Introduction to Web Systems and Technologies

TOPIC 01
MODULE 01

Prepared by DL...
May You All Stay Success in your Whole Life, Ameen! Summ Ameen... Just Remember us in your Deep Prayers. JazakAllah...

Web Systems and Technologies

Introduction to Web Systems and Technologies

Introduction to Web Systems and Technologies

Topics (Basics)

1. What is web and how does it work?
2. Structuring web contents using HTML
3. Setting visual layout using CSS
4. Making web interactive using JavaScript
5. Web backend programming using PHP and databases

Introduction to Web Systems and Technologies

Topics (Advanced)

6. Making websites secure
7. Introduction to web services
8. Front-end and back-end frameworks
9. Hosting websites
10. Search engine optimization (SEO)

Introduction to Web Systems and Technologies

Textbook

1. "Fundamentals of Web Development", 2nd Edition by Randy Connolly and Ricardo Hoar, 2018, Pearson

Reference websites

1. https://www.w3schools.com/
2. https://developer.mozilla.org/en-US/
3. https://reactjs.org/

Introduction to Web Systems and Technologies

Credits

- Material for the slides used in this course is provided by *Pearson Education*

Reference websites

1. https://www.w3schools.com/
2. https://developer.mozilla.org/en-US/
3. https://reactjs.org/

Web Systems and Technologies

Module 02

TOPIC 02

Ecosystem of Web

Web Systems and Technologies

TOPIC 02

Ecosystem of Web

Ecosystem of Web

A Complicated Ecosystem



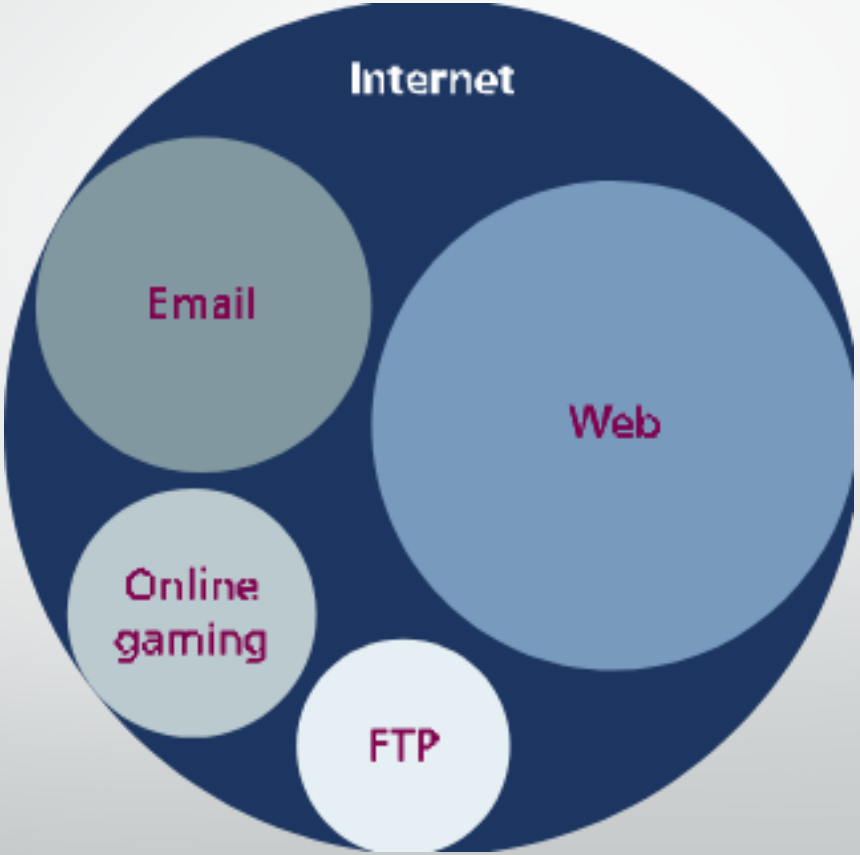
Ecosystem of Web

A Short History of the Internet

- Telephone Network
- Packet Networks
  - ARPANET (1969)
  - X.25 (1974)
  - USENET (1979)
  - TCP/IP (1983) ← INTERNET

Ecosystem of Web

Internet vs. (World Wide) Web



Ecosystem of Web

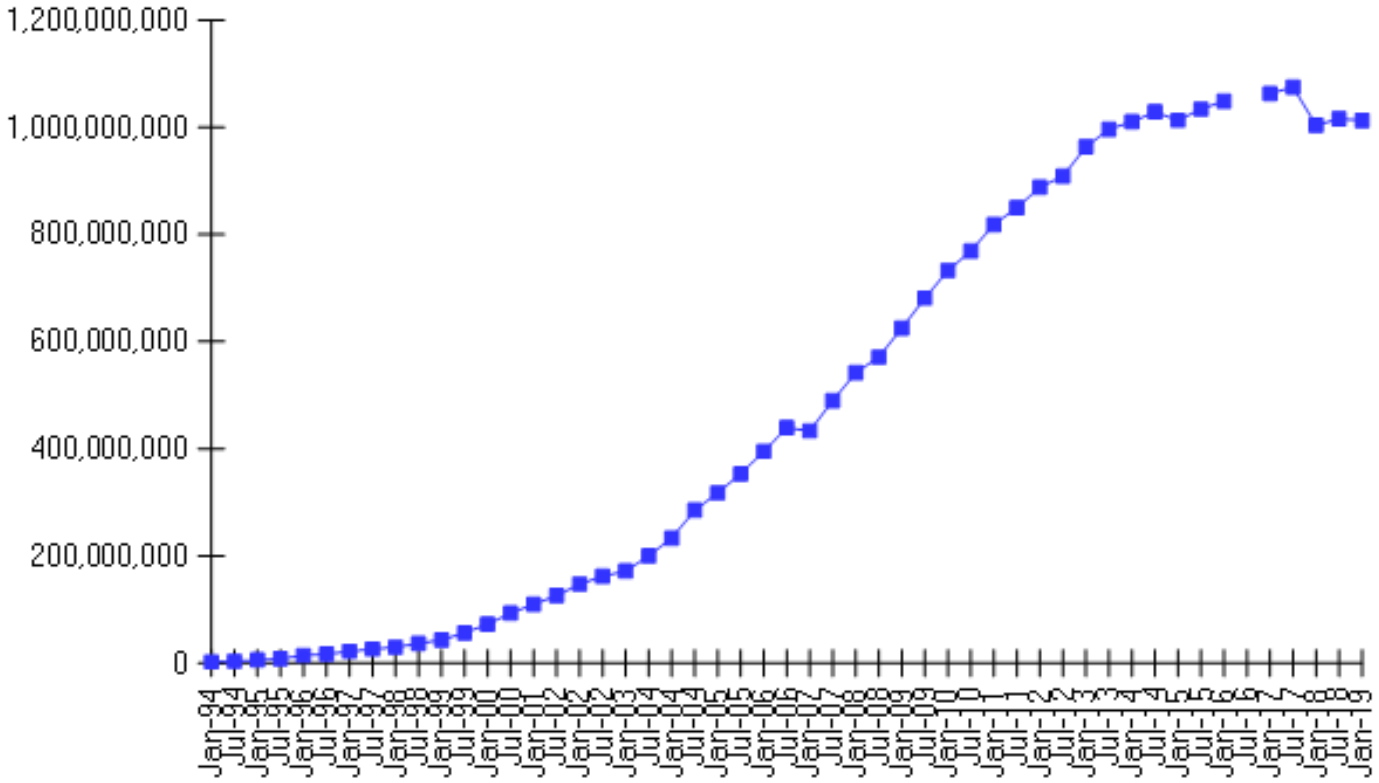
The Birth of the Web (1990)

- URLS (Uniform Resource Locator)
- HTTP (Hyper Text Transfer Protocol) used to transfer data on the web.
- SERVERS
- BROWSERS
- HTML (Hyper Text Markup Language) for creating web pages.

Ecosystem of Web

Growth of Internet

- Number of Internet Hosts



Source: Internet Systems Consortium (www.isc.org)

Ecosystem of Web

Web Apps vs. Desktop Apps

- Advantages
  - Accessible from any Internet-enabled computer
  - Cross platform and browser compatibility
  - Easier to roll out program updates
  - Fewer security concerns about local storage

Ecosystem of Web

Web Apps vs. Desktop Apps

- Disadvantages
  - Requires internet
  - Security concerns about sensitive private data being transmitted over the Internet
  - Concerns over the storage, licensing, and use of uploaded data
  - Problems with certain websites not having an identical appearance across all browsers
  - Restrictions on access to operating system resources
  - May need plugins for functioning

Web Systems and Technologies

TOPIC 03

MODULE 03

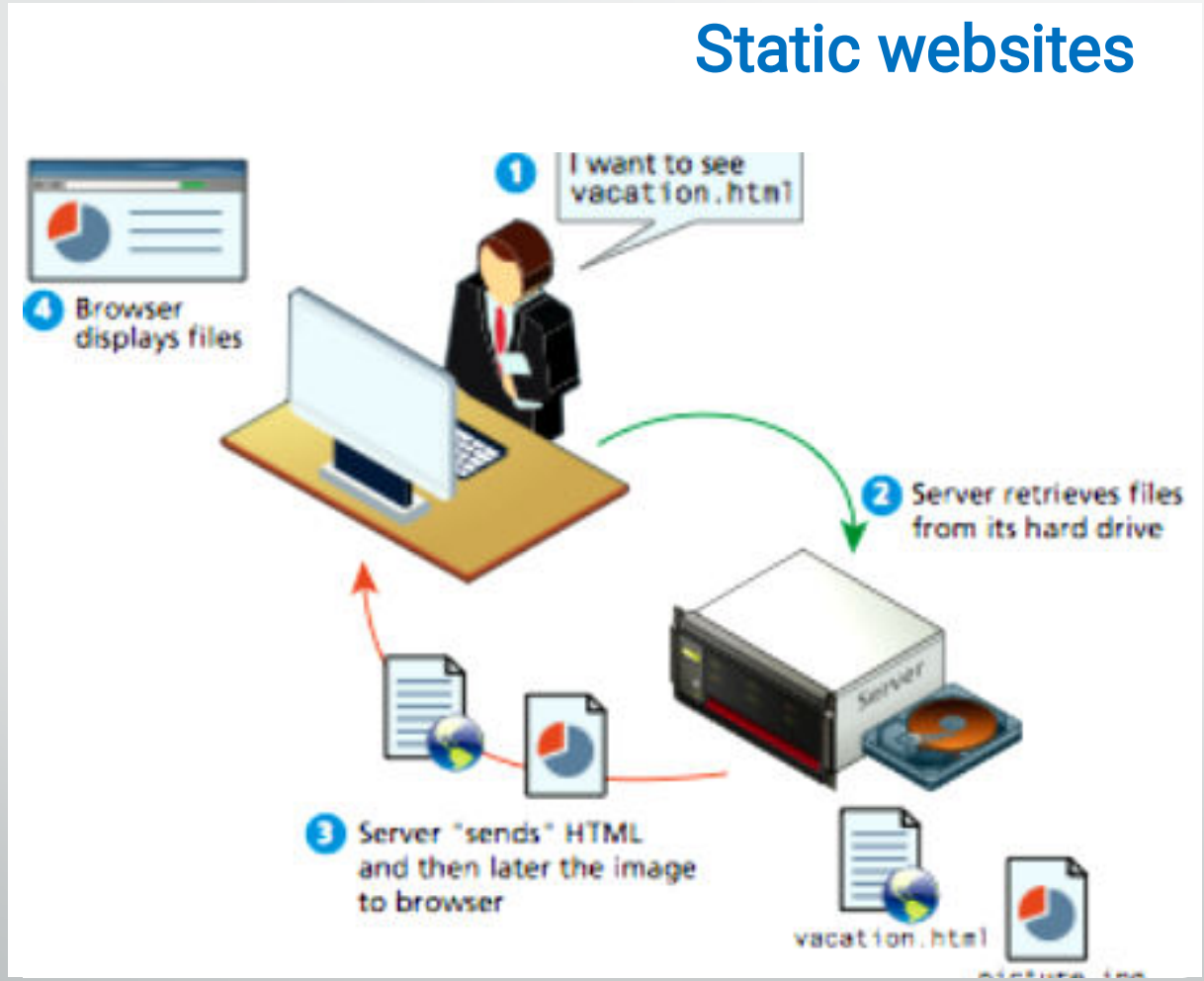
Static vs Dynamic Websites

Web Systems and Technologies

Static vs Dynamic Websites

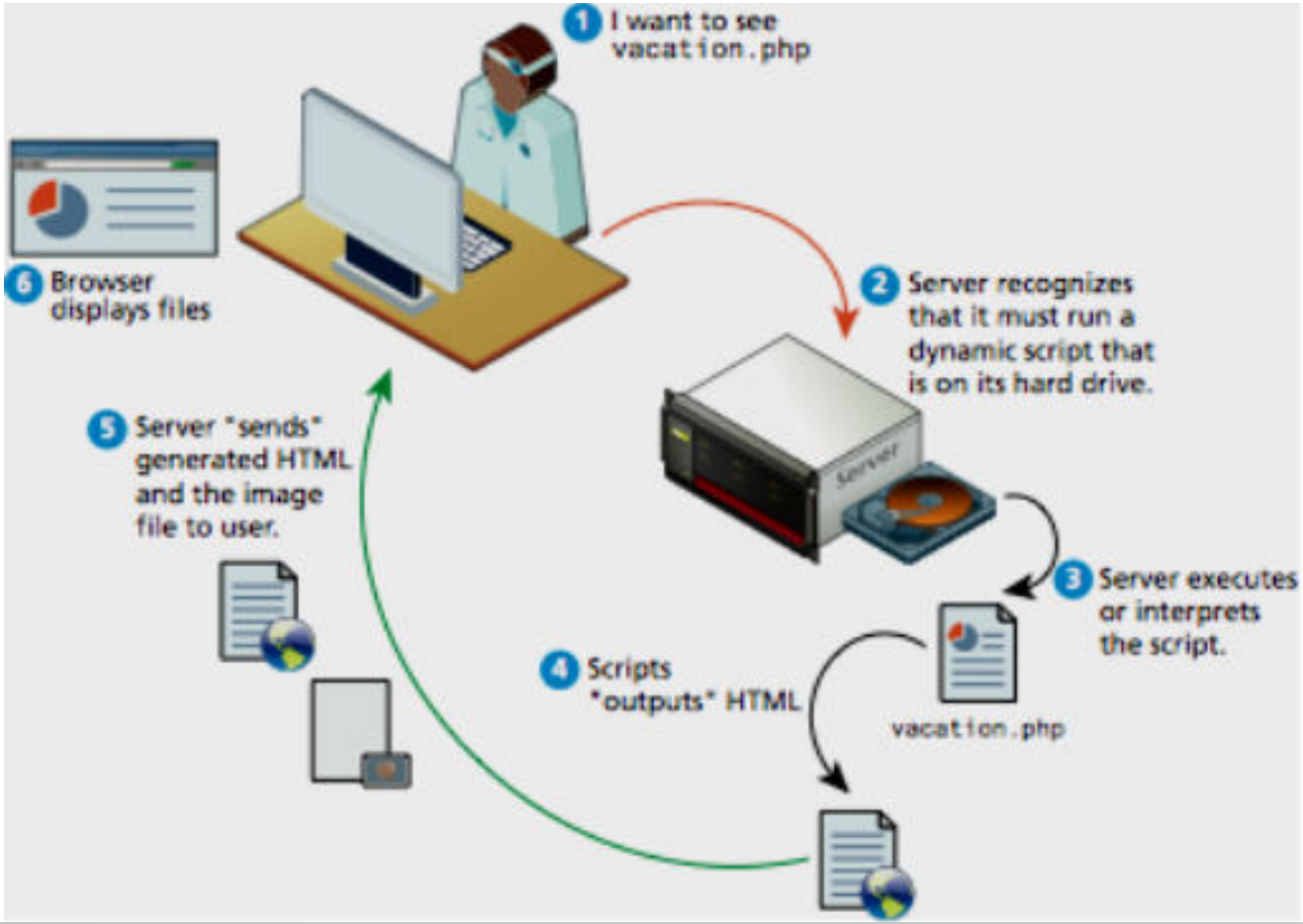
Static vs Dynamic Websites

Static websites



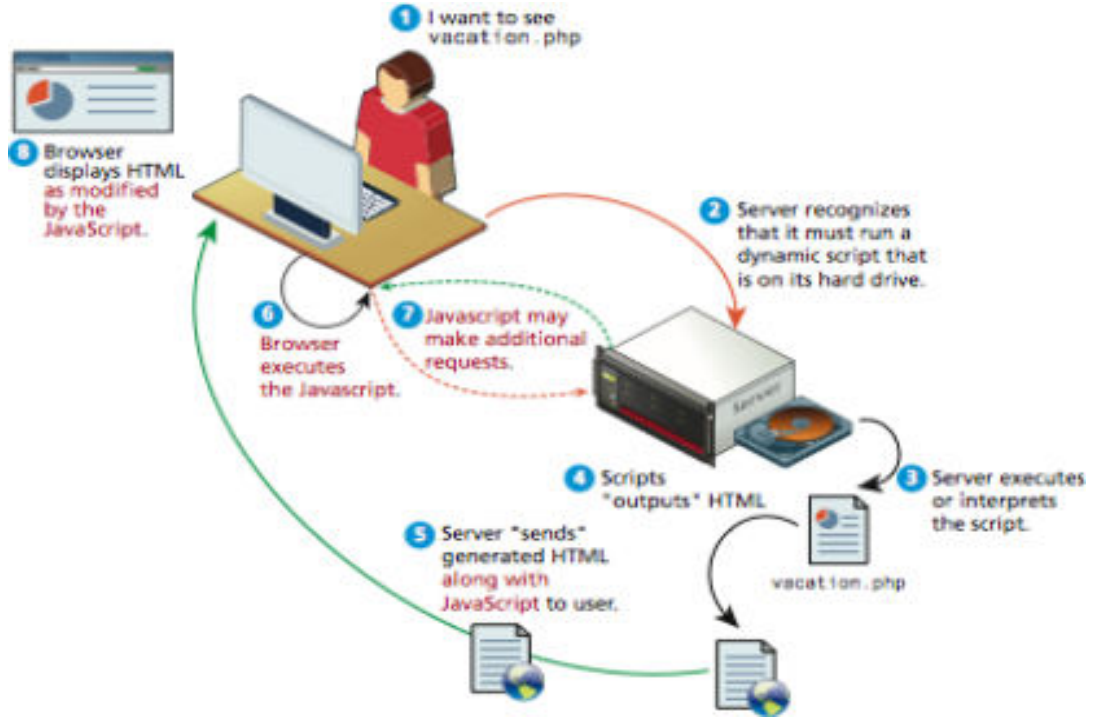
Static vs Dynamic Websites

(Traditional) Dynamic Websites



Static vs Dynamic Websites

(Modern) Dynamic Websites



Web Systems and Technologies

TOPIC 04

MODULE 04

Brief History of Web

Web Systems and Technologies

Brief History of Web

Brief history of Web

Web 1.0

- Contents generated by publishers
- Company generated companies
- Static websites
- Dynamic websites

Brief history of Web

Web 2.0

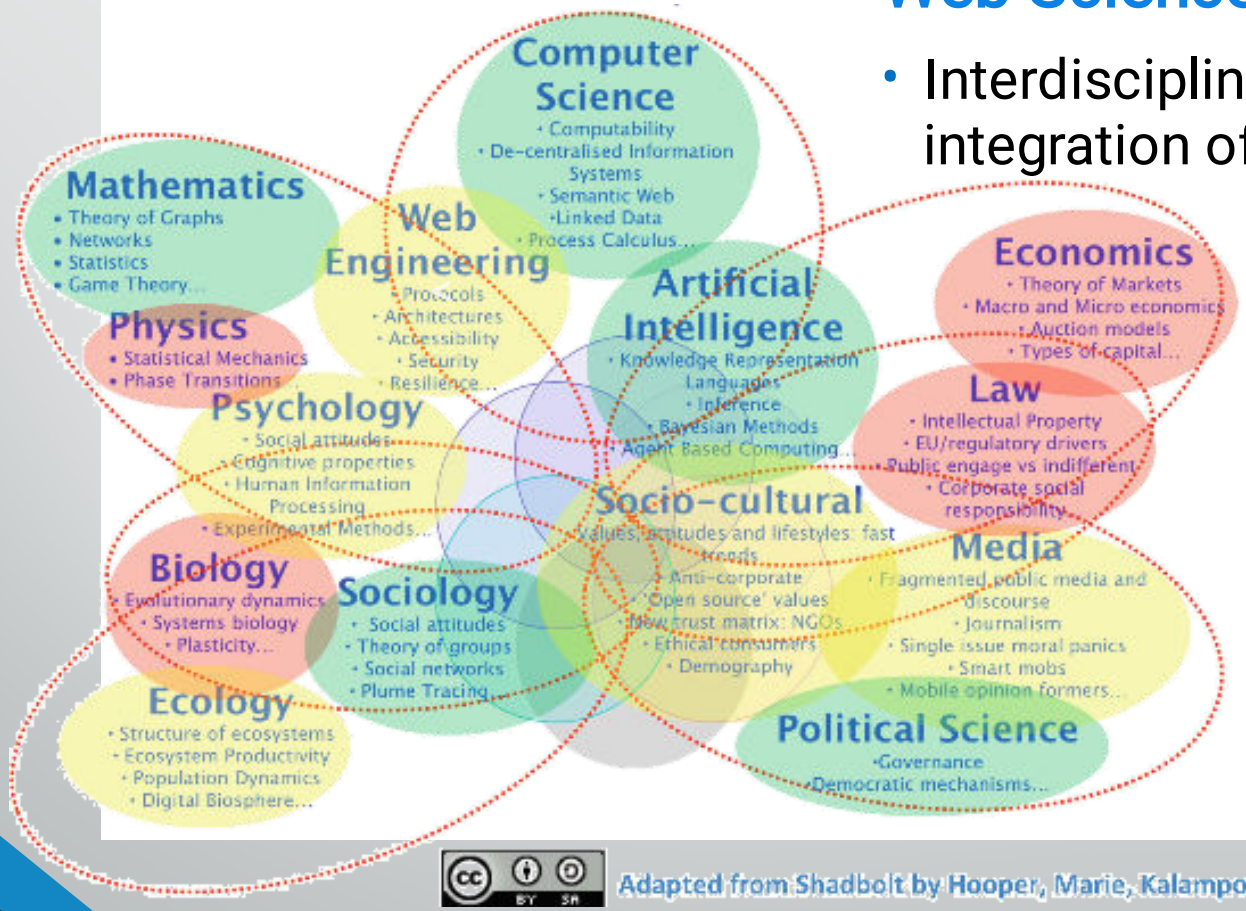
- User Generated Contents (UGC)
- Dynamic websites

Brief history of Web

Web 3.0

- Machine understandable contents
- Semantic web

Brief history of Web



Web Science

- Interdisciplinary study of sociotechnical integration of the world wide web

Adapted from Shadbolt by Hooper, Marie, Kalampokis

Web Systems and Technologies

TOPIC 05

MODULE 05

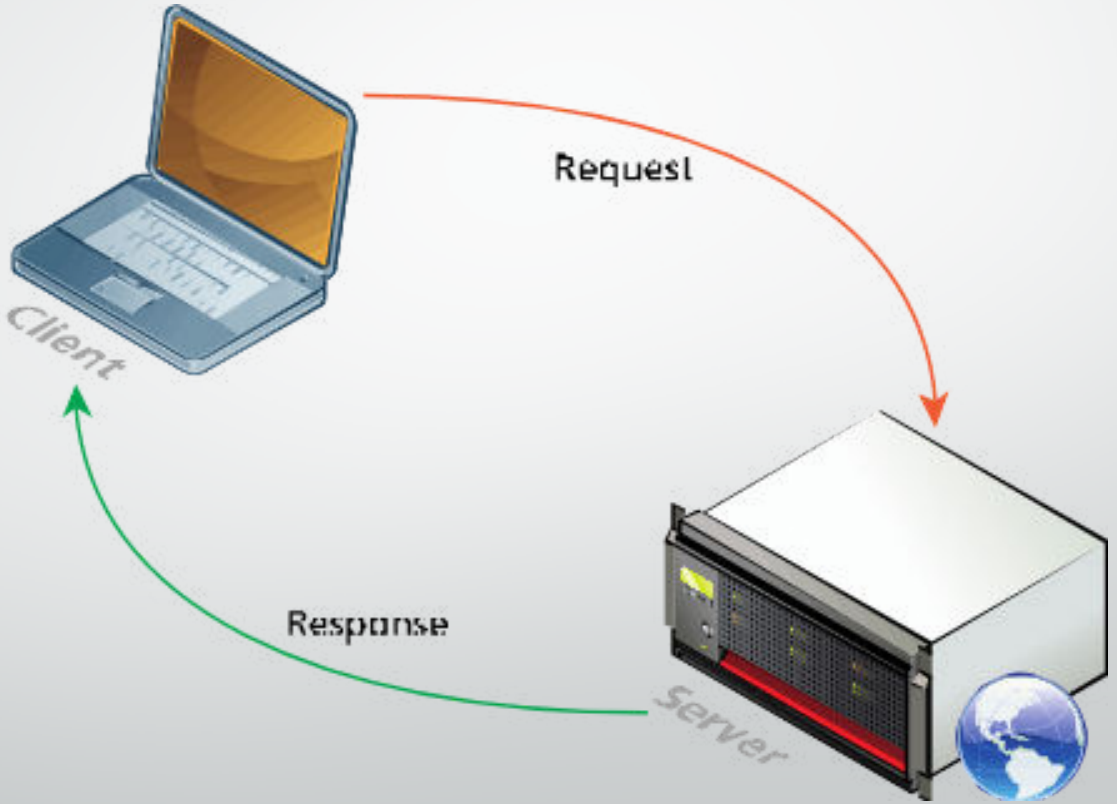
Client Server Model

Web Systems and Technologies

Client Server Model

Client Server Model

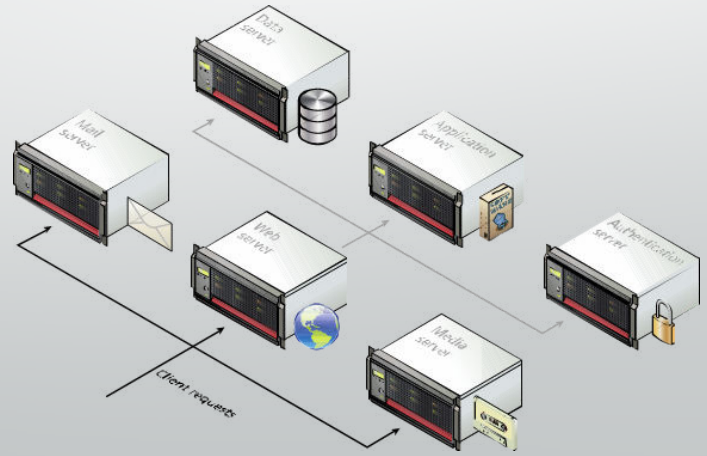
The request-response Loop



Client Server Model

Types of Servers

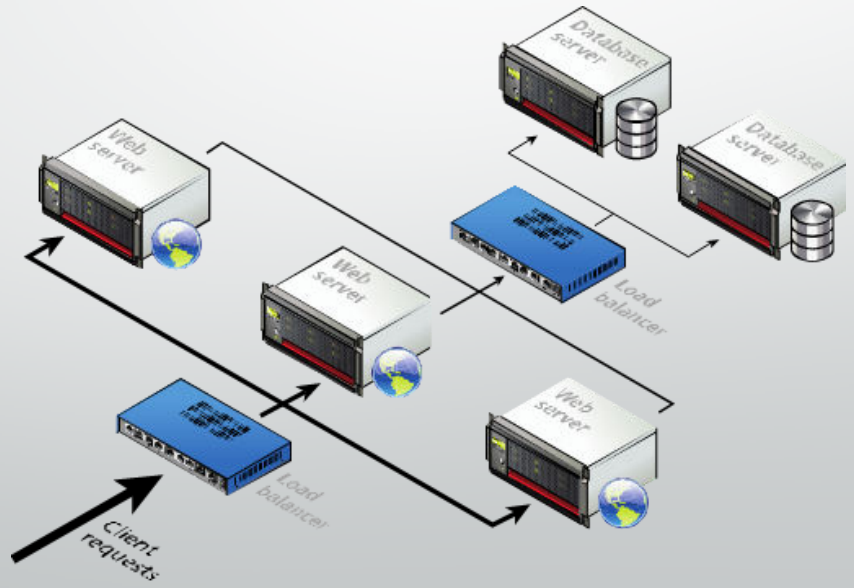
- Web Servers
- Application Servers
- Database Servers
- Mail Servers
- Media Servers
- Authentication Servers
- …



Client Server Model

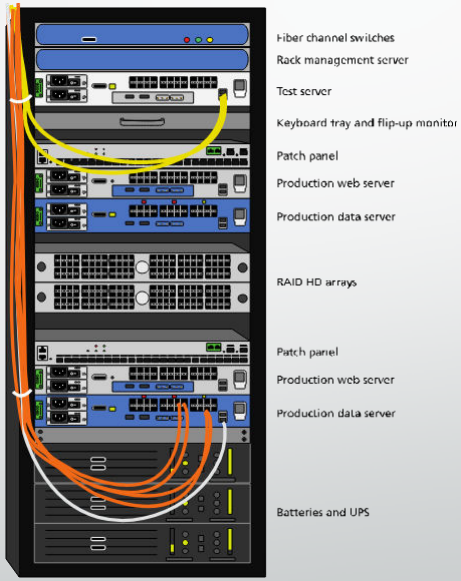
Server Farms

- Large websites receives millions of requests per minute
  - Cannot be handled by a single server
- Server farms are used for scalability



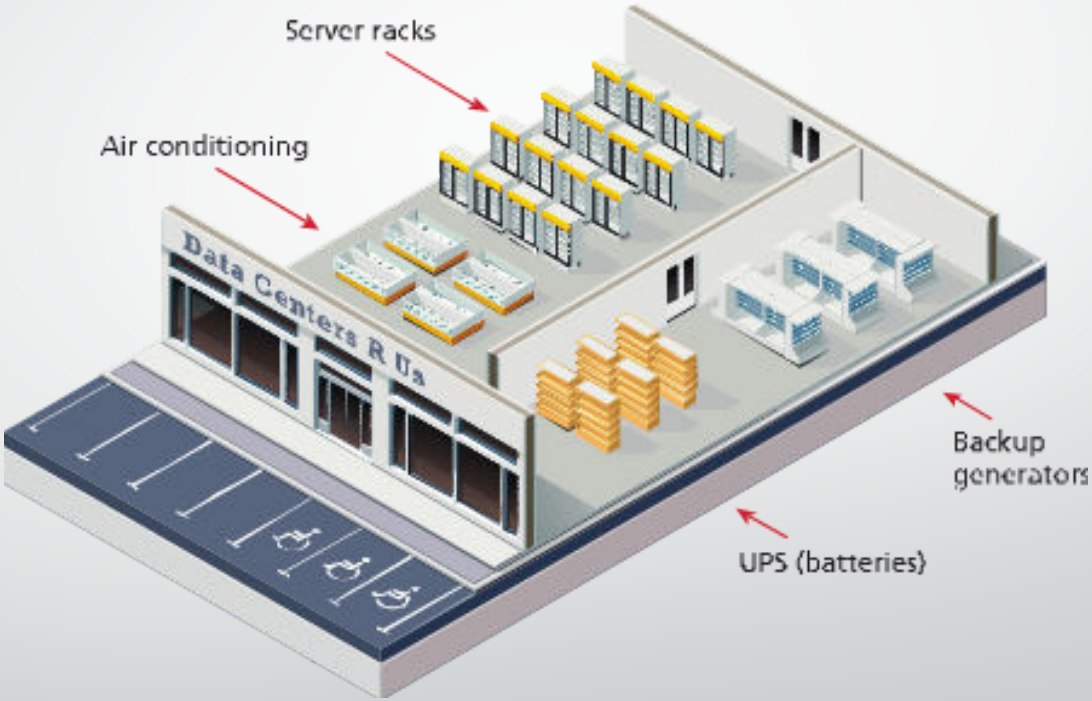
Client Server Model

Server Racks



Client Server Model

Data Centers



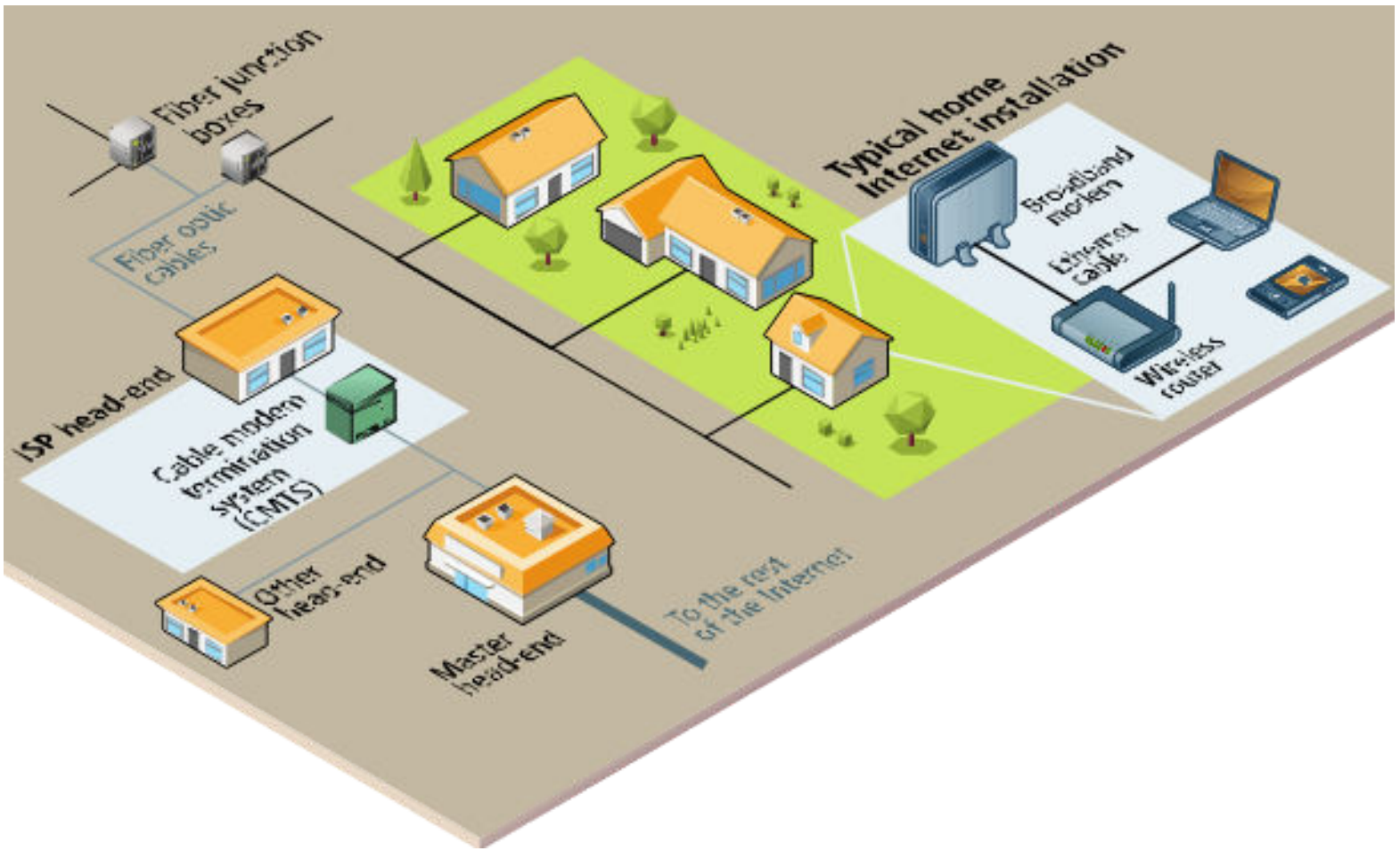
How Internet Works?

Web System and Tecnologies CS 521

TOPIC 06

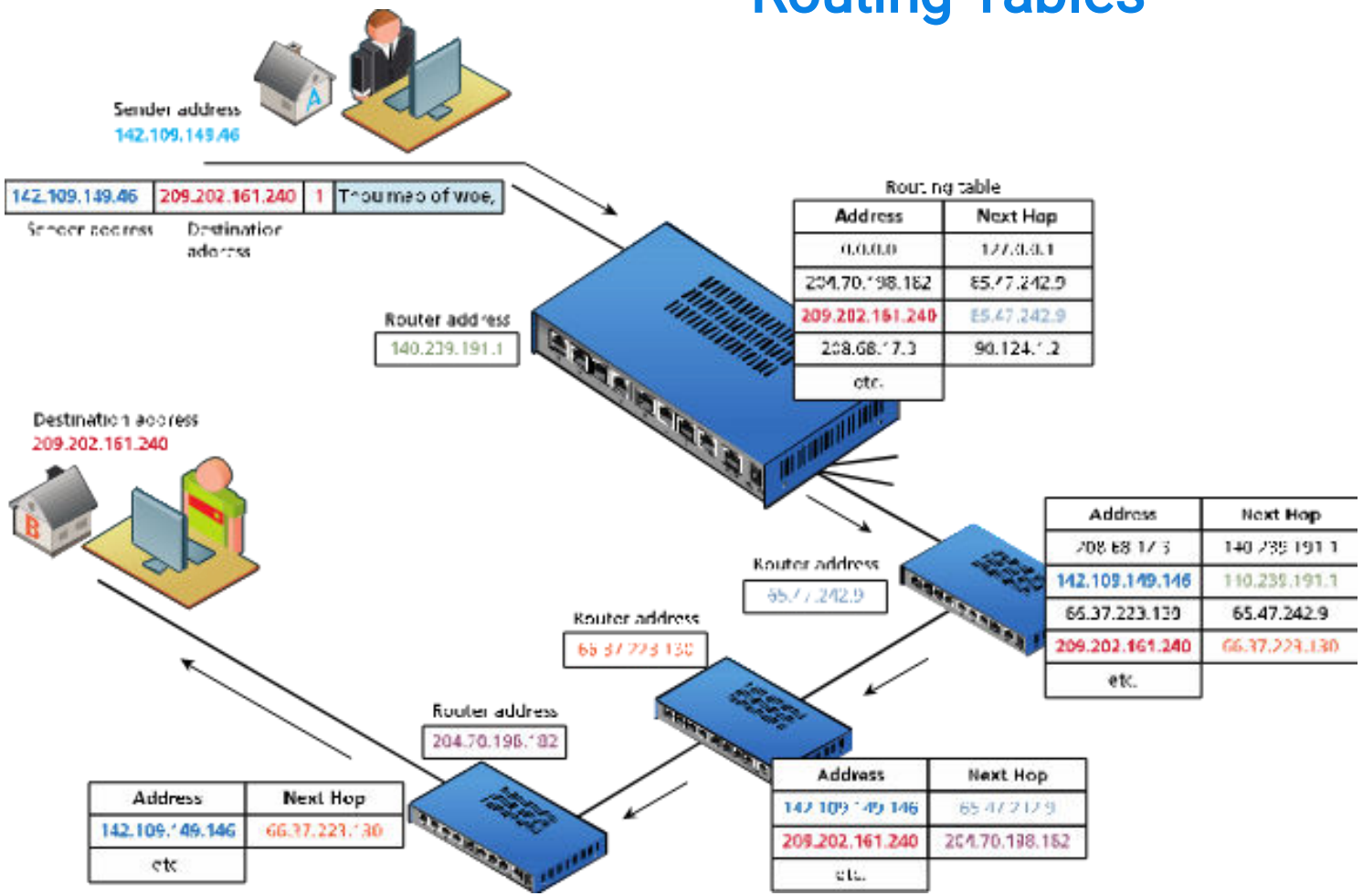
How Internet Works?

Internet from Home to ISP



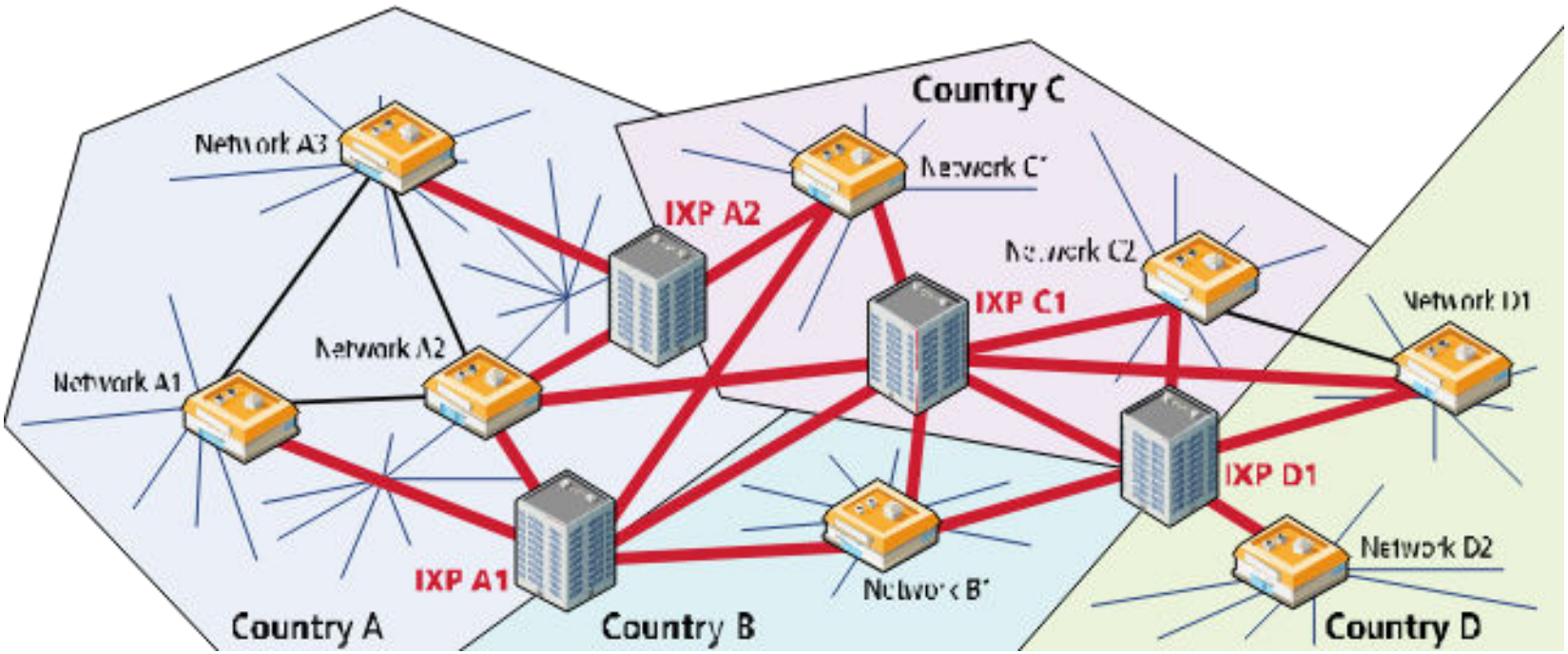
How Internet Works?

Routing Tables



How Internet Works?

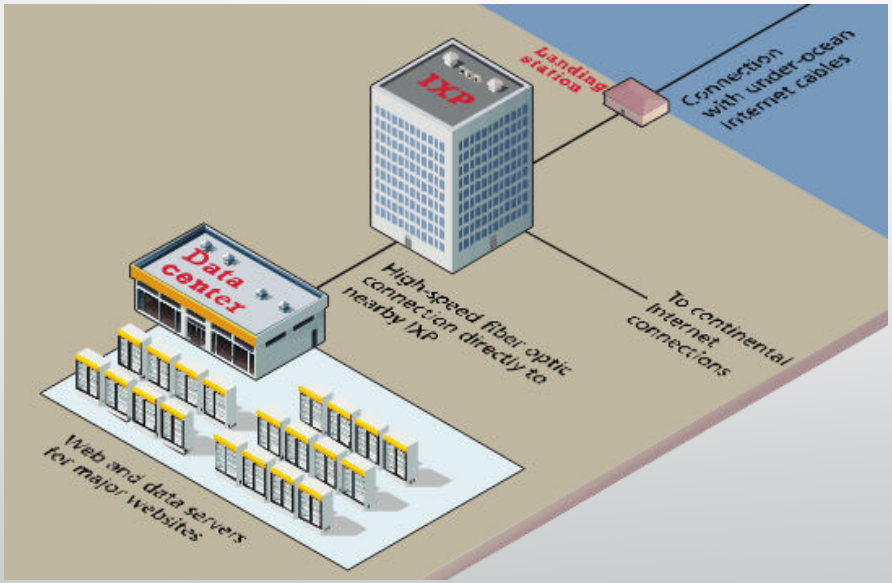
From ISP to the World



How Internet Works?

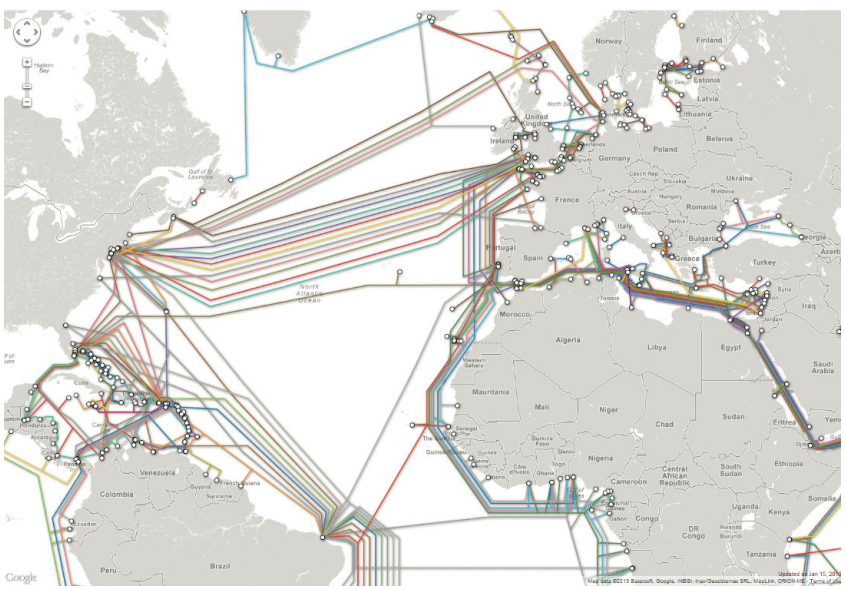
IXP

- IXP – Internet Access Point – Dedicated infrastructure to make the internet more connected.



How Internet Works?

Under-ocean Connectivity



The End

Internet Protocols

Web System and Tecnologies CS 521

Topic 07

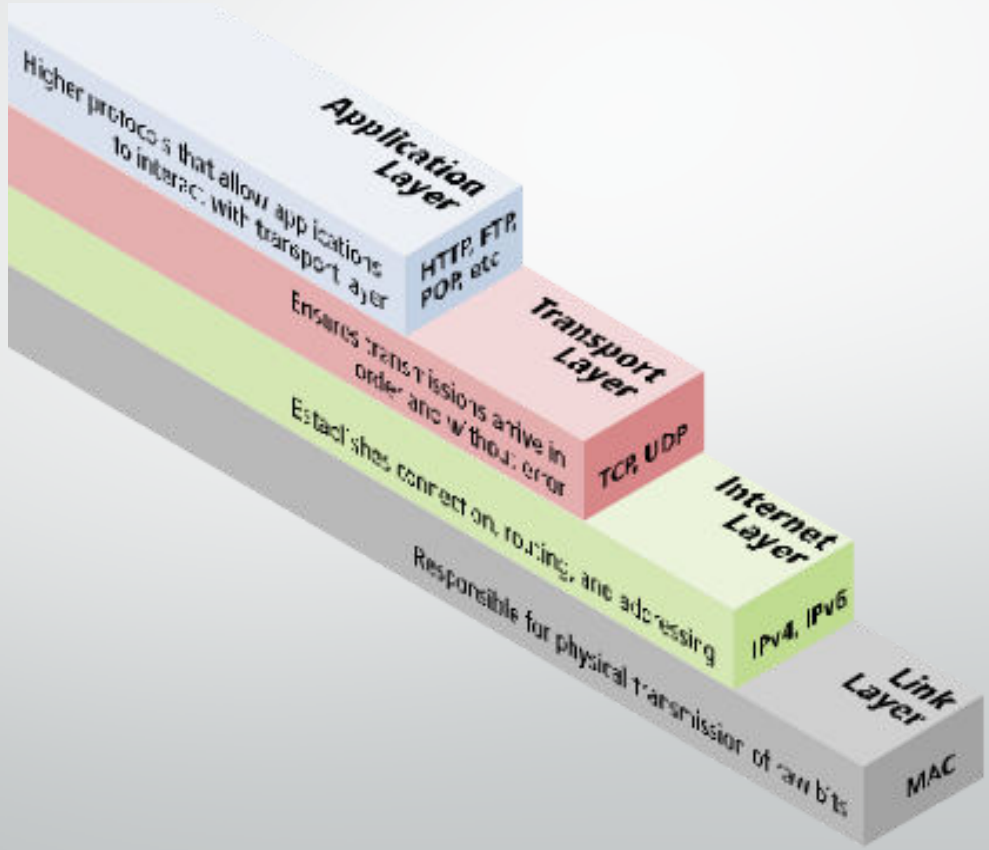
Internet Protocols

TCP/IP

- These protocols have been implemented in every operating system, and make fast web development possible.
- Networking is it’s own entire discipline.
- Web developer needs general awareness of what the suite of Internet protocols does

Internet Protocols

Layered Architecture:



Internet Protocols

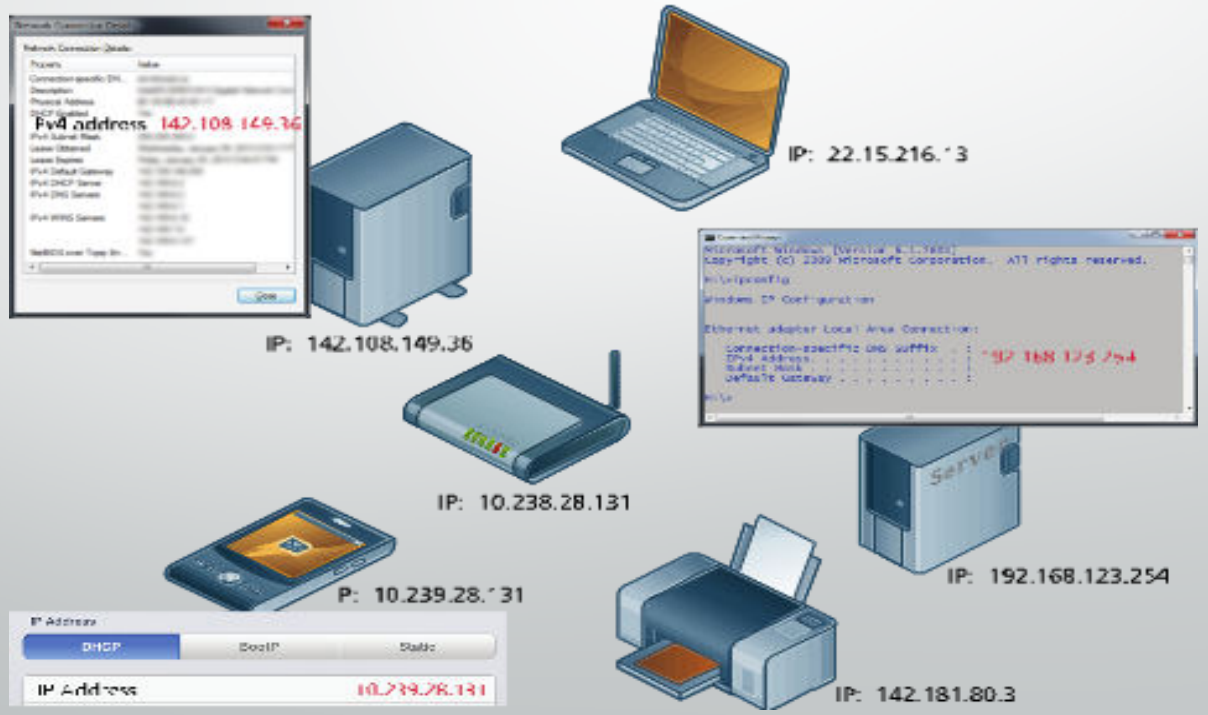
Link Layer

- Physical transmission of data across media (both wired and wireless) and establishing logical links.
- Handles issues like packet creation, transmission, reception, error detection, collisions, line sharing, and more.

Internet Protocols

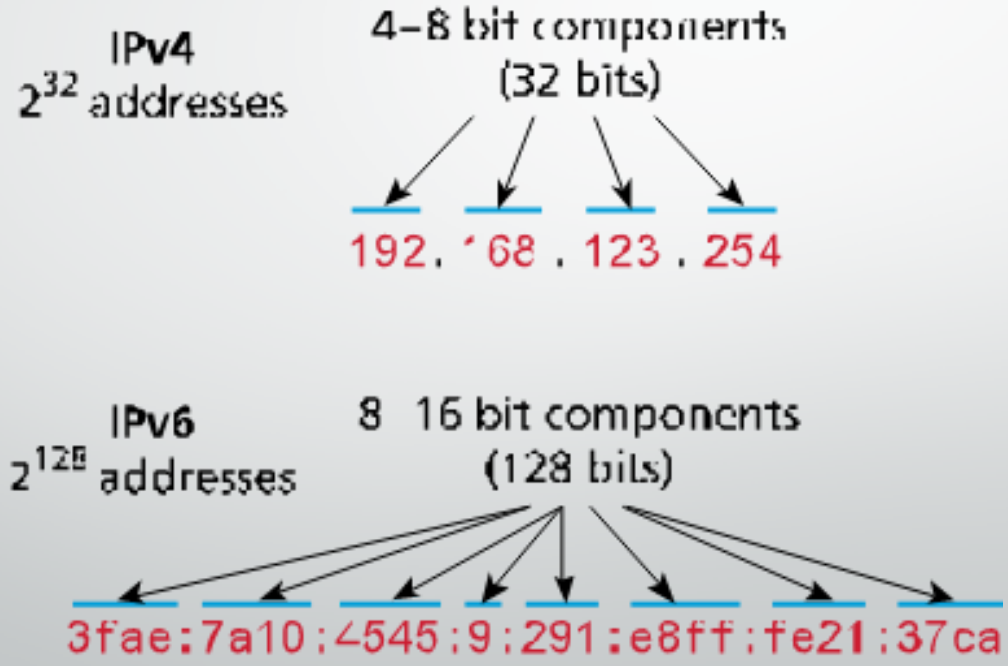
Internet Layer:

- The Internet layer provides “best effort” communication.
- Makes use of IP addresses



Internet Protocols

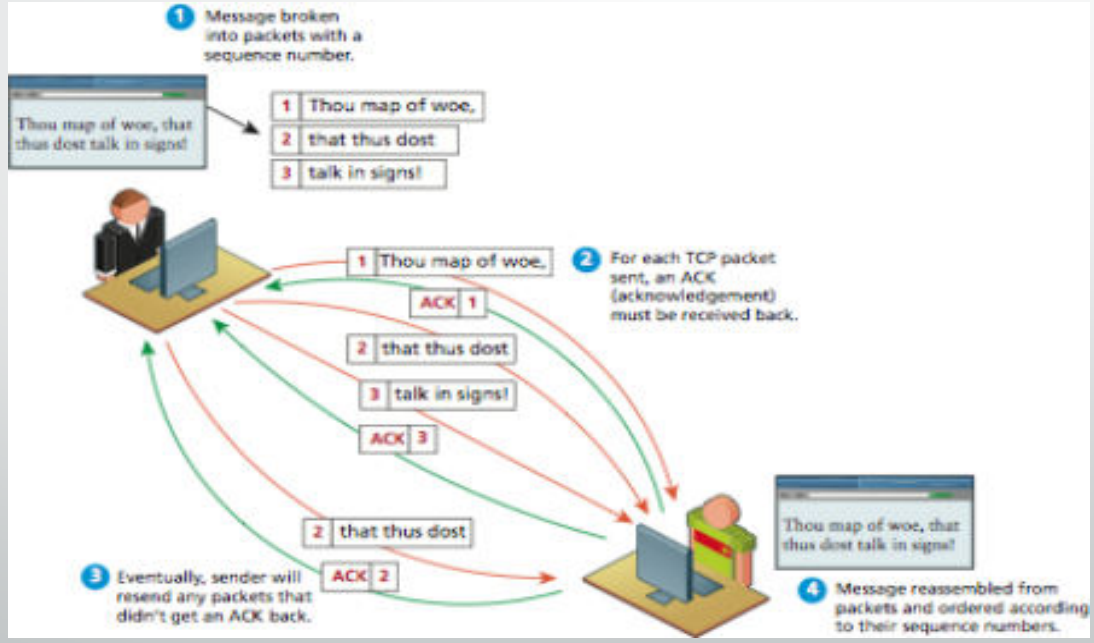
IP-Address:



Internet Protocols

Transport Layer:

- Ensures transmissions arrive in order and without error



Internet Protocols

Application Layer Protocols:

- HTTP. The Hypertext Transfer Protocol is used for web communication.
- SSH. The Secure Shell Protocol allows remote command-line connections to servers.

