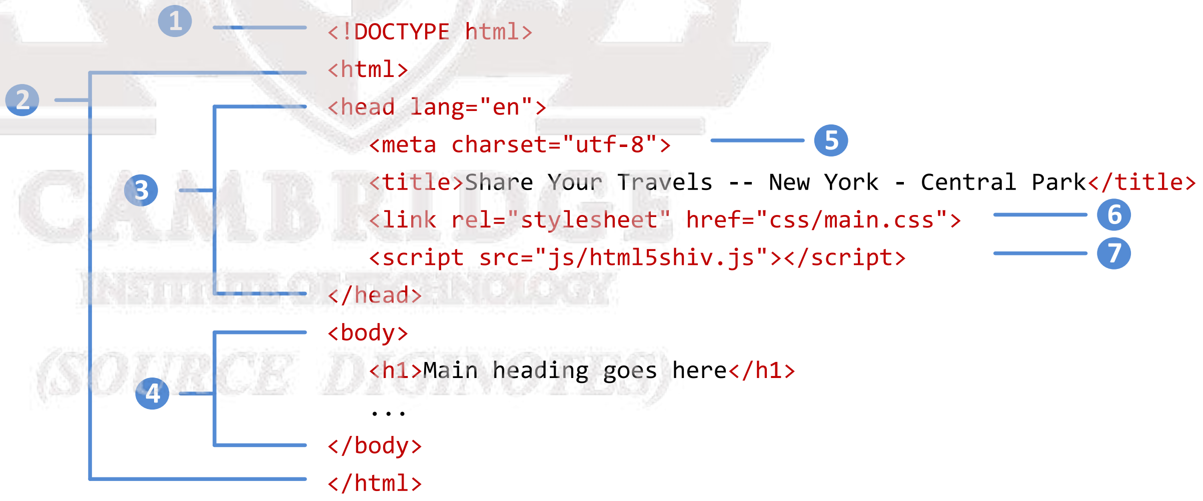
```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="utf-8">
    <title>Share Your Travels -- New York - Central Park</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/html5shiv.js"></script>
</head>
<body>
    <h1>Main heading goes here</h1>
    ...
</body>
</html>
```

1

DOCTYPE

(short for **Document Type Definition**)

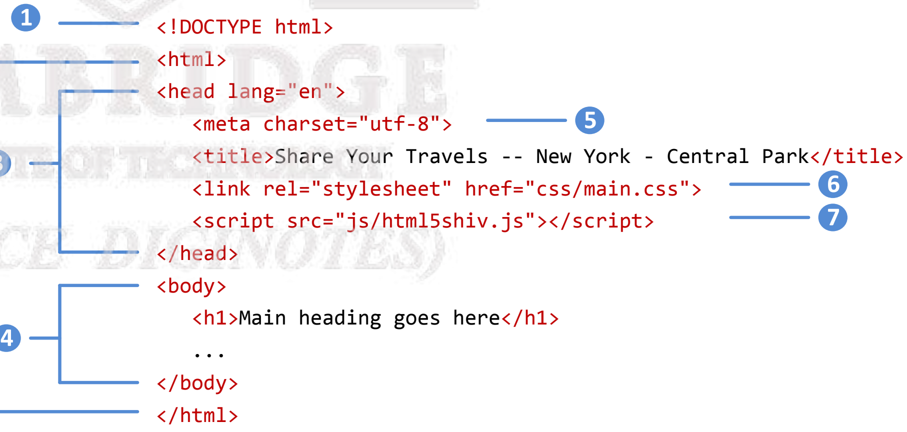
- Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.
- Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

HTML, Head, and Body

- HTML5 does not require the use of the <html>, <head>, and <body>.
- However, in XHTML they were required, and most web authors continue to use them.
- 2 The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

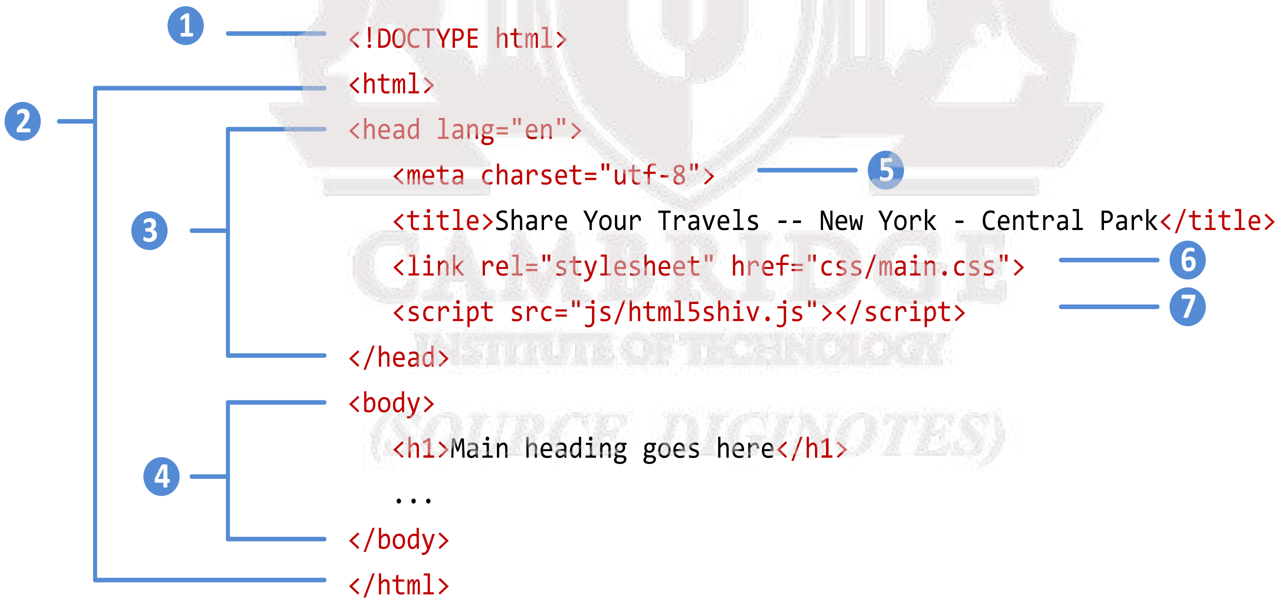
Head and Body

- HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.

3

- The head contains descriptive elements *about* the document.

4
- The body contains content that will be displayed by the browser.



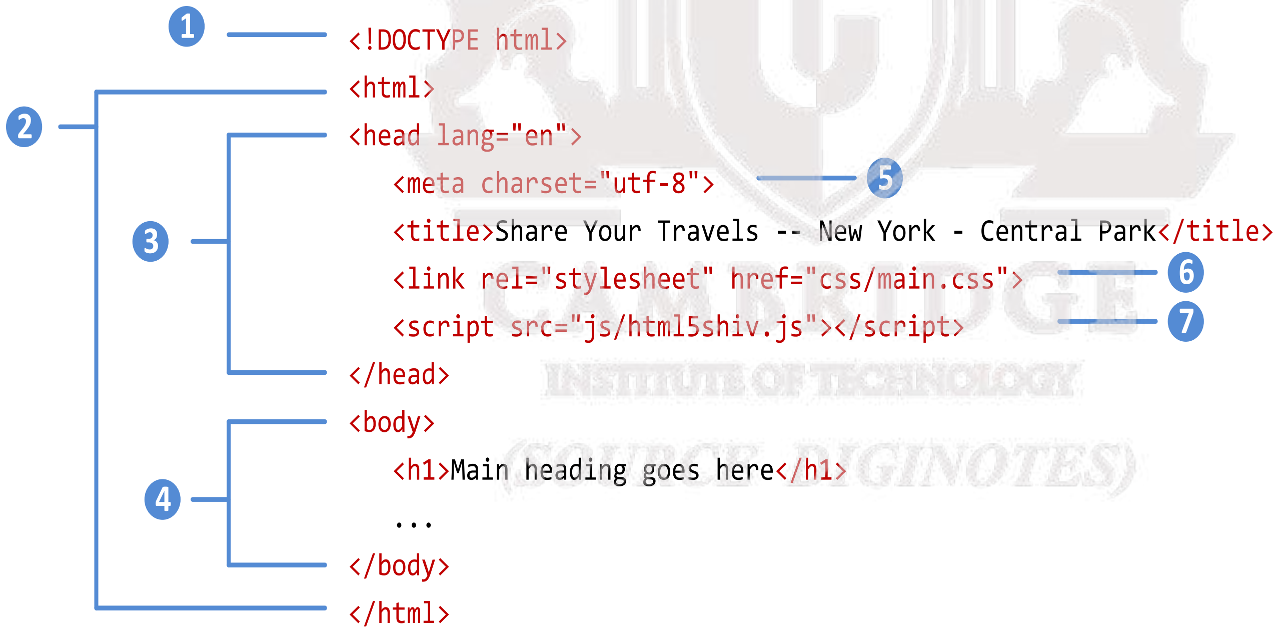
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

Inside the head

There are no brains

➢You will notice that the <head> element contains a variety of additional elements.

5 ➢The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.



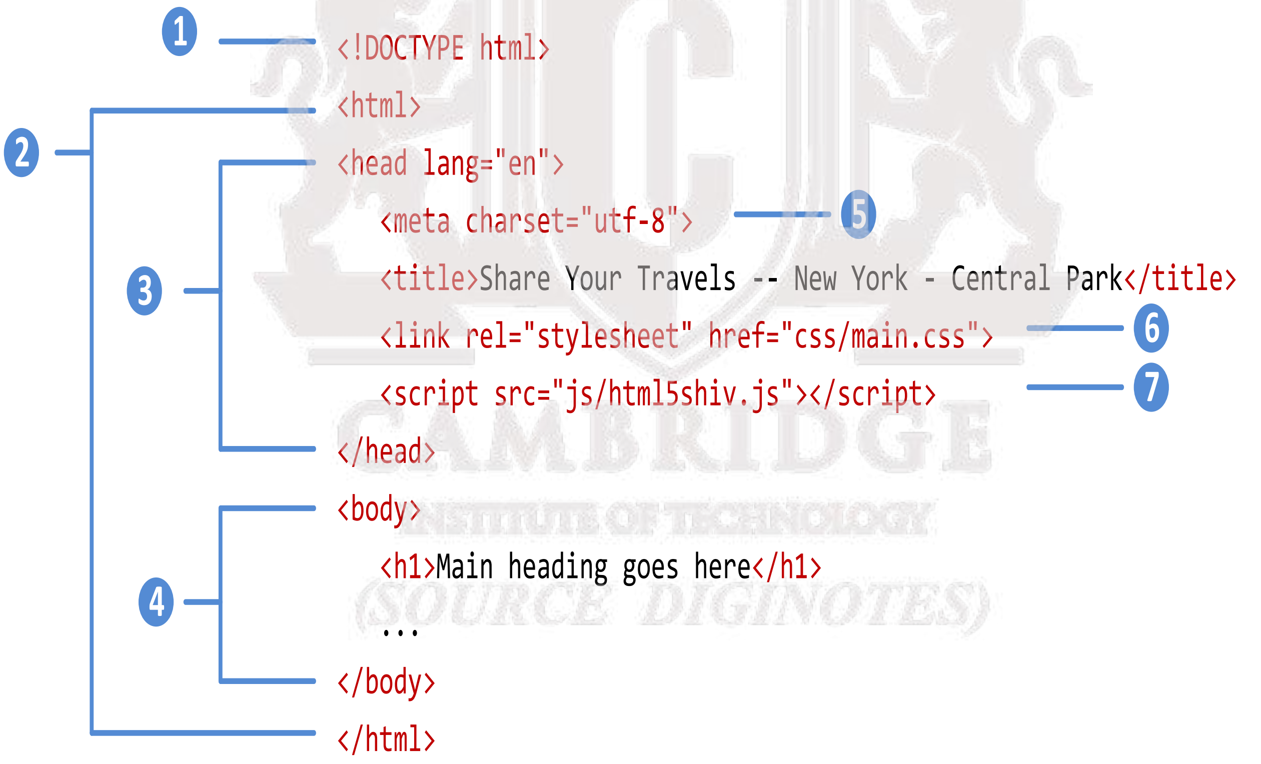
```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

Inside the head

No brains but metas, styles and javascripts

6 Our example specifies an external CSS style sheet file that is used with this document.

7 It also references an external Javascript file.



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

Section 5 of 6

QUICK TOUR OF HTML

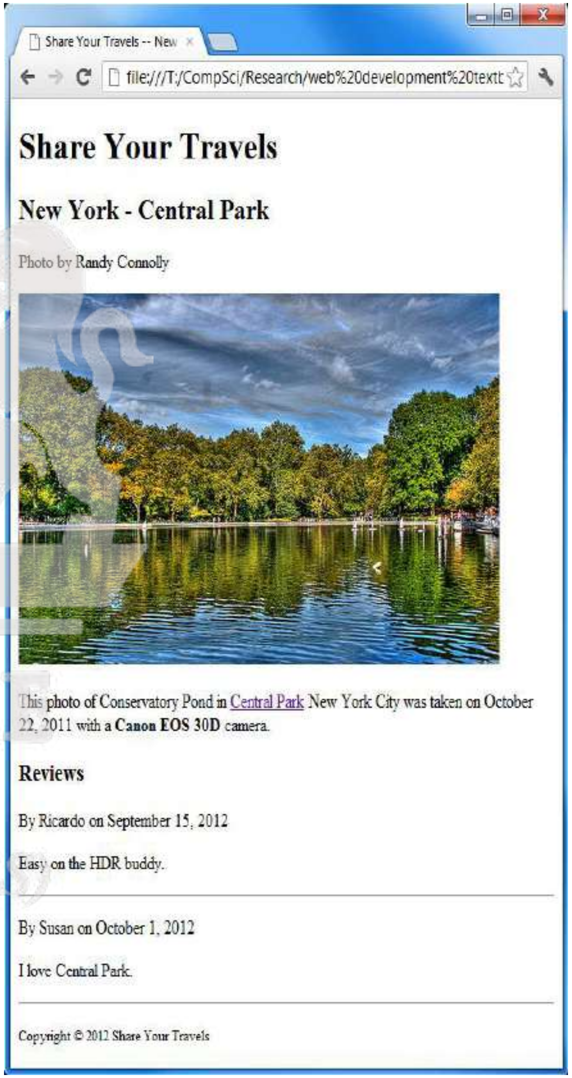
Why a quick tour?

- HTML5 contains many structural and presentation elements – too many to completely cover in this presentation.

- Rather than comprehensively cover all these elements, this presentation will provide a quick overview of the most common elements.

Sample Document

```
   <body>
      <h1>Share Your Travels</h1>
1     <h2>New York - Central Park</h2>
2     <p>Photo by Randy Connolly</p>
      <p>This photo of Conservatory Pond in
         <a href="http://www.centralpark.com/">Central Park</a>     3
         New York City was taken on October 22, 2011 with a
         <strong>Canon EOS 30D</strong> camera.
      </p>                                                4
5     <img src="images/central-park.jpg" alt="Central Park" />

      <h3>Reviews</h3>
6     <div>                                                       7
         <p>By Ricardo on <time>September 15, 2012</time></p>
         <p>Easy on the HDR buddy.</p>
      </div>

      <div>
         <p>By Susan on <time>October 1, 2012</time></p>
         <p>I love Central Park.</p>
      </div>   8

      <p><small>Copyright &copy; 2012 Share Your Travels</small></p>
   </body>           9
```


Share Your Travels -- New
file:///T:/CompSci/Research/web%20development%20textt

Share Your Travels

New York - Central Park

Photo by Randy Connolly

This photo of Conservatory Pond in Central Park New York City was taken on October 22, 2011 with a **Canon EOS 30D** camera.

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

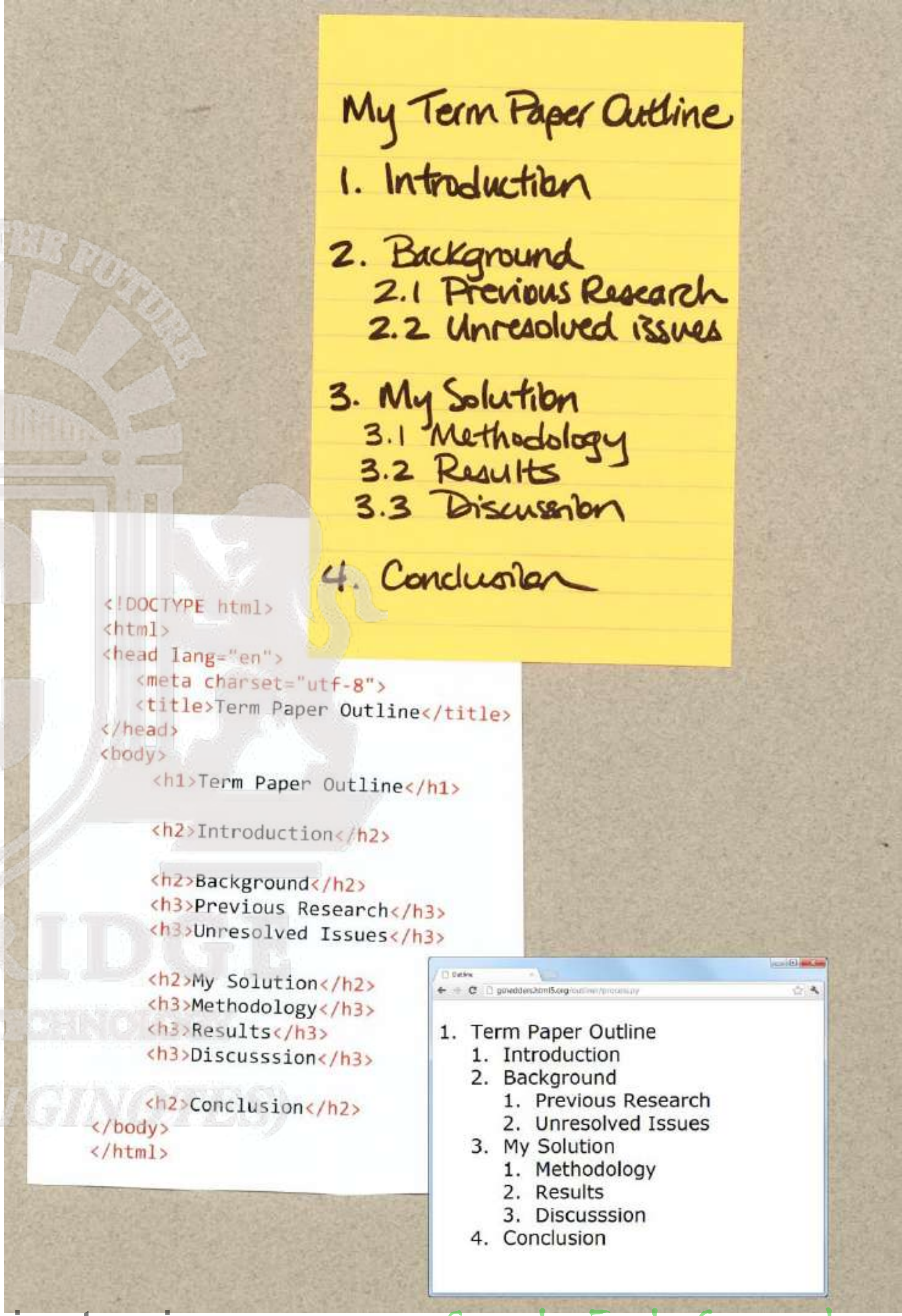
Copyright © 2012 Share Your Travels

1 Headings

<h1>, <h2>, <h3>, etc

❖HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.

❖Headings are an essential way for document authors use to show their readers the structure of the document.



My Term Paper Outline
1. Introduction
2. Background
  2.1 Previous Research
  2.2 Unresolved issues
3. My Solution
  3.1 Methodology
  3.2 Results
  3.3 Discussion
4. Conclusion

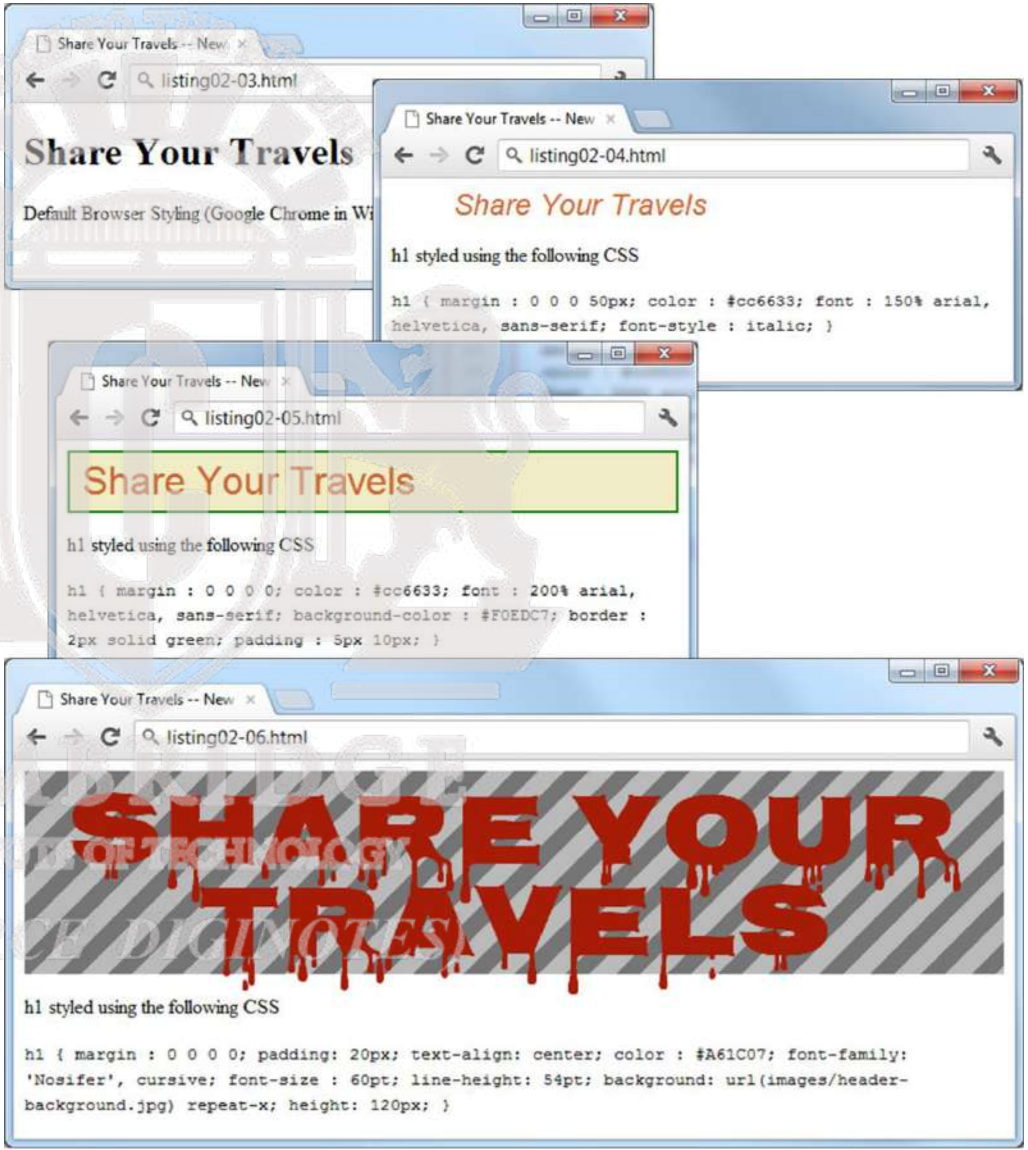
```html
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="utf-8">
    <title>Term Paper Outline</title>
</head>
<body>
    <h1>Term Paper Outline</h1>

    <h2>Introduction</h2>

    <h2>Background</h2>
    <h3>Previous Research</h3>
    <h3>Unresolved Issues</h3>

    <h2>My Solution</h2>
    <h3>Methodology</h3>
    <h3>Results</h3>
    <h3>Discusssion</h3>

    <h2>Conclusion</h2>
</body>
</html>
```

1. Term Paper Outline
   1. Introduction
   2. Background
      1. Previous Research
      2. Unresolved Issues
   3. My Solution
      1. Methodology
      2. Results
      3. Discusssion
   4. Conclusion

Headings

❖The browser has its own default styling for each heading level.

❖However, these are easily modified and customized via CSS.


Share Your Travels -- New
listing02-03.html
Share Your Travels -- New
listing02-04.html
Share Your Travels -- New
listing02-05.html
Share Your Travels -- New
listing02-06.html

Share Your Travels

Default Browser Styling (Google Chrome in Wi

Share Your Travels

h1 styled using the following CSS

```
h1 { margin : 0 0 0 50px; color : #cc6633; font : 150% arial,
helvetica, sans-serif; font-style : italic; }
```

Share Your Travels

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; color : #cc6633; font : 200% arial,
helvetica, sans-serif; background-color : #F0EDC7; border :
2px solid green; padding : 5px 10px; }
```

SHARE YOUR TRAVELS

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; padding: 20px; text-align: center; color : #A61C07; font-family:
'Nosifer', cursive; font-size : 60pt; line-height: 54pt; background: url(images/header-
background.jpg) repeat-x; height: 120px; }
```

Headings

Be semantically accurate

- In practice, specify a heading level that is semantically accurate.
- Do not choose a heading level because of its default presentation
  - e.g., choosing <h3> because you want your text to be bold and 16pt
- Rather, choose the heading level because it is appropriate
  - e.g., choosing <h3> because it is a third level heading and not a primary or secondary heading

2 Paragraphs

<p>

✓ Paragraphs are the most basic unit of text in an HTML document.

✓ Notice that the **<p>** tag is a container and can contain HTML and other **inline HTML elements**

✓ Inline HTML elements refers to HTML elements that do not cause a paragraph break but are part of the regular “flow” of the text.

6

Divisions

<div>

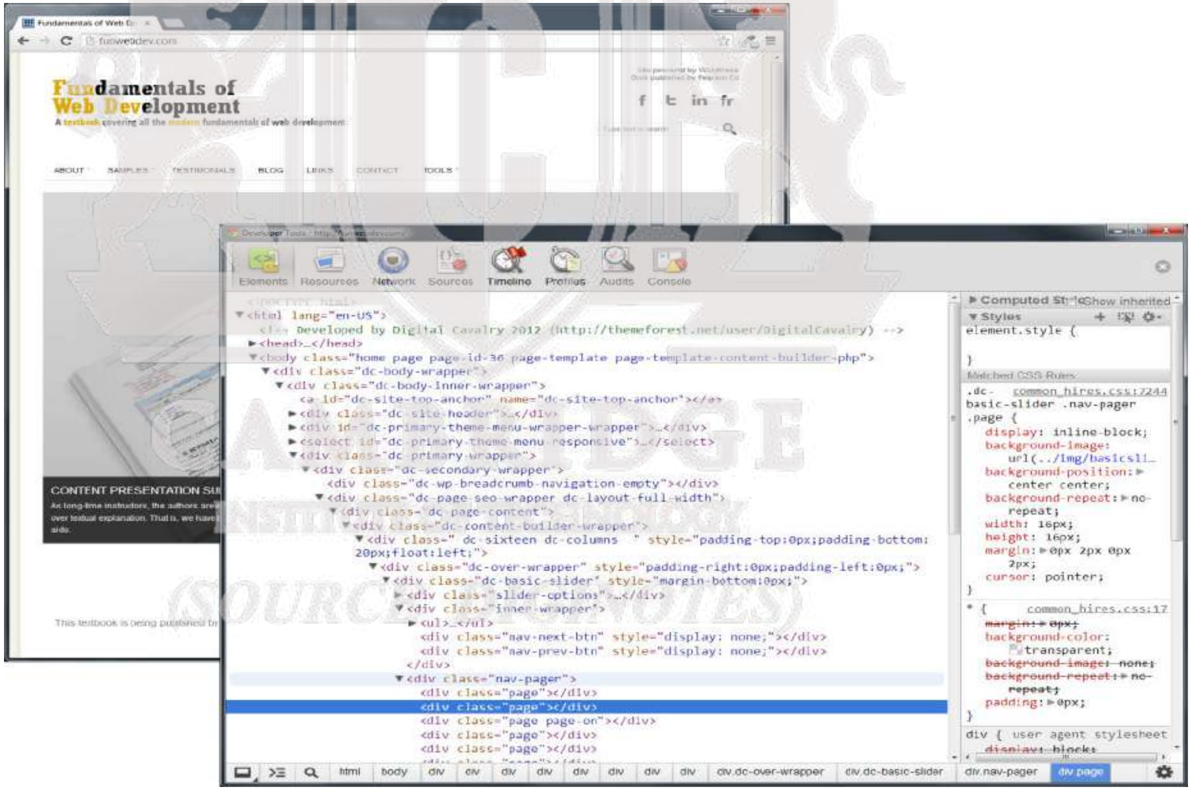
This **<div>** tag is also a container element and is used to create a logical grouping of content

- The <div> element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

Using div elements

Can you say "div-tastic"

❖HTML5 has a variety of new semantic elements (which we will examine later) that can be used to reduce somewhat the confusing mass of div within divs within divs that is so typical of contemporary web design.



```
<!DOCTYPE html>
▼<html lang="en-US">
  <!-- Developed by Digital Cavalry 2012 (http://themeforest.net/user/DigitalCavalry) -->
 ►<head>…</head>
 ▼<body class="home page page-id-36 page-template page-template-content-builder-php">
  ▼<div class="dc-body-wrapper">
   ▼<div class="dc-body-inner-wrapper">
      <a id="dc-site-top-anchor" name="dc-site-top-anchor"></a>
    ►<div class="dc-site-header">…</div>
    ►<div id="dc-primary-theme-menu-wrapper-wrapper">…</div>
    ►<select id="dc-primary-theme-menu-responsive">…</select>
    ▼<div class="dc-primary-wrapper">
     ▼<div class="dc-secondary-wrapper">
        <div class="dc-wp-breadcrumb-navigation-empty"></div>
      ▼<div class="dc-page-seo-wrapper dc-layout-full-width">
       ▼<div class="dc-page-content">
        ▼<div class="dc-content-builder-wrapper">
         ▼<div class=" dc-sixteen dc-columns  " style="padding-top:0px;padding-bottom:
          20px;float:left;">
          ▼<div class="dc-over-wrapper" style="padding-right:0px;padding-left:0px;">
           ▼<div class="dc-basic-slider" style="margin-bottom:0px;">
            ►<div class="slider-options">…</div>
            ▼<div class="inner-wrapper">
             ►<ul>…</ul>
              <div class="nav-next-btn" style="display: none;"></div>
              <div class="nav-prev-btn" style="display: none;"></div>
             </div>
            ▼<div class="nav-pager">
              <div class="page"></div>
              <div class="page"></div>
              <div class="page page-on"></div>
              <div class="page"></div>
              <div class="page"></div>
```

```
element.style {

}
.dc-    common_hires.css:7244
basic-slider .nav-pager
.page {
   display: inline-block;
   background-image:
      url(../img/basicsli…
   background-position: ►
      center center;
   background-repeat: ► no-
      repeat;
   width: 16px;
   height: 16px;
   margin: ► 0px 2px 0px
      2px;
   cursor: pointer;
}
* {        common_hires.css:17
   ~~margin: ► 0px;~~
   background-color:
      transparent;
   ~~background-image: none;~~
   ~~background-repeat: ► no-
      repeat;~~
   padding: ► 0px;
}
div { user agent stylesheet
   ~~display: block;~~
```

❸ Links

`<a>`

- Links are created using the **<a>** element (the “a” stands for anchor).
- A link has two main parts: the destination and the label.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination

Label (text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

Label (image)

Types of Links

You can use the anchor element to create a wide range of links:

- Links to external sites (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current site.
- Links to other places within the current page.
- Links to particular locations on another page.
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a Javascript function.

Different link destinations

Link to external site

```
<a href="http://www.centralpark.com">Central Park</a>
```

Link to resource on external site

```
<a href="http://www.centralpark.com/logo.gif">Central Park</a>
```

Link to another page on same site as this page

```
<a href="index.html">Home</a>
```

Link to another place on the same page

```
<a href="#top">Go to Top of Document</a>
```

Link to specific place on another page

```
<a href="productX.html#reviews">Reviews for product X</a>
```

Link to email

```
<a href="mailto://person@somewhere.com">Someone</a>
```

Link to javascript function

```
<a href="javascript://OpenAnnoyingPopup();">See This</a>
```

Link to telephone (automatically dials the number
when user clicks on it using a smartphone browser)

```
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

Link Text

Some guidance … or … don't "Click Here"

Links with the label "Click Here" were once a staple of the web.

Today, such links are frowned upon, since:

- they do not tell users where the link will take them
- as a verb "click" is becoming increasingly inaccurate when one takes into account the growth of mobile browsers.

Instead, textual link labels should be descriptive.

~~"Click here to see the race results"~~

"Race Results" or **"See Race Results".**

URL Absolute Referencing

For external resources

When referencing a page or resource on an external site, a full **absolute reference** is required: that is,

- the protocol (typically, http://),
- the domain name,
- any paths, and then finally
- the file name of the desired resource.

URL Relative Referencing

An essential skill

- We also need to be able to successfully reference files within our site.
- This requires learning the syntax for so-called **relative referencing**.
- When referencing a resource that is on the same server as your HTML document, then you can use briefer relative referencing. If the URL does not include the “http://” then the browser will request the current server for the file.

URL Relative Referencing

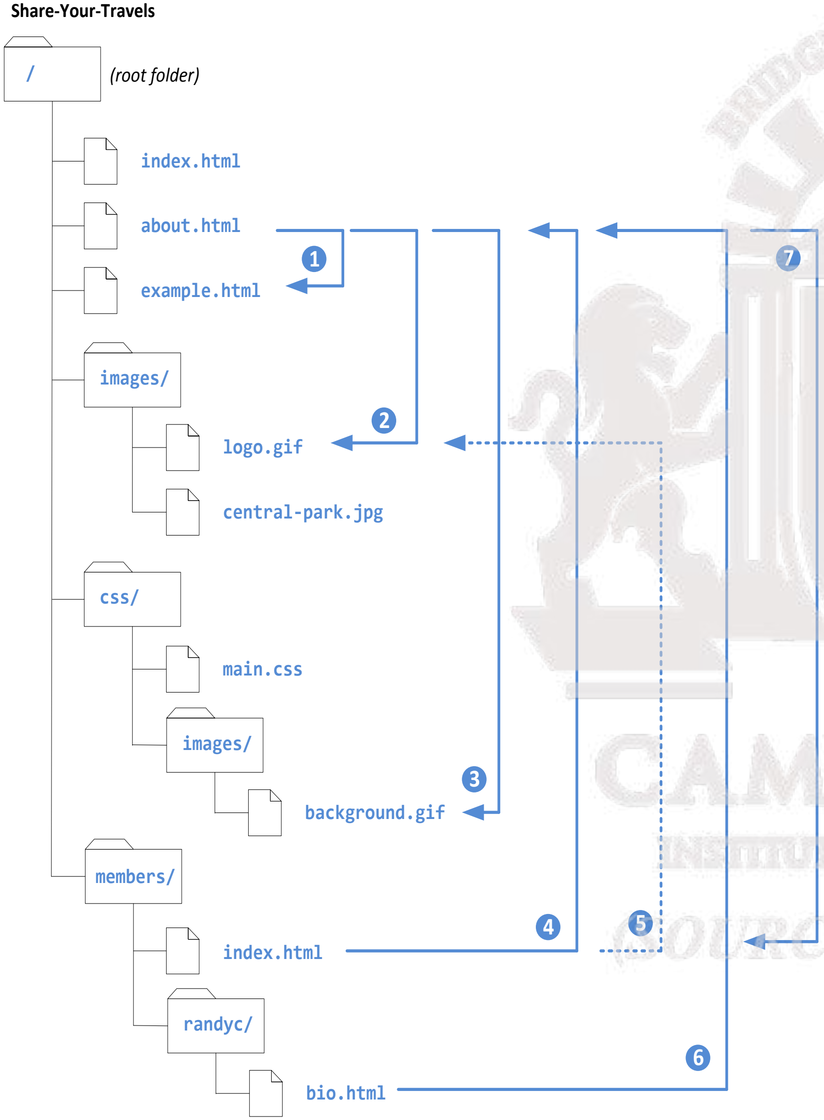
- If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their filename.
- However, most real-world sites contain too many files to put them all within a single directory.
- For these situations, a relative pathname is required along with the filename.
- The **pathname** tells the browser where to locate the file on the server.

Pathnames

Pathnames on the web follow Unix conventions.

- Forward slashes (“/”) are used to separate directory names from each other and from file names.
- Double-periods (“..”) are used to reference a directory “above” the current one in the directory tree.

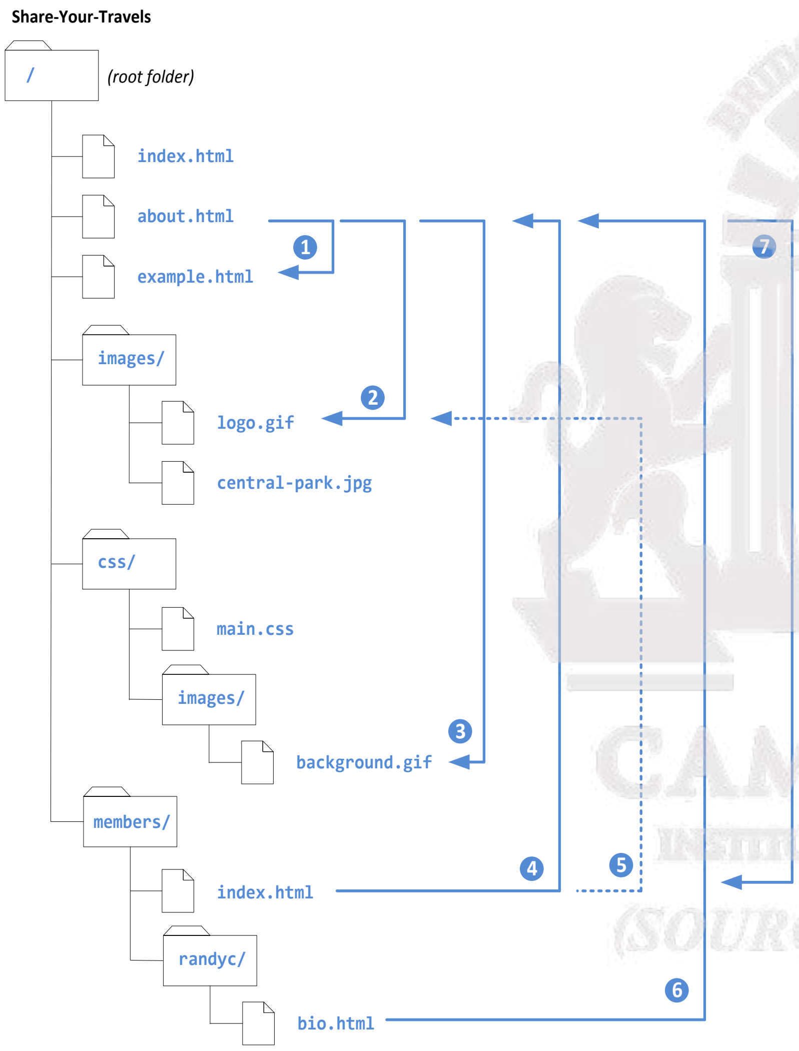
URL Relative Referencing



Relative Link Type	Example
1 **Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.18), use: `<a href="example.html">`
2 **Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: `<a href="images/logo.gif">`
3 **Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one, construct the full path by including each subdirectory name (separated by slashes) before the file name.	To link to background.gif from about.html, use: `<a href="css/images/background.gif">`

4 **Parent/Ancestor Directory**
Use “../” to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple “../”.
	To link to about.html from index.html in members, use:
	`<a href="../about.html">`
	To link to about.html from bio.html, use:
	`<a href="../../about.html">`

URL Relative Referencing



5 Sibling Directory

Use “../”to move up to the appropriate level, and then use the same technique as for child or grandchild directories.

To link to logo.gif from index.html in members, use:

```
<a href="../images/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="../../css/images/background.gif">
```

6 Root Reference

An alternative approach for ancestor and sibling references is to use the so-called root reference approach. In this approach, begin the reference with the root reference (the “/”) and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**

To link to about.html from bio.html, use:

```
<a href="/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="/images/background.gif">
```

7 Default Document

Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (apache) or default.html (IIS). **Again, this will only generally work on the web server.**

To link to index.html in members from about.html, use either:

```
<a href="members">
```

Or

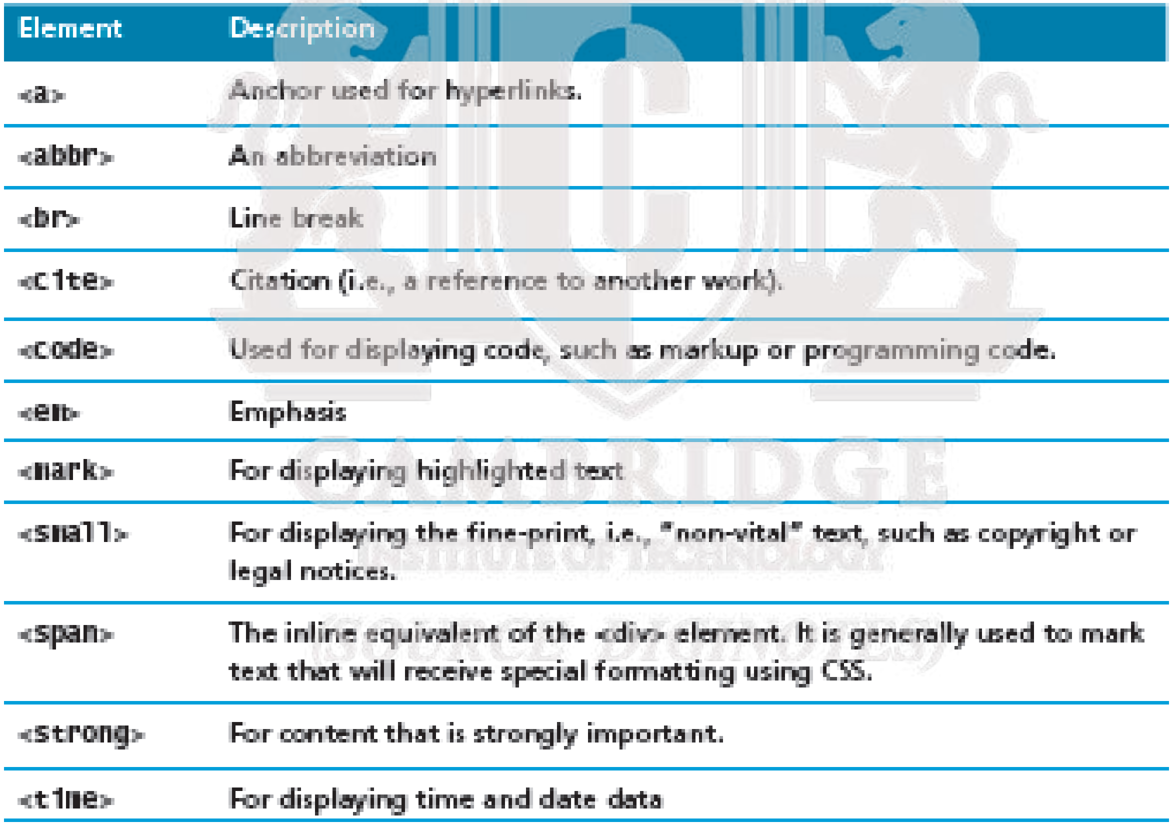
```
<a href="/members">
```

Inline Text Elements

Do not disrupt the flow

Inline elements do not disrupt the flow of text (i.e., cause a line break).

HTML5 defines over 30 of these elements.



Element	Description
<a>	Anchor used for hyperlinks.
<abbr>	An abbreviation
<br>	Line break
<cite>	Citation (i.e., a reference to another work).
<code>	Used for displaying code, such as markup or programming code.
<em>	Emphasis
<mark>	For displaying highlighted text
<small>	For displaying the fine-print, i.e., "non-vital" text, such as copyright or legal notices.
<span>	The inline equivalent of the <div> element. It is generally used to mark text that will receive special formatting using CSS.
<strong>	For content that is strongly important.
<time>	For displaying time and date data

TABLE 2.2 Common Text-Level Semantic Elements

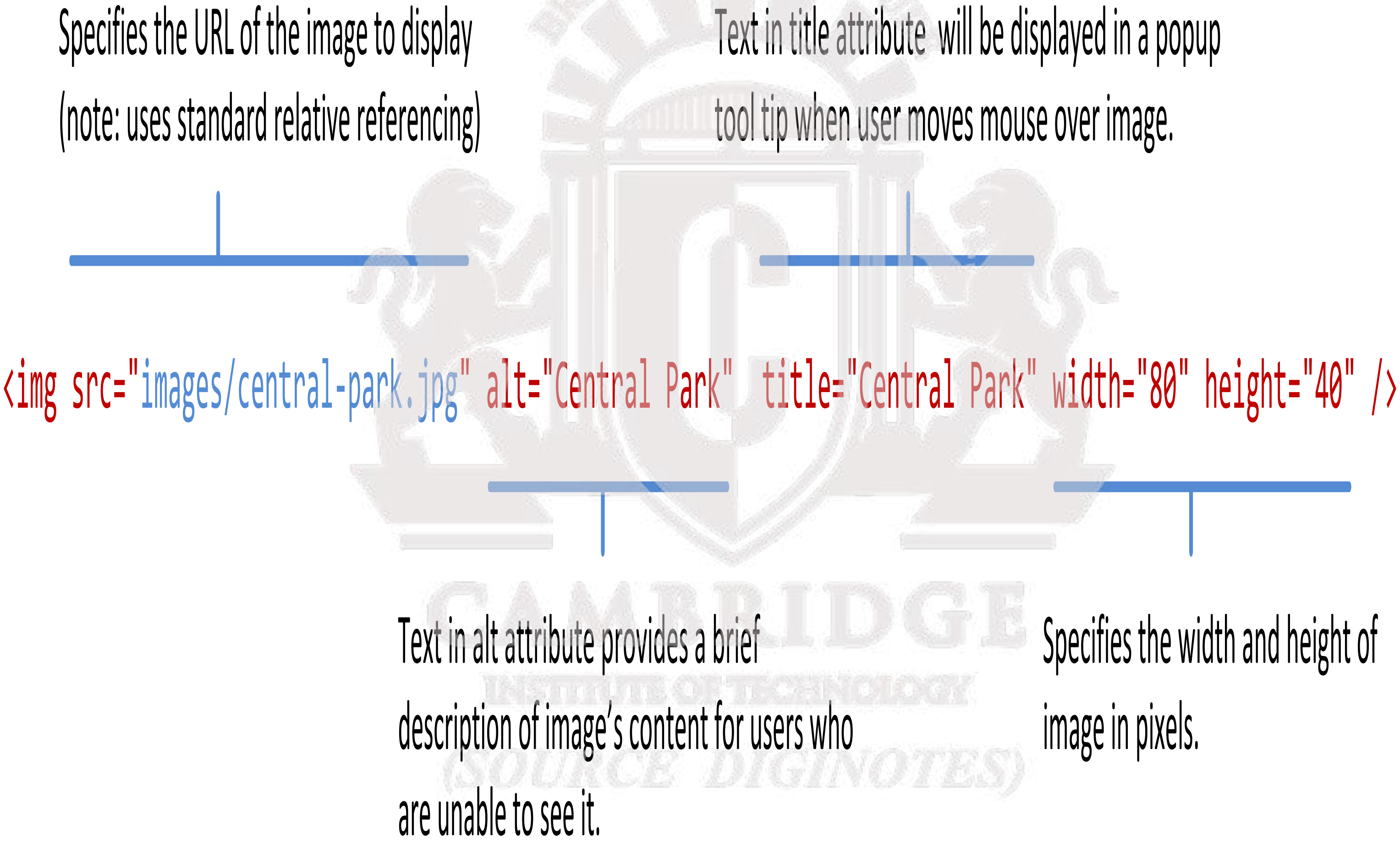
Images

✓While the **<img>** tag is the oldest method for displaying an image, it is not the only way.

✓For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

✓But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

Images



Specifies the URL of the image to display (note: uses standard relative referencing)

Text in title attribute will be displayed in a popup tool tip when user moves mouse over image.

```
<img src="images/central-park.jpg" alt="Central Park" title="Central Park" width="80" height="40" />
```

Text in alt attribute provides a brief description of image's content for users who are unable to see it.

Specifies the width and height of image in pixels.

Lists

HTML provides three types of lists

- **Unordered lists**. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

- **Ordered lists**. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

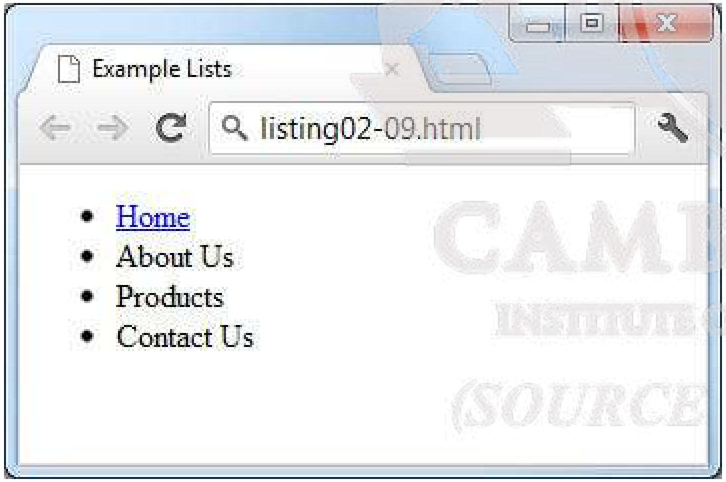
- **Definition lists**. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

Lists

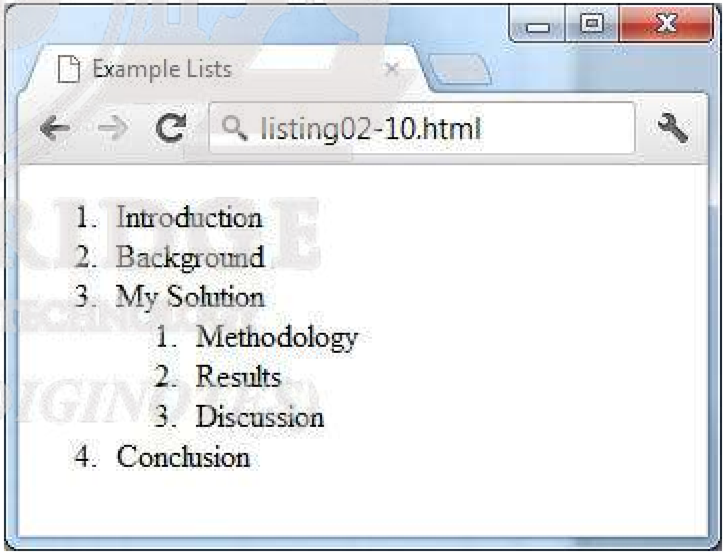
Notice that the list item element can contain other HTML elements

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li>About Us</li>
   <li>Products</li>
   <li>Contact Us</li>
</ul>
```

```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
     <ol>
        <li>Methodology</li>
        <li>Results</li>
        <li>Discussion</li>
     </ol>
   </li>
   <li>Conclusion</li>
</ol>
```



- Home
- About Us
- Products
- Contact Us



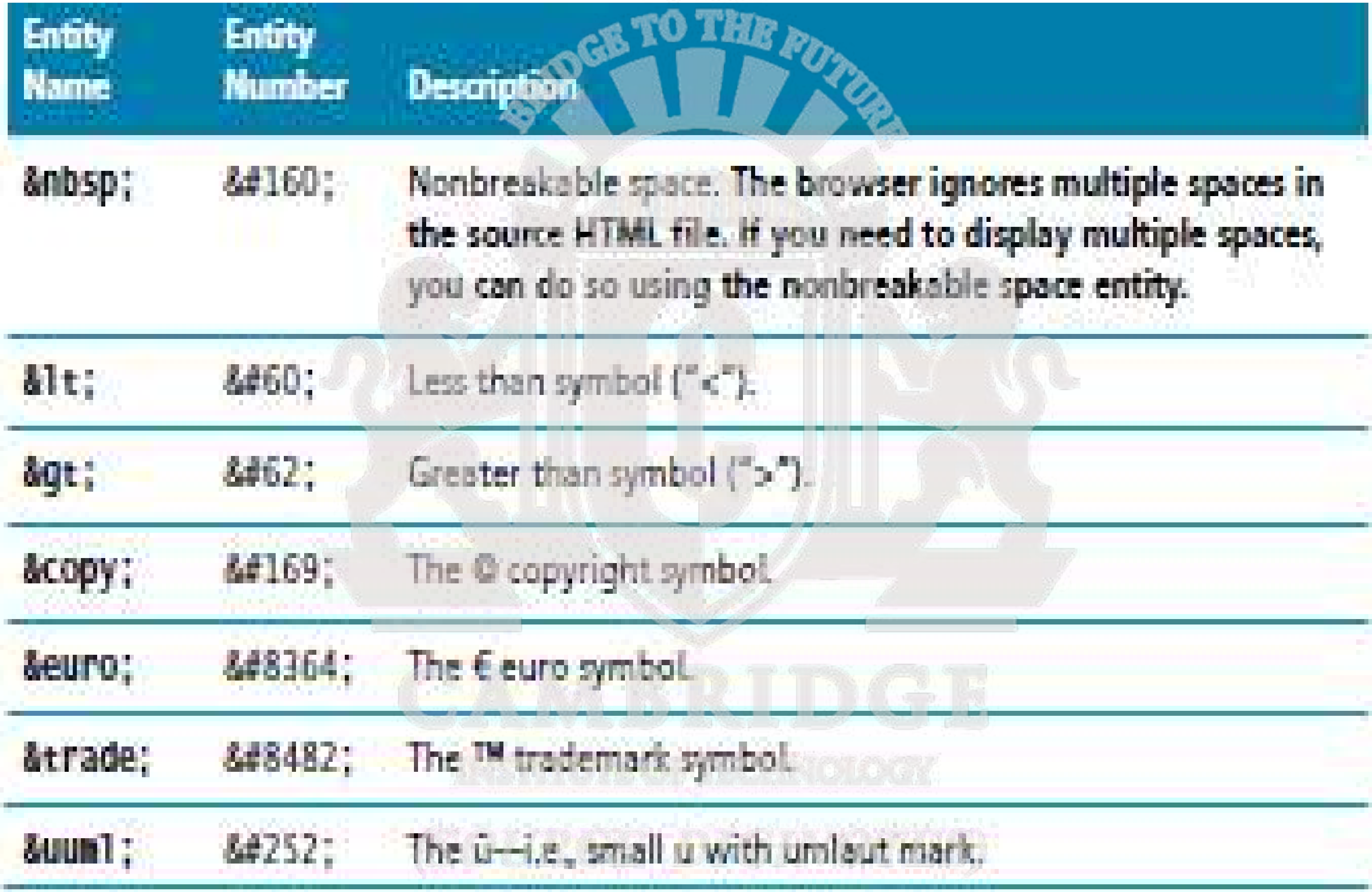
1. Introduction
2. Background
3. My Solution
    1. Methodology
    2. Results
    3. Discussion
4. Conclusion

Character Entities

❖These are special characters for symbols for which there is either no way easy way to type in via a keyboard (such as the copyright symbol or accented characters) or which have a reserved meaning in HTML (for instance the “<” or “>” symbols).

❖They can be used in an HTML document by using the entity name or the entity number.

e.g., &nbsp; and &copy;



Entity Name	Entity Number	Description
&nbsp;	&#160;	Nonbreakable space. The browser ignores multiple spaces in the source HTML file. If you need to display multiple spaces, you can do so using the nonbreakable space entity.
&lt;	&#60;	Less than symbol ("<").
&gt;	&#62;	Greater than symbol (">").
&copy;	&#169;	The © copyright symbol.
&euro;	&#8364;	The € euro symbol.
&trade;	&#8482;	The ™ trademark symbol.
&uuml;	&#252;	The ü—i.e., small u with umlaut mark.

TABLE 2.3 Common Character Entities

Section 6 of 6

HTML SEMANTIC ELEMENTS

1 10

Header and Footer

<header> <footer>

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.

1 10

Header and Footer

<header> <footer>

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site’s privacy policy, and
- perhaps twitter feeds or links to other social sites.

Header and Footer

❑Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements
</h2>
      <p>By <em>Randy Connolly</em></p>
      <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

2

Heading Groups

<hgroup>

The <hgroup> element can be used to group related headings together within one container.

```
<header>
   <hgroup>
       <h1>Chapter Two: HTML 1</h1>
       <h2>An Introduction</h2>
   </hgroup>
</header>
<article>
   <hgroup>
      <h2>HTML5 Semantic Structure Elements </h2>
      <h3>Overview</h3>
   </hgroup>
</article>
```

3 Navigation

<nav>

❖The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

❖Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element.

❖The <nav> element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.

Navigation

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   <nav role="navigation">
      <ul>
         <li><a href="index.html">Home</a></li>
         <li><a href="about.html">About Us</a></li>
         <li><a href="browse.html">Browse</a></li>
      </ul>
   </nav>
</header>
```

❺ ❻

Articles and Sections

<article> <section>

- □ The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

- □ The **<section>** element represents a section of a document, typically with a title or heading.

- □ According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.

Sections versus Divs

How to decide which to use

- The WHATWG specification warns readers that the <section> element is **not** a generic container element. HTML already has the <div> element for such uses.
- When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the <div> element instead.
- Another way to help you decide whether or not to use the <section> element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

If so, then use a <section>; otherwise use a <div>.

7 8

Figure and Figure Captions

<figure> <figcaption>

- The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.

- The **<figure>** element should **not** be used to wrap every image.

- For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

- Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

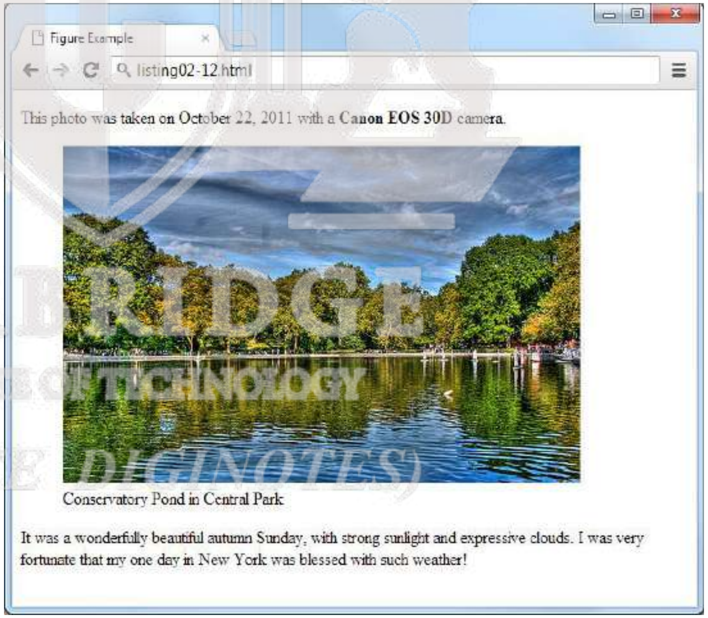
Figure and Figure Captions

Figure could be moved to a different location in document

…

But it has to exist in the document (i.e., the figure isn't optional)

```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
  <img src="images/central-park.jpg" alt="Central Park" /><br/>
  <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
<p>
It was a wonderfully beautiful autumn Sunday, with strong sunlight and
expressive clouds. I was very fortunate that my one day in New York was
blessed with such weather!
</p>
```



This photo was taken on October 22, 2011 with a **Canon EOS 30D** camera.

Conservatory Pond in Central Park

It was a wonderfully beautiful autumn Sunday, with strong sunlight and expressive clouds. I was very fortunate that my one day in New York was blessed with such weather!

Aside

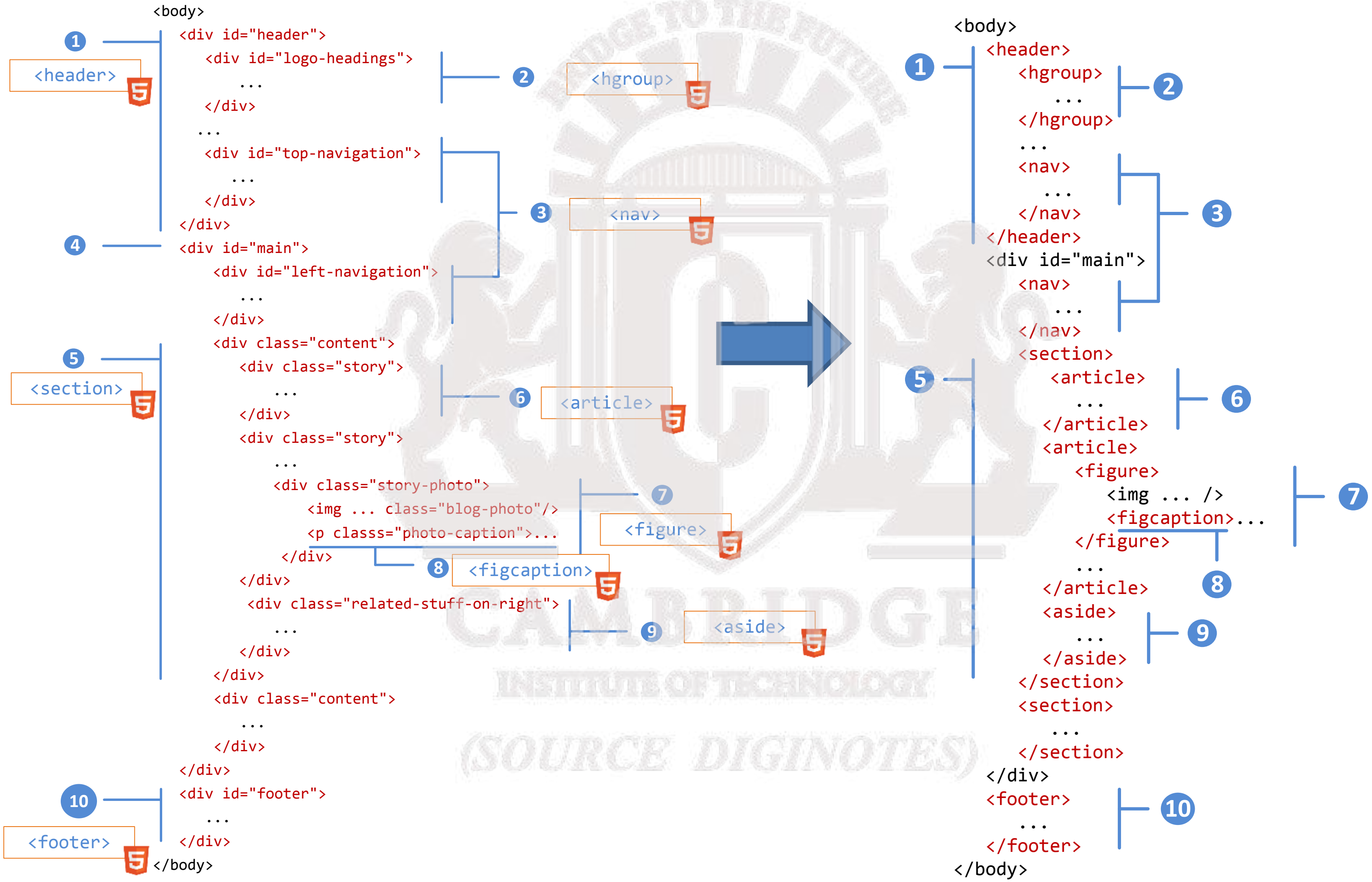
<aside>

❑The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

❑But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element "represents a section of a page that consists of content that is tangentially related to the content around the aside element."

❑The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

XHTML versus HTML5



```
<body>
  <div id="header">
    <div id="logo-headings">
      ...
    </div>
    ...
    <div id="top-navigation">
      ...
    </div>
  </div>
  <div id="main">
     <div id="left-navigation">
      ...
     </div>
     <div class="content">
       <div class="story">
         ...
       </div>
       <div class="story">
         ...
         <div class="story-photo">
           <img ... class="blog-photo"/>
           <p classs="photo-caption">...
        </div>
       </div>
        <div class="related-stuff-on-right">
         ...
       </div>
     </div>
     <div class="content">
      ...
     </div>
  </div>
  <div id="footer">
   ...
  </div>
</body>
```

```
<body>
   <header>
      <hgroup>
        ...
      </hgroup>
      ...
      <nav>
        ...
      </nav>
   </header>
   <div id="main">
      <nav>
        ...
      </nav>
      <section>
         <article>
           ...
        </article>
        <article>
           <figure>
              <img ... />
              <figcaption>...
           </figure>
           ...
        </article>
        <aside>
          ...
        </aside>
      </section>
      <section>
        ...
      </section>
   </div>
   <footer>
     ...
   </footer>
</body>
```

Source diginotes.in

CSS 1: Introduction

Chapter 3

Objectives

1 What is CSS?

2 CSS Syntax

3 Location of Styles

4 Selectors

5 The Cascade: How Styles Interact

6 The Box Model

7 CSS Text Styling

Section 1 of 7

WHAT IS CSS?

What is CSS?

You be styling soon

CSS is a W3C standard for describing the **presentation (or appearance)** of HTML elements.

With CSS, we can assign

- font properties,
- colors,
- sizes,
- borders,
- background images,
- even the position of elements.

What is CSS?

You be styling soon

❑CSS is a language in that it has its own syntax rules.

❑CSS can be added directly to any HTML element (via the style attribute), within the **<head>** element, or, most commonly, in a separate text file that contains only CSS.

Benefits of CSS

Why using CSS is a better way of describing presentation than HTML

- The degree of formatting control in CSS is significantly better than that provided in HTML.
- Web sites become significantly more maintainable because all formatting can be centralized into one, or a small handful, of CSS files.
- CSS-driven sites are more accessible.
- A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain less markup.
- CSS can be used to adopt a page for different output mediums.

CSS Versions

Let’s just say there’s more than 1

- W3C published the CSS Level 1 Recommendation in 1996.
- A year later, the CSS Level 2 Recommendation (also more succinctly labeled simply as CSS2) was published.
- Even though work began over a decade ago, an updated version of the Level 2 Recommendation, CSS2.1, did not become an official W3C Recommendation until June 2011.
- And to complicate matters even more, all through the last decade (and to the present day as well), during the same time the CSS2.1 standard was being worked on, a different group at the W3C was working on a CSS3 draft.

Browser Adoption

Insert obligatory snide comment about Internet Explorer 6 here

❖While Microsoft’s Internet Explorer was an early champion of CSS, its later versions (especially IE5, IE6, and IE7) for Windows had uneven support for certain parts of CSS2.

❖In fact, all browsers have left certain parts of the CSS2 Recommendation unimplemented.

CSS has a reputation for being a somewhat frustrating language.

- this reputation is well deserved!

Section 2 of 7

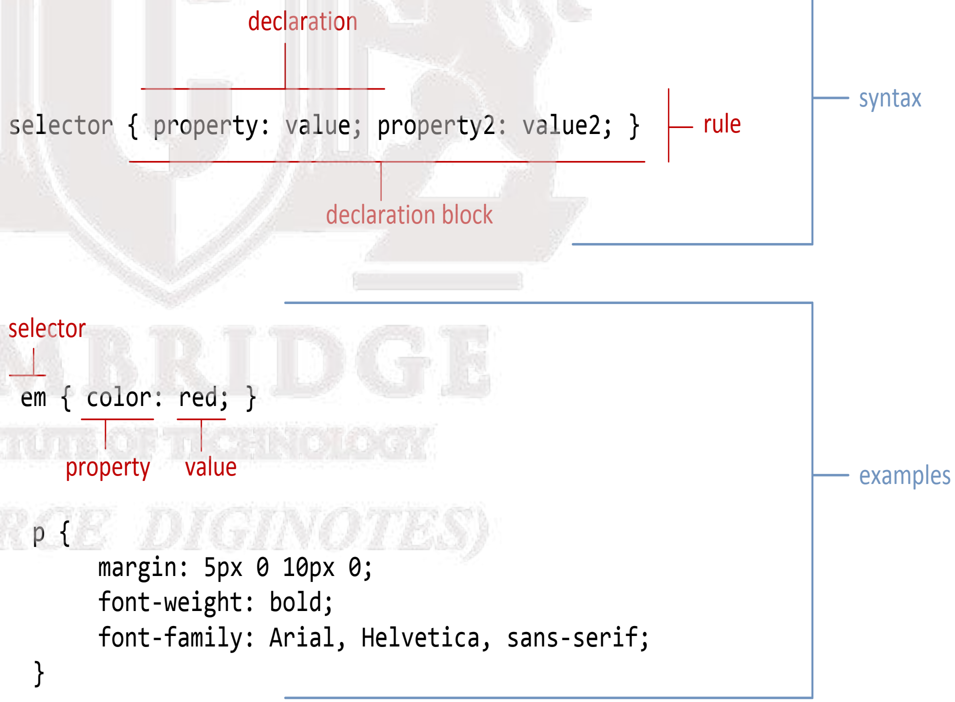
CSS SYNTAX

CSS Syntax

Rules, properties, values, declarations

❖A CSS document consists of one or more **style rules**.

❖A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).



```
selector { property: value; property2: value2; }
```

```
em { color: red; }
```

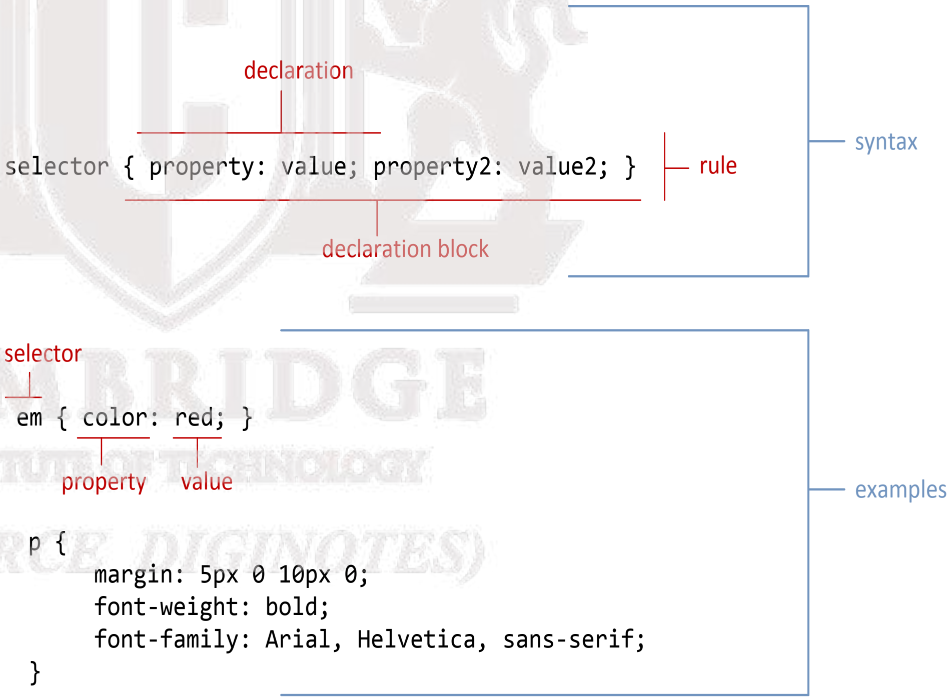
```
p {
      margin: 5px 0 10px 0;
      font-weight: bold;
      font-family: Arial, Helvetica, sans-serif;
}
```

Declaration Blocks

The series of declarations is also called the **declaration block**.

- A declaration block can be together on a single line, or spread across multiple lines.

- The browser ignores white space
- Each declaration is terminated with a semicolon.



```
selector { property: value; property2: value2; }
```

```
em { color: red; }

p {
      margin: 5px 0 10px 0;
      font-weight: bold;
      font-family: Arial, Helvetica, sans-serif;
}
```

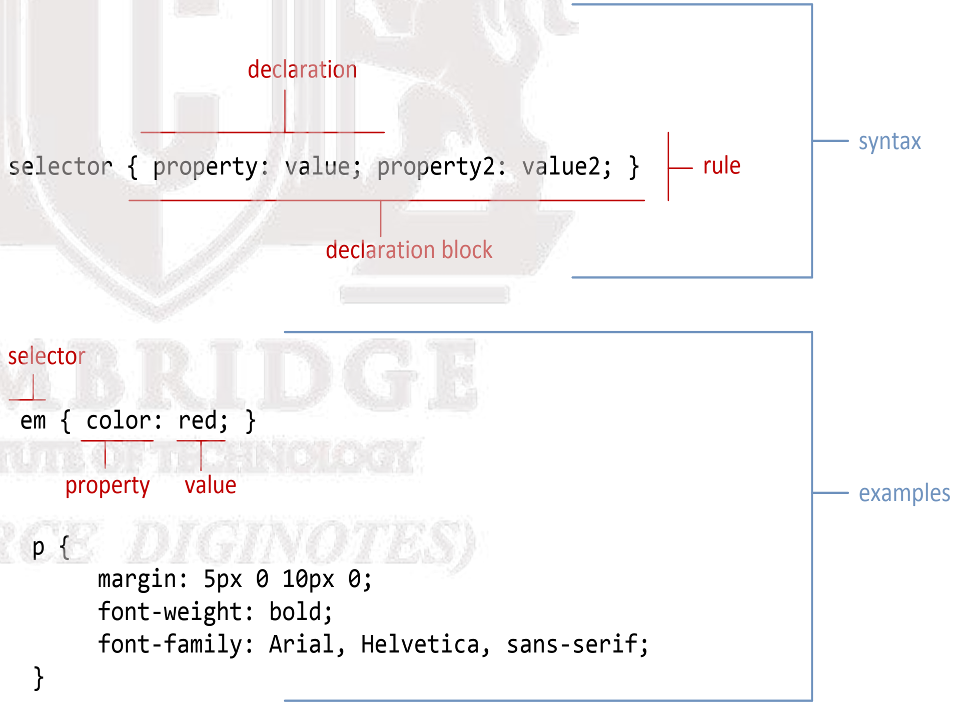
Selectors

Which elements

❖Every CSS rule begins with a **selector**.

❖The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive the style.



declaration

```
selector { property: value; property2: value2; }
```

rule

declaration block

syntax

selector

```
em { color: red; }
```

property

value

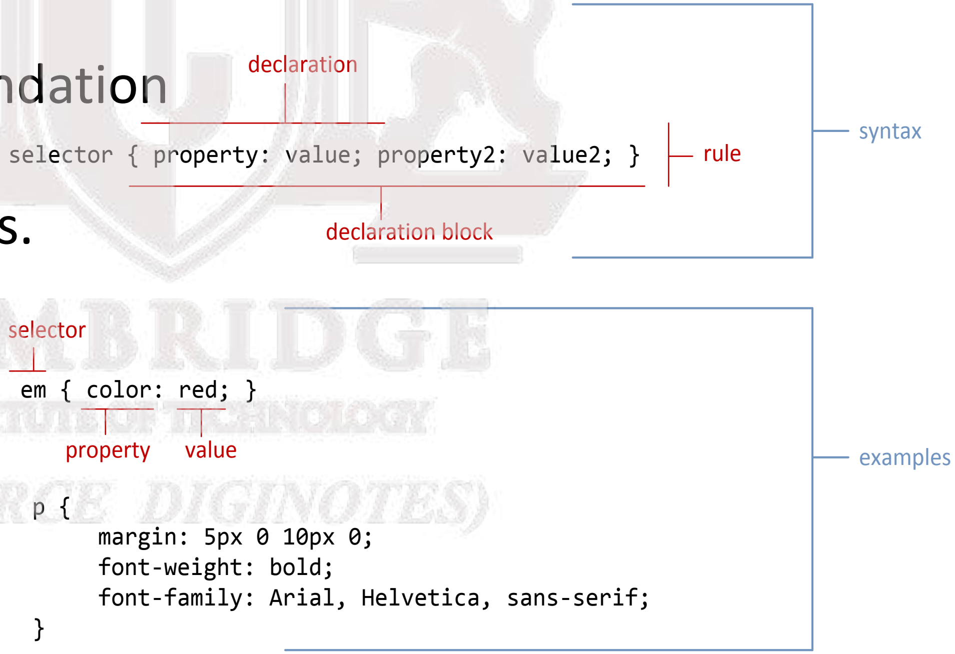
examples

```
p {
      margin: 5px 0 10px 0;
      font-weight: bold;
      font-family: Arial, Helvetica, sans-serif;
}
```

Properties

Which style properties of the selected elements

- ❖Each individual CSS declaration must contain a **property**.
- ❖These property names are predefined by the CSS standard.
- ❖The CSS2.1 Recommendation defines over a hundred different property names.



```
selector { property: value; property2: value2; }
```

```
em { color: red; }
```

```
p {
        margin: 5px 0 10px 0;
        font-weight: bold;
        font-family: Arial, Helvetica, sans-serif;
}
```

Properties

Common CSS properties

Property Type	Property
Fonts	font font-family font-size font-style font-weight @font-face
Text	letter-spacing line-height text-align text-decoration text-indent
Color and background	background background-color background-image background-position background-repeat color
Borders	border border-color border-width border-style border-top border-top-color border-top-width etc

Properties

Common CSS properties continued.

Property Type	Property
Spacing	padding padding-bottom, padding-left, padding-right, padding-top margin margin-bottom, margin-left, margin-right, margin-top
Sizing	height max-height max-width min-height min-width width
Layout	bottom, left, right, top clear display float overflow position visibility z-index
Lists	list-style list-style-image list-style-type

Values

What style value for the properties

Each CSS declaration also contains a value for a property.

•The unit of any given value is dependent upon the property.

•Some property values are from a predefined list of keywords.

•Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

Color Values

CSS supports a variety of different ways of describing color

Method	Description	Example
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	color: red; color: hotpink; /* CSS3 only */

RGB	Uses three different numbers between 0 and 255 to describe the Red, Green, and Blue values for the color.	color: rgb(255,0,0);
		color: rgb(255,105,180);

Hexadecimal	Uses a six-digit hexadecimal number to describe the red, green, and blue value of the color; each of the three RGB values is between 0 and FF (which is 255 in decimal). Notice that the hexadecimal number is preceded by a hash or pound symbol (#).	color: #FF0000;
		color: #FF69B4;

RGBa	Allows you to add an alpha, or transparency, value. This allows a background color or image to "show through" the color. Transparency is a value between 0.0 (fully transparent) and 1.0 (fully opaque).	color: rgb(255,0,0, 0.5);

HSL	Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well.	color: hsl(0,100%,100%);
		color: hsl(330,59%,100%);

Units of Measurement

There are multiple ways of specifying a unit of measurement in CSS

- Some of these are **relative units**, in that they are based on the value of something else, such as the size of a parent element.
- Others are **absolute units**, in that they have a real-world size.
  - Unless you are defining a style sheet for printing, it is recommended to avoid using absolute units.
  - Pixels are perhaps the one popular exception (though as we shall see later there are also good reasons for avoiding the pixel unit).

Relative Units

Unit	Description	Type
px	Pixel. In CSS2 this is a relative measure, while in CSS3 it is absolute (1/96 of an inch).	Relative (CSS2) Absolute (CSS3)
em	Equal to the computed value of the font-size property of the element on which it is used. When used for font sizes, the em unit is in relation to the font size of the parent.	Relative
%	A measure that is always relative to another value. The precise meaning of % varies depending upon which property it is being used.	Relative
ex	A rarely used relative measure that expresses size in relation to the x-height of an element's font.	Relative
ch	Another rarely used relative measure; this one expresses size in relation to the width of the zero ("0") character of an element's font.	Relative (CSS3 only)
rem	Stands for root em, which is the font size of the root element. Unlike em, which may be different for each element, the rem is constant throughout the document.	Relative (CSS3 only)
vw, vh	Stands for viewport width and viewport height. Both are percentage values (between 0 and 100) of the viewport (browser window). This allows an item to change size when the viewport is resized.	Relative (CSS3 only)

Source diginotes.in

Absolute Units

Unit	Description	Type
in	Inches	Absolute
cm	Centimeters	Absolute
mm	Millimeters	Absolute
pt	Points (equal to 1/72 of an inch)	Absolute
pc	Pica (equal to 1/6 of an inch)	Absolute

Comments in CSS

✓It is often helpful to add comments to your style sheets. Comments take the form:

/* comment goes here */

Section 3 of 7

LOCATION OF STYLES

Actually there are three …

Different types of style sheet

- ❖**Author-created style sheets** (what we are learning in this presentation).

- ❖**User style sheets** allow the individual user to tell the browser to display pages using that individual's own custom style sheet. This option is available in a browser usually in its accessibility options area.

- ❖The **browser style sheet** defines the default styles the browser uses for each HTML element.

Style Locations

Author Created CSS style rules can be located in three different locations

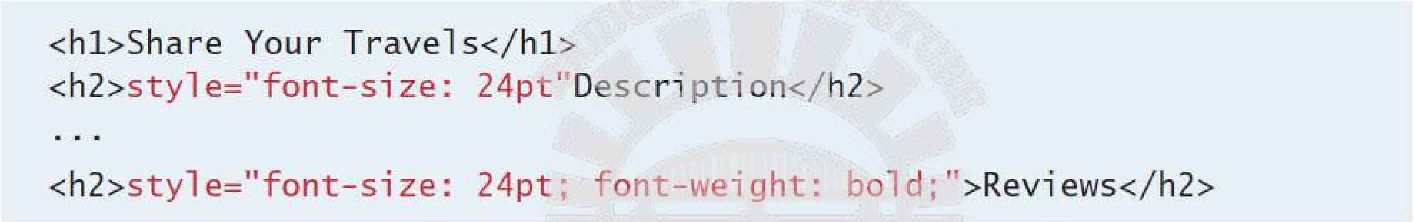
CSS style rules can be located in three different locations.

- Inline
- Embedded
- External

You can combine all 3!

Inline Styles

Style rules placed within an HTML element via the style attribute



```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 3.1 Internal styles example

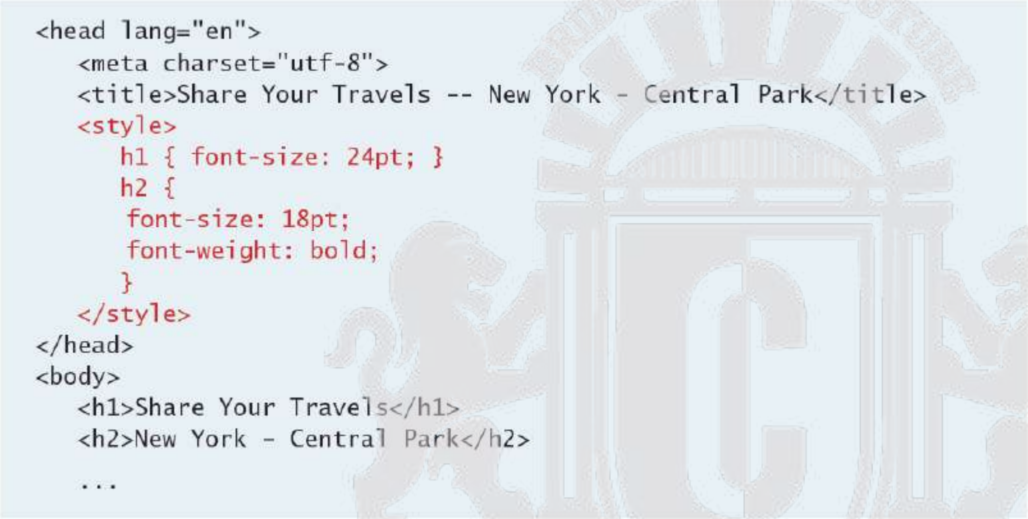
- ➢An inline style only affects the element it is defined within and will override any other style definitions for the properties used in the inline style.

- ➢Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability.

- ➢Inline styles can however be handy for quickly testing out a style change.

Embedded Style Sheet

Style rules placed within the <style> element inside the <head> element



```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <style>
      h1 { font-size: 24pt; }
      h2 {
       font-size: 18pt;
       font-weight: bold;
      }
   </style>
</head>
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   ...
```

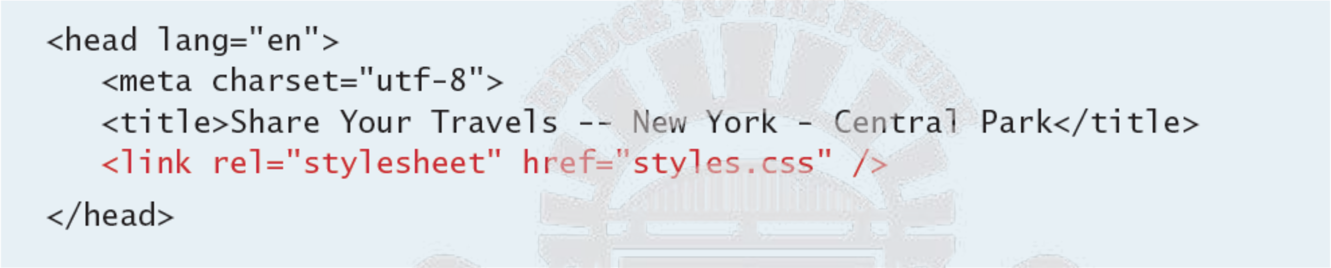
LISTING 3.2 Embedded styles example

- ➢While better than inline styles, using embedded styles is also by and large discouraged.

- ➢Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

External Style Sheet

Style rules placed within a external text file with the .css extension



```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

LISTING 3.3 Referencing an external style sheet

This is by far the most common place to locate style rules because it provides the best maintainability.

- When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.

- The browser is able to cache the external style sheet which can improve the performance of the site

Section 4 of 7

SELECTORS

Selectors

Things that make your life easier

When defining CSS rules, you will need to first need to use a **selector** to tell the browser which elements will be affected.

CSS selectors allow you to select

- individual elements
- multiple HTML elements,
- elements that belong together in some way, or
- elements that are positioned in specific ways in the document hierarchy.

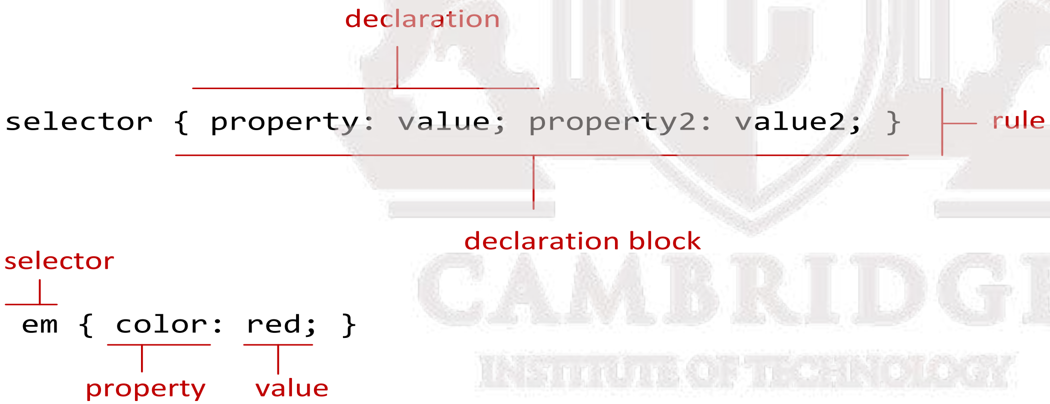
There are a number of different selector types.

Element Selectors

Selects all instances of a given HTML element

Uses the HTML element name.

You can select all elements by using the **universal element selector**, which is the * (asterisk) character.



declaration

```
selector { property: value; property2: value2; }
```

rule

declaration block

selector

```
 em { color: red; }
```

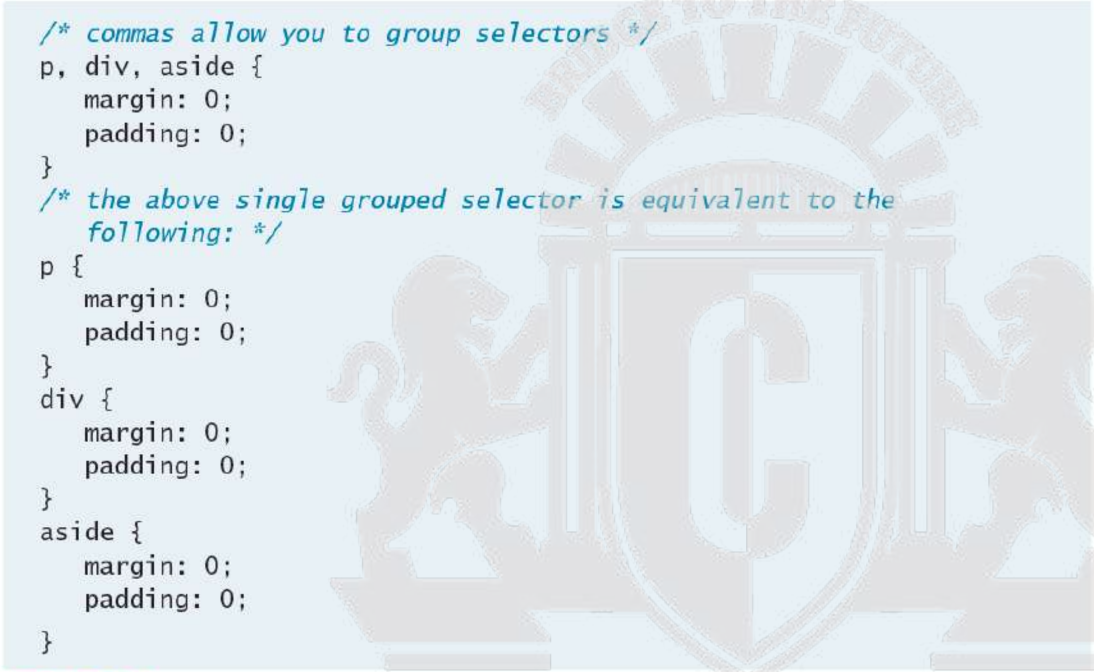
property

value

```
  p {
        margin: 5px 0 10px 0;
        font-weight: bold;
        font-family: Arial, Helvetica, sans-serif;
  }
```

Grouped Selectors

Selecting multiple things



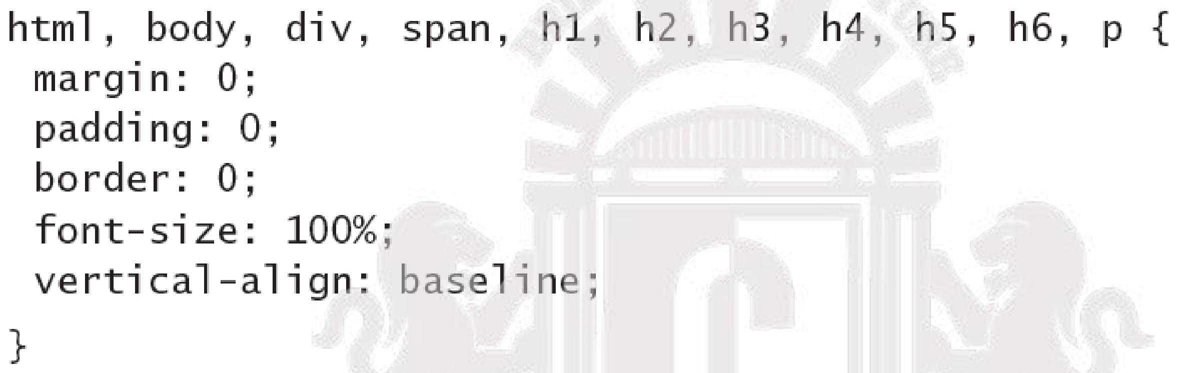
```
/* commas allow you to group selectors */
p, div, aside {
   margin: 0;
   padding: 0;
}
/* the above single grouped selector is equivalent to the
   following: */
p {
   margin: 0;
   padding: 0;
}
div {
   margin: 0;
   padding: 0;
}
aside {
   margin: 0;
   padding: 0;
}
```

LISTING 3.4 Sample grouped selector

❖You can select a group of elements by separating the different element names with commas.

❖This is a sensible way to reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule.

Reset



```
html, body, div, span, h1, h2, h3, h4, h5, h6, p {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  vertical-align: baseline;
}
```

❑Grouped selectors are often used as a way to quickly **reset** or remove browser defaults.

❑The goal of doing so is to reduce browser inconsistencies with things such as margins, line heights, and font sizes.

❑These reset styles can be placed in their own css file (perhaps called reset.css) and linked to the page **before** any other external styles sheets.

Class Selectors

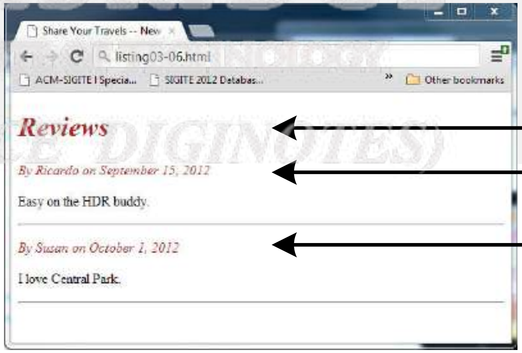
Simultaneously target different HTML elements

❖A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.

❖If a series of HTML element have been labeled with ***the same class attribute value***, then you can target them for styling by using a class selector, which takes the form: period (.) followed by the class name.

Class Selectors

```
<head>
   <title>Share Your Travels </title>
     <style>
         .first {
              font-style: italic;
              color: brown;
         }
     </style>
</head>
<body>
   <h1 class="first">Reviews</h1>
   <div>
      <p class="first">By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p class="first">By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

```
.first {
   font-style: italic;
   color: brown;
}
```

Id Selectors

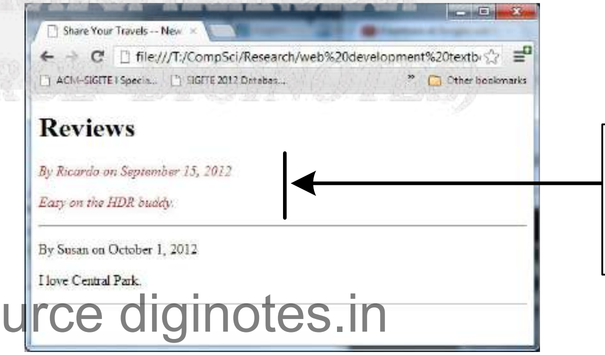
Target a specific element by its id attribute

- An **id selector** allows you to target a specific element by its id attribute regardless of its type or position.
- If an HTML element has been labeled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name.

Note: You should only be using an **id** once per page

Id Selectors

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
     <style>
          #latestComment {
               font-style: italic;
               color: brown;
          }
     </style>
</head>
<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```


Share Your Travels -- New
file:///T:/CompSci/Research/web%20development%20textb
ACM-SIGITE | Specia...
SIGITE 2012 Databas...
Other bookmarks

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

```
#latestComment {
   font-style: italic;
   color: brown;
}
```

Id versus Class Selectors

How to decide

✓Id selectors should only be used when referencing a single HTML element since an id attribute can only be assigned to a single HTML element.

✓Class selectors should be used when (potentially) referencing several related elements.

Attribute Selectors

Selecting via presence of element attribute or by the value of an attribute

- An **attribute selector** provides a way to select HTML elements by either the presence of an element attribute or by the value of an attribute.

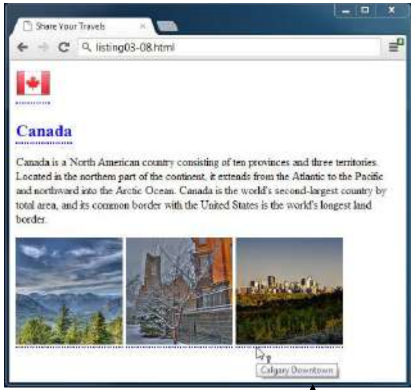
- This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them.

- Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.

Attribute Selectors

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
     <style>
          [title] {
               cursor: help;
               padding-bottom: 3px;
               border-bottom: 2px dotted blue;
               text-decoration: none;
          }
     </style>
</head>
<body>
   <div>
      <img src="images/flags/CA.png" title="Canada Flag" />
       <h2><a href="countries.php?id=CA" title="see posts from Canada">
            Canada</a>
        </h2>
       <p>Canada is a North American country consisting of … </p>
       <div>
          <img src="images/square/6114907897.jpg" title="At top of Sulpher
Mountain">
          <img src="images/square/6592317633.jpg" title="Grace Presbyterian
Church">
          <img src="images/square/6592914823.jpg" title="Calgary Downtown">
       </div>
   </div>
</body>
```

```
[title] {
   cursor: help;
   padding-bottom: 3px;
   border-bottom: 2px dotted blue;
   text-decoration: none;
}
```

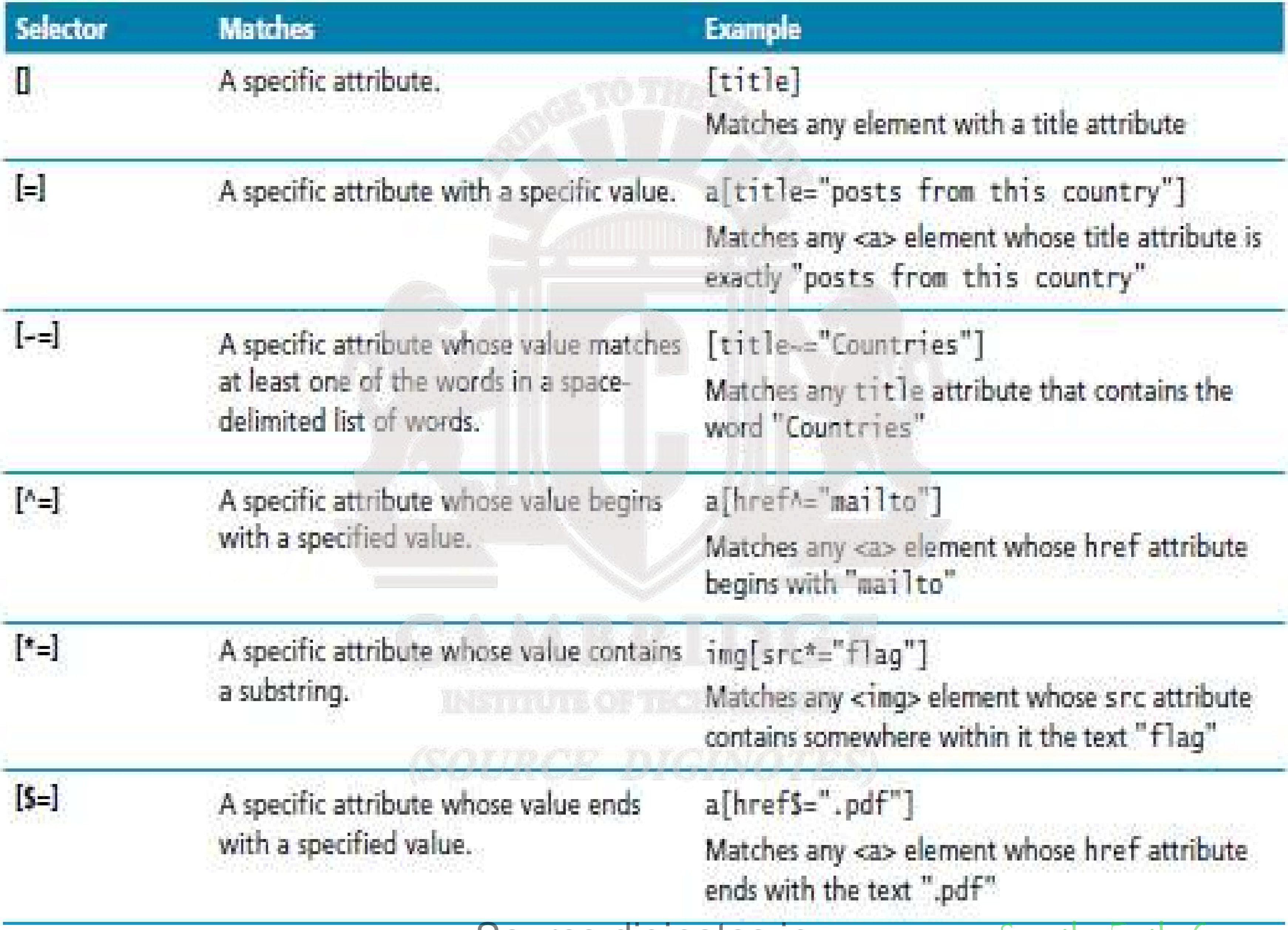


Canada

Canada is a North American country consisting of ten provinces and three territories. Located in the northern part of the continent, it extends from the Atlantic to the Pacific and northward into the Arctic Ocean. Canada is the world's second-largest country by total area, and its common border with the United States is the world's longest land border.

Calgary Downtown

Attribute Selectors



Selector	Matches	Example
[]	A specific attribute.	`[title]` Matches any element with a title attribute
[=]	A specific attribute with a specific value.	`a[title="posts from this country"]` Matches any <a> element whose title attribute is exactly `"posts from this country"`
[~=]	A specific attribute whose value matches at least one of the words in a space-delimited list of words.	`[title~="Countries"]` Matches any `title` attribute that contains the word `"Countries"`
[^=]	A specific attribute whose value begins with a specified value.	`a[href^="mailto"]` Matches any <a> element whose href attribute begins with `"mailto"`
[*=]	A specific attribute whose value contains a substring.	`img[src*="flag"]` Matches any <img> element whose src attribute contains somewhere within it the text `"flag"`
[$=]	A specific attribute whose value ends with a specified value.	`a[href$=".pdf"]` Matches any <a> element whose href attribute ends with the text ".pdf"

Pseudo Selectors

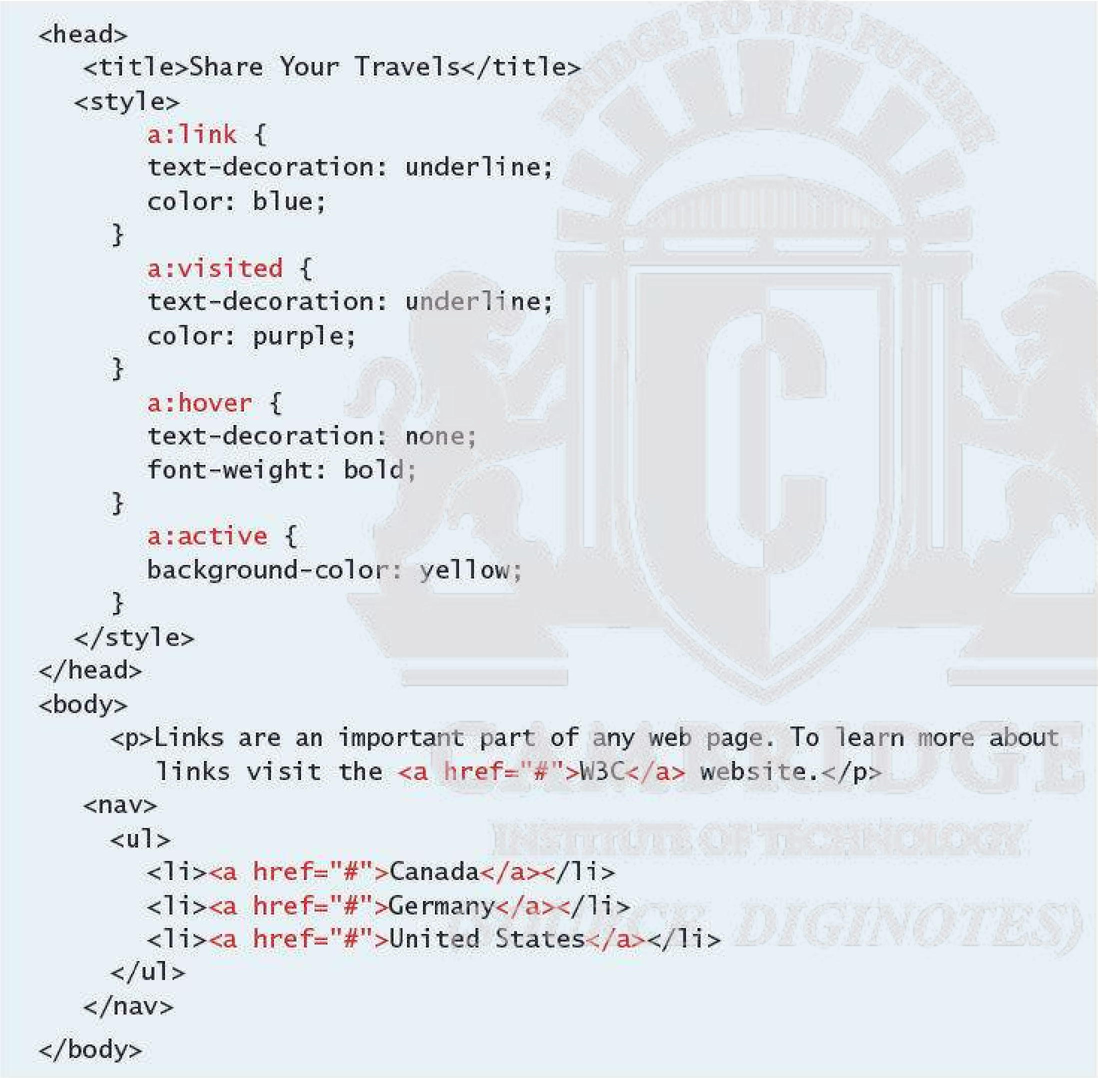
Select something that does not exist explicitly as an element

- ❖A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object.

- ❖A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.

- ❖The most common use of this type of selectors is for targeting link states.

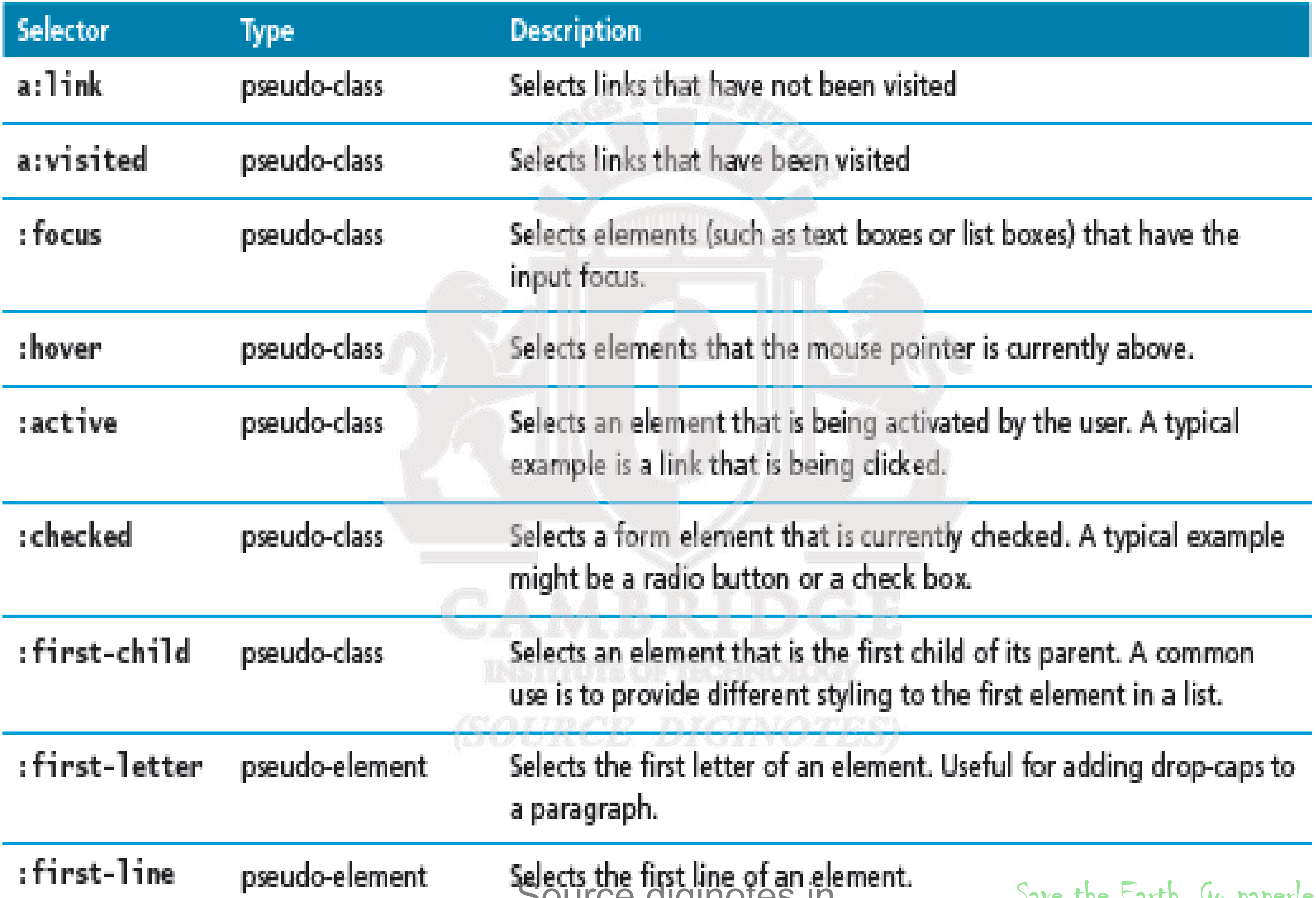
Pseudo Selectors



```
<head>
   <title>Share Your Travels</title>
  <style>
       a:link {
       text-decoration: underline;
       color: blue;
     }
       a:visited {
       text-decoration: underline;
       color: purple;
     }
       a:hover {
       text-decoration: none;
       font-weight: bold;
     }
       a:active {
       background-color: yellow;
     }
  </style>
</head>
<body>
     <p>Links are an important part of any web page. To learn more about
        links visit the <a href="#">W3C</a> website.</p>
   <nav>
     <ul>
       <li><a href="#">Canada</a></li>
       <li><a href="#">Germany</a></li>
       <li><a href="#">United States</a></li>
     </ul>
   </nav>
</body>
```

LISTING 3.8 Styling a link using pseudo-class selectors

Common Pseudo-Class and Pseudo-Element Selectors



Selector	Type	Description
a:link	pseudo-class	Selects links that have not been visited
a:visited	pseudo-class	Selects links that have been visited
:focus	pseudo-class	Selects elements (such as text boxes or list boxes) that have the input focus.
:hover	pseudo-class	Selects elements that the mouse pointer is currently above.
:active	pseudo-class	Selects an element that is being activated by the user. A typical example is a link that is being clicked.
:checked	pseudo-class	Selects a form element that is currently checked. A typical example might be a radio button or a check box.
:first-child	pseudo-class	Selects an element that is the first child of its parent. A common use is to provide different styling to the first element in a list.
:first-letter	pseudo-element	Selects the first letter of an element. Useful for adding drop-caps to a paragraph.
:first-line	pseudo-element	Selects the first line of an element.

Contextual Selectors

Select elements based on their ancestors, descendants, or siblings

- A **contextual selector** (in CSS3 also called **combinators**) allows you to select elements based on their ancestors, descendants, or siblings.

- That is, it selects elements based on their context or their relation to other elements in the document tree.

Contextual Selectors

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element	div p Selects a <p> element that is contained somewhere within a <div> element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct child of the specified element	div>h2 Selects an <h2> element that is a child of a <div> element.
Adjacent Sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element.	h3+p Selects the first <p> after any <h3>.
General Sibling	A specified element that shares the same parent as the specified element.	h3~p Selects all the <p> elements that share the same parent as the <h3>.

Source diginotes.in

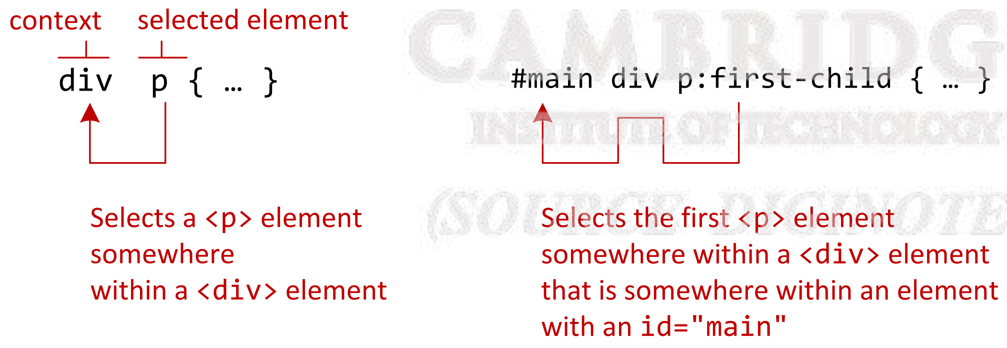
Save the Earth. Go paperless

Descendant Selector

Selects all elements that are contained within another element

❖While some of these contextual selectors are used relatively infrequently, almost all web authors find themselves using descendant selectors.

❖A **descendant selector** matches all elements that are contained within another element. The character used to indicate descendant selection is the space character.



context

selected element

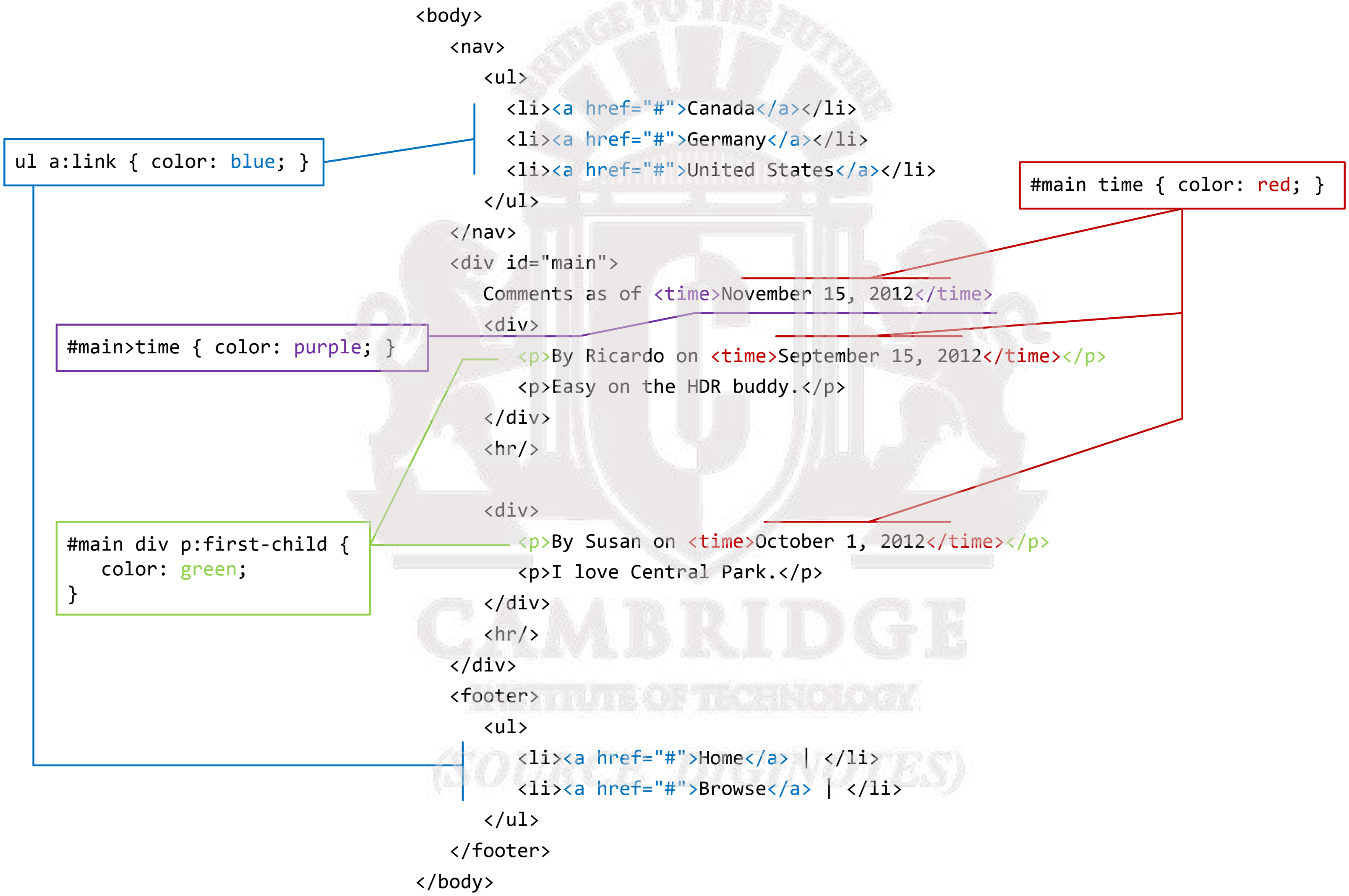
```
div  p { … }
```

Selects a <p> element somewhere within a <div> element

```
#main div p:first-child { … }
```

Selects the first <p> element somewhere within a <div> element that is somewhere within an element with an id="main"

Contextual Selectors in Action



```
<body>
   <nav>
      <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
      </ul>
   </nav>
   <div id="main">
      Comments as of <time>November 15, 2012</time>
      <div>
         <p>By Ricardo on <time>September 15, 2012</time></p>
         <p>Easy on the HDR buddy.</p>
      </div>
      <hr/>

      <div>
         <p>By Susan on <time>October 1, 2012</time></p>
         <p>I love Central Park.</p>
      </div>
      <hr/>
   </div>
   <footer>
      <ul>
         <li><a href="#">Home</a> | </li>
         <li><a href="#">Browse</a> | </li>
      </ul>
   </footer>
</body>
```

```
ul a:link { color: blue; }
```

```
#main time { color: red; }
```

```
#main>time { color: purple; }
```

```
#main div p:first-child {
   color: green;
}
```

Section 5 of 7

THE CASCADE: HOW STYLES INTERACT

Why Conflict Happens

In CSS that is

Because

- there are three different types of style sheets (author-created, user-defined, and the default browser style sheet),
- author-created stylesheets can define multiple rules for the same HTML element,

CSS has a system to help the browser determine how to display elements when different style rules conflict.

Cascade

How conflicting rules are handled in CSS

❑The "Cascade" in CSS refers to how conflicting rules are handled.

❑The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks.

❑The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.

Cascade Principles

CSS uses the following cascade principles to help it deal with conflicts:

- **inheritance,**
- **specificity,**
- **location**

Inheritance

Cascade Principle #1

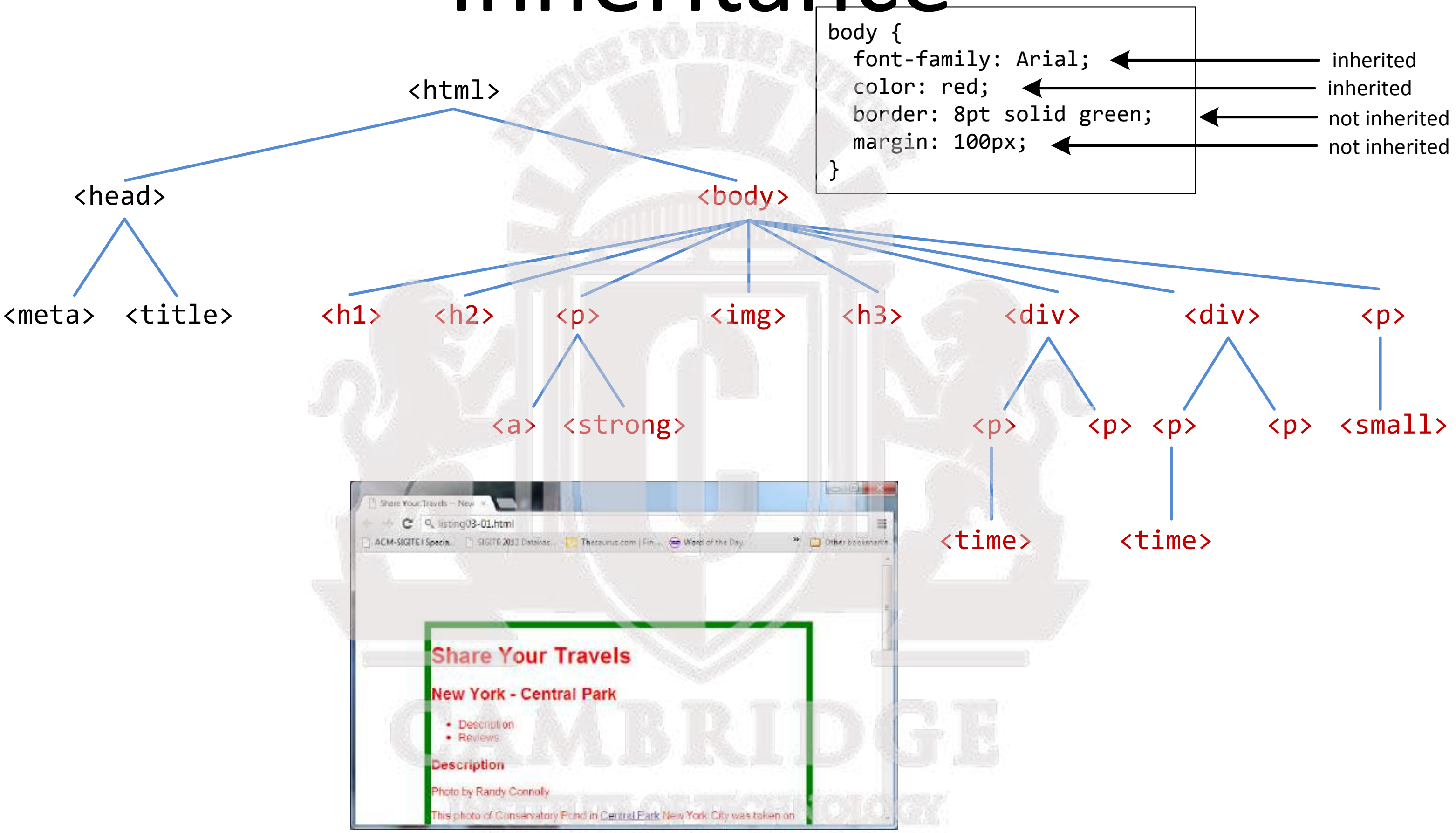
❖Many (but not all) CSS properties affect not only themselves but their descendants as well.

❖Font, color, list, and text properties are inheritable.

❖Layout, sizing, border, background and spacing properties are not.

Inheritance

```
body {
   font-family: Arial;
   color: red;
   border: 8pt solid green;
   margin: 100px;
}
```



Share Your Travels

New York - Central Park

- Description
- Reviews

Description

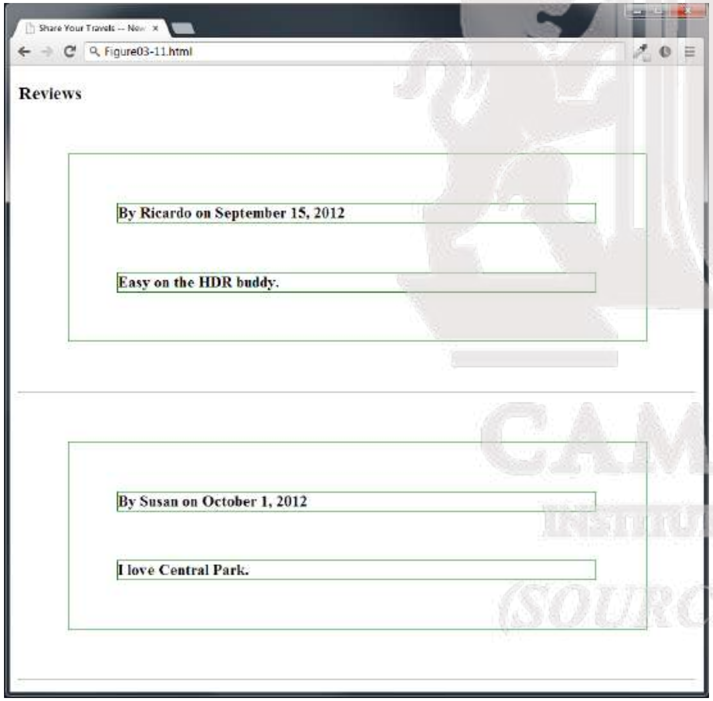
Photo by Randy Connolly

This photo of Conservatory Pond in Central Park New York City was taken on

Inheritance

How to force inheritance

It is possible to tell elements to inherit properties that are normally not inheritable.



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

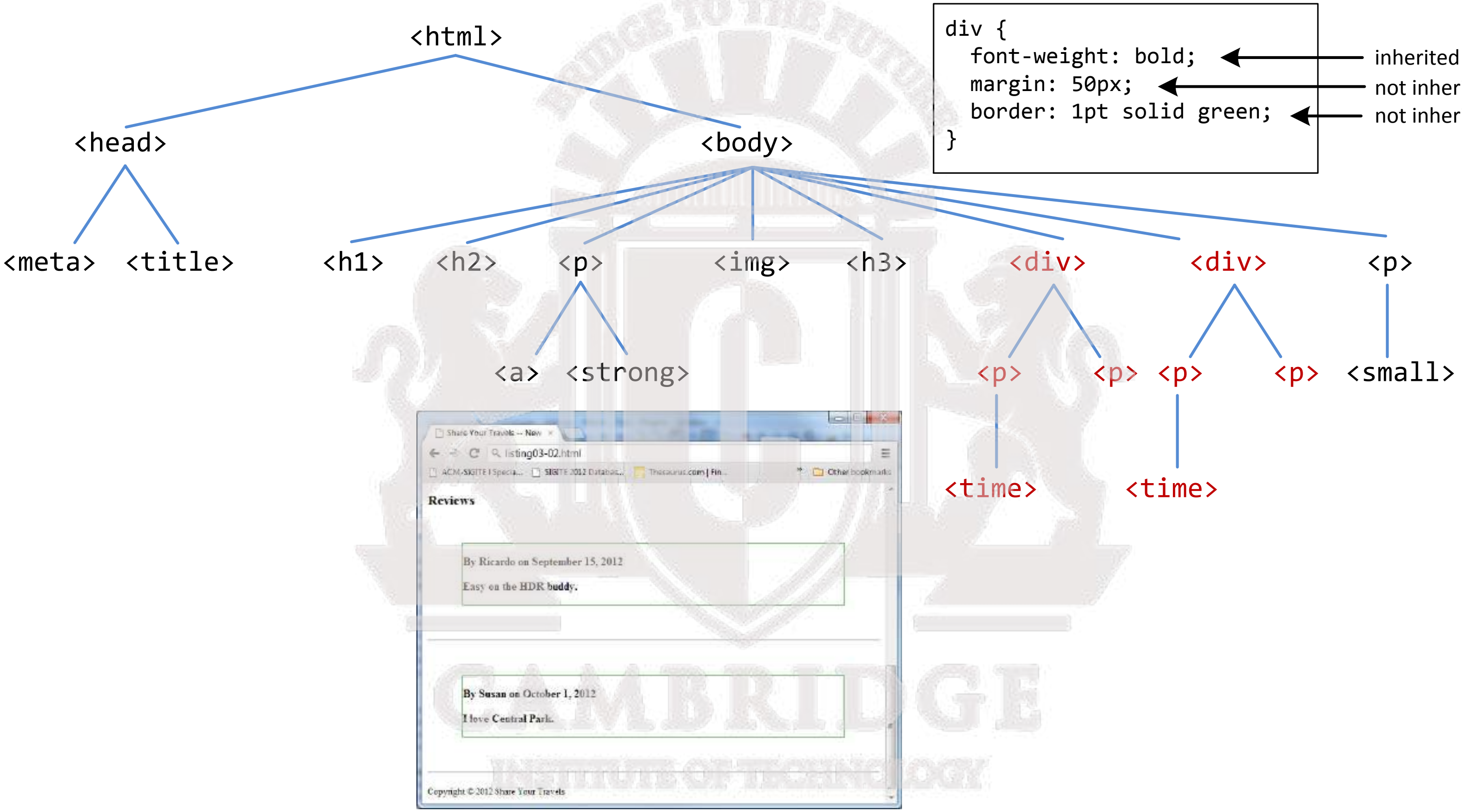
By Susan on October 1, 2012

I love Central Park.

```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
p {
  border: inherit;
  margin: inherit;
}
```

```
<h3>Reviews</h3>
<div>
   <p>By Ricardo on <time>September 15, 2012</time></p>
   <p>Easy on the HDR buddy.</p>
</div>
<hr/>

<div>
   <p>By Susan on <time>October 1, 2012</time></p>
   <p>I love Central Park.</p>
</div>
<hr/>
```

Inheritance



Share Your Travels -- New

listing03-02.html

ACM-SIGITE | Specia... SIGITE 2012 Databas... Thesaurus.com | Fin... Other bookmarks

Reviews

By Ricardo on September 15, 2012

Easy on the HDR **buddy.**

By Susan on October 1, 2012

I love Central Park.

Copyright © 2012 Share Your Travels

```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
```

- font-weight: bold; ← inherited
- margin: 50px; ← not inherited
- border: 1pt solid green; ← not inherited

Specificity
Cascade Principle #2

❖**Specificity** is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element.

❖The more *specific* the selector, the more it takes precedence (i.e., overrides the previous definition).

Specificity

How it works

❖The way that specificity works in the browser is that the browser assigns a weight to each style rule.

❖When several rules apply, the one with the greatest weight takes precedence.

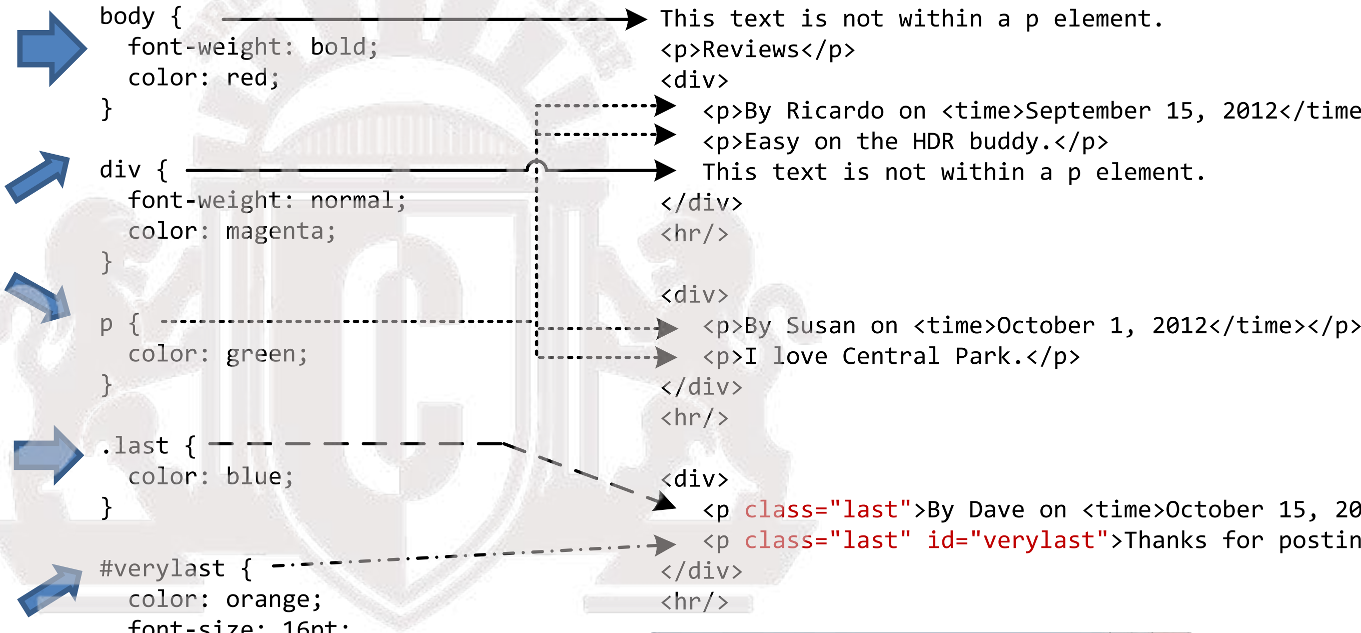
Specificity

These color and font-weight properties are inheritable and thus potentially applicable to all the child elements contained within the body.

However, because the <div> and <p> elements also have the same properties set, they *override* the value defined for the <body> element because their selectors (div and p) are more specific.

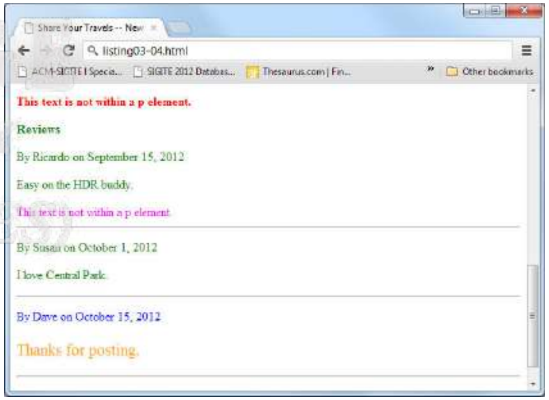
Class selectors are more specific than element selectors, and thus take precedence and override element selectors.

Id selectors are more specific than class selectors, and thus take precedence and override class selectors.



```
body {
  font-weight: bold;
  color: red;
}

div {
  font-weight: normal;
  color: magenta;
}

p {
  color: green;
}

.last {
  color: blue;
}

#verylast {
  color: orange;
  font-size: 16pt;
}
```

```
This text is not within a p element.
<p>Reviews</p>
<div>
   <p>By Ricardo on <time>September 15, 2012</time
   <p>Easy on the HDR buddy.</p>
   This text is not within a p element.
</div>
<hr/>

<div>
   <p>By Susan on <time>October 1, 2012</time></p>
   <p>I love Central Park.</p>
</div>
<hr/>

<div>
   <p class="last">By Dave on <time>October 15, 20
   <p class="last" id="verylast">Thanks for postin
</div>
<hr/>
```



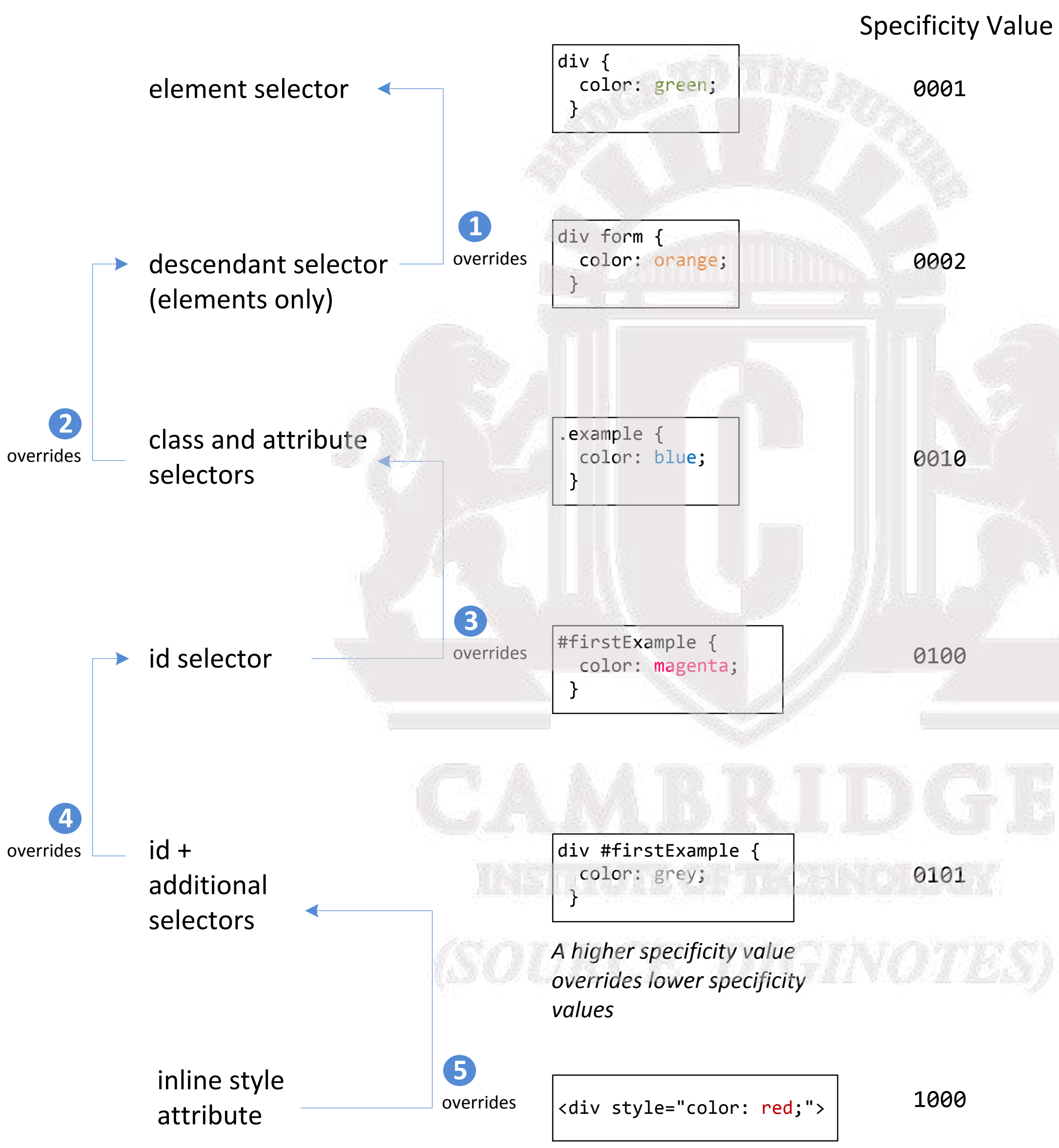
```
This text is not within a p element.

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

This text is not within a p element.
────────────────────────────────────
By Susan on October 1, 2012

I love Central Park.
────────────────────────────────────
By Dave on October 15, 2012

Thanks for posting.
────────────────────────────────────
```

Specificity Algorithm

The algorithm that is used to determine specificity is :

- ❖First count 1 if the declaration is from a 'style' attribute in the HTML, 0 otherwise (let that value = a).
- ❖Count the number of ID attributes in the selector (let that value = b).
- ❖Count the number of other attributes and pseudo-classes in the selector (let that value = c).
- ❖Count the number of element names and pseudo-elements in the selector (let that value = d).
- ❖Finally, concatenate the four numbers a+b+c+d together to calculate the selector's specificity.

Specificity Algorithm



Specificity Value

element selector

```
div {
  color: green;
}
```

0001

1 overrides

descendant selector (elements only)

```
div form {
  color: orange;
}
```

0002

2 overrides

class and attribute selectors

```
.example {
  color: blue;
}
```

0010

3 overrides

id selector

```
#firstExample {
  color: magenta;
}
```

0100

4 overrides

id + additional selectors

```
div #firstExample {
  color: grey;
}
```

0101

A higher specificity value overrides lower specificity values

5 overrides

inline style attribute

```
<div style="color: red;">
```

1000

Source diginotes.in

Save the Earth. Go paperless

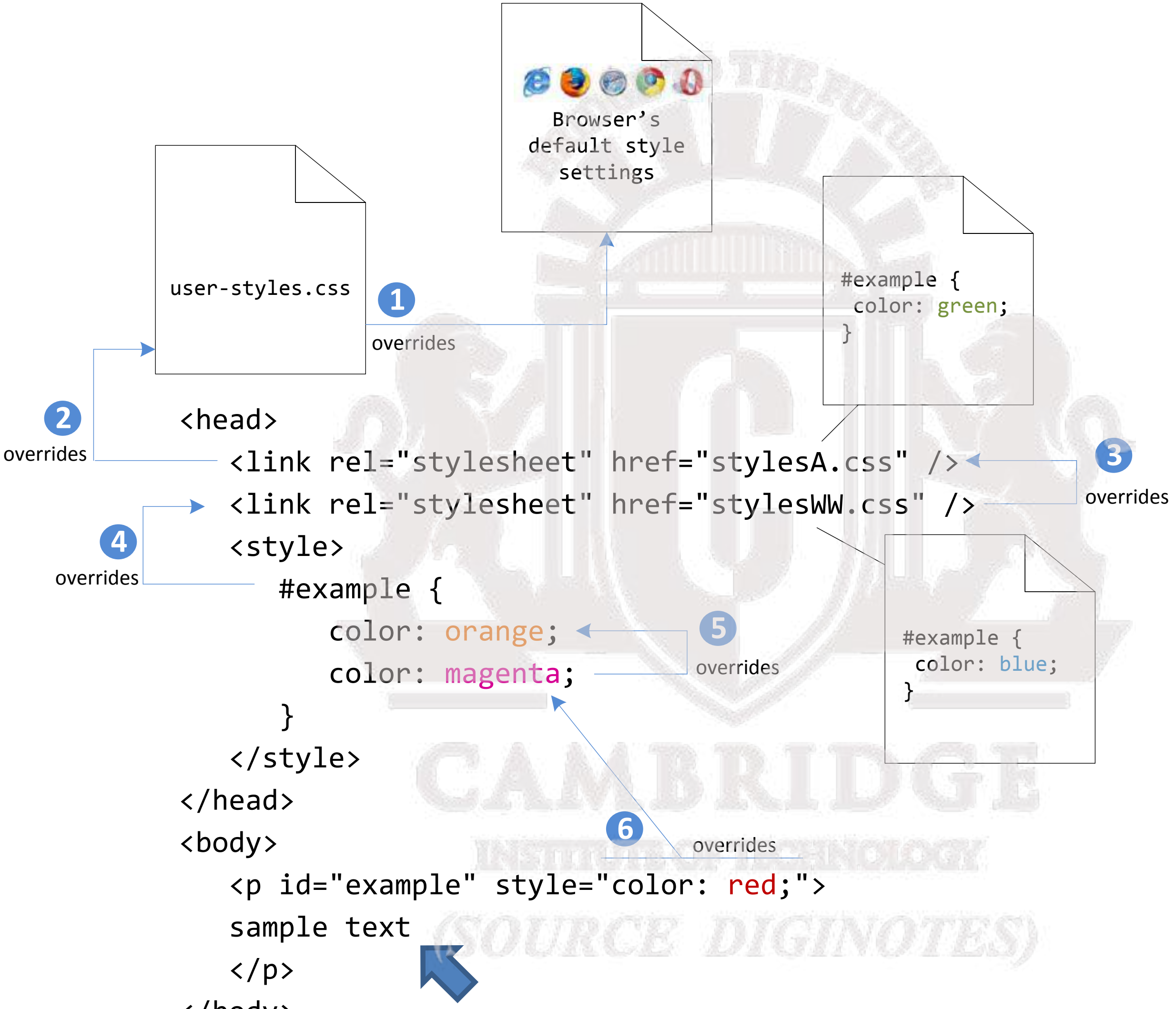
Location

Cascade Principle #3

❖When inheritance and specificity cannot determine style precedence, the principle of location will be used.

❖The principle of location is that when rules have the same specificity, then the latest are given more weight.

Location



```
#example {
 color: green;
}
```

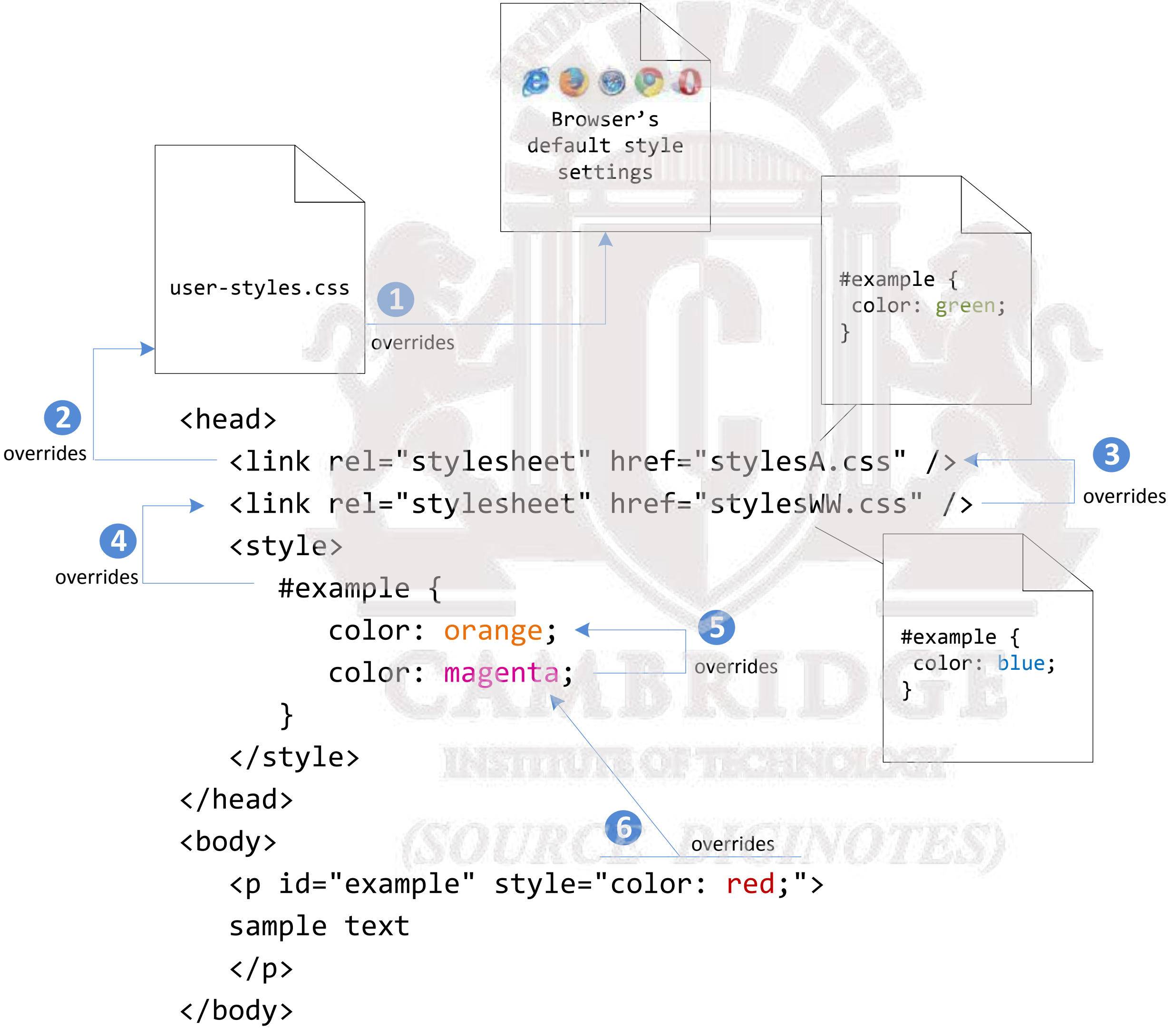
```
#example {
 color: blue;
}
```

```
<head>
   <link rel="stylesheet" href="stylesA.css" />
   <link rel="stylesheet" href="stylesWW.css" />
   <style>
      #example {
         color: orange;
         color: magenta;
      }
   </style>
</head>
<body>
   <p id="example" style="color: red;">
   sample text
   </p>
</body>
```

Can you guess what will be the color of the sample text ?

Location

What color would the sample text be if there wasn't an inline style definition?



Browser's default style settings

user-styles.css

1 overrides

2 overrides

```
#example {
 color: green;
}
```

3 overrides

```
#example {
 color: blue;
}
```

4 overrides

5 overrides

6 overrides

```
<head>
   <link rel="stylesheet" href="stylesA.css" />
   <link rel="stylesheet" href="stylesWW.css" />
   <style>
      #example {
         color: orange;
         color: magenta;
      }
   </style>
</head>
<body>
   <p id="example" style="color: red;">
   sample text
   </p>
</body>
```

Location

There's always an exception

- ❑There is one exception to the principle of location.

- ❑If a property is marked with !important in an author-created style rule, then it will override any other author-created style regardless of its location.

- ❑The only exception is a style marked with !important in an user style sheet; such a rule will override all others.

Section 6 of 7

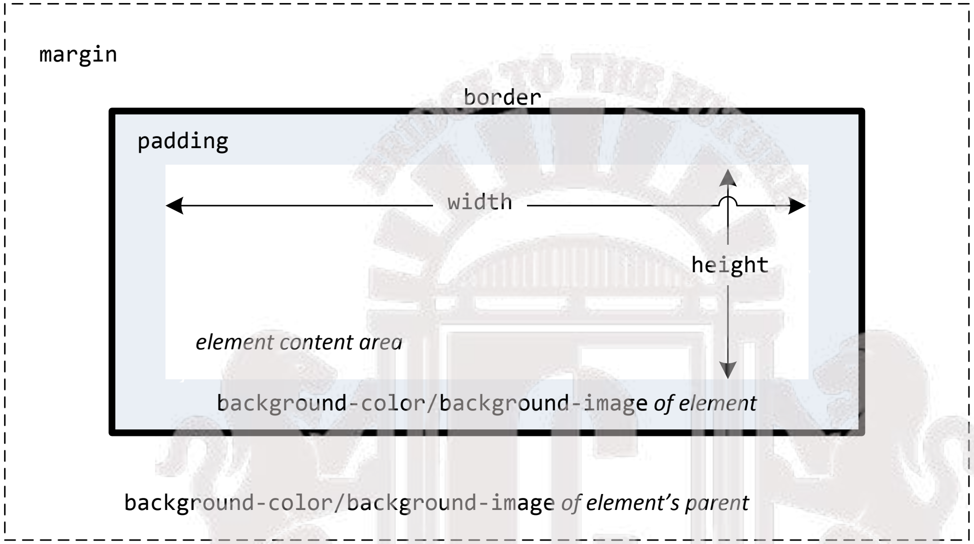
THE BOX MODEL

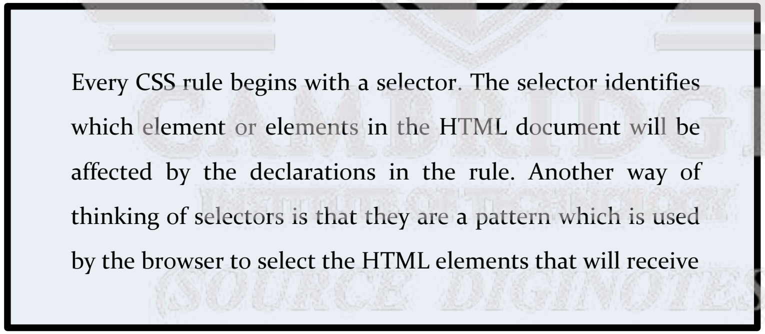
The Box Model

Time to think inside the box

- In CSS, all HTML elements exist within an **element box**.
- It is absolutely essential that you familiarize yourself with the terminology and relationship of the CSS properties within the element box.

The Box Model


margin
border
padding
width
height
element content area
background-color/background-image of element
background-color/background-image of element's parent



Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Background

Box Model Property #1

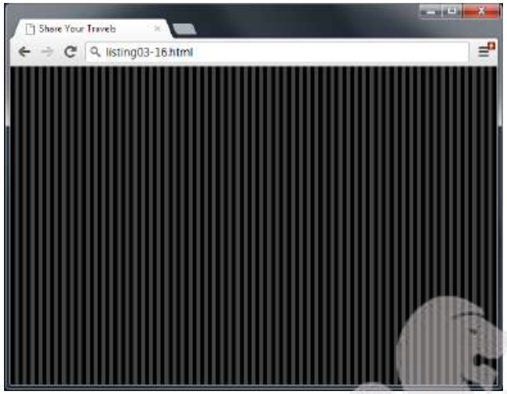
❖The background color or image of an element fills an element out to its border (if it has one that is).

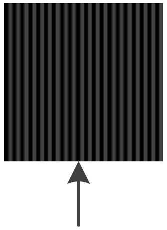
❖In contemporary web design, it has become extremely common too use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc) rather than using the <img> element.

Background Properties

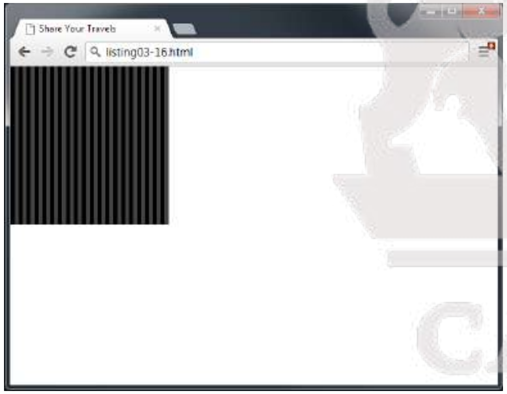
Property	Description
background	A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short-hand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. Possible values are: fixed, scroll.
background-color	Sets the background color of the element.
background-image	Specifies the background image (which is generally a jpeg, gif, or png file) for the element. Note that the URL is relative to the CSS file and not the HTML. CSS3 introduced the ability to specify multiple background images.
background-position	Specifies where on the element the background image will be placed. Some possible values include: bottom, center, left, and right. You can also supply a pixel or percentage numeric position value as well. When supplying a numeric value, you must supply a horizontal/vertical pair; this value indicates its distance from the top left corner of the element.
background-repeat	Determines whether the background image will be repeated. This is a common technique for creating a tiled background (it is in fact the default behavior). Possible values are: repeat, repeat-x, repeat-y, and no-repeat.
background-size	New to CSS3, this property lets you modify the size of the background image.