Internet Protocols

Application Layer Protocols:

- FTP. The File Transfer Protocol is used for transferring files between computers.
- POP/IMAP/SMTP. Email-related protocols for transferring and storing email.
- DNS. The Domain Name System protocol used for resolving domain names to IP addresses.

The End

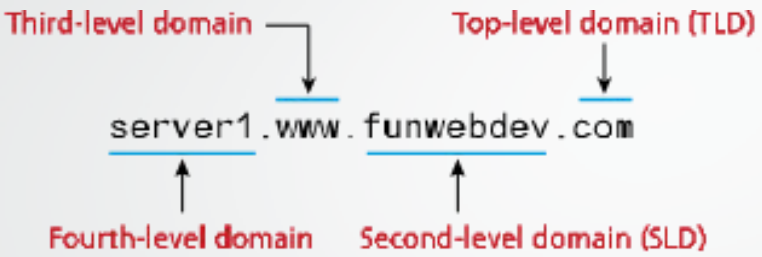
Domain Name System

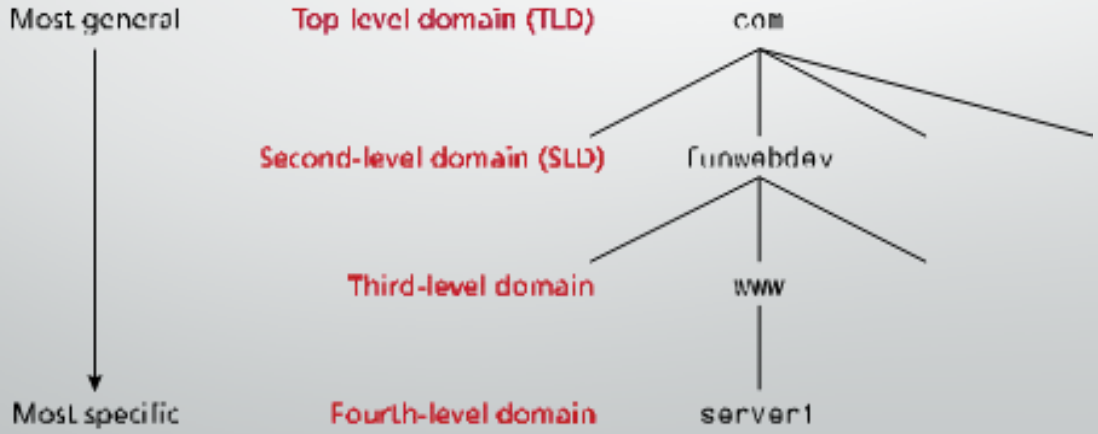
Web System and Tecnologies CS 521

TOPIC 08

Domain Name System

Name Levels in a Domain:





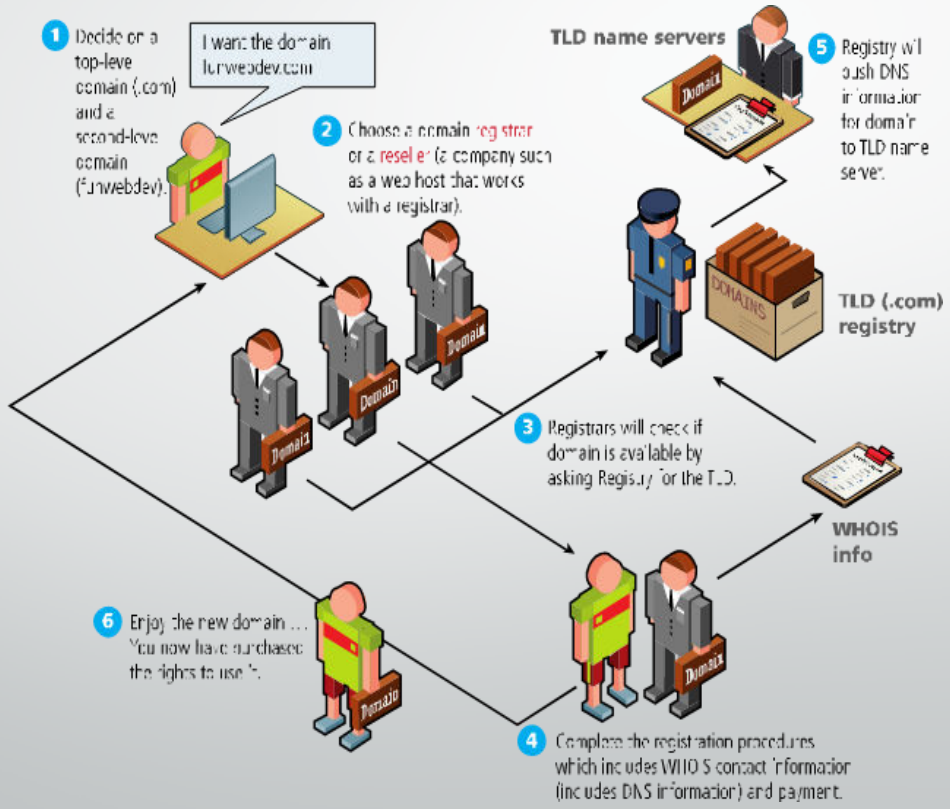
Domain Name System

Types of Top Level Domains:

- Generic top-level domain (gTLD)
  - Unrestricted. TLDs include .com , .net, .org, and .info.
  - Sponsored. TLDs including .gov , .mil, .edu, and others.
  - New TLDs.
- Country code top-level domain (ccTLD)
  - TLDs include .pk , .ca , .uk , .au. , …
  - Internationalized Domain Names

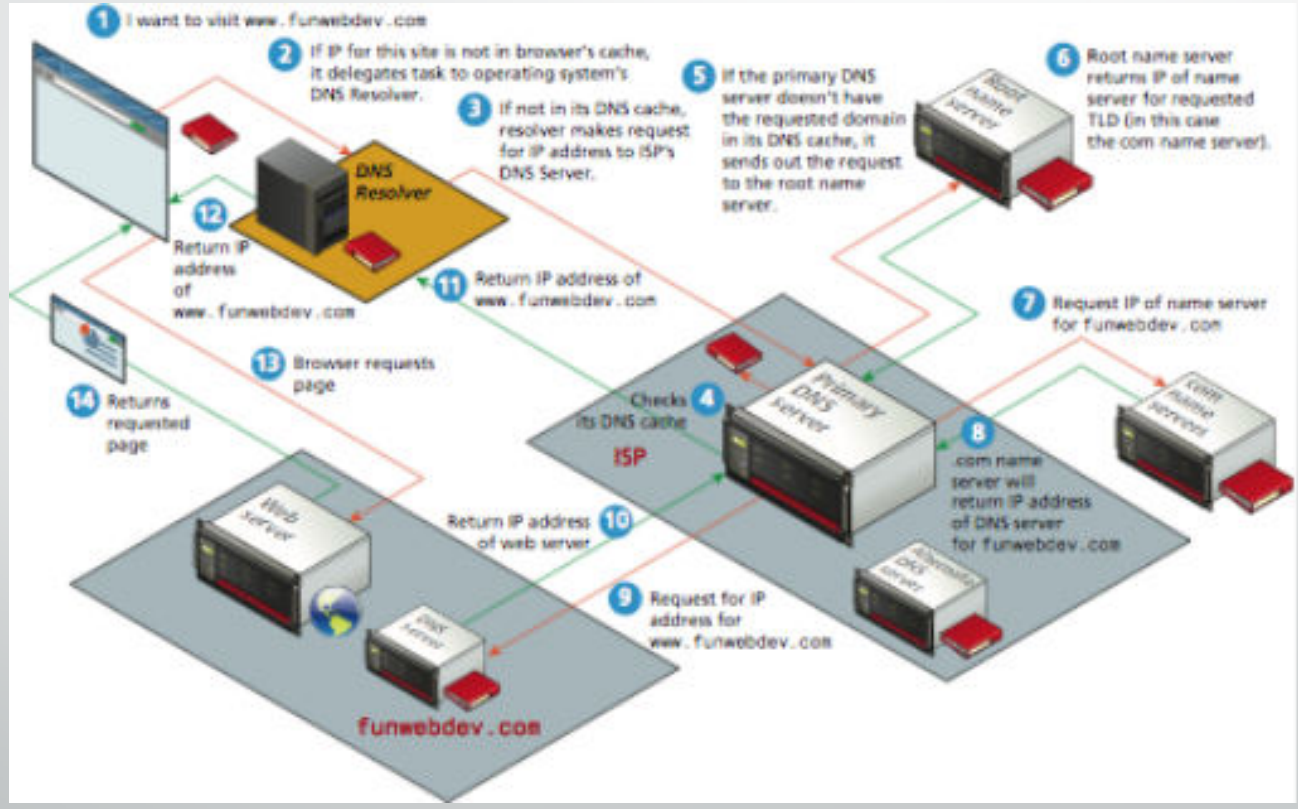
Domain Name System

Domain Registration:



Domain Name System

Address Resolution:



The End

Web System and Technologies CS 521

TOPIC 09

Uniform Resource Locators

Uniform Resource Locators

Components of a URL:

- Protocol
- Domain
- Path
- Query String
- Fragment

Uniform Resource Locators

Protocols:

- **ftp**://example.com/abc.txt → sends out an FTP request on port 21, while
- **http**://example.com/abc.txt → transmits an HTTP request on port 80



Uniform Resource Locators

Domain:

The domain identifies the server from which we are requesting resources.

- Since the DNS system is case insensitive, this part of the URL is case insensitive.
- Alternatively, an IP address can be used for the domain



Uniform Resource Locators

Port :

- The optional port attribute allows us to specify connections to ports other than the defaults
- Add a colon after the domain, then specify an integer port number.



Uniform Resource Locators

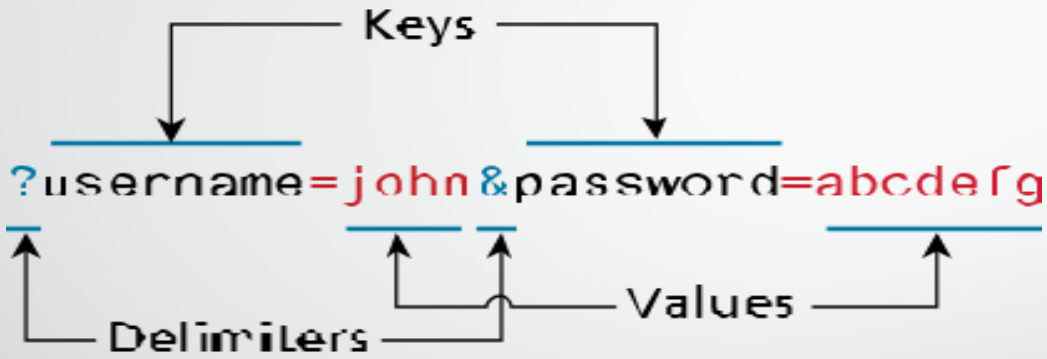
Path :

- Familiar concept to anyone who has ever used a computer file system.
- The root of a web server corresponds to a folder somewhere on that server.
  - On many Linux servers that path is /var/www/html/
  - On Windows IIS machines it is often c:/inetpub/wwwroot/
- The path is optional
  - If not provided, server decides which file to send



Uniform Resource Locators

Query String :



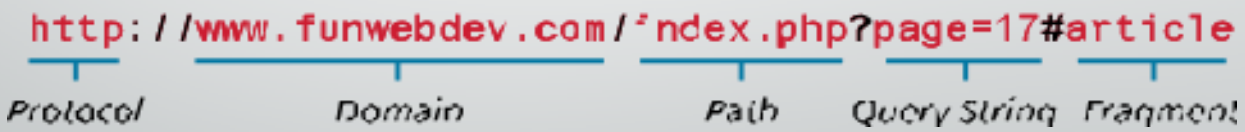


Uniform Resource Locators

Fragment :

- A way of requesting a portion of a page.

- Browsers will see the fragment in the URL, seek out the tag anchor in the HTML, and scroll the website to it



The End

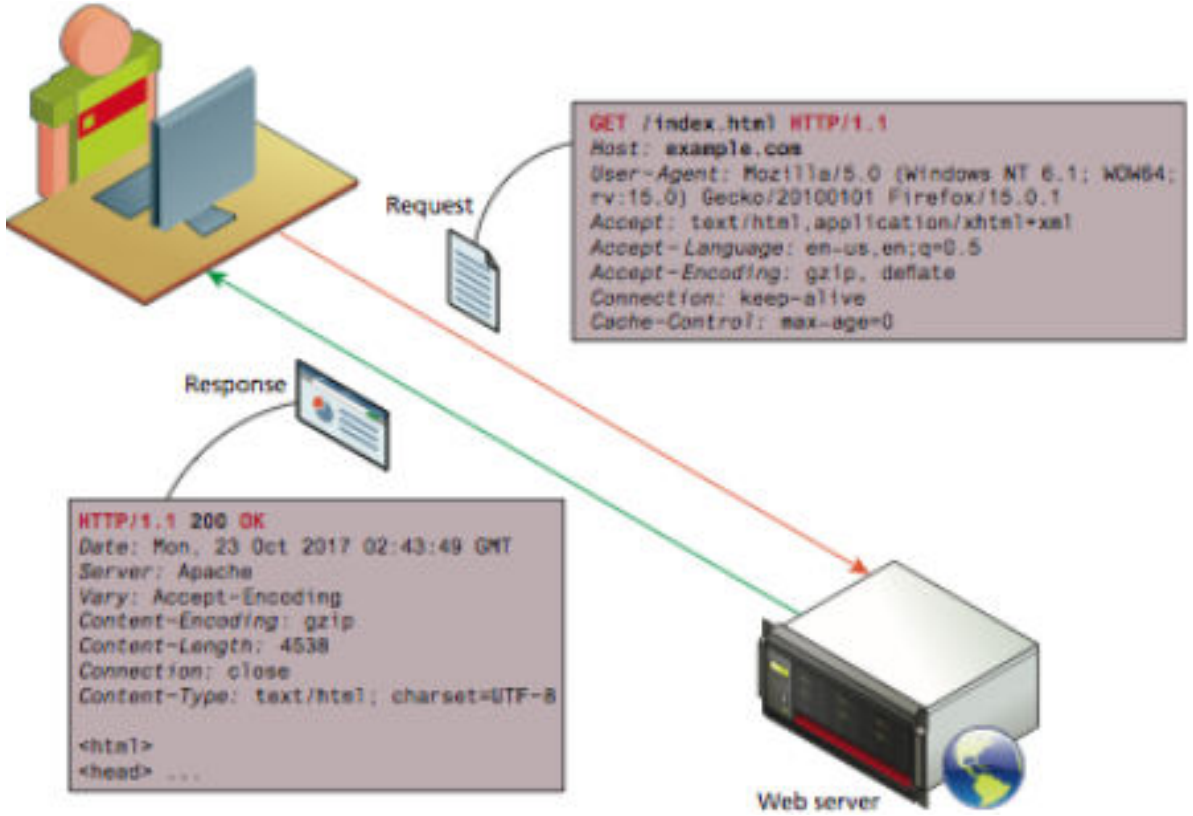
Hypertext Transfer Protocol

Web System and Tecnologies CS 521

TOPIC 10

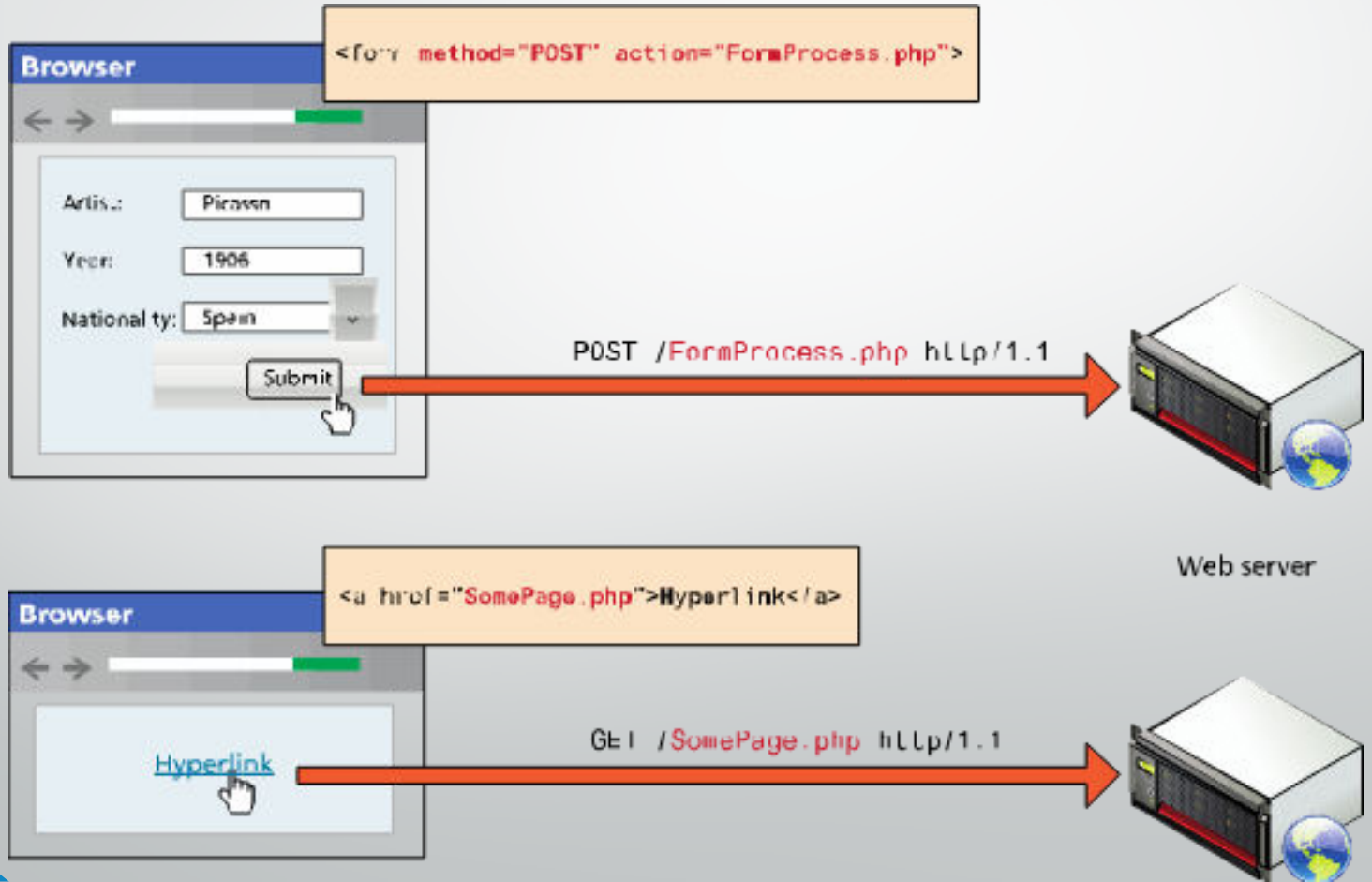
Hypertext Transfer Protocol (HTTP)

Headers



Hypertext Transfer Protocol (HTTP)

Request Methods



Hypertext Transfer Protocol (HTTP)

Response codes

- 200 codes are for successful responses,
- 300 are for redirection-related responses,
- 400 codes are client errors,
- 500 codes are server errors.

Hypertext Transfer Protocol (HTTP)

Response codes – Examples

- 200: OK
- 301: Moved Permanently
- 304: Not Modified
- 307: Temporary redirect
- 400: Bad Request
- 401: Unauthorized
- 404: Not found
- 414: Request URI too long
- 500: Internal server error

The End

Introduction to Web Servers

Web System and Tecnologies CS 521

TOPIC 11

Introduction to Web Servers

Web servers

- A **web server** is nothing more than a computer that responds to HTTP requests.
- Real-world web servers are often more powerful than your own desktop computer
- Webservers must choose an **application stack** to run a website. This application stack will include an
  - Operating System
  - Web Server Software
  - A Database
  - Scripting Language for Dynamic Requests

Introduction to Web Servers

Application Stacks

- Most commonly used LAMP
  - **L**inux operating system,
  - **A**pache web server,
  - **M**ySQL database, and
  - **P**HP scripting language

- Other stacks include WAMP, WISA, MEAN, …

Introduction to Web Servers

Operating systems

- Linux
- Mac
- Windows

Introduction to Web Servers

Webserver Software

- Apache
- Nginx
- IIS

Introduction to Web Servers

Database Software

- MySQL
- PostgreSQL
- Sqlite
- Oracle
- IBM DB2
- Microsoft SQL Server
- MongoDB

Introduction to Web Servers

Scripting Languages

- PHP
- ASP.NET
- Python
- Node.js

The End

Architecture of Web-based System

Web System and Tecnologies CS 521

TOPIC 12

Architecture of Web-based System

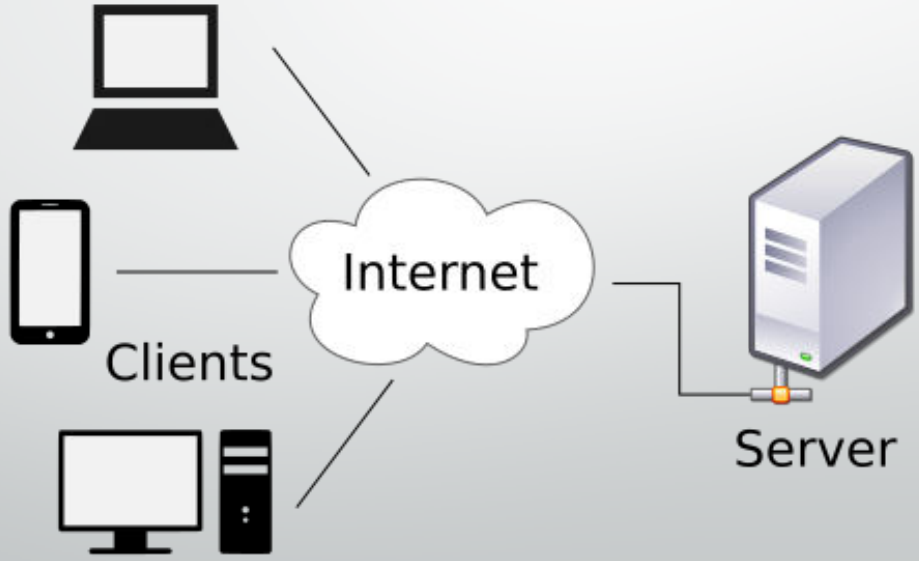
Multi-tier Architecture

- Web development can be realized at multiple tiers
- 2-tier
- 3-tier

Architecture of Web-based System

2-tier Architecture

- Tier 1 – Web Browser
- Tier 2 – Web server

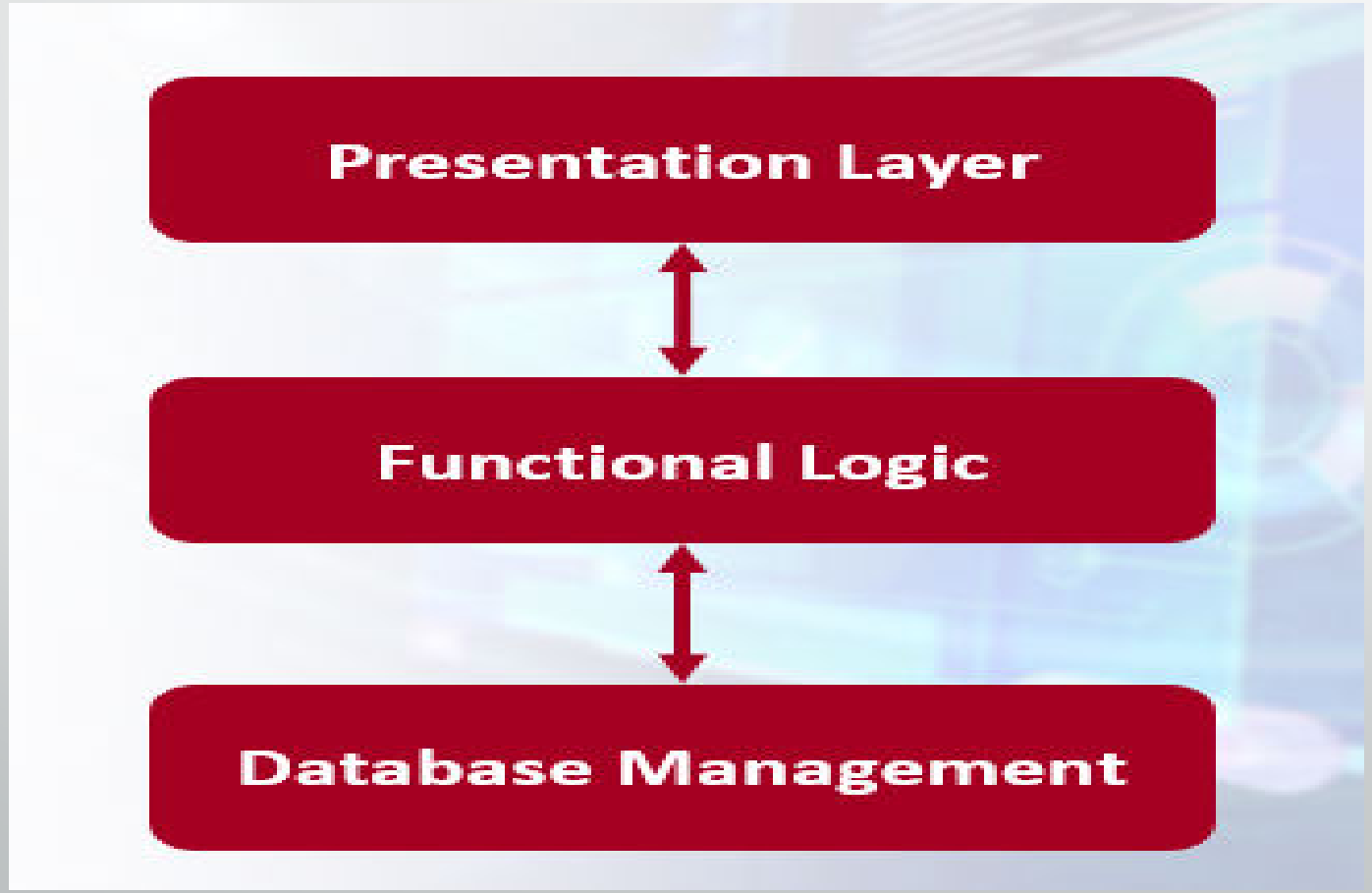


Credit: David Vignoni

https://en.wikipedia.org/wiki/File:Client-server-model.svg

Architecture of Web-based System

3-tier Architecture



The End

Introduction to HTML

Web System and Tecnologies CS 521

TOPIC 13

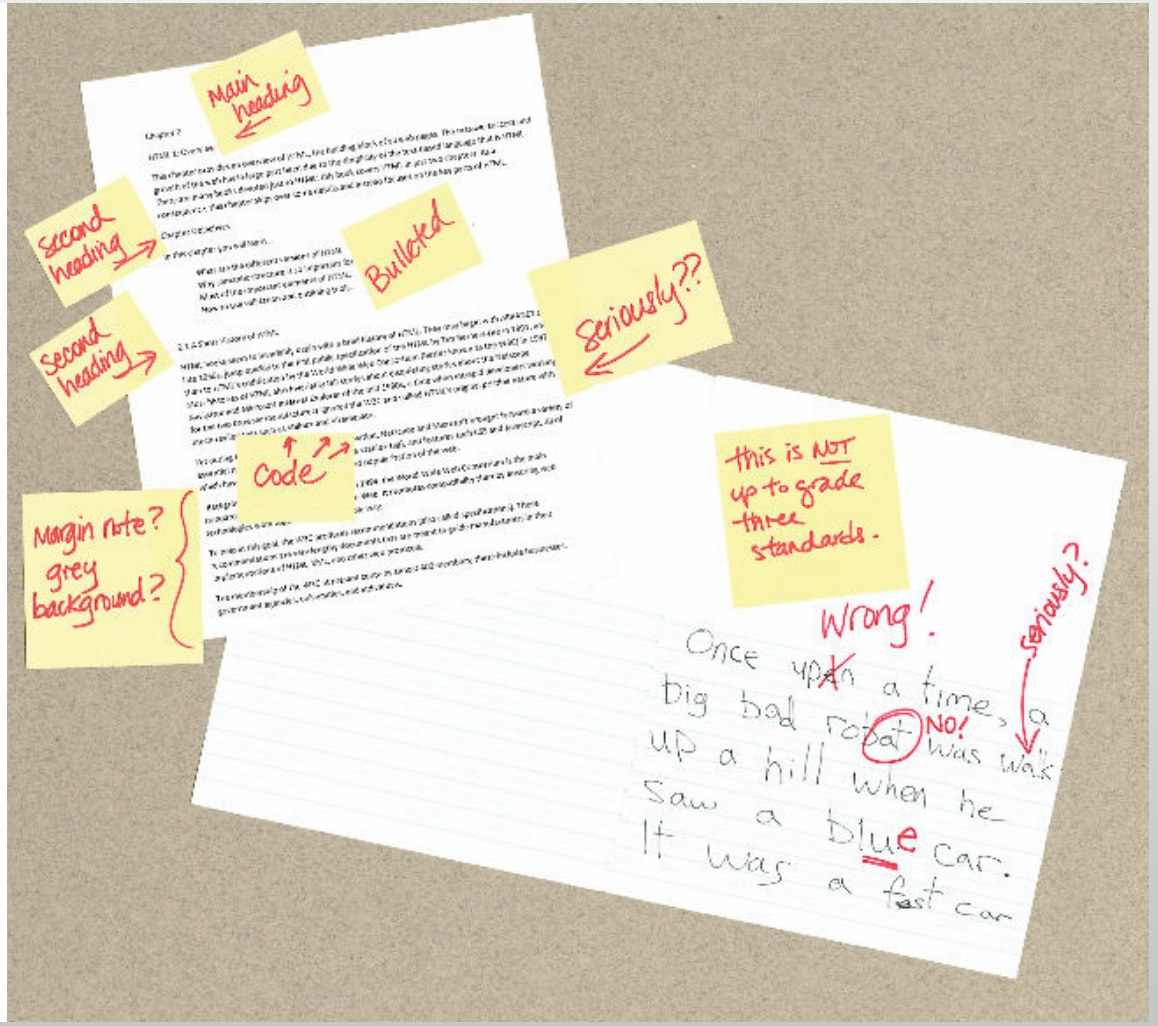
Introduction to HTML

HTML

- Hypertext Transfer **Markup** Language
- **markup** is a way to indicate information about the content that is distinct from the content
- HTML has been through many versions and branches, the details of which might matter if you ever see old HTML code

Introduction to HTML

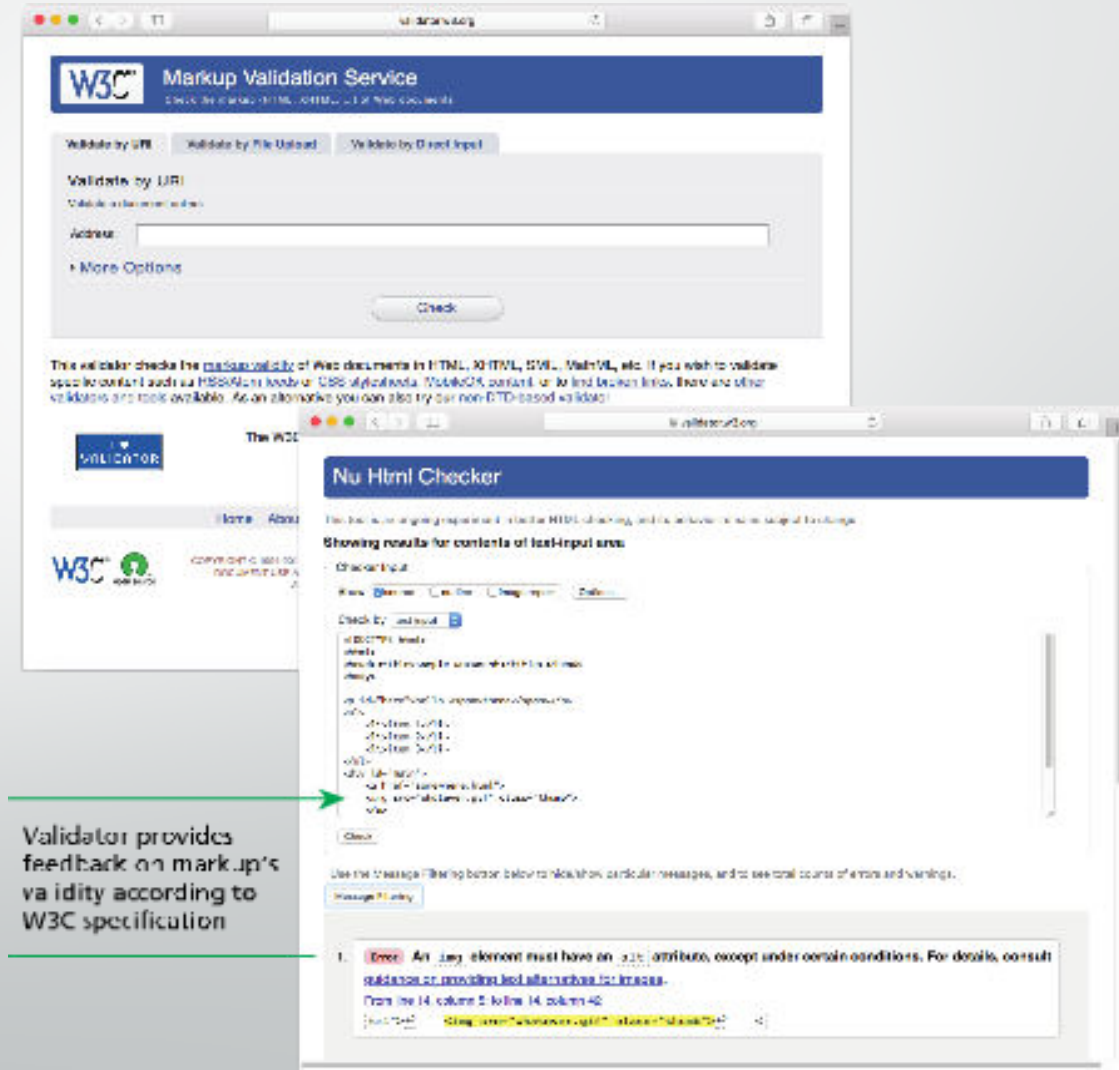
Markup



Introduction to HTML

(X)HTML and Validation

https://validator.w3.org/



Introduction to HTML

HTML 5

- Widely implemented in modern browsers
- The current W3C recommendation for web development

Introduction to HTML

Elements and Attributes

- **HTML documents** are composed of textual content and **HTML elements**
- **HTML element** encompasses
  - the **element name** within angle brackets (i.e., the **tag**) and
  - HTML elements can also contain **attributes, the content** within the tag.



Introduction to HTML

An Empty Element

- An **empty element** does not contain any text content; instead, it is an instruction to the browser to do something.
- In XHTML, empty elements had to be terminated by a trailing slash.
- In HTML5, the trailing slash in empty elements is optional



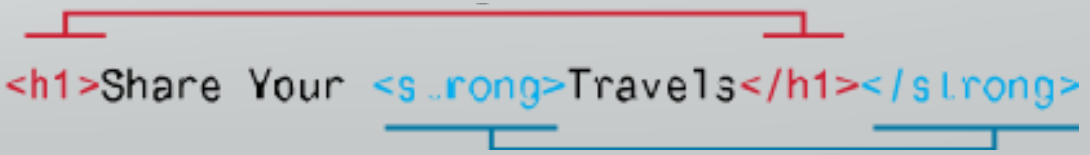
Introduction to HTML

Nesting elements

- Correct nesting
  - Nested element must be closed first

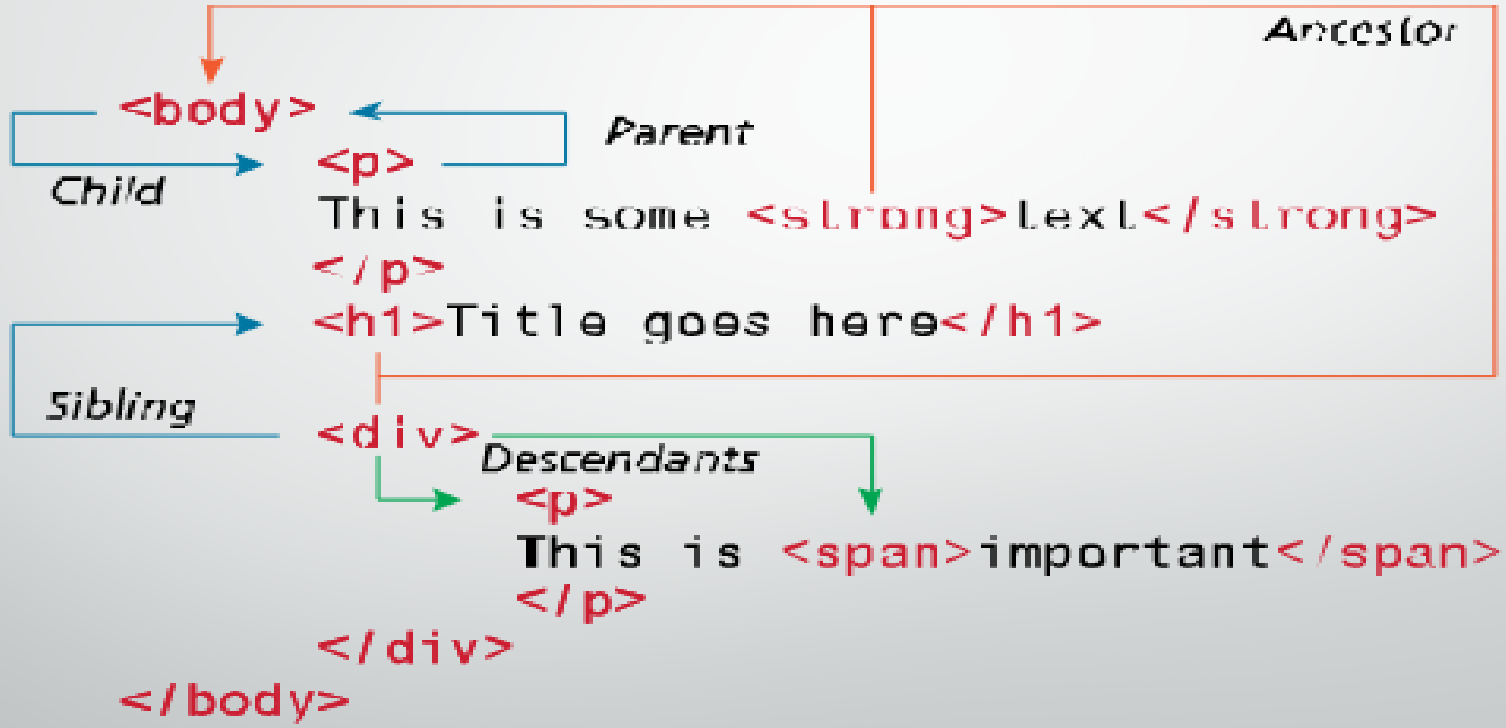


- Incorrect nesting
  - Outer element is closed first



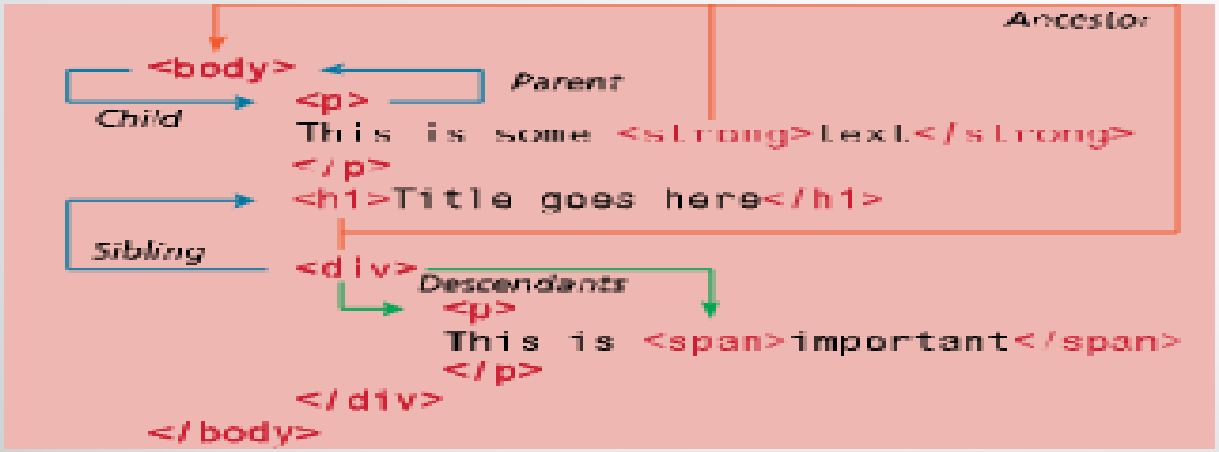
Introduction to HTML

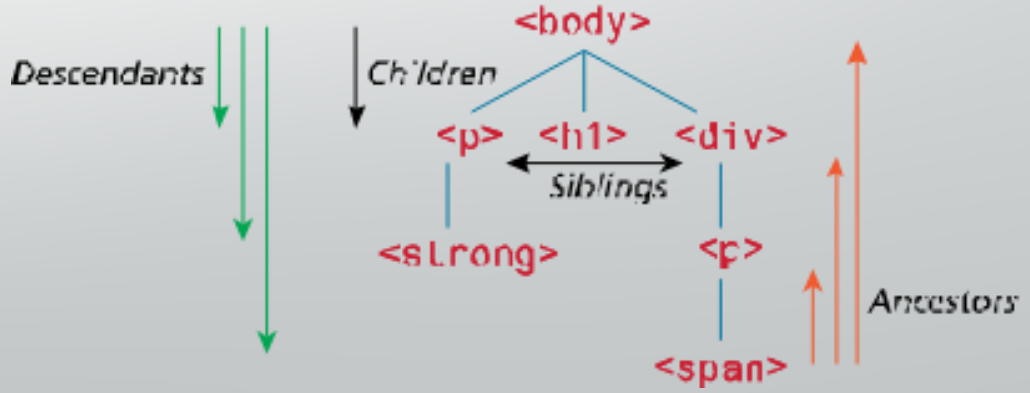
Nesting elements



Introduction to HTML

Tree of Nested Elements





The End

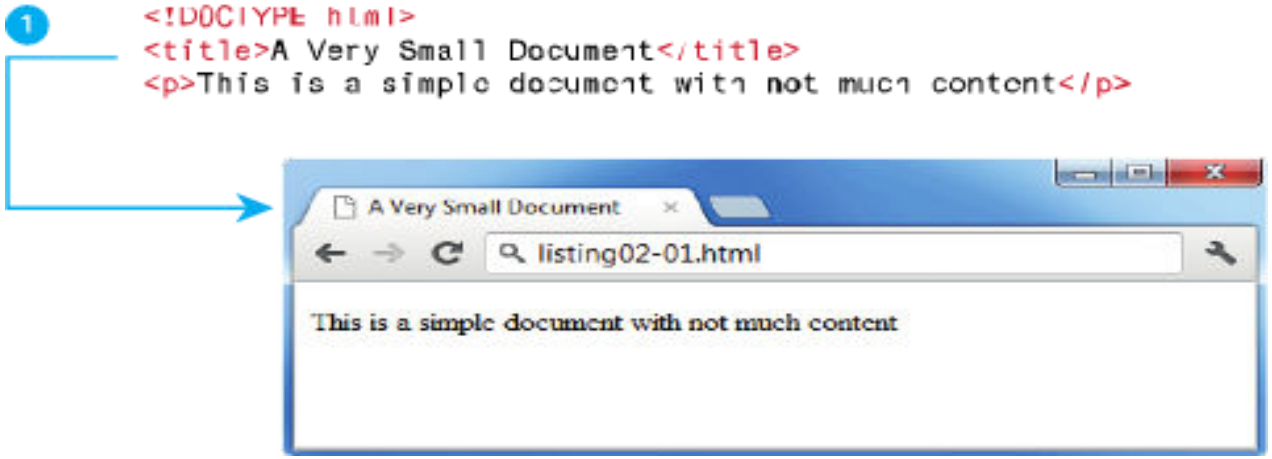
Common HTML Tags

Web System and Tecnologies CS 521

TOPIC 14

Common HTML Tags

A Simple Example



Common HTML Tags

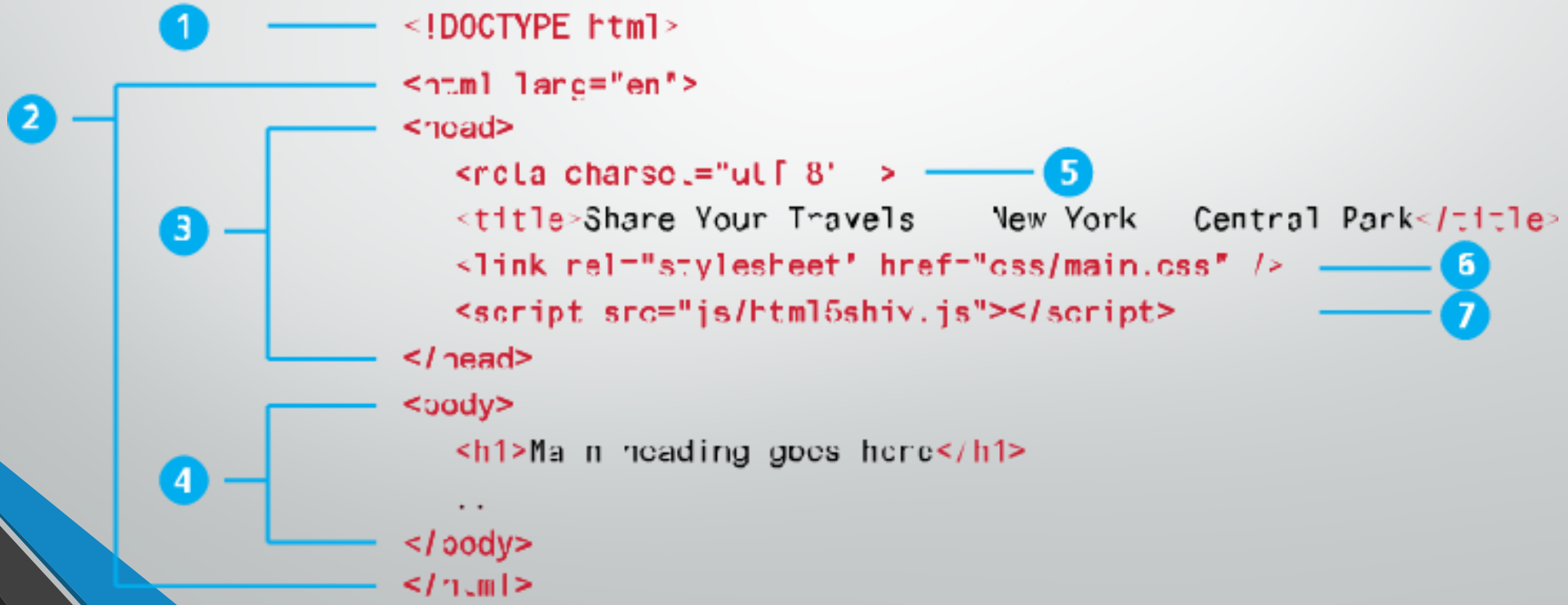
DOCTYPE

- DOCTYPE Short for **Document Type Definition** tells the browser what type of document it is about to process
- Written at the start of the HTML document

<!DOCTYPE html>

Common HTML Tags

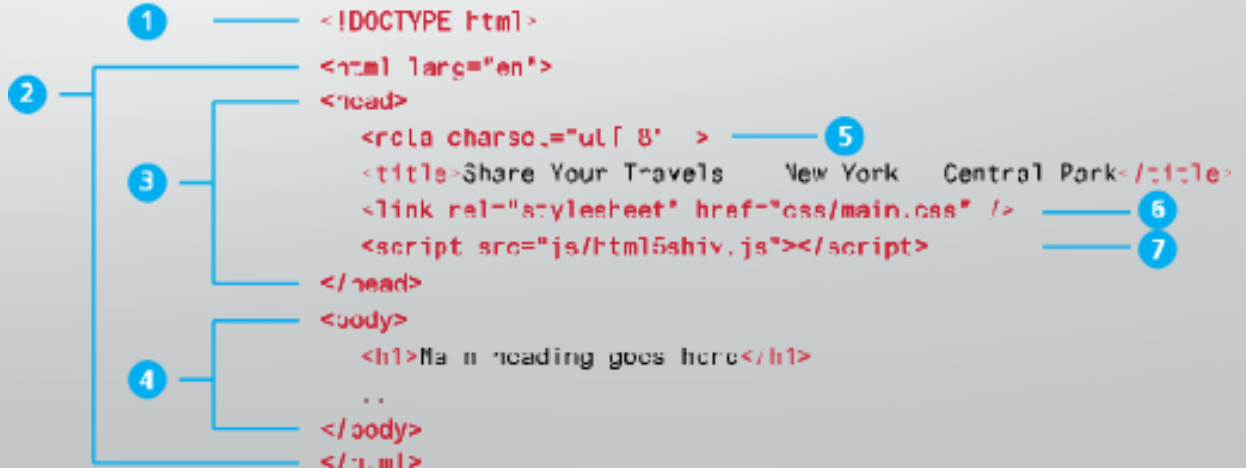
Another Example



Common HTML Tags

Head and Body Tags

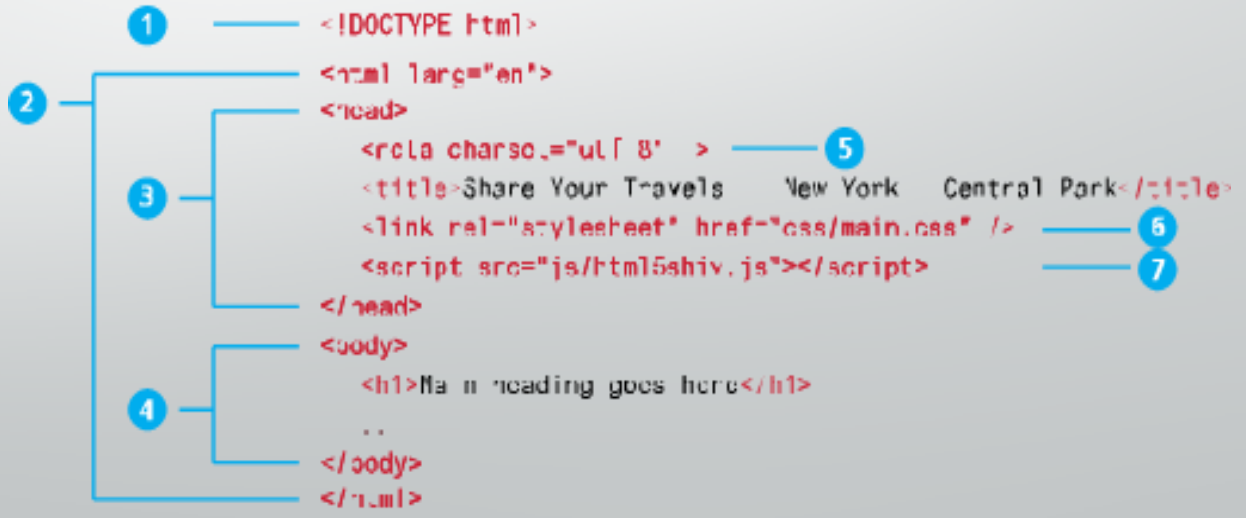
- HTML5 does not require the use of the **<html> , <head>** , and **<body>** elements (most developers still use them)
- **<html>** contains all the other HTML elements in the document (Item 2)
- **<head>** contains descriptive elements about the document, such (title, style sheets, JavaScript files etc.) (Item 3)
- **<body>** contains content to be displayed by the browser (Item 4)



Common HTML Tags

Meta, Link, and Script Tags

- The **<meta>** element (Item 5) declares that the character encoding for the document is UTF-8.
- Item 6 specifies an external CSS style sheet file with **<link>** that is used with this document.
- Item 7 references an external JavaScript file using **<script>**



Common HTML Tags

Another Example

```
<body>
    <h1>Share Your Travels</h1>
    <h2>New York - Central Park</h2>
    <p>Photo by Randy Connolly</p>
    <p>This photo of Conservatory Pond in
        <a href="http://www.centralpark.com/">Central Park</a>
        New York City was taken on October 22, 2016 with a
        <strong>Canon EOS 30D</strong> camera.
    </p>
    <img src="images/central-park.jpg" alt="Central Park" />

    <h3>Reviews</h3>
    <div>
        <p>By Ricardo on <time>2016-05-23</time></p>
        <p>Easy on the HDR buddy.</p>
    </div>
    <hr>
    <div>
        <p>By Susan on <time>2016-11-18</time></p>
        <p>I love Central Park.</p>
    </div>

    <p><small>Copyright &copy; 2017 Share Your Travels</small></p>
</body>
```

1 2 3 4 5 6 7 8 9 10

Common HTML Tags

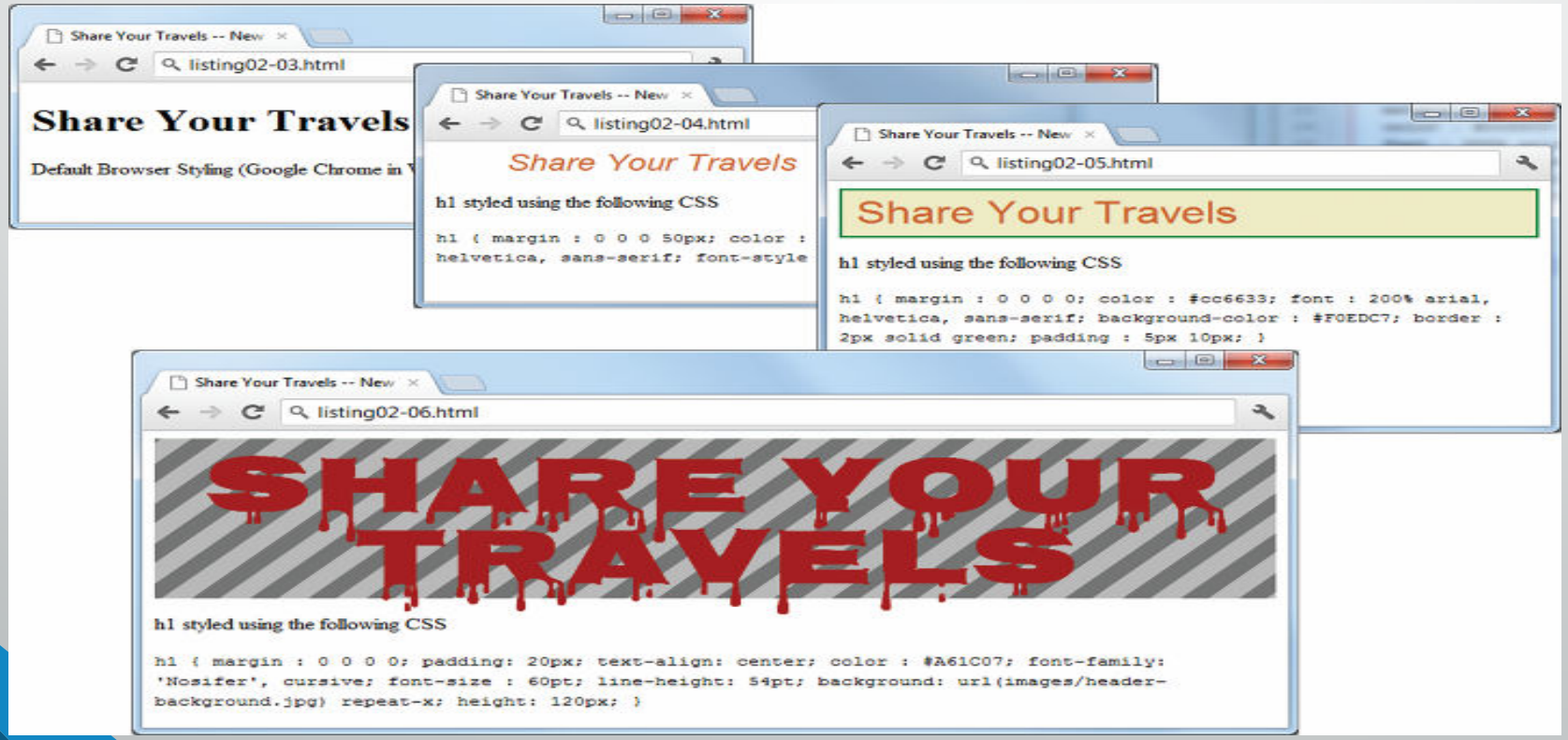
Rendered Document



Common HTML Tags

HTML is for structure!