Background Repeat

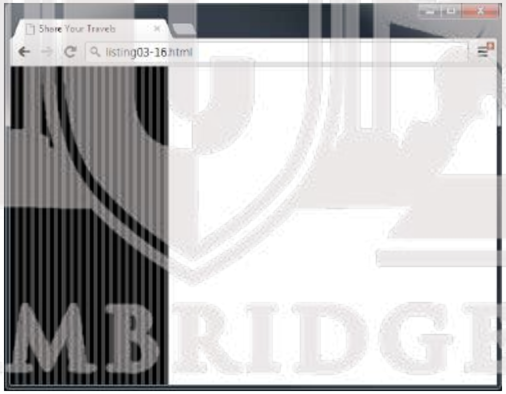




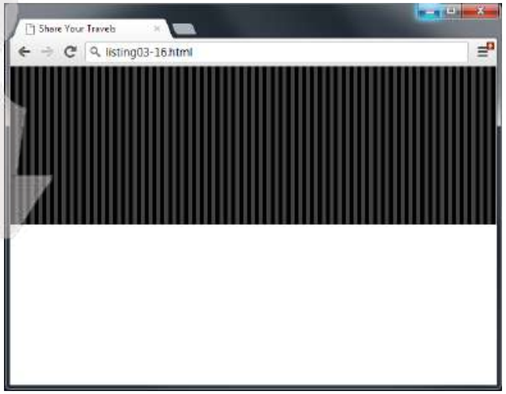
```
background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;
```



```
background-repeat: no-repeat;
```

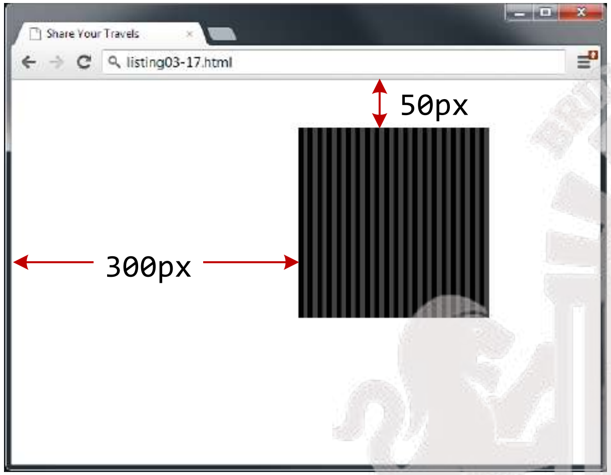


```
background-repeat: repeat-y;
```



```
background-repeat: repeat-x;
```

Background Position



```
body {
        background: white url(../images/backgrounds/body-background-tile.gif) no-repeat;
        background-position: 300px 50px;
}
```

Borders

Box Model Property #2

- Borders provide a way to visually separate elements.
- You can put borders around all four sides of an element, or just one, two, or three of the sides.

Borders

Property	Description
border	A combined short-hand property that allows you to set the style, width, and color of a border in one property. The order is important and must be: **border-style border-width border-color**
border-style	Specifies the line type of the border. Possible values are: solid, dotted, dashed, double, groove, ridge, inset, and outset.
border-width	The width of the border in a unit (but not percents). A variety of keywords (thin, medium, etc) are also supported.
border-color	The color of the border in a color unit.
border-radius	The radius of a rounded corner.
border-image	The URL of an image to use as a border.

Shortcut notation

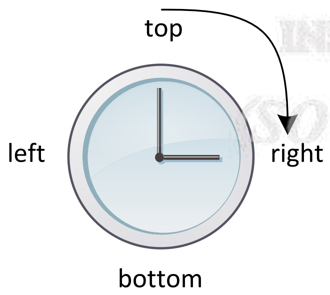
TRBL

With border, margin, and padding properties, there are long-form and shortcut methods to set the 4 sides

```
border-top-color: red;            /* sets just the top side */
border-right-color: green;        /* sets just the right side */
border-bottom-color: yellow;      /* sets just the bottom side */
border-left-color: blue;          /* sets just the left side */

border-color: red;                /* sets all four sides to red */

border-color: red green orange blue;      /* sets all four sides differently */
```

When using this multiple values shortcut, they are applied in clockwise order starting at the top.
Thus the order is: **top right bottom left.**

TRBL (Trouble)

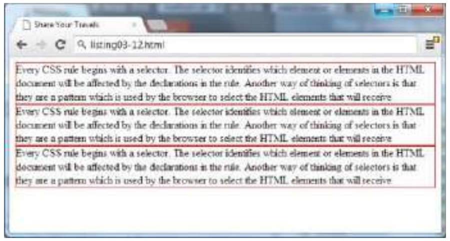


```
border-color: top right bottom left;
```

```
border-color: red green orange blue;
```

Margins and Padding

Box Model Properties #3 and #4

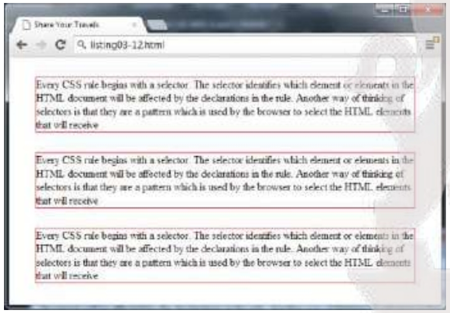


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 0;
   padding: 0;
}
```

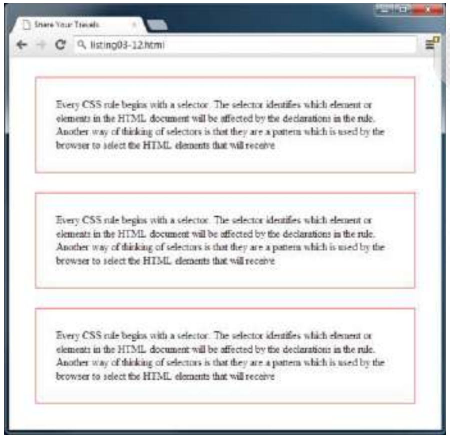


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 0;
}
```



Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

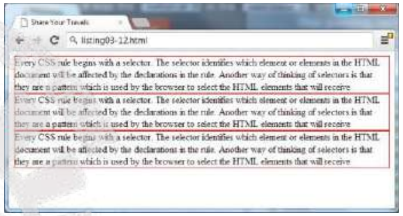
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 30px;
}
```

Margins

Why they will cause you trouble.



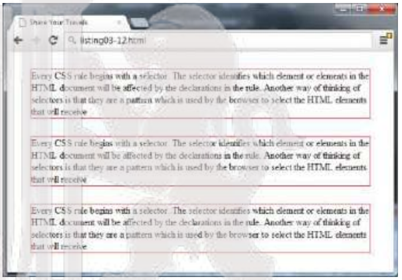
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 0;
   padding: 0;
}
```

Did you notice that the space between paragraphs one and two and between two and three is the same as the space before paragraph one and after paragraph three?



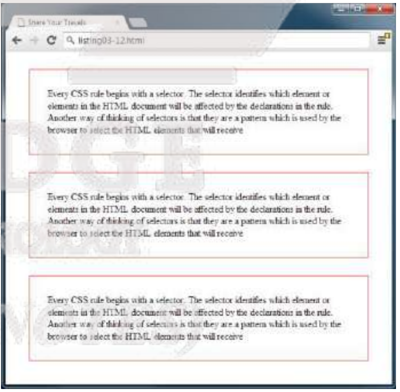
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 0;
}
```

This is due to the fact that adjoining vertical margins collapse.



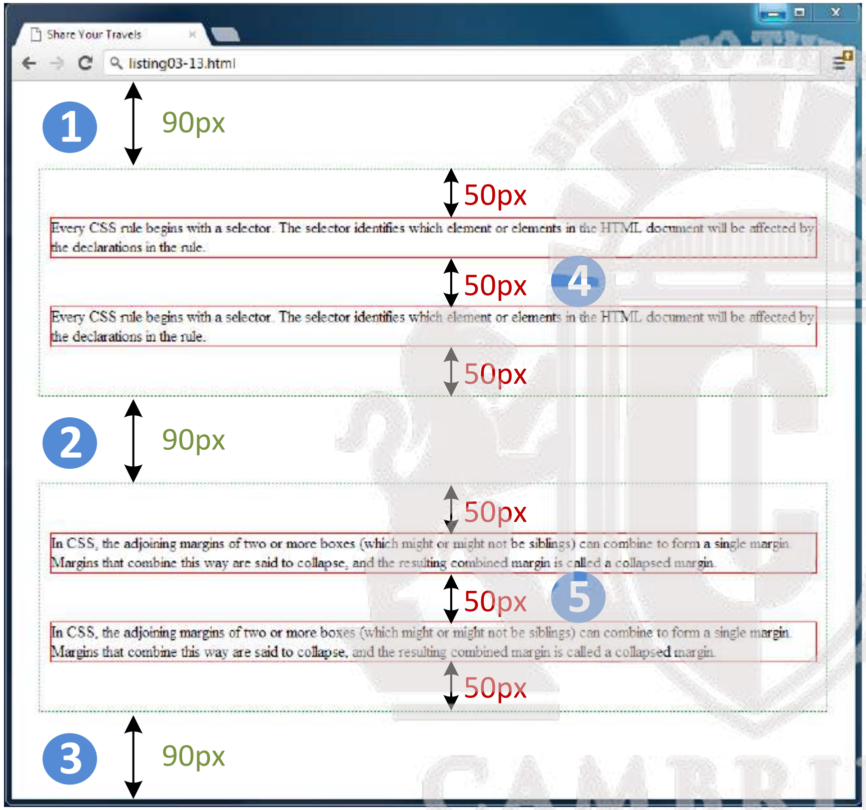
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 30px;
}
```

Collapsing Margins



Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

In CSS, the adjoining margins of two or more boxes (which might or might not be siblings) can combine to form a single margin. Margins that combine this way are said to collapse, and the resulting combined margin is called a collapsed margin.

In CSS, the adjoining margins of two or more boxes (which might or might not be siblings) can combine to form a single margin. Margins that combine this way are said to collapse, and the resulting combined margin is called a collapsed margin.

```
<div>
  <p>Every CSS rule ...</p>
  <p>Every CSS rule ...</p>
</div>
<div>
  <p>In CSS, the adjoining ... </p>
  <p>In CSS, the adjoining ... </p>
</div>
```

```
div {
   border: dotted 1pt green;
   padding: 0;
   margin: 90px 20px;
}
```

```
p {
   border: solid 1pt red;
   padding: 0;
   margin: 50px 20px;
}
```

If overlapping margins did not collapse, then margin space for 2 would be 180p (90pixels for the bottom margin of the first <div> + 90 pixels for the top margin of the second <div>), while the margins 4 and 5 for would be 100px.

However, as you can see this is not the case.

Collapsing Margins

How it works

When the **vertical** margins of two elements touch,

- the largest margin value of the elements will be displayed
- the smaller margin value will be collapsed to zero.

Horizontal margins, on the other hand, **never** collapse.

To complicate matters even further, there are a large number of special cases in which adjoining vertical margins do **not** collapse.

Width and Height

Box Model Properties #5 and #6

- ❖The width and height properties specify the size of the element’s content area.

- ❖Perhaps the only rival for collapsing margins in troubling our students, box dimensions have a number of potential issues.

Width and Height

Potential Problem #1

□Only block-level elements and non-text inline elements such as images have a **width** and **height** that you can specify.

□By default the width of and height of elements is the actual size of the content.

□For text,

- this is determined by the font size and font face;

For images,

- the width and height of the actual image in pixels determines the element box’s dimensions.

Width and Height

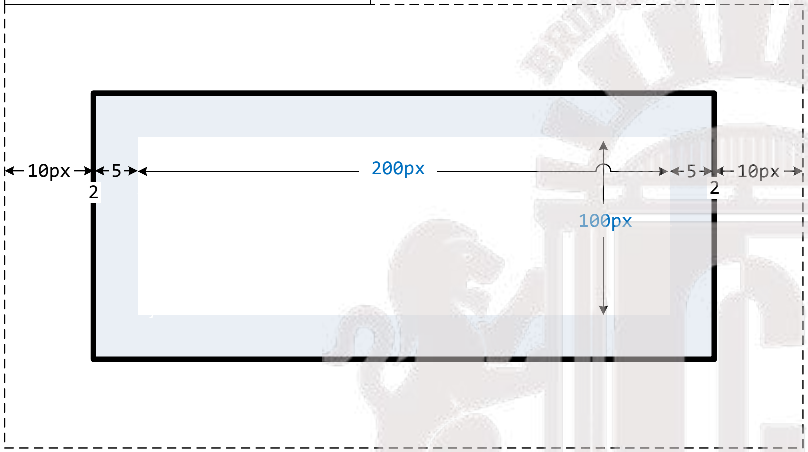
Potential Problem #2

✓Since the width and the height refer to the size of the content area, by default, the total size of an element is equal to not only its content area, but also to the sum of its padding, borders, and margins.

```
div {
  box-sizing: content-box;
  width: 200px;
  height: 100px;
  padding: 5px;
  margin: 10px;
  border: solid 2pt black;
}
```

True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px

True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px

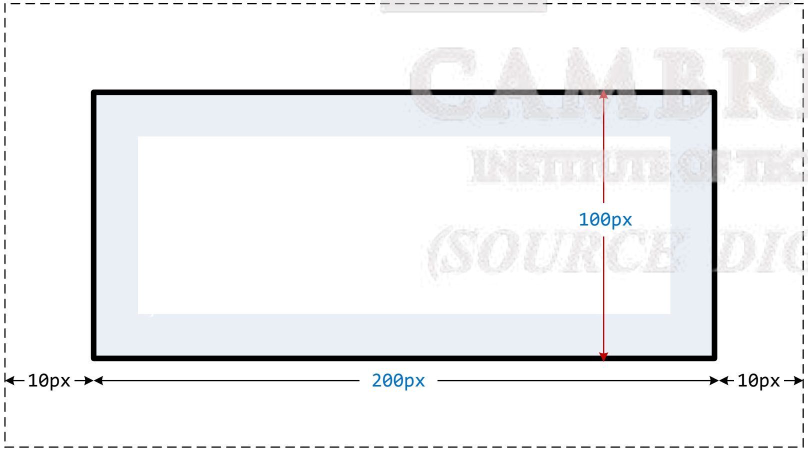


Default

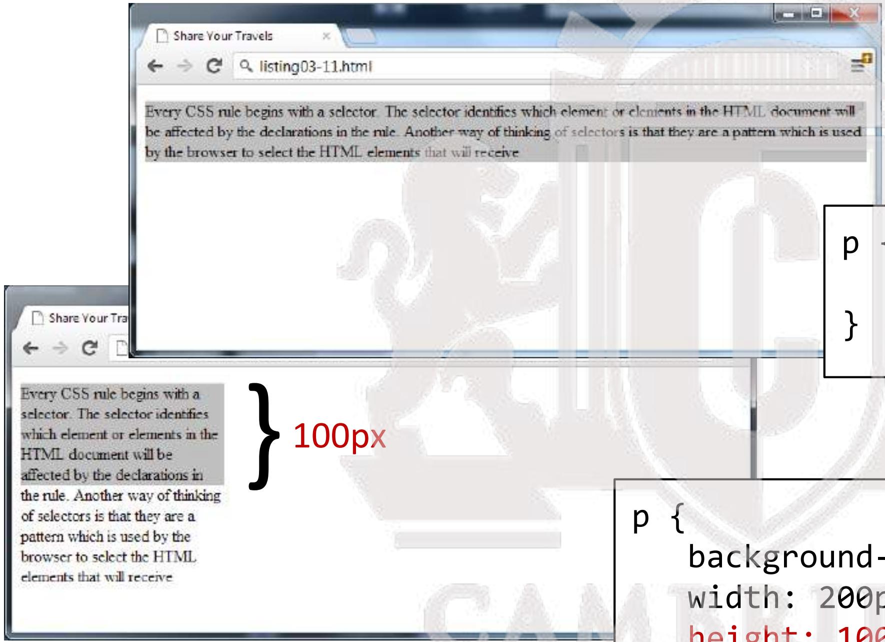
```
div {
  ...
  box-sizing: border-box;
}
```

True element width = 10 + 200 + 10 = 220 px

True element height = 10 + 100 + 10 = 120 px



Width and Height



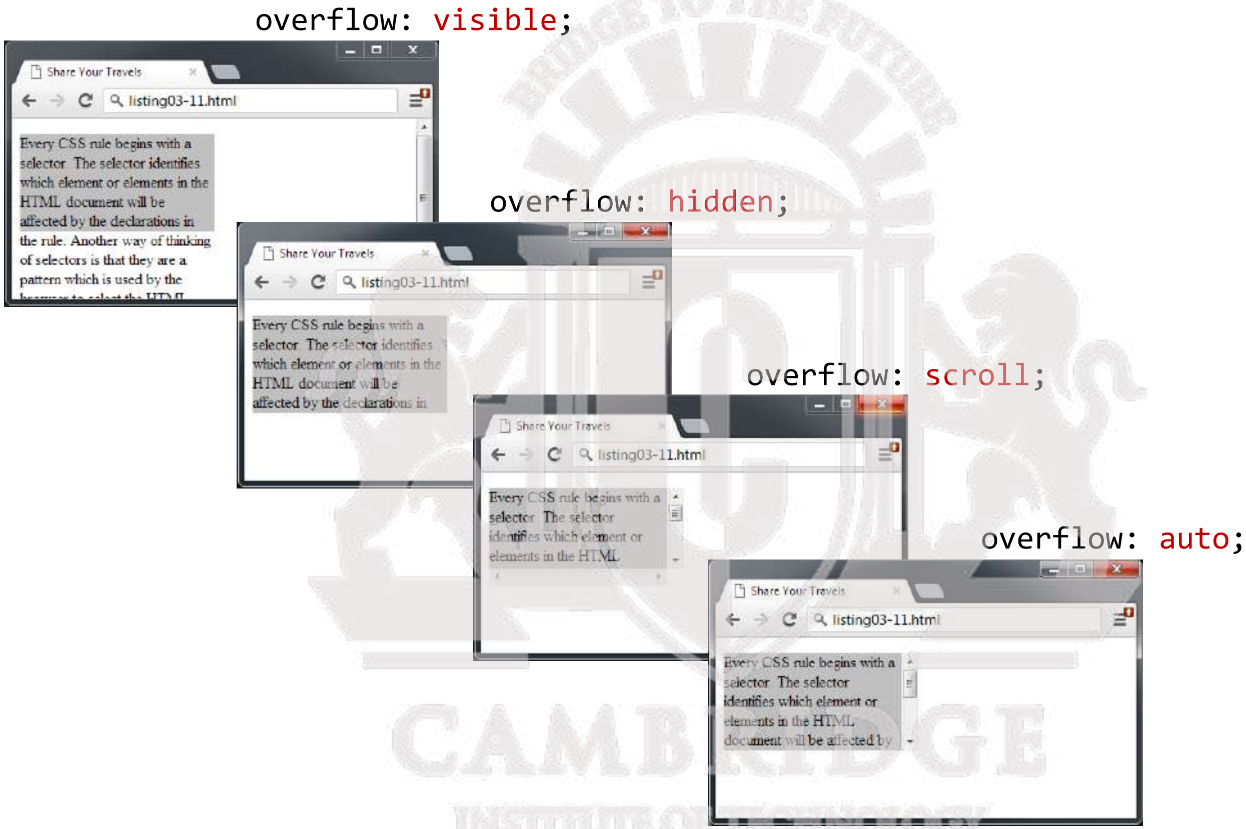
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

```
p {
   background-color: silver;
}
```

```
p {
   background-color: silver;
   width: 200px;
   height: 100px;
}
```

Overflow Property



`overflow: visible;`

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML

`overflow: hidden;`

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in

`overflow: scroll;`

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML

`overflow: auto;`

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by

Sizing Elements

Time to embrace ems and percentages

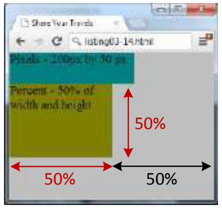
❖While the previous examples used pixels for its measurement, many contemporary designers prefer to use percentages or em units for widths and heights.

- When you use percentages, the size is relative to the size of the parent element.

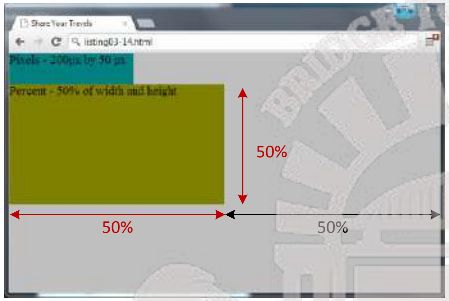
- When you use ems, the size of the box is relative to the size of the text within it.

The rationale behind using these relative measures is to make one's design scalable to the size of the browser or device that is viewing it.

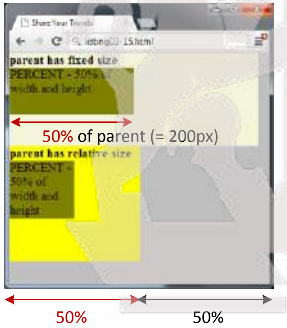
```
<style>
  html,body {
     margin:0;
     width:100%;
     height:100%;
     background: silver;
  }
  .pixels {
     width:200px;
     height:50px;
     background: teal;
  }
  .percent {
     width:50%;
     height:50%;
     background: olive;
  }
```



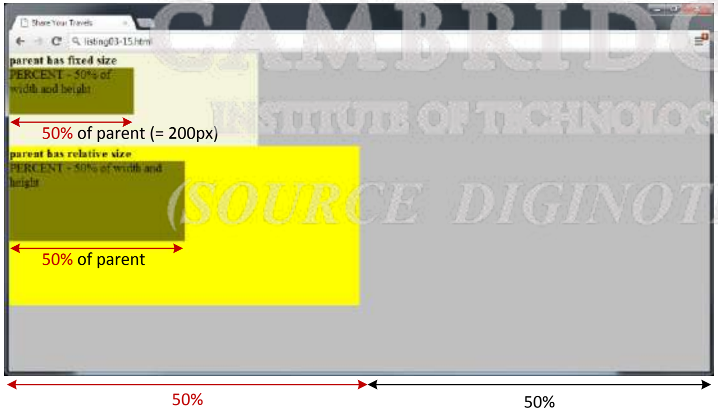
```
<body>
   <div class="pixels">
     Pixels - 200px by 50 px
   </div>
   <div class="percent">
     Percent - 50% of width and height
   </div>
</body>
```



```
  .parentFixed {
     width:400px;
     height:150px;
     background: beige;
  }
  .parentRelative {
     width:50%;
     height:50%;
     background: yellow;
  }
</style>
```



```
<body>
<div class="parentFixed">
   <strong>parent has fixed size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
<div class="parentRelative">
   <strong>parent has relative size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
</body>
```



Developer Tools

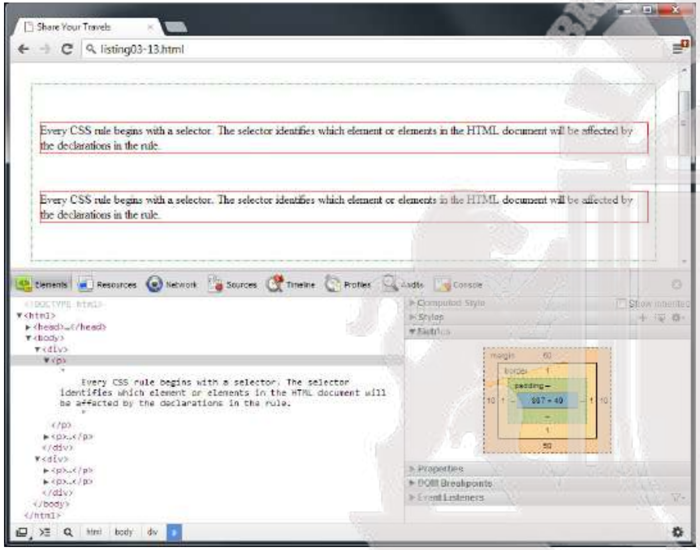
Help is on the way

❖Developer tools in current browsers make it significantly easier to examine and troubleshot CSS than was the case a decade ago.

❖You can use the various browsers' CSS inspection tools to examine, for instance, the box values for a selected element.

Developer Tools

Chrome – Inspect Element

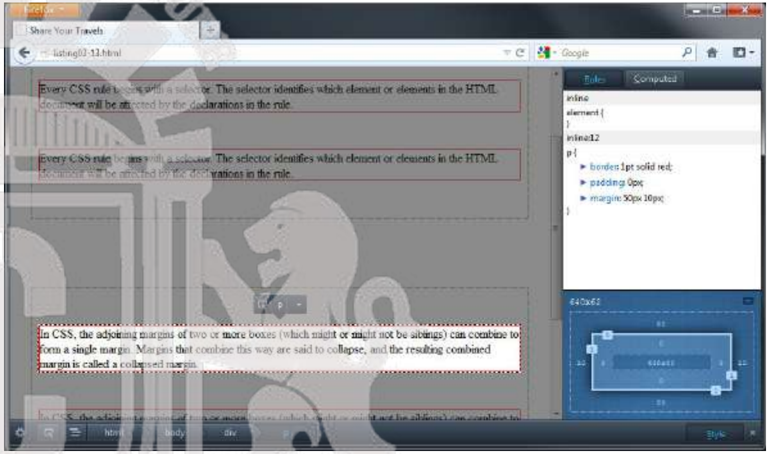


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<div>
      ▼<p>
          Every CSS rule begins with a selector. The selector
        identifies which element or elements in the HTML document will
        be affected by the declarations in the rule.
        </p>
      ▶<p>…</p>
      </div>
    ▼<div>
      ▶<p>…</p>
      ▶<p>…</p>
      </div>
    </body>
</html>
```

Firefox – Inspect



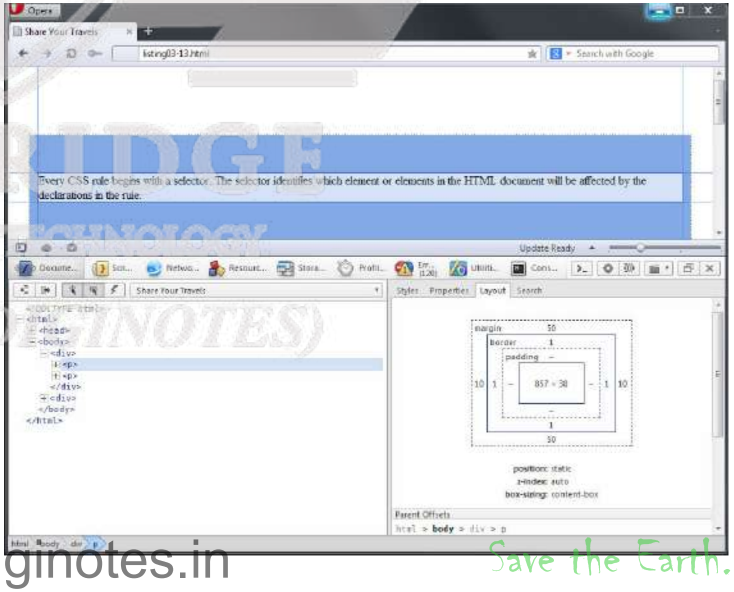
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

In CSS, the adjoining margins of two or more boxes (which might or might not be siblings) can combine to form a single margin. Margins that combine this way are said to collapse, and the resulting combined margin is called a collapsed margin.

```
inline
element {
}
inline:12
p {
  ▸ border: 1pt solid red;
  ▸ padding: 0px;
  ▸ margin: 50px 10px;
}
```

Opera – Inspect Element

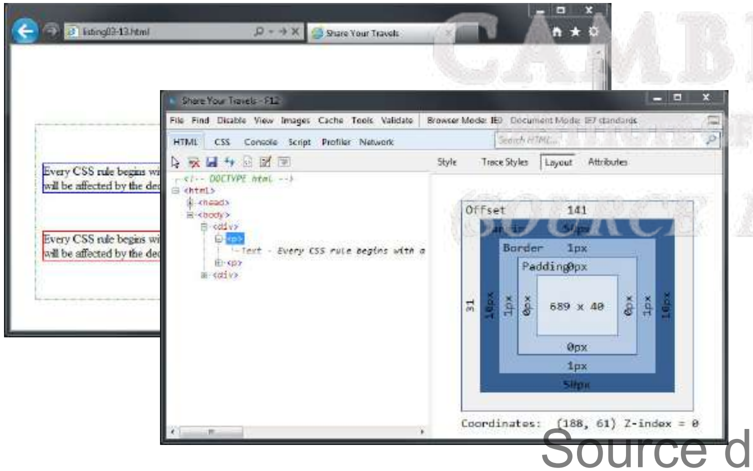


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

```
<!DOCTYPE html>
<html>
  <head>
  <body>
    <div>
      <p>
      <p>
    </div>
    <div>
  </body>
</html>
```

position: static
z-index: auto
box-sizing: content-box

Internet Explorer – Developer Tools



Every CSS rule begins wi... will be affected by the de...

Every CSS rule begins wi... will be affected by the de...

```
<!-- DOCTYPE html -->
<html>
  <head>
  <body>
    <div>
      <p>
        Text - Every CSS rule begins with a
      <p>
    <div>
```

Coordinates: (188, 61) Z-index = 0

Section 7 of 7

TEXT STYLING

Text Properties

Two basic types

CSS provides two types of properties that affect text.

- **font properties** that affect the font and its appearance.
- **paragraph properties** that affect the text in a similar way no matter which font is being used.

Font-Family

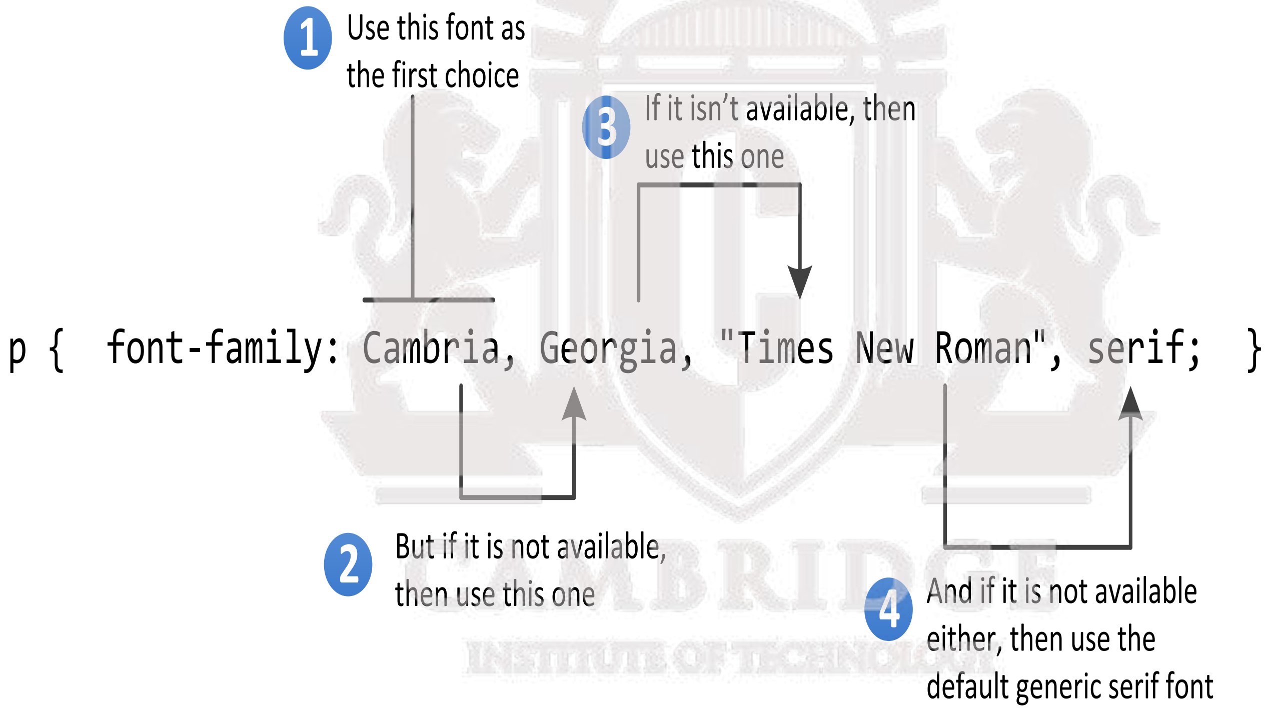
A few issues here

❖A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different.

❖However, just because a given font is available on the web developer's computer, it does not mean that that same font will be available for all users who view the site.

❖For this reason, it is conventional to supply a so-called **web font stack**, that is, a series of alternate fonts to use in case the original font choice in not on the user's computer.

Specifying the Font-Family



```
p {  font-family: Cambria, Georgia, "Times New Roman", serif;  }
```

1 Use this font as the first choice

2 But if it is not available, then use this one

3 If it isn't available, then use this one

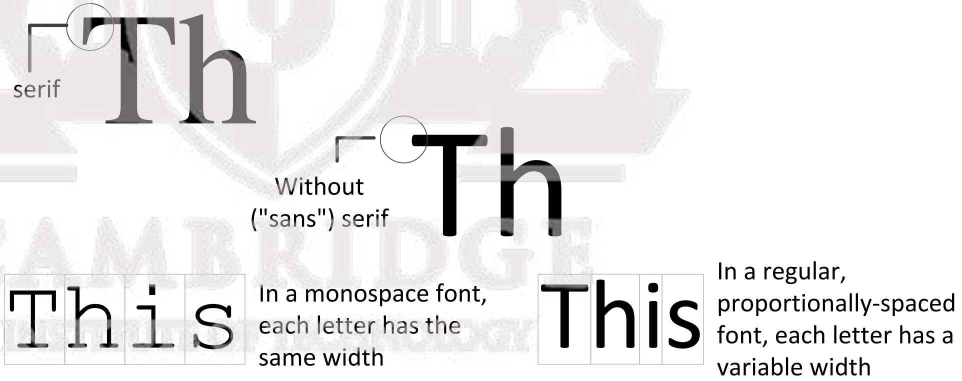
4 And if it is not available either, then use the default generic serif font

Generic Font-Family

The font-family property supports five different generic families.

The browser supports a typeface from each family.

	Generic Font-Family Name
This	serif
This	sans-serif
This	monospace
This	cursive
This	fantasy



In a monospace font, each letter has the same width

In a regular, proportionally-spaced font, each letter has a variable width

Decorative and cursive fonts vary from system to system; rarely used as a result.

@font-face

The future is now

- ❖Over the past few years, the most recent browser versions have begun to support the **@font-face** selector in CSS.
- ❖This selector allows you to use a font on your site even if it is not installed on the end user's computer.
- ❖Due to the on-going popularity of open source font sites such as Google Web Fonts (http://www.google.com/webfonts) and Font Squirrel (http://www.fontsquirrel.com/), @font-face seems to have gained a critical mass of widespread usage.

Font Sizes

Mo control, mo problems

The issue of font sizes is unfortunately somewhat tricky.

In a print-based program such as a word processor, specifying a font size in points is unproblematic.

However, absolute units such as points and inches do not translate very well to pixel-based devices.

Somewhat surprisingly, pixels are also a problematic unit.

  Newer mobile devices in recent years have been increasing pixel densities so that a given CSS pixel does not correlate to a single device pixel.

Font Sizes

Welcome ems and percents again

If we wish to create web layouts that work well on different devices, we should learn to use relative units such as **em** units or **percentages** for our font sizes (and indeed for other sizes in CSS as well).

One of the principles of the web is that the user should be able to change the size of the text if he or she so wishes to do so.

Using percentages or em units ensures that this user action will work.

How to use ems and percents

When used to specify a font size, both em units and percentages are relative to the parent’s font size.

How to use ems and percents

<body>	Browser’s default text size is usually 16 pixels
<p>	100% or 1em is 16 pixels
<h3>	125% or 1.125em is 18 pixels
<h2>	150% or 1.5em is 24 pixels
<h1>	200% or 2em is 32 pixels

```
/* using 16px scale */

body { font-size: 100%; }
h3 { font-size: 1.125em; }  /* 1.25 x 16 = 18 */
h2 { font-size: 1.5em; }    /* 1.5 x 16  = 24 */
h1 { font-size: 2em; }      /* 2 x 16 = 32 */
```

```
<body>
  <p>this will be about 16 pixels</p>
  <h1>this will be about 32 pixels</h1>
  <h2>this will be about 24 pixels</h2>
  <h3>this will be about 18 pixels</h3>
  <p>this will be about 16 pixels</p>
</body>
```

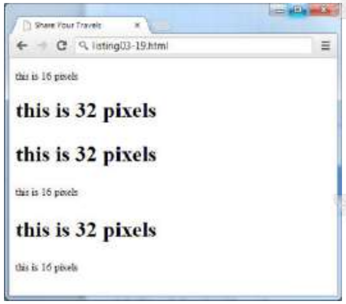
How to use ems and percents

It might seem easy … but it’s not …

- While this looks pretty easy to master, things unfortunately can quickly become quite complicated.
- Remember that percents and em units are relative to their parents, so if the parent font size changes, this affects all of its contents.

ems and percents

```
<body>
  <p>this is 16 pixels</p>
  <h1>this is 32 pixels</h1>
  <article>
     <h1>this is 32 pixels</h1>
     <p>this is 16 pixels</p>
     <div>
        <h1>this is 32 pixels</h1>
        <p>this is 16 pixels</p>
     </div>
  </article>
</body>
```



this is 16 pixels

this is 32 pixels

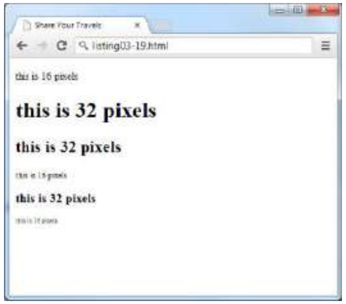
this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }       /* 1 x 16 = 16px */
h1   { font-size: 2em; }       /* 2 x 16 = 32px */
```



this is 16 pixels

this is 32 pixels

this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

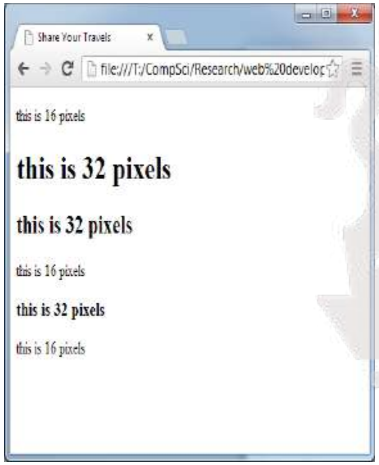
```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }
h1   { font-size: 2em; }

article { font-size: 75% }  /* h1 = 2 * 16 * 0.75 = 24px
                                p = 1 * 16 * 0.75 = 12px  */

div  { font-size: 75% }     /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                                p = 1 * 16 * 0.75 * 0.75 = 9px   */
```

The rem unit

Solution to font sizing hassles?

- CSS3 now supports a new relative measure, the **rem** (for root em unit).
- This unit is always relative to the size of the root element (i.e., the <html> element).
- However, since Internet Explorer prior to version 9 do not support the rem units, you need to provide some type of fallback for those browsers.

The rem unit



this is 16 pixels

this is 32 pixels

this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

```
/* using 16px scale */

body { font-size: 100%; }
p {
       font-size: 16px;  /* for older browsers: won’t scale properly though */
       font-size: 1rem;  /* for new browsers: scales and simple too */
}
h1   { font-size: 2em; }

article { font-size: 75% }  /* h1 = 2 * 16 * 0.75 = 24px
                               p = 1 * 16 = 16px  */

div  { font-size: 75% }      /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                                p = 1 * 16 = 16px   */
```

WEB TECHNOLOGY AND ITS APPLICATIONS

17CS71

Mr. GANESH D R
ASSISTANT PROFESSOR,
DEPT OF CSE, CITECH

WEB TECHNOLOGY AND ITS APPLICATIONS
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)
SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	**10 Hours**
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	**10 Hours**
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	**10 Hours**
Module – 4	
PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays, $_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	**10 Hours**
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	**10 Hours**

HTML Tables and Forms

Chapter 4

Section 1 of 6

INTRODUCING TABLES

HTML Tables

A grid of cells

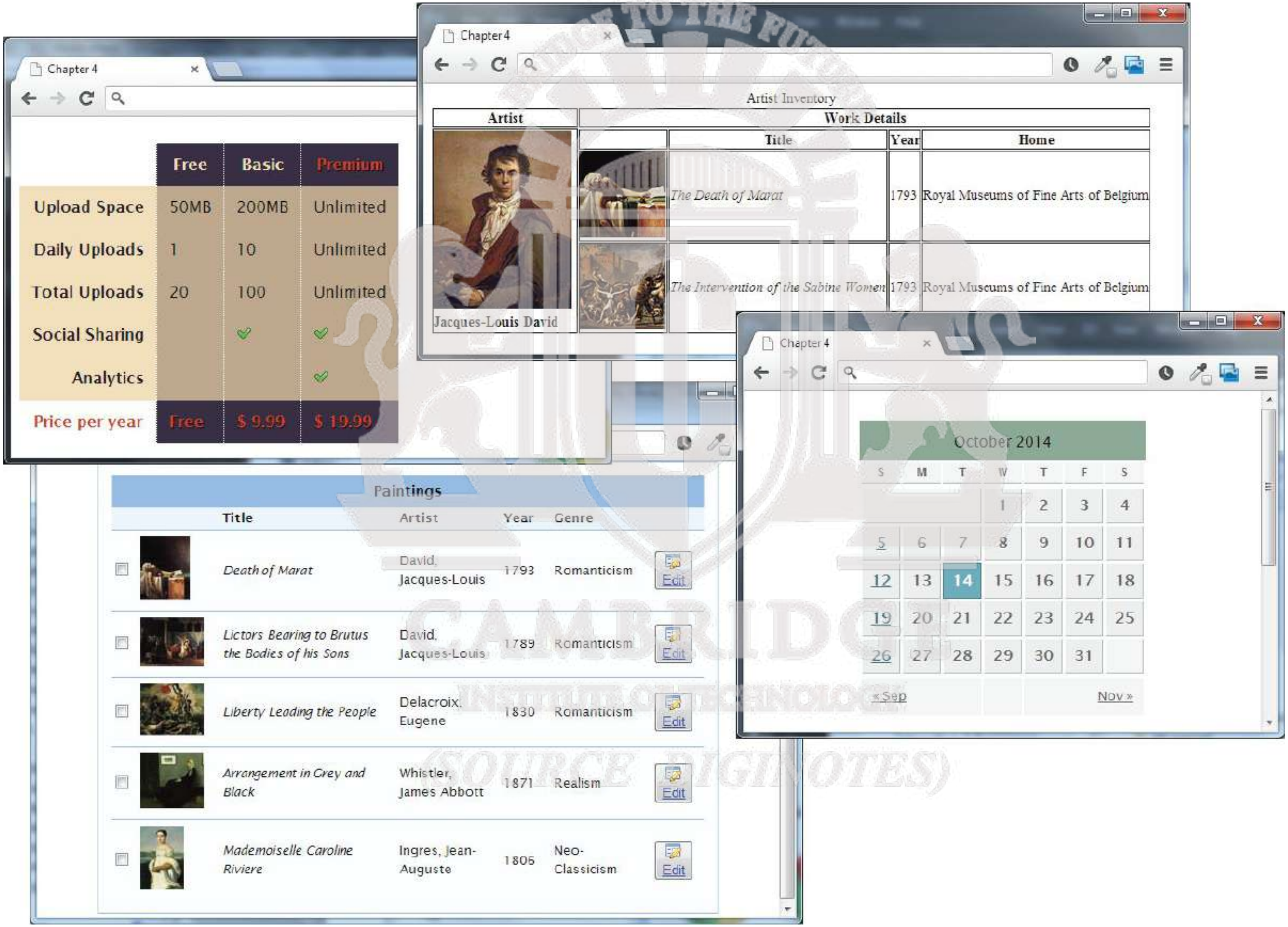
- ➢A table in HTML is created using the **<table>** element

Tables can be used to display:

- •Many types of content
  - •Calendars, financial data, lists, etc…
- •Any type of data
  - •Images
  - •Text
  - •Links
  - •Other tables

HTML Tables

Example usages



	Free	Basic	Premium
Upload Space	50MB	200MB	Unlimited
Daily Uploads	1	10	Unlimited
Total Uploads	20	100	Unlimited
Social Sharing		✓	✓
Analytics			✓
Price per year	Free	$ 9.99	$ 19.99

Artist Inventory

Artist	Work Details			
		Title	Year	Home
Jacques-Louis David		*The Death of Marat*	1793	Royal Museums of Fine Arts of Belgium
		The Intervention of the Sabine Women	1793	Royal Museums of Fine Arts of Belgium

Paintings

		Title	Artist	Year	Genre	
□		*Death of Marat*	David, Jacques-Louis	1793	Romanticism	Edit
□		*Lictors Bearing to Brutus the Bodies of his Sons*	David, Jacques-Louis	1789	Romanticism	Edit
□		*Liberty Leading the People*	Delacroix, Eugene	1830	Romanticism	Edit
□		*Arrangement in Grey and Black*	Whistler, James Abbott	1871	Realism	Edit
□		*Mademoiselle Caroline Riviere*	Ingres, Jean-Auguste	1806	Neo-Classicism	Edit

October 2014

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
« Sep					Nov »	

Tables Basics

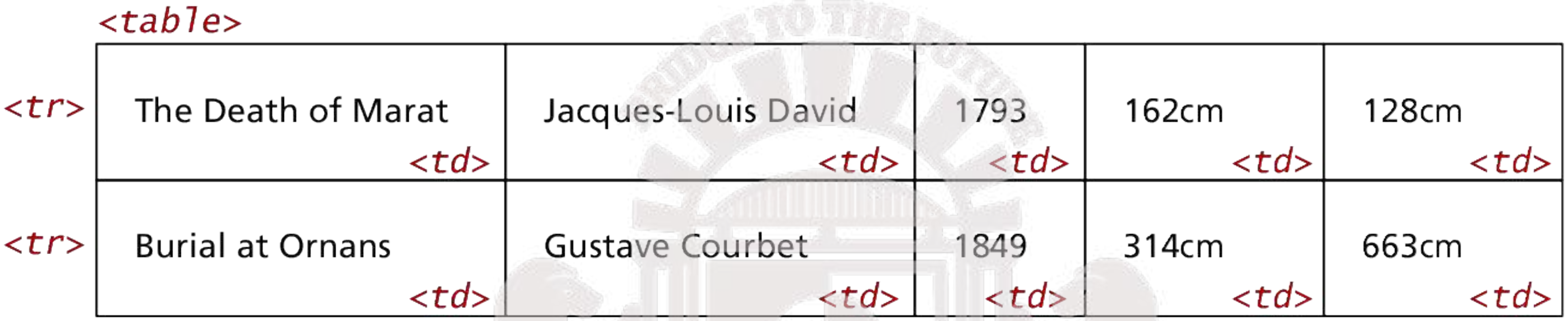
Rows and cells

- an HTML **<table>** contains any number of rows (**<tr>**)
- each row contains any number of table data cells (**<td>**)
- Content goes inside of **<td></td>** tags

```
<table>
    <tr>
        <td>The Death of Marat</td>
    </tr>
</table>
```

content

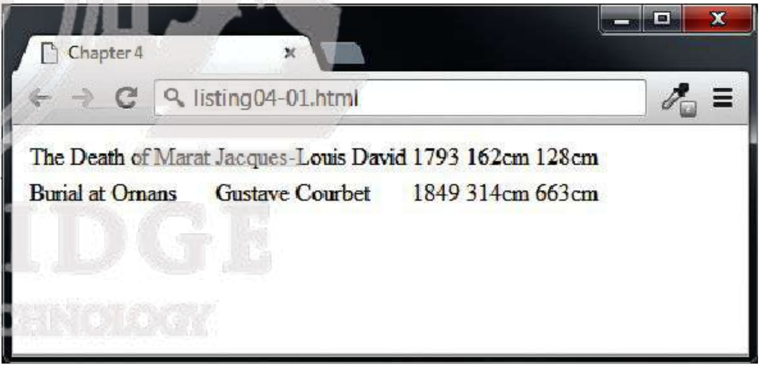
A basic Example



<table>

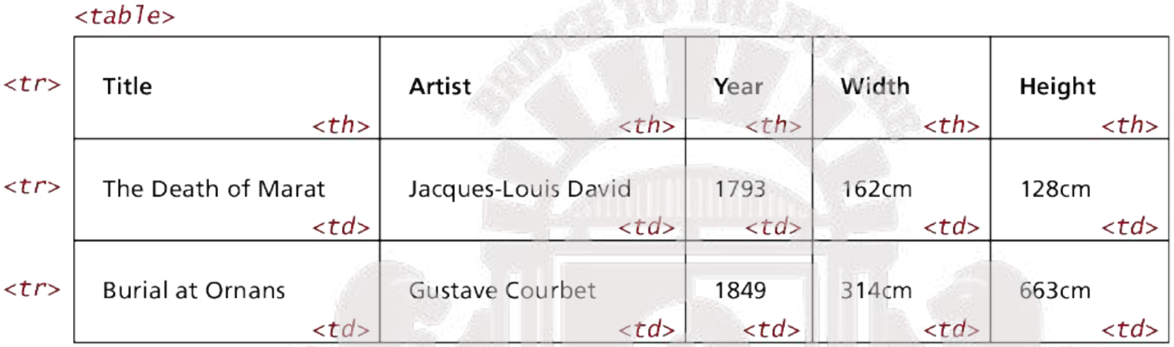
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```



```
The Death of Marat Jacques-Louis David 1793 162cm 128cm
Burial at Ornans    Gustave Courbet    1849 314cm 663cm
```

With Table Headings

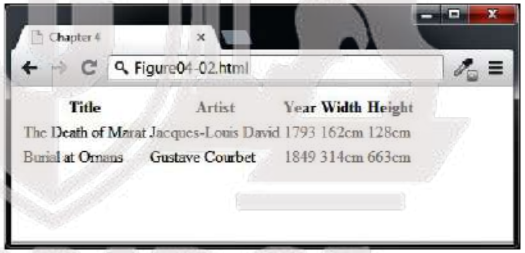


`<table>`

`<tr>`	Title `<th>`	Artist `<th>`	Year `<th>`	Width `<th>`	Height `<th>`
`<tr>`	The Death of Marat `<td>`	Jacques-Louis David `<td>`	1793 `<td>`	162cm `<td>`	128cm `<td>`
`<tr>`	Burial at Ornans `<td>`	Gustave Courbet `<td>`	1849 `<td>`	314cm `<td>`	663cm `<td>`

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th>Width</th>
      <th>Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```

th



Title	Artist	Year	Width	Height
The Death of Marat	Jacques-Louis David	1793	162cm	128cm
Burial at Ornans	Gustave Courbet	1849	314cm	663cm

Why Table Headings

A table heading <th>

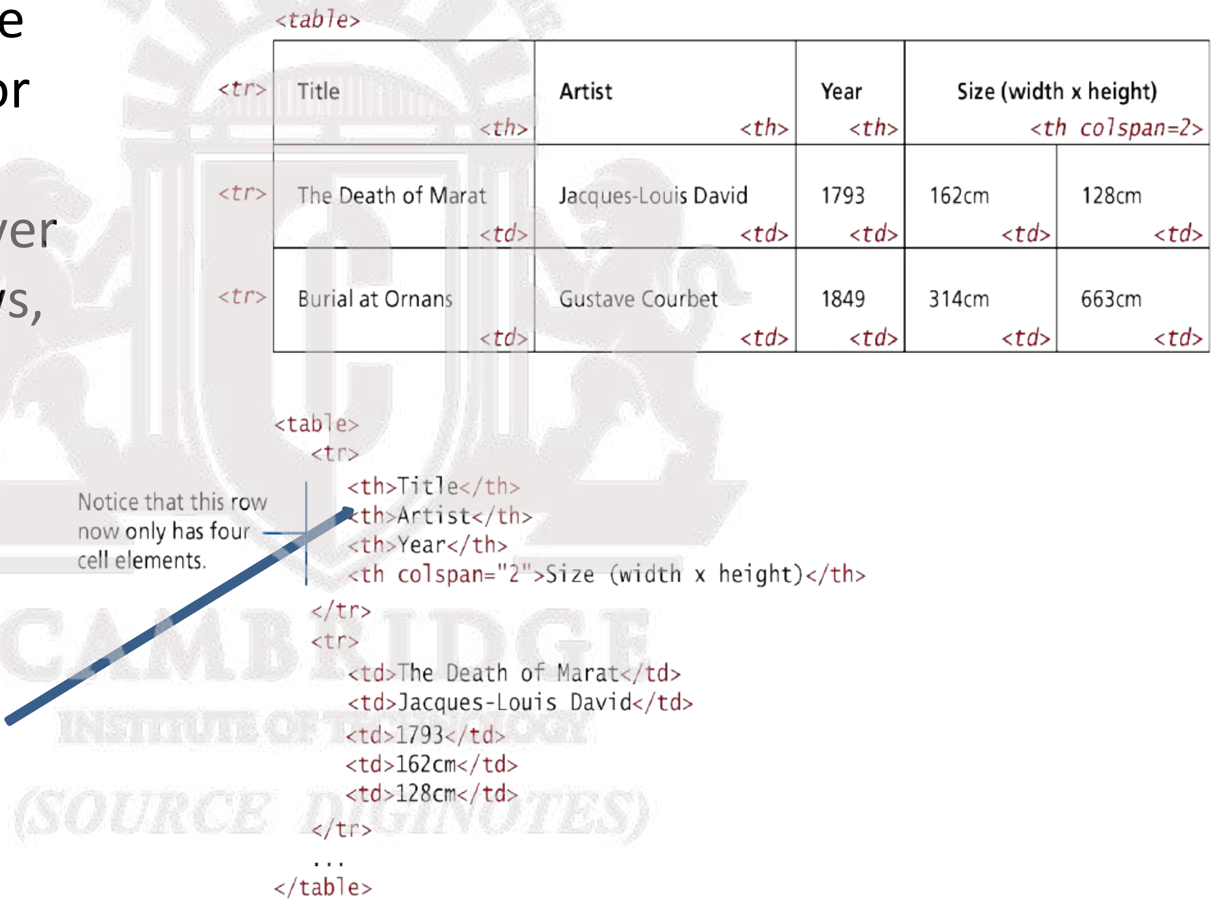
- Browsers tend to make the content within a <th> element bold
- <th> element for accessibility (it helps those using screen readers)
- Provides some semantic info about the row being a row of headers

Spanning Rows and Columns

Span Span Span a Row

Each row must have the same number of <td> or <th> containers. If you want a given cell to cover several columns or rows,

use the **colspan or rowspan attributes**



<table>

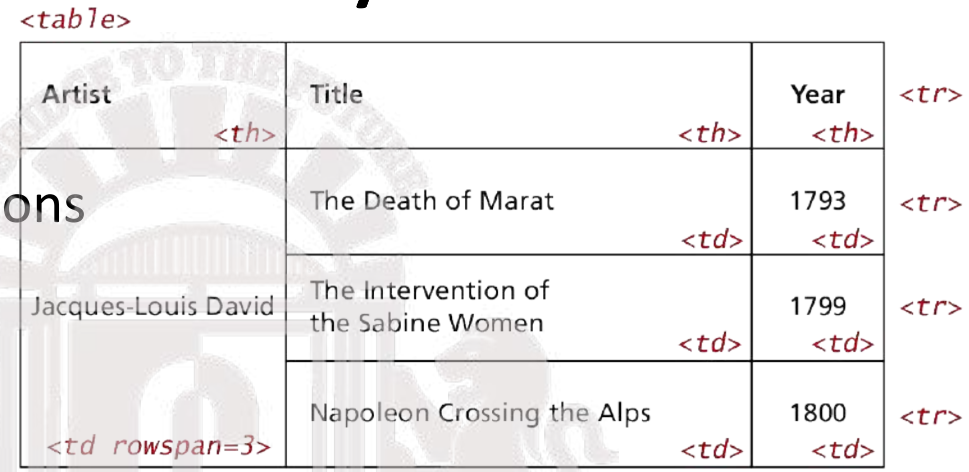
<tr>	Title <th>	Artist <th>	Year <th>	Size (width x height) <th colspan=2>	
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th colspan="2">Size (width x height)</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   ...
</table>
```

Notice that this row now only has four cell elements.

Using Tables for Layout

- Popular in 1990s
  - It works in many situations
- Results in table bloat
- Not semantic
- Larger HTML pages
- Browser quirks



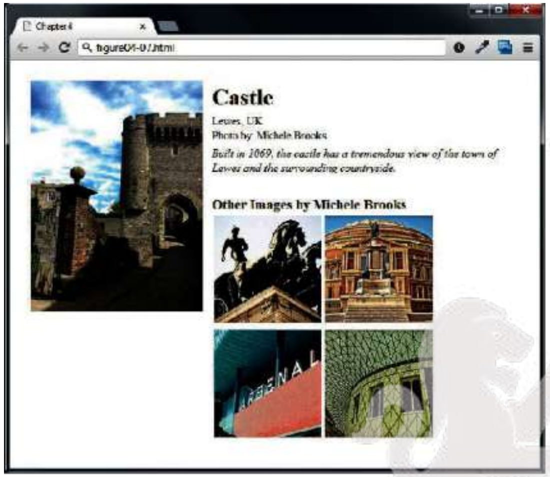
<table>

Artist <th>	Title <th>	Year <th>	<tr>
Jacques-Louis David <td rowspan=3>	The Death of Marat <td>	1793 <td>	<tr>
	The Intervention of the Sabine Women <td>	1799 <td>	<tr>
	Napoleon Crossing the Alps <td>	1800 <td>	<tr>

```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

Notice that these two rows now only have two cell elements.

Example Table layouts



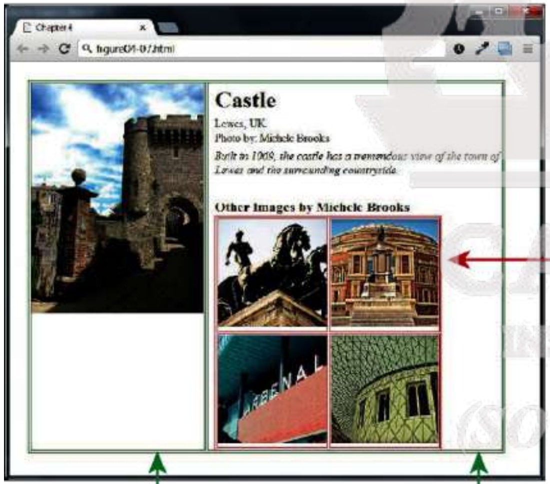
Castle

Lewes, UK

Photo by: Michele Brooks

Built in 1069, the castle has a tremendous view of the town of Lewes and the surrounding countryside.

Other Images by Michele Brooks



Castle

Lewes, UK

Photo by: Michele Brooks

Built in 1069, the castle has a tremendous view of the town of Lewes and the surrounding countryside.

Other Images by Michele Brooks

```
<table>
  <tr>
    <td>
      <img src="images/959.jpg" alt="Castle"/>
    </td>
    <td>
      <h2>Castle</h2>
      <p>Lewes, UK</p>
      <p>Photo by: Michele Brooks</p>
      <p>Built in 1069, the castle has a tremendous
         view of the town of Lewes and the
         surrounding countryside.
      </p>


      <h3>Other Images by Michele Brooks</h3>

      <table>
        <tr>
           <td><img src="images/464.jpg" /></td>
           <td><img src="images/537.jpg" /></td>
        </tr>
        <tr>
           <td><img src="images/700.jpg" /></td>
           <td><img src="images/828.jpg" /></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Additional table tags

- <caption>
- <col>,<colgroup>
- <thead>
- <tfoot>
- <tbody>

A title for the table is good for accessibility.

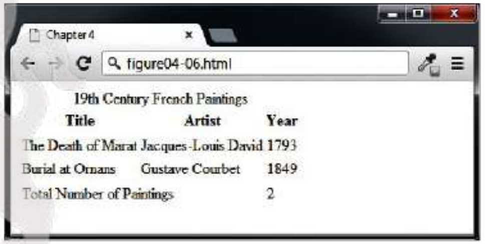
These describe our columns, and can be used to aid in styling.

Table header could potentially also include other <tr> elements.

Yes, the table footer comes *before* the body.

Potentially, with styling the browser can scroll this information, while keeping the header and footer fixed in place.

```
<table>
   <caption>19th Century French Paintings</caption>
   <col class="artistName" />
   <colgroup id="paintingColumns">
       <col />
       <col />
   </colgroup>

   <thead>
       <tr>
           <th>Title</th>
           <th>Artist</th>
           <th>Year</th>
       </tr>
   </thead>

   <tfoot>
       <tr>
           <td colspan="2">Total Number of Paintings</td>
           <td>2</td>
       </tr>
   </tfoot>

   <tbody>
       <tr>
           <td>The Death of Marat</td>
           <td>Jacques-Louis David</td>
           <td>1793</td>
       </tr>
       <tr>
           <td>Burial at Ornans</td>
           <td>Gustave Courbet</td>
           <td>1849</td>
       </tr>
   </tbody>

</table>
```



19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
Total Number of Paintings		2

Section 2 of 6

STYLING TABLES

Styling Tables

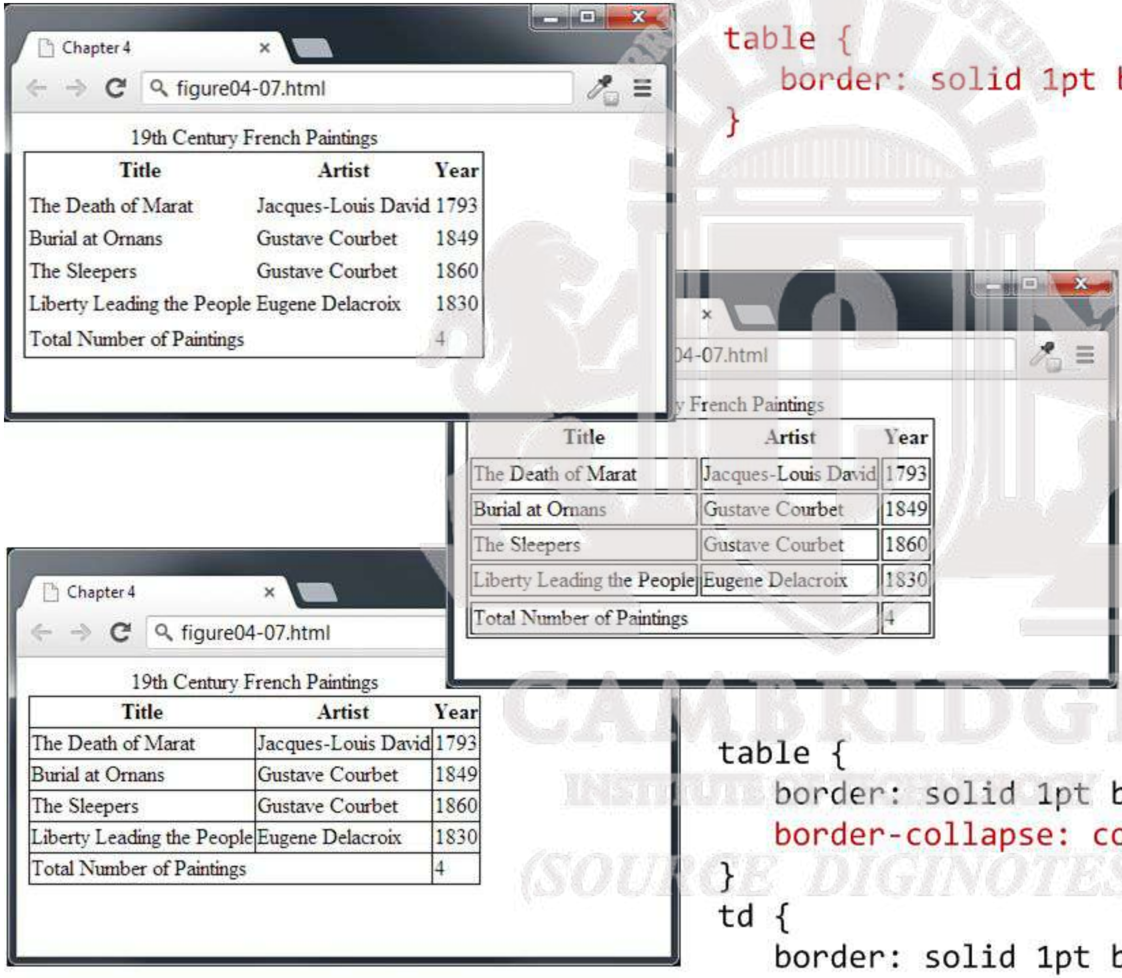
The old way’s deprecated

In HTML5 it is left to CSS, However legacy support for deprecated HTML attributes still exist

- **width, height**—for setting the width and height of cells
- **cellspacing**—for adding space between every cell in the table
- **cellpadding**—for adding space between the content of the cell and its border
- **bgcolor**—for changing the background color of any table element
- **background**—for adding a background image to any table element
- **align**—for indicating the alignment of a table in relation to the surrounding container

Styling Tables

Borders



19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {
    border: solid 1pt black;
}
```

y French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {
    border: solid 1pt black;
}
td {
    border: solid 1pt black;
}
```

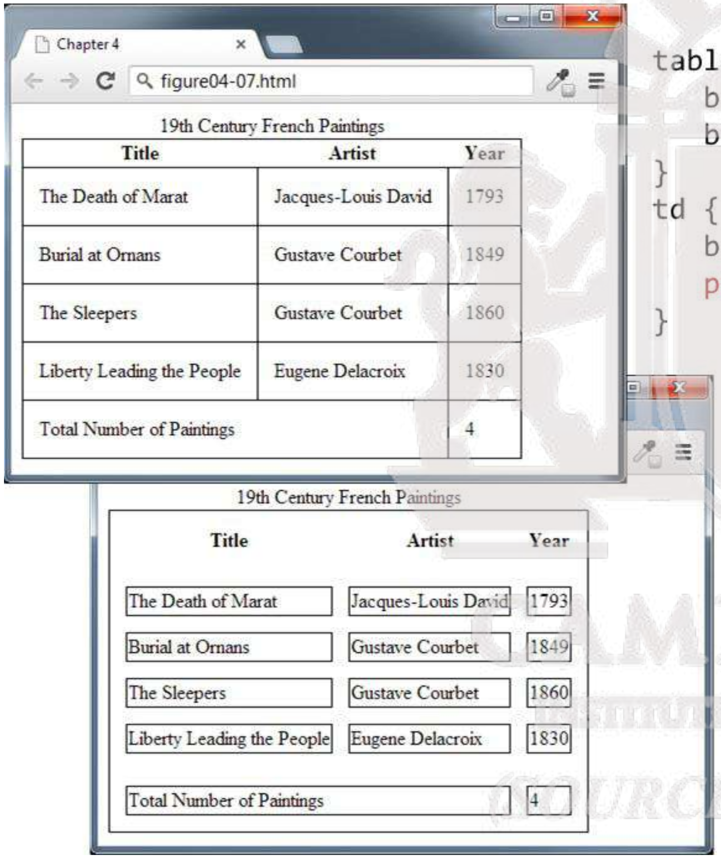
19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {
    border: solid 1pt black;
    border-collapse: collapse;
}
td {
    border: solid 1pt black;
}
```

Styling Tables

Padding and spacing



19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

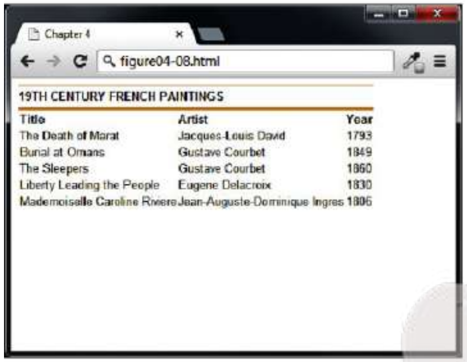
```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
   padding: 10pt;
}
```

19th Century French Paintings

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Total Number of Paintings		4

```
table {
   border: solid 1pt black;
   border-spacing: 10pt;
}
td {
   border: solid 1pt black;
}
```

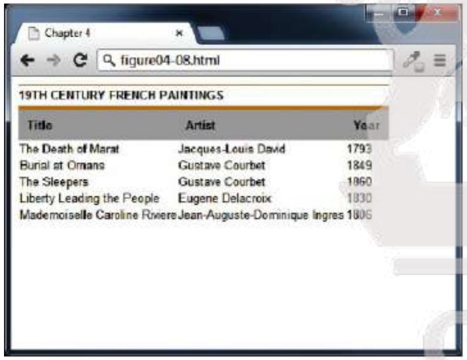
Styling Tables



19TH CENTURY FRENCH PAINTINGS

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

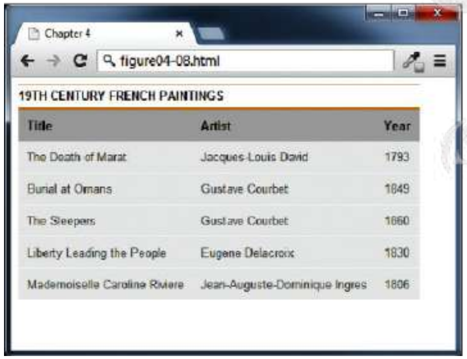
```
table {
   font-size: 0.8em;
   font-family: Arial, Helvetica, sans-serif;
   border-collapse: collapse;
   border-top: 4px solid #DCA806;
   border-bottom: 1px solid white;
   text-align: left;
}
caption {
   font-weight: bold;
   padding: 0.25em 0 0.25em 0;
   text-align: left;
   text-transform: uppercase;
   border-top: 1px solid #DCA806;
}
```



19TH CENTURY FRENCH PAINTINGS

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
thead tr {
   background-color: #CACACA;
}
th {
   padding: 0.75em;
}
```



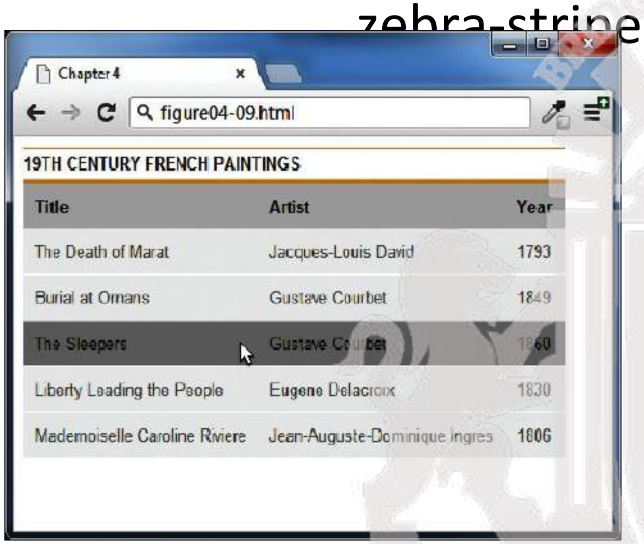
19TH CENTURY FRENCH PAINTINGS

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
tbody tr {
   background-color: #F1F1F1;
   border-bottom: 1px solid white;
   color: #6E6E6E;
}
tbody td {
   padding: 0.75em;
}
```

Nth-Child

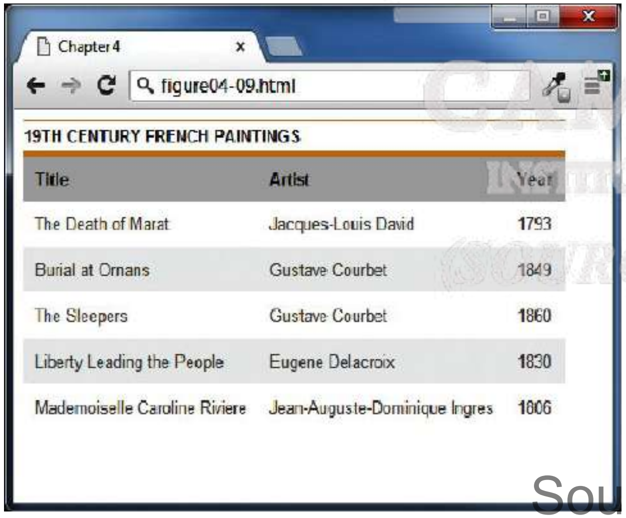
Nifty Table styling tricks: hover effect and zebra-stripes



19TH CENTURY FRENCH PAINTINGS

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
tbody tr:hover {
    background-color: #9e9e9e;
    color: black;
}
```



19TH CENTURY FRENCH PAINTINGS

Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
The Sleepers	Gustave Courbet	1860
Liberty Leading the People	Eugene Delacroix	1830
Mademoiselle Caroline Riviere	Jean-Auguste-Dominique Ingres	1806

```
tbody tr:nth-child(odd) {
    background-color: white;
}
```

Section 3 of 6

INTRODUCING FORMS

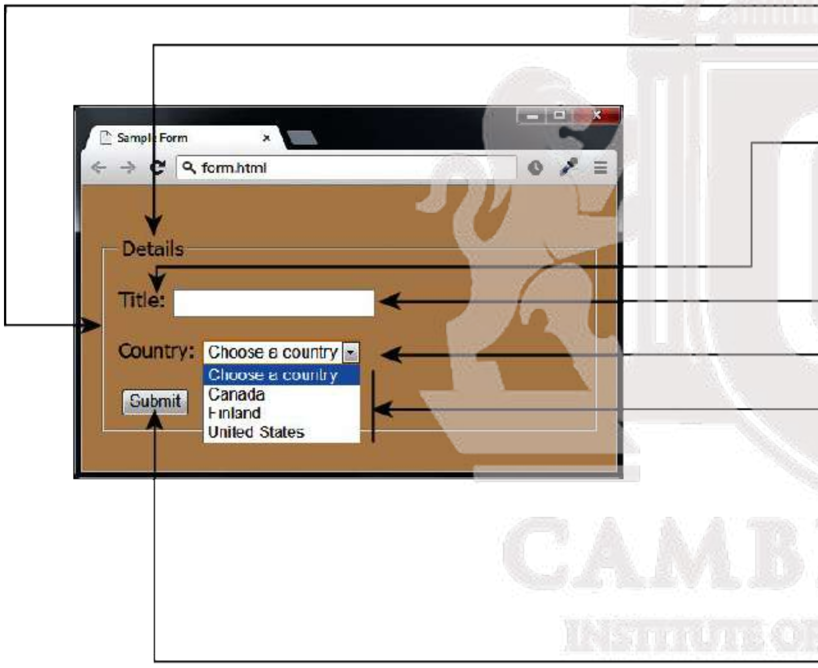
HTML Forms

Richer way to interact with server

Forms provide the user with an alternative way to interact with a web server.

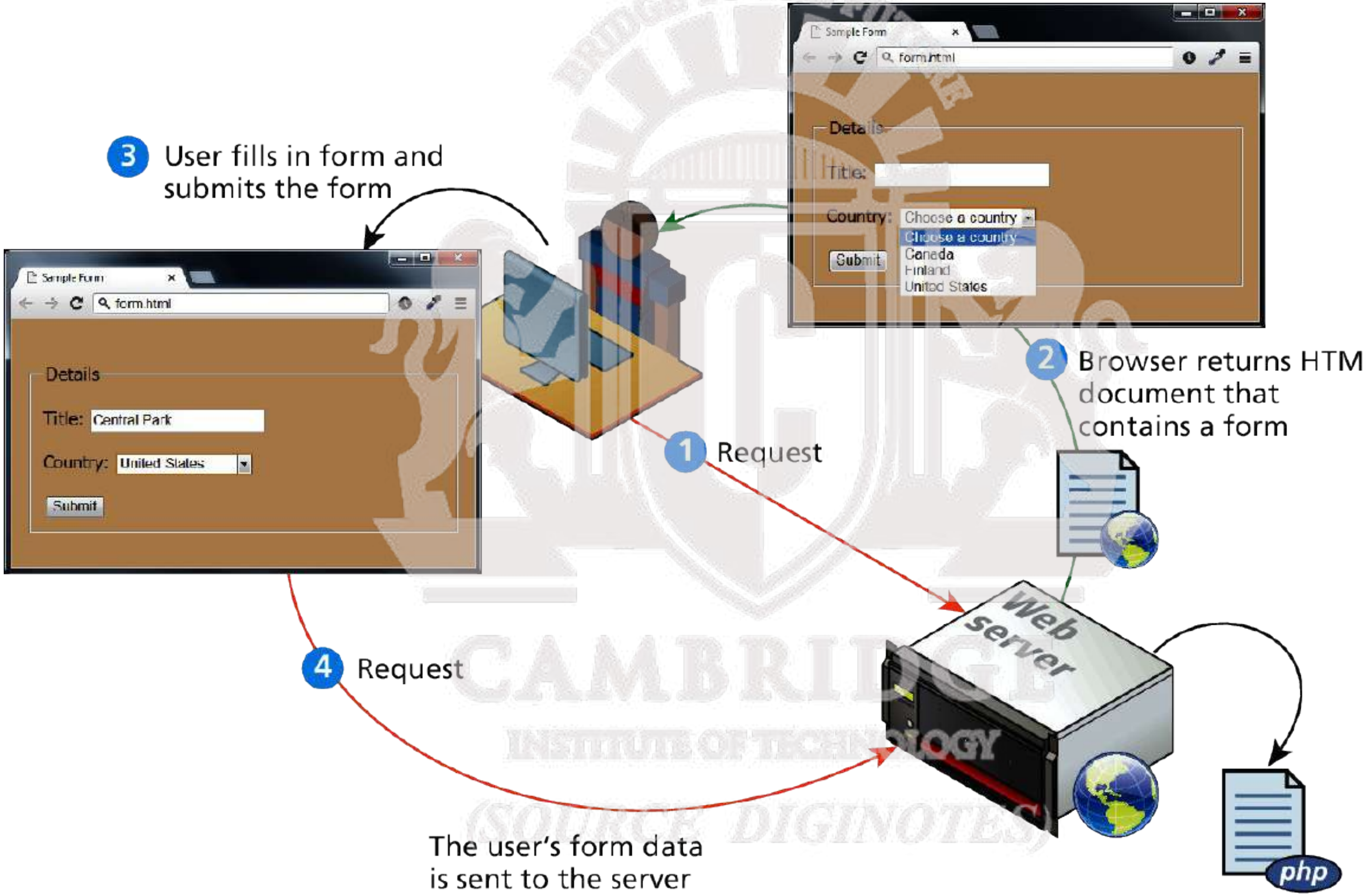
- Forms provide rich mechanisms like:
  - Text input
  - Password input
  - Options Lists
  - Radio and check boxes

Form Structure



```
<form method="get" action="process.php">
  <fieldset>
    <legend>Details</legend>
    <p>
      <label>Title: </label>
      <input type="text" name="title" />
    </p>
    <p>
      <label>Country: </label>
      <select name="where">
        <option>Choose a country</option>
        <option>Canada</option>
        <option>Finland</option>
        <option>United States</option>
      </select>
    </p>
    <input type="submit" />
  </fieldset>
</form>
```

How forms interact with servers



Title: Central Park

Country: United States

1 Request

2 Browser returns HTML document that contains a form

3 User fills in form and submits the form

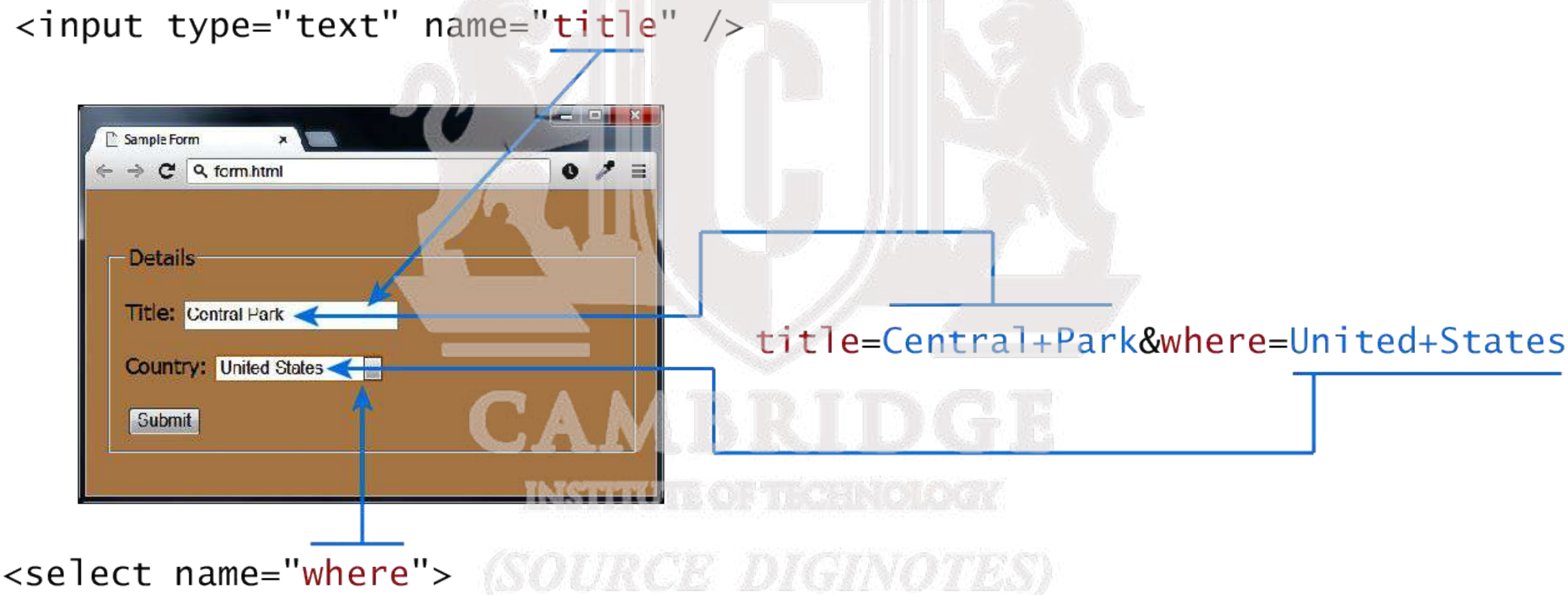
4 Request

The user's form data is sent to the server within the request.

5 This request is usually for some type of server-side script that will process the form data.

Query Strings

At the end of the day, another string



```
<input type="text" name="title" />
```

Sample Form

form.html

Details

Title: Central Park

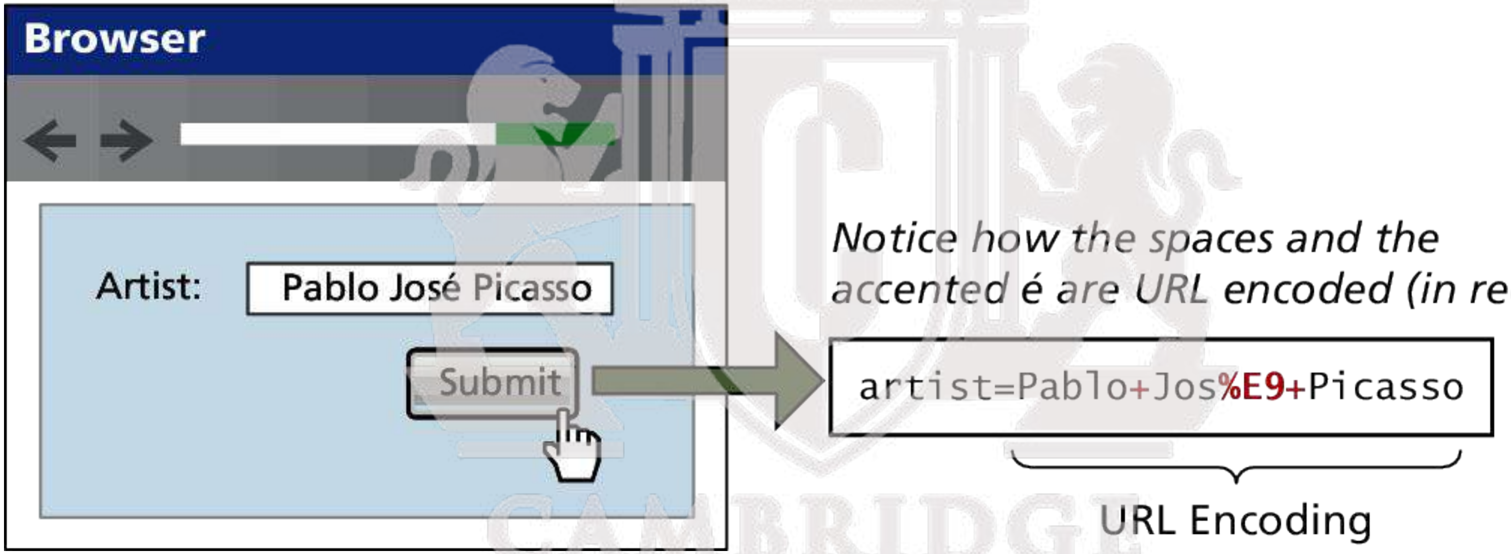
Country: United States

```
title=Central+Park&where=United+States
```

```
<select name="where">
```

URL encoding

Special symbols



Notice how the spaces and the accented é are URL encoded (in red).

```
artist=Pablo+Jos%E9+Picasso
```

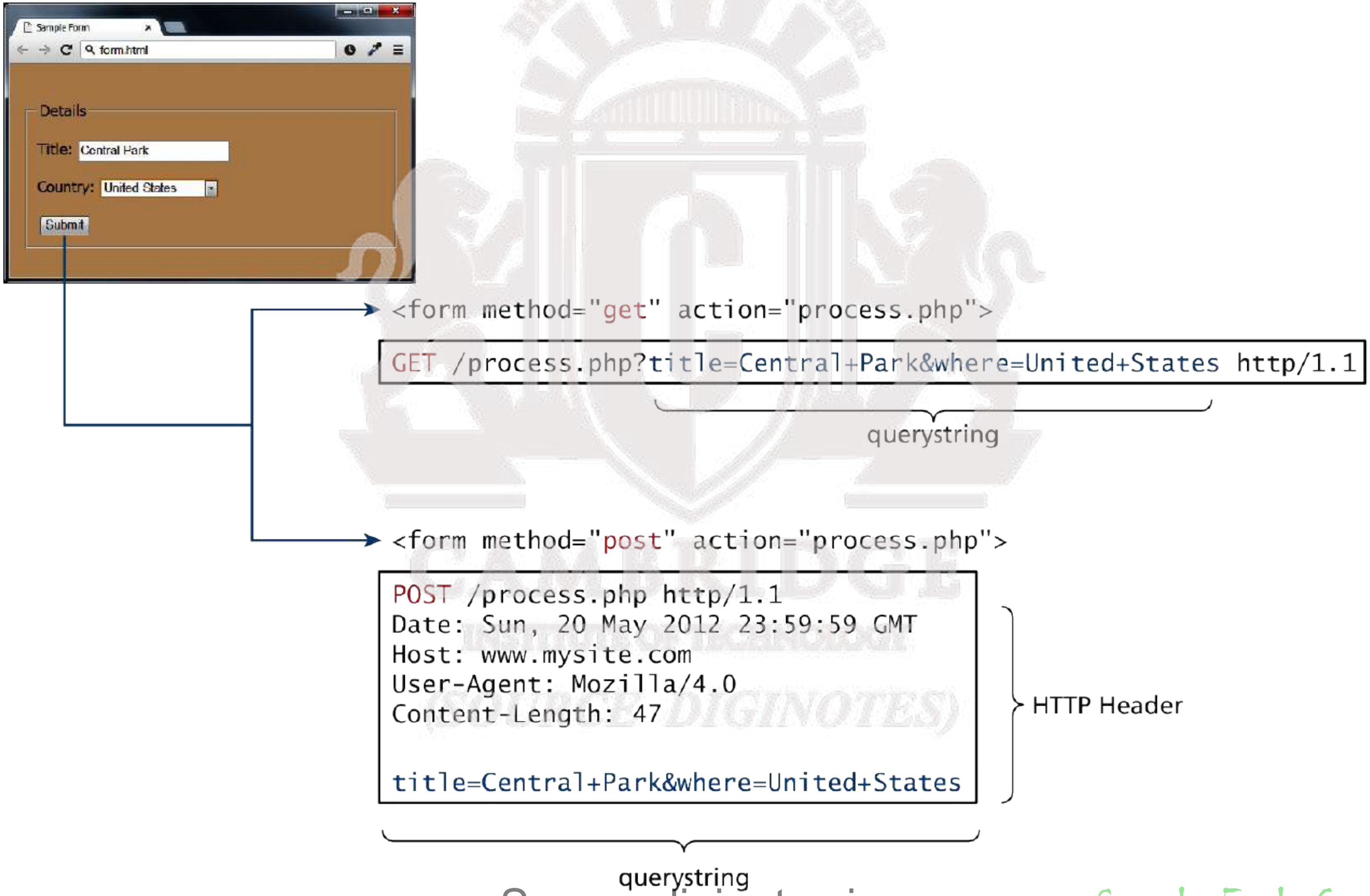
URL Encoding

<form> element

Two essential features of any form, namely the **action** and the **method** attributes.

- The **action** attribute specifies the URL of the server-side resource that will process the form data
- The **method** attribute specifies how the query string data will be transmitted from the browser to the server.
  - GET
  - POST

GET vs POST



Details

Title: Central Park

Country: United States

```
<form method="get" action="process.php">
```

```
GET /process.php?title=Central+Park&where=United+States http/1.1
```

```
<form method="post" action="process.php">
```

```
POST /process.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47

title=Central+Park&where=United+States
```

GET vs POST

Advantages and Disadvantages

- Data can be clearly seen in the address bar.
- Data remains in browser history and cache.
- Data can be bookmarked
- Limit on the number of characters in the form data returned.

POST

- Data can contain binary data.
- Data is hidden from user.
- Submitted data is not stored in cache, history, or bookmarks.

Section 4 of 6

FORMS CONTROL ELEMENTS

Form-Related HTML Elements

Type	Description
<button>	Defines a clickable button.
<datalist>	An HTML5 element form defines lists to be used with other form elements.
<fieldset>	Groups related elements in a form together.
<form>	Defines the form container.
<input>	Defines an input field. HTML5 defines over 20 different types of input.
<label>	Defines a label for a form input element.
<legend>	Defines the label for a fieldset group.
<option>	Defines an option in a multi-item list.
<optgroup>	Defines a group of related options in a multi-item list.
<select>	Defines a multi-item list.
<textarea>	Defines a multiline text entry box.

Source diginotes.in

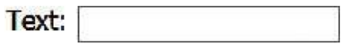
Text Input Controls

Type	Description
text	Creates a single line text entry box. **<input type="text" name="title" />**
textarea	Creates a multiline text entry box. **<textarea rows="3" ... />**
password	Creates a single line text entry box for a password **<input type="password" ... />**
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. **<input type="search" ... />**
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. **<input type="email" ... />**
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. **<input type="tel" ... />**
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. **<input type="url" ... />**

Text Input Controls

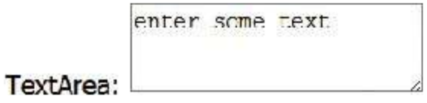
Classic

```
<input type="text" ... />
```



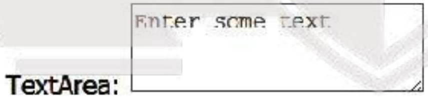
Text:

```
<textarea>
  enter some text
</textarea>
```



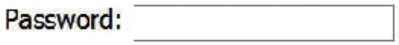
TextArea: enter some text

```
<textarea placeholder="enter some text">
</textarea>
```

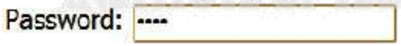


TextArea: Enter some text

```
<input type="password" ... />
```



Password:

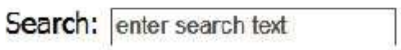


Password: ----

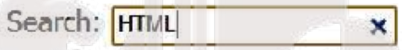
Text Input Controls

HTML5

```
<input type="search" placeholder="enter search text" ... />
```

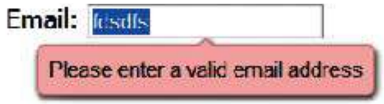


Search: enter search text



Search: HTML ×

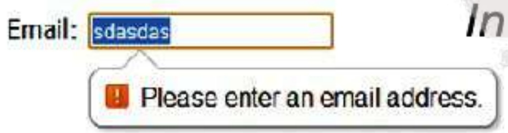
```
<input type="email" ... />
```



Email: ldsdfs

Please enter a valid email address

In Opera

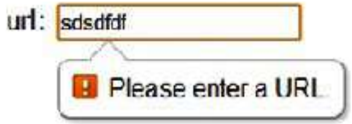


Email: sdasdas

Please enter an email address.

In Chrome

```
<input type="url" ... />
```



url: sdsdfdf

Please enter a URL.

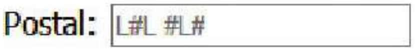
```
<input type="tel" ... />
```

Tel:

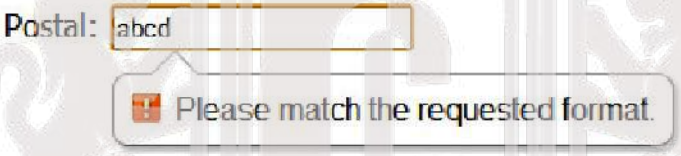
HTML5 advanced controls

Pattern attribute

```
<input type="text" ... placeholder="L#L #L#" pattern="[a-z][0-9][a-z] [0-9][a-z][0-9]"  />
```



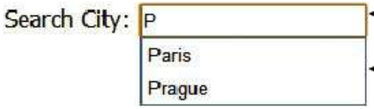
Postal: L#L #L#



Postal: abcd

Please match the requested format.

datalist



Search City: P

```
<input type="text" name="city" list="cities"  />

<datalist id="cities">
   <option>Calcutta</option>
   <option>Calgary</option>
   <option>London</option>
   <option>Los Angeles</option>
   <option>Paris</option>
   <option>Prague</option>
</datalist>
```

Source diginotes.in

Save the Earth. Go paperless

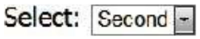
Select Lists

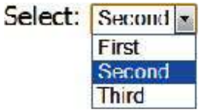
Chose an option, any option.

- **<select>** element is used to create a multiline box for selecting one or more items
  - The options are defined using the **<option>** element
  - can be hidden in a dropdown or multiple rows of the list can be visible
  - Option items can be grouped together via the **<optgroup>** element.

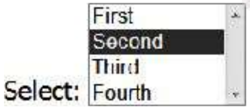
Select Lists

Select List Examples

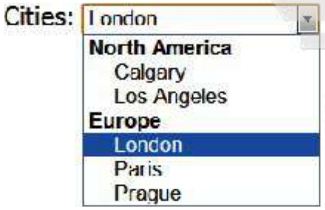




```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```



```
<select size="3" ... >
```

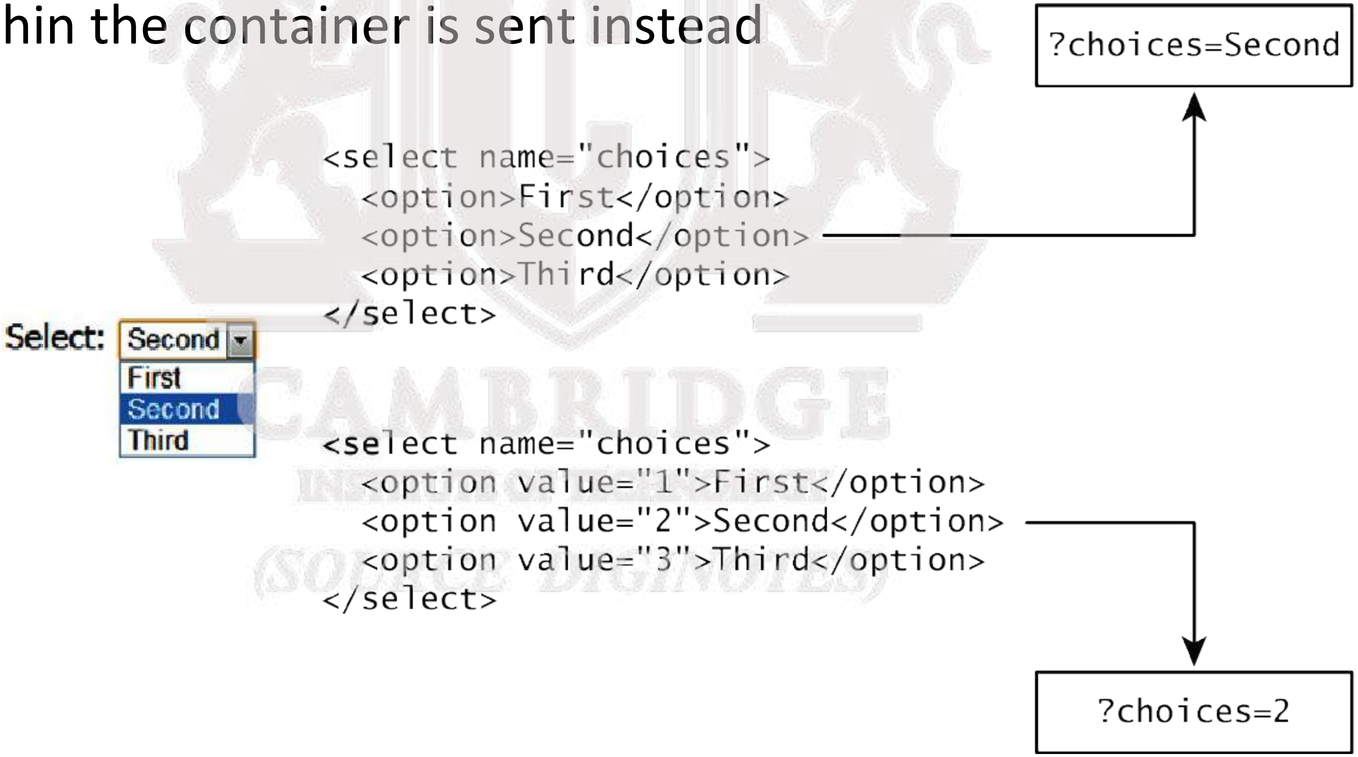


```
<select … >
 <optgroup label="North America">
   <option>Calgary</option>
   <option>Los Angeles</option>
 </optgroup>
 <optgroup label="Europe">
   <option>London</option>
   <option>Paris</option>
   <option>Prague</option>
 </optgroup>
</select>
```

Which Value to send

Select Lists Cont.

- The **value** attribute of the <option> element is used to specify what value will be sent back to the server.
- The value attribute is optional; if it is not specified, then the text within the container is sent instead



```
<select name="choices">
   <option>First</option>
   <option>Second</option>
   <option>Third</option>
</select>
```

?choices=Second

```
<select name="choices">
   <option value="1">First</option>
   <option value="2">Second</option>
   <option value="3">Third</option>
</select>
```

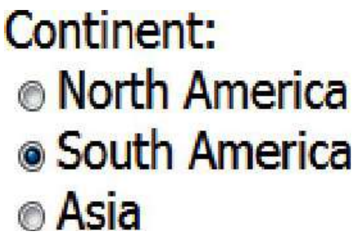
?choices=2

Radio Buttons

Radio buttons are useful when you want the user to select a single item from a small list of choices and you want all the choices to be visible

- radio buttons are added via the **<input type="radio">** element
- The buttons are mutually exclusive (i.e., only one can be chosen) by sharing the same name attribute
- The checked attribute is used to indicate the default choice
- the value attribute works in the same manner as with the <option> element

Radio Buttons



Continent:

○ North America

● South America

○ Asia

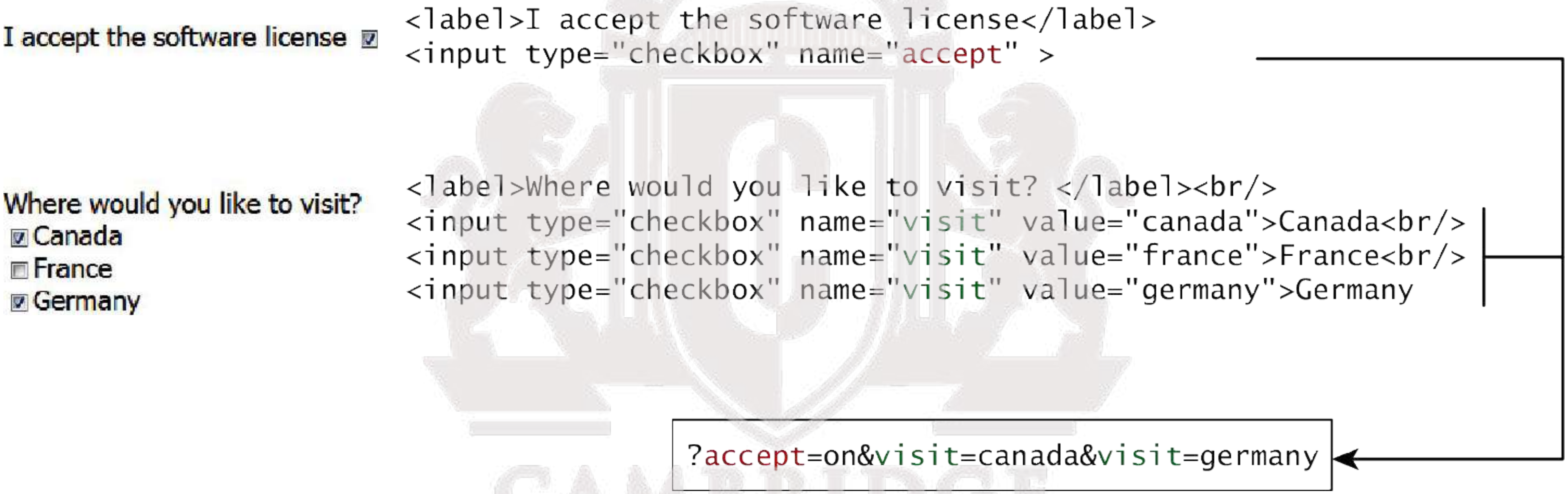
```
<input type="radio" name="where" value="1">North America<br/>
<input type="radio" name="where" value="2" checked>South America<br/>
<input type="radio" name="where" value="3">Asia
```

Checkboxes

Checkboxes are used for getting yes/no or on/off responses from the user.

- checkboxes are added via **the <input type="checkbox”>** Element
- You can also group checkboxes together by having them share the same name attribute
- Each checked checkbox will have its value sent to the server
- Like with radio buttons, the checked attribute can be used to set the default value of a checkbox

Checkboxes



I accept the software license ☑

Where would you like to visit?
☑ Canada
☐ France
☑ Germany

```
<label>I accept the software license</label>
<input type="checkbox" name="accept" >
```

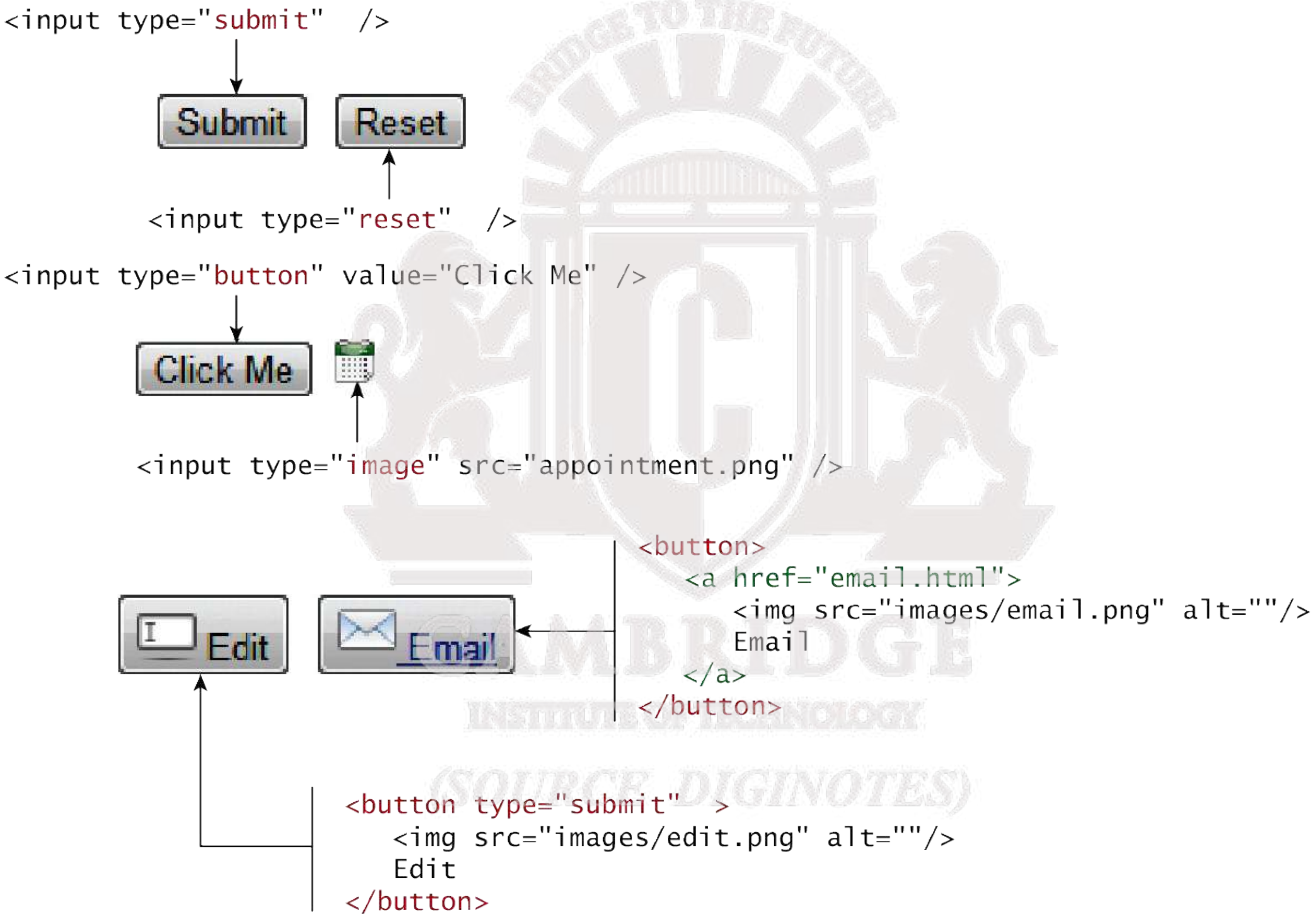
```
<label>Where would you like to visit? </label><br/>
<input type="checkbox" name="visit" value="canada">Canada<br/>
<input type="checkbox" name="visit" value="france">France<br/>
<input type="checkbox" name="visit" value="germany">Germany
```

```
?accept=on&visit=canada&visit=germany
```

Button Controls

Type	Description
<input type="submit">	Creates a button that submits the form data to the server.
<input type="reset">	Creates a button that clears any of the user’s already entered form data.
<input type="button">	Creates a custom button. This button may require Javascript for it to actually perform any action.
<input type="image">	Creates a custom submit button that uses an image for its display.
<button>	Creates a custom button. The <button> element differs from <input type="button"> in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the type="submit" attribute.

Button Controls



```
<input type="submit"  />
```

```
<input type="reset"  />
```

```
<input type="button" value="Click Me" />
```

```
<input type="image" src="appointment.png" />
```

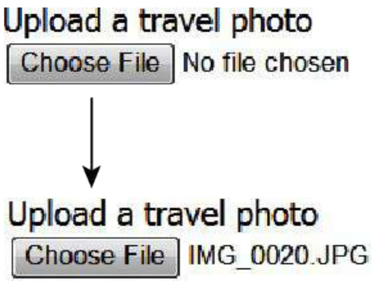
```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

```
<button type="submit" >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

Specialized Controls

I’m so special

- **<input type=hidden>**
- **<input type=file>**



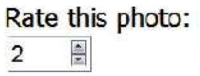
```
<form method="post" enctype="multipart/form-data" ... >
   ...
   <label>Upload a travel photo</label>
   <input type="file" name="photo" />
   ...
</form>
```

Number and Range

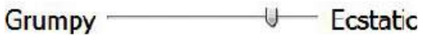
- Typically input values need be **validated**. Although server side validation is required, optional client side pre-validation is good practice.

- The number and range controls Added in HTML5 provide a way to input numeric values that **eliminates the need for JavaScript numeric validation!!!**

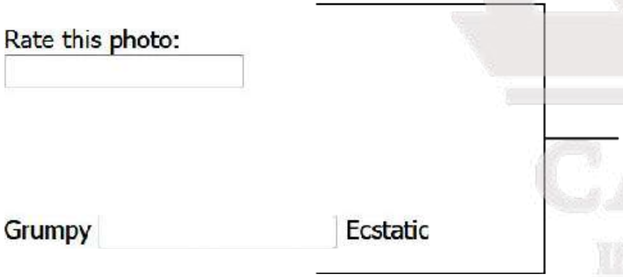
Number and Range



```
<label>Rate this photo: <br/>
<input type="number" min="1" max="5" name="rate" />
```

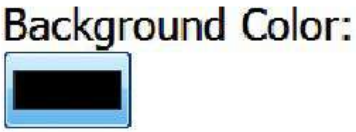


```
Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```

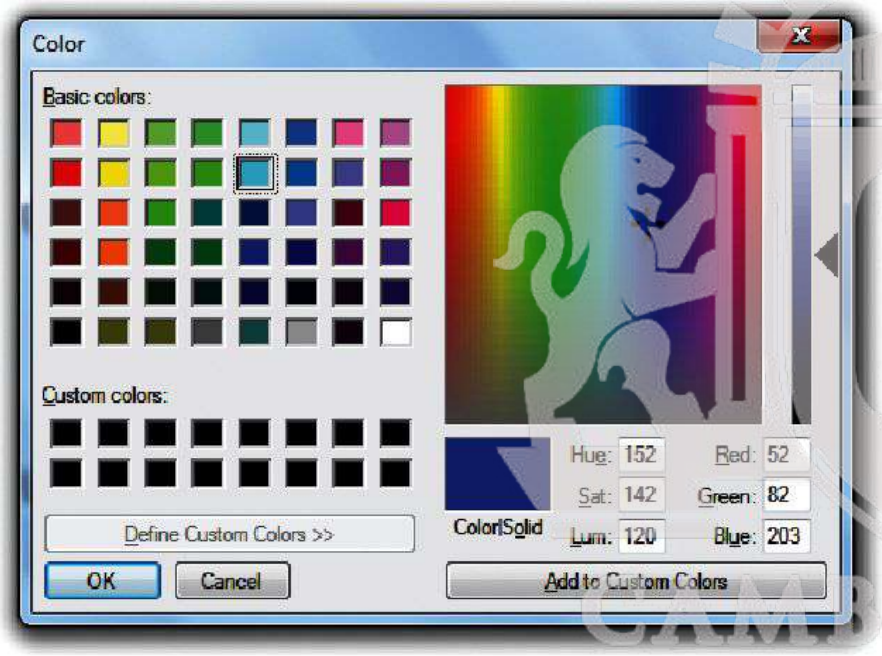


Controls as they appear in browser that doesn't support these input types

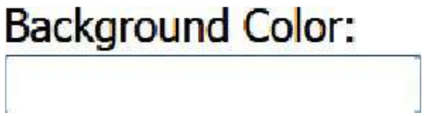
Color


Background Color:

```
<label>Background Color: <br/>
<input type="color" name="back" />
```



Hue: 152
Sat: 142
Lum: 120
Red: 52
Green: 82
Blue: 203


Background Color:

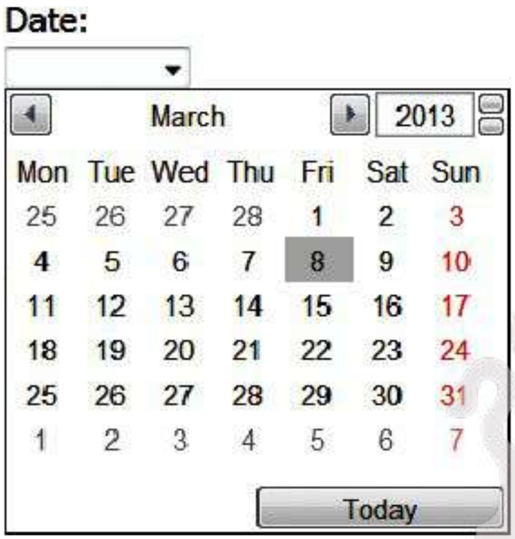
Control as it appears in browser that doesn't support this input type

Date and Time Controls

- Dates and times often need validation when gathering this information from a regular text input control.

- From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on.

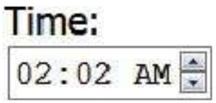
HTML5 Date and Time Controls



Date:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

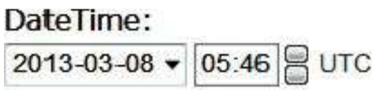
```
<label>Date: <br/>
<input type="date" ... />
```



Time:

02:02 AM

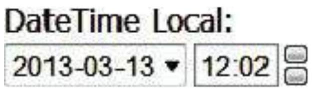
```
<input type="time" ... />
```



DateTime:

2013-03-08 ▾ 05:46 UTC

```
<input type="datetime" ... />
```

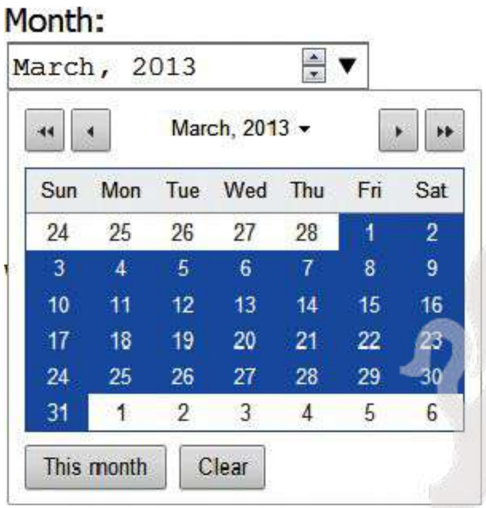


DateTime Local:

2013-03-13 ▾ 12:02

```
<input type="datetime-local" ... />
```

HTML5 Date and Time Controls

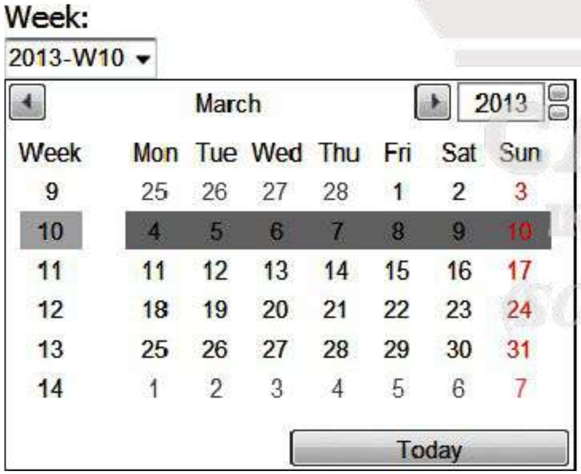


Month: March, 2013

March, 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

```
<input type="month" ... />
```



Week: 2013-W10

March 2013

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
9	25	26	27	28	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

```
<input type="week" ... />
```

Source diginotes.in

HTML Controls

Type	Description
date	Creates a general date input control. The format for the date is "yyyy-mm-dd".
time	Creates a time input control. The format for the time is "HH:MM:SS", for hours:minutes:seconds.
datetime	Creates a control in which the user can enter a date and time.
datetime-local	Creates a control in which the user can enter a date and time without specifying a time zone.
month	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm".
week	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##".

Other Controls

You mean there's more

- The <progress> and <meter> elements can be used to provide feedback to users,
    - but requires JavaScript to function dynamically.
- The <output> element can be used to hold the output from a calculation.
- The <keygen> element can be used to hold a private key for public-key encryption

Section 5 of 6

TABLE AND FORM ACCESSIBILITY

Web Accessibility

- Not all web users are able to view the content on web pages in the same manner.

- The term **web accessibility** refers to the assistive technologies, various features of HTML that work with those technologies, and different coding and design practices that can make a site more usable for people with visual, mobility, auditory, and cognitive disabilities.

- In order to improve the accessibility of websites, the W3C created the **Web Accessibility Initiative (WAI)**

  - Web Content Accessibility Guidelines

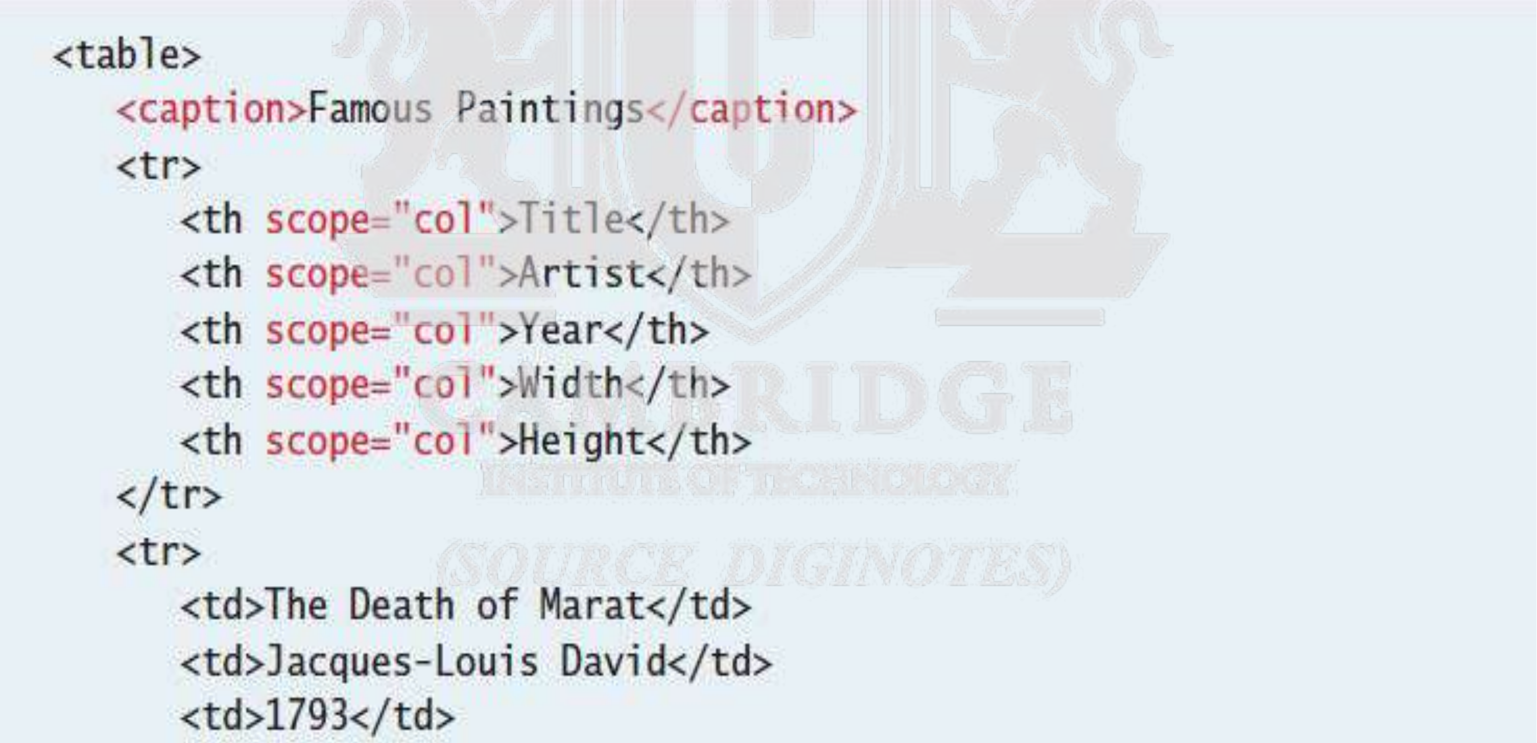
Web Content Accessibility Guidelines

- Provide text alternatives for any nontext content so that it can be changed into other forms people need, such as large print, braille, speech, symbols, or simpler language.
- Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
- Make all functionality available from a keyboard.
- Provide ways to help users navigate, find content, and determine where they are.

Accessible Tables

1. Describe the table’s content using the <caption> element

2. Connect the cells with a textual description in the header



```
<table>
   <caption>Famous Paintings</caption>
   <tr>
      <th scope="col">Title</th>
      <th scope="col">Artist</th>
      <th scope="col">Year</th>
      <th scope="col">Width</th>
      <th scope="col">Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
```

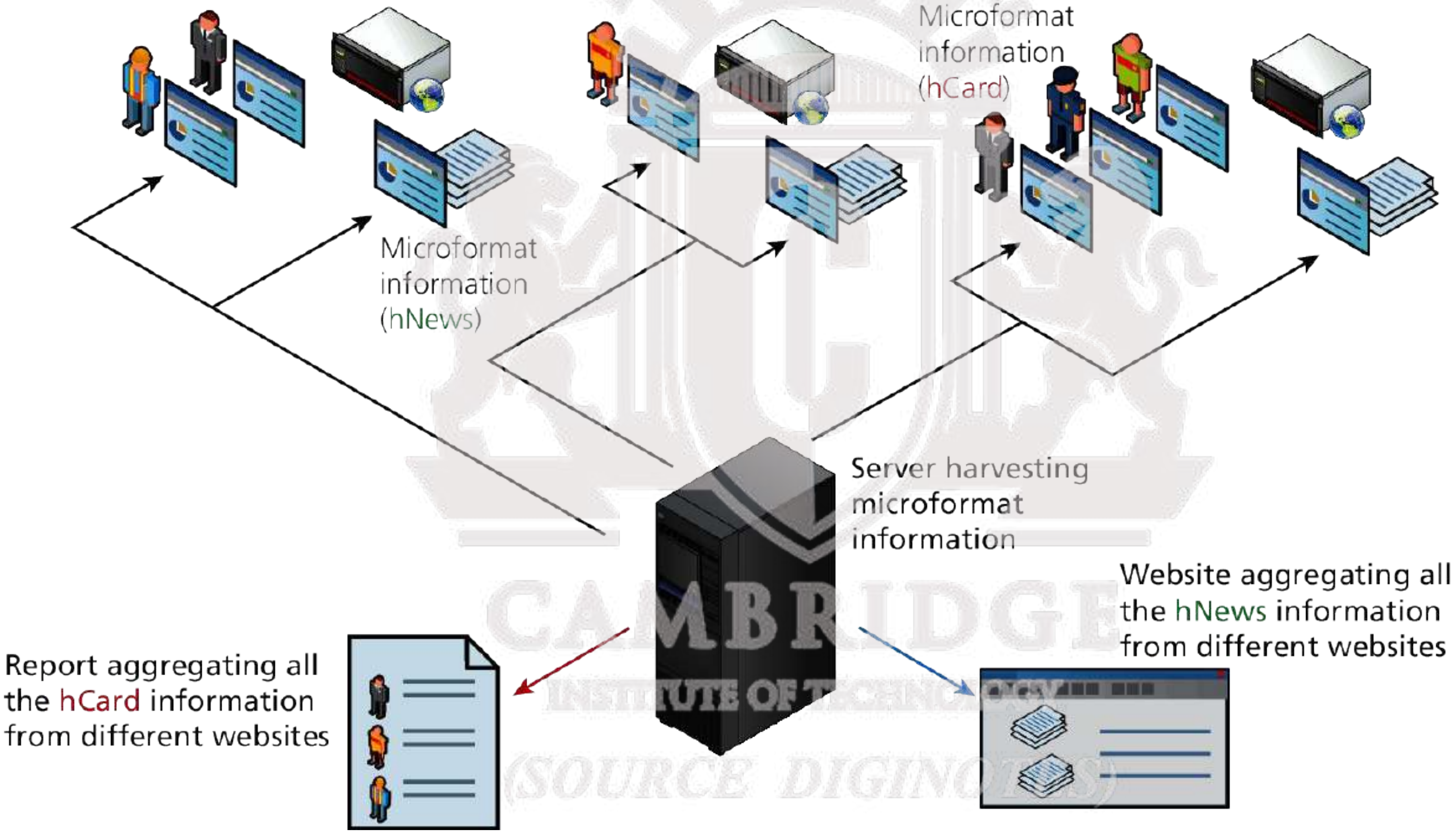
Section 6 of 6

MICROFORMATS

Microformats

- The web has millions of pages in it. Yet despite the incredible variety, there is a surprising amount of similar information from site to site.
- Most sites have some type of Contact us pages.
- Similarly, many sites contain calendar of upcoming events or information about products or news.
- The idea behind microformats is that if this type of common information were tagged in similar ways, then automated tools would be able to gather and transform it.
- A **microformat** is a small pattern of HTML markup and attributes to represent common blocks of information such as people, events, and news stories so that the information in them can be extracted and indexed by software agents.
- One of the most common microformat is hcard, which is used to semantically mark up contact information of a person.

Microformat


Microformat information (hCard)
Microformat information (hNews)
Server harvesting microformat information
Website aggregating all the hNews information from different websites
Report aggregating all the hCard information from different websites

Source diginotes.in

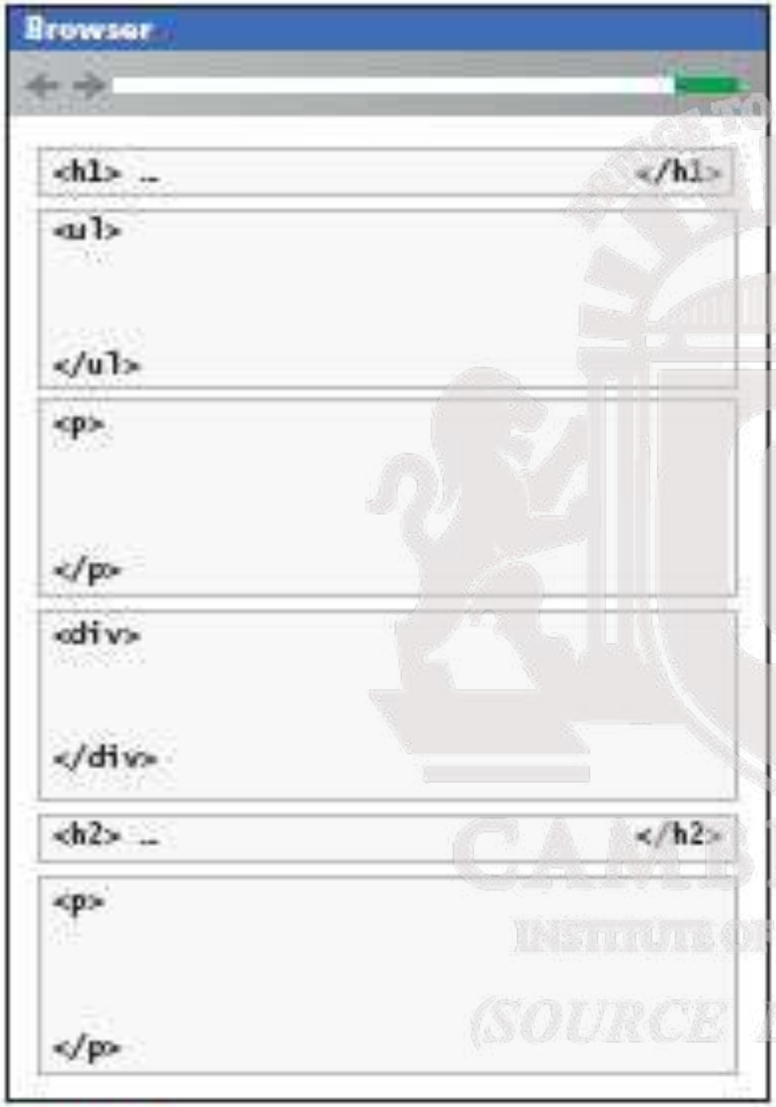
Advanced CSS: Layout

Chapter 5

5.1 Normal Flow

Normal flow, which refers here to how the browser will normally display block level elements and inline elements

- Block Level elements such as p, div, table, ul and table are each contained on their own line.
- They begin with a line break.
- 2 block level elements cant exist on the same line as in fig



Each block exists on its own line and is displayed in normal flow from the browser window's top to its bottom.

By default each block-level element fills up the entire width of its parent (in this case, it is the <body>, which is equivalent to the width of the browser window).

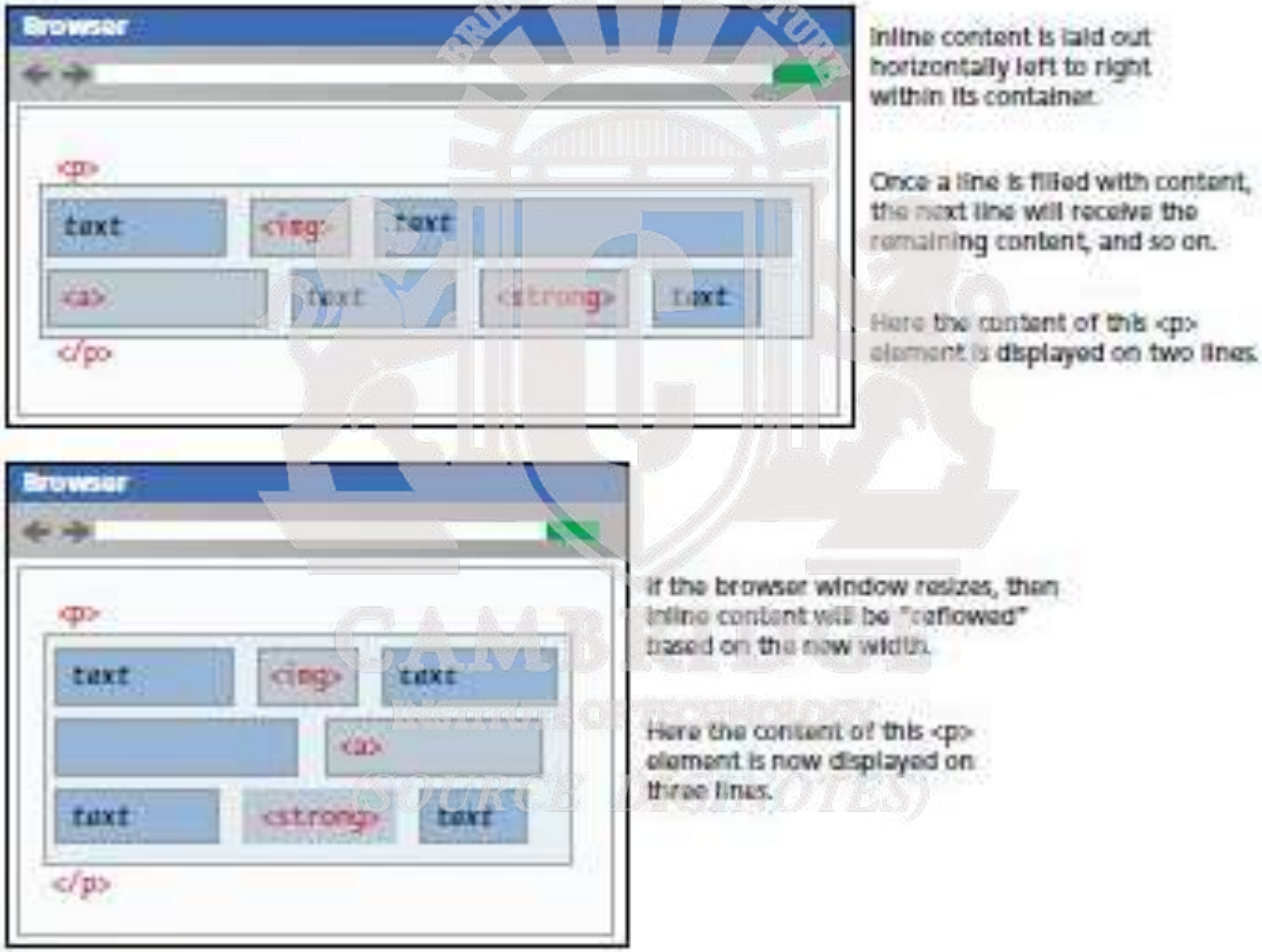
You can use CSS box model properties to customize, for instance, the width of the box and the margin space between other block-level elements.

FIGURE 5.1 Block-level elements

Inline Elements

- Inline elements do not form their own blocks but instead are displayed within lines.
- Normal text in an html document is inline, as are elements such as em, a, tag, and span.
- Inline elements line up next to one another horizontally from left to right on the same line.
- When there is no space left on the line the content moves to new line as in figure

```
<p>
This photo <img src="photo-con.png" alt="" /> of Conservatory Pond in
<a href="http://www.centralpark.com/">Central Park</a> New York City
was taken on October 22, 2015 with a <strong>Canon EOS 30D</strong>
camera.
</p>
```



Inline content is laid out horizontally left to right within its container.

Once a line is filled with content, the next line will receive the remaining content, and so on.

Here the content of this <p> element is displayed on two lines.

If the browser window resizes, then inline content will be "reflowed" based on the new width.

Here the content of this <p> element is now displayed on three lines.

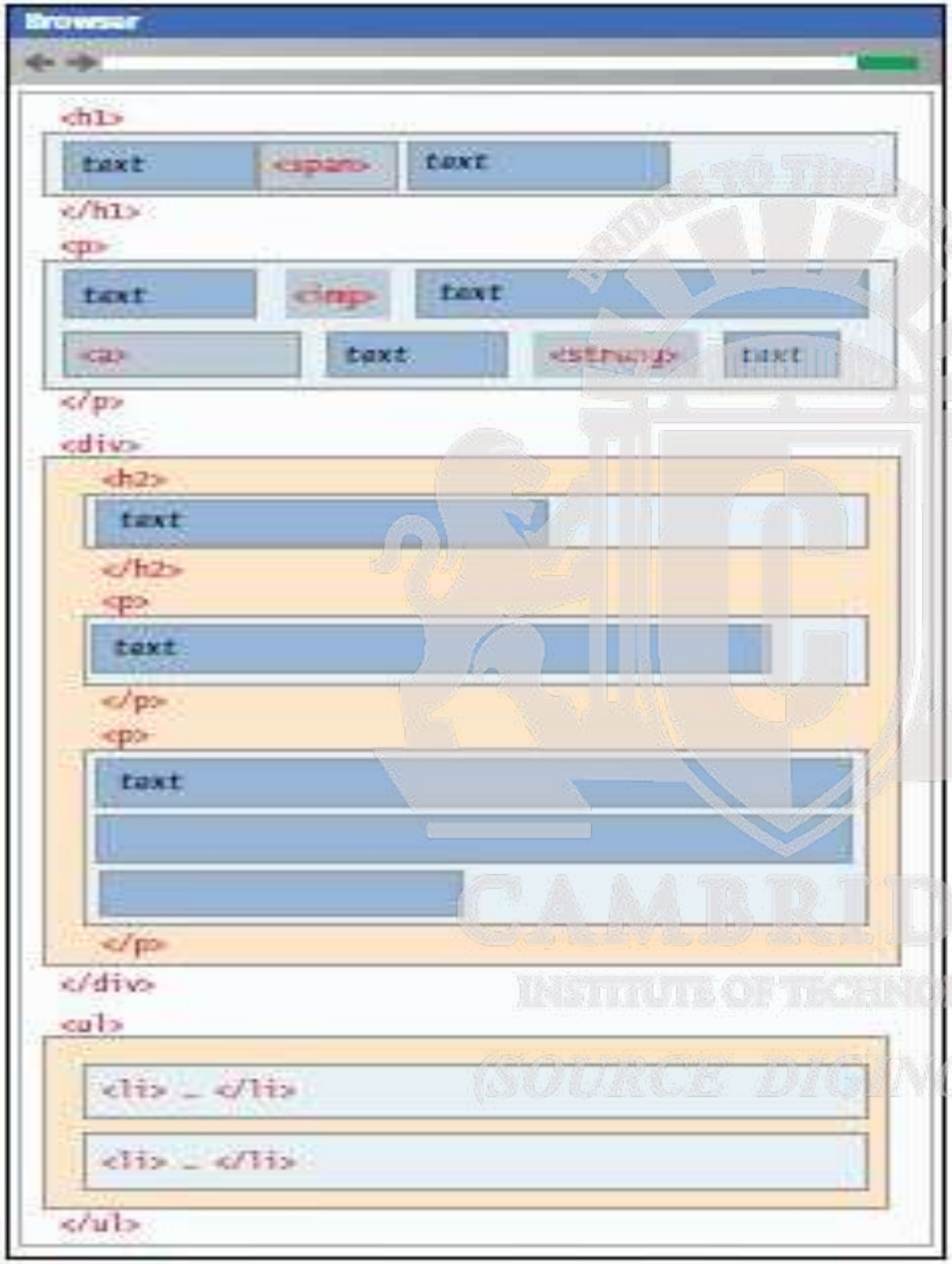
FIGURE 5.2 Inline elements

Inline Elements

- Inline elements are actually two types.
  - Replaced Inline elements.
    - Content and appearance is defined by some external resource,

      Ex – tag and some form elements
  - Nonreplaced Inline elements.
    - Content defined within the document, which includes all other inline elements.

- In a Document with normal flow, block level elements and inline elements work together as in figure.
  - Block level will flow from top to bottom
  - Inline level will flow from left to right



A document consists of block-level elements stacked from top to bottom.

Within a block, inline content is horizontally placed left to right.

Some block-level elements can contain other block-level elements (in this example, a <div> can contain other blocks).

In such a case, the block-level content inside the parent is stacked from top to bottom within the container (<div>).

FIGURE 5.3 Block and inline elements together

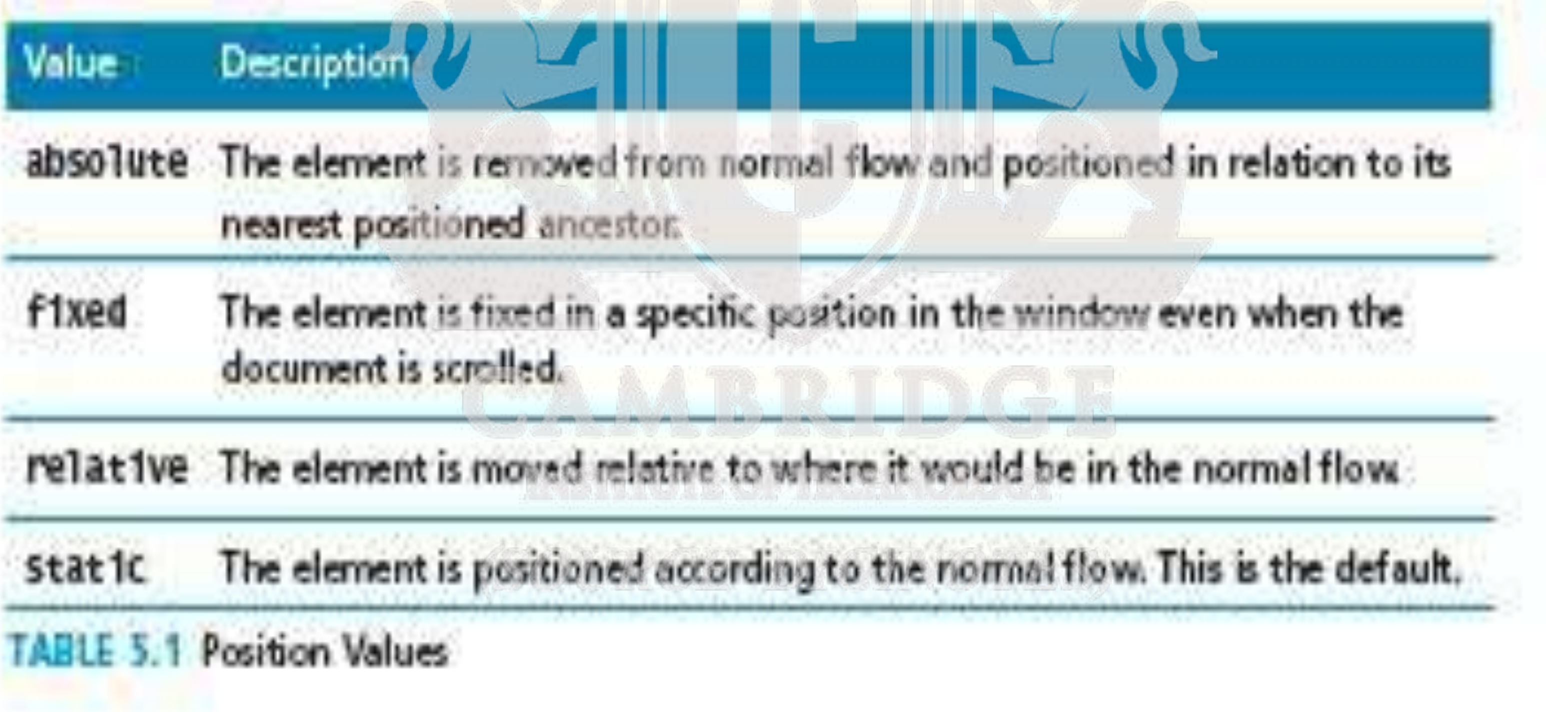
- It is possible to change

- These two rules will make all span elements behave like block level elements and all li elements like inline

```
Span {display : block;}
l1 {display : inline;}
```

5.2 Positioning Elements

- The position property is used to specify the type of positioning and the possible values are

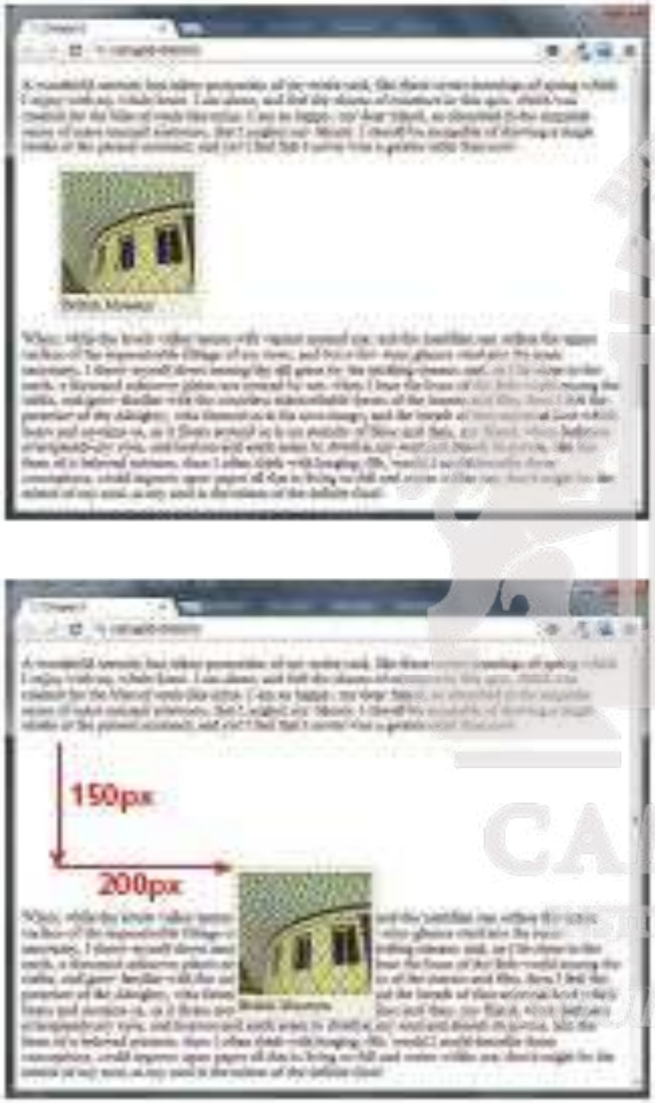


Value	Description
absolute	The element is removed from normal flow and positioned in relation to its nearest positioned ancestor.
fixed	The element is fixed in a specific position in the window even when the document is scrolled.
relative	The element is moved relative to where it would be in the normal flow.
static	The element is positioned according to the normal flow. This is the default.

TABLE 5.1 Position Values

5.2.1 Relative positioning

- Element is displaced out of its normal flow position and moved relative to where it would have been placed.
- When an element is positioned relatively, it is displaced out of its normal flow position and moved relative to where it would have been placed.
- The other content around the relatively positioned element "remembers" the element old position in the flow, thus the space the element would have occupied as in fig



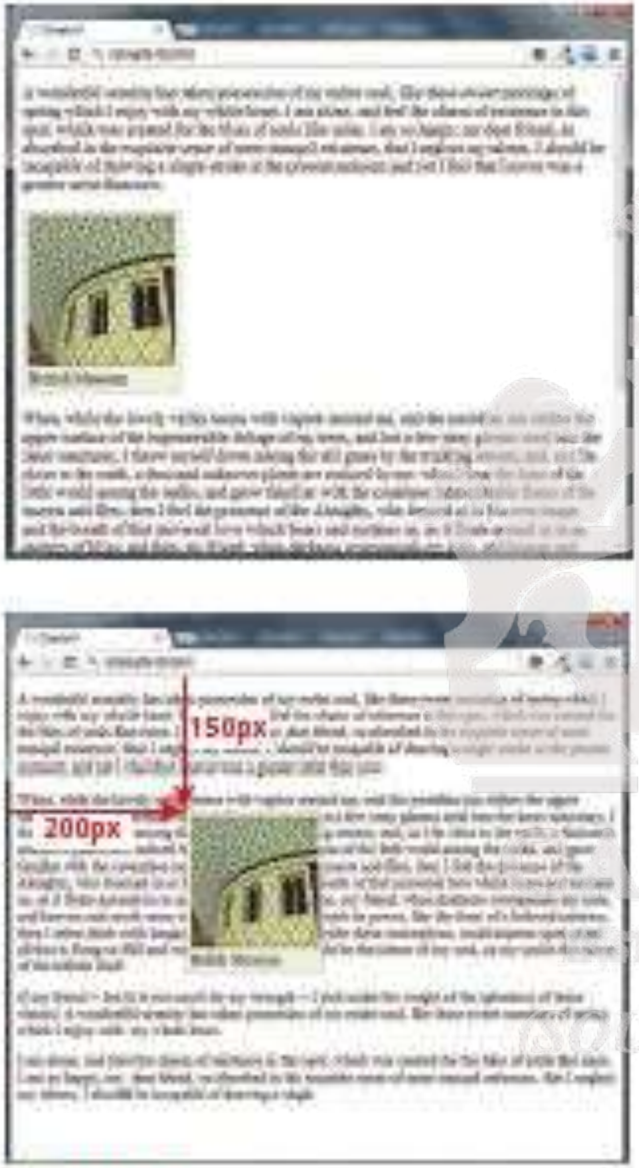
```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …
```

```
figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: relative;
   top: 150px;
   left: 200px;
}
```

FIGURE 5.4 Relative positioning

5.2.2 Absolute positioning

- When an element is positioned absolutely, it is removed completely from normal flow.
- Thus, unlike with relative positioning, space is not left for the moved element, as it no longer in the normal flow.
- Its position is moved in relation to its container block.



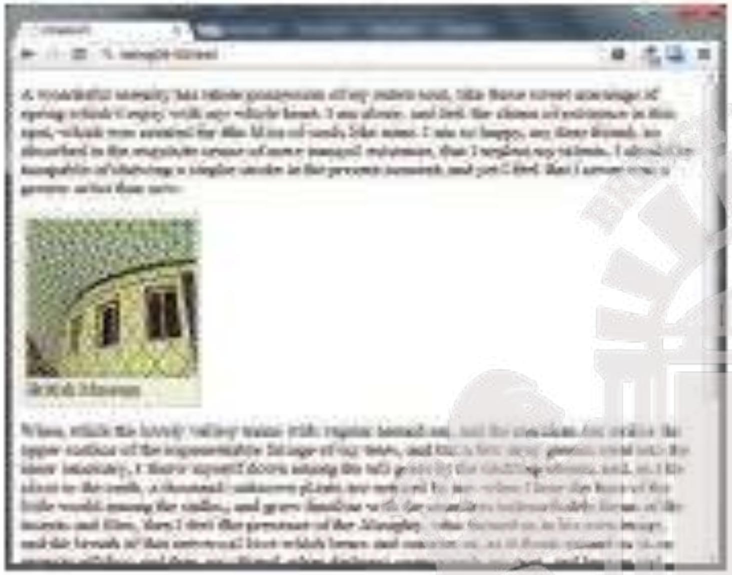
```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```

```
figure {
   margin: 0;
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: absolute;
   top: 150px;
   left: 200px;
}
```

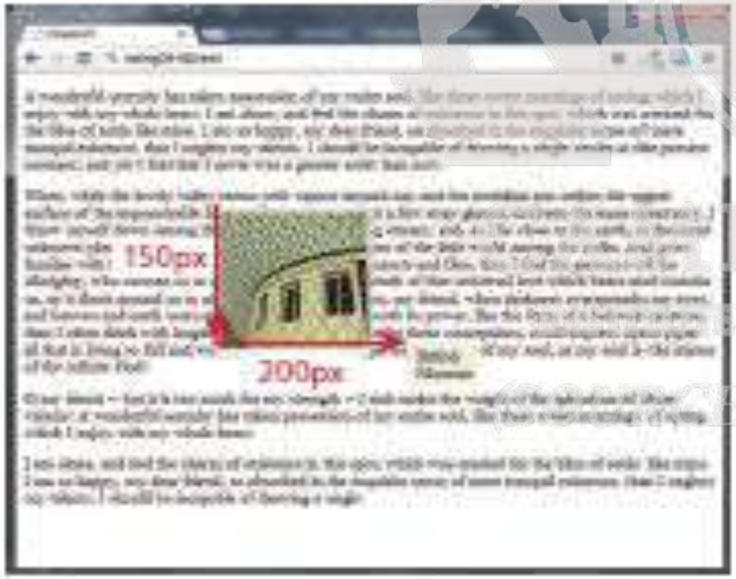
FIGURE 5.5 Absolute positioning

- A moved element via absolute position is actually relative to its nearest positioned ancestor container(i.e, a block level element whose position is fixed, relative or absolute).

- In the below fig, the <figcaption> is absolutely positioned, it is moved 150px down and 200 px to the left of its nearest positional ancestor, which happens to be its parent (<figure> element)



```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```

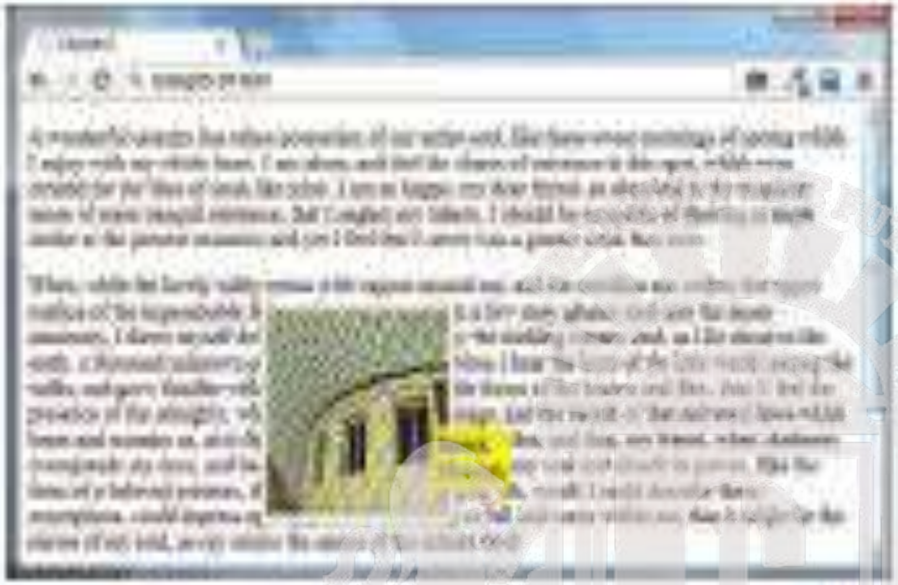


```
figure {
   margin: 0;
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: absolute;
   top: 150px;
   left: 200px;
}
figcaption {
   background-color: #EDEDDD;
   padding: 5px;
   position: absolute;
   top: 150px;
   left: 200px;
}
```

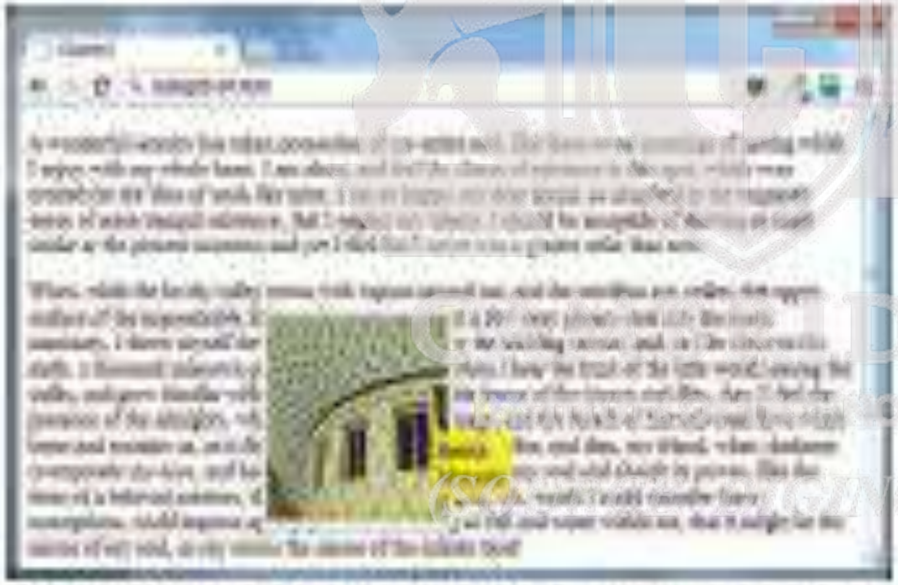
FIGURE 5.6 Absolute position is relative to nearest positioned ancestor container.

5.2.3 Z - Index

- Looking at above fig, you may wonder what would have happened if the <figcaption> had been moved so that it overlapped the <figure>.
- Each positioned element has a stacking order defined by the z – index property.
- Items closest to the viewer have a larger z-index.
- Working with z- index can be tricky.
- First, only positioned elements will make use of their z-index.
- Second, as in below fig, simply setting the z-index value of elements will not necessarily move them on top or behind other items.

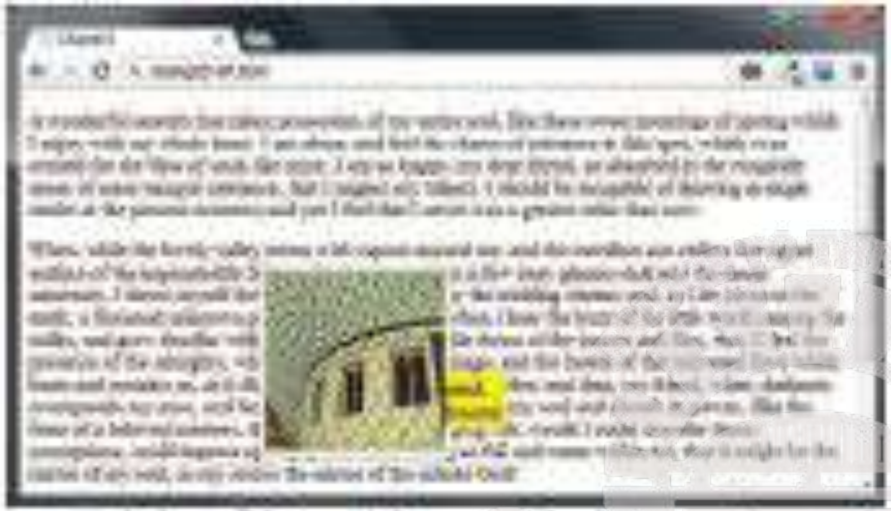


```
figure {
   position: absolute;
   top: 150px;
   left: 200px;
}
figcaption {
   position: absolute;
   top: 90px;
   left: 140px;
}
```



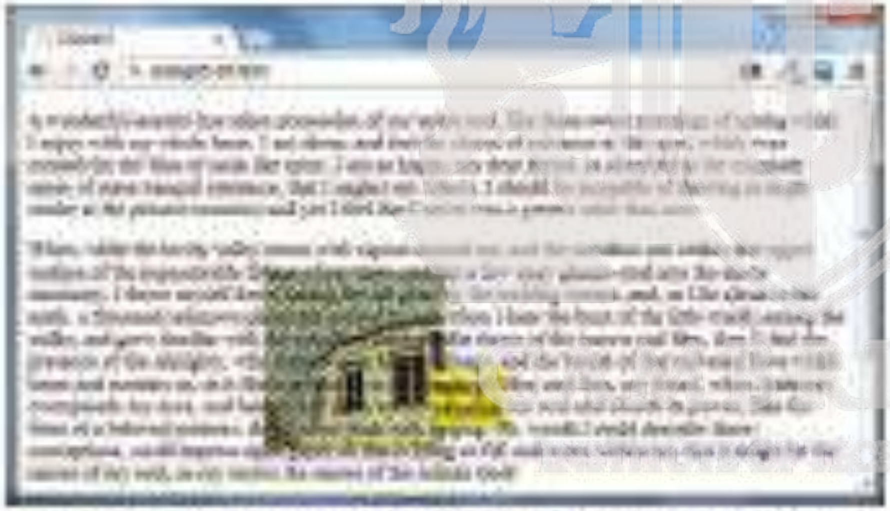
```
figure {
   -
   z-index: 5;
}
figcaption {
   -
   z-index: 1;
}
```

Note that this did not move the <figure> on top of the <figcaption> as one might expect. This is due to the nesting of the caption within the figure.



```
figure {
   …
   z-index: 1;
}
figcaption {
   …
   z-index: -1;
}
```

Instead the `<figcaption>` `z-index` must be set below 0. The `<figure>` `z-index` could be any value equal to or above 0.



```
figure {
   …
   z-index: -1;
}
figcaption {
   …
   z-index: 1;
}
```

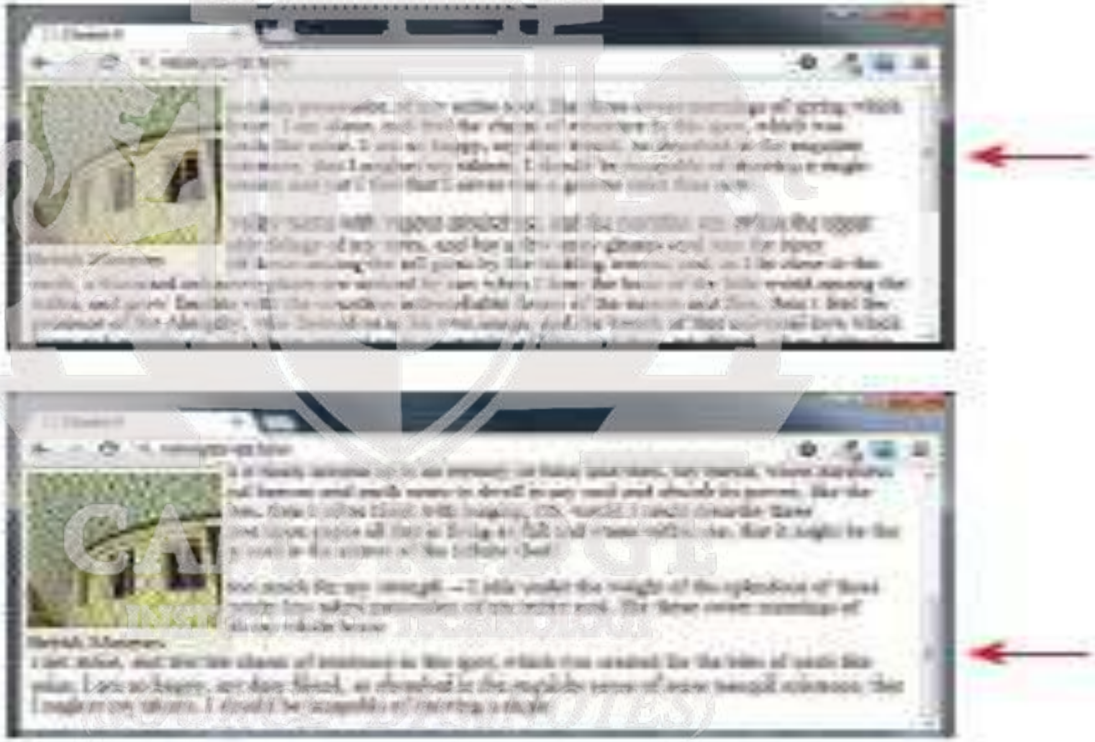
If the `<figure>` `z-index` is given a value less than 0, then any of its positioned descendants change as well. Thus both the `<figure>` and `<figcaption>` move underneath the body text.

FIGURE 5.7 Z-Index

5.2.4 Fixed

- The fixed position value is used relatively infrequently.
- It’s a type of absolute positioning, except that the positioning values are in relation to the viewport.
- Elements with fixed positioning do not move when the user scrolls up or down the page.

```
figure {
    ...
    position: fixed;
    top: 0;
    left: 0;
}
```

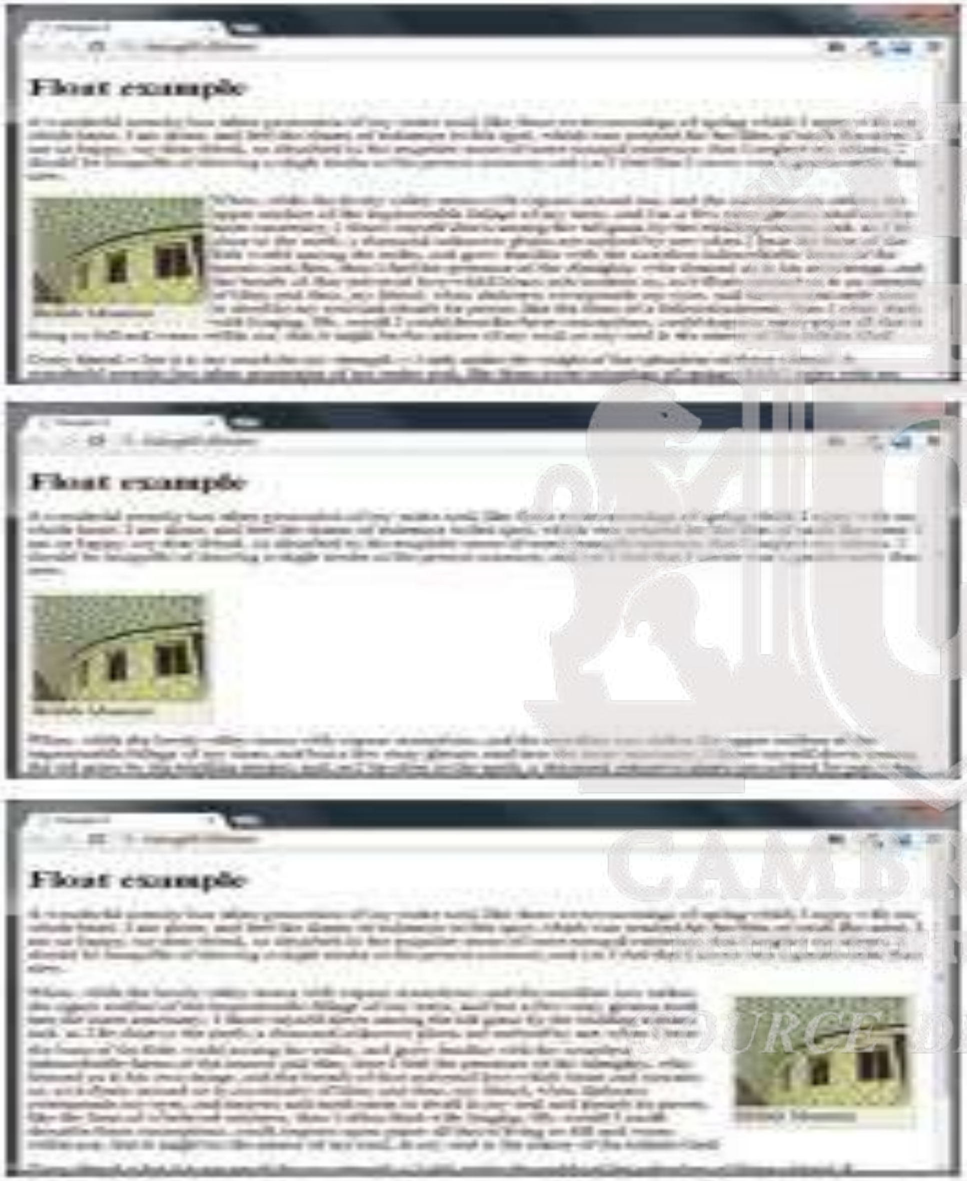


Notice that figure is fixed in its position regardless of what part of the page is being viewed.

FIGURE 5.8 Fixed position

5.3 Floating elements

- It is possible to displace an element out of its position in the normal flow via the css float property.
- It means to far left or far right of its containing block and rest of the content is “re-flowed” around the floated element, as in below fig



```
<h1>Float example</h1>
<p>A wonderful serenity has taken ...</p>
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …</p>

figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   margin: 0;
   padding: 5px;
   width: 150px;
}
```

Notice that a floated block-level element must have a width specified.

```
figure {
   …
   width: 150px;
   float: left;
}
```

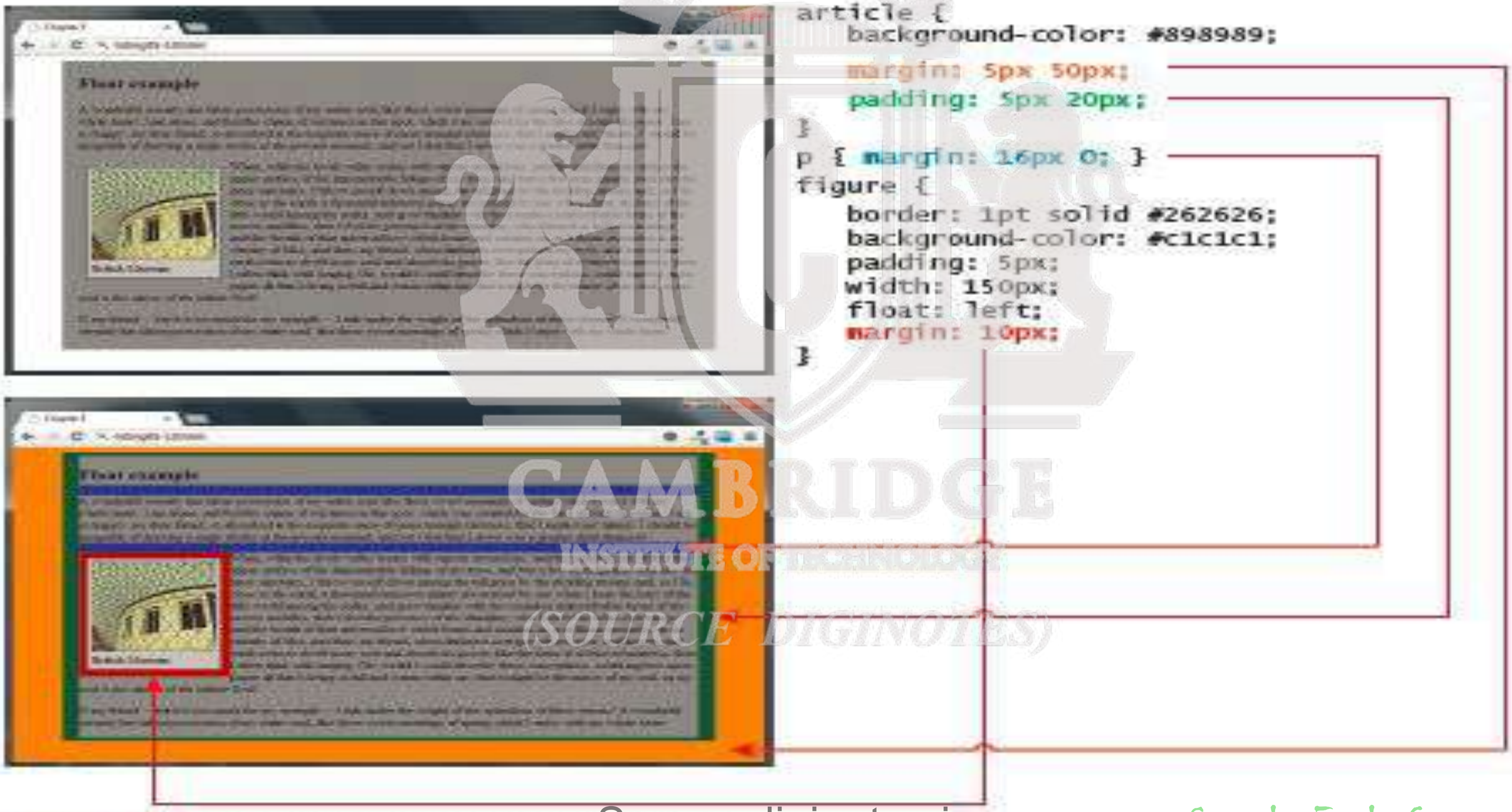
```
figure {
   …
   width: 150px;
   float: right;
   margin: 10px;
}
```

FIGURE 5.9 Floating an element

5.3.1 Floating with a container

- Float will help to move left or right of its container also called as container block
- In above fig container is HTML document itself so figures moves left or right of browser window.
- In below example the floated figure is contained within an <article> element that is indented from the browser's edge.

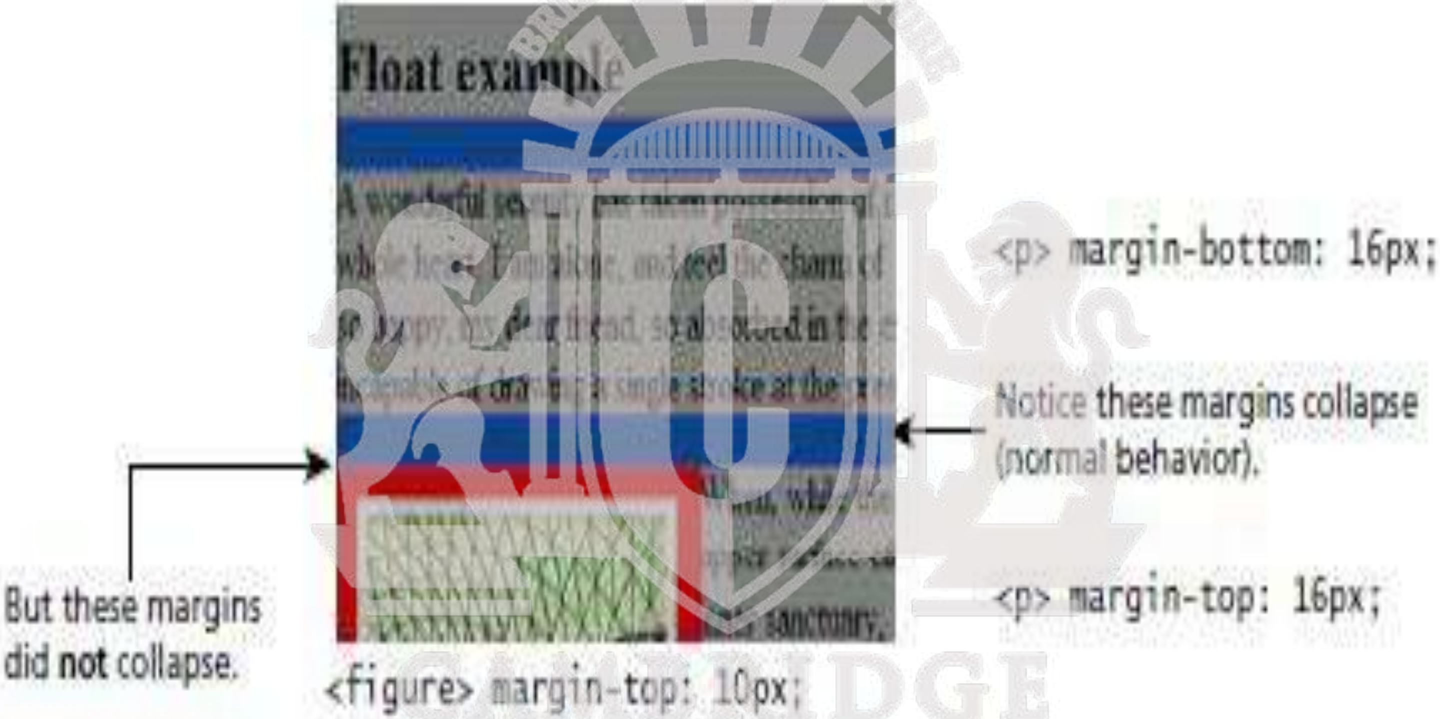
```
<article>
  <h1>Float example</h1>
  <p>A wonderful serenity has taken possession of … </p>

  <figure>
    <img src="images/828.jpg" alt="" />
    <figcaption>British Museum</figcaption>
  </figure>

  <p>When, while the lovely valley teems with ...</p>

  <p>O my friend -- but it is too much for my ...</p>
</article>
```



```
article {
   background-color: #898989;
   margin: 5px 50px;
   padding: 5px 20px;
}
p { margin: 16px 0; }
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 150px;
   float: left;
   margin: 10px;
}
```

FIGURE 5.10 Floating to the containing block

- There is an important change happening which can be seen only by zooming the above figure.
- The overlapping margins for the adjacent <p> elements behave normally and collapse.
- But notice that the top margin for the floated <figure> and the bottom margin for the <p> element above it do not collapse.



<p> margin-bottom: 16px;

Notice these margins collapse (normal behavior).

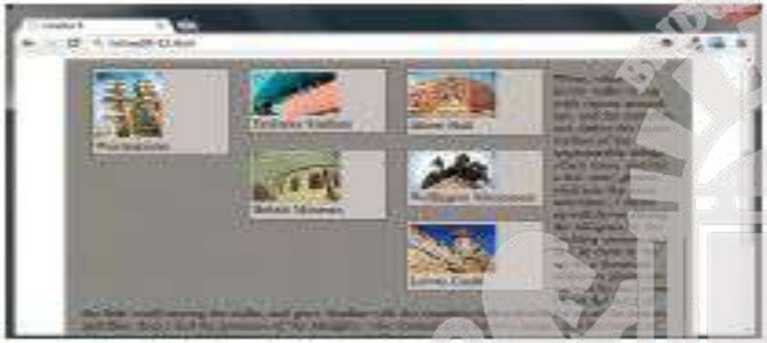
<p> margin-top: 16px;

But these margins did **not** collapse.

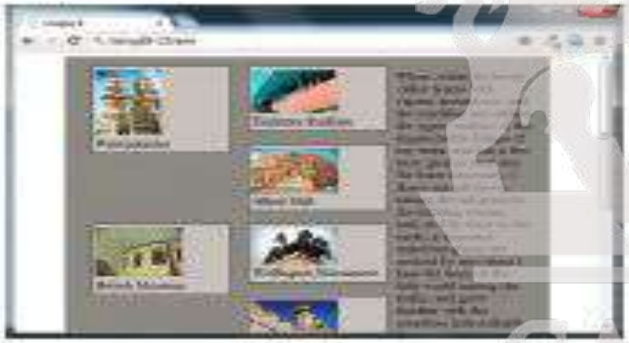
<figure> margin-top: 10px;

FIGURE 5.11 Margins do not collapse on floated block-level elements.

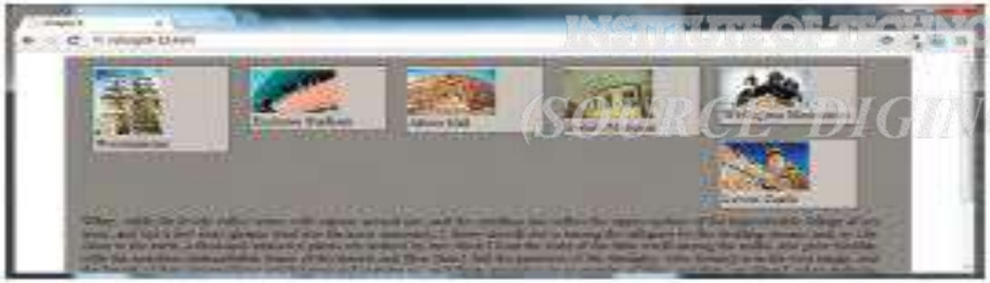
5.3.2 Floating Multiple items side by side



```
<article>
   <figure>
      <img src="images/tiny/275.jpg" alt="" />
      <figcaption>Westminister</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/700.jpg" alt="" />
      <figcaption>Emirates Stadium</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/537.jpg" alt="" />
      <figcaption>Albert Hall</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/828.jpg" alt="" />
      <figcaption>British Museum</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/464.jpg" alt="" />
      <figcaption>Wellington Monument</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/224.jpg" alt="" />
      <figcaption>Lewes Castle</figcaption>
   </figure>
   <p>When, while the lovely valley teems ...
</article>
```

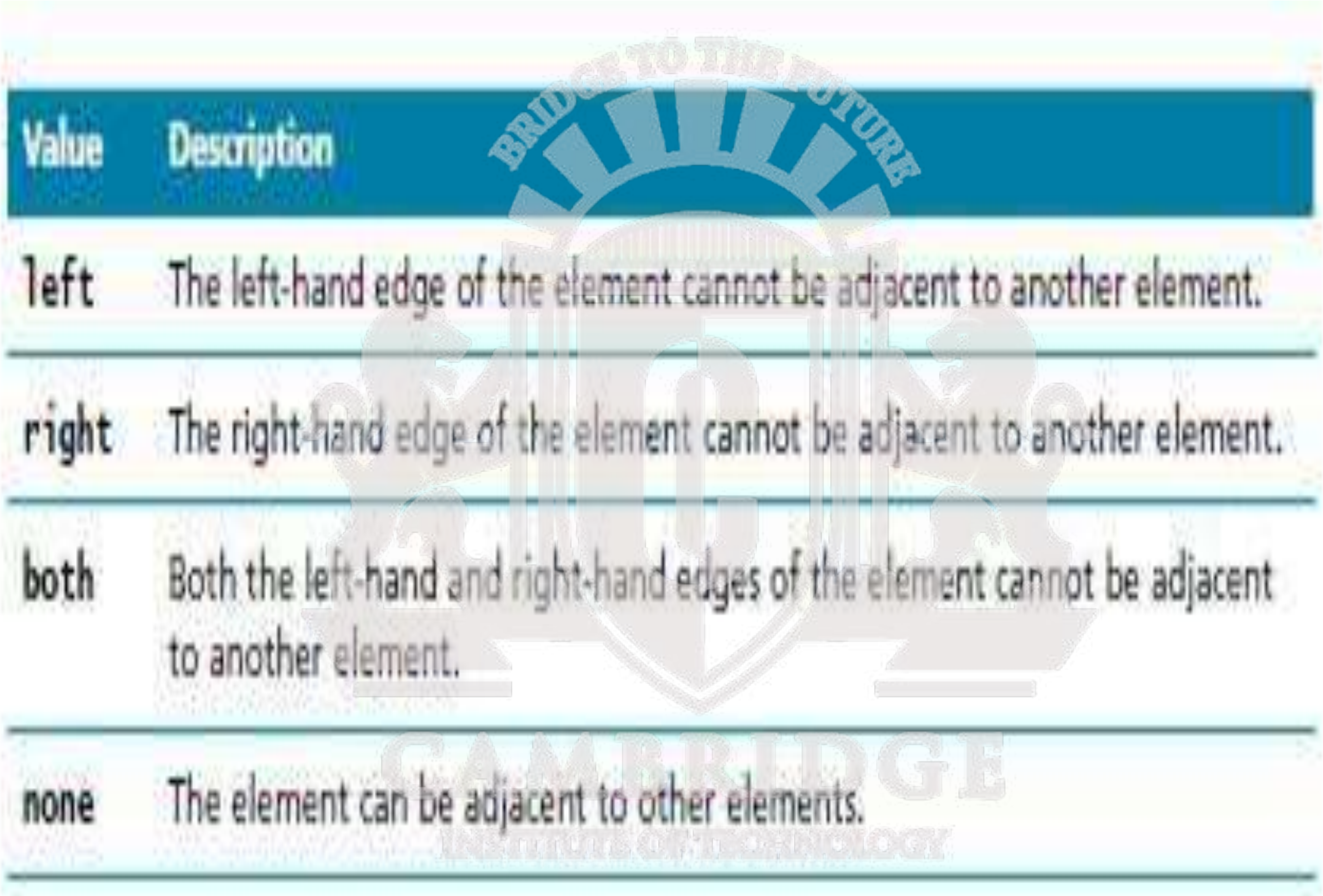


```
figure {
   ...
   width: 150px;
   float: left;
}
```



As the window resizes, the content in the containing block (the `<article>` element), will try to fill the space that is available to the right of the floated elements.

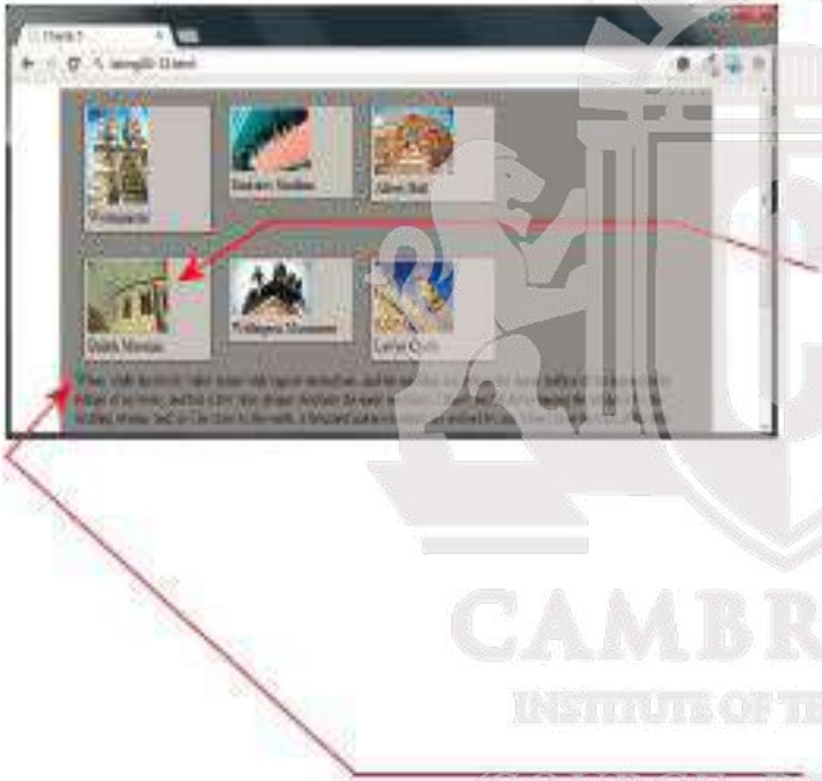
FIGURE 5.12 Problems with multiple floats



Value	Description
left	The left-hand edge of the element cannot be adjacent to another element.
right	The right-hand edge of the element cannot be adjacent to another element.
both	Both the left-hand and right-hand edges of the element cannot be adjacent to another element.
none	The element can be adjacent to other elements.

TABLE 5.2 Clear Property

```
.first { clear: left; }
```



```
<article>
  <figure>
     <img src="images/tiny/275.jpg" alt="" />
     <figcaption>Westminister</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/700.jpg" alt="" />
     <figcaption>Emirates Stadium</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/537.jpg" alt="" />
     <figcaption>Albert Hall</figcaption>
  </figure>
  <figure class="first">
     <img src="images/tiny/828.jpg" alt="" />
     <figcaption>British Museum</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/464.jpg" alt="" />
     <figcaption>Wellington Monument</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/224.jpg" alt="" />
     <figcaption>Lewes Castle</figcaption>
  </figure>
  <p class="first">When, while the lovely valley ...
</article>
```

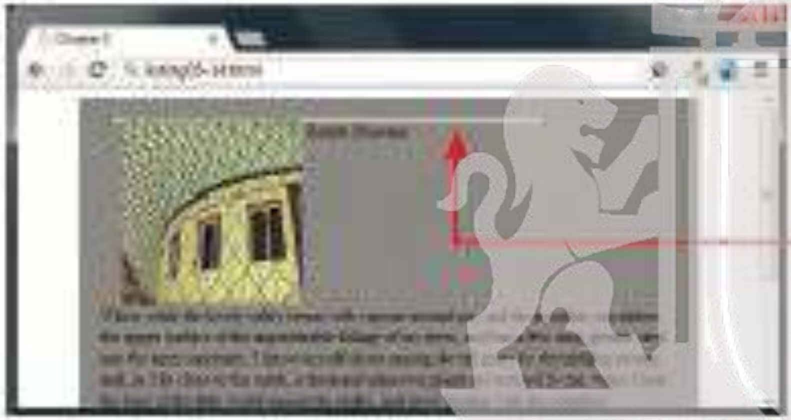
FIGURE 5.13 Using the clear property

5.3.3 Containing Floats

- Another problem that can occur with floats is when an element is floated within a containing block that contains only floated content.
- In that case, the containing block essentially disappears(fig).
- In the below figure, the <figure> containing block contains only an <img> and a figcaption element and both of these elements are floated to the left.
- That means both elements have been removed from the normal flow, from the browser's perspective, since the <figure> contains no normal flow content, it essentially has nothing in it, hence it has a content height of zero.

- Best solution for this problem is to use overflow property.

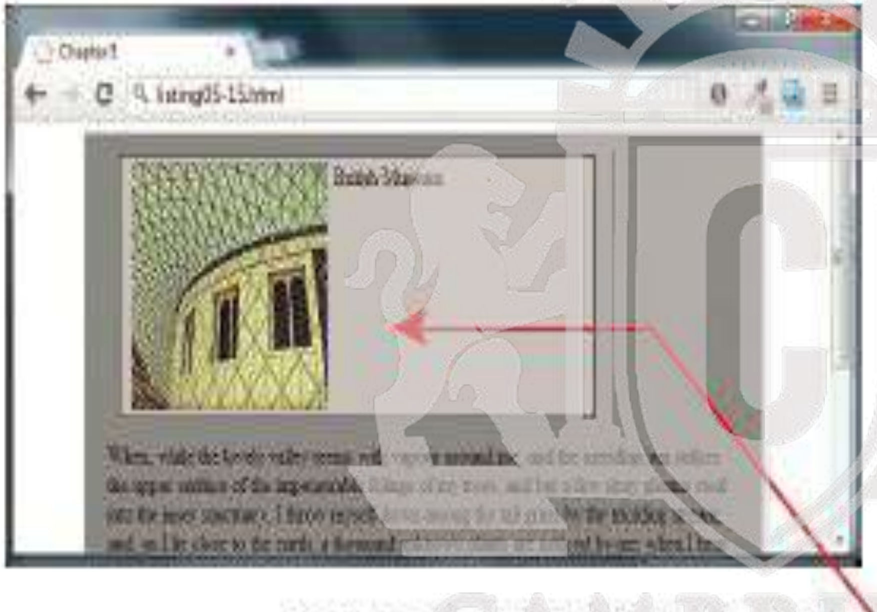
```
<article>
 <figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
 </figure>
 <p class="first">When, while the lovely valley …
</article>
```



Notice that the <figure> element's content area has shrunk down to zero (it now just has padding space and borders).

```
figure img {
   width: 170px;
   float: left;
   margin: 0 5px;
}
figure figcaption {
   width: 100px;
   float: left;
}
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 400px;
   margin: 10px;
}
.first { clear: left; }
```

FIGURE 5.14 Disappearing parent containers



```
figure img {
   width: 170px;
   float: left;
   margin: 0 5px;
}
figure figcaption {
   width: 100px;
   float: left;
}
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 400px;
   margin: 10px;
   overflow: auto;
}
```

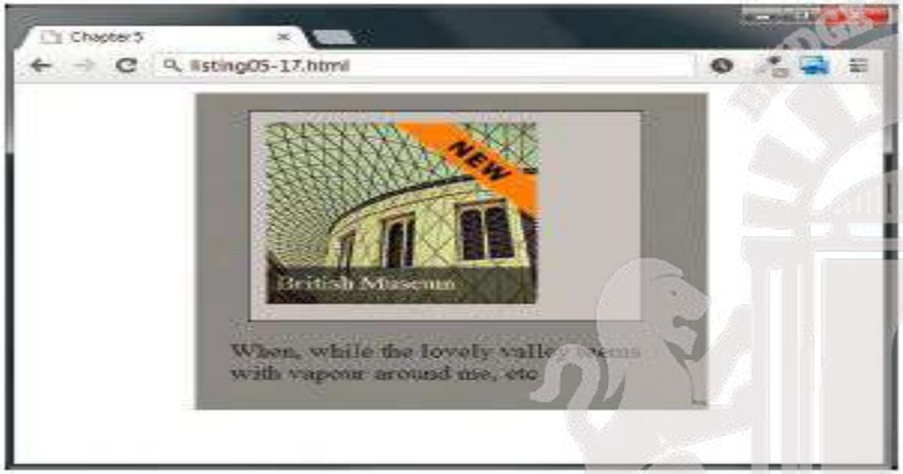
Setting the overflow property to auto solves the problem.

FIGURE 5.15 Using the overflow property

5.3.4 Overlaying and Hiding elements

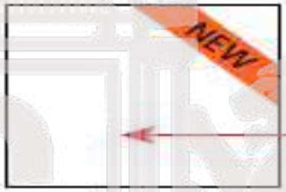
- One of the most common design tasks with CSS is to place two elements on top of each other or to selectively hide and display elements. Positioning is important for both tasks.

```
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
   <img src="images/new-banner.png" alt="" class="overlayed"/>
</figure>
```



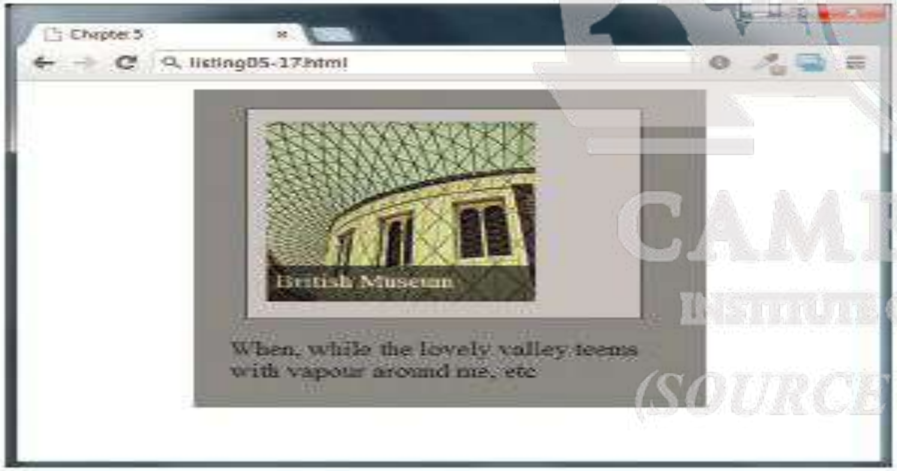
When, while the lovely valley teems with vapour around me, etc

```
.overlayed {
   position: absolute;
   top: 10px;
   left: 10px;
}
```



Transparent area

new-banner.png



When, while the lovely valley teems with vapour around me, etc

```
.overlayed {
   position: absolute;
   top: 10px;
   left: 10px;
   display: none;
}
```

This hides the overlayed image.

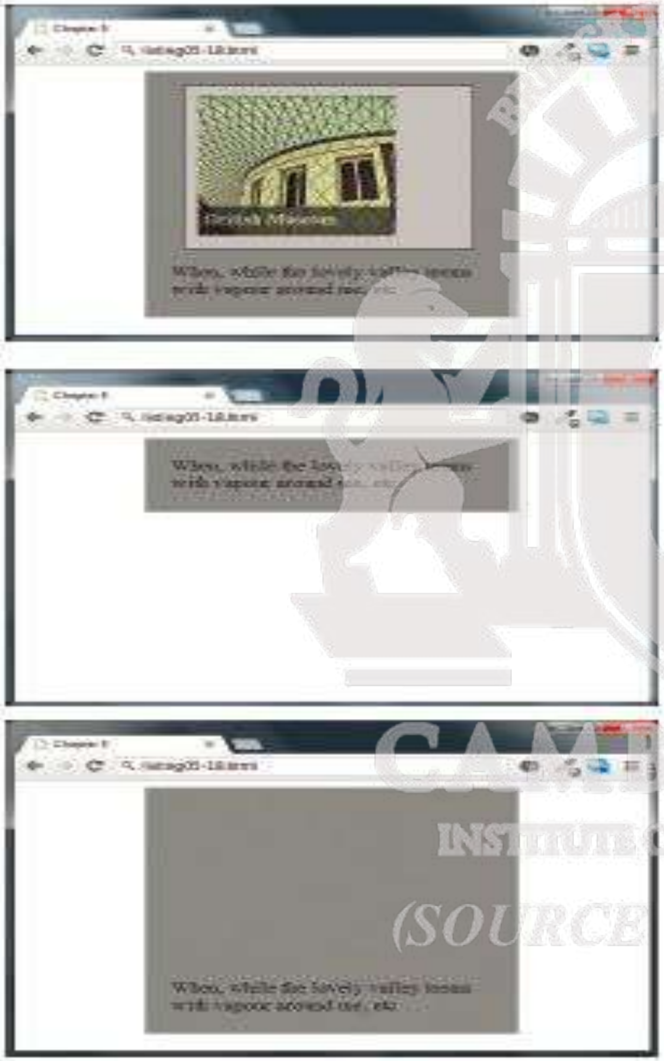
```
.hide {
   display: none;
}
```

This is the preferred way to hide: by adding this class to another element. This makes it clear in the markup that an element is not visible.

```
<img ... class="overlayed hide"/>
```

FIGURE 5.17 Using the display property

- There are two different ways to hide elements in css:
  - Display property
    - It takes an item out of the flow – element no longer exists
  - Visibility property
    - Hides the element, but the space for that element remains



When, while the lovely valley teems with vapour around me, etc.

```
figure {
   ...
   display: auto;
}
```

When, while the lovely valley teems with vapour around me, etc.

```
figure {
   ...
   display: none;
}
```

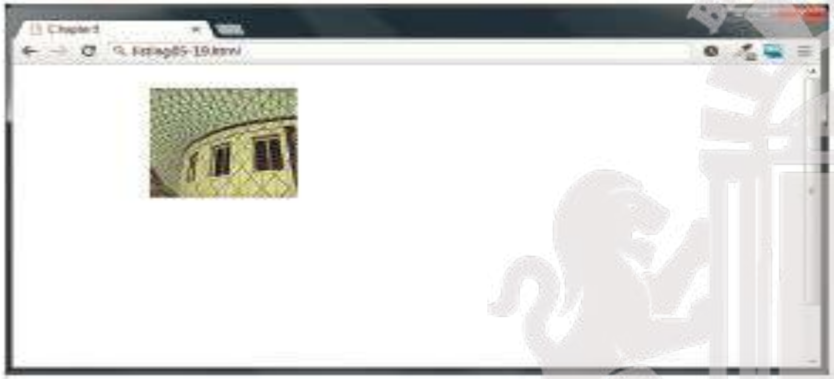
When, while the lovely valley teems with vapour around me, etc.

```
figure {
   ...
   visibility: hidden;
}
```

FIGURE 5.18 Comparing display to visibility

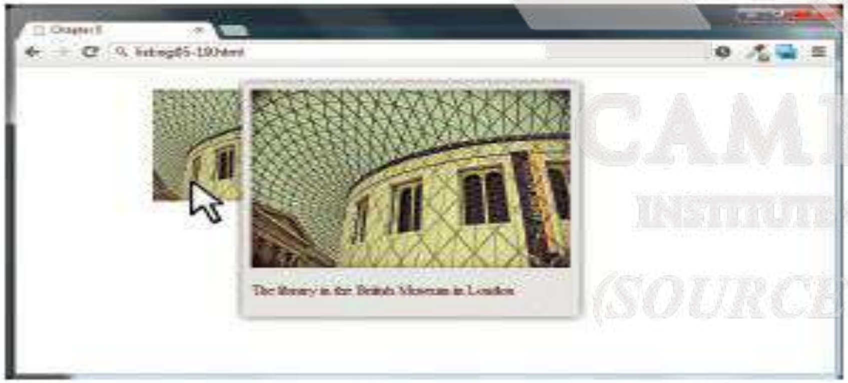
- It is also possible to make use of these properties by using
  - Hover and pseudo class.

```
<figure class="thumbnail">
  <img src="images/828.jpg" alt="" />
  <figcaption class="popup">
     <img src="images/828-bigger.jpg" alt="" />
     <p>The library in the British Museum in London</p>
  </figcaption>
</figure>
```



```
figcaption.popup {
   padding: 10px;
   background: #e1e1e1;
   position: absolute;

   /* add a drop shadow to the frame */
   -webkit-box-shadow: 0 0 15px #A9A9A9);
   -moz-box-shadow: 0 0 15px #A9A9A9;
   box-shadow: 0 0 15px #A9A9A9;

   /* hide it until there is a hover */
   visibility: hidden;
}
```

When the page is displayed, the larger version of the image, which is within the <figcaption> element, is hidden.



```
figure.thumbnail:hover figcaption.popup {
   position: absolute;
   top: 0;
   left: 100px;

   /* display image upon hover */
   visibility: visible;

}
```

When the user moves/hovers the mouse over the thumbnail image, the visibility property of the <figcaption> element is set to visible.

FIGURE 5.19 Using hover with display

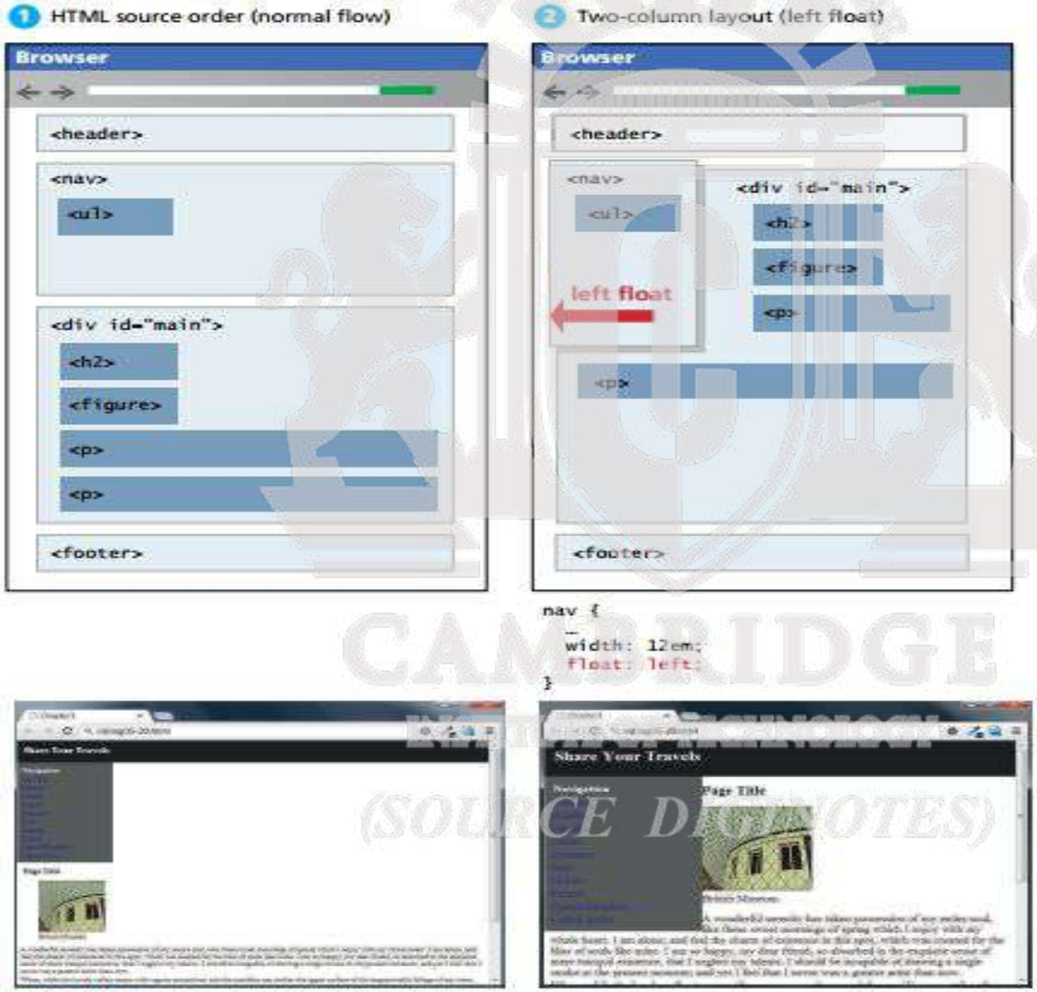
5.4 Constructing Multicolumn Layouts

The previous sections showed two different ways to move items out of the normal top-down flow, namely, by using positioning and by using floats.

They are the raw techniques that you can use to create more complex layouts

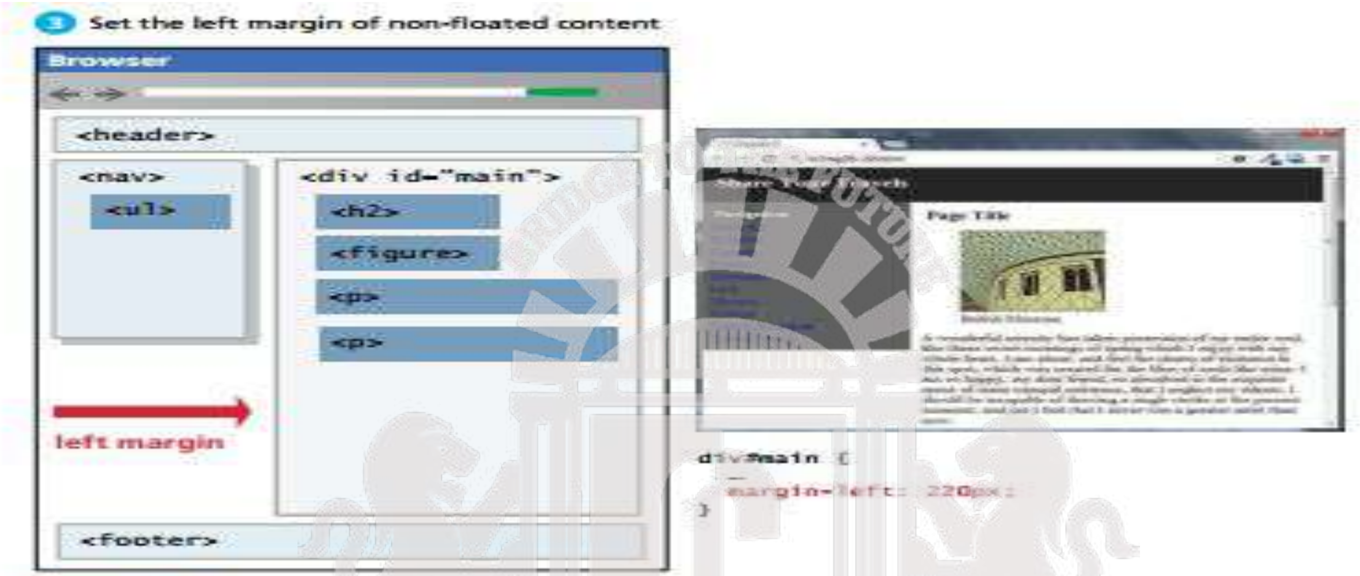
5.4.1Using Floats to Create Columns

- The first step is to float the content container that will be on the left-hand side.



```
nav {
  …
  width: 12em;
  float: left;
}
```

FIGURE 5.20 Creating two-column layout, step one



```
div#main {
   margin-left: 220px;
}
```

FIGURE 5.21 Creating two-column layout, step two

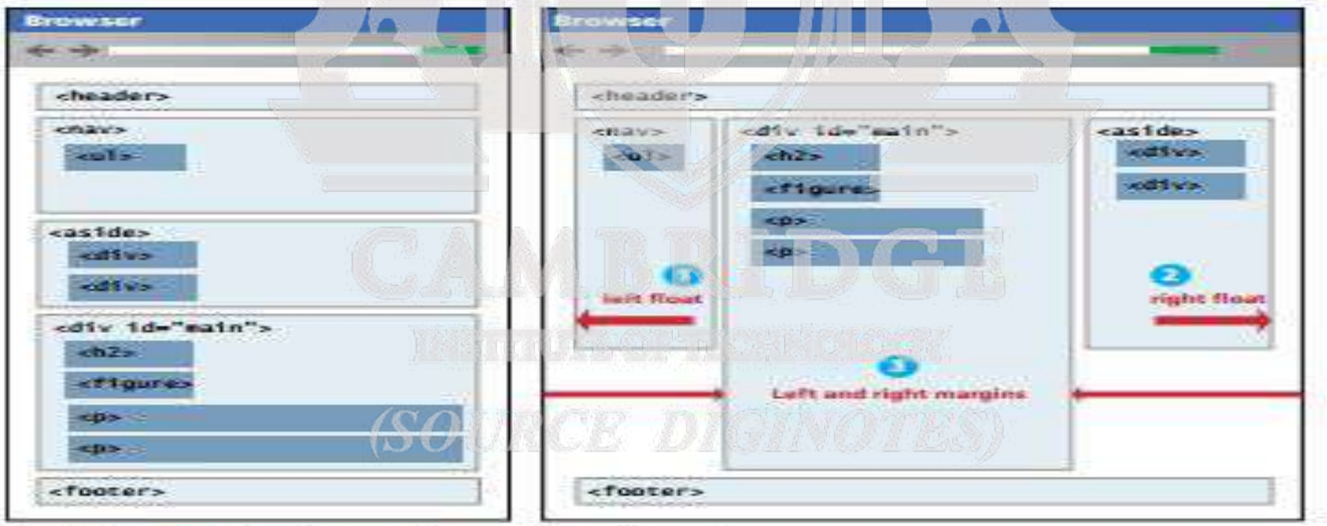


FIGURE 5.22 Creating a three-column layout

Main problem :

- This is the main problem with the floated approach: that we can’t necessarily put the source in an SEO-optimized order (which would be to put the main page content *before* the navigation and the aside).

- There are in fact ways to put the content in an SEO-optimized order with floats, but typically this requires
making use of certain tricks such as giving the main content negative margins

Using positioning to Create Columns

- Positioning can also be used to create a multicolumn layout. Typically, the approach will be to absolute position the elements.

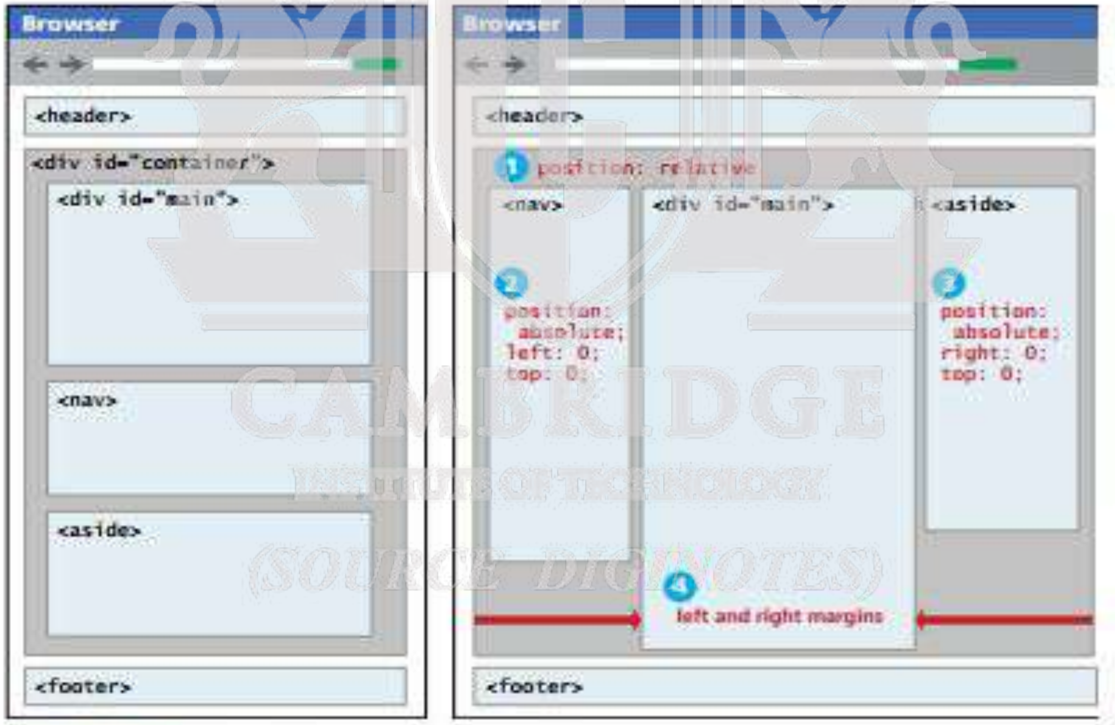
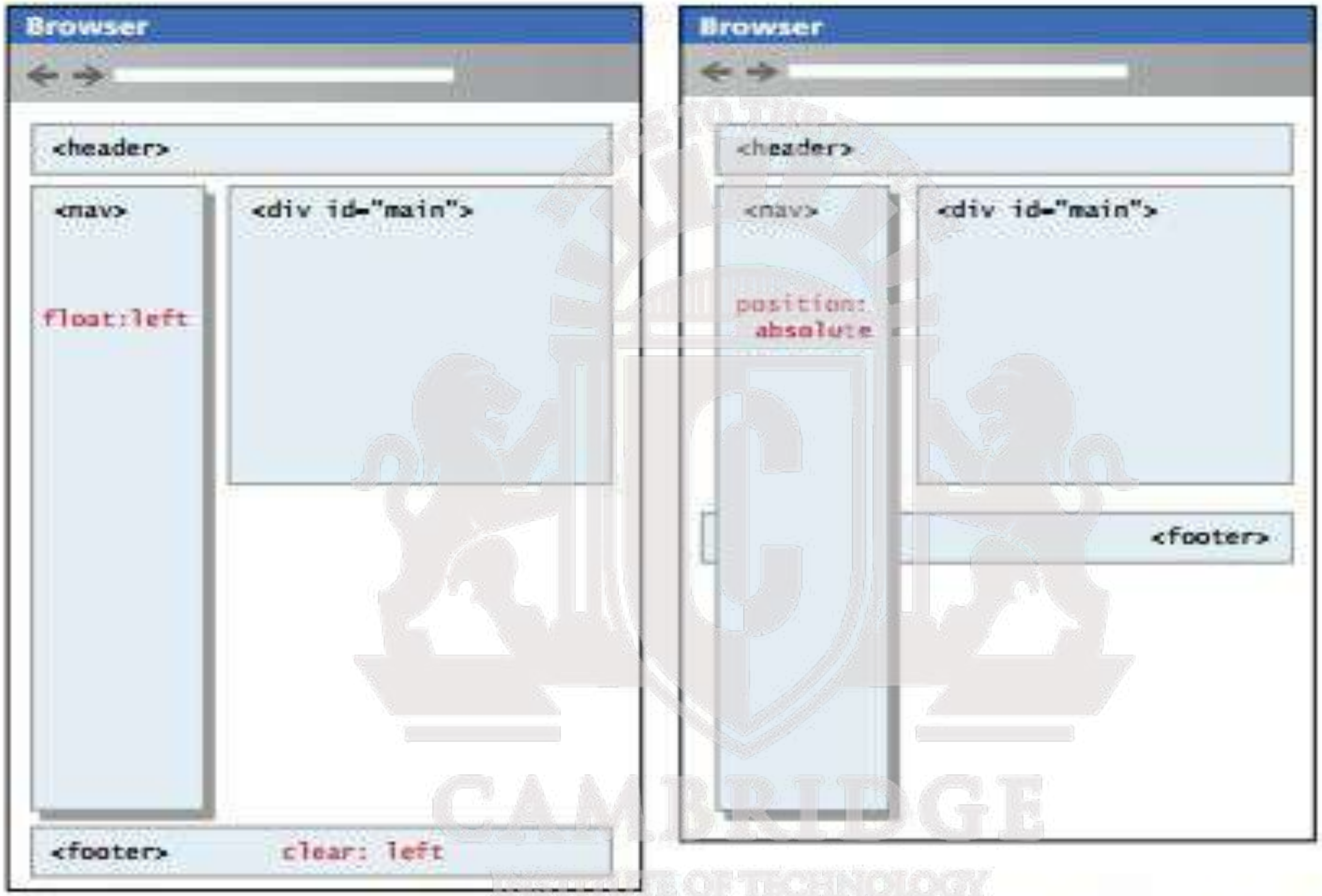


FIGURE 5.24 Three-column layout with positioning

Source diginotes.in



Elements that are floated leave behind space for them in the normal flow. We can also use the clear property to ensure later elements are below the floated element.

Absolute positioned elements are taken completely out of normal flow, meaning that the positioned element may overlap subsequent content. The clear property will have no effect since it only responds to floated elements.

FIGURE 5.25 Problems with absolute positioning

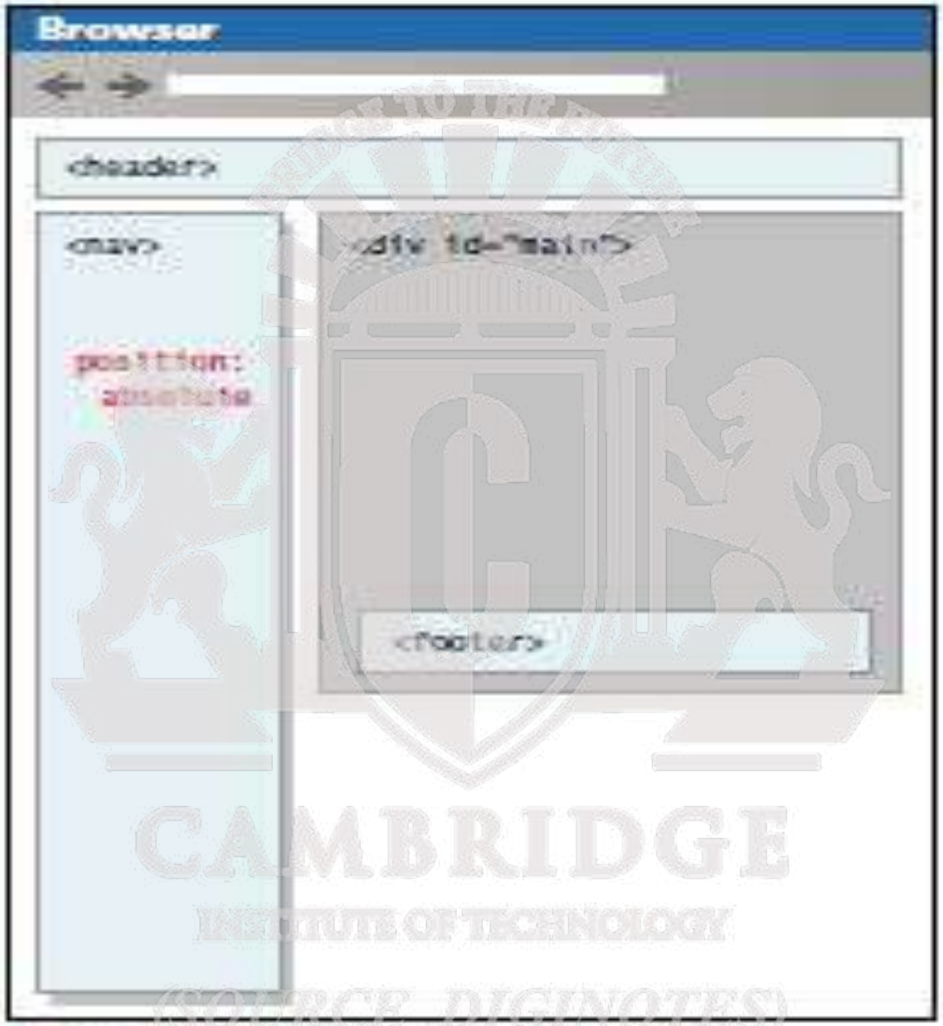


FIGURE 5.26 Solution to footer problem

Section 5 of 7

APPROACHES TO CSS LAYOUT

Approaches to CSS Layout

One of the main problems faced by web designers is that the size of the screen used to view the page can vary quite a bit.

- 21-inch wide screen monitor that can display 1920 x 1080 pixels
- older iPhone with a 3.5 screen and a resolution of 320x480 px

Satisfying both users can be difficult; the approach to take for one type of site content might not work as well with another site with different content.

Approaches to CSS Layout

Most designers take one of two basic approaches to dealing with the problems of screen size.

- Fixed Layout
- Liquid Layout
- Hybrid Layout

Fixed Layout

It isn't even broken

In a **fixed layout**, the basic width of the design is set by the designer, typically corresponding to an “ideal” width based on a “typical” monitor resolution

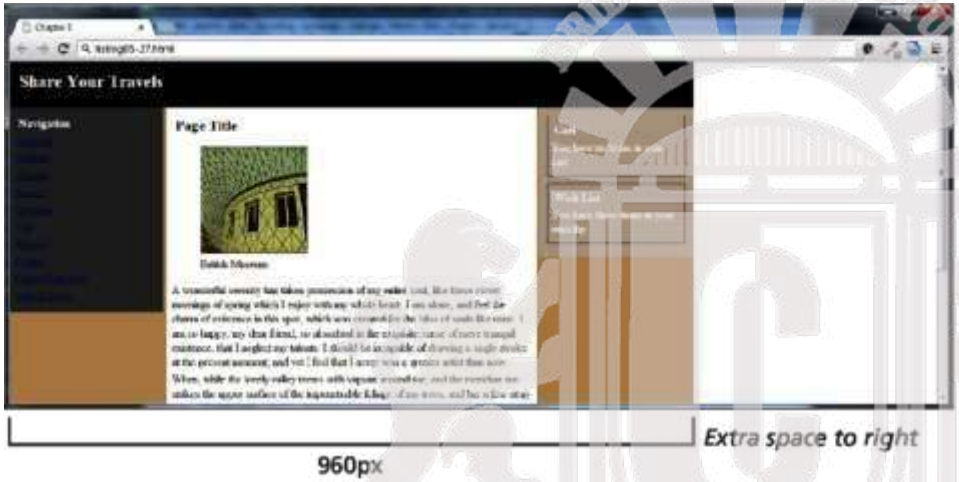
The advantage of a fixed layout is that it is easier to produce and generally has a predictable visual result.

Fixed layouts have drawbacks.

- For larger screens, there may be an excessive amount of blank space to the left and/or right of the content.
- It is also optimized for typical desktop monitors; however, as more and more user visits are happening via smaller mobile devices

Fixed Layout

Notice the fixed size

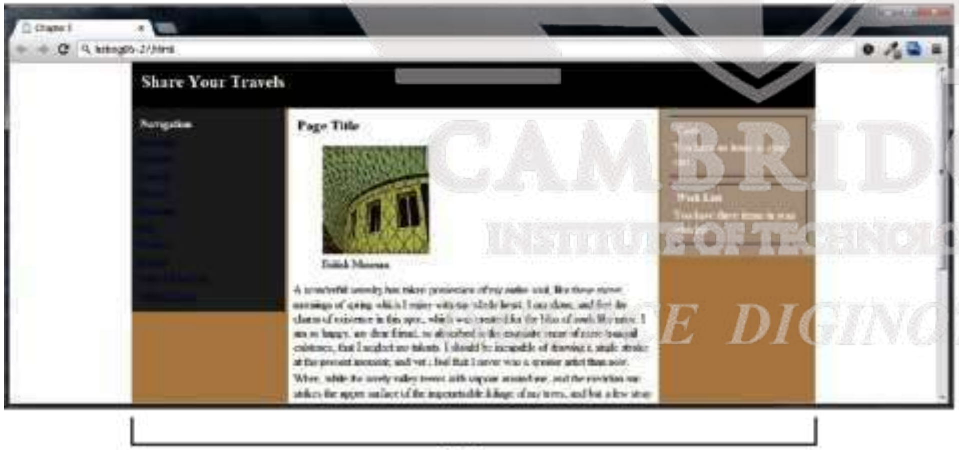


960px

Extra space to right

```
div#wrapper {
  width: 960px;
  background-color: tan;
}
```

```
<body>
   <div id="wrapper">
      <header>
      ...
      </header>
      <div id="main">
      ...
      </div>
      <footer>
      ...
      </footer>
   </div>
</body>
```



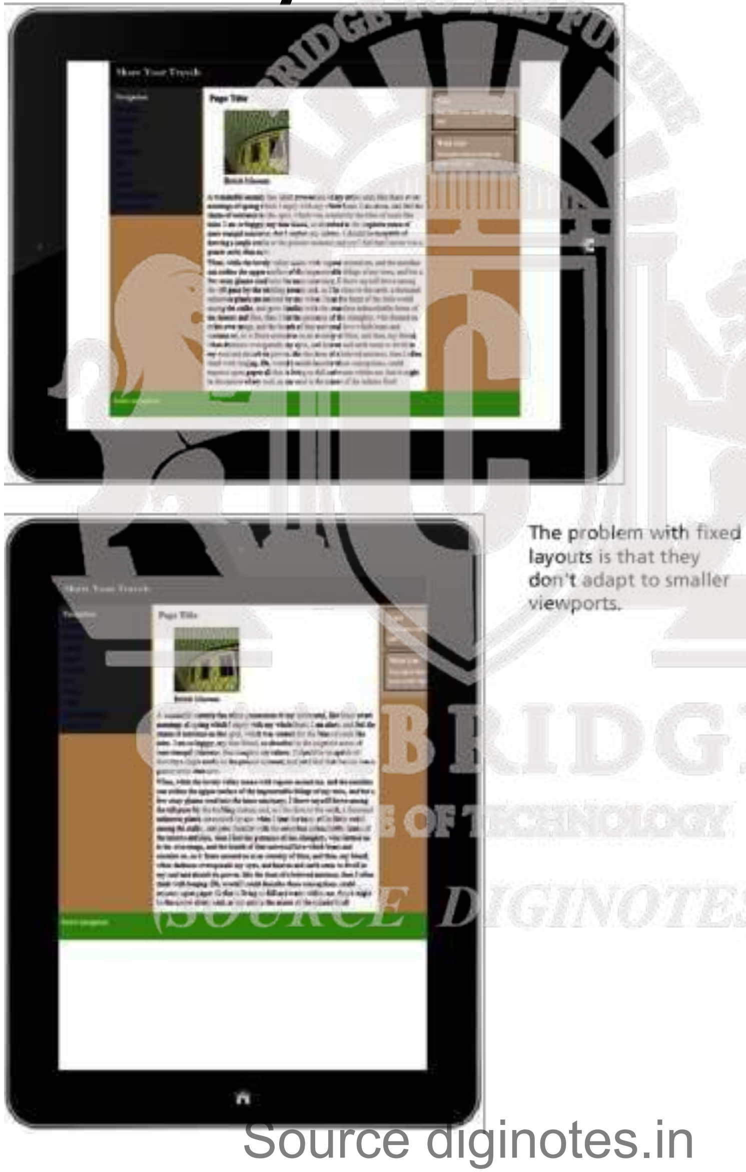
960px

Equal space to the left and to right

```
div#wrapper {
  width: 960px;
  margin-left: auto;
  margin-right: auto;
  background-color: tan;
}
```

Source diginotes.in

Problem with fixed layouts



The problem with fixed layouts is that they don't adapt to smaller viewports.

Liquidy goodness

Liquid Layout

In a **liquid layout** (also called a **fluid layout**) widths are not
specified using pixels, but percentage values.

Advantage:

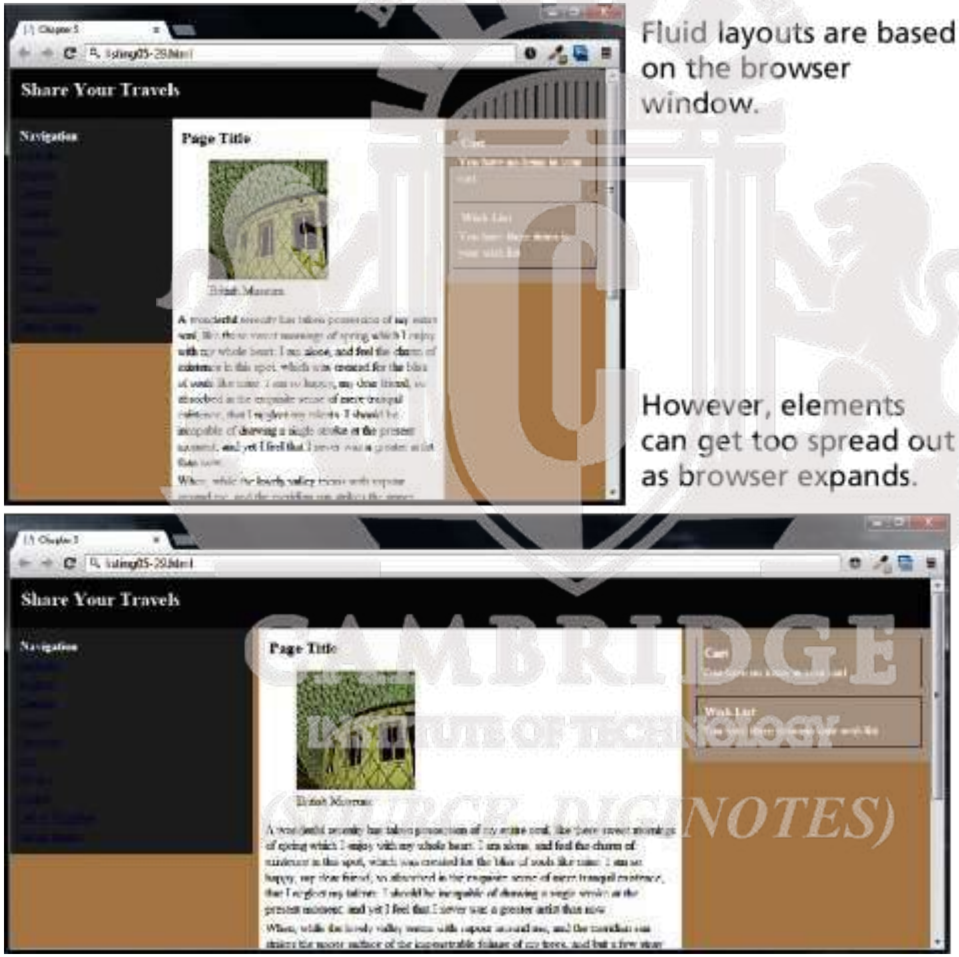
- adapts to different browser sizes,

Disadvantages:

- Liquid layouts can be more difficult to create because some elements, such as images, have fixed pixel sizes
- the line length (which is an important contributing factor to readability) may become too long or too short

Liquidy goodness

Liquid Layout



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray

Fluid layouts are based on the browser window.

However, elements can get too spread out as browser expands.

Hybrid Layout

Such a smug feeling

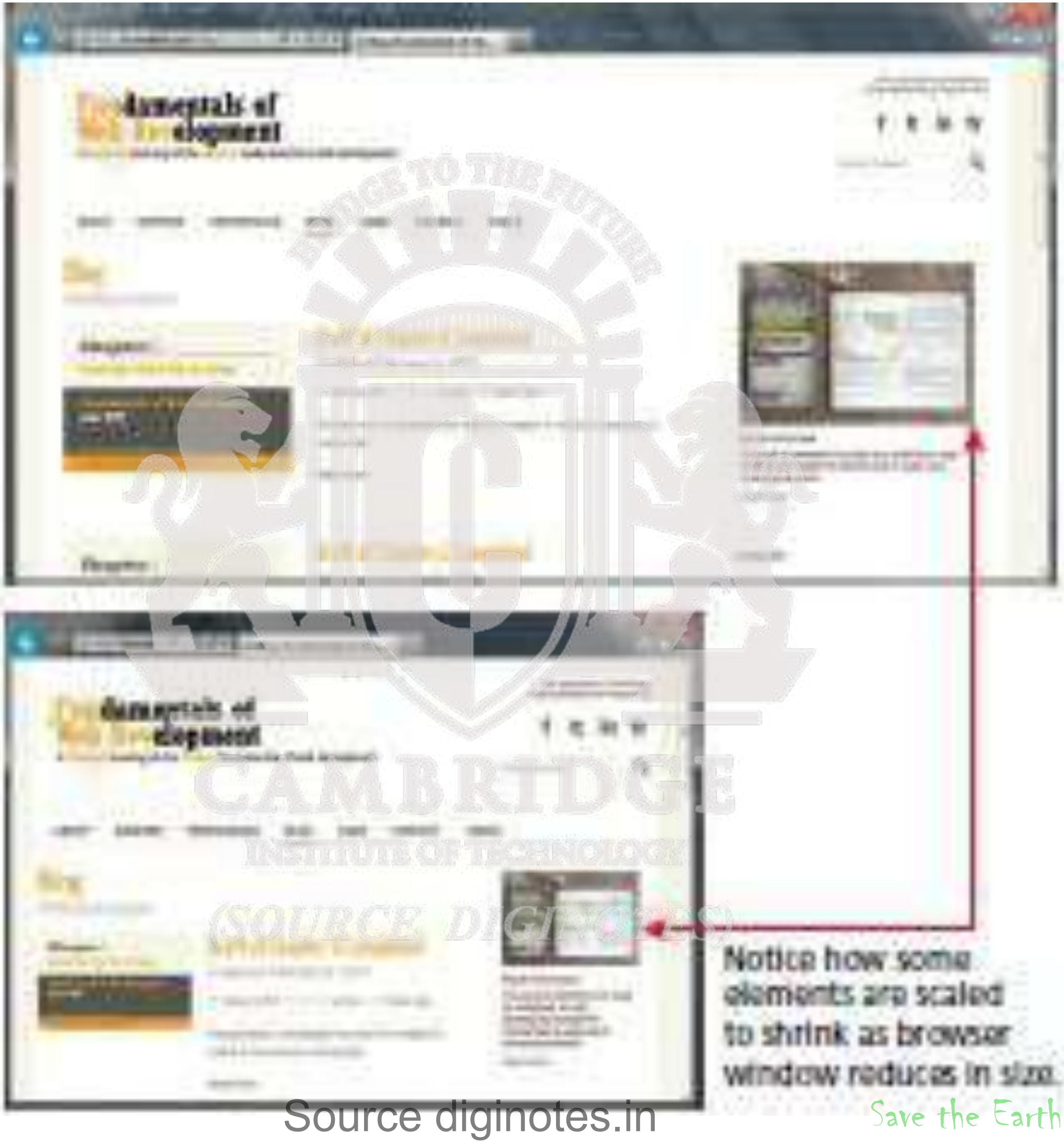
A **hybrid layout** combines pixel and percentage measurements.

- Fixed pixel measurements might make sense for a sidebar column containing mainly graphic advertising images that must always be displayed and which always are the same width.

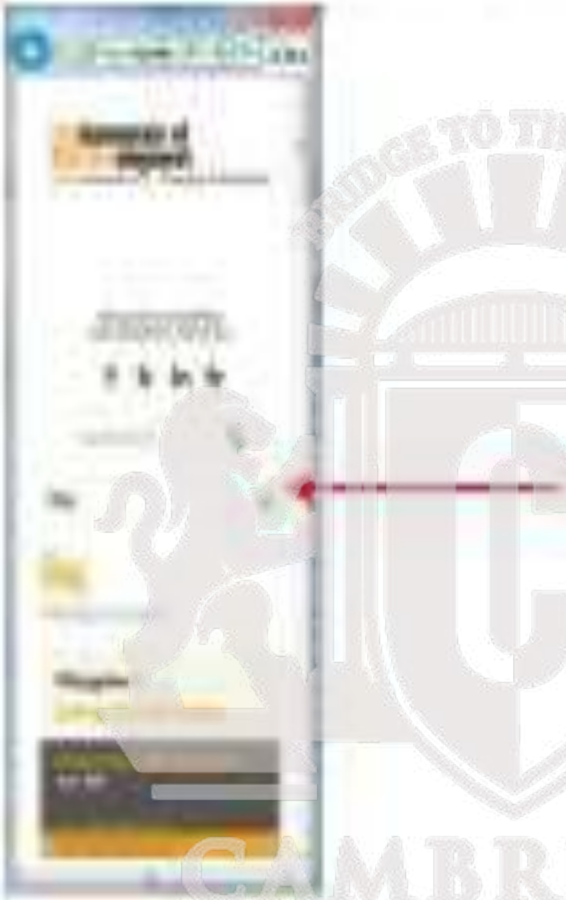
- percentages would make more sense for the main content or navigation areas, with perhaps min and max size limits in pixels set for the navigation areastime

5.6 Responsive design

- In the past several years, a lot of attention has been given to so called Responsive layout designs.
- In a responsive design, the page “responds” to changes in the browser size that go beyond the width scaling of a liquid layout.
- We had problems with liquid layout for images.
- In a responsive design layout, images will be scaled down and navigation elements will be replaced as the browser shrinks.


Fundamentals of Web Development
Fundamentals of Web Development

Notice how some elements are scaled to shrink as browser window reduces in size.



When browser shrinks below a certain threshold, then layout and navigation elements change as well.

In this case, the <ol> list of hyperlinks changes to a <select> and the two-column design changes to one column.

FIGURE 5.30 Responsive layouts

- There are 4 important key components to make responsive design work

1. Liquid layouts.
2. Scaling images to viewport size.
3. Setting viewports via the <meta> tag
4. Customizing the css for different viewports using media queries.

❖ Responsive designs begin with liquid layout, ie., in which most elements have their widths specified as percentages.

❖ Making images scale in size is quite straight forward

```
img   {
                max-width=100%;
          }
```

5.6.1 Setting Viewport

- If you have ever used a modern mobile browser, you may have been surprised to see how the web page was scaled to fit into the small screen of the browser.
- The way this works is the mobile browser renders the page on a canvas called the **viewport.**

- On iphones, - the viewport width is 980 px, and then that viewport is scaled to fit the current width of the any device

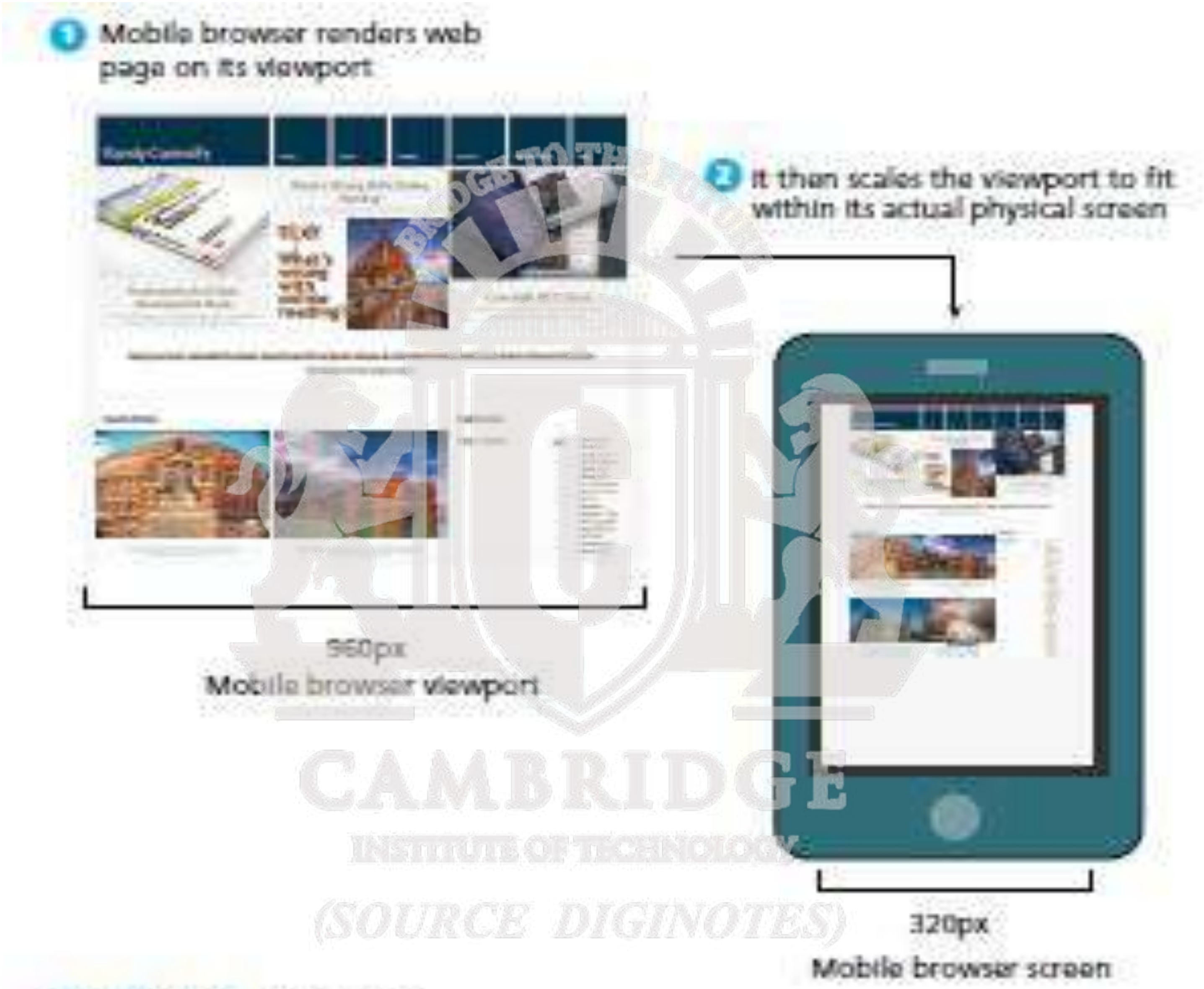

1 Mobile browser renders web page on its viewport
2 It then scales the viewport to fit within its actual physical screen
960px
Mobile browser viewport
320px
Mobile browser screen

FIGURE 5.31 Viewports

- The mobile safari browser introduced the viewport meta tag as a way for developers to control the size of that initial viewport.
- The web page will tell the mobile browser the viewport size to use via the viewport <meta> element, as below

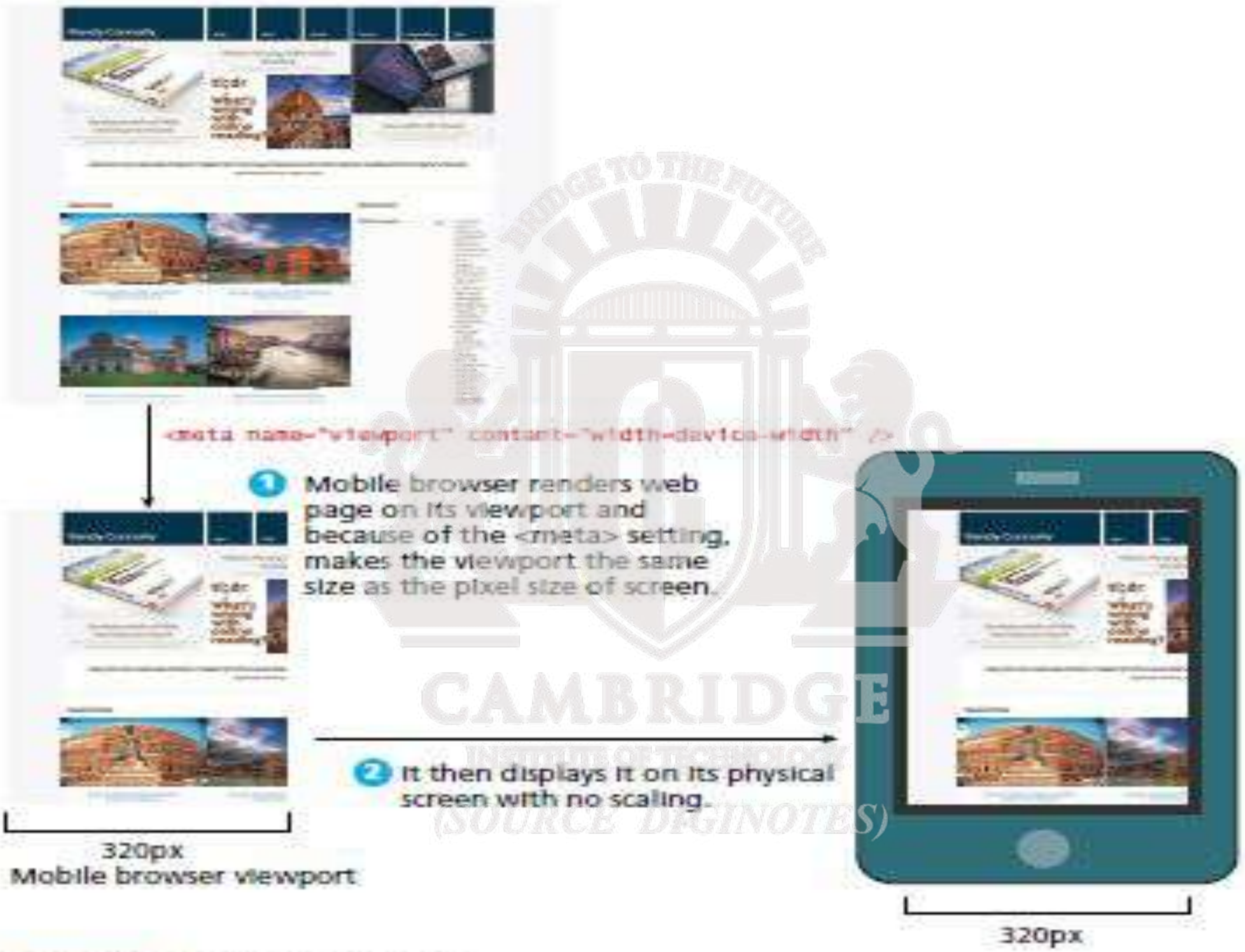
```
<html>
<head>
<meta name=“viewport” content=“width-device-width”/>
```

(Setting the viewport)

- By above, the page is telling the browser there is no need of any scaling and it makes the viewport as many pixels wide as device screen width.

  Ex- if device has a screen that is 320 px wide, then the viewport width will be 320 px,

  if its 480px wide, then viewport will be 480



<meta name="viewport" content="width=device-width" />

1 Mobile browser renders web page on its viewport and because of the <meta> setting, makes the viewport the same size as the pixel size of screen.

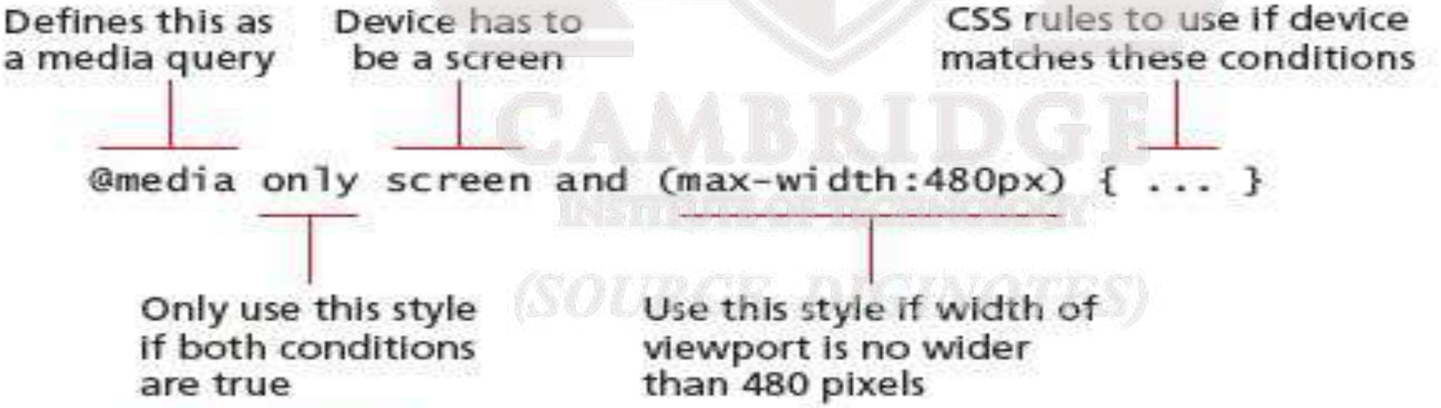
2 It then displays it on its physical screen with no scaling.

FIGURE 5.32 Setting the viewport

- However, since only setting the viewport as in above figure- shrank but still cropped the content.
- Setting the viewport is only one step in creating a responsive design.
- There needs to be a way to transform the look of the site for the smaller screen of the mobile device, which can be done by using media queries.

5.6.2 Media Queries

- The other key component of responsive design is CSS media queries.
- A media query is a way to apply style rules based on the medium that is displaying the file.
- Syntax of media queries.



Defines this as a media query

Device has to be a screen

CSS rules to use if device matches these conditions

```
@media only screen and (max-width:480px) { ... }
```

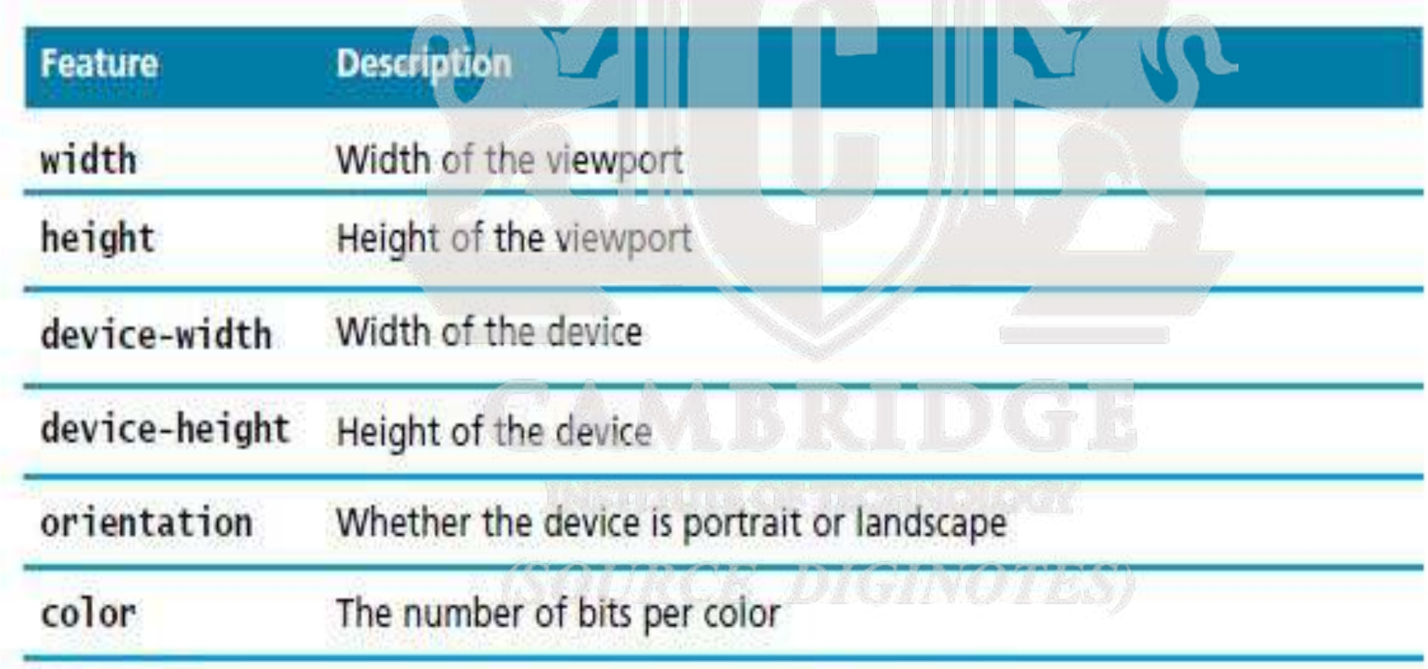
Only use this style if both conditions are true

Use this style if width of viewport is no wider than 480 pixels

FIGURE 5.33 Sample media query

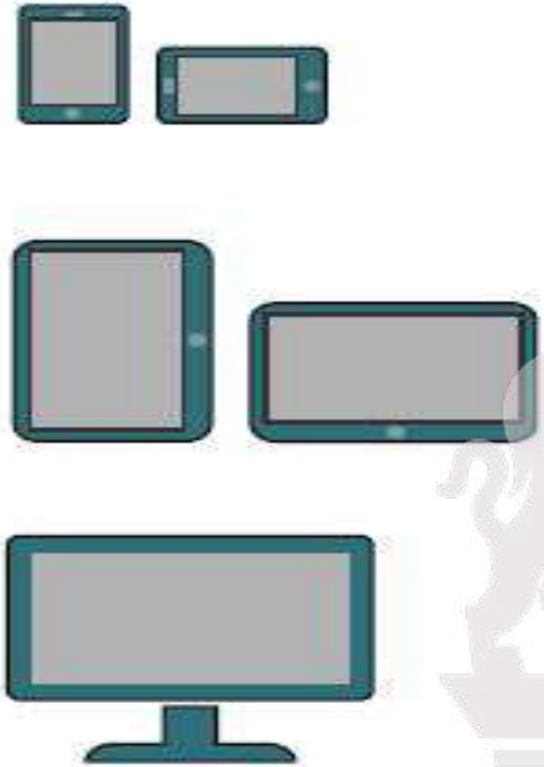
- These queries are Boolean expressions and can be added to your css file or to the link element to conditionally use a different external CSS file based .

- Below table shows the list of features you can use with media queries.



Feature	Description
width	Width of the viewport
height	Height of the viewport
device-width	Width of the device
device-height	Height of the device
orientation	Whether the device is portrait or landscape
color	The number of bits per color

TABLE 5.3 Browser Features You Can Examine with Media Queries



styles.css

```
/* rules for phones */
@media only screen and (max-width:480px)
{
  #slider-image { max-width: 100%; }
  #flash-ad { display: none; }
    ...
}


/* CSS rules for tablets */
@media only screen and (min-width: 481px)
      and (max-width: 768px)
{
    ...
}


/* CSS rules for desktops */
@media only screen and (min-width: 769px)
{
    ...
}
```

Instead of having all the rules in a single file, we can put them in separate files and add media queries to <link> elements.

```
<link rel="stylesheet" href="mobile.css"  media="screen and (max-width:480px)" />
<link rel="stylesheet" href="tablet.css"  media="screen and (min-width:481px)
    and (max-width:768px)" />
<link rel="stylesheet" href="desktop.css" media="screen and (min-width:769px)" />

<!--[if lt IE 9]>
<link rel="stylesheet" media="all" href="style-ie.css"/>
<![endif]-->
```

Handles Internet Explorer 8 and earlier using IE conditional comments.

FIGURE 5.34 Media queries in action

5.7 CSS FRAMEWORK

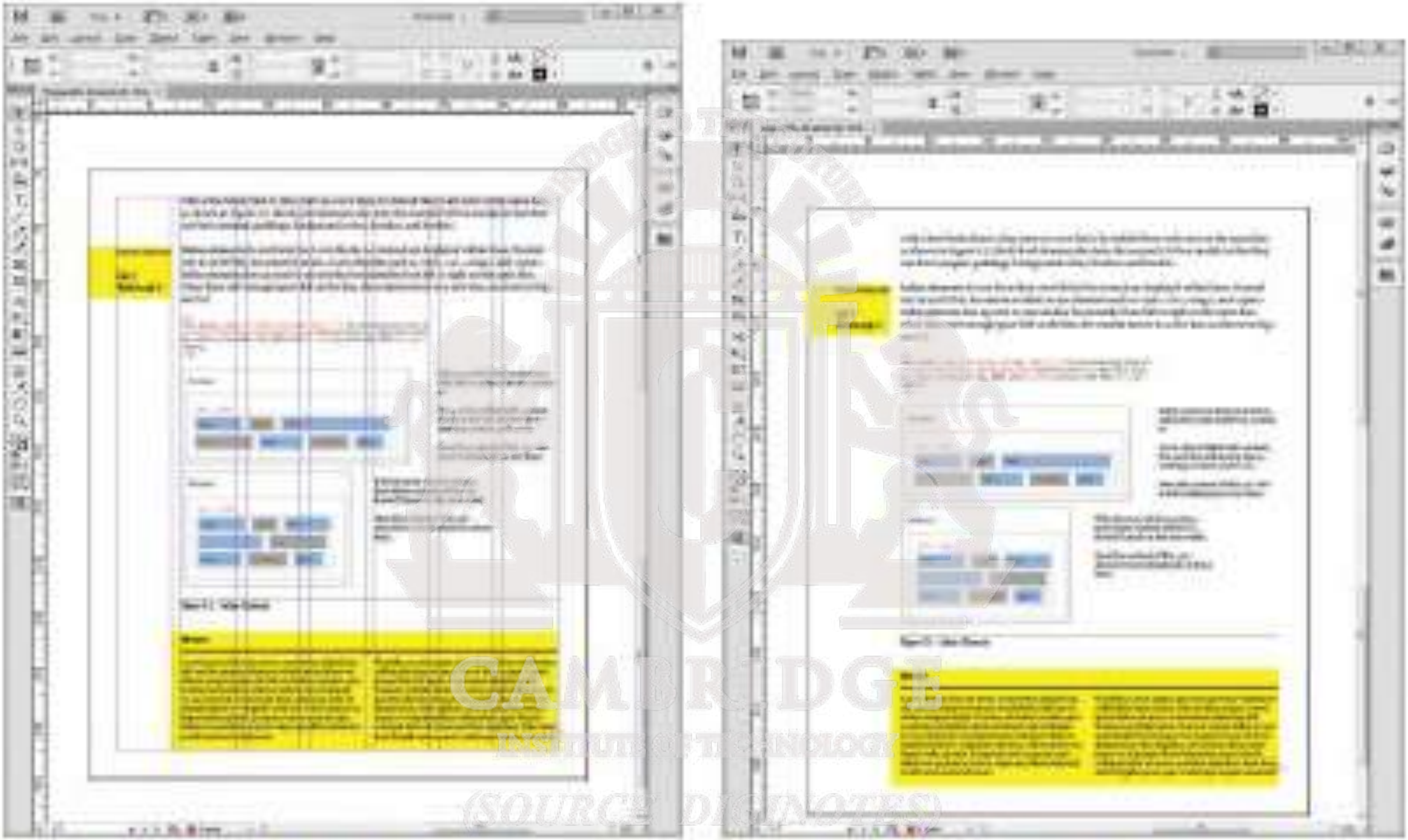
- A **CSS framework** is a precreated set of CSS classes or other software tools that make it easier to use and work with CSS.
- They are two main types of CSS framework:

1. Grid systems
2. CSS preprocessors.

Grid systems

- **Grid systems** make it easier to create multicolumn layouts.
- There are many CSS grid systems; some of the most popular are Bootstrap **twitter.github.com /bootstrap**),Blueprint (**www.blueprintcss.org**), and 960 (**960.gs**).
- The most important of these capabilities is a grid system.

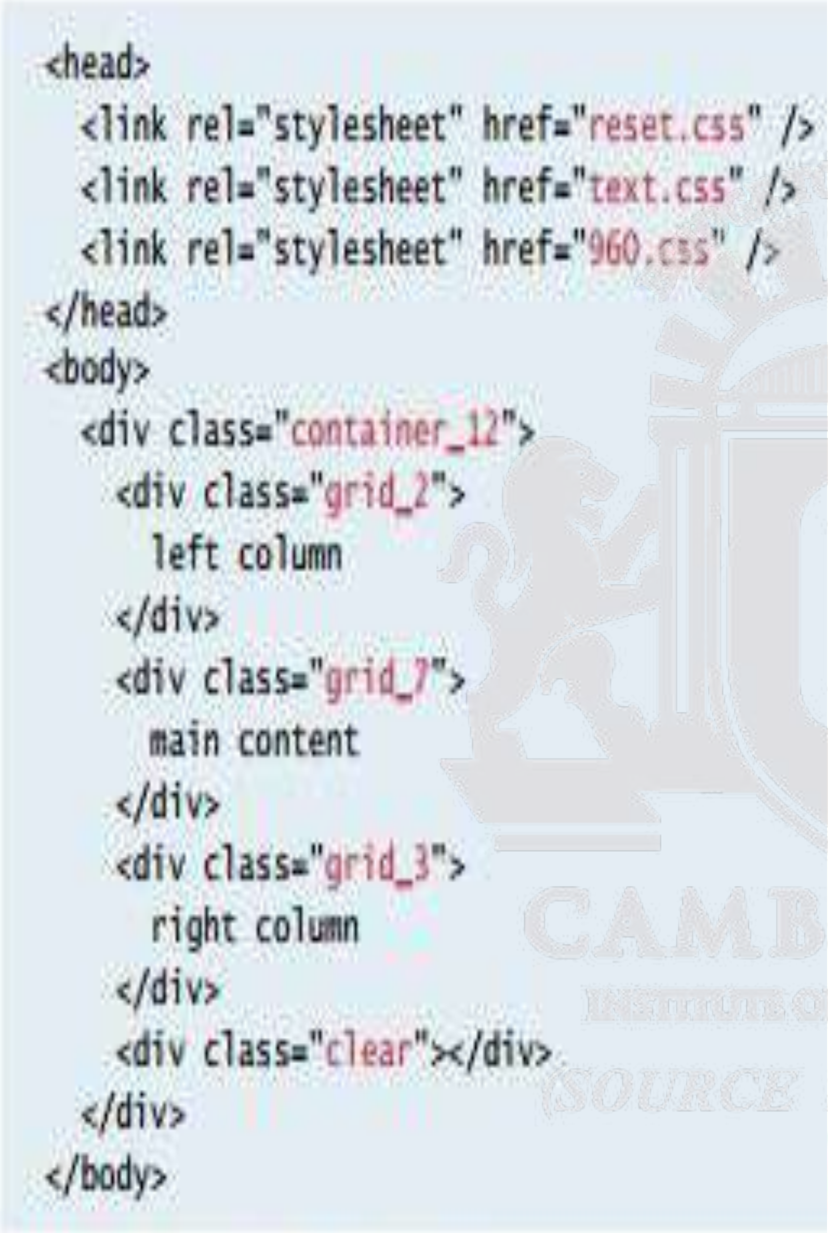
- CSS frameworks provide similar grid features. The 960 framework uses either a 12- or 16-column grid.
- Bootstrap uses a 12-column grid.
- Blueprint uses a 24-column grid.
- The grid is constructed using <div> elements with classes defined by the framework.
- The HTML elements for the rest of your site are then placed within these <div> elements.



Most page design begins with a grid. In this case, a seven-column grid is being used to layout page elements in Adobe InDesign.

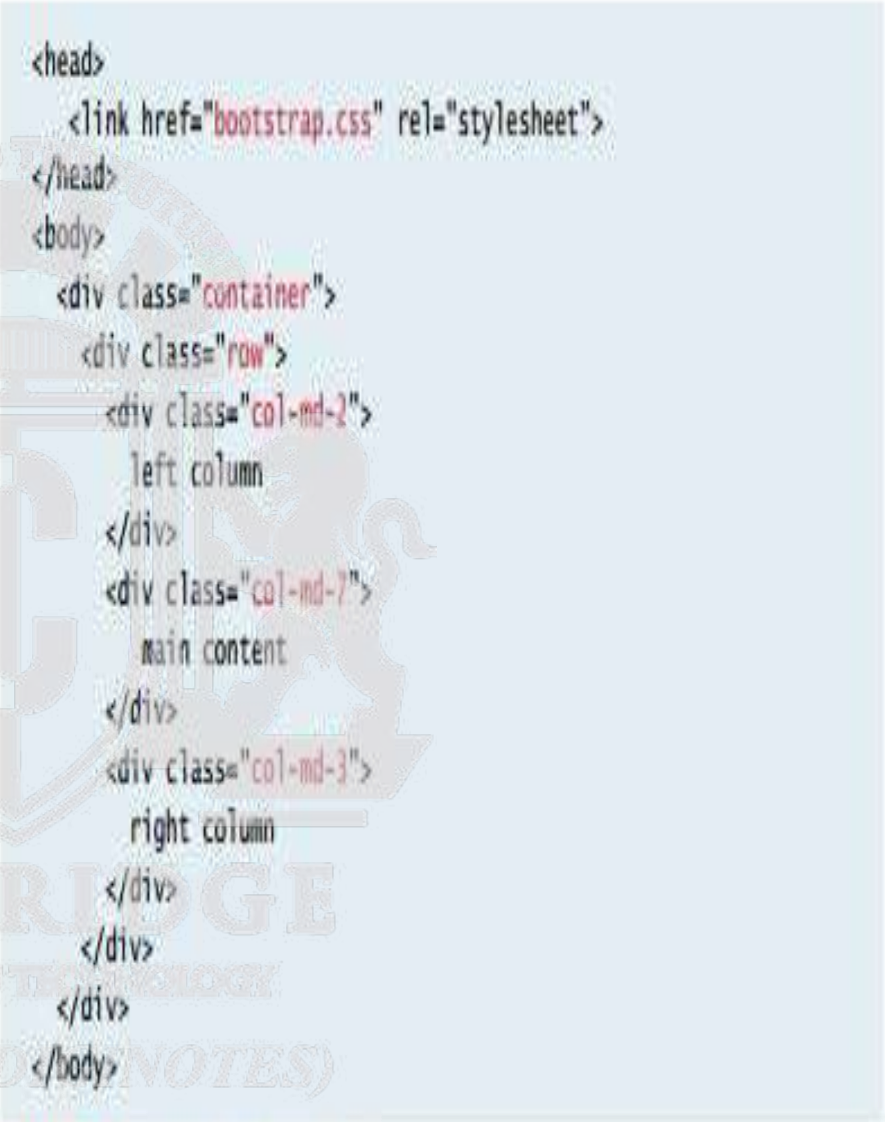
Without the gridlines visible, the elements on the page do not look random, but planned and harmonious.

FIGURE 5.35 Using a grid in print design



```
<head>
  <link rel="stylesheet" href="reset.css" />
  <link rel="stylesheet" href="text.css" />
  <link rel="stylesheet" href="960.css" />
</head>
<body>
  <div class="container_12">
    <div class="grid_2">
      left column
    </div>
    <div class="grid_7">
      main content
    </div>
    <div class="grid_3">
      right column
    </div>
    <div class="clear"></div>
  </div>
</body>
```

LISTING 5.2 Using the 960 grid



```
<head>
  <link href="bootstrap.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-md-2">
        left column
      </div>
      <div class="col-md-7">
        main content
      </div>
      <div class="col-md-3">
        right column
      </div>
    </div>
  </div>
</body>
```

LISTING 5.3 Using the Bootstrap grid

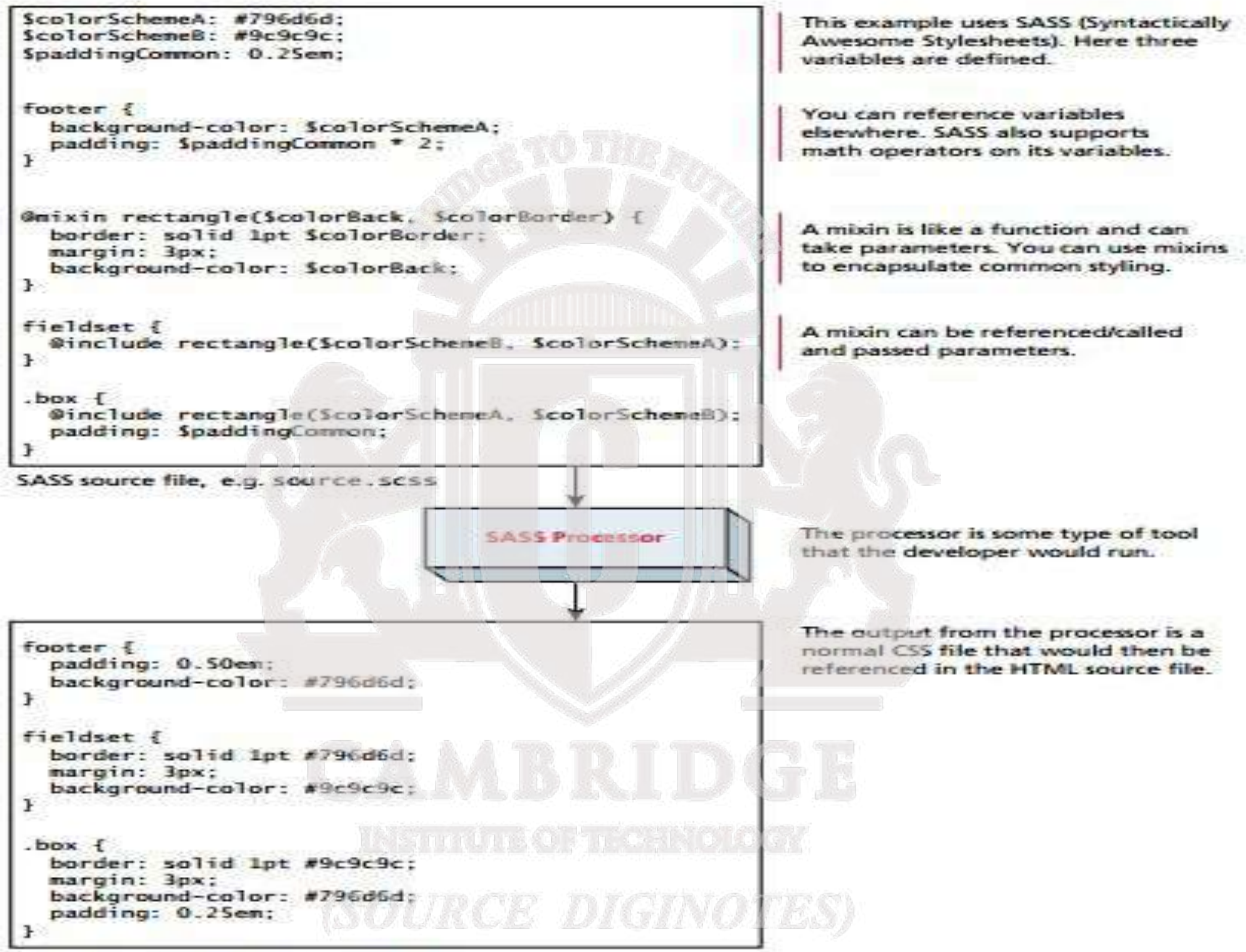
We will be using the Bootstrap framework, which is an open-source system, but was originally created by the designers at Twitter

FIGURE 5.36 Examples using just built-in Bootstrap classes

Css preprocessor

- **CSS preprocessors** are tools that allow the developer to write CSS that takes advantage of programming ideas such as variables, inheritance, calculations, and functions.
- A CSS preprocessor is a tool that takes code written in some type of preprocessed language and then converts that code into normal CSS
- The advantage of a CSS preprocessor is that it can provide additional functionalities that are not available in CSS.

- In a programming language, a developer can use variables, nesting, functions, or inheritance to handle duplication and avoid copy-and-pasting and search-and-replacing.
- CSS preprocessors such as LESS, SASS, and Stylus provide this type of functionality.



```
$colorSchemeA: #796d6d;
$colorSchemeB: #9c9c9c;
$paddingCommon: 0.25em;


footer {
  background-color: $colorSchemeA;
  padding: $paddingCommon * 2;
}


@mixin rectangle($colorBack, $colorBorder) {
  border: solid 1pt $colorBorder;
  margin: 3px;
  background-color: $colorBack;
}

fieldset {
  @include rectangle($colorSchemeB, $colorSchemeA);
}

.box {
  @include rectangle($colorSchemeA, $colorSchemeB);
  padding: $paddingCommon;
}
```

SASS source file, e.g. source.scss

This example uses SASS (Syntactically Awesome Stylesheets). Here three variables are defined.

You can reference variables elsewhere. SASS also supports math operators on its variables.

A mixin is like a function and can take parameters. You can use mixins to encapsulate common styling.

A mixin can be referenced/called and passed parameters.

The processor is some type of tool that the developer would run.

The output from the processor is a normal CSS file that would then be referenced in the HTML source file.

```
footer {
  padding: 0.50em;
  background-color: #796d6d;
}

fieldset {
  border: solid 1pt #796d6d;
  margin: 3px;
  background-color: #9c9c9c;
}

.box {
  border: solid 1pt #9c9c9c;
  margin: 3px;
  background-color: #796d6d;
  padding: 0.25em;
}
```

Generated CSS file, e.g., styles.css

FIGURE 5.37 Using a CSS preprocessor

MODULE 2 END

WEB TECHNOLOGY AND ITS APPLICATIONS

17CS71

Mr. GANESH D R
ASSISTANT PROFESSOR,
DEPT OF CSE, CITECH

WEB TECHNOLOGY AND ITS APPLICATIONS

[As per Choice Based Credit System (CBCS) scheme]

(Effective from the academic year 2017 - 2018)

SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	**10 Hours**
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	**10 Hours**
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	**10 Hours**
Module – 4	
PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays, $_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	**10 Hours**
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	**10 Hours**

MODULE 3 - SYLLABUS

- JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server’s Responsibilities, Quick Tour of PHP, Program Control, Functions

JavaScript: Client-Side Scripting

Chapter 6

Section 1 of 8

WHAT IS JAVASCRIPT

What is JavaScript

- JavaScript runs right inside the browser
- JavaScript is dynamically typed (Weakly Typed)
- JavaScript is object oriented in that almost everything in the language is an object
- The objects in JavaScript are prototype-based rather than class-based, which means that while JavaScript shares some syntactic features of PHP, Java or C#, it is also quite different from those languages

What isn’t JavaScript

It’s not Java

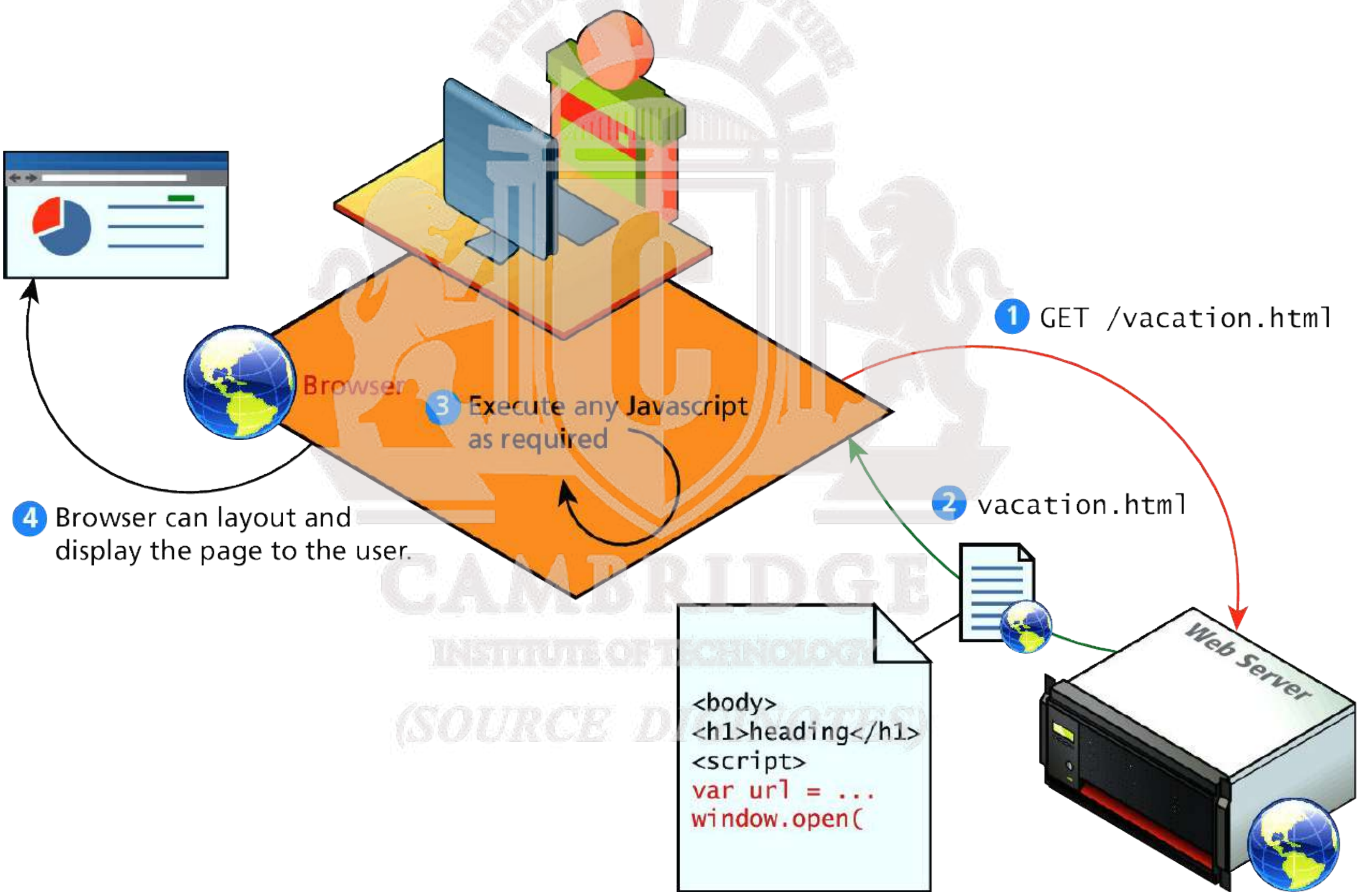
❖Although it contains the word *Java*, JavaScript and Java are vastly different programming languages with different uses.

❖Java is a full-fledged compiled, object-oriented language, popular for its ability to run on any platform with a JVM installed.

❖Conversely, JavaScript is one of the world’s most popular languages, with fewer of the object-oriented features of Java, and runs directly inside the browser, without the need for the JVM.

Client-Side Scripting

Let the client compute



1 GET /vacation.html

2 vacation.html

3 Execute any Javascript as required

4 Browser can layout and display the page to the user.

```
<body>
<h1>heading</h1>
<script>
var url = ...
window.open(
```

Source diginotes.in

Save the Earth. Go paperless

Client-Side Scripting

It’s good

There are many **advantages** of client-side scripting:

- Processing can be offloaded from the server to client machines, thereby reducing the load on the server.
- The browser can respond more rapidly to user events than a request to a remote server ever could, which improves the user experience.
- JavaScript can interact with the downloaded HTML in a way that the server cannot, creating a user experience more like desktop software than simple HTML ever could.

Client-Side Scripting

There are challenges

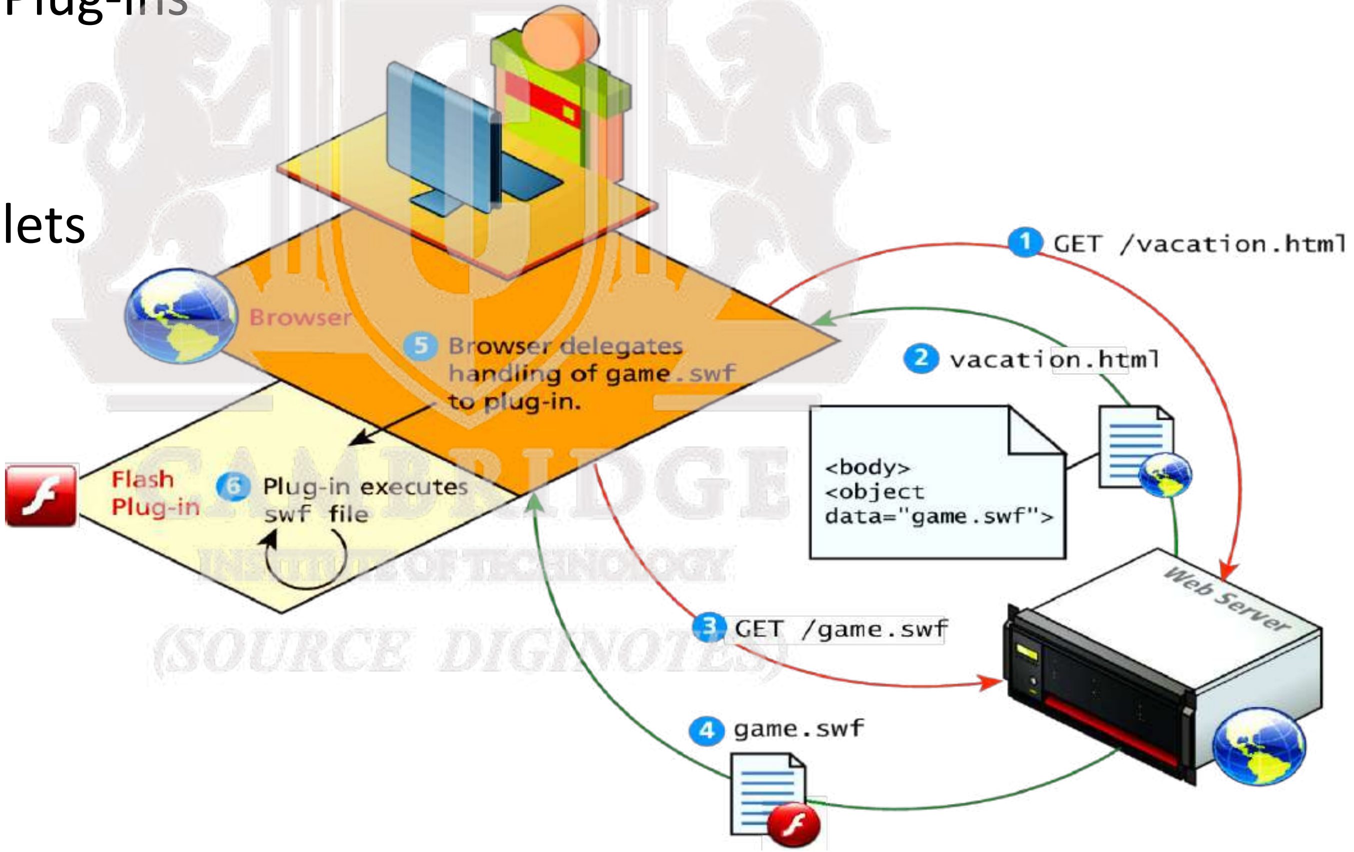
The disadvantages of client-side scripting are mostly related to how programmers use JavaScript in their applications.

- There is no guarantee that the client has JavaScript enabled
- The idiosyncrasies between various browsers and operating systems make it difficult to test for all potential client configurations. What works in one browser, may generate an error in another.
- JavaScript-heavy web applications can be complicated to debug and maintain.

Client-Side Flash

JavaScript is not the only type of client-side scripting.

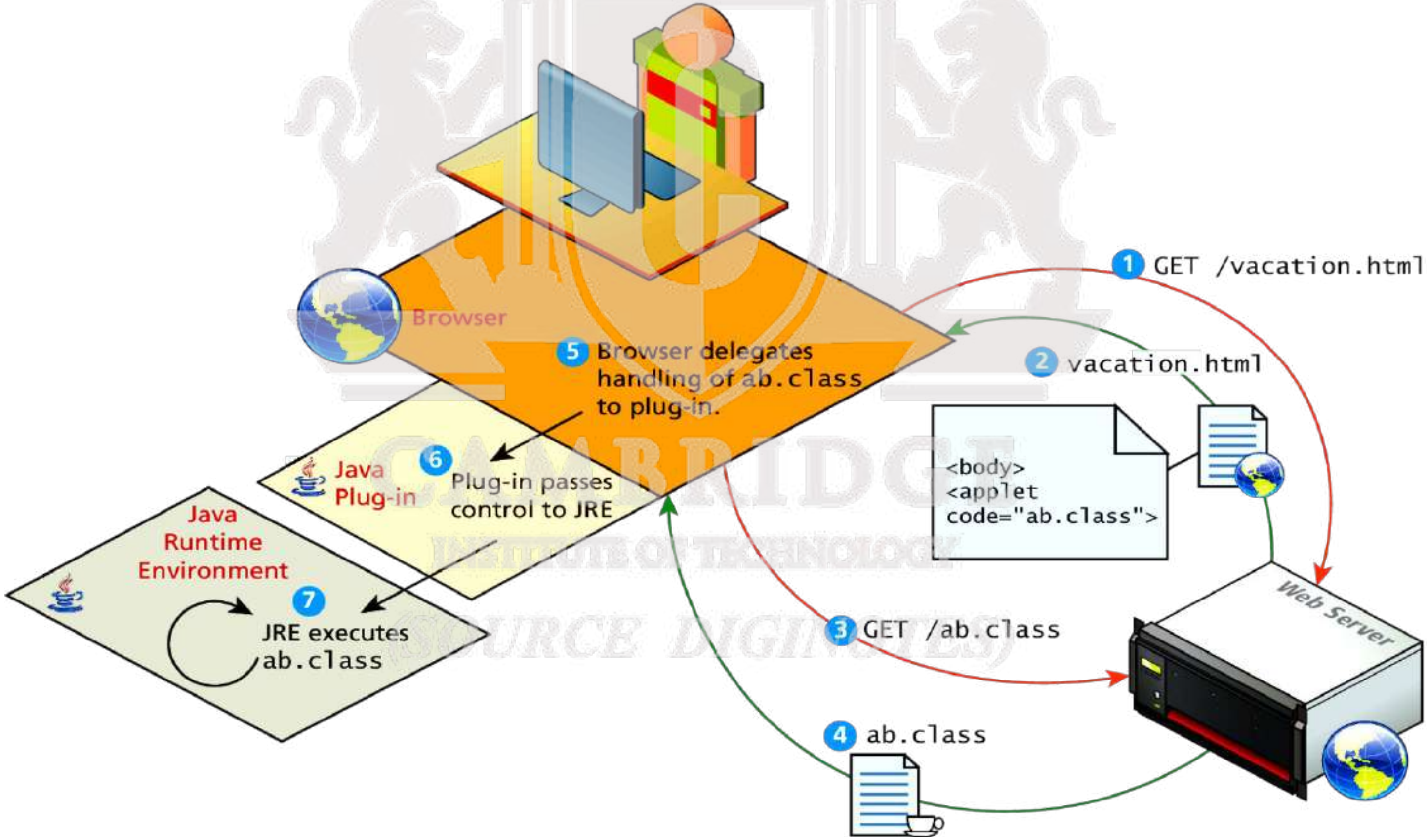
- Browser Plug-ins
- Flash
- Java Applets



Client-Side Applets

Java Applets

Java applets are written in and are separate objects included within an HTML document via the <applet> tag



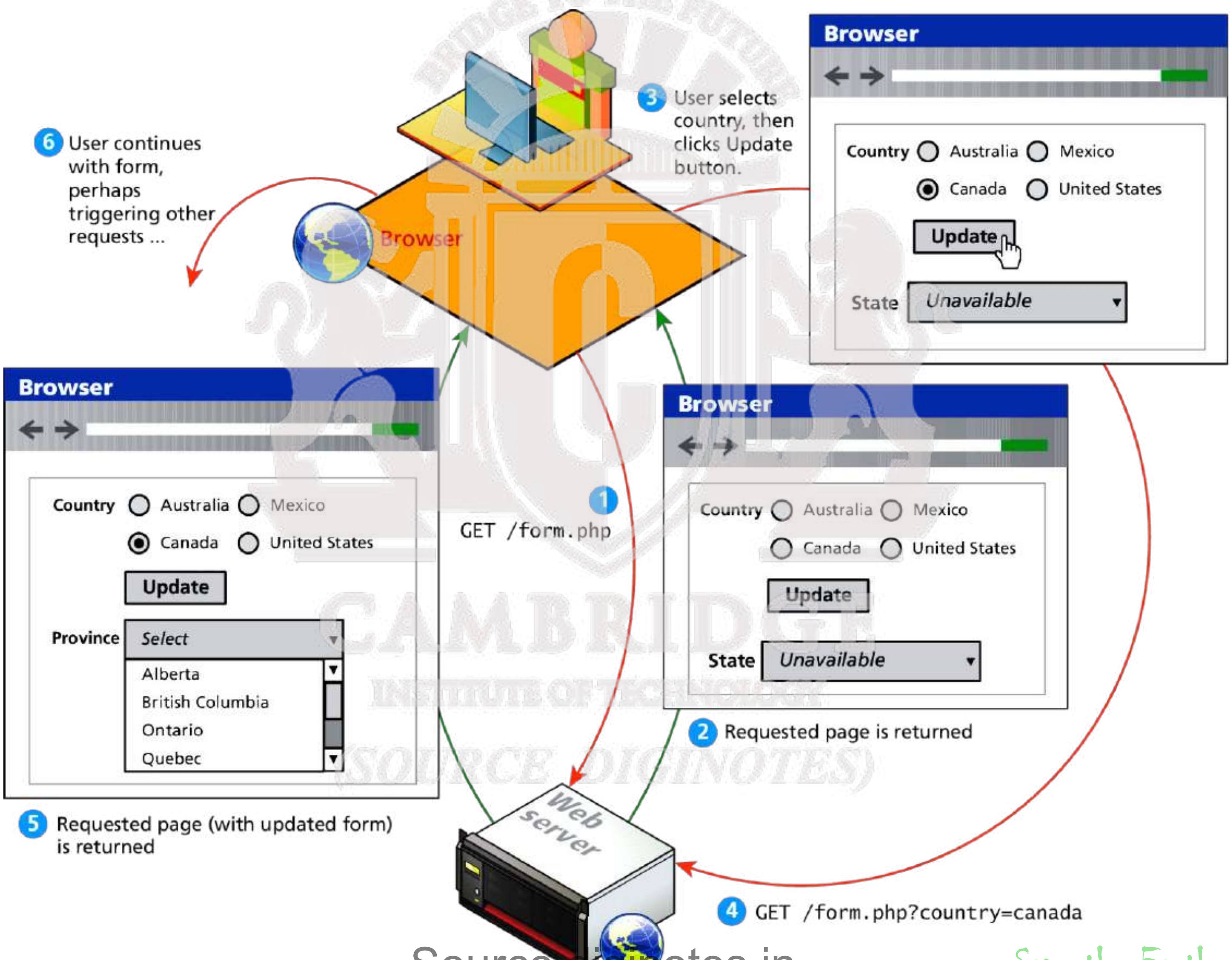
Source diginotes.in

JavaScript History

- JavaScript was introduced by Netscape in their Navigator browser back in 1996.
- It was originally called LiveScript
- JavaScript is in fact an implementation of a standardized scripting language called **ECMAScript**
- JavaScript was only slightly useful, and quite often, very annoying to many users

HTTP request-response loop

Without JavaScript



1 GET /form.php

2 Requested page is returned

3 User selects country, then clicks Update button.

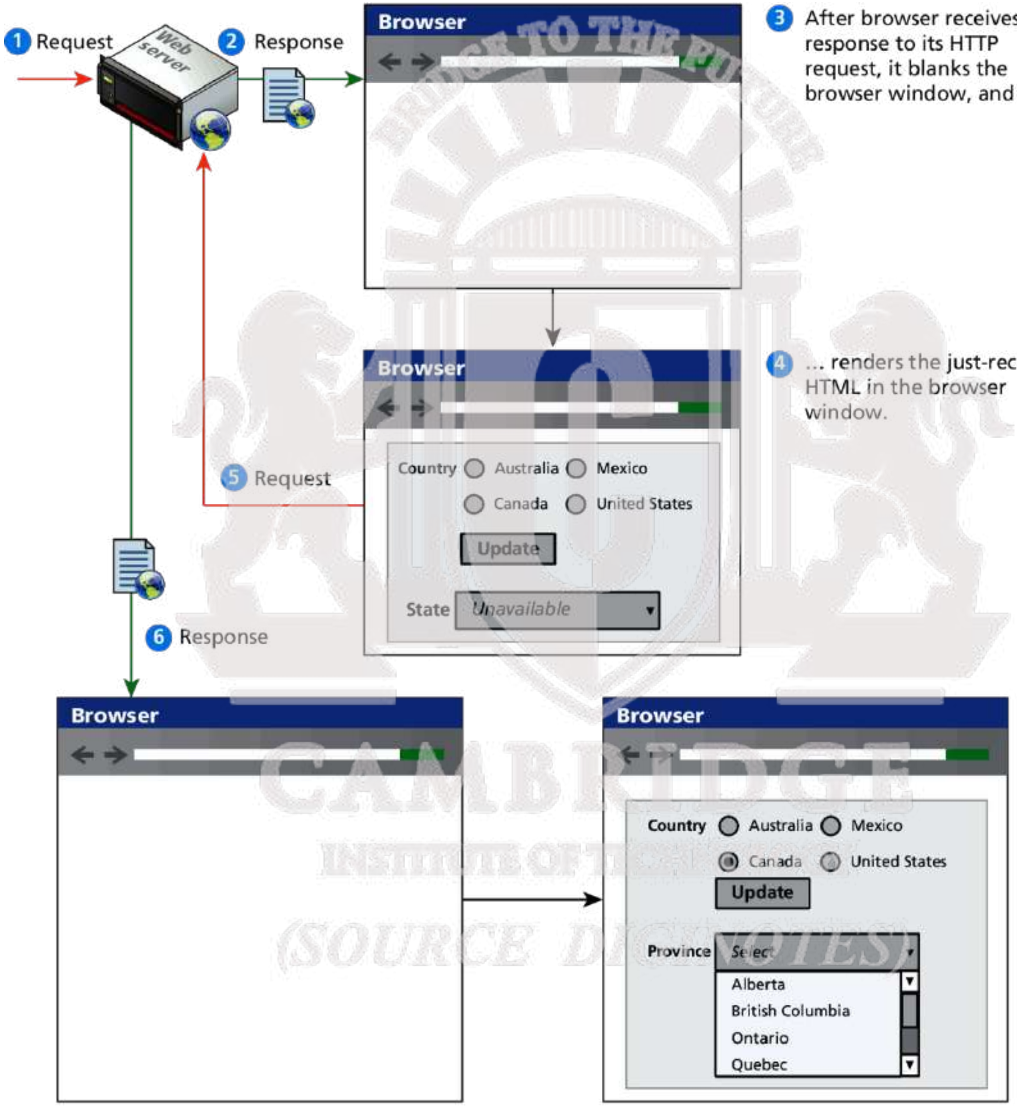
4 GET /form.php?country=canada

5 Requested page (with updated form) is returned

6 User continues with form, perhaps triggering other requests ...

HTTP request-response loop

Detail



3 After browser receives a response to its HTTP request, it blanks the browser window, and ...

4 ... renders the just-received HTML in the browser window.

7 Another new response has been received, so browser window is blanked and ...

8 ... renders the just-received HTML in the browser window.

JavaScript in Modern Times

AJAX

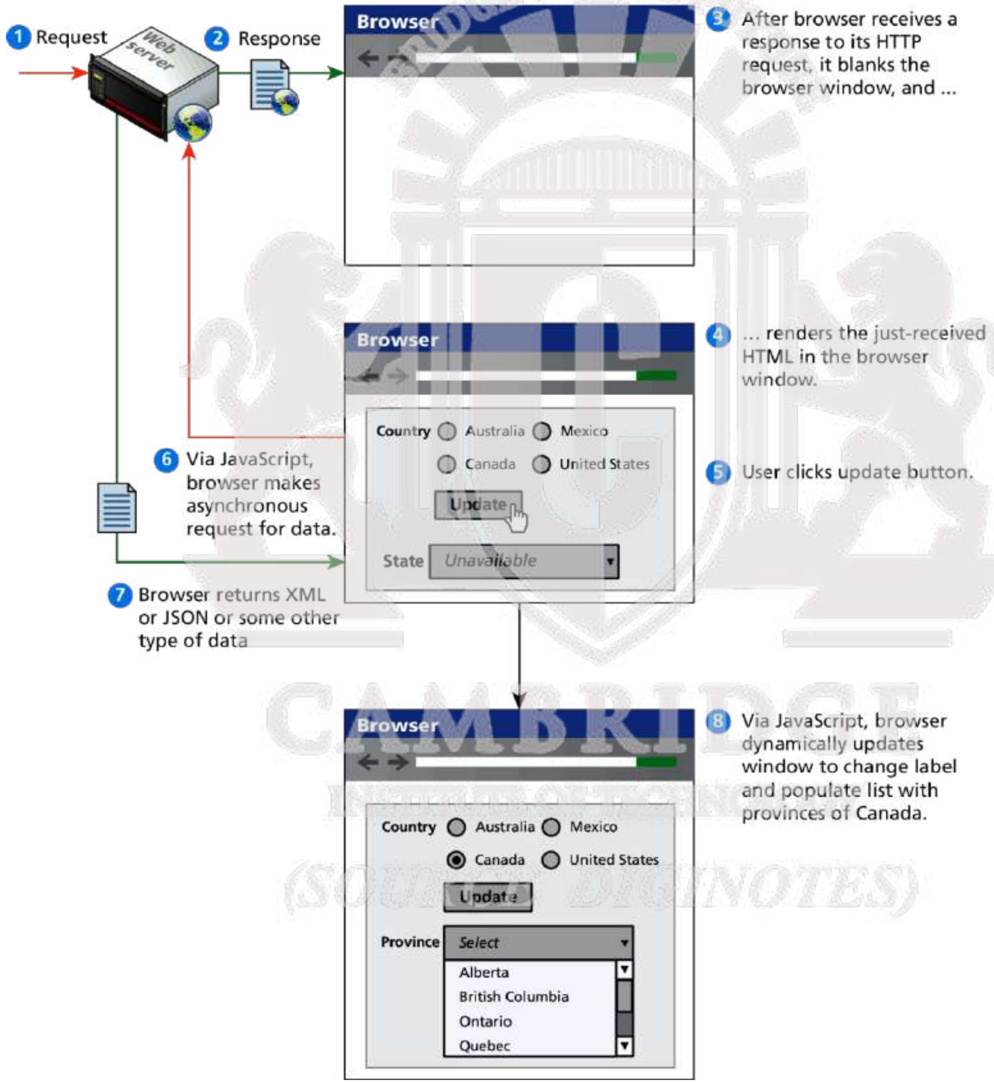
JavaScript became a much more important part of web development in the mid 2000s with **AJAX**.

AJAX is both an acronym as well as a general term.

- As an acronym it means **A**synchronous **J**avaScript **A**nd **X**ML.
- The most important feature of AJAX sites is the asynchronous data requests.

Asynchronous data requests

The better AJAX way



1 Request

2 Response

3 After browser receives a response to its HTTP request, it blanks the browser window, and ...

4 ... renders the just-received HTML in the browser window.

5 User clicks update button.

6 Via JavaScript, browser makes asynchronous request for data.

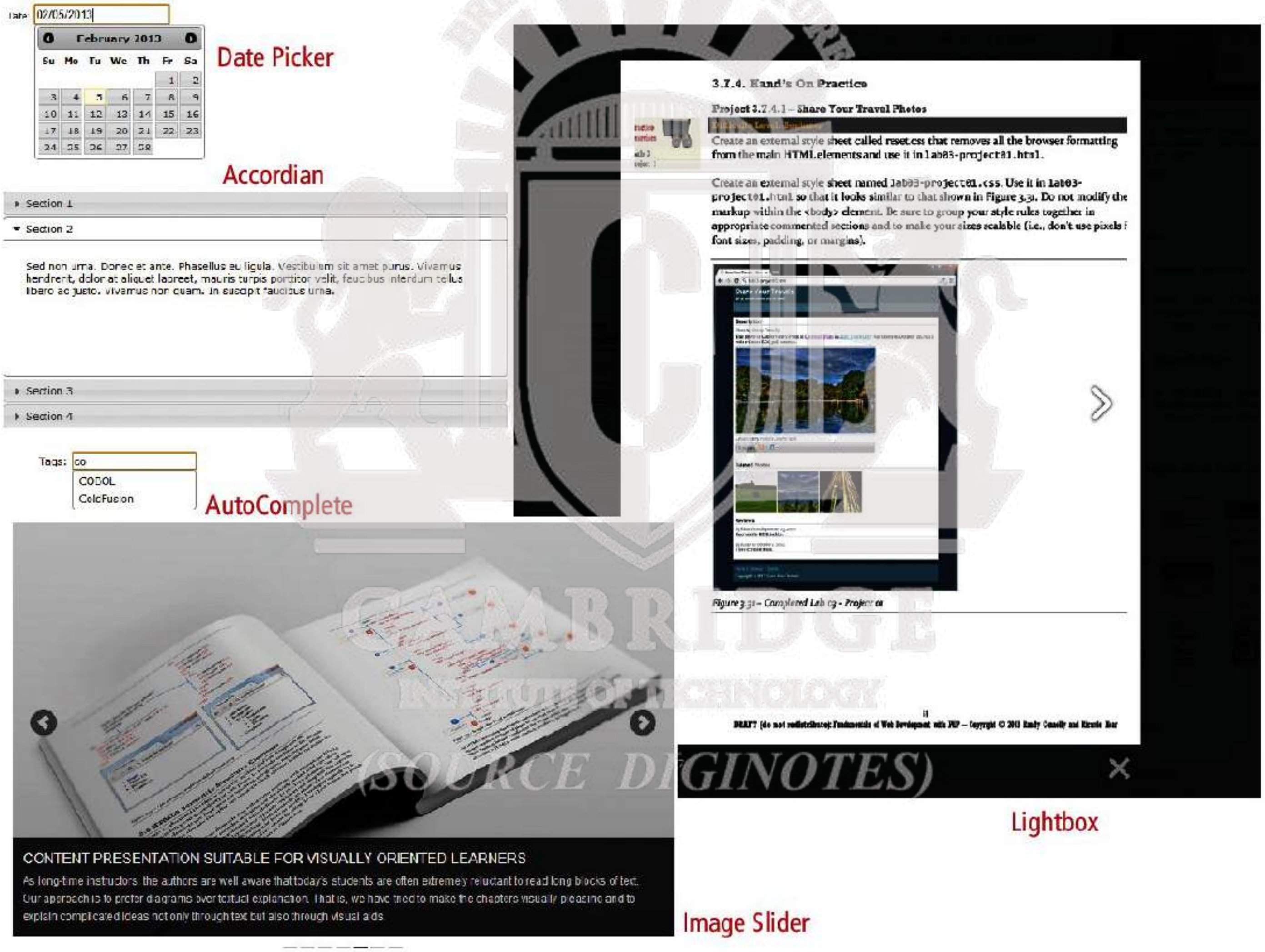
7 Browser returns XML or JSON or some other type of data

8 Via JavaScript, browser dynamically updates window to change label and populate list with provinces of Canada.

Source diginotes.in

Frameworks

Lots of this is done for you, once you get the basics



Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum sit amet purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turpis porttitor velit, faucibus interdum tellus libero ac justo. Vivamus non quam. In suscipit faucibus urna.

3.7.4. Hand's On Practice

Project 3.7.4.1 – Share Your Travel Photos

Difficulty Level: Beginner

Create an external style sheet called reset.css that removes all the browser formatting from the main HTML elements and use it in lab03-project01.html.

Create an external style sheet named lab03-project01.css. Use it in lab03-project01.html so that it looks similar to that shown in Figure 3.31. Do not modify the markup within the <body> element. Be sure to group your style rules together in appropriate commented sections and to make your sizes scalable (i.e., don't use pixels f font sizes, padding, or margins).

Figure 3.31 – Completed Lab 03 - Project 01

DRAFT (do not redistribute) Fundamentals of Web Development with PHP – Copyright © 2013 Randy Connolly and Ricardo Hoar

CONTENT PRESENTATION SUITABLE FOR VISUALLY ORIENTED LEARNERS

As long-time instructors the authors are well aware that today's students are often extremely reluctant to read long blocks of text. Our approach is to prefer diagrams over textual explanation. That is, we have tried to make the chapters visually pleasing and to explain complicated ideas not only through text but also through visual aids.

Section 2 of 8

JAVASCRIPT DESIGN PRINCIPLES

Layers

They help organize

- When designing software to solve a problem, it is often helpful to abstract the solution a little bit to help build a cognitive model in your mind that you can then implement.
- Perhaps the most common way of articulating such a cognitive model is via the term **layer**.
- In object-oriented programming, a software **layer** is a way of conceptually grouping programming classes that have similar functionality and dependencies.

Layers

Common Layers

- **Presentation layer.**

  Classes focused on the user interface.

- **Business layer.**

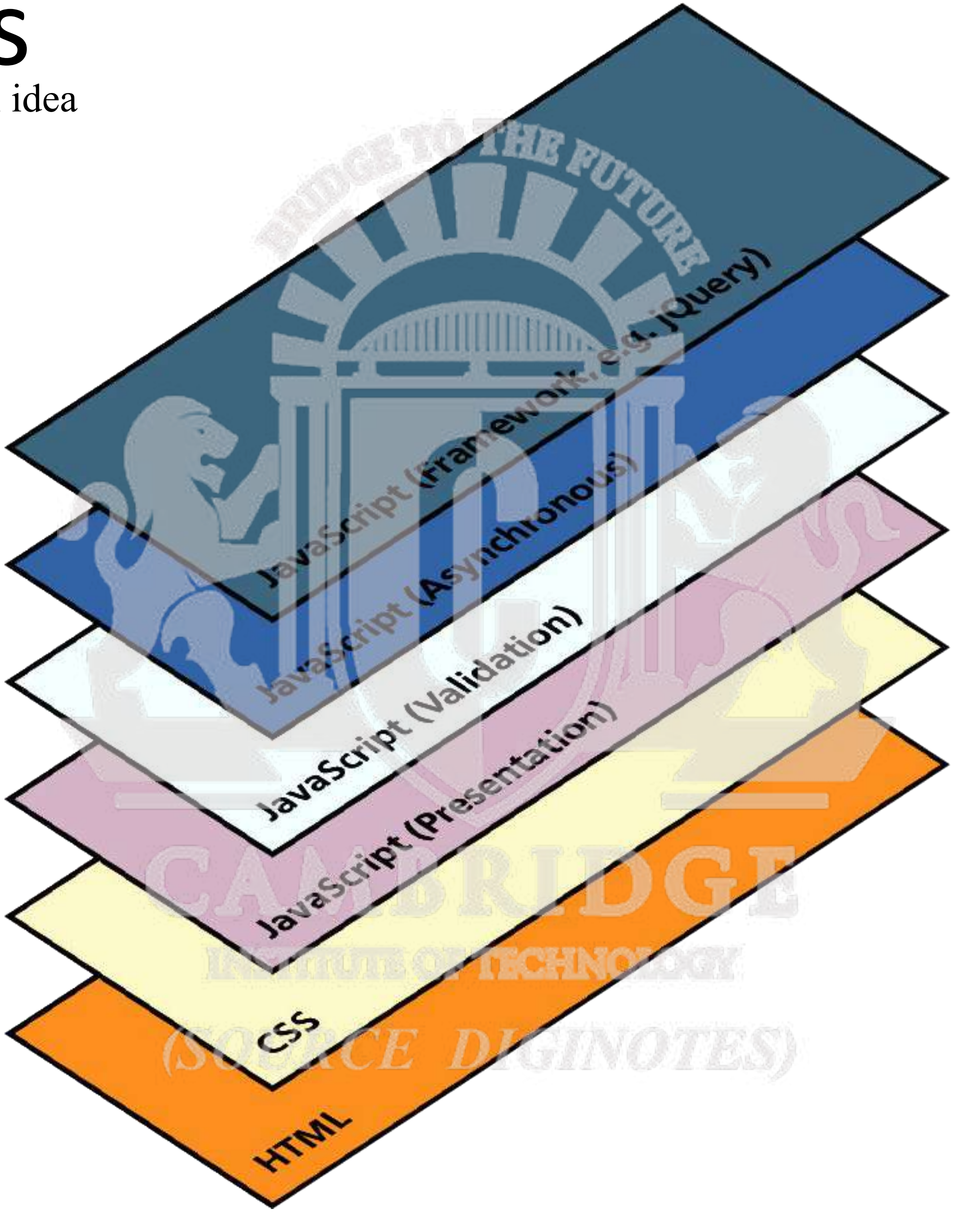
  Classes that model real-world entities, such as customers, products, and sales.

- **Data layer.**

  Classes that handle the interaction with the data sources.

Layers

Just a conceptual idea


JavaScript (framework, e.g. jQuery)
JavaScript (Asynchronous)
JavaScript (Validation)
JavaScript (Presentation)
CSS
HTML

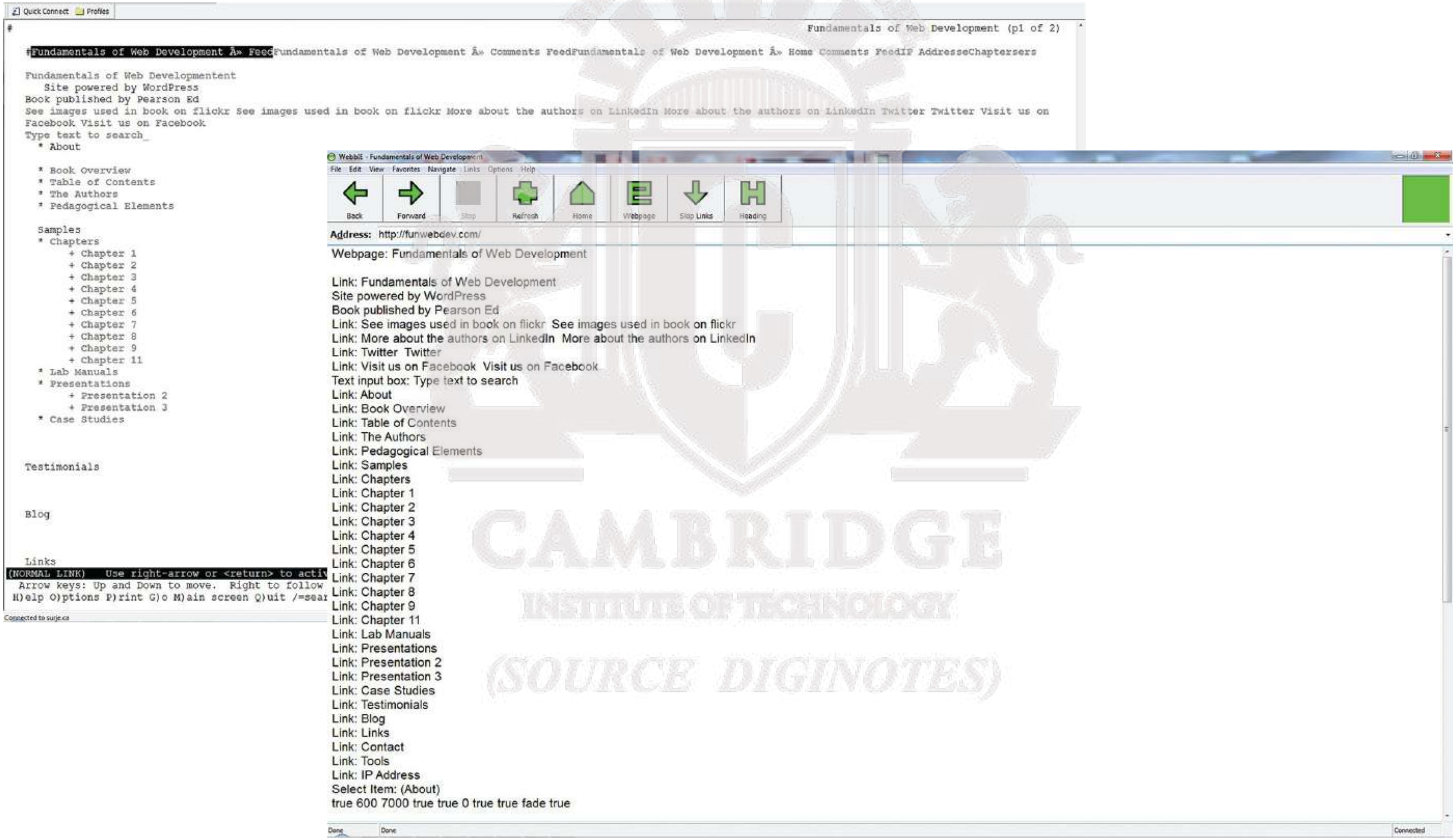
Users Without Javascript

They do exist

- **Web crawler**. A web crawler is a client running on behalf of a search engine to download your site, so that it can eventually be featured in their search results.
- **Browser plug-in**. A browser plug-in is a piece of software that works within the browser, that might interfere with JavaScript.
- **Text-based client**. Some clients are using a text-based browser.
- **Visually disabled client**. A visually disabled client will use special web browsing software to read the contents of a web page out loud to them.

Users Without Javascript

Lynx, and WebIE



```
#                                                Fundamentals of Web Development (p1 of 2)

  #Fundamentals of Web Development Â» FeedFundamentals of Web Development Â» Comments FeedFundamentals of Web Development Â» Home Comments FeedIP AddresseChaptersers

  Fundamentals of Web Developmentent
     Site powered by WordPress
  Book published by Pearson Ed
  See images used in book on flickr See images used in book on flickr More about the authors on LinkedIn More about the authors on LinkedIn Twitter Twitter Visit us on
  Facebook Visit us on Facebook
  Type text to search_
     * About

     * Book Overview
     * Table of Contents
     * The Authors
     * Pedagogical Elements

    Samples
     * Chapters
          + Chapter 1
          + Chapter 2
          + Chapter 3
          + Chapter 4
          + Chapter 5
          + Chapter 6
          + Chapter 7
          + Chapter 8
          + Chapter 9
          + Chapter 11
     * Lab Manuals
     * Presentations
          + Presentation 2
          + Presentation 3
     * Case Studies



  Testimonials



  Blog



  Links
(NORMAL LINK)   Use right-arrow or <return> to activ
  Arrow keys: Up and Down to move.  Right to follow
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=sear
```

Connected to surje.ca

Address: http://funwebdev.com/

Webpage: Fundamentals of Web Development

Link: Fundamentals of Web Development
Site powered by WordPress
Book published by Pearson Ed
Link: See images used in book on flickr See images used in book on flickr
Link: More about the authors on LinkedIn More about the authors on LinkedIn
Link: Twitter Twitter
Link: Visit us on Facebook Visit us on Facebook
Text input box: Type text to search
Link: About
Link: Book Overview
Link: Table of Contents
Link: The Authors
Link: Pedagogical Elements
Link: Samples
Link: Chapters
Link: Chapter 1
Link: Chapter 2
Link: Chapter 3
Link: Chapter 4
Link: Chapter 5
Link: Chapter 6
Link: Chapter 7
Link: Chapter 8
Link: Chapter 9
Link: Chapter 11
Link: Lab Manuals
Link: Presentations
Link: Presentation 2
Link: Presentation 3
Link: Case Studies
Link: Testimonials
Link: Blog
Link: Links
Link: Contact
Link: Tools
Link: IP Address
Select Item: (About)
true 600 7000 true true 0 true true fade true

Done Done Connected

Graceful Degradation and Progressive Enhancement

Over the years, browser support for different JavaScript objects has varied. Something that works in the current version of Chrome might not work in IE version 8; something that works in a desktop browser might not work in a mobile browser.

There are two strategies:

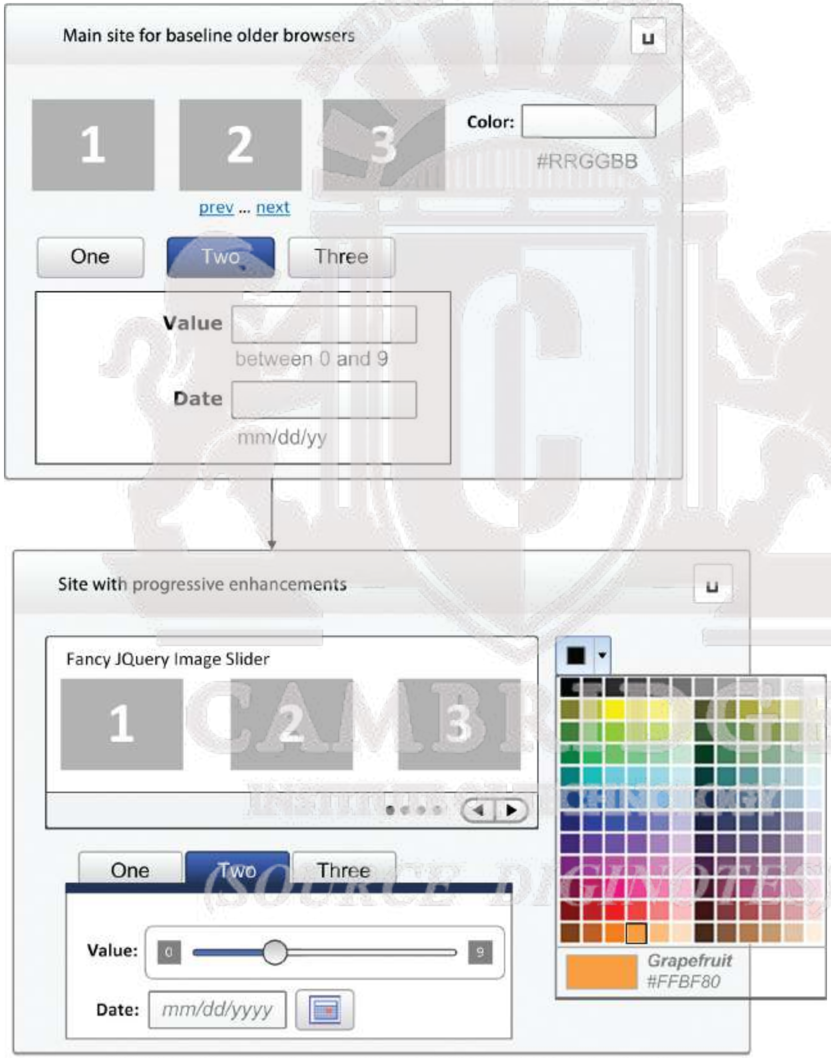
- **graceful degradation**
- **progressive enhancement**

Progressive Enhancement

- In this case, the developer creates the site using CSS, JavaScript, and HTML features that are supported by all browsers of a certain age or newer.

- To that baseline site, the developers can now “progressively” (i.e., for each browser) “enhance” (i.e., add functionality) to their site based on the capabilities of the users’ browsers.

Progressive Enhancement



Users with more current browsers will experience a progressively richer and enhanced user interface.

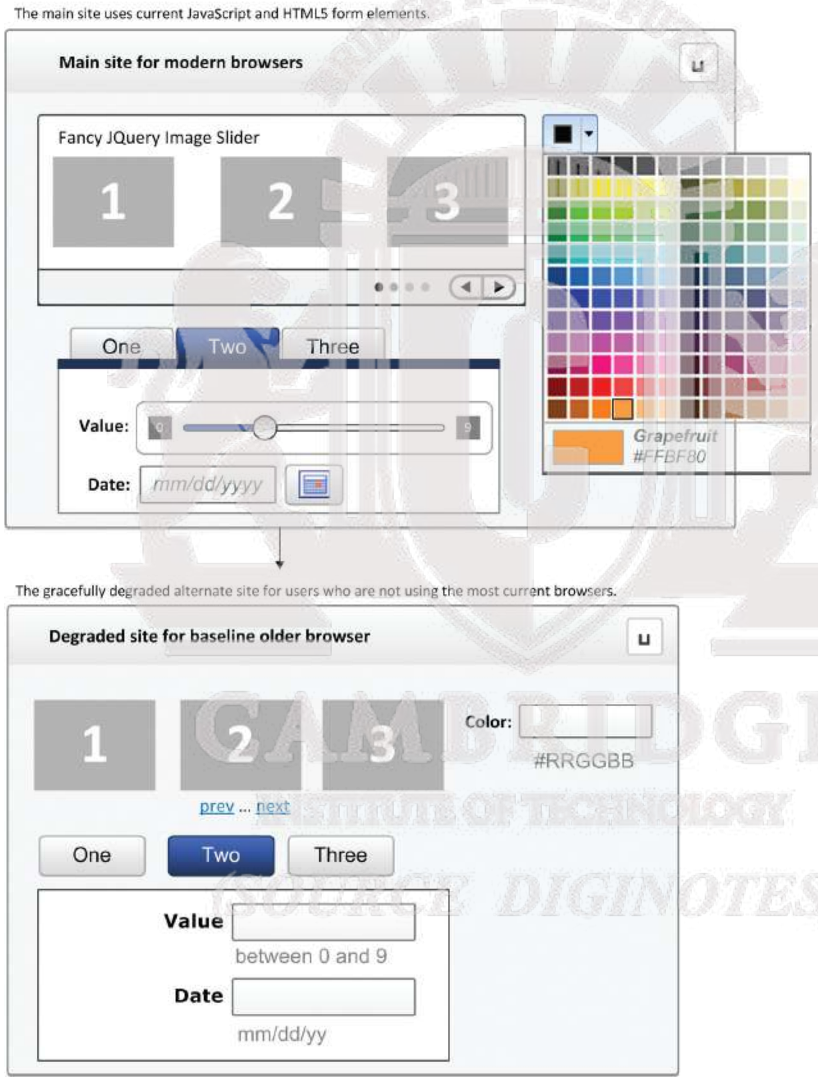
Source diginotes.in

Save the Earth. Go paperless

Graceful Degradation

- ❖With this strategy you develop your site for the abilities of current browsers.
- ❖For those users who are not using current browsers, you might provide an alternate site or pages for those using older browsers that lack the JavaScript (or CSS or HTML5) used on the main site.
- ❖The idea here is that the site is “degraded” (i.e., loses capability) “gracefully” (i.e., without pop-up JavaScript error codes or without condescending messages telling users to upgrade their browsers)

Graceful Degradation


The main site uses current JavaScript and HTML5 form elements.

The gracefully degraded alternate site for users who are not using the most current browsers.

Source diginotes.in

Section 3 of 8

WHERE DOES JAVASCRIPT GO?

Where does JavaScript go?

JavaScript can be linked to an HTML page in a number of ways.

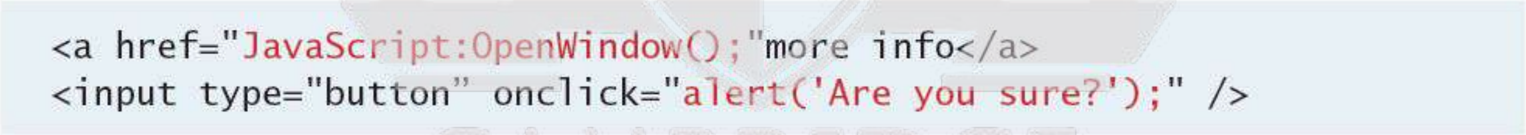
- Inline
- Embedded
- External

Inline JavaScript

Mash it in

Inline JavaScript refers to the practice of including JavaScript code directly within certain HTML attributes

Inline JavaScript is a real maintenance nightmare



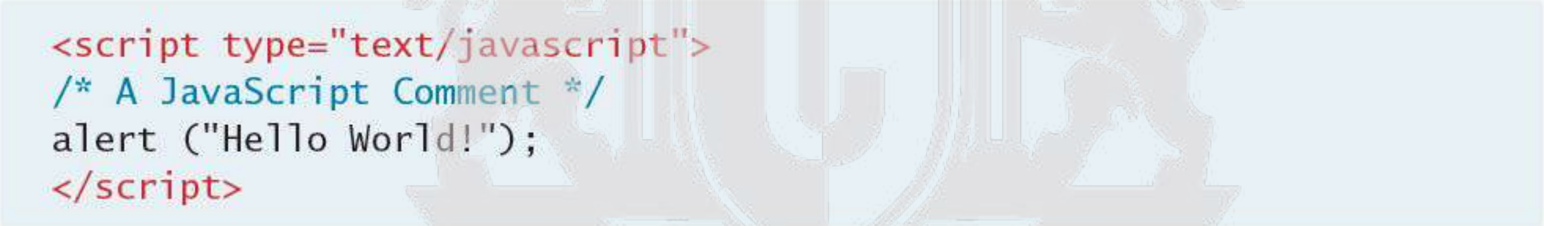
```
<a href="JavaScript:OpenWindow();"more info</a>
<input type="button” onclick="alert('Are you sure?');" />
```

LISTING 6.1 Inline JavaScript example

Embedded JavaScript

Better

Embedded JavaScript refers to the practice of placing JavaScript code within a <script> element



```
<script type="text/javascript">
/* A JavaScript Comment */
alert ("Hello World!");
</script>
```

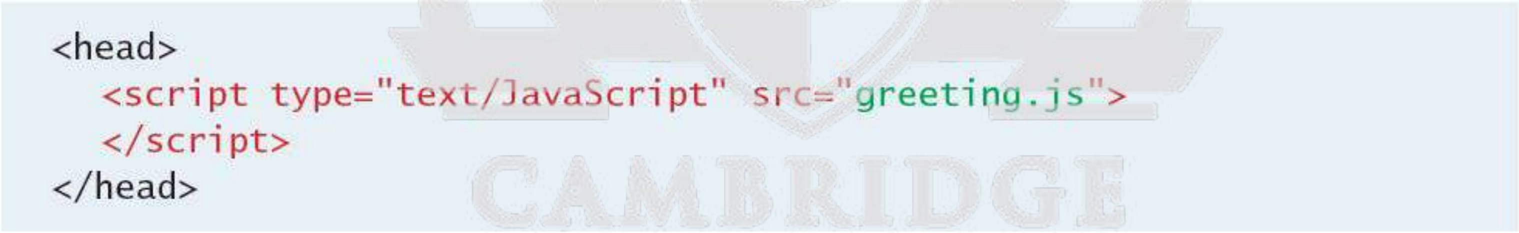
LISTING 6.2 Embedded JavaScript example

External JavaScript

Better

JavaScript supports this separation by allowing links to an external file that contains the JavaScript.

By convention, JavaScript external files have the extension .js.