- CSS is used for styling



Common HTML Tags

Headings

- HTML provides six levels of headings (h1 - h6)
- Headings are also used by the browser to create a document outline for the page
- Don't use different headings for setting font size (will be covered in CSS)



Common HTML Tags

Paragraphs and Divisions

- **<p>** tag is a container for text and other HTML elements
- ** ** also a container element and is used to create a logical grouping of content



Common HTML Tags

Inline Text Elements

- Do not disrupt the flow
  - <a>
  - <abbr>
  -
  - <cite>
  - <code>
  -
  - <mark>
  - <small>
  -
  -
  - <time>
  - …

… <inline element> …

Common HTML Tags

Block Elements

- Disrupt the existing flow and take their own space
  -
  - <h1>-<h6>
  - <hr>
  -
  -
  -
  - <p>
  - <table>
  - …

…

<block element>

…

Common HTML Tags

Images

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

Text in title attribute will be displayed in a pop up 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

Common HTML Tags

Character Entities

- These are special characters for symbols for which there is either no easy way to type them via a keyboard
- or which have a reserved meaning in HTML (like "<")

Entity	Description
	Nonbreakable space
<	<
>	>
©	©
™	™

The End

Web Systems and Technologies

TOPIC 15

Links

Links

Links (Anchor tags)

- Links are an essential feature of all web pages
- Links use the <a> element (the “a” stands for anchor).



Links

Examples

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>
...
<a name="top">
```

Defines anchor for a link to another place on same page

Links

Examples

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>
```

Links

Relative Referencing

Relative Link Type	Example
Same directory	<a href="example.html">
Child Directory	<a href="images/logo.gif">
Grandchild/Descendant Directory	<a href="css/images/background.gif">
Parent/Ancestor Directory	<a href="../about.html"> <a href="../../about.html”>
Sibling Directory	<a href="../images/about.html">
Root Reference	<a href="/images/background.gif">

Web Systems and Technologies

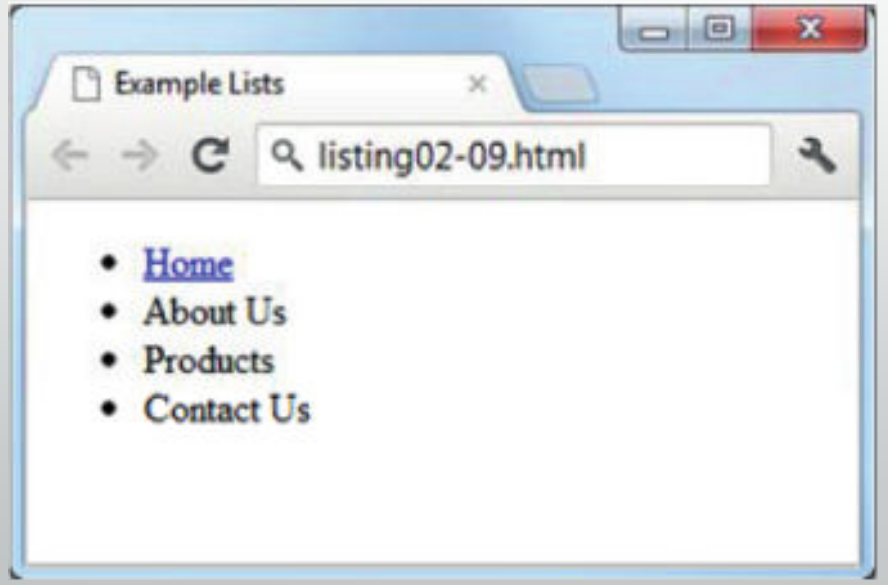
TOPIC 16

Lists

Lists

Unordered Lists

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



Lists

Ordered Lists

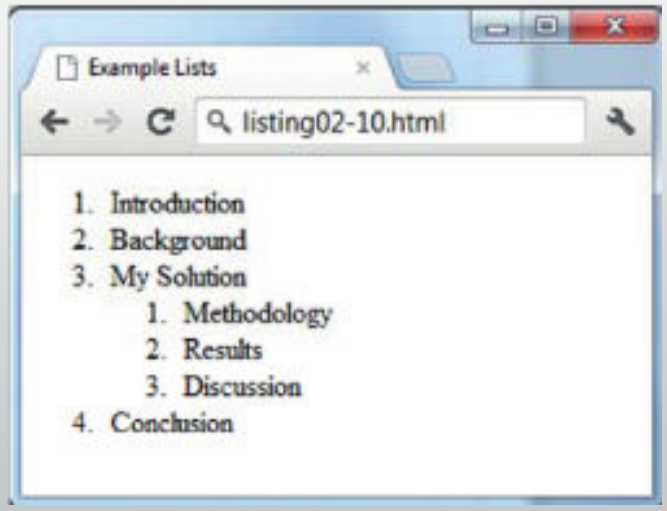
```
<ol>
    <li>Introduction</li>
    <li>Background</li>
    <li>My Solution</li>
    <li>Conclusion</li>
</ol>
```

1. Introduction
2. Background
3. My Solution
4. Conclusion

Lists

Nested Lists

```
<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>
```



Lists

Description Lists

- Used for definitions and descriptions

```
<dl>
    <dt>HTML</dt>
    <dd>Hyper Text Markup Language</dd>
    <dt>CSS</dt>
    <dd>Cascading Style Sheet</dd>
</dl>
```

HTML
    Hyper Text Markup Language
CSS
    Cascading Style Sheet

END

Web Systems and Technologies

TOPIC 17

Semantics in HTML 5

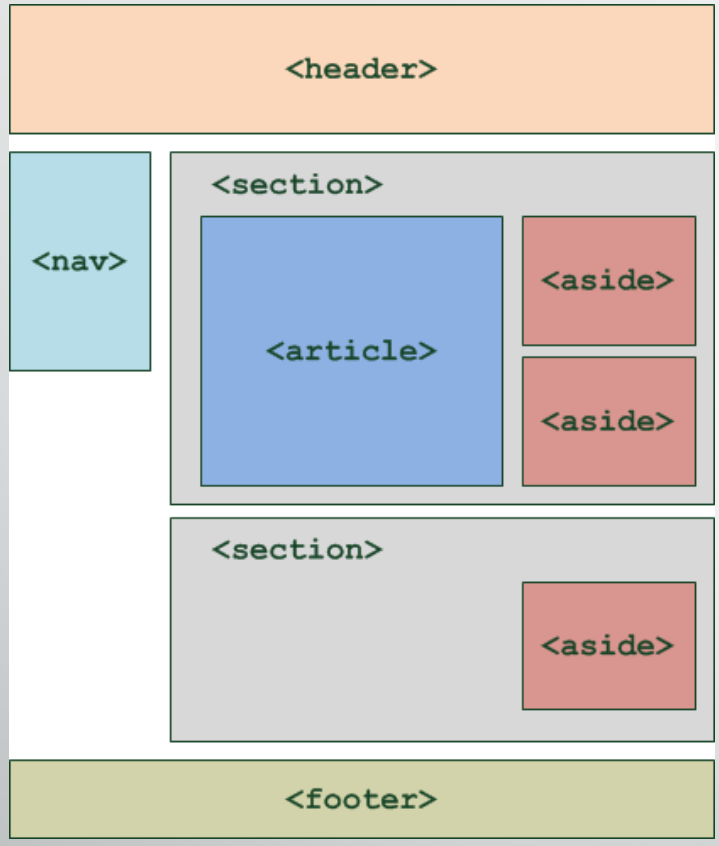
Semantics in HTML 5

Semantic Markup

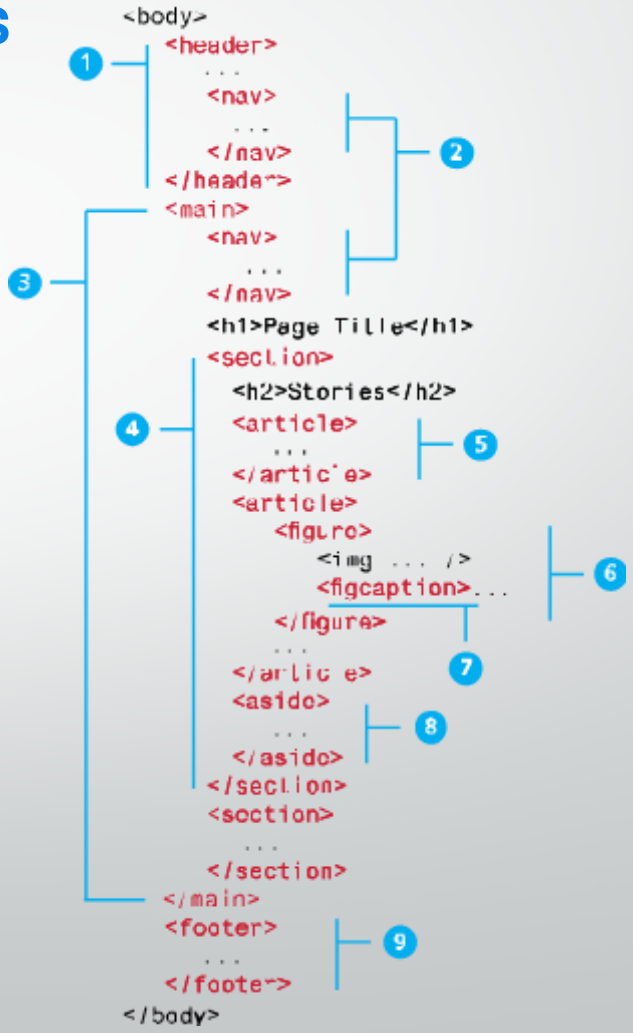
- Semantic tags are used for organizing contents and labeling contents (adding meaning)
- Do not change the structure of the document
- Make HTML code
  - maintainable
  - accessible

Semantics in HTML 5

Semantic tags

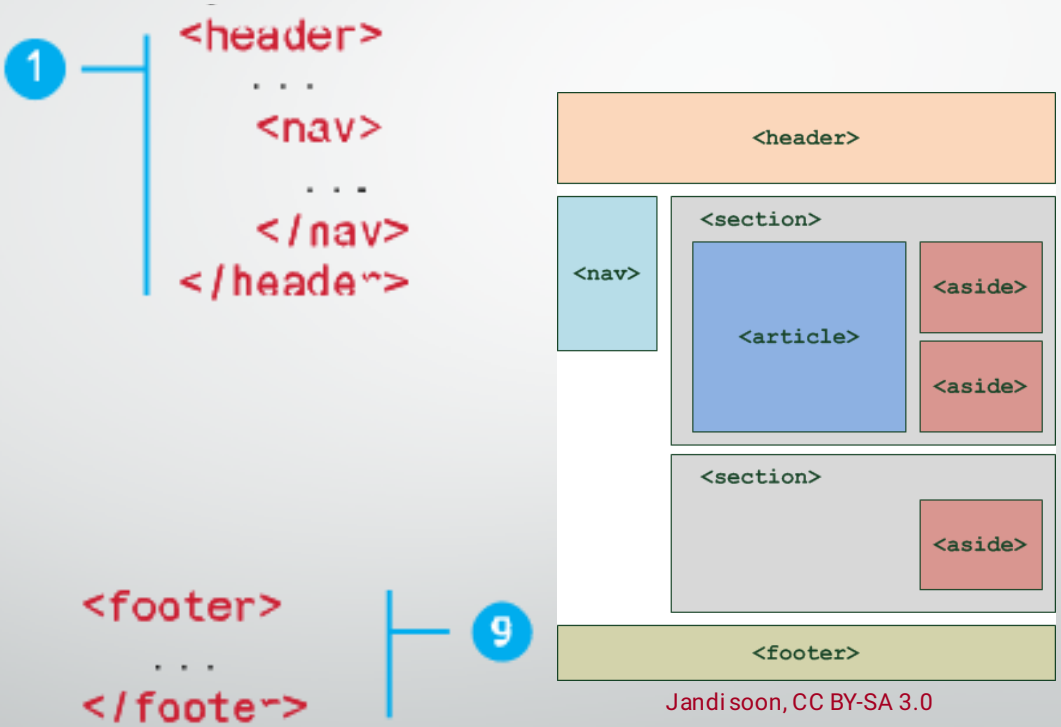


Jandi soon, CC BY-SA 3.0



Semantics in HTML 5

Header and Footer



Jandi soon, CC BY-SA 3.0

Semantics in HTML 5

Header

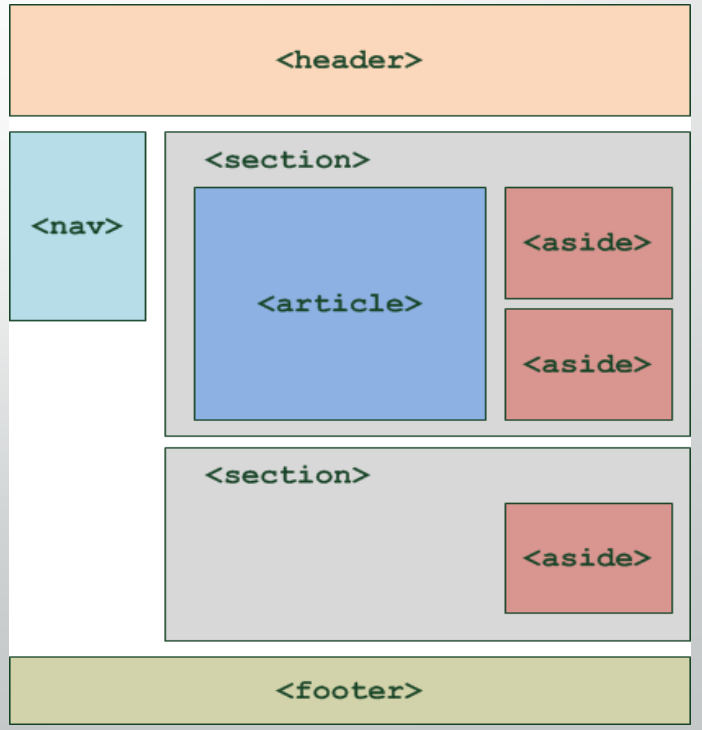
- Intended to usually contain the section's heading (an h1– h6 element), but not required
- The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.

```
<header>
    <img src="logo.gif" alt="logo" />
    <h1>Fundamentals of Web Development</h1>
    ...
</header>
```

Semantics in HTML 5

Navigation (nav)

```
<header>
    <img src="logo.gif" alt="logo" />
    <h1>Fundamentals of Web Development</h1>
    <nav>
        <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>
```

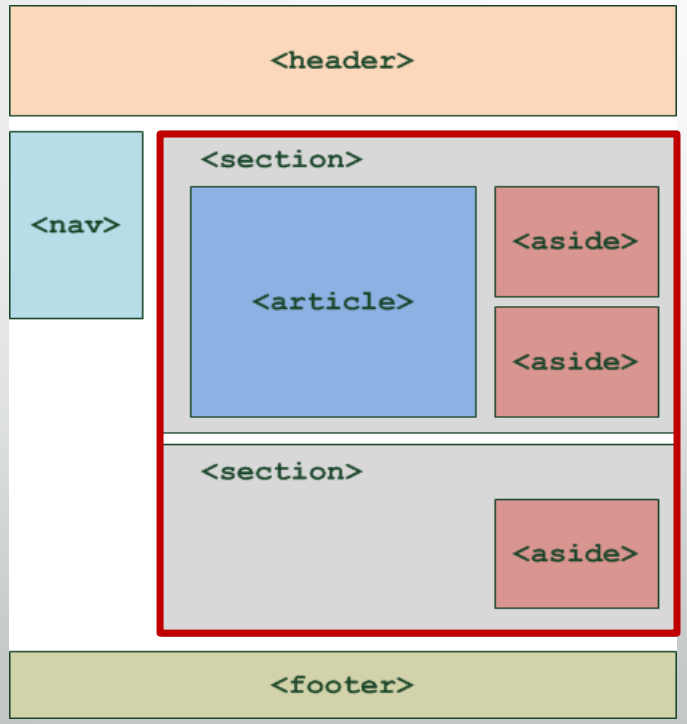


Jandi soon, CC BY-SA 3.0

Semantics in HTML 5

Main

- **<main>** is meant to contain the main unique content of the document.

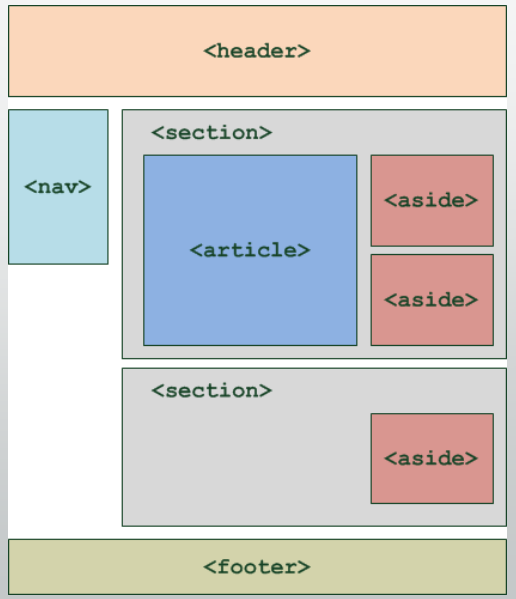


Jandi soon, CC BY-SA 3.0

Semantics in HTML 5

Section and Article

- **<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

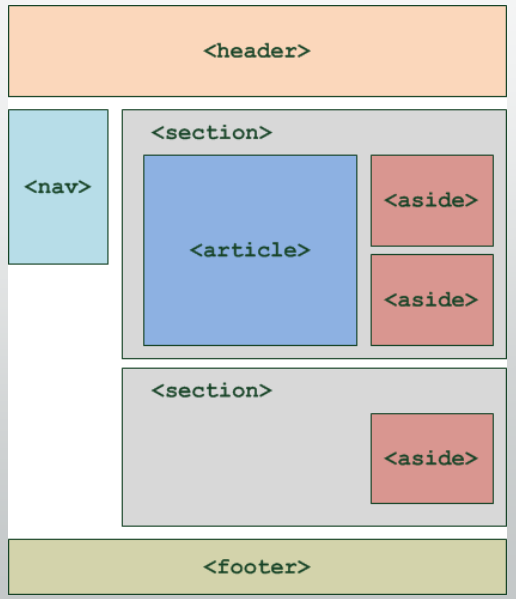


Jandi soon, CC BY-SA 3.0

Semantics in HTML 5

Aside

- The **<aside>** element can be used for sidebars, pull quotes, groups of advertising images, or any other grouping of nonessential elements



Jandi soon, CC BY-SA 3.0

Semantics in HTML 5

Figures and Figure Captions

Figure could be moved to a different location in document.
...

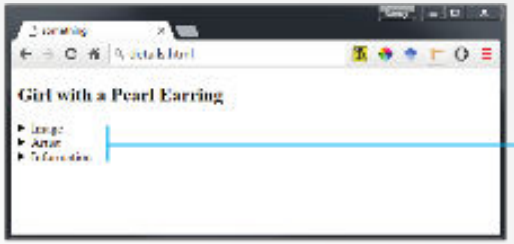
But it has to exist in the document (i.e., the figure isn't optional).

```
<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>
```



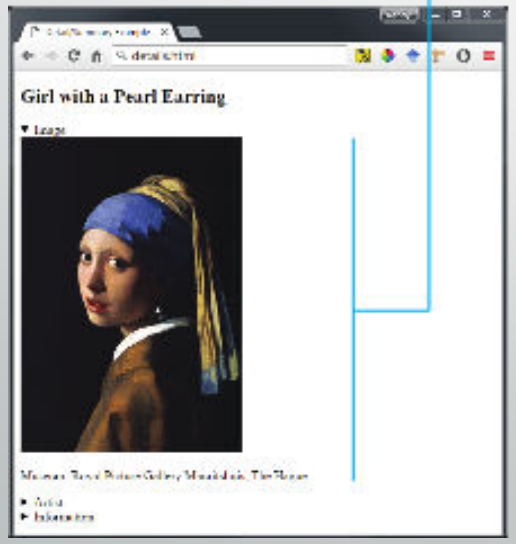
Semantics in HTML 5

Details and Summary




```
<body>
   <h2>Girl with a Pear' Earring</h2>
   <details>
     <summary>Image</summary>
     <img src="images/106020.jpg"><br>
     <p>Museum: Roya' Picture Ga'lery Mauritshuis ...
   </details>
   <details>
     <summary>Artist</summary>
     <p><strong>Jan Vermeer</strong> was a Dutch ...
   </details>
   <details>
     <summary>Information</summary>
     <p>
        Date: 1665<br>
        Medium: Oil on Canvas
     </p>
  </details>
</body>
```

Clicking on the summary label reveals the rest of the content with the <details> container



Web Systems and Technologies

TOPIC 18

Tools Insight

Tools Insights

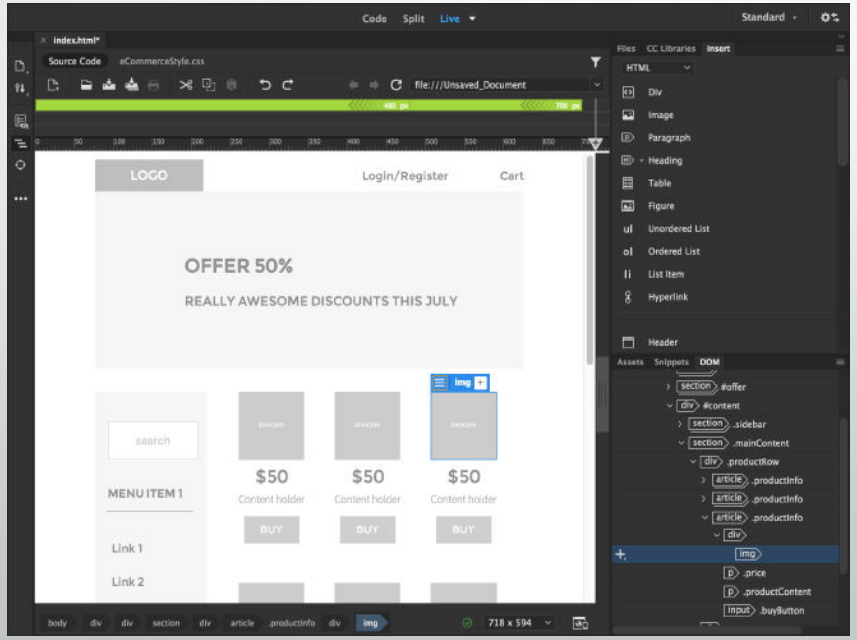
A variety of tools available for web development

- WYSIWYG
- Code editors
- IDEs
- Cloud based environments
- Code playgrounds

Tools Insights

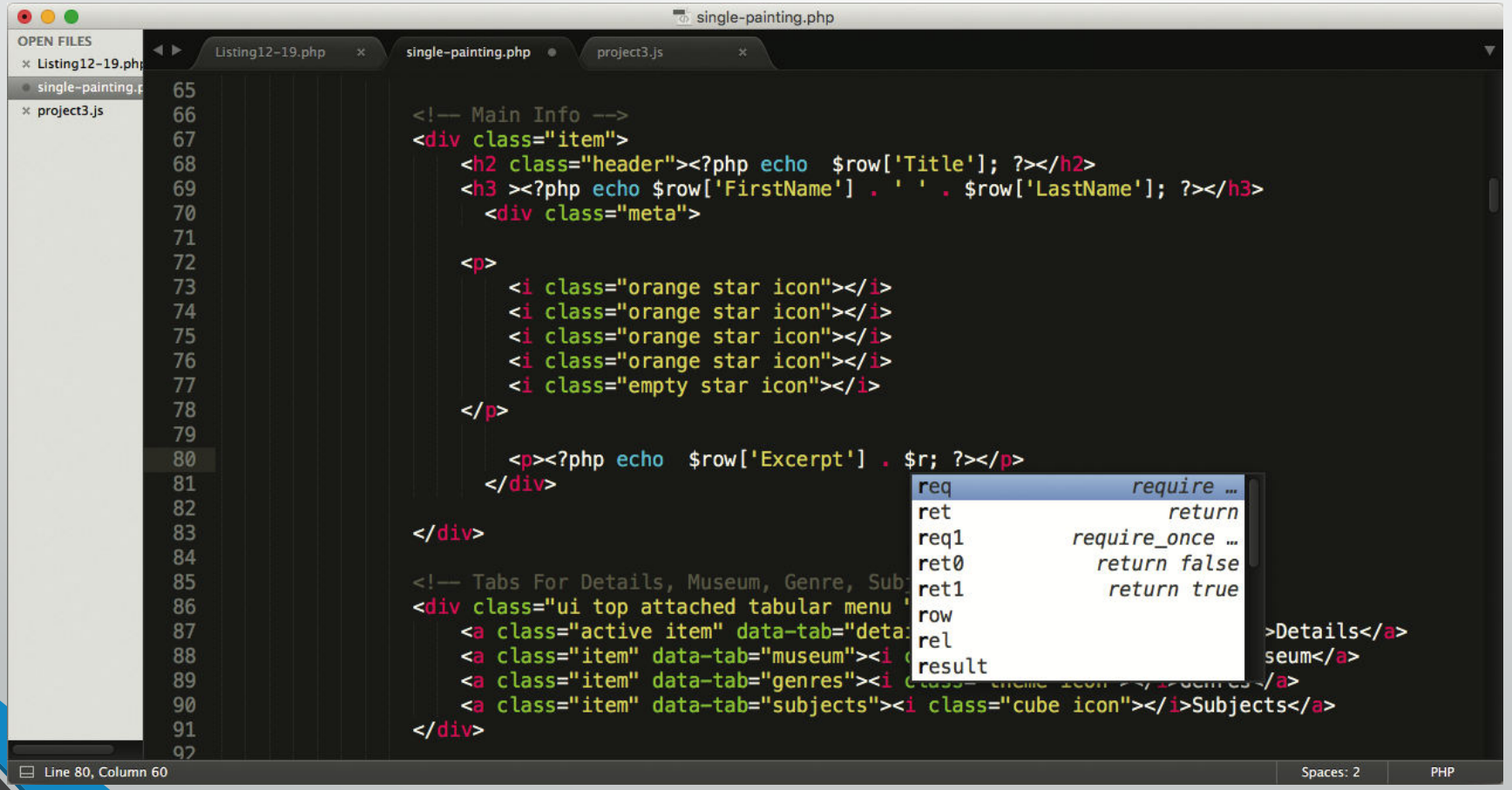
WYSIWYG Editors

- Dreamweaver
- Google Web Designer
- Seamonkey composer



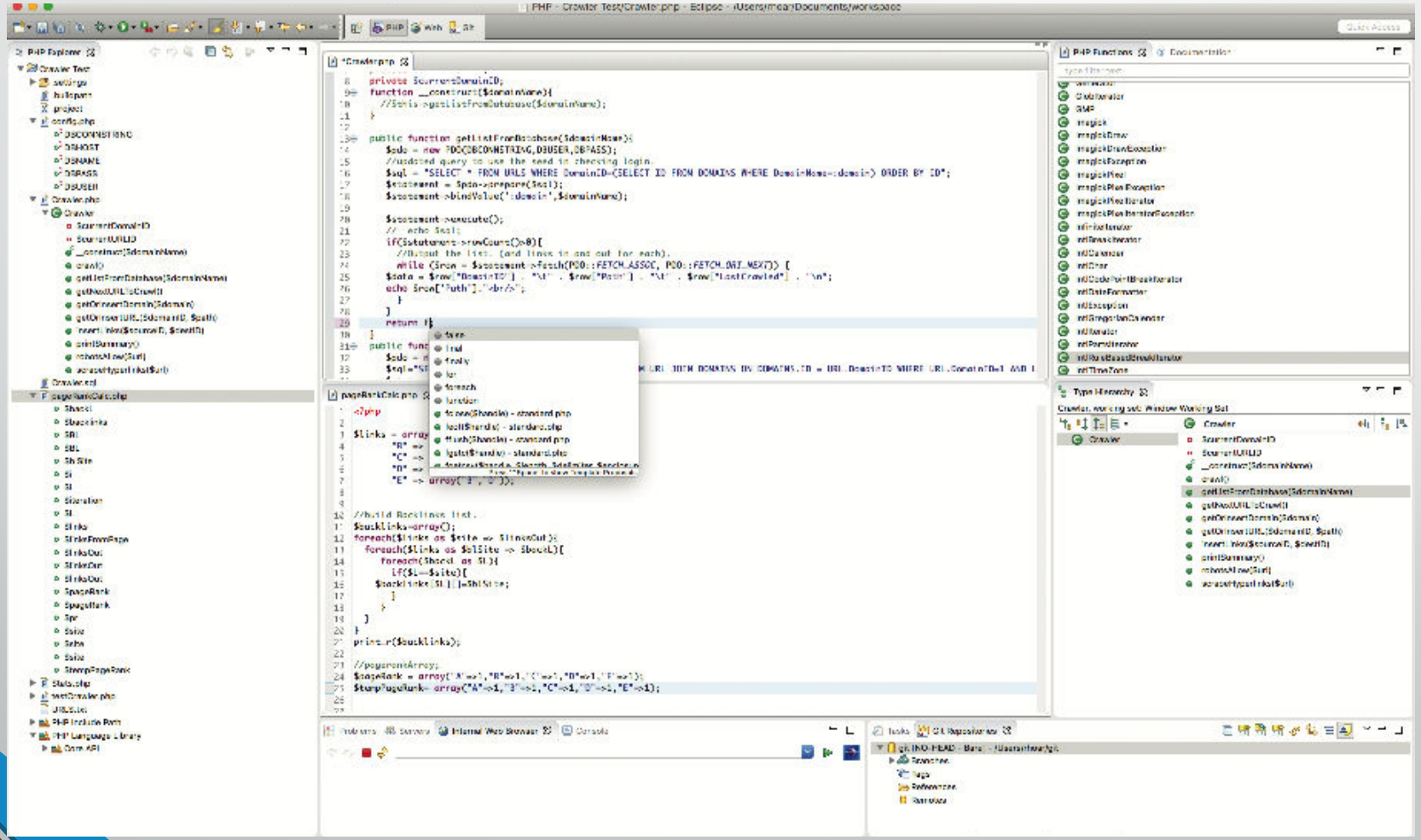
Tools Insights

Code editors



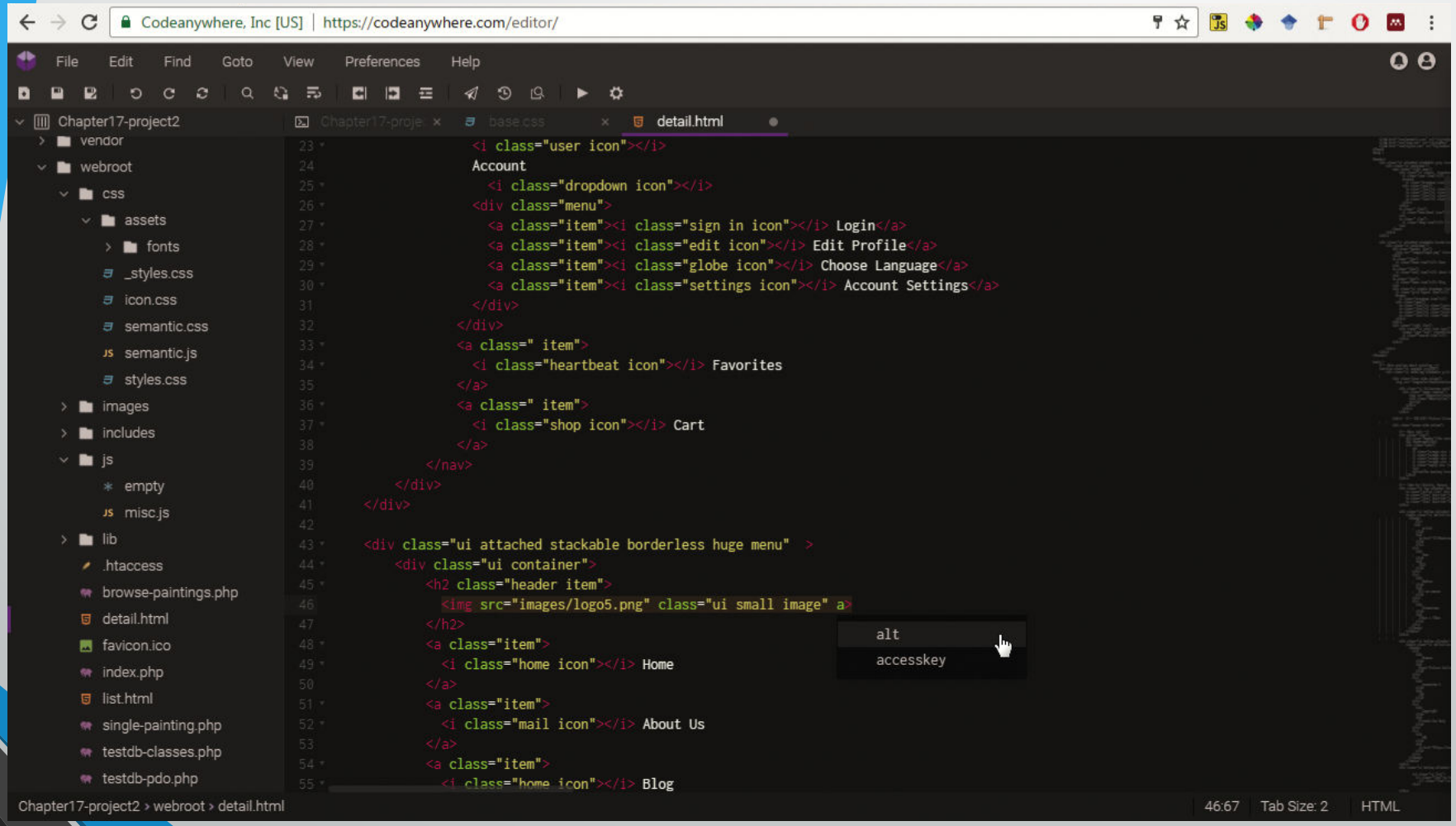
Tools Insights

IDEs



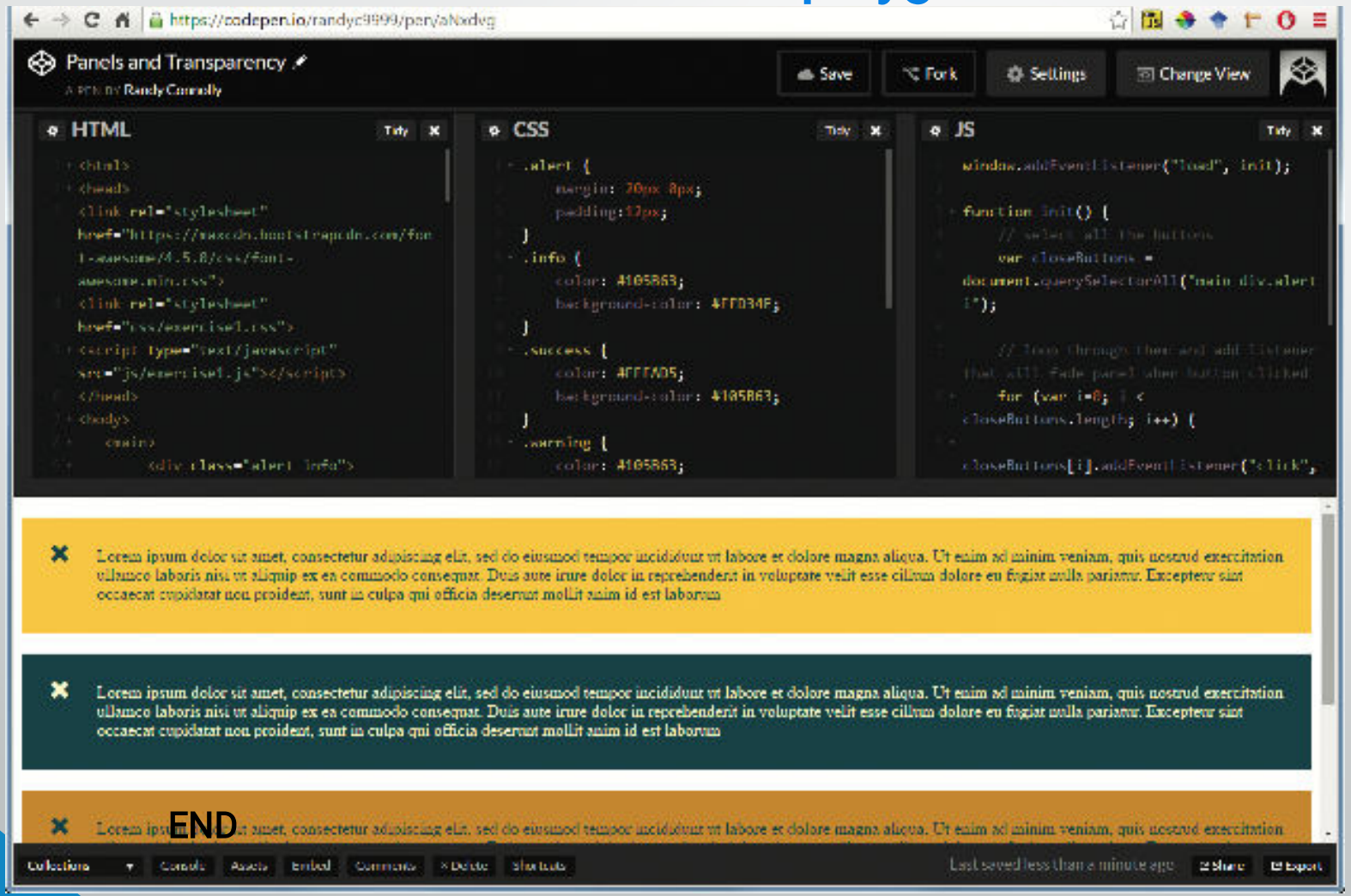
Tools Insights

Cloud-based Environments



Tools Insights

Code playgrounds



END

Web Systems and Technologies

TOPIC 19

Introduction to CSS

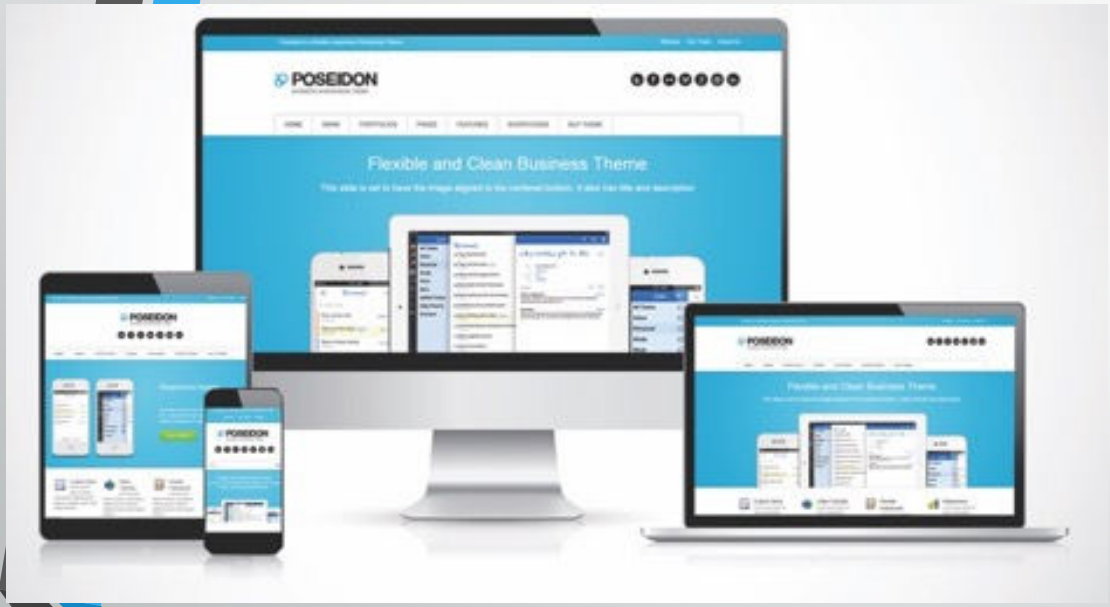
Introduction to CSS

Cascading Style Sheets (CSS)

- CSS is a W3C standard for describing the appearance of HTML elements
- With CSS, we can assign font properties, colors, sizes, borders, background images, and even position elements on the page
- 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.

Introduction to CSS

Benefits of CSS



- Improved control over formatting
- Improved site maintainability
- Improved accessibility
- Improved page-download speed
- Improved output flexibility (responsive design)

Introduction to CSS

CSS Versions

- 1996 the CSS Level 1 Recommendation was published
- June 2011 - CSS2.1 becomes an official W3C Recommendation
- From CSS 3 onwards, recommendations are made based on modules. Following modules have been recommended:
  - Cascading and Inheritance
  - Color Module
  - Fonts Module
  - Media Queries
  - Selectors
  - Basic User Interface Module

Introduction to CSS

Browser Adoption

- Historically implementation of CSS was not consistent across major browsers
- CSS has a reputation for being a somewhat frustrating language due to browser implementation differences
- CSS was not really designed to be a layout language, so authors often find it tricky dealing with floating elements, relative positions, inconsistent height handling, overlapping margins, and non-intuitive naming

Web Systems and Technologies

TOPIC 20

CSS Syntax (Properties)

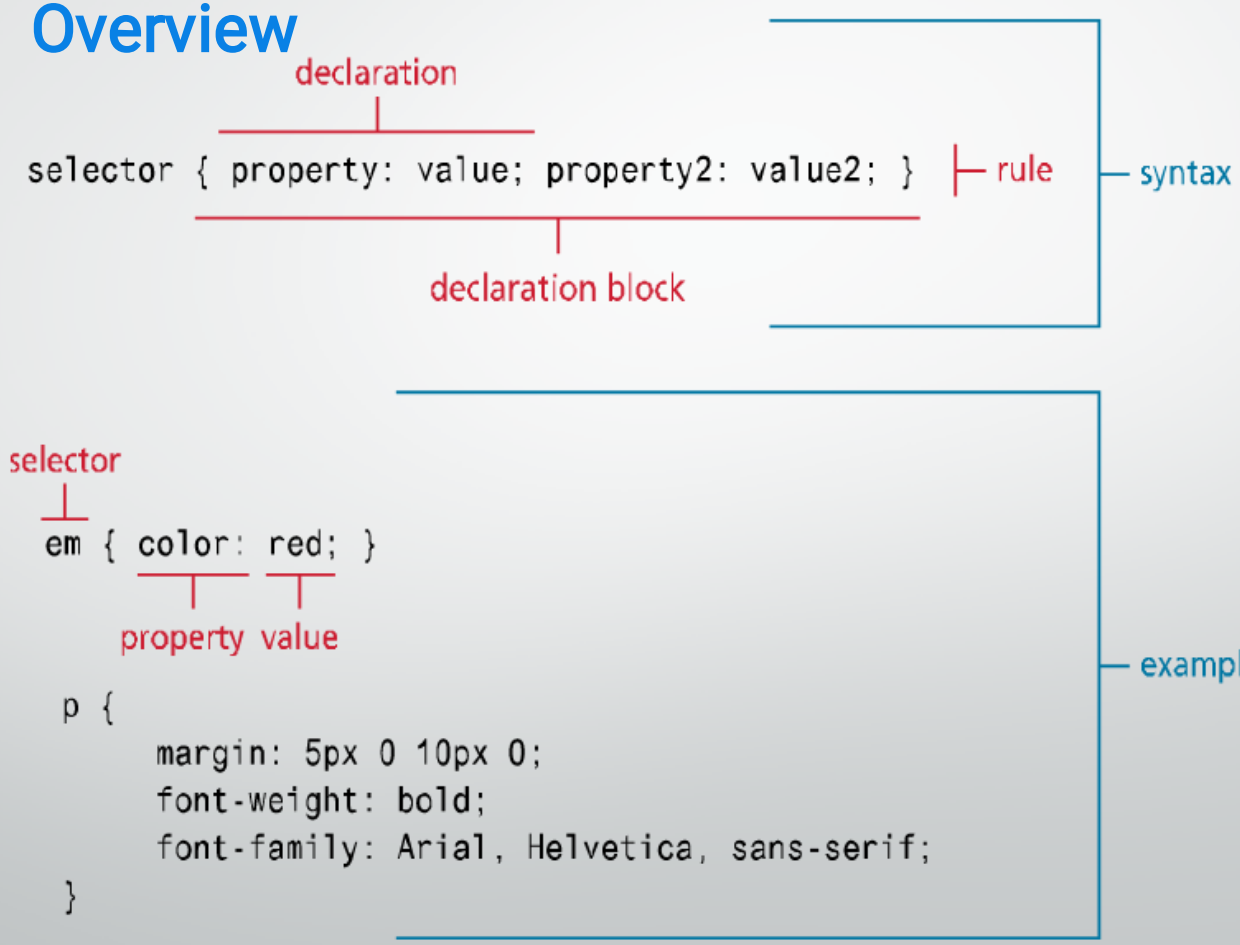
CSS Syntax

Overview

- 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:value** pairs (each pair is also called a declaration)
- The series of declarations is also called the declaration block

CSS Syntax

Overview



CSS Syntax

Selectors

- 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
- Many ways to write selectors (stay tuned…)

CSS Syntax

Properties

- Each individual CSS declaration must contain a property.
- The CSS2.1 recommendation defines over a hundred different property names

CSS Syntax

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

CSS Syntax

Properties

Property Type	Property
Color and Background	background background-color background-image background-position background-repeat box-shadow color opacity
Borders	border border-color border-width border-style border-top, border-left, ... border-image border-radius

CSS Syntax

Properties

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

CSS Syntax

Properties

Property Type	Property
Lists	list-style list-style-image list-style-type
Effects	animation filter perspective transform transition

Web Systems and Technologies

TOPIC 21

CSS Syntax (Values)

CSS Syntax

Values

- 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

CSS Syntax

Colors

- Name
- Hexadecimal
- RGB
- RGBa
- HSL

CSS Syntax

Relative and absolute Units

- Relative
  - px
  - em
  - %
  - vw,vh
- Absolute
  - in
  - cm
  - pt

Web Systems and Technologies

TOPIC 22

Location of Styles

Location of Styles

Inline Styles

- **Inline styles** are 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>
```

- Using inline styles is generally discouraged
- Handy for quickly testing out a style change

Location of Styles

Embedded Style Sheet

- Also called internal styles are style rules placed within the **<style>** element (inside the <head> element of an HTML document)

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

Location of Styles

External Style Sheet

- External style sheets are style rules placed within an external text file with the .css extension

```
<head>
  <meta charset="utf-8">
  <title>Share Your Travels</title>
  <link rel="stylesheet" href="styles.css" />
</head>
```

Location of Styles

External Style Sheet

Three different types of style sheets:

1. Author-created style sheets
2. User style sheets
3. Browser style sheets

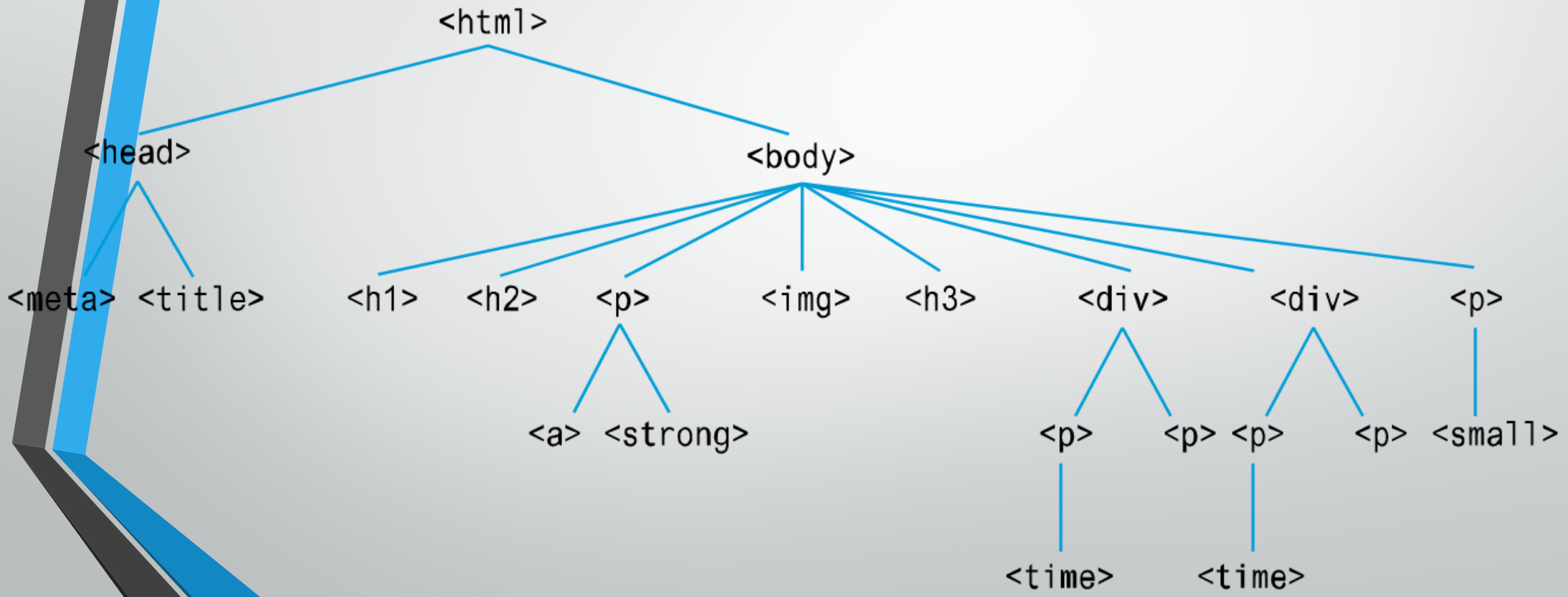
Web Systems and Technologies

TOPIC 23

CSS Selectors
(Element, Class, ID)

Selectors

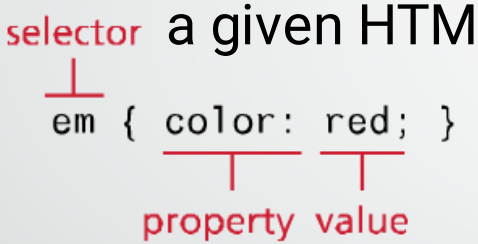
HTML DOM Tree



Selectors

Element Selectors

- **Element selectors** select all instances of a given HTML element.




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

Selectors

Class Selectors

- A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree using the same class attribute value.

Selectors

Class Selectors

```
<head>
        <title>Share Your Travels </title>
        <style>
                .first {
                  font-style: italic;
                  color: red;
                }
        </style>
</head>


<body>
        <h1 class="first">Reviews</h1>
        <div>
          <p class="first">By Ricardo on <time>2016-05-23</time></p>
          <p>Easy on the HDR buddy.</p>
        </div>
        <hr/>
        <div>
          <p class="first">By Susan on <time>2016-11-18</time></p>
          <p>I love Central Park.</p>
        </div>
        <hr/>
</body>
```

Selectors

ID Selectors

- An **id selector** allows you to target a specific element by its id attribute regardless of its type or position

Selectors

ID Selectors

```
<head>
        <title>Share Your Travels </title>
        <style>
                #latestComment {
                 font-style: italic;
                 color: red;
                }
        </style>
</head>



<body>
        <h1 class="first">Reviews</h1>
        <div id="latestComment">
         <p class="first">By Ricardo on <time>2016-05-23</time></p>
         <p>Easy on the HDR buddy.</p>
        </div>
        <hr/>
        <div>
         <p class="first">By Susan on <time>2016-11-18</time></p>
         <p>I love Central Park.</p>
        </div>
        <hr/>
    ody>
```

CSS Selectors (Attributes)

Web System and Tecnologies CS 521

TOPIC 24

CSS Selectors (Attributes)

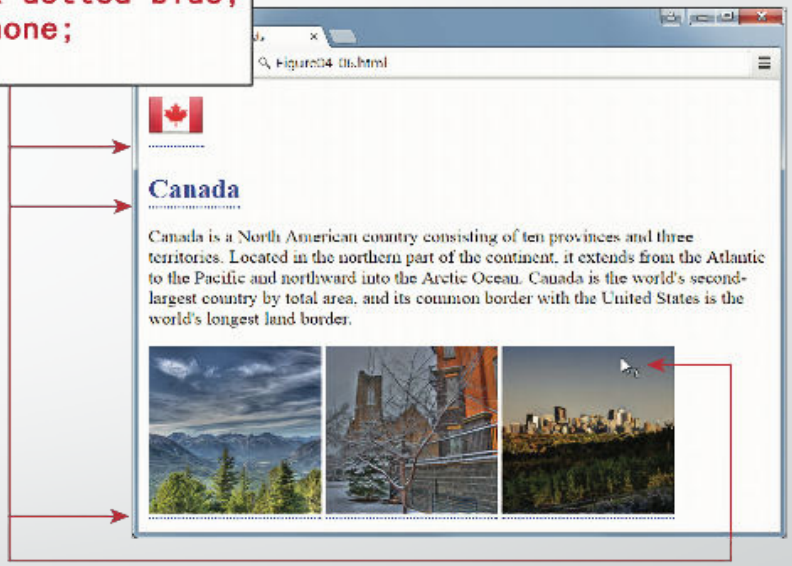
Attribute Selectors

- An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute
    - [title] { … }

CSS Selectors (Attributes)

```
<head>
    <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 Sulphur 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;
}
```



CSS Selectors (Attributes)

Selector	Matches
[]	A specific attribute.
[=]	A specific attribute with a specific value.
[~=]	A specific attribute whose value matches at least one of the words in a space delimited list of words.
[^=]	A specific attribute whose value **begins** with a specified value.
[*=]	A specific attribute whose value **contains** a substring.
[$=]	A specific attribute whose value ends with a specified value.

The End

CSS Selectors (Pseudo)

Web System and Tecnologies CS 521

TOPIC 25

CSS Selectors (Pseudo)

Pseudo-Element and Pseudo-Class Selectors:

- 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

CSS Selectors (Pseudo)

Pseudo-Classes and -Elements

- Pseudo-Classes
  - a:link
  - a:visited
  - :hover
  - :active
  - :focus
  - :checked
  - :first-child
- Pseudo-Elements
  - ::first-letter
  - ::first-line
  - ::before
  - ::after
  - ::marker

CSS Selectors (Pseudo)

Pseudo-Element and Pseudo-Class Selectors:

```
<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>
```

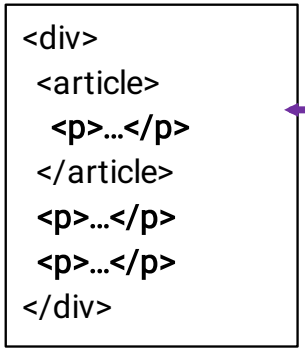
The End

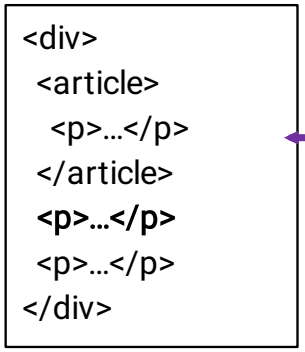
CSS Selectors (Contextual)

Web System and Tecnologies CS 521

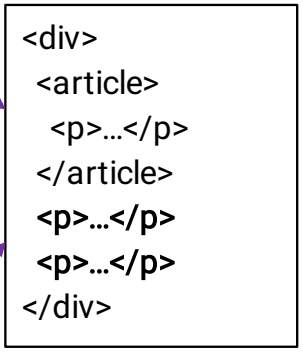
TOPIC 26

CSS Selectors (Contextual)

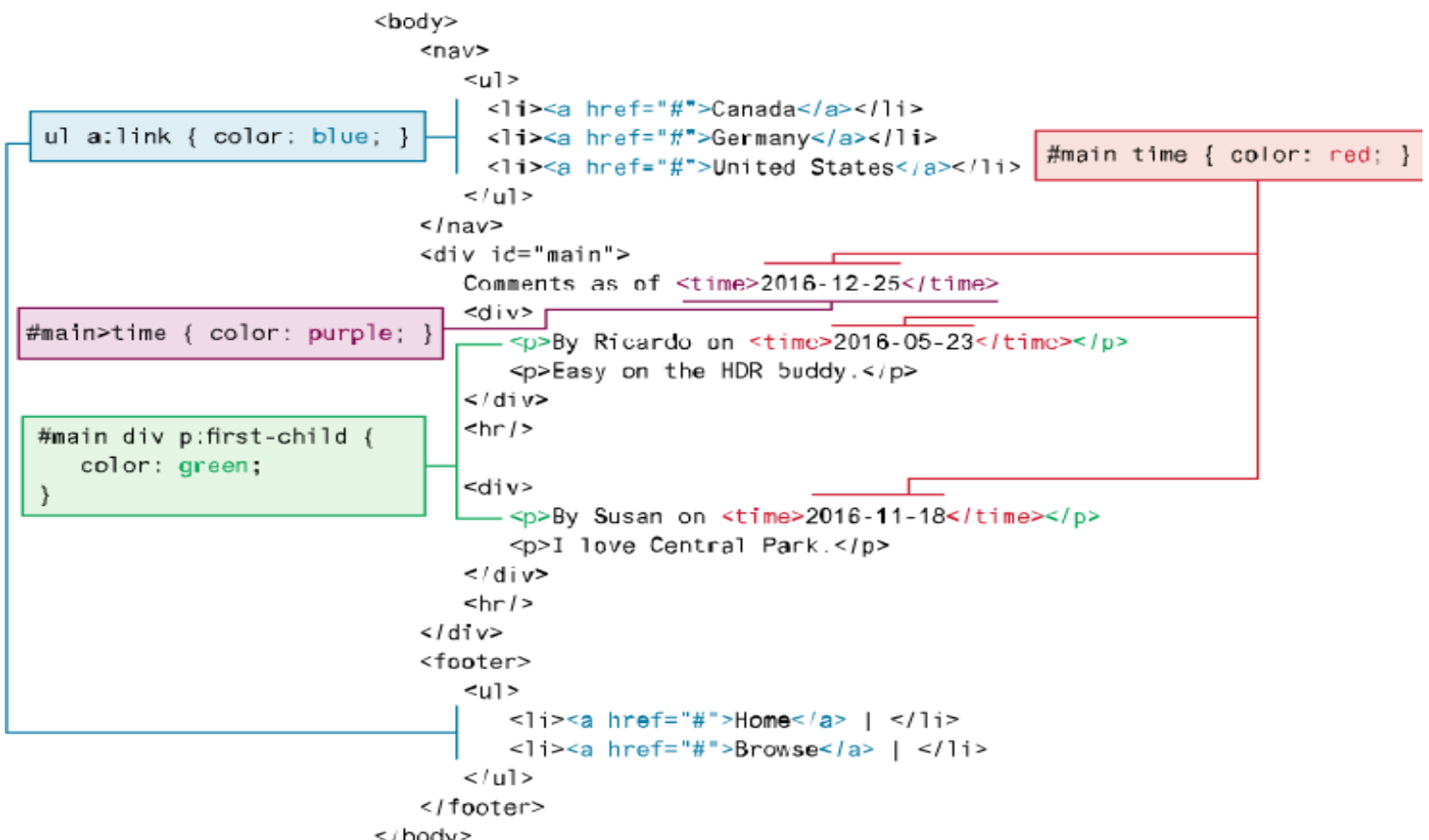




Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	**div p** Selects a <p> element that is contained somewhere within a element.
Child	A specified element that is a direct child of the specified element.	**div>p** Selects a <p> element that is a child of a 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>.



CSS Selectors (Contextual)



The End

The Cascade: How Styles Interact

Web System and Tecnologies CS 521

TOPIC 27

The Cascade: How Styles Interact

CSS:

- The “Cascade" in CSS refers to how conflicting rules are handled.
- 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
- CSS uses the following cascade principles to help it deal with conflicts:
  - Inheritance
  - Specificity
  - Location

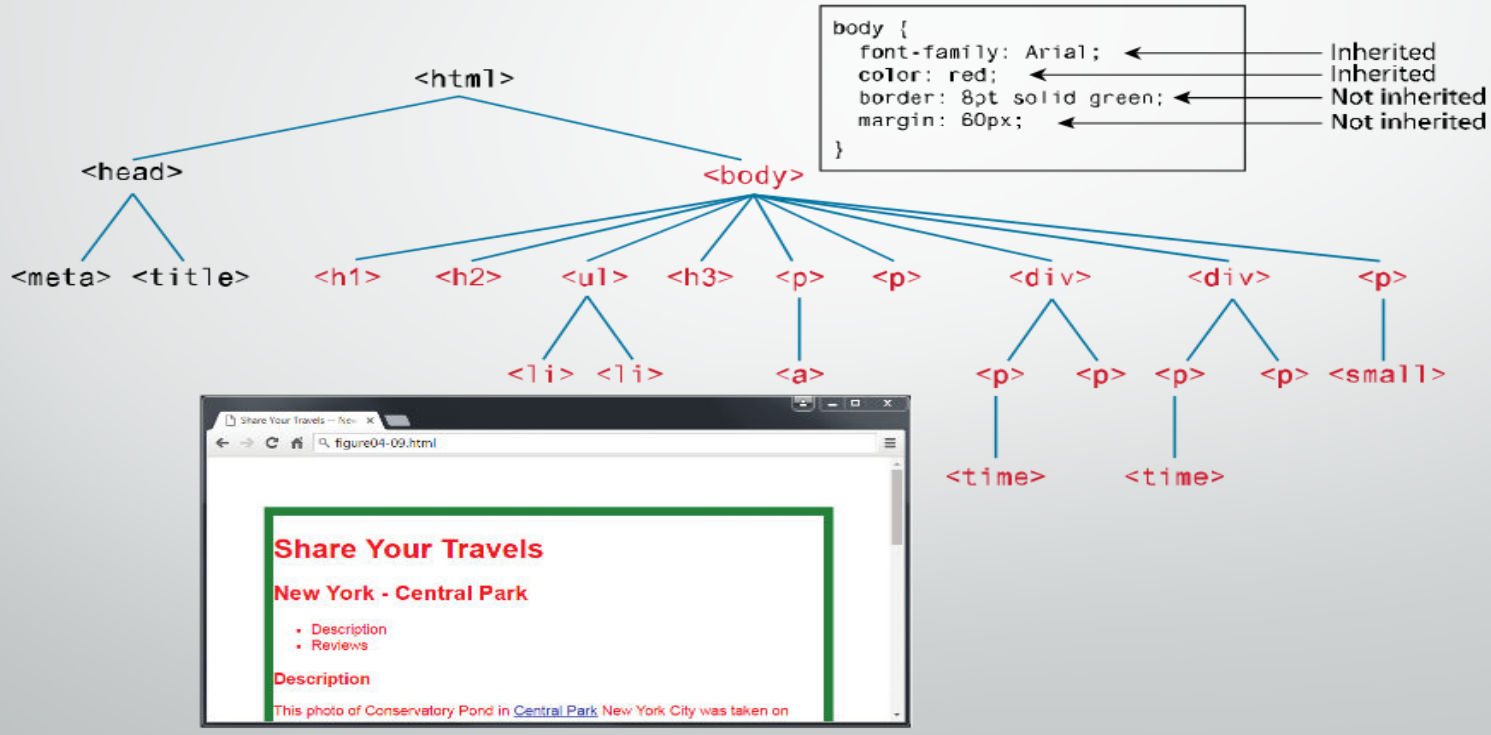
The Cascade: How Styles Interact

Inheritance:

- Many (but not all) CSS properties affect not only themselves but their descendants as well
- Inheritable:
  - font, color, list, and text properties
- Not inheritable:
  - layout, sizing, border, background, and spacing properties

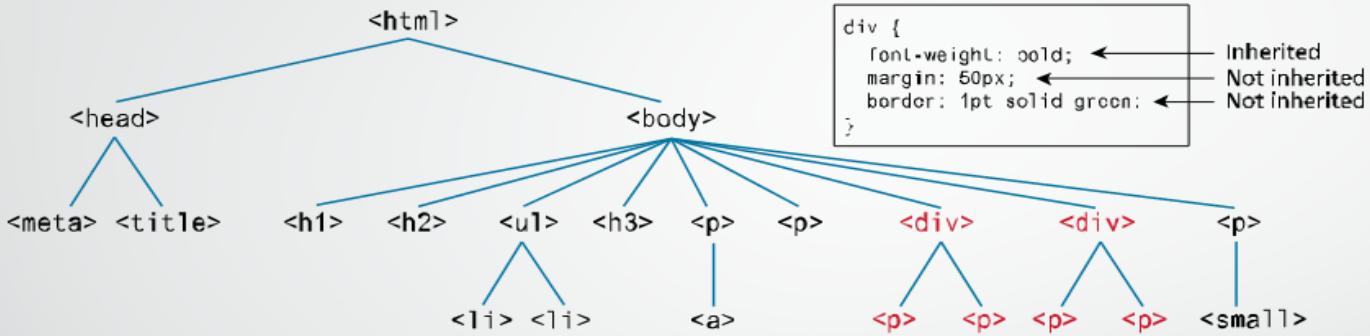
The Cascade: How Styles Interact

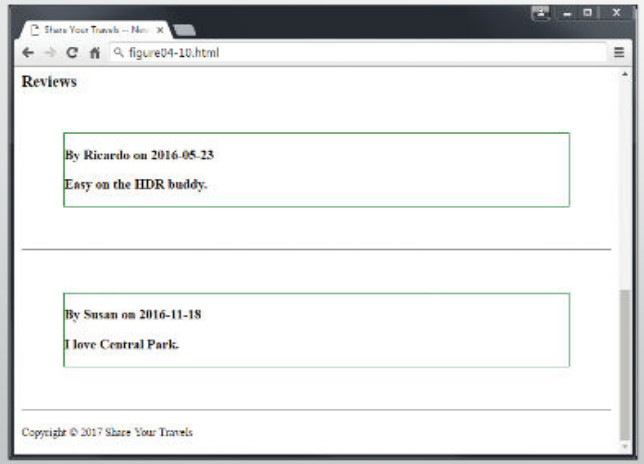
Inheritance:



The Cascade: How Styles Interact

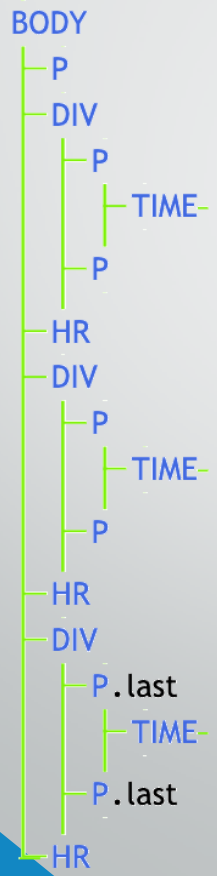
Inheritance:





The Cascade: How Styles Interact

Specificity:




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

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

p {
  color: green;
}

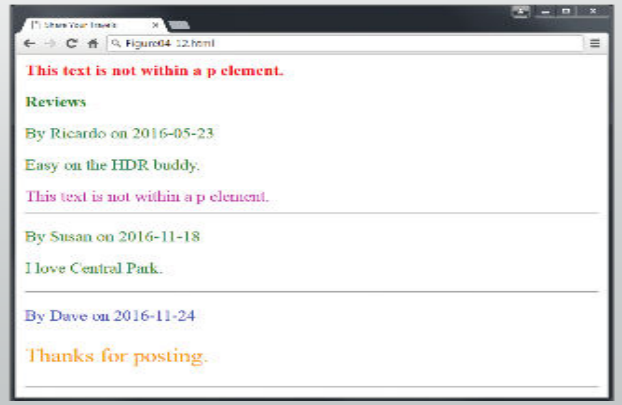
.last {
  color: blue;
}

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

```
<body>
   This text is not within a p element.
   <p>Reviews</p>
   <div>
      <p>By Ricardo on <time>2016-05-23</time></p>
      <p>Easy on the HDR buddy.</p>
      This text is not within a p element.
   </div>
   <hr/>

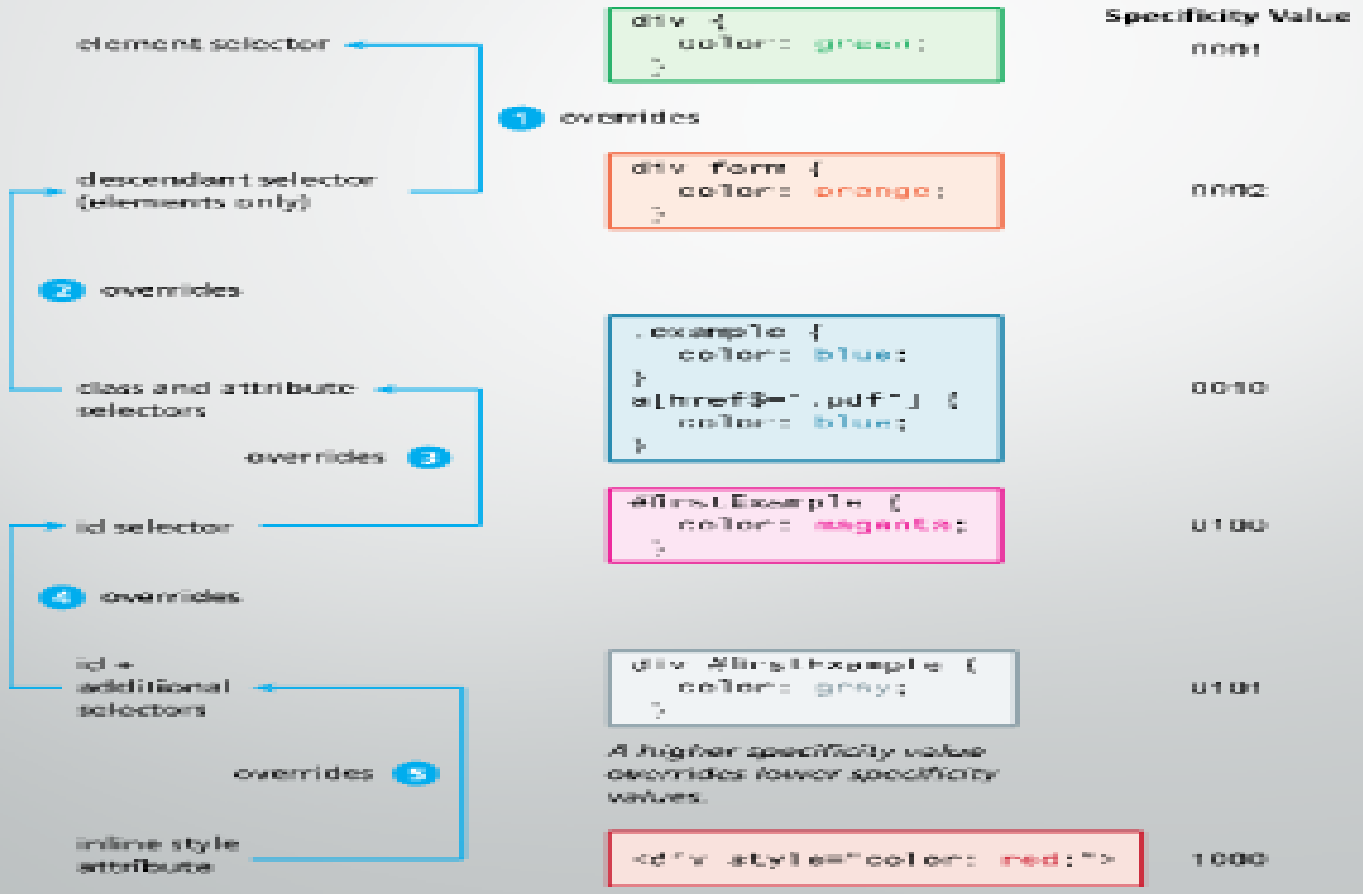
   <div>
      <p>By Susan on <time>2016-11-18</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>

   <div>
      <p class="last">By Dave on <time>2016-11-24</time></p>
      <p class="last" id="verylast">Thanks for posting.</p>
   </div>
   <hr/>
</body>
```



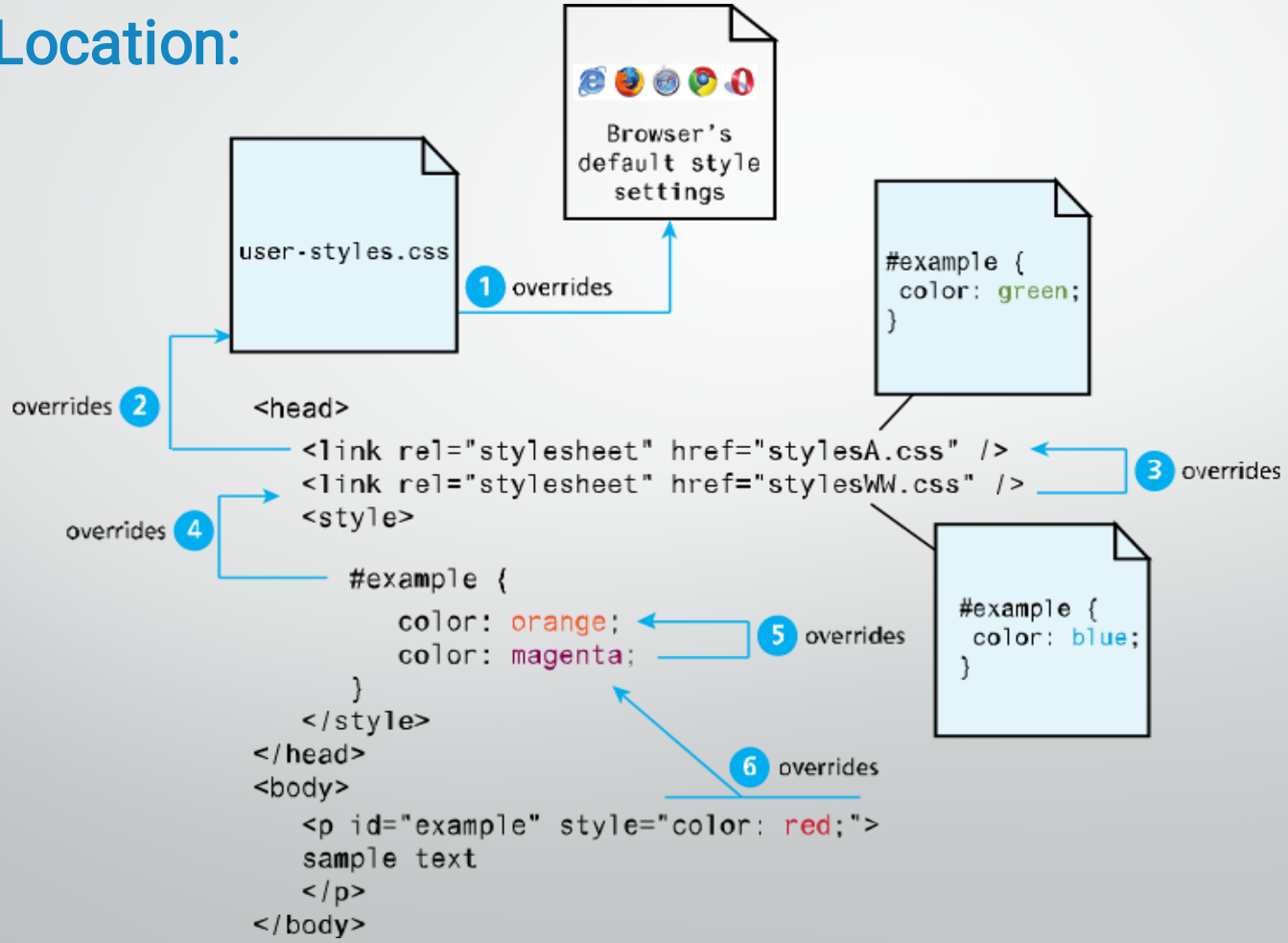
The Cascade: How Styles Interact

Specificity Algorithm:



The Cascade: How Styles Interact

Location:



The End



The Box Mode

TOPIC 28

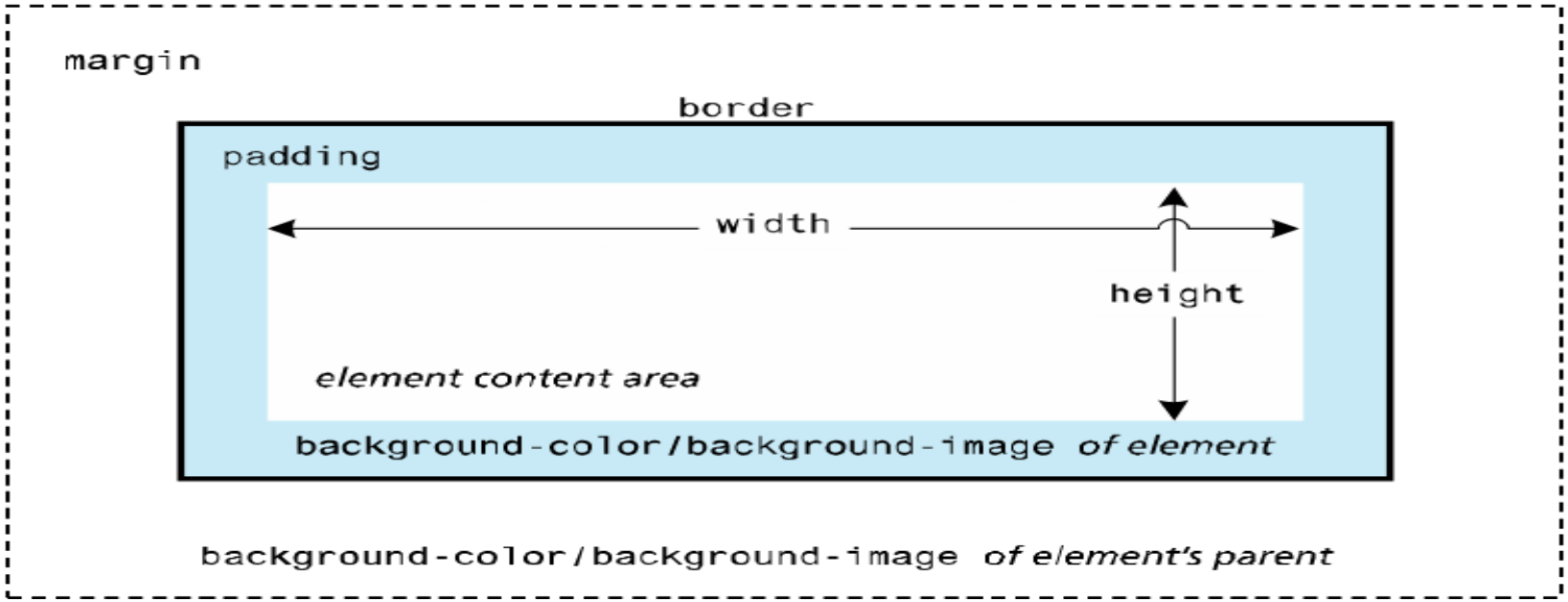
WITH

VOICE NOTE

Web System and Tecnologies CS 521

The Box Model

Overview:



The Box Model

Background:

- Background
- Background-attachment
- Background-color
- Background-image
- Background-position
- Background-repeat
- Background-size

The Box Model

BackGround:





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



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



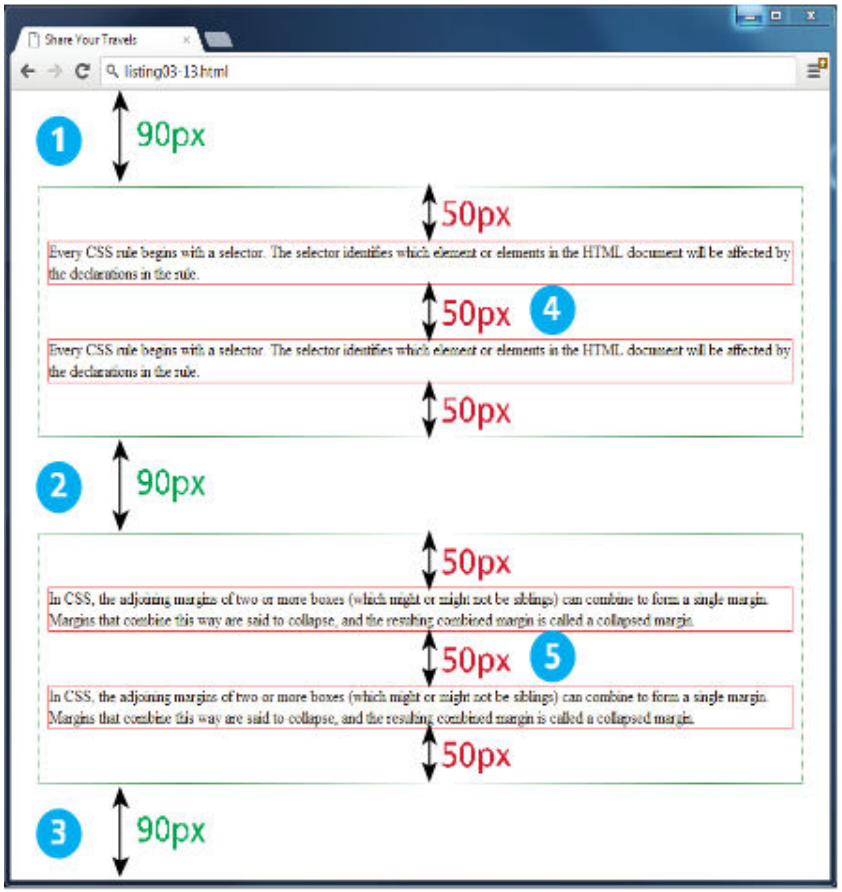
```
background-repeat: repeat-y;
```



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

The Box Model

Collapsing Margins:

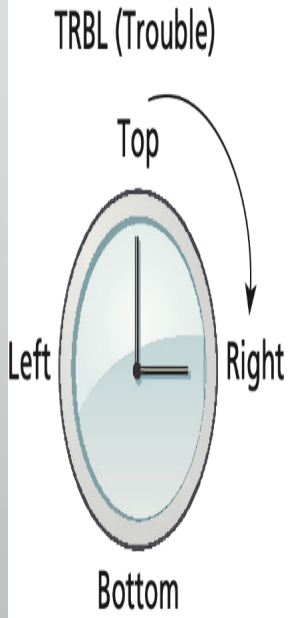



```
<div>
  <p>Every CSS rule ...</p>
  <p>Every CSS rule ...</p>
</div>
<div>
  <p>In CSS, the adjoining ... </p>
  <p>In CSS, the adjoining ... </p>
</div>
```

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

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

The Box Model




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

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

CSS TRBL Shortcuts:

- 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 */
- Alternately, we can set all four sides at once:
- border-color: red; /* sets all four sides to red */
- border-color: red green orange blue; /* sets 4 colors */

The End

The Box Model

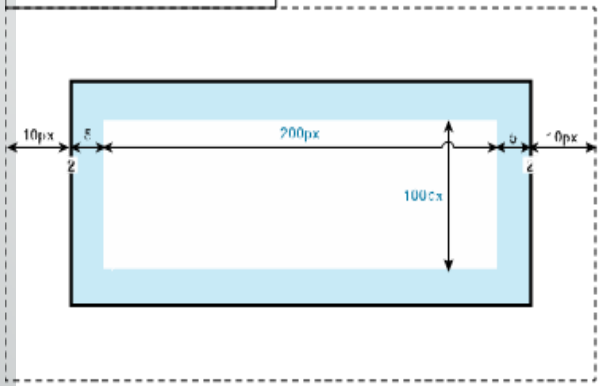
Web System and Tecnologies CS 521

TOPIC 29

The Box Model

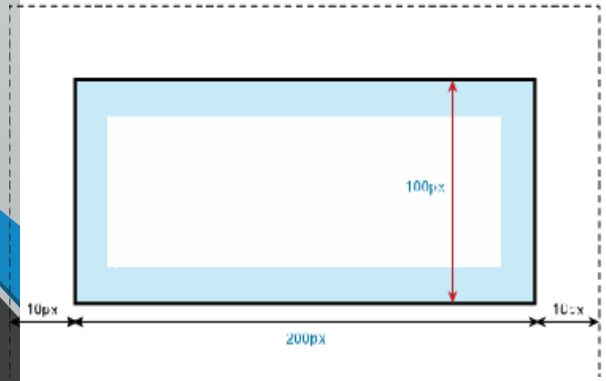
```
div {
  box-sizing: content-box;
  width: 200px;
  height: 100px;
  padding: 5px;
  margin: 10px;
  border: solid 2pt black;
}
```

True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px




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

True element width = 10 + 200 + 10 = 220 px
True element height = 10 + 100 + 10 = 120 px

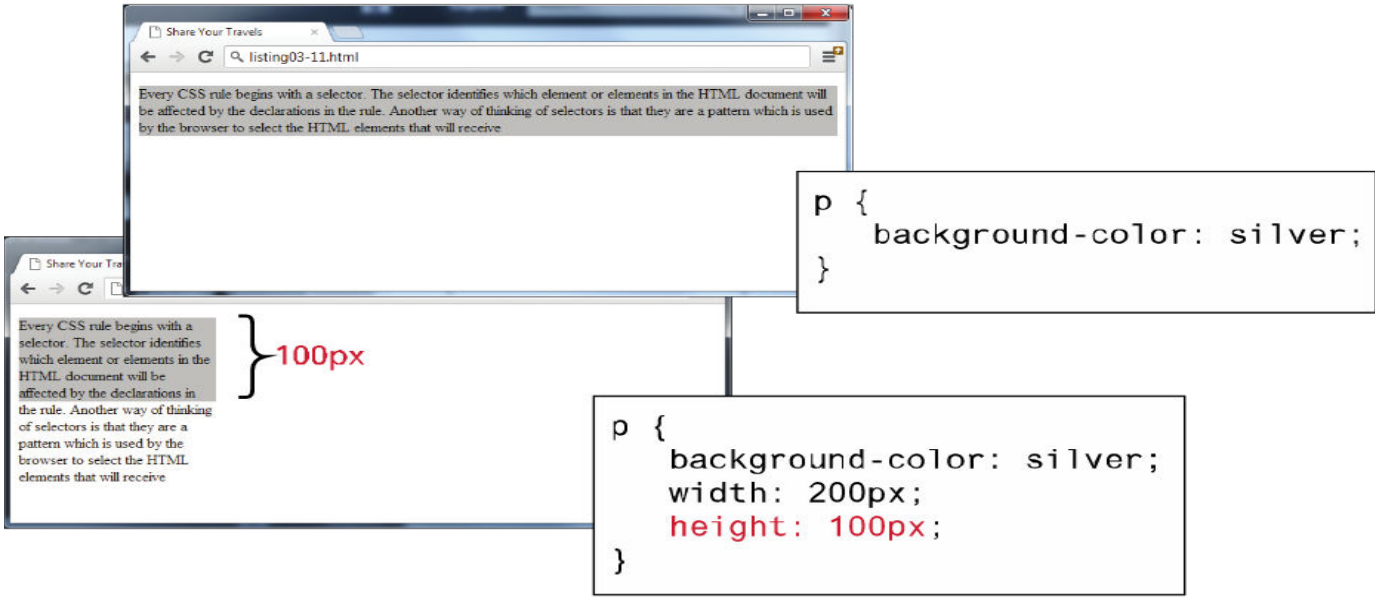


Box Dimensions:

- Content-box
  - width/height applied to contents only
  - borders and padding counted separately
  - margins counted separately
- Border-box
  - width/height include border and padding
  - margins counted separately

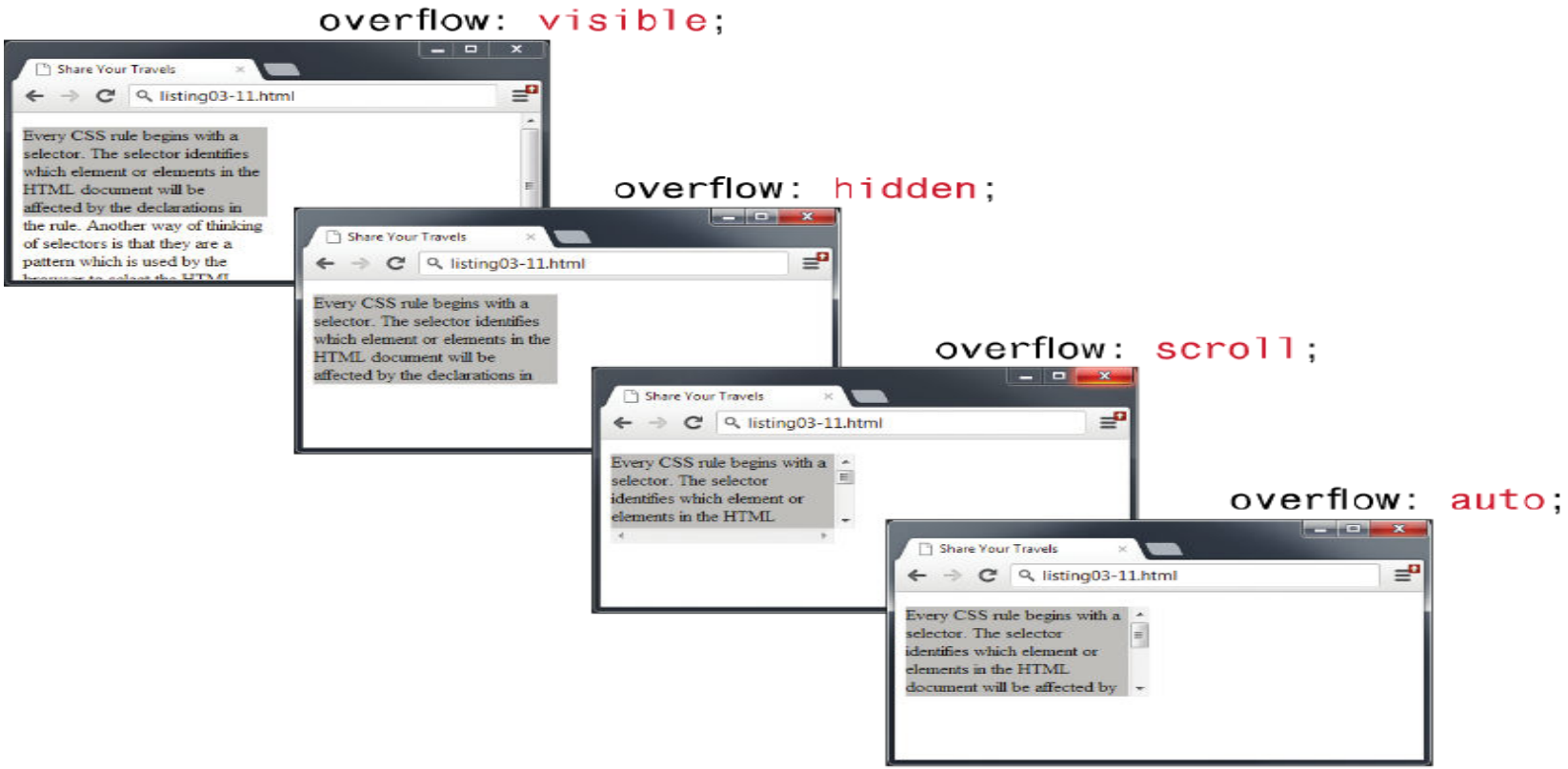
The Box Model

Limitation of Height Property:



The Box Model

OverFlow Property:



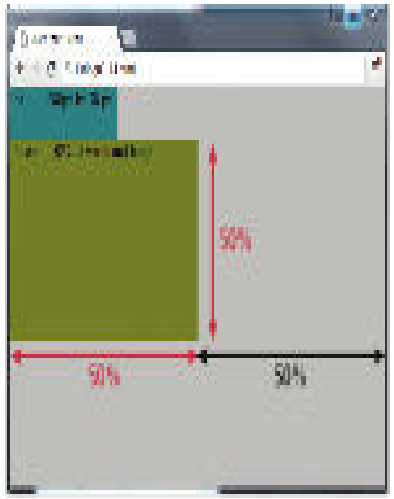
The Box Model

Box sizing Using Percentage:

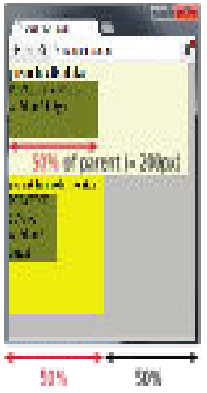
```
<style>
  html,body {
    margin:0;
    width:100%;
    height:100%;
    background: silver;
  }
  .pixels {
    width:200px;
    height:50px;
    background: teal;
  }
  .percent {
    width:50%;
    height:50%;
    background: olive;
  }
```




```
<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>
```



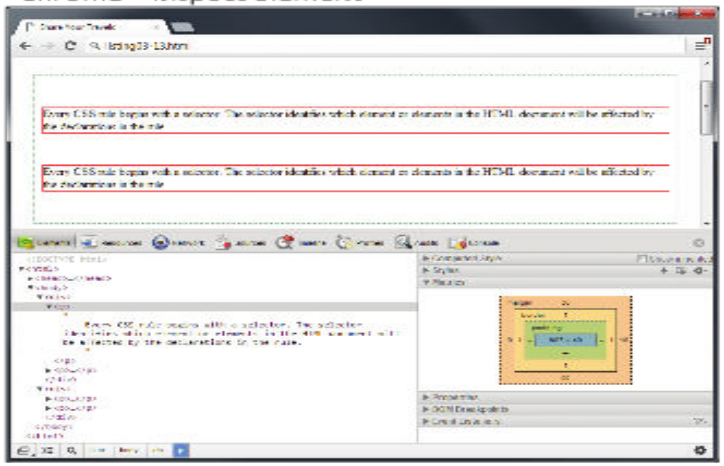

```
<body>
<div class="parentFixed">
  <strong>parent has fixed size</strong>
  <div class="percent">
    PERCENT - 50% of width and height
  </div>
</div>
<div class="parentRelative">
  <strong>parent has relative size</strong>
  <div class="percent">
    PERCENT - 50% of width and height
  </div>
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



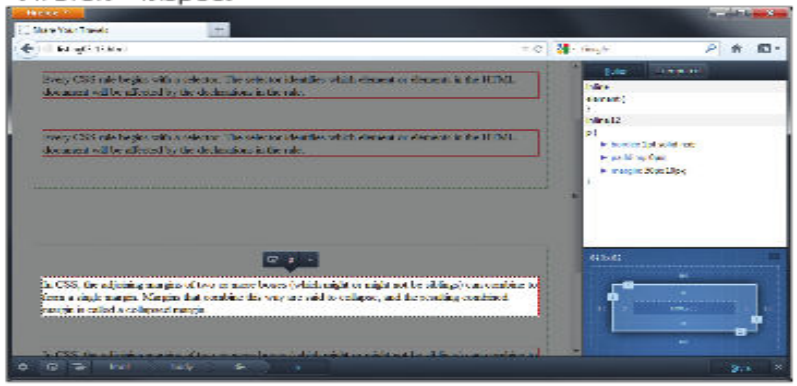
The Box Model

Inspecting CSS using Modern Browsers:

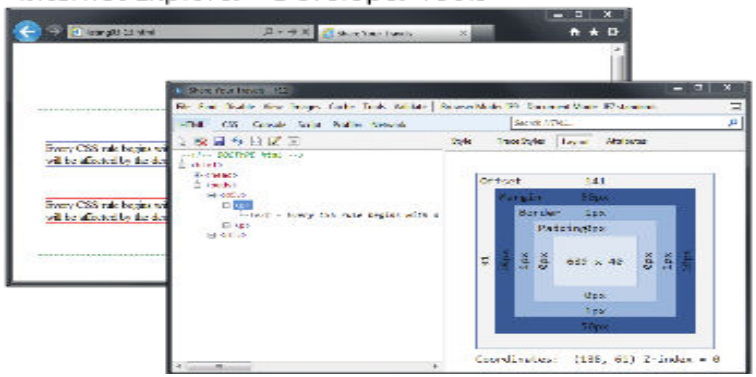
Chrome – Inspect Element



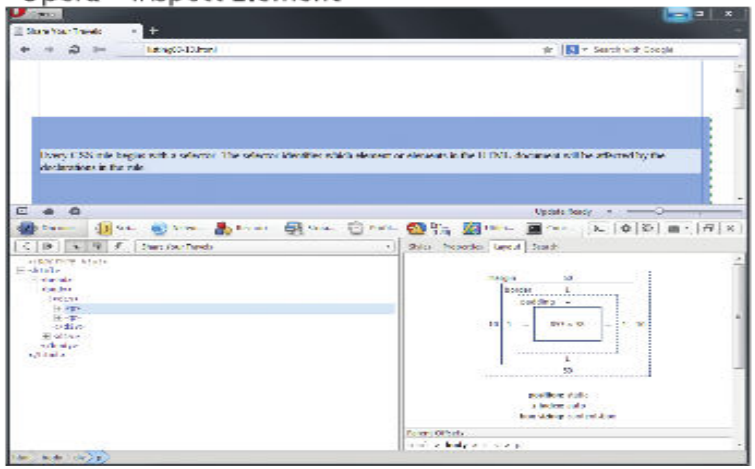
Firefox – Inspect



Internet Explorer – Developer Tools



Opera – Inspect Element



The End



CSS Text Styling

TOPIC 30

WITH

VOICE NOTE

Web System and Tecnologies CS 521

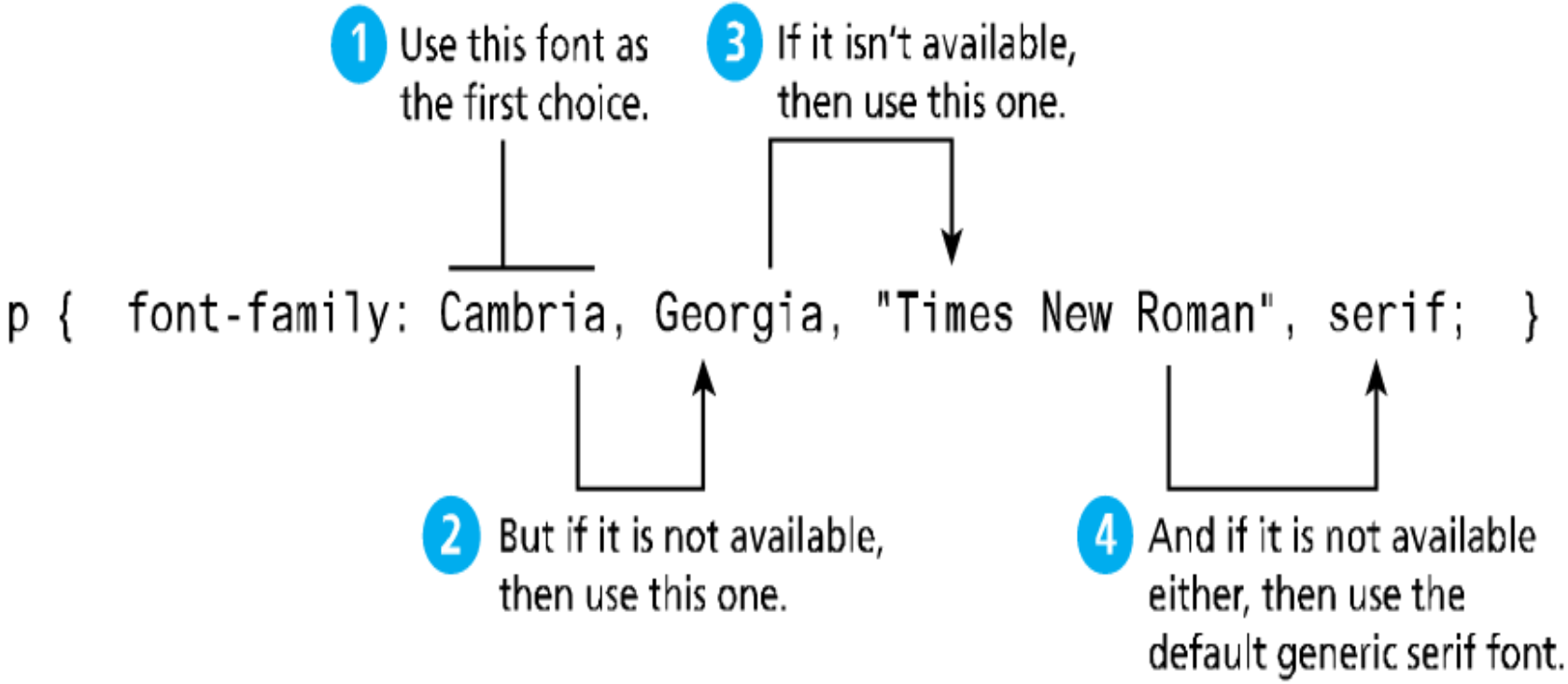
CSS Text Styling

Font Family:

Property	Description
font	A combined shorthand property that allows you to set the family, style, size, variant, and weight in one property. **style weight variant size font-family**
font-family	Specifies the typeface/font to use. More than one can be specified.
font-size	The size of the font in one of the measurement units
font-style	Specifies whether italic, oblique, or normal
font-variant	Specifies either small-caps text or none
font-weight	Specifies either normal, bold, bolder, lighter, or a value between 100 and 900 in multiples of 100, where larger number represents weightier (i.e., bolder) text.

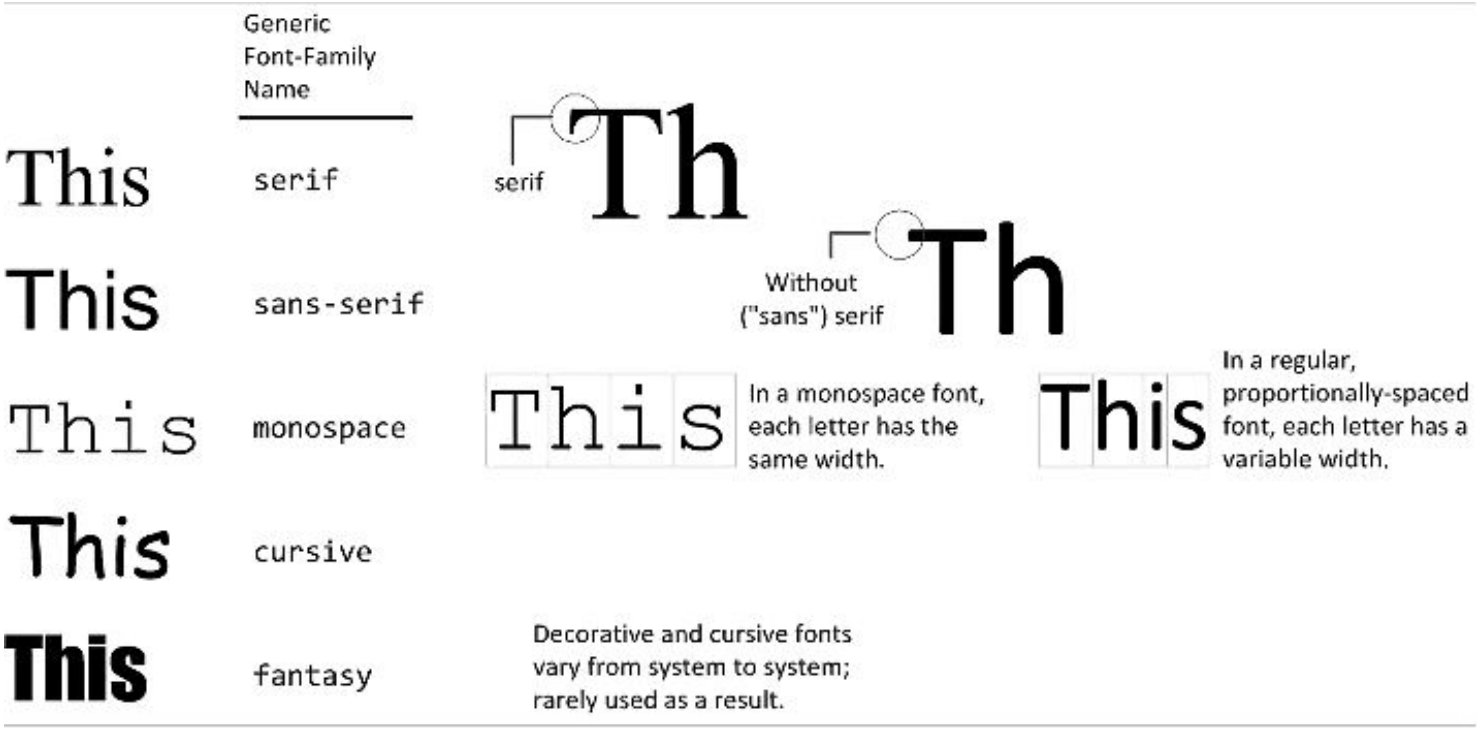
CSS Text Styling

Specifying the Font Family:



CSS Text Styling

Different Font Families:



CSS Text Styling

Font Sizes:

<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

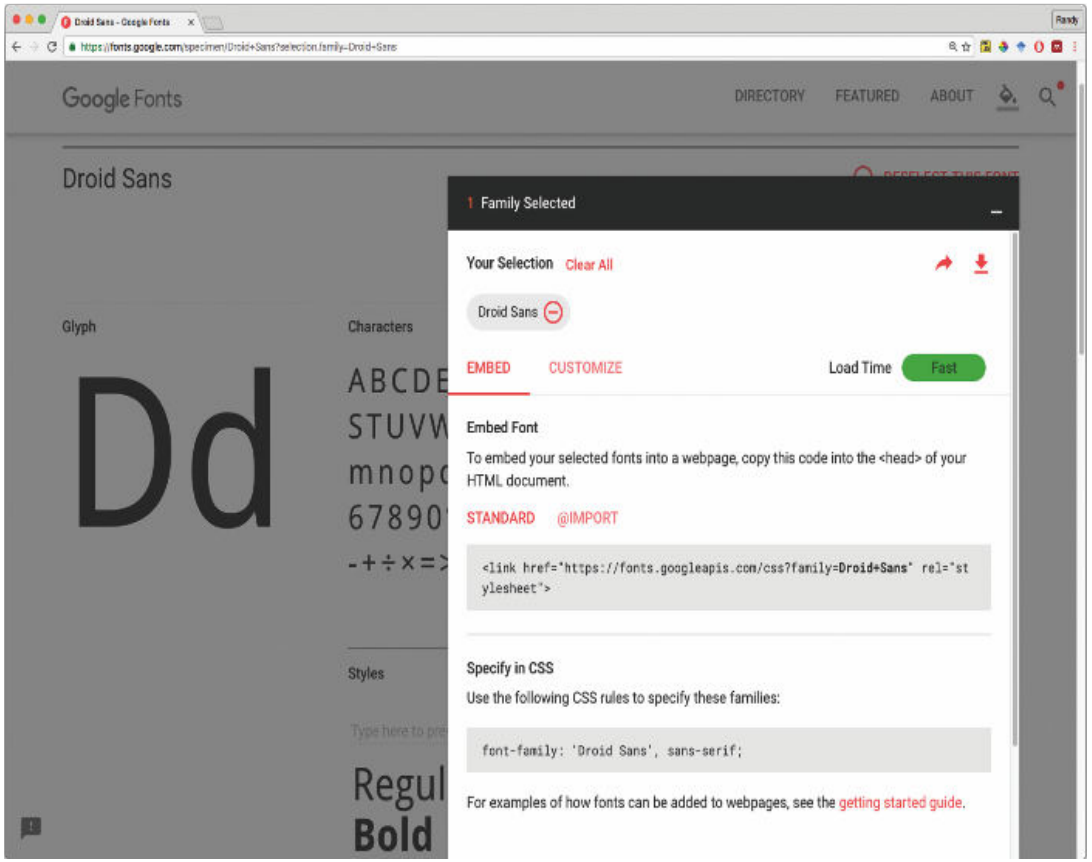
```
/* using 16px scale */

body { font-size: 100%; }
p { font-size: 1em; }        /* 1.0   x 16 = 16 */
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>
  Browser's default text size is usually 16 pixels
  <p>100% or 1em is 16 pixels</p>
  <h3>125% or 1.125em is 18 pixels</h3>
  <h2>150% or 1.5em is 24 pixels</h2>
  <h1>200% or 2em is 32 pixels</h1>
</body>
```

CSS Text Styling

Embeded Fonts:



CSS Text Styling

Paragraph Properties:

- Just as there are properties that affect the font in CSS, there are also a range of CSS properties that affect text independently of the font.
  - letter-spacing
  - line-height
  - text-align
  - text-decoration
  - text-direction
  - text-shadow
  - …

CSS Text Styling

Just look at Text Shadow:



You will likely want the shadow color to be partly transparent

y offset
x offset
blur size
shadow color

1 `text-shadow: 20px 20px 10px rgba(0,0.0,0.5);`

2
```
text-shadow: 4px 4px 0 #5C6BC0,
             8px 8px 0 #7986CB,
            12px 12px 0 #9FA8DA;
```

multiple shadows can be defined (separated by commas)

3 `text-shadow: 0 1px 1px #1A237E;`

4 `box-shadow: 0px 0px 30px #1A237E;`

box shadows work in the same way as text shadows

The End

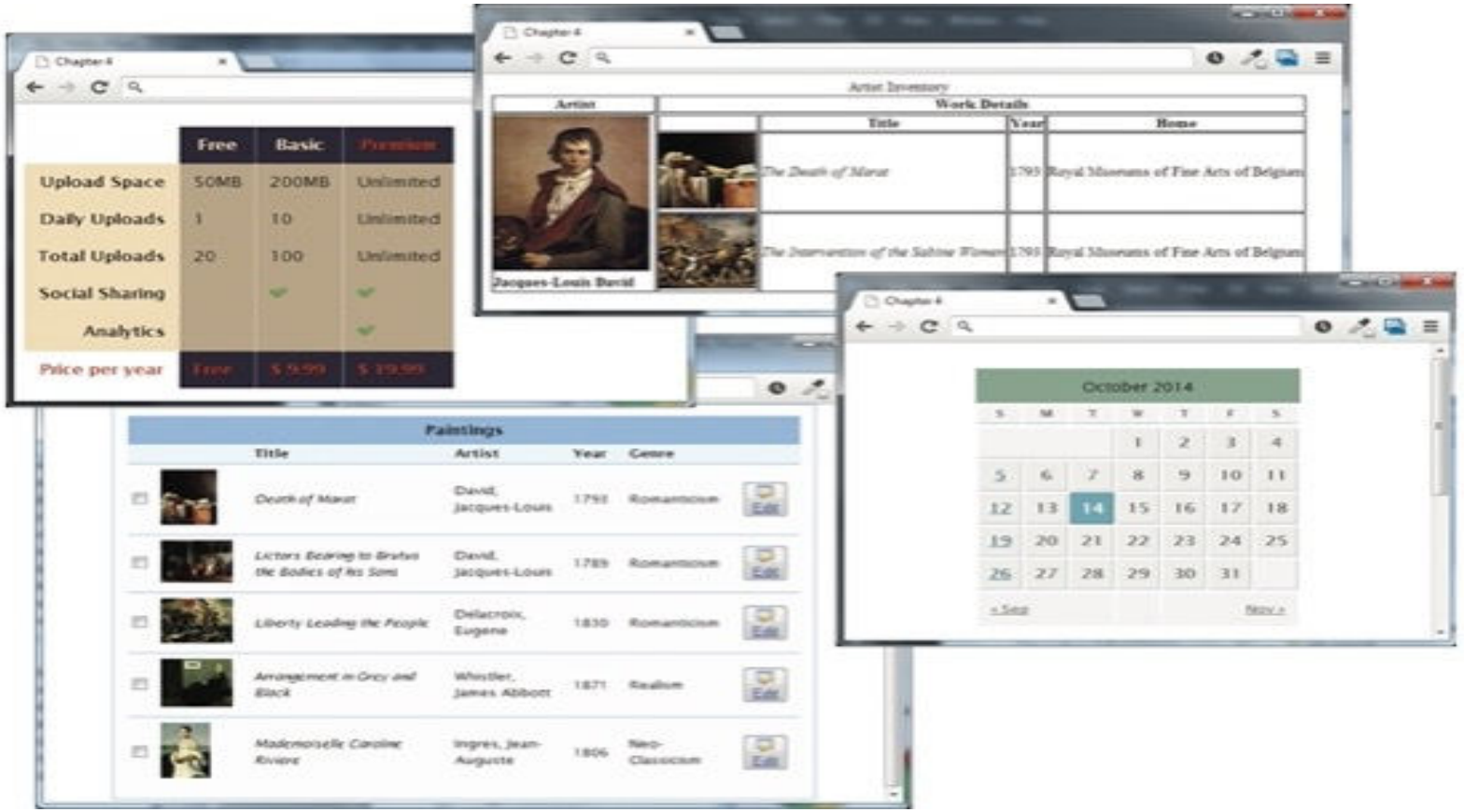
Introducing Tables

Web System and Tecnologies CS 521

TOPIC 31

Introducing Tables

Basic Table Structure:



Introducing Tables

Basic Table Structure:

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

```
<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>
```



Introducing Tables

Adding Headings:

<table>

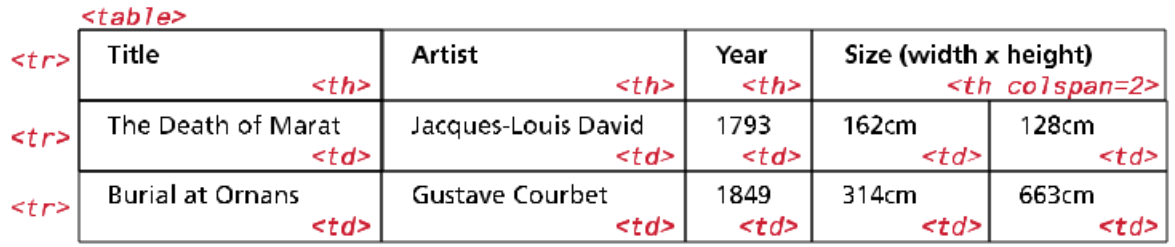
<tr>	Title <th>	Artist <th>	Year <th>	Width <th>	Height <th>
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<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>
```



Introducing Tables

Spanning Columns:



Notice that this row now only has four cell elements.