```
<head>
  <script type="text/JavaScript" src="greeting.js">
  </script>
</head>
```

LISTING 6.3 External JavaScript example

Advanced Inclusion

In production sites, advanced techniques are used

- Generate embedded styles to reduce requests
- Code still managed in a external file
- <iframe> loading
- Asynchronous load from another JavaScript file
- Faster initial load

Section 4 of 8

SYNTAX

JavaScript Syntax

We will briefly cover the fundamental syntax for the most common programming constructs including

- **variables**,
- **assignment**,
- **conditionals**,
- **loops**, and
- **arrays**

before moving on to advanced topics such as **events** and **classes**.

JavaScript's Reputation

Precedes it?

JavaScript's reputation for being quirky not only stems from its strange way of implementing object-oriented principles but also from some odd syntactic *gotchas:*

- Everything is type sensitive, including function, class, and variable names.
- The scope of variables in blocks is not supported. This means variables declared inside a loop may be accessible outside of the loop, counter to what one would expect.
- There is a === operator, which tests not only for equality but type equivalence.
- **Null** and **undefined** are two distinctly different states for a variable.
- Semicolons are not required, but are permitted (and encouraged).
- There is no integer type, only number, which means floating-point rounding errors are prevalent even with values intended to be integers.

Variables

var

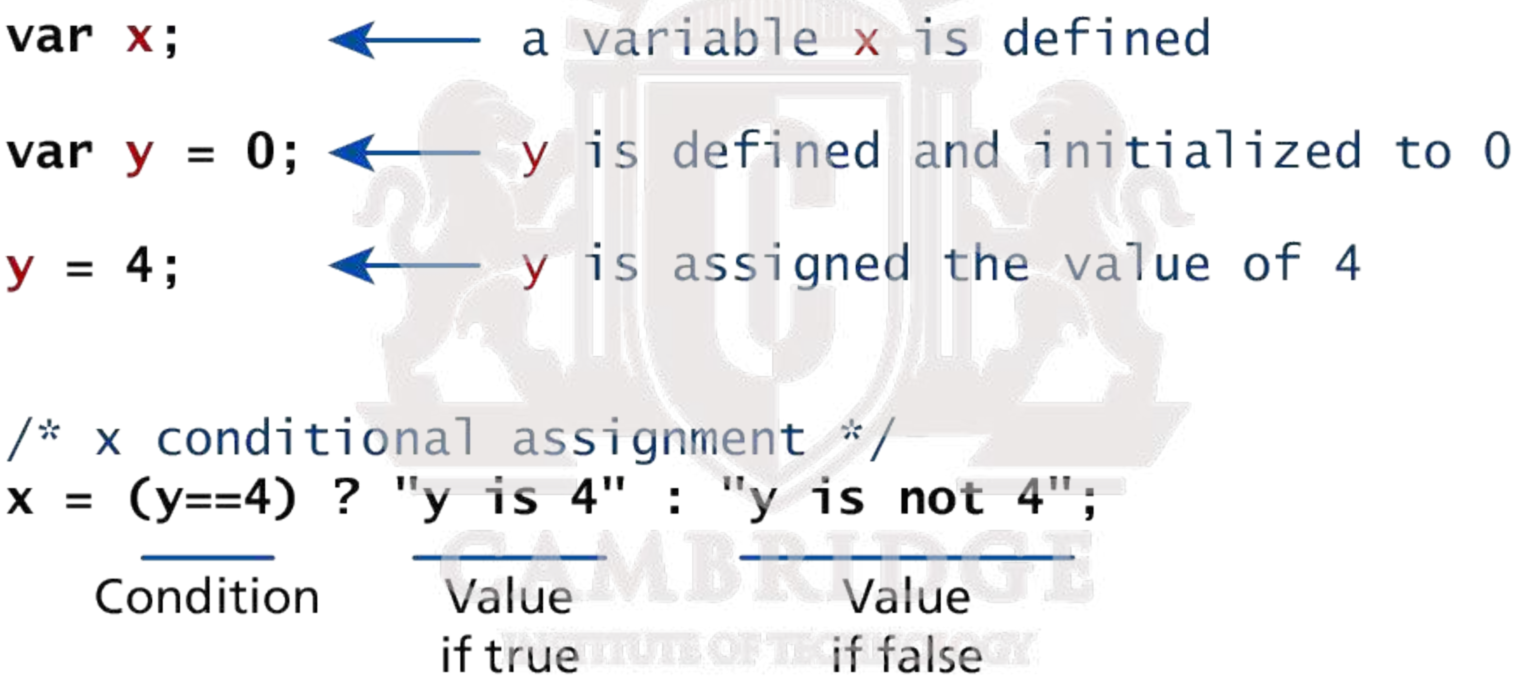
❖**Variables** in JavaScript are **dynamically typed**, meaning a variable can be an integer, and then later a string, then later an object, if so desired.

❖This simplifies variable declarations, so that we do not require the familiar type fields like *int*, *char*, and *String*. Instead we use **var**

❖**Assignment** can happen at declaration-time by appending the value to the declaration, or at run time with a simple right-to-left assignment

Variables

Assignment



```
var x;        ←  a variable x is defined

var y = 0;    ←  y is defined and initialized to 0

y = 4;        ←  y is assigned the value of 4


/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

Condition

Value if true

Value if false

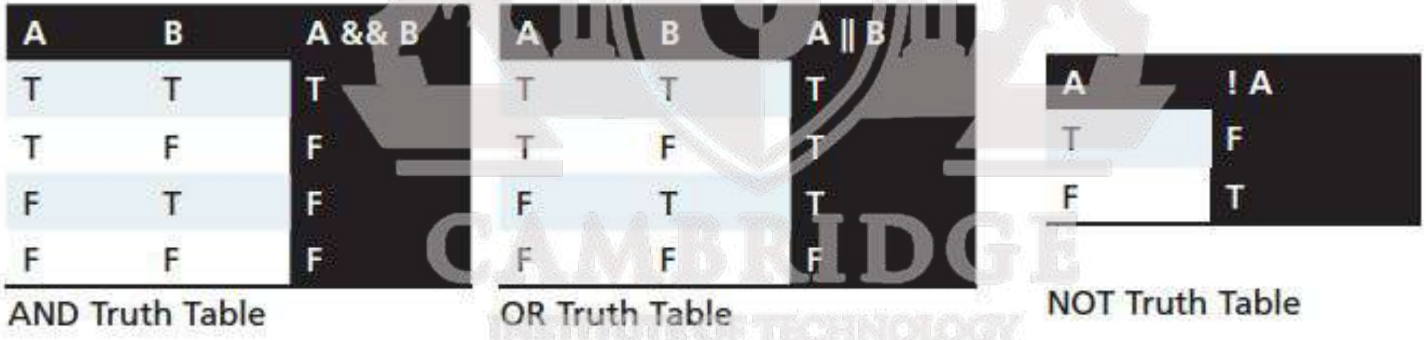
Comparison Operators

True or not True

Operator	Description	Matches (x=9)
==	Equals	(x==9) is true (x=="9") is true
===	Exactly equals, including type	(x==="9") is false (x===9) is true
< , >	Less than, Greater Than	(x<5) is false
<= , >=	Less than or equal, greater than or equal	(x<=9) is true
!=	Not equal	(4!=x) is true
!==	Not equal in either value or type	(x!=="9") is true (x!==9) is false

Logical Operators

The Boolean operators and, or, and not and their truth tables are listed in Table 6.2. Syntactically they are represented with && (and), || (or), and ! (not).



A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

AND Truth Table

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

OR Truth Table

A	! A
T	F
F	T

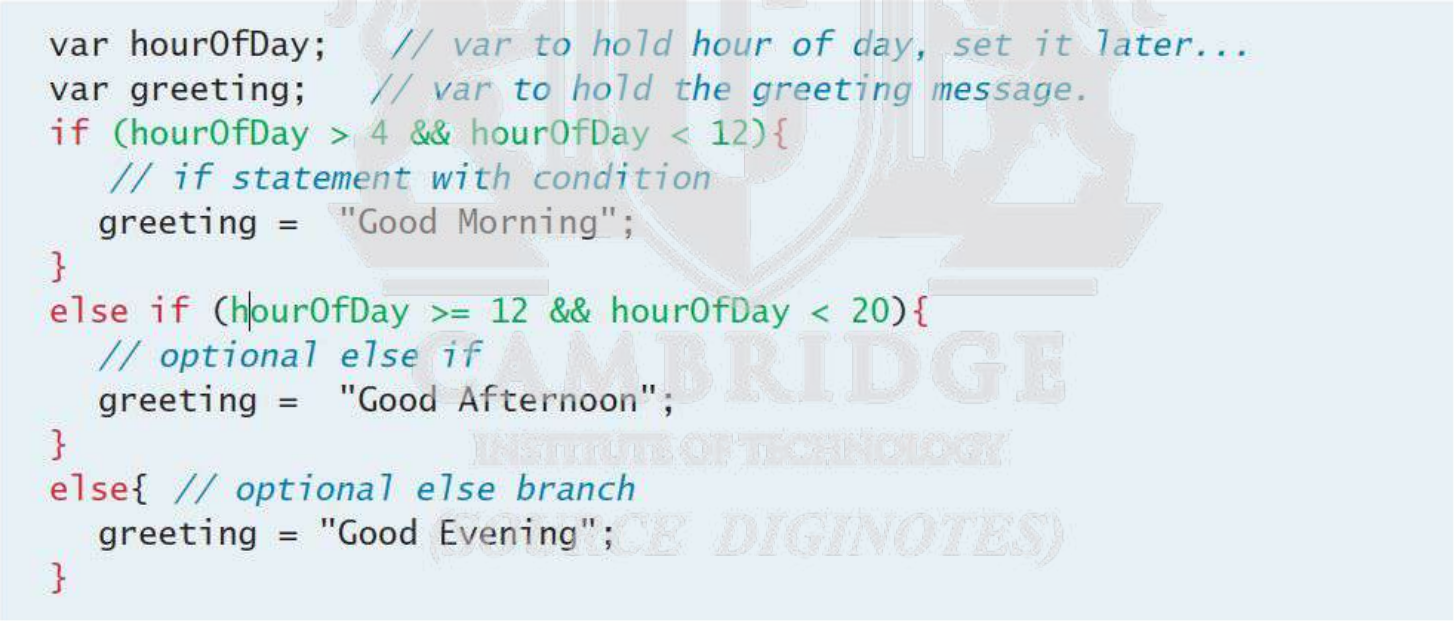
NOT Truth Table

TABLE 6.2 AND, OR, and NOT Truth Tables

Conditionals

If, else if, …, else

JavaScript’s syntax is almost identical to that of PHP, Java, or C when it comes to conditional structures such as if and if else statements. In this syntax the condition to test is contained within ( ) brackets with the body contained in { } blocks.



```
var hourOfDay;    // var to hold hour of day, set it later...
var greeting;    // var to hold the greeting message.
if (hourOfDay > 4 && hourOfDay < 12){
   // if statement with condition
   greeting =  "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 20){
   // optional else if
   greeting =  "Good Afternoon";
}
else{ // optional else branch
   greeting = "Good Evening";
}
```

LISTING 6.4 Conditional statement setting a variable based on the hour of the day

Loops

Round and round we go

- Like conditionals, loops use the ( ) and { } blocks to define the condition and the body of the loop.
- You will encounter the **while** and **for** loops
- While loops normally initialize a **loop control variable** before the loop, use it in the condition, and modify it within the loop.

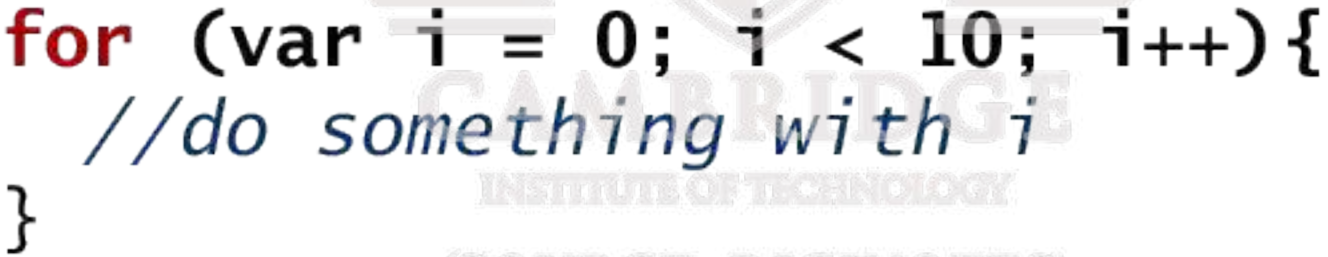
```
var i=0;  // initialise the Loop Control Variable
while(i < 10){ //test the loop control variable
      i++;  //increment the loop control variable
}
```

For Loops

Counted loops

A **for loop** combines the common components of a loop: initialization, condition, and post-loop operation into one statement.

This statement begins with the **for** keyword and has the components placed between **( )** brackets, semicolon **(;)** separated as shown



```
for (var i = 0; i < 10; i++){
  //do something with i
}
```

Functions

Functions are the building block for modular code in JavaScript, and are even used to build **pseudo-classes**, which you will learn about later.

They are defined by using the reserved word **function** and then the function name and (optional) parameters.

Since JavaScript is dynamically typed, functions do not require a return type, nor do the parameters require type.

Functions

Example

Therefore a function to raise x to the yth power might be defined as:

```
function power(x,y){
        var pow=1;
        for (var i=0;i<y;i++){
                pow = pow*x;
        }
        return pow;
}
```

And called as

```
power(2,10);
```

Alert

Not really used anymore, console instead

✓The alert() function makes the browser show a pop-up to the user, with whatever is passed being the message displayed. The following JavaScript code displays a simple hello world message in a pop-up:

alert ("Good Morning");

✓Using alerts can get tedious fast. When using debugger tools in your browser you can write output to a log with:

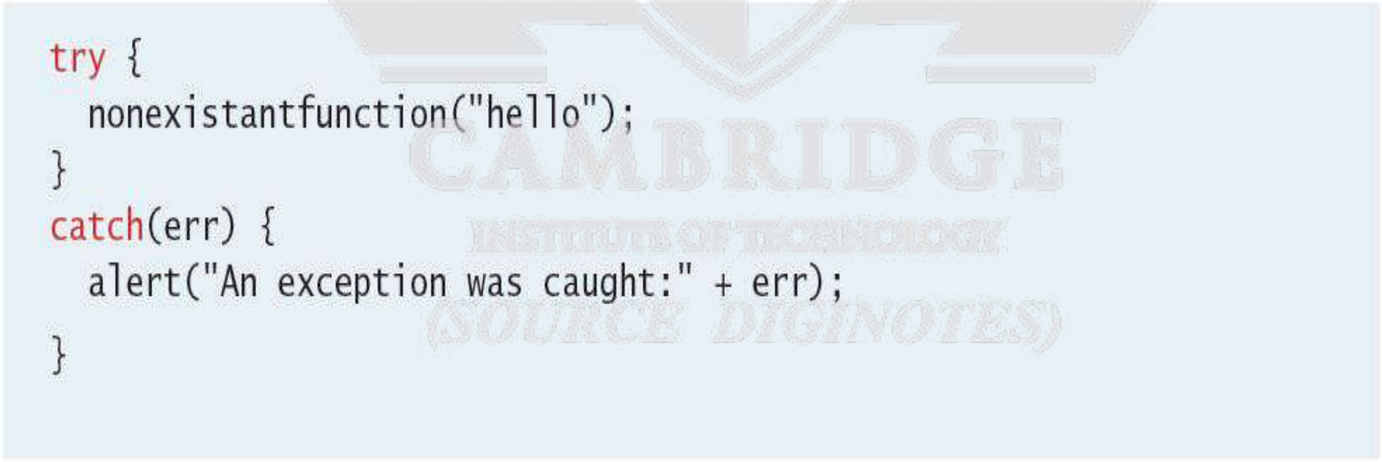
console.log("Put Messages Here");

And then use the debugger to access those logs.

Errors using try and catch

- When the browser's JavaScript engine encounters an error, it will *throw* an **exception**. These exceptions interrupt the regular, sequential execution of the program and can stop the JavaScript engine altogether.

- However, you can optionally catch these errors preventing disruption of the program using the **try–catch block**



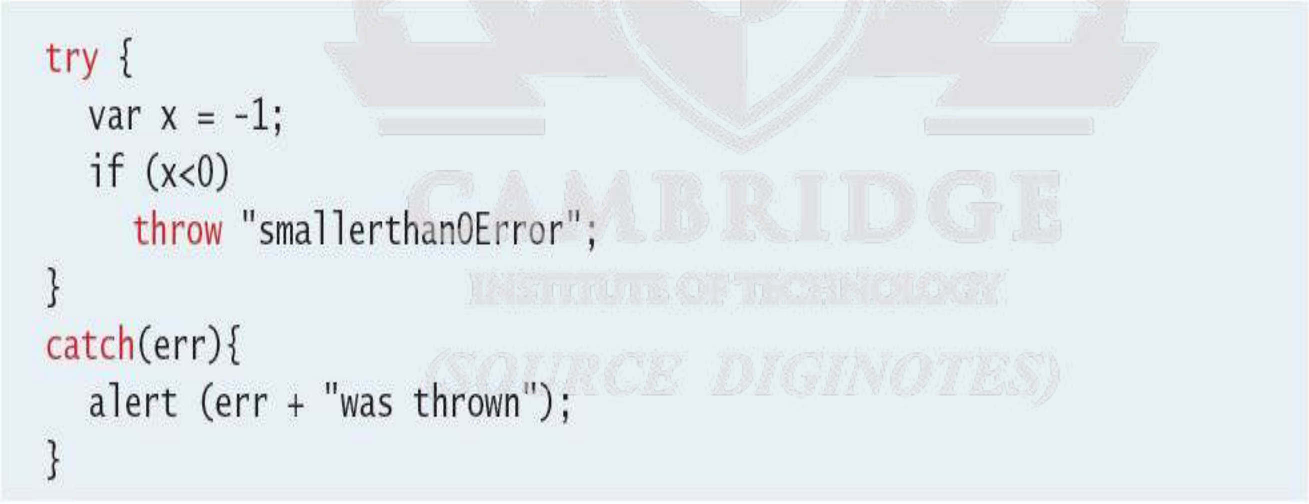
```
try {
  nonexistantfunction("hello");
}
catch(err) {
  alert("An exception was caught:" + err);
}
```

LISTING 6.5 Try-catch statement

Throw your own

Exceptions that is.

- ❑Although try-catch can be used exclusively to catch built-in JavaScript errors, it can also be used by your programs, to throw your own messages.
- ❑The throw keyword stops normal sequential execution, just like the built-in exceptions



```
try {
   var x = -1;
   if (x<0)
      throw "smallerthan0Error";
}
catch(err){
   alert (err + "was thrown");
}
```

LISTING 6.6 Throwing a user-defined exception

Tips

With Exceptions

- ❖Try-catch and throw statements should be used for *abnormal* or *exceptional* cases in your program.
- ❖Throwing an exception disrupts the sequential execution of a program. When the exception is thrown all subsequent code is not executed until the catch statement is reached.
- ❖This reinforces why try-catch is for exceptional cases.

Section 5 of 8

JAVASCRIPT OBJECTS

JavaScript Objects

Objects not Classes

- ❖JavaScript is not a full-fledged object-oriented programming language.
- ❖It does not support many of the patterns you'd expect from an object-oriented language like inheritance and polymorphism.

The language does, however, support objects.

JavaScript Objects

Not full-fledged O.O.

- ➢Objects can have **constructors**, **properties**, and **methods** associated with them.
- ➢There are objects that are included in the JavaScript language; you can also define your own kind of objects.

Constructors

- Normally to create a new object we use the new keyword, the class name, and ( ) brackets with *n* optional parameters inside, comma delimited as follows:

```
var someObject = new ObjectName(p1,p2,..., pn);
```

- For some classes, shortcut constructors are defined

```
var greeting = "Good Morning";
```

```
var greeting = new String("Good Morning");
```

Properties

Use the dot

- Each object might have properties that can be accessed, depending on its definition.
- When a property exists, it can be accessed using **dot notation** where a dot between the instance name and the property references that property.

```
//show someObject.property to the user
    alert(someObject.property);
```

Methods

Use the dot, with brackets

- Objects can also have methods, which are **functions** associated with an instance of an object. These methods are called using the same dot notation as for properties, but instead of accessing a variable, we are calling a method.

  someObject.**doSomething()**;

- Methods may produce different output depending on the object they are associated with because *they can utilize the internal properties of the object.*

Objects Included in JavaScript

A number of useful objects are included with JavaScript including:

- Array
- Boolean
- Date
- Math
- String
- Dom objects

Arrays

- Arrays are one of the most used data structures. In practice, this class is defined to behave more like a linked list in that it can be resized dynamically, but the implementation is browser specific, meaning the efficiency of insert and delete operations is unknown.
- The following code creates a new, empty array named greetings:

```
var greetings = new Array();
```

Arrays

Initialize with values

- ✓ To initialize the array with values, the variable declaration would look like the following:

```
var greetings = new Array("Good Morning", "Good Afternoon");
```

or, using the square bracket notation:

```
var greetings = ["Good Morning", "Good Afternoon"];
```

Arrays

Access and Traverse

- ✓To access an element in the array you use the familiar square bracket notation from Java and C-style languages, with the index you wish to access inside the brackets.

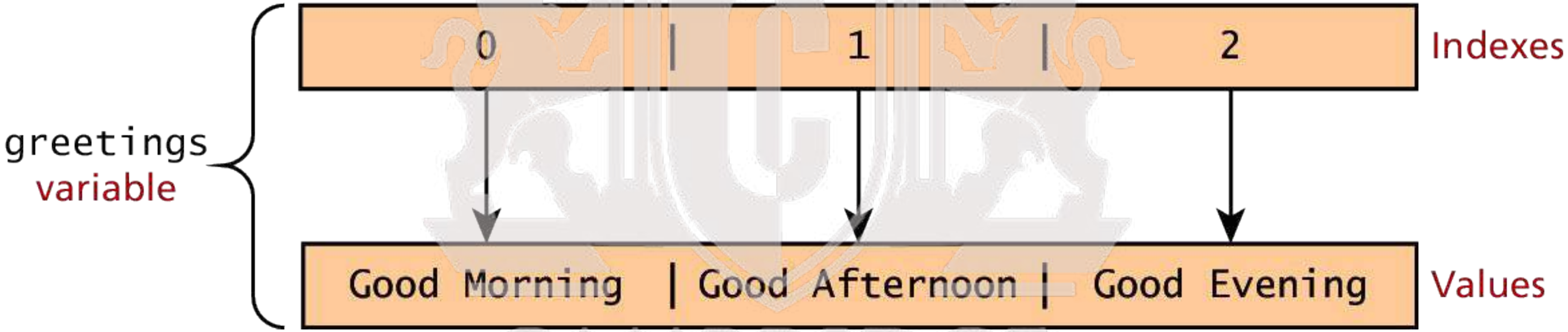
```
        alert ( greetings[0] );
```

- ✓One of the most common actions on an array is to traverse through the items sequentially. Using the Array object’s **length** property to determine the maximum valid index. We have:

```
           for (var i = 0; i < greetings.length; i++){
        alert(greetings[i]);
              }
```

Arrays

Index and Value



greetings variable

0 | 1 | 2

Indexes

Good Morning | Good Afternoon | Good Evening

Values

Arrays

Modifying an array

- ✓To add an item to an existing array, you can use the **push** method.

```
greetings.push("Good Evening");
```

- ✓The **pop** method can be used to remove an item from the back of an array.

- ✓Additional methods: concat(), slice(), join(), reverse(), shift(), and sort()

Math

- The **Math class** allows one to access common mathematic functions and common values quickly in one place.

- This static class contains methods such as max(), min(), pow(), sqrt(), and exp(), and trigonometric functions such as sin(), cos(), and arctan().

- Many mathematical constants are defined such as PI, E, SQRT2, and some others

Math.PI; *// 3.141592657*

Math.sqrt(4); *// square root of 4 is 2.*

Math.random(); *// random number between 0 and 1*

String

- The **String class** has already been used without us even knowing it.
- **Constructor usage**

```
var greet = new String("Good"); // long form constructor

var greet = "Good"; // shortcut constructor
```

- **Length of a string**

```
alert (greet.length); // will display "4"
```

String

Concatenation and so much more

```
var str = greet.concat("Morning"); // Long form concatenation

var str = greet + "Morning"; // + operator concatenation
```

Many other useful methods exist within the String class, such as

- accessing a single character using charAt()
- searching for one using indexOf().

▪Strings allow splitting a string into an array, searching and matching with split(), search(), and match() methods.

Date

Not that kind

❑The Date class is yet another helpful included object you should be aware of.

❑It allows you to quickly calculate the current date or create date objects for particular dates.

❑To display today’s date as a string, we would simply create a new object and use the toString() method.

```
var d = new Date();

// This outputs Today is Mon Nov 12 2012 15:40:19 GMT-0700

alert ("Today is "+ d.toString());
```

Window

- ❖The window object in JavaScript corresponds to the browser itself. Through it, you can access the current page's URL, the browser's history, and what's being displayed in the status bar, as well as opening new browser windows.
- ❖In fact, the alert() function mentioned earlier is actually a method of the window object.

Section 6 of 8

THE DOCUMENT OBJECT MODEL (DOM)

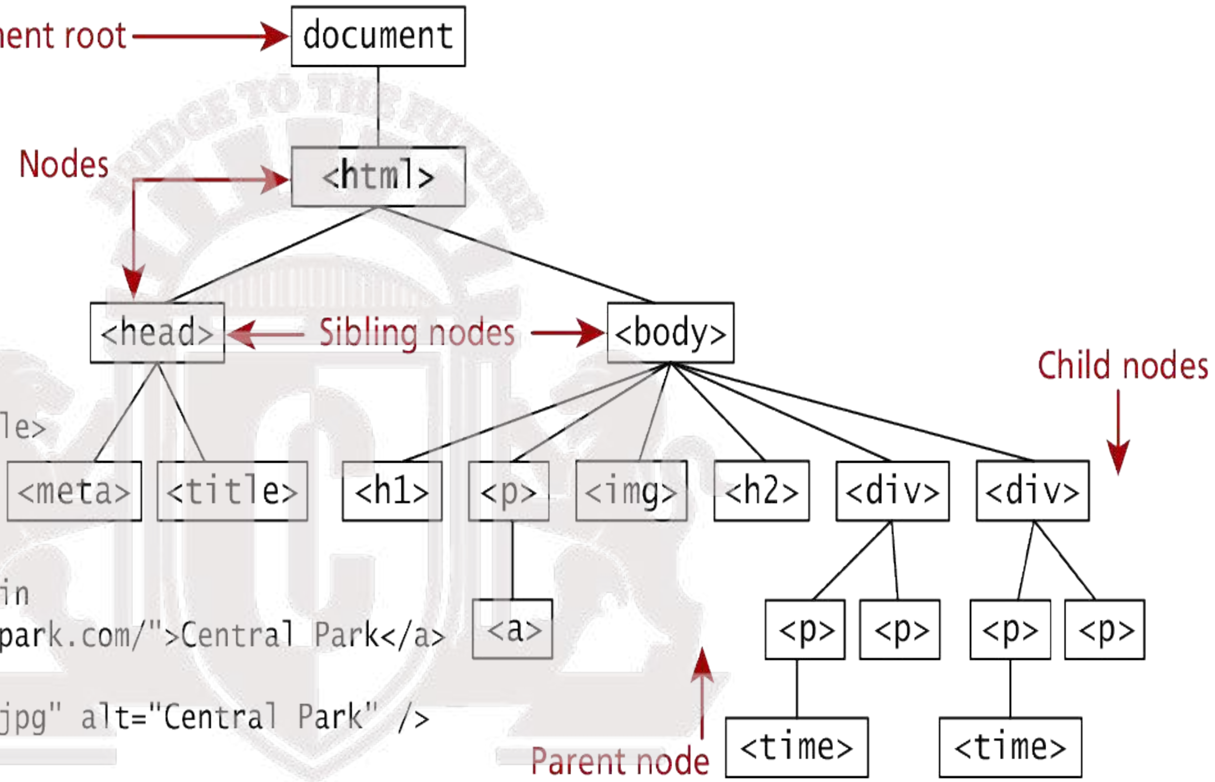
The DOM

Document Object Model

- JavaScript is almost always used to interact with the HTML document in which it is contained.
- This is accomplished through a programming interface (API) called the **Document Object Model.**

- According to the W3C, the DOM is a:

  Platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.



```
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
</head>
<body>
   <h1>Share Your Travels</h1>
   <p>Photo of Conservatory Pond in
      <a href="http://www.centralpark.com/">Central Park</a>
   </p>
   <img src="images/central-park.jpg" alt="Central Park" />
   <h2>Reviews</h2>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
</body>
</html>
```

DOM Nodes

- In the DOM, each element within the HTML document is called a **node.** If the DOM is a tree, then each node is an individual branch.

  There are:

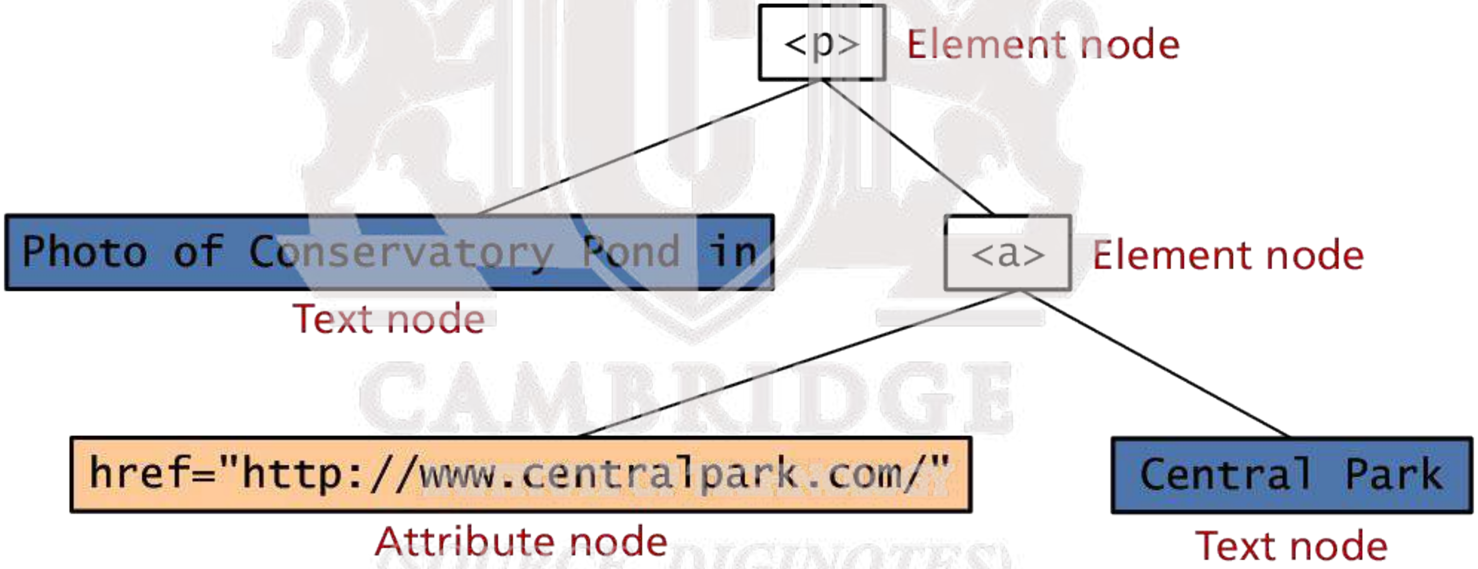
  - element nodes,
  - text nodes, and
  - attribute nodes

- All nodes in the DOM share a common set of properties and methods.

DOM Nodes

Element, text and attribute nodes

```
<p>Photo of Conservatory Pond in
   <a href="http://www.centralpark.com/">Central Park</a>
</p>
```



Source diginotes.in

DOM Nodes

Essential Node Object properties

Property	Description
attributes	Collection of node attributes
childNodes	A NodeList of child nodes for this node
firstChild	First child node of this node.
lastChild	Last child of this node.
nextSibling	Next sibling node for this node.
nodeName	Name of the node
nodeType	Type of the node
nodeValue	Value of the node
parentNode	Parent node for this node.
previousSibling	Previous sibling node for this node.

Source diginotes.in

Document Object

One root to ground them all

❑The **DOM document object** is the root JavaScript object representing the entire HTML document.

❑It contains some properties and methods that we will use extensively in our development and is globally accessible as **document**.

```
        // specify the doctype, for example html

var a = document.doctype.name;

            // specify the page encoding, for example ISO-8859-1

var b = document.inputEncoding;
```

Document Object

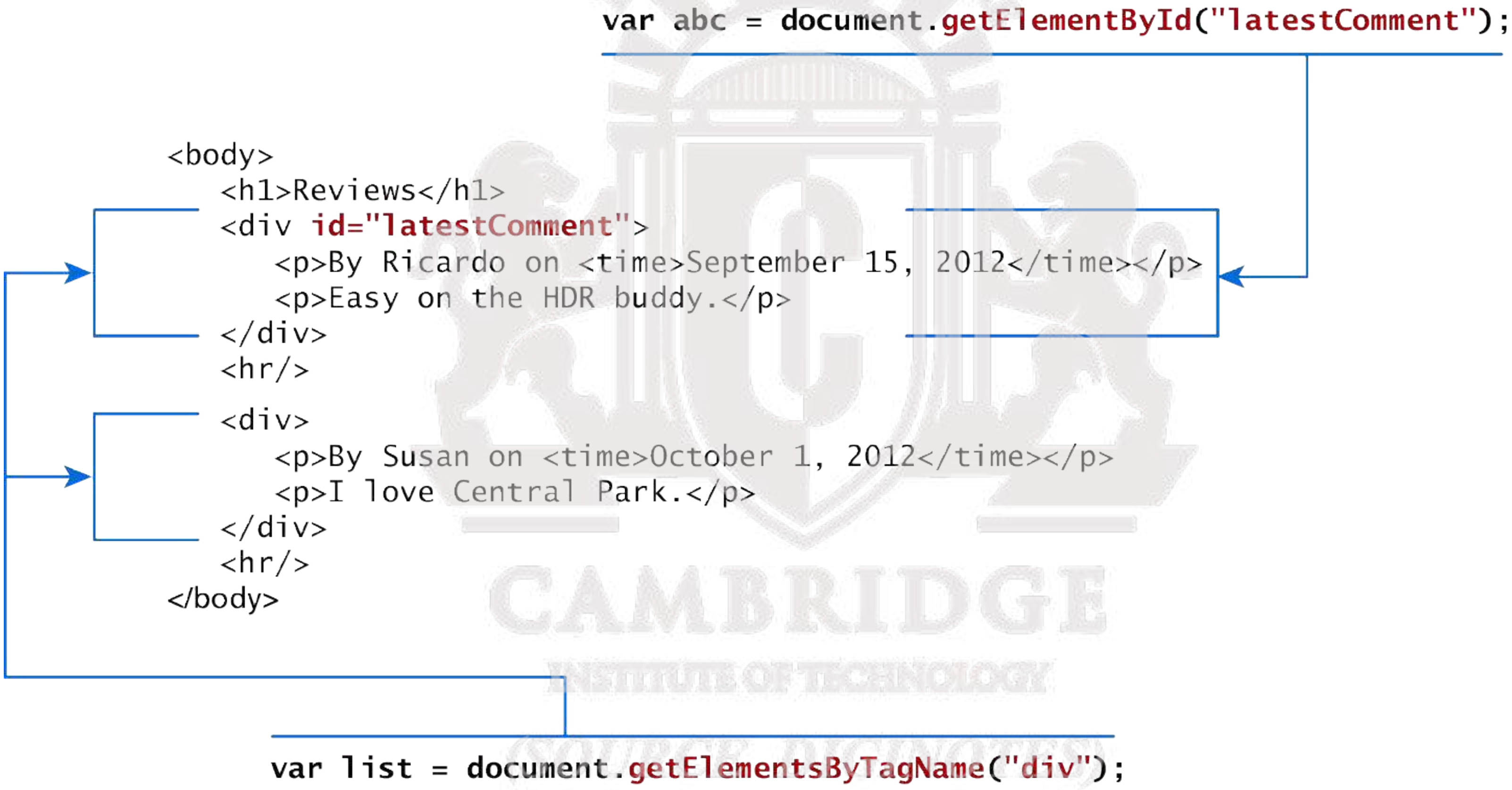
Document Object Methods

Method	Description
createAttribute()	Creates an attribute node
createElement()	Creates an element node
createTextNode()	Create a text node
getElementById(id)	Returns the element node whose id attribute matches the passed id parameter.
getElementsByTagName(name)	Returns a nodeList of elements whose tag name matches the passed name parameter.

Source diginotes.in

Accessing nodes

getElementById(), getElementsByTagName()



```
var abc = document.getElementById("latestComment");

<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>

var list = document.getElementsByTagName("div");
```

Element node Object

The type of object returned by the method document.getElementById() described in the previous section is an **element node** object.

This represents an HTML element in the hierarchy, contained between the opening <> and closing </> tags for this element.

- can itself contain more elements

Element node Object

Essential Element Node Properties

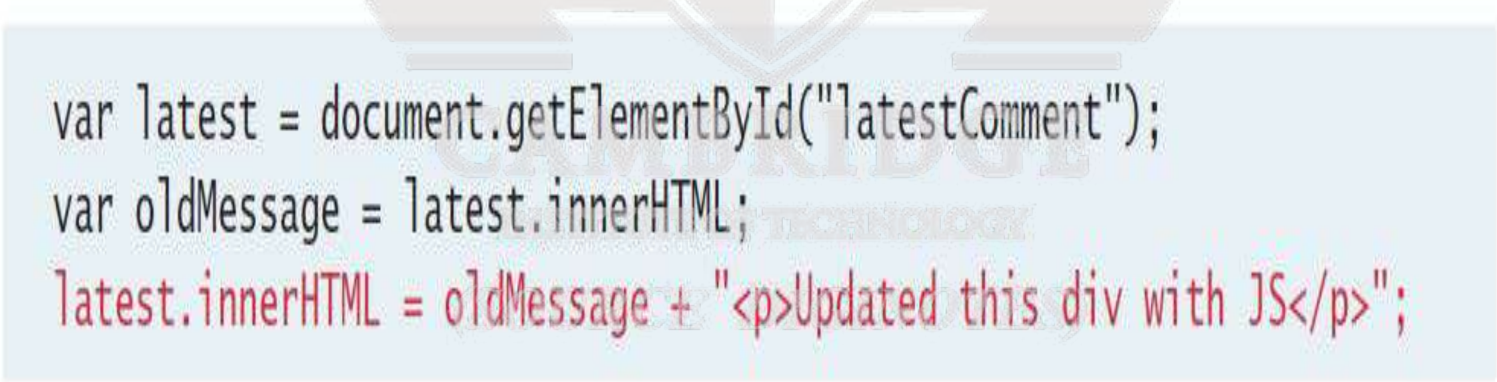
Property	Description
className	The current value for the class attribute of this HTML element.
id	The current value for the id of this element.
innerHTML	Represents all the things inside of the tags. This can be read or written to and is the primary way which we update particular div's using JS.
style	The style attribute of an element. We can read and modify this property.
tagName	The tag name for the element.

Source diginotes.in

Modifying a DOM element

- The document.write() method is used to create output to the HTML page from JavaScript. The modern JavaScript programmer will want to write to the HTML page, but in a particular location, not always at the bottom

- Using the DOM document and HTML DOM element objects, we can do exactly that using the **innerHTML** property



```
var latest = document.getElementById("latestComment");
var oldMessage = latest.innerHTML;
latest.innerHTML = oldMessage + "<p>Updated this div with JS</p>";
```

LISTING 6.8 Changing the HTML using innerHTML

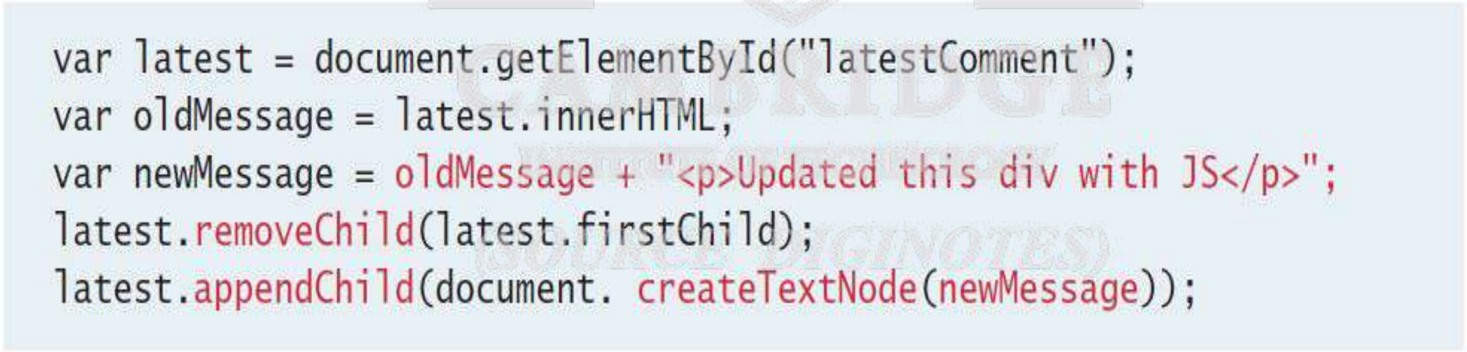
Source diginotes.in

Modifying a DOM element

More verbosely, and validated

- ➢Although the innerHTML technique works well (and is very fast), there is a more verbose technique available to us that builds output using the DOM.

- ➢DOM functions createTextNode(), removeChild(), and appendChild() allow us to modify an element in a more rigorous way



```
var latest = document.getElementById("latestComment");
var oldMessage = latest.innerHTML;
var newMessage = oldMessage + "<p>Updated this div with JS</p>";
latest.removeChild(latest.firstChild);
latest.appendChild(document. createTextNode(newMessage));
```

LISTING 6.9 Changing the HTML using createTextNode( ) and appendChild( )

Changing an element’s style

- ✓We can add or remove any style using the **style** or **className** property of the Element node.
- ✓Its usage is shown below to change a node’s background color and add a three-pixel border.

```
var commentTag = document.getElementById("specificTag");
   commentTag.style.backgroundColour = "#FFFF00";
commentTag.style.borderWidth="3px";
```

Changing an element’s style

With class

- The className property is normally a better choice, because it allows the styles to be created outside the code, and thus be better accessible to designers.

```
var commentTag = document.getElementById("specificTag");

commentTag.className = "someClassName";
```

- HTML5 introduces the classList element, which allows you to add, remove, or toggle a CSS class on an element.

```
label.classList.addClass("someClassName");
```

More Properties

Some Specific HTML DOM Element Properties for Certain Tag Types

Property	Description	Tags
href	The href attribute used in a tags to specify a URL to link to.	a
name	The name property is a bookmark to identify this tag. Unlike id which is available to all tags, name is limited to certain form related tags.	a, input, textarea , form
src	Links to an external URL that should be loaded into the page (as opposed to href which is a link to follow when clicked)	img, input, iframe, script
value	The value is related tot he value attribute of input tags. Often the value of an input field is user defined, and we use value to get that user input.	Input, textarea , submit

Section 7 of 8

JAVASCRIPT EVENTS

JavaScript Events

- A JavaScript **event** is an action that can be detected by JavaScript.
- We say then that an event is *triggered* and then it can be *caught* by JavaScript functions, which then do something in response.

JavaScript Events

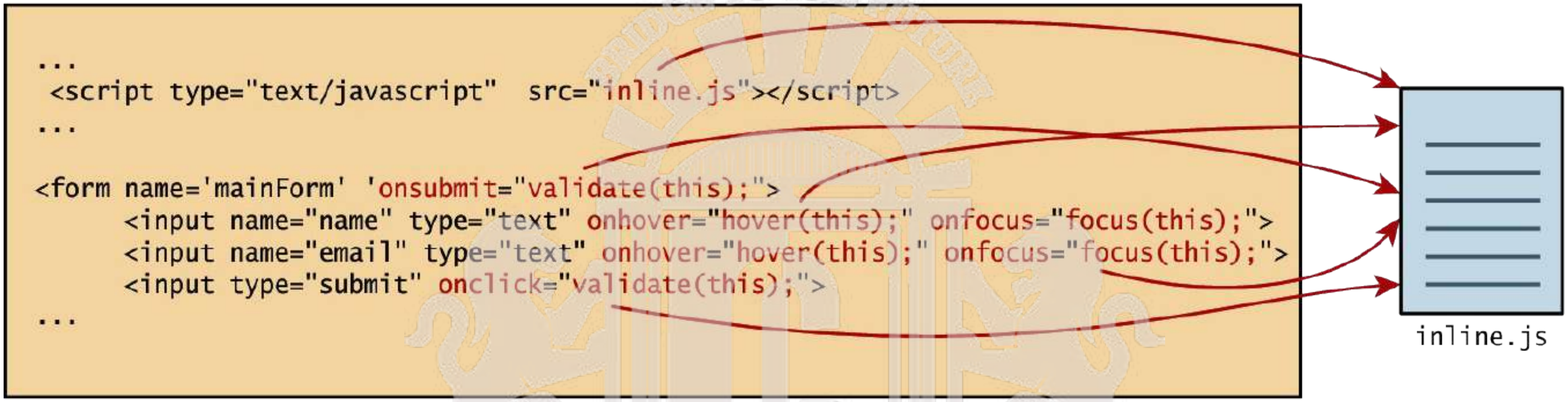
A brave new world

- In the original JavaScript world, events could be specified right in the HTML markup with *hooks* to the JavaScript code (and still can).
- As more powerful frameworks were developed, and website design and best practices were refined, this original mechanism was supplanted by the **listener** approach.

JavaScript Events

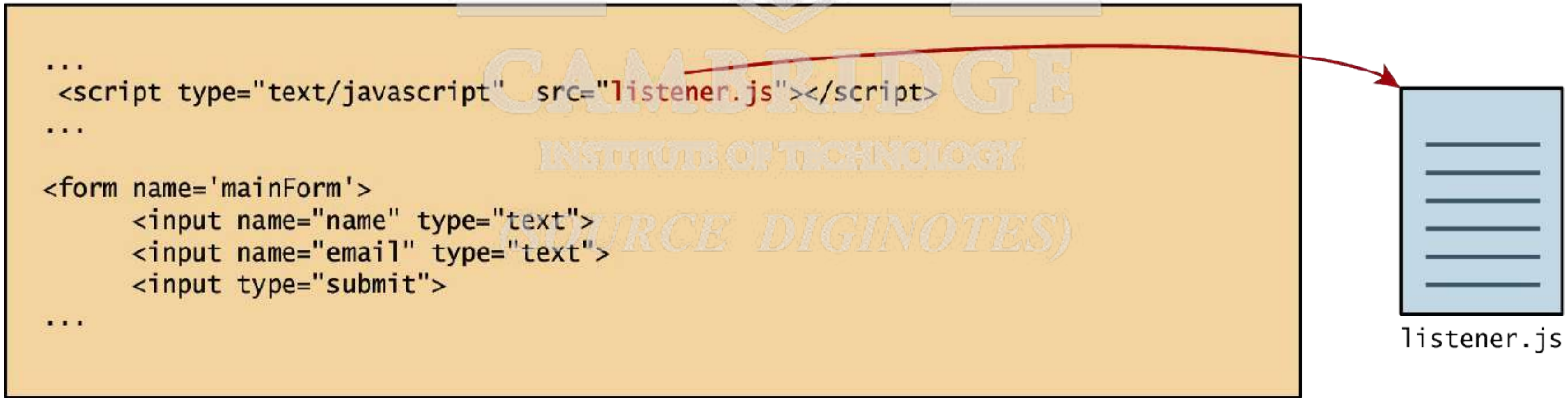
Two approaches

Old, Inline technique



```
...
 <script type="text/javascript"  src="inline.js"></script>
...

<form name='mainForm' 'onsubmit="validate(this);">
      <input name="name" type="text" onhover="hover(this);" onfocus="focus(this);">
      <input name="email" type="text" onhover="hover(this);" onfocus="focus(this);">
      <input type="submit" onclick="validate(this);">
...
```

New, Layered Listener technique



```
...
 <script type="text/javascript"  src="listener.js"></script>
...

<form name='mainForm'>
      <input name="name" type="text">
      <input name="email" type="text">
      <input type="submit">
...
```

Inline Event Handler Approach

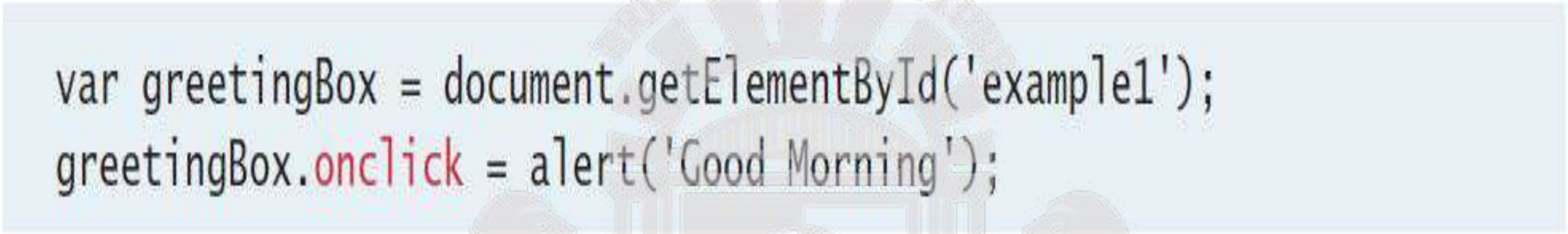
- ❖For example, if you wanted an alert to pop-up when clicking a <div> you might program:

```
<div id="example1" onclick="alert('hello')">Click for pop-up</div>
```

- ❖The problem with this type of programming is that the HTML markup and the corresponding JavaScript logic are woven together. It does not make use of layers;
- ❖that is, it does not separate content from behavior.

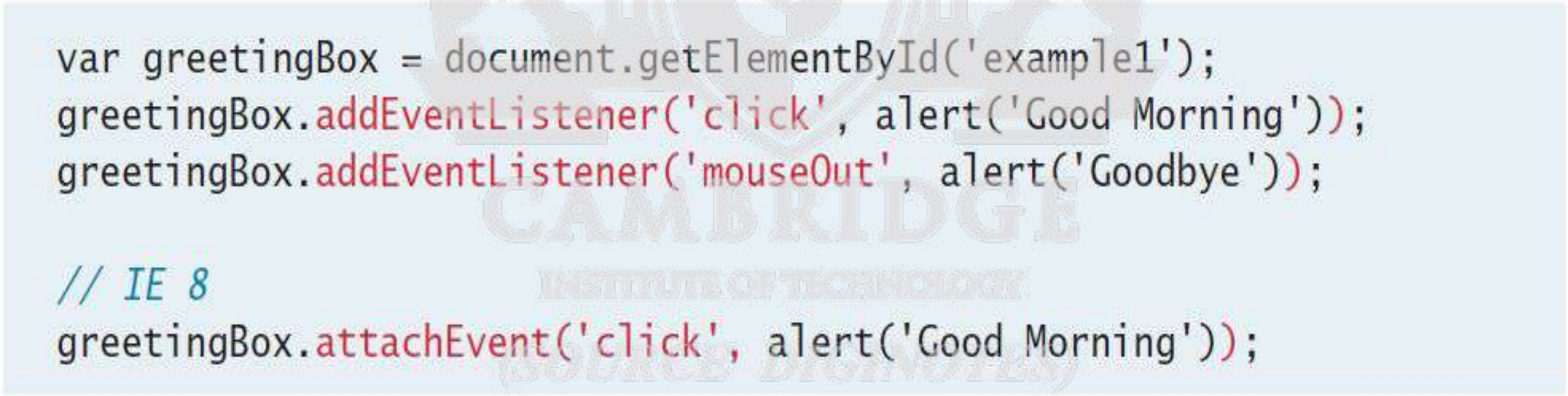
Listener Approach

Two ways to set up listeners



```
var greetingBox = document.getElementById('example1');
greetingBox.onclick = alert('Good Morning');
```

LISTING 6.10 The “old” style of registering a listener.



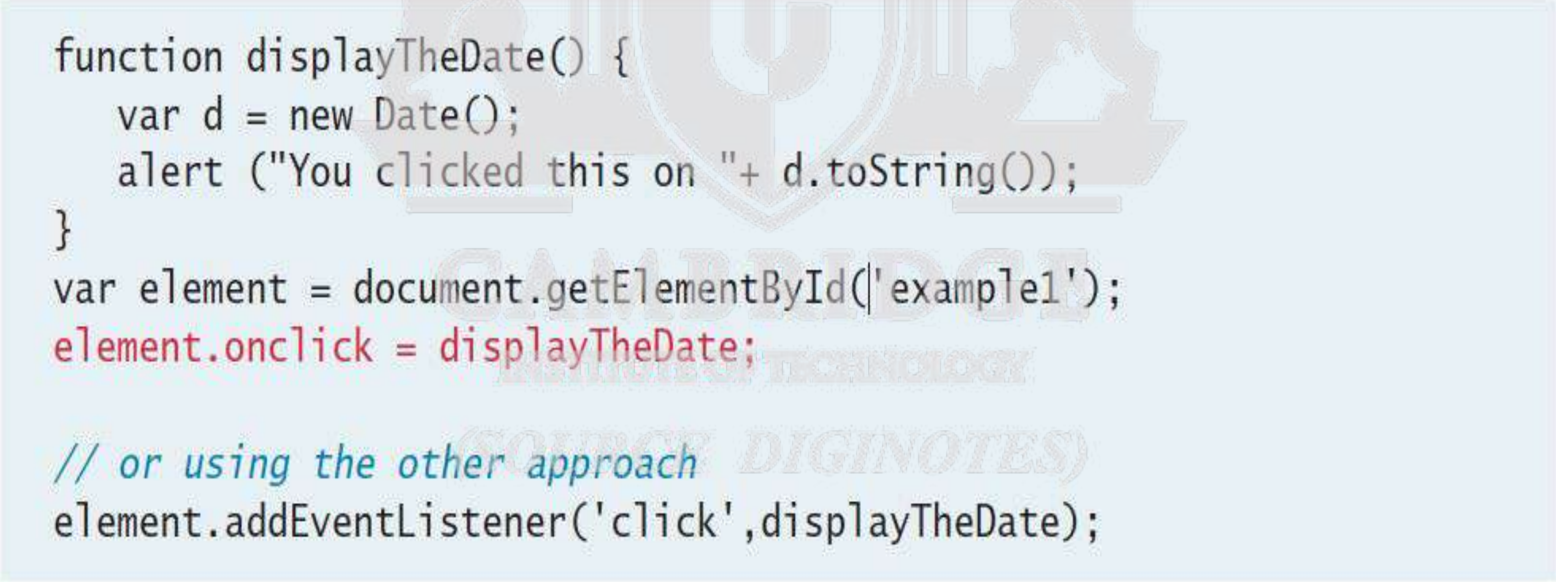
```
var greetingBox = document.getElementById('example1');
greetingBox.addEventListener('click', alert('Good Morning'));
greetingBox.addEventListener('mouseOut', alert('Goodbye'));

// IE 8
greetingBox.attachEvent('click', alert('Good Morning'));
```

LISTING 6.11 The “new” DOM2 approach to registering listeners.

Listener Approach

Using functions

- ❑What if we wanted to do something more elaborate when an event is triggered? In such a case, the behavior would have to be encapsulated within a function, as shown in Listing 6.12.



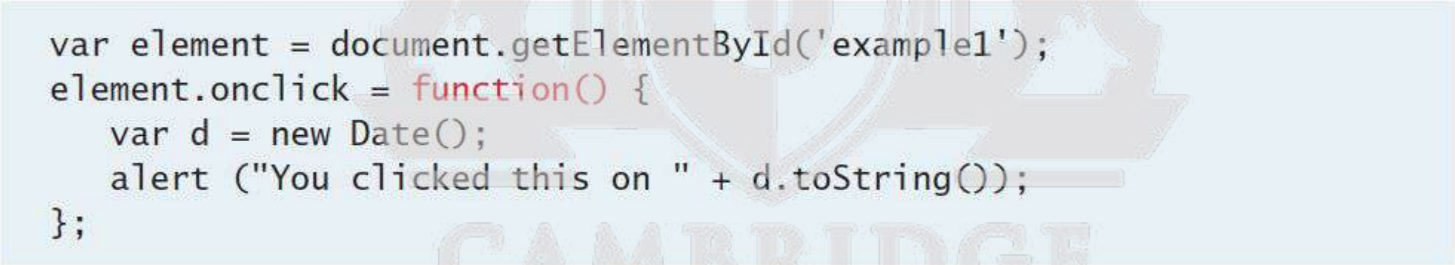
```
function displayTheDate() {
    var d = new Date();
    alert ("You clicked this on "+ d.toString());
}
var element = document.getElementById('example1');
element.onclick = displayTheDate;

// or using the other approach
element.addEventListener('click',displayTheDate);
```

LISTING 6.12 Listening to an event with a function

Listener Approach

Anonymous functions

An alternative to that shown in Listing 6.12 is to use an anonymous function (that is, one without a name), as shown in Listing 6.13.



```
var element = document.getElementById('example1');
element.onclick = function() {
   var d = new Date();
   alert ("You clicked this on " + d.toString());
};
```

LISTING 6.13 Listening to an event with an anonymous function

Event Object

✓No matter which type of event we encounter, they are all **DOM event objects** and the event handlers associated with them can access and manipulate them.

✓Typically we see the events passed to the function handler as a parameter named *e*.

```
function someHandler(e) {
    // e is the event that triggered this handler.
      }
```

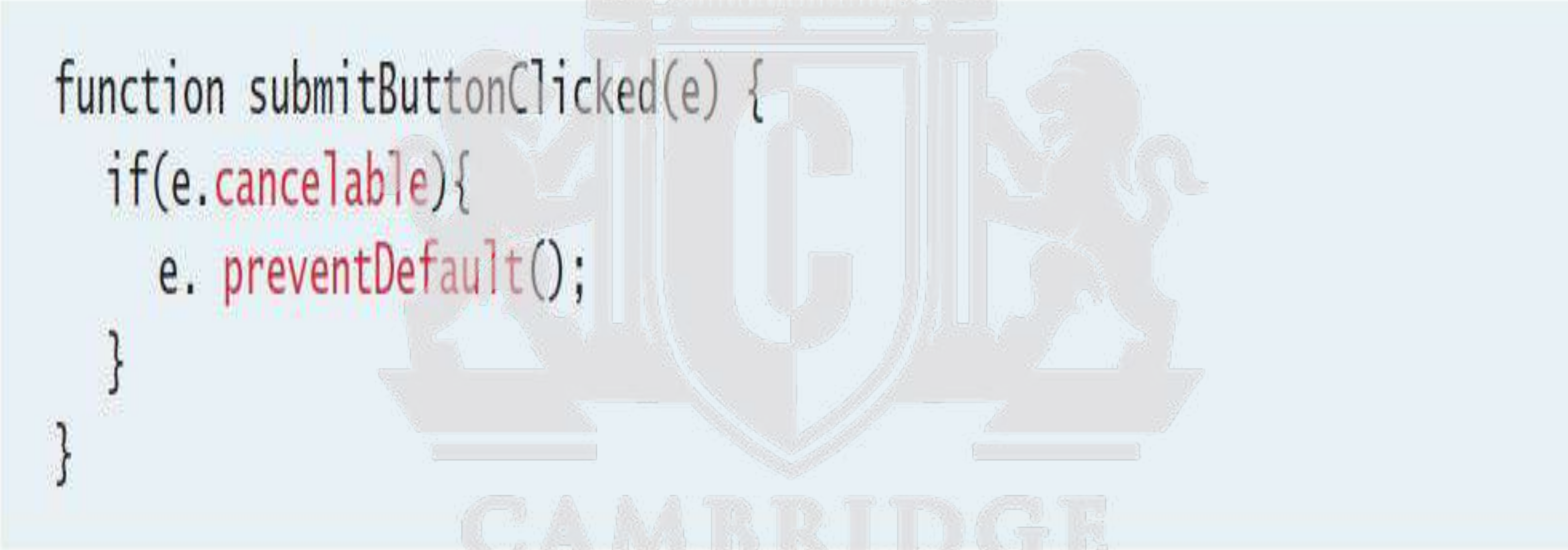
Event Object

Several Options

- **Bubbles**. If an event’s bubbles property is set to true then there must be an event handler in place to handle the event or it will bubble up to its parent and trigger an event handler there.
- **Cancelable**. The Cancelable property is also a Boolean value that indicates whether or not the event can be cancelled.
- **preventDefault**. A cancelable default action for an event can be stopped using the preventDefault() method in the next slide

Event Object

Prevent the default behaviour



```
function submitButtonClicked(e) {
  if(e.cancelable){
    e. preventDefault();
  }
}
```

LISTING 6.14 A sample event handler function that prevents the default event

Event Types

There are several classes of event, with several types of event within each class specified by the W3C:

- mouse events
- keyboard events
- form events
- frame events

Mouse events

Event	Description
onclick	The mouse was clicked on an element
ondblclick	The mouse was double clicked on an element
onmousedown	The mouse was pressed down over an element
onmouseup	The mouse was released over an element
onmouseover	The mouse was moved (not clicked) over an element
onmouseout	The mouse was moved off of an element
onmousemove	The mouse was moved while over an element

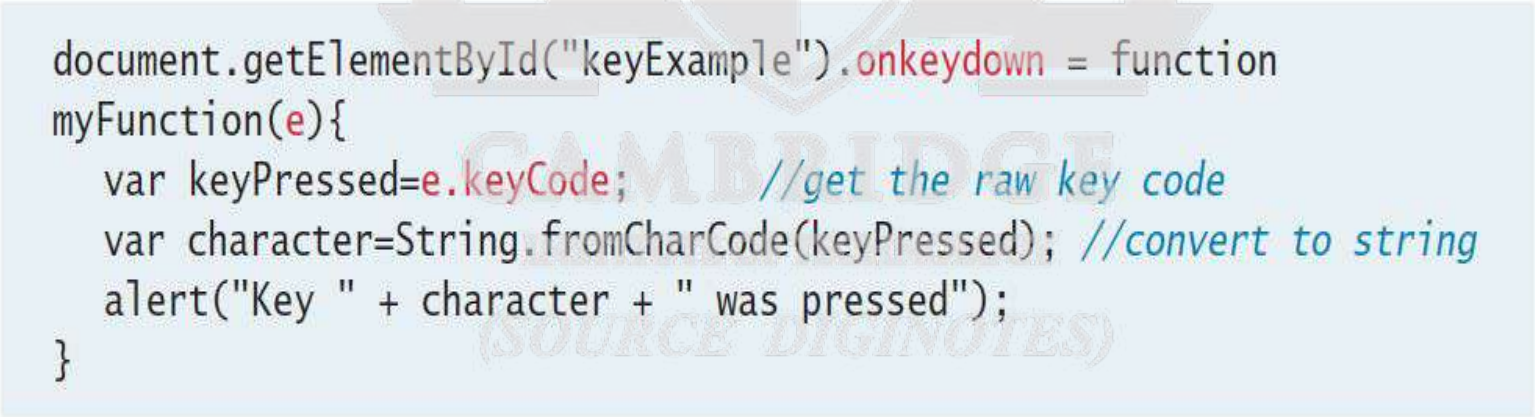
Keyboard events

Event	Description
onkeydown	The user is pressing a key (this happens first)
onkeypress	The user presses a key (this happens after onkeydown)
onkeyup	The user releases a key that was down (this happens last)

Keyboard events

Example

- <input type="text" id="keyExample">
- The input box above, for example, could be listened to and each key pressed echoed back to the user as an alert as shown in Listing 6.15.



```
document.getElementById("keyExample").onkeydown = function
myFunction(e){
   var keyPressed=e.keyCode;        //get the raw key code
   var character=String.fromCharCode(keyPressed); //convert to string
   alert("Key " + character + " was pressed");
}
```

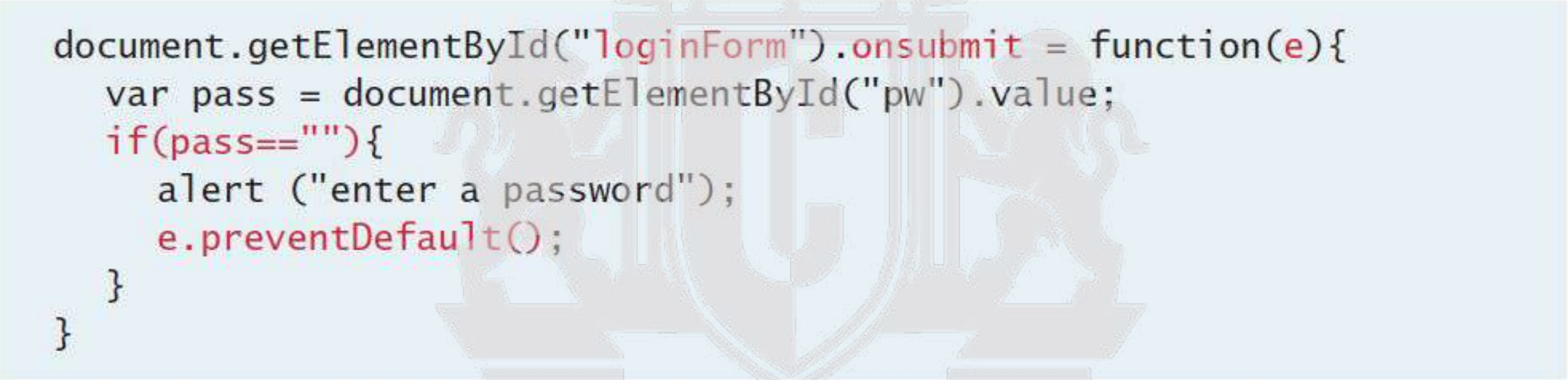
LISTING 6.15 Listener that hears and alerts keypresses

Form Events

Event	Description
onblur	A form element has lost focus (that is, control has moved to a different element, perhaps due to a click or Tab key press.
onchange	Some <input>, <textarea> or <select> field had their value change. This could mean the user typed something, or selected a new choice.
onfocus	Complementing the onblur event, this is triggered when an element gets focus (the user clicks in the field or tabs to it)
onreset	HTML forms have the ability to be reset. This event is triggered when that happens.
onselect	When the users selects some text. This is often used to try and prevent copy/paste.
onsubmit	When the form is submitted this event is triggered. We can do some pre-validation when the user submits the form in JavaScript before sending the data on to the server.

Form Events

Example



```
document.getElementById("loginForm").onsubmit = function(e){
   var pass = document.getElementById("pw").value;
   if(pass==""){
      alert ("enter a password");
      e.preventDefault();
   }
}
```

LISTING 6.16 Catching the onsubmit event and validating a password to not be blank

Frame Events

- **Frame events** are the events related to the browser frame that contains your web page.
- The most important event is the **onload** event, which tells us an object is loaded and therefore ready to work with. If the code attempts to set up a listener on this not-yet-loaded <div>, then an error will be triggered.

```
        window.onload= function(){
     //all JavaScript initialization here.
}
```

Frame Events

Table of frame events

Event	Description
onabort	An object was stopped from loading
onerror	An object or image did not properly load
onload	When a document or object has been loaded
onresize	The document view was resized
onscroll	The document view was scrolled
onunload	The document has unloaded

Section 8 of 8

FORMS

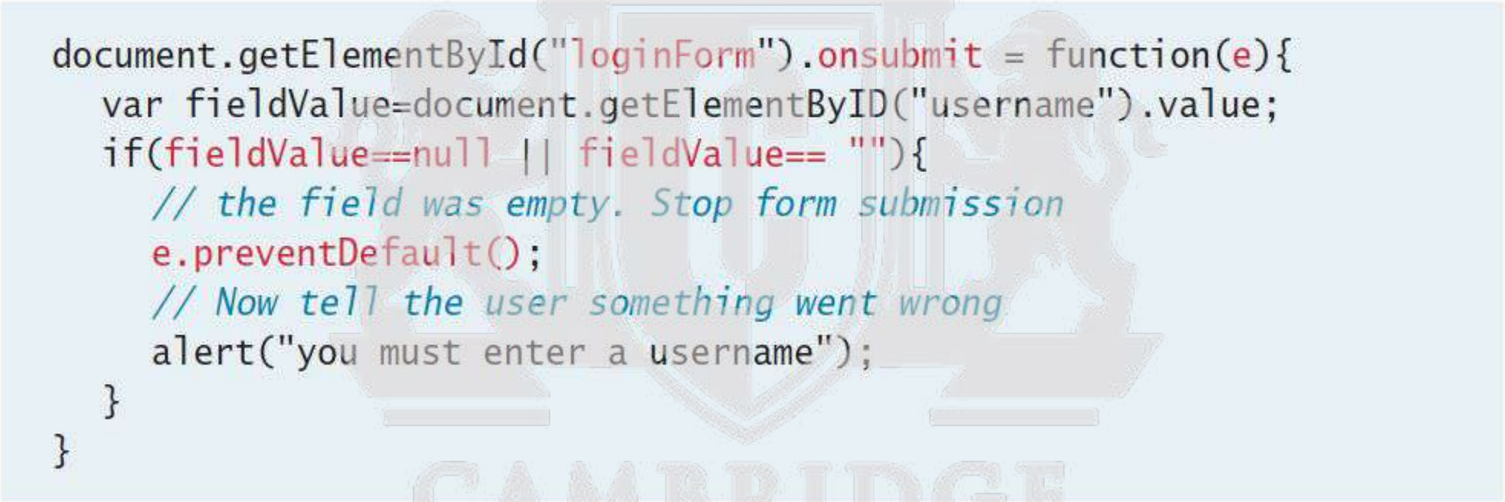
Validating Forms

You mean pre-validating right?

- Writing code to prevalidate forms on the client side will reduce the number of incorrect submissions, thereby reducing server load.
- There are a number of common validation activities including email validation, number validation, and data validation.

Validating Forms

Empty field



```
document.getElementById("loginForm").onsubmit = function(e){
   var fieldValue=document.getElementByID("username").value;
   if(fieldValue==null || fieldValue== ""){
      // the field was empty. Stop form submission
      e.preventDefault();
      // Now tell the user something went wrong
      alert("you must enter a username");
   }
}
```

LISTING 6.18 A simple validation script to check for empty fields

Validating Forms

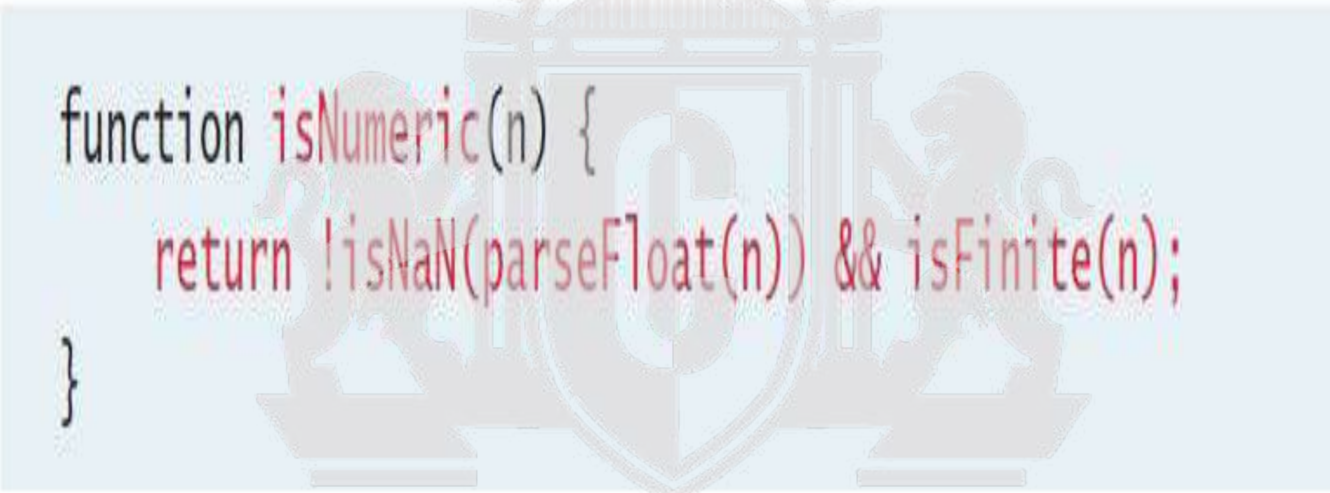
Empty field

❖If you want to ensure a checkbox is ticked, use code like that below.

```
  var inputField=document.getElementByID("license");
              if (inputField.type=="checkbox")
                   {
                 if (inputField.checked)
             //Now we know the box is checked
                    }
```

Validating Forms

Number Validation



```
function isNumeric(n) {
    return !isNaN(parseFloat(n)) && isFinite(n);
}
```

LISTING 6.19 A function to test for a numeric value

Submitting Forms

- ❖Submitting a form using JavaScript requires having a node variable for the form element. Once the variable, say, formExample is acquired, one can simply call the submit() method:

```
var formExample = document.getElementById("loginForm");

formExample.submit();
```

- ✓This is often done in conjunction with calling **preventDefault()** on the onsubmit event.

Introduction to Server-Side Development with PHP

Chapter 8

Section 1 of 5

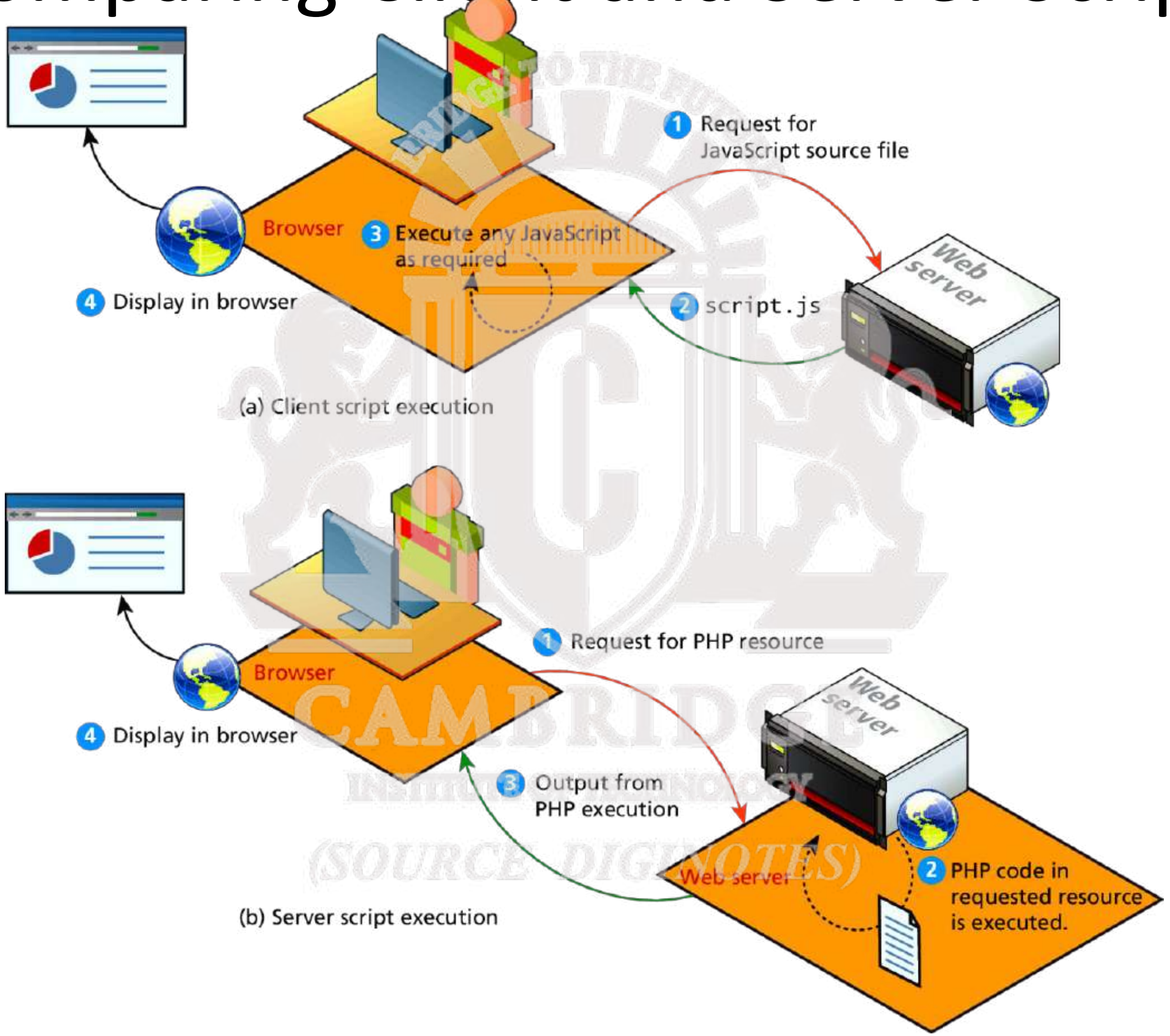
WHAT IS SERVER-SIDE DEVELOPMENT

What is Server-Side Development

- The basic hosting of your files is achieved through a web server.
- Server-side development is much more than web hosting: it involves the use of a programming technology like PHP or ASP.NET to create scripts that dynamically generate content

Consider distinction between client side and server side…

Comparing Client and Server Scripts

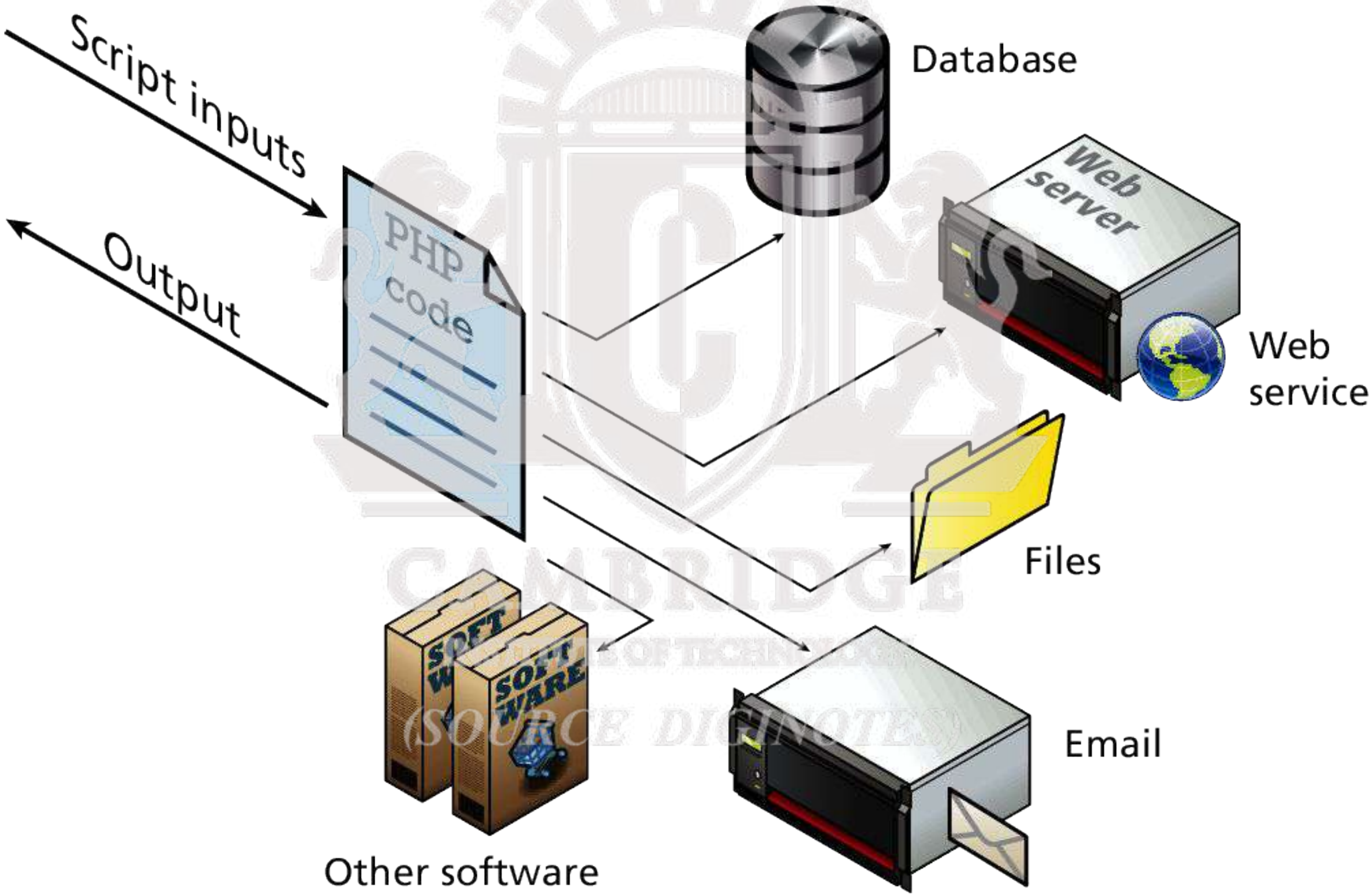


Request for JavaScript source file
Browser
3 Execute any JavaScript as required
4 Display in browser
2 script.js
Web server
(a) Client script execution
1 Request for PHP resource
Browser
Web server
4 Display in browser
3 Output from PHP execution
Web server
2 PHP code in requested resource is executed.
(b) Server script execution

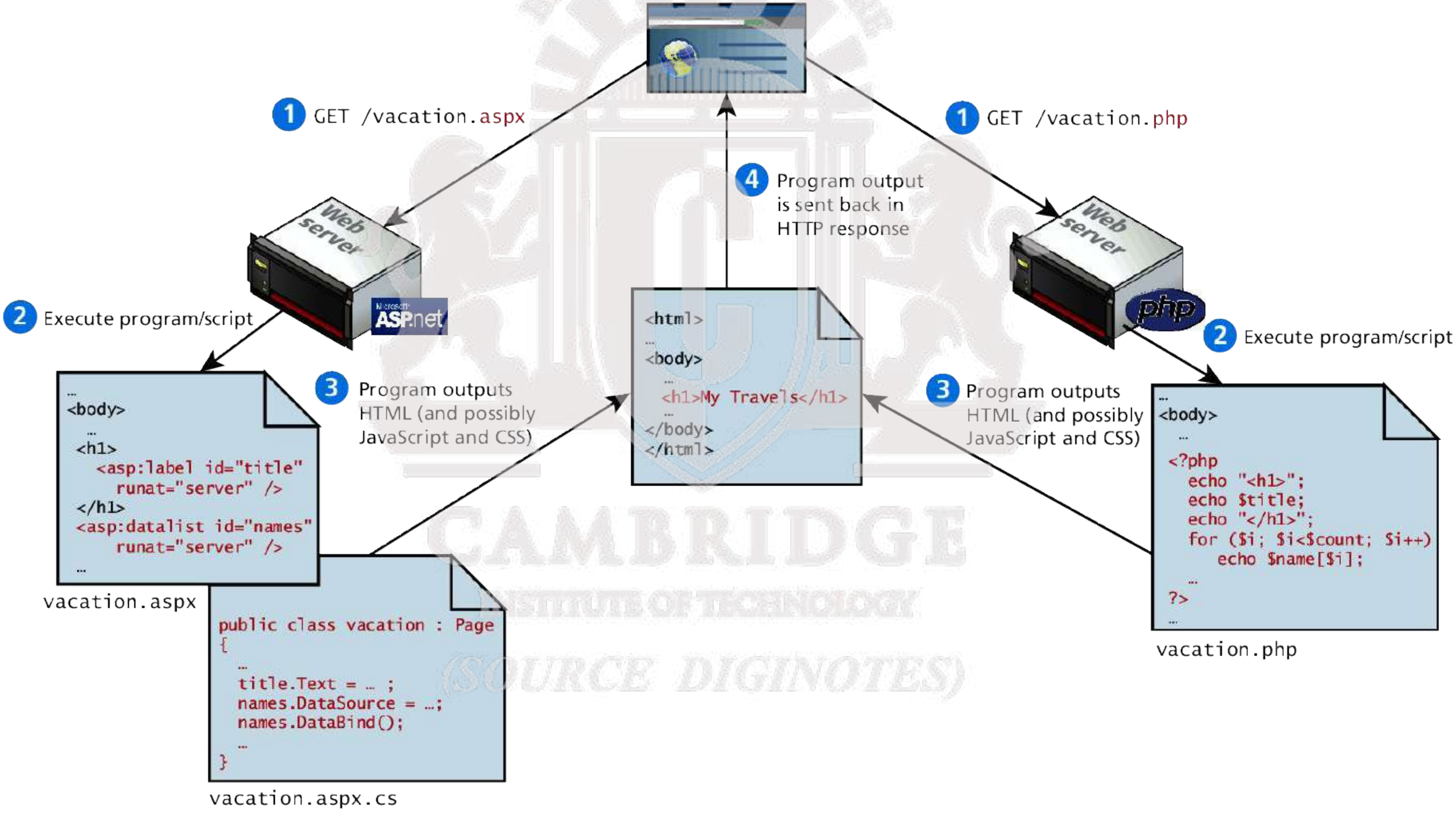
Source diginotes.in

Server-Side Script Resources

So many tools in your kit



Web Development Technologies



1 GET /vacation.aspx
Web server
ASP.net
2 Execute program/script
3 Program outputs HTML (and possibly JavaScript and CSS)
4 Program output is sent back in HTTP response
1 GET /vacation.php
Web server
php
2 Execute program/script
3 Program outputs HTML (and possibly JavaScript and CSS)
vacation.aspx
vacation.aspx.cs
vacation.php

vacation.aspx

```
…
<body>
  …
 <h1>
   <asp:label id="title"
     runat="server" />
 </h1>
 <asp:datalist id="names"
     runat="server" />
 …
```

vacation.aspx.cs

```
public class vacation : Page
{
  …
  title.Text = … ;
  names.DataSource = …;
  names.DataBind();
  …
}
```

```
<html>
…
<body>
  …
  <h1>My Travels</h1>
  …
</body>
</html>
```

vacation.php

```
…
<body>
  …
 <?php
   echo "<h1>";
   echo $title;
   echo "</h1>";
   for ($i; $i<$count; $i++)
      echo $name[$i];
   …
 ?>
 …
```

Comparing Server-Side Technologies

- **ASP (Active Server Pages)**. Like PHP, ASP code (using the VBScript programming language) can be embedded within the HTML. ASP programming code is interpreted at run time, hence it can be slow in comparison to other technologies.
- **ASP.NET**. ASP.NET is part of Microsoft's .NET Framework and can use any .NET programming language (though C# is the most commonly used). ASP.NET uses an explicitly object-oriented approach. It also uses special markup called web server controls that encapsulate common web functionality such as database-driven lists, form validation, and user registration wizards. ASP.NET pages are compiled into an intermediary file format called MSIL that is analogous to Java's byte-code. ASP.NET then uses a Just-In-Time compiler to compile the MSIL into machine executable code so its performance can be excellent. However, ASP.NET is essentially limited to Windows servers.

Comparing Server-Side Technologies

- **JSP (Java Server Pages)**. JSP uses Java as its programming language and like ASP.NET it uses an explicit object-oriented approach and is used in large enterprise web systems and is integrated into the J2EE environment. Since JSP uses the Java Runtime Engine, it also uses a JIT compiler for fast execution time and is cross-platform. While JSP's usage in the web as a whole is small, it has a substantial market share in the intranet environment, as well as with very large and busy sites.

- **Node.js**. This is a more recent server environment that uses JavaScript on the server side, thus allowing developers already familiar with JavaScript to use just a single language for both client-side and server-side development. Unlike the other development technologies listed here, node.js also is its own web server software, thus eliminating the need for Apache, IIS, or some other web server software.

Comparing Server-Side Technologies

- **Perl**. Until the development and popularization of ASP, PHP, and JSP, Perl was the language typically used for early server-side web development. As a language, it excels in the manipulation of text. It was commonly used in conjunction with the **Common Gateway Interface (CGI)**, an early standard API for communication between applications and web server software.

- **PHP**. Like ASP, PHP is a dynamically typed language that can be embedded directly within the HTML, though it now supports most common object-oriented features, such as classes and inheritance. By default, PHP pages are compiled into an intermediary representation called **opcodes** that are analogous to Java's byte-code or the .NET Framework's MSIL. Originally, PHP stood for *personal home pages*, although it now is a recursive acronym that means *PHP: Hypertext Processor*.

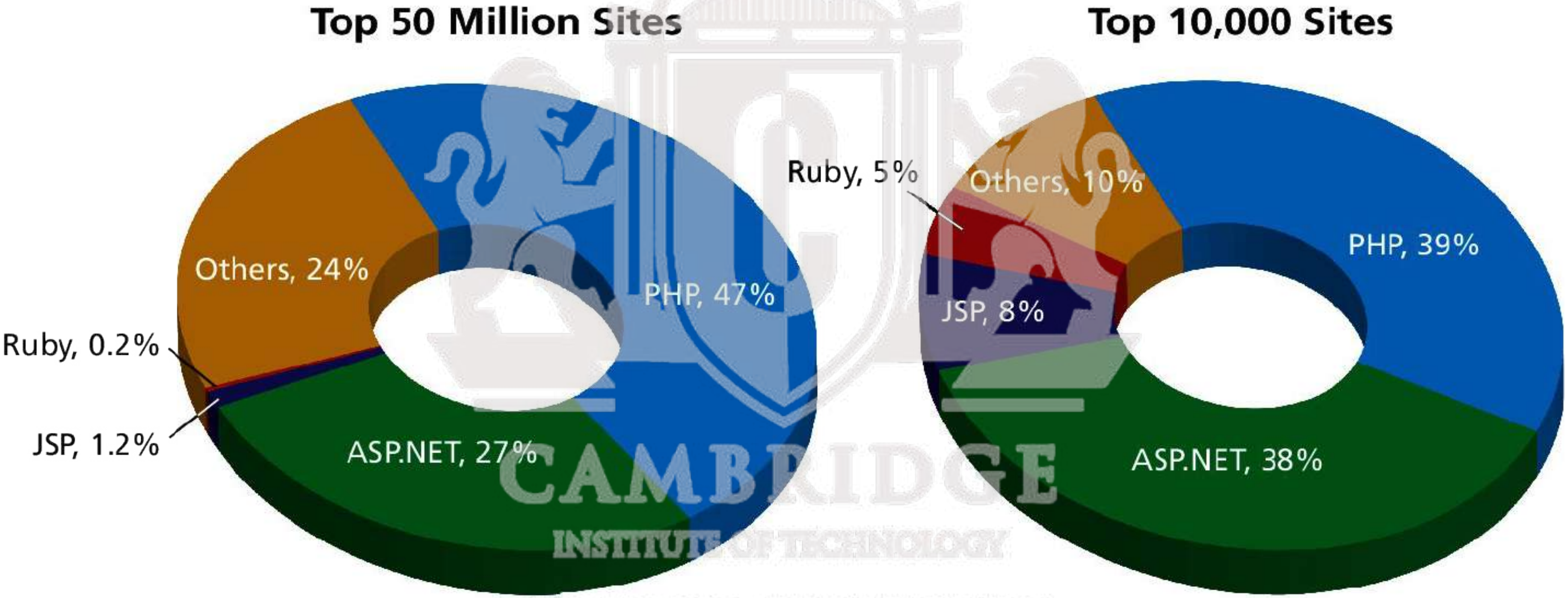
Comparing Server-Side Technologies

- **Python**. This terse, object-oriented programming language has many uses, including being used to create web applications. It is also used in a variety of web development frameworks such as Django and Pyramid.

- **Ruby on Rails**. This is a web development framework that uses the Ruby programming language. Like ASP.NET and JSP, Ruby on Rails emphasizes the use of common software development approaches, in particular the MVC design pattern. It integrates features such as templates and engines that aim to reduce the amount of development work required in the creation of a new site.

Market Share

Of web development environments



Source diginotes.in

Section 2 of 5

WEB SERVER'S RESPONSABILITIES

A Web Server's Responsibilities

A web server has many responsibilities:

- handling HTTP connections
- responding to requests for static and dynamic resources
- managing permissions and access for certain resources
- encrypting and compressing data
- managing multiple domains and URLs
- managing database connections
- managing cookies and state
- uploading and managing files

LAMP stack

WAMP, MAMP, …

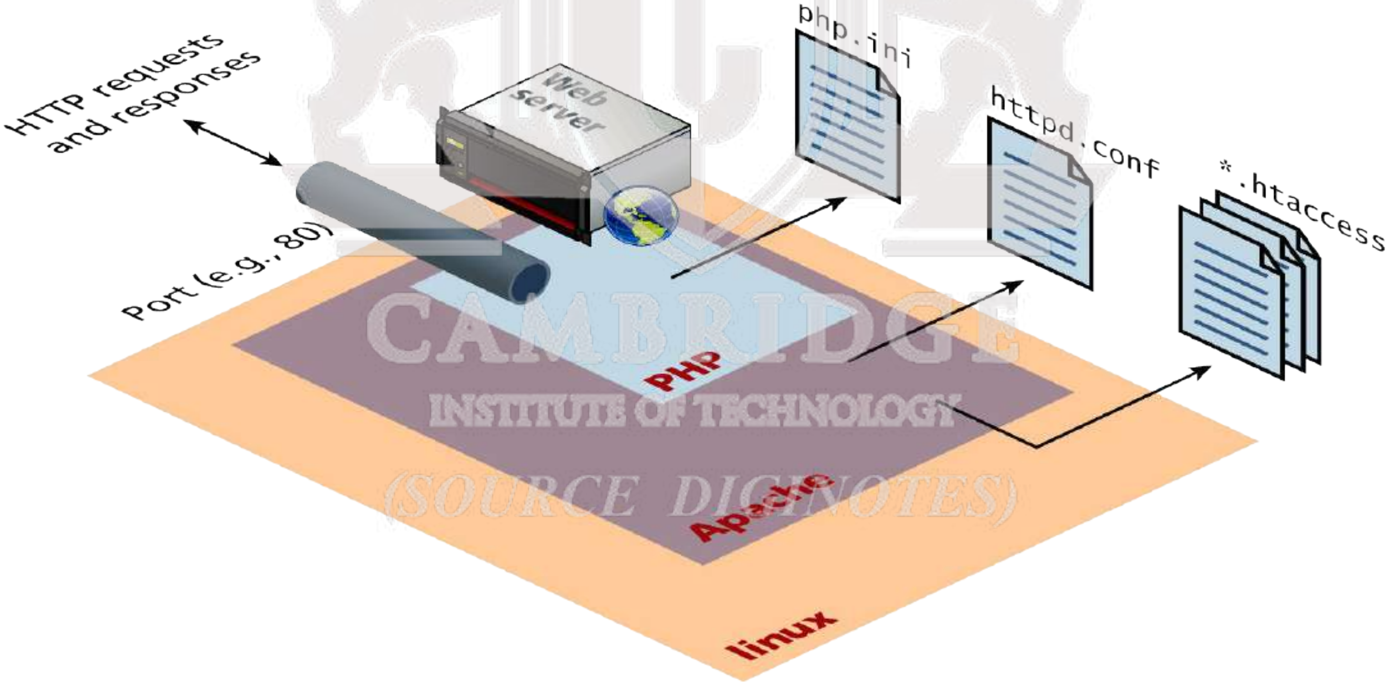
You will be using the LAMP software stack

- Linux operating system
- Apache web server
- MySQL DBMS
- PHP scripting language

Apache and Linux

LA

Consider the **Apache** web server as the intermediary that interprets HTTP requests that arrive through a network port and decides how to handle the request, which often requires working in conjunction with PHP.



Source diginotes.in

Apache

Continued

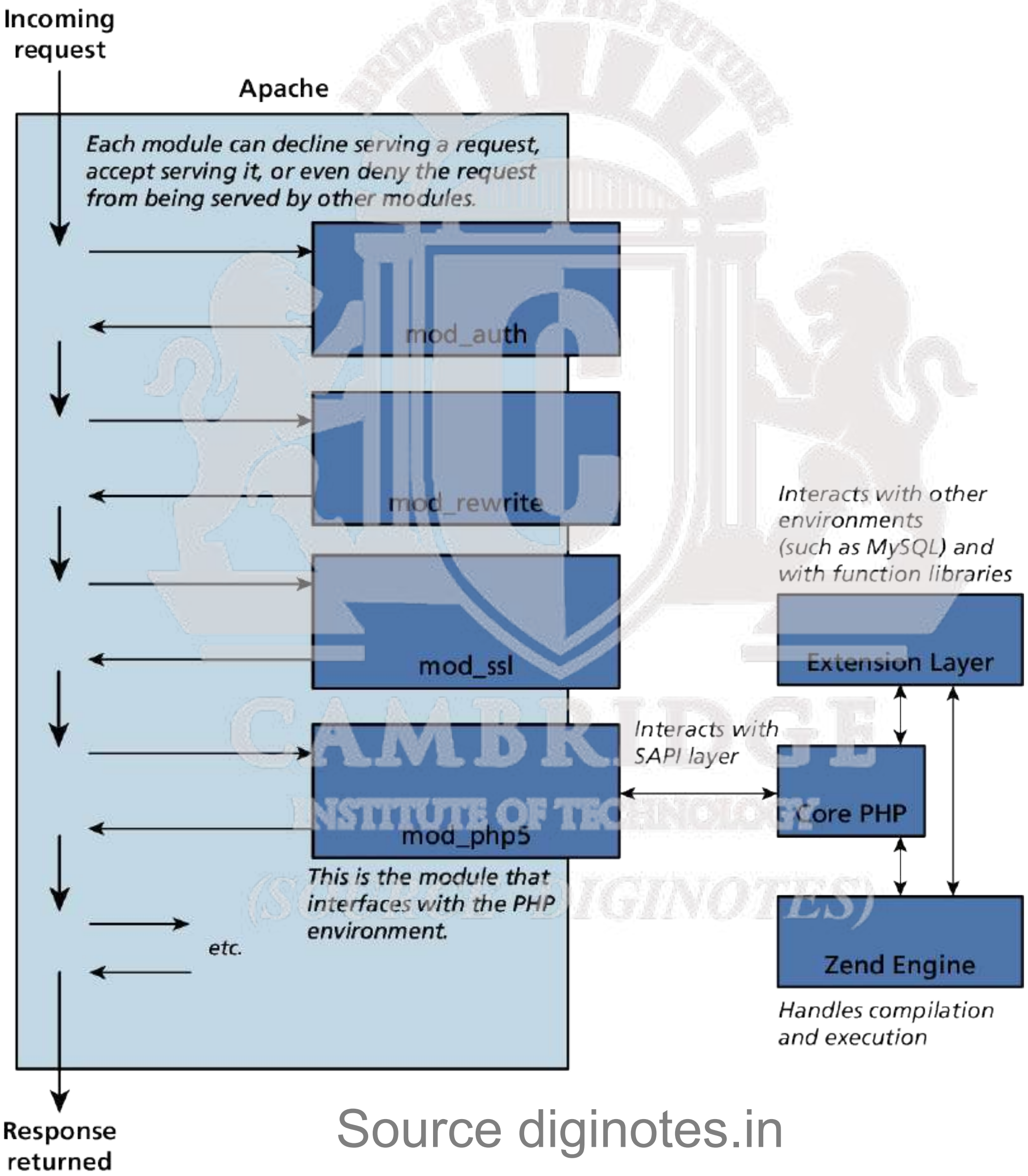
- Apache runs as a daemon on the server. A **daemon** is an executing instance of a program (also called a **process**) that runs in the background, waiting for a specific event that will activate it.

- When a request arrives, Apache then uses modules to determine how to respond to the request.

- In Apache, a **module** is a compiled extension (usually written in the C programming language) to Apache that helps it *handle* requests. For this reason, these modules are also sometimes referred to as **handlers**.

Apache and PHP

PHP Module in Apache



Incoming request
Apache
Each module can decline serving a request, accept serving it, or even deny the request from being served by other modules.
mod_auth
mod_rewrite
mod_ssl
mod_php5
This is the module that interfaces with the PHP environment.
etc.
Interacts with SAPI layer
Interacts with other environments (such as MySQL) and with function libraries
Extension Layer
Core PHP
Zend Engine
Handles compilation and execution
Response returned

Source diginotes.in

Apache Threads

Multi-thread and multi-process

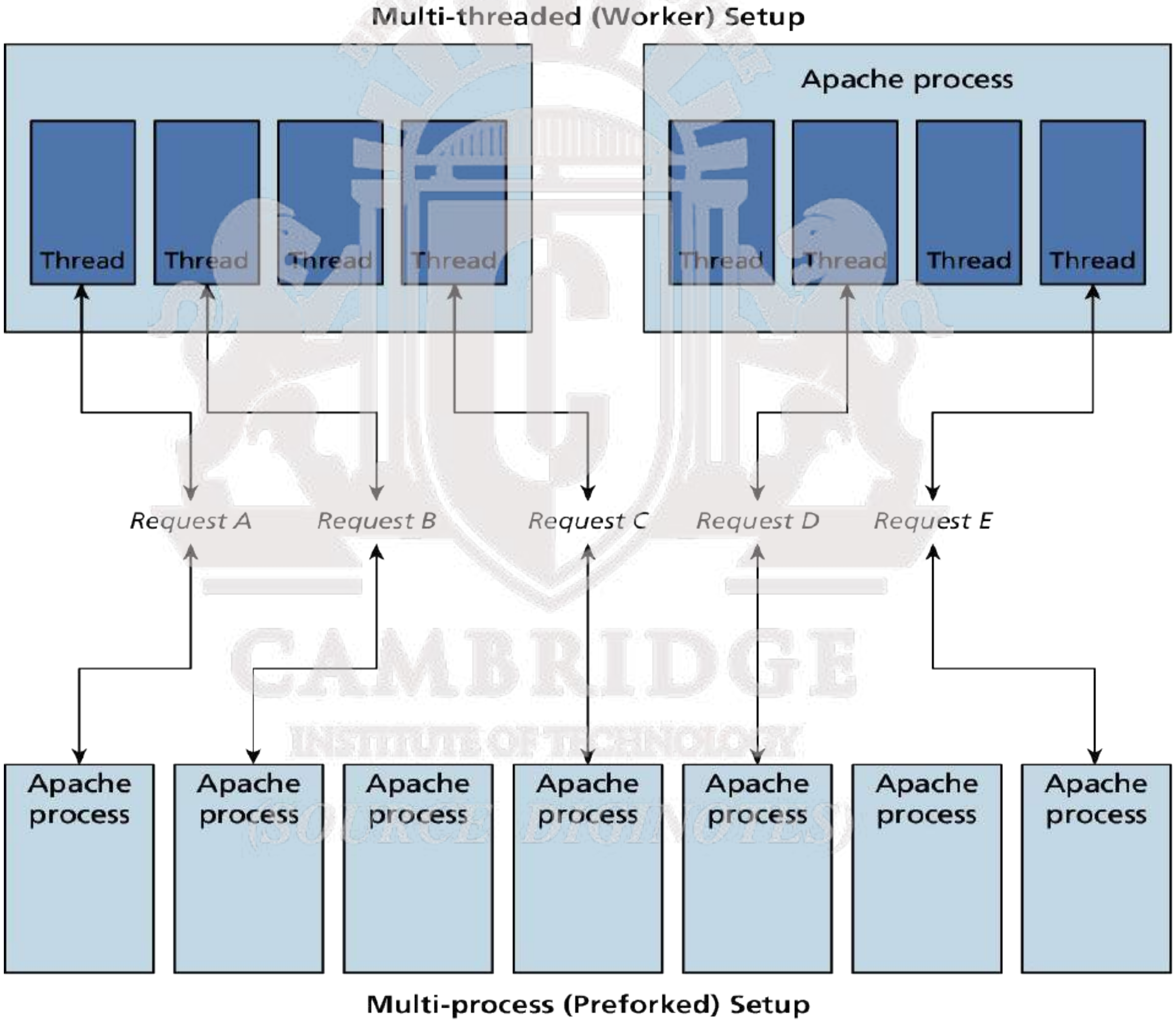
Apache runs in two possible modes:

- **multi-process** (also called **preforked**)
- **multi-threaded** (also called **worker**)

The default installation of Apache runs using the multi-process mode.

Apache Threads

Multi-thread and multi-process



Source diginotes.in

PHP Internals

PHP itself is written in C

There are 3 main modules

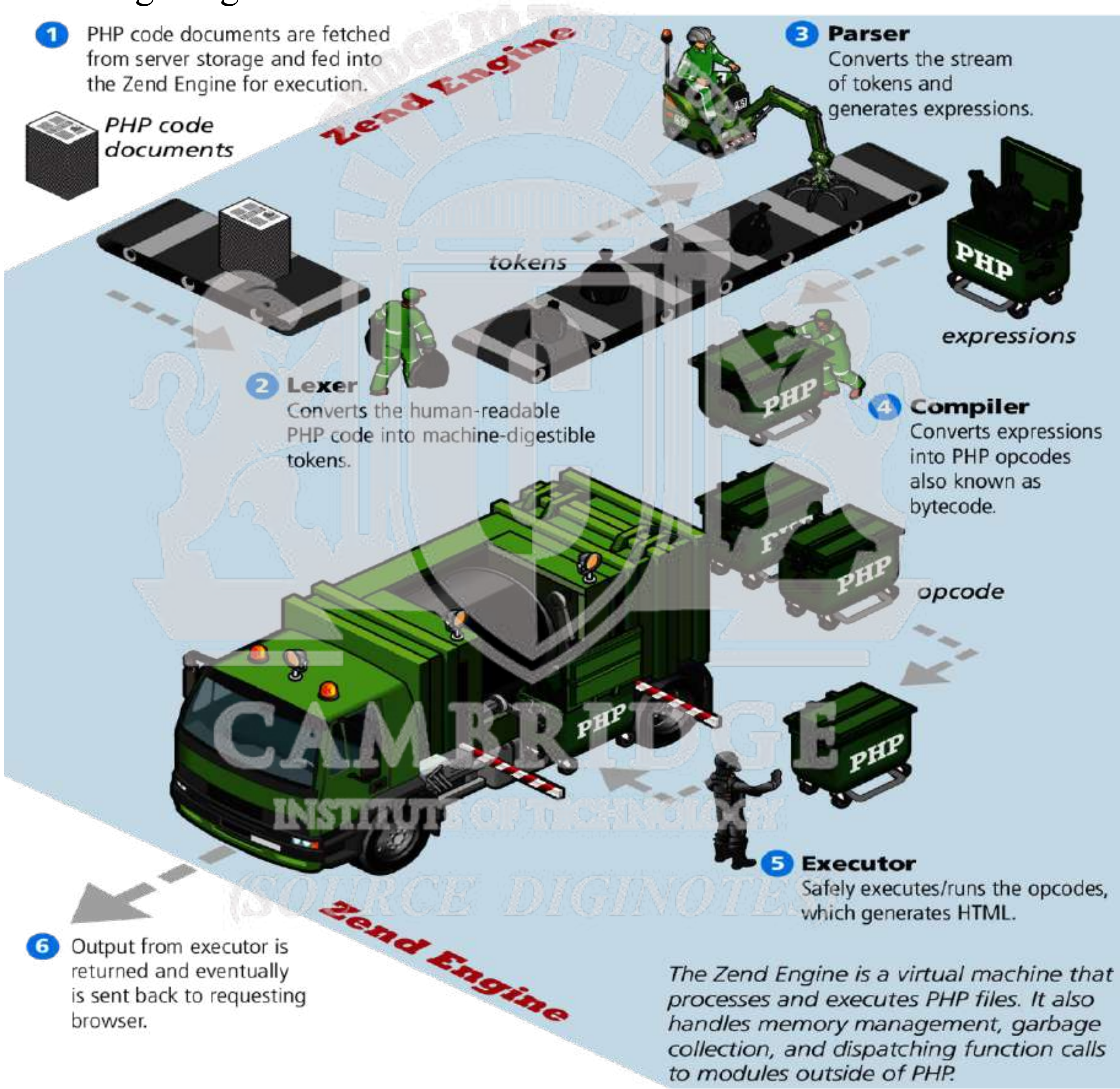
1. **PHP core**. The Core module defines the main features of the PHP environment, including essential functions for variable handling, arrays, strings, classes, math, and other core features.

2. **Extension layer**. This module defines functions for interacting with services outside of PHP. This includes libraries for MySQL, FTP, SOAP web services, and XML processing, among others.

3. **Zend Engine**. This module handles the reading in of a requested PHP file, compiling it, and executing it.

Zend Engine

No, your code is not garbage.



1 PHP code documents are fetched from server storage and fed into the Zend Engine for execution.

2 **Lexer** Converts the human-readable PHP code into machine-digestible tokens.

3 **Parser** Converts the stream of tokens and generates expressions.

4 **Compiler** Converts expressions into PHP opcodes also known as bytecode.

5 **Executor** Safely executes/runs the opcodes, which generates HTML.

6 Output from executor is returned and eventually is sent back to requesting browser.

The Zend Engine is a virtual machine that processes and executes PHP files. It also handles memory management, garbage collection, and dispatching function calls to modules outside of PHP.

Installing LAMP locally

Turn this key

❖The easiest and quickest way to do so is to use the

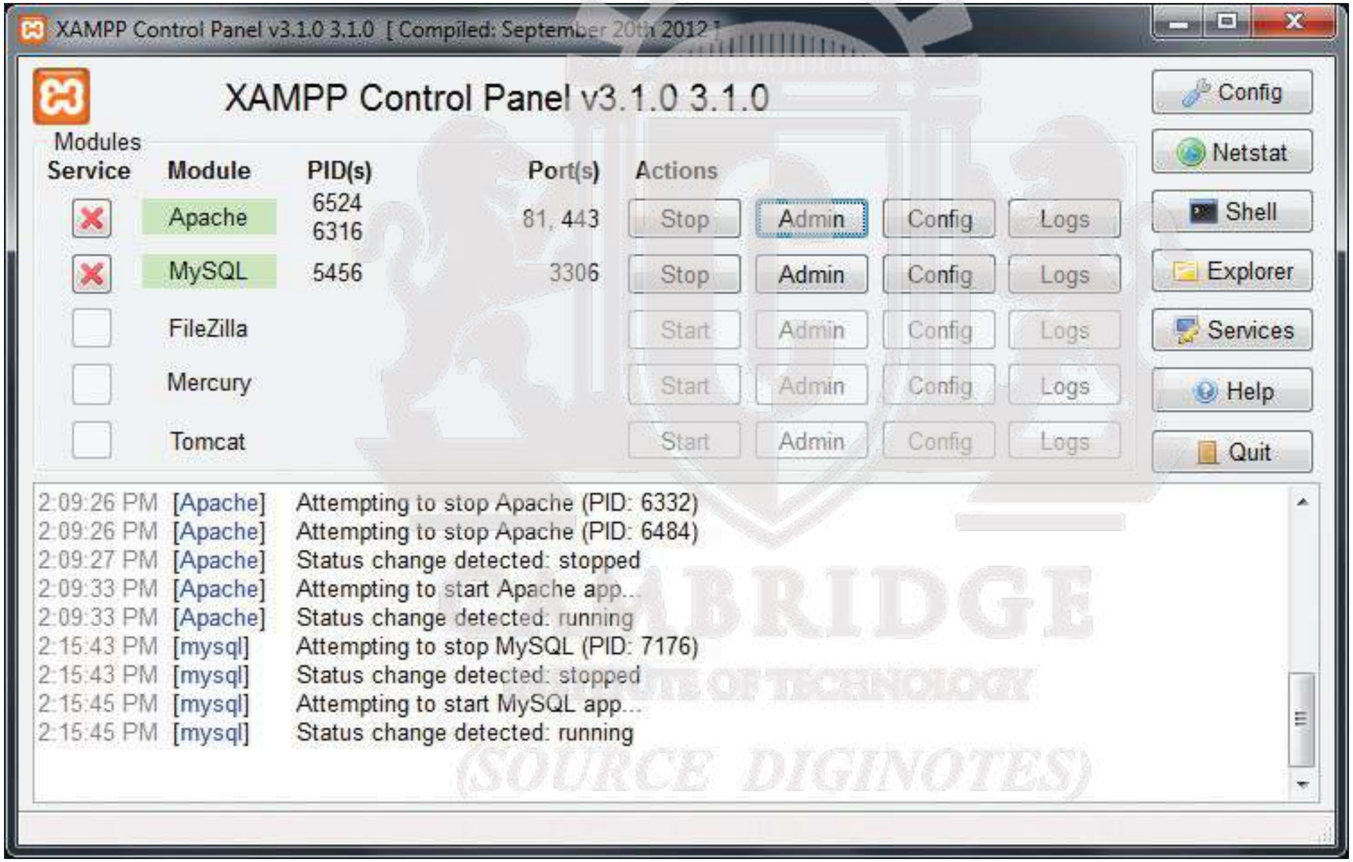
- **XAMPP** For Windows installation package
- **MAMP** for Mac installation package

❖Both of these installation packages install and configure Apache, PHP, and MySQL.

❖Later we can come back and configure these systems in more detail.

XAMPP Control Panel

Turn this key



XAMPP Control Panel v3.1.0 3.1.0 [Compiled: September 20th 2012]

XAMPP Control Panel v3.1.0 3.1.0

Modules

Service	Module	PID(s)	Port(s)	Actions			
☒	Apache	6524 6316	81, 443	Stop	Admin	Config	Logs
☒	MySQL	5456	3306	Stop	Admin	Config	Logs
□	FileZilla			Start	Admin	Config	Logs
□	Mercury			Start	Admin	Config	Logs
□	Tomcat			Start	Admin	Config	Logs

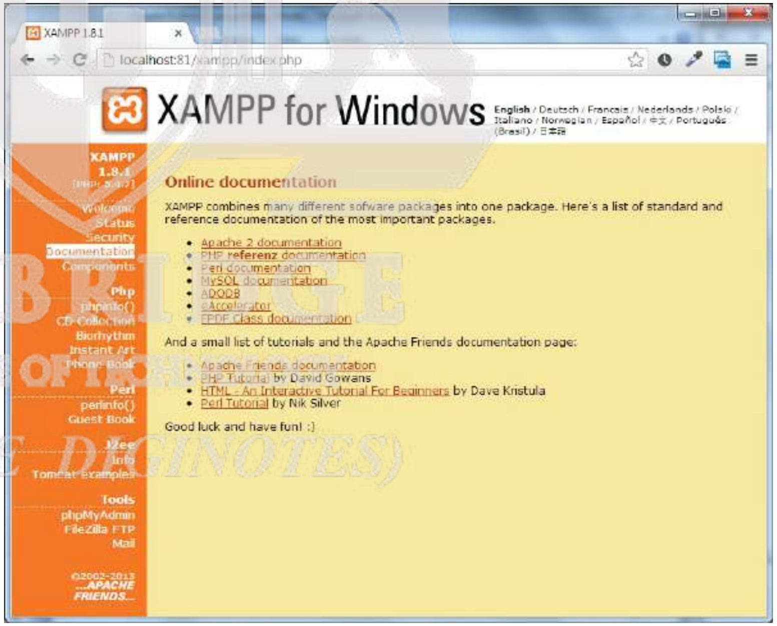
```
2:09:26 PM  [Apache]   Attempting to stop Apache (PID: 6332)
2:09:26 PM  [Apache]   Attempting to stop Apache (PID: 6484)
2:09:27 PM  [Apache]   Status change detected: stopped
2:09:33 PM  [Apache]   Attempting to start Apache app...
2:09:33 PM  [Apache]   Status change detected: running
2:15:43 PM  [mysql]    Attempting to stop MySQL (PID: 7176)
2:15:43 PM  [mysql]    Status change detected: stopped
2:15:45 PM  [mysql]    Attempting to start MySQL app...
2:15:45 PM  [mysql]    Status change detected: running
```

Source diginotes.in

XAMPP Settings

Defaults are

- PHP requests in your browser will need to use the **localhost** domain (127.0.0.1)
- PHP files will have to be saved somewhere within the **C:\xampp\htdocs** folder



Online documentation

XAMPP combines many different sofware packages into one package. Here's a list of standard and reference documentation of the most important packages.

- Apache 2 documentation
- PHP **referenz** documentation
- Perl documentation
- MySQL documentation
- ADODB
- eAccelerator
- FPDF Class documentation

And a small list of tutorials and the Apache Friends documentation page:

- Apache Friends documentation
- PHP Tutorial by David Gowans
- HTML - An Interactive Tutorial For Beginners by Dave Kristula
- Perl Tutorial by Nik Silver

Good luck and have fun! :)

Section 3 of 5

QUICK TOUR OF PHP

Quick Tour

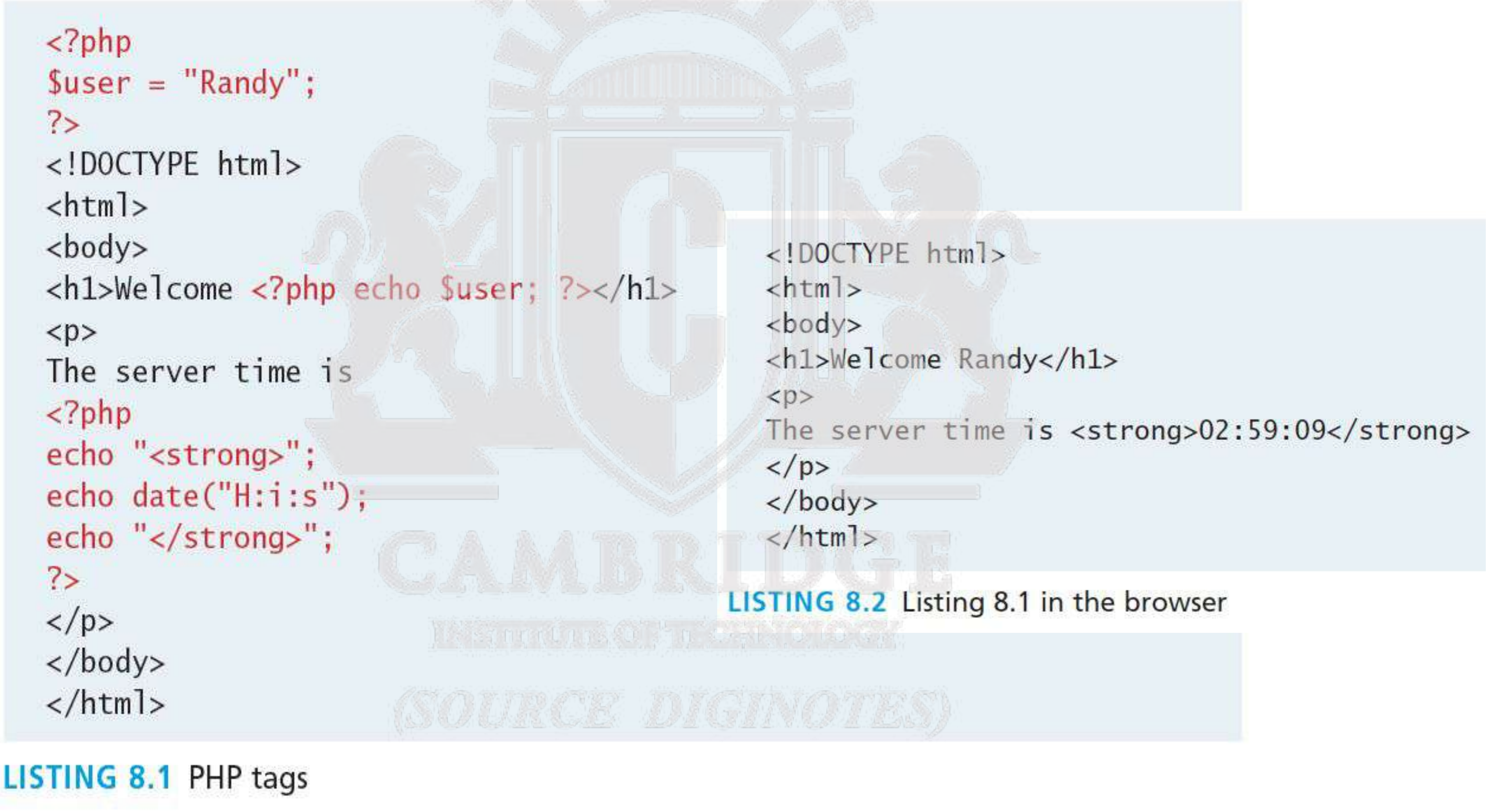
- PHP, like JavaScript, is a dynamically typed language.
- it uses classes and functions in a way consistent with other object-oriented languages such as C++, C#, and Java
- The syntax for loops, conditionals, and assignment is identical to JavaScript
- Differs when you get to functions, classes, and in how you define variables

PHP Tags

The most important fact about PHP is that the programming code can be embedded directly within an HTML file.

- A PHP file will usually have the extension **.php**
- programming code must be contained within an opening **<?php** tag and a matching closing **?>** tag
- any code outside the tags is echoed directly out to the client

PHP Tags



```
<?php
$user = "Randy";
?>
<!DOCTYPE html>
<html>
<body>
<h1>Welcome <?php echo $user; ?></h1>
<p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p>
</body>
</html>
```

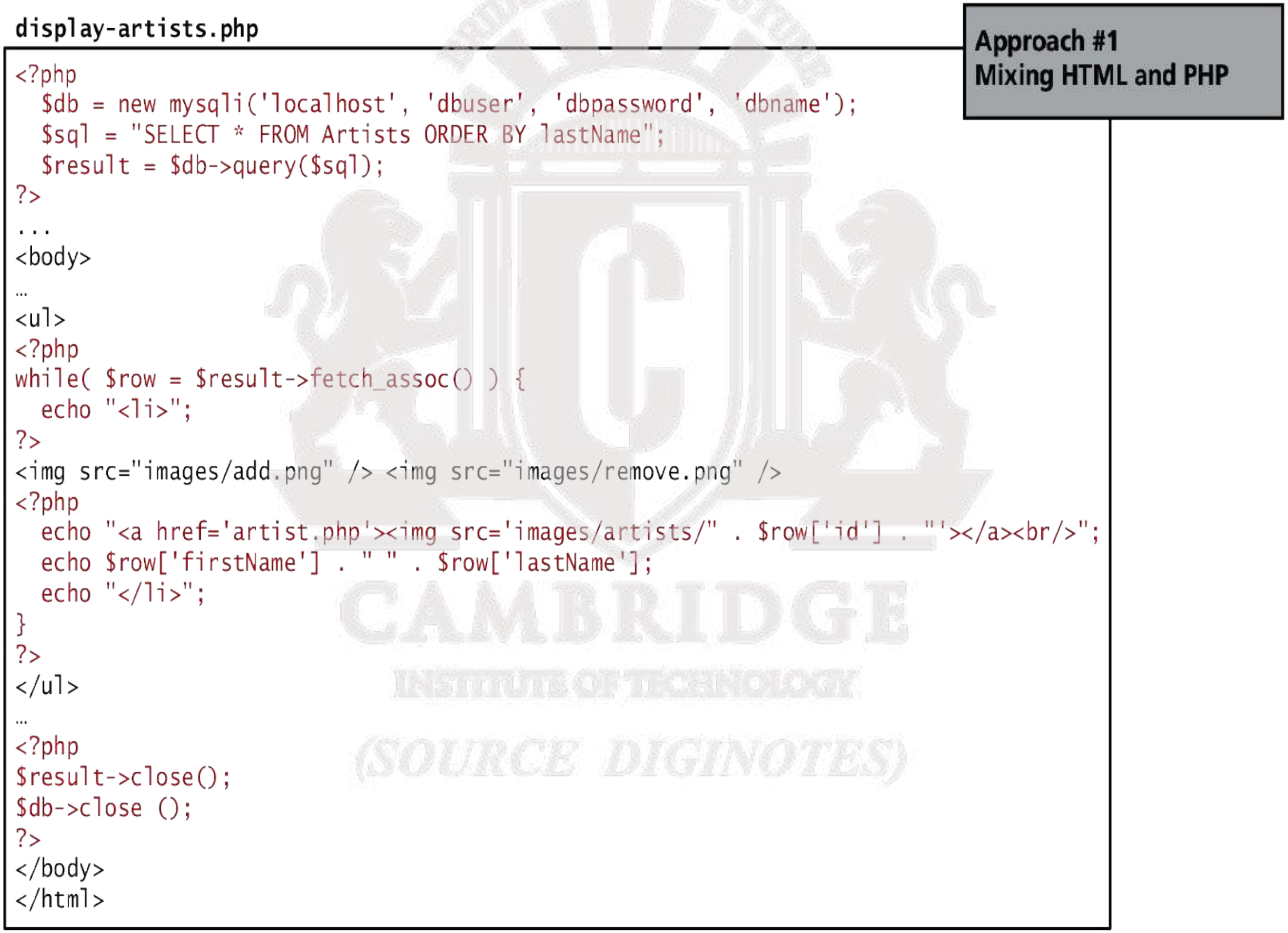
LISTING 8.1 PHP tags

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

LISTING 8.2 Listing 8.1 in the browser

HTML and PHP

Two approaches



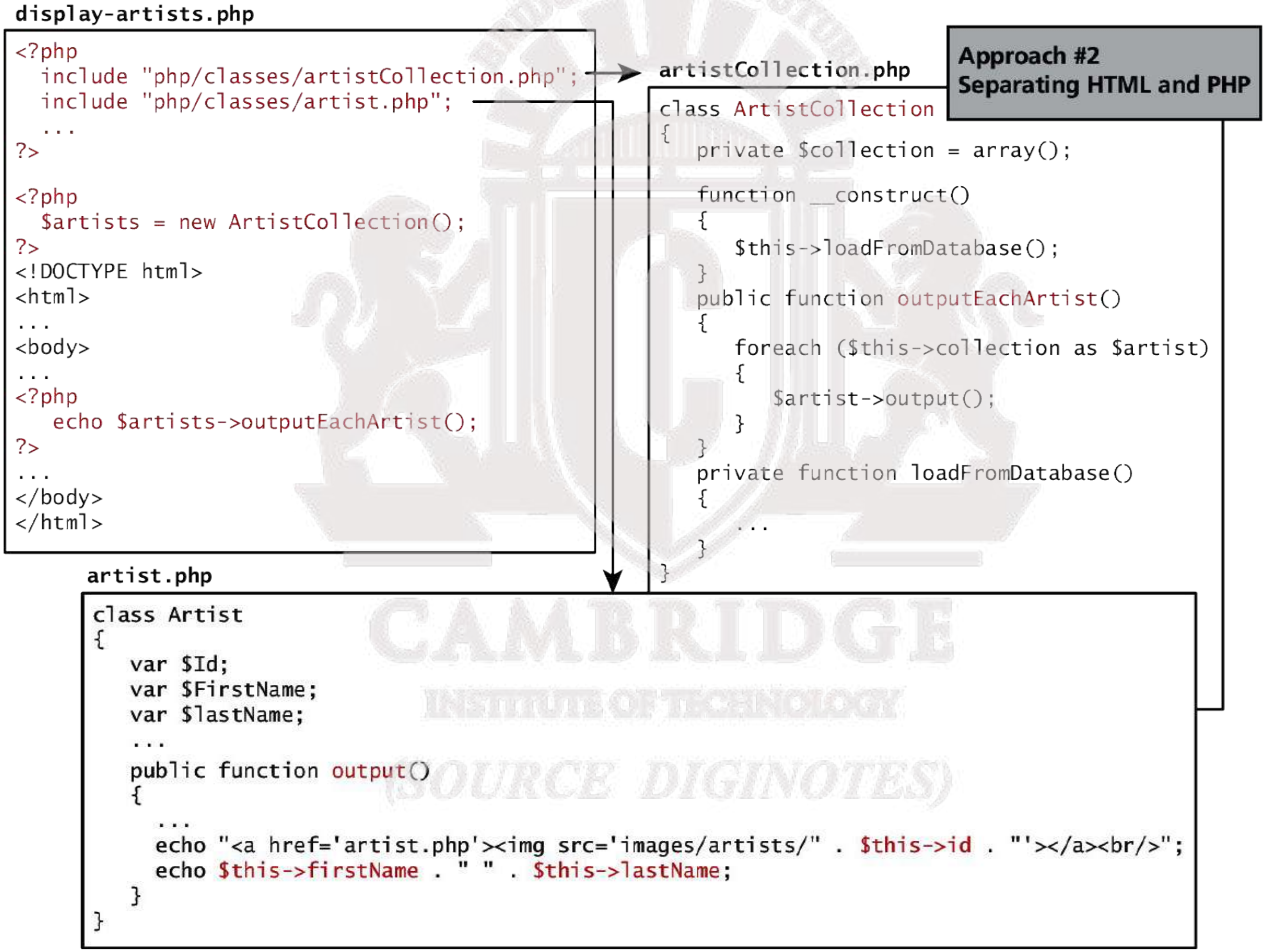
display-artists.php

Approach #1
Mixing HTML and PHP

```
<?php
  $db = new mysqli('localhost', 'dbuser', 'dbpassword', 'dbname');
  $sql = "SELECT * FROM Artists ORDER BY lastName";
  $result = $db->query($sql);
?>
. . .
<body>
…
<ul>
<?php
while( $row = $result->fetch_assoc() ) {
  echo "<li>";
?>
<img src="images/add.png" /> <img src="images/remove.png" />
<?php
  echo "<a href='artist.php'><img src='images/artists/" . $row['id'] . "'></a><br/>";
  echo $row['firstName'] . " " . $row['lastName'];
  echo "</li>";
}
?>
</ul>
…
<?php
$result->close();
$db->close ();
?>
</body>
</html>
```

HTML and PHP

Two approaches



Approach #2
Separating HTML and PHP

display-artists.php

```
<?php
  include "php/classes/artistCollection.php";
  include "php/classes/artist.php";
  ...
?>

<?php
  $artists = new ArtistCollection();
?>
<!DOCTYPE html>
<html>
...
<body>
...
<?php
   echo $artists->outputEachArtist();
?>
...
</body>
</html>
```

artistCollection.php

```
class ArtistCollection
{
   private $collection = array();

   function __construct()
   {
      $this->loadFromDatabase();
   }
   public function outputEachArtist()
   {
      foreach ($this->collection as $artist)
      {
         $artist->output();
      }
   }
   private function loadFromDatabase()
   {
      ...
   }
}
```

artist.php

```
class Artist
{
   var $Id;
   var $FirstName;
   var $lastName;
   ...
   public function output()
   {
     ...
     echo "<a href='artist.php'><img src='images/artists/" . $this->id . "'></a><br/>";
     echo $this->firstName . " " . $this->lastName;
   }
}
```

PHP Comments

3 kinds

The types of comment styles in PHP are:

- **Single-line comments**. Lines that begin with a # are comment lines and will not be executed.
- **Multiline (block) comments**. These comments begin with a /* and encompass everything that is encountered until a closing */ tag is found.
- **End-of-line comments**. Whenever // is encountered in code, everything up to the end of the line is considered a comment.

PHP Comments

3 kinds

```
<?php

# single-line comment

/*

This is a multiline comment.

They are a good way to document functions or complicated
blocks of code

*/

$artist = readDatabase(); // end-of-line comment

?>
```

Variables

- Variables in PHP are **dynamically typed.**
- Variables are also **loosely typed** in that a variable can be assigned different data types over time
- To declare a variable you must preface the variable name with the dollar ($) symbol.

$count = 42;

Data Types

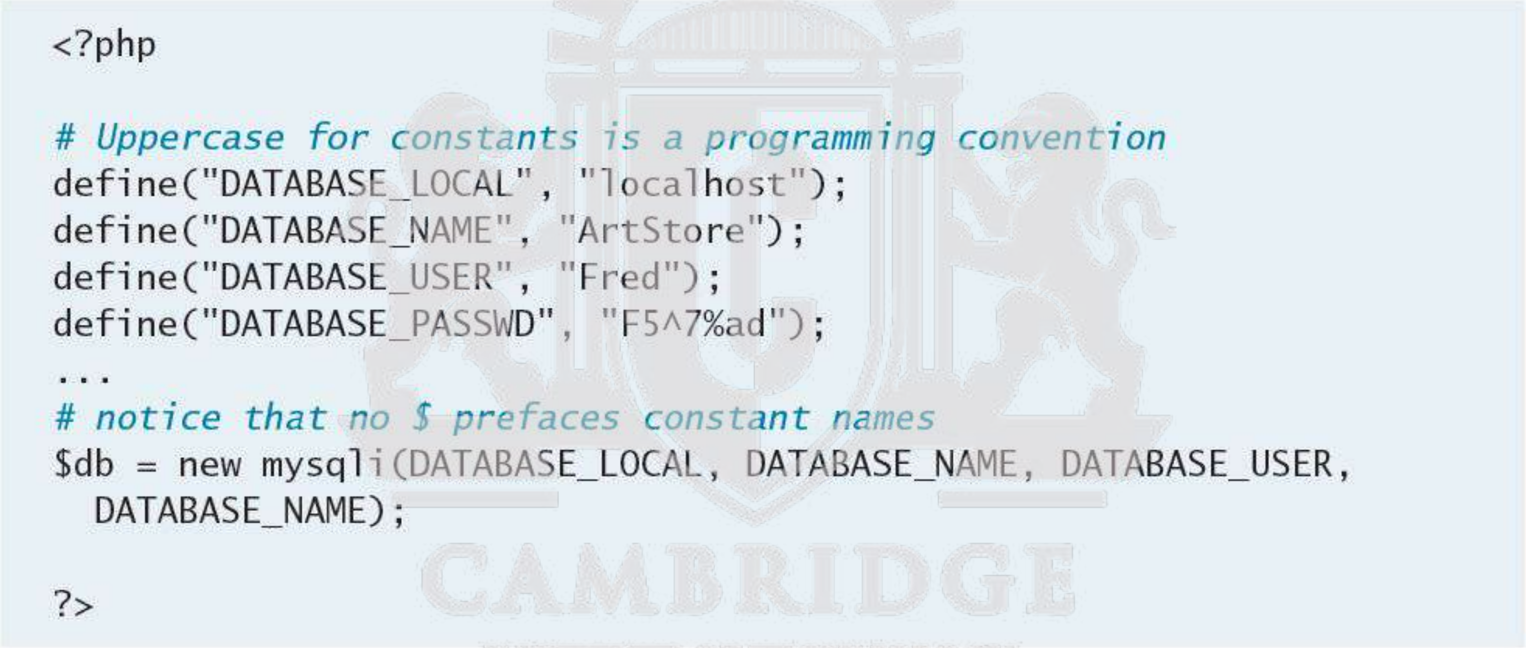
Data Type	Description
Boolean	A logical true or false value
Integer	Whole numbers
Float	Decimal numbers
String	Letters
Array	A collection of data of any type (covered in the next chapter)
Object	Instances of classes

Constants

A **constant** is somewhat similar to a variable, except a constant's value never changes . . . in other words it stays constant.

- Typically defined near the top of a PHP file via the **define()** function
- once it is defined, it can be referenced without using the $ symbol

Constants



```
<?php

# Uppercase for constants is a programming convention
define("DATABASE_LOCAL", "localhost");
define("DATABASE_NAME", "ArtStore");
define("DATABASE_USER", "Fred");
define("DATABASE_PASSWD", "F5^7%ad");
...
# notice that no $ prefaces constant names
$db = new mysqli(DATABASE_LOCAL, DATABASE_NAME, DATABASE_USER,
  DATABASE_NAME);

?>
```

LISTING 8.4 PHP constants

Writing to Output

Hello World

To output something that will be seen by the browser, you can use the echo() function.

echo ("hello"); //long form

echo "hello"; //shortcut

String Concatenation

Easy

Strings can easily be appended together using the concatenate operator, which is the period (.) symbol.

```
$username = ”World";

echo "Hello". $username;
```

Will Output **Hello World**

String Concatenation

Example

```
$firstName = "Pablo";

$lastName = "Picasso";

/*

Example one:

These two lines are equivalent. Notice that you can reference PHP
variables within a string literal defined with double quotes. The
resulting output for both lines is: <em>Pablo Picasso</em>

*/

echo "<em>" . $firstName . " ". $lastName. "</em>";

echo "<em> $firstName $lastName </em>";
```

String Concatenation

Example

```
/*

Example two:

These two lines are also equivalent. Notice that you can use either the single quote symbol or double quote symbol for string literals.

*/

echo "<h1>";

echo '<h1>';
```

String Concatenation

Example

*/**

Example three:

These two lines are also equivalent. In the second example, the escape character (the backslash) is used to embed a double quote within a string literal defined within double quotes.