```
<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>
```

Introducing Tables

Spanning Rows:

<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>Title</th>
        <th>Artist</th>
        <th>Year</th>
    </tr>
    <tr>
        <td rowspan="3">Jacques-Louis David</td>
        <td>The Death of Marat</td>
        <td>1793</td>
    </lr>
    <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.

The End

Web Systems and Technologies

TOPIC 32

Introducing Tables (2)

Introducing Tables

- **Using Tables for Layout**






```
<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/484.jpg" /></td>
          <td><img src="images 537.jpg"  ></td>
        </tr>
        <tr>
          <td><img src="images/700.jpg" /></td>
          <td><img src='images/829.jpg' /></ td>
        </tr>
      </table>
    </td>
  < tr>
</table>
```

Introducing Tables

- ## Additional Table Elements

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>
```



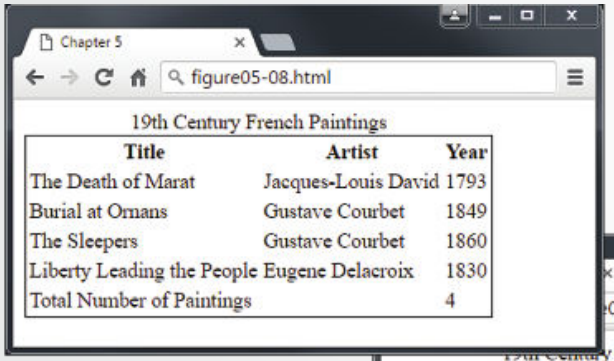
Web Systems and Technologies

TOPIC 33

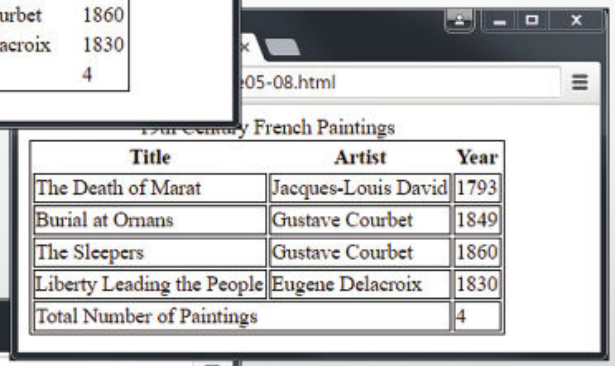
Styling Tables

Styling Tables

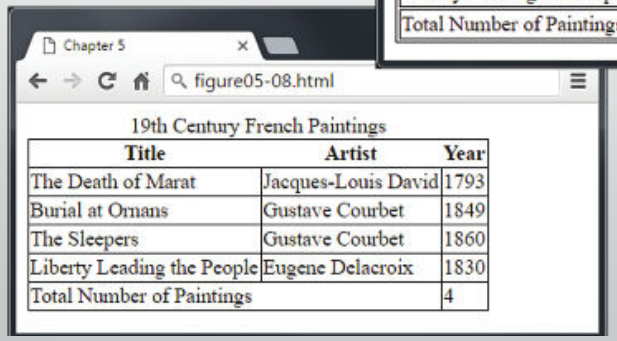
- Table Borders




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



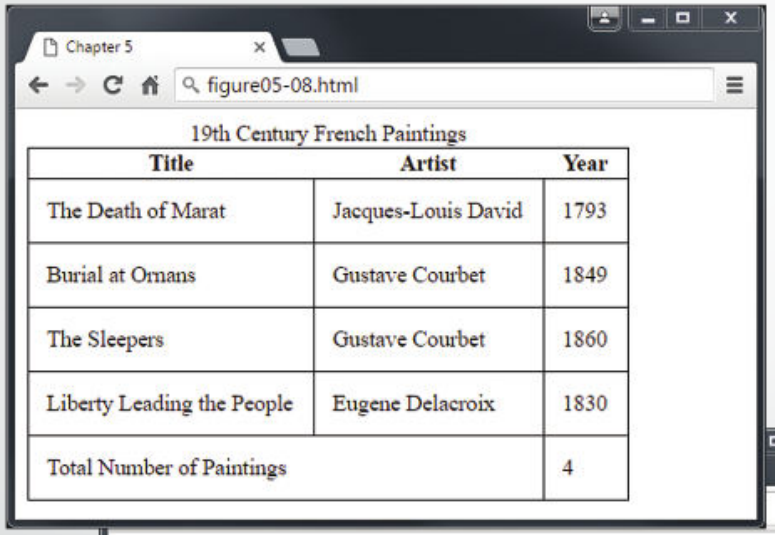

```
table {
   border: solid 1pt black;
}
td {
   border: solid 1pt black;
}
```



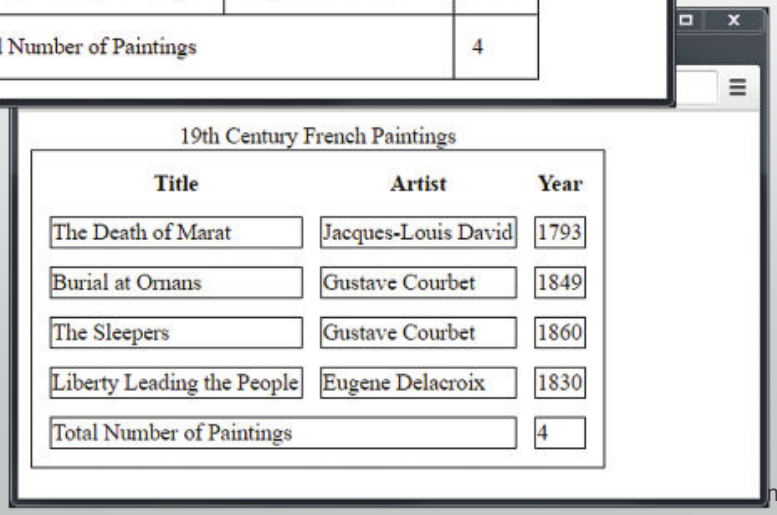

```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
}
```

Styling Tables

- Table Borders



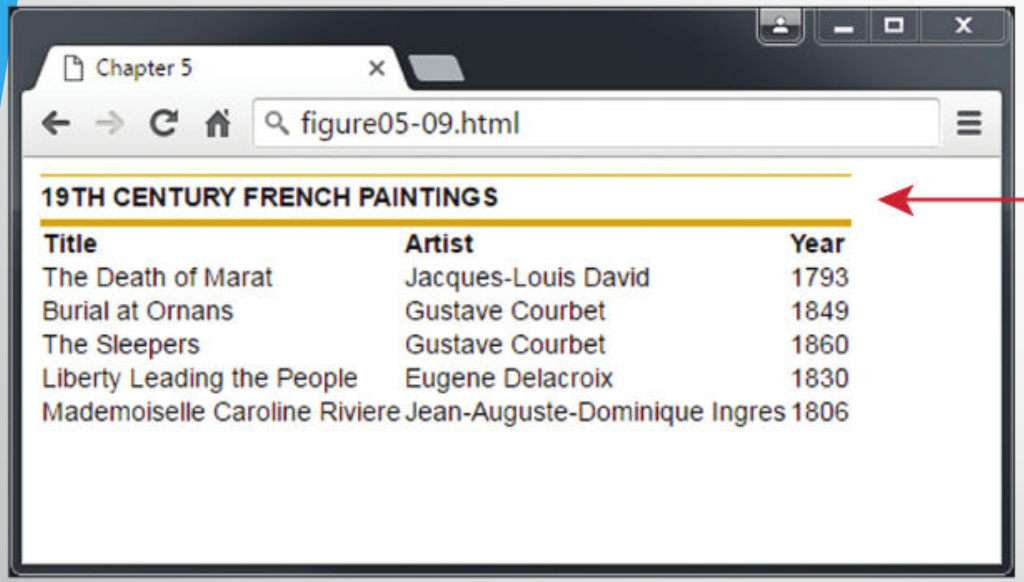

```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
   padding: 10pt;
}
```




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

Styling Tables

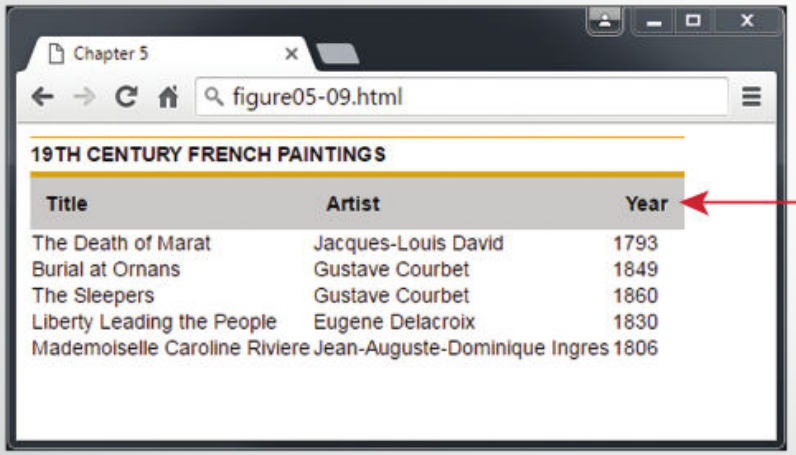
- **Boxes**




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

Styling Tables

- Boxes




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

Styling Tables

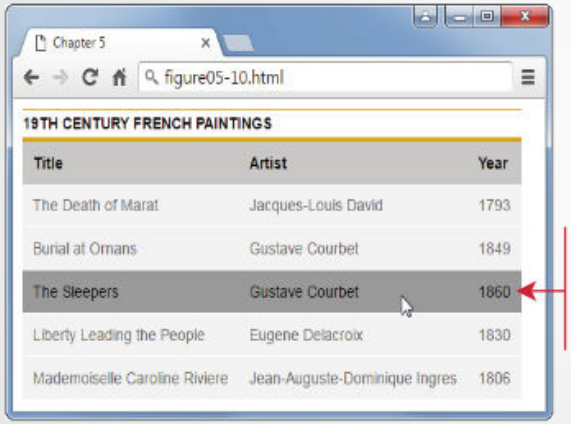
- **Boxes**



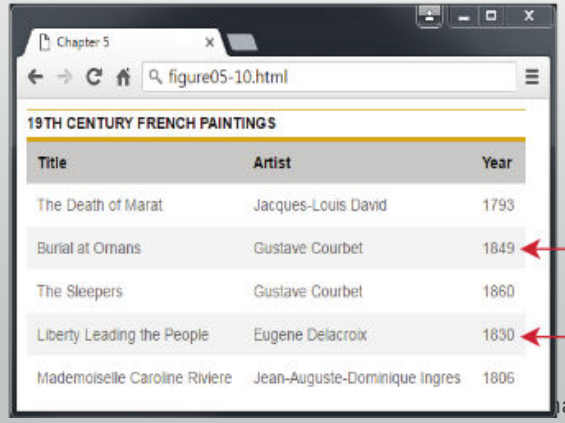

```
tbody tr {
   background-color: #F1F1F1;
   border-bottom: 1px solid white;
   color: #6E6E6E;
}
tbody td {
   padding: 0.75em;
}
```

Styling Tables

- Hover & Zebras




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




```
tbody tr:nth-child(even) {
   background-color: lightgray;
}
```

Styling Tables

- **Accessible Tables**
- Use tables for data, not layout
- Use the <caption> element
- Connect cells with a textual description in the header

```
<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>
```

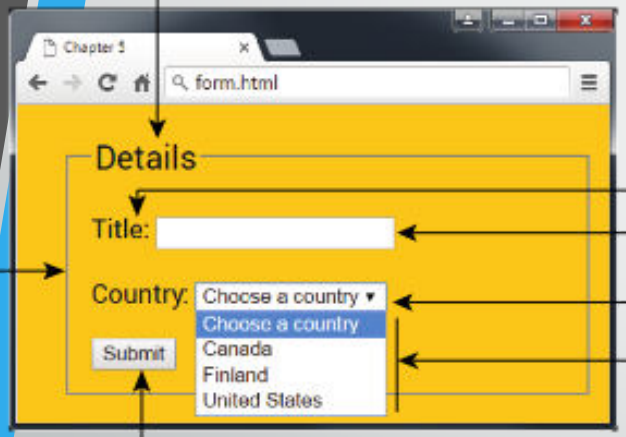
Web Systems and Technologies

TOPIC 34

Introducing Forms

Introducing Forms

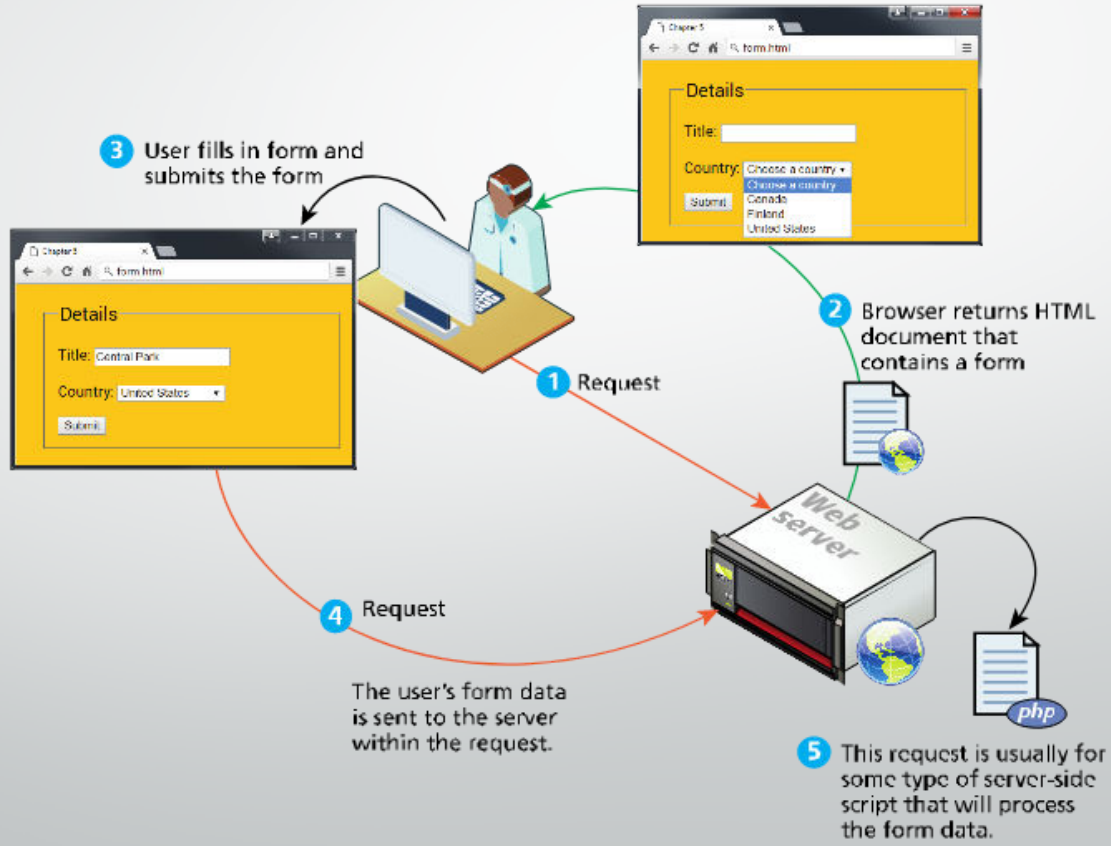
- Form Structure




```
<form method="post" 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>
```

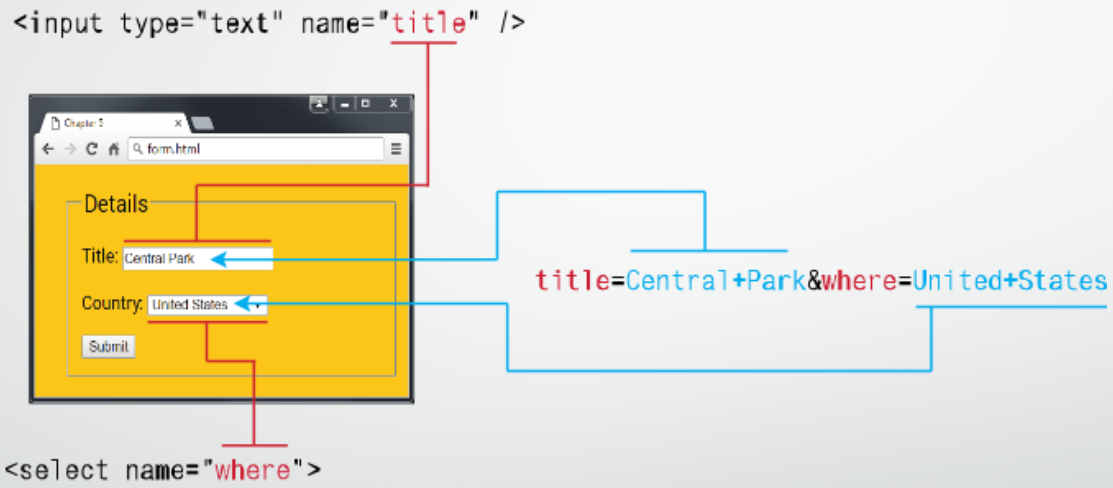
Introducing Forms

- How Forms Work



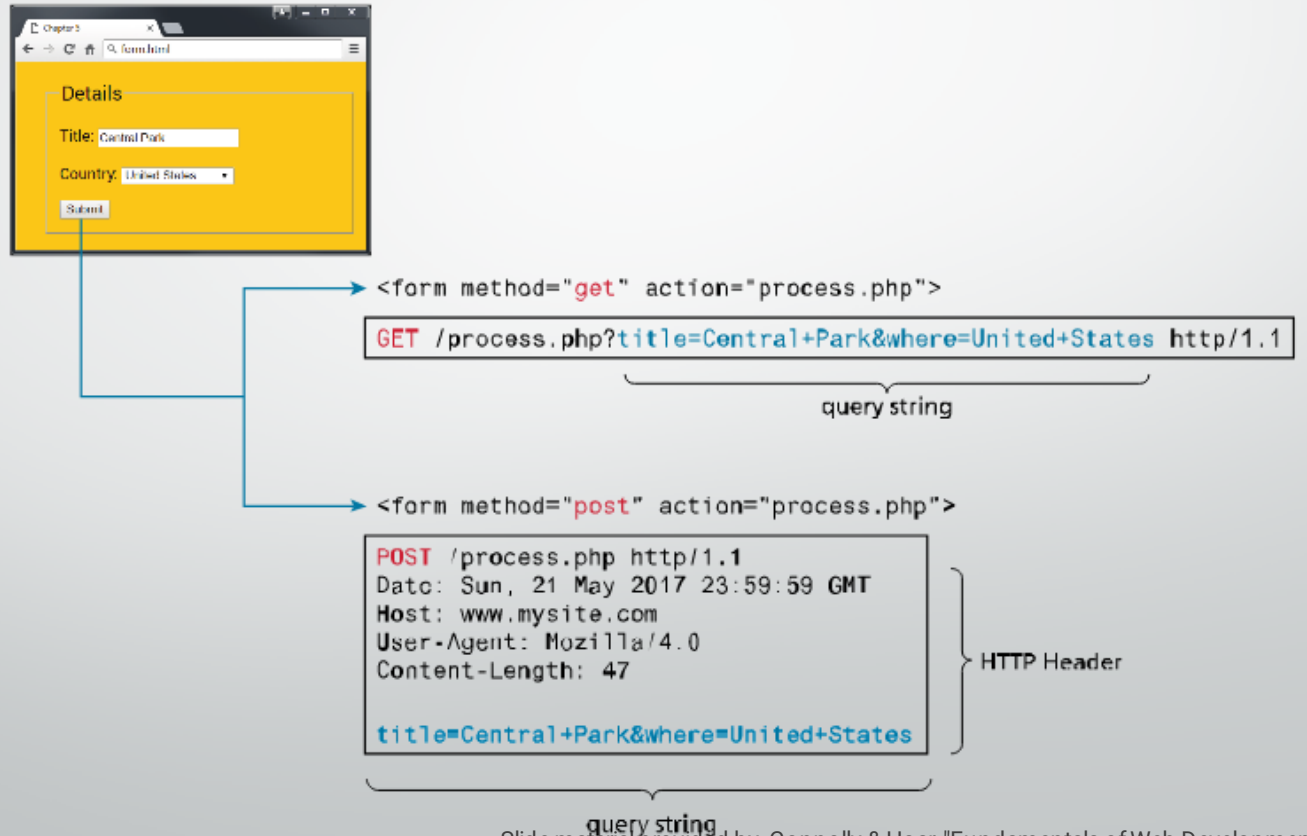
Introducing Forms

- **Query Strings**



Introducing Forms

- GET and POST



Introducing Forms

- HTML Form elements

  <form>
  <button>
  <input>
  <label>
  <datalist>
  <fieldset>
  <legend>
  <select>
  <option>
  <optgroup>
  <textarea>
  <output>

Web Systems and Technologies

TOPIC 35

Form Control Elements (1)

Form Control Elements

- Text Input Controls

text



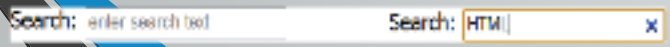
textarea



password



search



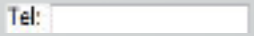
email



url



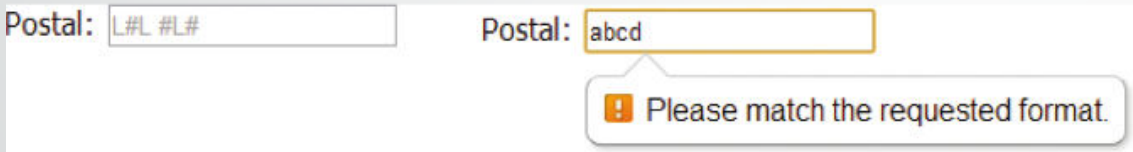
tel

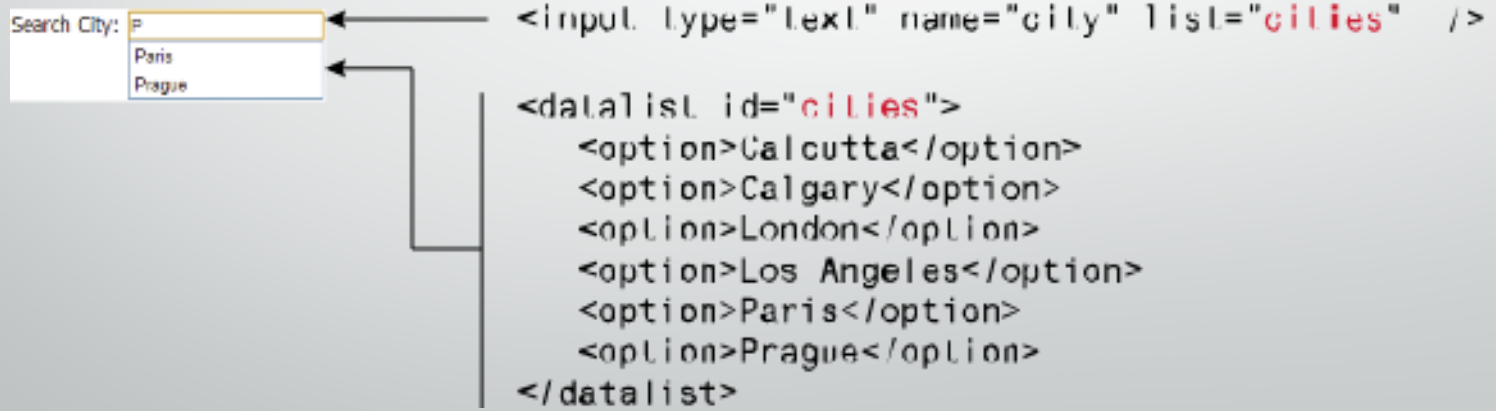


Form Control Elements

- New in HTML5 – pattern and datalist

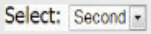
```
<input type="text" ... placeholder="L#L #L#" pattern="[a-z][0-9][a-z] [0-9][a-z][0-9]"  />
```

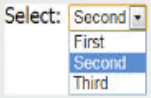




Form Control Elements

- **Choice Controls (<select>)**

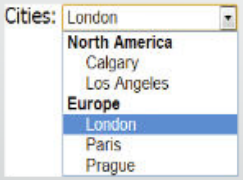





```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```



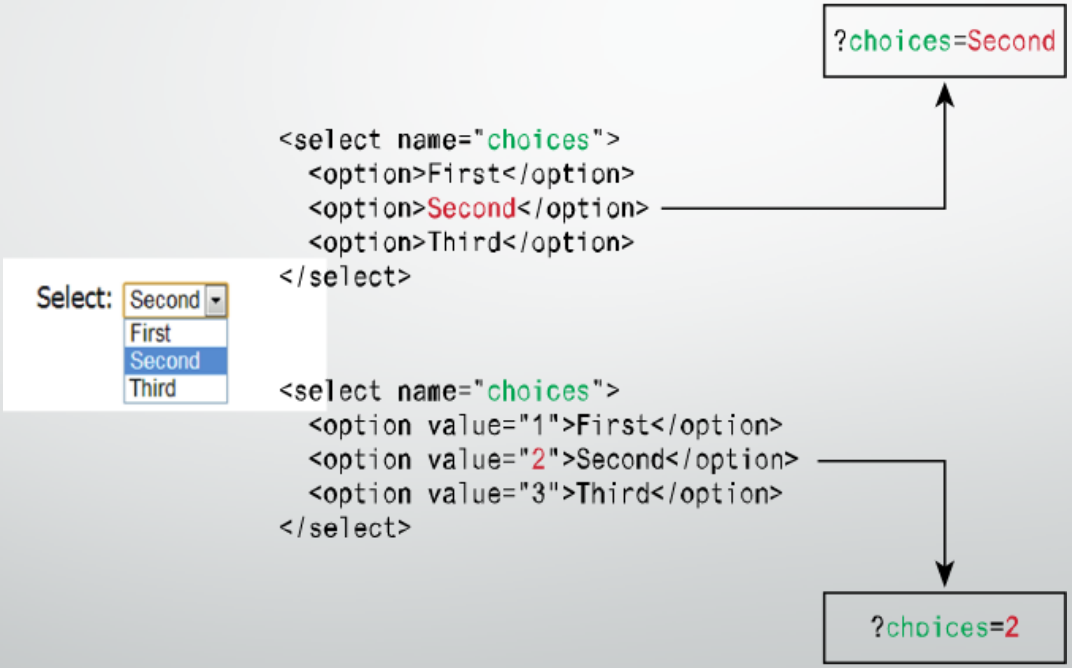

```
<select size="3" ... >
```




```
<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>
```

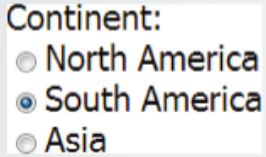
Form Control Elements

- **Choice Controls <select> using value attribute**



Form Control Elements

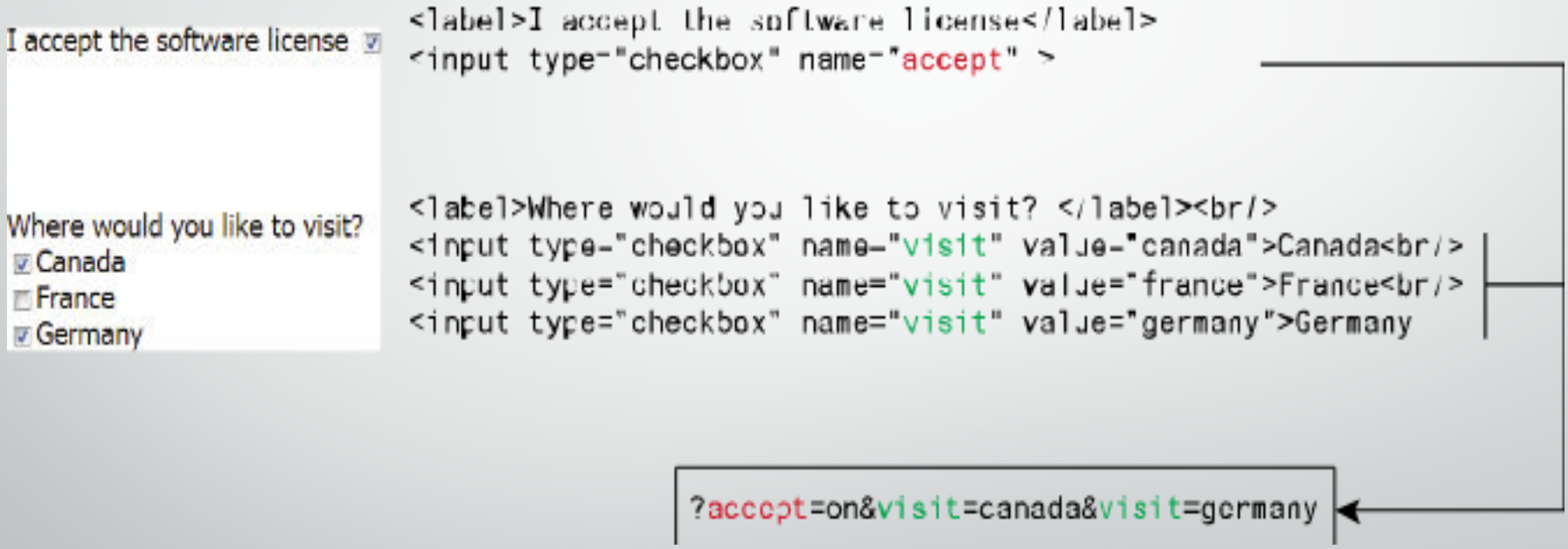
- **Button Controls (radio)**




```
<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
```

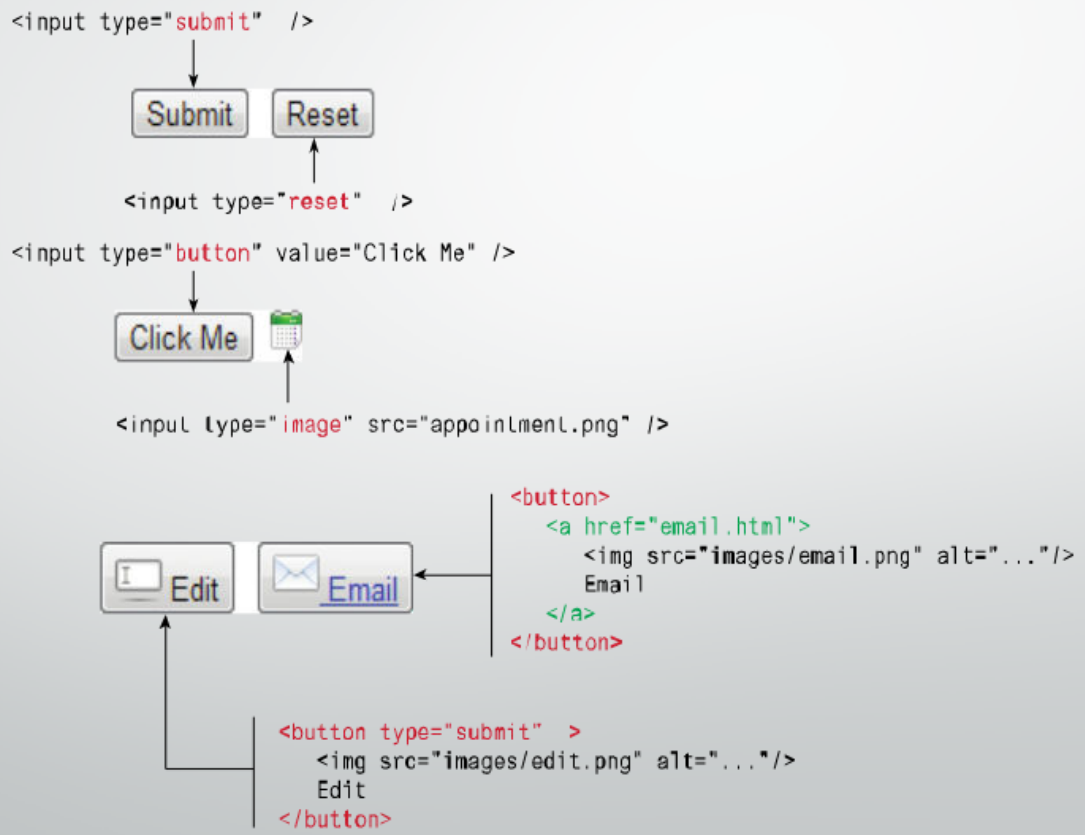
Form Control Elements

- **Button Controls (checkbox)**



Form Control Elements

- **Button Controls (checkbox)**



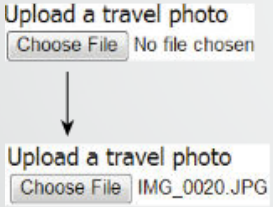
Web Systems and Technologies

TOPIC 36

Form Control Elements (2)

Form Control Elements

- **Specialized Controls (file upload)**




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

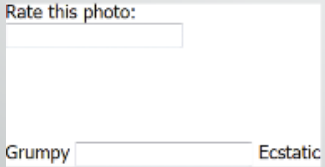
Form Control Elements

- Specialized Controls (number and range controls)




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

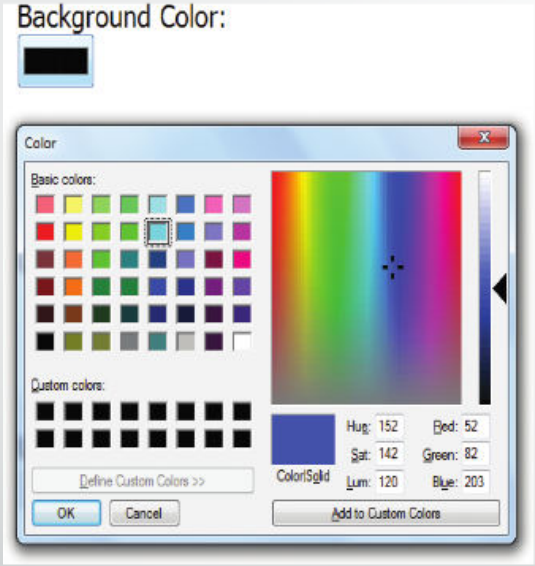
Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```



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

Form Control Elements

- **Specialized Controls (colour)**



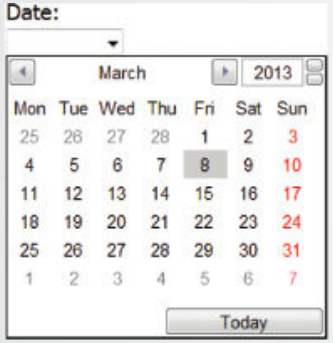

```
<label>Background Color: <br/>
<input type="color" name="back" />
```

Background Color:

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

Form Control Elements

- **Date and Time Controls**



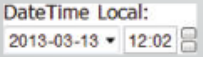

```
<label>Date: <br/>
<input type="date" ... />
```




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



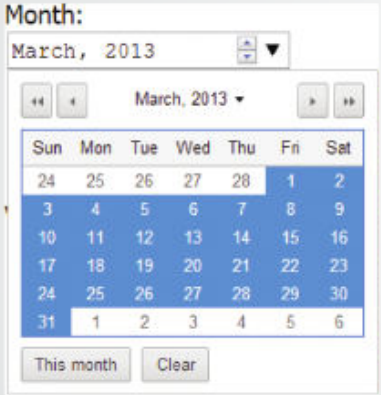

```
<input type="datetime" ... />
```



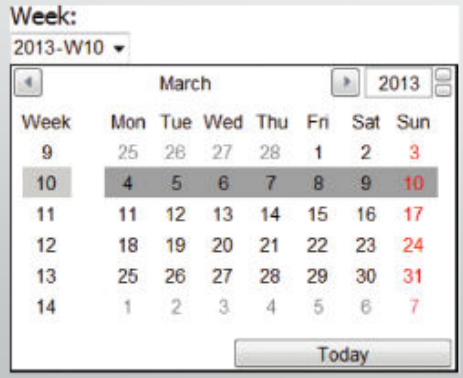

```
<input type="datetime-local" ... />
```

Form Control Elements

- **Date and Time Controls**




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




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

Form Control Elements

- **Accessible Form**

```
<label for="f-title">Title: </label>

<input type="text" name="title" id="f-title"/>


<label for="f-country">Country: </label>

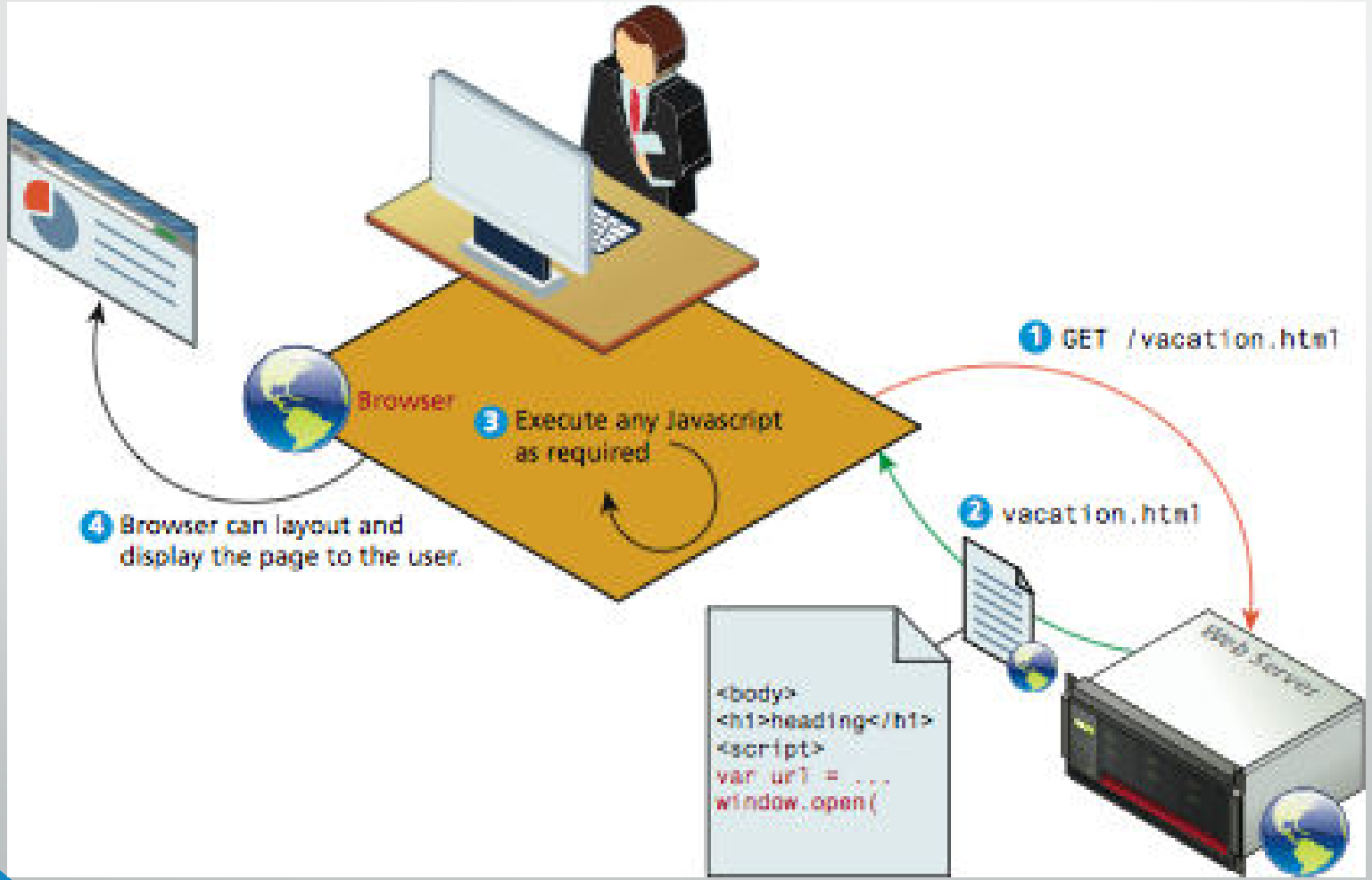
<select name="where" id="f-country">
    <option>Choose a country</option>
    <option>Canada</option>
    <option>Finland</option>
    <option>United States</option>
  </select>
```

Web System and Tecnologies CS 521

WEEK 05 TOPIC 37 TO 45

Topic 37: Introduction to JavaScript

Client Side Scripting



Introduction to JavaScript

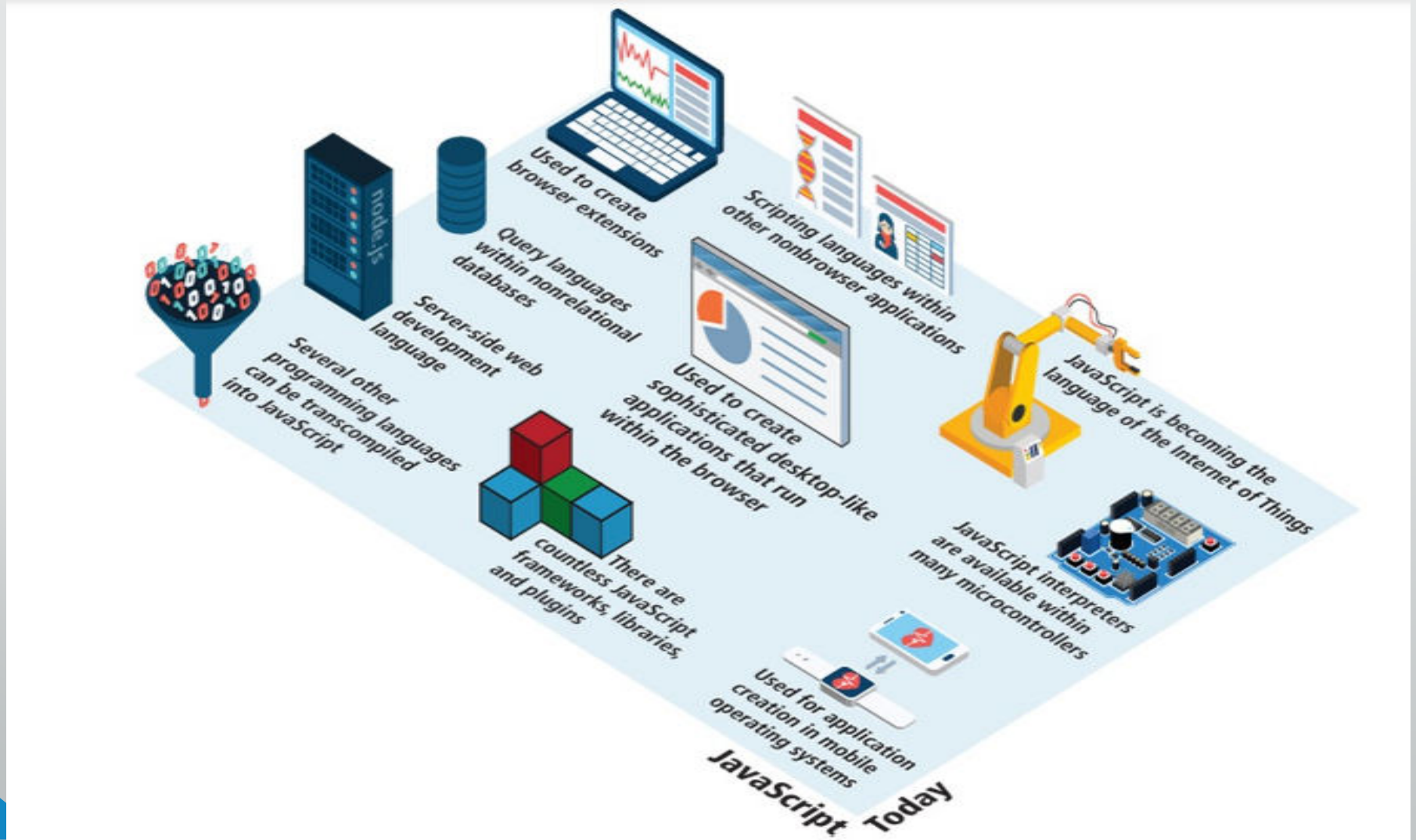
JavaScript History

- JavaScript was introduced by Netscape in their Navigator browser back in 1996
- JavaScript that is supported by your browser contains language features
  - not included in the current ECMAScript specification and
  - missing certain language features from that specification
- Commonly used version of ECMAScript is the Sixth Edition (ES6)
  - 12th Edition (ECMAScript 2021) released in 2021

JavaScript and Web 2.0

- Early JavaScript had only a few common uses
- 2000s onward saw more sophisticated uses for JavaScript
- AJAX as both an acronym and a general term

JavaScript in Contemporary Software Development



TOPIC 38

Writing and Storing the Script

Web System and Tecnologies CS 521

Writing and Storing the Script

Inline JavaScript

- Inline JavaScript refers to the practice of includingJavaScript code directly within certain HTML attributes

```
<a href="javaScript:OpenWindow();">more info</a>
```

```
<input type="button" onClick="alert('Are you sure?');" />
```

Writing and Storing the Script

Embedded JavaScript

- 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>
```

Writing and Storing the Script

External JavaScript

- External JavaScript files typically contain function definitions, data definitions, and entire frameworks.

```
<head>
    <script type="text/javascript" src="greeting.js"></script>
</head>
```

Writing and Storing the Script

Users without JavaScript

- Web crawler
- Browser plug-in
- Text-based client
- Visually disabled client
- The <NoScript> Tag

Introduction to JavaScript

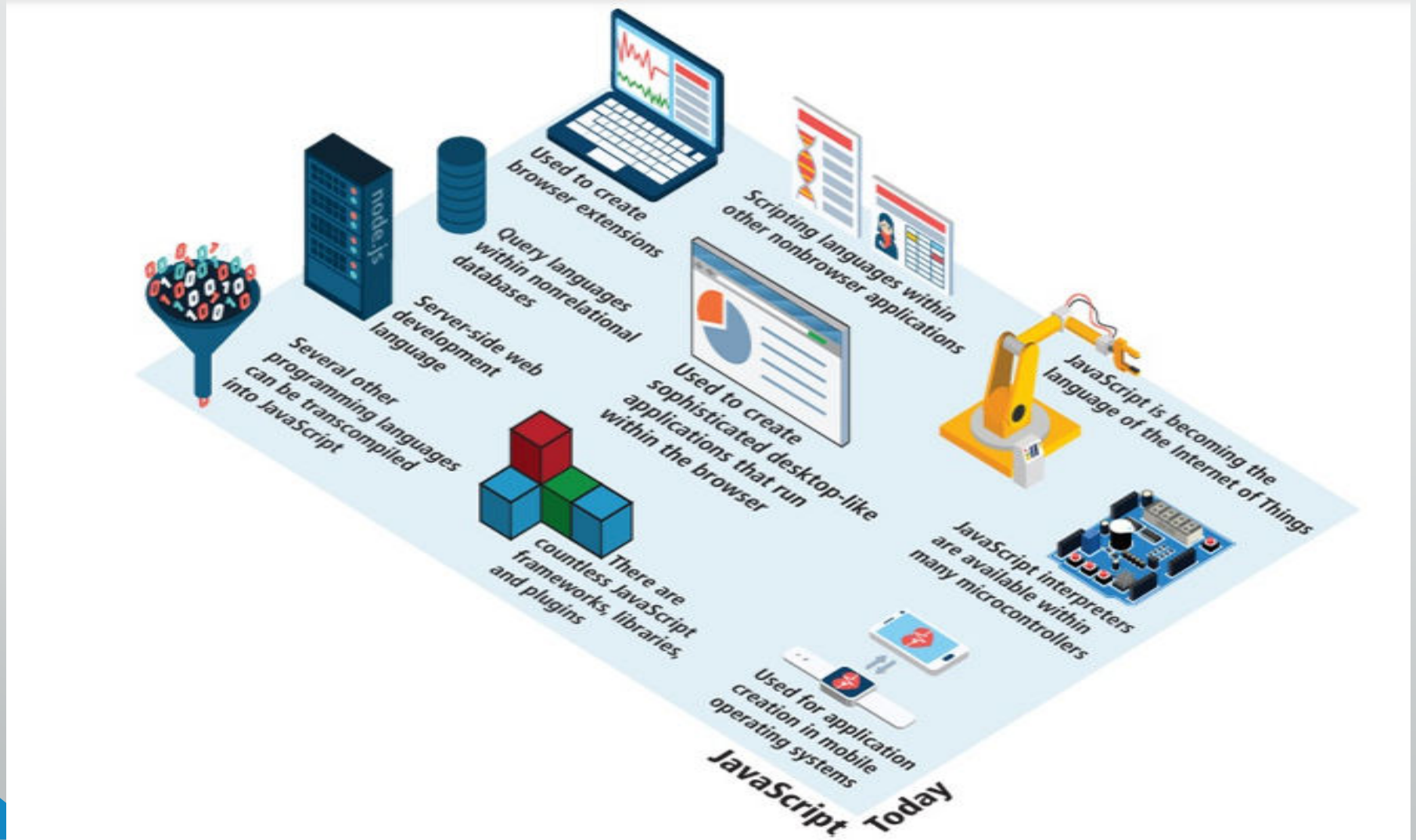
JavaScript History

- JavaScript was introduced by Netscape in their Navigator browser back in 1996
- JavaScript that is supported by your browser contains language features
  - not included in the current ECMAScript specification and
  - missing certain language features from that specification
- Commonly used version of ECMAScript is the Sixth Edition (ES6)
  - 12th Edition (ECMAScript 2021) released in 2021

JavaScript and Web 2.0

- Early JavaScript had only a few common uses
- 2000s onward saw more sophisticated uses for JavaScript
- AJAX as both an acronym and a general term

JavaScript in Contemporary Software Development



Variables and Data Types

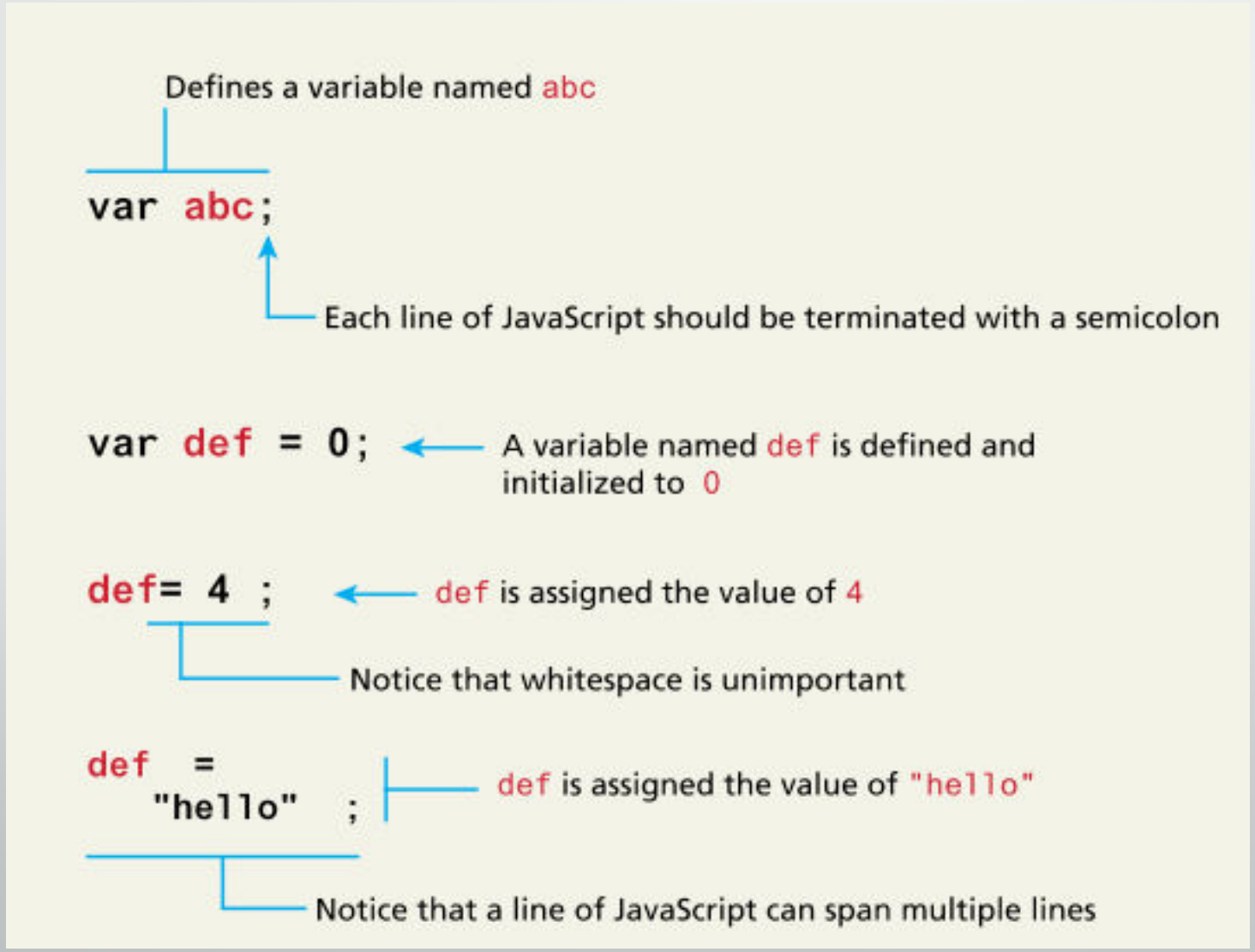
TOPIC 39

Web Systems and Technologies

Variables

- Variables in JavaScript are dynamically typed
- This simplifies variable declarations, since we do not require the familiar data-type identifiers
- Instead we simply use the var keyword

Example variable declarations and Assignments



Data Types

- Two basic data types:
  - Reference types (usually referred to as objects) and
  - primitive types (Boolean, Number, String, …)

Reference Types

```
var abc = 27;
var def = "hello";
```

variables with primitive types

```
var foo = [45, 35, 25];
```

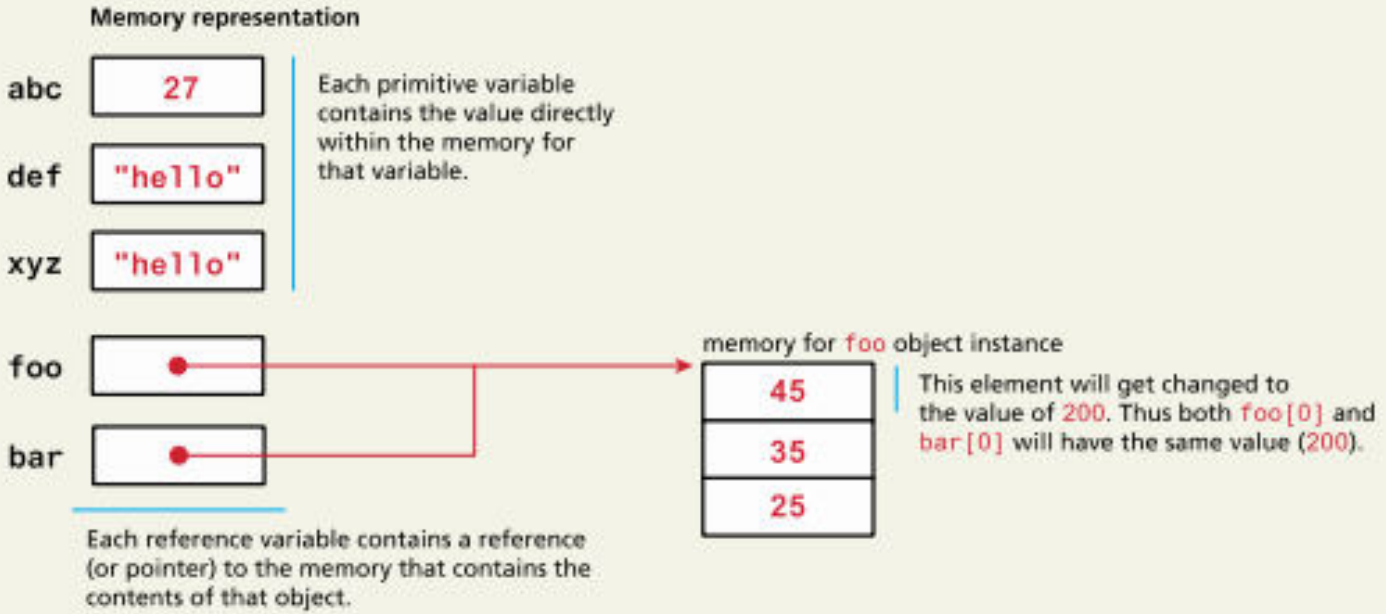
variable with reference type (i.e., array object)

```
var xyz = def;
var bar = foo;
```

these new variables differ in important ways (see below)

```
bar[0] = 200;
```

changes value of the first element of array



JavaScript Output

TOPIC 40

Web Systems and Technologies

Output

- alert()
  - Displays content within a pop-up box.
- document.write()
  - Outputs the content (as markup)
- console.log()
  - Displays content in the Browser’s JavaScript console.

Message (alert box)

- Syntax
  - alert(“Hello World”);
- Avoid frequent use

Document

- Use the JavaScript's document object

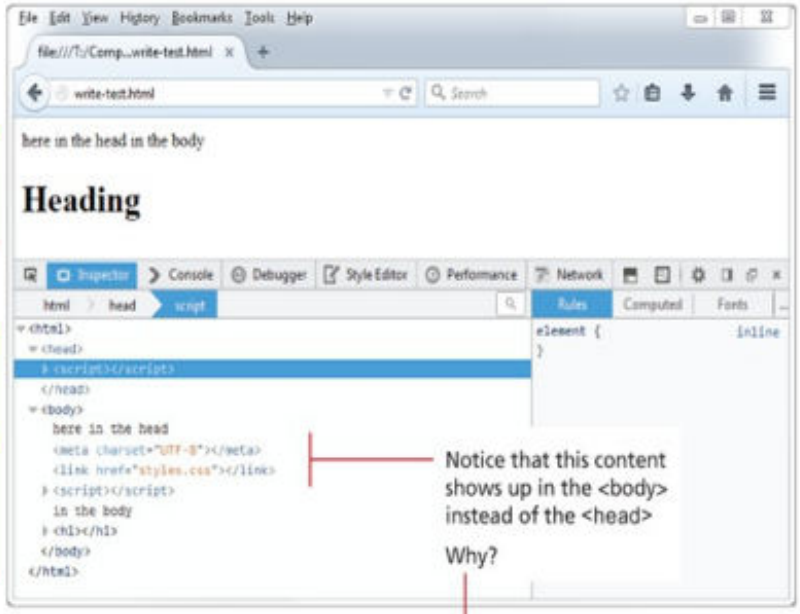
```
var name = "Randy";
document.write("<h1>Title</h1>");
// this uses the concatenate operator (+)
document.write("Hello " + name + " and welcome");
```

Fun with document.write()

```
<html>
<head>
<script>
    document.write('here in the head');

    document.write('<meta charset="UTF-8">');
    document.write('<link href=styles.css>');
</script>
</head>
<body>
<script>
    document.write("in the body");
    document.write("<h1>Heading</h1>");
</script>
</body>
</html>
```

We want this to appear here in the <head>



Generated content

Notice that this content shows up in the <body> instead of the <head>

Why?

Browser Inspector displays HTML content that is being displayed (static and dynamic)

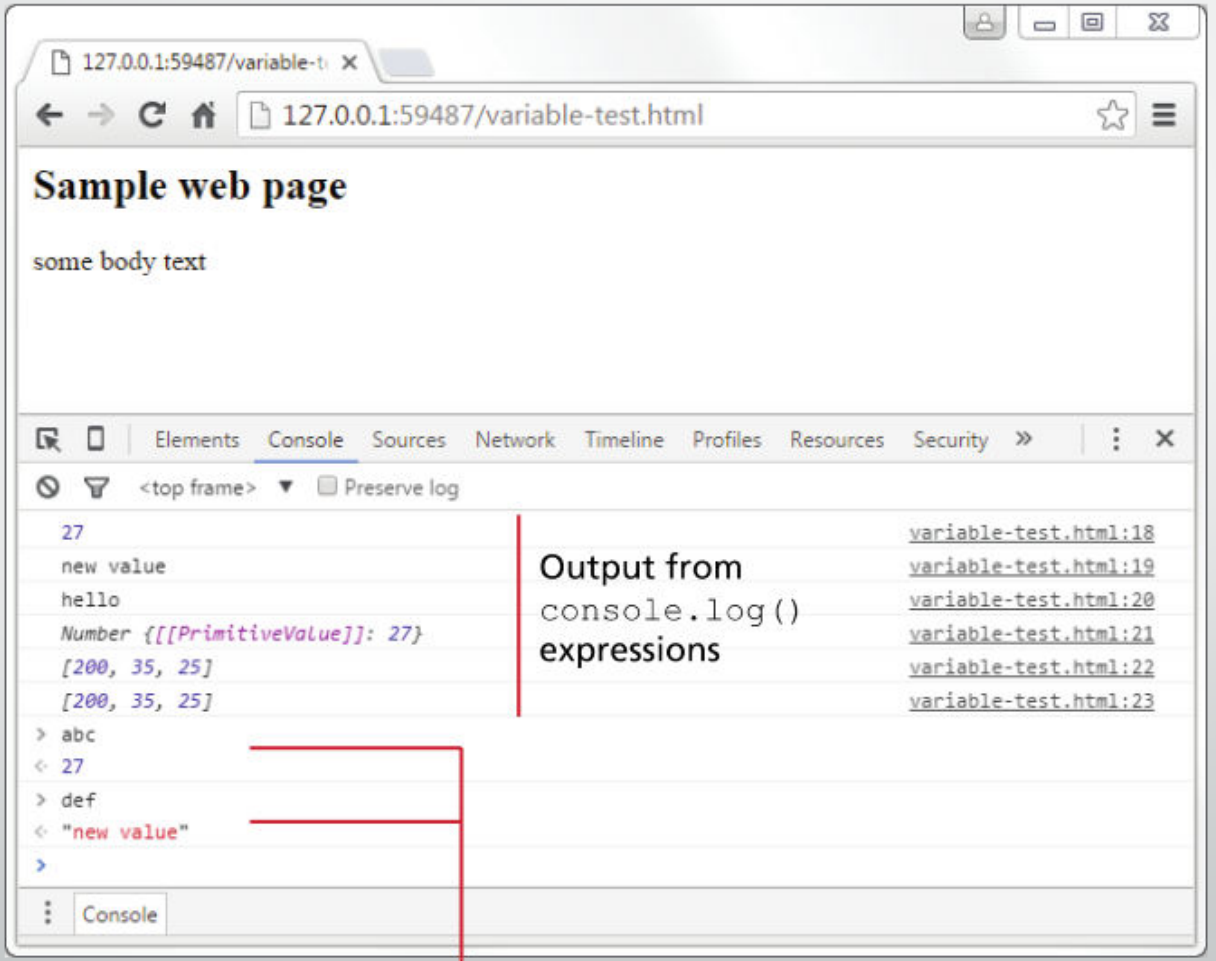
The appearance of this line will shift the following write() calls to the <body>

Console’s Log

- Allmost all modern browsers provide a JavaScript for developers and debugging
- Use console.log() to display output in the console

Chrome JavaScript Console

page content



Using console interactively to query value of JavaScript variables

Conditions

TOPIC 41

Web Systems and Technologies

Conditional Statements

- If ... else if ... else
- Switch
- Conditional Assignment

Conditional Statements

```
if (hourOfDay > 4 && hourOfDay < 12) {
        greeting = "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 18) {
        greeting = "Good Afternoon";
}
else {
        greeting = "Good Evening";
}
```

Switch Statement

```
switch (artType) {
        case "PT":
                output = "Painting";
                break;
        case "SC":
                output = "Sculpture";
                break;
        default:
        output = "Other";
}
```

Conditional Assignment

```
/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

Condition — Value if true — Value if false

```
/* equivalent to */
if (y==4) {
   x = "y is 4";
}
else {
   x = "y is not 4";
}
```

Truthy and Falsy

In JavaScript, a value is said to be **truthy** if it translates to true, while a value is said to be **falsy** if it translates to false.

- Almost all values in JavaScript are truthy
- false, null, "", '', 0, NaN, and undefined are falsy

TOPIC 42: Arrays

Arrays are one of the most commonly used data structures in programming.

JavaScript provides two main ways to define an array.

- object literal notation
- use the Array() constructor

Arrays (Object Literal Notation)

The literal notation approach is generally preferred since it involves less typing, is more readable, and executes a little bit quicker

```
var years = [1855, 1648, 1420];

var countries = ["Canada", "France",
                 "Germany", "Nigeria",
                 "Thailand", "United States"];

var mess = [53, "Canada", true, 1420];
```

Array Constructor

- We can use the Array constructor to create an array

```
var countries = new Array("Canada",          "France"
,"Germany",          "Nigeria","Thailand", "United
            States");

var A = new Array(5);
```

Arrays

- Some common features

arrays in JavaScript are zero indexed

[] notation for access

.length gives the length of the array

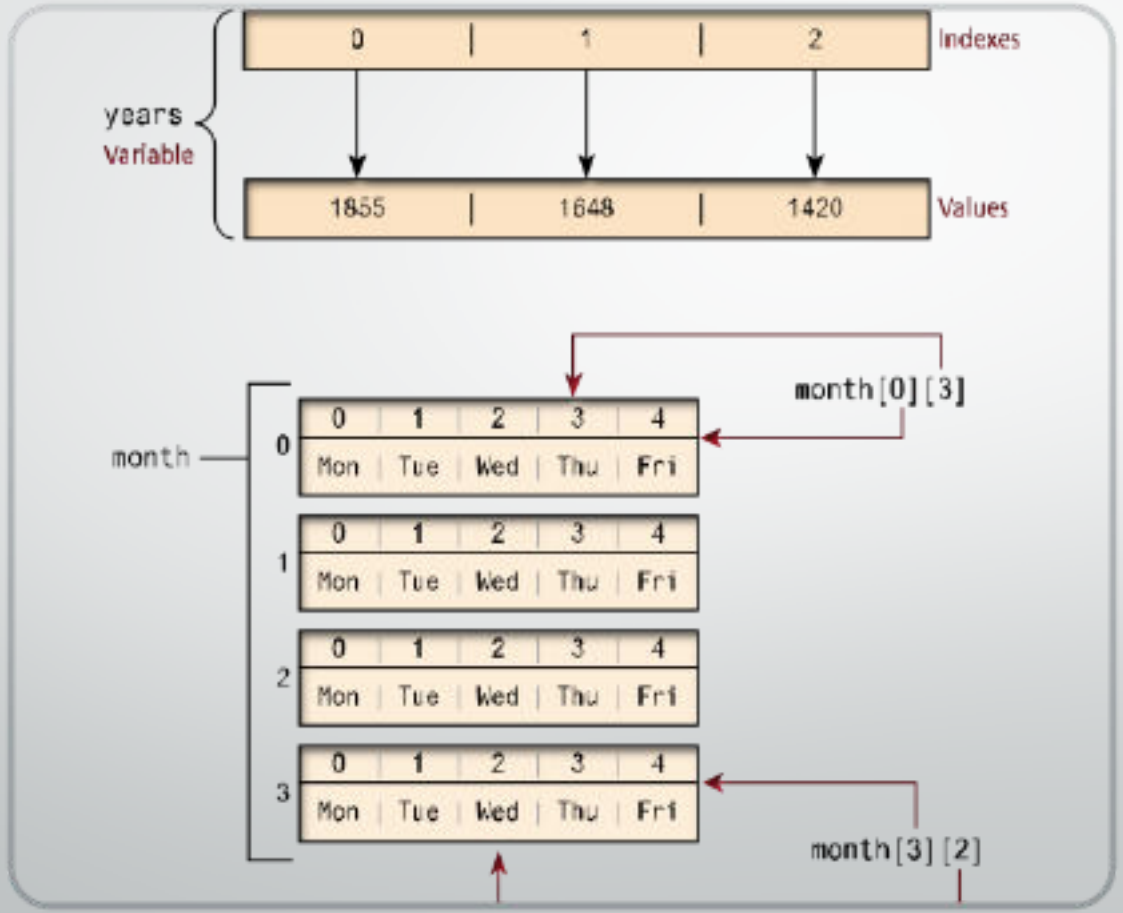
.push()

.pop()

concat(), slice(), join(), reverse(), shift(), and sort()

Arrays

- Arrays Illustrated



TOPIC 43: Loops

- Types of Loops

Used for repetitively executing statements

while

do while

for

forEach

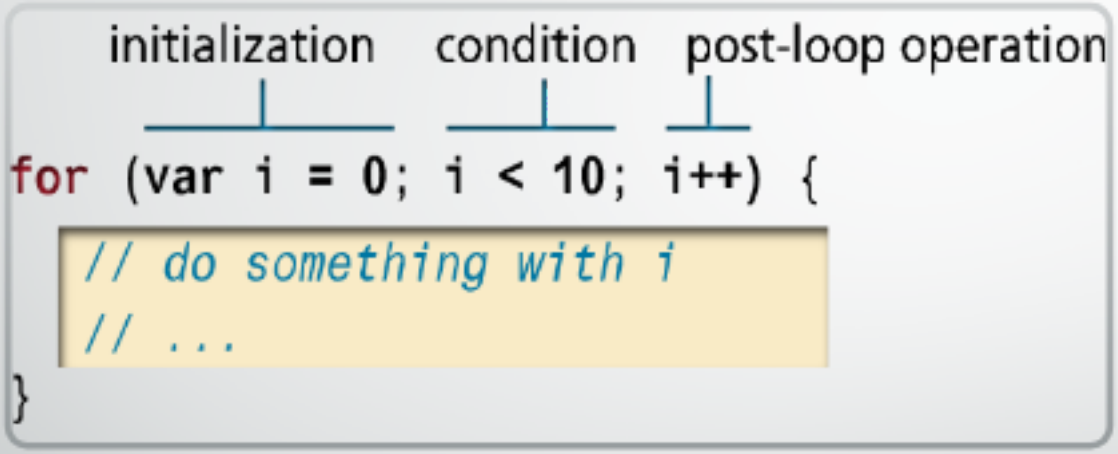
Loops

- While and do . . . while Loops

```
var count = 0;
while (count < 10) {
    // do something
    // ...
    count++;
    }
count = 0;
do {
// do something
    // ...
    count++;
} while (count < 10);
```

Loops

- For Loop



Loops

- ForEach Loop

- forEach is used with an array
- Processes each element of an array

```
var countries = ["Canada", "France",
                 "Germany", "Nigeria",
                 "Thailand", "United States"];
countries.forEach( function(country) {
                 console.log(country);
                 }
)
```

TOPIC 44: Objects

Object Creation - Object Literal Notation

```
var objName = {
    name1: value1,
    name2: value2,
        // ...
    nameN: valueN
    };
```

Objects

Object Creation - Object Literal Notation

- Access using either of:

objName.name1

objName["name1"]

Objects

Object Creation – Constructed Form

```
// first create an empty object
var objName = new Object();
// then define properties for this object
objName.name1 = value1;
objName.name2 = value2;
```

Web Systems and Technologies

TOPIC 44

Functions (1)

Functions

Function Declarations vs. Function Expressions

- **Functions** are the building block for modular code in JavaScript

```
function subtotal(price,quantity) {
        return price * quantity;
}
```

- The above is formally called a **function declaration**, called or invoked by using the () operator

```
var result = subtotal(10,2);
```

Functions

Function Declarations vs. Function Expressions

```
// defines a function using a function expression
var sub = function subtotal(price,quantity) {
return price * quantity;
};
// invokes the function
var result = sub(10,2);
```

It is conventional to leave out the function name in function expressions

Functions

- Anonymous Function Expressions

```
// defines a function using an anonymous function expression

var calculateSubtotal = function (price,quantity) {

return price * quantity;

};

// invokes the function

var result = calculateSubtotal(10,2);
```

Functions

Arrow Functions

- Have a shorter syntax
- Can convert normal functions to arrow
- Useful for anonymous and call functions

```
//normal function
function square(n) {
  return n*n;
}

//arrow function
sq = (n) => n*n

//Another example
sum = (n,m) => n+m
```

Web Systems and Technologies

WEEK 6

TOPIC 46 TO 54

Web Systems and Technologies

- Functions-Nested
- Functions-Callback
- Functions-Objects
- Functions-Scope
- Exception handling using Try and Catch
- JavaScript built in Objects (Data, Math, Navigator, String, and etc.)
- Object prototypes: Prototypes for Extending Objects
- Introduction to the Document Object Model (DOM)
- Accessing DOM Clements

Nested Functions (2)

TOPIC 46

Nested Functions

```
function calculateTotal(price,quantity) {
        var subtotal = price * quantity;
        return subtotal + calculateTax(subtotal);
        // this function is nested
        function calculateTax(subtotal) {
                var taxRate = 0.05;
                var tax = subtotal * taxRate;
                return tax;
        }
}
```

Hoisting in JavaScript

Function declaration is hoisted to the beginning of its scope

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

Variable declaration is hoisted to the beginning of its scope

BUT

Variable assignment is not hoisted

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    var calculateTax = function (subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    };
}
```

THUS

The value of the `calculateTax` variable here is undefined

Callback Functions

TOPIC 47

Callback Functions

```
var calculateTotal = function (price, quantity, tax) {
    var subtotal = price * quantity;
    return subtotal + tax(subtotal);
};

var calcTax = function (subtotal) {
    var taxRate = 0.05;
    var tax = subtotal * taxRate;
    return tax;
};

var temp = calculateTotal(50,2,calcTax);
```

2 The local parameter variable tax is a reference to the calcTax() function

1 Passing the calcTax() function object as a parameter

We can say that calcTax variable here is a callback function



Callback Functions

Passing an anonymous function definition as a callback function parameter

```
var temp = calculateTotal( 50, 2,
                           function (subtotal) {
                              var taxRate = 0.05;
                              var tax = subtotal * taxRate;
                              return tax;
                           }
                         );
```



Functions - Objects

TOPIC 48

Objects and Functions Together

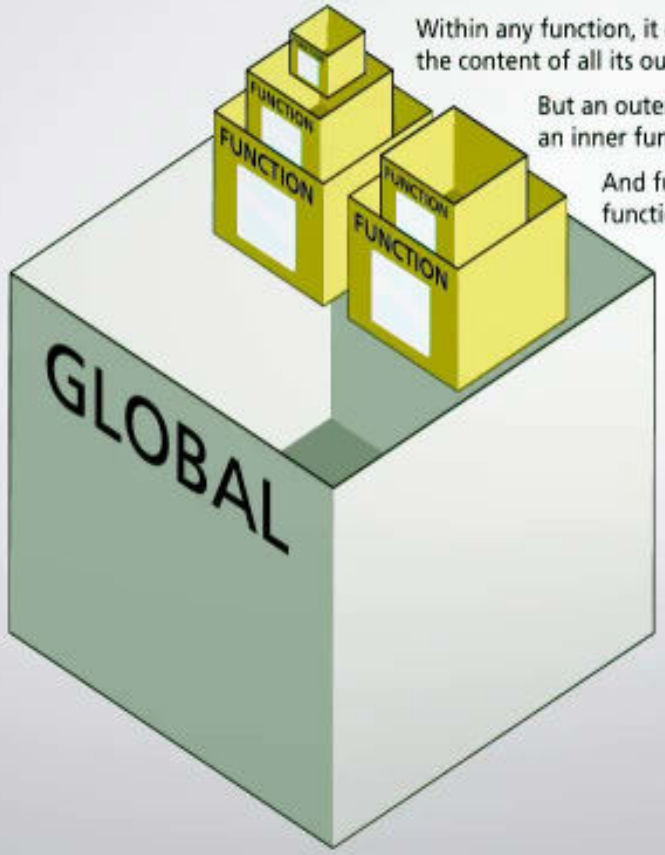
```
var order = {
    salesDate : "May 5, 2017",
    product : {
        type: "laptop",
        price: 500.00,
        output: function () {
            return this.type + ' $' + this.price;
        }
    },
    customer : {
        name: "Sue Smith",
        address: "123 Somewhere St",
        output: function () {
            return this.name + ', ' + this.address;
        }
    },
    output: function () {
            return 'Date' + this.salesDate;
    }
};
```

Functions (5) Scope

TOPIC 49

Scope in JavaScript

Each function is like a box
with a one-way window



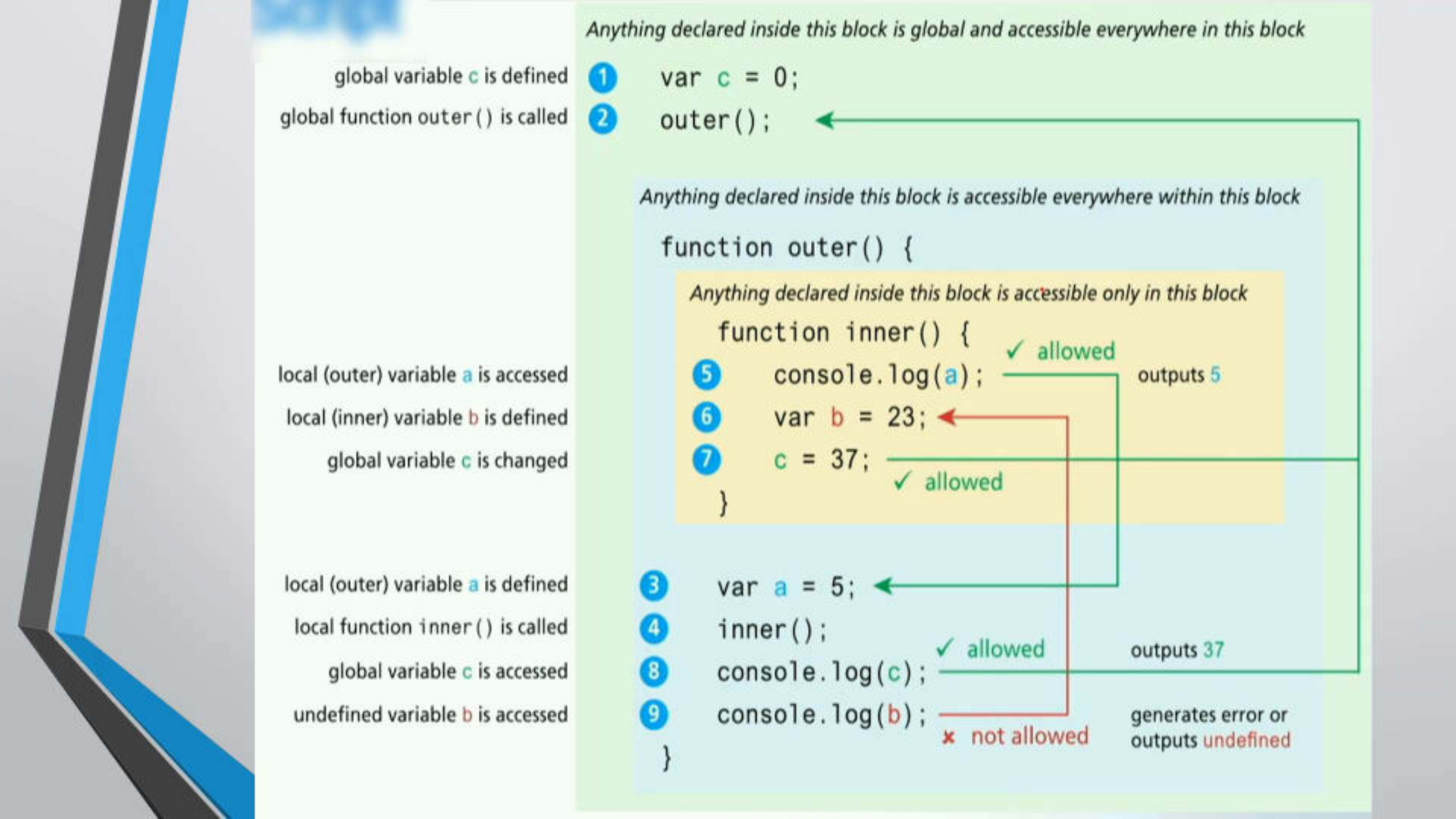
Within any function, it can see out at the content of all its outer boxes

But an outer function can't look into an inner function

And functions can't see into other functions at the same level

All functions can see anything within global scope

Scope ends at global ... functions can't see outside of the global box


Anything declared inside this block is global and accessible everywhere in this block
global variable c is defined
1 var c = 0;
global function outer() is called
2 outer();
Anything declared inside this block is accessible everywhere within this block
function outer() {
Anything declared inside this block is accessible only in this block
function inner() {
local (outer) variable a is accessed
5 console.log(a);
✓ allowed
outputs 5
local (inner) variable b is defined
6 var b = 23;
global variable c is changed
7 c = 37;
✓ allowed
}
local (outer) variable a is defined
3 var a = 5;
local function inner() is called
4 inner();
global variable c is accessed
8 console.log(c);
✓ allowed
outputs 37
undefined variable b is accessed
9 console.log(b);
✘ not allowed
generates error or
outputs undefined
}

Remember that scope is determined at design-time

```
var myGlobal = 55;

function outer() {

    var foo = 66;

    function middle() {

        var bar = 77;

        function inner() {

            var foo = 88;

            bar = foo + myGlobal;
        }
    }
}
```

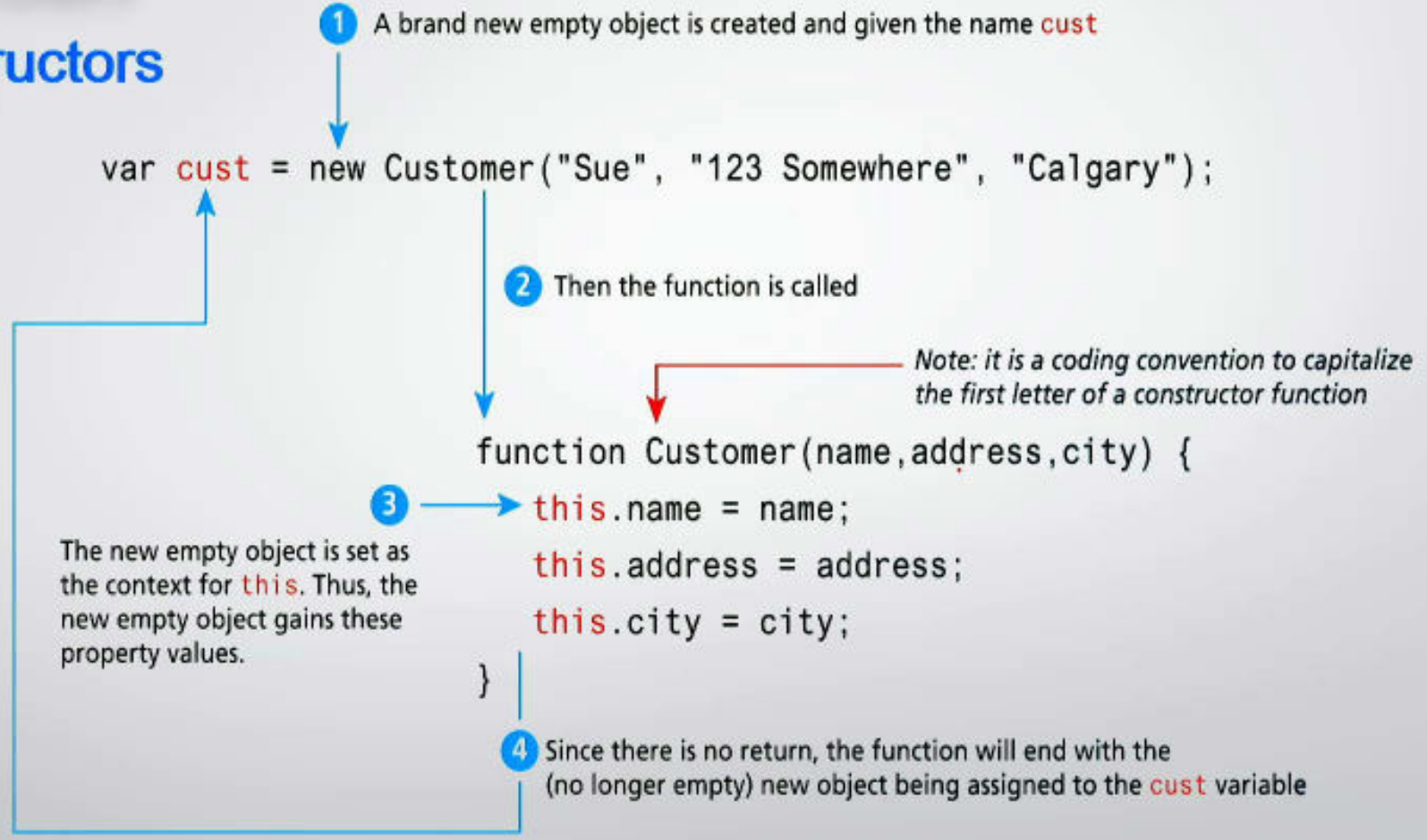
1 looks first within current function

2 then looks within first containing function

3 then looks within next containing function

4 then finally looks within global scope

Function Constructors



Exception Handling

TOPIC 50

Introduction

- Errors occur at runtime are called exceptions
- Might result in unexpected behavior if unhandled
- Handling exceptions avoid unexpected termination

Examples

- TypeError

```
var x
console.log(x.a)
```

- RangeError

```
var A = new Array(-5)
```

- ReferenceError

```
function f() {
  var y = 2;
}
console.log(y)
```

Handling Exceptions

- Exceptions are handled using try/catch
- Statements expected to result in an exception are put in try block
- Exception handling is done in catch block

```
try {
    var A = new Array(-5);
} catch (e) {
    console.log(e);
}
```

Throwing Exceptions

- Exceptions can be generated (thrown) by the programmer using **throw**

```
try {
    …
    if (age < 0 || age > 150) {
      throw("InvalidAge: Valid age
is between 0 and 150");
    }
} catch (e) {
    console.log(e);
}
```

Catching Specific Exceptions

- Use instanceof to catch specific exceptions

```
try {
    var A = new Array(-5);

} catch (e) {
    if(e instanceof RangeError) {
        ...
    }
}
```

The *finally* Block

- The finally block is executed irrespective of exception being thrown or not
- Useful for releasing resources

```
try {
    //connect to a database
} catch (e) {
    //an error occurred
    //rollback
} finally {
    //disconnect from the database
}
```

Web Systems and Technologies

Built-in Objects

TOPIC 51

Built-in Objects

Working with Maths

- Math object provides many functions
- Called through Math object, e.g., Math.abs(-5) returns 5
  - Trigonometry functions: sin(), cos(), …
  - Rounding: floor(), ceil(), round(), …
  - Statistical: min(), max(), …
  - Math.random() returns a value between [0,1).

```
function rand_int_between(min,max) {
  return Math.floor(Math.random()*
         (max-min+1)+min);
}
```

Built-in Objects

Working with Dates

- Date object provides functions to work with dates and times
  - Date.now(): milliseconds passed since Jan 1, 1970
  - Date.parse(): converts a string to a date, e.g., Date.parse('Sep 1, 2021')
  - toDateString(): converts a date object to a string in human readable form

```
var today = new Date();
console.log(today.toDateString())
```

Built-in Objects

Working with Patterns

- RegExp object is used for matching text with patterns
  - test(): tests if a string follows a pattern

```
//Example
var pattern = /\d+/
var pattern = new RegExp('\\d+')
var num = 123;
var str = 'abc';
console.log(pattern.test(num))
console.log(pattern.test(str))
```

Web Systems and Technologies

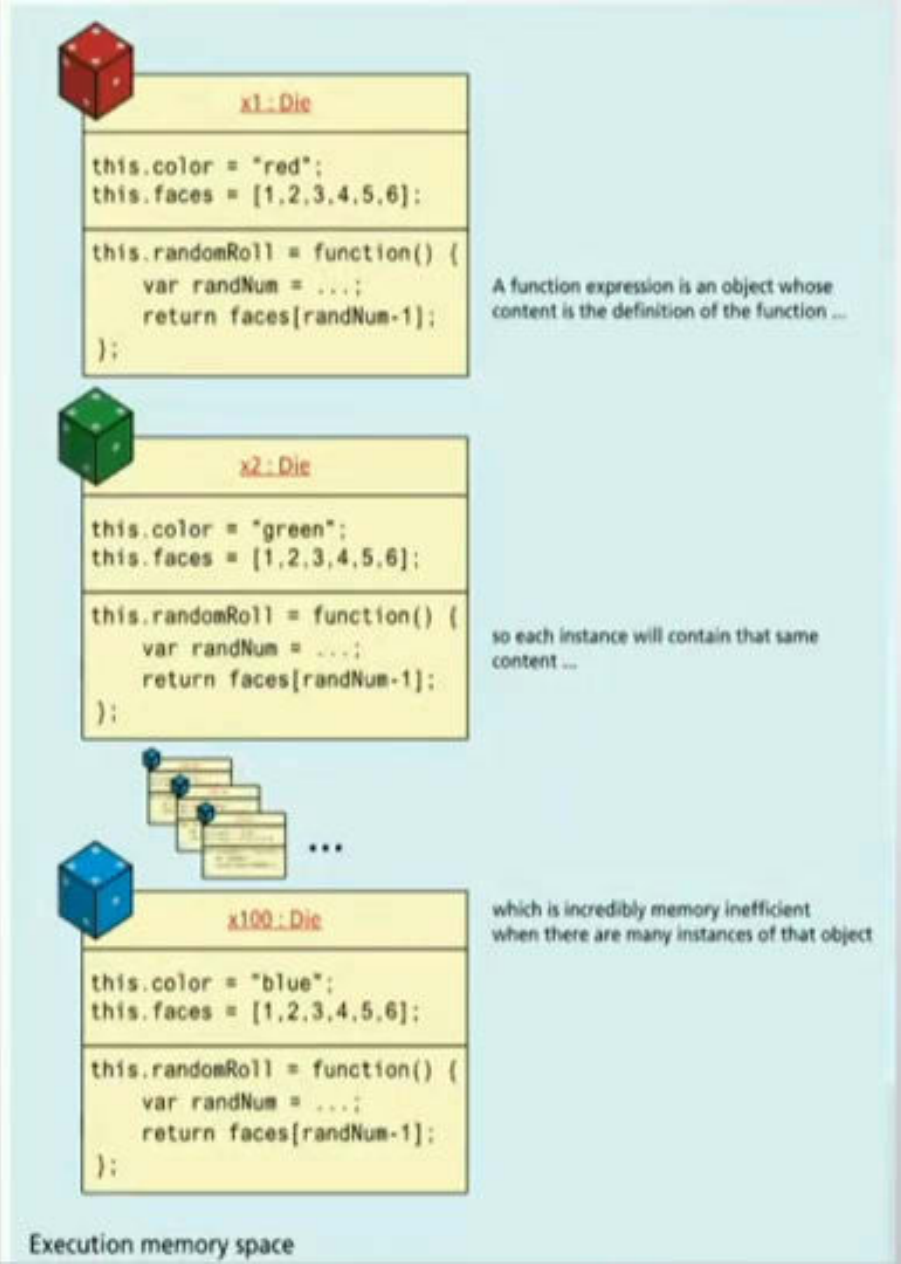
Object Prototypes

TOPIC 52

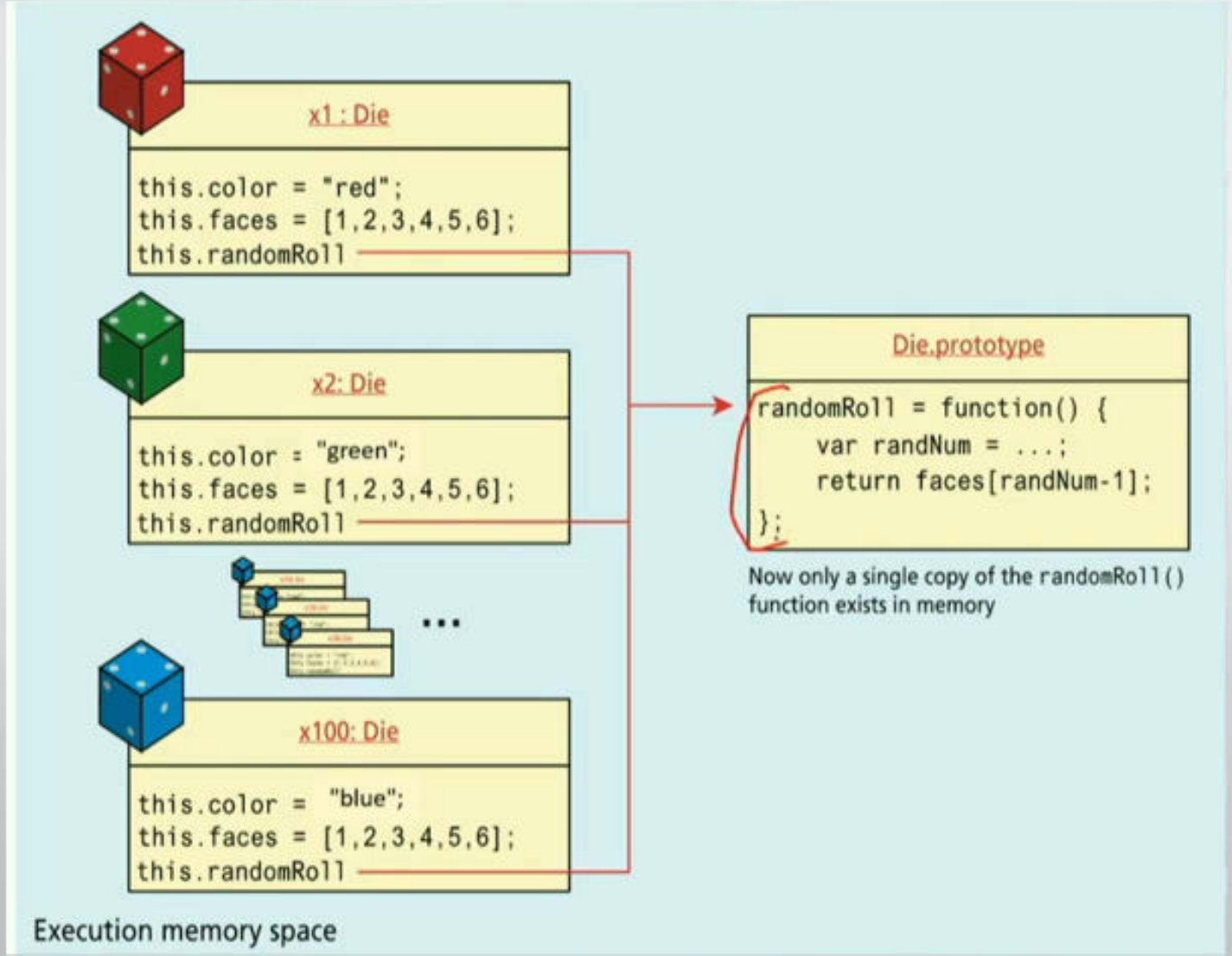
Introduction

- Prototypes are an essential syntax mechanism in JavaScript, and are used to make JavaScript behave more like an object-oriented language

Methds Get Duplicated



Using Prototype reduces Duplication at Runtime



Using Prototypes to Extend Other Objects

```
String.prototype.countChars = function (c) {
        var count=0;
        for (var i=0;i<this.length;i++) {
                if (this.charAt(i) == c)
                        count++;
                }
        return count;
}
var msg = "Hello World";
console.log(msg + "has" + msg.countChars("l") + "
letter l's");
```

The Document Object Model (DOM)

TOPIC 53

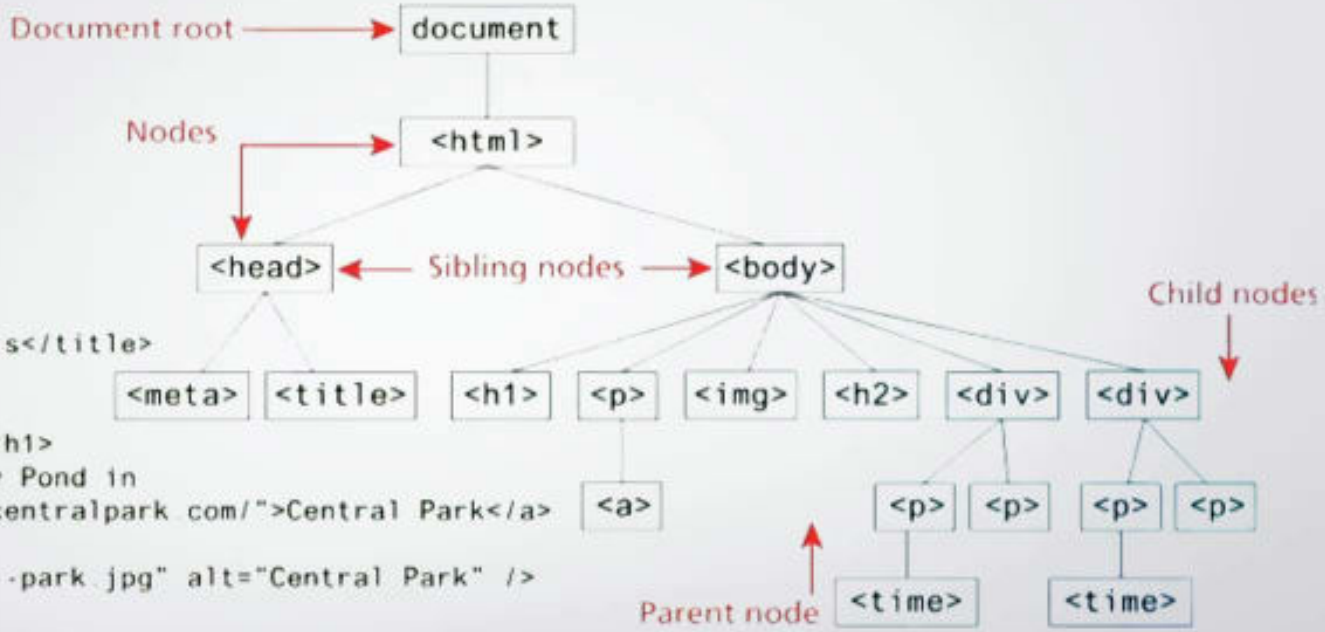
Web Systems and Technologies

Introduction

- Any HTML document can be represented in form of a tree called Document Object Model
- Available as a data structure in JavaScript
- Helps in accessing and manipulating HTML elements through JavaScript

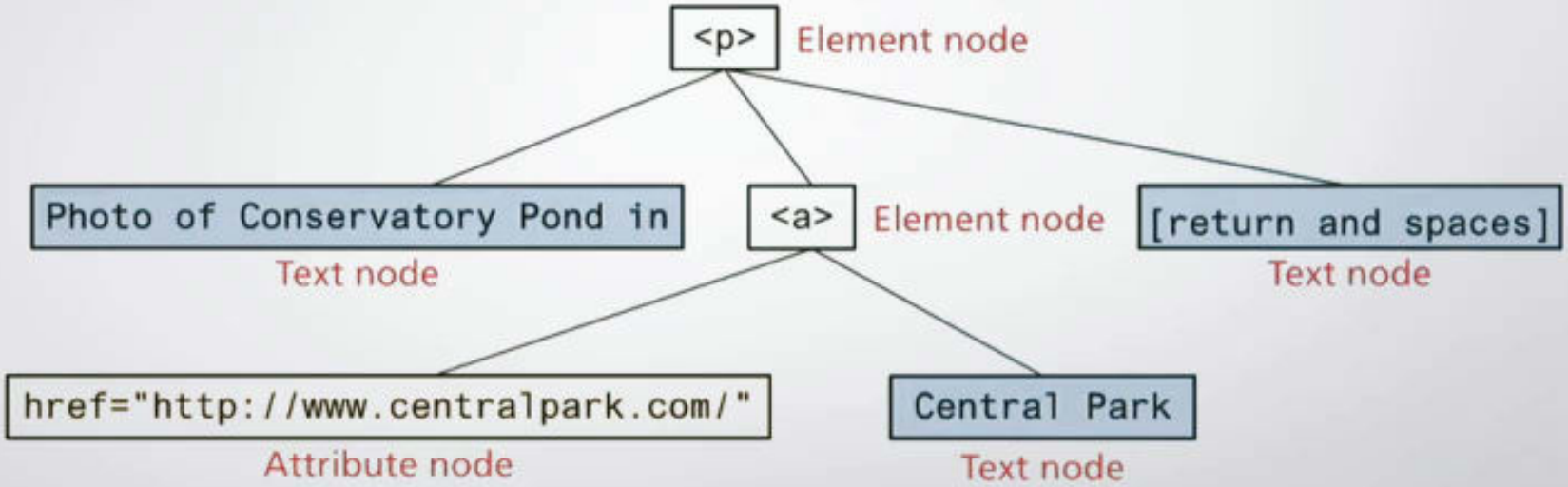
Overview

```
<html>
<head>
   <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>2016-05-23</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <div>
      <p>By Susan on <time>2016-11-18</time></p>
      <p>I love Central Park.</p>
   </div>
</body>
</html>
```



Nodes

```
<p>Photo of Conservatory Pond in
   <a href="http://www.centralpark.com/">Central Park</a>
</p>
```



DOM Related Data Types

- Document
  - The root document
- Node
  - Every object in the HTML document is a node (elements, attributes, text, etc.)
- NodeList
  - Array of elements
- Element
  - Refers to an HTML element or a node
- Attribute
  - Refers to an attribute

Accessing DOM Elements

TOPIC 54

Web Systems and Technologies

Document Object

- The DOM **document** object is the root JavaScript object representing the entire HTML document

```
// retrieve the URL of the current page
var a = document.URL;
// retrieve the page encoding, for
example ISO-8859-1
var b = document.inputEncoding;
```

Selection Methods

- Classic
  - getElementById()
  - getElementsByTagName()
  - getElementsByClassName()
- Newer
  - querySelector() and
  - querySelectorAll()

Selection Methods

```
var node = document.getElementById("latest");
```

```
<body>
   <h1>Reviews</h1>
   <div id="latest">
      <p>By Ricardo on <time>2016-05-23</time></p>
      <p class="comment">Easy on the HDR buddy.</p>
   </div>
   <hr/>
   <div>
      <p>By Susan on <time>2016-11-18</time></p>
      <p class="comment">I love Central Park.</p>
   </div>
   <hr/>
</body>
```

```
var list2 = document.getElementByClassName("comment");
```

```
var list1 = document.getElementsByTagName("div");
```



Query Selector

querySelectorAll("nav ul a:link")

querySelectorAll("#main div time")

querySelector("#main>time")

querySelector("footer")

```
<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>

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

Element Node Object

- Element Node object represents an HTML element in the hierarchy, contained between the opening <> and closing </> tags for this element. Every node has
  - classList
  - className
  - id
  - innerHTML
  - style
  - tagName

Element Node Object (2)

- Some properties are not common in all the elements
  - href
  - name
  - src
  - value

Web Systems and Technologies

WEEK 7

TOPIC 55 TO 63

Web Systems and Technologies

- Modifying DOM Elements
- Creating DOM elements
- Event Handling
- Capturing and Bubbling
- Event Types
- Form Validation; Client Side Input Validation
- Introduction to jQuery
- Query Selectors
- jQuery Manipulators

TOPIC 55

Modifying the DOM

Web Systems and Technologies

Changing an Element's Style

```
<style>
    .box {
        margin: 2em; padding: 0;
        border: solid 1pt black;
    }
   .yellowish { background-color: #EFE63F; }
   .hide { display: none; }
</style>
<main>
   <div class="box">
    ...
   </div>
</main>
```

```
var node = document.querySelector("main div");
```

Equivalent to:

1. `node.className = "yellowish";` → This replaces the existing class specification with this one. Thus the `<div>` no longer has the box class

   `<div class="yellowish">`

2. `node.classList.remove("yellowish");`
   `node.classList.add("box");` → Removes the specified class specification and adds the box class

   `<div class="">`
   `<div class="box">`

3. `node.classList.add("yellowish");` → Adds a new class to the existing class specification

   `<div class="box yellowish">`

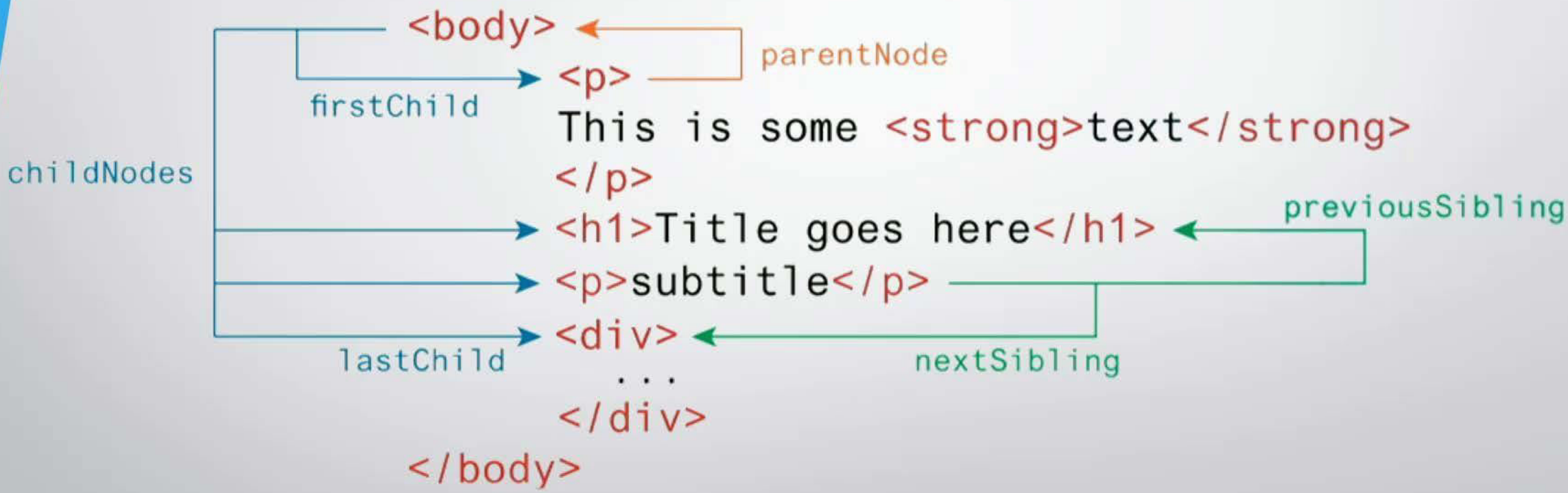
4. `node.classList.toggle("hide");` → If it isn't in the class specification, then add it

   `<div class="box yellowish hide">`

5. `node.classList.toggle("hide");` → If it is in the class specification, then remove it

   `<div class="box yellowish">`

Dom Family


<body>
parentNode

firstChild
This is some text

childNodes
previousSibling
<h1>Title goes here</h1>
subtitle

lastChild
nextSibling
...

</body>

Changing an Element's Content

```
node.innerHTML = "foo<em>bar</em>";
```

TOPIC 56

Creating DOM Elements

Web Systems and Technologies

Creating DOM elements

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
</div>
```

Visualizing the DOM elements

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
</div>
```

1. Create a new text node

```
var text = document.createTextNode("this is dynamic");
```

`"this is dynamic"`

2. Create a new empty <p> element

```
var p = document.createElement("p");
```

`<p></p>`

3. Add the text node to new <p> element

```
p.appendChild(text);
```

`<p> "this is dynamic" </p>`

4. Add the <p> element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

Creating DOM elements

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
</div>
```

Visualizing the DOM elements

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
</div>
```

1. Create a new text node

```
var text = document.createTextNode("this is dynamic");
```

`"this is dynamic"`

2. Create a new empty <p> element

```
var p = document.createElement("p");
```

`<p></p>`

3. Add the text node to new <p> element

```
p.appendChild(text);
```

`<p> "this is dynamic" </p>`

4. Add the <p> element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
    <p>this is dynamic</p>
</div>
```

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
    <p> "this is dynamic" </p>
</div>
```

TOPIC 57

Events

Introduction

- JavaScript event is an action that can be detected by JavaScript
- Many of them are initiated by user actions, some are generated by the browser itself
- We say that an event is *triggered* and then it is *handled* by JavaScript functions

Event Handling Approaches- Inline Hook

HTML document using the inline hooks

```
...
 <script type="text/javascript" src="inline.js"></script>
...
<form name='mainForm' onsubmit="validate(this);">
      <input name="name" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input name="email" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input type="submit"
          onclick="function (e) {
                    ...
                  }">
```

inline.js

```
function validate(node) {
  ...
}
function check(node) {
  ...
}
function highlight(node) {
  ...
}
```

Notice that you can define an entire event handling function within the markup. This is NOT recommended!

Event-Handling Approaches – Event Property Approach

- var myButton = document.getElementById('example');
- myButton.onclick = alert('some message');

Event-Handling Approaches – Event Listener Approach

- var myButton = document.getElementById('example');
- myButton.addEventListener('click', alert('some message'));
- myButton.addEventListener('mouseout', funcName);

Event-Handling Approaches – Event Listener Approach (anon function)

```
myButton.addEventListener('click', function() {
        var d = new Date();
        alert("You clicked this on "+
d.toString());
});
```

Event Object

- When an event is triggered, the browser will construct an event object that contains information about the event.