**/*

```
echo '<img src="23.jpg" >';

echo "<img src=\"23.jpg\" >";
```

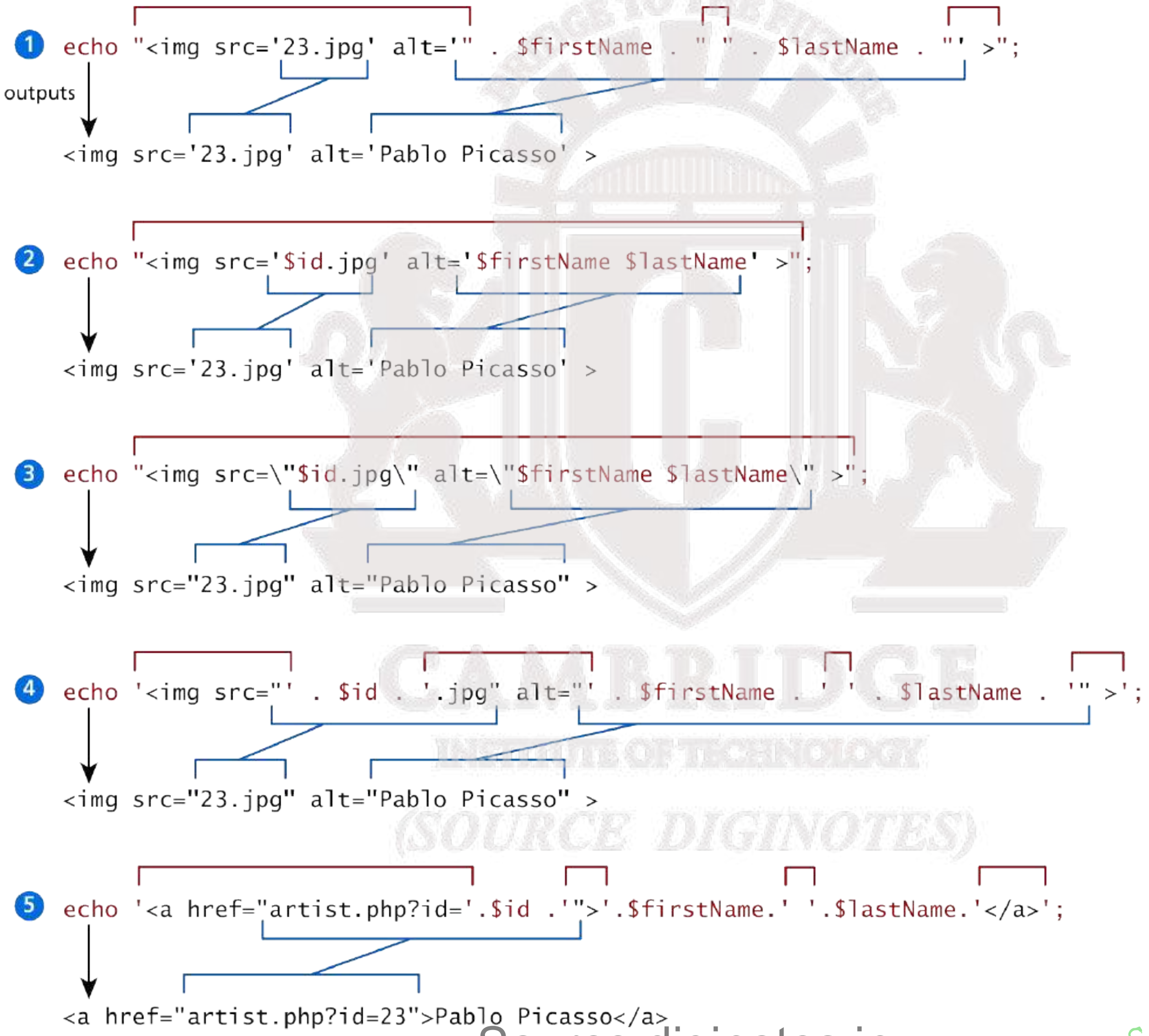
String escape Sequences

Sequence	Description
\n	Line feed
\t	Horizontal tab
\\	Backslash
\$	Dollar sign
\"	Double quote

Complicated Concatenation

```
echo "<img src='23.jpg' alt='". $firstName . " ". $lastName . "' >";

echo "<img src='$id.jpg' alt='$firstName $lastName' >";

echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\" >";

echo '<img src="' . $id. '.jpg" alt="' . $firstName . ' ' . $lastName . '" >';

echo '<a href="artist.php?id=' .$id .'">' .$firstName .' ' . $lastName .'</a>';
```

Illustrated Example



```
1 echo "<img src='23.jpg' alt='" . $firstName . " " . $lastName . "' >";

<img src='23.jpg' alt='Pablo Picasso' >

2 echo "<img src='$id.jpg' alt='$firstName $lastName' >";

<img src='23.jpg' alt='Pablo Picasso' >

3 echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\" >";

<img src="23.jpg" alt="Pablo Picasso" >

4 echo '<img src="' . $id . '.jpg" alt="' . $firstName . ' ' . $lastName . '" >';

<img src="23.jpg" alt="Pablo Picasso" >

5 echo '<a href="artist.php?id='.$id .'">'.$firstName.' '.$lastName.'</a>';

<a href="artist.php?id=23">Pablo Picasso</a>
```

PrintF

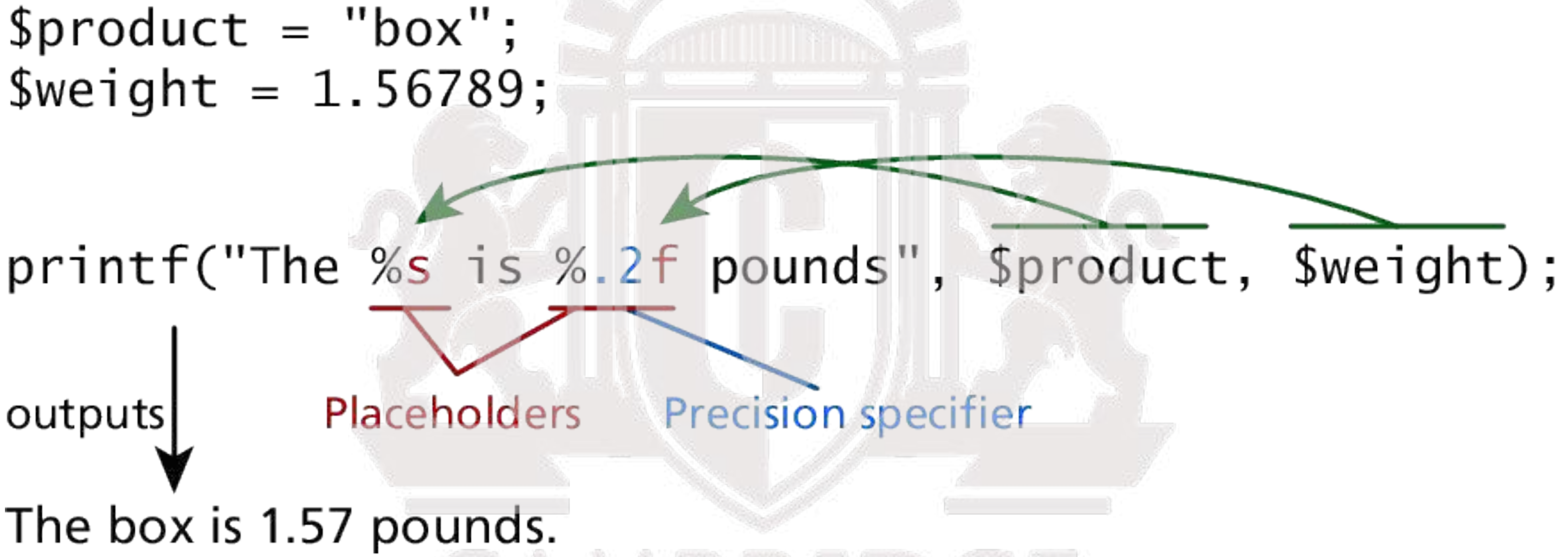
Good ol' printf

As an alternative, you can use the **printf()** function.

- derived from the same-named function in the C programming language
- includes variations to print to string and files (sprintf, fprintf)
- takes at least one parameter, which is a string, and that string optionally references parameters, which are then integrated into the first string by placeholder substitution
- Can also apply special formatting, for instance, specific date/time formats or number of decimal places

PrintF

Illustrated example



```
$product = "box";
$weight = 1.56789;

printf("The %s is %.2f pounds", $product, $weight);
```

outputs

Placeholders

Precision specifier

```
The box is 1.57 pounds.
```

PrintF

Type specifiers

Each placeholder requires the percent (%) symbol in the first parameter string followed by a type specifier.

- b for binary
- d for signed integer
- f for float
- o for octal
- x for hexadecimal

PrintF

Precision

- Precision allows for control over how many decimal places are shown. Important for displaying calculated numbers to the user in a “pretty” way.

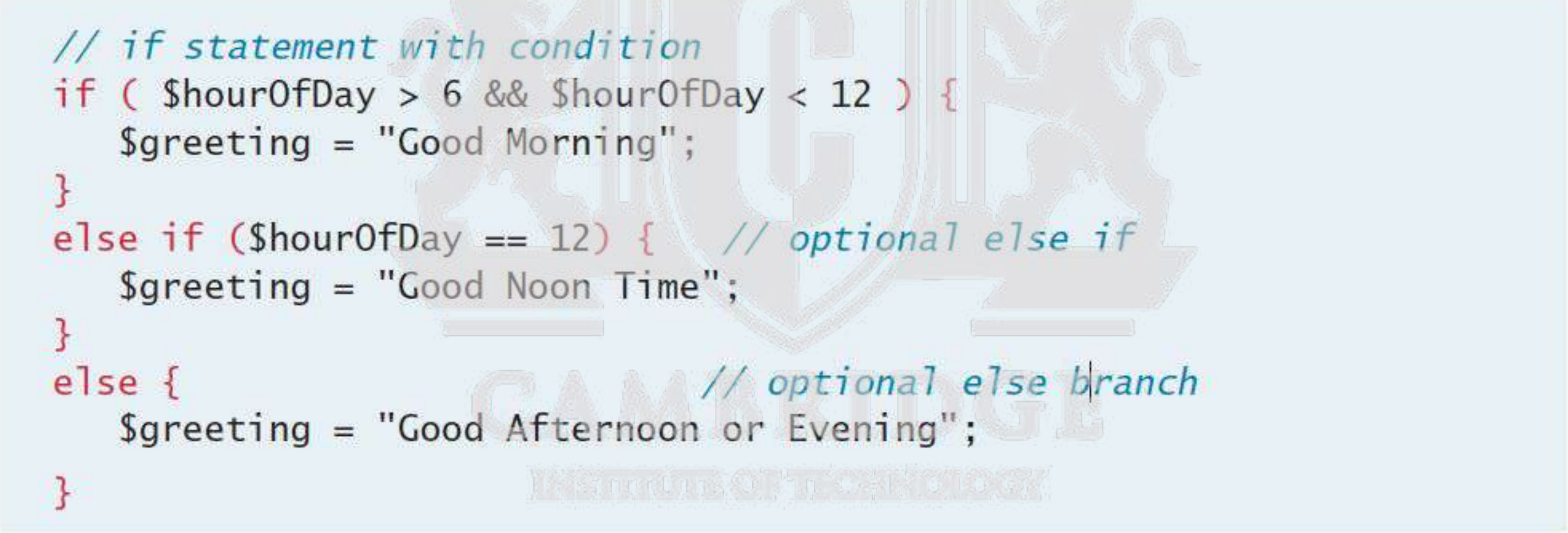
- Precision is achieved in the string with a period (.) followed by a number specifying how many digits should be displayed for floating-point numbers.

Section 4 of 5

PROGRAM CONTROL

If…else

The syntax for conditionals in PHP is almost identical to that of JavaScript

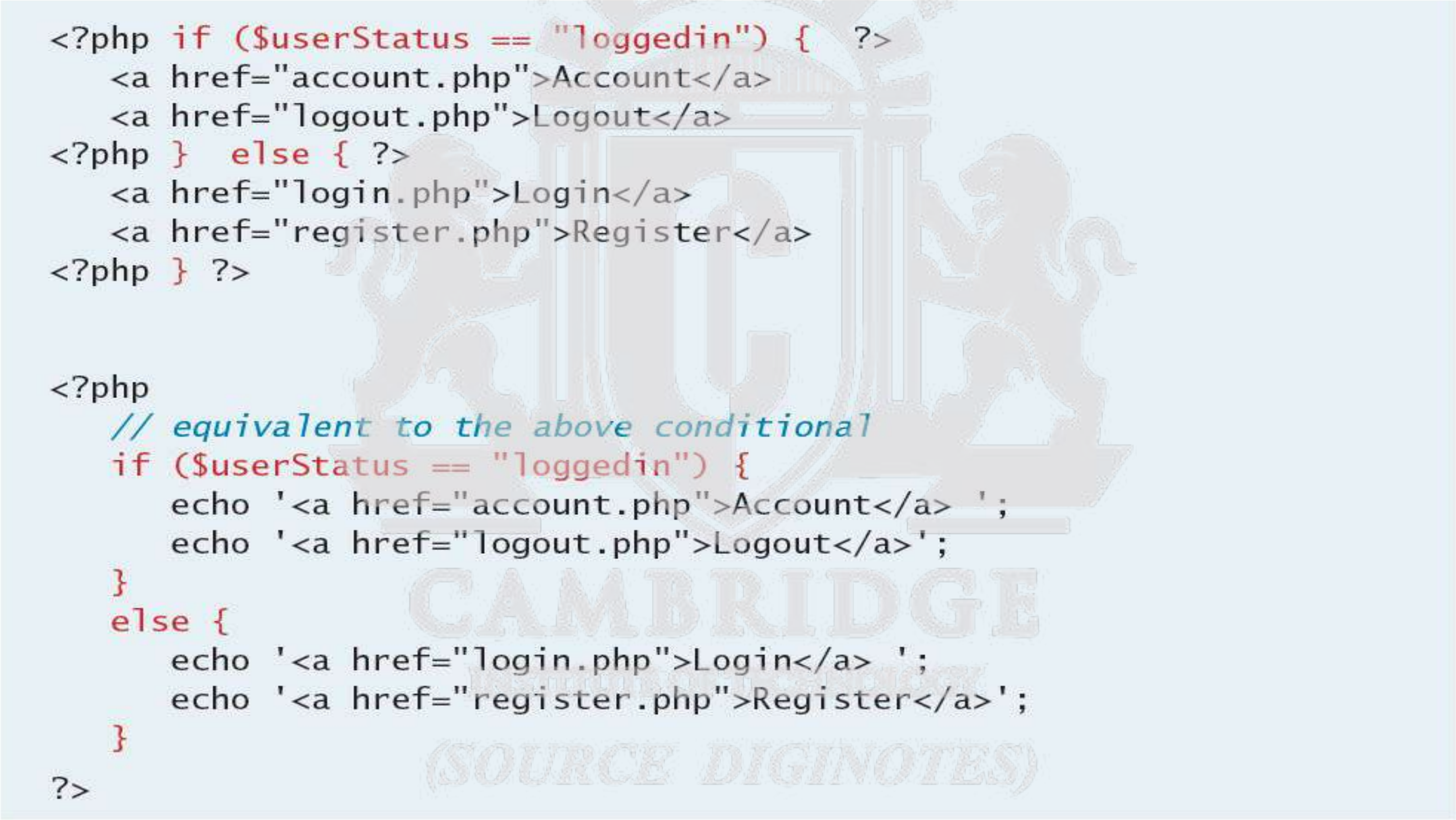


```
// if statement with condition
if ( $hourOfDay > 6 && $hourOfDay < 12 ) {
   $greeting = "Good Morning";
}
else if ($hourOfDay == 12) {   // optional else if
   $greeting = "Good Noon Time";
}
else {                        // optional else branch
   $greeting = "Good Afternoon or Evening";
}
```

LISTING 8.7 Conditional statement using if . . . else

If…else

Alternate syntax

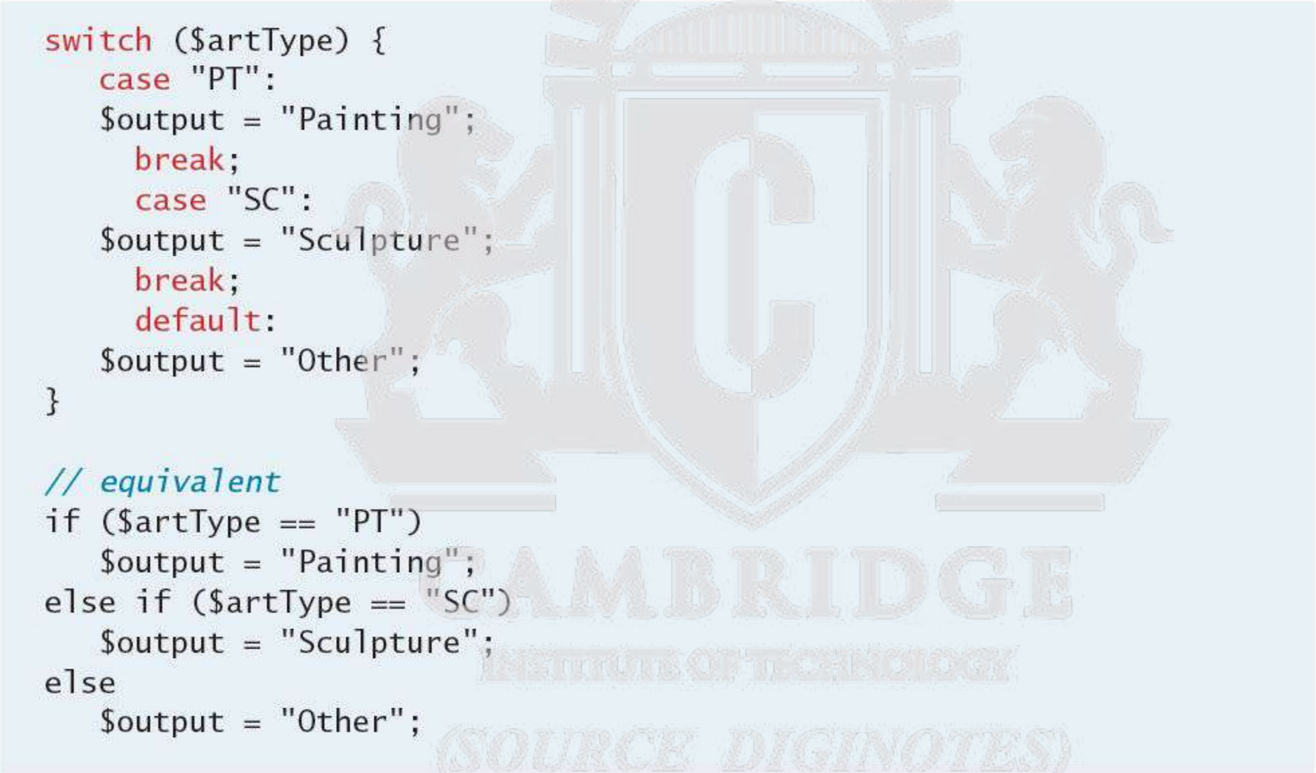


```
<?php if ($userStatus == "loggedin") {  ?>
   <a href="account.php">Account</a>
   <a href="logout.php">Logout</a>
<?php }  else { ?>
   <a href="login.php">Login</a>
   <a href="register.php">Register</a>
<?php } ?>


<?php
   // equivalent to the above conditional
   if ($userStatus == "loggedin") {
      echo '<a href="account.php">Account</a> ';
      echo '<a href="logout.php">Logout</a>';
   }
   else {
      echo '<a href="login.php">Login</a> ';
      echo '<a href="register.php">Register</a>';
   }
?>
```

LISTING 8.8 Combining PHP and HTML in the same script

Switch…case

Nearly identical

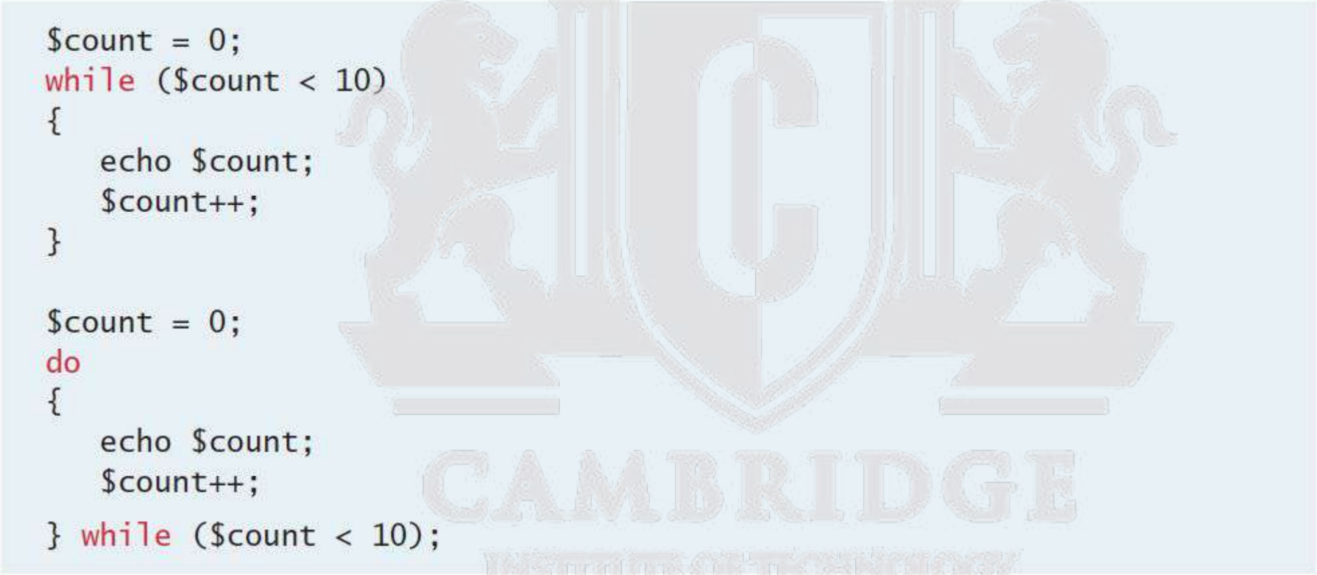


```
switch ($artType) {
   case "PT":
   $output = "Painting";
     break;
     case "SC":
   $output = "Sculpture";
     break;
     default:
   $output = "Other";
}

// equivalent
if ($artType == "PT")
   $output = "Painting";
else if ($artType == "SC")
   $output = "Sculpture";
else
   $output = "Other";
```

LISTING 8.9 Conditional statement using switch

While and Do..while

Identical to other languages

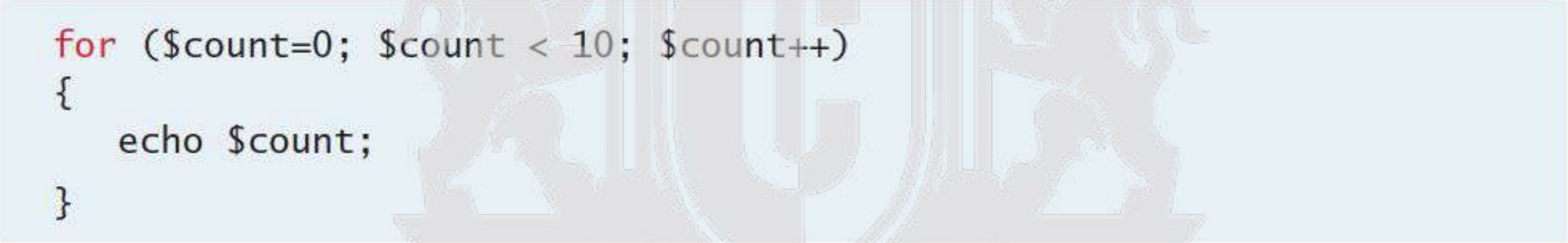


```
$count = 0;
while ($count < 10)
{
    echo $count;
    $count++;
}

$count = 0;
do
{
    echo $count;
    $count++;
} while ($count < 10);
```

LISTING 8.10 while loops

For

Identical to other languages



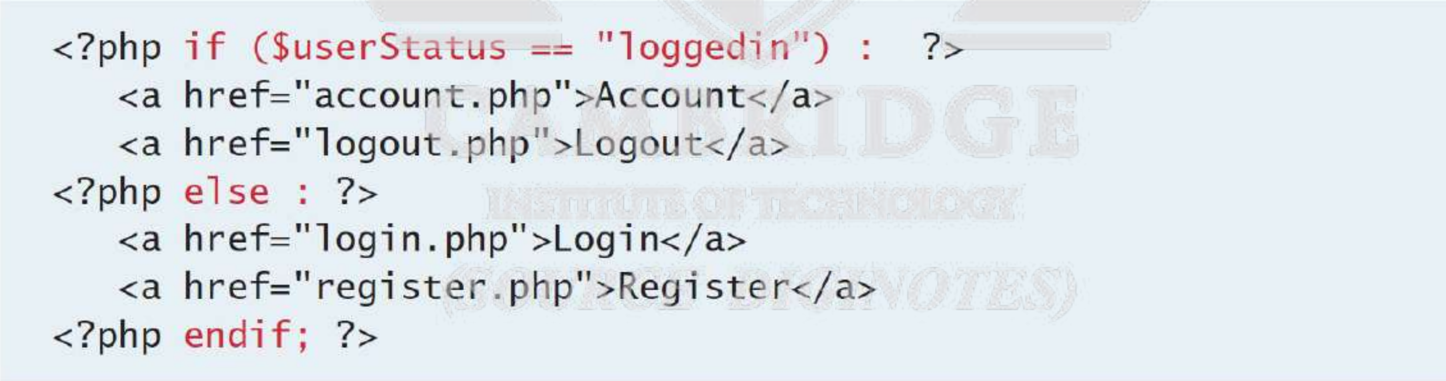
```
for ($count=0; $count < 10; $count++)
{
   echo $count;
}
```

LISTING 8.11 for loops

Alternate syntax for Control Structures

PHP has an alternative syntax for most of its control structures. In this alternate syntax

- the colon (:) replaces the opening curly bracket,
- while the closing brace is replaced with endif;, endwhile;, endfor;, endforeach;, or endswitch;



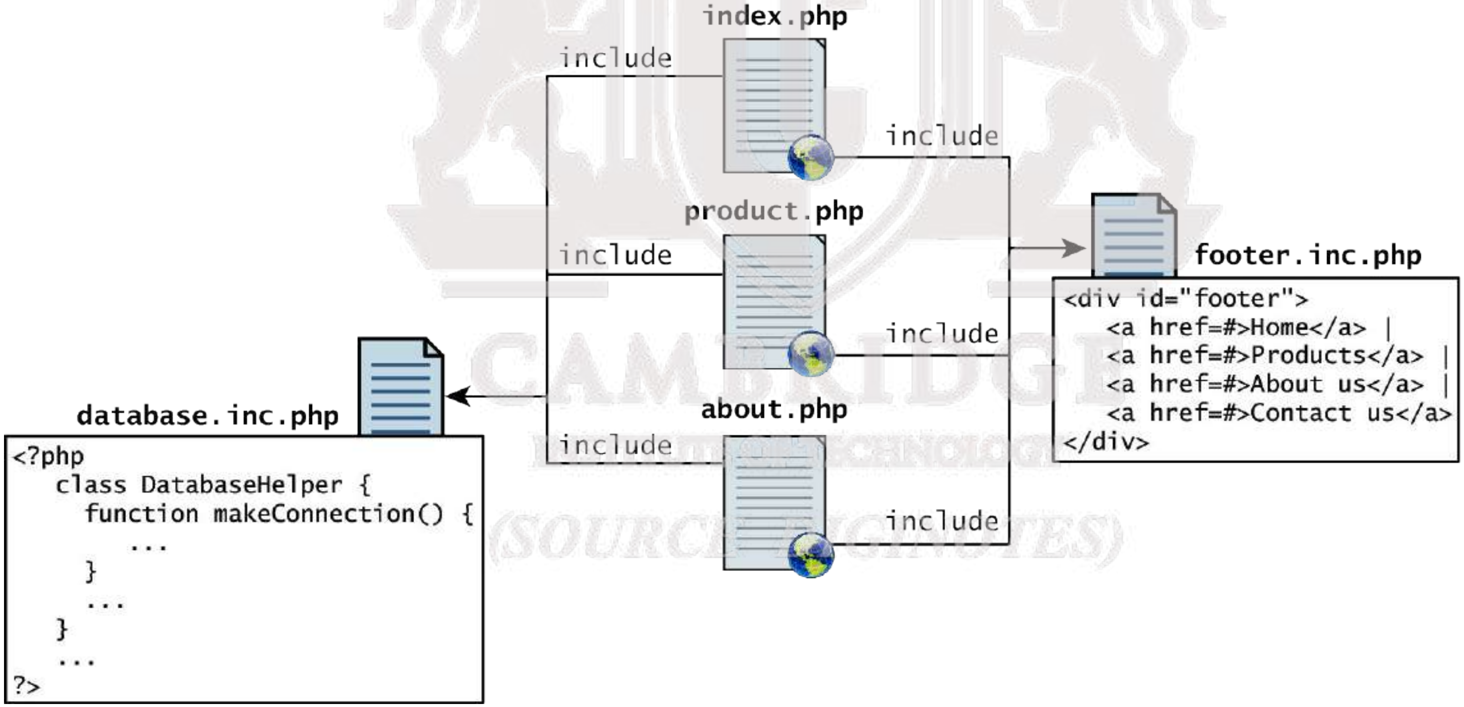
```
<?php if ($userStatus == "loggedin") :  ?>
   <a href="account.php">Account</a>
   <a href="logout.php">Logout</a>
<?php else : ?>
   <a href="login.php">Login</a>
   <a href="register.php">Register</a>
<?php endif; ?>
```

LISTING 8.12 Alternate syntax for control structures

Include Files

Organize your code

PHP does have one important facility that is generally unlike other nonweb programming languages, namely the ability to include or insert content from one file into another.



```
<?php
   class DatabaseHelper {
     function makeConnection() {
         ...
     }
       ...
   }
   ...
?>
```

```
<div id="footer">
   <a href=#>Home</a> |
   <a href=#>Products</a> |
   <a href=#>About us</a> |
   <a href=#>Contact us</a>
</div>
```

Include Files

Organize your code

PHP provides four different statements for including files, as shown below.

include "somefile.php";

include_once "somefile.php";

require "somefile.php";

require_once "somefile.php";

With include, a warning is displayed and then execution continues. With require, an error is displayed and execution stops.

Include Files

Scope

Include files are the equivalent of copying and pasting.

- Variables defined within an include file will have the scope of the line on which the include occurs
- Any variables available at that line in the calling file will be available within the called file
- If the include occurs inside a function, then all of the code contained in the called file will behave as though it had been defined inside that function

Section 5 of 5

FUNCTIONS

Functions

You mean we don't write everything in main?

Just as with any language, writing code in the main function (which in PHP is equivalent to coding in the markup between <?php and ?> tags) is not a good habit to get into.

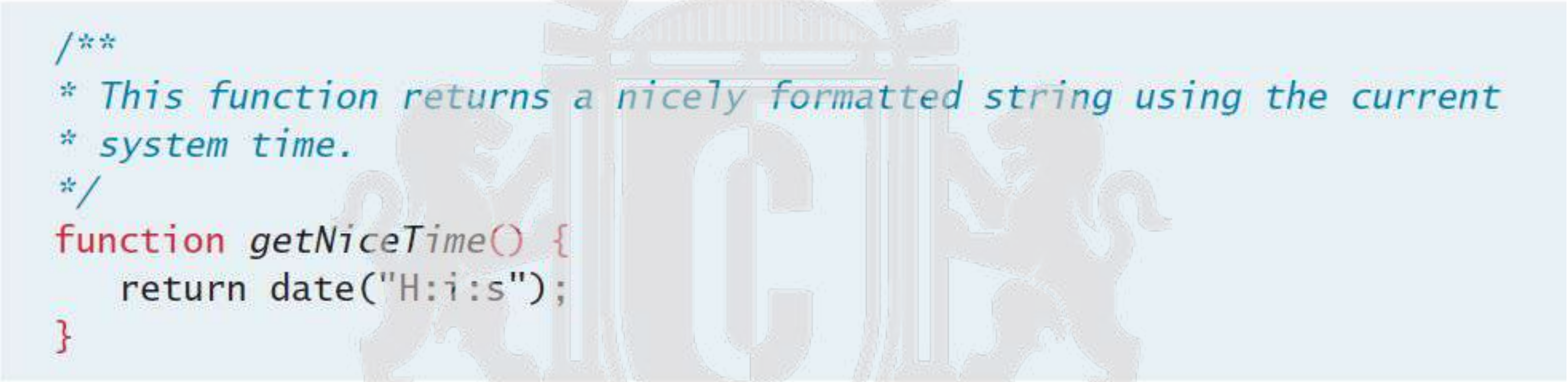
A **function** in PHP contains a small bit of code that accomplishes one thing. In PHP there are two types of function: user-defined functions and built-in functions.

1. A **user-defined function** is one that you the programmer define.

2. A **built-in function** is one of the functions that come with the PHP environment

Functions

syntax



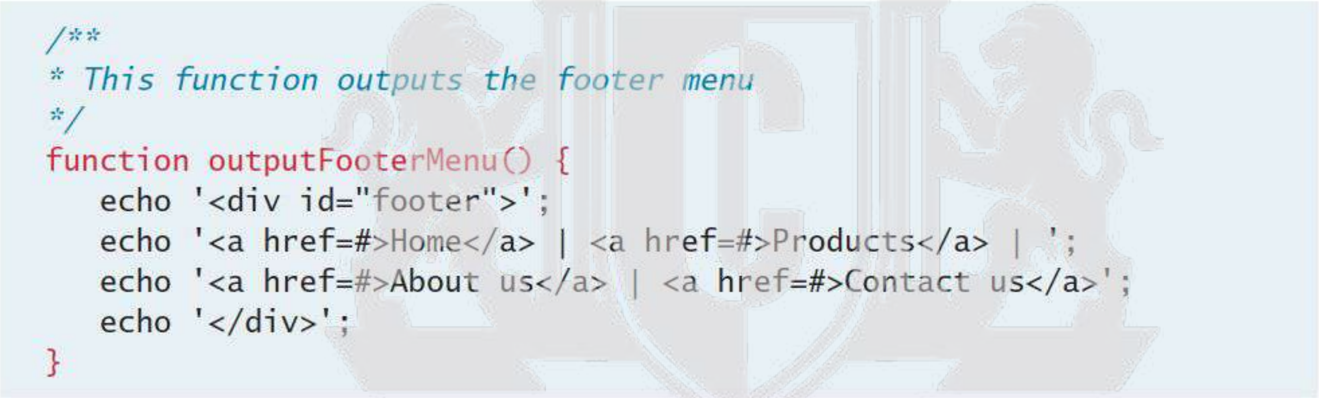
```
/**
* This function returns a nicely formatted string using the current
* system time.
*/
function getNiceTime() {
   return date("H:i:s");
}
```

LISTING 8.13 The definition of a function to return the current time as a string

While the example function in Listing 8.13 returns a value, there is no requirement for this to be the case.

Functions

No return – no big deal.



```
/**
* This function outputs the footer menu
*/
function outputFooterMenu() {
   echo '<div id="footer">';
   echo '<a href=#>Home</a> | <a href=#>Products</a> | ';
   echo '<a href=#>About us</a> | <a href=#>Contact us</a>';
   echo '</div>';
}
```

LISTING 8.14 The definition of a function without a return value

Call a function

- Now that you have defined a function, you are able to use it whenever you want to. To call a function you must use its name with the () brackets.

- Since getNiceTime() returns a string, you can assign that return value to a variable, or echo that return value directly, as shown below.

  $output = getNiceTime();

  echo getNiceTime();

  If the function doesn't return a value, you can just call the function:

  outputFooterMenu();

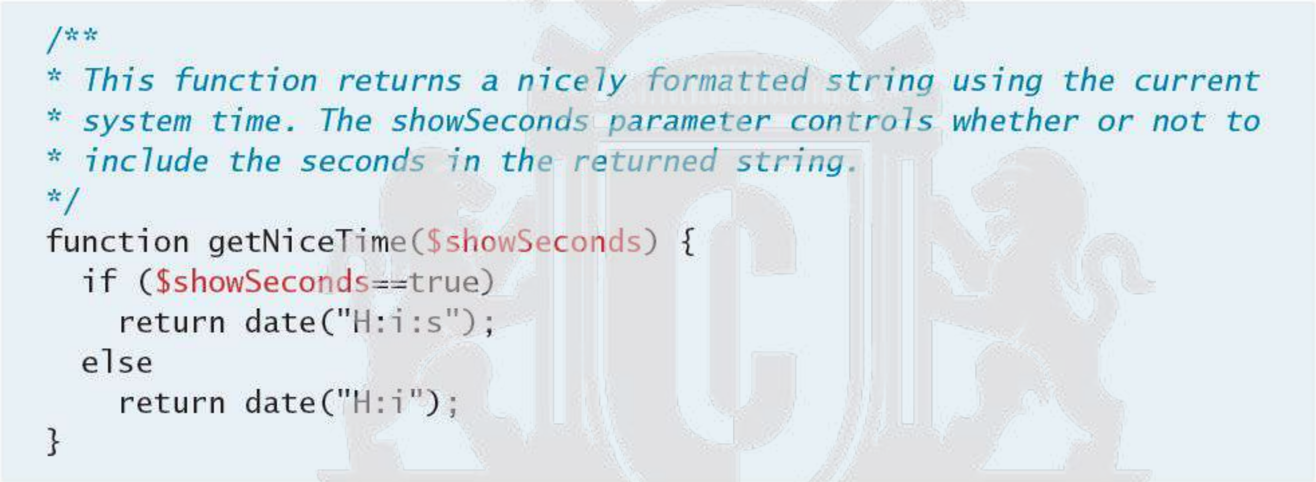
Parameters

Parameters are the mechanism by which values are passed into functions.

To define a function with parameters, you must decide

- how many parameters you want to pass in,
- and in what order they will be passed
- Each parameter must be named

Parameters



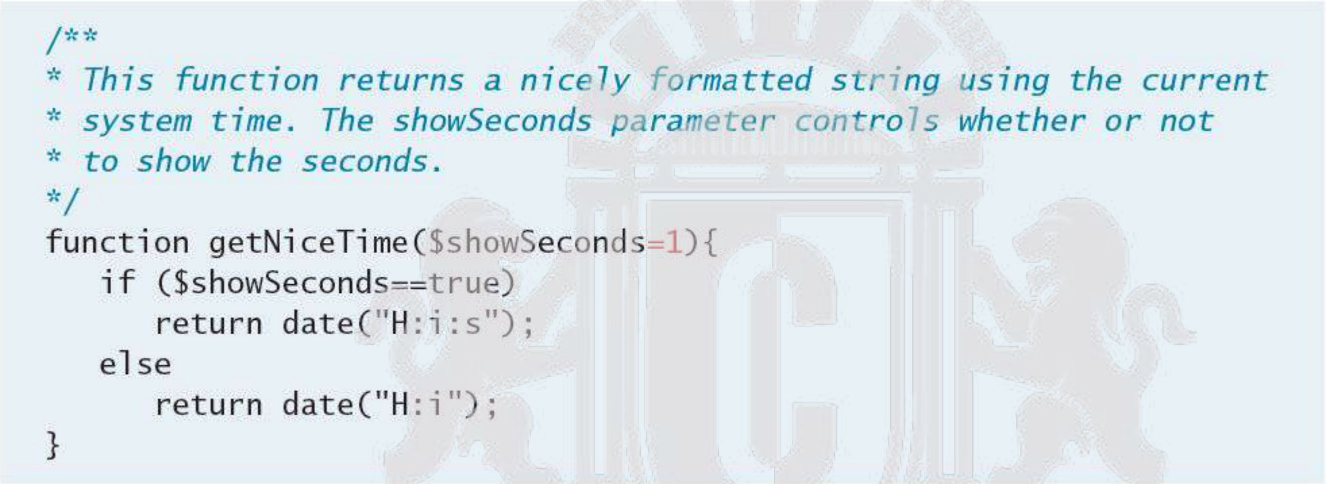
```
/**
* This function returns a nicely formatted string using the current
* system time. The showSeconds parameter controls whether or not to
* include the seconds in the returned string.
*/
function getNiceTime($showSeconds) {
  if ($showSeconds==true)
    return date("H:i:s");
  else
    return date("H:i");
}
```

LISTING 8.15 A function to return the current time as a string with an integer parameter

Thus to call our function, you can now do it in two ways:

echo getNiceTime(1); *// this will print seconds*
echo getNiceTime(0); *// will not print seconds*

Parameter Default Values



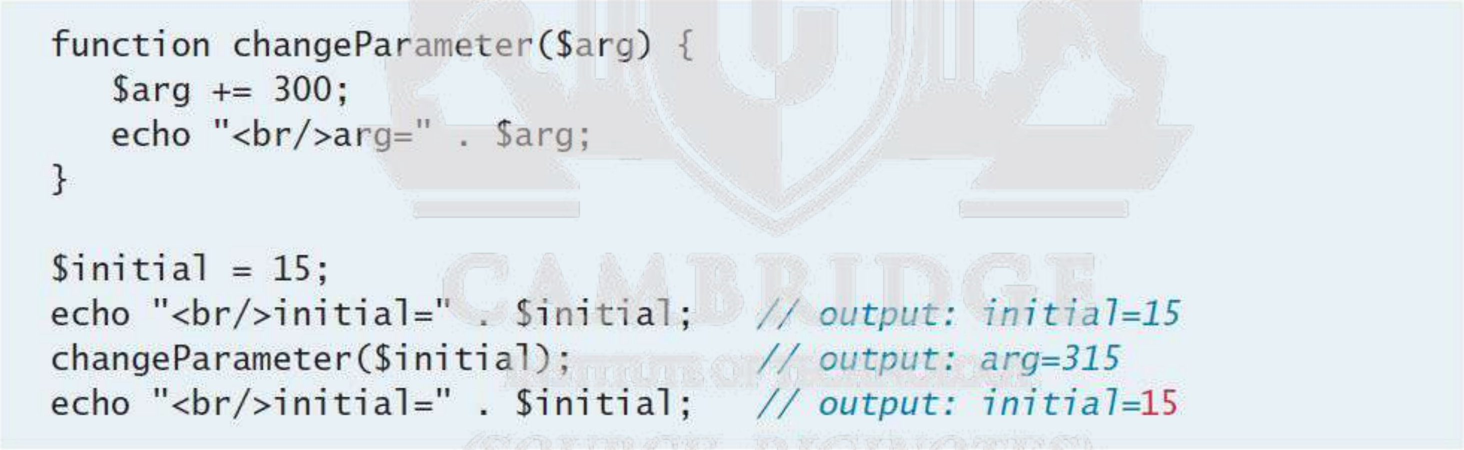
```
/**
* This function returns a nicely formatted string using the current
* system time. The showSeconds parameter controls whether or not
* to show the seconds.
*/
function getNiceTime($showSeconds=1){
   if ($showSeconds==true)
      return date("H:i:s");
   else
      return date("H:i");
}
```

LISTING 8.16 A function to return the current time with a parameter that includes a default

Now if you were to call the function with no values, the $showSeconds parameter would take on the default value, which we have set to 1, and return the string with seconds.

Pass Parameters by Value

By default, arguments passed to functions are **passed by value** in PHP. This means that PHP passes a copy of the variable so if the parameter is modified within the function, it does not change the original.



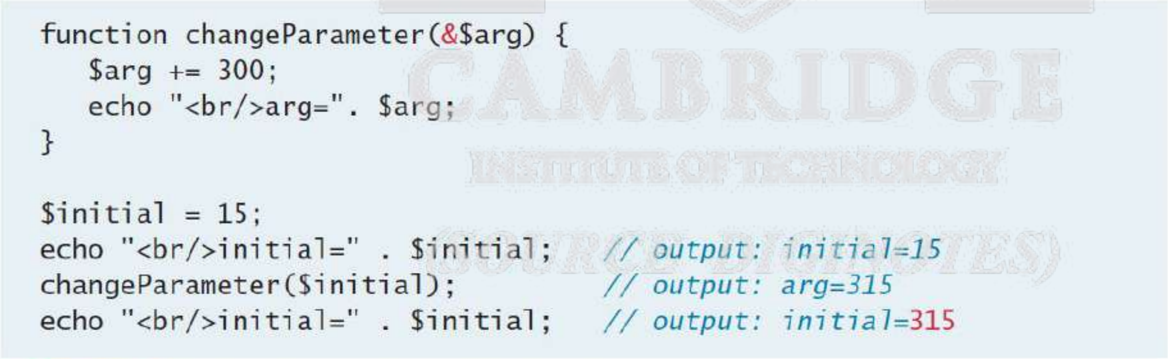
```
function changeParameter($arg) {
   $arg += 300;
   echo "<br/>arg=" . $arg;
}

$initial = 15;
echo "<br/>initial=" . $initial;    // output: initial=15
changeParameter($initial);          // output: arg=315
echo "<br/>initial=" . $initial;    // output: initial=15
```

LISTING 8.17 Passing a parameter by value

Pass Parameters by Reference

➢PHP also allows arguments to functions to be **passed by reference**, which will allow a function to change the contents of a passed variable.

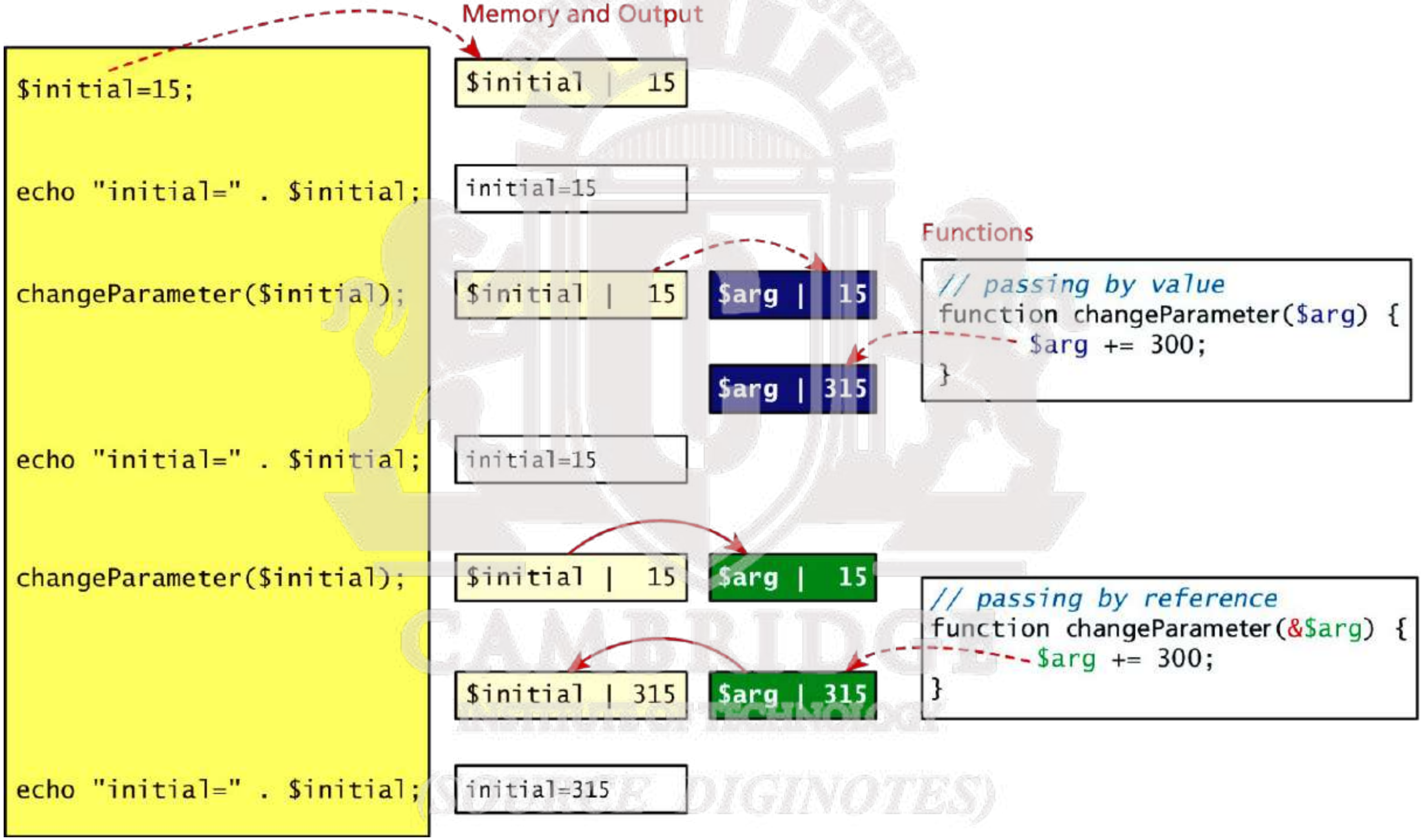
➢The mechanism in PHP to specify that a parameter is passed by reference is to add an ampersand (&) symbol next to the parameter name in the function declaration



```
function changeParameter(&$arg) {
   $arg += 300;
   echo "<br/>arg=". $arg;
}

$initial = 15;
echo "<br/>initial=" . $initial;   // output: initial=15
changeParameter($initial);         // output: arg=315
echo "<br/>initial=" . $initial;   // output: initial=315
```

LISTING 8.18 Passing a parameter by reference

Value vs Reference



```
$initial=15;

echo "initial=" . $initial;

changeParameter($initial);

echo "initial=" . $initial;

changeParameter($initial);

echo "initial=" . $initial;
```

Memory and Output

```
$initial | 15
initial=15
$initial | 15     $arg | 15
                  $arg | 315
initial=15
$initial | 15     $arg | 15
$initial | 315    $arg | 315
initial=315
```

Functions

```
// passing by value
function changeParameter($arg) {
    $arg += 300;
}
```

```
// passing by reference
function changeParameter(&$arg) {
    $arg += 300;
}
```

Variable Scope in functions

- All variables defined within a function (such as parameter variables) have **function scope**, meaning that they are only accessible within the function.

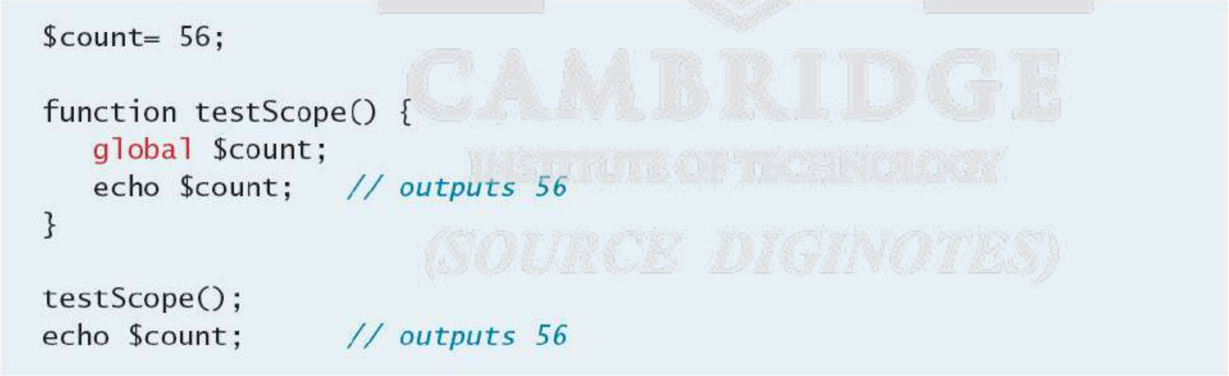
- Any variables created outside of the function in the main script are unavailable within a function.

```
$count= 56;

function testScope() {
    echo $count; // outputs 0 or generates run-time
            //warning/error

}

testScope();
echo $count; // outputs 56
```

Global variables

Sometimes unavoidable

- ❖Variables defined in the main script are said to have **global scope.**
- ❖Unlike in other programming languages, a global variable is not, by default, available within functions.
- ❖PHP does allow variables with global scope to be accessed within a function using the **global** keyword



```
$count= 56;

function testScope() {
   global $count;
   echo $count;    // outputs 56
}

testScope();
echo $count;       // outputs 56
```

LISTING 8.19 Using the global keyword

What You’ve Learned

1 Server-Side Development

2 Web Server’s Responsabilities

3 Quick Tour of PHP

4 Program Control

5 Functions

WEB TECHNOLOGY AND ITS APPLICATIONS

17CS71

Mr. GANESH D R
ASSISTANT PROFESSOR,
DEPT OF CSE, CITECH

WEB TECHNOLOGY AND ITS APPLICATIONS

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)
SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	**10 Hours**
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	**10 Hours**
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	**10 Hours**
Module – 4	
PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays, $_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	**10 Hours**
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	**10 Hours**

MODULE 4 - SYLLABUS

- PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays,$_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes andObjects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling

PHP Arrays and Superglobals

Chapter 9

Section 1 of 5

ARRAYS

Arrays

Background

An array is a data structure that

- Collects a number of related elements together in a single variable.
- Allows the set to be Iterated
- Allows access of any element

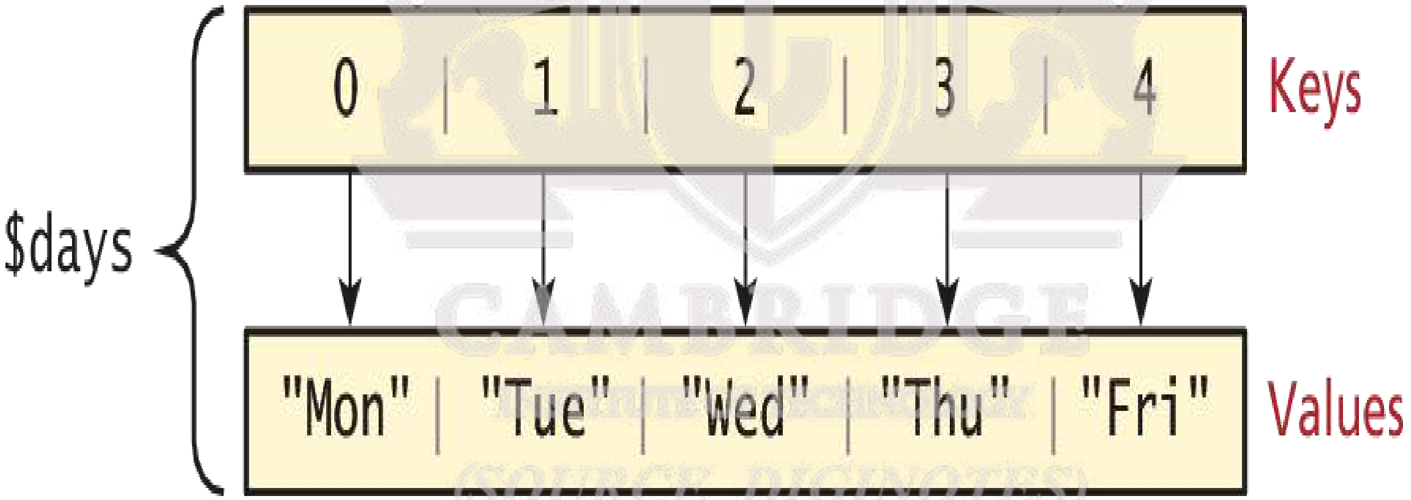
Since PHP implements an array as a dynamic structure:

- Add to the array
- Remove from the array

Arrays

Key Value

In PHP an array is actually an **ordered map**, which associates each value in the array with a key.



Arrays

Keys

Array keys are the means by which you reer to single element in the array.

In most programming languages array keys are limited to integers, start at 0, and go up by 1.

In PHP, array keys *must* be either integers or strings and need not be sequential.

- Don’t mix key types i.e. “1” vs 1
- If you don’t explicitly define them they are 0,1,…

Arrays

Values

Array values, unlike keys, are not restricted to integers and strings.

They can be any object, type, or primitive supported in PHP.

You can even have objects of your own types, so long as the keys in the array are integers and strings.

Arrays

Defining an array

The following declares an empty array named days:

```
$days = array();
```

You can also initialize it with a comma-delimited list of values inside the ( ) braces using either of two following syntaxes:

```
$days = array("Mon","Tue","Wed","Thu","Fri");

$days = ["Mon","Tue","Wed","Thu","Fri"]; // alternate
```

Arrays

Defining an array

You can also declare each subsequent element in the array individually:

```
$days = array();

$days[0] = "Mon"; //set 0th key’s value to “Mon”

$days[1] = "Tue";

// also alternate approach

$daysB = array();

$daysB[] = "Mon"; //set the next sequential value to “Mon”

$daysB[] = "Tue";
```

Arrays

Access values

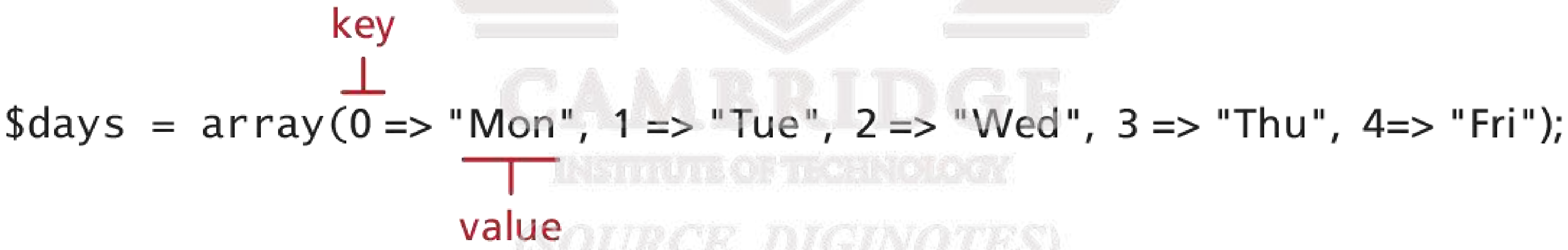
To access values in an array you refer to their key using the square bracket notation.

```
echo "Value at index 1 is ". $days[1];
```

Keys and Values

In PHP, you are also able to explicitly define the keys in addition to the values.

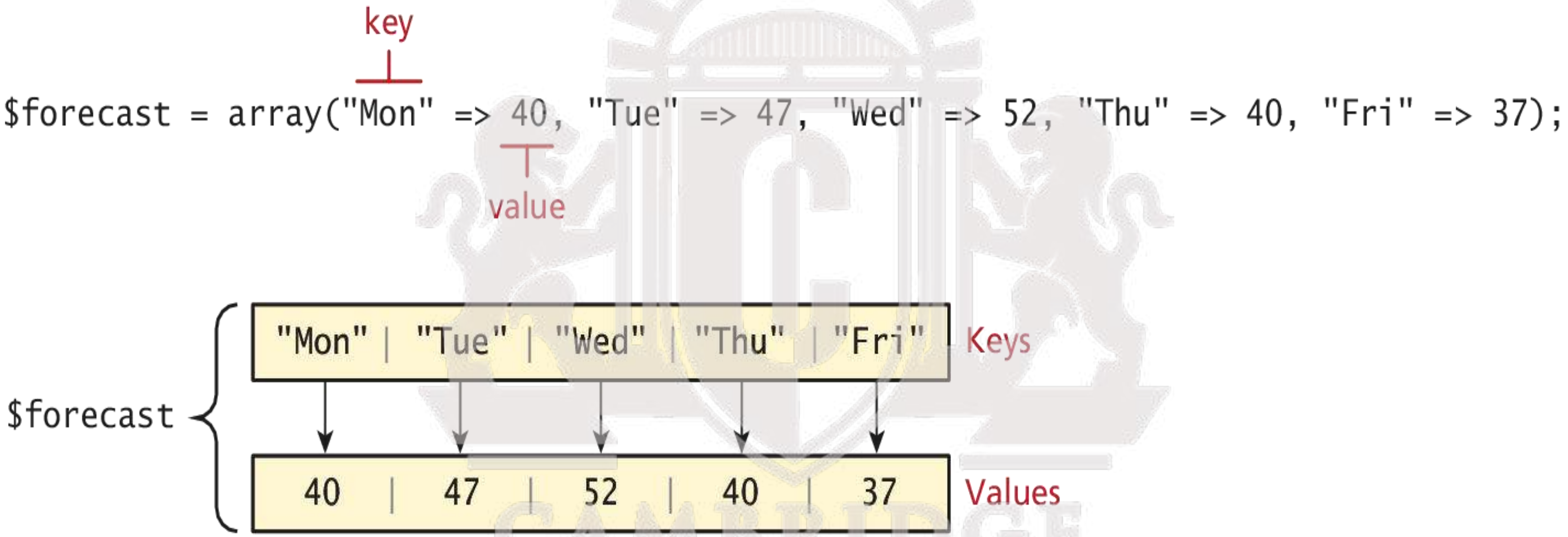
This allows you to use keys other than the classic 0, 1, 2, . . . , n to define the indexes of an array.



```
$days = array(0 => "Mon", 1 => "Tue", 2 => "Wed", 3 => "Thu", 4=> "Fri");
```

Super Explicit

Array declaration with string keys, integer values



```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```

```
echo $forecast["Tue"];   // outputs 47
echo $forecast["Thu"];   // outputs 40
```

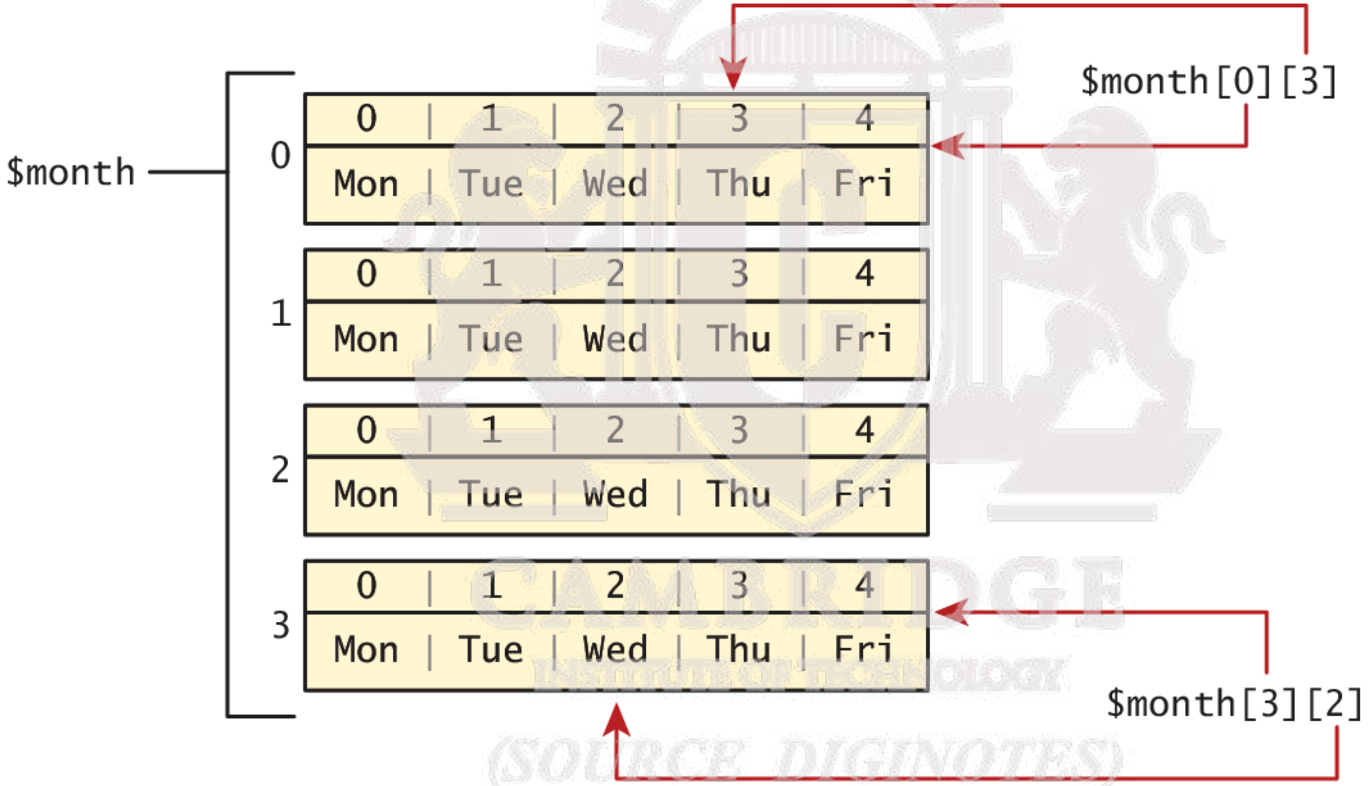
Multidimensional Arrays

Creation

```
$month = array(
            array("Mon","Tue","Wed","Thu","Fri"),
            array("Mon","Tue","Wed","Thu","Fri"),
            array("Mon","Tue","Wed","Thu","Fri"),
            array("Mon","Tue","Wed","Thu","Fri")
);
echo $month[0][3]; // outputs Thu
```

Multidimensional Arrays

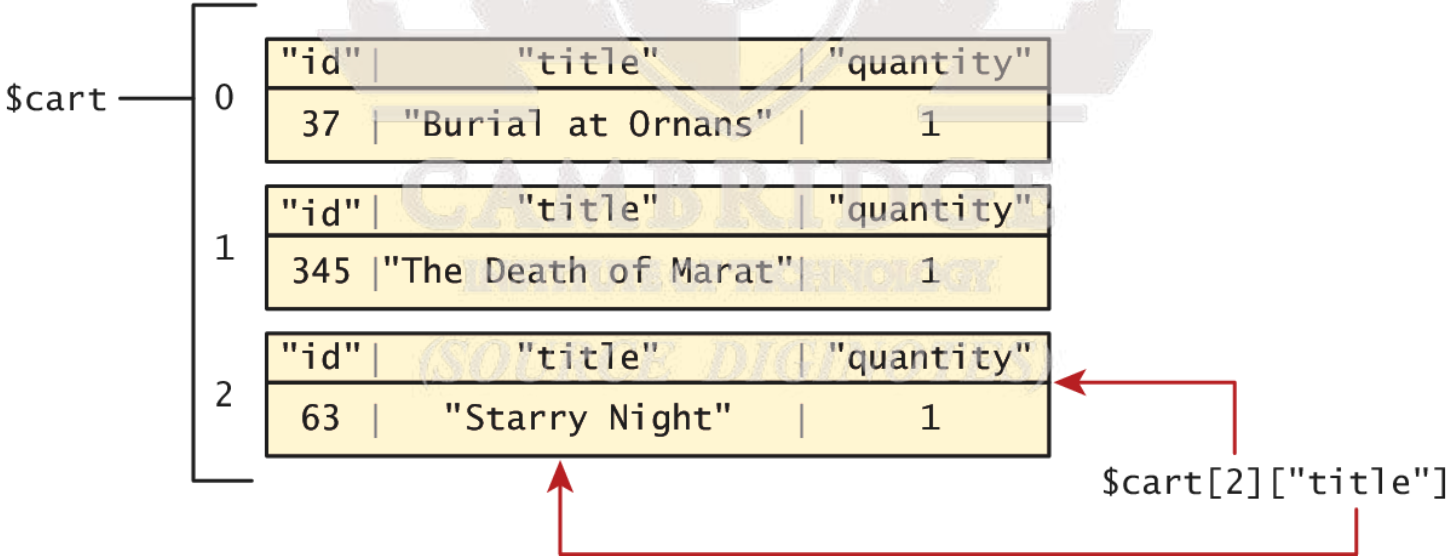
Access


$month[0][3]
$month
0
0 | 1 | 2 | 3 | 4
Mon | Tue | Wed | Thu | Fri
1
0 | 1 | 2 | 3 | 4
Mon | Tue | Wed | Thu | Fri
2
0 | 1 | 2 | 3 | 4
Mon | Tue | Wed | Thu | Fri
3
0 | 1 | 2 | 3 | 4
Mon | Tue | Wed | Thu | Fri
$month[3][2]

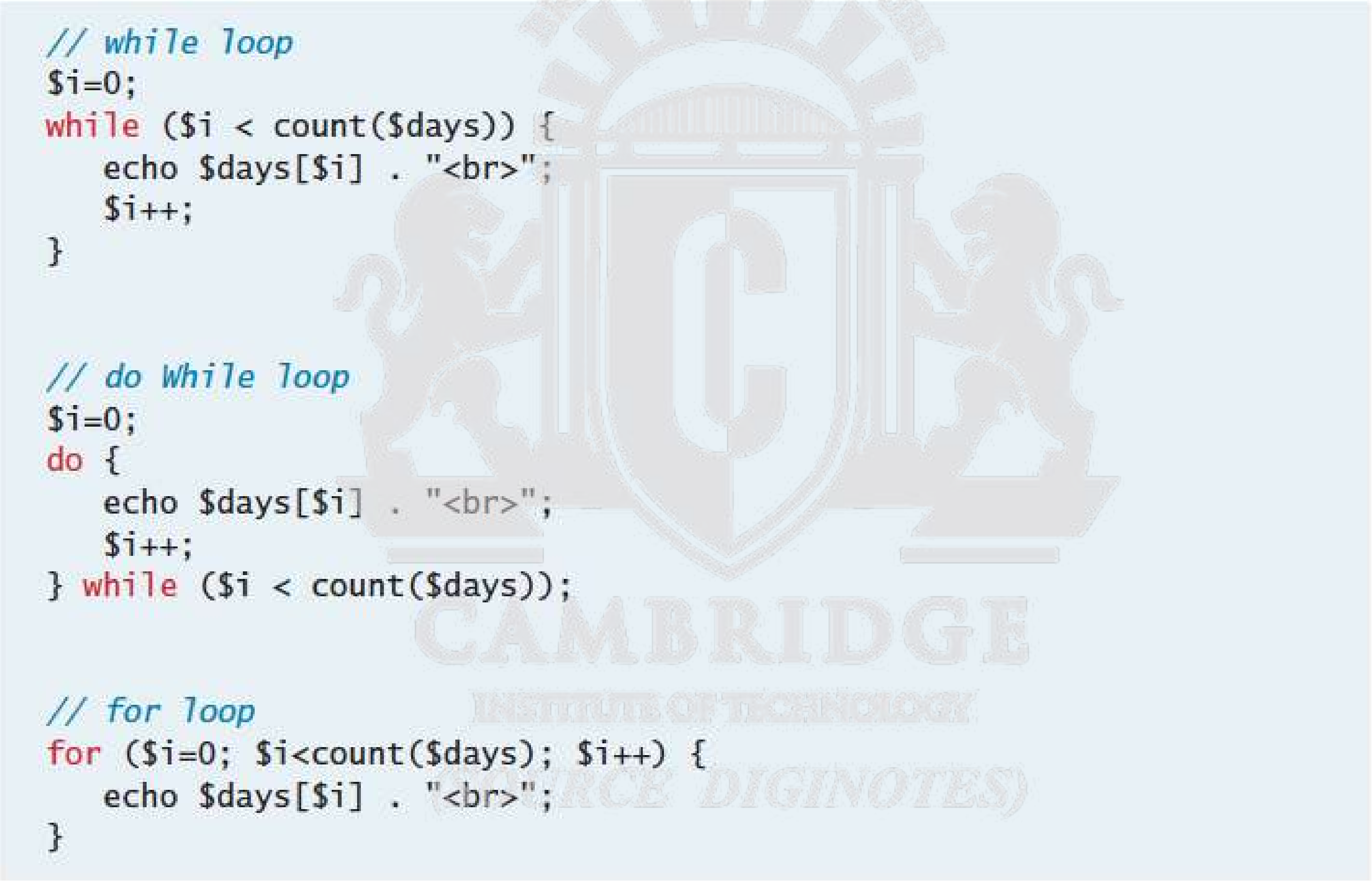
Multidimensional Arrays

Another example

```
$cart = array();

$cart[] = array("id" => 37, "title" => "Burial at Ornans", "quantity" => 1);

$cart[] = array("id" => 345, "title" => "The Death of Marat", "quantity" => 1);

$cart[] = array("id" => 63, "title" => "Starry Night", "quantity" => 1);
```


$cart
0
"id" | "title" | "quantity"
37 | "Burial at Ornans" | 1
1
"id" | "title" | "quantity"
345 |"The Death of Marat"| 1
2
"id" | "title" | "quantity"
63 | "Starry Night" | 1
$cart[2]["title"]

Iterating through an array



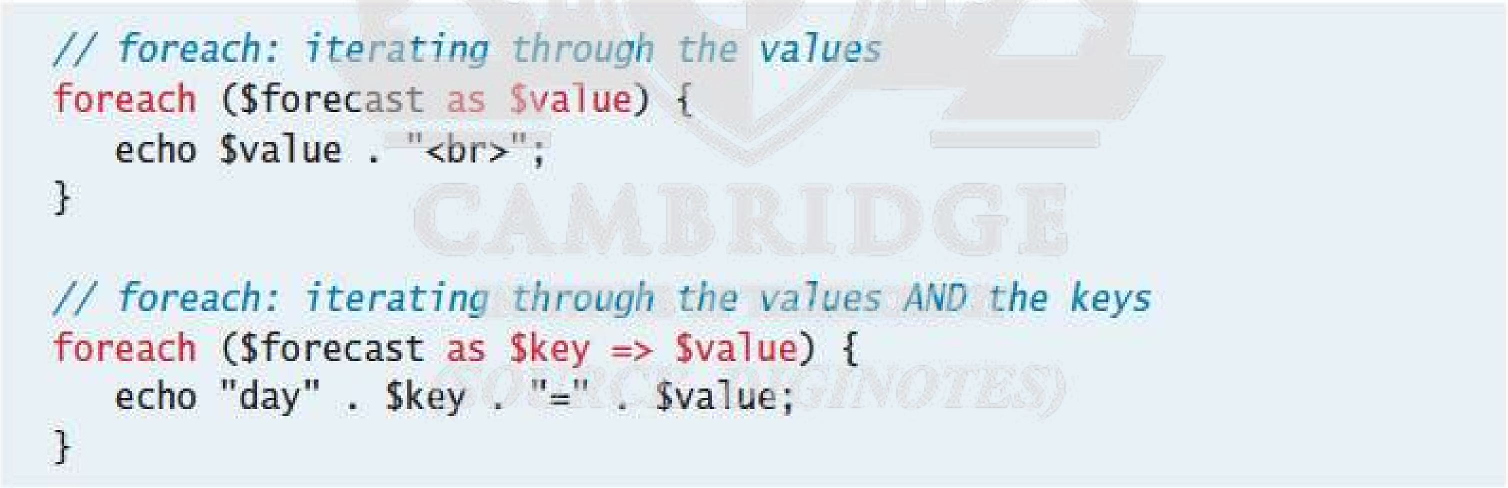
```
// while loop
$i=0;
while ($i < count($days)) {
   echo $days[$i] . "<br>";
   $i++;
}


// do While loop
$i=0;
do {
   echo $days[$i] . "<br>";
   $i++;
} while ($i < count($days));


// for loop
for ($i=0; $i<count($days); $i++) {
   echo $days[$i] . "<br>";
}
```

LISTING 9.2 Iterating through an array using while, do while, and for loops

Iterating through an array

Foreach loop is pretty nice

The challenge of using the classic loop structures is that when you have nonsequential integer keys (i.e., an associative array), you can't write a simple loop that uses the $i++ construct. To address the dynamic nature of such arrays, you have to use iterators to move through such an array.



```
// foreach: iterating through the values
foreach ($forecast as $value) {
   echo $value . "<br>";
}

// foreach: iterating through the values AND the keys
foreach ($forecast as $key => $value) {
   echo "day" . $key . "=" . $value;
}
```

LISTING 9.3 Iterating through an associative array using a foreach loop

Source diginotes.in

Adding to an array

To an array

An element can be added to an array simply by using a key/index that hasn’t been used

```
$days[5] = "Sat";
```

A new element can be added to the end of any array

```
$days[ ] = "Sun";
```

Adding to an array

And quickly printing

PHP is more than happy to let you “skip” an index

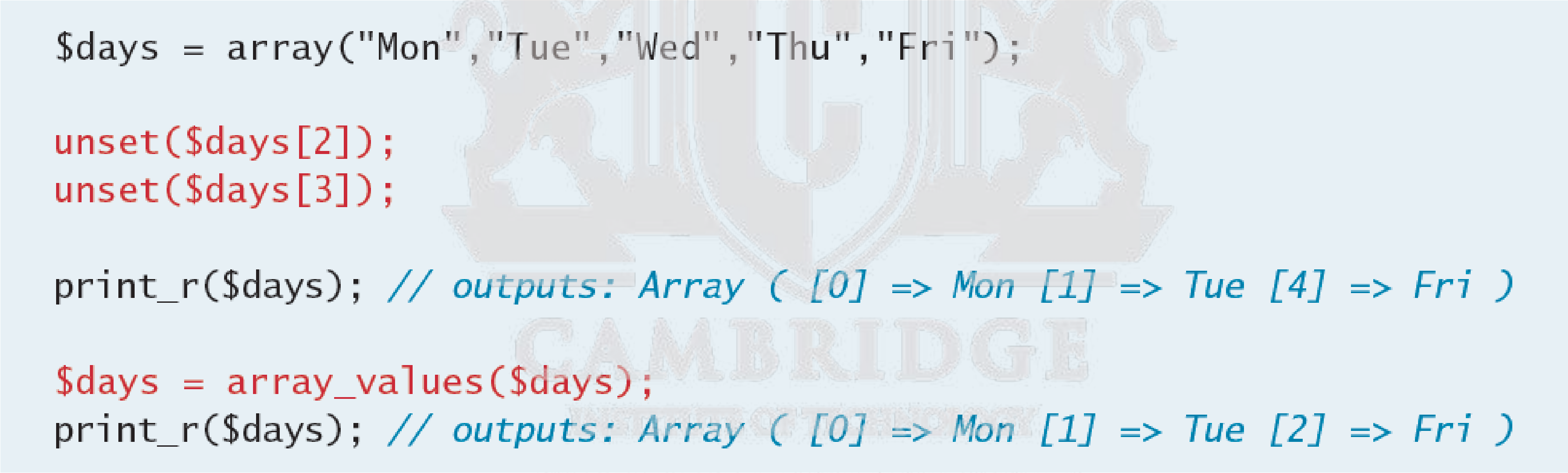
```
$days = array("Mon","Tue","Wed","Thu","Fri");

$days[7] = "Sat";

print_r($days);
```

```
Array ([0] => Mon [1] => Tue [2] => Wed [3] => Thu [4] => Fri [7] => Sat)’
```

If we try referencing $days[6], it will return a **NULL** value

Deleting from an array

You can explicitly delete array elements using the unset() function



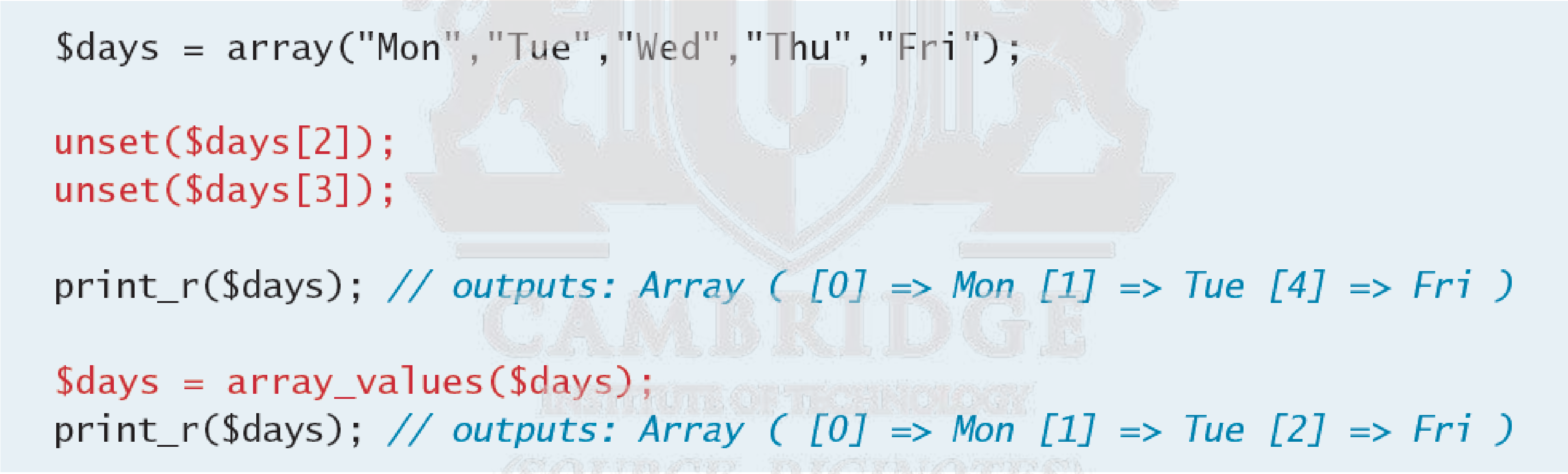
```
$days = array("Mon","Tue","Wed","Thu","Fri");

unset($days[2]);
unset($days[3]);

print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [4] => Fri )

$days = array_values($days);
print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [2] => Fri )
```

LISTING 9.4 Deleting elements

Deleting from an array

You can explicitly delete array elements using the unset() function.

array_values() reindexes the array numerically



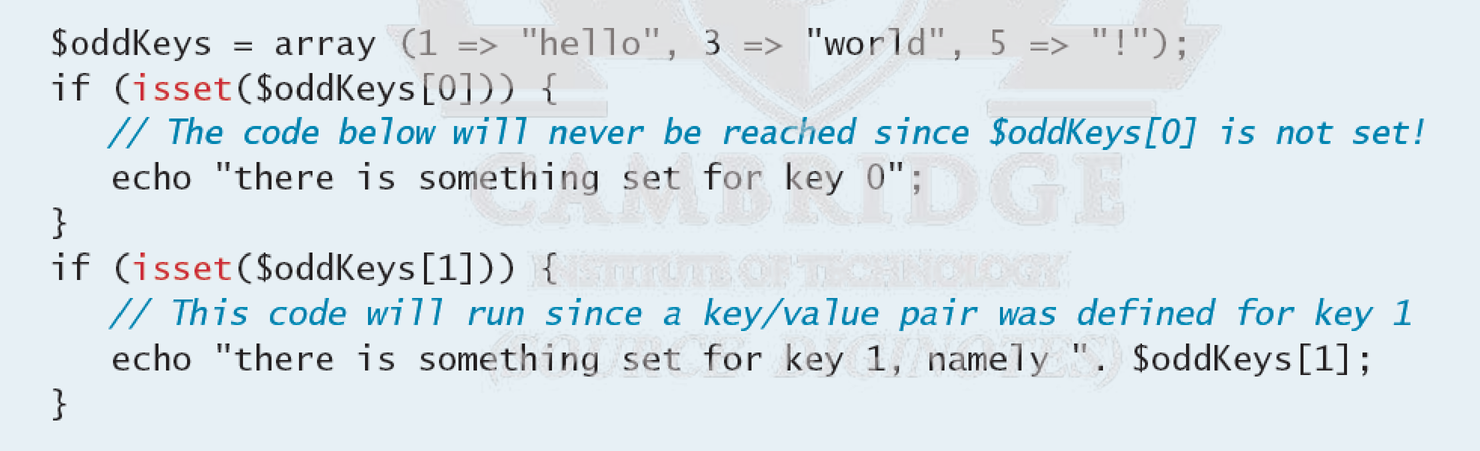
```
$days = array("Mon","Tue","Wed","Thu","Fri");

unset($days[2]);
unset($days[3]);

print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [4] => Fri )

$days = array_values($days);
print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [2] => Fri )
```

LISTING 9.4 Deleting elements

Checking for a value

Since array keys need not be sequential, and need not be integers, you may run into a scenario where you want to check if a value has been set for a particular key.

To check if a value exists for a key, you can therefore use the isset() function, which returns true if a value has been set, and false otherwise



```
$oddKeys = array (1 => "hello", 3 => "world", 5 => "!");
if (isset($oddKeys[0])) {
   // The code below will never be reached since $oddKeys[0] is not set!
   echo "there is something set for key 0";
}
if (isset($oddKeys[1])) {
   // This code will run since a key/value pair was defined for key 1
   echo "there is something set for key 1, namely ". $oddKeys[1];
}
```

LISTING 9.5 Illustrating nonsequential keys and usage of isset( )

Array Sorting

Sort it out

There are many built-in sort functions, which sort by key or by value. To sort the $days array by its values you would simply use:

sort($days);

As the values are all strings, the resulting array would be:

Array ([0] => Fri [1] => Mon [2] => Sat [3] => Sun [4] => Thu [5] => Tue [6] => Wed)

A better sort, one that would have kept keys and values associated together, is:

asort($days);

Array ([4] => Fri [0] => Mon [5] => Sat [6] => Sun [3] => Thu [1] => Tue [2] => Wed)

More array operations

Too many to go over in depth here…

- array_keys($someArray)
- array_values($someArray)
- array_rand($someArray, $num=1)
- array_reverse($someArray)
- array_walk($someArray, $callback, optionalParam)
- in_array($needle, $haystack)
- shuffle($someArray)
- …

Superglobal Arrays

PHP uses special predefined associative arrays called **superglobal variables** that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information.

They are called superglobal because they are always in scope, and always defined.

9.1.7 Superglobal Arrays

PHP uses special predefined associative arrays called superglobal variables that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information (see Table 9.1). They are called superglobal because these arrays are always in scope and always exist, ready for the programmer to access or modify them without having to use the `global` keyword as in Chapter 8.

Name	Description
$GLOBALS	Array for storing data that needs superglobal scope
$_COOKIES	Array of cookie data passed to page via HTTP request
$_ENV	Array of server environment data
$_FILES	Array of file items uploaded to the server
$_GET	Array of query string data passed to the server via the URL
$_POST	Array of query string data passed to the server via the HTTP header
$_REQUEST	Array containing the contents of $_GET, $_POST, and $_COOKIES
$_SESSION	Array that contains session data
$_SERVER	Array containing information about the request and the server

TABLE 9.1 Sugerglobal Variables

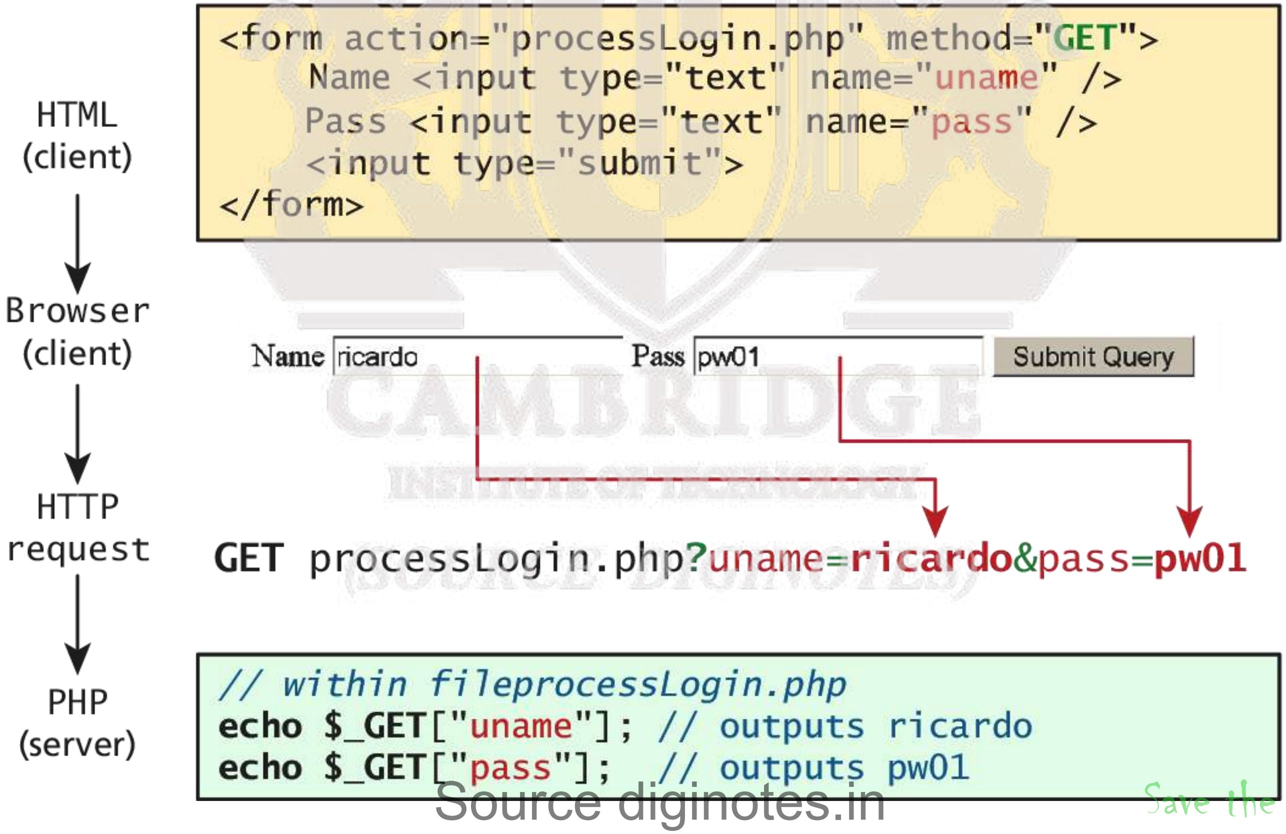
Section 2 of 5

$_GET AND $_POST SUPERGLOBAL ARRAYS

$_GET and $_POST

Sound familiar?

The $_GET and $_POST arrays are the most important superglobal variables in PHP since they allow the programmer to access data sent by the client in a query string.



```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Name ricardo Pass pw01 Submit Query

```
GET processLogin.php?uname=ricardo&pass=pw01
```

```
// within fileprocessLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```

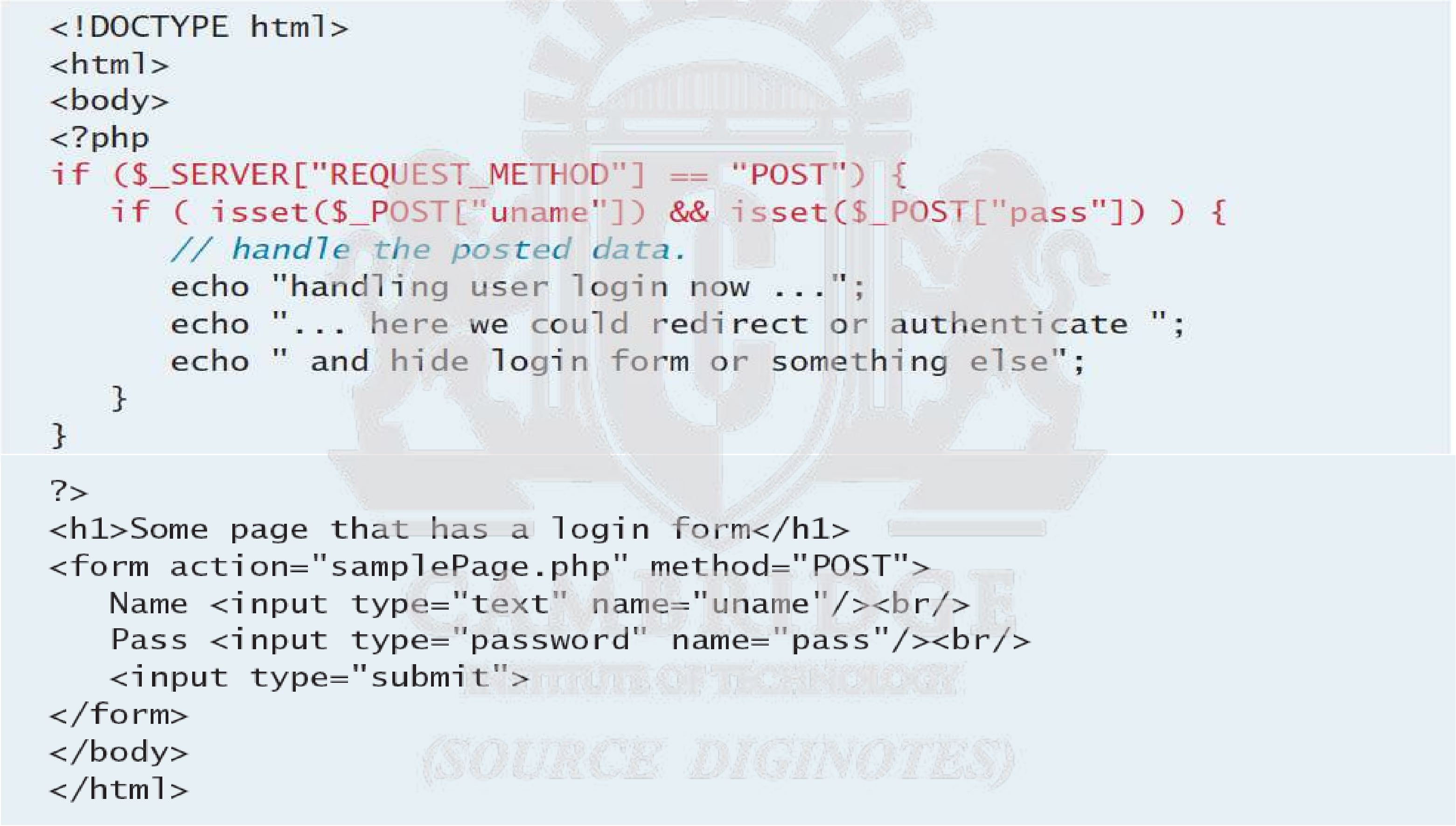
$_GET and $_POST

Sound familiar?

- Get requests parse query strings into the $_GET array
- Post requests are parsed into the $POST array

This mechanism greatly simplifies accessing the data posted by the user, since you need not parse the query string or the POST request headers!

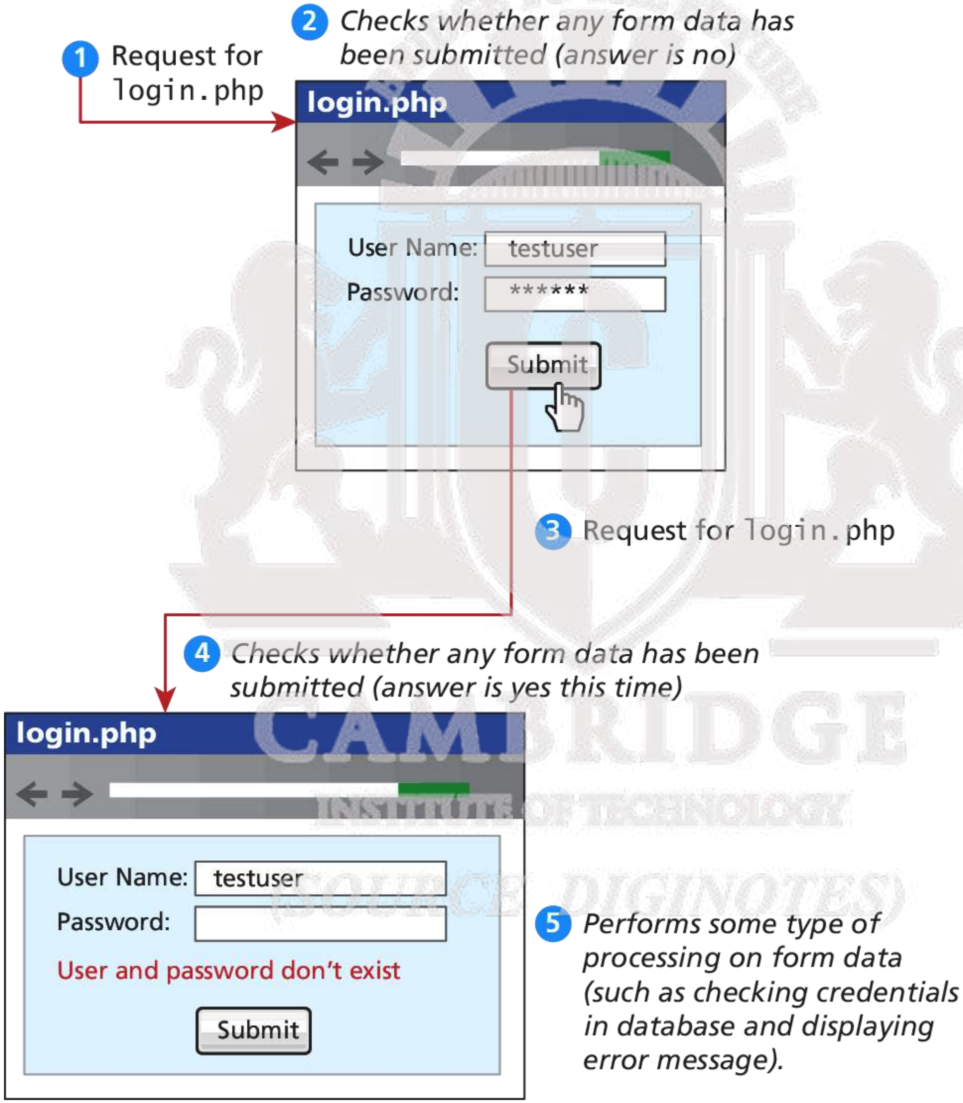
Determine if any data sent



```
<!DOCTYPE html>
<html>
<body>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if ( isset($_POST["uname"]) && isset($_POST["pass"]) ) {
      // handle the posted data.
      echo "handling user login now ...";
      echo "... here we could redirect or authenticate ";
      echo " and hide login form or something else";
   }
}
?>
<h1>Some page that has a login form</h1>
<form action="samplePage.php" method="POST">
   Name <input type="text" name="uname"/><br/>
   Pass <input type="password" name="pass"/><br/>
   <input type="submit">
</form>
</body>
</html>
```

LISTING 9.6 Using isset() to check query string data

Determine if any data sent



1 Request for `login.php`

2 *Checks whether any form data has been submitted (answer is no)*

login.php

User Name: testuser

Password: ******

Submit

3 Request for `login.php`

4 *Checks whether any form data has been submitted (answer is yes this time)*

login.php

User Name: testuser

Password:

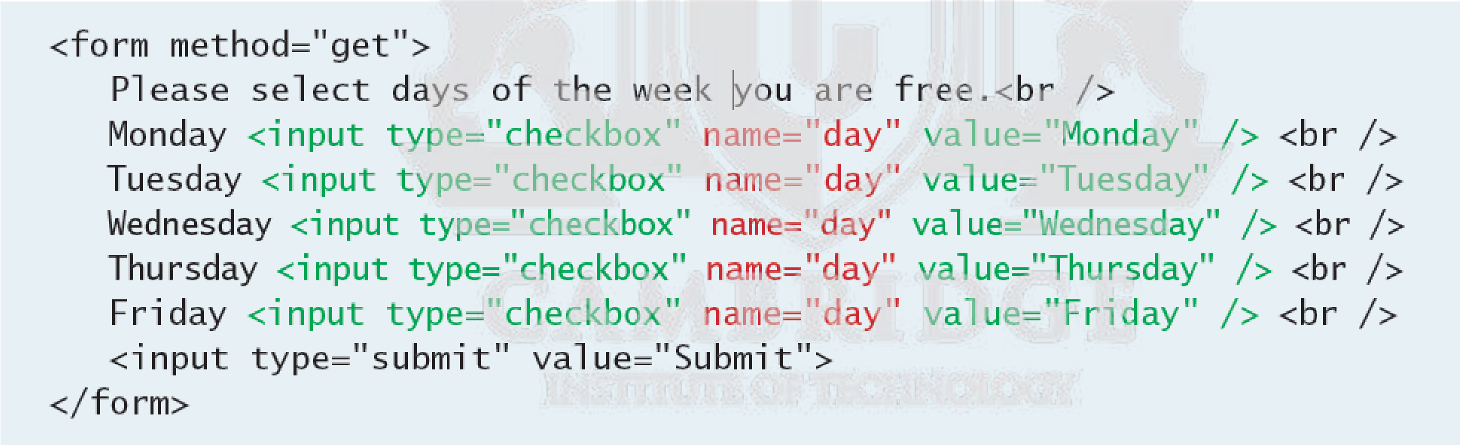
User and password don't exist

Submit

5 *Performs some type of processing on form data (such as checking credentials in database and displaying error message).*

Accessing Form Array Data

Sometimes in HTML forms you might have multiple values associated with a single name;



```
<form method="get">
   Please select days of the week you are free.<br />
   Monday <input type="checkbox" name="day" value="Monday" /> <br />
   Tuesday <input type="checkbox" name="day" value="Tuesday" /> <br />
   Wednesday <input type="checkbox" name="day" value="Wednesday" /> <br />
   Thursday <input type="checkbox" name="day" value="Thursday" /> <br />
   Friday <input type="checkbox" name="day" value="Friday" /> <br />
   <input type="submit" value="Submit">
</form>
```

LISTING 9.7 HTML that enables multiple values for one name

Accessing Form Array Data

HTML tweaks for arrays of data

Unfortunately, if the user selects more than one day and submits the form, the $_GET['day'] value in the superglobal array *will only contain the last value from the list* that was selected.

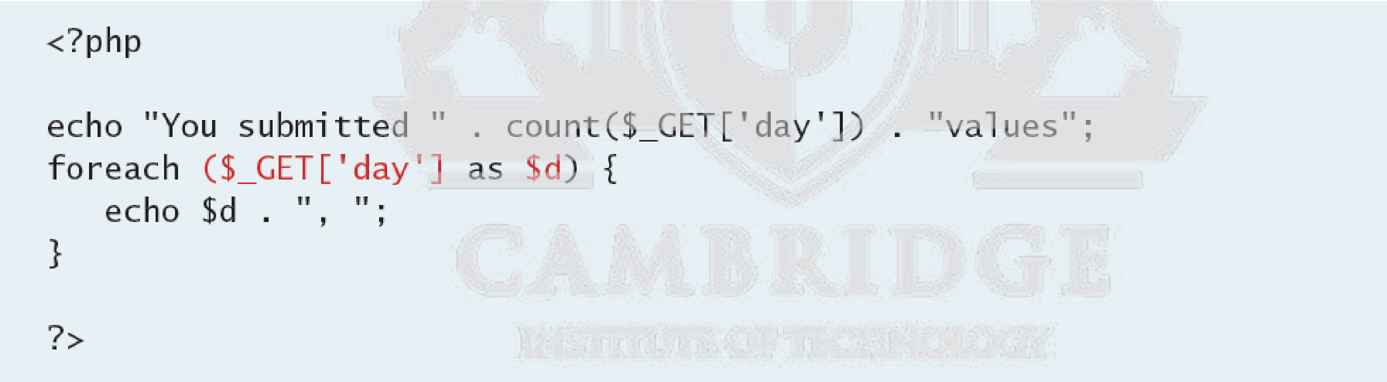
To overcome this limitation, you must change the name attribute for each checkbox from day to day[].

```
Monday <input type="checkbox" name="day[]" value="Monday" />

Tuesday <input type="checkbox" name="day[]" value="Tuesday" />
```

Accessing Form Array Data

Meanwhile on the server

After making this change in the HTML, the corresponding variable $_GET['day'] will now have a value that is of type array.



```
<?php

echo "You submitted " . count($_GET['day']) . "values";
foreach ($_GET['day'] as $d) {
   echo $d . ", ";
}

?>
```

LISTING 9.8 PHP code to display an array of checkbox variables

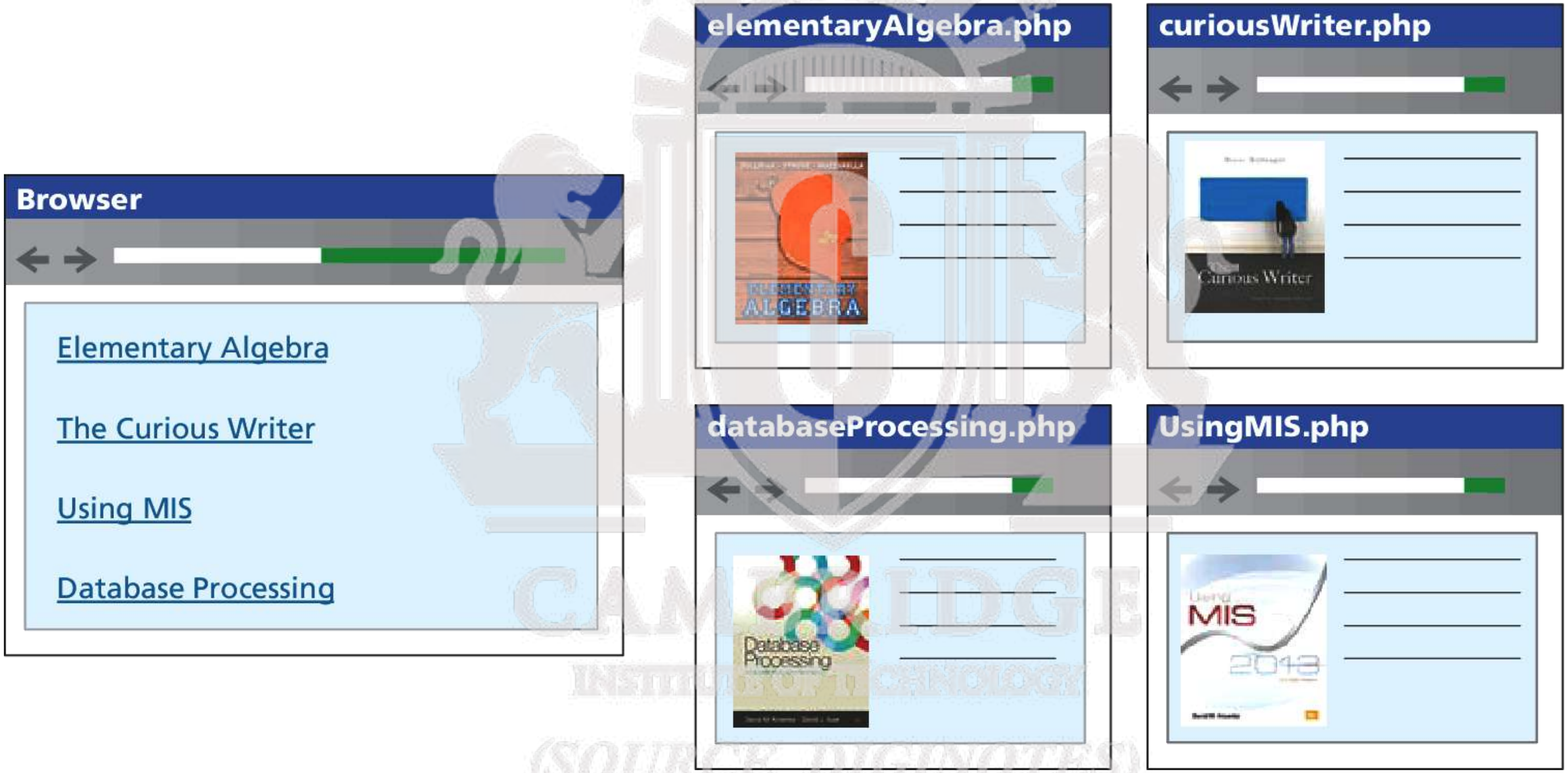
Using Query String in Links

Design idea

Imagine a web page in which we are displaying a list of book links. One approach would be to have a separate page for each book.

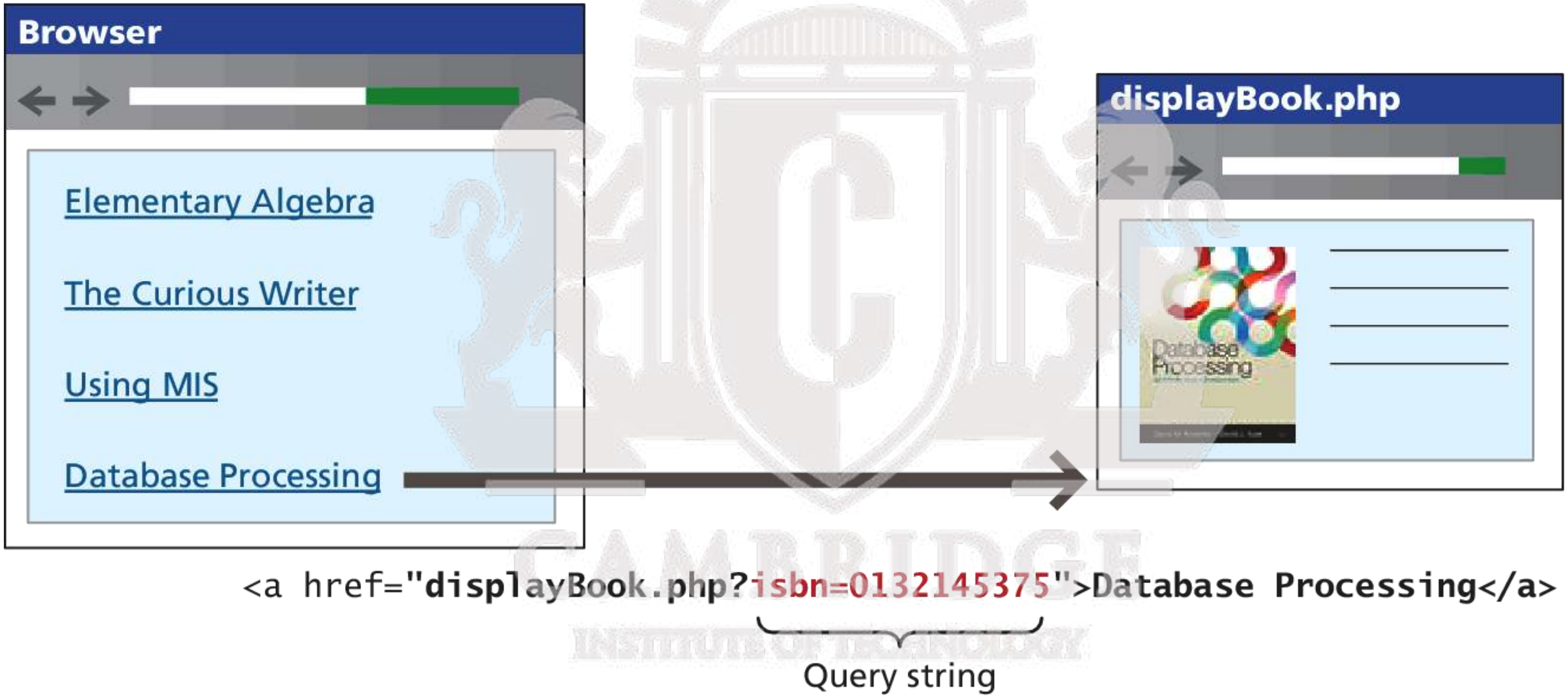
Using Query Strings in links

Not a great setup



Using Query Strings in links

Use the query string to reduce code duplication



```
<a href="displayBook.php?isbn=0132145375">Database Processing</a>
```

Source diginotes.in

Sanitizing Query Strings

Just because you are expecting a proper query string, doesn’t mean that you are going to get a properly constructed query string.

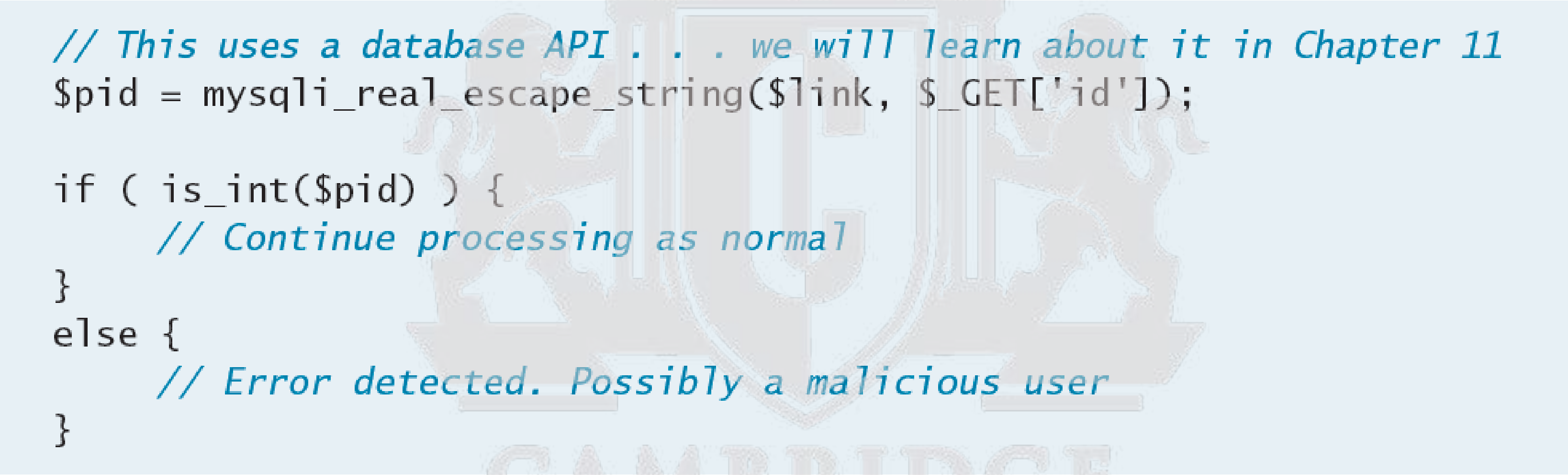
- **distrust all user input**

The process of checking user input for incorrect or missing information is sometimes referred to as the process of **sanitizing user inputs.**

Learn more about this in Chapter 11/12.

Sanitation

Don’t forget trim()



```
// This uses a database API . . . we will learn about it in Chapter 11
$pid = mysqli_real_escape_string($link, $_GET['id']);

if ( is_int($pid) ) {
     // Continue processing as normal
}
else {
     // Error detected. Possibly a malicious user
}
```

LISTING 9.9 Simple sanitization of query string values

Section 3 of 5

$_SERVER ARRAY

$_SERVER

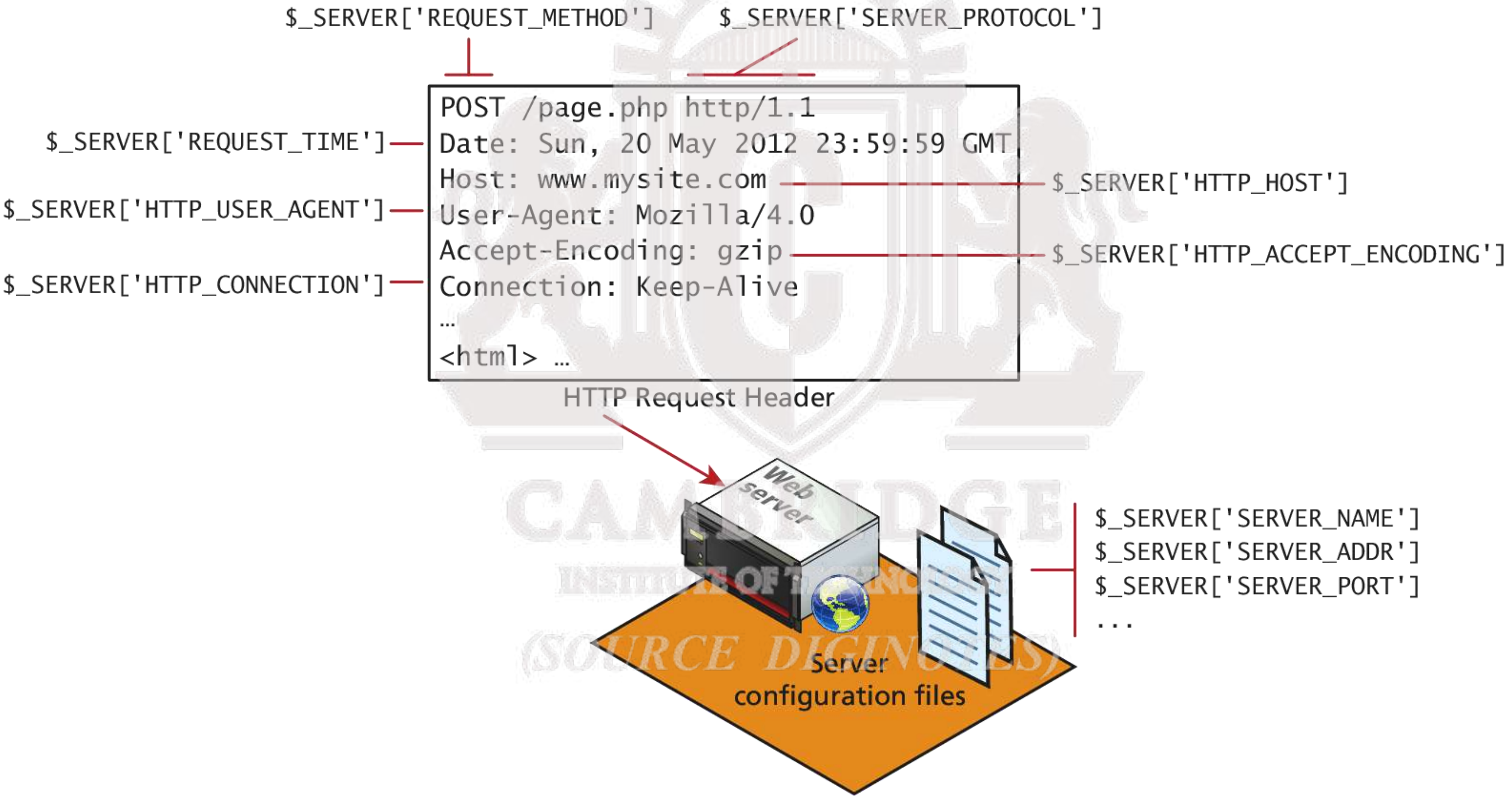
The $_SERVER associative array contains

- HTTP request headers (send by client)
- configuration options for PHP

To use the $_SERVER array, you simply refer to the relevant case-sensitive keyname:

```
echo $_SERVER["SERVER_NAME"] . "<br/>";

echo $_SERVER["SERVER_SOFTWARE"] . "<br/>";

echo $_SERVER["REMOTE_ADDR"] . "<br/>";
```

$_SERVER



$_SERVER['REQUEST_METHOD']

$_SERVER['SERVER_PROTOCOL']

$_SERVER['REQUEST_TIME']

$_SERVER['HTTP_USER_AGENT']

$_SERVER['HTTP_CONNECTION']

```
POST /page.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Accept-Encoding: gzip
Connection: Keep-Alive
…
<html> …
```

$_SERVER['HTTP_HOST']

$_SERVER['HTTP_ACCEPT_ENCODING']

HTTP Request Header

Server configuration files

$_SERVER['SERVER_NAME']
$_SERVER['SERVER_ADDR']
$_SERVER['SERVER_PORT']
...

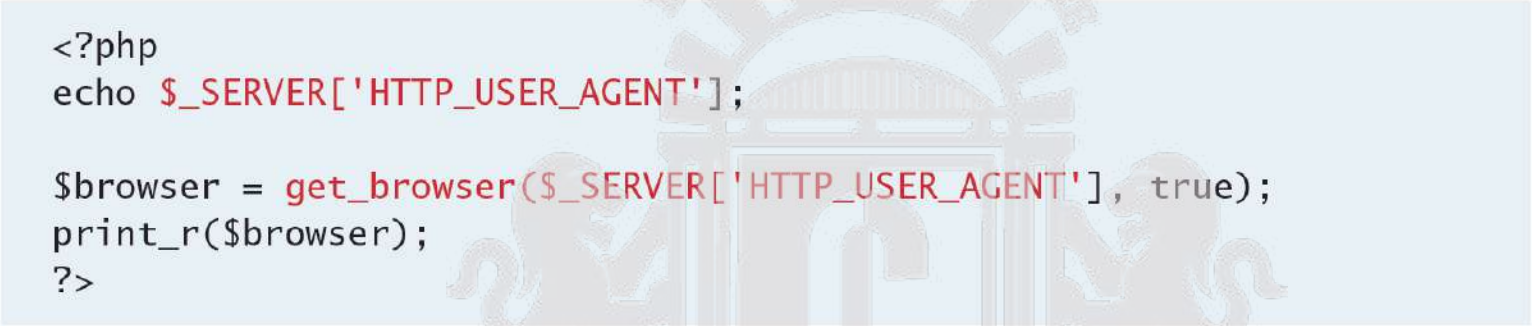
SERVER INFORMATION KEYS

- SERVER_NAME contains the name of the site that was requested
- SERVER_ADDR tells us the IP of the server
- DOCUMENT_ROOT tells us the location from which you are currently running your script
- SCRIPT_NAME key that identifies the actual script being executed

Request Header Keys

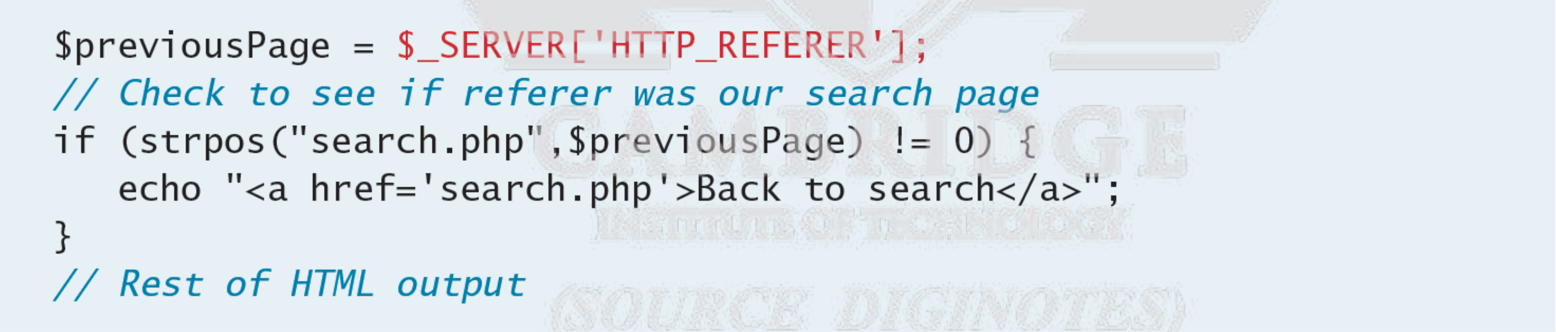
- REQUEST_METHOD returns the request method that was used to access the page: that is, GET, HEAD, POST, PUT
- REMOTE_ADDR key returns the IP address of the requestor
- HTTP_USER_AGENT contains the operating system and browser that the client is using
- HTTP_REFERER contains the address of the page that referred us to this one (if any) through a link

Header Access Examples



```
<?php
echo $_SERVER['HTTP_USER_AGENT'];

$browser = get_browser($_SERVER['HTTP_USER_AGENT'], true);
print_r($browser);
?>
```

LISTING 9.10 Accessing the user-agent string in the HTTP headers



```
$previousPage = $_SERVER['HTTP_REFERER'];
// Check to see if referer was our search page
if (strpos("search.php",$previousPage) != 0) {
   echo "<a href='search.php'>Back to search</a>";
}
// Rest of HTML output
```

LISTING 9.11 Using the HTTP_REFERER header to provide context-dependent output

Security

Headers can be forged

All headers can be forged!

- The HTTP_REFERER header can lie about where the referral came from
- The USER_AGENT can lie about the operating system and browser the client is using.

9.4 $_Files Array

- The $_FILES associative array contains items that have been uploaded to the current script.

<input type = “file”>

Creates a user interface for uploading a file from the client to server.

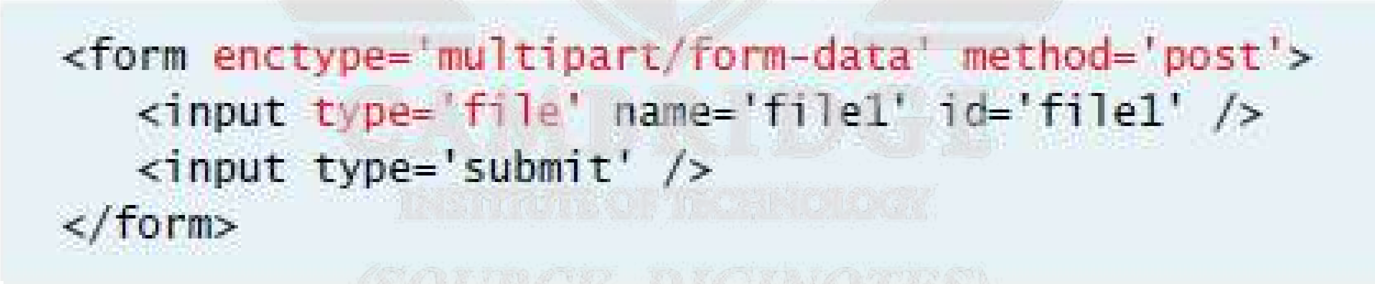
- A server Script must process the upload files in some way ($_FILES array helps in this process)

9.4.1 HTML Required for File Uploads

- To allow users to upload files, there are some specific things you must do,
  - First, you must ensure that the HTML form uses the HTTP post method, since transmitting a flie through the URL is not possible.
  - Second, You must add the enctype= “multipart/form-data” attribute to the html form that is performing the upload so that the HTTP request can submit multiple pieces of data (HTTP post body, the HTTP file attachment itself)

– Finally you must include an input type of file in your form.

This will show up with a browse button beside it so the user can select a file from their computer to be uploaded.

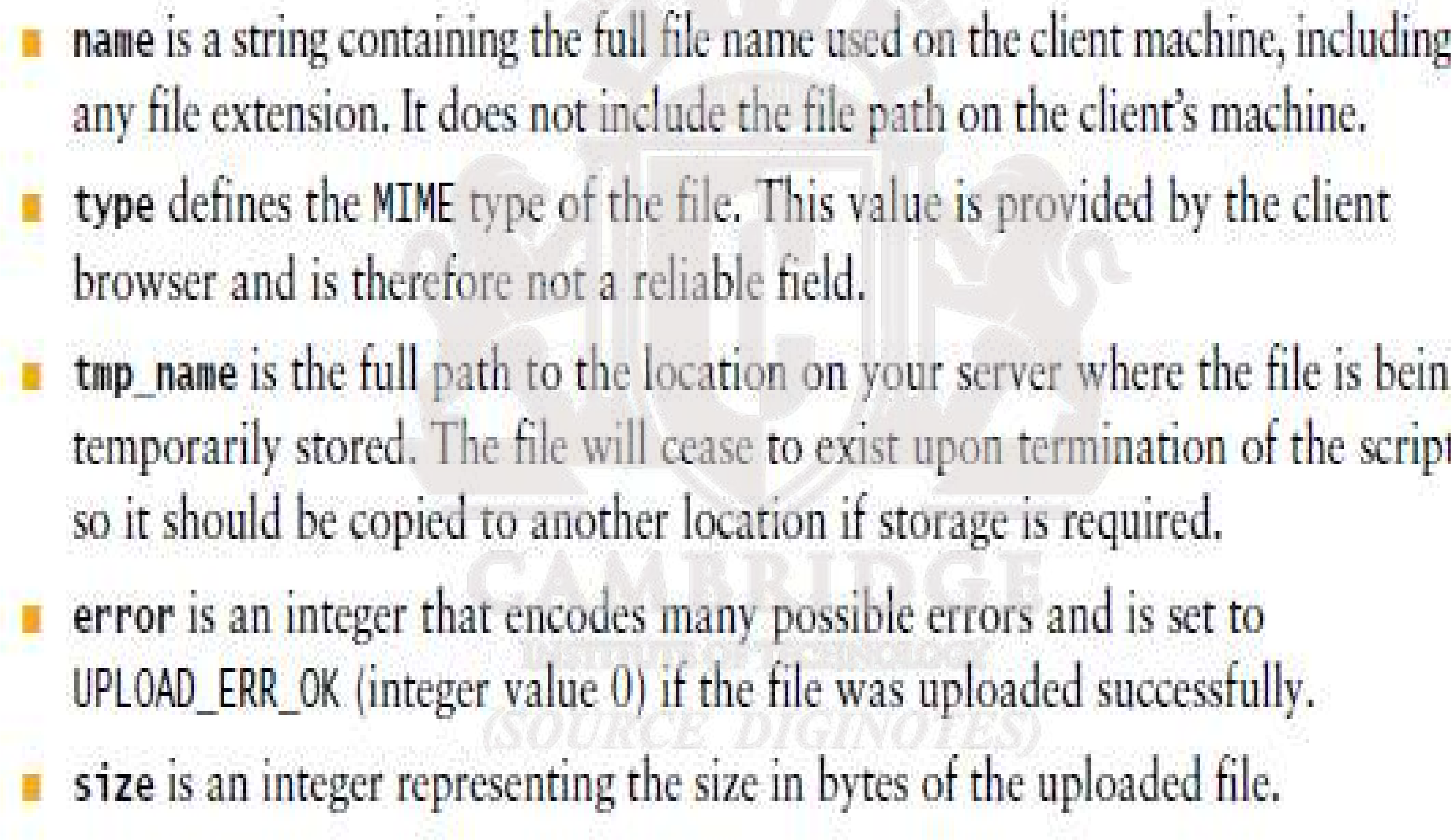


```
<form enctype='multipart/form-data' method='post'>
   <input type='file' name='file1' id='file1' />
   <input type='submit' />
</form>
```

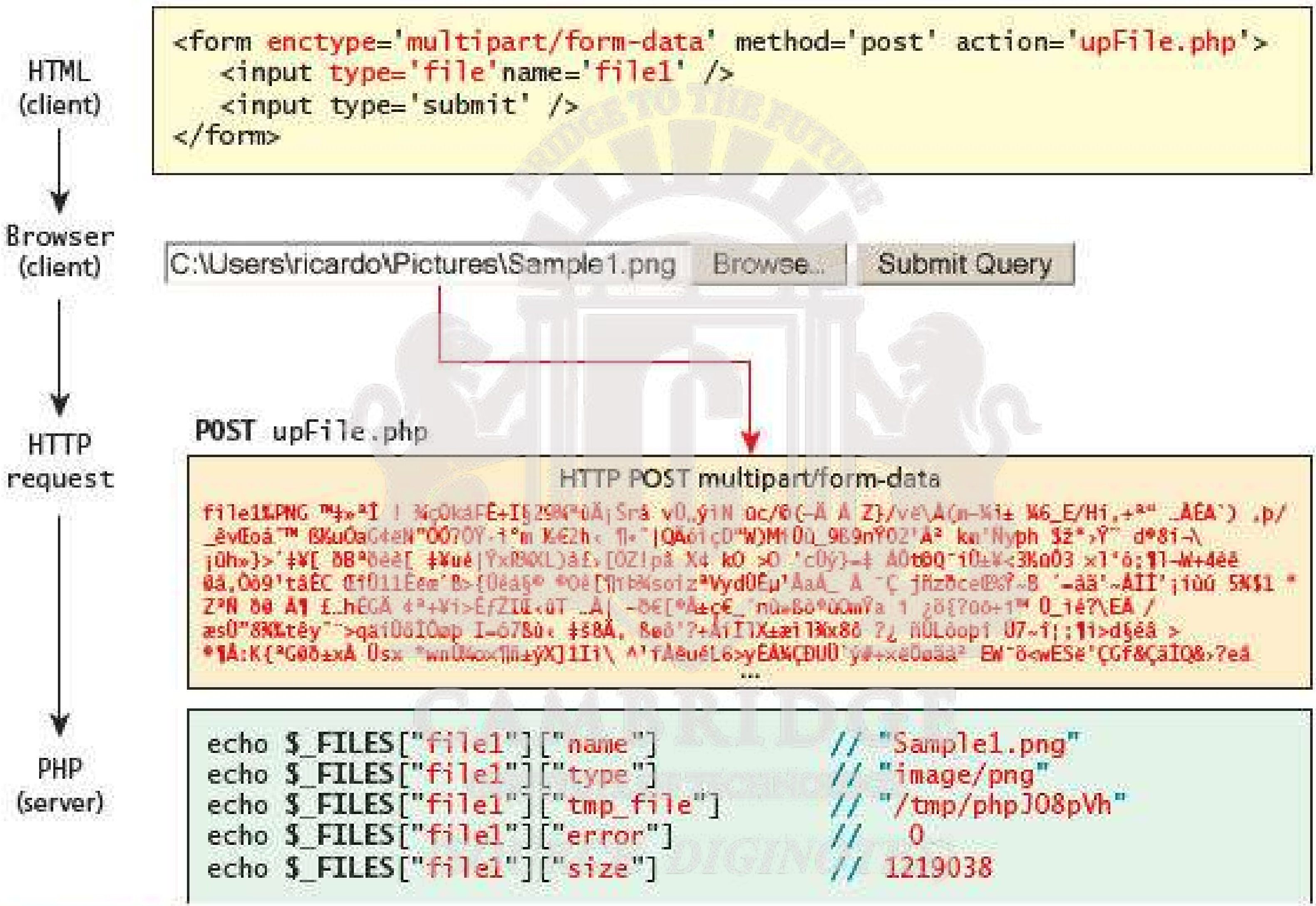
LISTING 9.12 HTML for a form that allows an upload

9.4.2 Handling the File Upload in PHP

- The Corresponding PHP file responsible for handling the upload will utilize the superglobal $_FILES array.
- This array will contain a key = value pair for each file uploaded in the post.
- The key for each element will be the name attribute from the HTML form, while the value will be an array containing information about the file as well as the file itself.
- The keys in that array are the name, type, tmp_name, error and size.



- **name** is a string containing the full file name used on the client machine, including any file extension. It does not include the file path on the client's machine.
- **type** defines the MIME type of the file. This value is provided by the client browser and is therefore not a reliable field.
- **tmp_name** is the full path to the location on your server where the file is bein temporarily stored. The file will cease to exist upon termination of the scrip so it should be copied to another location if storage is required.
- **error** is an integer that encodes many possible errors and is set to UPLOAD_ERR_OK (integer value 0) if the file was uploaded successfully.
- **size** is an integer representing the size in bytes of the uploaded file.



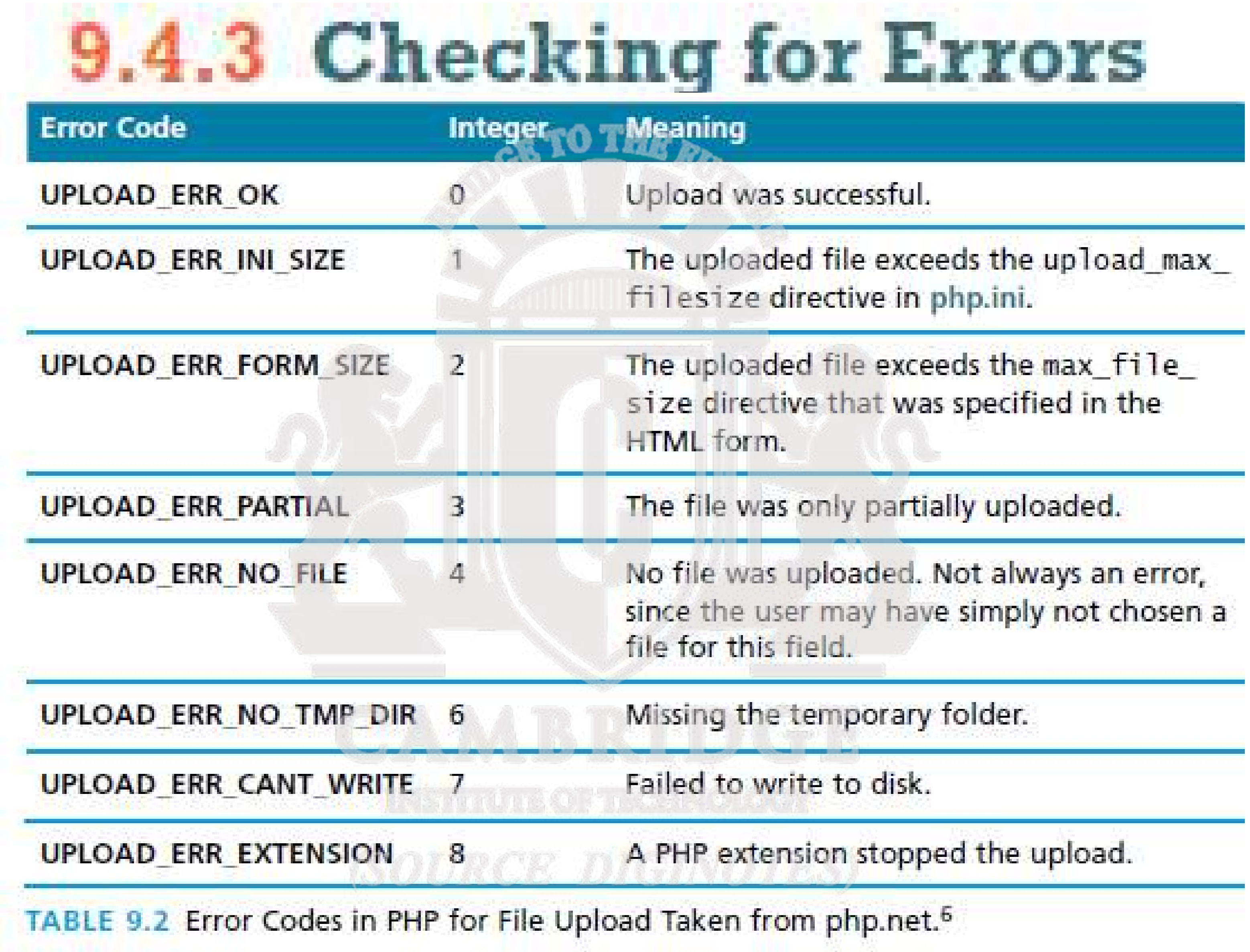
```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file'name='file1' />
   <input type='submit' />
</form>
```

C:\Users\ricardo\Pictures\Sample1.png
Browse...
Submit Query
POST upFile.php
HTTP POST multipart/form-data

```
file1‰PNG ™‡»ªÎ ! ¾çÒkäFÊ+I§29¾"ûÄ¡Šrä vÒ„ýiN ûc/®(-Ä Â Z}/vë\Å(m-¾i± ¾6_E/Hi,+ª" _ÅÉÀ`) .p/
_èvŒoä¯™ ß¾uÒaG¢eN"ÒÖ?ÖY·ï°m ¾€2h‹ ¶•¨|QÅóiçD"W)Mr0ù_9ß9nŸ02'Ä² k¤'Nyph $ž*›Ÿ¯ d*8i¬\
¡Ûh»}>´‡¥[ ðB³ðèê[ ‡¥uè|ŸxR¾XL)ä£›[ÒZ!pä X¢ kÒ >Ò ´cÖý}—‡ ÂÒtðQ¯iÜ±¥<3‰ö03 ×1'ó;¶]—W+4èê
ØäˆÒö9'täÊC Œï011Êëæ´ß>{Ûèä§ª *Öè[¶ïb‰soizªVydÒÊµ'ÄäÅ_ Å ¯Ç jñzðceŒ%?~ß ´—äã'~ÄÌÏ'¡iùû 5%$1 *
ZªN ðø Ä¶ £…hÈGÄ ¢ª+¥i>ÈƒZIŒ‹ûT …Ä¦ -ð€[*Å±ç€_´ñÛ»ßö³ÜØmŸa 1 ¿ð{?oó+1™ 0_iè?\ÉÄ /
æsÒ"8‰£têy¯¨>qaiÛßÏÒæp I–ó7ßù‹ ‡§ßÅ, ßeö'?+ÄïÏ1X±æi1‰x8ð ?¿ ñÜLöopí Ü7~í¦;¶i>d§èä >
*¶Ä:K{ªGØð±xÄ Ûsx ªwnÜ¾o×¶ñ±ýX]1Ii\ ^'fÄ®uéL6>yÉÄ‰ÇÐÙÜ´ý#+×èØøää² ÈW¯ö<wESé'ÇGf&ÇäÏQ&›?eä
...
```

```
echo $_FILES["file1"]["name"]          // "Sample1.png"
echo $_FILES["file1"]["type"]          // "image/png"
echo $_FILES["file1"]["tmp_file"]      // "/tmp/phpJO8pVh"
echo $_FILES["file1"]["error"]         //   0
echo $_FILES["file1"]["size"]          // 1219038
```

FIGURE 9.12 Data flow from HTML form through POST to PHP $_FILES array

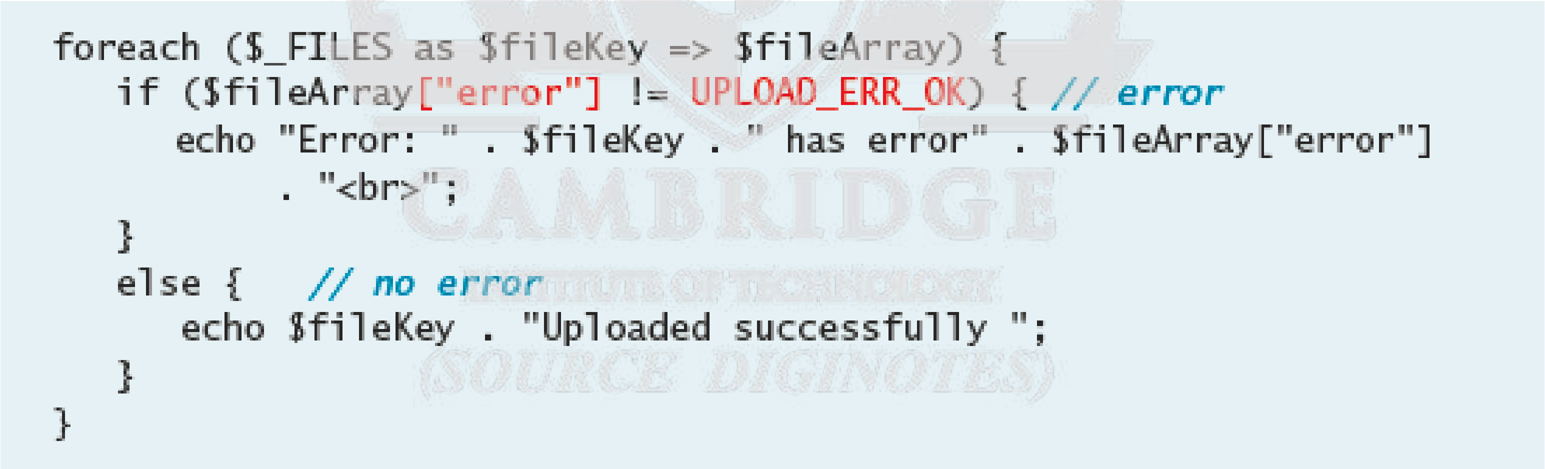


9.4.3 Checking for Errors

Error Code	Integer	Meaning
UPLOAD_ERR_OK	0	Upload was successful.
UPLOAD_ERR_INI_SIZE	1	The uploaded file exceeds the `upload_max_filesize` directive in php.ini.
UPLOAD_ERR_FORM_SIZE	2	The uploaded file exceeds the `max_file_size` directive that was specified in the HTML form.
UPLOAD_ERR_PARTIAL	3	The file was only partially uploaded.
UPLOAD_ERR_NO_FILE	4	No file was uploaded. Not always an error, since the user may have simply not chosen a file for this field.
UPLOAD_ERR_NO_TMP_DIR	6	Missing the temporary folder.
UPLOAD_ERR_CANT_WRITE	7	Failed to write to disk.
UPLOAD_ERR_EXTENSION	8	A PHP extension stopped the upload.

TABLE 9.2 Error Codes in PHP for File Upload Taken from php.net.[6]

- A proper file upload script will therefore check each uploaded file by checking the various error codes as below,

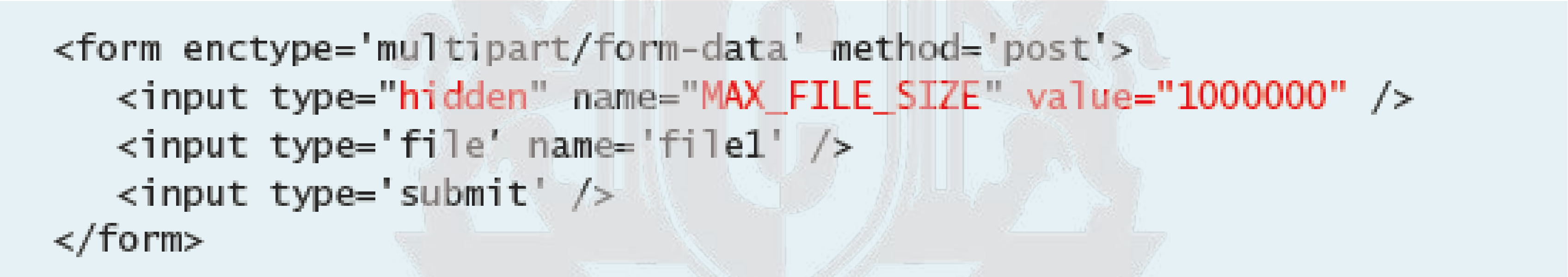


```
foreach ($_FILES as $fileKey => $fileArray) {
   if ($fileArray["error"] != UPLOAD_ERR_OK) { // error
      echo "Error: " . $fileKey . " has error" . $fileArray["error"]
            . "<br>";
   }
   else {   // no error
      echo $fileKey . "Uploaded successfully ";
   }
}
```

LISTING 9.13 Checking each file uploaded for errors

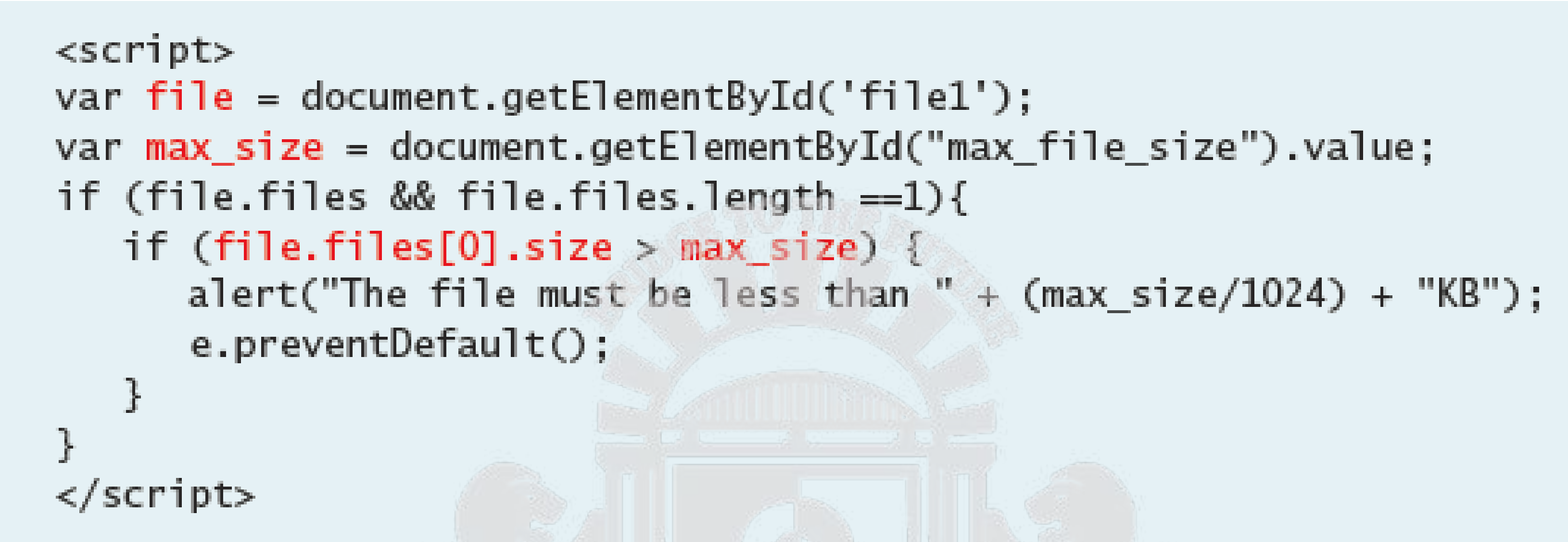
9.4.4 File Size Restrictions

- There are three main mechanisms for maintaining uploaded file size restrictions:
  - Via HTML in the input form
  - Via JavaScript in the input form
  - Via PHP coding.



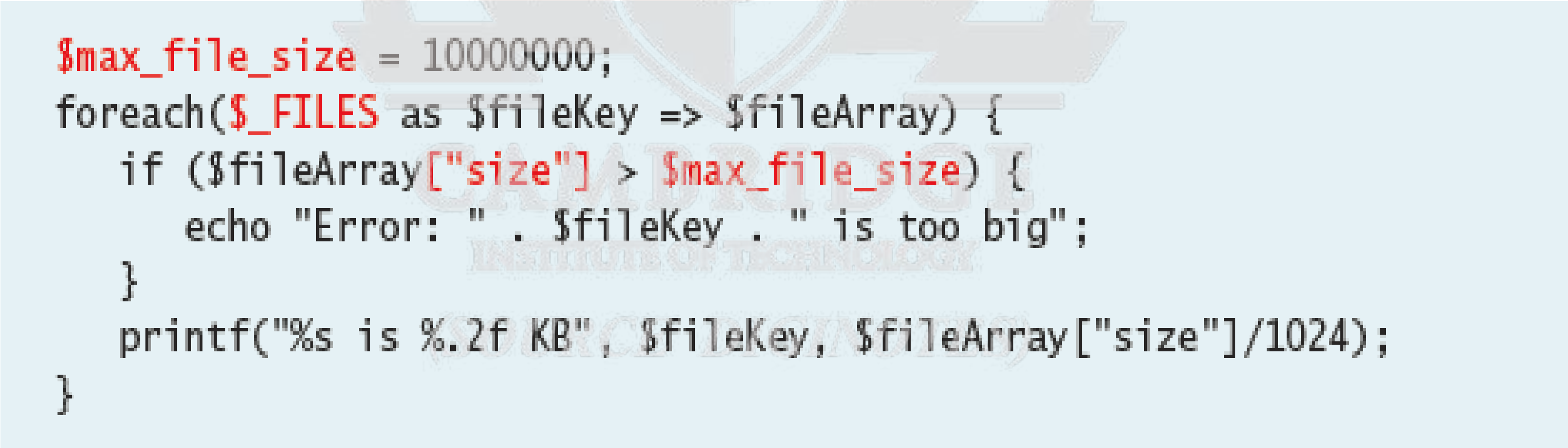
```
<form enctype='multipart/form-data' method='post'>
   <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
   <input type='file' name='file1' />
   <input type='submit' />
</form>
```

LISTING 9.14 Limiting upload file size via HTML



```
<script>
var file = document.getElementById('file1');
var max_size = document.getElementById("max_file_size").value;
if (file.files && file.files.length ==1){
   if (file.files[0].size > max_size) {
      alert("The file must be less than " + (max_size/1024) + "KB");
      e.preventDefault();
   }
}
</script>
```

LISTING 9.15 Limiting upload file size via JavaScript

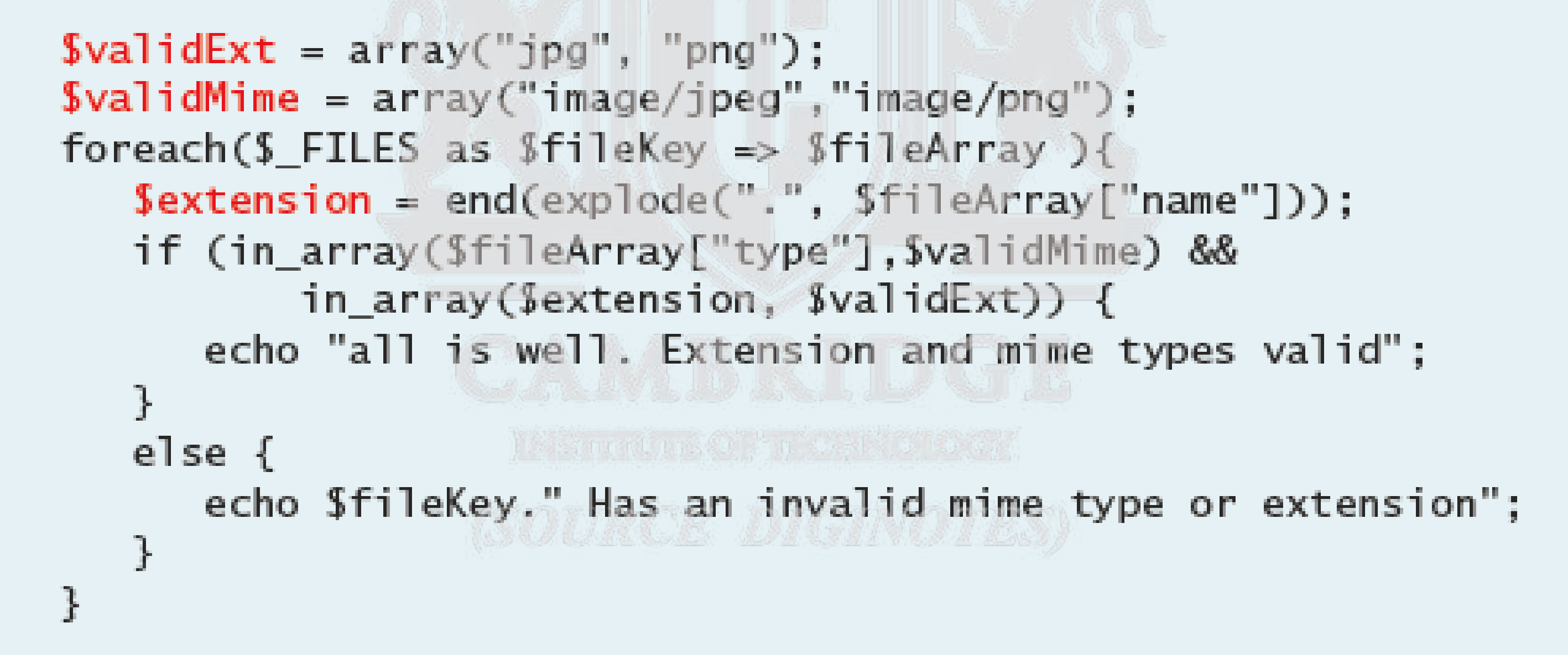


```
$max_file_size = 10000000;
foreach($_FILES as $fileKey => $fileArray) {
   if ($fileArray["size"] > $max_file_size) {
      echo "Error: " . $fileKey . " is too big";
   }
   printf("%s is %.2f KB", $fileKey, $fileArray["size"]/1024);
}
```

LISTING 9.16 Limiting upload file size via PHP

9.4.5 Limiting the Type of File Upload

- You should also restrict the type of file uploaded.

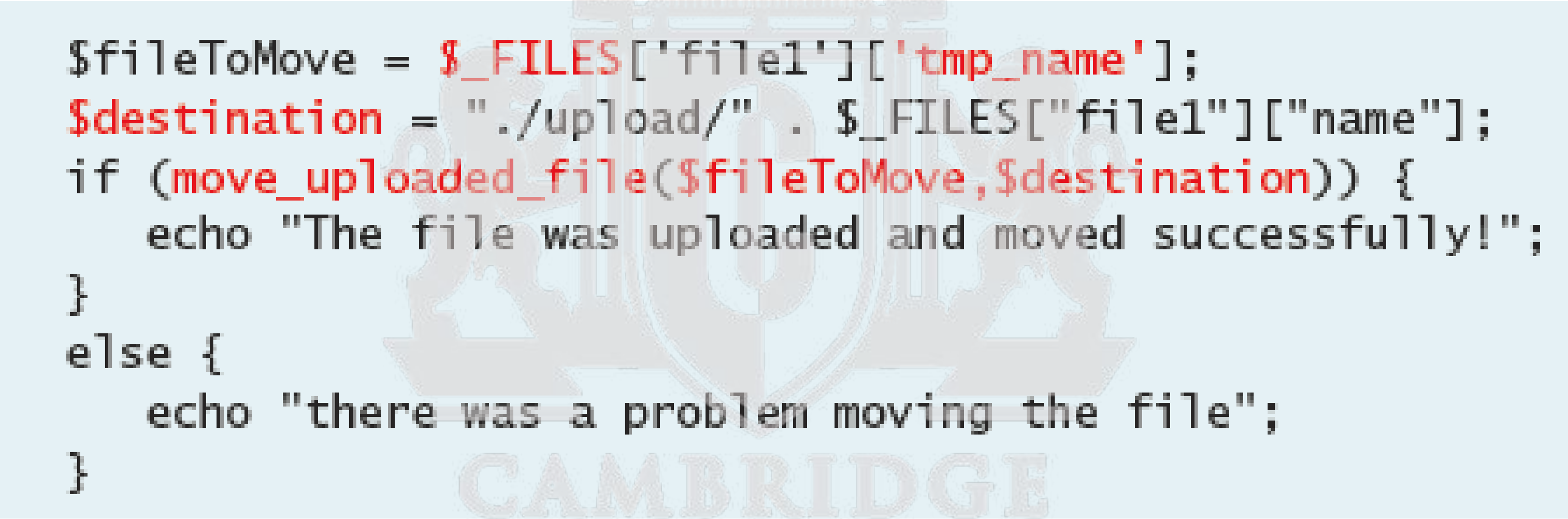


```
$validExt = array("jpg", "png");
$validMime = array("image/jpeg","image/png");
foreach($_FILES as $fileKey => $fileArray ){
   $extension = end(explode(".", $fileArray["name"]));
   if (in_array($fileArray["type"],$validMime) &&
          in_array($extension, $validExt)) {
      echo "all is well. Extension and mime types valid";
   }
   else {
      echo $fileKey." Has an invalid mime type or extension";
   }
}
```

LISTING 9.17 PHP code to look for valid mime types and file extensions

9.4.6 Moving the File

- You can make use of PHP function move_uploaded_file, which takes in the temporary file location and the file's final destination.
- This function will work only if the source file exist and if the destination location is writable by web server.