```
div.addEventListener('click', function(e)
{
        // find out where the user clicked
        var x = e.clientX;
        …
```

TOPIC 58

Capturing and Bubbling

Web Systems and Technologies

Event Handling

- Events are handled in three phases
- Capturing
  - Event propagation from parent towards the most nested child
- Target
  - Identifying the target
- Bubbling
  - Elements registered for the events are triggered from the most nested child towards the outermost

TOPIC 59

Event Types – Mouse, Keyboard, Touch

EVENT TYPES

- Mouse events
- Keyboard events
- Touch events
- Form events
- Textfield events

Mouse Events

- **click** The mouse was clicked on an element
- **dblclick** The mouse was double clicked on an element
- **mousedown** The mouse was pressed down over an element
- **mouseup** The mouse was released over an element
- **mouseover** The mouse was moved (not clicked) over an element
- **mouseout** The mouse was moved off of an element
- **mousemove** The mouse was moved while over an element

Keyboard Events

- **keydown** The user is pressing a key (this happens first)
- **keypress** The user presses a key (this happens after keydown)
- **keyup** The user releases a key that was down (this happens last)

Touch Events

- Touch events are a relatively new category of events that can be triggered by devices with touch screens
- The different events (e.g., touchstart, touchmove, and touchend) are analogous to some of the mouse events (mousedown, mousemove, and mouseup).

TOPIC 60

Form Validation

Web Systems and Technologies

Form Validation

- HTML Level
- JavaScript Level
- Webserver Level

Form Events

- Form events
  - submit
  - reset
  - formdata
- Textfield events
  - blur
  - change
  - focus
  - select
  - …

TOPIC 61

Introduction to jQuery

Web Systems and Technologies

The Most Popular Framework

- Makes working with JavaScript and DOM easy
- Cross browser compatibility
- Most popular JavaScript library (as of 2021). According to w3techs.com
  - Used by 78% of all website
  - 96% market share

Including jQuery

- Use a Content Delivery Network (CDN)
  - <script src="https://code.jquery.com/jquery-3.6.0.min.js"> </script>
- Can also be
  - Downloaded and used on your web server
  - Installed via package manager (npm, yarn, etc.)

TOPIC 62

jQuery Selectors

Web Systems and Technologies

jQuery Selector

- The power of jQuery resides in the function named jQuery()
- There's also an alias for this function named $()
- You can combine CSS selectors with the $() notation to select DOM objects that match CSS attributes

jQuery Selector

- Selecting using regular JavaScript

```
var node =
document.getElementById("here");
var link =
document.querySelectorAll("ul li");
```

- Equivalent selection using jQuery

```
var node = $("#here");
var link = $("ul li");
```

- The $() function always returns a set of results

Basic Selectors

- $("*")—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 element with a given HTML id attribute

Examples

```
<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>
```

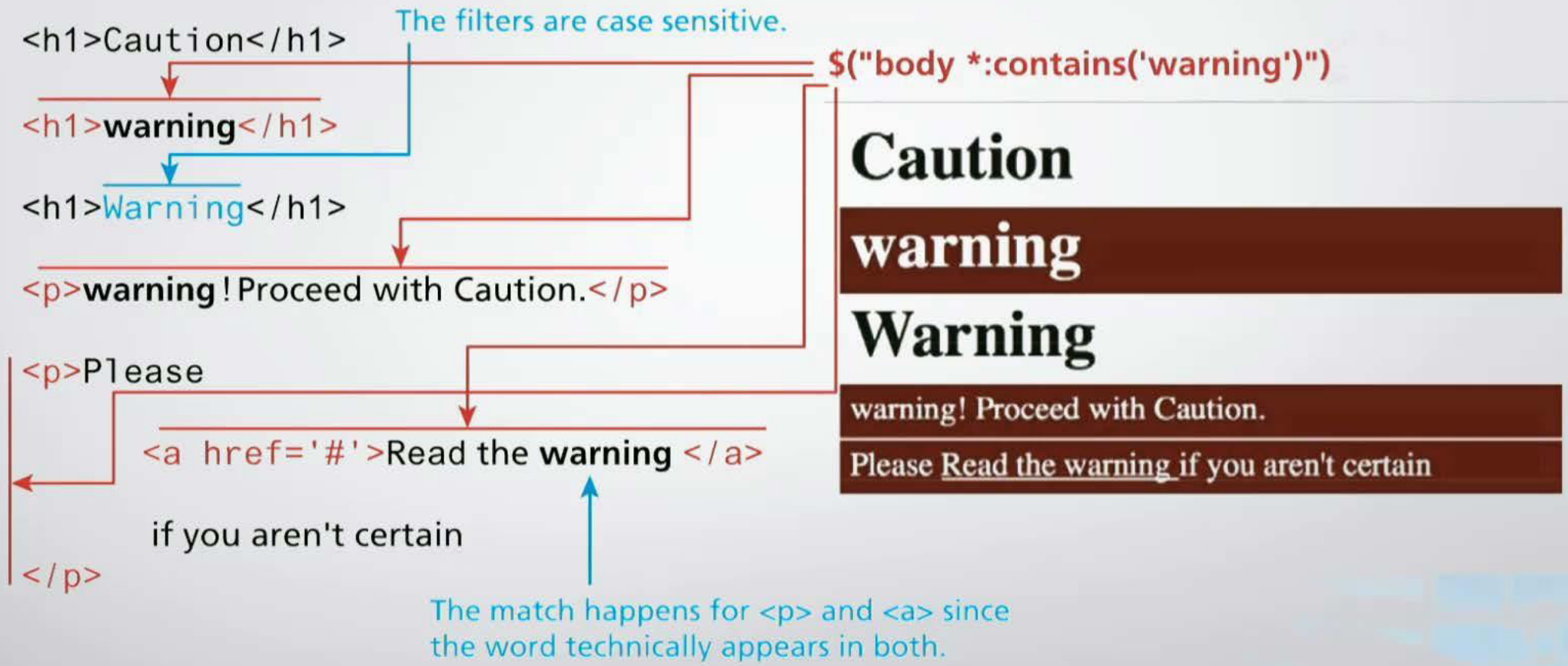
$("ul a:link")

$("#main time")

$("#main>time")

$("#main div p:first-child")

jQuery's Content Filter Selection



TOPIC 63

jQuery Manipulators

Web Systems and Technologies

Common Element Manipulations – HTML

- We can both set and get innerHTML through the html() method

```
//link is assigned the html code contained
by the first p
var html = $("p").html();

//change the innerHTML of an element
$("p#message").html("<b>Invalid Email</b>");

//change innerHTML of all matching elements
$("p.warning").html("No warning");
```

- text() can be used in similar manner for innerText

Common Element Manipulations – HTML Attributes

- We can both set and get an attribute value by using the attr() method

```
// 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");
```

Common Element Manipulations – HTML Properties

- The prop() method retrieves and set the value of a property, often used for form elements

```
<input class="meh" type="checkbox"
checked="checked">
```

```
var theBox = $(".meh");
theBox.prop("checked");
// evaluates to TRUE
```

- val() used for getting/setting values of elements having the value property (e.g., <input type="text"…>

Common Element Manipulations – Changing CSS

- jQuery provides the extremely intuitive css() method

```
var color =
$("#element").css("background-color");
// get the color

$("#element").css("background-color",
"red"); // set color to red
```

Web Systems and Technologies

WEEK 8

TOPIC 64 TO 72

Only audio/video lectures

Web Systems and Technologies

64 • Query Manipulators - Example
65 • Events in Query
66 • DOM Manipulation using jQuery
67 • Animations using jQuery
68 • Introduction to Ajax
69 • Asynchronous File Transmission
70 • Introduction to server side development, Comparison of server side technologies
71 • Installation of Apache webserver, PHP and MySQL for Local Development
72 • Introduction to PHP

Web Systems and Technologies

WEEK 9

TOPIC 73 TO 81

Web Systems and Technologies

- 73 -Introduction to PHP (String Concatenation)
- 74 - Program Control - Conditional Program Statements
- 75 Program Control - Loops
- 76 Functions in PHP (1)
- 77 Functions in PHP(2)
- 78 Arrays in PHP
- 79 Sorting Arrays
- 80 Arrays - Super Global
- 81 $_GET and $_POST Arrays

TOPIC 73

Introduction to PHP – String Concatenation

Web Systems and Technologies

String Concatenation

- Strings can be concatenated using (.) operator

```
$username = "Ricardo";
echo "Hello". $username;
//outputs Hello Ricardo
```

More concatenation examples

1 `echo "<img src='23.jpg' alt='" . $firstName . " " . $lastName . "' >";`

outputs

`<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" >`

More concatenation examples

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>
```

TOPIC 74

Program Control – Conditionals

Web Systems and Technologies

Comparison Operators

- Equal to operator ==

```
5 == '5'; //true
```

- Identical operator ===

```
5 === '5'; //false
```

if ...else

```
// if statement
if ( $hourOfDay > 6 && $hourOfDay < 12)
{
  $greeting = "Good Morning";
}
//optional else if
else if ($hourOfDay == 12) {
  $greeting = "Good Noon Time";
}
// optional else branch
else {
  $greeting = "Good Afternoon or
Evening";
}
```

Alternate Syntax for Control Structure

```
<?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; ?>
```

switch . . . case

- Uses equal to operator

```
switch ($artType) {
  case "PT":
    $output = "Painting";
    break;
  case "SC":
    $output = "Sculpture";
    break;
  default:
  $output = "Other";
}
```

match

- Introduced in PHP 8
  - Uses identical operator
  - Can return a value
  - Break not required
  - No default case (throws exception if not matched) or use conditions for default

```
$output = match ($artType) {
   "PT" => "Painting",
   "SC" => "Sculpture",
};
```

TOPIC 75

Program Control – Loops

Web Systems and Technologies

while and do . . . while

```
$count = 0;
while ($count < 10){
   echo $count;
   $count++;
}
```

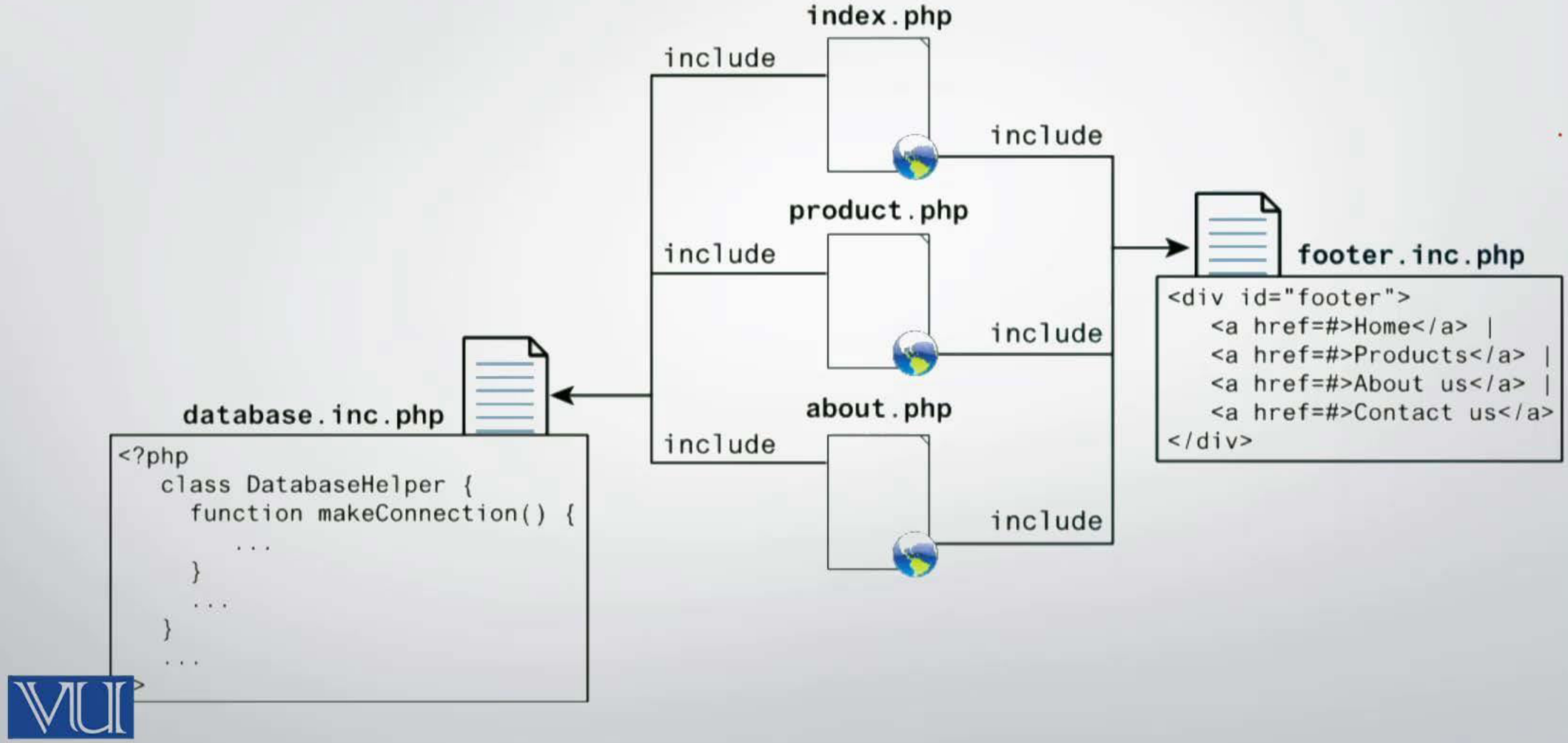
while and do . . . while

```
$count = 0;
do {
   echo $count;
   // this one increments the count
by 2 each time
   $count = $count + 2;
} while ($count < 10);
```

for

```
for ($count=0; $count < 100; $count+=5)
{
   echo $count;
}
```

Include Files



Include Files

- include "somefile.php";
- include_once "somefile.php";

TOPIC 76

Functions (1)

Web Systems and Technologies

Copyright © 2021 Victoria University of Wellington

Function Syntax

- Function starts with the keyword **function**
- Can take zero or more parameters
- May or may not return a value

```
function square($num) {
  return $num*$num;
}

$sq = square(5);
```

Function Syntax – Return Type Declaration

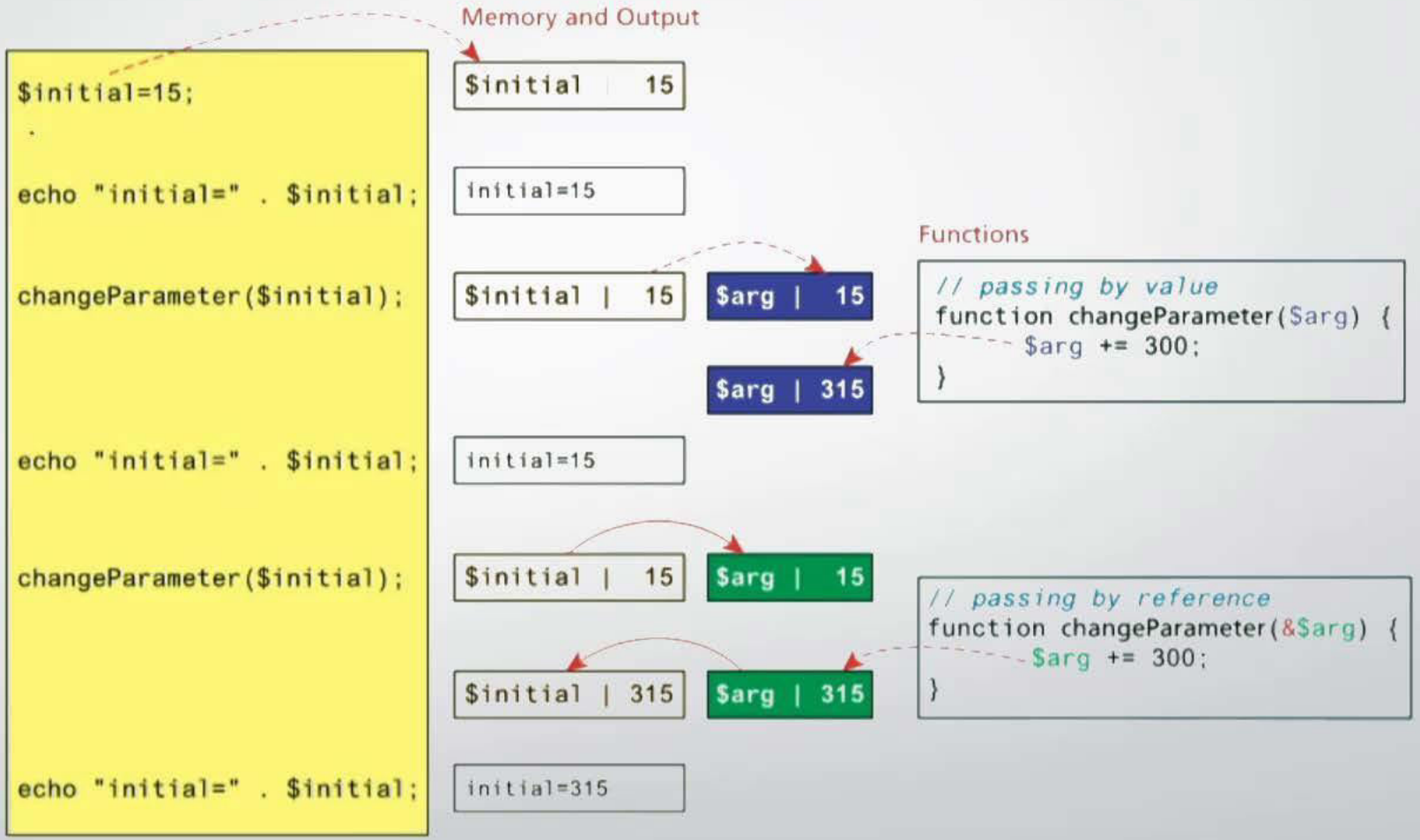
- A Return Type Declaration explicitly defines a function's return type by adding a colon and the return type after the parameter list when defining a function

```
function mustReturnString() : string {
  return "hello";
}
```

Parameters – Passing Parameters by Reference

- By default, arguments passed to functions are **passed by value** in PHP
- 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

Parameters – Passing Parameters by Reference



Parameters – Default Values

- Parameters can have default values

```
function func($a=1, $b=2, $c=3) {
  echo $a." ".$b." ".$c;
}

func(5,10,15);     //5 10 15
func(5,10);        //5 10 3
func();            //1 2 3
```

TOPIC 77

Functions (2)

Web Systems and Technologies

Parameters – Default Values

- Parameters can have default values

```
function func($a=1, $b=2, $c=3) {
  echo $a." ".$b." ".$c;
}

func(5,10,15);     //5 10 15
func(5,10);        //5 10 3
func();            //1 2 3
```

Parameters – Named Arguments

- Named arguments supported in PHP 8

```
function func($a=1, $b=2, $c=3) {
  echo $a." ".$b." ".$c;
}

func(5,10,15);   //5 10 15
func();          //1 2 3
func(c: 5);      //1 2 5
```

Variable Scope within Functions

- Variables defined within a function have function scope
- While variables defined in the main script are said to have global scope, by default, not available within functions
- PHP does allow variables with global scope to be accessed within a function using the **global** keyword

```
$x=10;
function func() {
  global $x;
  echo $x;
}
```

TOPIC 78

Arrays

Web Systems and Technologies

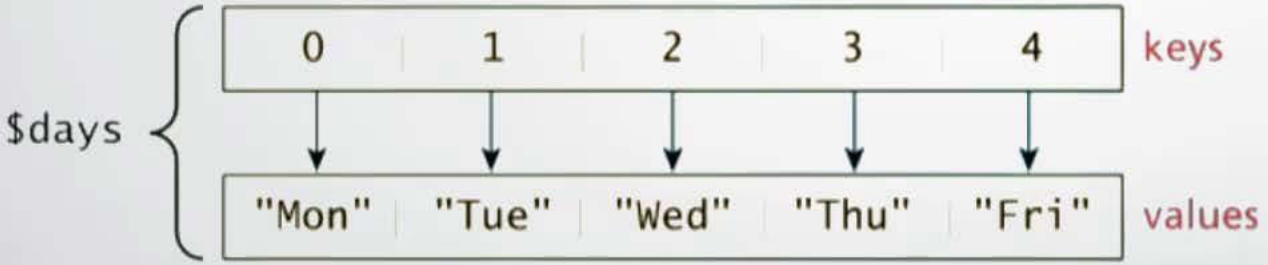
Defining and Accessing an Array

- Using array function

```
$days =
array("Mon","Tue","Wed","Thu","Fri");
```

- Using array notation

```
$days = ["Mon","Tue","Wed","Thu","Fri"];
```

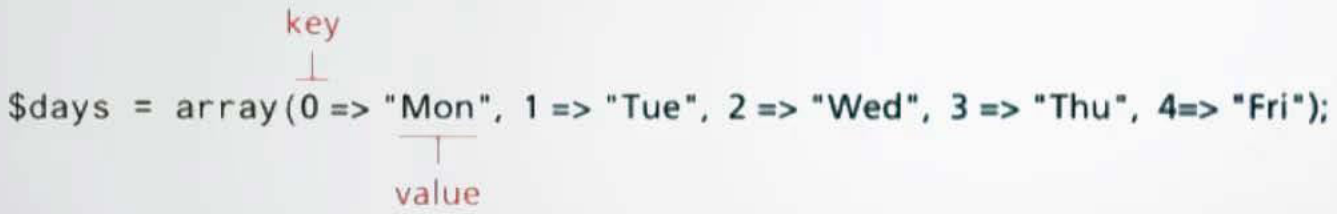


- For debugging, use *var_dump($days)* or *print_r($days)* to see contents of the array (or any variable). Place output in <pre></pre>

```
<pre><?php var_dump($days); ?></pre>
```

Defining and Accessing an Array

- All arrays in PHP are generally referred to as **associative arrays**



Arrays

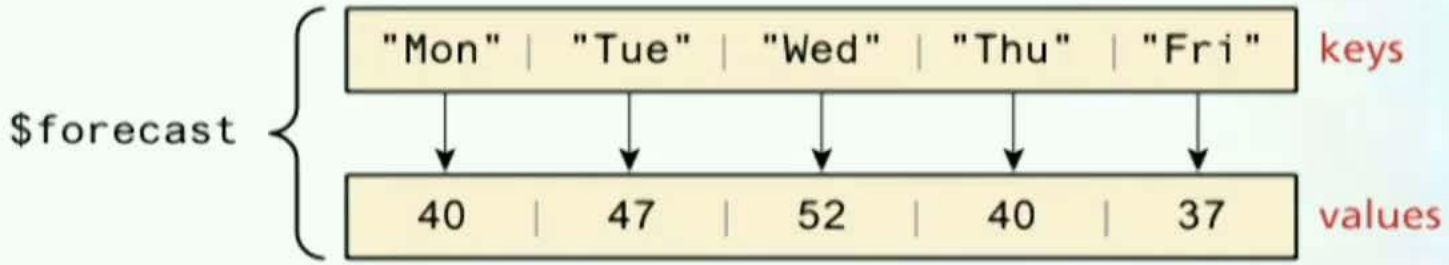
Defining and Accessing an Array

- You can use integer and string keys, not necessarily in order

```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```

key: "Mon", "Tue", "Wed", "Thu", "Fri"

value: 40, 47, 52, 40, 37

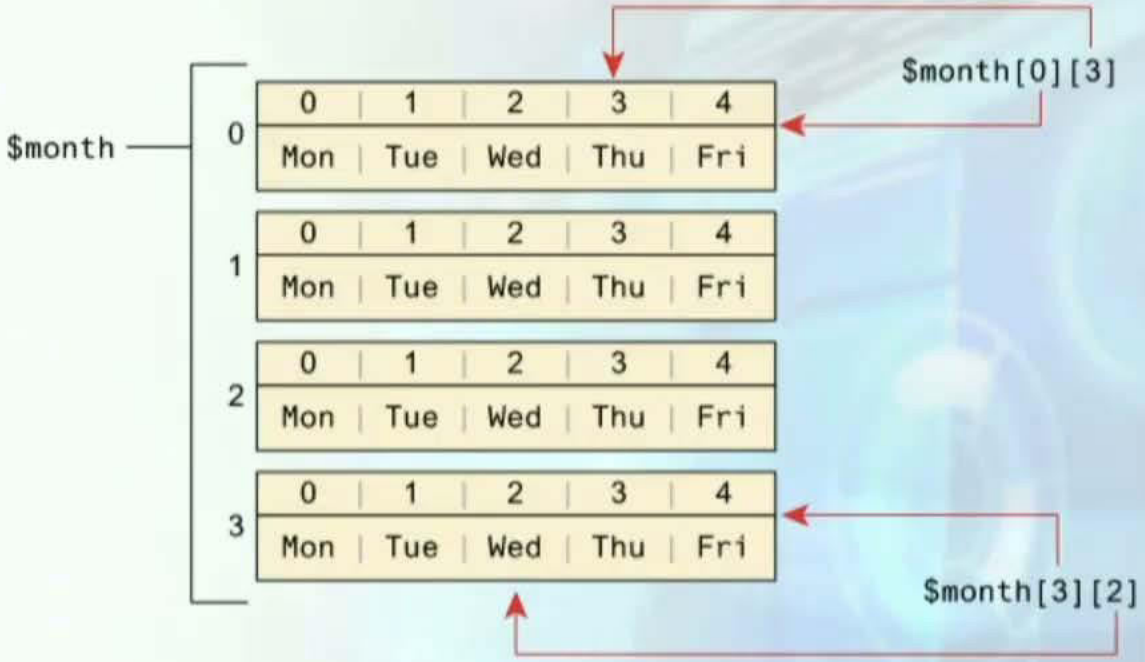



```
echo $forecast["Tue"];   // outputs 47
echo $forecast["Thu"];   // outputs 40
```

Arrays

Multidimensional Arrays

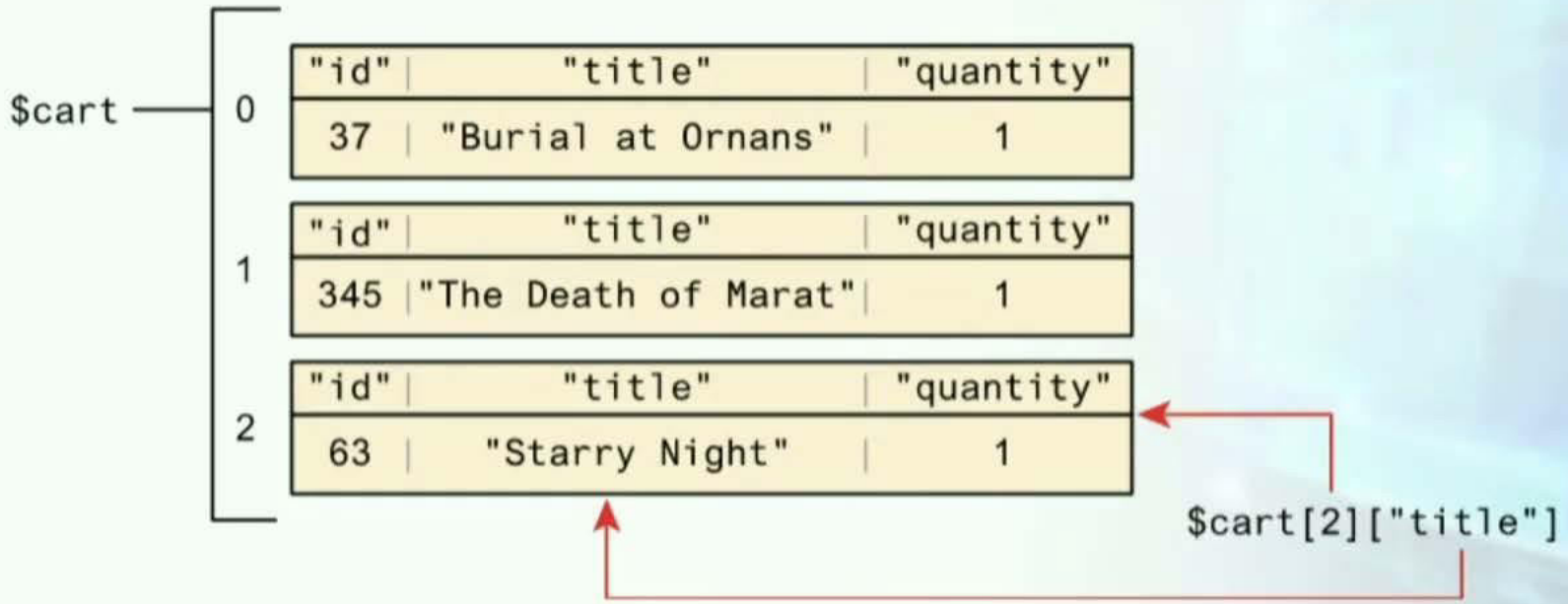
```
$month = array
(
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri")
);
```



Arrays

Multidimensional Arrays

```
$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);
```



Iterating through an Array - for

```
// for loop
for ($i=0; $i<count($days); $i++) {
  echo $days[$i] . "<br>";
}
```

Iterating through an Array - foreach

```php
//iterating through the values
foreach ($forecast as $value) {
  echo $value . "<br>";
}
// iterating through the values AND the keys
foreach ($forecast as $key => $value) {
  echo "day[" . $key . "]=" . $value;
}
```

Adding and Deleting Elements

- An element can be added to an array simply by using a key/index that hasn't been used, as shown below:

```
$days[5]= "Sat";
```

- As an alternative to specifying the index, a new element can be added to the end of any array using empty square brackets after the array name, as follows:

```
$days[]="Sun";
```

- Delete with unset()

TOPIC 79

Arrays – Sorting

Web Systems and Technologies

Array Sorting

- 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)
```

- asort($days);
  - The resulting array in this case keeps associations so is:

```
Array ([4] => Fri [0] => Mon [5] =>
Sat [6] => Sun [3] => Thu [1] =>
Tue [2] => Wed)
```

More Array Operations

```
array_keys($someArray)

array_values($someArray)

array_rand($someArray, $num=1)

shuffle($someArray)

array_reverse($someArray)

array_walk($someArray, $callback,
$optionalParam)

in_array($needle, $haystack,
$optionalStrict)
```

TOPIC 80

Arrays – Super Globals

Web Systems and Technologies

Copyrights © 2021 Virtual University of Pakistan

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

Superglobal Arrays

- $GLOBALS
  - Array for reading and storing data that needs superglobal scope
- $_COOKIES
  - Array of cookie data passed to page via HTTP request
- $_SESSION
  - Array that contains session data

Superglobal Arrays

- $_ENV
  - Array of server environment data (mostly OS related and paths)
- $_SERVER
  - Array containing information about the request and the server

Superglobal Arrays

- $_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
- $_FILES
  - Array of file items uploaded to the server

TOPIC 81

$_GET and $_POST Superglobal Arrays

Web Systems and Technologies

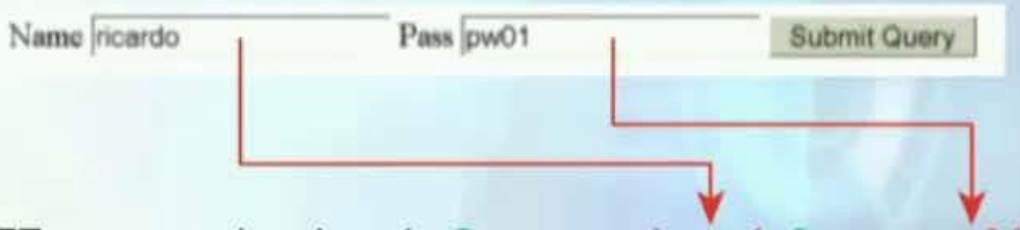
Copyright © 2024 Virtual University of Pakistan

Reading Form Data – GET

HTML (client)

```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



HTTP request

```
GET processLogin.php?uname=ricardo&pass=pw01
```

PHP (server)

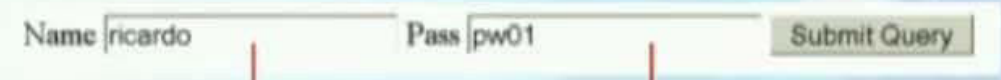
```
// within processLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```

Reading Form Data – POST

HTML (client)

```
<form action="processLogin.php" method="POST">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



HTTP request

POST processLogin.php

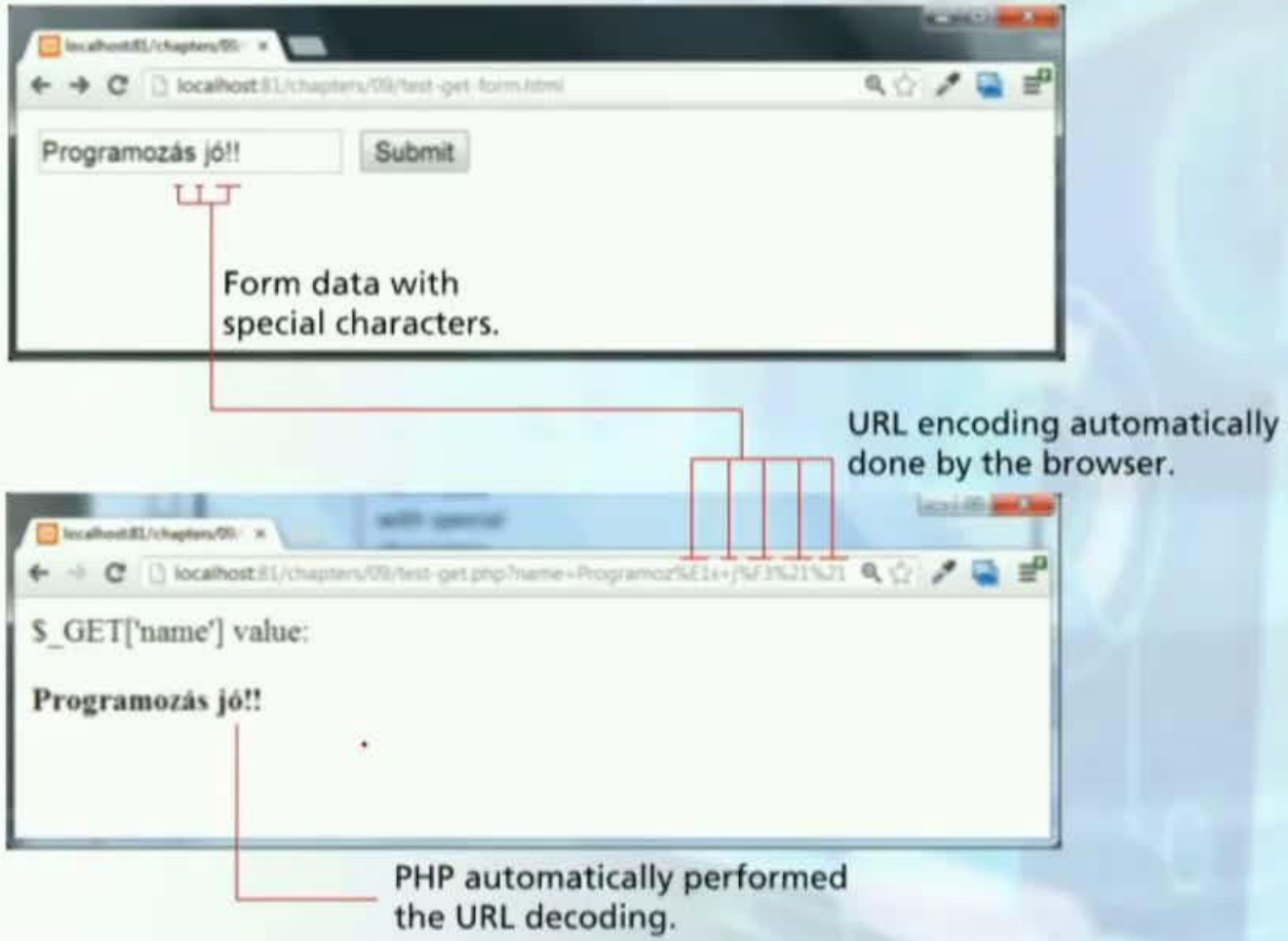
HTTP POST request body:

uname=ricardo&pass=pw01

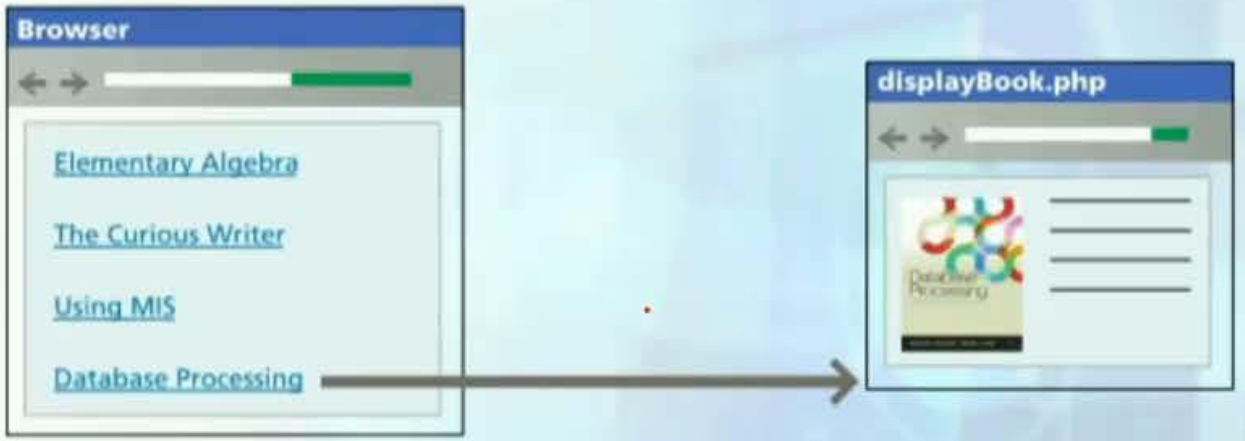
PHP (server)

```
//File processLogin.php
echo $_POST["uname"]; //outputs "ricardo";
echo $_POST["pass"];  //outputs "pw01";
```

Note URL Encoding and Decoding



Using Query Strings in Hyperlinks



Sanitizing Query Strings

- That is, just because you are expecting a proper query string, it doesn't mean that you are going to get one. Your program must be able to handle:
  - If query string parameter doesn't exist
  - If query string parameter doesn't contain a value
  - If query string parameter value isn't the correct type or is out of acceptable range
  - If value is required for a database lookup, but provided value doesn't exist in the database table

Web Systems and Technologies

WEEK 10

TOPIC 82 TO 90

Web Systems and Technologies

- 82 $_GET and $_POST arrays - Example
- 83 $_SERVER Array
- 84 $_FILES Array
- 85 Reading and Writing Files
- 86 Classes and Objects in PHP; Object Properties
- 87 Constructor; Methods
- 88 Visibility
- 89 Static members; Constants
- 90 Databases and Web Development

TOPIC 82

$_GET and $ _POST Superglobal Arrays - Example

Web Systems and Technologies

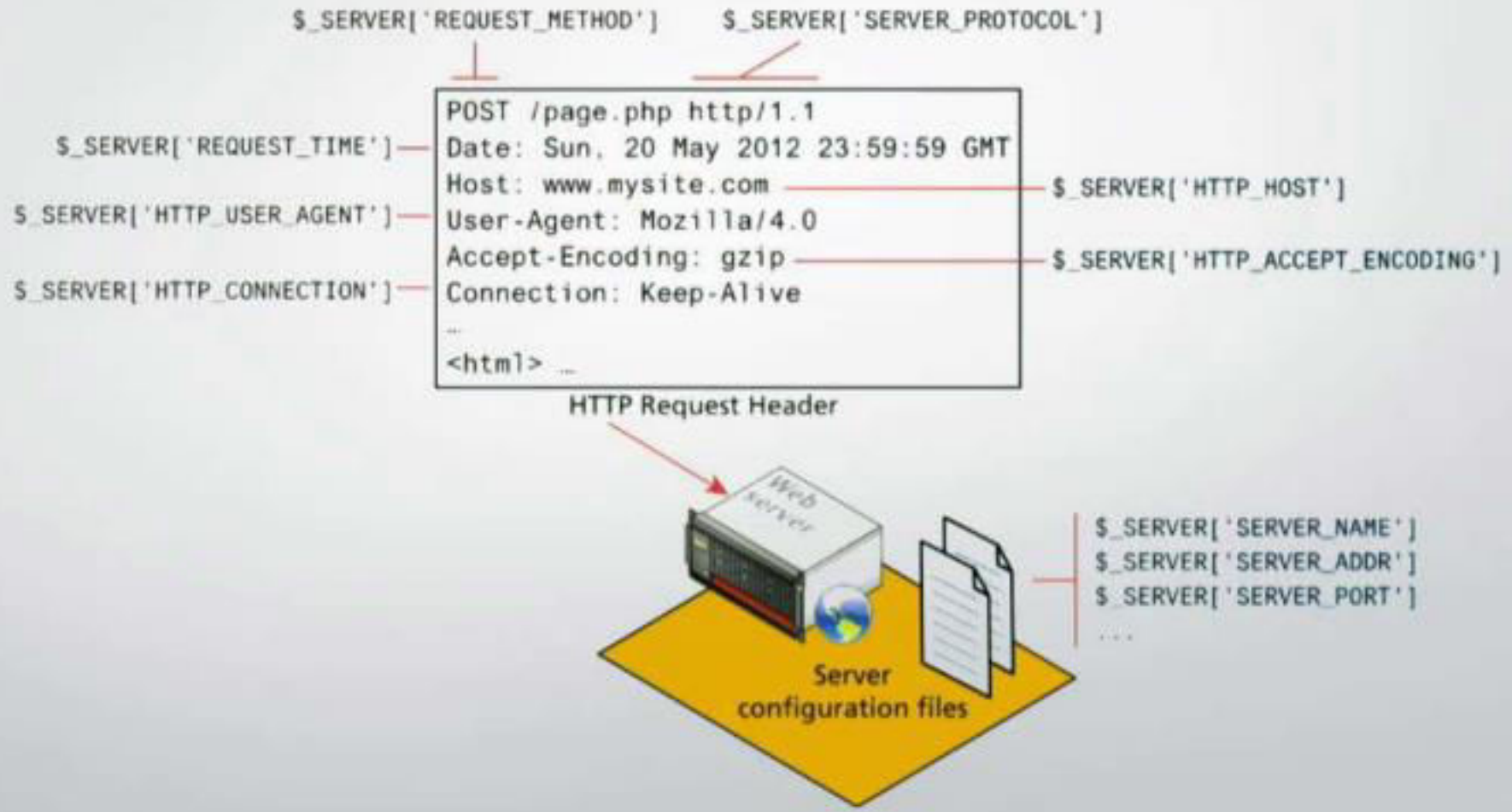
$_SERVER Array

TOPIC 83

Web Systems and Technologies

$_SERVER Array

Server Information Keys



Request Header Information Keys

- Advanced browser detection, need to get browscap.ini and configure php.ini [browscap] to use it

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
$browser =
get_browser($_SERVER['HTTP_USER_AGENT']
, true);
print_r($browser);
?>
```

Request Header Information Keys

```
$previousPage =
$_SERVER['HTTP_REFERER'];

// Check to see if referer was our
search page

if (strpos($previousPage,"search.php")
!= 0) {

  echo "<a href='search.php'>Back to
search</a>";

}

// Rest of HTML output
```

TOPIC 84

$_FILES Array

Web Systems and Technologies

HTML Required for File Uploads

- First, you must ensure that the HTML form uses the HTTP POST method
- Second, you must add the enctype="multipart/form-data" attribute to the HTML form that is performing the upload
- Finally you must include an input type of file in your form.

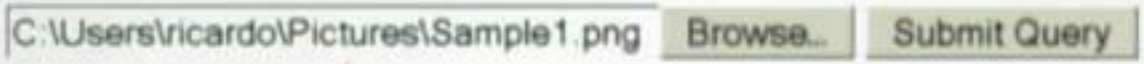
```
<form enctype='multipart/form-data'
method='post'>
  <input type='file' name='file1'
id='file1'>
  <input type='submit'>
</form>
```

Handling the File Upload in PHP

HTML (client)

```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file' name='file1'>
   <input type='submit' value="Submit Query">
</form>
```

Browser (client)



HTTP request

POST upFile.php

HTTP POST multipart/form-data

```
file1%PNG ™‡›•Ì ! %çÕkáFÊ÷Ì§29%*úÅ¡Šrá vÛ.ÿiN üc/©(–Å Å Z}/vè\Å(m–%i± %6_Ê/Hi.+*" _AÉA`) .þ/
_êvßoá'™ ß%uÔaGé#N^ÔÔ?ÔŸ·ì'm ½Æ2h‹ ¶•`¦QÅðiçD"W)MìÛü_9ß9nŸ02'Å' km'Ñyþh $2'›Ÿ` d®81¬\
¡ûh»}>´‡¥[ ðB*ðèè[ ‡¥uè|ŸxR%XL}áE›[ÕZ!pá X¢ kÔ >Ô 'cÛÿ}¤‡ ÅÔt©Q`iÜ±¥<3%üÔ3 ×1'ò¡¶1¬W+4èê
®à,Ôo9'tâÆC ®1Ü11Êée`ß>{Üèaĝ® ®Öè|¶ìÞ%soiz*VydÜÊµ'ÅøÅ_ Å "Ç jñzöce®%Ÿ-B `¬àã'~ÀII'¡íüü 5%$1 "
Z³Ñ ðº Å¶ £_hÉGÀ ¢°÷¥ì>Éƒ2Ï¢‹ûT _Å| –ð€|®Å±ç€_`nü»ßð®üÐmŸa ì ¿ô{?üö÷1™ Ü_ìê?\ÉÅ /
æsÚ"®%%tëy``>qàíÜöÎÔøþ Ì÷ó7ßü› ‡ßßÅ, ßæò'?÷ÀìÎ1Xtæ11%x8ô 7¿ ñÜLöoþ1 Ü7~1¦;¶ì>d§éà >
®¶Å:K(*Gëõ±xÅ Üsx `wnÜ%o×¶ñ±ýX]1Ii\ ^'fÅéuéL6>yÉÅ%ÇÐUÜ`ý#÷×èÕøàa° ÊW"ó<wÉSè'ÇGf&ÇàÎQß›?eà
...
```

```
echo $_FILES["file1"]["name"]      // "Sample1.png"
echo $_FILES["file1"]["type"]      // "image/png"
echo $_FILES["file1"]["tmp_file"]  // "/tmp/phpJ08pVh"
echo $_FILES["file1"]["error"]     //  0
echo $_FILES["file1"]["size"]      // 1219038
```

PHP (server)



Checking for Errors

```
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 ";
  }
}
```

File Size Restrictions

- You can limit in multiple ways
- HTML form attributes in inputs (browser)
- JavaScript (browser)
- PHP validation (server)

Limiting the Type of File Upload

```
$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";
  }
}
```

Moving the File

```
$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.";
}
```

TOPIC 85

Reading/Writing Files

Web Systems and Technologies

Two ways

- There are two basic techniques for read/writing files in PHP:
  - Stream access. Just read a small portion of the file at a time
  - All-In-Memory access. Read the entire file into memory (i.e., into a PHP variable). While not appropriate for large files, it does make processing of the file extremely easy.

Stream Access

- To those of you familiar with functions like fopen() , fclose() , and fgets() from the C programming language, this first technique will be familiar

- Open fopen()
- Read data fgets()
- Close the file fclose()

In-Memory File Access

- file() Reads the entire file and returns an array, with each array element corresponding to one line in the file.
- file_get_contents() Reads the entire file and returns a string variable.
- file_put_contents() Writes the contents of a string variable out to a file.

Classes and Objects in PHP

TOPIC 86

Web Systems and Technologies

Defining Classes

```
class Artist {
    public $firstName;
           $lastName;
    public $birthDate;
    public $birthCity;
           $deathDate;
    }
```

Instantiating Objects

- Use the new keyword

```
$picasso = new Artist();
$dali = new Artist();
```

Properties

- Once you have instances of an object, you can access and modify the properties using the object's 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";
```

Web Systems and Technologies

TOPIC 87

Classes and Objects in PHP Constructor & Methods

Classes and Objects in PHP

Constructor

```
class Artist {
function __construct($firstName, $lastName, $city, $birth,
$death=null) {
$this->firstName = $firstName;
$this->lastName = $lastName;
$this->birthCity = $city;
$this->birthDate = $birth;
$this->deathDate = $death;
}
}
```

Classes and Objects in PHP

Instantiating using Constructors

```
$picasso = new
Artist("Pablo","Picasso","Malaga","Oct 25,1881", "Apr 8,1973");
$dali = new
Artist("Salvador","Dali","Figures","May 11 1904", "Jan 23 1989");
```

Classes and Objects in PHP

Methods

- Methods define the tasks each instance of a class can perform and are useful since they associate behavior with objects

```
class Artist {
// ...
public function outputAsTable() {
//...
}
}

$a->outputAsTable()
```

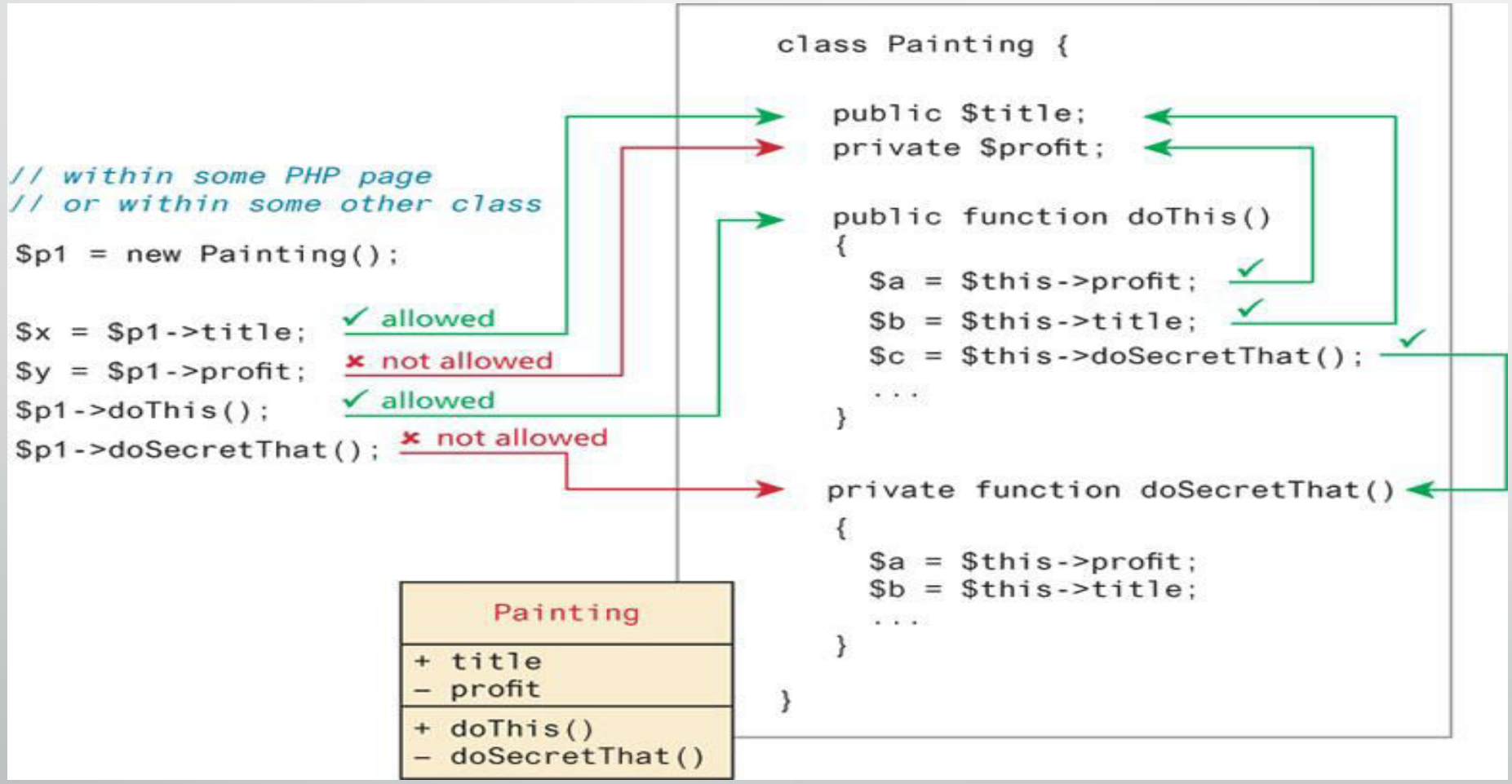
Web Systems and Technologies

TOPIC 88

Classes and Objects in PHP Visibility

Classes and Objects in PHP

Visibility



Web Systems and Technologies

TOPIC 89

Classes and Objects in PHP Static Members & Constants

Classes and Objects in PHP

Static Members

- A **Static member** is a property or method that all instances of a class share
- Static value is shared across all instances

```
class Artist {
public static $artistCount = 0;
public static function addArtist() {
self:: $artistCount++;
}
}
Artist::addArtist();
print(Artist::$artistCount);
```

Classes and Objects in PHP

Class Constants

- **Constant Values** can be stored more efficiently as class constants so long as they are not calculated or updated

```
Const   EARLIEST_DATE ='January1, 1200';
```

- They can be accessed both inside and outside theclass using self::EARLIEST_DATE in the class and classReference::EARLIEST_DATE

Web Systems and Technologies

TOPIC 90

Databases and Web Development

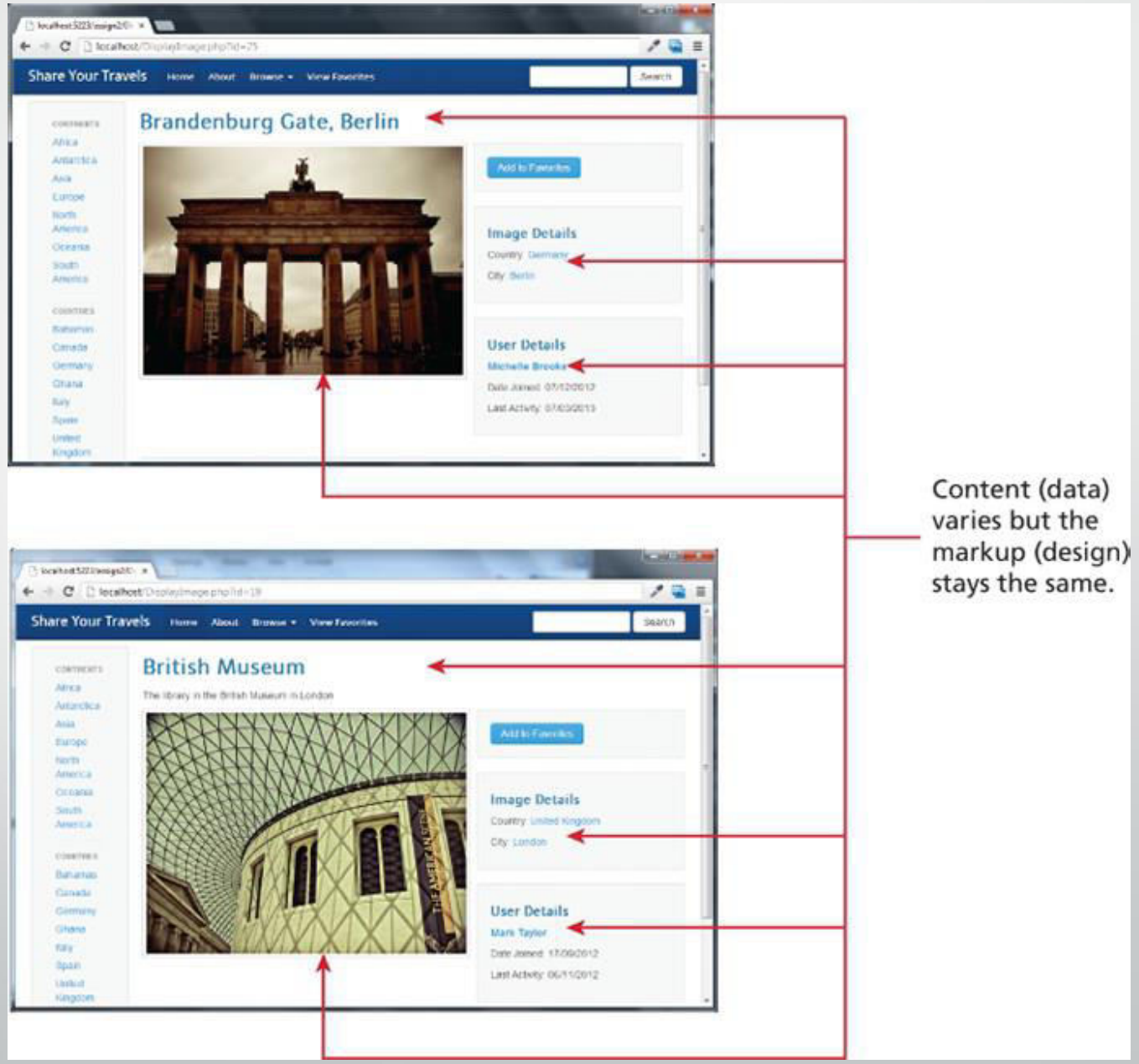
Databases and Web Development

The Role of Databases in Web Development

- HTML pages remain constant
- Using databases with PHP allows to make parts of HTML dynamic(variable)

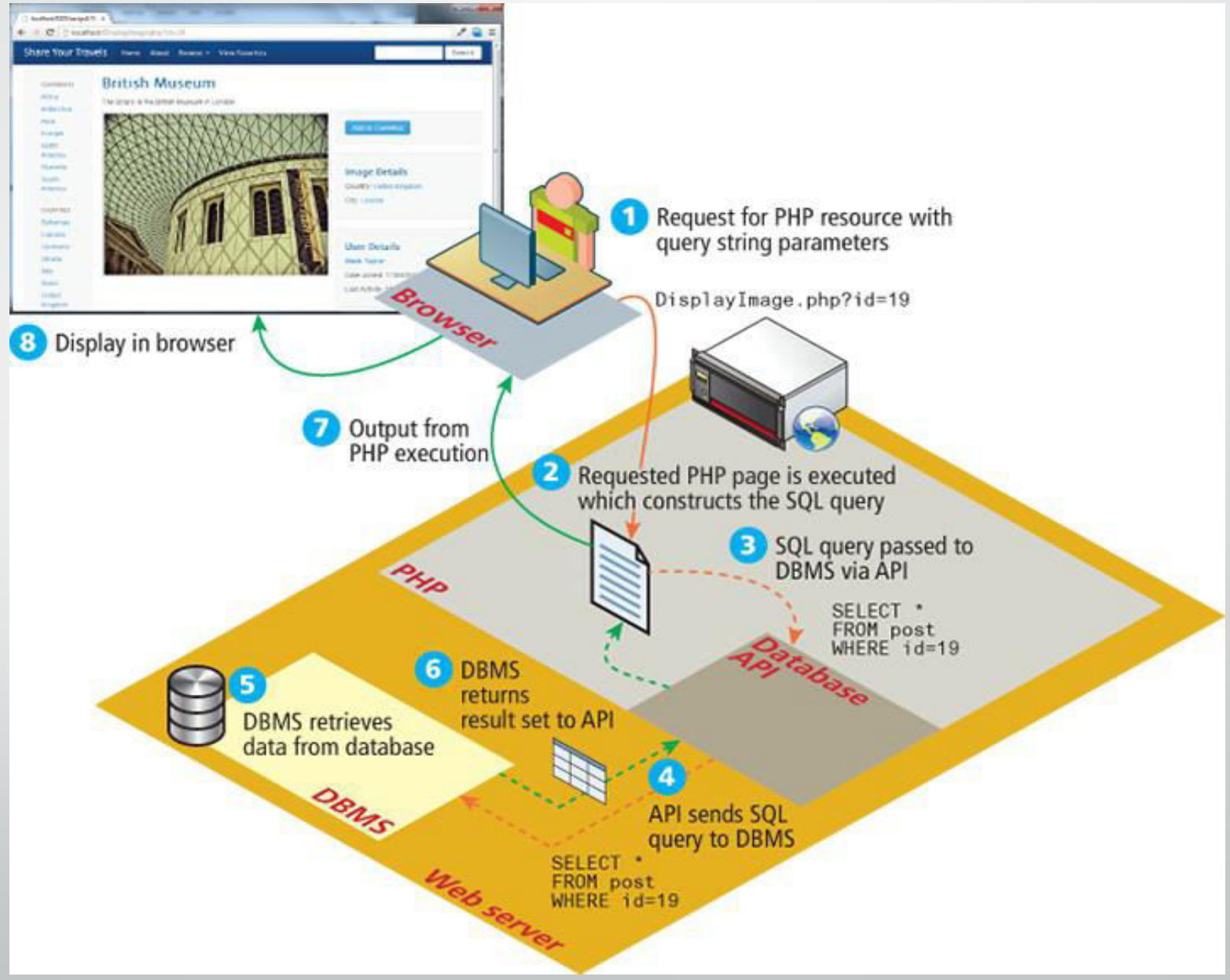
Databases and Web Development

Role of databases in web development



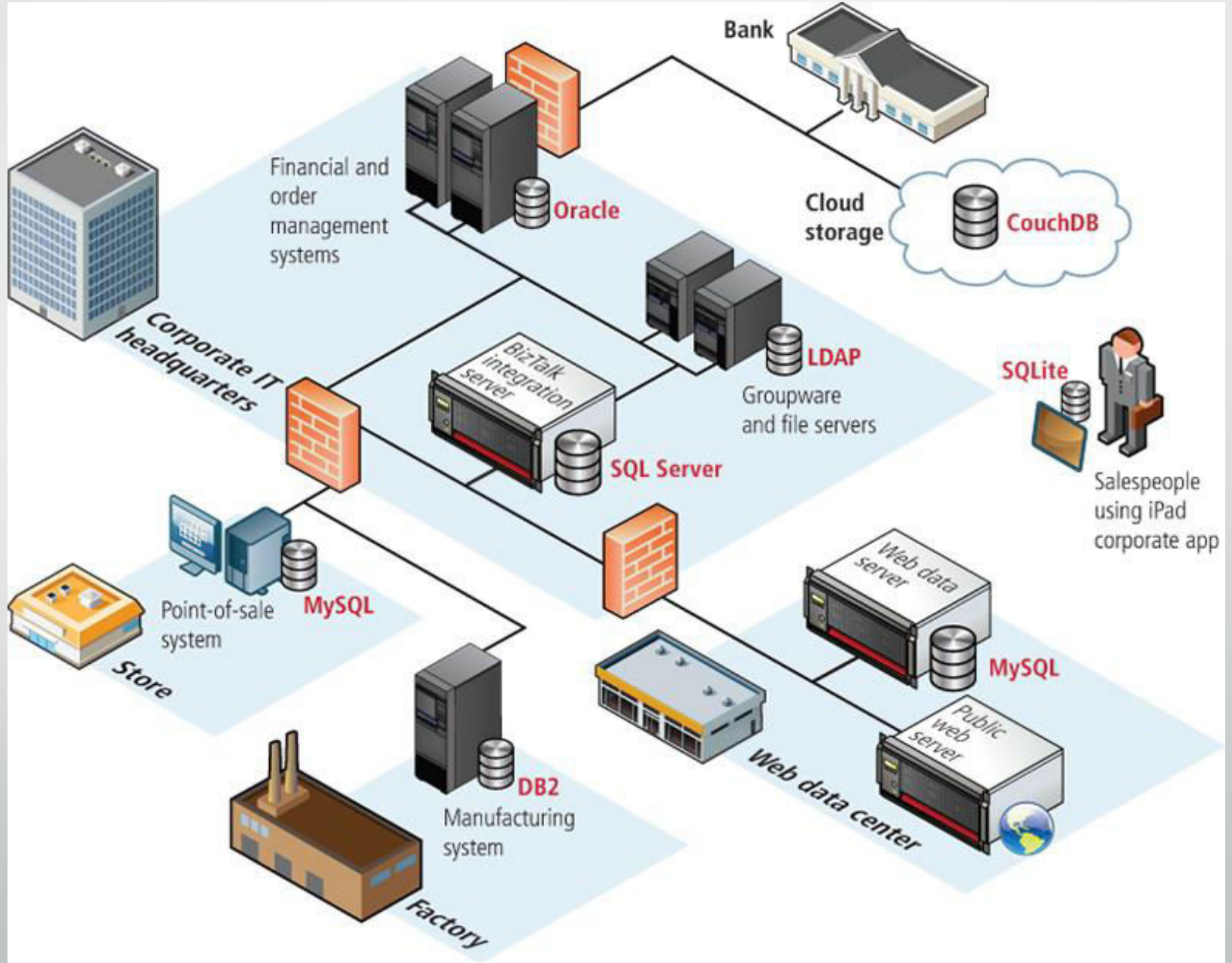
Databases and Web Development

How websites use databases



Databases and Web Development

Database options



Web Systems and Technologies

WEEK 11

TOPIC 91 TO 99

SQL,Database,PHP

- 91 SQL - Select, Insert, Update, Delete
- 92 Introduction to MySQL
- 93 phpMyAdmin
- 94 Accessing MySQL in PHP
- 95 Fetching Data
- 96 Performing Transactions
- 97 Example: Database Creation
- 98 Example: Database Insertion
- 99 Example: Database Search

SQL – Select, Insert, and Update

TOPIC 91

Web Systems and Technologies

SELECT Statement

SQL keyword that indicates the type of query (in this case a query to retrieve data)

SQL keyword for specifying the tables

```
SELECT ISBN10, Title FROM Books
```

Fields to retrieve

Table to retrieve from

```
SELECT * FROM Books
```

Wildcard to select all fields

Note: While the wildcard is convenient, especially when testing, for production code it is usually avoided; instead of selecting every field, you should select just the fields you need.