```
$fileToMove = $_FILES['file1']['tmp_name'];
$destination = "./upload/" . $_FILES["file1"]["name"];
if (move_uploaded_file($fileToMove,$destination)) {
   echo "The file was uploaded and moved successfully!";
}
else {
   echo "there was a problem moving the file";
}
```

LISTING 9.18 Using move_uploaded_file() function

9.5 Reading/Writing Files

- There are two basic techniques for read/writing files in PHP
  - Stream Access : In this technique, our code will read just a small portion of the file at a time. While this does require more careful programming, it is the most efficient approach when reading large files.
  - All – In – Memory access: In this technique, we can read the entire file into memory (I.e., into PHP variable). While not appropriate for large files, it does make processing of file extremely easy.

9.5.1 Stream Access

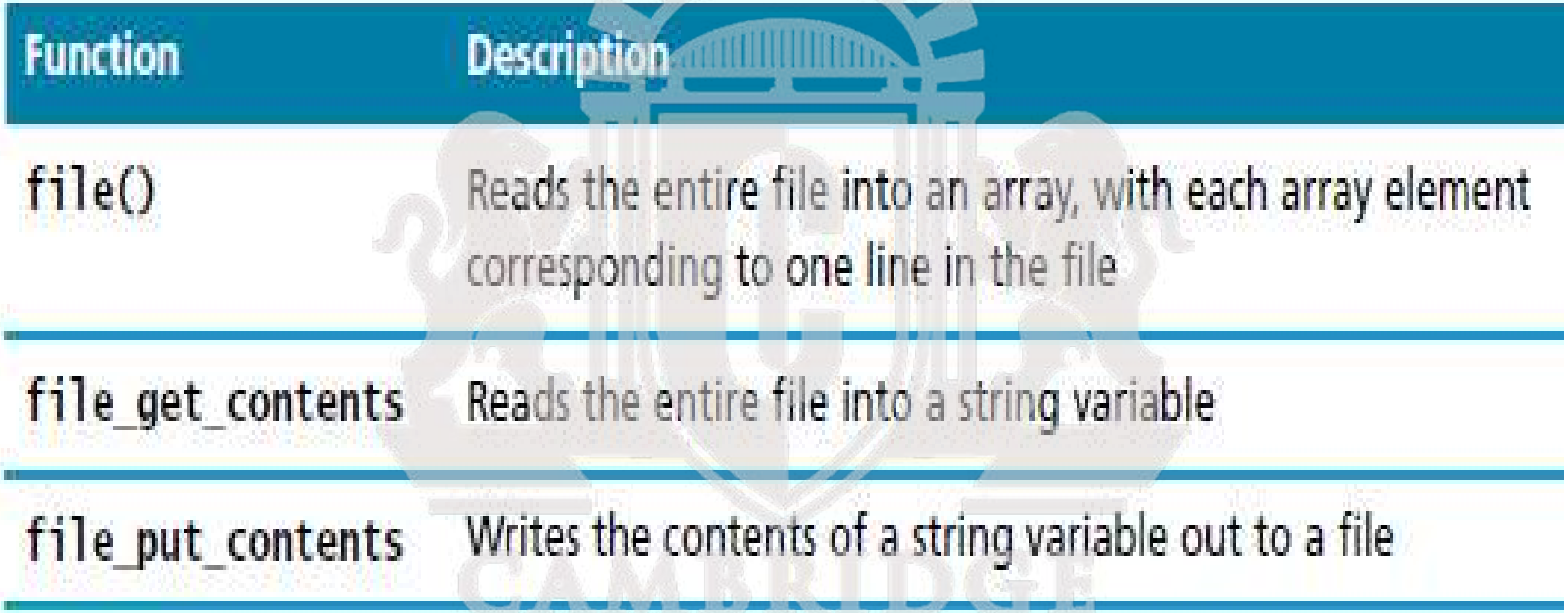
The function fopen() takes a file location or URL and access mode as parameters. The returned value is a stream resource, which you can then read sequentially. Some of the common modes are “r” for read, “rw” for read and write, and “c,” which creates a new file for writing.

Once the file is opened, you can read from it in several ways. To read a single line, use the fgets() function, which will return false if there is no more data, and if it reads a line it will advance the stream forward to the next one so you can use the === check to see if you have reached the end of the file. To read an arbitrary amount of data (typically for binary files), use fread() and for reading a single character use fgetsc(). Finally, when finished processing the file you must close it using fclose(). Listing 9.19 illustrates a script using fopen(), fgets(), and fclose() to read a file and echo it out (replacing new lines with <br> tags).

```
$f = fopen("sample.txt", "r");
$ln = 0;
while ($line = fgets($f)) {
   $ln++;
   printf("%2d: ", $ln);
   echo $line . "<br>";
}
fclose($f);
```

LISTING 9.19 Opening, reading lines, and closing a file

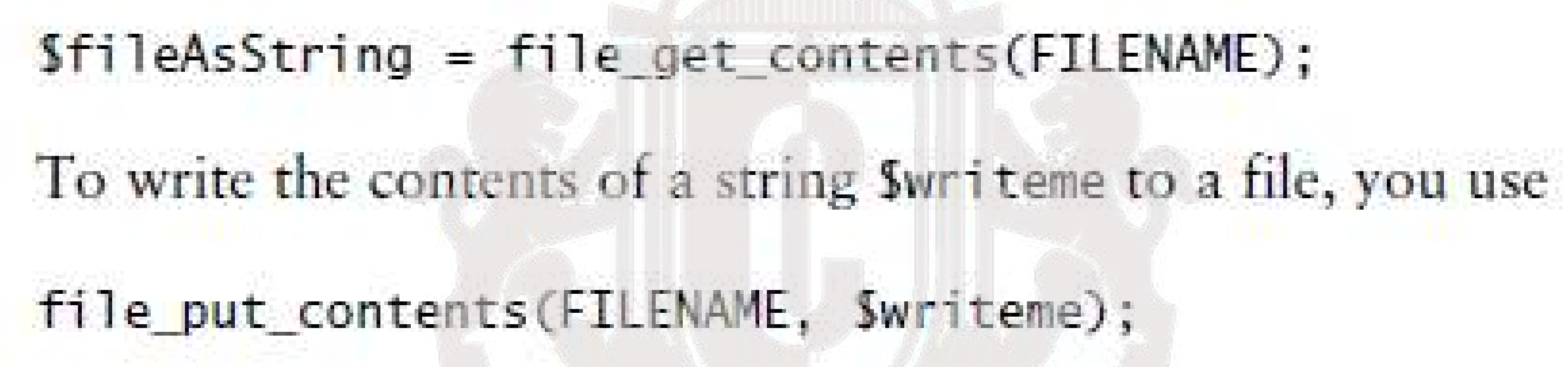
9.5.2 In-Memory File Access



Function	Description
file()	Reads the entire file into an array, with each array element corresponding to one line in the file
file_get_contents	Reads the entire file into a string variable
file_put_contents	Writes the contents of a string variable out to a file

TABLE 9.3 In-Memory File Functions

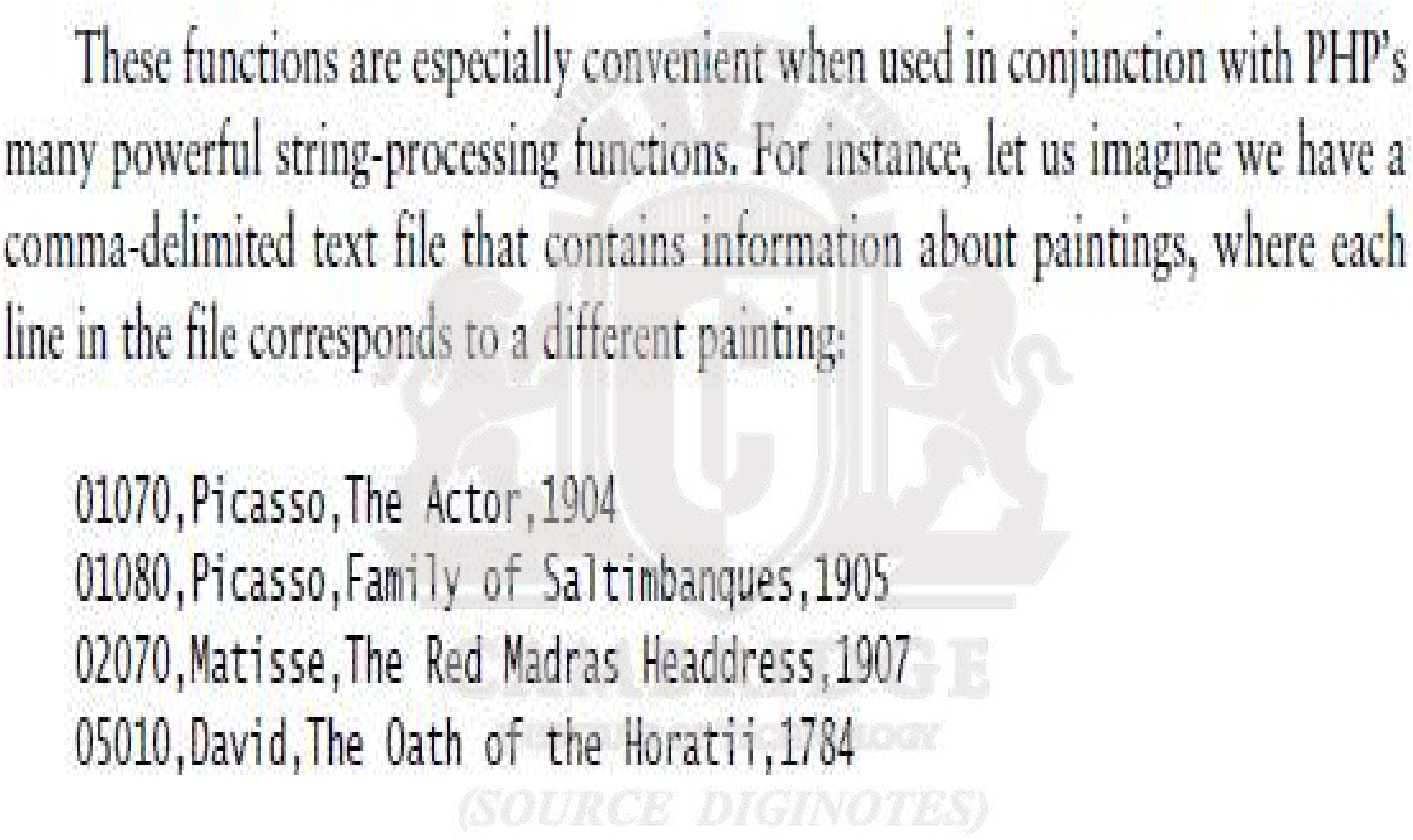
To Read an entire file into variable



```
$fileAsString = file_get_contents(FILENAME);
```

To write the contents of a string `$writeme` to a file, you use

```
file_put_contents(FILENAME, $writeme);
```



These functions are especially convenient when used in conjunction with PHP's many powerful string-processing functions. For instance, let us imagine we have a comma-delimited text file that contains information about paintings, where each line in the file corresponds to a different painting:

```
01070,Picasso,The Actor,1904
01080,Picasso,Family of Saltimbanques,1905
02070,Matisse,The Red Madras Headdress,1907
05010,David,The Oath of the Horatii,1784
```

To read and then parse this text file is quite straightforward, as shown in Listing 9.20.

```
// read the file into memory; if there is an error then stop processing
$paintings = file($filename) or die('ERROR: Cannot find file');

// our data is comma-delimited
$delimiter = ',';

// loop through each line of the file
foreach ($paintings as $painting) {

   // returns an array of strings where each element in the array
   // corresponds to each substring between the delimiters
   $paintingFields = explode($delimiter, $painting);

   $id= $paintingFields[0];
   $artist = $paintingFields[1];
   $title = $paintingFields[2];
   $year = $paintingFields[3];

   // do something with this data
   . . .
}
```

LISTING 9.20 Processing a comma-delimited file

PHP Classes and Objects

Chapter 10

Section 1 of 3

OBJECT-ORIENTED OVERVIEW

Overview

Object-Oriented Overview

PHP is a full-fledged object-oriented language with many of the syntactic constructs popularized in languages like Java and C++.

Earlier versions of PHP do not support all of these object-oriented features,

- PHP versions after 5.0 do

Terminology

Object-Oriented Terminology

❑The notion of programming with objects allows the developer to think about an item with particular **properties** (also called attributes or **data members**) and methods (functions).

❑The structure of these **objects** is defined by **classes**, which outline the properties and methods like a blueprint.

❑Each variable created from a class is called an object or **instance**, and each object maintains its own set of variables, and behaves (largely) independently from the class once created.

Relationship between Class and Objects

SULLIVAN · S
ELEMENTARY ALGEBRA
Database Processing
David M. Kroenke · David J. Auer
Curious Writer
Bruce Ballenger

Book class

Defines properties such as:
title, author, and number of pages

Objects (or instances of the Book class)

Each instance has its own title, author, and number of pages property values

UML

The Unified Modelling Language

- The standard diagramming notation for object-oriented design is **UML (Unified Modeling Language).**

- Class diagrams and object diagrams, in particular, are useful to us when describing the properties, methods, and relationships between classes and objects.

- For a complete definition of UML modeling syntax, look at the Object Modeling Group’s living specification

UML Class diagram

By example

Every Artist has a

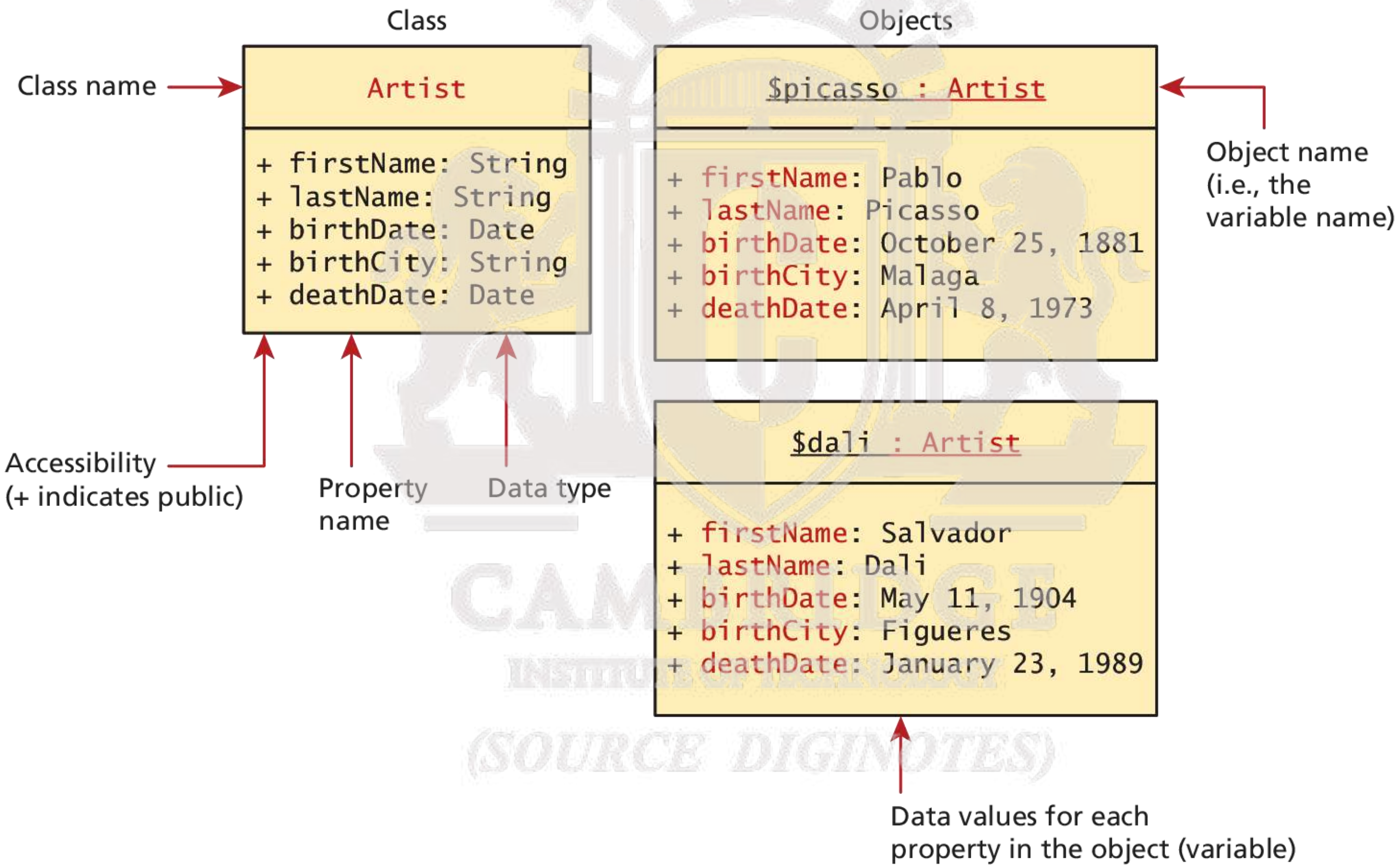
- first name,
- last name,
- birth date,
- birth city, and
- death date.

✓Using objects we can encapsulate those properties together into a class definition for an Artist.

✓UML articulates that design

UML Class diagram

Class and a couple of objects



Class

Objects

Class name

Artist

+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date

$picasso : Artist

+ firstName: Pablo
+ lastName: Picasso
+ birthDate: October 25, 1881
+ birthCity: Malaga
+ deathDate: April 8, 1973

Object name (i.e., the variable name)

$dali : Artist

+ firstName: Salvador
+ lastName: Dali
+ birthDate: May 11, 1904
+ birthCity: Figueres
+ deathDate: January 23, 1989

Accessibility (+ indicates public)

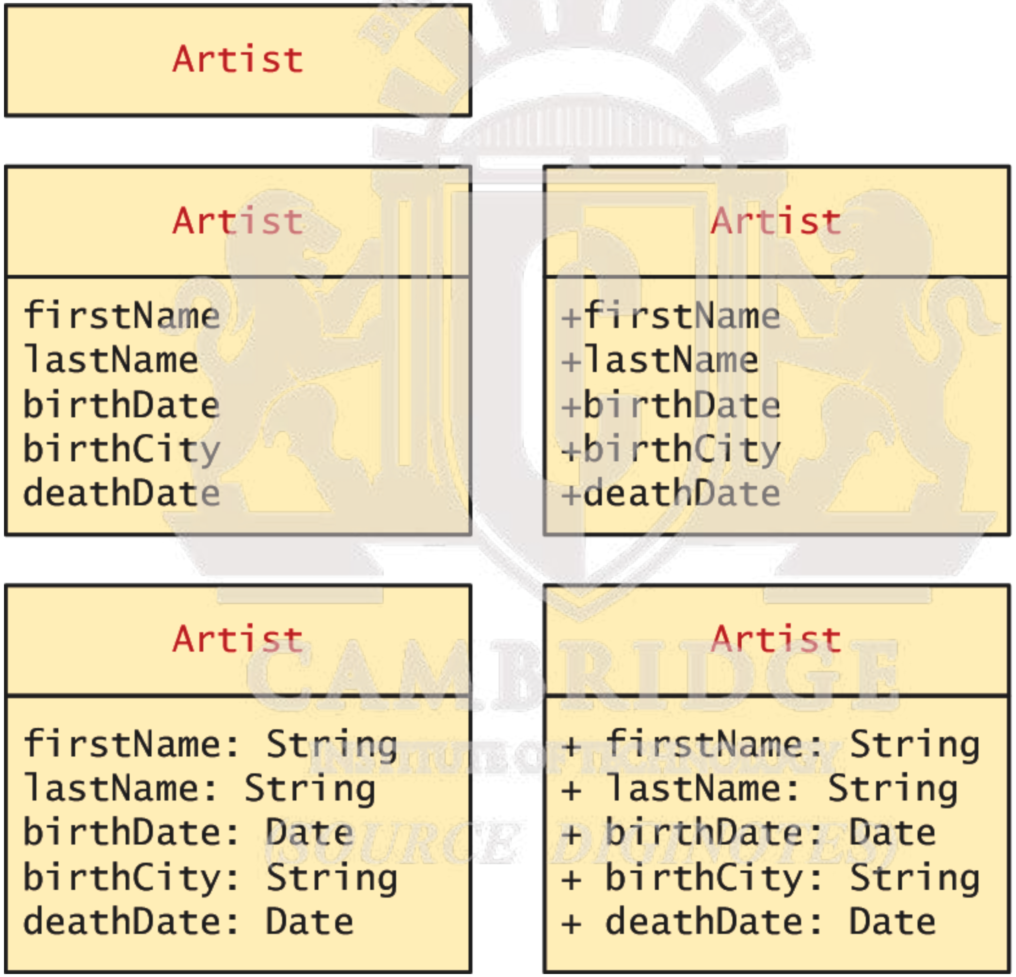
Property name

Data type

Data values for each property in the object (variable)

UML Class diagram

Different levels of detail



Artist

Artist

firstName
lastName
birthDate
birthCity
deathDate

Artist

+firstName
+lastName
+birthDate
+birthCity
+deathDate

Artist

firstName: String
lastName: String
birthDate: Date
birthCity: String
deathDate: Date

Artist

+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date

Server and Desktop Objects

Not the same

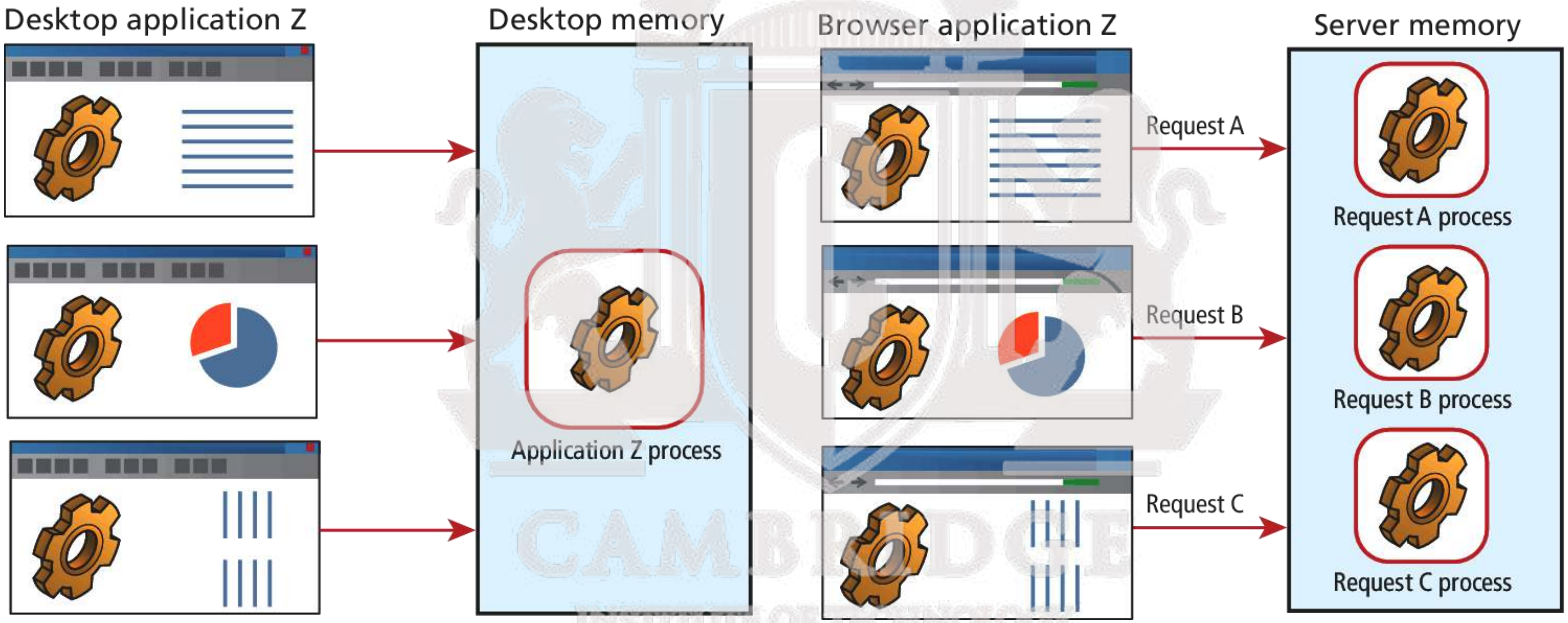
❖While desktop software can load an object into memory and make use of it for several user interactions, a PHP object is loaded into memory only for the life of that HTTP request.

❖We must use classes differently than in the desktop world, since the object must be recreated and loaded into memory

❖Unlike a desktop, there are potentially many thousands of users making requests at once, so not only are objects destroyed upon responding to each request, but memory must be shared between many simultaneous requests, each of which may load objects into memory or each request that requires it

Server and Desktop Objects

Not the same



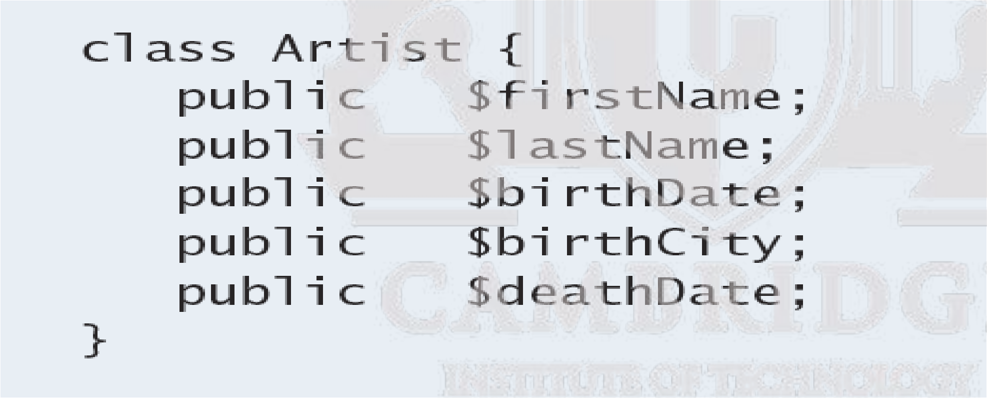
Section 2 of 3

OBJECTS AND CLASSES IN PHP

Defining Classes

In PHP

The PHP syntax for defining a class uses the class keyword followed by the class name and { } braces



```
class Artist {
    public   $firstName;
    public   $lastName;
    public   $birthDate;
    public   $birthCity;
    public   $deathDate;
}
```

LISTING 10.1 A simple Artist class

Instantiating Objects

In PHP

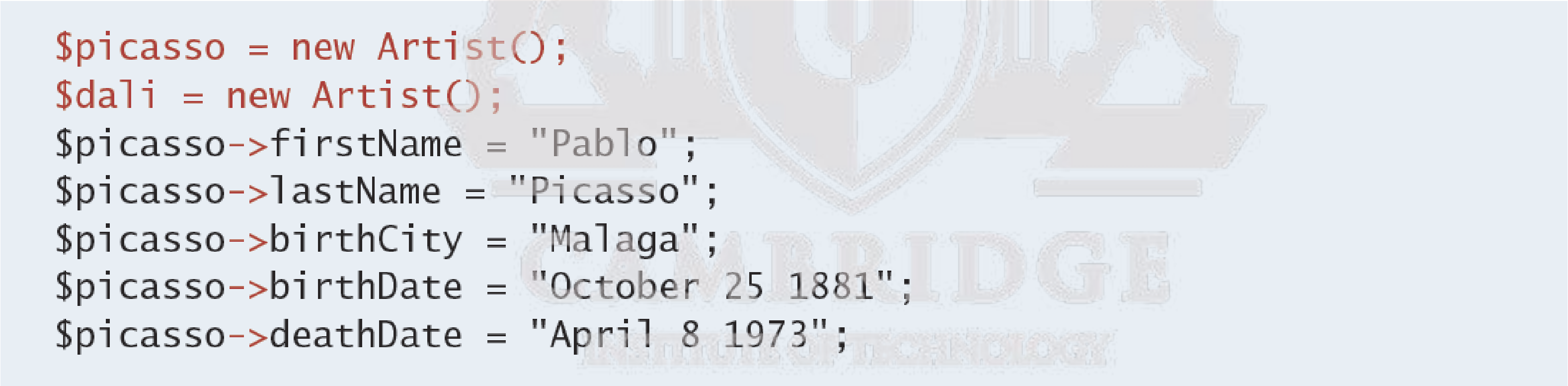
Defining a class is not the same as using it. To make use of a class, one must **instantiate** (create) objects from its definition using the *new* keyword.

```
$picasso = new Artist();

$dali = new Artist();
```

Properties

The things in the objects

Once you have instances of an object, you can access and modify the properties of each one separately using the variable name and an arrow (->).



```
$picasso = new Artist();
$dali = new Artist();
$picasso->firstName = "Pablo";
$picasso->lastName = "Picasso";
$picasso->birthCity = "Malaga";
$picasso->birthDate = "October 25 1881";
$picasso->deathDate = "April 8 1973";
```

LISTING 10.2 Instantiating two Artist objects and setting one of those object's properties

Constructors

A Better way to build

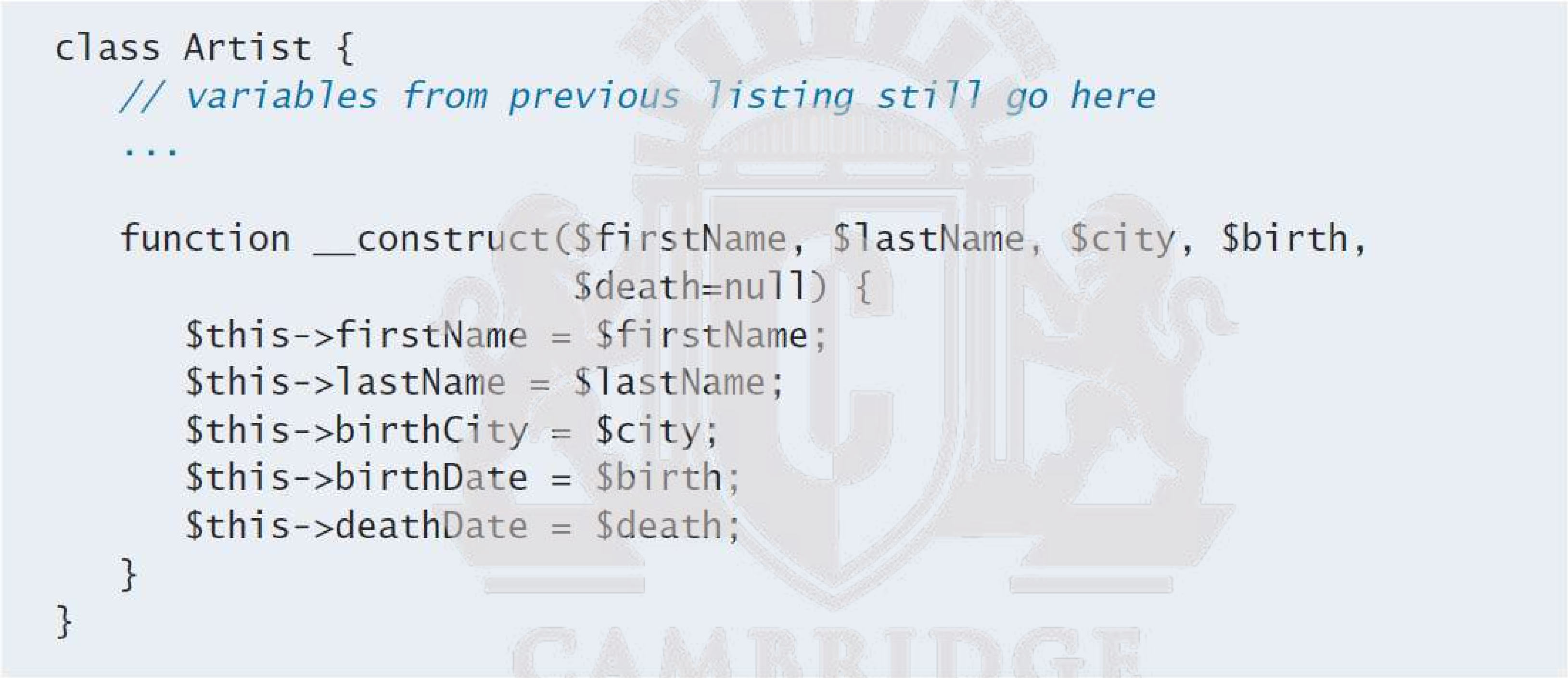
Constructors let you specify parameters during instantiation to initialize the properties within a class right away.

In PHP, constructors are defined as functions (as you shall see, all methods use the function keyword) with the name **__construct().**

Notice that in the constructor each parameter is assigned to an internal class variable using the $this-> syntax. you **must** always use the $this syntax to reference all properties and methods associated with this particular instance of a class.

Constructors

An Example



```
class Artist {
   // variables from previous listing still go here
   ...

   function __construct($firstName, $lastName, $city, $birth,
                        $death=null) {
      $this->firstName = $firstName;
      $this->lastName = $lastName;
      $this->birthCity = $city;
      $this->birthDate = $birth;
      $this->deathDate = $death;
   }
}
```

LISTING 10.3 A constructor added to the class definition

Constructors

Using the constructor

```
$picasso = new Artist("Pablo","Picasso","Malaga","Oct 25,1881","Apr 8,1973");

$dali = new Artist("Salvador","Dali","Figures","May 11 1904", "Jan 23 1989");
```

Methods

Functions In a class

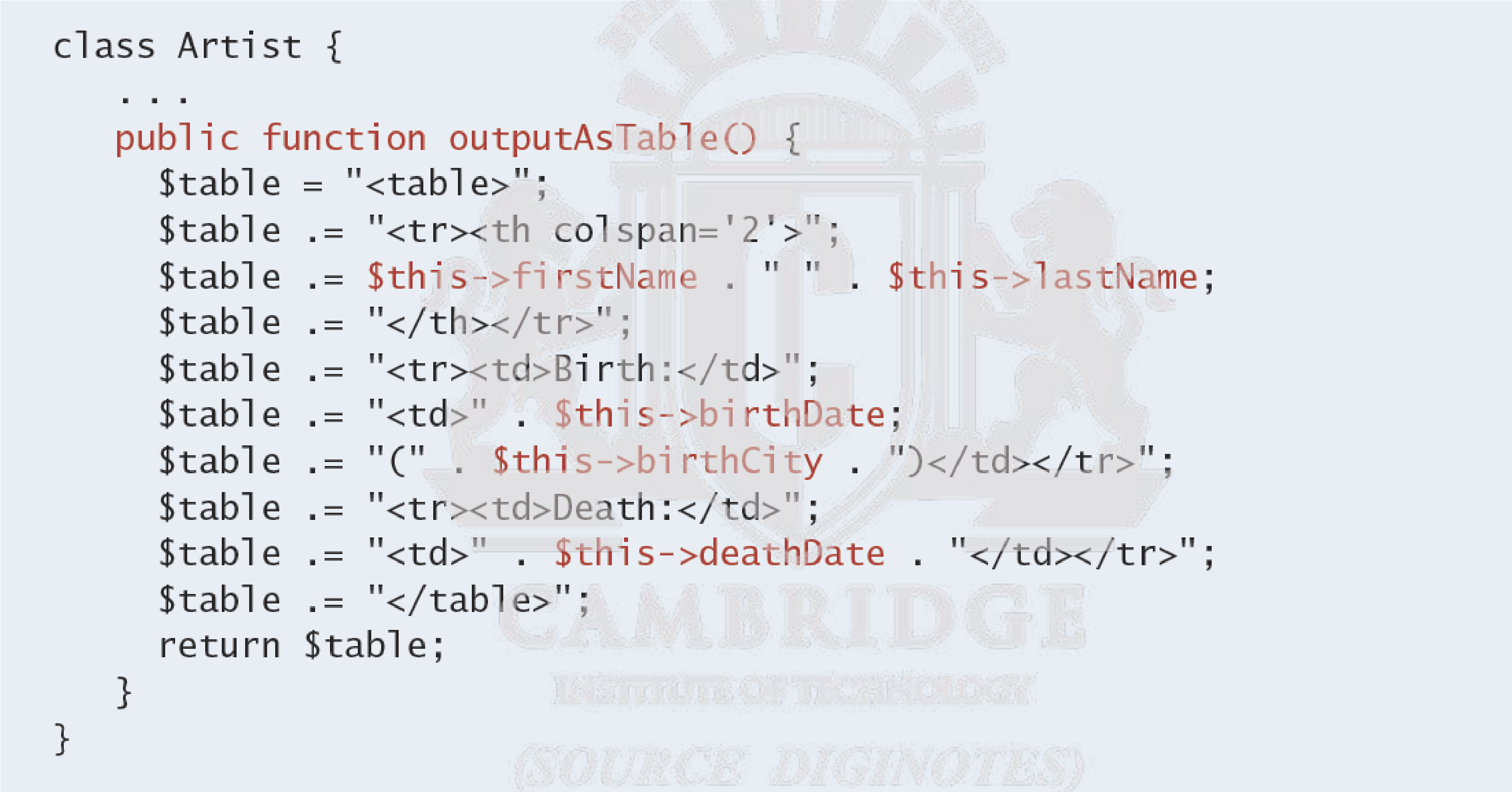
Methods and are like functions, except they are associated with a class.

They define the tasks each instance of a class can perform and are useful since they associate behavior with objects.

```
$picasso = new Artist( . . . )

echo $picasso->outputAsTable();
```

Methods

The example definition

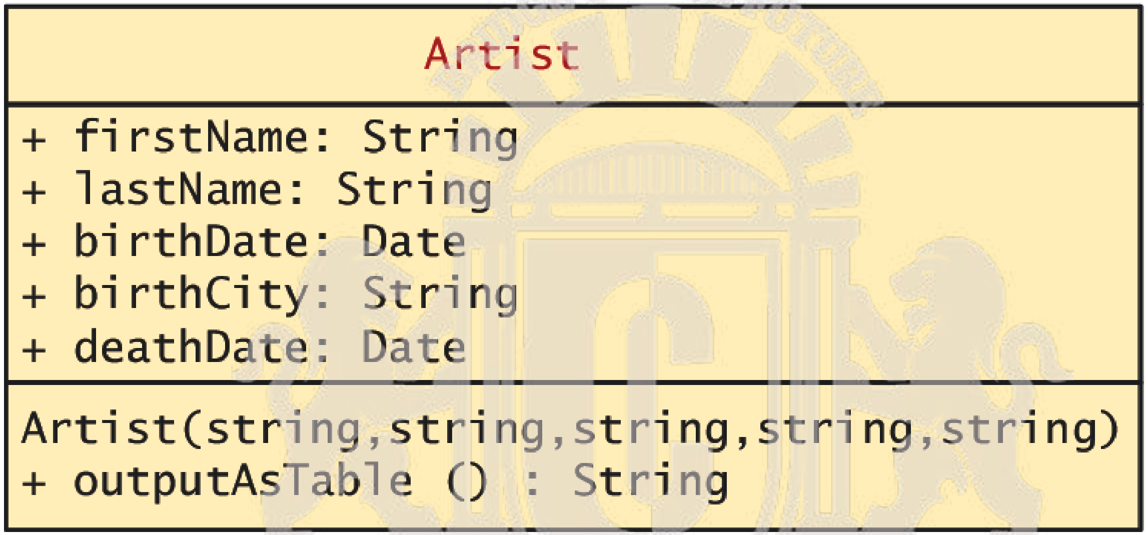


```
class Artist {
   . . .
   public function outputAsTable() {
      $table = "<table>";
      $table .= "<tr><th colspan='2'>";
      $table .= $this->firstName . " " . $this->lastName;
      $table .= "</th></tr>";
      $table .= "<tr><td>Birth:</td>";
      $table .= "<td>" . $this->birthDate;
      $table .= "(" . $this->birthCity . ")</td></tr>";
      $table .= "<tr><td>Death:</td>";
      $table .= "<td>" . $this->deathDate . "</td></tr>";
      $table .= "</table>";
      return $table;
   }
}
```

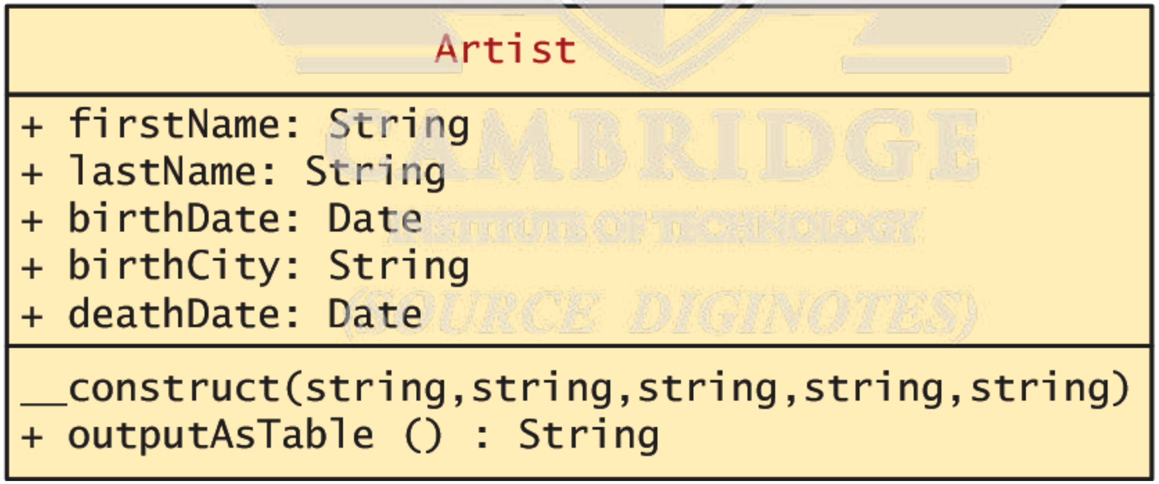
LISTING 10.4 Method definition

Methods

UML class diagrams adding the method



```
Artist
+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date
Artist(string,string,string,string,string)
+ outputAsTable () : String
```



```
Artist
+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date
__construct(string,string,string,string,string)
+ outputAsTable () : String
```

Visibility

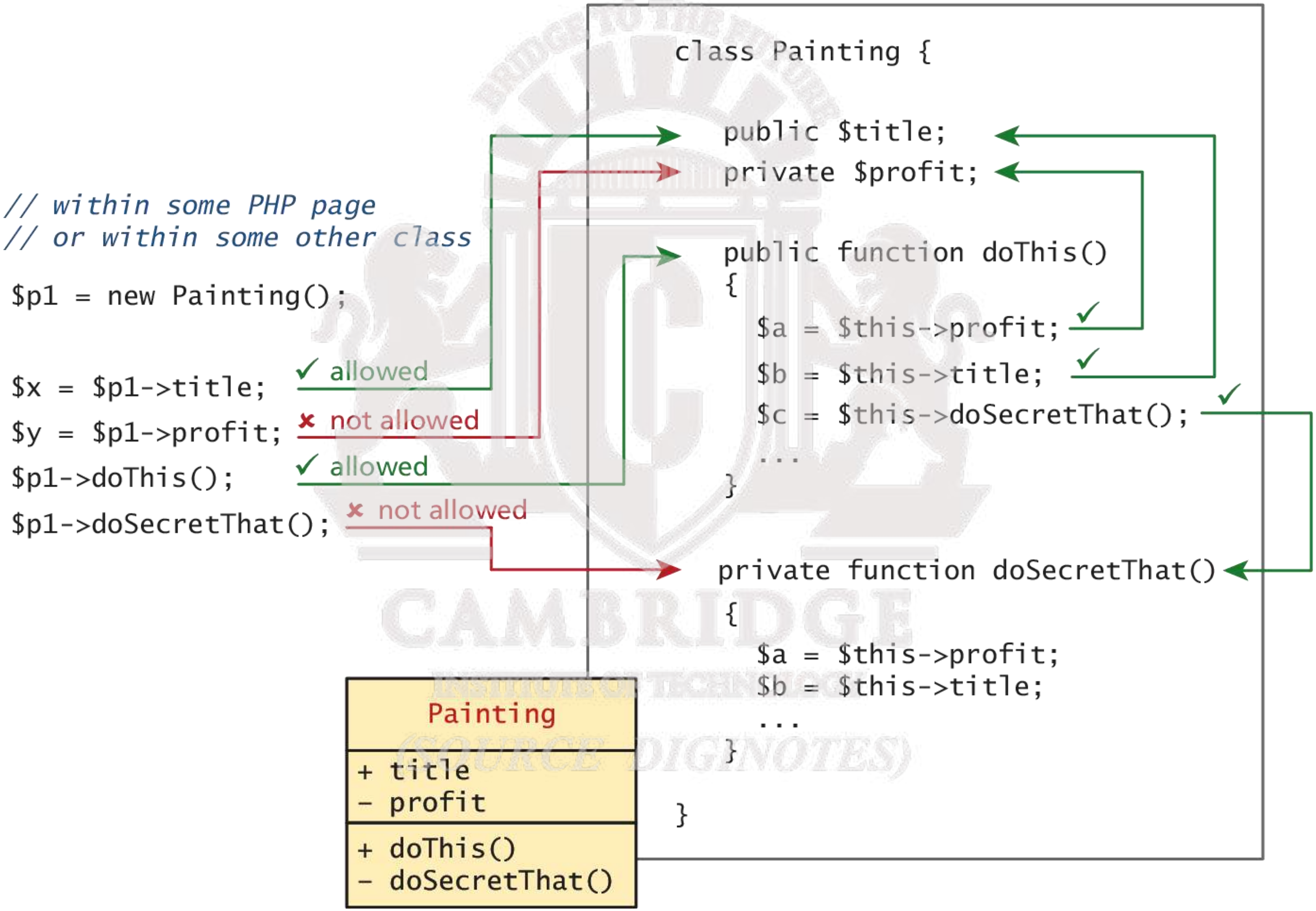
Or accessibility

The **visibility** of a property or method determines the accessibility of a **class member** and can be set to:

- **Public** the property or method is accessible to any code that has a reference to the object
- **Private** sets a method or variable to only be accessible from within the class
- **Protected** is related to inheritance…

Visibility

Or accessibility



```
// within some PHP page
// or within some other class

$p1 = new Painting();

$x = $p1->title;        ✓ allowed
$y = $p1->profit;       ✘ not allowed
$p1->doThis();          ✓ allowed
$p1->doSecretThat();      ✘ not allowed
```

```
class Painting {

    public $title;
    private $profit;

    public function doThis()
    {
        $a = $this->profit;  ✓
        $b = $this->title;   ✓
        $c = $this->doSecretThat();  ✓
        ...
    }

   private function doSecretThat()
    {
        $a = $this->profit;
        $b = $this->title;
        ...
    }

}
```

```
    Painting
+ title
- profit
+ doThis()
- doSecretThat()
```

Source diginotes.in

Save the Earth. Go paperless

Static Members

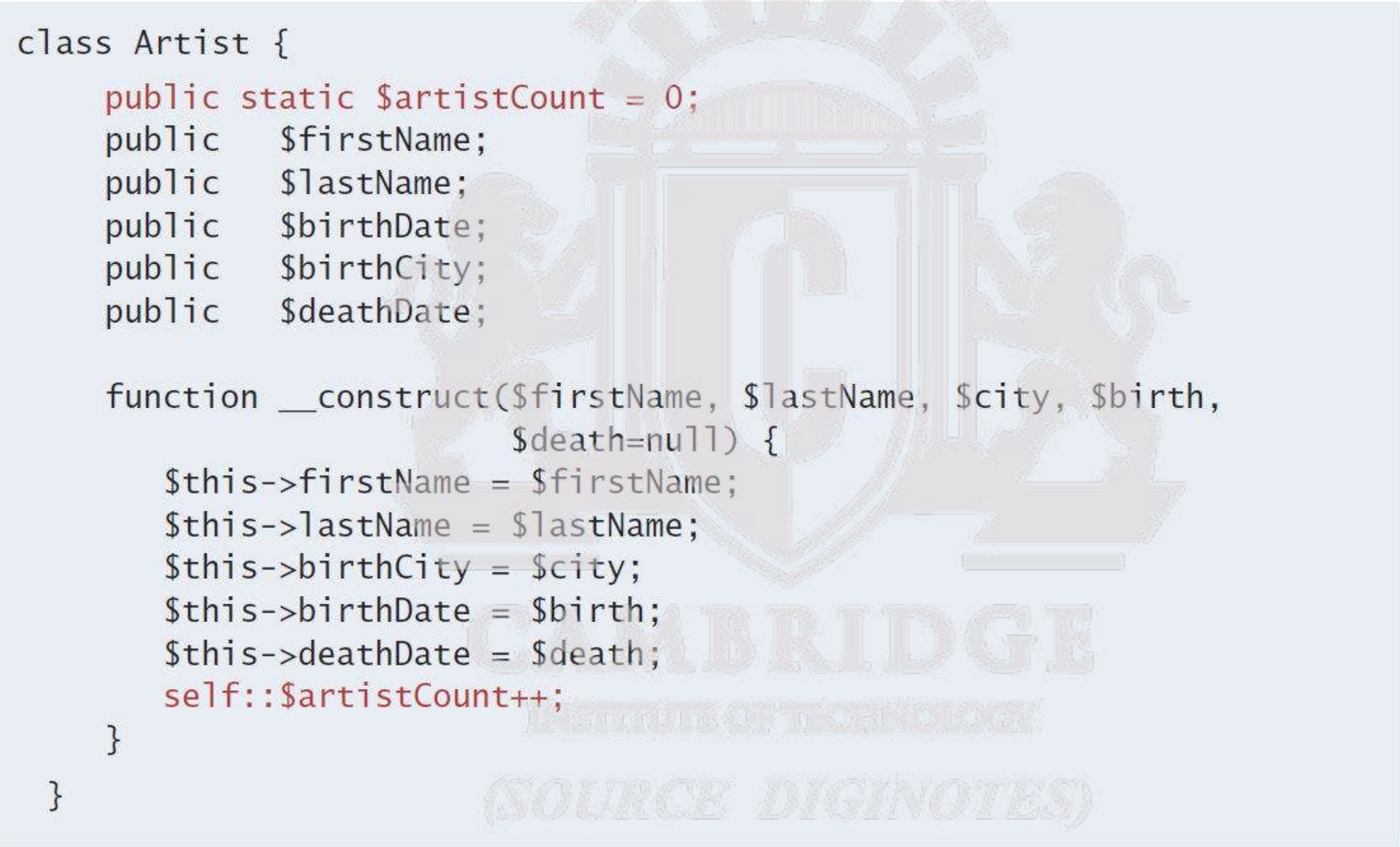
❖A **static** member is a property or method that all instances of a class share.

❖Unlike an instance property, where each object gets its own value for that property, there is only one value for a class's static property.

❖Static members use the self:: syntax and are not associated with one object

❖They can be accessed without any instance of an Artist object by using the class name, that is, via **Artist::$artistCount.**

Static Members

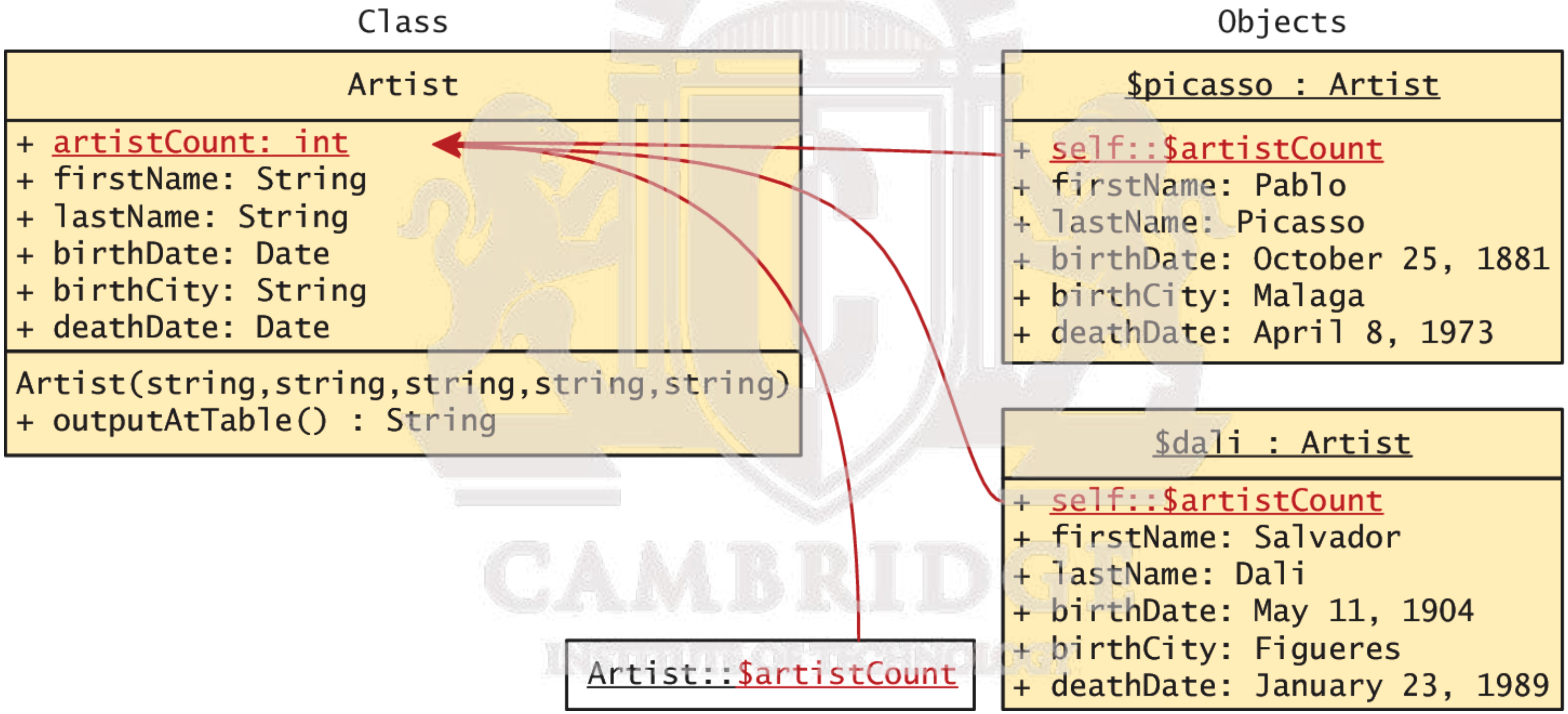


```
class Artist {
    public static $artistCount = 0;
    public   $firstName;
    public   $lastName;
    public   $birthDate;
    public   $birthCity;
    public   $deathDate;

    function __construct($firstName, $lastName, $city, $birth,
                         $death=null) {
       $this->firstName = $firstName;
       $this->lastName = $lastName;
       $this->birthCity = $city;
       $this->birthDate = $birth;
       $this->deathDate = $death;
       self::$artistCount++;
    }
  }
```

LISTING 10.5 Class definition modified with static members

Static Members

Uml again



Class

```
Artist
+ artistCount: int
+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date
Artist(string,string,string,string,string)
+ outputAtTable() : String
```

Objects

```
$picasso : Artist
+ self::$artistCount
+ firstName: Pablo
+ lastName: Picasso
+ birthDate: October 25, 1881
+ birthCity: Malaga
+ deathDate: April 8, 1973
```

```
$dali : Artist
+ self::$artistCount
+ firstName: Salvador
+ lastName: Dali
+ birthDate: May 11, 1904
+ birthCity: Figueres
+ deathDate: January 23, 1989
```

```
Artist::$artistCount
```

Class constants

Never changes

Constant values can be stored more efficiently as class constants so long as they are not calculated or updated

They are added to a class using the **const** keyword.

const EARLIEST_DATE = 'January 1, 1200';

Unlike all other variables, constants don't use the $ symbol when declaring or using them.

Accessed both inside and outside the class using

- **self::EARLIEST_DATE** in the class and
- **classReference::EARLIEST_DATE** outside.

Section 3 of 3

OBJECT ORIENTED DESIGN

Data Encapsulation

What is it?

- Perhaps the most important advantage to object-oriented design is the possibility of **encapsulation**, which generally refers to restricting access to an object's internal components.
- Another way of understanding encapsulation is: it is the hiding of an object's implementation details
- A properly encapsulated class will define an interface to the world in the form of its public methods, and leave its data, that is, its properties, hidden (that is, private).

Data Encapsulation

Getters and setters

If a properly encapsulated class makes its properties private, then how do you access them?

- **getters**
- **setters**

Data Encapsulation

Getters

A getter to return a variable's value is often very straightforward and should not modify the property.

```
public function getFirstName() {

        return $this->firstName;

}
```

Data Encapsulation

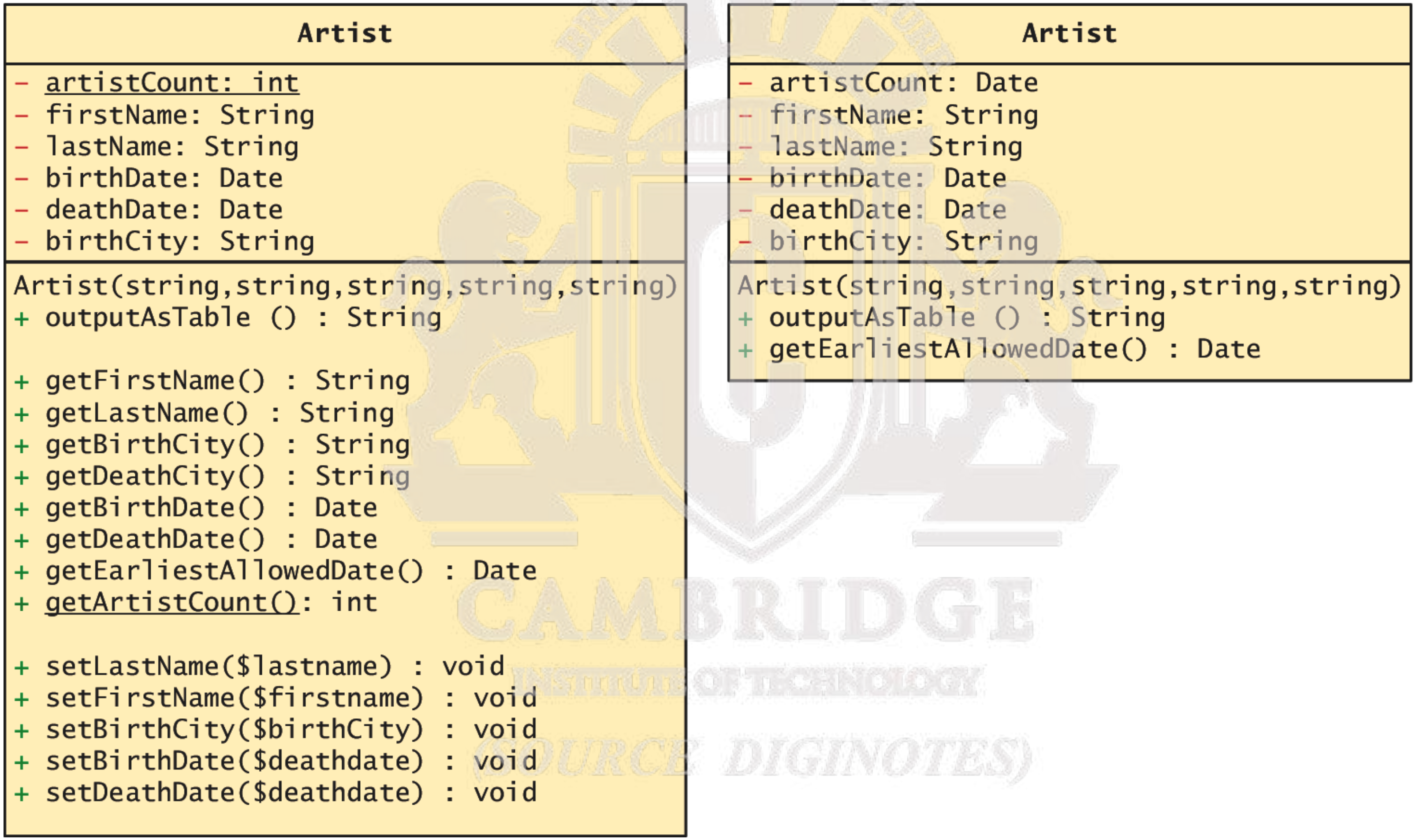
Setters

Setter methods modify properties, and allow extra logic to be added to prevent properties from being set to strange values.

```
public function setBirthDate($birthdate){
          // set variable only if passed a valid date string
          $date = date_create($birthdate);
          if ( ! $date ) {
                $this->birthDate = $this->getEarliestAllowedDate();
          }
          else {
          // if very early date then change it to
          // the earliest allowed date
                    if ( $date < $this->getEarliestAllowedDate() ) {
                      $date = $this->getEarliestAllowedDate();
                    }
                    $this->birthDate = $date;
          }

}
```

Data Encapsulation

UML

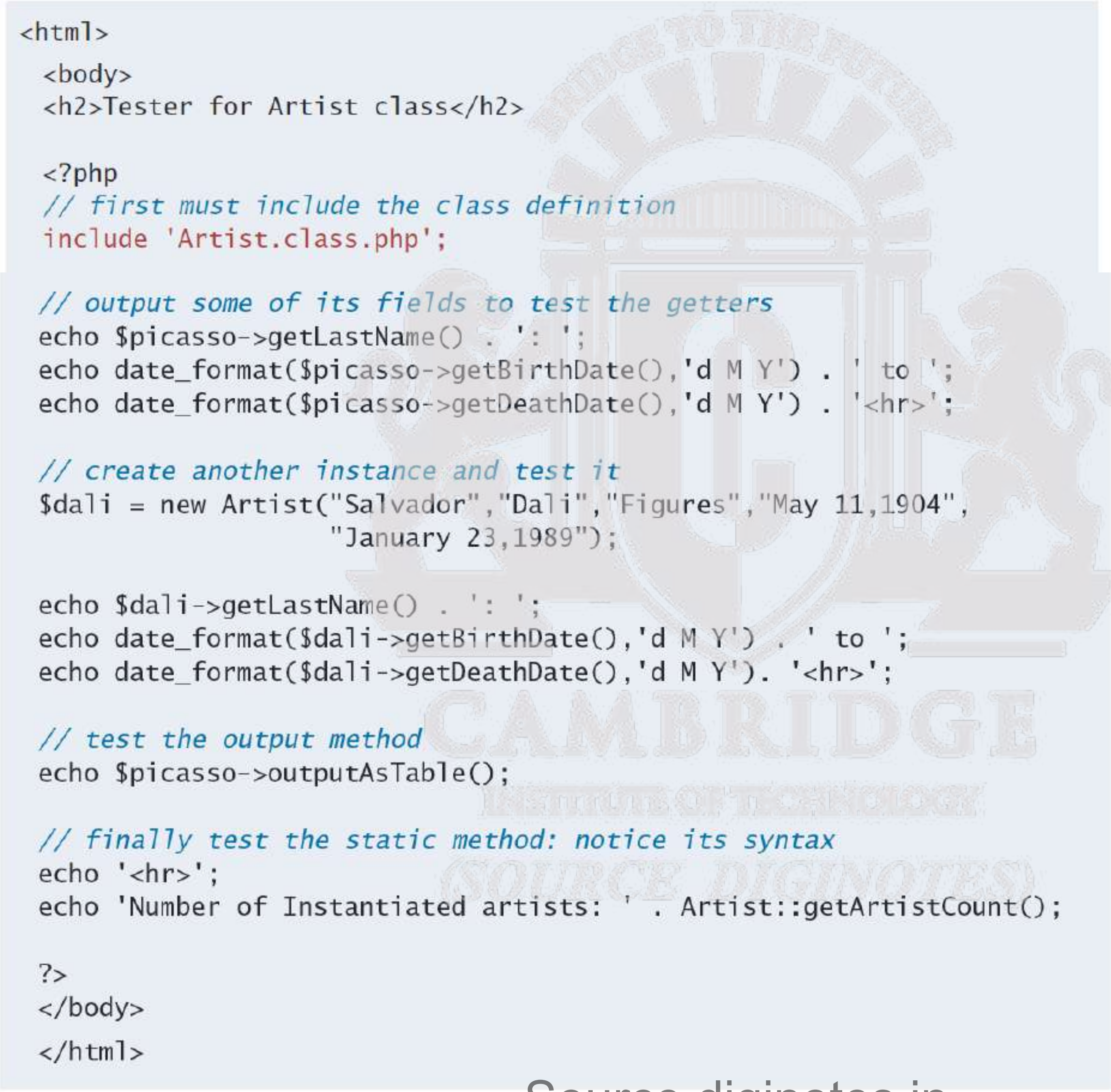


```
Artist
- artistCount: int
- firstName: String
- lastName: String
- birthDate: Date
- deathDate: Date
- birthCity: String
Artist(string,string,string,string,string)
+ outputAsTable () : String

+ getFirstName() : String
+ getLastName() : String
+ getBirthCity() : String
+ getDeathCity() : String
+ getBirthDate() : Date
+ getDeathDate() : Date
+ getEarliestAllowedDate() : Date
+ getArtistCount(): int

+ setLastName($lastname) : void
+ setFirstName($firstname) : void
+ setBirthCity($birthCity) : void
+ setBirthDate($deathdate) : void
+ setDeathDate($deathdate) : void
```

```
Artist
- artistCount: Date
- firstName: String
- lastName: String
- birthDate: Date
- deathDate: Date
- birthCity: String
Artist(string,string,string,string,string)
+ outputAsTable () : String
+ getEarliestAllowedDate() : Date
```

Data Encapsulation

Using an encapsulated class



```
<html>
 <body>
 <h2>Tester for Artist class</h2>

 <?php
 // first must include the class definition
 include 'Artist.class.php';

 // output some of its fields to test the getters
 echo $picasso->getLastName() . ': ';
 echo date_format($picasso->getBirthDate(),'d M Y') . ' to ';
 echo date_format($picasso->getDeathDate(),'d M Y') . '<hr>';

 // create another instance and test it
 $dali = new Artist("Salvador","Dali","Figures","May 11,1904",
                    "January 23,1989");

 echo $dali->getLastName() . ': ';
 echo date_format($dali->getBirthDate(),'d M Y') . ' to ';
 echo date_format($dali->getDeathDate(),'d M Y'). '<hr>';

 // test the output method
 echo $picasso->outputAsTable();

 // finally test the static method: notice its syntax
 echo '<hr>';
 echo 'Number of Instantiated artists: ' . Artist::getArtistCount();

 ?>
 </body>
 </html>
```

LISTING 10.7 Using the encapsulated class

Inheritance

Inheritance enables you to create new PHP classes that reuse, extend, and modify the behavior that is defined in another PHP class.

- PHP only allows you to inherit from one class at a time
- A class that is inheriting from another class is said to be a **subclass** or a **derived class**
- The class that is being inherited from is typically called a **superclass** or a **base class**

A PHP class is defined as a subclass by using the ***extends*** keyword.

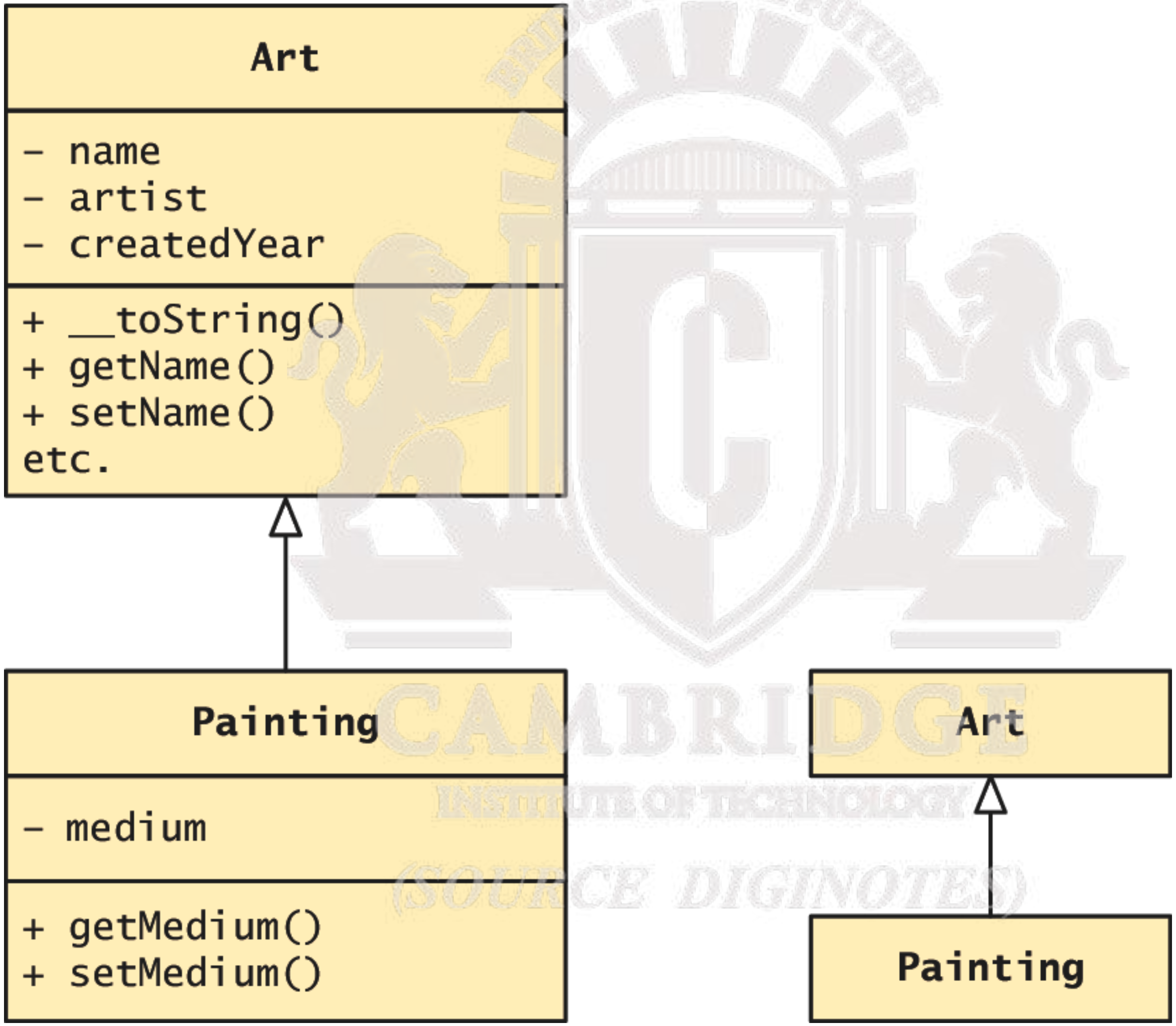
class Painting **extends** Art { . . . }

Example usage

```
$p = new Painting();

. . .

echo $p->getName(); // defined in base class

echo $p->getMedium(); // defined in subclass
```

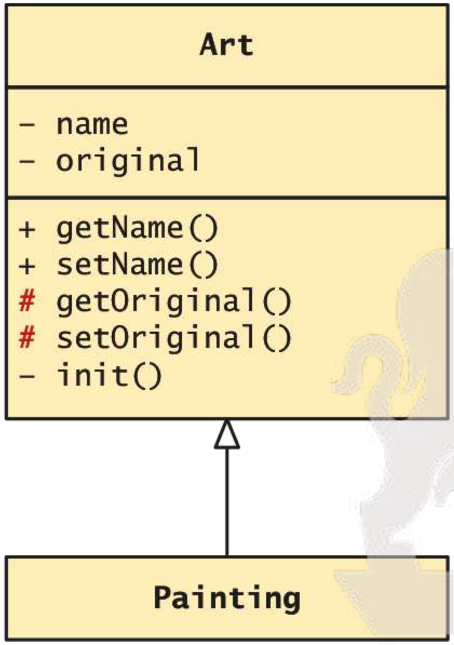
Inheritance

There’s UML for that too



Protected access modifier

Remember Protected?



Art

- name
- original

+ getName()
+ setName()
# getOriginal()
# setOriginal()
- init()

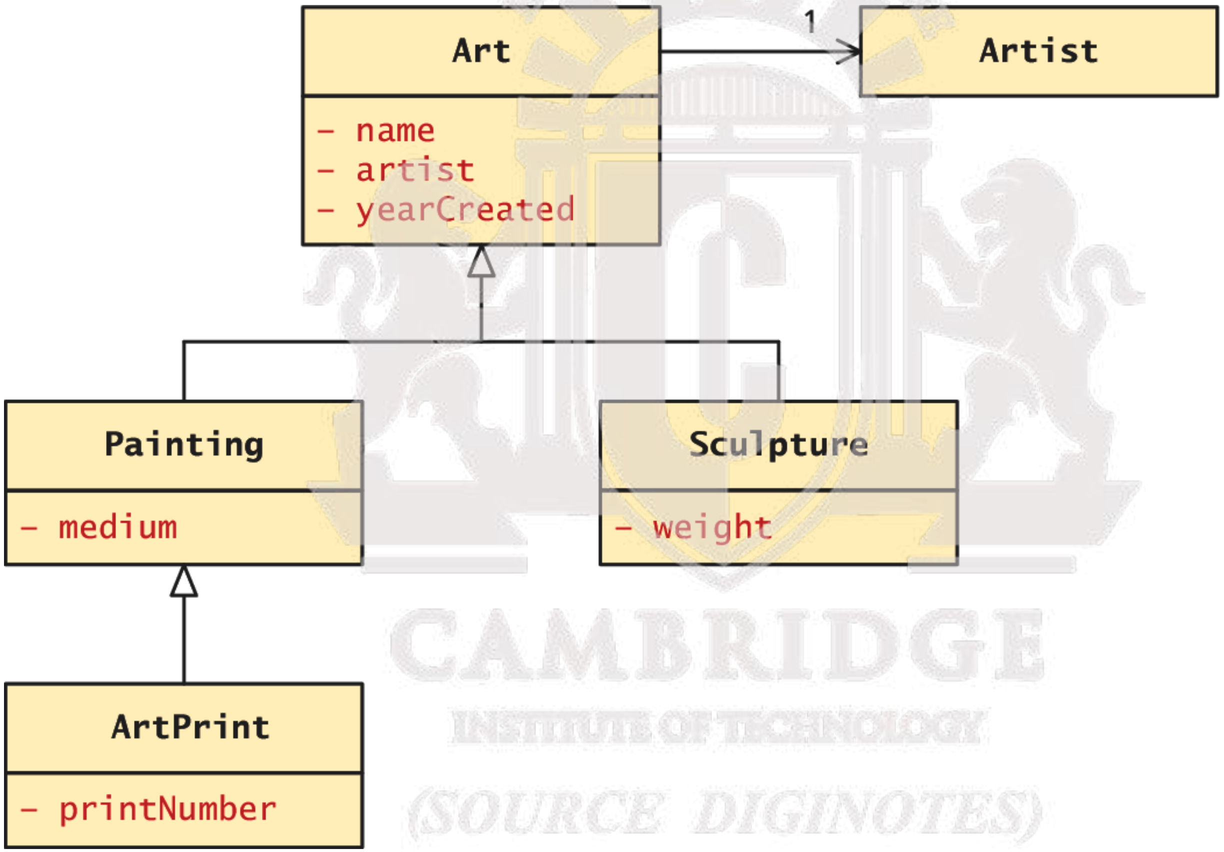
Painting

```
  class Painting extends Art {
     …
     private function foo() {
        …
        // these are allowed
✓       $w = parent::getName();
✓       $x = parent::getOriginal();

        // this is not allowed
✗       $y = parent::init();
     }
  }
```

```
  // in some page or other class
  $p = new Painting();
  $a = new Art();

  // neither of these references are allowed
✗ $w = $p->getOriginal();
✗ $y = $a->getOriginal();
```

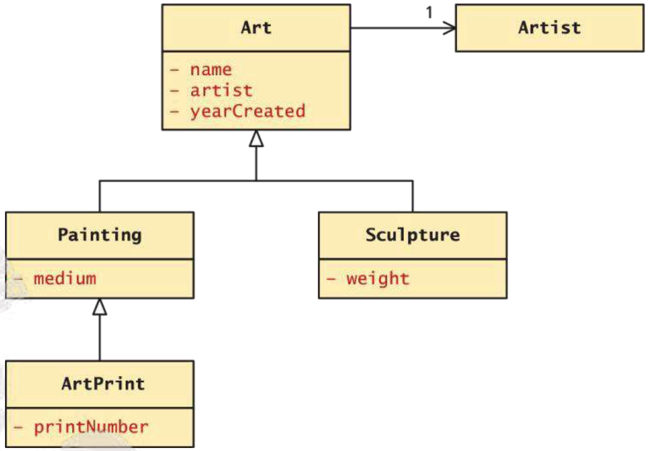
A More Complex Example

Using inheritance



Extended example

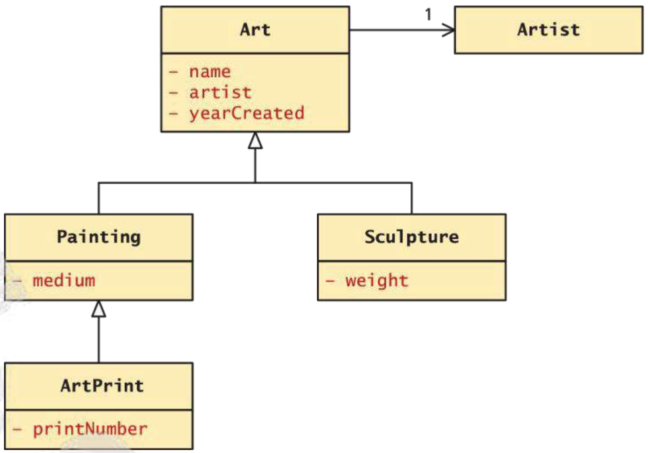
All art has certain properties



```
/* The abstract class that
 contains functionality required by
 all types of Art */

abstract class Art {

        private $name;

        private $artist;

        private $yearCreated;

        //... constructor, getters, setters
```

Extended example

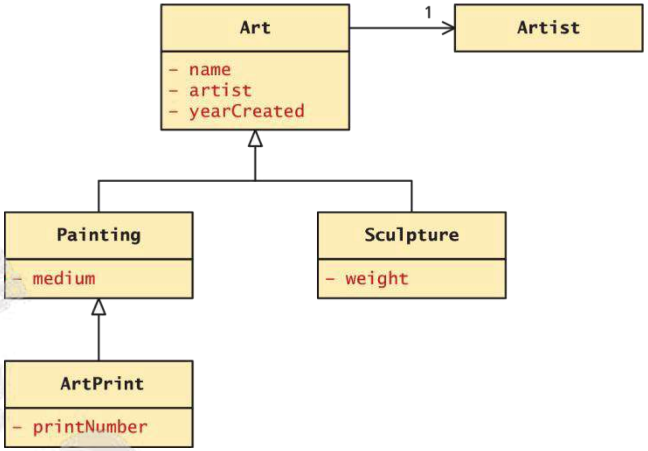
Painting require a “medium”



```
class Painting extends Art {

        private $medium;

        //...constructor, getters, setters

        public function __toString() {

        return parent::__toString() . ", Medium: " .
                                        $this->getMedium();

    }

}
```

Extended example

Sculptures have weight



```
class Sculpture extends Art {
        private $weight;
        //...constructor, getters, setters
        public function __toString() {
                return parent::__toString() . ", Weight: " .
                                $this->getWeight() ."kg";
        }
}
```

Extended example

Using the classes

```
…

$picasso = new Artist("Pablo","Picasso","Malaga","May 11,904”,"Apr 8, 1973");

$guernica = new Painting("1937",$picasso,"Guernica”, "Oil on canvas");

$woman = new Sculpture("1909",$picasso,"Head of a Woman", 30.5);

?>

<h2>Paintings</h2>

<p><em>Use the __toString() methods </em></p>

<p><?php echo $guernica; ?></p>

<h2>Sculptures</h2>

<p> <?php echo $woman; ?></p>
```

Polymorphism

No thank you, I’ll have water

➢**Polymorphism** is the notion that an object can in fact be multiple things at the same time.

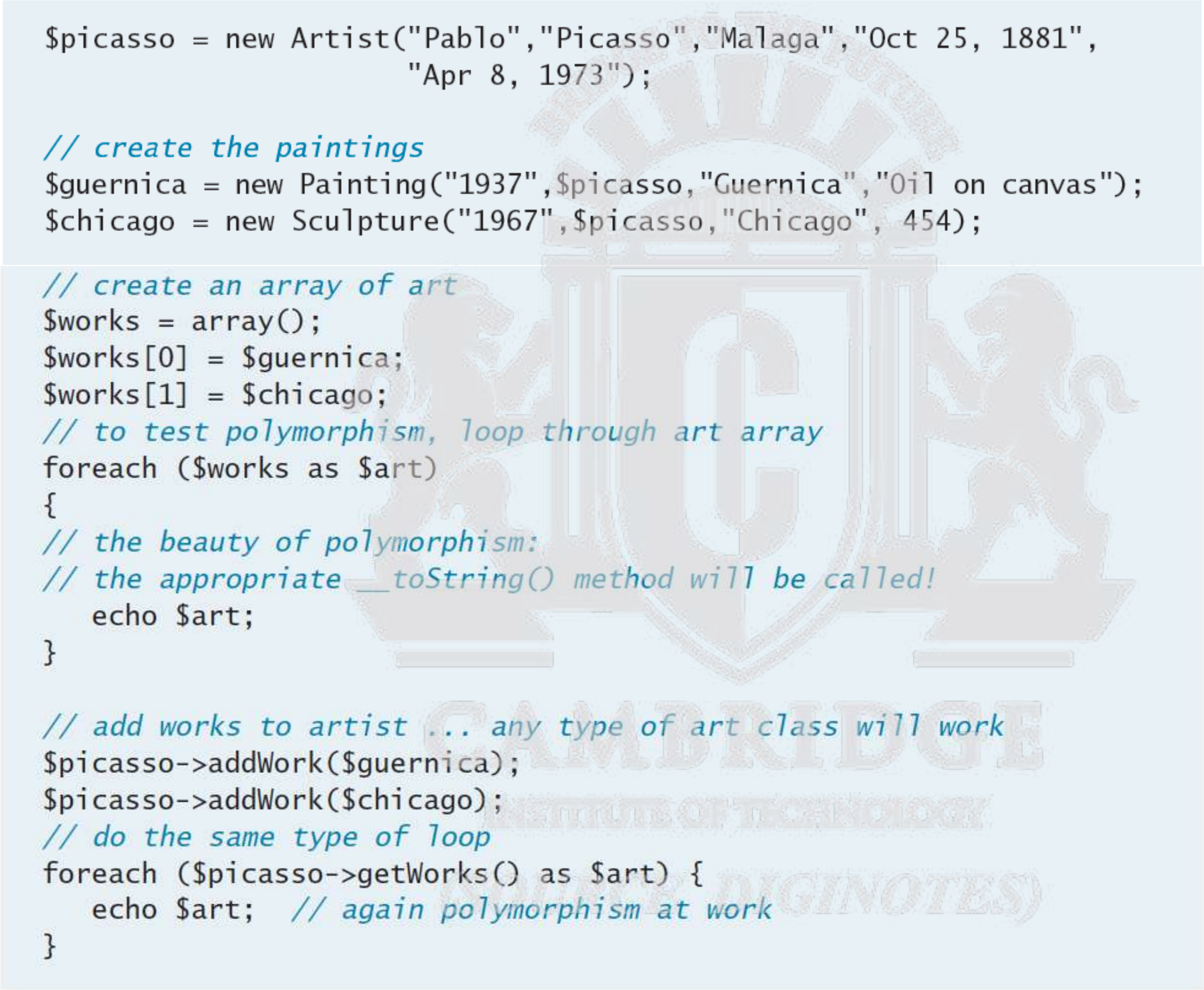
➢Consider an instance of a Painting object named $guernica created as follows:

```
$guernica = new Painting("1937",$picasso,"Guernica","Oil on canvas");
```

➢The variable $guernica is both a *Painting* object and an *Art* object due to its inheritance.

➢The advantage of polymorphism is that we can manage a list of Art objects, and call the same overridden method on each.

Polymorphism



```
$picasso = new Artist("Pablo","Picasso","Malaga","Oct 25, 1881",
                      "Apr 8, 1973");

// create the paintings
$guernica = new Painting("1937",$picasso,"Guernica","Oil on canvas");
$chicago = new Sculpture("1967",$picasso,"Chicago", 454);

// create an array of art
$works = array();
$works[0] = $guernica;
$works[1] = $chicago;
// to test polymorphism, loop through art array
foreach ($works as $art)
{
// the beauty of polymorphism:
// the appropriate __toString() method will be called!
   echo $art;
}

// add works to artist ... any type of art class will work
$picasso->addWork($guernica);
$picasso->addWork($chicago);
// do the same type of loop
foreach ($picasso->getWorks() as $art) {
   echo $art;  // again polymorphism at work
}
```

LISTING 10.10 Using polymorphism

Interfaces

Defining the interface

❖An object **interface** is a way of defining a formal list of methods that a class **must** implement without specifying their implementation.

❖Interfaces are defined using the interface keyword, and look similar to standard PHP classes, except an interface contains no properties and its methods do not have method bodies defined.

```
interface Viewable {
          public function getSize();
          public function getPNG();
}
```

Interfaces

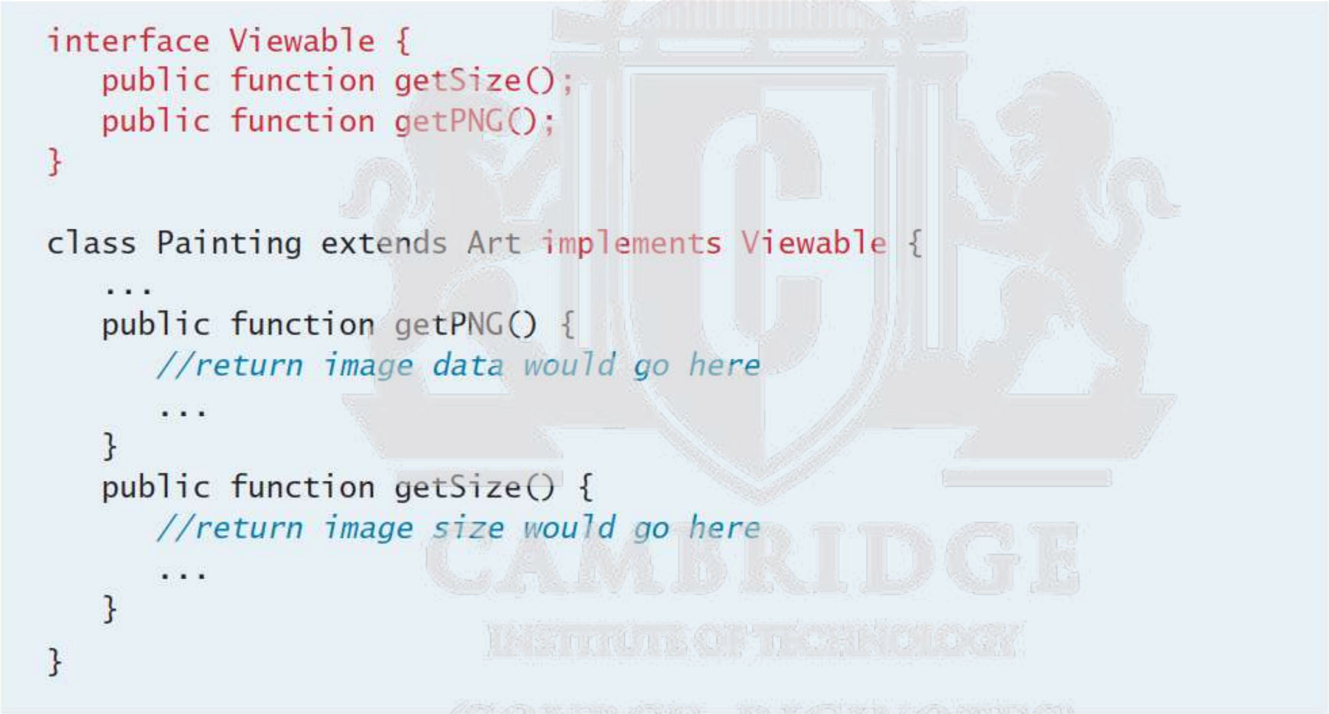
Implementing the Interface

- In PHP, a class can be said to *implement* an interface, using the implements keyword:

class *Painting* extends Art **implements** Viewable { ... }

- This means then that the class *Painting* must provide implementations for the getSize() and getPNG() methods.

Interface Example

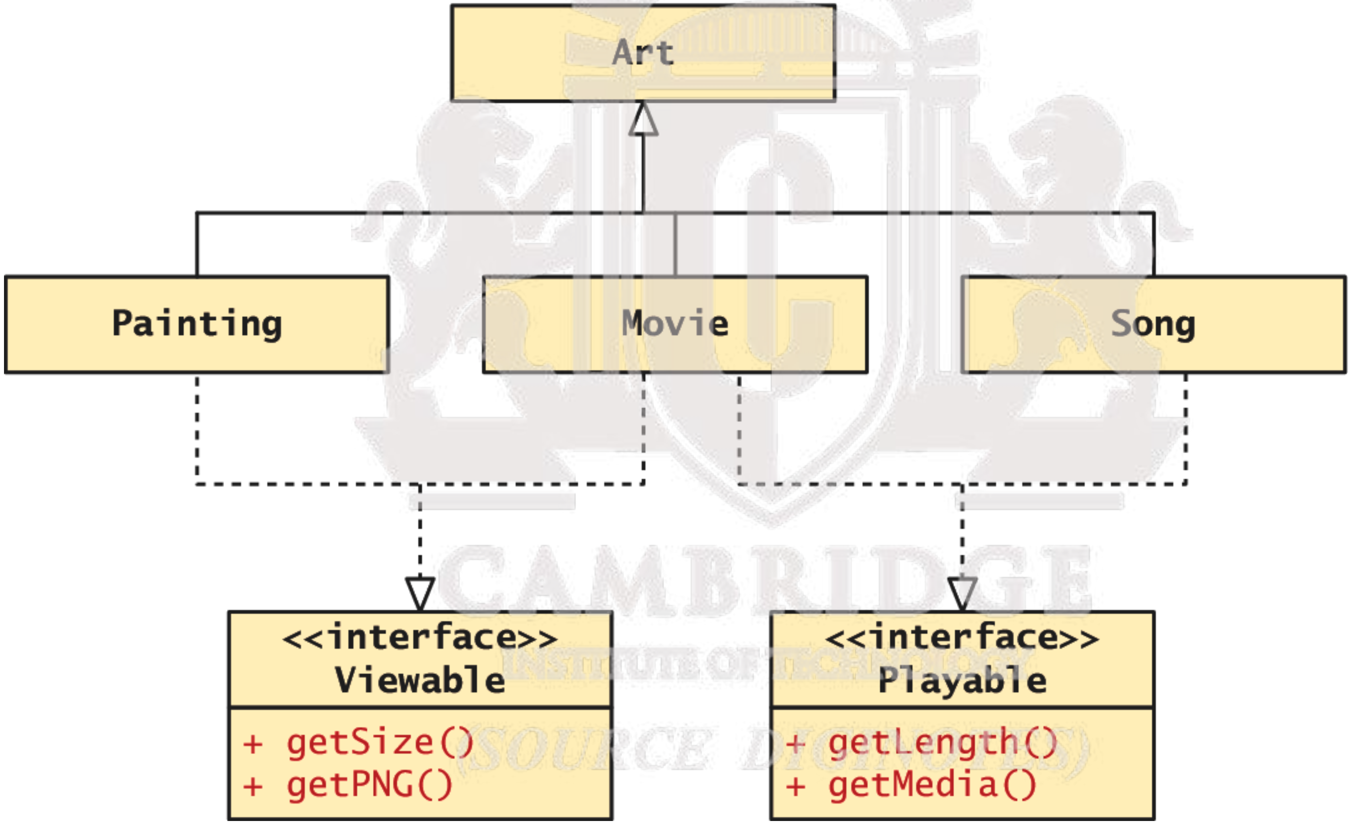


```
interface Viewable {
   public function getSize();
   public function getPNG();
}

class Painting extends Art implements Viewable {
   ...
   public function getPNG() {
      //return image data would go here
      ...
   }
   public function getSize() {
      //return image size would go here
      ...
   }
}
```

LISTING 10.11 Painting class implementing an interface

Interfaces

An Extended example



Error Handling and Validation

Chapter 12

Section 1 of 6

WHAT ARE ERRORS AND EXCEPTIONS?

Types of Errors

- Expected errors

  Things that you expect to go wrong. Bad user input, database connection, etc…

- Warnings

  problems that generate a PHP warning message but will not halt the execution of the page

- Fatal errors

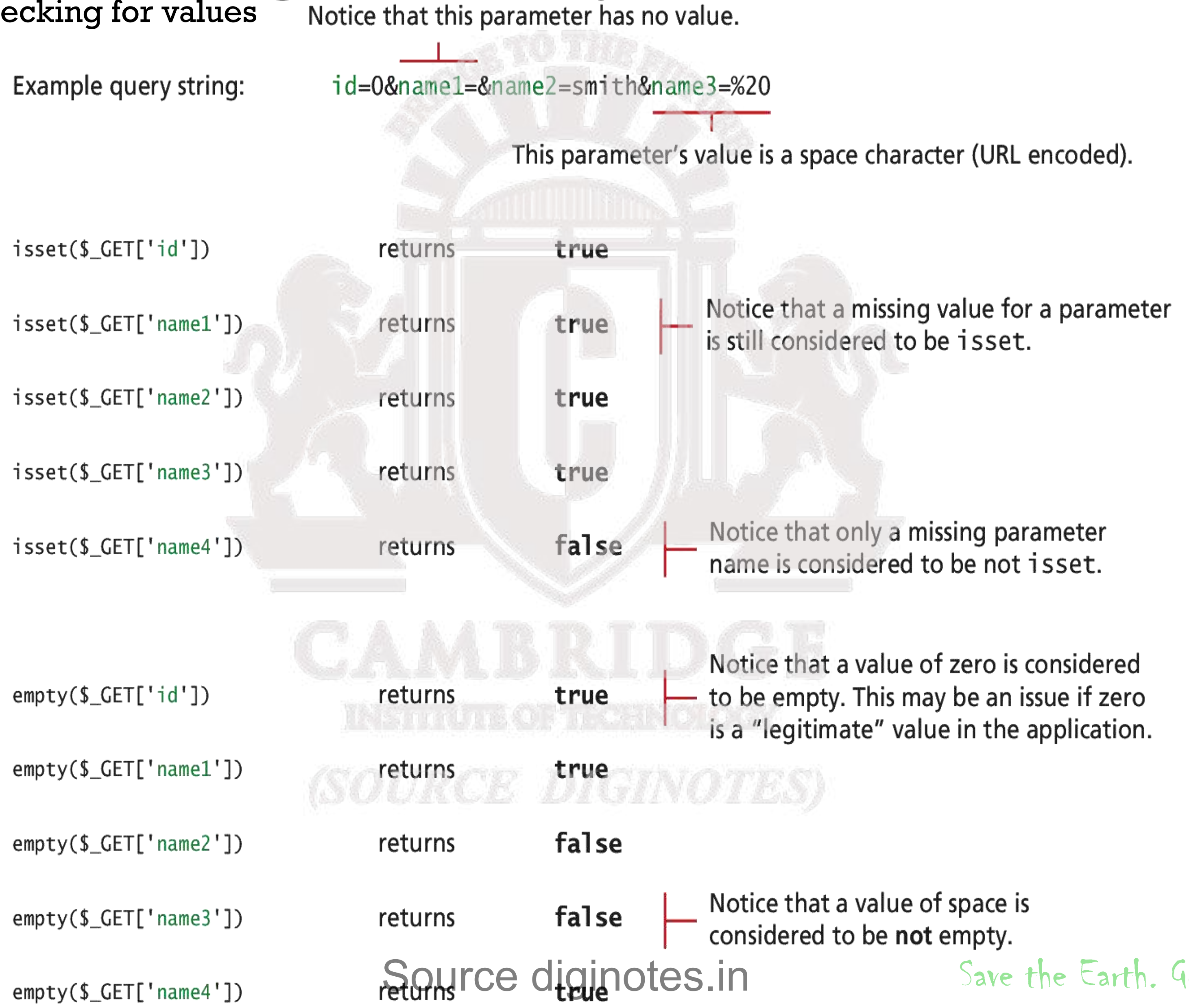
  are serious in that the execution of the page will terminate unless handled in some way

Isset() : returns true if a variable is not null.

Empty() : returns true if a variable is null, false, zero or an empty string.

Checking user input

Checking for values



Notice that this parameter has no value.

Example query string: `id=0&name1=&name2=smith&name3=%20`

This parameter's value is a space character (URL encoded).

```
isset($_GET['id'])        returns    true
isset($_GET['name1'])     returns    true
isset($_GET['name2'])     returns    true
isset($_GET['name3'])     returns    true
isset($_GET['name4'])     returns    false
```

Notice that a missing value for a parameter is still considered to be `isset`.

Notice that only a missing parameter name is considered to be not `isset`.

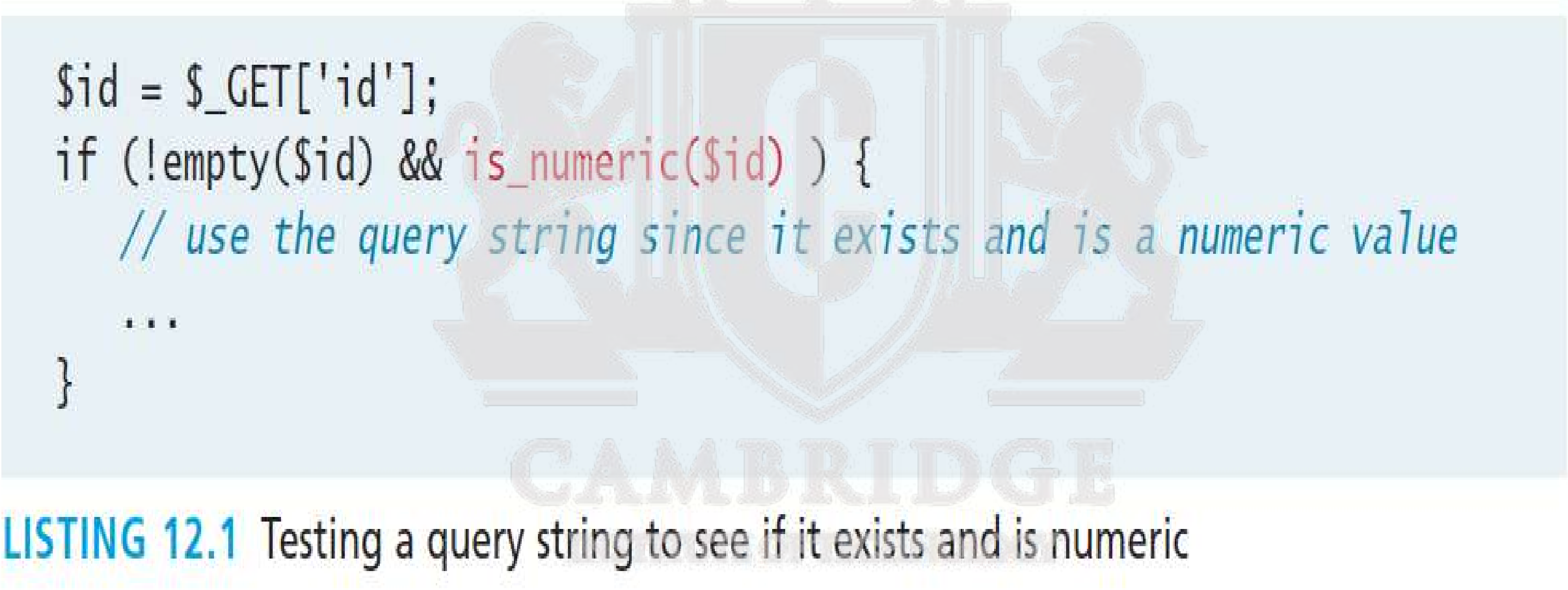
```
empty($_GET['id'])        returns    true
empty($_GET['name1'])     returns    true
empty($_GET['name2'])     returns    false
empty($_GET['name3'])     returns    false
empty($_GET['name4'])     returns    true
```

Notice that a value of zero is considered to be empty. This may be an issue if zero is a "legitimate" value in the application.

Notice that a value of space is considered to be **not** empty.

Checking user input

Checking for a number



```
$id = $_GET['id'];
if (!empty($id) && is_numeric($id) ) {
   // use the query string since it exists and is a numeric value
   ...
}
```

LISTING 12.1 Testing a query string to see if it exists and is numeric

Exceptions vs Errors

Not the same thing

- An **error** is some type of problem that generates a nonfatal warning message or that generates an error message that terminates the program’s execution.
- An **exception** refers to objects that are of type Exception and which are used in conjunction with the object-oriented try . . . catch language construct for dealing with runtime errors.

Section 2 of 6

PHP ERROR REPORTING

PHP error reporting

Lots of control

PHP has a flexible and customizable system for reporting warnings and errors that can be set programmatically at runtime or declaratively at design-time within the **php.ini** file. There are three main error reporting flags:

- error_reporting
- display_errors
- log_errors

The error_reporting setting

What is an error?

The **error_reporting** setting specifies which type of errors are to be reported.

It can be set programmatically inside **any** PHP file:

```
error_reporting(E_ALL);
```

It can also be set within the **php.ini** file:

```
error_reporting = E_ALL
```

The error_reporting setting

Some error reporting constants

Constant Name	Value	Description
E_ALL	8191	Report all errors and warnings
E_ERROR	1	Report all fatal runtime errors
E_WARNING	2	Report all nonfatal runtime errors (that is, warnings)
	0	No reporting

The display_errors setting

To show or not to show

The **display_error** setting specifies whether error messages should or should not be displayed in the browser.

It can be set programmatically via the ini_set() function:

```
ini_set('display_errors','0');
```

It can also be set within the **php.ini** file:

```
display_errors = Off
```

The log_error setting

To record or not to record

The **log_error** setting specifies whether error messages should or should not be sent to the server error log.

It can be set programmatically via the ini_set() function:

```
ini_set('log_errors','1');
```

It can also be set within the **php.ini** file:

```
log_errors = On
```

The log_error setting

Where to store.

The location to store logs in can be set programatically:

```
ini_set('error_log', '/restricted/my-errors.log');
```

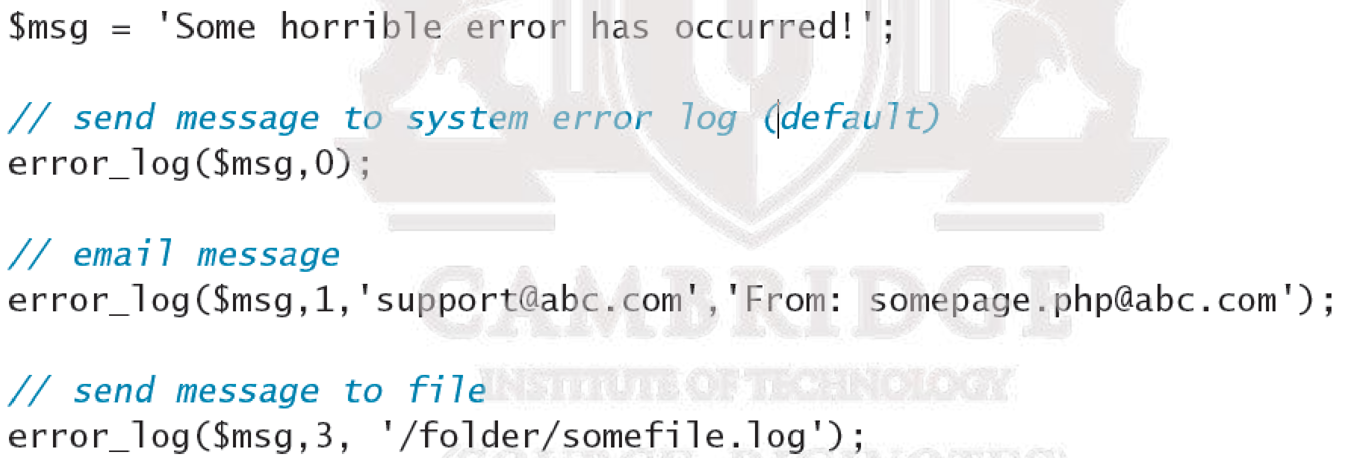
It can also be set within the **php.ini** file:

```
error_log = /restricted/my-errors.log
```

The log_error setting

Error_log()

You can also programmatically send messages to the error log at any time via the error_log() function



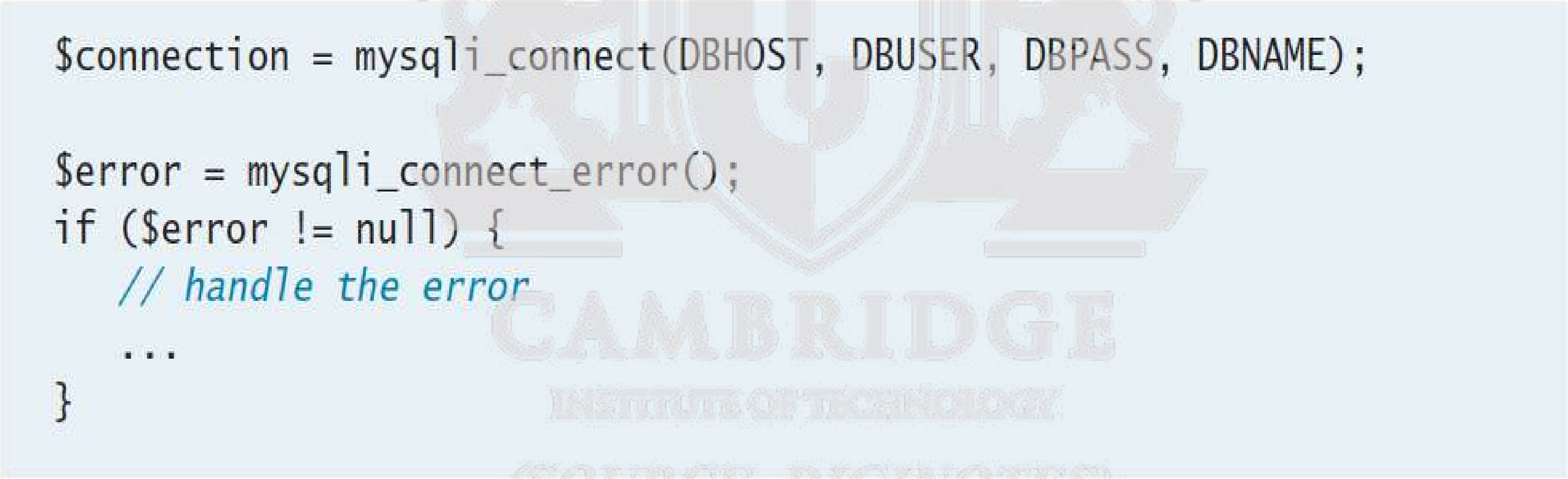
```
$msg = 'Some horrible error has occurred!';

// send message to system error log (default)
error_log($msg,0);

// email message
error_log($msg,1,'support@abc.com','From: somepage.php@abc.com');

// send message to file
error_log($msg,3, '/folder/somefile.log');
```

Section 3 of 6

PHP ERROR AND EXCEPTION HANDLING

Procedural Error Handling

Recall connecting to a database, that there may be an error…



```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);

$error = mysqli_connect_error();
if ($error != null) {
   // handle the error
   ...
}
```

LISTING 12.2 Procedural approach to error handling

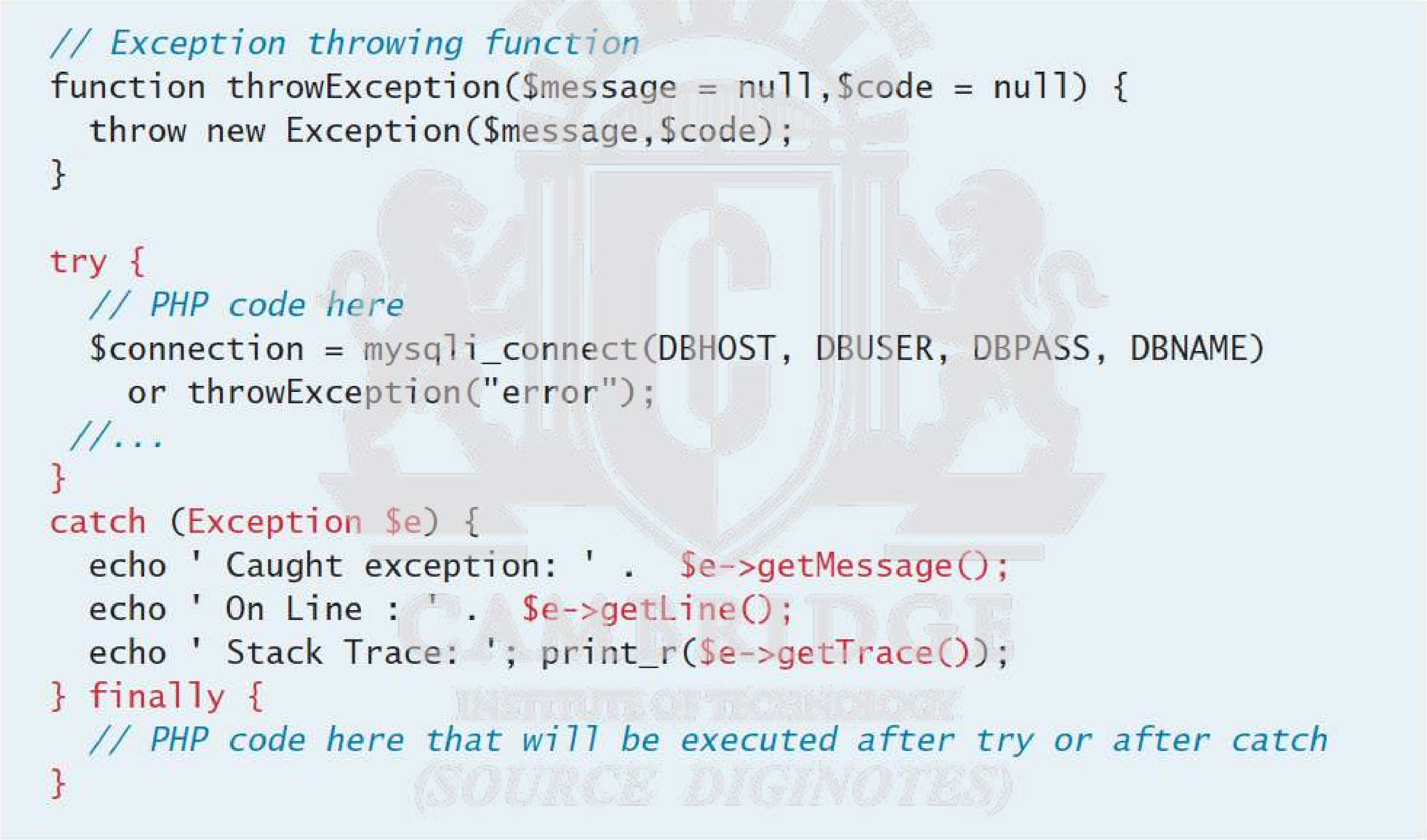
OO Exception Handling

Try, catch, finally

- When a runtime error occurs, PHP *throws* an *exception*.
- This exception can be *caught* and handled either by the function, class, or page that generated the exception or by the code that called the function or class.
- If an exception is not caught, then eventually the PHP environment will handle it by terminating execution with an “Uncaught Exception” message.

OO Exception Handling

Try, catch, finally



```
// Exception throwing function
function throwException($message = null,$code = null) {
  throw new Exception($message,$code);
}

try {
  // PHP code here
  $connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME)
    or throwException("error");
 //...
}
catch (Exception $e) {
  echo ' Caught exception: ' .  $e->getMessage();
  echo ' On Line : ' . $e->getLine();
  echo ' Stack Trace: '; print_r($e->getTrace());
} finally {
  // PHP code here that will be executed after try or after catch
}
```

LISTING 12.3 Example of try . . . catch block

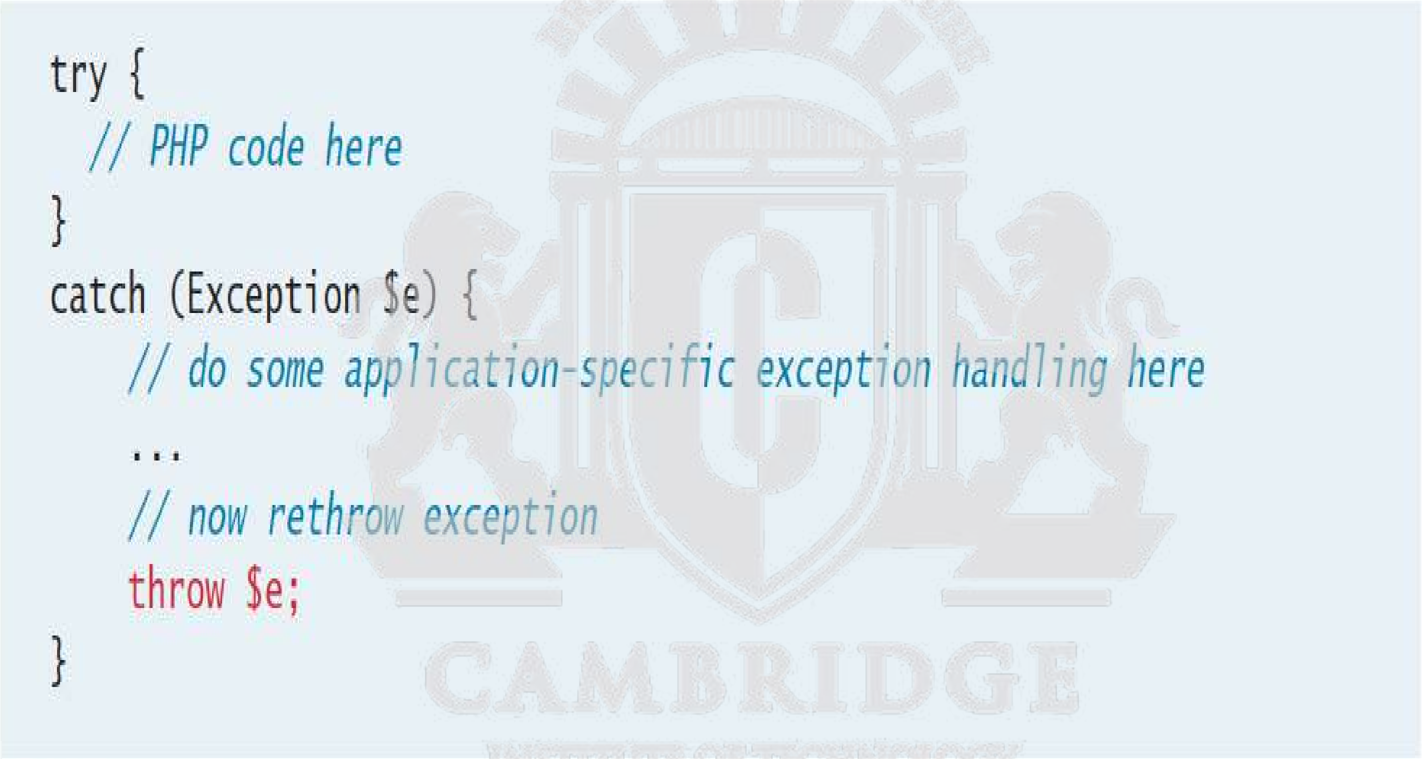
OO Exception Handling

Finally

- ❖The finally block is optional. Any code within it will always be executed *after* the code in the try or in the catch blocks, even if that code contains a return statement.
- ❖The finally block is only available in PHP 5.5 and later

Throw your own exception

Object oriented way of dealing with the unexpected



```
try {
  // PHP code here
}
catch (Exception $e) {
    // do some application-specific exception handling here
    ...
    // now rethrow exception
    throw $e;
}
```

LISTING 12.5 Rethrowing an exception

Custom Handlers

Error and Exception Handlers

- ❖It is possible to define your own handler for uncaught errors and exceptions, the mechanism for doing so varies depending upon whether you are using the procedural or object oriented mechanism for responding to errors.

- ❖If using the procedural approach(i.e, not using try…catch) you can define a custom error handling function and then register it with

  set_error_handler()

- ❖ If you are using the object oriented exception approach with try… catch blocks, you can define a custom exception handling function and then register it with

  set_exception_handler()

Custom Handlers

Error and Exception Handlers

✓What should a custom error or exception handler do?

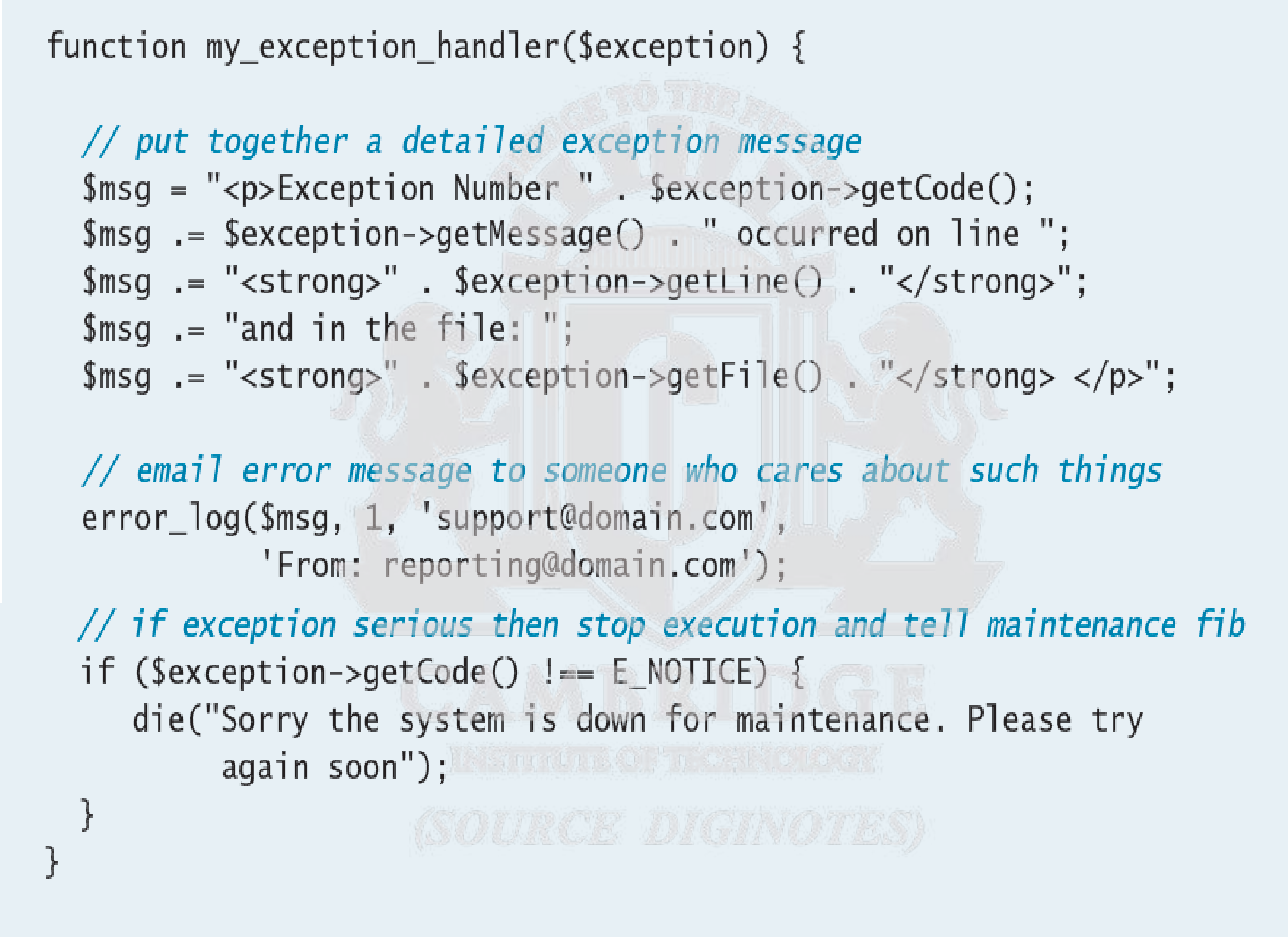
✓It should provide the *developer* with detailed information about the state of the application when the exception occurred, information about the exception, and when it happened.

✓It should hide any of those details from the regular end user, and instead provide the user with a generic message such as “Sorry but there was a problem”

✓Once a handler function is defined, it must be registered, using the following code:

```
set_exception_handler('my_exception_handler');
```

Custom Handlers



```
function my_exception_handler($exception) {

   // put together a detailed exception message
   $msg = "<p>Exception Number " . $exception->getCode();
   $msg .= $exception->getMessage() . " occurred on line ";
   $msg .= "<strong>" . $exception->getLine() . "</strong>";
   $msg .= "and in the file: ";
   $msg .= "<strong>" . $exception->getFile() . "</strong> </p>";

   // email error message to someone who cares about such things
   error_log($msg, 1, 'support@domain.com',
             'From: reporting@domain.com');

   // if exception serious then stop execution and tell maintenance fib
   if ($exception->getCode() !== E_NOTICE) {
      die("Sorry the system is down for maintenance. Please try
           again soon");
   }
}
```

LISTING 12.6 Custom exception handler

END OF MODULE 4

WEB TECHNOLOGY AND ITS APPLICATIONS

17CS71

Mr. GANESH D R
ASSISTANT PROFESSOR,
DEPT OF CSE, CITECH

WEB TECHNOLOGY AND ITS APPLICATIONS
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)
SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	**10 Hours**
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	**10 Hours**
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	**10 Hours**
Module – 4	
PHP Arrays and Superglobals, Arrays, $_GET and $_POST Superglobal Arrays, $_SERVER Array, $_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	**10 Hours**
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	**10 Hours**

MODULE 5 - SYLLABUS

- Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.

MANAGING STATE

Chapter 13

Section 1 of 8

THE PROBLEM OF STATE IN WEB APPLICATIONS

State in Web Applications

Not like a desktop application

❖Until now we have seen how to process user inputs, output information and read & write from other storage media.

❖But here we will be examining a development problem that is unique to the world of web development.

❖HOW CAN ONE REQUEST SHARE INFORMATION WITH ANOTHER REQUEST?

❖Single user desktop applications do not have this challenge at all because the program information for the user is stored in memory(or in external storage) and thus can be easily accesses through out the applications.

❖Remember Web applications differ from desktop applications

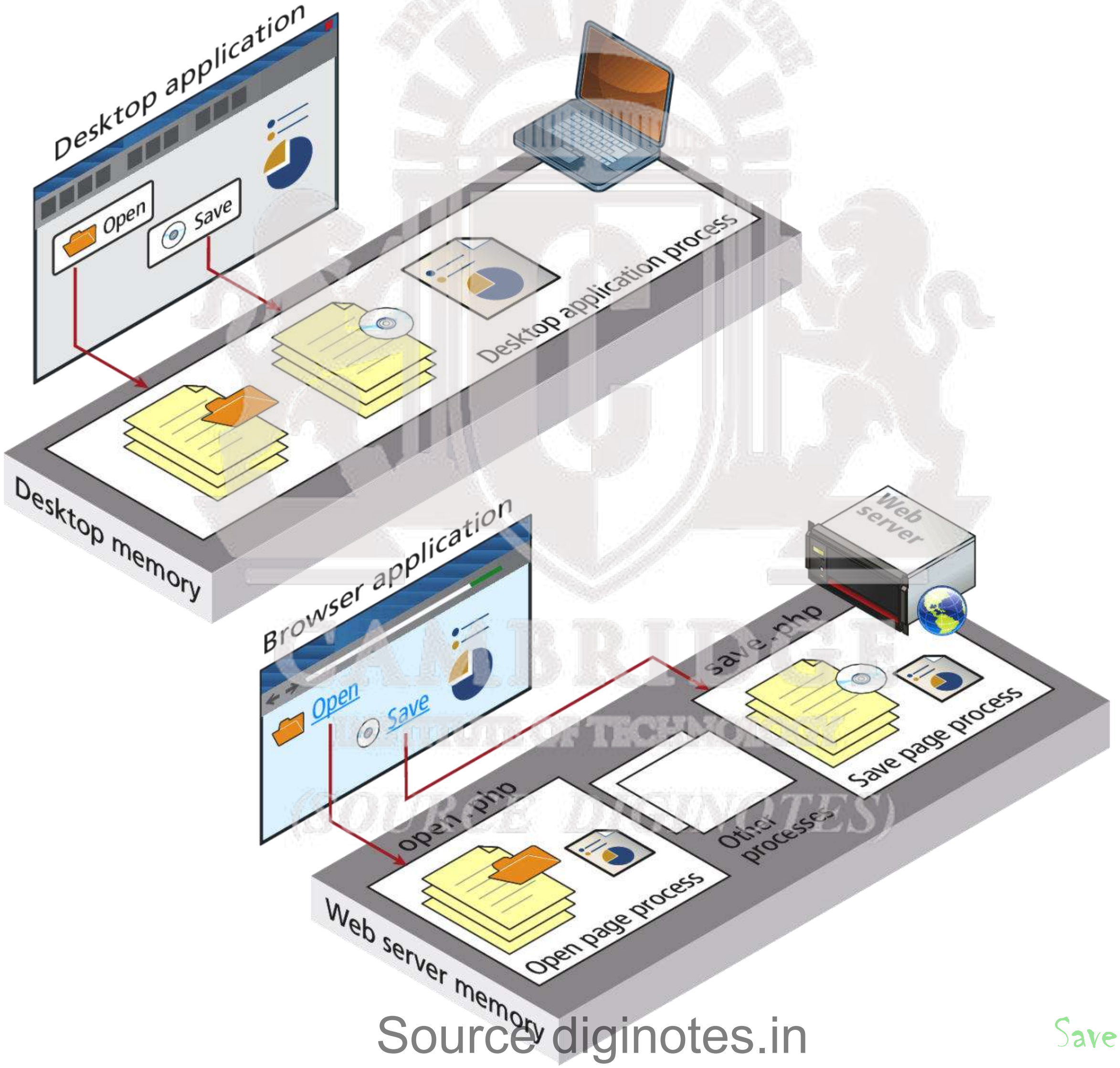
State in Web Applications

Not like a desktop application

❖Unlike the unified single process that is the typical desktop application, a web application consists of a series of disconnected HTTP requests to a web server where each request for a server page is essentially a request to run a separate program as in below fig

State in Web Applications

Not like a desktop application



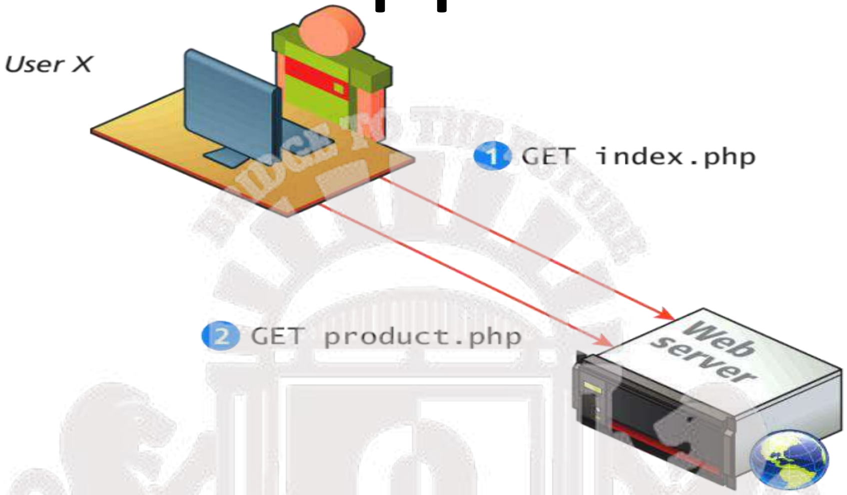
Source diginotes.in

- The web server sees only requests.

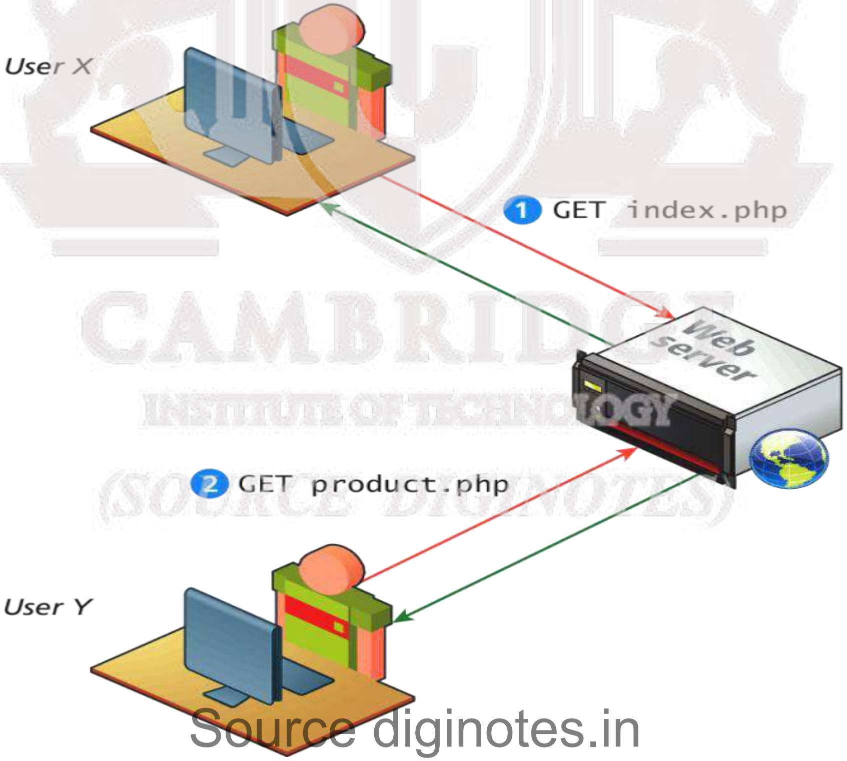
- The HTTP protocol does not, without program intervention, distinguish two requests by one source from two requests from two different sources, as in below figure

State in Web Applications

What's the issue?



... is for the server not really any different than ...



❖There are many occasions when we want the web server to connect requests together.

❖Consider the scenario of a web shopping cart, as in below fig.

❖In such a case, the user(website owner) most certainly wants the server to recognize that the request to add an item to the cart and the subsequent request to check out and pay for the item in the cart are connected to the same individual

State in Web Applications

What's the desired outcome

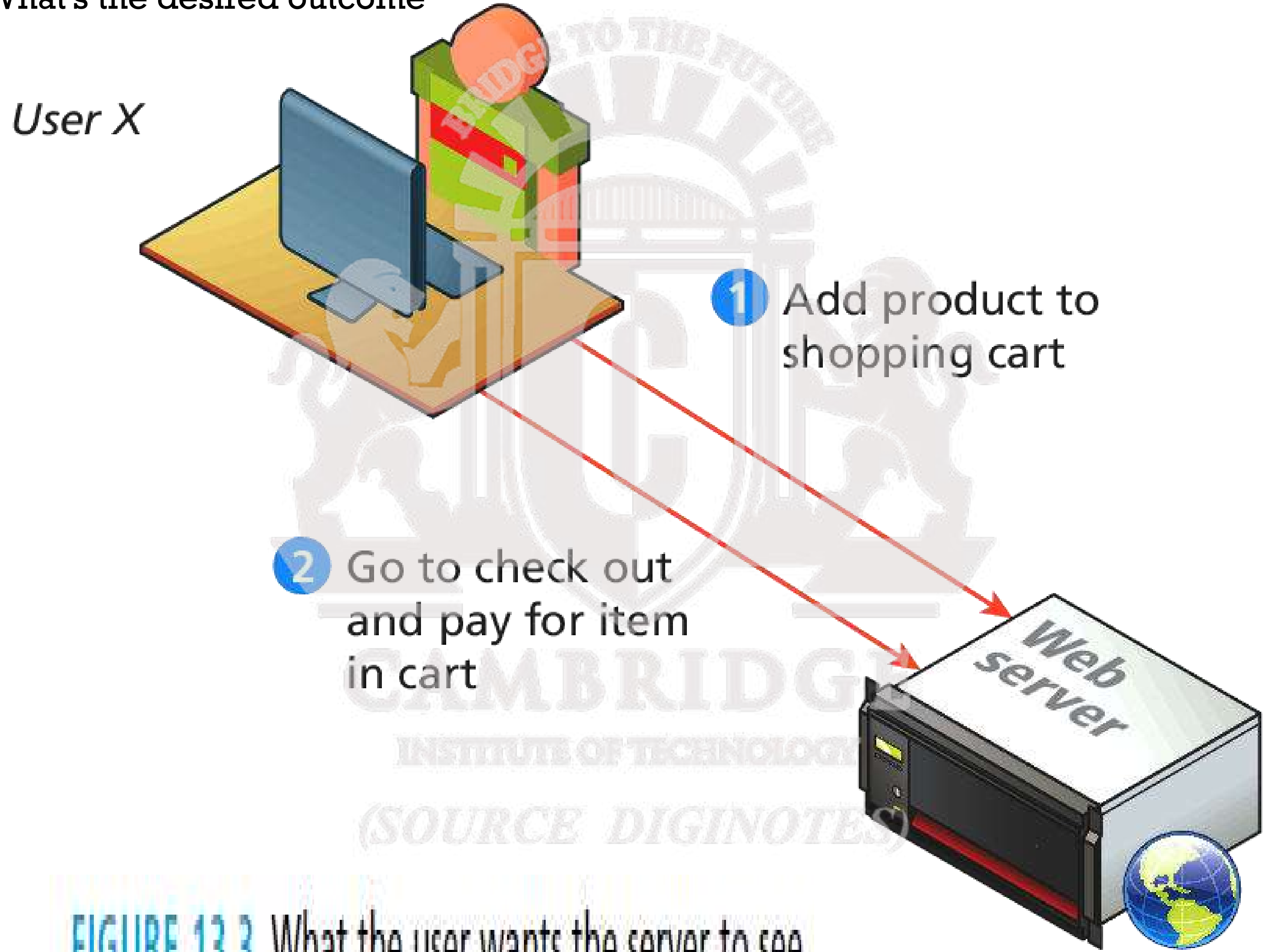

User X
1 Add product to shopping cart
2 Go to check out and pay for item in cart
Web server

FIGURE 13.3 What the user wants the server to see

State in Web Applications

How do we reach our desired outcome?

❑The rest of the chapter we will explain how web developers and web development environments work together through the constraints of HTTP to solve this particular problem.

❑How does one web page pass information to another page?

❑What mechanisms are available within HTTP to pass information to the server in our requests?

In HTTP, we can pass information using:

- Query strings
- Cookies

Section 2 of 8

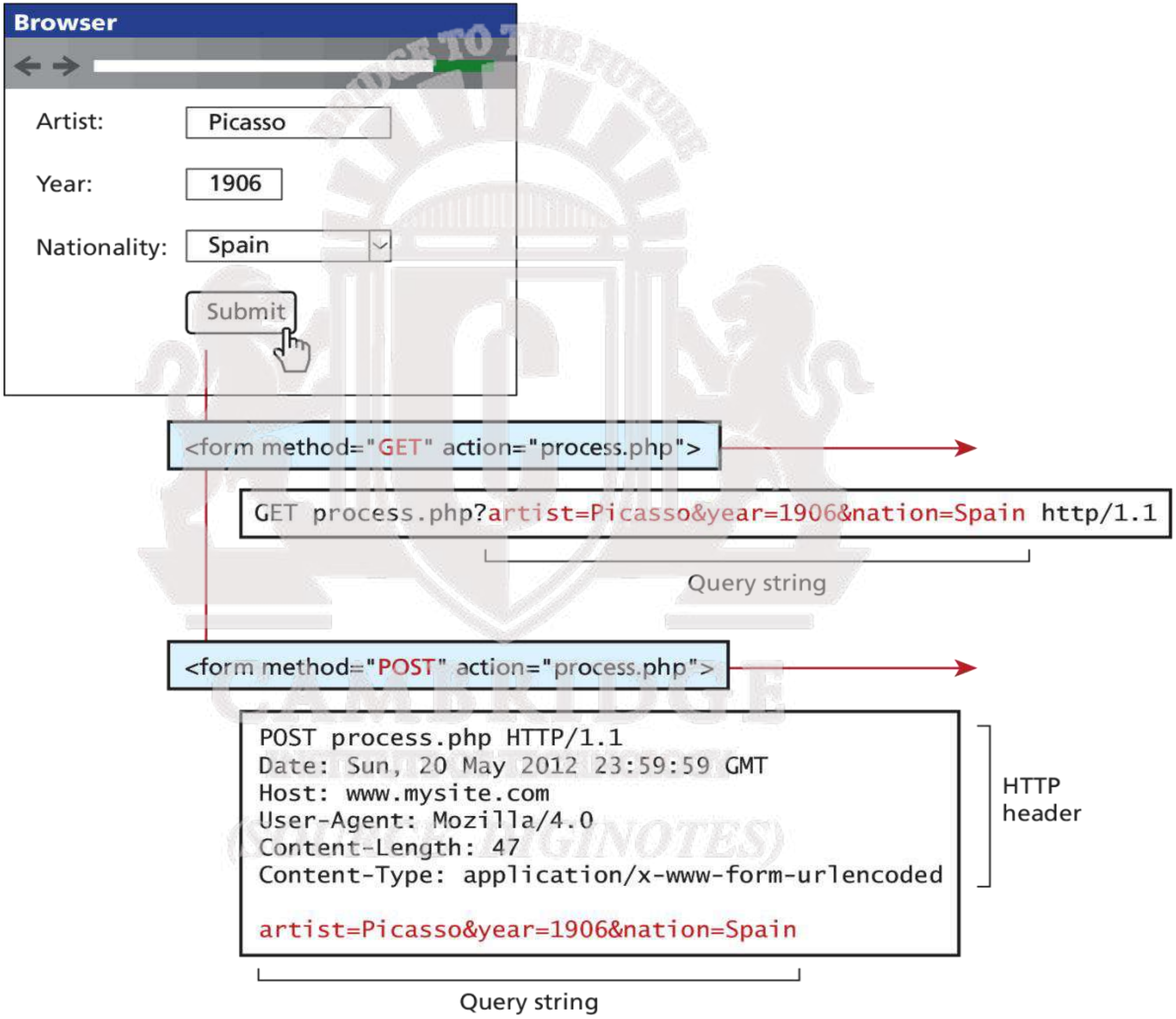
PASSING INFORMATION VIA QUERY STRINGS

❑A web page can pass query string information from the browser to the server using one of the two methods:

- A query string within the URL(GET)
- A query string within the HTTP header(POST)

Info in Query Strings

Recall GET and POST



Artist: Picasso

Year: 1906

Nationality: Spain

```
<form method="GET" action="process.php">
```

```
GET process.php?artist=Picasso&year=1906&nation=Spain http/1.1
```

```
<form method="POST" action="process.php">
```

```
POST process.php HTTP/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

artist=Picasso&year=1906&nation=Spain
```

Source diginotes.in

Section 3 of 8

PASSING INFORMATION VIA THE URL PATH

Passing Info via URL Path

An Idealized looking link structure

❖Passing information from one page to another is done by query strings but they have drawback.

❖The URLs that result can be long and complicated.

❖while there is some dispute about whether dynamic URLs(i.e, ones with query string parameters) or static URLs are better from a search engine result optimization (or SEO)

❖Dynamic URLs (i.e., query string parameters) are a pretty essential part of web application development.

✓How can we do without them?

❖The answer is to rewrite the dynamic URL into a static one (and vice versa). This process is commonly called **URL rewriting**.

➢In below fig, the top four commerce – related results for the search term

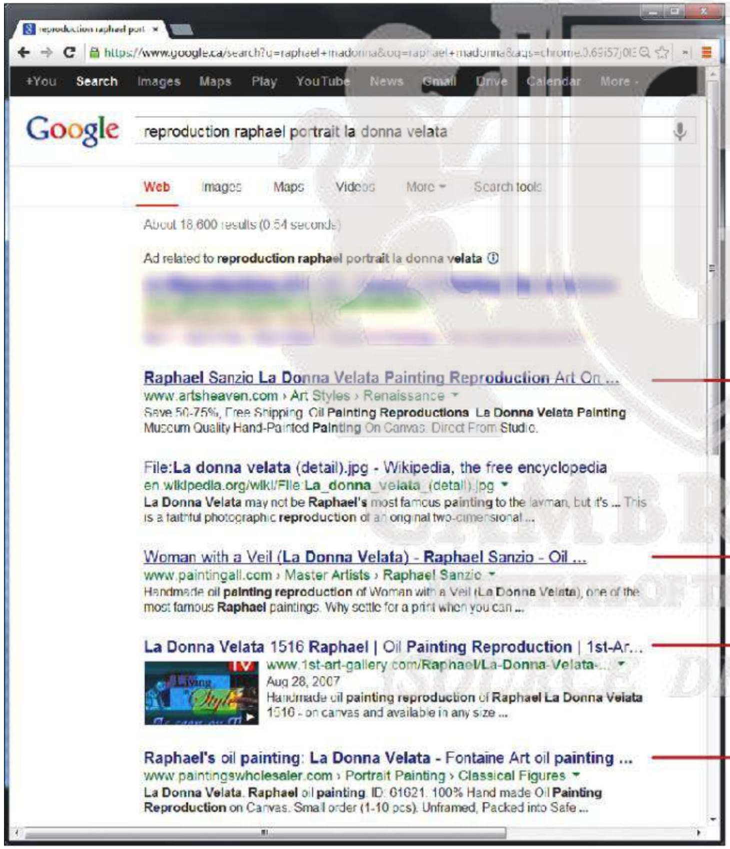
“ Reproduction Raphael portrait la donna velata” are shown along with their URLs.

➢Notice how the top three do not use query string parameters but instead put the relevant information within the folder path or the file name.

URL rewriting

Search Engine (Fine… and Human) Friendly

```
http://www.1st-art-gallery.com/Raphael/La-Donna-Velata-1516.html

http://www.paintingall.com/raphael-sanzio-woman-with-a-veil-la-donna-velata.html

http://www.artsheaven.com/raphael-la-donna-velata.html
```



reproduction raphael port ×
https://www.google.ca/search?q=raphael+madonna&oq=raphael+madonna&aqs=chrome.0.69i57j0l3
+You
Search
Images
Maps
Play
YouTube
News
Gmail
Drive
Calendar
More
Google
Web
Images
Maps
Videos
More
Search tools

reproduction raphael portrait la donna velata

About 18,600 results (0.54 seconds)

Ad related to **reproduction raphael** portrait la donna **velata**

Raphael Sanzio **La Donna Velata Painting Reproduction** Art On ...
www.artsheaven.com › Art Styles › Renaissance
Save 50-75%, Free Shipping. Oil **Painting Reproductions**. **La Donna Velata Painting** Museum Quality Hand-Painted **Painting** On Canvas. Direct From Studio.

File:**La donna velata** (detail).jpg - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/File:**La_donna_velata**_(detail).jpg
La Donna Velata may not be **Raphael's** most famous **painting** to the layman, but it's ... This is a faithful photographic **reproduction** of an original two-dimensional ...

Woman with a Veil (**La Donna Velata**) - **Raphael** Sanzio - Oil ...
www.paintingall.com › Master Artists › **Raphael** Sanzio
Handmade oil **painting reproduction** of Woman with a Veil (**La Donna Velata**), one of the most famous **Raphael** paintings. Why settle for a print when you can ...

La Donna Velata 1516 **Raphael** | Oil **Painting Reproduction** | 1st-Ar...
www.1st-art-gallery.com/**Raphael/La-Donna-Velata**-...
Aug 28, 2007
Handmade oil **painting reproduction** of **Raphael La Donna Velata** 1516 - on canvas and available in any size ...

Raphael's oil **painting**: **La Donna Velata** - Fontaine Art oil **painting** ...
www.paintingswholesaler.com › Portrait Painting › Classical Figures
La Donna Velata. **Raphael** oil **painting**. ID: 01621. 100% Hand made Oil **Painting Reproduction** on Canvas. Small order (1-10 pcs). Unframed, Packed into Safe ...

```
http://www.paintingswholesaler.com/detail.asp?vcode=6umd7krr1yqi161c&title=La+Donna+Velata
```

URL rewriting

Search Engine (Fine… and Human) Friendly

- We can try doing our own rewriting. Let us begin with the following URL with its query string information:

  www.somedomain.com/DisplayArtist.php?artist=16

- One typical alternate approach would be to rewrite the URL to:

  www.somedomain.com/artists/16.php

- Notice that the query string name and value have been turned into path names. One could improve this to make it more SEO friendly using the following:

  www.somedomain.com/artists/Mary-Cassatt

URL rewriting in Apache and linux

You are not yet ready grasshoper

❖The mod_rewrite module uses a rule-based rewriting engine that utilizes Perl compatible regular expressions to change the URLs so that the requested URL can be mapped or redirected to another URL internally.

Section 4 of 8

COOKIES

Cookies

mmmmm

- **Cookies** are a client-side approach for persisting state information.

- They are name=value pairs that are saved within one or more text files that are managed by the browser.

Cookies

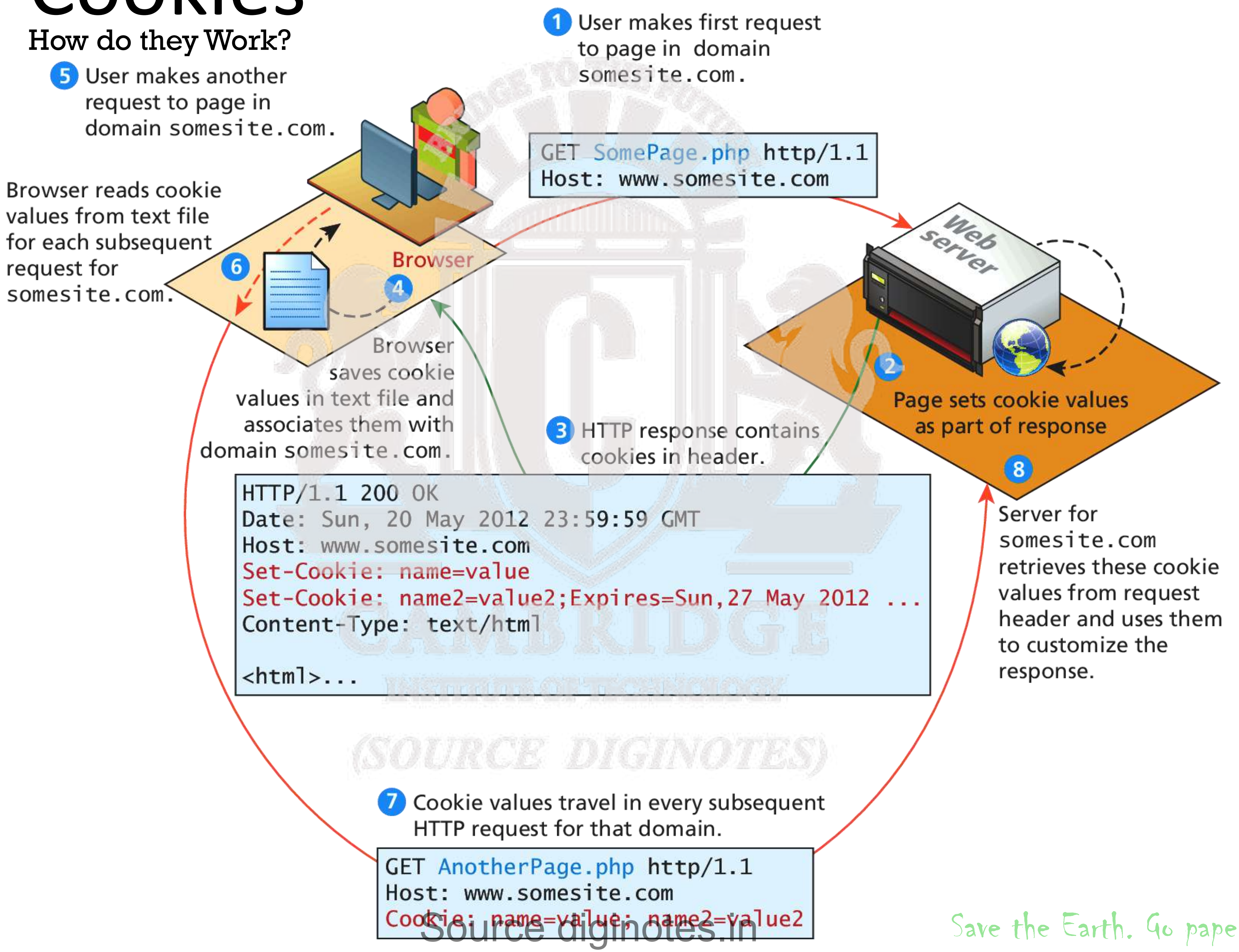
How do they Work?

❖While cookie information is stored and retrieved by the browser, the information in a cookie travels within the HTTP header.

- Sites that use cookies should not depend on their availability for critical features
- The user can delete cookies or tamper with them

Cookies

How do they Work?



1 User makes first request to page in domain somesite.com.

```
GET SomePage.php http/1.1
Host: www.somesite.com
```

2 Page sets cookie values as part of response

3 HTTP response contains cookies in header.

```
HTTP/1.1 200 OK
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.somesite.com
Set-Cookie: name=value
Set-Cookie: name2=value2;Expires=Sun,27 May 2012 ...
Content-Type: text/html

<html>...
```

4 Browser saves cookie values in text file and associates them with domain somesite.com.

5 User makes another request to page in domain somesite.com.

6 Browser reads cookie values from text file for each subsequent request for somesite.com.

7 Cookie values travel in every subsequent HTTP request for that domain.

```
GET AnotherPage.php http/1.1
Host: www.somesite.com
Cookie: name=value; name2=value2
```

8 Server for somesite.com retrieves these cookie values from request header and uses them to customize the response.

Cookies

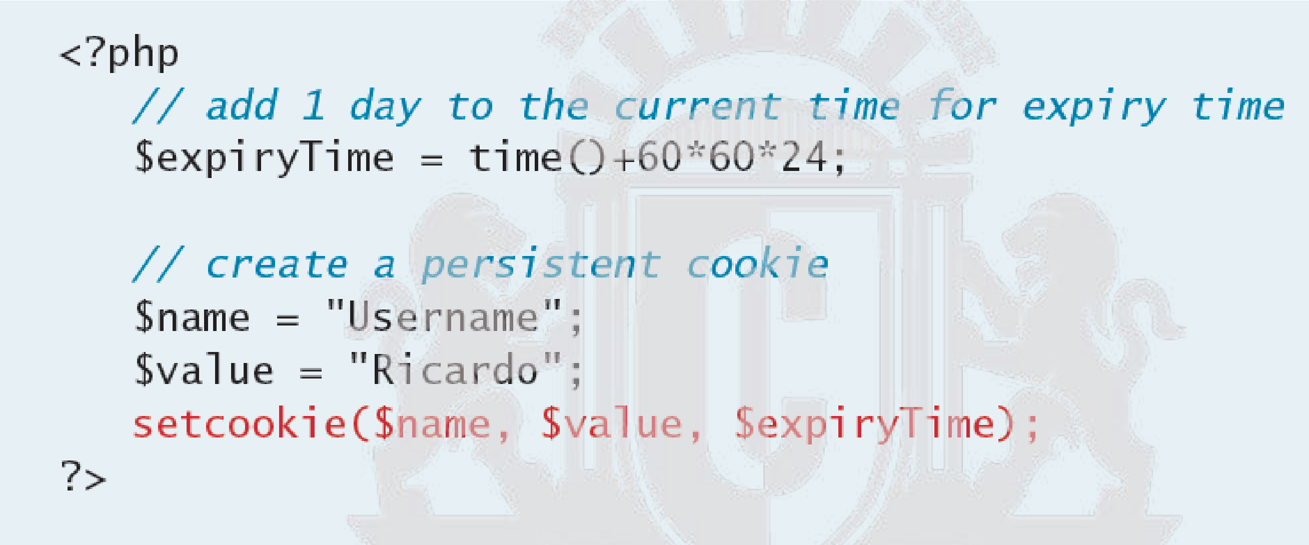
Chocolate and peanut butter

Two kinds of Cookie

- A **session cookie** has no expiry stated and thus will be deleted at the end of the user browsing session.
- **Persistent cookies** have an expiry date specified;

Using Cookies

Writing a cookie



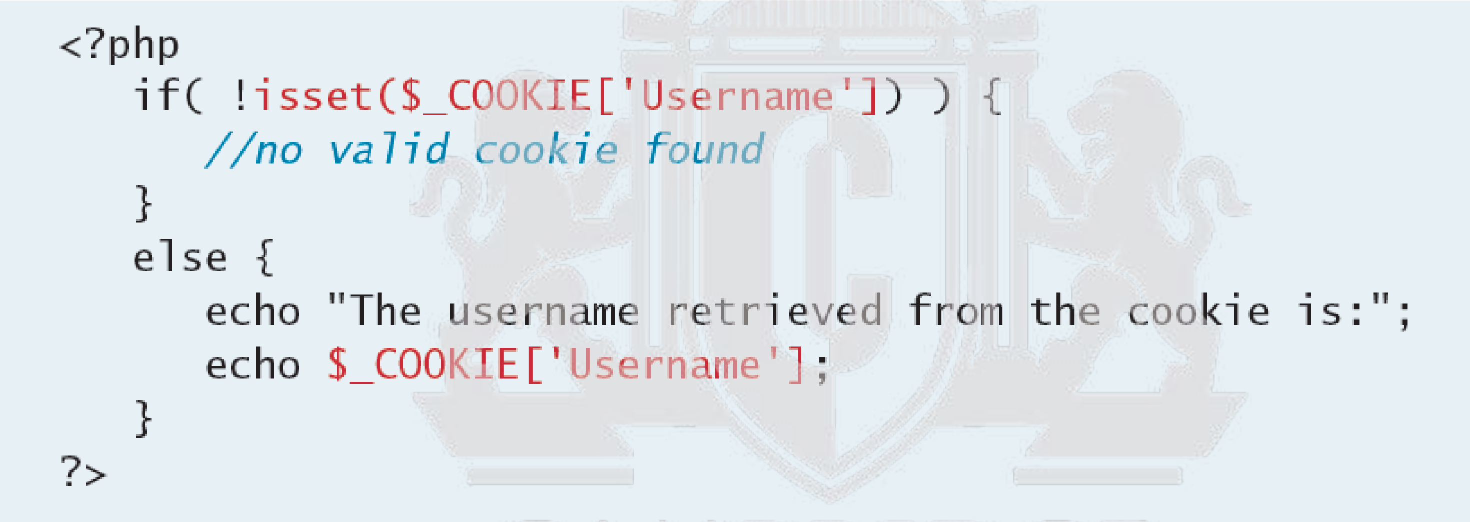
```
<?php
   // add 1 day to the current time for expiry time
   $expiryTime = time()+60*60*24;

   // create a persistent cookie
   $name = "Username";
   $value = "Ricardo";
   setcookie($name, $value, $expiryTime);
?>
```

LISTING 13.1 Writing a cookie

It is important to note that cookies must be written before any other page output.

Using Cookies

Reading a cookie



```
<?php
    if( !isset($_COOKIE['Username']) ) {
        //no valid cookie found
    }
    else {
        echo "The username retrieved from the cookie is:";
        echo $_COOKIE['Username'];
    }
?>
```

LISTING 13.2 Reading a cookie

Using Cookies

Common usages

❖In addition to being used to track authenticated users and shopping carts, cookies can implement:

- “Remember me” persistent cookie
- Store user preferences
- Track a user’s browsing behavior

Section 5 of 8

SERIALIZATION

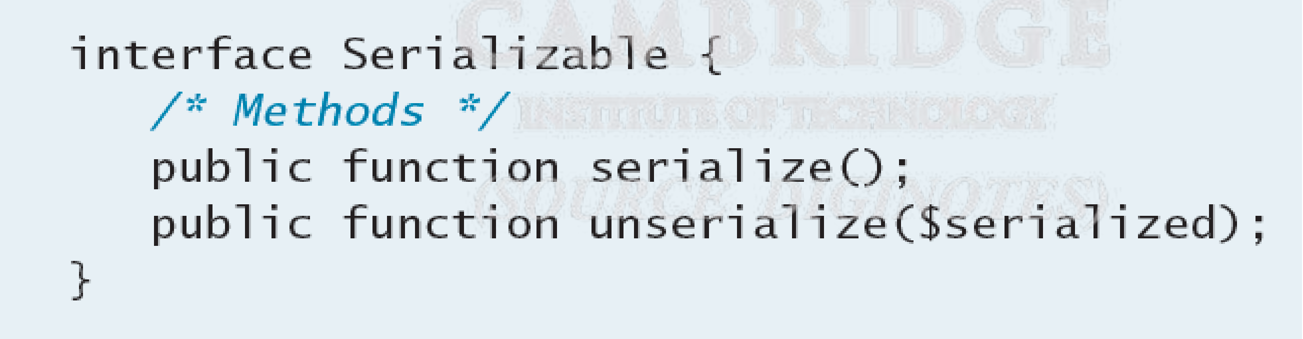
Serialization

Down to 0s and 1s

•**Serialization** is the process of taking a complicated object and reducing it down to zeros and ones for either storage or transmission.

•In PHP objects can easily be reduced down to a binary string using the **serialize()** function.

•The string can be reconstituted back into an object using the **unserialize()** method

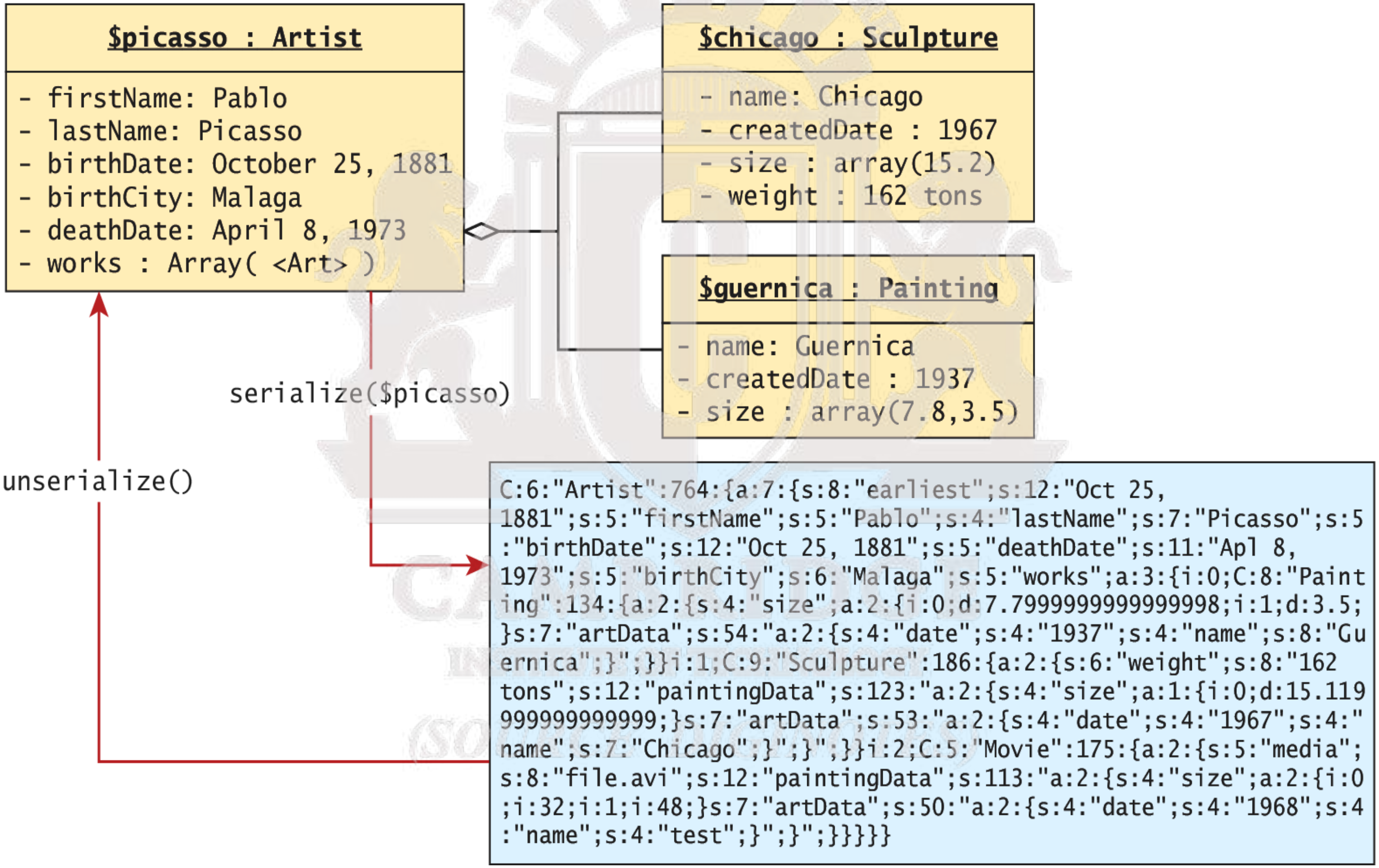


```
interface Serializable {
   /* Methods */
   public function serialize();
   public function unserialize($serialized);
}
```

LISTING 13.3 The Serializable interface

Serialization

Serialization and deserialization



```
$picasso : Artist
- firstName: Pablo
- lastName: Picasso
- birthDate: October 25, 1881
- birthCity: Malaga
- deathDate: April 8, 1973
- works : Array( <Art> )
```

```
$chicago : Sculpture
- name: Chicago
- createdDate : 1967
- size : array(15.2)
- weight : 162 tons
```

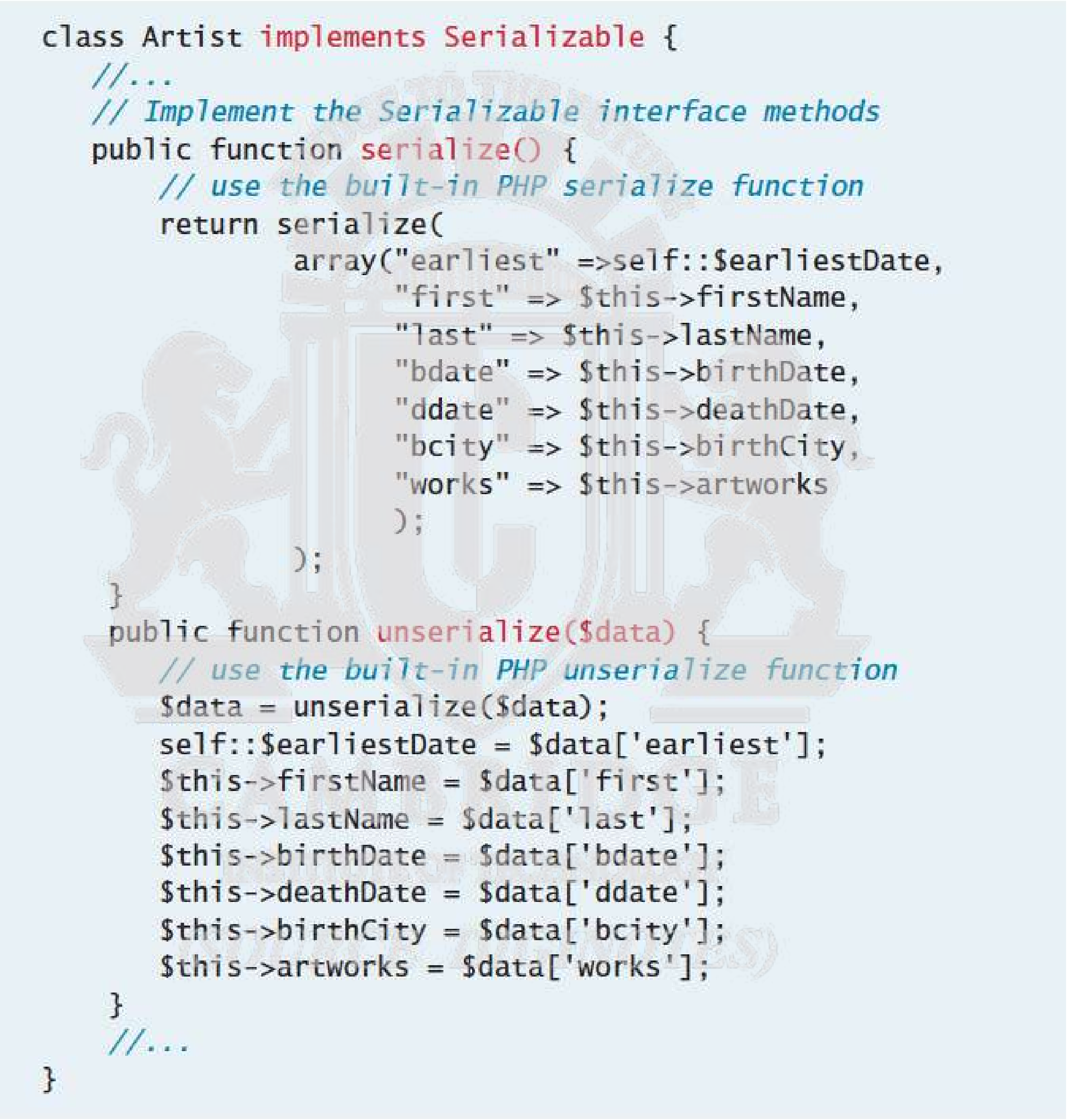
```
$guernica : Painting
- name: Guernica
- createdDate : 1937
- size : array(7.8,3.5)
```

serialize($picasso)

unserialize()

```
C:6:"Artist":764:{a:7:{s:8:"earliest";s:12:"Oct 25,
1881";s:5:"firstName";s:5:"Pablo";s:4:"lastName";s:7:"Picasso";s:5
:"birthDate";s:12:"Oct 25, 1881";s:5:"deathDate";s:11:"Apl 8,
1973";s:5:"birthCity";s:6:"Malaga";s:5:"works";a:3:{i:0;C:8:"Paint
ing":134:{a:2:{s:4:"size";a:2:{i:0;d:7.7999999999999998;i:1;d:3.5;
}s:7:"artData";s:54:"a:2:{s:4:"date";s:4:"1937";s:4:"name";s:8:"Gu
ernica";}";}}i:1;C:9:"Sculpture":186:{a:2:{s:6:"weight";s:8:"162
tons";s:12:"paintingData";s:123:"a:2:{s:4:"size";a:1:{i:0;d:15.119
999999999999;}s:7:"artData";s:53:"a:2:{s:4:"date";s:4:"1967";s:4:"
name";s:7:"Chicago";}";}";}}i:2;C:5:"Movie":175:{a:2:{s:5:"media";
s:8:"file.avi";s:12:"paintingData";s:113:"a:2:{s:4:"size";a:2:{i:0
;i:32;i:1;i:48;}s:7:"artData";s:50:"a:2:{s:4:"date";s:4:"1968";s:4
:"name";s:4:"test";}";}";}}}}}
```

Serialization



```
class Artist implements Serializable {
   //...
   // Implement the Serializable interface methods
   public function serialize() {
       // use the built-in PHP serialize function
       return serialize(
               array("earliest" =>self::$earliestDate,
                     "first" => $this->firstName,
                     "last" => $this->lastName,
                     "bdate" => $this->birthDate,
                     "ddate" => $this->deathDate,
                     "bcity" => $this->birthCity,
                     "works" => $this->artworks
                     );
               );
    }
    public function unserialize($data) {
       // use the built-in PHP unserialize function
       $data = unserialize($data);
       self::$earliestDate = $data['earliest'];
       $this->firstName = $data['first'];
       $this->lastName = $data['last'];
       $this->birthDate = $data['bdate'];
       $this->deathDate = $data['ddate'];
       $this->birthCity = $data['bcity'];
       $this->artworks = $data['works'];
    }
    //...
}
```

LISTING 13.4 Artist class modified to implement the Serializable interface

Serialization

Consider our Artist class

The output of calling serialize($picasso) is:

```
C:6:"Artist":764:{a:7:{s:8:"earliest";s:13:"Oct 25,
1881";s:5:"firstName";s:5:"Pablo";s:4:"lastName";s:7:"Picasso";s:5:"birthDate";s:13:"Oct
25, 1881";s:5:"deathDate";s:11:"Apl 8, 1973";s:5:"birthCity”;s:6:"Malaga";s:5:"works";
a:3:{i:0;C:8:"Painting":134:{a:2:{s:4:"size”;a:2:{i:0;d:7.7999999999999998;i:1;d:3.5;}s:7:"a
rtData";s:54:"a:2:{s:4:"date";s:4:"1937";s:4:"name";s:8:"Guernica";}";}}i:1;C:9:"Sculpture”
:186:{a:2:{s:6:"weight";s:8:"162 tons";s:13:"paintingData";
s:133:"a:2:{s:4:"size";a:1:{i:0;d:15.119999999999999;}s:7:"artData";s:53:”a:2:{s:4:"date";s
:4:"1967";s:4:"name";s:7:"Chicago";}";}";}}i:2;C:5:"Movie":175:{a:2:{s:5:"media";s:8:"file.a
vi";s:13:"paintingData";s:113:"a:2:{s:4:"size";a:2:{i:0;i:32;i:1;i:48;}s:7:"artData";s:50:"a:2:{
s:4:"date";s:4:"1968";s:4:"name";s:4:"test";}";}";}}}}}
```

If the data above is assigned to $data, then the following line will instantiate a new object identical to the original:

$picassoClone = unserialize($data);

Application of Serialization

Remember our state problem

- Since each request from the user requires objects to be reconstituted, using serialization to store and retrieve objects can be a rapid way to maintain state between requests.

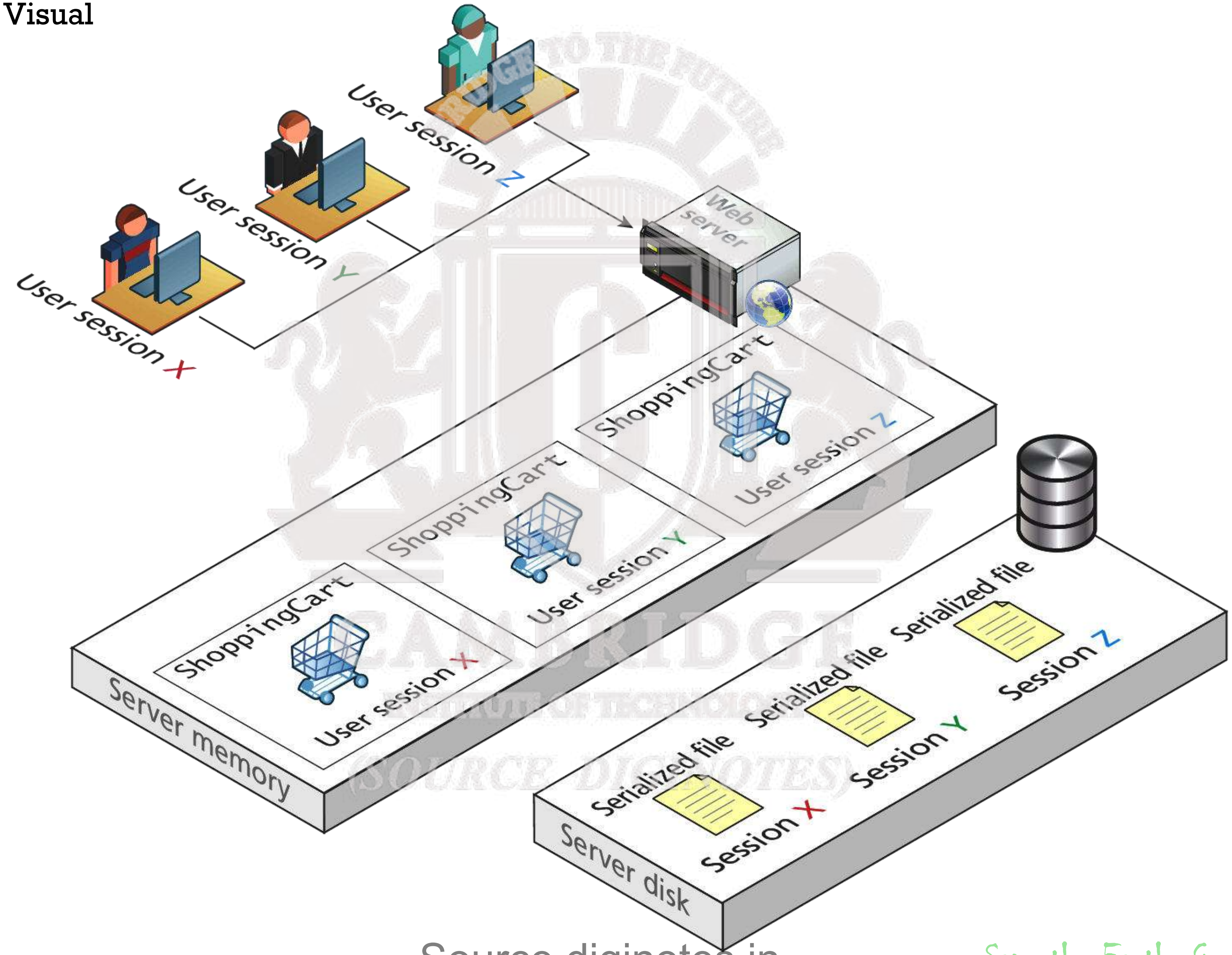
- At the end of a request you store the state in a serialized form, and then the next request would begin by deserializing it to reestablish the previous state.

Section 6 of 8

SESSION STATE

Session State

Visual



Source diginotes.in

Session State

❖All modern web development environments provide some type of session state mechanism.

❖**Session state** is a server-based state mechanism that lets web applications store and retrieve objects of any type for each unique user session.

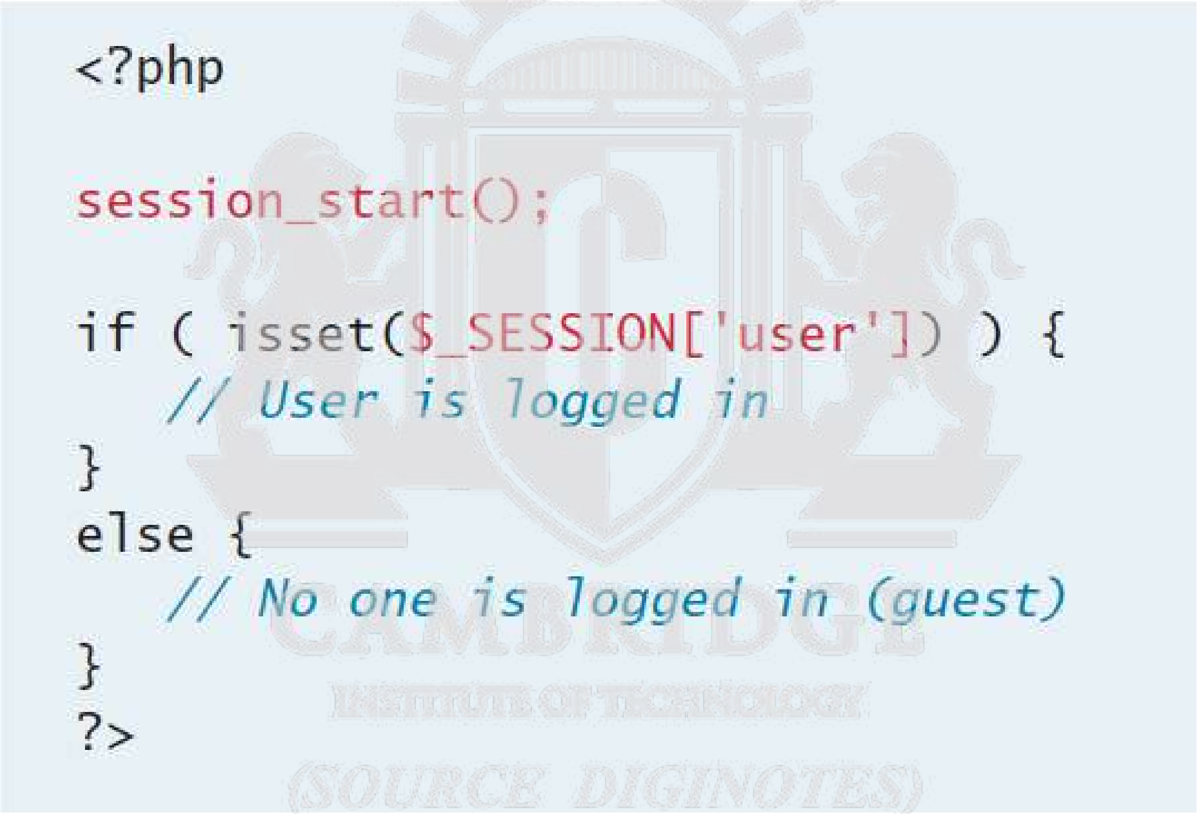
❖Session state is ideal for storing more complex objects or data structures that are associated with a user session.

❖In PHP, session state is available to the via the $_SESSION variable

❖Must use session_start() to enable sessions.

Session State

Accessing State

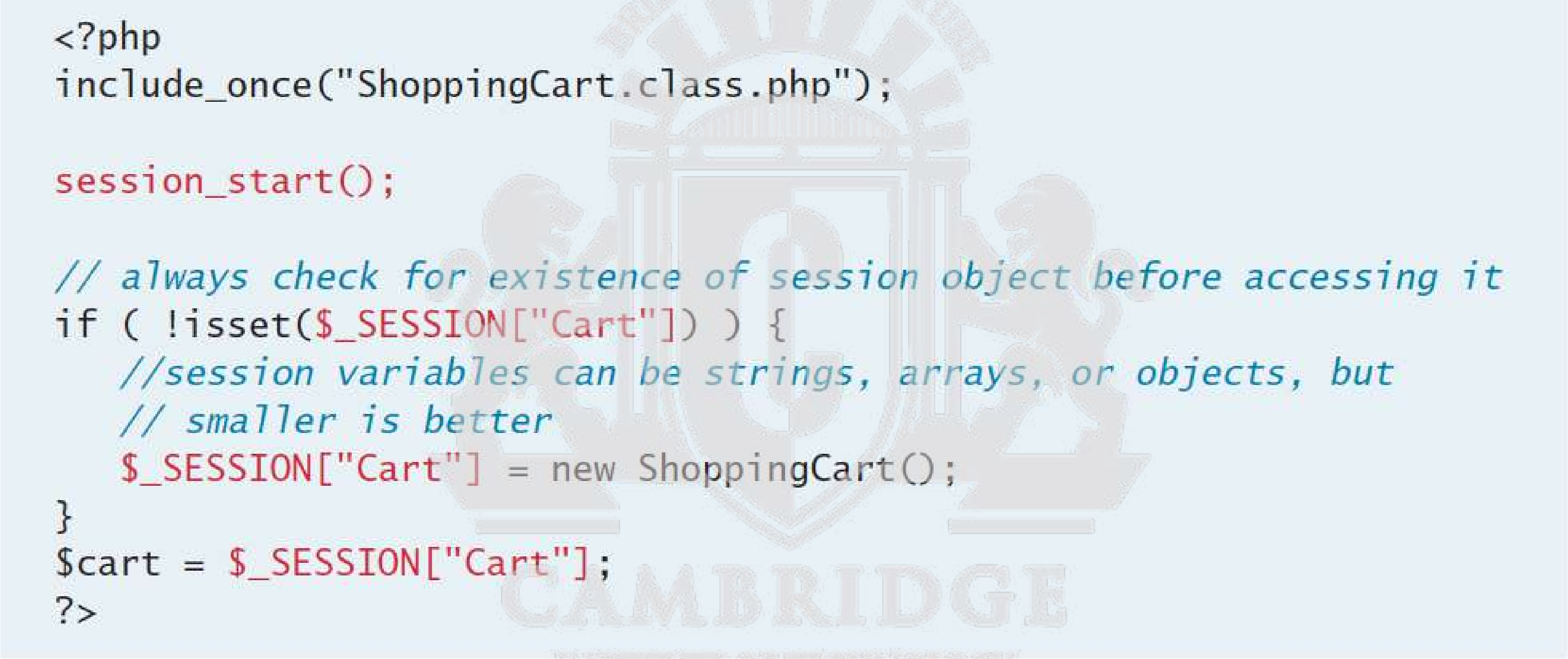


```
<?php

session_start();

if ( isset($_SESSION['user']) ) {
    // User is logged in
}
else {
    // No one is logged in (guest)
}
?>
```

LISTING 13.5 Accessing session state

Session State

Checking Session existance

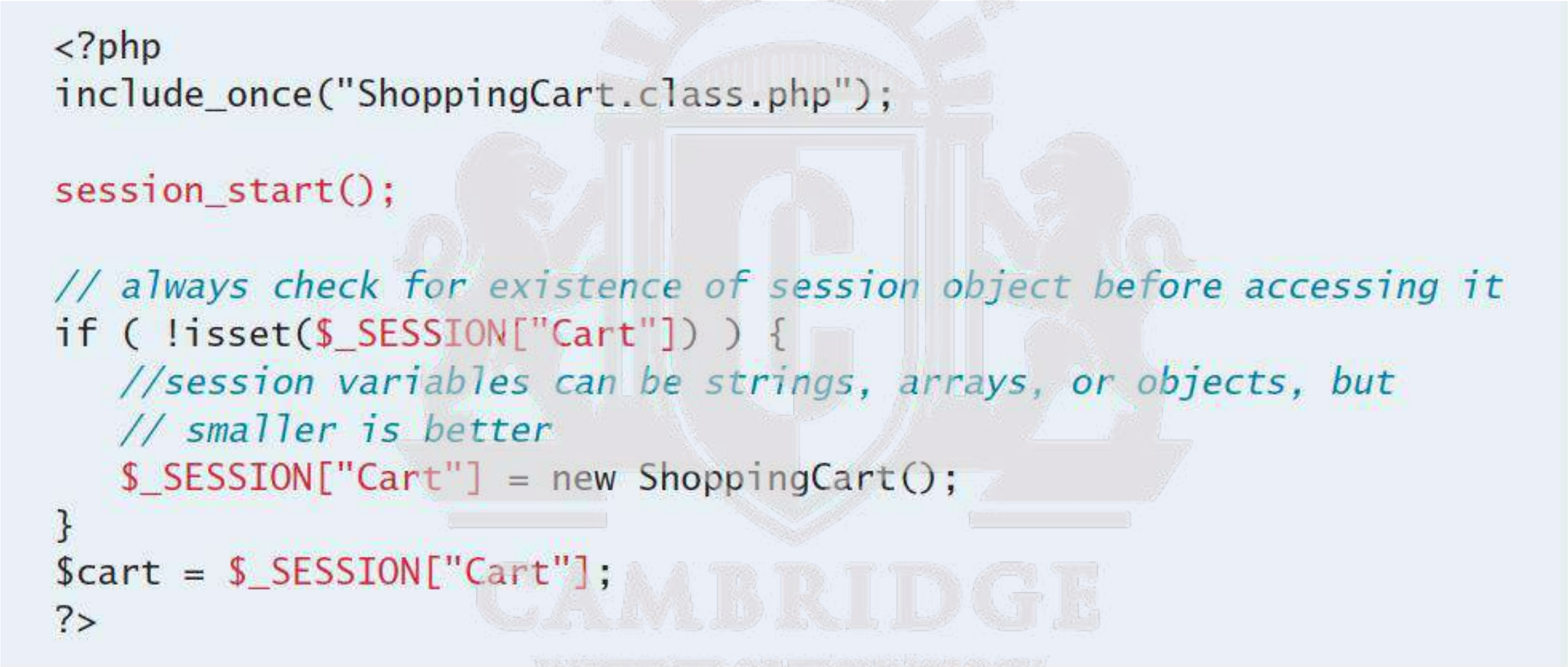


```php
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
   //session variables can be strings, arrays, or objects, but
   // smaller is better
   $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

Session State

Checking Session existence

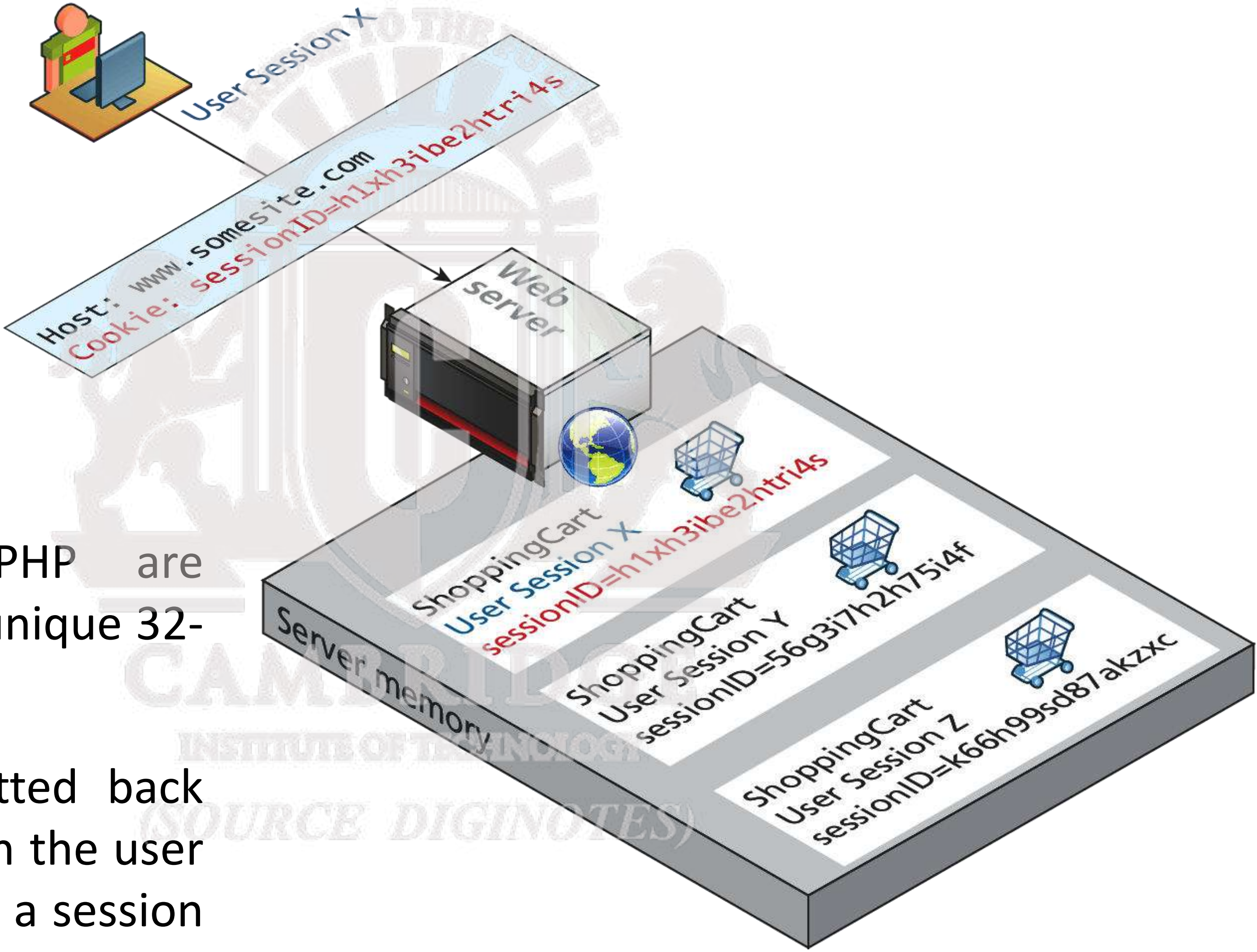


```
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
   //session variables can be strings, arrays, or objects, but
   // smaller is better
   $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

How does state session work?

It's magic right?



- ➢Sessions in PHP are identified with a unique 32-byte session ID.
- ➢This is transmitted back and forth between the user and the server via a session cookie

Source diginotes.in

How does state session work?

It’s magic right?

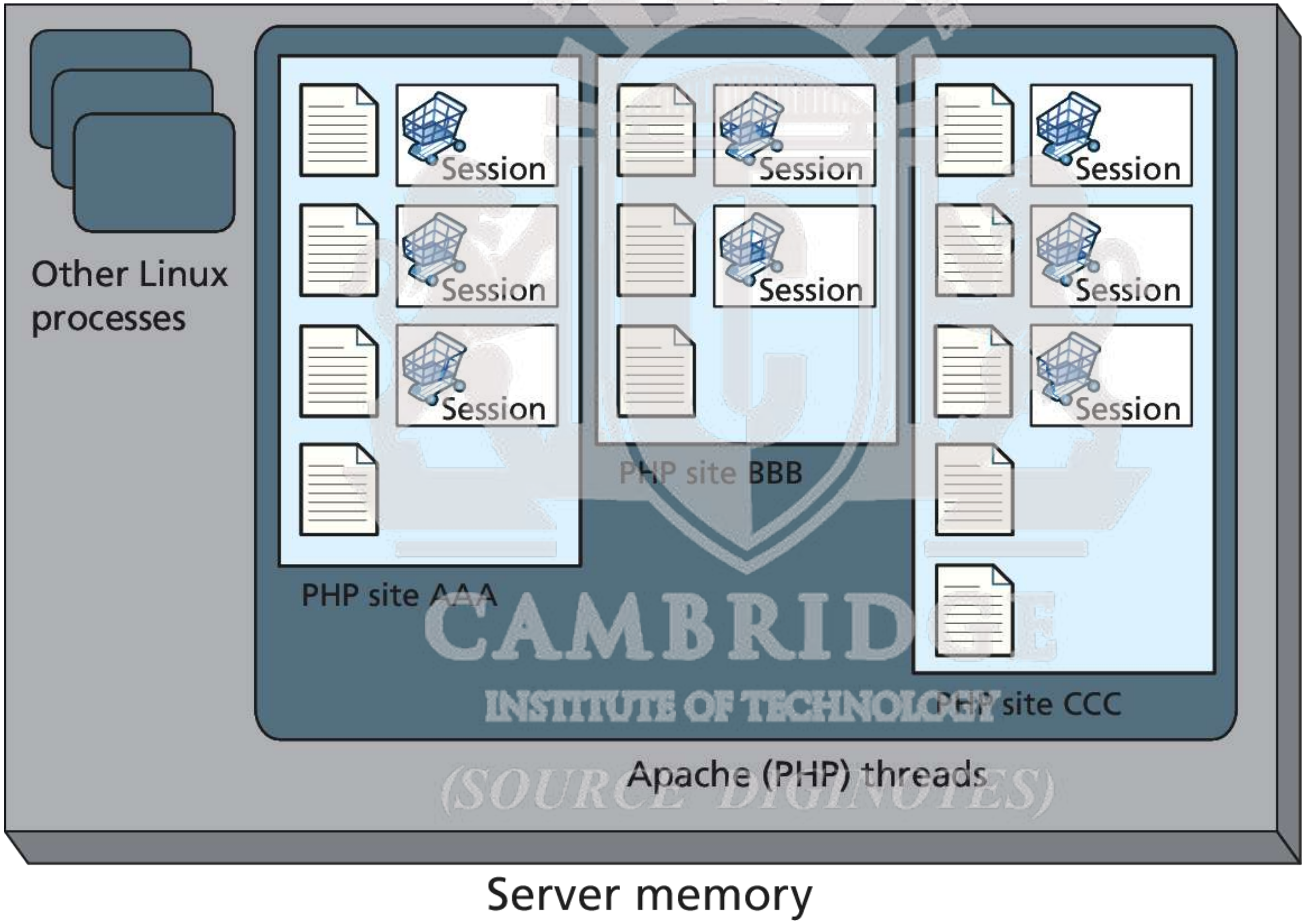
- For a brand new session, PHP assigns an initially empty dictionary-style collection that can be used to hold any state values for this session.
- When the request processing is finished, the session state is saved to some type of state storage mechanism, called a session state provider
- When a new request is received for an already existing session, the session’s dictionary collection is filled with the previously saved session data from the session state provider.

Session Storage

- It is possible to configure many aspects of sessions including where the session files are saved.
- The decision to save sessions to files rather than in memory (like ASP.NET) addresses the issue of memory usage that can occur on shared hosts as well as persistence between restarts.
- Inexpensive web hosts may sometimes stuff hundreds or even thousands of sites on each machine.
- Server memory may be storing not only session information, but pages being executed, and caching information

Session Storage

Applications and Server Memory



Source diginotes.in

Session Storage

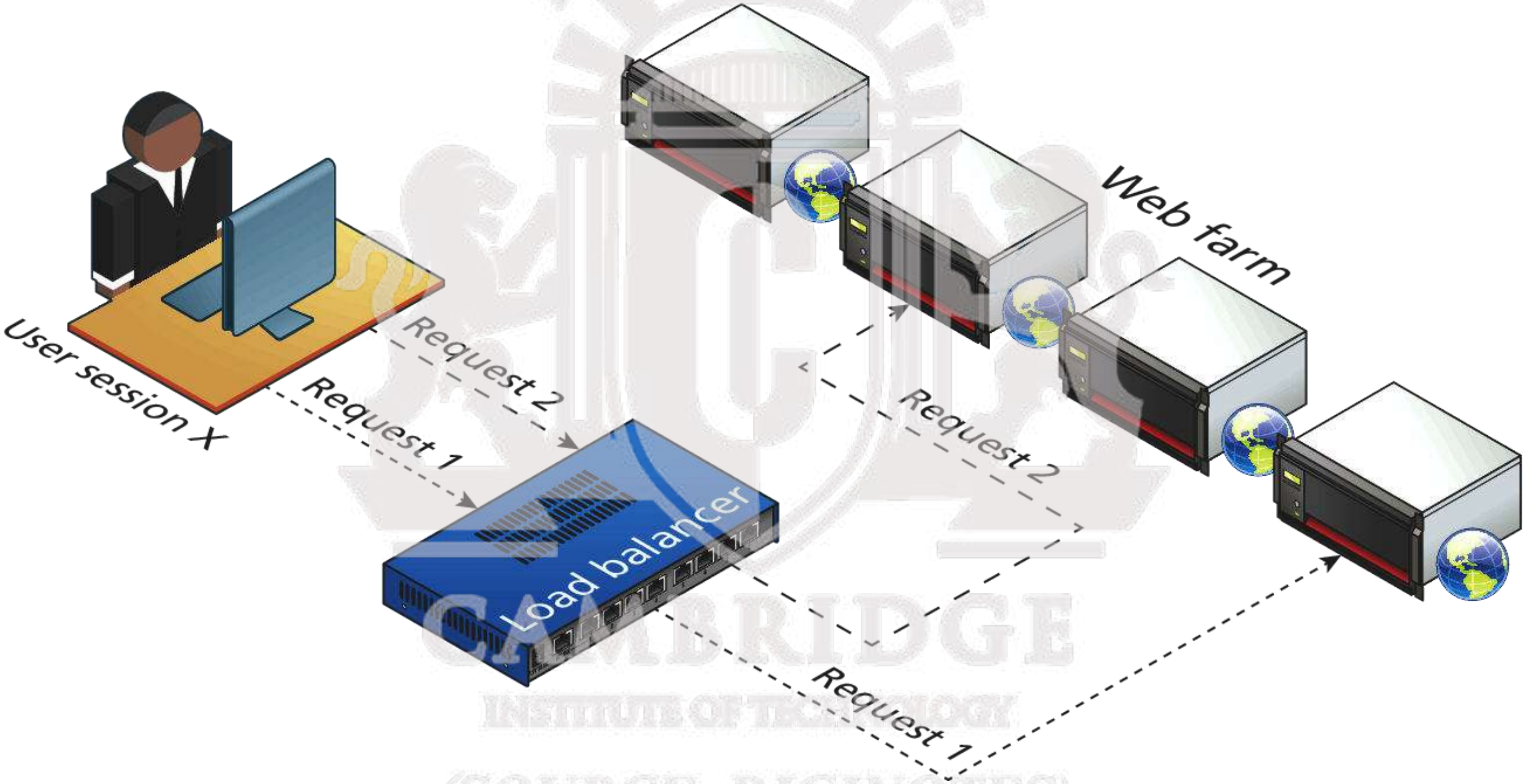
High Volume considerations

❖Higher-volume web applications often run in an environment in which multiple web servers (also called a web farm) are servicing requests.

❖In such a situation the in-process session state will not work, since one server may service one request for a particular session, and then a completely different server may service the next request for that session

Session Storage

Web Farm Sessions: Visualizing the problem



Source diginotes.in

Session Storage

Visualizing the problem

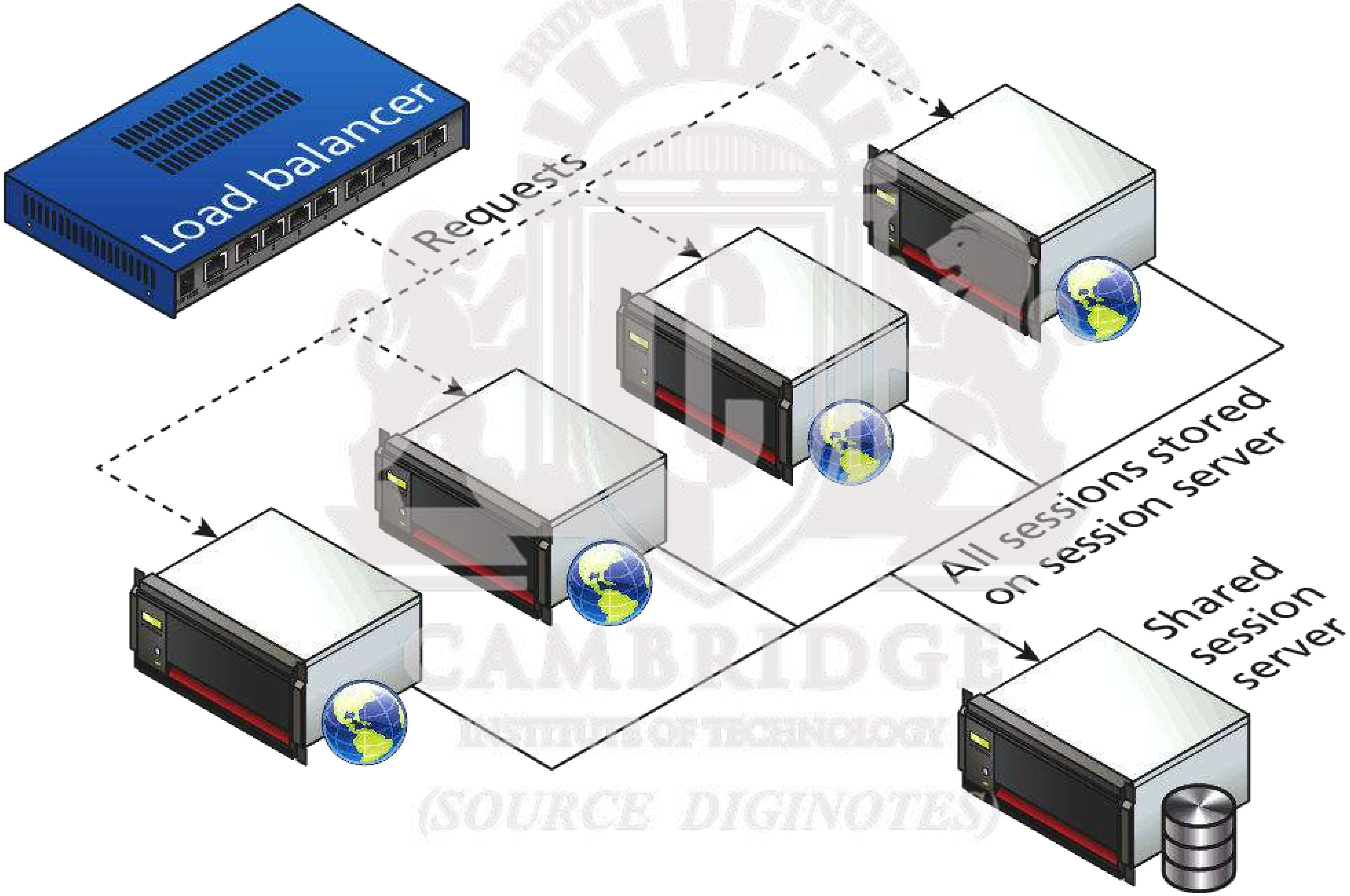
There are effectively two categories of solution to this problem.

1. Configure the load balancer to be “session aware” and relate all requests using a session to the same server.

2. Use a shared location to store sessions, either in a database, memcache, or some other shared session state mechanism

Session Storage

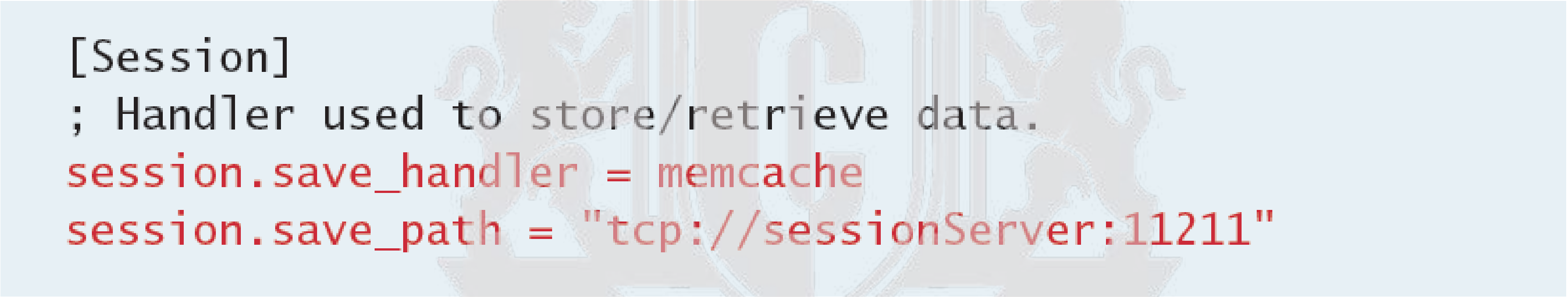
Shared location with memcache



Source diginotes.in

Session Storage

Shared location configuration in php.ini (on each webserver)



```
[Session]
; Handler used to store/retrieve data.
session.save_handler = memcache
session.save_path = "tcp://sessionServer:11211"
```

LISTING 13.7 Configuration in php.ini to use a shared location for sessions

13.7 HTML5 Web Storage

❖Web storage is a new javascript only API introduced in HTML5.

❖IT is meant to be a replacement to cookies, in that web storage is managed by the browser.

❖Unlike cookies web storage data is not transported to and from the server with every request and response

❖Web storage is not limited to 4k size barrier of cookies.

❖Limit of 5 MB but browsers are allowed to store more per domain.

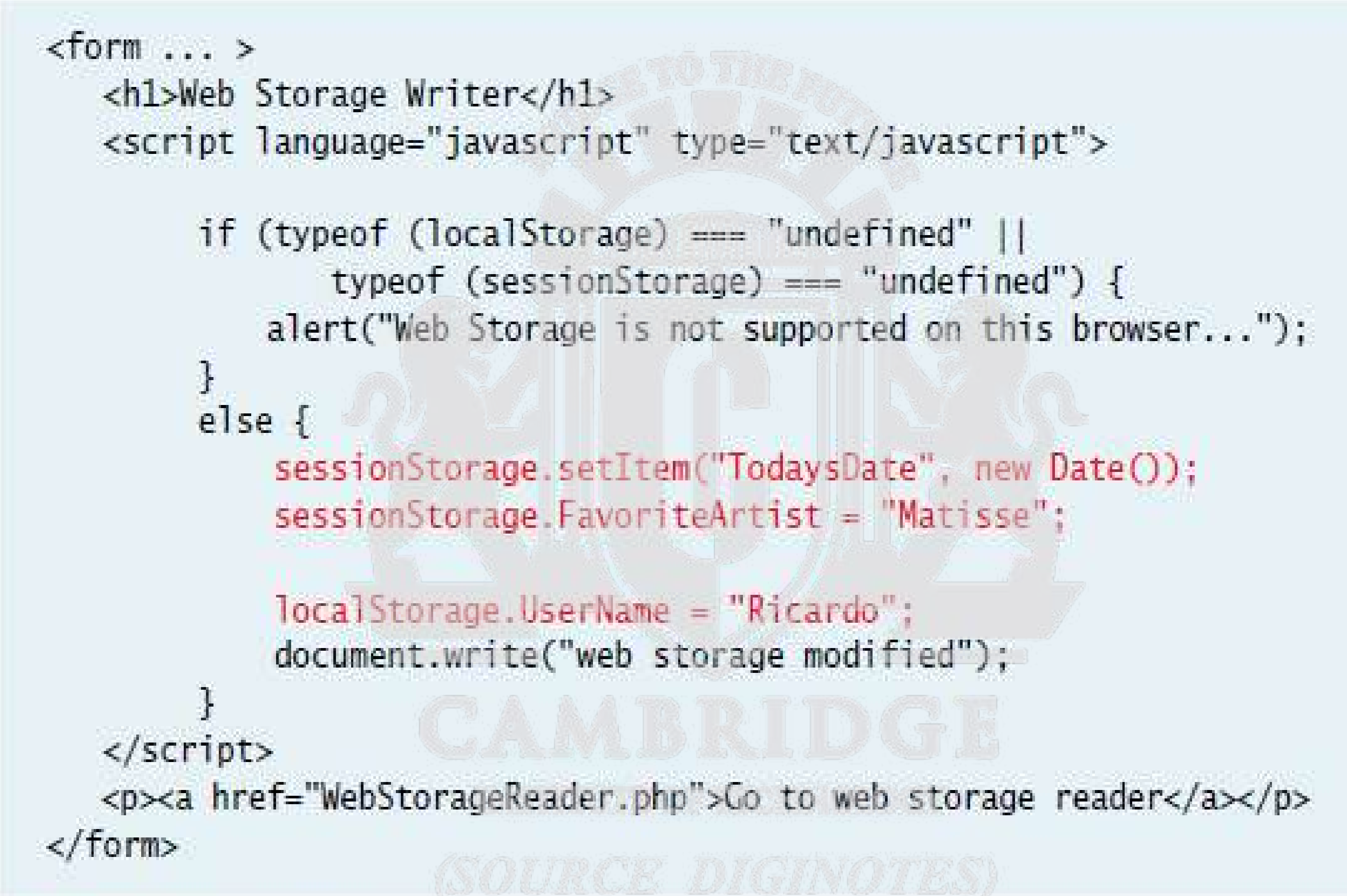
❖Just as there were two types of cookies, there are two types of web storage

- ❖Local storage – It is for saving information that will persist between browser sessions.

- ❖Session storage – It is for information that will be lost once the browser session is finished.

13.7.1 Using Web Storage

- Below code illustrates the javaScript code for writitng information to web stroage.
- There are two ways to store values in web storage
- Using setItem() function, or using the property shortcut (Eg: session stoargear.FavoriteArtist().

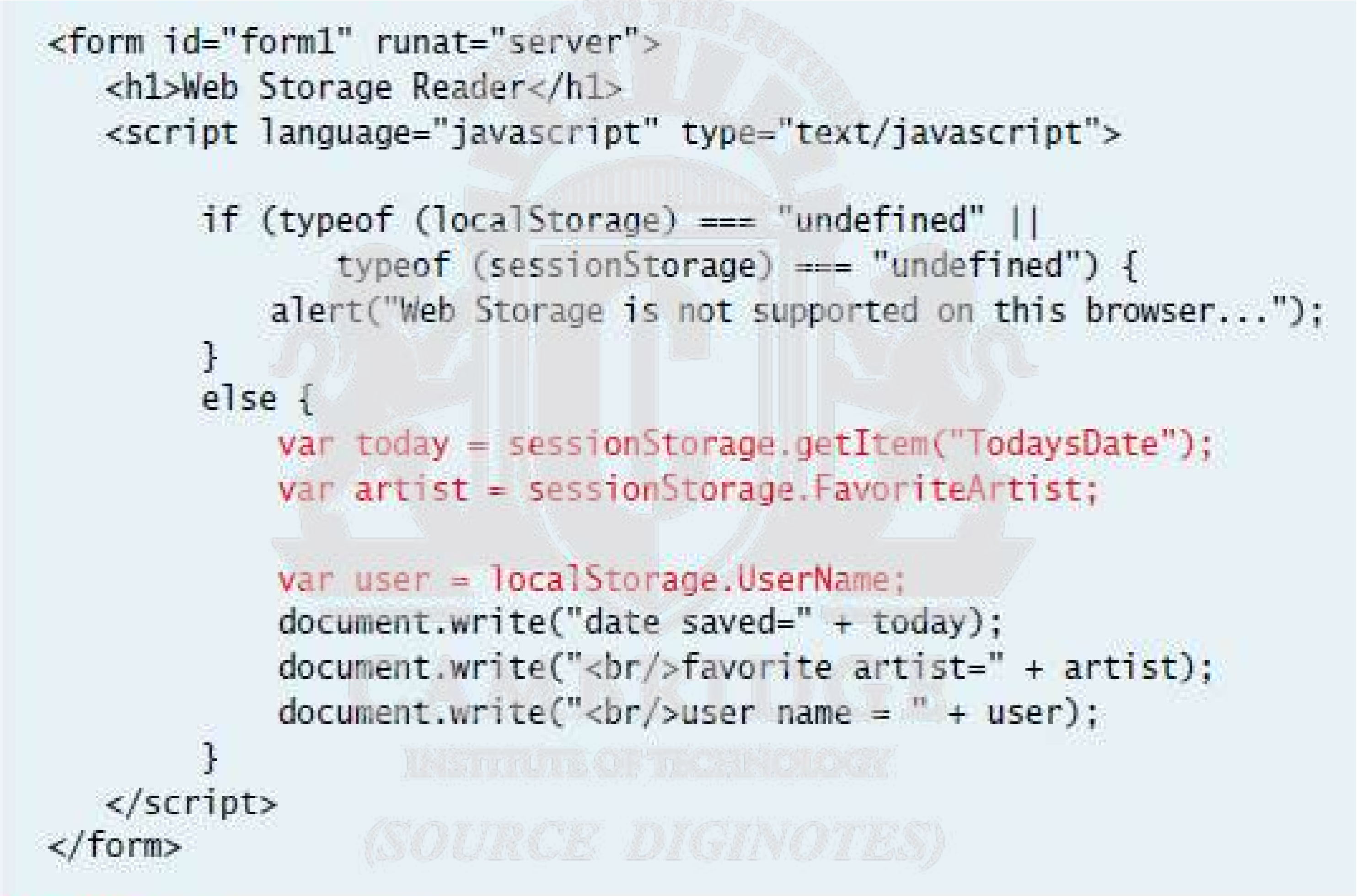


```
<form ... >
   <h1>Web Storage Writer</h1>
   <script language="javascript" type="text/javascript">

      if (typeof (localStorage) === "undefined" ||
             typeof (sessionStorage) === "undefined") {
         alert("Web Storage is not supported on this browser...");
      }
      else {
          sessionStorage.setItem("TodaysDate", new Date());
          sessionStorage.FavoriteArtist = "Matisse";

          localStorage.UserName = "Ricardo";
          document.write("web storage modified");
      }
   </script>
   <p><a href="WebStorageReader.php">Go to web storage reader</a></p>
</form>
```

LISTING 13.8 Writing web storage

- Below code illustrates the process of reading from web storage is equally straight forward.

- The difference between sessionStorage and localStorage in this example is that if you close the browser after writing and then run the code (Above code), only the local storage item will still contain a value.



```
<form id="form1" runat="server">
   <h1>Web Storage Reader</h1>
   <script language="javascript" type="text/javascript">

      if (typeof (localStorage) === "undefined" ||
            typeof (sessionStorage) === "undefined") {
         alert("Web Storage is not supported on this browser...");
      }
      else {
         var today = sessionStorage.getItem("TodaysDate");
         var artist = sessionStorage.FavoriteArtist;

         var user = localStorage.UserName;
         document.write("date saved=" + today);
         document.write("<br/>favorite artist=" + artist);
         document.write("<br/>user name = " + user);
      }
   </script>
</form>
```

LISTING 13.9 Reading web storage

13.7.2 Why Would We Use Web Storage?

❖Cookies have the disadvantage of being limited in size, potentially disabled by the user, other security attacks and being sent in every single request and response to and from a given domian.

❖Web storage is not as a cookie replacement but as a local cache for relatively static items available to javascript.

❖One use of web storage is to store static content downloaded asynchronously such as xml or JSON from a web service in web storage, this reducing server load for subsequent requests by the session.

❖Below fig illustrates an example of how web storage could be used as a mechanism for reducing server data request, there by speeding up the display of the page on the browser as well as reducing load on the server.

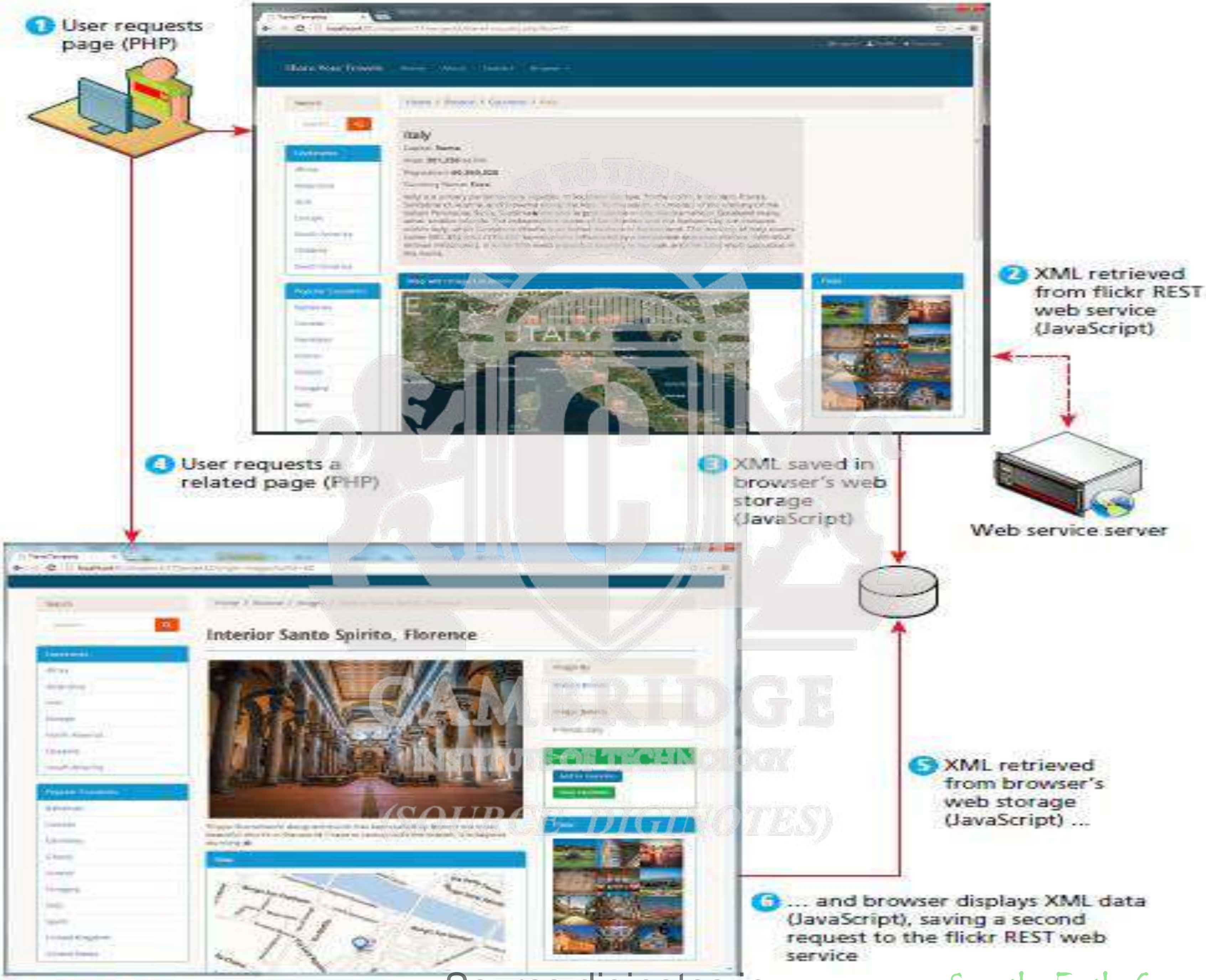

1 User requests page (PHP)
Italy
2 XML retrieved from flickr REST web service (JavaScript)
Web service server
3 XML saved in browser's web storage (JavaScript)
4 User requests a related page (PHP)
Interior Santo Spirito, Florence
5 XML retrieved from browser's web storage (JavaScript) ...
6 ... and browser displays XML data (JavaScript), saving a second request to the flickr REST web service

FIGURE 13.13 Using web storage

13.8 Caching

13.8 Caching

❖Caching is the vital way to improve the performance of web applications. Your browser uses caching to speed up the user experience by using locally stored versions of images and other files rather than re – requesting the files from the server.

❖A server side developer only had limited control over browser caching.

❖Why is this necessary?

❖Every time a PHP page is requested, it must be fetched, parsed and executed by the PHP engine and end result is HTML that is sent back to the requestor.

- On way to address this problem is to cache the generated markup in server memory so that subsequent requests can be served from memory rather than from the execution of the page.

- There are two basic strategies to caching web applications.

  - PAGE OUTPUT CACHING

    It saves the rendered output of a page or user control and reuses the output instead of reprocessing the page when a user requests the page again

  - APPLICATION DATA CACHING

    It allows the developer to programmatically cache data.

13.8.1 Page Output Caching

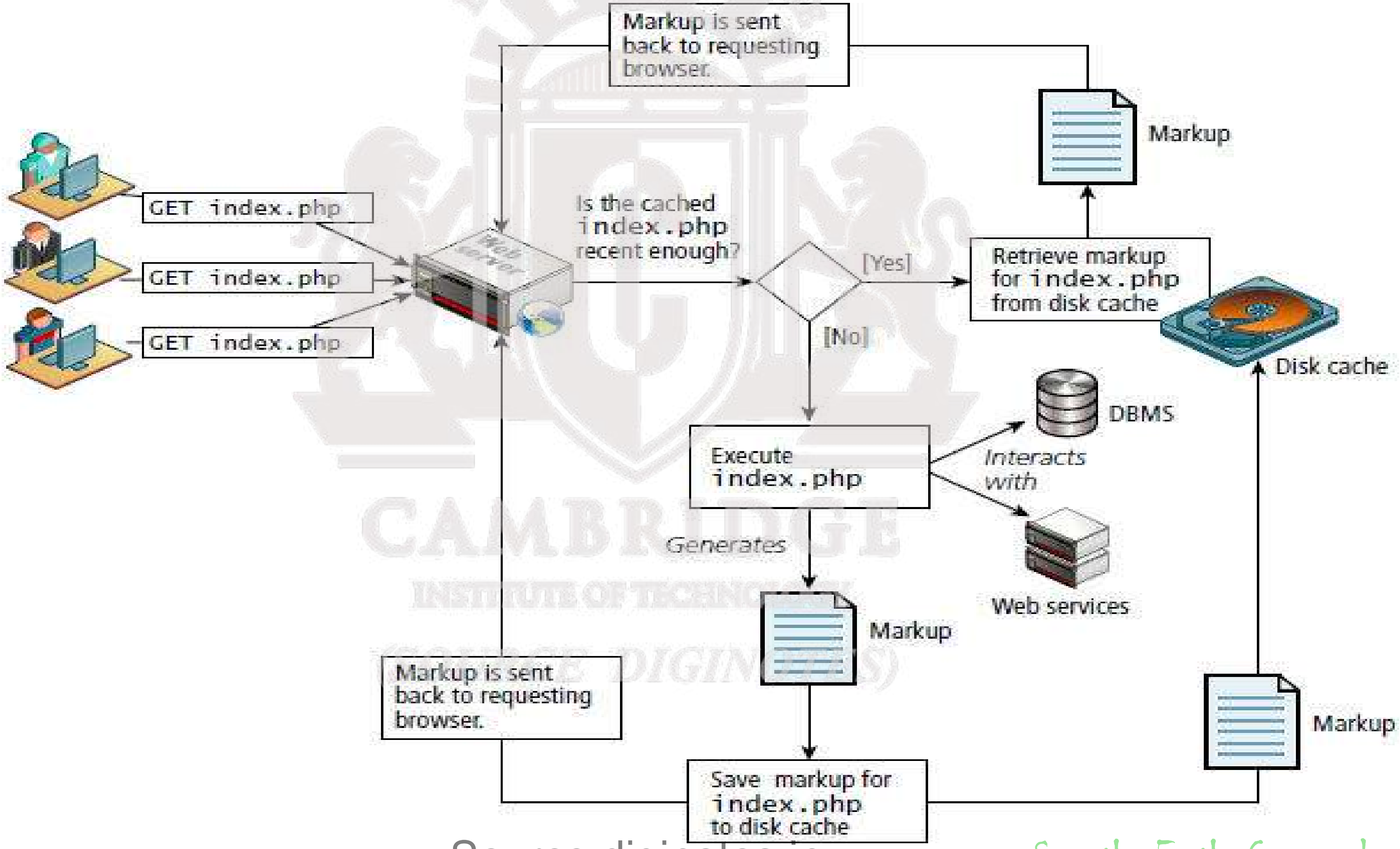

Markup is sent back to requesting browser.
Markup
GET index.php
GET index.php
GET index.php
Web server
Is the cached index.php recent enough?
[Yes]
Retrieve markup for index.php from disk cache
[No]
Disk cache
DBMS
Execute index.php
Interacts with
Web services
Generates
Markup
Markup is sent back to requesting browser.
Markup
Save markup for index.php to disk cache

FIGURE 13.14 Page output caching

- ❖There are two models for page caching:
  - ❖Full page caching
  - ❖Partial page caching :

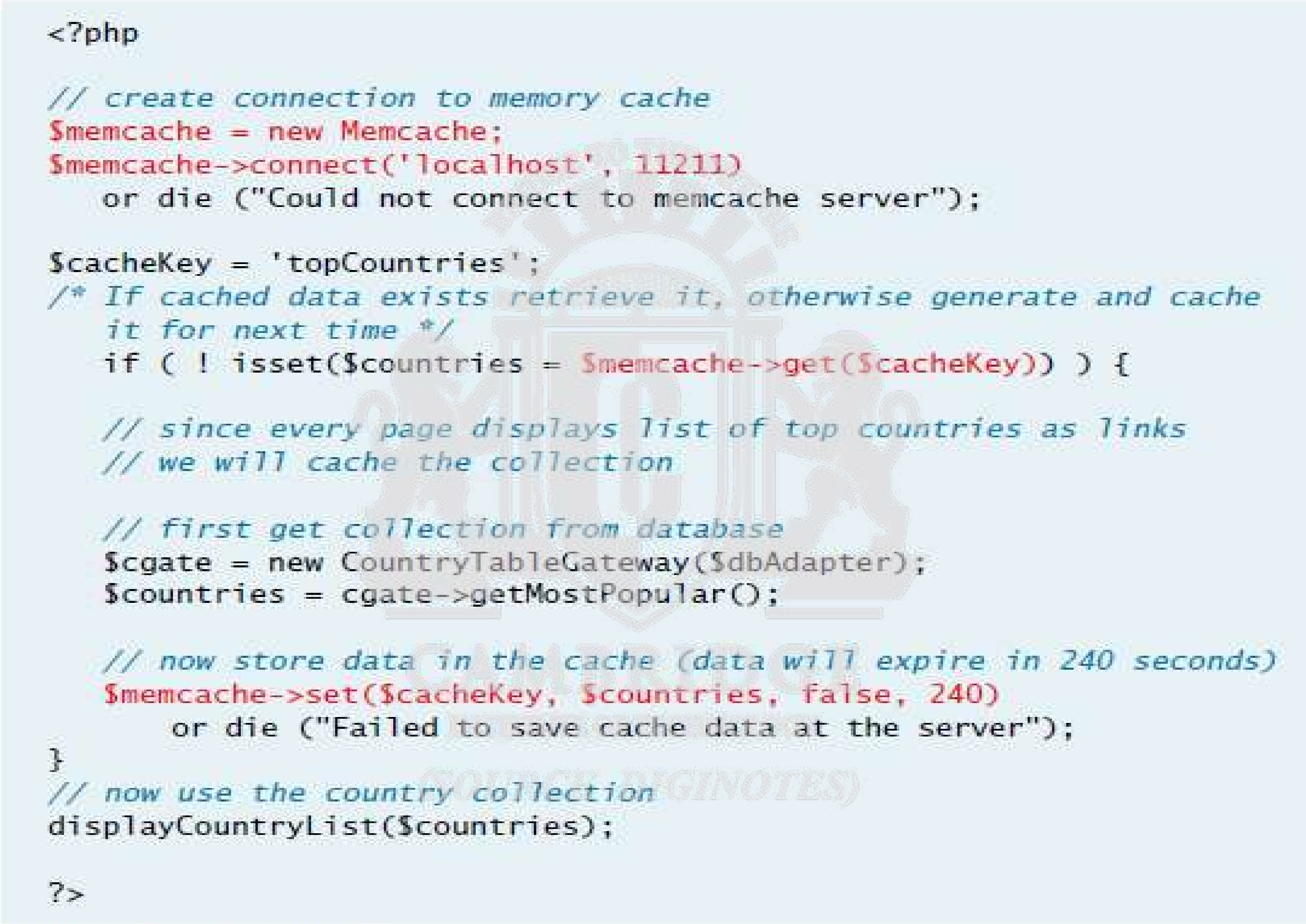
    Only specific parts of page are cached while other parts are dynamically generated in the normal manner.

13.8.2 Application Data Caching

❖One of the biggest drawbacks with page output caching is that performance gains will only be had if the entire cached page is the same for numerous requests.

❖An alternate strategy is to use application data caching in which a page will programmatically place commonly used collections of data that require time intensive queries from the database or web server into cache memory, and then other pages that also need that same data can use the cache version rather than re – retrieve it from its original location.

❖While the default installation of PHP does not come with an application caching ability, a widely available free PECL extension called memcache is used for this ability.



```php
<?php

// create connection to memory cache
$memcache = new Memcache;
$memcache->connect('localhost', 11211)
   or die ("Could not connect to memcache server");

$cacheKey = 'topCountries';
/* If cached data exists retrieve it, otherwise generate and cache
   it for next time */
   if ( ! isset($countries = $memcache->get($cacheKey)) ) {

   // since every page displays list of top countries as links
   // we will cache the collection

   // first get collection from database
   $cgate = new CountryTableGateway($dbAdapter);
   $countries = cgate->getMostPopular();

   // now store data in the cache (data will expire in 240 seconds)
   $memcache->set($cacheKey, $countries, false, 240)
       or die ("Failed to save cache data at the server");
}
// now use the country collection
displayCountryList($countries);

?>
```

LISTING 13.10 Using memcache

Advanced JavaScript & JQuery

Chapter 15

Section 1 of 6

JAVASCRIPT PSEUDO-CLASSES

15.1 JavaScript Pseudo-Classes

❖JavaScript has no formal class mechanism, it does support objects (DOM).

❖While most OO languages that supports objects also support classes formally, JavaScript does not.

❖Instead we define Pseudo – classes through a variety of syntax.

15.1.1 Using Object Literals

Without Classes

•An array in JavaScript can be instantiated

```
var daysofweek = [“sun” , ”mon” , “tue”…..];
```

•An object can be instantiated using object literals.

•**Object literals** are a list of key-value pairs with colons between the key and value with commas separating key-value pairs.

•Object literals are also known as **Plain Objects** in jQuery.

Object Oriented Design

Without Classes

- JavaScript has no formal class mechanism.

Simple Variable
var car = "Fiat";

Then this
var car = {type:"Fiat", model:500, color:"white"};

Properties	Methods:
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.brake()
car.color = white	car.stop()

Using Object Literals

Consider a die (single dice)

```
//define a 6 faced dice, with a red color.

var oneDie =     { color : "FF0000",

                   faces : [1,2,3,4,5,6]

                 };
```

These elements can be accessed using dot notation.

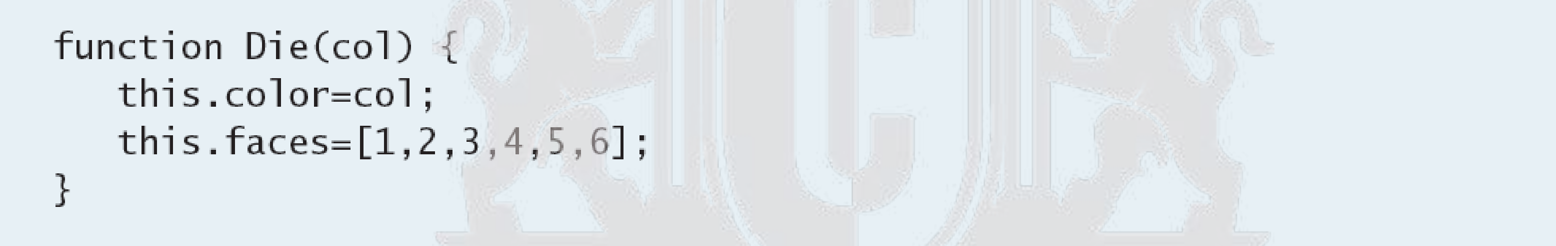
For instance

```
oneDie.color="0000FF";
```

15.1.2Emulate Classes with functions

We told you this would get weird

Use functions to encapsulate variables and methods together.



```
function Die(col) {
   this.color=col;
   this.faces=[1,2,3,4,5,6];
}
```

LISTING 15.1 Very simple Die pseudo-class definition as a function

Instantiation looks much like in PHP:

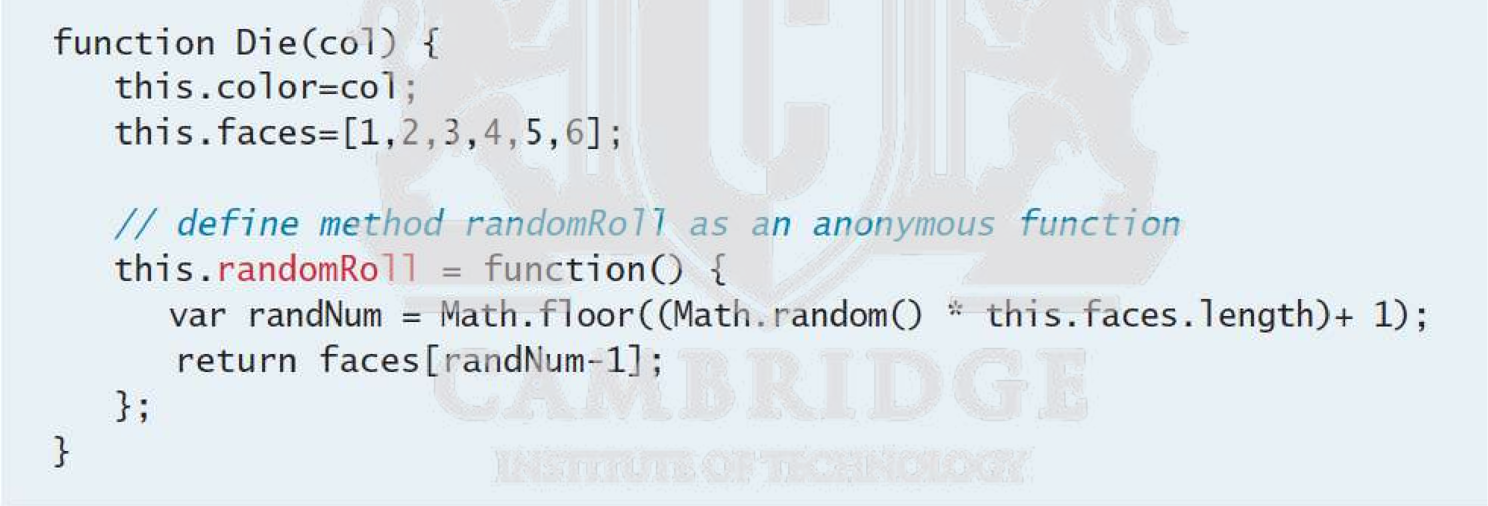
var oneDie = new Die("0000FF");

Emulate Classes with functions

Adding methods to the objects

- To define a method in an object’s function one can either define it internally or use a reference to a function define outside the class.

- One technique for adding a method inside of a class definition is by assigning an anonymous function to a variable



```
function Die(col) {
    this.color=col;
    this.faces=[1,2,3,4,5,6];

    // define method randomRoll as an anonymous function
    this.randomRoll = function() {
        var randNum = Math.floor((Math.random() * this.faces.length)+ 1);
        return faces[randNum-1];
    };
}
```

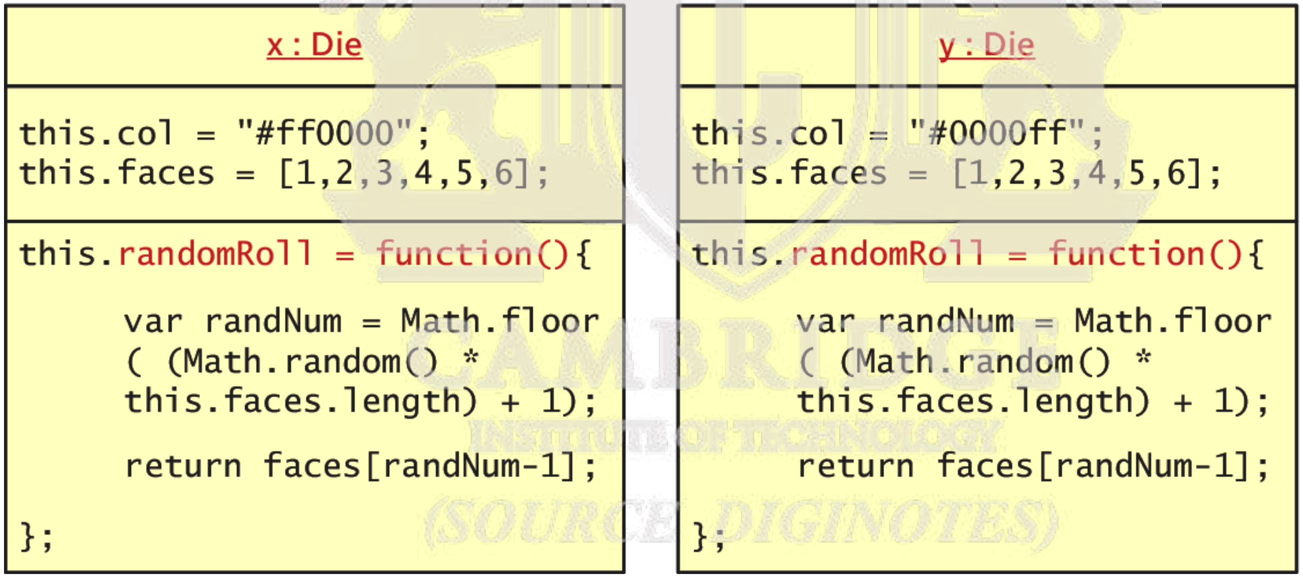
LISTING 15.2 Die pseudo-class with an internally defined method

var oneDie = new Die("0000FF");
console.log(oneDie.randomRoll() + " was rolled");

Emulate Classes with functions

Not very efficient

- This mechanism for methods in effective, it is not a memory efficient approach because each inline method is redefined for each new object



x : Die

```
this.col = "#ff0000";
this.faces = [1,2,3,4,5,6];
```

```
this.randomRoll = function(){

      var randNum = Math.floor
      ( (Math.random() *
      this.faces.length) + 1);

      return faces[randNum-1];

};
```

y : Die

```
this.col = "#0000ff";
this.faces = [1,2,3,4,5,6];
```

```
this.randomRoll = function(){

      var randNum = Math.floor
      ( (Math.random() *
      this.faces.length) + 1);

      return faces[randNum-1];

};
```

FIGURE 15.1 Illustrating duplicated method definition

15.1.3 Using Prototypes

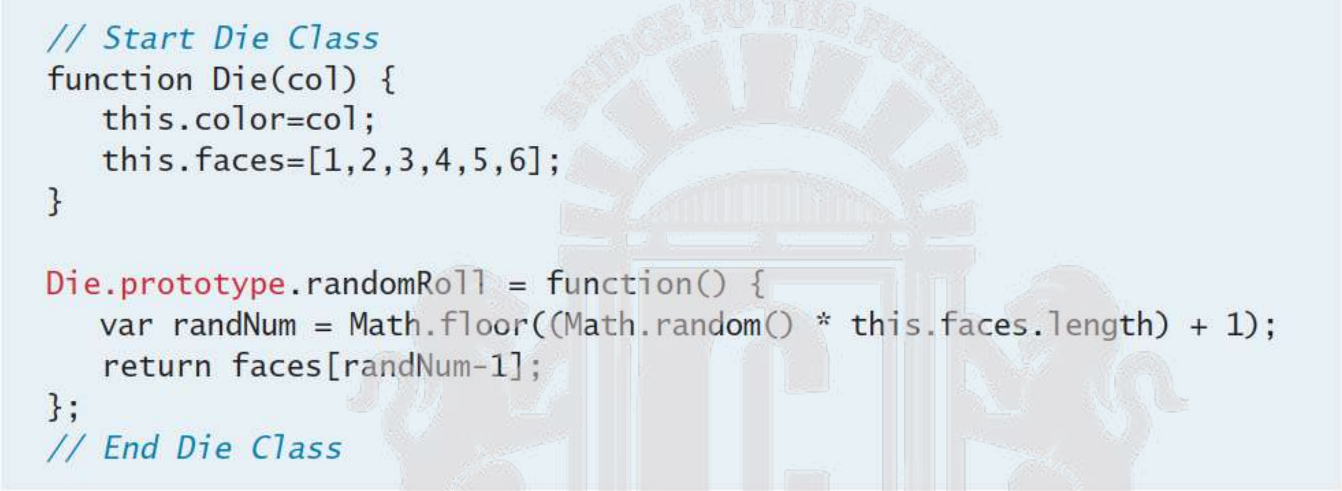
Prototypes

So you can use a *prototype* of the class.

- **Prototypes** are used to make JavaScript behave more like an object-oriented language.
- The prototype properties and methods are defined *once* for all instances of an *object*.
- Every object has a prototype

Using Prototypes

moving the randomRoll() function into the **prototype**.



```
// Start Die Class
function Die(col) {
   this.color=col;
   this.faces=[1,2,3,4,5,6];
}

Die.prototype.randomRoll = function() {
   var randNum = Math.floor((Math.random() * this.faces.length) + 1);
   return faces[randNum-1];
};
// End Die Class
```

LISTING 15.3 The Die pseudo-class using the prototype object to define methods

❖This definition is better because it defines the method only once, no matter how many instance of die are created.

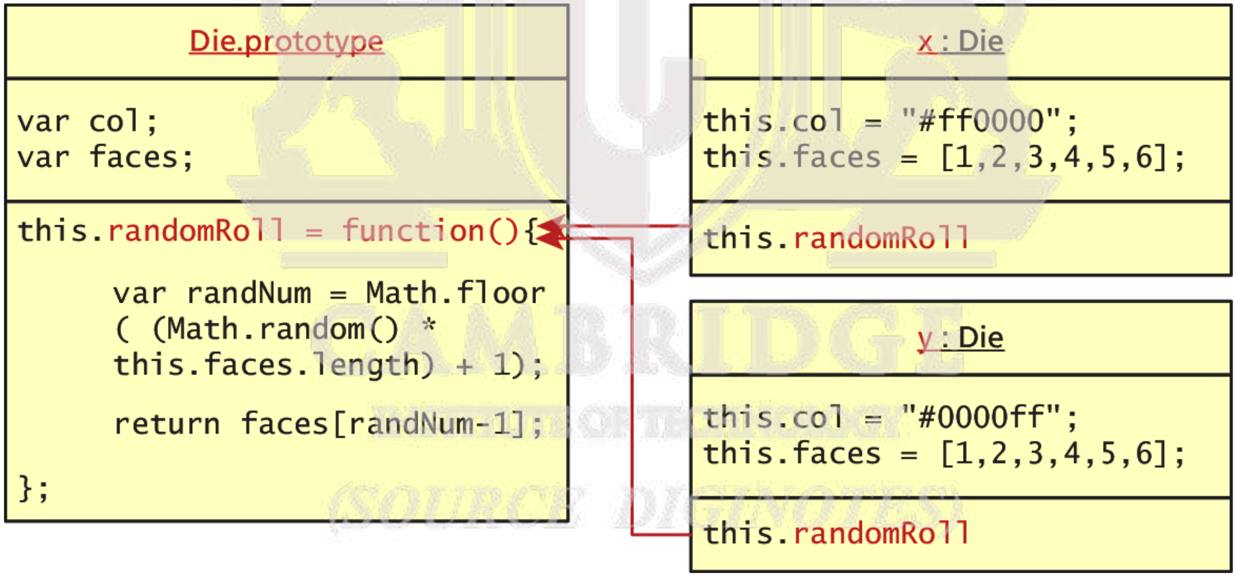
❖How many ever created it will be reference to that one method. (fig below)

Using Prototypes

No duplication of methods

Since all instances of a Die share the same prototype object, the function declaration only happens one time and is shared with all Die instances.

FIGURE 15.2 Illustration of JavaScript prototypes as pseudo-classes



```
var col;
var faces;

this.randomRoll = function(){

      var randNum = Math.floor
      ( (Math.random() *
      this.faces.length) + 1);

      return faces[randNum-1];

};
```

```
this.col = "#ff0000";
this.faces = [1,2,3,4,5,6];

this.randomRoll
```

```
this.col = "#0000ff";
this.faces = [1,2,3,4,5,6];

this.randomRoll
```

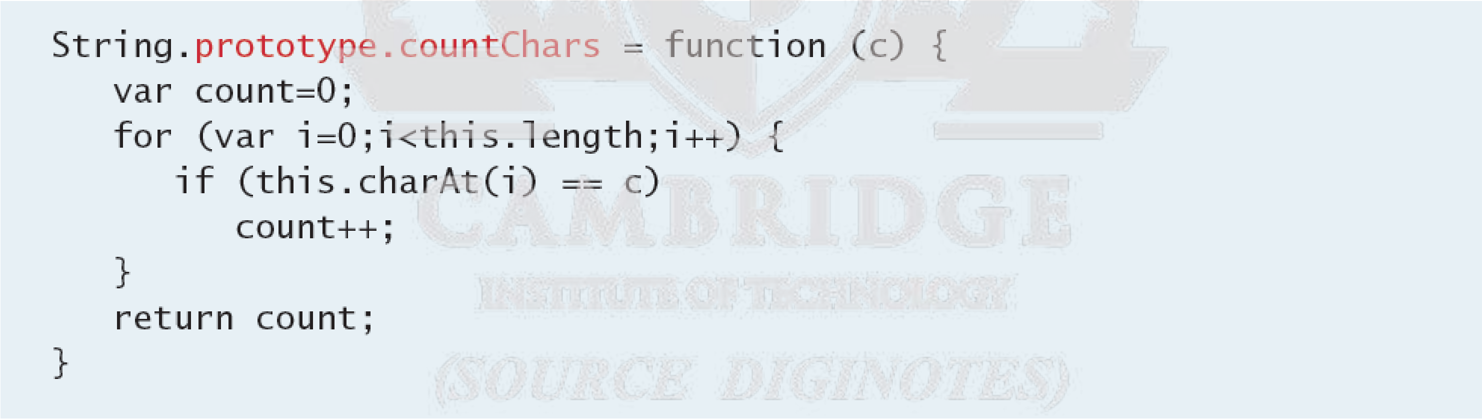
A prototype is an object from which other objects inherit.

More about Prototypes

Extend any Object

- Every object (and method) in JavaScript has a prototype.

- In addition to using prototypes for our own pseudo-classes, prototypes enable you to *extend* existing classes by adding to their prototypes



```
String.prototype.countChars = function (c) {
   var count=0;
   for (var i=0;i<this.length;i++) {
      if (this.charAt(i) == c)
         count++;
   }
   return count;
}
```

LISTING 15.4 Adding a method named countChars to the String class

Section 2 of 6

JQUERY FOUNDATIONS

jQuery

Framework

- ❑A **library** or **framework** is software that you can utilize in your own software, which provides some common implementations of standard ideas.
- ❑Many developers find that once they start using a framework like jQuery, there’s no going back to “pure” JavaScript because the framework offers so many useful shortcuts and breif ways of doing things

jQuery

A 1 slide history

In August 2005 jQuery founder John Resig was looking into how to better combine **CSS selectors with succinct JavaScript notation**.

- Within 1 year AJAX and animations were added
- Additional modules
  - jQuery UI extension
  - mobile device support
- Continues to improve.

jQuery

Not the only one, but a popular one

jQuery is now the most popular JavaScript library currently in use as supported by the statistics from BuiltWith.com

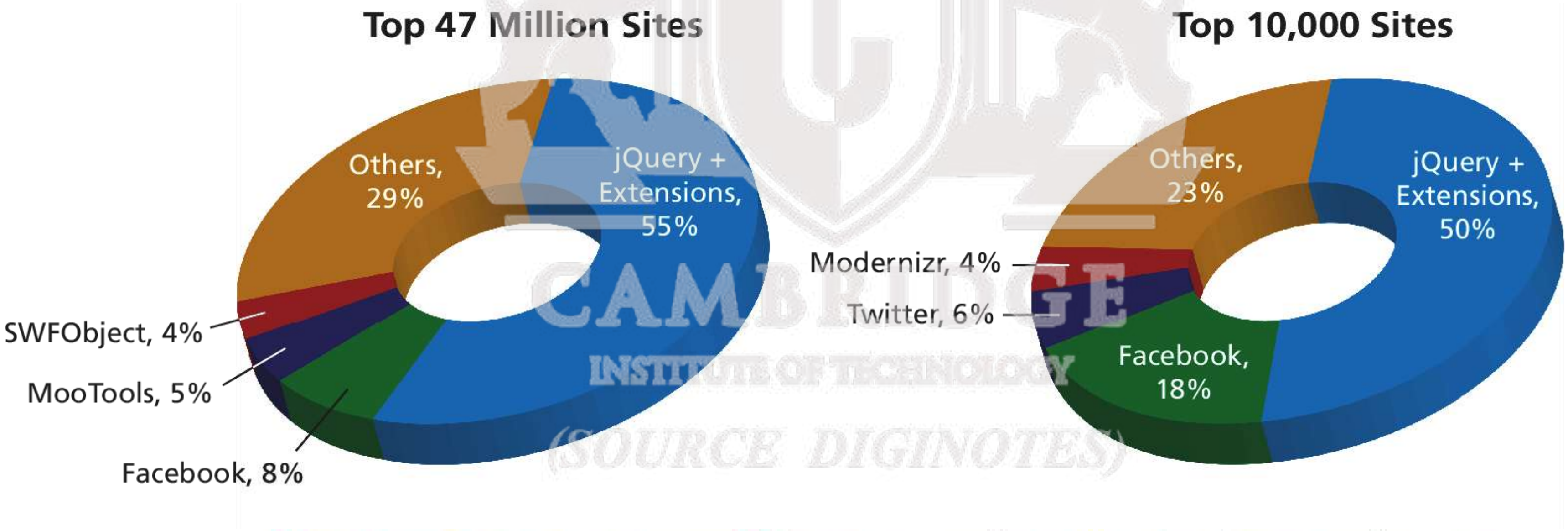


FIGURE 15.3 Comparison of the most popular JavaScript frameworks (data courtesy of BuiltWith.com)

Including jQuery

Let's get started

You must either:

- link to a locally hosted version of the library
- Use an approved third-party host, such as Google, Microsoft, or jQuery itself

Including jQuery

Content Delivery Network

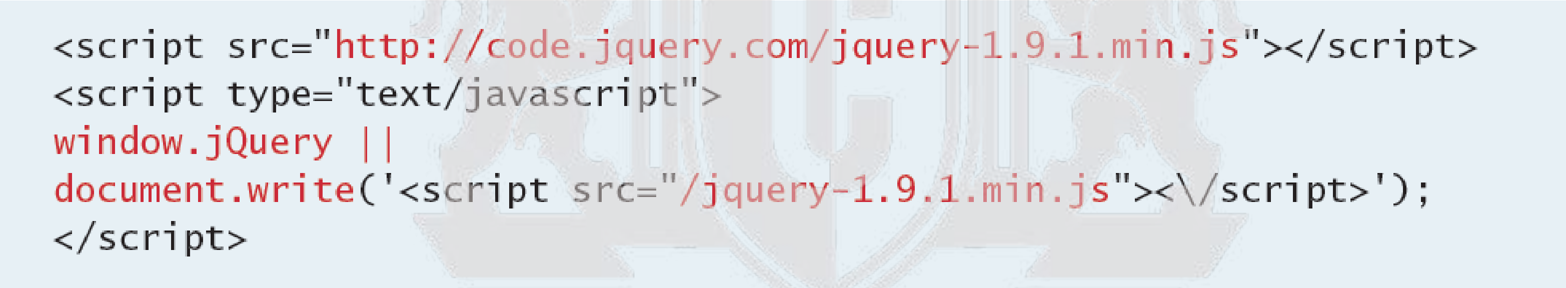
Using a third-party **content delivery network (CDN)** is advantageous for several reasons.

- The bandwidth of the file is offloaded to reduce the demand on your servers.
- The user may already have cached the third-party file and thus not have to download it again, thereby reducing the total loading time.

A disadvantage to the third-party CDN is that your jQuery will fail if the third-party host fails (unlikely but possible)

Including jQuery

Content Delivery Network and fallback



```
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script type="text/javascript">
window.jQuery ||
document.write('<script src="/jquery-1.9.1.min.js"><\/script>');
</script>
```

LISTING 15.5 jQuery loading using a CDN and a local fail-safe if the CDN is offline

jQuery Selectors

Should ring a bell

- When discussing basic JavaScript we introduced the **getElementByID()** and **querySelector()** selector functions in JavaScript.

- Although the advanced **querySelector()** methods allow selection of DOM elements based on CSS selectors, it is only implemented in newest browsers

- jQuery introduces its own way to select an element, which under the hood supports a myriad of older browsers for you!

jQuery Selectors

The easiest way to select an element yet

✓The relationship between DOM objects and selectors is so important in JavaScript programming that the pseudo-class bearing the name of the framework,

jQuery()

✓Is reserved for selecting elements from the DOM.

✓Because it is used so frequently, it has a shortcut notation and can be written as

$()

Basic Selectors

All the way back to CSS

The four basic selectors are:

- $("*") **Universal selector** matches all elements (and is slow).
- $("tag") **Element selector** matches all elements with the given element name.
- $(".class") **Class selector** matches all elements with the given CSS class.
- $("#id") **Id selector** matches all elements with a given HTML id attribute.

Basic Selectors

All the way back to CSS

For example, to select the single <div> element with id="grab" you would write:

var singleElement = $("#grab");

To get a set of all the <a> elements the selector would be:

var allAs = $("a");

These selectors replace the use of getElementById() entirely.

More CSS Selectors

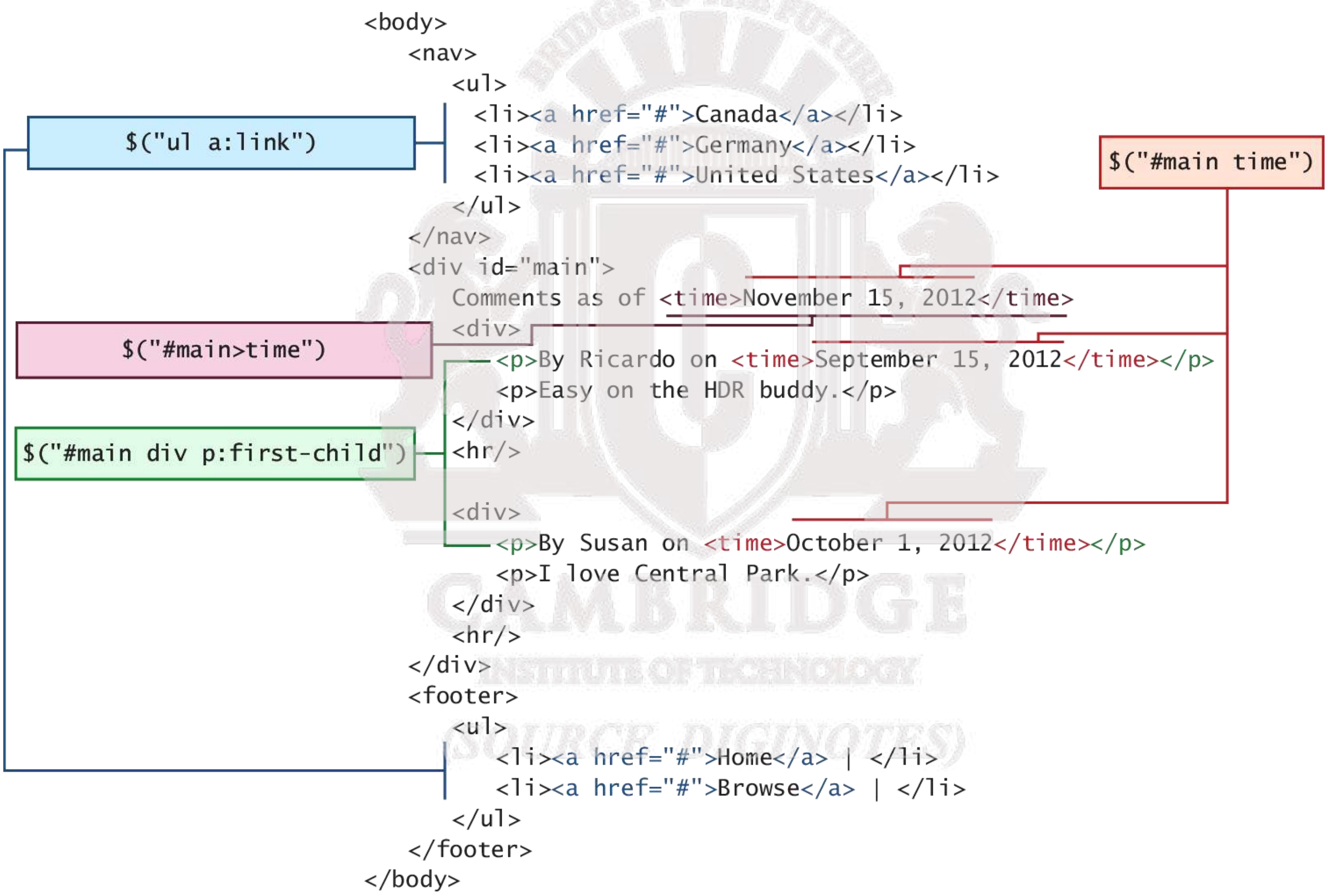
jQuery's selectors are powerful indeed

In addition to these basic selectors, you can use the other CSS selectors that were covered in Chapter 3 on CSS:

- attribute selectors,
- pseudo-element selectors, and
- contextual selectors

More CSS Selectors

jQuery's selectors are powerful indeed



```
<body>
   <nav>
      <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
      </ul>
   </nav>
   <div id="main">
      Comments as of <time>November 15, 2012</time>
      <div>
         <p>By Ricardo on <time>September 15, 2012</time></p>
         <p>Easy on the HDR buddy.</p>
      </div>
      <hr/>

      <div>
         <p>By Susan on <time>October 1, 2012</time></p>
         <p>I love Central Park.</p>
      </div>
      <hr/>
   </div>
   <footer>
      <ul>
         <li><a href="#">Home</a> | </li>
         <li><a href="#">Browse</a> | </li>
      </ul>
   </footer>
</body>
```

Attribute Selector

Really a review of CSS

Recall from CSS that you can select

- by attribute with square brackets
  - [attribute]
- Specify a value with an equals sign
  - [attribute=value]
- Search for a particular value in the beginning, end, or anywhere inside a string
  - [attribute^=value]
  - [attribute$=value]
  - [attribute*=value]

Attribute Selector

Really a review of CSS

Consider a selector to grab all <img> elements with an src attribute beginning with */artist/* as:

```
var artistImages = $("img[src^='/artist/']");
```

Pseudo-Element Selector

Not to be confused with the pseudo-classes in JavaScript

pseudo-element selectors are also from CSS and allow you to append to any selector using the colon and one of

- :link
- :visited
- :focus
- :hover
- :active
- :checked
- :first-child, :first-line, and :first-letter

Pseudo-Element Selector

Not to be confused with the pseudo-classes in JavaScript

Selecting all links that have been visited, for example, would be specified with:

```
var visitedLinks = $("a:visited");
```

Contextual Selector

Put it into context

Contextual selectors are also from CSS. Recall that these include:

- descendant (space)
- child (>)
- adjacent sibling (+)
- and general sibling (~).

To select all <p> elements inside of <div> elements you would write

var para = $("div p");

Content Filters

Above and Beyond CSS

The **content filter** is the only jQuery selector that allows you to append filters to all of the selectors you've used thus far and match a particular pattern.

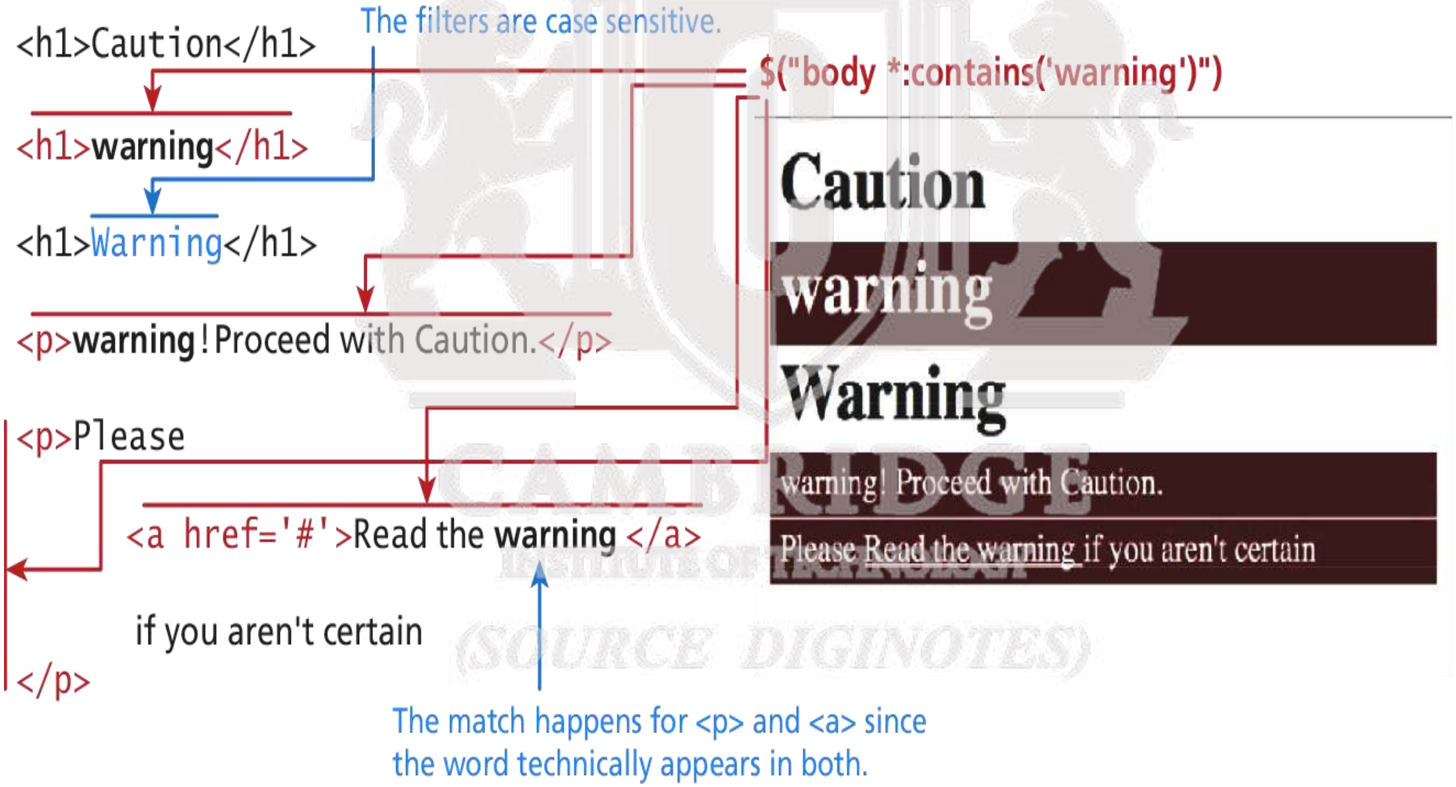
Select elements that have:

- a particular child using :has()
- have no children using :empty
- match a particular piece of text with :contains()

Content Filters

Above and Beyond CSS

$("body *:contains('warning')”)



```
<h1>Caution</h1>

<h1>warning</h1>

<h1>Warning</h1>

<p>warning!Proceed with Caution.</p>

<p>Please
  <a href='#'>Read the warning </a>

  if you aren't certain
</p>
```

The filters are case sensitive.

$("body *:contains('warning')")

The match happens for <p> and <a> since the word technically appears in both.

Caution
warning
Warning
warning! Proceed with Caution.
Please Read the warning if you aren't certain

Form Selectors

Selector	CSS Equivalent	Description
$(:button)	$("button, input[type='button']")	Selects all buttons
$(:checkbox)	$('[type=checkbox]‘)	Selects all checkboxes
$(:checked)	No Equivalent	Selects elements that are checked. This includes radio buttons and checkboxes.
$(:disabled)	No Equivalent	Selects form elements that are disabled.
$(:enabled)	No Equivalent	Opposite of :disabled
$(:file)	$('[type=file]')	Selects all elements of type file
$(:focus)	$(document.activeElement)	The element with focus
$(:image)	$('[type=image]')	Selects all elements of type image
$(:input)	No Equivalent	Selects all <input>, <textarea>, <select>, and <button> elements.
$(:password)	$('[type=password]')	Selects all password fields
$(:radio)	$('[type=radio]')	Selects all radio elements
$(:reset)	$('[type=reset]')	Selects all the reset buttons
$(:selected)	No Equivalent	Selects all the elements that are currently selected of type <option>. It does not include checkboxes or radio buttons.
$(:submit)	$('[type=submit]')	Selects all submit input elements
$(:text)	No Equivalent	Selects all input elements of type text. $('[type=text]') is almost the same, except that $(:text) includes <input> fields with no type specified.

jQuery Attributes

Back to HTML now.

In order to understand how to fully manipulate the elements you now have access to, one must understand an element's *attributes* and *properties*.

The core set of attributes related to DOM elements are the ones specified in the HTML tags.

- The *href* attribute of an <a> tag
- The *src* attribute of an <img>
- The *class* attribute of most elements

jQuery Attributes

And some examples

In jQuery we can both set and get an attribute value by using the **attr()** method on any element from a selector.

```
// var link is assigned the href attribute of the first <a> tag
var link = $("a").attr("href");

// change all links in the page to http://funwebdev.com
$("a").attr("href","http://funwebdev.com");

// change the class for all images on the page to fancy
$("img").attr("class","fancy");
```

HTML Properties

Full circle

- Many HTML tags include *properties* as well as attributes, the most common being the *checked* property of a radio button or checkbox.

- The prop() method is the preferred way to retrieve and set the value of a property although, attr() may return some (less useful) values.

- <input **class ="meh"** type="checkbox" checked="checked">

- Is accessed by jQuery as follows:

- **var theBox = $(".meh");**
theBox.prop("checked"); *// evaluates to TRUE*
theBox.attr("checked"); *// evaluates to "checked"*

Changing CSS

With jQuery

- jQuery provides the extremely intuitive **css()** methods.
- To get a css value use the css() method with 1 parameter:

```
$color = $("#colourBox").css("background-color"); // get the color
```

- To set a CSS variable use css() with two parameters: the first being the CSS attribute, and the second the value.

```
        // set color to red
            $("#colourBox").css("background-color", "#FF0000");
```

Shortcut Methods

With jQuery

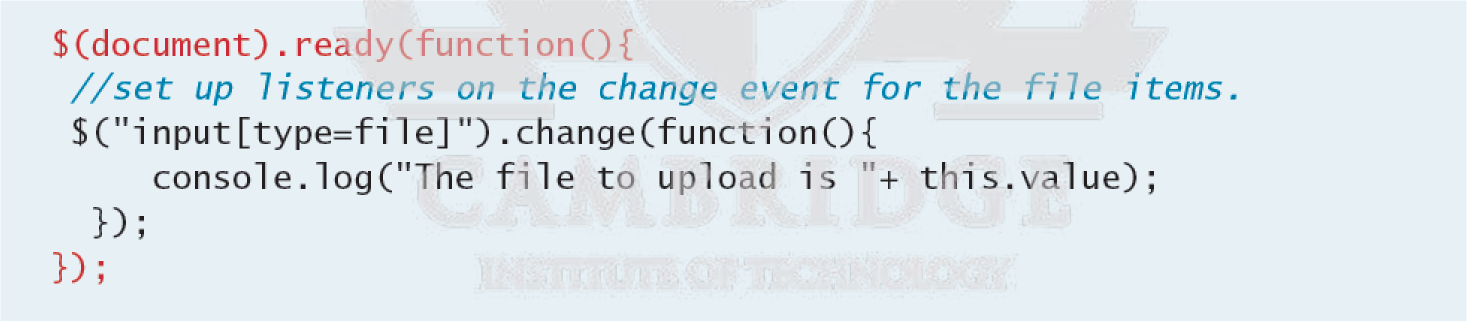
- The **html()** method is used to get the HTML contents of an element. If passed with a parameter, it updates the HTML of that element.
- The **val()** method returns the value of the element.
- The shortcut methods **addClass**(className) / **removeClass**(className) add or remove a CSS class to the element being worked on. The className used for these functions can contain a space-separated list of classnames to be added or removed.
- The **hasClass**(classname) method returns true if the element has the className currently assigned. False, otherwise.

jQuery Listeners

Set up after page load

In JavaScript, you learned why having your **listeners** set up inside of the window.onload() event was a good practice.

With jQuery we do the same thing but use the $(document).ready() event



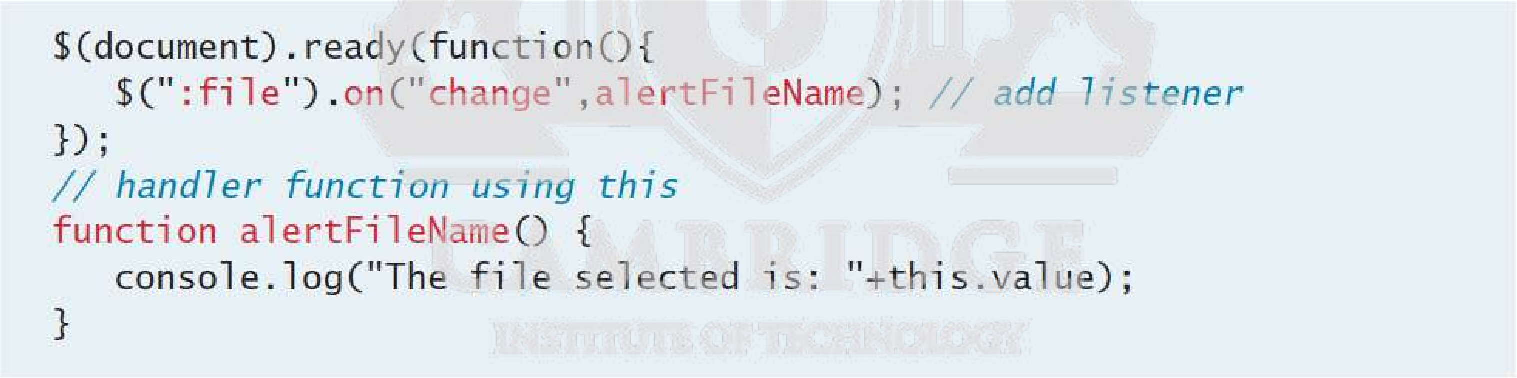
```
$(document).ready(function(){
 //set up listeners on the change event for the file items.
 $("input[type=file]").change(function(){
     console.log("The file to upload is "+ this.value);
  });
});
```

LISTING 15.6 jQuery code to listen for file inputs changing, all inside the document's ready event

jQuery Listeners

Listener Management

While pure JavaScript uses the addEventListener() method, jQuery has on() and off() methods as well as shortcut methods to attach events.



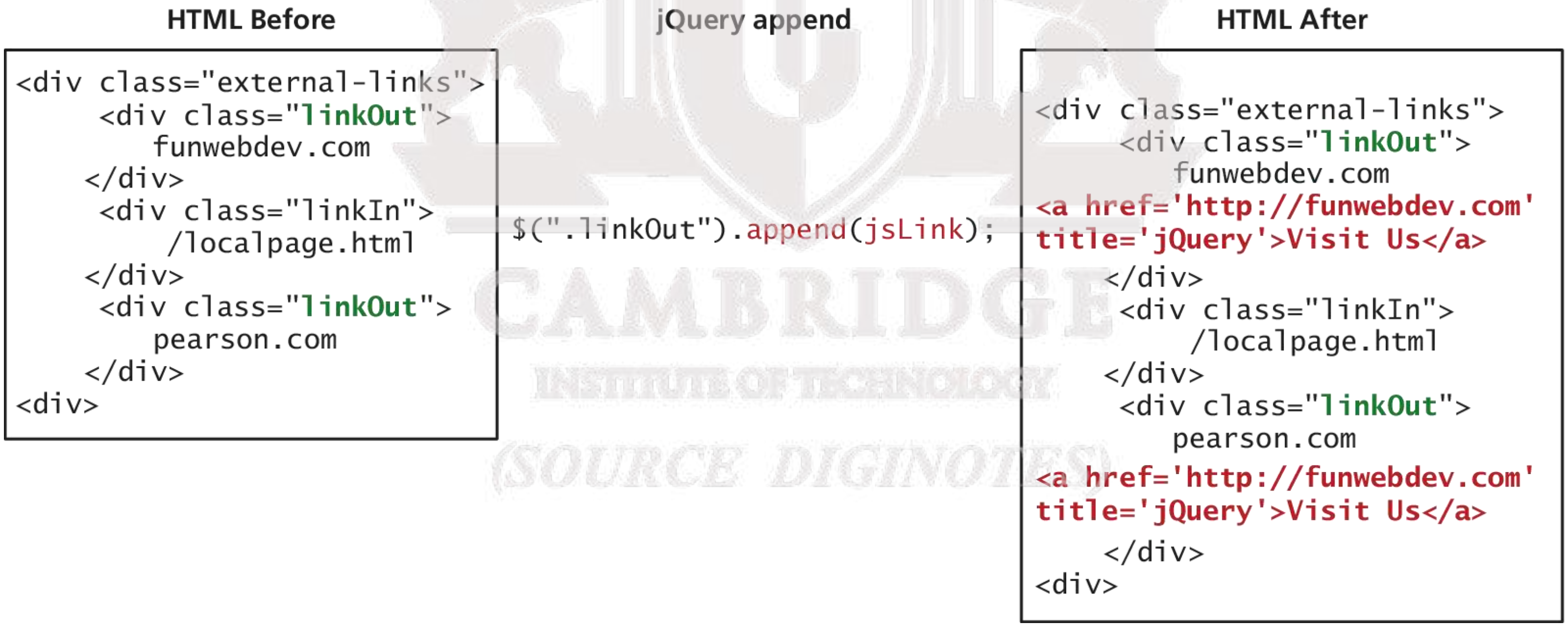
```
$(document).ready(function(){
   $(":file").on("change",alertFileName); // add listener
});
// handler function using this
function alertFileName() {
   console.log("The file selected is: "+this.value);
}
```

LISTING 15.7 Using the listener technique in jQuery with on and off methods

Modifying the DOM

Appending DOM Elements

•The append() method takes as a parameter an HTML string, a DOM object, or a jQuery object. That object is then added as the last child to the element(s) being selected.



HTML Before

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
    </div>
     <div class="linkIn">
         /localpage.html
    </div>
     <div class="linkOut">
        pearson.com
    </div>
<div>
```

jQuery append

```
$(".linkOut").append(jsLink);
```

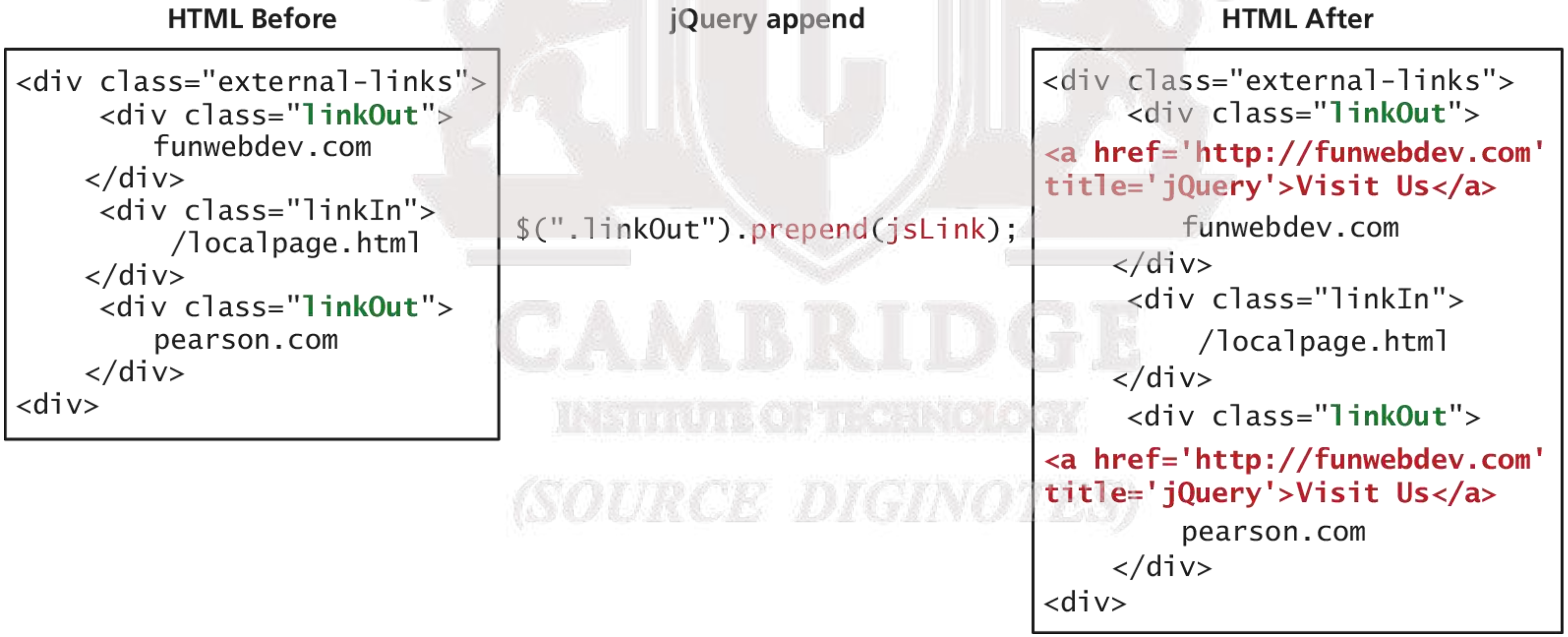
HTML After

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
    </div>
     <div class="linkIn">
         /localpage.html
    </div>
     <div class="linkOut">
        pearson.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
    </div>
<div>
```

Modifying the DOM

Prepending DOM Elements

The prepend() and prependTo() methods operate in a similar manner except that they add the new element as the first child rather than the last.



HTML Before

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
    </div>
     <div class="linkIn">
        /localpage.html
    </div>
     <div class="linkOut">
        pearson.com
    </div>
<div>
```

jQuery append

```
$(".linkOut").prepend(jsLink);
```

HTML After

```
<div class="external-links">
     <div class="linkOut">
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
        funwebdev.com
    </div>
     <div class="linkIn">
        /localpage.html
    </div>
     <div class="linkOut">
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
        pearson.com
    </div>
<div>
```

Modifying the DOM

Wrapping Existing DOM in New Tags

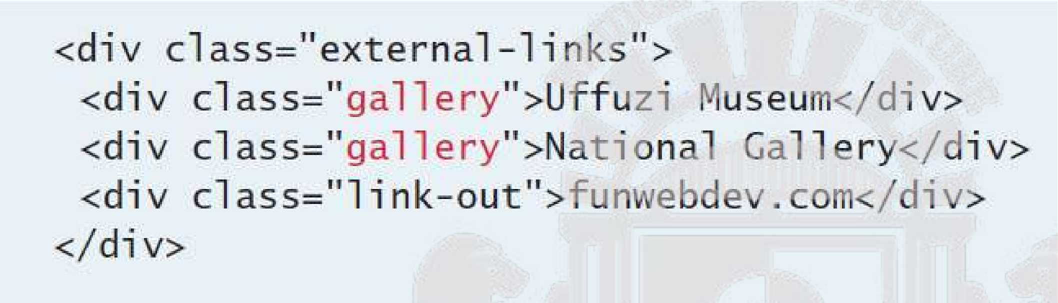
❖A more advanced technique might make use of the content of each div being modified. In that case we use a callback function in place of a simple element.

❖The wrap() method is a callback function, which is called for each element in a set (often an array).

❖Each element then becomes this for the duration of one of the wrap() function’s executions, allowing the unique title attributes as shown in Listing 15.12.

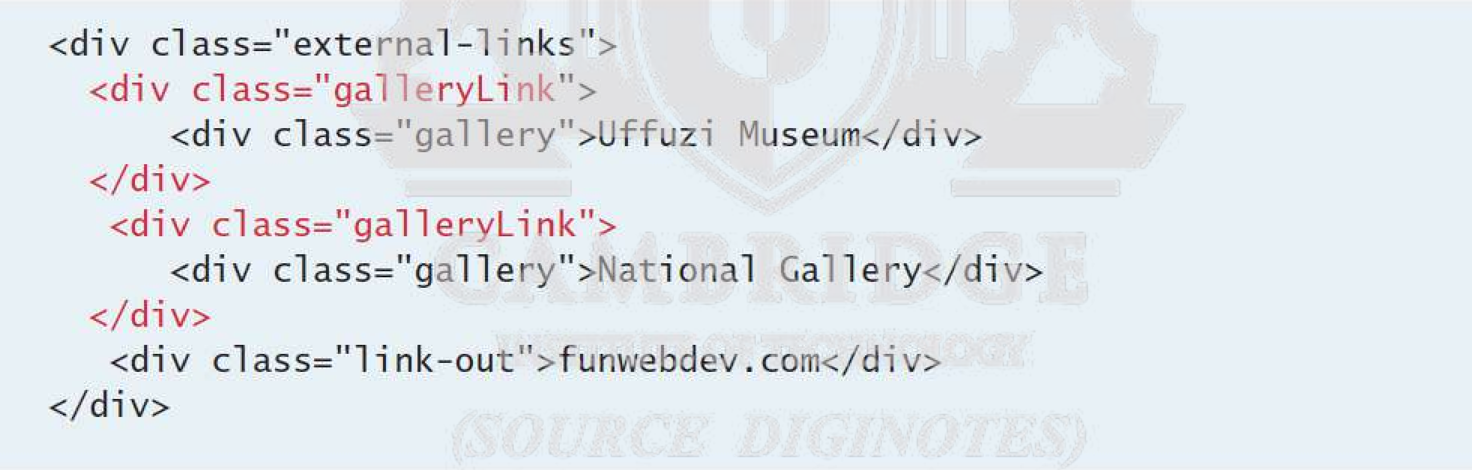
Modifying the DOM

Wrapping Existing DOM in New Tags



```
<div class="external-links">
 <div class="gallery">Uffuzi Museum</div>
 <div class="gallery">National Gallery</div>
 <div class="link-out">funwebdev.com</div>
</div>
```

```
$(".gallery").wrap('<div class="galleryLink"/>');
```



```
<div class="external-links">
  <div class="galleryLink">
      <div class="gallery">Uffuzi Museum</div>
  </div>
   <div class="galleryLink">
      <div class="gallery">National Gallery</div>
  </div>
   <div class="link-out">funwebdev.com</div>
</div>
```

LISTING 15.10 HTML from Listing 15.9 modified by executing the wrap statement above

Section 3 of 6

AJAX

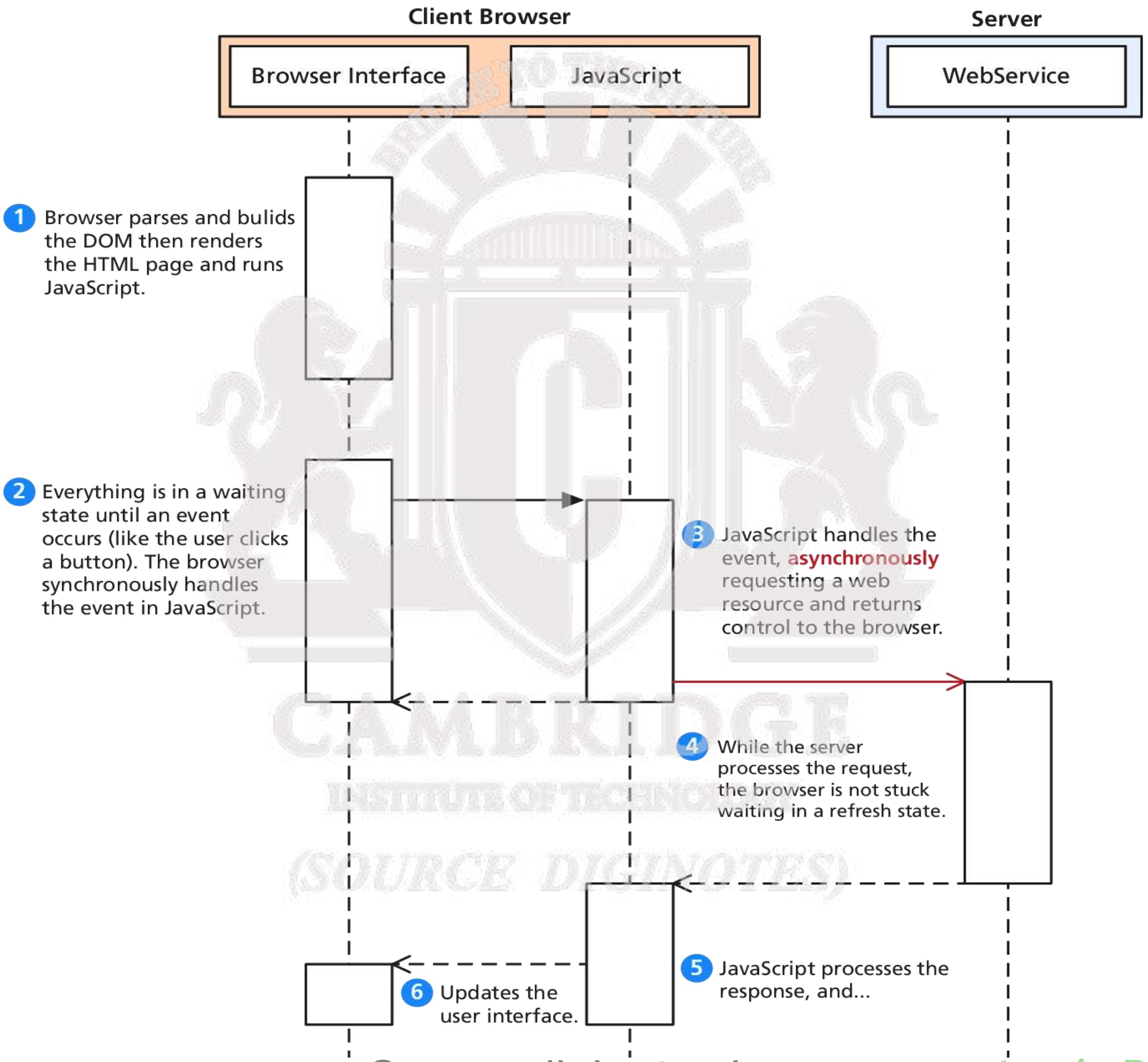
AJAX

Asynchronous JavaScript with XML

❑Asynchronous JavaScript with XML (AJAX) is a term used to describe a paradigm that allows a web browser to send messages back to the server without interrupting the flow of what’s being shown in the browser.

AJAX

UML of an asynchronous request



1 Browser parses and bulids the DOM then renders the HTML page and runs JavaScript.

2 Everything is in a waiting state until an event occurs (like the user clicks a button). The browser synchronously handles the event in JavaScript.

3 JavaScript handles the event, **asynchronously** requesting a web resource and returns control to the browser.

4 While the server processes the request, the browser is not stuck waiting in a refresh state.

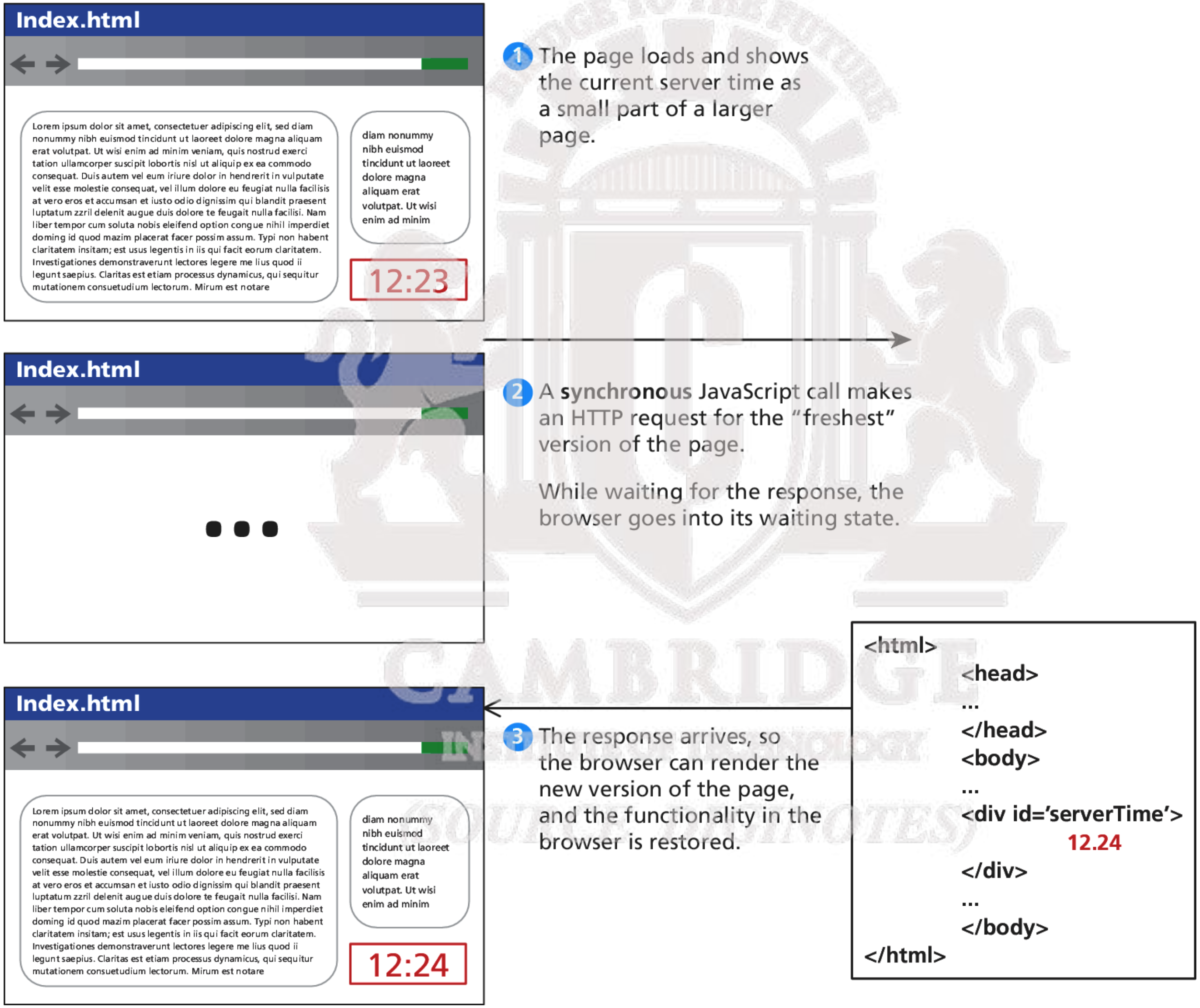
5 JavaScript processes the response, and...

6 Updates the user interface.

Source diginotes.in

AJAX

Consider a webpage that displays the server's time


Index.html
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
12:23
Index.html
Index.html
12:24

1 The page loads and shows the current server time as a small part of a larger page.

2 A synchronous JavaScript call makes an HTTP request for the "freshest" version of the page.

While waiting for the response, the browser goes into its waiting state.

3 The response arrives, so the browser can render the new version of the page, and the functionality in the browser is restored.

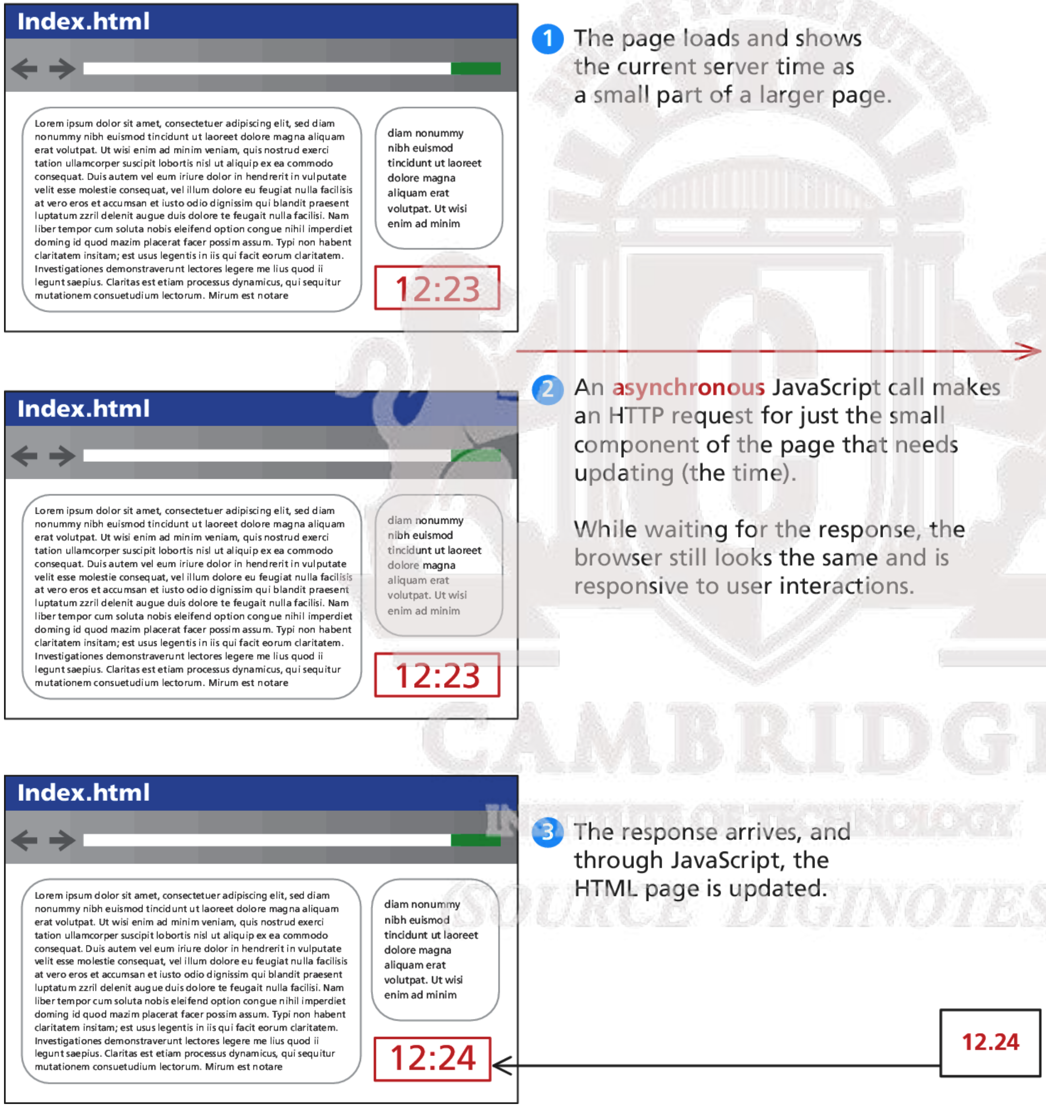
```
<html>
          <head>
          ...
          </head>
          <body>
          ...
          <div id='serverTime'>
                    12.24
          </div>
          ...
          </body>
</html>
```

Source diginotes.in

Save the Earth. Go paperless

AJAX

Consider a webpage that displays the server's time


Index.html
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
12:23
Index.html
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
12:23
Index.html
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
12:24
12.24

1. The page loads and shows the current server time as a small part of a larger page.

2. An asynchronous JavaScript call makes an HTTP request for just the small component of the page that needs updating (the time).

   While waiting for the response, the browser still looks the same and is responsive to user interactions.

3. The response arrives, and through JavaScript, the HTML page is updated.

AJAX

Making Asynchronous requests

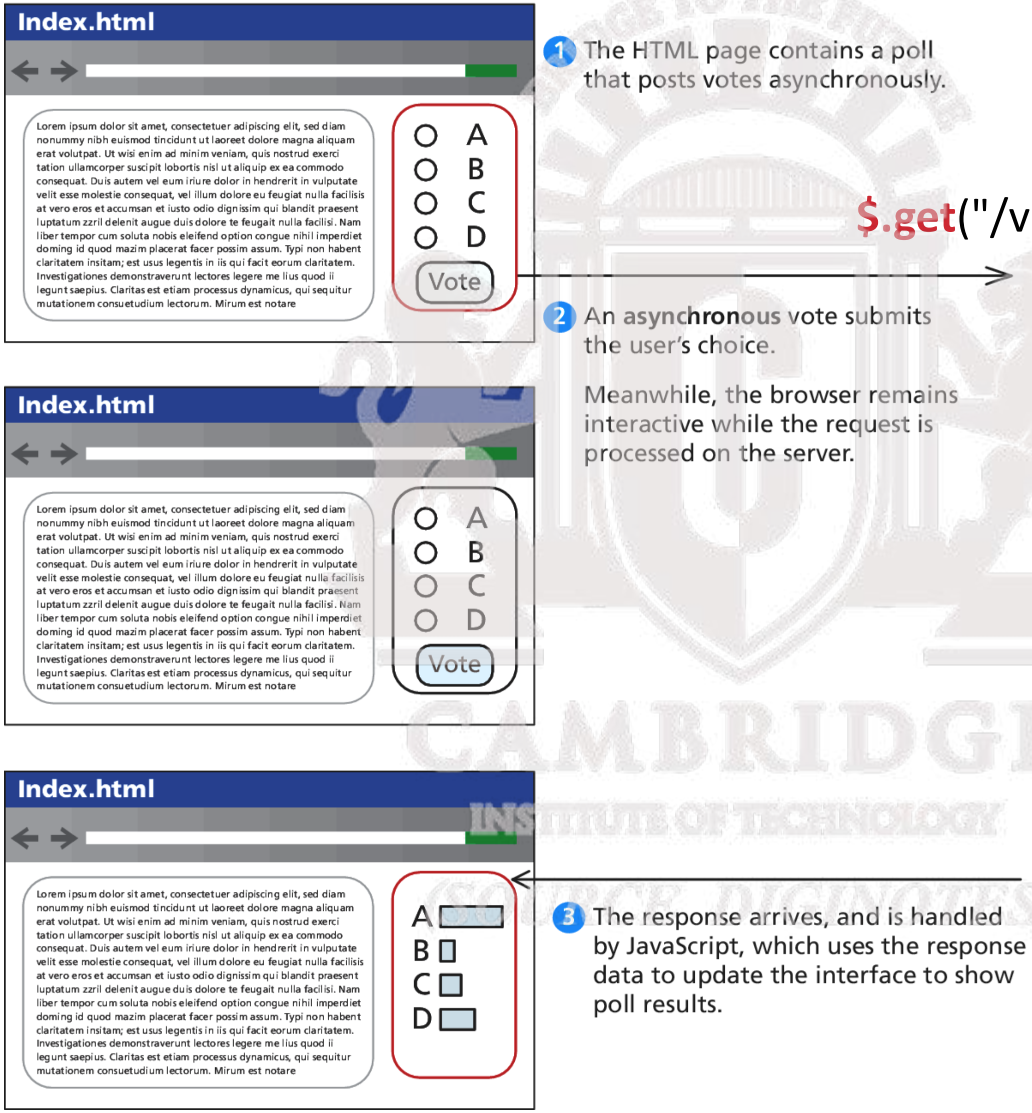
- jQuery provides a family of methods to make asynchronous requests. We will start simple and work our way up.

- Consider the very simple server time example we just saw. If currentTime.php returns a single string and you want to load that value asynchronously into the <div id="timeDiv"> element, you could write:

- **$("#timeDiv").load("currentTime.php");**

AJAX

GET Requests



1 The HTML page contains a poll that posts votes asynchronously.

```
$.get("/vote.php?option=C");
```

2 An asynchronous vote submits the user’s choice.

Meanwhile, the browser remains interactive while the request is processed on the server.

3 The response arrives, and is handled by JavaScript, which uses the response data to update the interface to show poll results.

AJAX

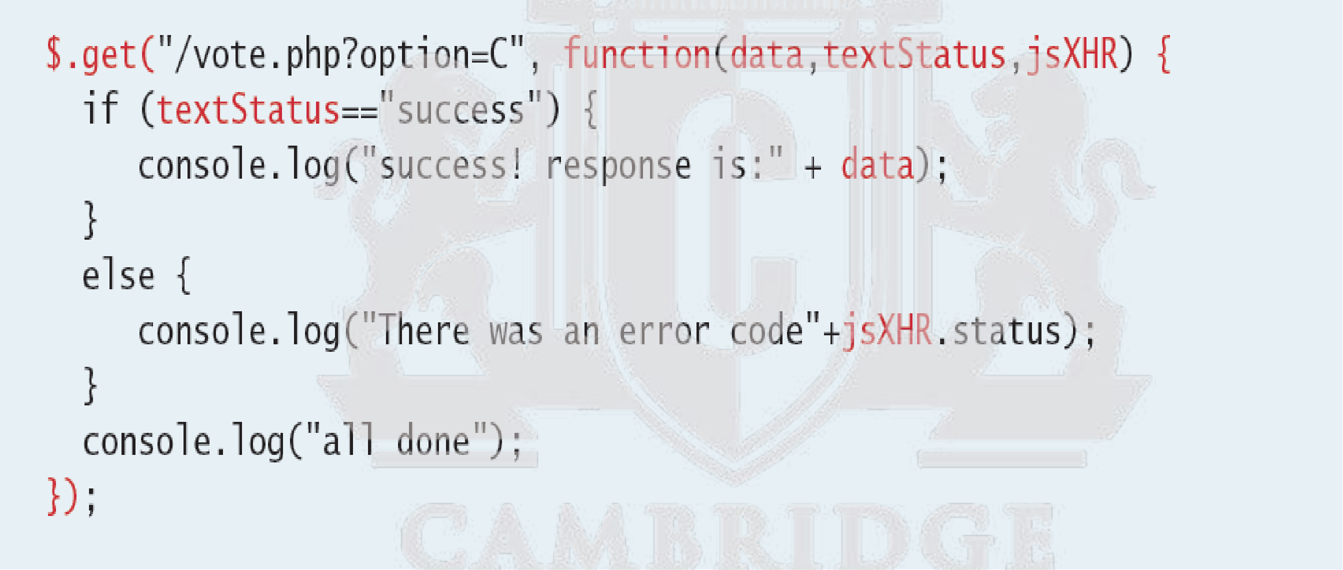
GET Requests – formal definition

jQuery.get (**url [, data] [, success(data, textStatus, jqXHR)] [, dataType]**)

- **url** is a string that holds the location to send the request.
- **data** is an optional parameter that is a query string or a *Plain Object.*
- **success(data,textStatus,jqXHR)** is an optional *callback* function that executes when the response is received.
  - **data** holding the body of the response as a string.
  - **textStatus** holding the status of the request (i.e., “success”).
  - **jqXHR** holding a jqXHR object, described shortly.
- **dataType** is an optional parameter to hold the type of data expected from the server.

AJAX

GET Requests – an example



```
$.get("/vote.php?option=C", function(data,textStatus,jsXHR) {
  if (textStatus=="success") {
     console.log("success! response is:" + data);
  }
  else {
     console.log("There was an error code"+jsXHR.status);
  }
  console.log("all done");
});
```

LISTING 15.13 jQuery to asynchronously get a URL and outputs when the response arrives

AJAX

The jqXHR Object

•All of the $.get() requests made by jQuery return a **jqXHR** object to encapsulate the response from the server.

•In practice that means the data being referred to in the callback from Listing 15.13 is actually an object with backward compatibility with XML HttpRequest.

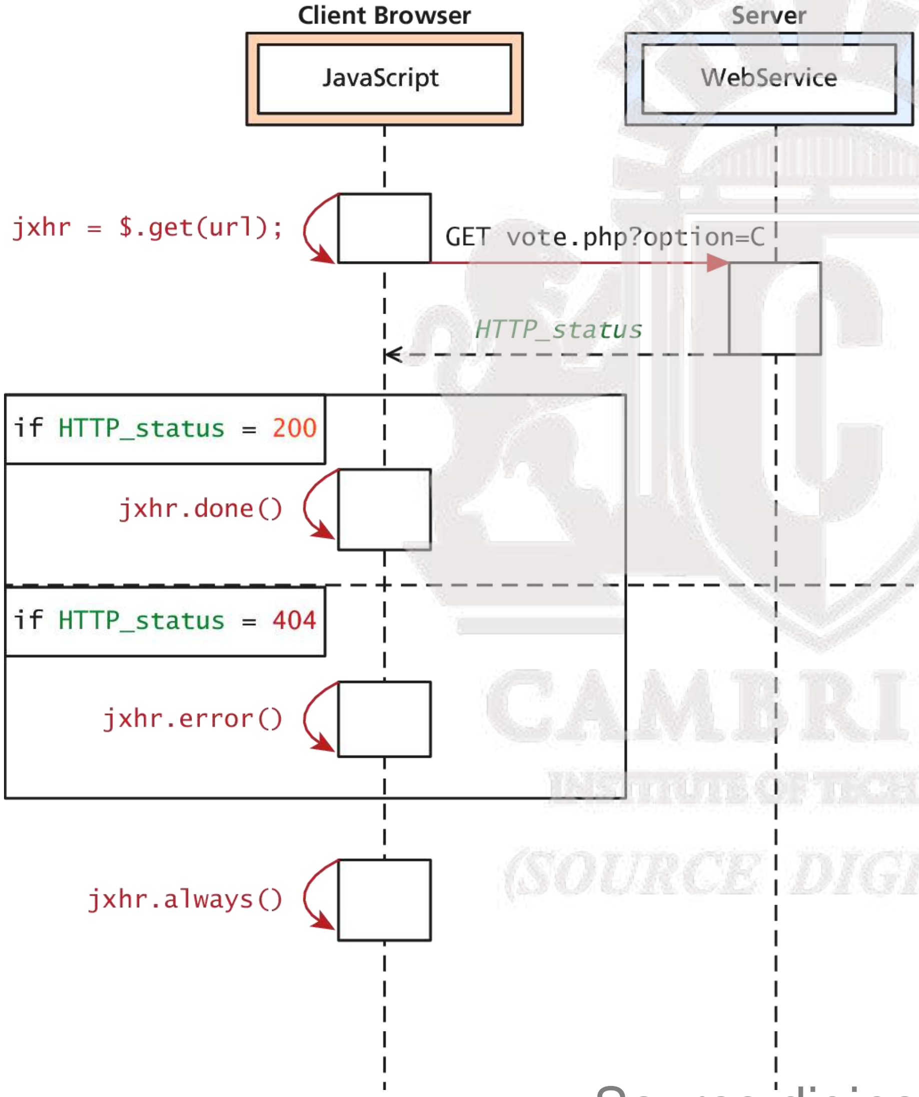
AAJAX

jqXHR - XMLHttpRequest compatibility

- **abort()** stops execution and prevents any callback or handlers from receiving the trigger to execute.
- **getResponseHeader()** takes a parameter and gets the current value of that header.
- **readyState** is an integer from 1 to 4 representing the state of the request. The values include 1:successful call open() 2: sending, 3: response being processed, and 4: completed.
- **responseXML** and/or **responseText** the main response to the request.
- **setRequestHeader(name, value)** when used before actually instantiating the request allows headers to be changed for the request.
- **status** is the HTTP request status codes (200 = ok)
- **statusText** is the associated description of the status code.

jqXHR

Actually quite easy to use



jqXHR objects have methods

- done()
- fail()
- always()

which allow us to structure our code in a more modular way than the inline callback

Source diginotes.in

Save the Earth. Go paperless

POST requests

Via jQuery AJAX

- POST requests are often preferred to GET requests because one can post an unlimited amount of data, and because they do not generate viewable URLs for each action.

- GET requests are typically not used when we have forms because of the messy URLs and that limitation on how much data we can transmit.

- With POST it is possible to transmit files, something which is not possible with GET.

POST requests

Via jQuery AJAX

- The HTTP 1.1 definition describes GET as a **safe method** meaning that they should not change anything, and should only read data.

- POSTs on the other hand are not safe, and should be used whenever we are changing the state of our system (like casting a vote). get() method.

- POST syntax is almost identical to GET.

- jQuery.**post** (url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

POST requests

Via jQuery AJAX

If we were to convert our vote casting code it would simply change the first line from

```
var jqxhr = $.get("/vote.php?option=C");
```

to

```
var jqxhr = $.post("/vote.php", "option=C");
```

POST requests

Serialize() will seriously help

serialize() can be called on any form object to return its current key-value pairing as an & separated string, suitable for use with post().

```
var postData = $("#voteForm").serialize();

$.post("vote.php", postData);
```

Ajax

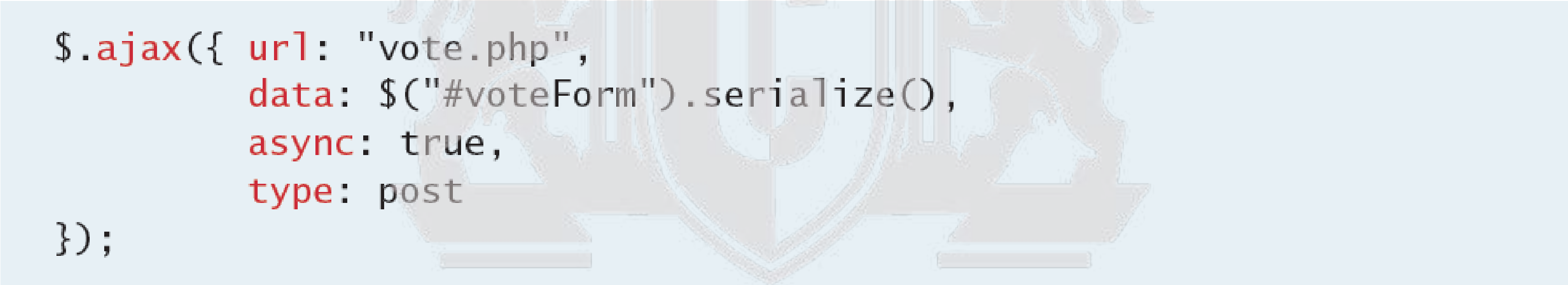
You have complete control

- It turns out both the $.get() and $.post() methods are actually shorthand forms for the jQuery().ajax() method
- The ajax() method has two versions. In the first it takes two parameters: a URL and a Plain Object, containing any of over 30 fields.
- A second version with only one parameter is more commonly used, where the URL is but one of the key-value pairs in the Plain Object.

Ajax

More verbose

The one line required to post our form using get() becomes the more verbose code



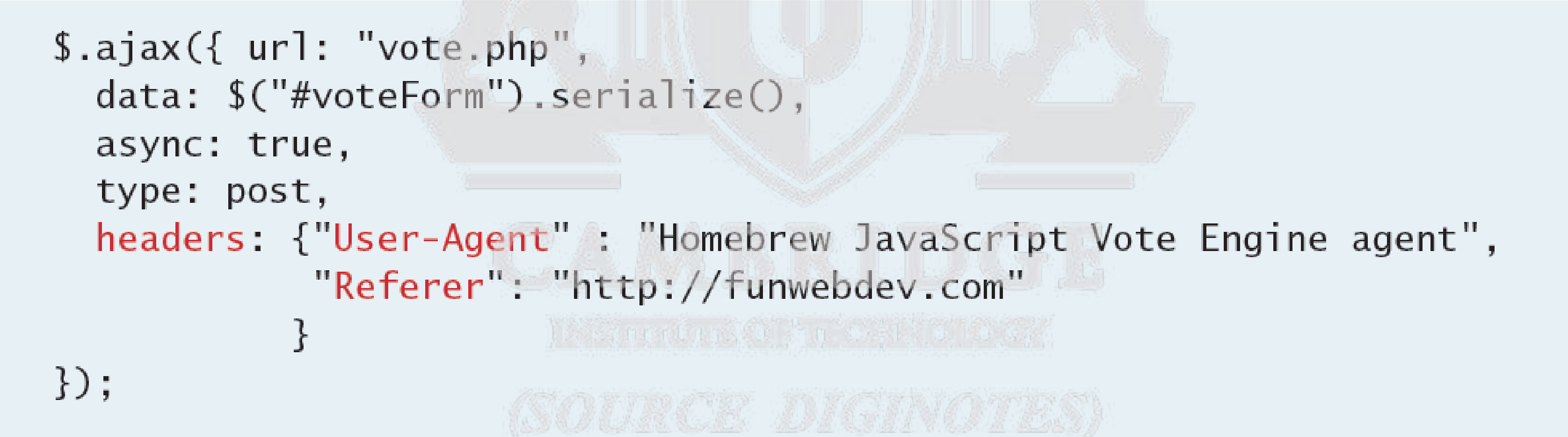
```
$.ajax({ url: "vote.php",
         data: $("#voteForm").serialize(),
         async: true,
         type: post
});
```

LISTING 15.15 A raw AJAX method code to make a post

Ajax

You have complete control

To pass HTTP headers to the ajax() method, you enclose as many as you would like in a Plain Object. To illustrate how you could override User-Agent and Referer headers in the POST



```
$.ajax({ url: "vote.php",
  data: $("#voteForm").serialize(),
  async: true,
  type: post,
  headers: {"User-Agent" : "Homebrew JavaScript Vote Engine agent",
             "Referer": "http://funwebdev.com"
           }
});
```

LISTING 15.16 Adding headers to an AJAX post in jQuery

CORS

Cross Origin Resource Sharing

❑Since modern browsers prevent cross-origin requests by default (which is good for security), sharing content legitimately between two domains becomes harder.

❑**Cross-origin resource sharing (CORS)** uses new headers in the HTML5 standard implemented in most new browsers.

❑If a site wants to allow any domain to access its content through JavaScript, it would add the following header to all of its responses.

❑**Access-Control-Allow-Origin: ***

CORS

Cross Origin Resource Sharing

✓A better usage is to specify specific domains that are allowed, rather than cast the gates open to each and every domain. To allow our domain to make cross site requests we would add the header:

✓Access-Control-Allow-Origin: www.funwebdev.com

✓Rather than the wildcard *.

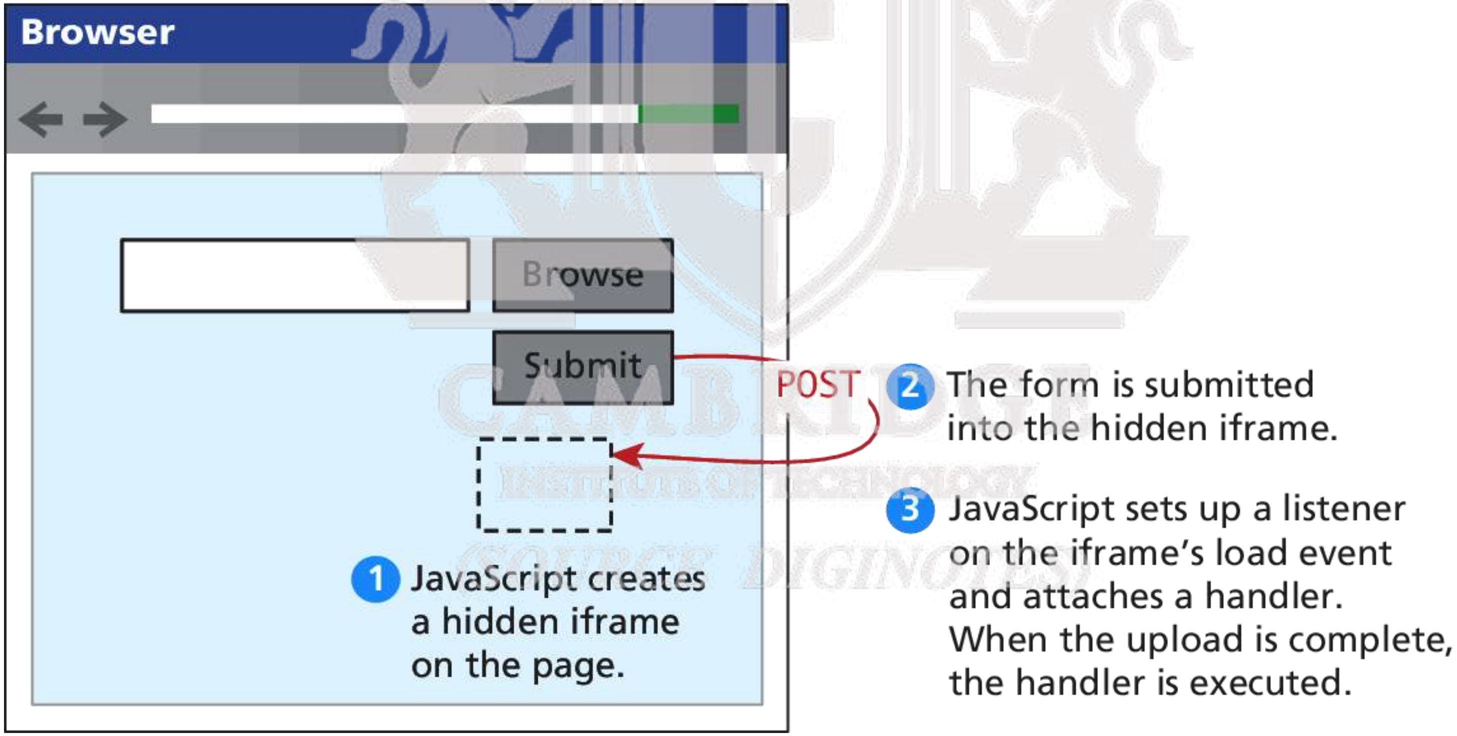
Section 4 of 6

ASYNCHRONOUS FILE TRANSMISSION

Asynchronous File Transmission

The old iFrame technique

The original workaround to allow the asynchronous posting of files was to use a hidden <iframe> element to receive the posted files.



1 JavaScript creates a hidden iframe on the page.

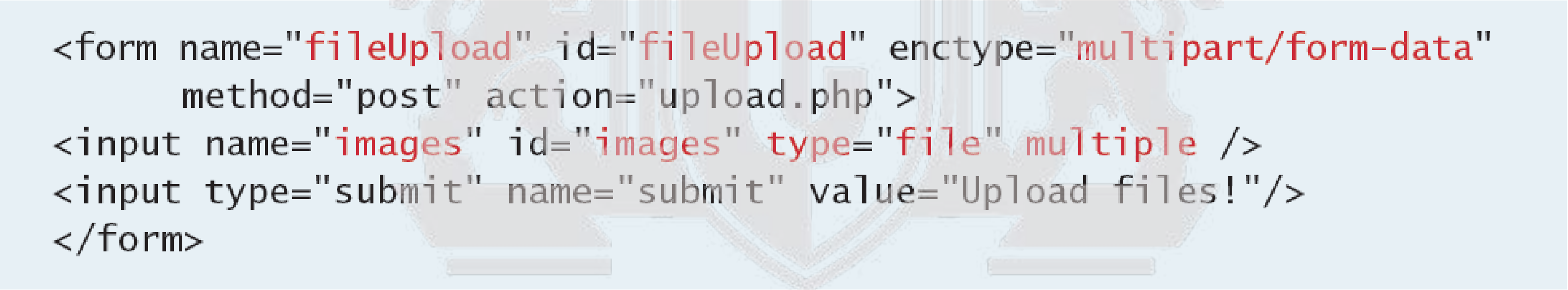
2 The form is submitted into the hidden iframe.

3 JavaScript sets up a listener on the iframe's load event and attaches a handler. When the upload is complete, the handler is executed.

Asynchronous File Transmission

A Primer

Consider a simple form as defined below:

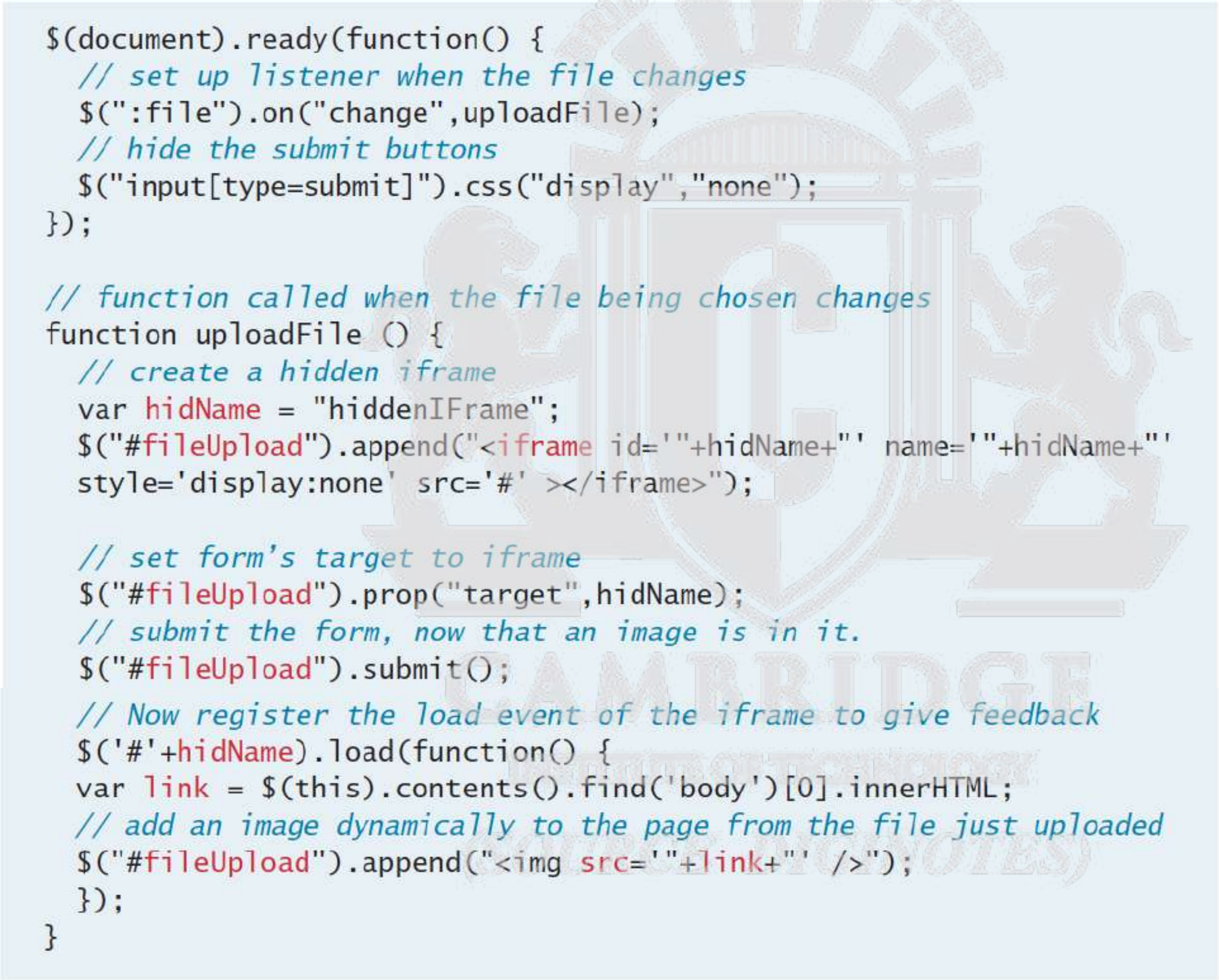


```
<form name="fileUpload" id="fileUpload" enctype="multipart/form-data"
      method="post" action="upload.php">
<input name="images" id="images" type="file" multiple />
<input type="submit" name="submit" value="Upload files!"/>
</form>
```

LISTING 15.17 Simple file upload form

Asynchronous File Transmission

The old iFrame technique



```
$(document).ready(function() {
  // set up listener when the file changes
  $(":file").on("change",uploadFile);
  // hide the submit buttons
  $("input[type=submit]").css("display","none");
});

// function called when the file being chosen changes
function uploadFile () {
  // create a hidden iframe
  var hidName = "hiddenIFrame";
  $("#fileUpload").append("<iframe id='"+hidName+"' name='"+hidName+"'
  style='display:none' src='#' ></iframe>");

  // set form's target to iframe
  $("#fileUpload").prop("target",hidName);
  // submit the form, now that an image is in it.
  $("#fileUpload").submit();
  // Now register the load event of the iframe to give feedback
  $('#'+hidName).load(function() {
  var link = $(this).contents().find('body')[0].innerHTML;
  // add an image dynamically to the page from the file just uploaded
  $("#fileUpload").append("<img src='"+link+"' />");
  });
}
```

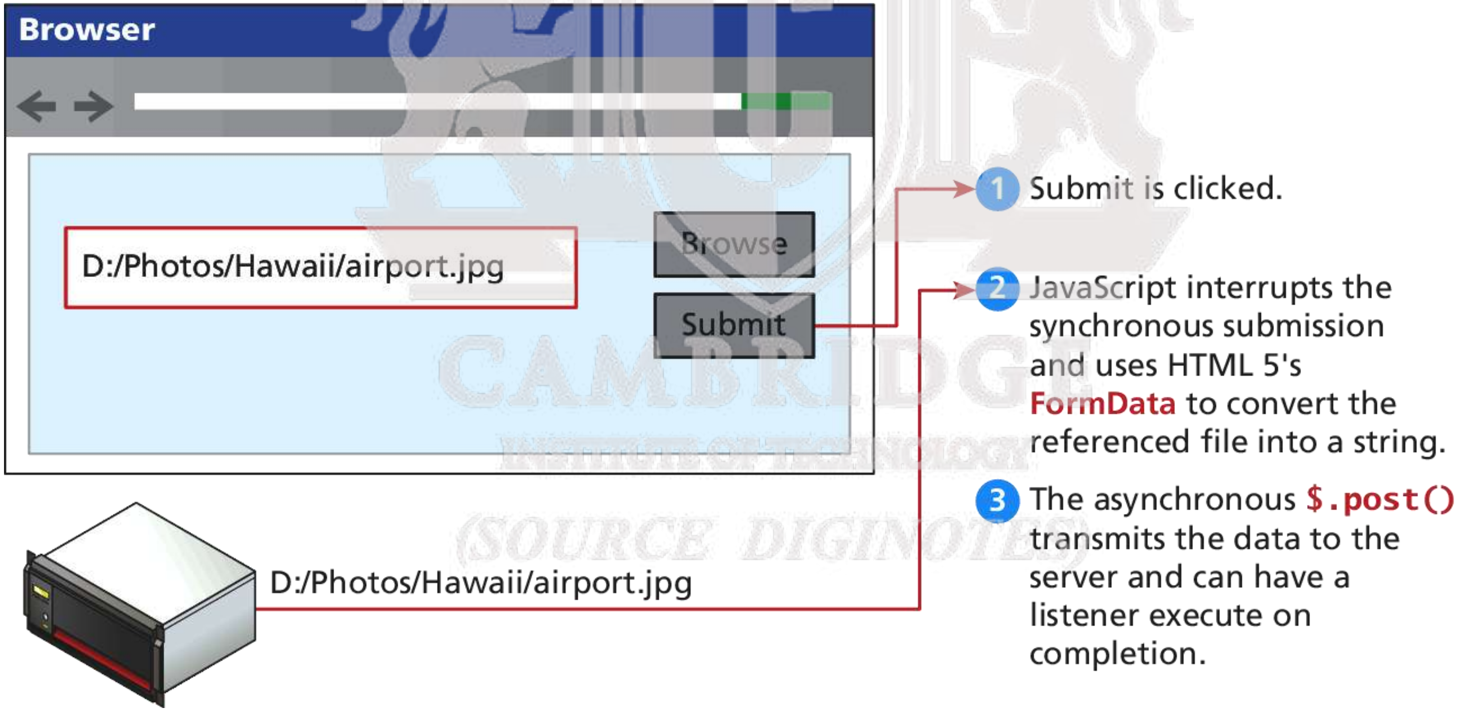
LISTING 15.18 Hidden iFrame technique to upload files

Source diginotes.in

Asynchronous File Transmission

New Form Data technique

•Using the **FormData** interface and File API, which is part of HTML5, you no longer have to trick the browser into posting your file data asynchronously.



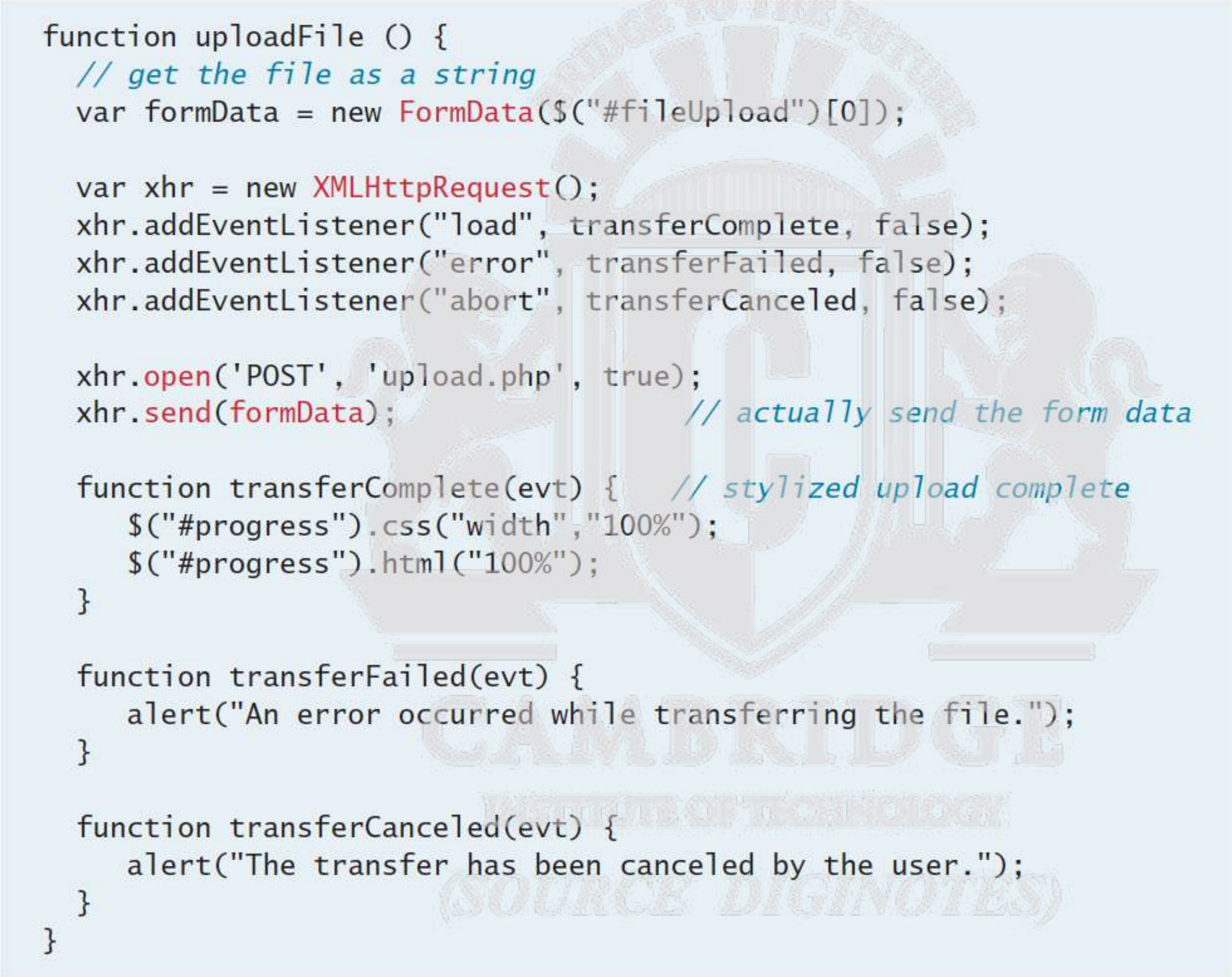
1 Submit is clicked.

2 JavaScript interrupts the synchronous submission and uses HTML 5's **FormData** to convert the referenced file into a string.

3 The asynchronous `$.post()` transmits the data to the server and can have a listener execute on completion.

Asynchronous File Transmission

New Form Data technique



```
function uploadFile () {
  // get the file as a string
  var formData = new FormData($("#fileUpload")[0]);

  var xhr = new XMLHttpRequest();
  xhr.addEventListener("load", transferComplete, false);
  xhr.addEventListener("error", transferFailed, false);
  xhr.addEventListener("abort", transferCanceled, false);

  xhr.open('POST', 'upload.php', true);
  xhr.send(formData);                  // actually send the form data

  function transferComplete(evt) {    // stylized upload complete
     $("#progress").css("width","100%");
     $("#progress").html("100%");
  }

  function transferFailed(evt) {
     alert("An error occurred while transferring the file.");
  }

  function transferCanceled(evt) {
     alert("The transfer has been canceled by the user.");
  }
}
```

LISTING 15.19 Using the new FormData interface from the XHR2 Specification to post files asychronously

Asynchronous File Transmission

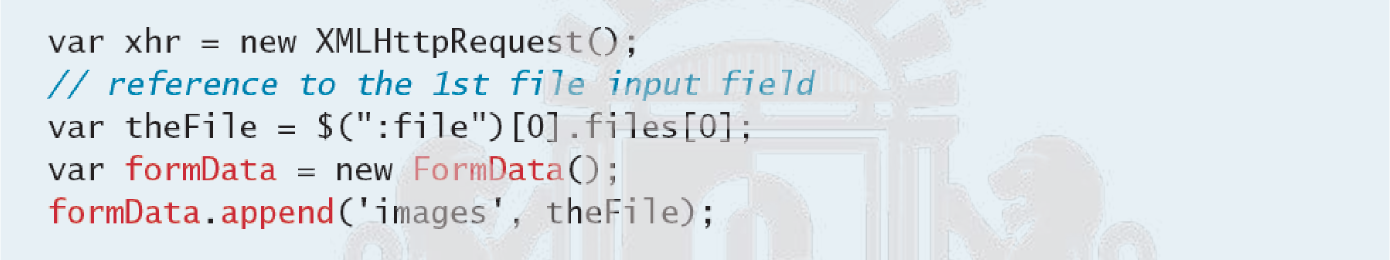
Advanced modern technique

- When we consider uploading multiple files, you may want to upload a single file, rather than the entire form every time. To support that pattern, you can access a single file and post it by appending the raw file to a FormData object.

- The advantage of this technique is that you submit each file to the server asynchronously as the user changes it; and it allows multiple files to be transmitted at once.

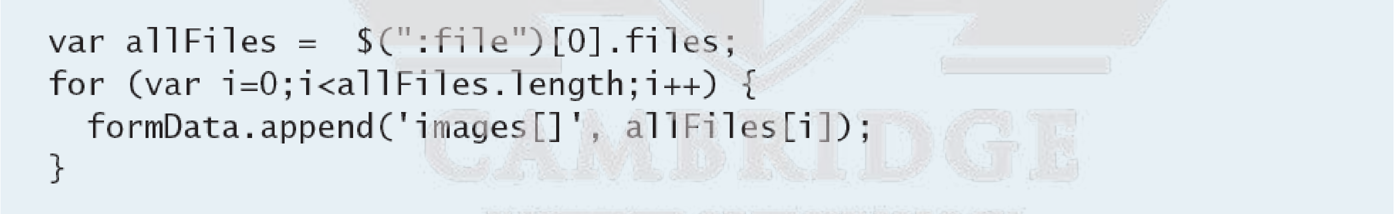
Asynchronous File Transmission

Advanced modern technique



```
var xhr = new XMLHttpRequest();
// reference to the 1st file input field
var theFile = $(":file")[0].files[0];
var formData = new FormData();
formData.append('images', theFile);
```

LISTING 15.20 Posting a single file from a form



```
var allFiles =  $(":file")[0].files;
for (var i=0;i<allFiles.length;i++) {
  formData.append('images[]', allFiles[i]);
}
```

LISTING 15.21 Looping through multiple files in a file input and appending the data for posting

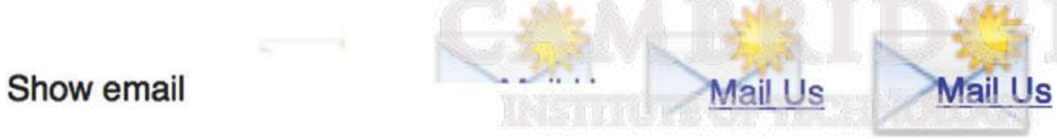
Section 5 of 6

ANIMATION

Animation

Hide() and Show()

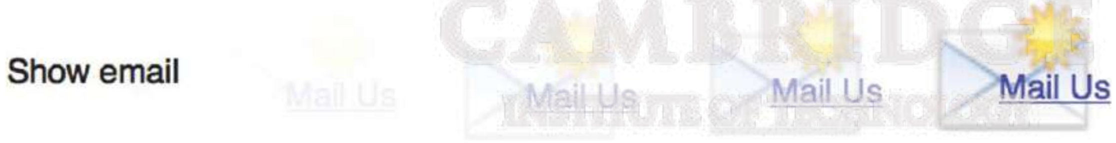
The hide() and show() methods can be called with no arguments to perform a default animation. Another version allows two parameters: the duration of the animation (in milliseconds) and a callback method to execute on completion.



Animation

fadeIn() and fadeOut()

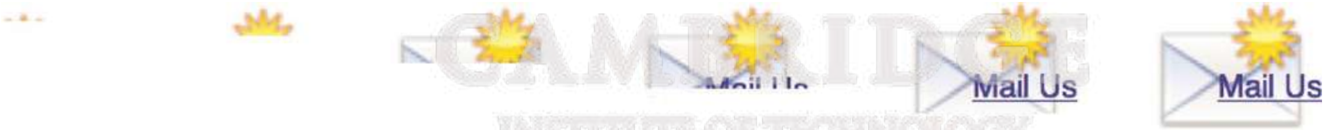
The fadeIn() and fadeOut() shortcut methods control the opacity of an element. The parameters passed are the duration and the callback, just like hide() and show(). Unlike hide() and show(), there is no scaling of the element, just strictly control over the transparency.



Animation

SlideUp() and SlideDown()

slideUp() and slideDown() do not touch the opacity of an element, but rather gradually change its height.



email icon from **http://openiconlibrary.sourceforge.net.**

Animation

Toggle()

As you may have seen, the shortcut methods come in pairs, which make them ideal for toggling between a shown and hidden state. Using a toggle method means you don't have to check the current state and then conditionally call one of the two methods;

- To toggle between the visible and hidden states you can use the **toggle()** methods.
- To toggle between fading in and fading out, use the **fadeToggle()** method
- To toggle between the two sliding states can be achieved using the **slideToggle()** method.

Raw Animation

Full control

❖The animate() method has several versions, but the one we will look at has the following form:

❖.animate(properties, options);

❖The properties parameter contains a Plain Object with all the CSS styles of the final state of the animation.

❖The options parameter contains another Plain Object with any of the following options set:

Raw Animation

Options parameter

- **always** is the function to be called when the animation completes or stops with a fail condition. This function will always be called (hence the name).
- **done** is a function to be called when the animation completes.
- **duration** is a number controlling the duration of the animation.
- **fail** is the function called if the animation does not complete.
- **progress** is a function to be called after each step of the animation.

Raw Animation

Options parameter

- **queue** is a Boolean value telling the animation whether to wait in the queue of animations or not. If false, the animation begins immediately.
- **step** is a function you can define that will be called periodically while the animation is still going.
- Advanced options called **easing** and **specialEasing** allow for advanced control over the speed of animation.

Raw Animation

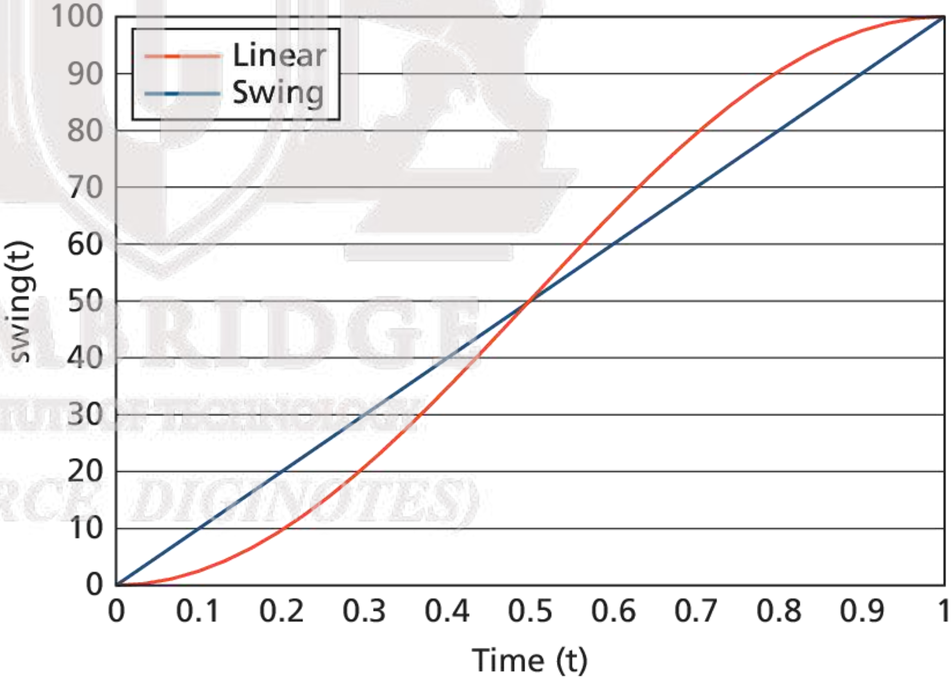
Easing functions – advanced animation

In web development, **easing functions** are used to simulate that natural type of movement. They are mathematical equations that describe how fast or slow the transitions occur at various points during the animation.

Included in jQuery are

- linear
- swing

easing functions.



Raw Animation

Easing functions – advanced animation

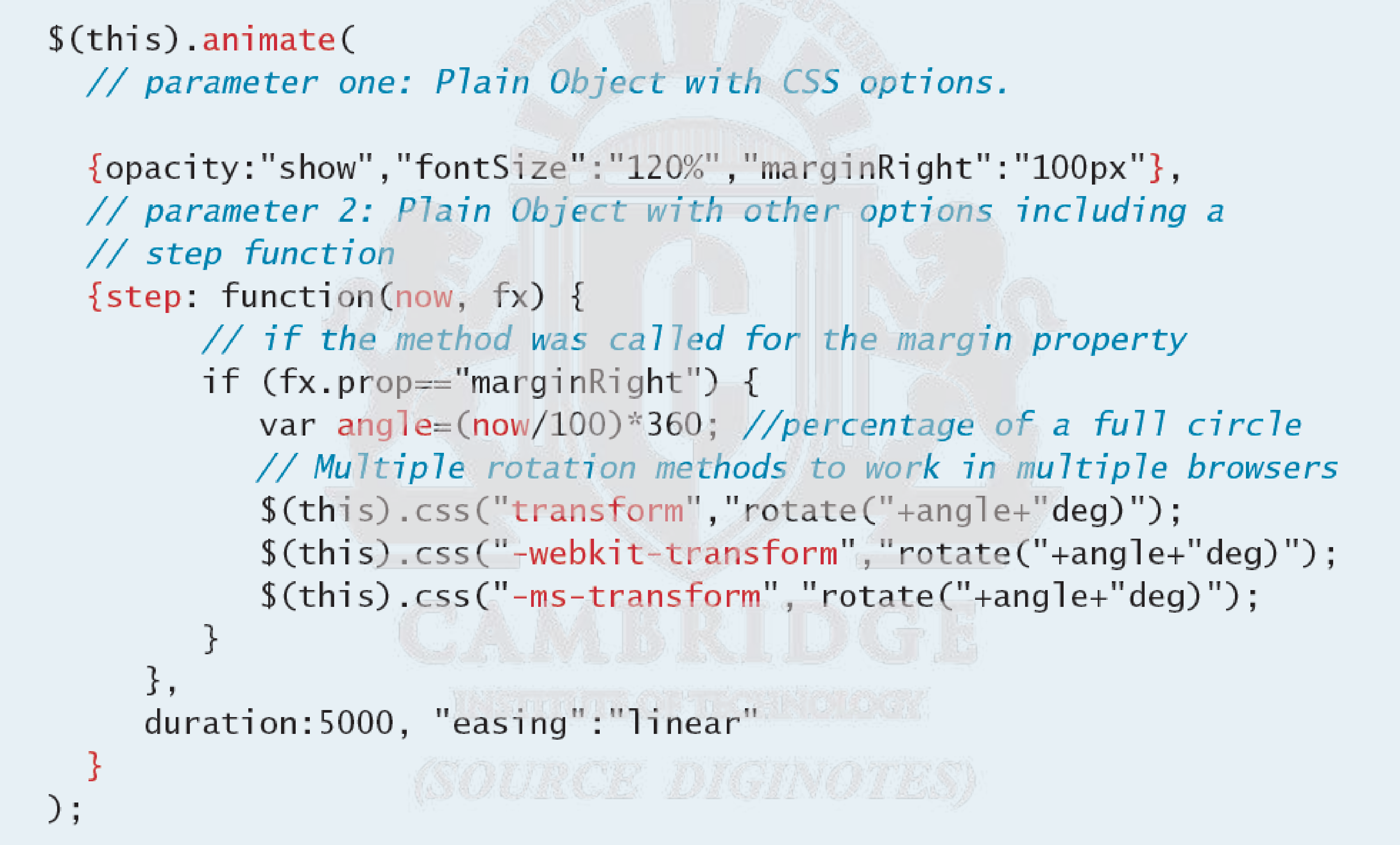
- For example, the function defining swing for values of time *t* between 0 and 1 is

$$\text{swing}(t) = -\frac{1}{2}\cos(t\pi) + 0.5$$

- The jQuery UI extension provides over 30 easing functions, including cubic functions and bouncing effects, so you should not have to define your own.

Advanced example

rotating

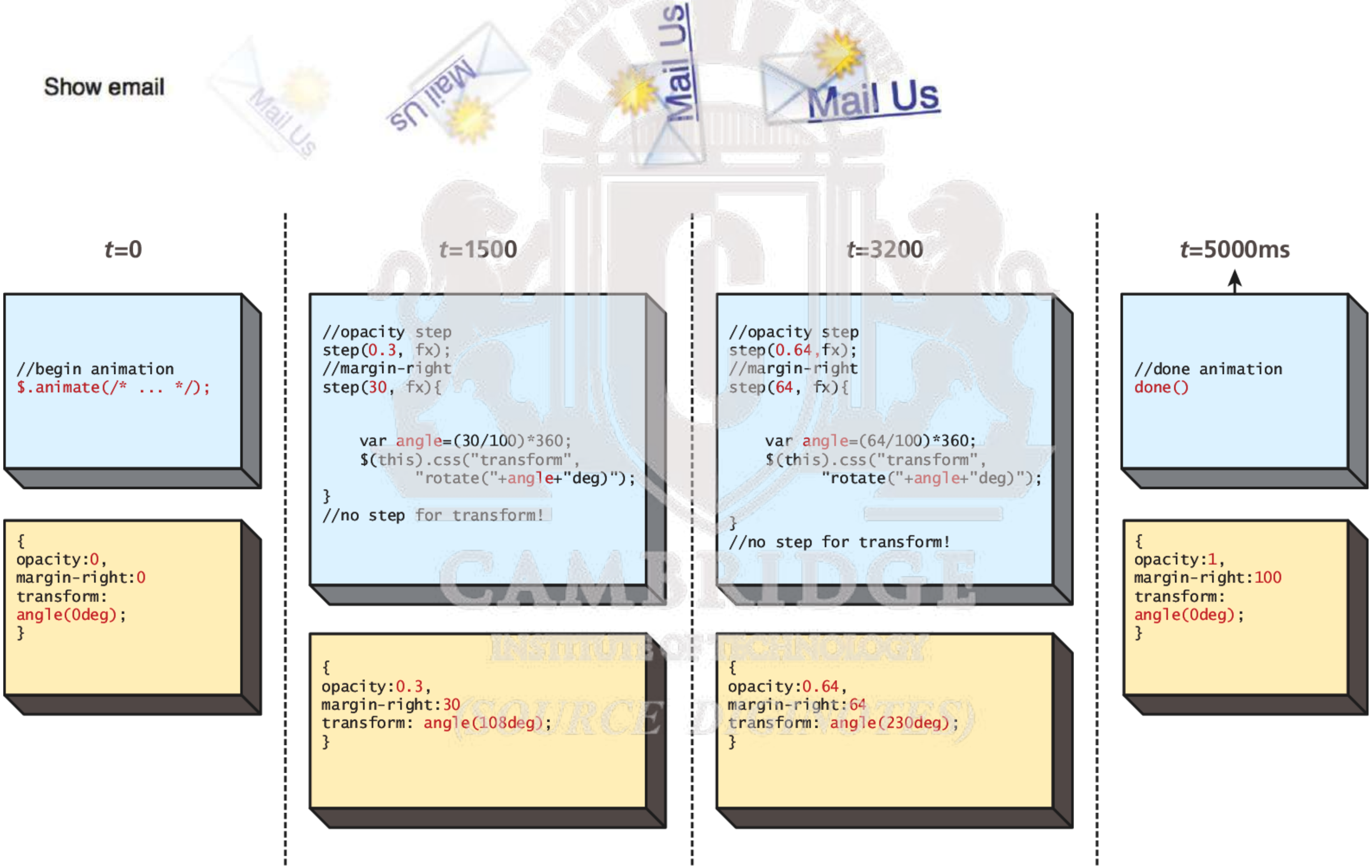


```
$(this).animate(
  // parameter one: Plain Object with CSS options.

  {opacity:"show","fontSize":"120%","marginRight":"100px"},
  // parameter 2: Plain Object with other options including a
  // step function
  {step: function(now, fx) {
        // if the method was called for the margin property
        if (fx.prop=="marginRight") {
           var angle=(now/100)*360; //percentage of a full circle
           // Multiple rotation methods to work in multiple browsers
           $(this).css("transform","rotate("+angle+"deg)");
           $(this).css("-webkit-transform","rotate("+angle+"deg)");
           $(this).css("-ms-transform","rotate("+angle+"deg)");
        }
     },
     duration:5000, "easing":"linear"
  }
);
```

LISTING 15.23 Use of animate() with a step function to do CSS3 rotation

Raw Animation

Rotating



t=0

```
//begin animation
$.animate(/* ... */);
```

```
{
opacity:0,
margin-right:0
transform:
angle(0deg);
}
```

t=1500

```
//opacity step
step(0.3, fx);
//margin-right
step(30, fx){

    var angle=(30/100)*360;
    $(this).css("transform",
          "rotate("+angle+"deg)");
}
//no step for transform!
```

```
{
opacity:0.3,
margin-right:30
transform: angle(108deg);
}
```

t=3200

```
//opacity step
step(0.64,fx);
//margin-right
step(64, fx){

    var angle=(64/100)*360;
    $(this).css("transform",
          "rotate("+angle+"deg)");

}
//no step for transform!
```

```
{
opacity:0.64,
margin-right:64
transform: angle(230deg);
}
```

t=5000ms

```
//done animation
done()
```

```
{
opacity:1,
margin-right:100
transform:
angle(0deg);
}
```

Section 6 of 6

BACKBONE MVC FRAMEWORKS

Backbone

Another framework

•Backbone is an MVC framework that further abstracts JavaScript with libraries intended to adhere more closely to the MVC model

•This library is available from **http://backbonejs.org** and relies on the underscore library, available from http://underscorejs.org/.

•Include with:

•**<script src="underscore-min.js"></script>**

•**<script src="backbone-min.js"></script>**

Backbone

Models

- In Backbone, you build your client scripts around the concept of **models**.

- Backbone.js defines **models** as *the heart of any JavaScript application, containing the interactive data as well as a large part of the logic surrounding it: conversions, validations, computed properties, and access control.*

- The Models you define using Backbone must *extend* Backbone.Model

Backbone

Collections

- ✓In addition to models, Backbone introduces the concept of **Collections**, which are normally used to contain lists of Model objects.

- ✓These collections have advanced features and like a database can have indexes to improve search performance.

- ✓A collection is defined by extending from Backbone’s Collection object.

Backbone

Views

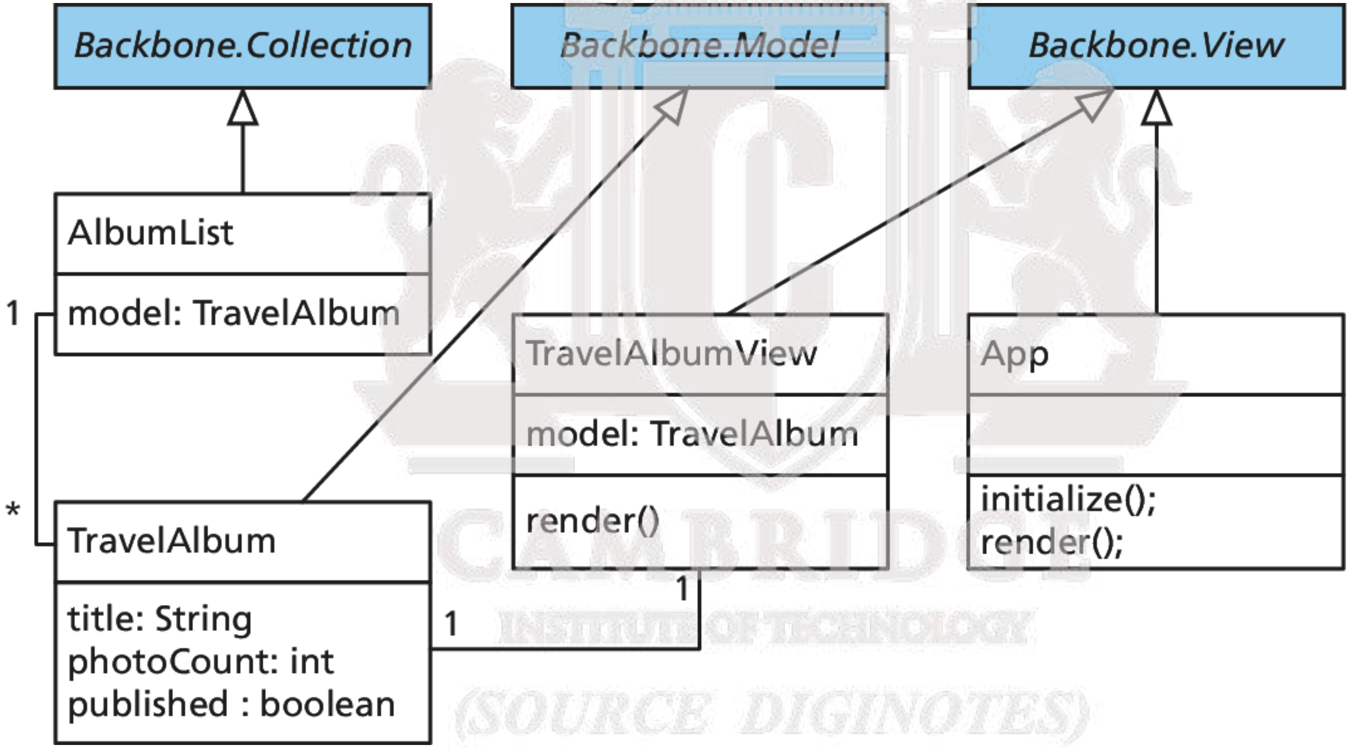
oViews allow you to translate your models into the HTML that is seen by the users.

oThey attach themselves to methods and properties of the Collection and define methods that will be called whenever Backbone determines the view needs refreshing.

oYou must always override the render() method since it defines the HTML that is output.

Backbone

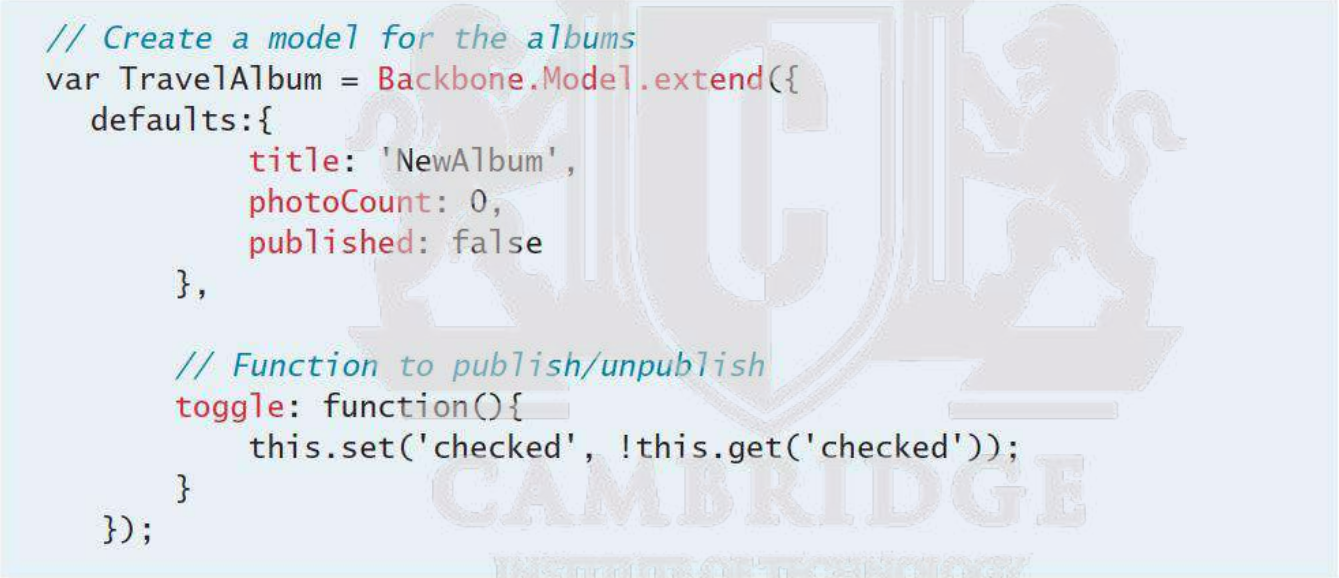
Example



Source diginotes.in

Backbone

A Model Example

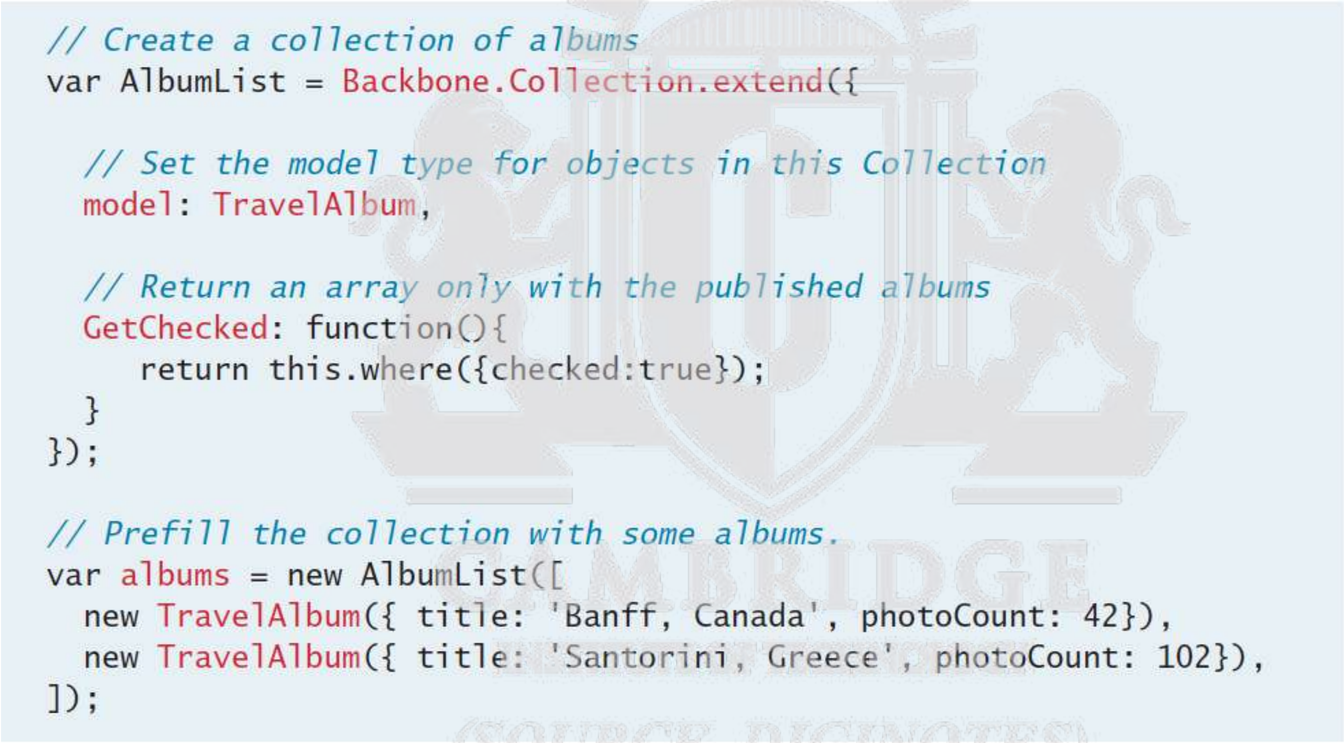


```
// Create a model for the albums
var TravelAlbum = Backbone.Model.extend({
  defaults:{
           title: 'NewAlbum',
           photoCount: 0,
           published: false
       },

       // Function to publish/unpublish
       toggle: function(){
           this.set('checked', !this.get('checked'));
       }
   });
```

LISTING 15.25 A PhotoAlbum Model extending from Backbone.Model

Backbone

A Collection Example

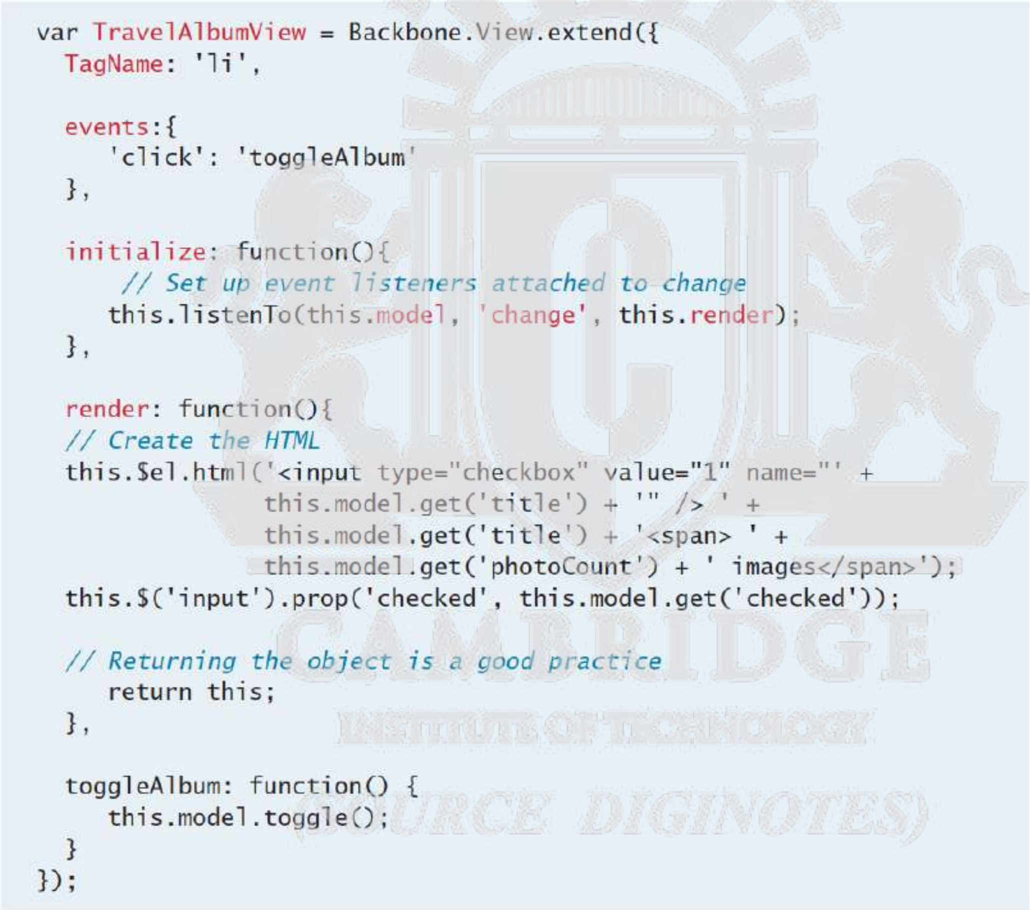


```
// Create a collection of albums
var AlbumList = Backbone.Collection.extend({

  // Set the model type for objects in this Collection
  model: TravelAlbum,

  // Return an array only with the published albums
  GetChecked: function(){
      return this.where({checked:true});
  }
});

// Prefill the collection with some albums.
var albums = new AlbumList([
  new TravelAlbum({ title: 'Banff, Canada', photoCount: 42}),
  new TravelAlbum({ title: 'Santorini, Greece', photoCount: 102}),
]);
```

LISTING 15.26 Demonstration of a Backbone.js Collection defined to hold PhotoAlbums

Backbone

A View Example

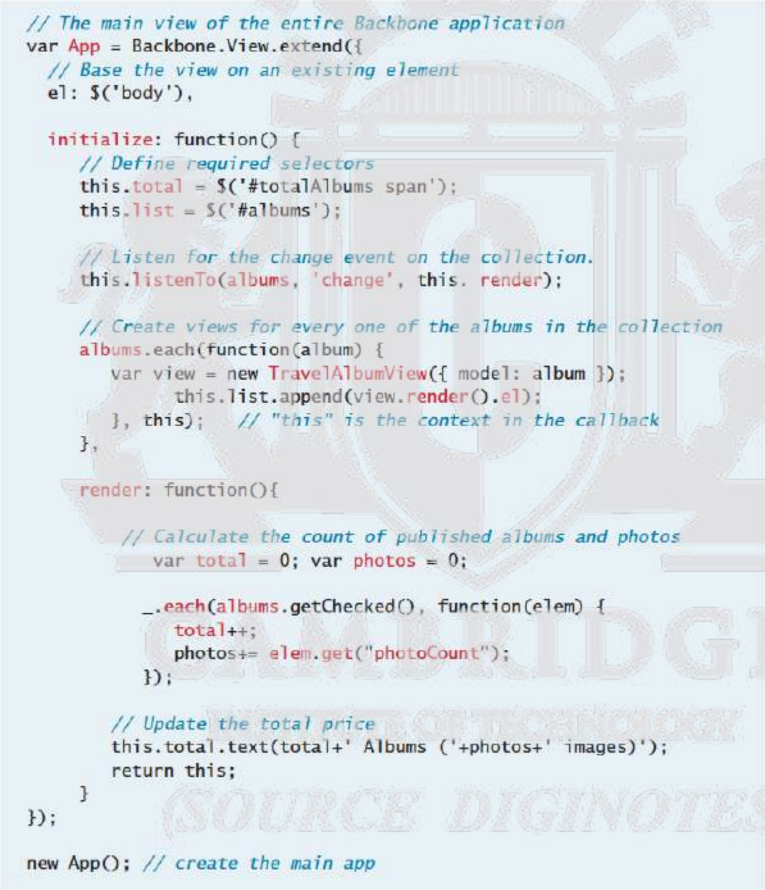


```
var TravelAlbumView = Backbone.View.extend({
  TagName: 'li',

  events:{
     'click': 'toggleAlbum'
  },

  initialize: function(){
      // Set up event listeners attached to change
     this.listenTo(this.model, 'change', this.render);
  },

  render: function(){
  // Create the HTML
  this.$el.html('<input type="checkbox" value="1" name="' +
                this.model.get('title') + '" /> ' +
                this.model.get('title') + '<span> ' +
                this.model.get('photoCount') + ' images</span>');
  this.$('input').prop('checked', this.model.get('checked'));

  // Returning the object is a good practice
     return this;
  },

  toggleAlbum: function() {
     this.model.toggle();
  }
});
```

LISTING 15.27 Deriving custom View objects for our model and Collection

Backbone

Bring it all together



```
// The main view of the entire Backbone application
var App = Backbone.View.extend({
  // Base the view on an existing element
  el: $('body'),

  initialize: function() {
     // Define required selectors
     this.total = $('#totalAlbums span');
     this.list = $('#albums');

     // Listen for the change event on the collection.
     this.listenTo(albums, 'change', this.render);

     // Create views for every one of the albums in the collection
     albums.each(function(album) {
        var view = new TravelAlbumView({ model: album });
             this.list.append(view.render().el);
        }, this);   // "this" is the context in the callback
     },

     render: function(){

        // Calculate the count of published albums and photos
          var total = 0; var photos = 0;

          _.each(albums.getChecked(), function(elem) {
             total++;
             photos+= elem.get("photoCount");
          });

        // Update the total price
        this.total.text(total+' Albums ('+photos+' images)');
        return this;
     }
});

new App(); // create the main app
```

LISTING 15.28 Defining the main app's view and making use of the Collections and models defined earlier

XML Processing and Web Services

Chapter 17

Section 1 of 7

XML OVERVIEW

XML Overview

❑XML is a markup language, but unlike HTML, XML can be used to mark up any type of data.

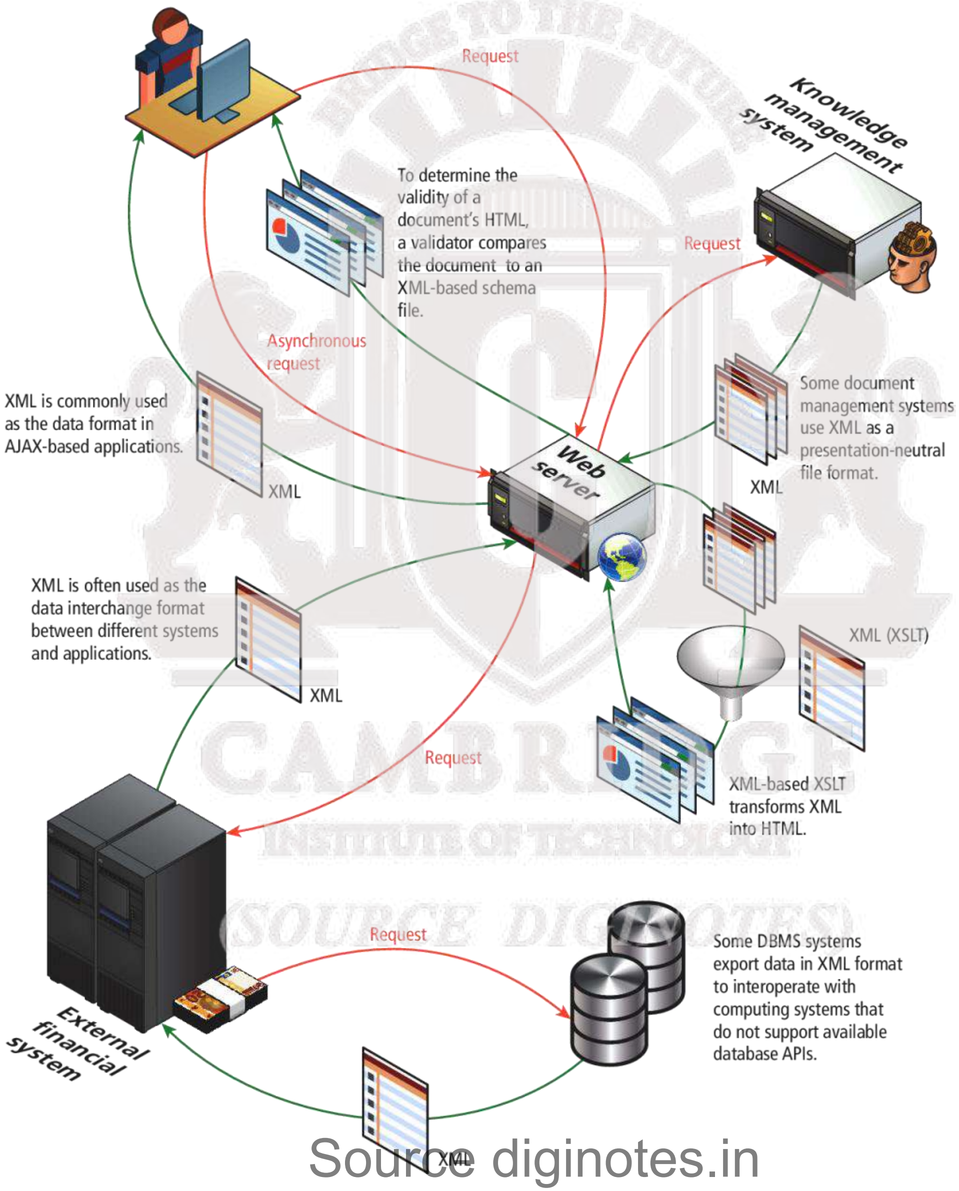
❑Benefits of XML data is that as plain text, it can be read and transferred between applications and different operating systems as well as being human-readable.

❑XML is used on the web server to communicate asynchronously with the browser

❑Used as a data interchange format for moving information between systems

XML Overview

Used in many systems



To determine the validity of a document's HTML, a validator compares the document to an XML-based schema file.

XML is commonly used as the data format in AJAX-based applications.

Some document management systems use XML as a presentation-neutral file format.

XML is often used as the data interchange format between different systems and applications.

XML-based XSLT transforms XML into HTML.

Some DBMS systems export data in XML format to interoperate with computing systems that do not support available database APIs.

Source diginotes.in

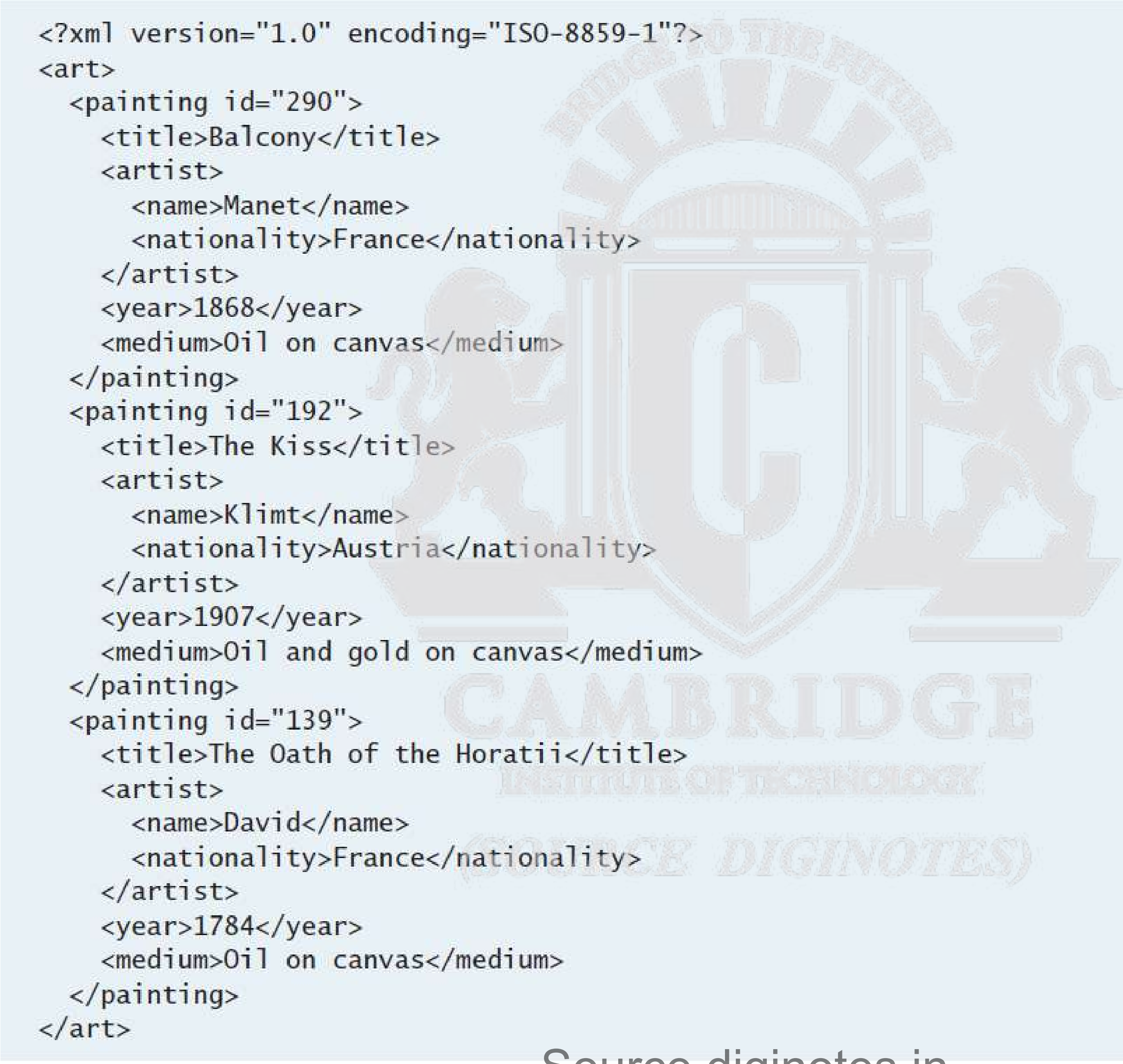
Well Formed XML

For a document to be **well-formed XML**, it must follow the syntax rules for XML:

- Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
- Element names can’t start with a number.
- There must be a single-root element. A **root element** is one that contains all the other elements; for instance, in an HTML document, the root element is <html>.
- All elements must have a closing element (or be self-closing).
- Elements must be properly nested.
- Elements can contain attributes.
- Attribute values must always be within quotes.
- Element and attribute names are case sensitive.

Well Formed XML

Sample Document



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
  <painting id="192">
    <title>The Kiss</title>
    <artist>
      <name>Klimt</name>
      <nationality>Austria</nationality>
    </artist>
    <year>1907</year>
    <medium>Oil and gold on canvas</medium>
  </painting>
  <painting id="139">
    <title>The Oath of the Horatii</title>
    <artist>
      <name>David</name>
      <nationality>France</nationality>
    </artist>
    <year>1784</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

LISTING 17.1 Sample XML document

Valid XML

Requires a DTD

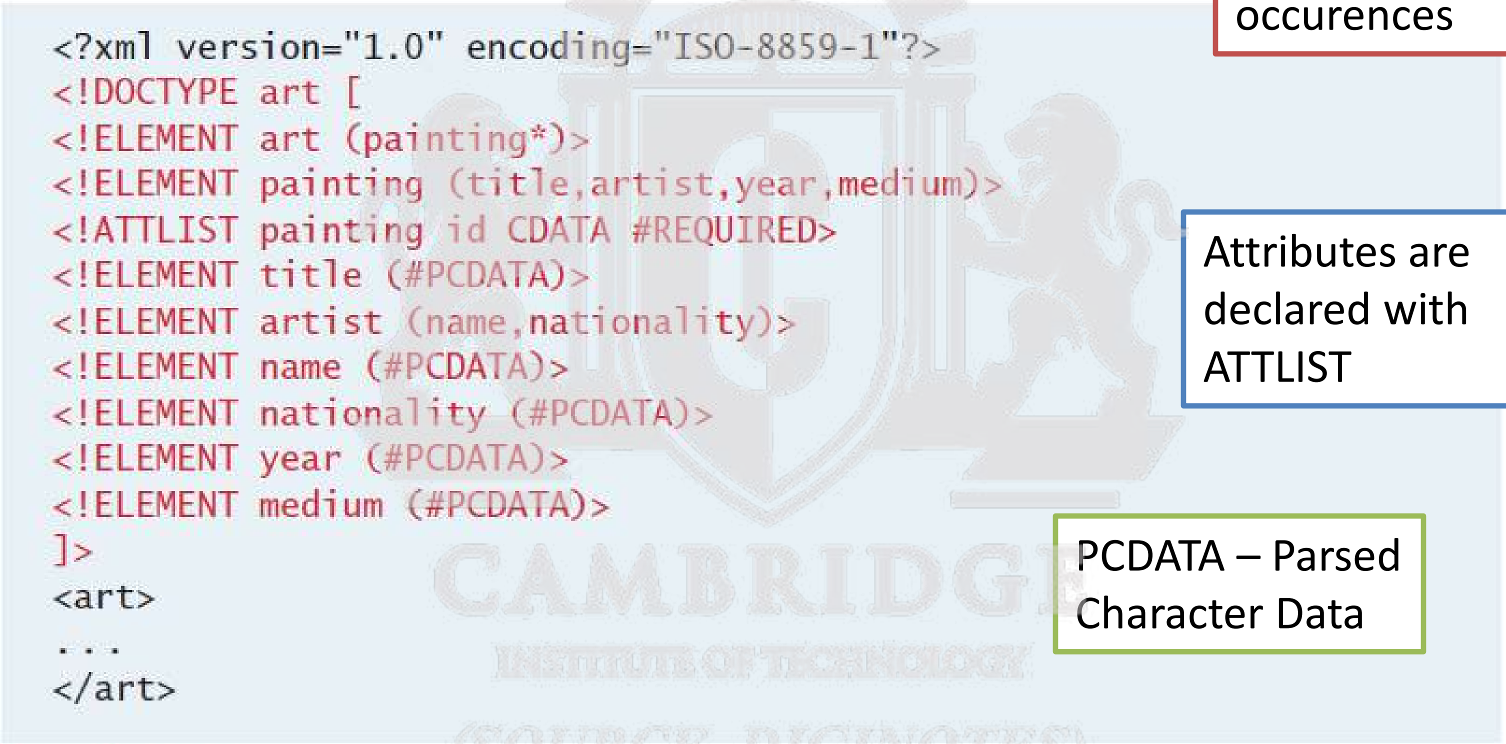
- A **valid XML** document is one that is well formed and whose element and content conform to a document type definition (DTD) or its schema.

- A DTD tells the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements.

- A DTD can be defined within an XML document or within an external file.

Data Type Definition

Example



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE art [
<!ELEMENT art (painting*)>
<!ELEMENT painting (title,artist,year,medium)>
<!ATTLIST painting id CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (name,nationality)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT medium (#PCDATA)>
]>
<art>
. . .
</art>
```

LISTING 17.2 Example DTD

The * allows zero or more occurences

Attributes are declared with ATTLIST

PCDATA – Parsed Character Data

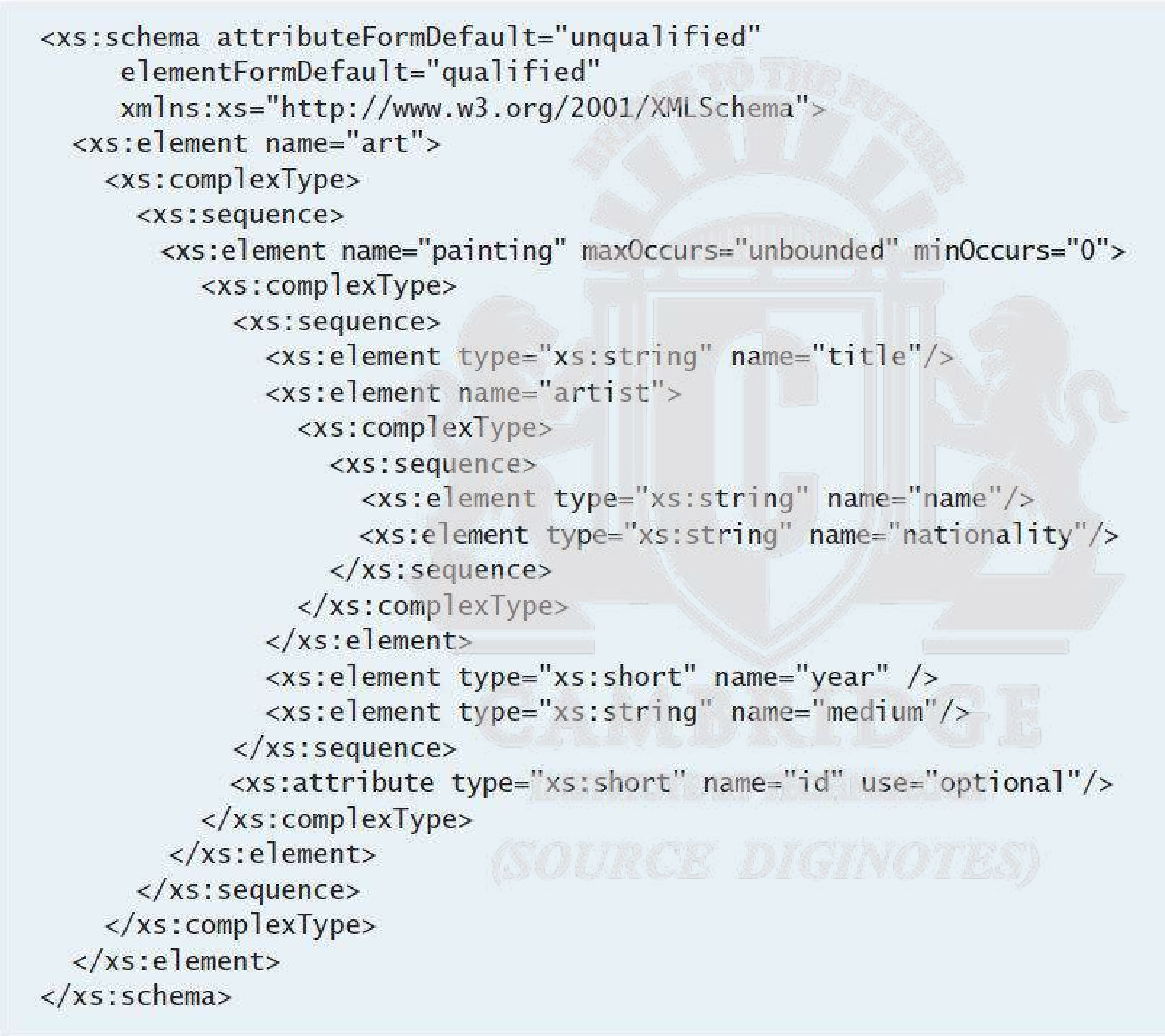
Data Type Definition

Example

- The main drawback with DTDs is that they can only validate the existence and ordering of elements. They provide no way to validate the values of attributes or the textual content of elements.

- For this type of validation, one must instead use XML schemas, which have the added advantage of using XML syntax. Unfortunately, schemas have the corresponding disadvantage of being long-winded and harder for humans to read and comprehend; for this reason, they are typically created with tools.

XML Schema



```
<xs:schema attributeFormDefault="unqualified"
     elementFormDefault="qualified"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="art">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="painting" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="title"/>
              <xs:element name="artist">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element type="xs:string" name="nationality"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element type="xs:short" name="year" />
              <xs:element type="xs:string" name="medium"/>
            </xs:sequence>
            <xs:attribute type="xs:short" name="id" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

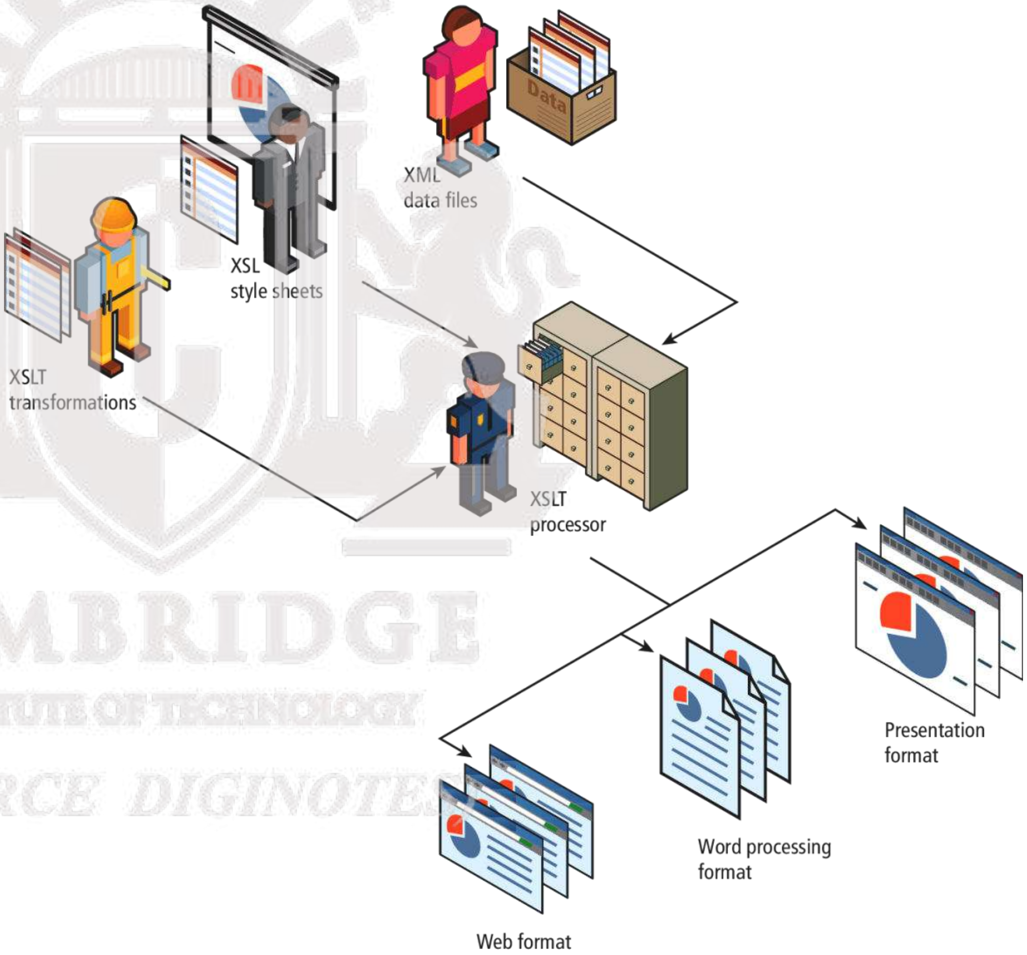
LISTING 17.3 Example schema

EXtensible Stylesheet Language Transformation

XSLT

XML Stylesheet Transformations

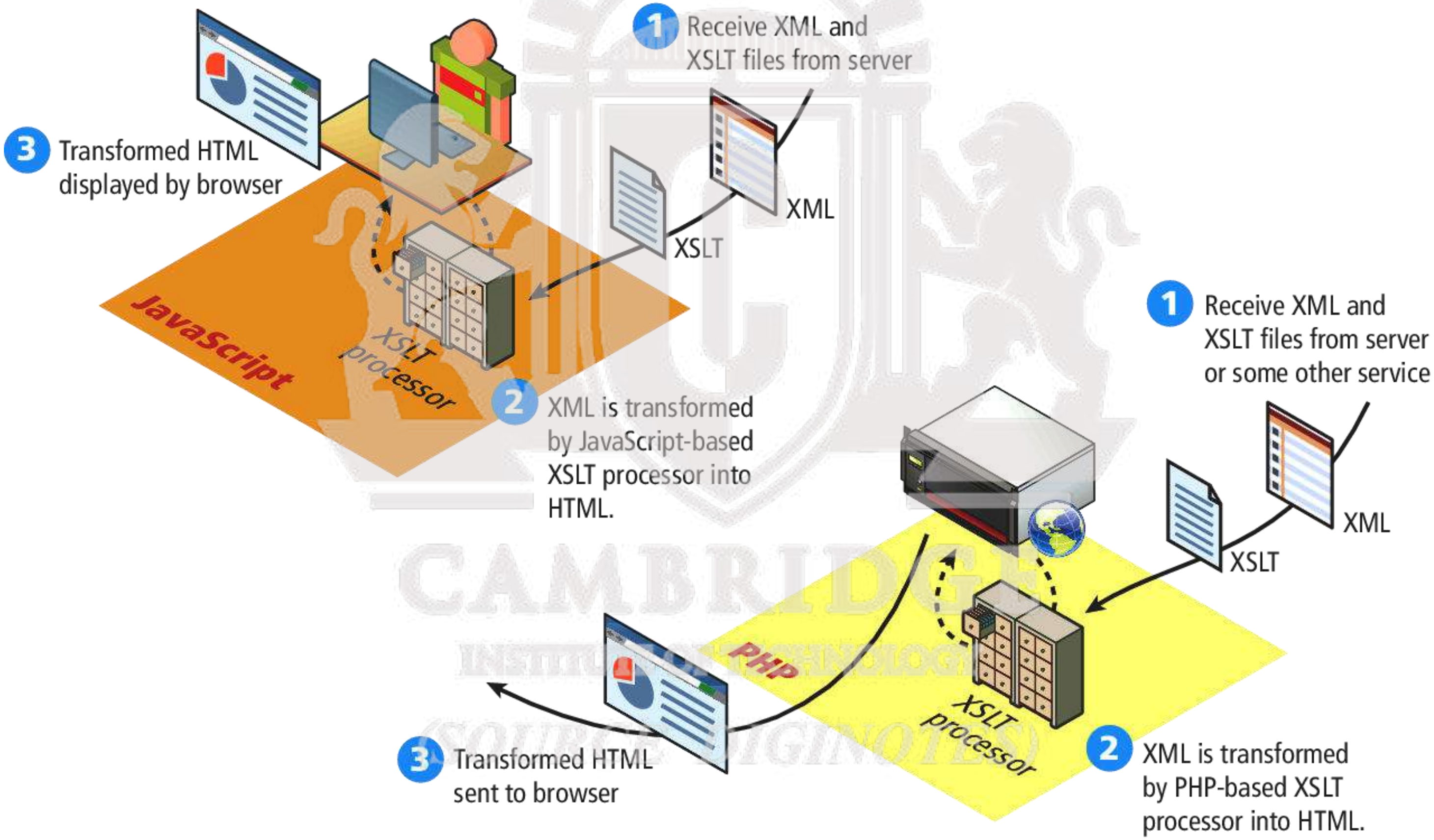
XSLT is an XML-based programming language that is used for transforming XML into other document formats



XSLT

Another usage

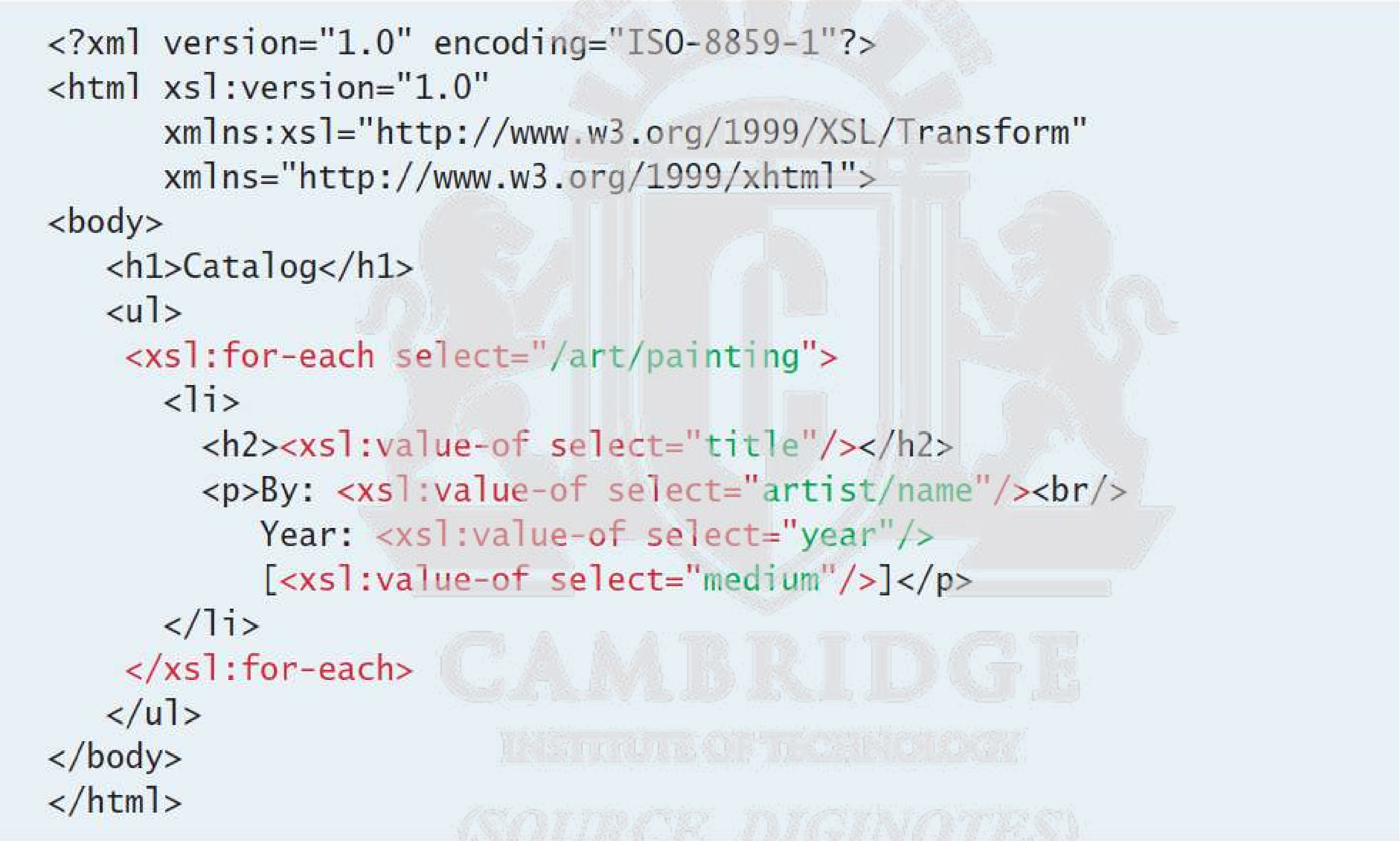
XSLT is also used on the server side and within JavaScript


1 Receive XML and XSLT files from server
XML
XSLT
3 Transformed HTML displayed by browser
JavaScript
XSLT processor
2 XML is transformed by JavaScript-based XSLT processor into HTML.
1 Receive XML and XSLT files from server or some other service
XML
XSLT
PHP
XSLT processor
2 XML is transformed by PHP-based XSLT processor into HTML.
3 Transformed HTML sent to browser

Source diginotes.in

XSLT

Example XSLT document that converts the XML from Listing 17.1 into an HTML list

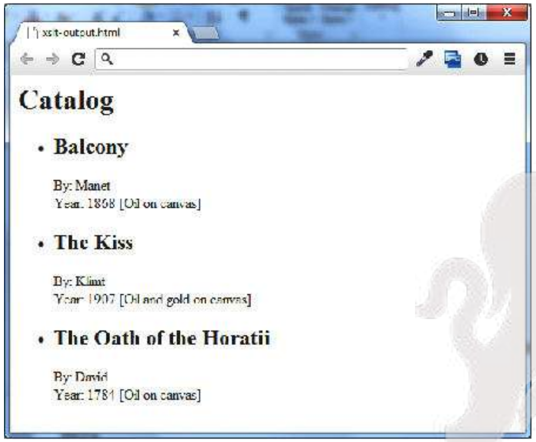


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xsl:version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://www.w3.org/1999/xhtml">
<body>
   <h1>Catalog</h1>
   <ul>
    <xsl:for-each select="/art/painting">
      <li>
        <h2><xsl:value-of select="title"/></h2>
        <p>By: <xsl:value-of select="artist/name"/><br/>
           Year: <xsl:value-of select="year"/>
           [<xsl:value-of select="medium"/>]</p>
      </li>
    </xsl:for-each>
   </ul>
</body>
</html>
```

LISTING 17.4 An example XSLT document

XSLT

An XML parser is still needed to perform the actual transformation



Catalog

- **Balcony**

  By: Manet
  Year: 1868 [Oil on canvas]

- **The Kiss**

  By: Klimt
  Year: 1907 [Oil and gold on canvas]

- **The Oath of the Horatii**

  By: David
  Year: 1784 [Oil on canvas]

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<h1>Catalog</h1>
<ul>
  <li>
    <h2>Balcony</h2>
    <p>By: Manet<br/>
    Year: 1868 [Oil on canvas]</p>
  </li>
  <li>
    <h2>The Kiss</h2>
    <p>By: Klimt<br/>
    Year: 1907 [Oil and gold on canvas]</p>
  </li>
  <li>
    <h2>The Oath of the Horatii</h2>
    <p>By: David<br/>Year: 1784 [Oil on canvas]</p>
  </li>
</ul>
</body>
</html>
```

XPath
Another XML Technology

❖**XPath** is a standardized syntax for searching an XML document and for navigating to elements within the XML document

❖XPath is typically used as part of the programmatic manipulation of an XML document in PHP and other languages

❖XPath uses a syntax that is similar to the one used in most operating systems to access directories.

XPath

Learn through example

/art/painting[@id='192']/artist/name

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
  <painting id="192">
    <title>The Kiss</title>
    <artist>
      <name>Klimt</name>
      <nationality>Austria</nationality>
    </artist>
    <year>1907</year>
    <medium>Oil and gold on canvas</medium>
  </painting>
  <painting id="139">
    <title>The Oath of the Horatii</title>
    <artist>
      <name>David</name>
      <nationality>France</nationality>
    </artist>
    <year>1784</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

/art/painting[year > 1800]

/art/painting[3]/@id

This used when you want to look for particular data, like just the artist’s name for a particular painting

Source diginotes.in

Save the Earth. Go paperless

Section 2 of 7

XML PROCESSING

XML Processing

Two types

XML processing in PHP, JavaScript, and other modern development environments is divided into two basic styles:

- The **in-memory approach**, which involves reading the entire XML file into memory into some type of data structure with functions for accessing and manipulating the data.
- The **event or pull approach**, which lets you pull in just a few elements or lines at a time, thereby avoiding the memory load of large XML files.

XML Processing

In JavaScript

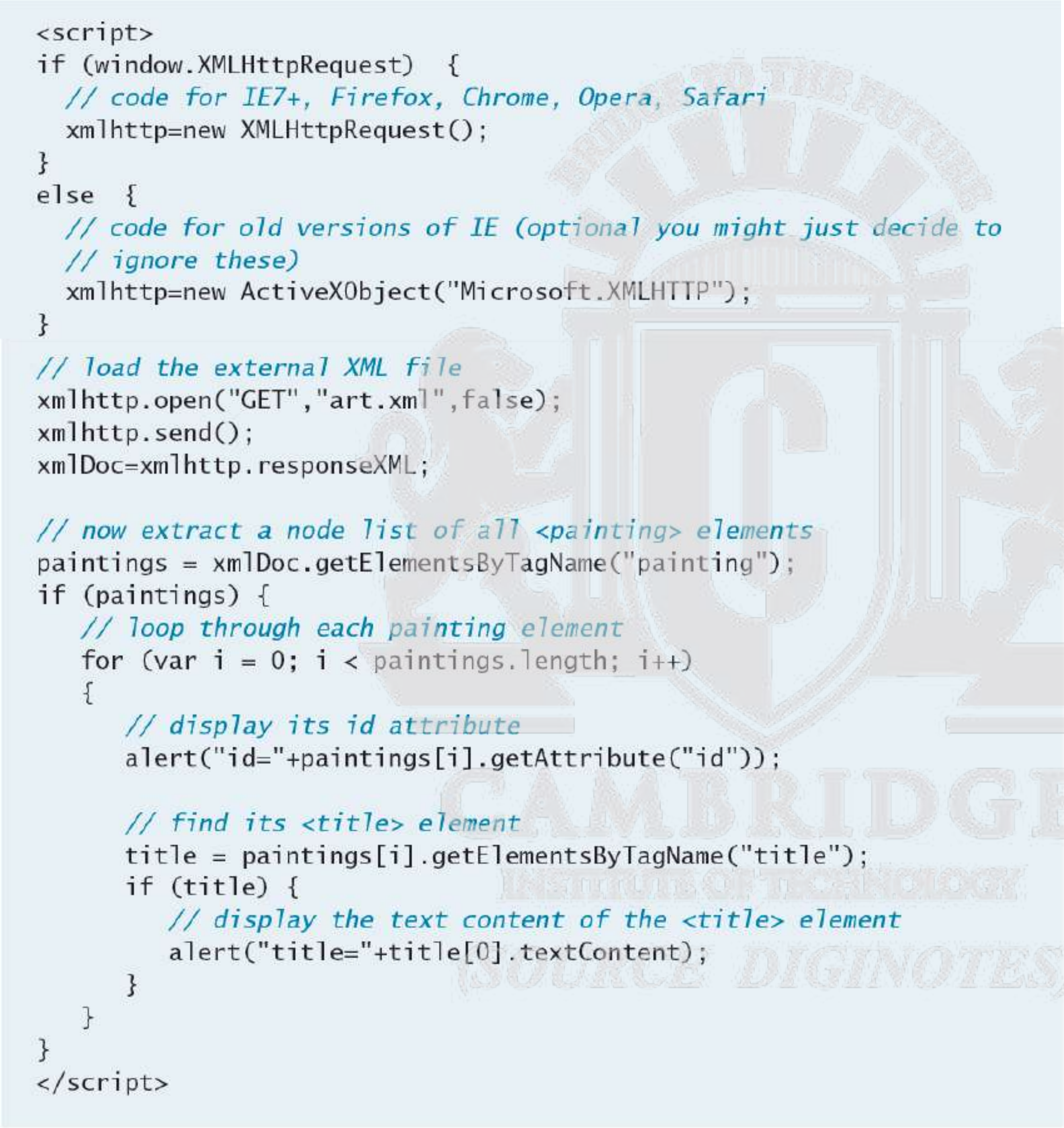
•All modern browsers have a built-in XML parser and their JavaScript implementations support an in-memory XML DOM API.

You can use the already familiar DOM functions such as

- getElementById(),
- getElementsByTagName()
- createElement()

to access and manipulate the data.

XML Processing

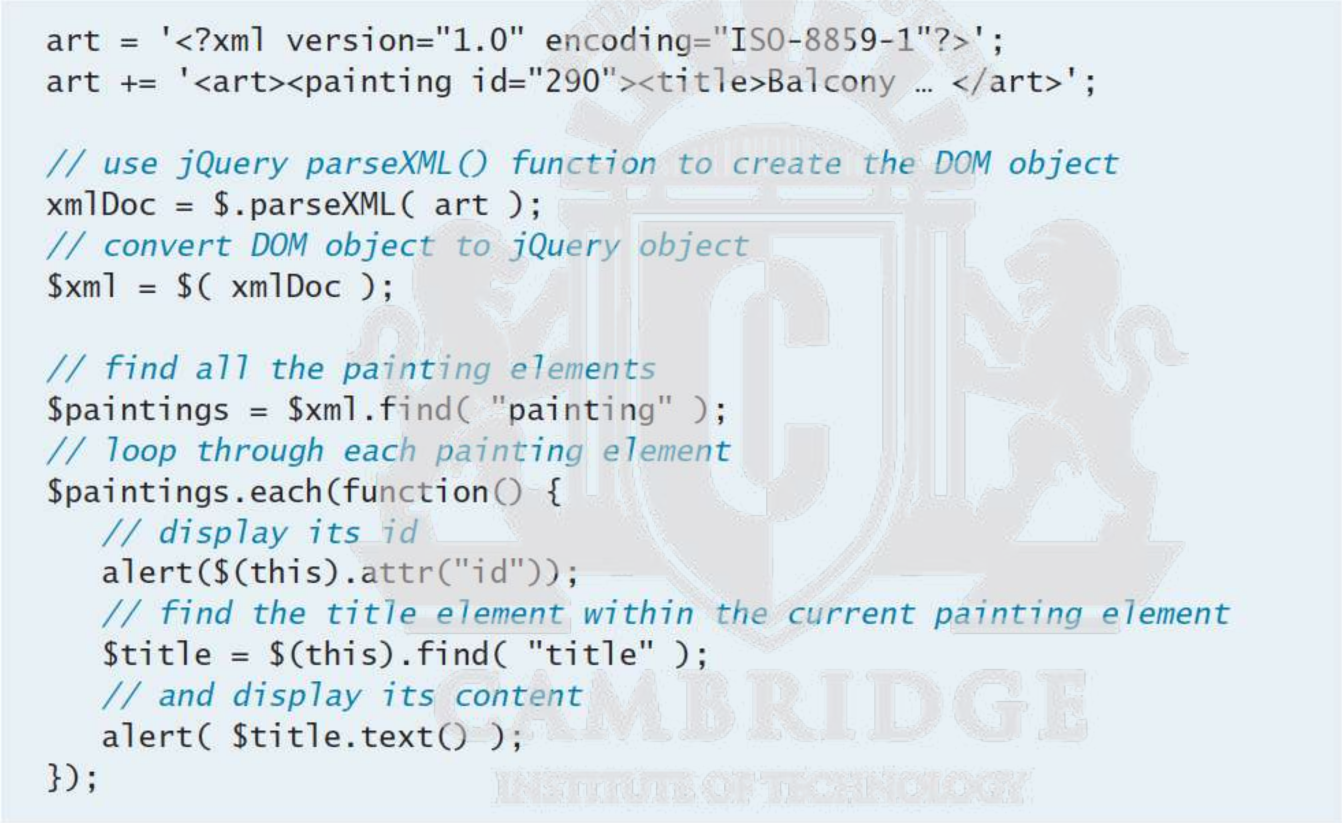


```
<script>
if (window.XMLHttpRequest)  {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
}
else  {
  // code for old versions of IE (optional you might just decide to
  // ignore these)
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
// load the external XML file
xmlhttp.open("GET","art.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

// now extract a node list of all <painting> elements
paintings = xmlDoc.getElementsByTagName("painting");
if (paintings) {
   // loop through each painting element
   for (var i = 0; i < paintings.length; i++)
   {
      // display its id attribute
      alert("id="+paintings[i].getAttribute("id"));

      // find its <title> element
      title = paintings[i].getElementsByTagName("title");
      if (title) {
         // display the text content of the <title> element
         alert("title="+title[0].textContent);
      }
   }
}
</script>
```

LISTING 17.5 Loading and processing an XML document via JavaScript

XML Processing

With JQuery



```
art = '<?xml version="1.0" encoding="ISO-8859-1"?>';
art += '<art><painting id="290"><title>Balcony … </art>';

// use jQuery parseXML() function to create the DOM object
xmlDoc = $.parseXML( art );
// convert DOM object to jQuery object
$xml = $( xmlDoc );

// find all the painting elements
$paintings = $xml.find( "painting" );
// loop through each painting element
$paintings.each(function() {
   // display its id
   alert($(this).attr("id"));
   // find the title element within the current painting element
   $title = $(this).find( "title" );
   // and display its content
   alert( $title.text() );
});
```

LISTING 17.6 XML processing using jQuery

XML Processing
With PHP

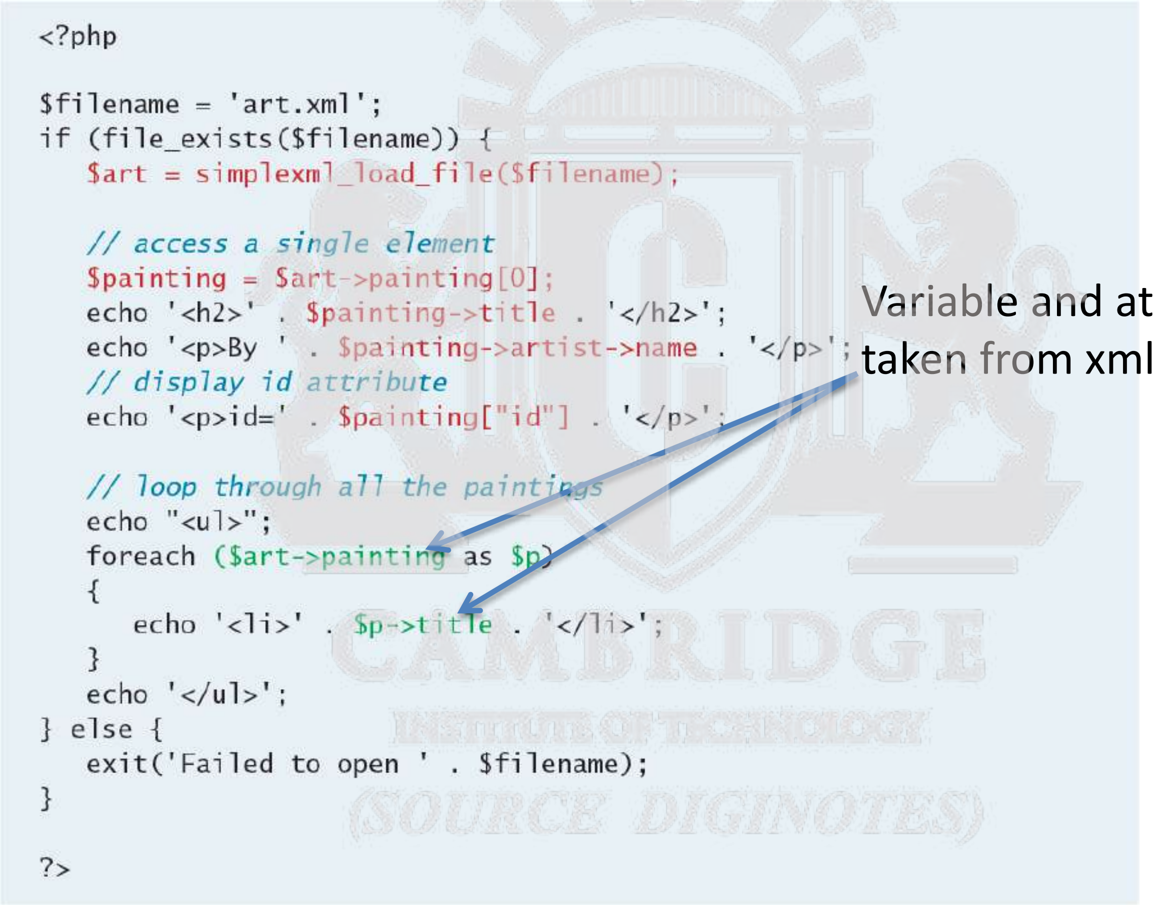
PHP provides several extensions or APIs for working with XML including:

- The **SimpleXML** extension which loads the data into an object that allows the developer to access the data via array properties and modifying the data via methods.

- The **XMLReader** is a read-only pull-type extension that uses a cursor-like approach similar to that used with database processing

XML Processing

With PHP using Simple XML



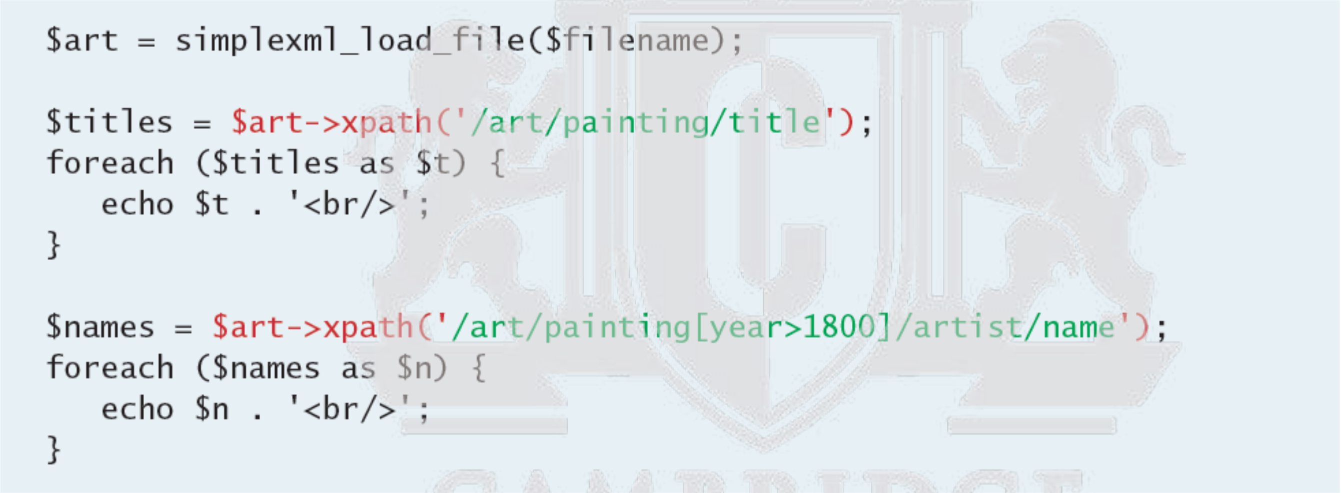
```
<?php

$filename = 'art.xml';
if (file_exists($filename)) {
   $art = simplexml_load_file($filename);

   // access a single element
   $painting = $art->painting[0];
   echo '<h2>' . $painting->title . '</h2>';
   echo '<p>By ' . $painting->artist->name . '</p>';
   // display id attribute
   echo '<p>id=' . $painting["id"] . '</p>';

   // loop through all the paintings
   echo "<ul>";
   foreach ($art->painting as $p)
   {
      echo '<li>' . $p->title . '</li>';
   }
   echo '</ul>';
} else {
   exit('Failed to open ' . $filename);
}

?>
```

LISTING 17.8 Using simple XML

Variable and attribute names taken from xml

XML Processing

With PHP using Simple XML and XPath

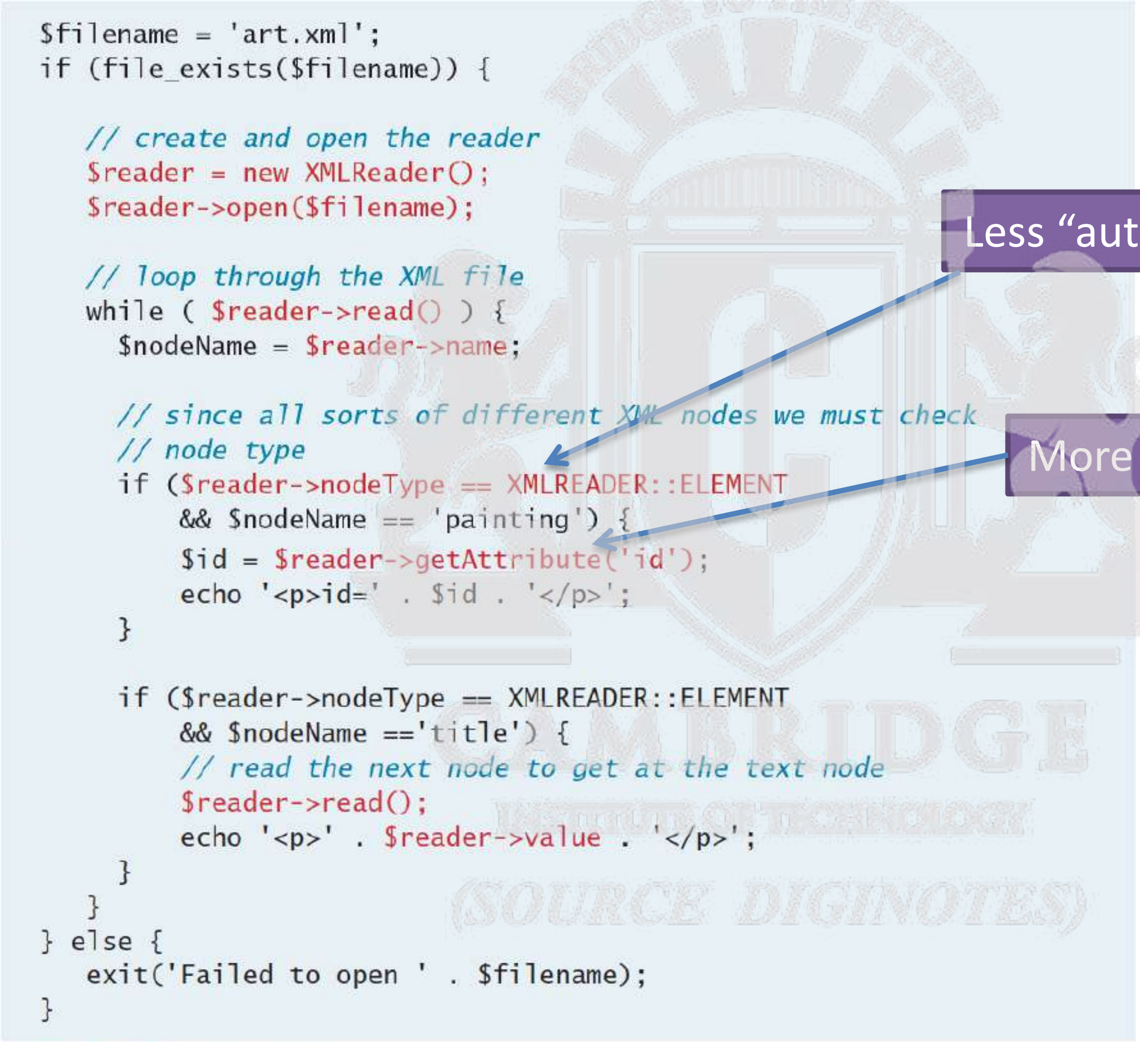


```
$art = simplexml_load_file($filename);

$titles = $art->xpath('/art/painting/title');
foreach ($titles as $t) {
   echo $t . '<br/>';
}

$names = $art->xpath('/art/painting[year>1800]/artist/name');
foreach ($names as $n) {
   echo $n . '<br/>';
}
```

LISTING 17.9 Using XPath with SimpleXML

XML Processing

With PHP using XMLReader



```
$filename = 'art.xml';
if (file_exists($filename)) {

   // create and open the reader
   $reader = new XMLReader();
   $reader->open($filename);

   // loop through the XML file
   while ( $reader->read() ) {
     $nodeName = $reader->name;

     // since all sorts of different XML nodes we must check
     // node type
     if ($reader->nodeType == XMLREADER::ELEMENT
         && $nodeName == 'painting') {
         $id = $reader->getAttribute('id');
         echo '<p>id=' . $id . '</p>';
     }

     if ($reader->nodeType == XMLREADER::ELEMENT
         && $nodeName =='title') {
         // read the next node to get at the text node
         $reader->read();
         echo '<p>' . $reader->value . '</p>';
     }
   }
} else {
   exit('Failed to open ' . $filename);
}
```

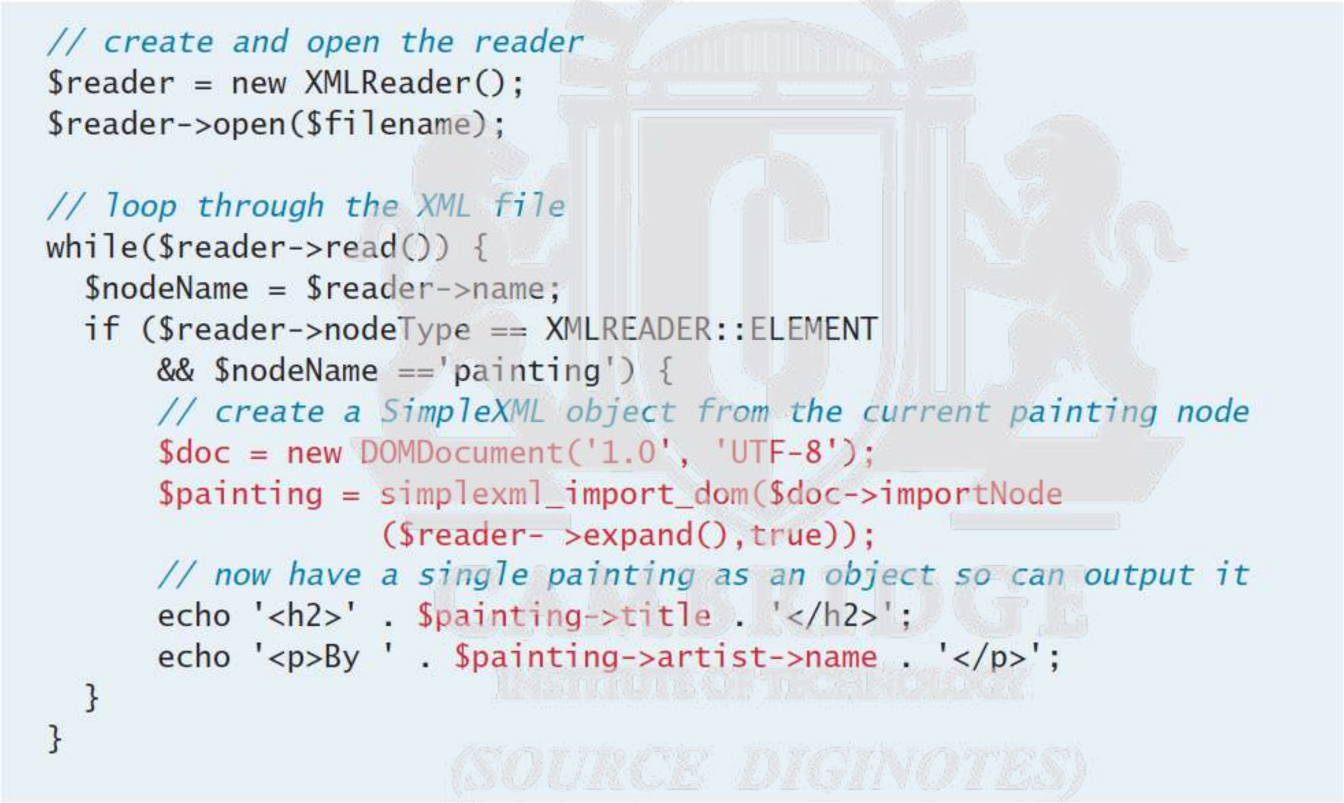
LISTING 17.10 Using XMLReader

Less “automatic”

More Verbose

XML Processing

Why choose when you can use both



```
// create and open the reader
$reader = new XMLReader();
$reader->open($filename);

// loop through the XML file
while($reader->read()) {
  $nodeName = $reader->name;
  if ($reader->nodeType == XMLREADER::ELEMENT
      && $nodeName =='painting') {
      // create a SimpleXML object from the current painting node
      $doc = new DOMDocument('1.0', 'UTF-8');
      $painting = simplexml_import_dom($doc->importNode
                  ($reader- >expand(),true));
      // now have a single painting as an object so can output it
      echo '<h2>' . $painting->title . '</h2>';
      echo '<p>By ' . $painting->artist->name . '</p>';
  }
}
```

LISTING 17.11 Combining XMLReader and SimpleXML

Section 3 of 7

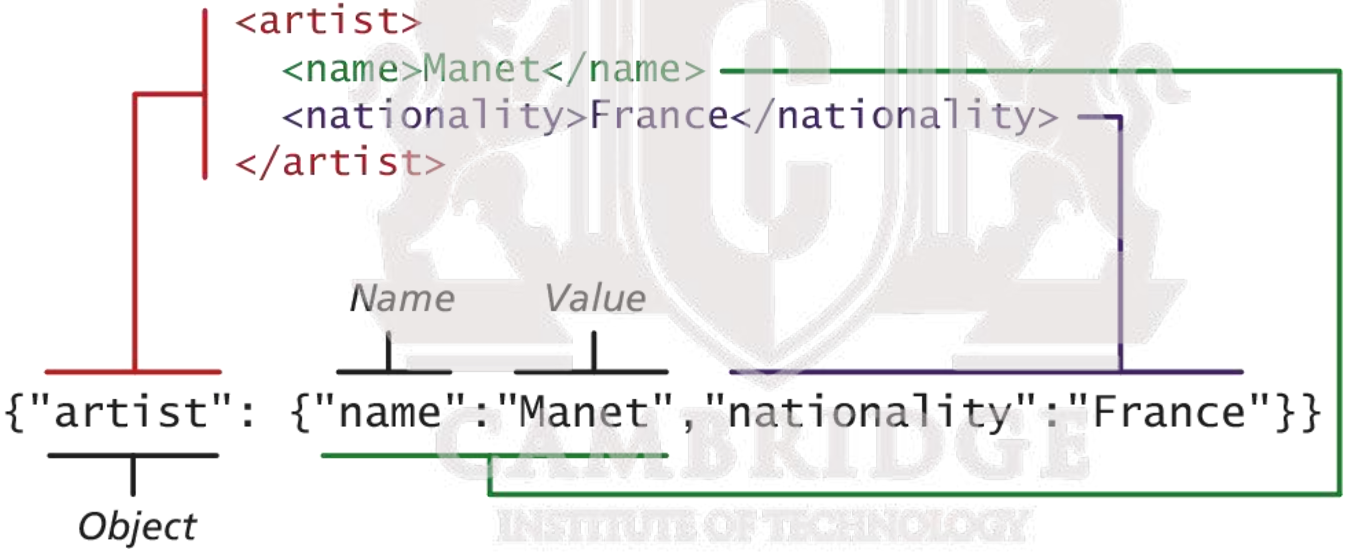
JSON

JSON

- ❑**JSON** stands for JavaScript Object Notation (though its use is not limited to JavaScript)
- ❑Like XML, JSON is a data serialization format. It provides a more concise format than XML.
- ❑Many REST web services encode their returned data in the JSON data format instead of XML.

JSON

An example XML object in JSON

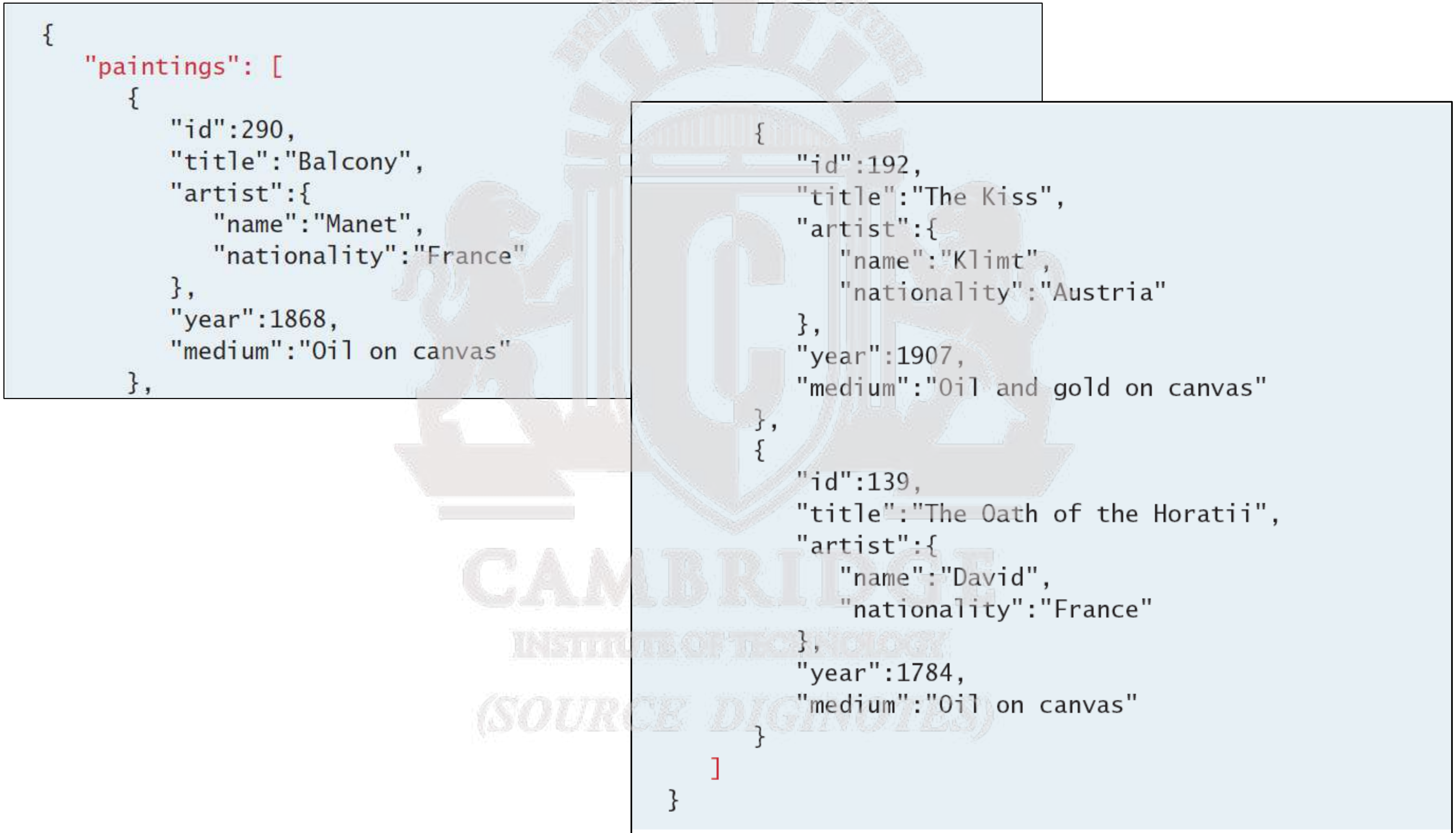


```
<artist>
  <name>Manet</name>
  <nationality>France</nationality>
</artist>
```

```
{"artist": {"name":"Manet","nationality":"France"}}
```

JSON

An example XML object in JSON



```
{
   "paintings": [
      {
         "id":290,
         "title":"Balcony",
         "artist":{
            "name":"Manet",
            "nationality":"France"
         },
         "year":1868,
         "medium":"Oil on canvas"
      },
      {
         "id":192,
         "title":"The Kiss",
         "artist":{
            "name":"Klimt",
            "nationality":"Austria"
         },
         "year":1907,
         "medium":"Oil and gold on canvas"
      },
      {
         "id":139,
         "title":"The Oath of the Horatii",
         "artist":{
            "name":"David",
            "nationality":"France"
         },
         "year":1784,
         "medium":"Oil on canvas"
      }
   ]
}
```

LISTING 17.12 JSON representation of XML data from Listing 17.1

Using JSON in JavaScript

Creating JSON JavaScript objects

it is easy to make use of the JSON format in JavaScript:

```
var a = {"artist": {"name":"Manet","nationality":"France"}};

alert(a.artist.name + " " + a.artist.nationality);
```

When the JSON information will be contained within a string (say when downloading) the JSON.parse() function can be used to transform the string containing into a JavaScript object

Using JSON in JavaScript

Convert string to JSON object and vice versa

```
var text = '{"artist": {"name":"Manet","nationality":"France"}}';
```

```
var a = JSON.parse(text);
```

```
alert(a.artist.nationality);
```

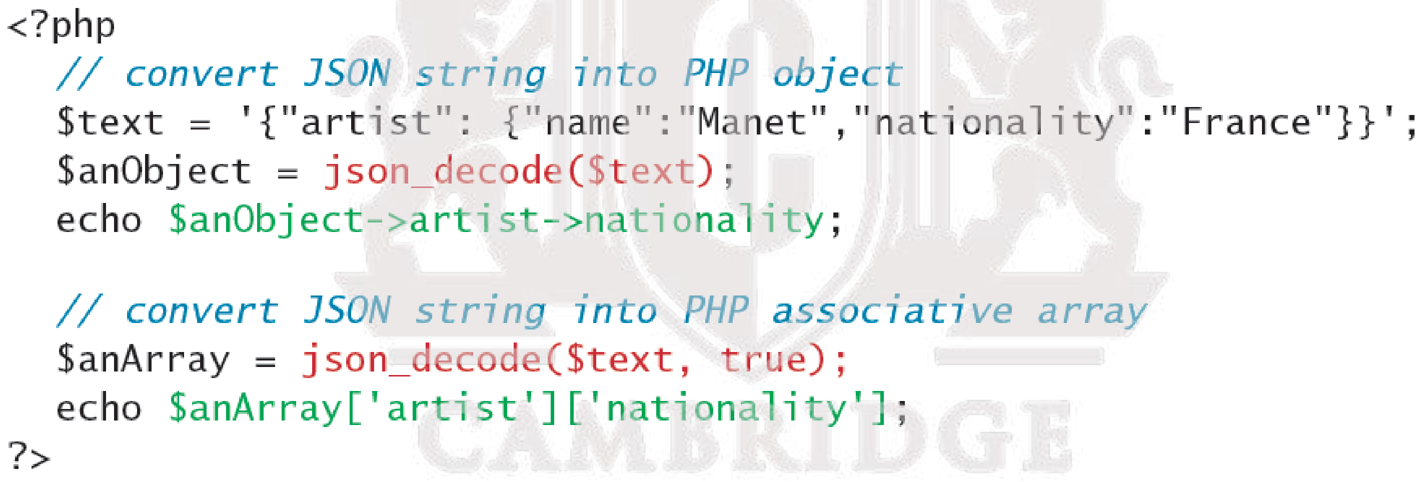
JavaScript also provides a mechanism to translate a JavaScript object into a JSON string:

```
var text = JSON.stringify(artist);
```

Using JSON in PHP

JSON on the server

Converting a JSON string into a PHP object is quite straightforward:



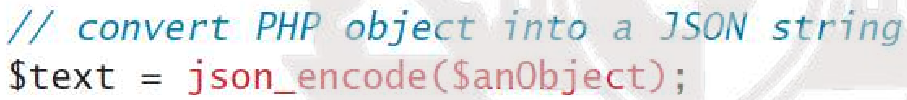
```
<?php
   // convert JSON string into PHP object
   $text = '{"artist": {"name":"Manet","nationality":"France"}}';
   $anObject = json_decode($text);
   echo $anObject->artist->nationality;

   // convert JSON string into PHP associative array
   $anArray = json_decode($text, true);
   echo $anArray['artist']['nationality'];
?>
```

Notice that the json_decode() function can return either a PHP object or an associative array.

Using JSON in PHP

Go the other way

To go the other direction (i.e., to convert a PHP object into a JSON string), you can use the json_encode() function.



```
// convert PHP object into a JSON string
$text = json_encode($anObject);
```

Section 4 of 7

OVERVIEW OF WEB SERVICES

Web Services

An overview

❑Web services are the most common example of a computing paradigm commonly referred to as **service-oriented computing** (SOC).

❑A **service** is a piece of software with a platform-independent interface that can be dynamically located and invoked.

❑**Web services** are a relatively standardized mechanism by which one software application can connect to and communicate with another software application using web protocols.

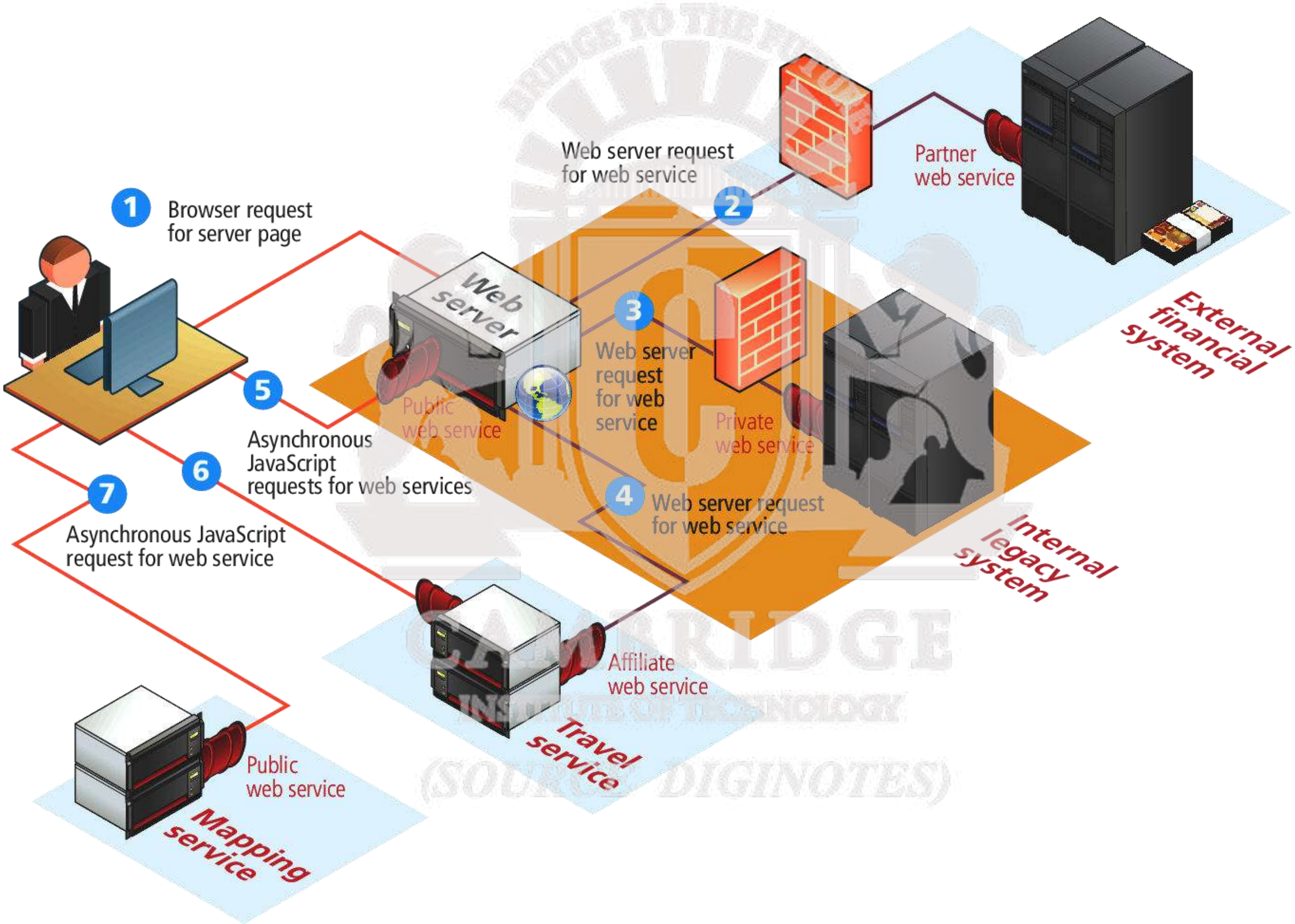
Web Services

Benefits

- they can provide interoperability between different software applications running on different platforms
- they can be used to implement a **service-oriented architecture** (SOA)
- they can be offered by different systems within an organization as well as by different organizations

Web Services

Visual Overview



1 Browser request for server page
2 Web server request for web service
3 Web server request for web service
4 Web server request for web service
5 Asynchronous JavaScript requests for web services
6
7 Asynchronous JavaScript request for web service
Web server
Public web service
Partner web service
Private web service
Affiliate web service
Public web service
External financial system
Internal legacy system
Travel service
Mapping service

Source diginotes.in

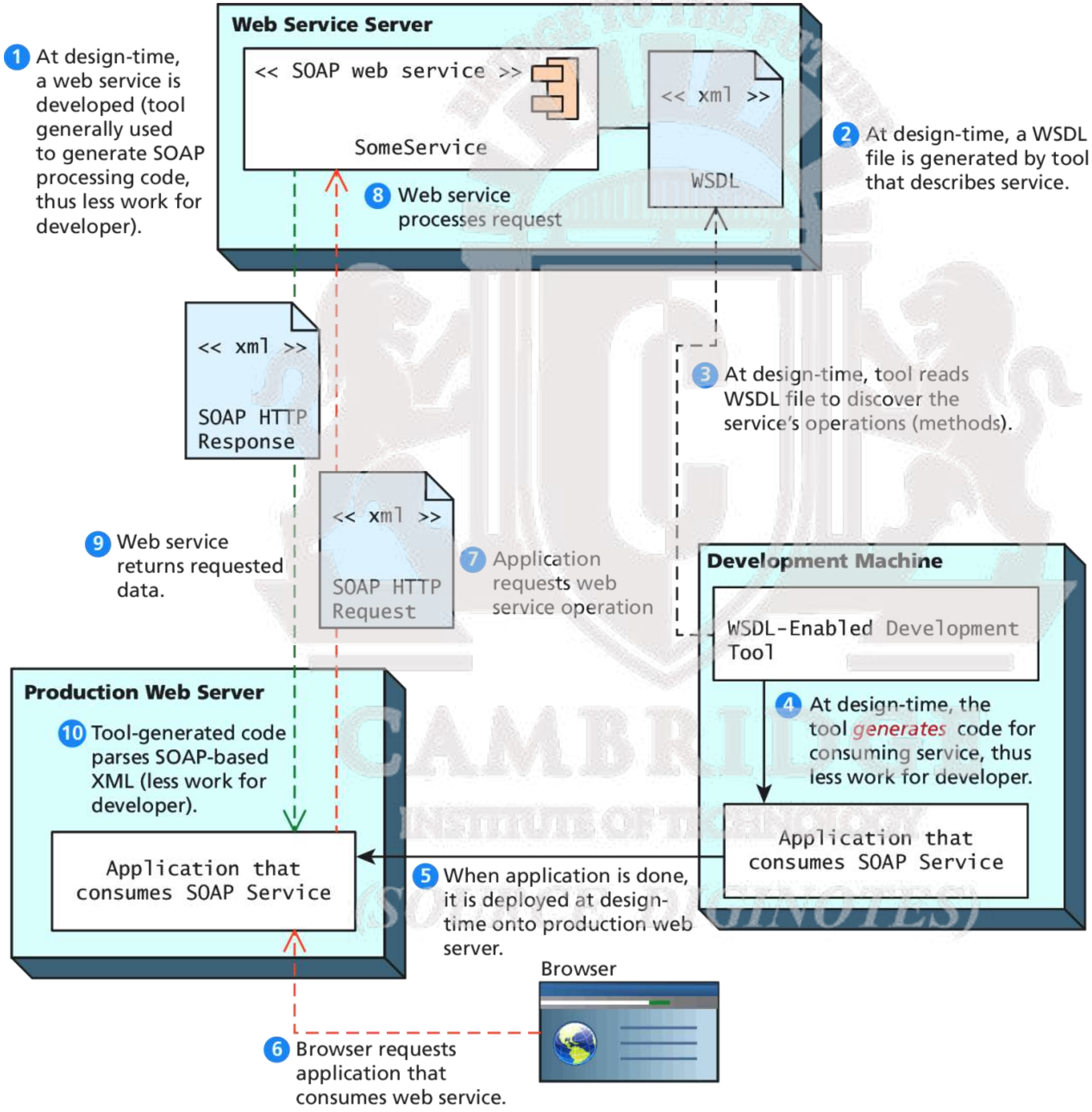
Web Services
SOAP Services

SOAP is the message protocol used to encode the service invocations and their return values via XML within the HTTP header.

- SOAP and WSDL are complex XML schemas
- akin to using a compiler: its output may be complicated to understand
- the enthusiasm for SOAP-based web services had cooled.

Web Services

SOAP Services



Web Service Server

<< SOAP web service >>

SomeService

<< xml >>

WSDL

1 At design-time, a web service is developed (tool generally used to generate SOAP processing code, thus less work for developer).

2 At design-time, a WSDL file is generated by tool that describes service.

3 At design-time, tool reads WSDL file to discover the service's operations (methods).

Development Machine

WSDL-Enabled Development Tool

4 At design-time, the tool *generates* code for consuming service, thus less work for developer.

Application that consumes SOAP Service

5 When application is done, it is deployed at design-time onto production web server.

Production Web Server

Application that consumes SOAP Service

Browser

6 Browser requests application that consumes web service.

<< xml >>

SOAP HTTP Request

7 Application requests web service operation

8 Web service processes request

<< xml >>

SOAP HTTP Response

9 Web service returns requested data.

10 Tool-generated code parses SOAP-based XML (less work for developer).

Web Services

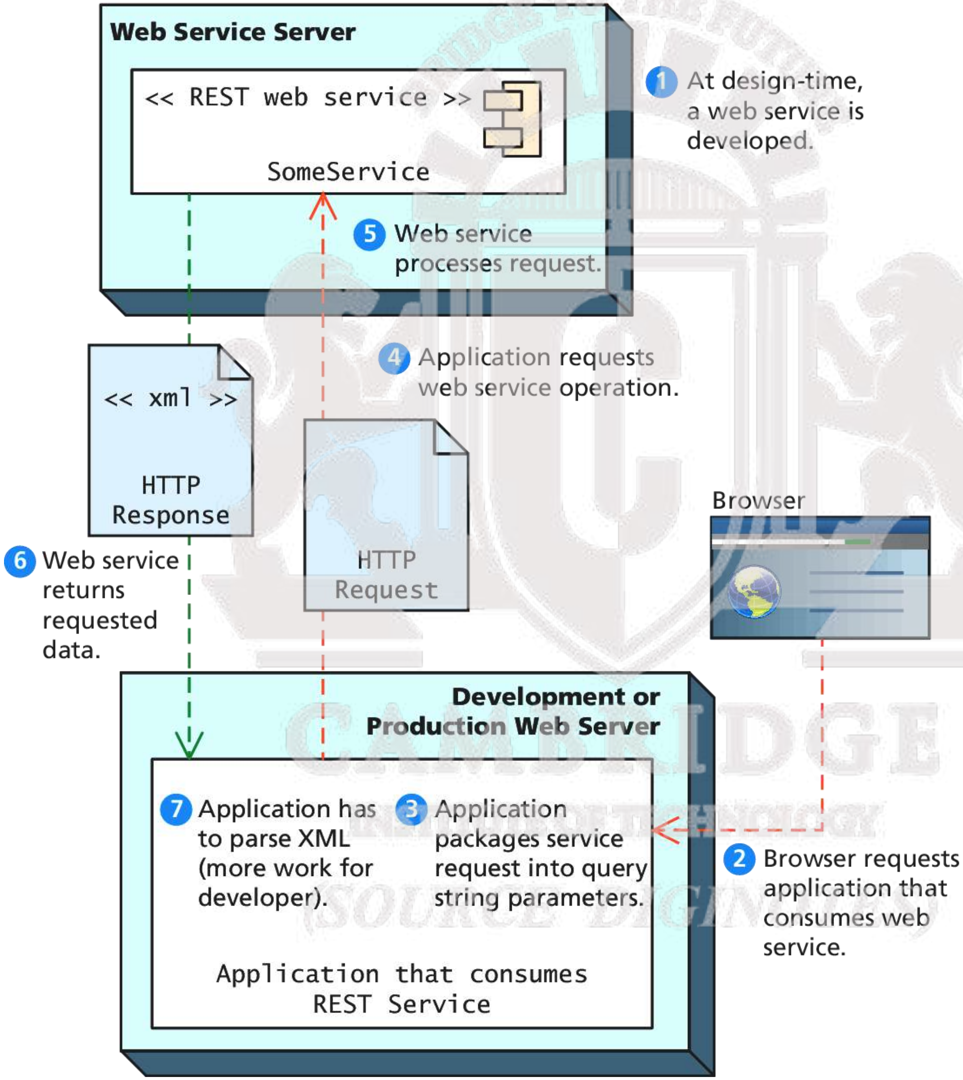
REST Services (an alternate to SOAP)

REST stands for Representational State Transfer.

- RESTful web service does away with the service description layer, and needs no separate protocol for encoding message requests and responses.
- It simply uses HTTP URLs for requesting a resource/object (and for encoding input parameters).
- The serialized representation of this object, usually an XML or JSON stream, is then returned to the requestor as a normal HTTP response.
- REST appears to have almost completely displaced SOAP services.

Web Services

REST Services


Web Service Server
<< REST web service >>
SomeService
1 At design-time, a web service is developed.
5 Web service processes request.
<< xml >>
HTTP Response
4 Application requests web service operation.
HTTP Request
Browser
6 Web service returns requested data.
Development or Production Web Server
7 Application has to parse XML (more work for developer).
3 Application packages service request into query string parameters.
2 Browser requests application that consumes web service.
Application that consumes REST Service

Source diginotes.in

An Example Web Service

We will only use REST from here on in

•Consider the Google Geocoding API.

•The Google Geocoding API provides a way to perform geocoding operations via an HTTP GET request, and thus is an especially useful example of a RESTful web service.

•**Geocoding** typically refers to the process of turning a real-world address into geographic coordinates, which are usually latitude and longitude values

•**Reverse geocoding** is the process of converting geographic coordinates into a human-readable address.

An Example Web Service

More details

In this case the request will take the following form:

http://maps.googleapis.com/maps/api/geocode/xml?***address***

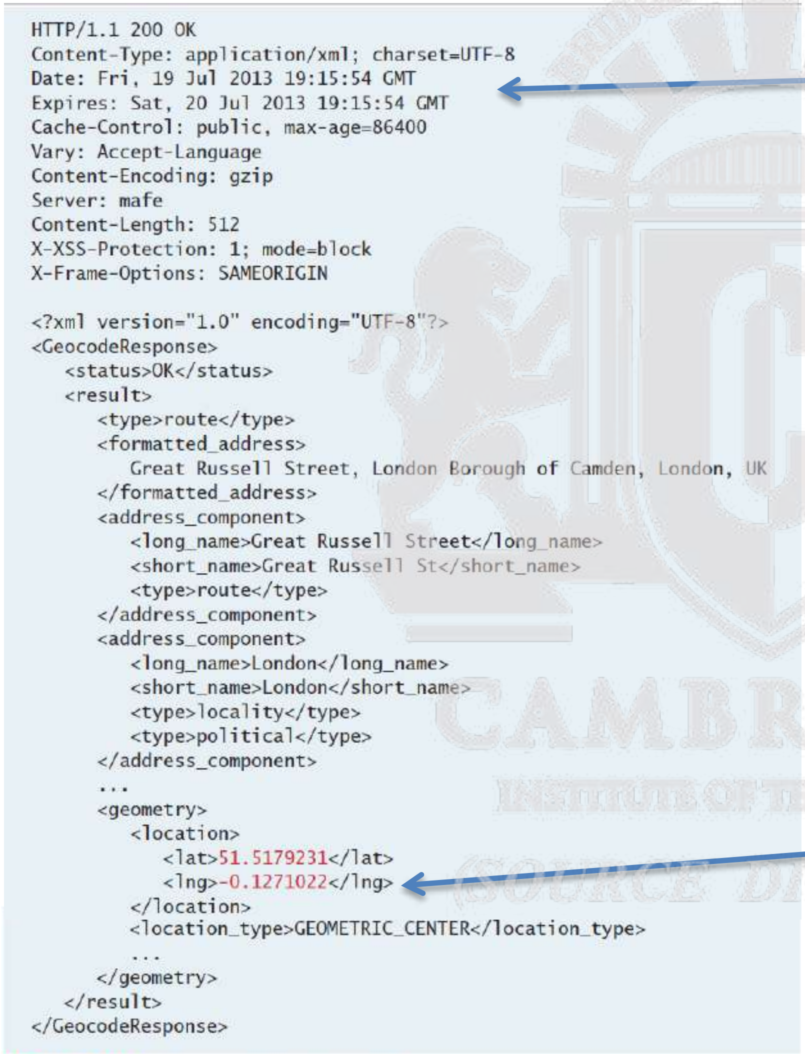
An example geocode request would look like the following:

http://maps.googleapis.com/maps/api/geocode/xml?address=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG&sensor=false

From trendsmap.com

An Example Web Service

The Response



```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8
Date: Fri, 19 Jul 2013 19:15:54 GMT
Expires: Sat, 20 Jul 2013 19:15:54 GMT
Cache-Control: public, max-age=86400
Vary: Accept-Language
Content-Encoding: gzip
Server: mafe
Content-Length: 512
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN

<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
   <status>OK</status>
   <result>
      <type>route</type>
      <formatted_address>
         Great Russell Street, London Borough of Camden, London, UK
      </formatted_address>
      <address_component>
         <long_name>Great Russell Street</long_name>
         <short_name>Great Russell St</short_name>
         <type>route</type>
      </address_component>
      <address_component>
         <long_name>London</long_name>
         <short_name>London</short_name>
         <type>locality</type>
         <type>political</type>
      </address_component>
      ...
      <geometry>
         <location>
            <lat>51.5179231</lat>
            <lng>-0.1271022</lng>
         </location>
         <location_type>GEOMETRIC_CENTER</location_type>
         ...
      </geometry>
   </result>
</GeocodeResponse>
```

LISTING 17.13 HTTP response from web service

The response is a standard HTTP response with headers

This response is XML

The lat/lng is in there somewhere

Identifying and Authenticating Service Requests

Most web services are not open. Instead they typically employ one of the following techniques:

- **Identity**. Each web service request must identify who is making the request.
- **Authentication**. Each web service request must provide additional evidence that they are who they say they are.

Identity examples

Real World ways of limiting service

- Web services that make use of an API key typically require the user (i.e., the developer) to register online with the service for an API key. This API key is then added to the GET request as a query string parameter.

- For instance, to request to the Microsoft Bing Maps web service will look like the following :

- http://dev.virtualearth.net/REST/v1/Locations?o=xml&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&**key**=**[*BING API KEY HERE*]**

Authentication

Real World ways of limiting service

❑Some web services are providing private/proprietary information or are involving financial transactions.

❑In this case, these services not only may require an API key, but they also require some type of user name and password in order to perform an authorization.

❑Many of the most well-known web services instead make use of the OAuth standard.

END OF MODULE 5