```
select iSbN10, title
FROM BOOKS
ORDER BY title
```

SQL keyword to indicate sort order

Field to sort on

Note: SQL doesn't care if a command is on a single line or multiple lines, nor does it care about the case of keywords or table and field names. Line breaks and keyword capitalization are often used to aid in readability.

```
SELECT ISBN10, Title FROM Books
ORDER BY CopyrightYear DESC, Title ASC
```

Keywords indicating that sorting should be in descending or ascending order (which is the default)

Several sort orders can be specified: in this case the data is sorted first on year, then on title

Use the WHERE Clause

```
SELECT isbn10, title FROM books
WHERE copyrightYear > 2010
```

SQL keyword that indicates to return only those records whose data matches the criteria expression

Expressions take form: field *operator* value

```
SELECT isbn10, title FROM books
WHERE category = 'Math' AND copyrightYear = 2014
```

Comparisons with strings require string literals (single or double quote)

Join together



Because the field name ArtistID is ambiguous, need to preface it with table name

Table 1

```
SELECT Artists.ArtistID, Title, YearOfWork, Name FROM Artists
INNER JOIN ArtWorks ON  Artists.ArtistID = ArtWorks.ArtistID
```

SQL keywords indicate the type of join

Table 2

Primary key in Table 1

Foreign key in Table 2




```
SELECT Books.BookID, Books.Title, Authors.Name, Books.CopyrightYear
FROM Books
INNER JOIN (Authors INNER JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorId)
ON Books.BookID = BookAuthors.BookId
```

Member Group By

This aggregate function returns a count of the number of records

Defines an alias for the calculated value

```
SELECT Count(ArtWorkID) AS NumPaintings
FROM ArtWorks
WHERE YearOfWork > 1900
```

Count number of paintings after year 1900

Note: This SQL statement returns a single record with a single value in it.

NumPaintings
745

```
SELECT Nationality, Count(ArtistID) AS NumArtists
FROM Artists
GROUP BY Nationality
```

SQL keywords to group output by specified fields

Note: This SQL statement returns as many records as there are unique values in the group-by field.

Nationality	NumArtists
Belgium	4
England	15
France	36
Germany	27
Italy	53

INSERT, UPDATE, and DELETE Statements

SQL keywords for inserting (adding) a new record

Table name

Fields that will receive the data values

```
INSERT INTO ArtWorks (Title, YearOfWork, ArtistID)
VALUES ('Night Watch', 1642, 105)
```

Values to be inserted. Note that string values must be within quotes (single or double).

Note: Primary key fields are often set to AUTO_INCREMENT, which means the DBMS will set it to a unique value when a new record is inserted.

```
INSERT INTO ArtWorks
SET Title='Night Watch', YearOfWork=1642, ArtistID=105
```

Nonstandard alternate MySQL syntax, which is useful when inserting record with many fields (less likely to insert wrong data into a field).

```
UPDATE ArtWorks
SET Title='Night Watch', YearOfWork=1642, ArtistID=105
WHERE ArtWorkID=54
```

It is essential to specify which record to update, otherwise it will update all the records!

Specify the values for each updated field. *Note: Primary key fields that are AUTO_INCREMENT cannot have their values updated.*

```
DELETE FROM ArtWorks
WHERE ArtWorkID=54
```

It is essential to specify which record to delete, otherwise it will delete all the records!

Transactions

- By starting the transaction, all database modifications within the transaction will only be permanently saved in the database if they all work

START TRANSACTION

INSERT INTO orders . . .

INSERT INTO orderDetails . . .

UPDATE inventory . . .

/* if we have made it here everything has worked so commit changes */

COMMIT

/* if we replace COMMIT with ROLLBACK then the three database changes would be "undone" */

Introduction to MySQL

TOPIC 92

Web Systems and Technologies

Command-Line Interface

```
Database changed
mysql> SHOW TABLES;
+--------------------------+
| Tables_in_book_database |
+--------------------------+
| authors                  |
| bindingtypes             |
| bookauthors              |
| books                    |
| categories               |
| disciplines              |
| imprints                 |
| productionstatuses       |
| subcategories            |
+--------------------------+
9 rows in set (0.00 sec)

mysql> SHOW COLUMNS IN authors;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int(11)      | NO   | PRI | NULL    | auto_increment |
| FirstName   | varchar(255) | YES  |     | NULL    |                |
| LastName    | varchar(255) | YES  |     | NULL    |                |
| Institution | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM authors WHERE FirstName LIKE "A%";
+-----+-------------+-----------+-----------------------------------------------+
| ID  | FirstName   | LastName  | Institution                                   |
+-----+-------------+-----------+-----------------------------------------------+
|   2 | Andrew      | Abel      | Wharton School of the University of Pennsylvania |
|  25 | Allen       | Center    | NULL                                          |
|  37 | Allen       | Dooley    | Santa Ana College                             |
|  40 | Andrew      | DuBrin    | Rochester Institute of Technology             |
|  56 | Allan       | Hambley   | NULL                                          |
|  57 | Arden       | Hamer     | Indiana University of Pennsylvania            |
|  82 | Arthur      | Keown     | Virginia Polytechnic Instit. and State University |
| 102 | Annie       | McKee     | NULL                                          |
| 119 | Arthur      | O'Sullivan | NULL                                         |
| 172 | Allyn       | Washington | Dutchess Community College                   |
| 194 | Anne Frances | Wysocki  | University of Wisconsin, Milwaukee            |
| 198 | Alice M.    | Gillam    | University of Wisconsin-Milwaukee             |
| 214 | Anthony P.  | O'Brien   | Lehigh University                             |
| 216 | Alvin C.    | Burns     | NULL                                          |
| 225 | Abbey       | Deitel    | NULL                                          |
| 252 | Alvin       | Arens     | Michigan State University                     |
| 258 | Ali         | Ovlia     | NULL                                          |
| 270 | Anne        | Winkler   | NULL                                          |
| 275 | Alan        | Marks     | DeVry University                              |
+-----+-------------+-----------+-----------------------------------------------+
19 rows in set (0.00 sec)

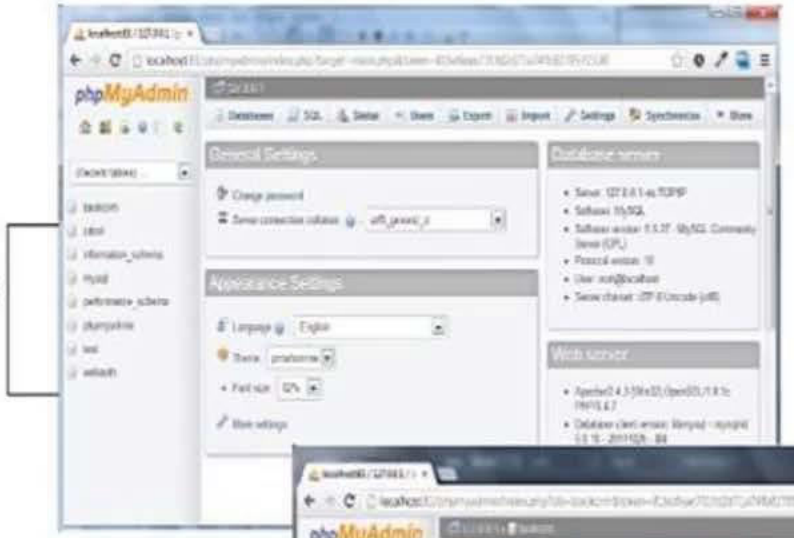
mysql>
```

Command-Line Interface

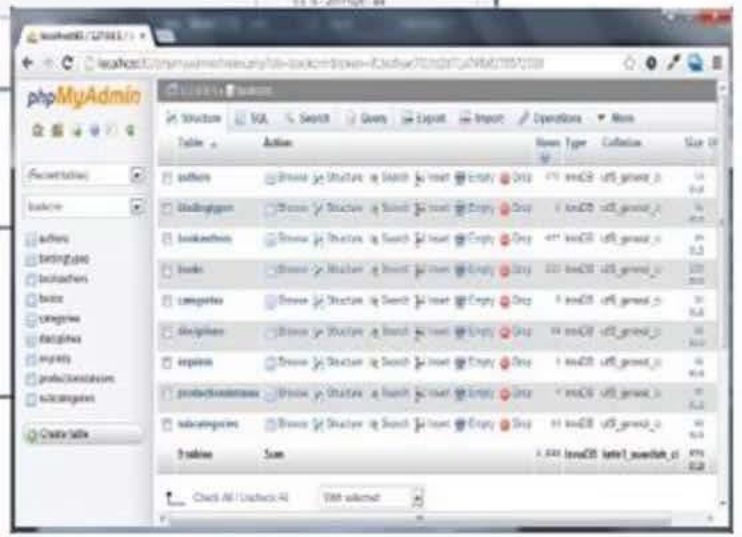
- To launch an interactive MySQL command-line session, you must specify the host, username, and database name to connect to as shown below:
- **mysql -h 192.168.1.14 -u bookUser –p**
- To import commands from a file called commands.sql , for example, we would use the < operation:
- **mysql –h 192.168.1.14 –u bookUser –p < commands.sql**

phpMyAdmin

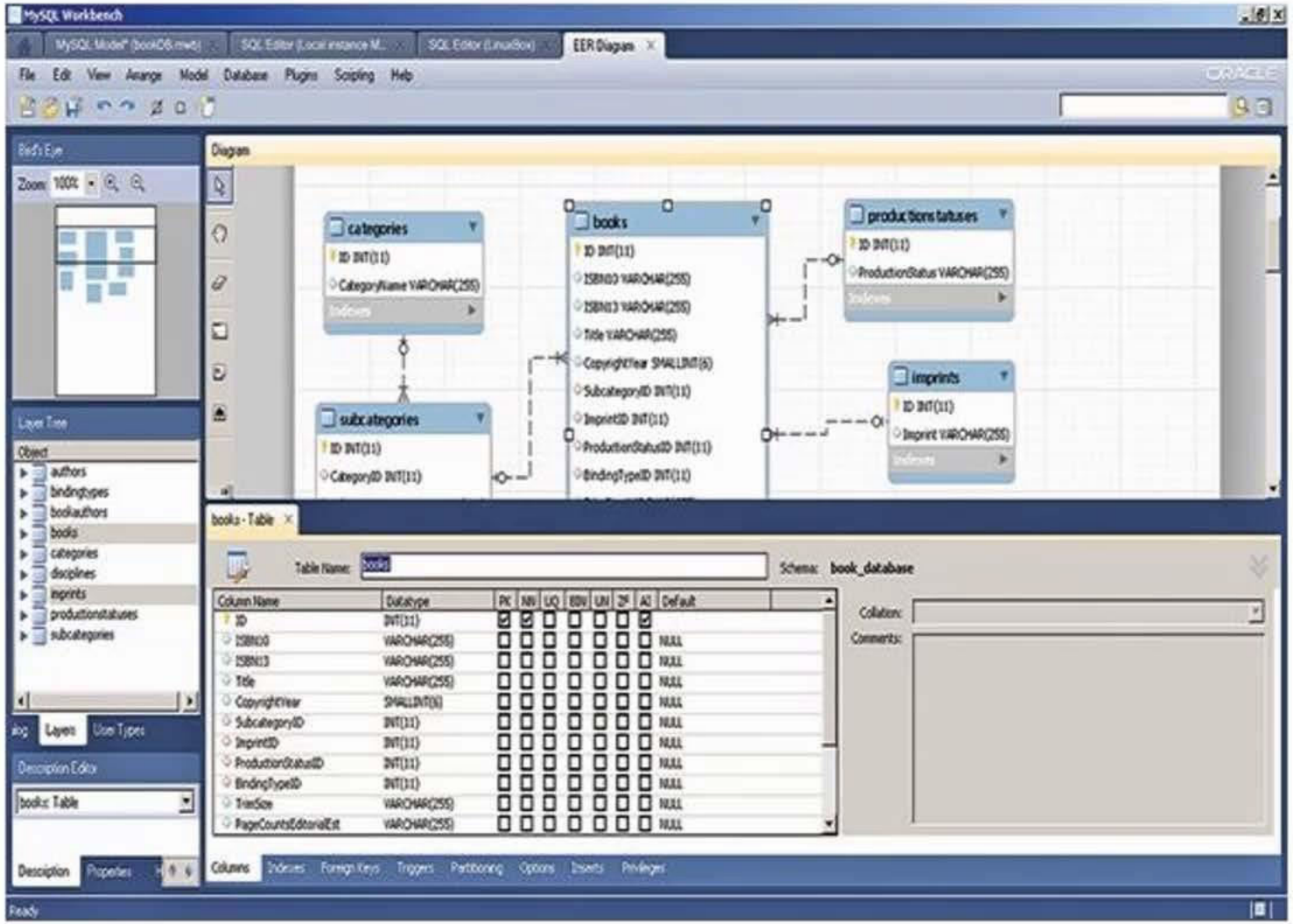
MySQL has a number of predefined databases it uses for its own operation.



phpMyAdmin allows you to view and manipulate any table in a database.



MySQL Workbench



Using phpMyAdmin

TOPIC 93

Web Systems and Technologies

Accessing MySQL in PHP

TOPIC 94

Web Systems and Technologies

PHP MySQL APIs

- **MySQL extension**. This was the original extension to PHP for working with MySQL and has been replaced with the newer mysqli extension
- **mysqli extension**. This extension provides both a procedural and an object-oriented approach. This extension also supports most of the latest features of MySQL
- **PHP data objects (PDOs)**. provides an abstraction layer that with the appropriate drivers can be used with any database, and not just MySQL databases.However, it is not able to make use of all the latest features of MySQL

Deciding on a Database API

- While PDO is unable to take advantage of some features of MySQL, there is a lot of merit to the fact that PDO can create database-independent PHP code
- Like many things in the web world, there is no single best choice.
- As the chapter (and book) proceed, we will standardize on the object-oriented, database-independent PDO approach.

Basic Connection Algorithm

1. Connect to the database
2. Handle connection errors
3. Execute the SQL query
4. Process the results
5. Free resources and close connection

Basic Connection Algorithm

```php
<?php

try {
    $connString = "mysql:host=localhost;dbname=bookcrm";
    $user = "testuser";
    $pass = "mypassword";

    $pdo = new PDO($connString,$user,$pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "SELECT * FROM Categories ORDER BY CategoryName";
    $result = $pdo->query($sql);

    while ($row = $result->fetch()) {
        echo $row['ID'] . " - " . $row['CategoryName'] . "<br/>";
    }
    $pdo = null;
}
catch (PDOException $e) {
    die( $e->getMessage() );
}

?>
```

1. Connection string, username, password
2. Exception handling
3. Query
4. Loop through results
5. Close connection

Connecting to a Database (mysqli peocedural)

```
// modify these variables for your
installation

$host = "localhost";

$database = "bookcrm";

$user = "testuser";

$pass = "mypassword";

$connection = mysqli_connect($host,
$user, $pass, $database);
```

Connecting to a Database (PDO Object-oriented)

```
// modify these variables for your
installation

$connectionString =
"mysql:host=localhost;dbname=bookcrm";

$user = "testuser";

$pass = "mypassword";

$pdo = new PDO($connectionString,
$user, $pass);
```

Handling Connection Errors - mysqli

```
$connection = mysqli_connect(DBHOST,
DBUSER, DBPASS, DBNAME);

// mysqli_connect_errno returns the
last error code

if ( mysqli_connect_errno() ) {

        die( mysqli_connect_error() );
        // die() is equivalent to exit()

}
```

Handling Connection Errors - PDO

```
try {
        $connString =
"mysql:host=localhost;dbname=bookcrm";
        $user = DBUSER;
        $pass = DBPASS;
        $pdo = new
PDO($connString,$user,$pass);
        . . .
}
catch (PDOException $e) {
        die( $e->getMessage() );
}
```

Web Systems and Technologies

TOPIC 95

Accessing MySQL in PHP
Fetching Data

Accessing MySQL in PHP

Executing the query

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";

// returns a mysqli_result object

$result = mysqli_query($connection, $sql);

OR

$result = $pdo->query($sql);
```

Accessing MySQL in PHP

Processing the Query Results

```
$sql = “SELECT * FROM Categories ORDER BY CategoryName”;
// run the query
$result = $pdo->query($sql);
// fetch a record from result set into an associative array
while ($row = $result->fetch()) {
// the keys match the field names from the table
echo $row['ID'] . “ - ” . $row['CategoryName'];
echo “<br/>”;
}
```

Accessing MySQL in PHP

Processing the Query Results

```
$sql = "select * from Paintings";
$result = $pdo->query($sql);
```

ID	Title	Artist	Year
345	The Death of Marat	David	1793
400	The School of Athens	Raphael	1510
408	Bacchus and Ariadne	Titian	1520
425	Girl with a Pearl Earring	Vermeer	1665
438	Starry Night	Van Gogh	1889

$result
Result set is a type of cursor to the retrieved data

```
$row = $result->fetch()
```

$row
Associative array

ID	Title	Artist	Year	keys
345	Death of Marat	David	1793	values

Accessing MySQL in PHP

Freeing Resources and Closing Connection

```
// close the  connection

mysqli_close($connection);

// closes connection and frees the resources used by the PDO
object

$pdo = null;
```

Accessing MySQL in PHP

Working with Parameters

```
$sql = “UPDATE Categories SET CategoryName='Web' WHERE

CategoryName='Business'”;

$count = $pdo->exec($sql);

echo "<p>Updated ” . $count . “ rows</p>”;
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (?,?,?,?,?,?,?)";
$statement = $pdo->prepare($sql);
$statement->bindValue(1, $_POST['isbn']);
$statement->bindValue(2, $_POST['title']);
$statement->bindValue(3, $_POST['year']);
$statement->bindValue(4, $_POST['imprint']);
$statement->bindValue(5, $_POST['status']);
$statement->bindValue(6, $_POST['size']);
$statement->bindValue(7, $_POST['desc']);
$statement->execute();
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders with Array

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (?,?,?,?,?,?,?)";
$statement = $pdo->prepare($sql);
$statement->execute
(array($_POST['isbn'],
 $_POST['title'], $_POST['year'],
 $_POST['imprint'], $_POST['status'],
 $_POST['size'],$_POST['desc']);
```

Accessing MySQL in PHP

Working with Parameters – Technique 2 ? Named Parameters

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (:isbn,
:title, :year, :imprint, :status, :size, :desc) ";
$statement = $pdo->prepare($sql);
$statement->bindValue(':isbn', $_POST['isbn']);
$statement->bindValue(':title', $_POST['title']);
$statement->bindValue(':year', $_POST['year']);
$statement->bindValue(':imprint', $_POST['imprint']);
$statement->bindValue(':status', $_POST['status']);
$statement->bindValue(':size', $_POST['size']);
$statement->bindValue(':desc', $_POST['desc']);
$statement->execute();
```

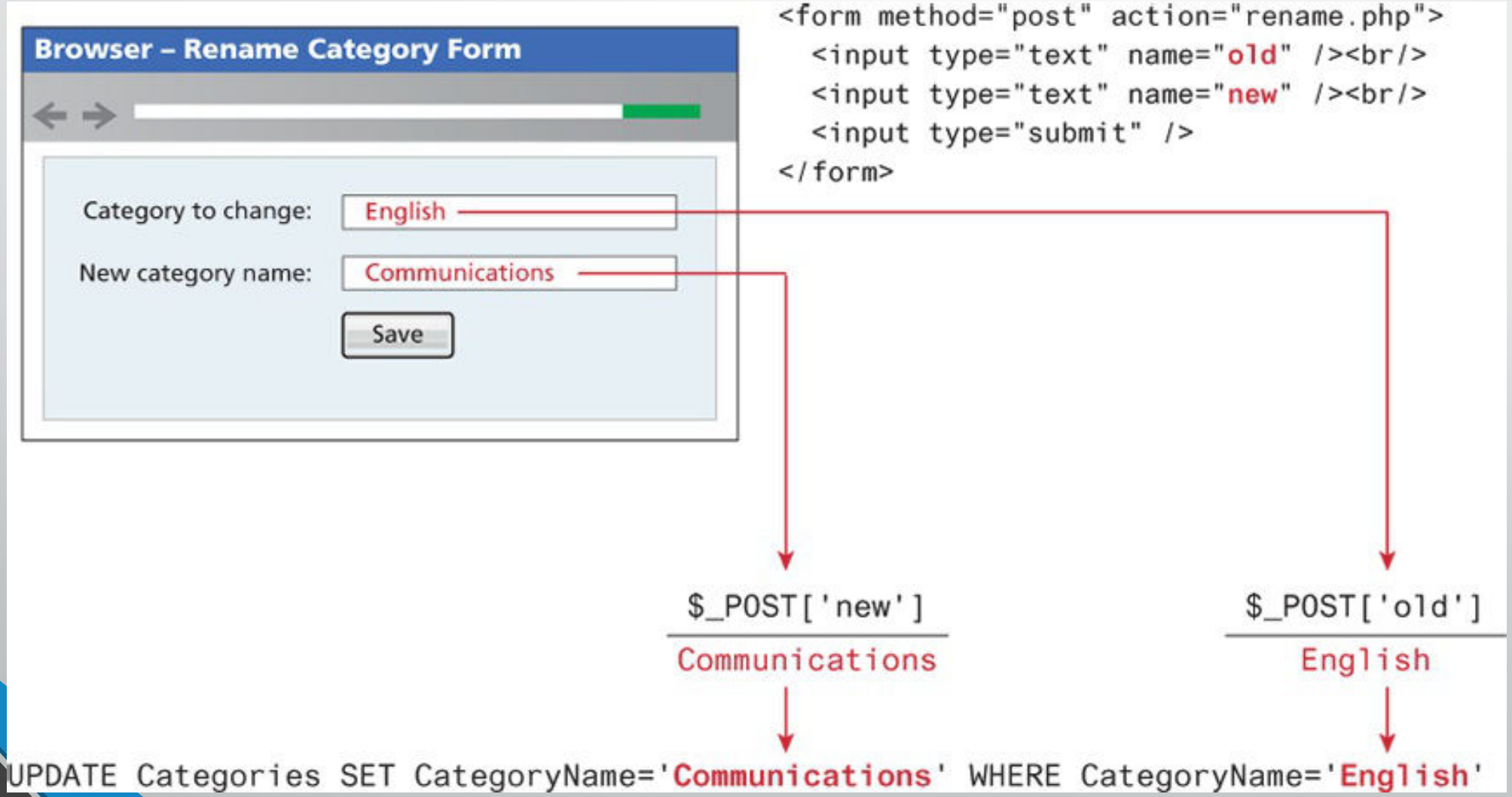
Accessing MySQL in PHP

Working with Parameters – Technique 2 ? Named Parameters with Array

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (:isbn,
:title, :year, :imprint, :status, :size, :desc) ";
$statement = $pdo->prepare($sql);
$statement->execute(array(':isbn' => $_POST['isbn'],
                          ':title'=> $_POST['title'],
                          ':year'=> $_POST['year'],
                          ':imprint'=> $_POST['imprint'],
                          ':status'=> $_POST['status'],
                          ':size'=> $_POST['size'],
                          ':desc'=> $_POST['desc']));
```

Accessing MySQL in PHP

Getting User Input into a Query



Web Systems and Technologies

TOPIC 96

Accessing MySQL in PHP
Performing Transactions

Accessing MySQL in PHP

Using Transactions

```
$pdo = new PDO($connString,$user,$pass);
try {
    // begin a transaction
    $pdo->beginTransaction();
    // a set of queries: if one fails, an exception will be thrown
    $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Philosophy')");
    $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Art')");
    // if we arrive here, it means that no exception was thrown
    $pdo->commit();
} catch (Exception $e) {
    // we must rollback the transaction since an error occurred with insert
    $pdo->rollback();
}
```

Topic 97 to 99 are examples

Web Systems and Technologies

WEEK 12

TOPIC 100 TO 108

Web Systems and Technologies

- 100 Example: Database Update
- 101 Error Handling in PHP; Types of Error
- 102 PHP Error Reporting
- 103 PHP Error and Exception Handling
- 104 Regular Expressions
- 105 Regular Expressions - Examples
- 106 PHP Filters
- 107 Validating User Input
- 108 Where to Perform Validation

TOPIC 100

Example - database

Web Systems and Technologies

TOPIC 101

Types of Errors

Web Systems and Technologies

Copyright © 2021 Virtual University of Pakistan

Types of Errors

- Expected errors
- Warning errors
- Fatal errors

Types of Errors

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**	Notice that a missing value for a parameter is still considered to be `isset`.
`isset($_GET['name2'])`	returns	**true**	
`isset($_GET['name3'])`	returns	**true**	
`isset($_GET['name4'])`	returns	**false**	Notice that only a missing parameter name is considered to be not `isset`.
`empty($_GET['id'])`	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.
`empty($_GET['name1'])`	returns	**true**	
`empty($_GET['name2'])`	returns	**false**	
`empty($_GET['name3'])`	returns	**false**	Notice that a value of space is considered to be **not** empty.
`empty($_GET['name4'])`	returns	**true**	

PHP Error Reporting

TOPIC 102

Web Systems and Technologies

The error_reporting Setting

- error_reporting specifies which type of errors are to be reported
- Can be configured using

  ini_set(...);

- Or within the php.ini file

The display_errors Setting

- The display_error setting specifies whether error messages should or should not be displayed in the browser
- ini_set('display_errors','0');
- It can also be set within the php.ini file:
- display_errors = Off

The log_errors Setting

- The log_errors setting specifies whether error messages should or should not be sent to the server error

- ini_set('log_errors','1');
- It can also be set within the php.ini file:
- log errors = On

PHP Error and Exception Handling

TOPIC 103

Web Systems and Technologies

Exceptions

- 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.

Procedural Error Handling

```
$connection = mysqli_connect(DBHOST,
DBUSER, DBPASS, DBNAME);
$error = mysqli_connect_error();
if ($error != null) {
  // handle the error
  ...
}
```

Object-Oriented Exception Handling

```
function throwException($message = null,$code = null) {
  throw new Exception($message,$code);
}
try {
  $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 {
  ...
}
```



TOPIC 104

Regular Expressions

Web Systems and Technologies

Introduction

- A regular expression is a set of special characters that define a pattern
- Regular expressions are a concise way to eliminate the conditional logic that would be necessary to ensure that input data follows a specific format
- PHP, JavaScript, Java, the .NET environment, and most other modern languages support regular expressions (each slightly different)

Regular Expression Syntax

- A literal is just a character you wish to match in the target
- A metacharacter is a special symbol that acts as a command to the regular expression parser
  - . [ ] ( ) ^ $ | * ? { } +
- Regular Expression Patterns can be combined to form complex expressions

Regular Expression Syntax Patterns

- ^qwerty$ If used at the very start and end of the regular expression, it means that the entire string (and not just a substring) must match the rest of the regular expression contained between the ^ and the $ symbols.
- \t Matches a tab character.
- \n Matches a new-line character.
- . Matches any character other than \n.

Regular Expression Syntax Patterns

- [qwerty] Matches any single character of the set contained within the brackets.
- [^qwerty] Matches any single character not contained within the brackets.
- [a-z] Matches any single character within range of characters.
- \w Matches any word character. Equivalent to [a-zA-Z0-9].
- \W Matches any nonword character.

Regular Expression Syntax Patterns

- \s Matches any white-space character.
- \S Matches any nonwhite-space character.
- \d Matches any digit.
- \D Matches any nondigit.
- * Indicates zero or more matches.
- + Indicates one or more matches.
- ? Indicates zero or one match.

Regular Expression Syntax Patterns

- {n} Indicates exactly n matches.
- {n,} Indicates n or more matches.
- {n, m} Indicates at least n but no more than m matches.
- | Matches any one of the terms separated by the | character. Equivalent to Boolean OR.
- () Groups a subexpression. Grouping can make a regular expression easier to understand.

Web Systems and Technologies

TOPIC 105

Regular Expressions - Examples

Examples

- CNIC
- Email Addresses
- Mobile numbers in Pakistan

PHP Filters

TOPIC 106

Web Systems and Technologies

Introduction

- Filters in PHP validate input against predefined patterns
  - Validate filters
  - Sanitize filters
- Use the function
  - filter_var($value, FILTER)

Validate Filters

Validate Filters

- Check input against a pre-defined pattern
- Validate email (FILTER_VALIDATE_EMAIL)

```
$email = filter_var('abc@mail.com',FILTER_VALIDATE_EMAIL);
//$email = abc@mail.com
$email = filter_var('abc@.com',FILTER_VALIDATE_EMAIL);
//$email = null
```

- Other validate filteres
  - FILTER_VALIDATE_BOOL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_INT
  - FILTER_VALIDATE_MAC
  - FILTER_VALIDATE_IP
  - FILTER_VALIDATE_URL
  - FILTER_VALIDATE_REGEXP
  - ...

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

```
$cnic = filter_var('12345-1234567-8', FILTER_VALIDATE_REGEXP,
        array("options" => array("regexp" => "/^\\d{5}-?\\d{7}-?\\d$/")));
if($cnic == null) {
  echo "Invalid CNIC";
} else {
  echo "Valid CNIC $cnic";
}
```

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

```
$cnic = filter_var('12345-1234567-8', FILTER_VALIDATE_REGEXP,
        array("options" => array("regexp" => "/^\\d{5}-?\\d{7}-?\\d$/")));
if($cnic == null) {
  echo "Invalid CNIC";
} else {
  echo "Valid CNIC $cnic";
}
```

Sanitize Filters

- Remove or replace specific characters
- FILTER SANITIZE EMAIL
- FILTER SANITIZE STRING
- FILTER SANITIZE FULL SPECIAL CHARS

Sanitize Filters

- Remove or replace specific characters
- FILTER_SANITIZE_EMAIL
  - Removes all characters except letters, digits and !#$%&'*+-=?^_`{|}~@.[].
  - filter_var('my(email)@mail/.com/', SANITIZE) == 'myemail@mail.com'
- FILTER_SANITIZE_STRING
  - Strips HTML tags
  - filter_var('<h1>A" \' B</h1>',FILTER_SANITIZE_FULL_SPECIAL_CHARS) == ' A" ' B '
- FILTER_SANITIZE_FULL_SPECIAL_CHARS
  - filter_var('<h1>A" \' B</h1>',FILTER_SANITIZE_FULL_SPECIAL_CHARS) == '<h1>A" ' B</h1>'

Sanitize Filters

- FILTER_SANITIZE_ADD_SLASHES
- FILTER_SANITIZE_ ENCODED
- FILTER_SANITIZE_ NUMBER_FLOAT
- FILTER_SANITIZE_ NUMBER_INT
- FILTER_SANITIZE_ SPECIAL_CHARS
- FILTER_SANITIZE_ URL

Validating User Input

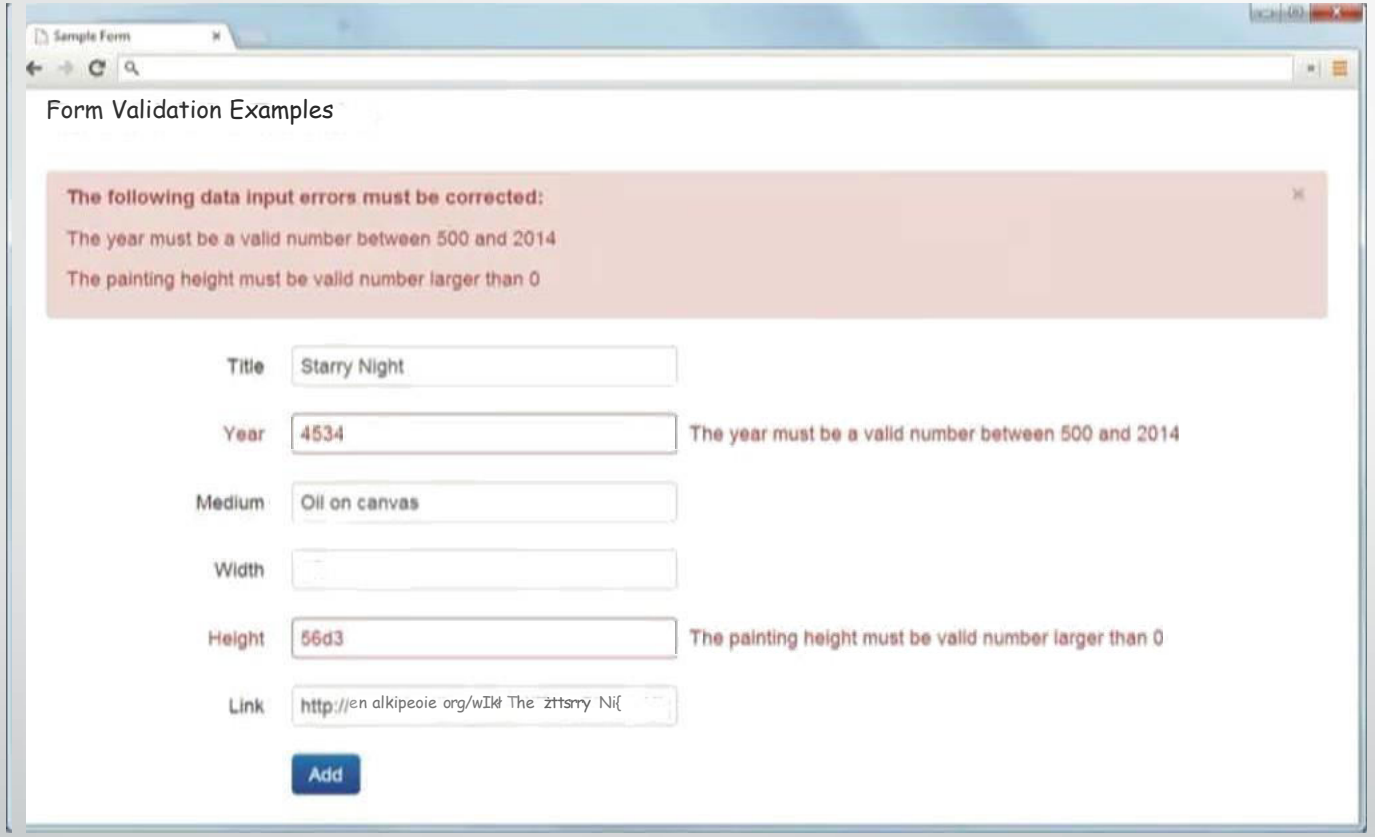
TOPIC 107

Web Systems and Technologies

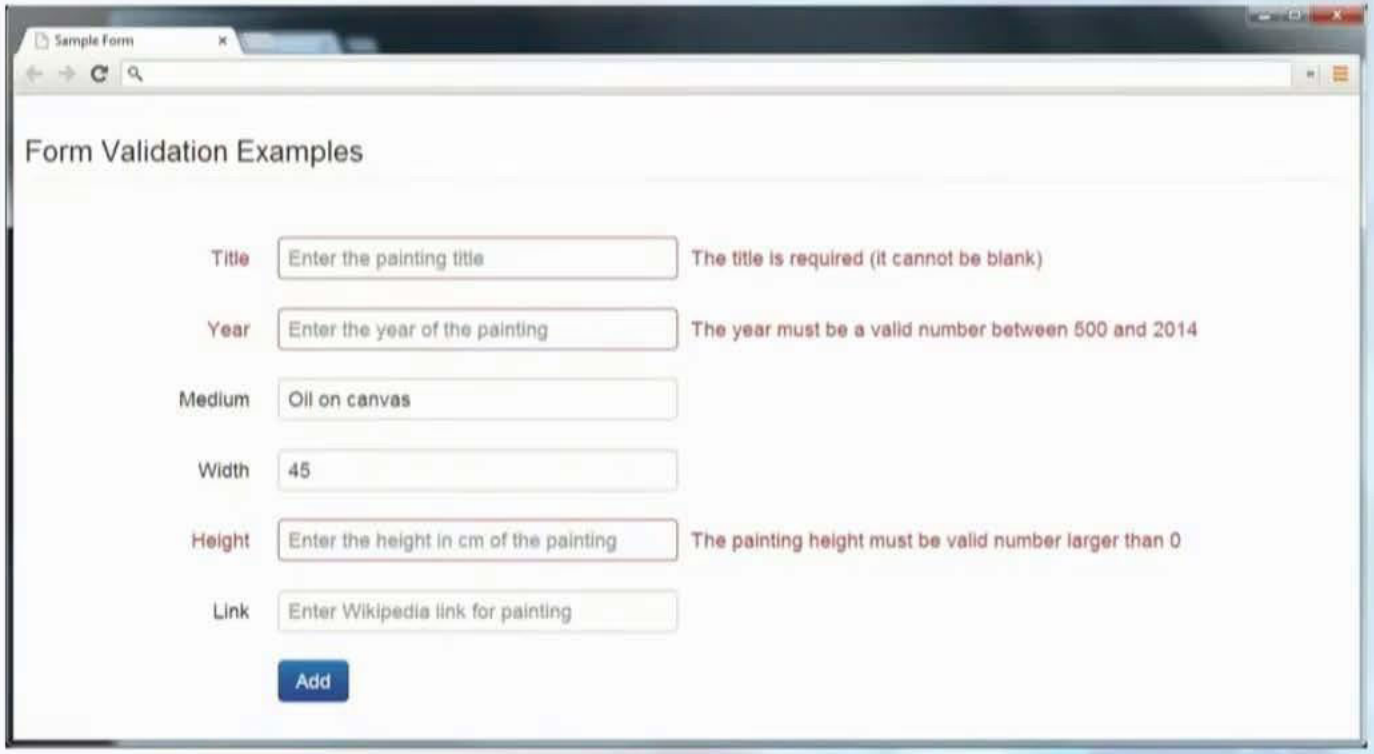
Types of Input Validation

- Required information
- Correct data type
- Correct format
- Comparison
- Range check
- Custom

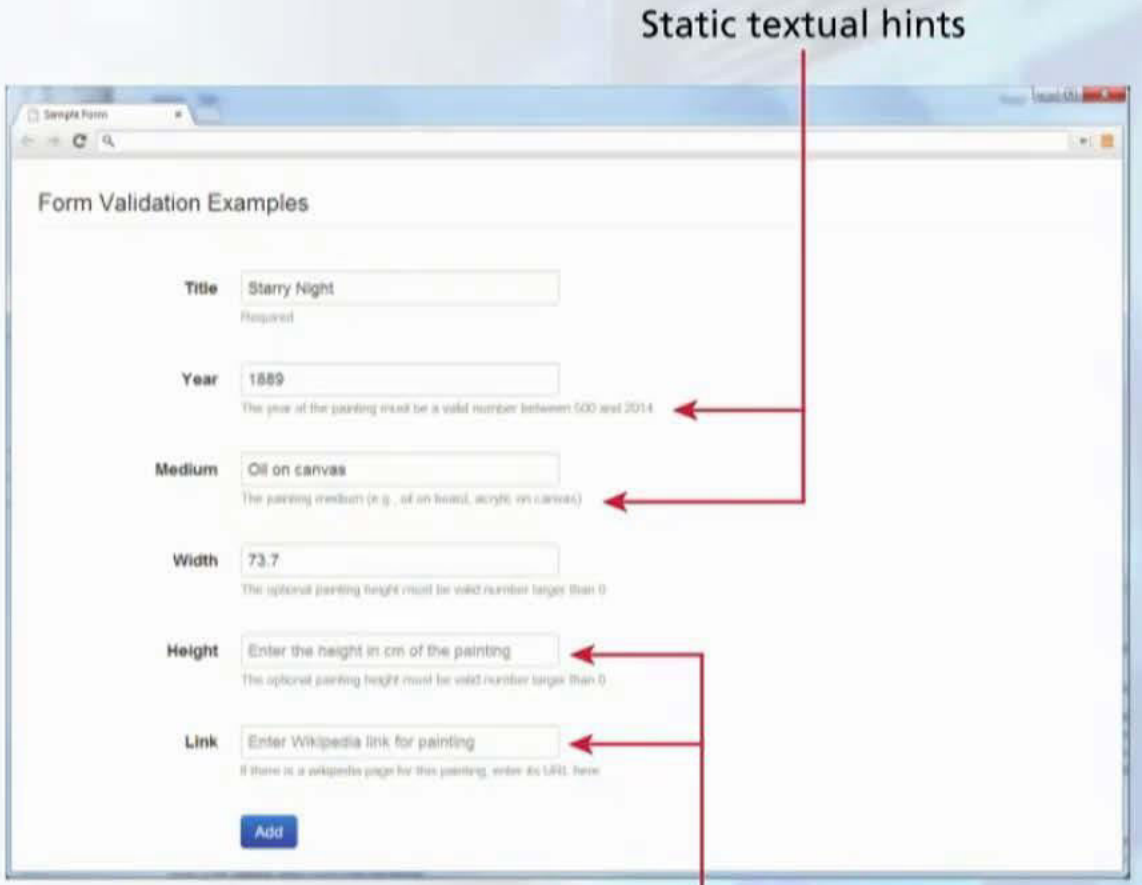
Notifying the user



How to Reduce Validation Errors – Show Where Error Located



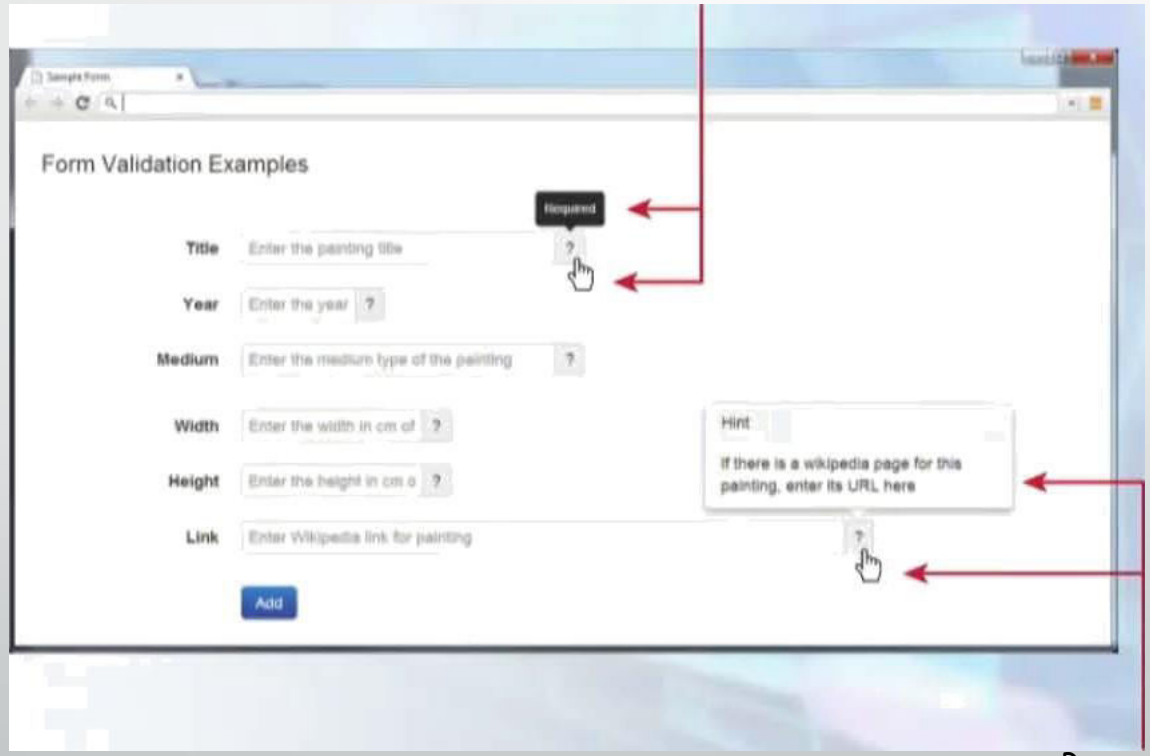
How to Reduce Validation Errors – Providing Textual Hints




```
<input type="text" … placeholder="Enter the height ...">
```

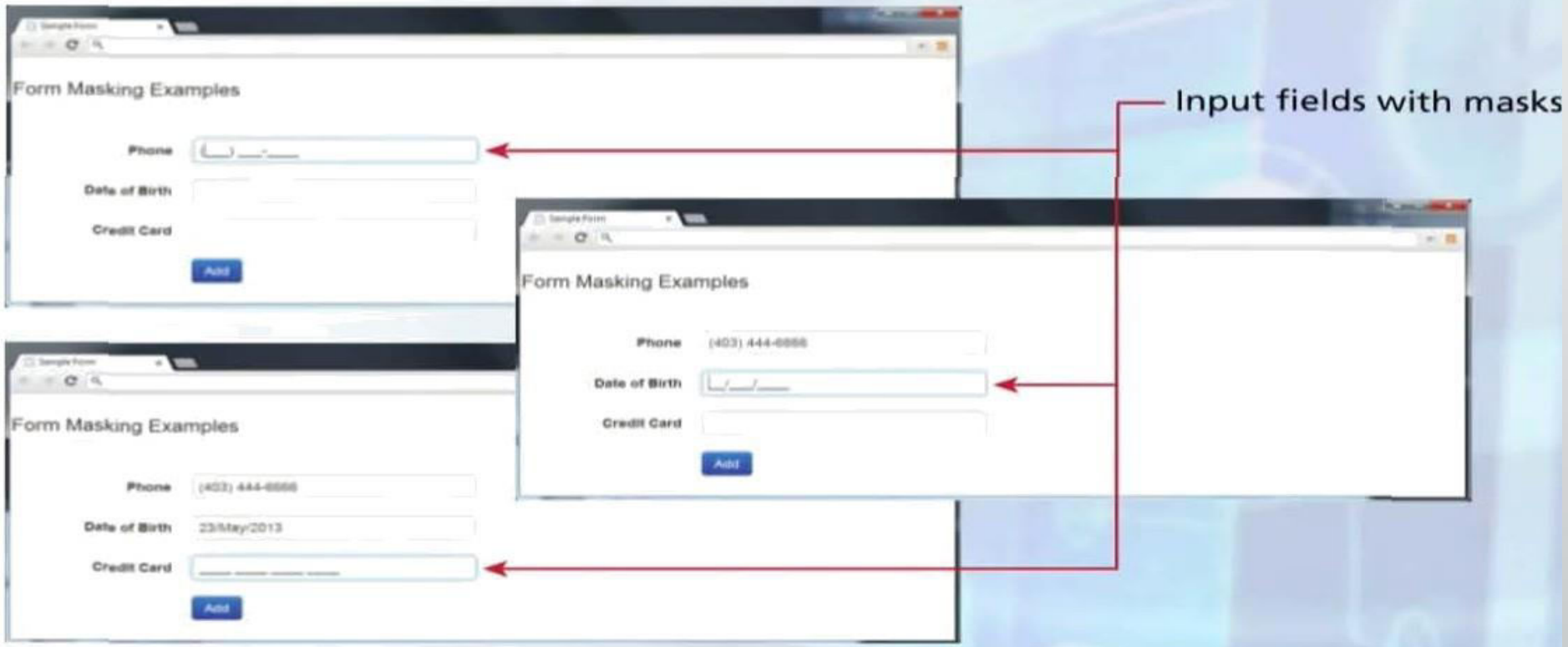
HOW TO REDUCE VALIDATION ERRORS- USE TOOLS TIPS

Pop-up tool tip (appears when mouse hovered over Icon)



Pop-over

How to Reduce Validation Errors Use Input Masks

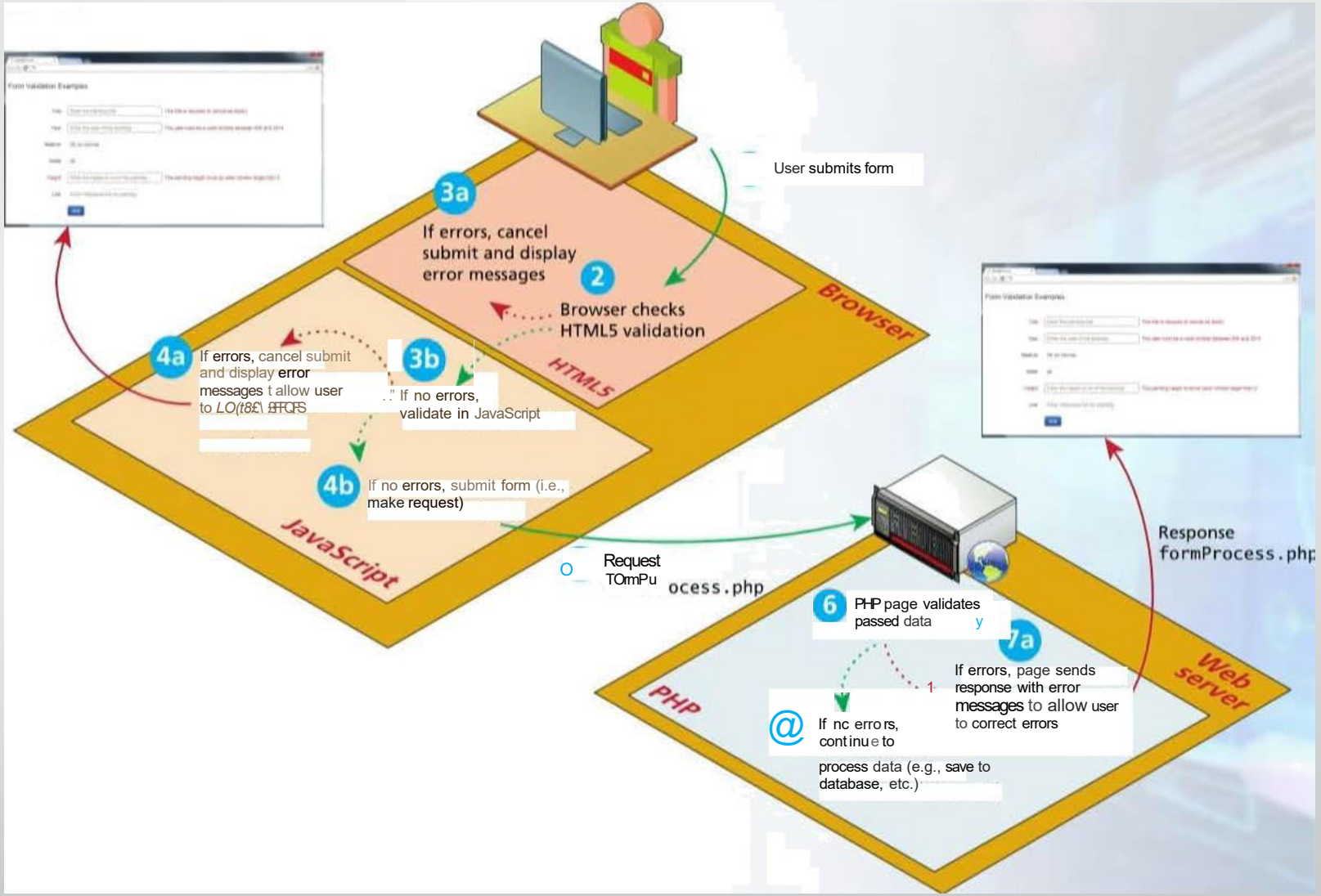


Where to Perform Validation?

TOPIC 108

Web Systems and Technologies

Bigger Picture



Validation at the JavaScript Level

- Client process
- Can reduce server load
- Can be bypassed

Validation at the PHP Level

- Validation on the server side using PHP is the most important form of validation and the only one that is absolutely essential

Web Systems and Technologies

WEEK 13

TOPIC 109 TO 117

Web Systems and Technologies

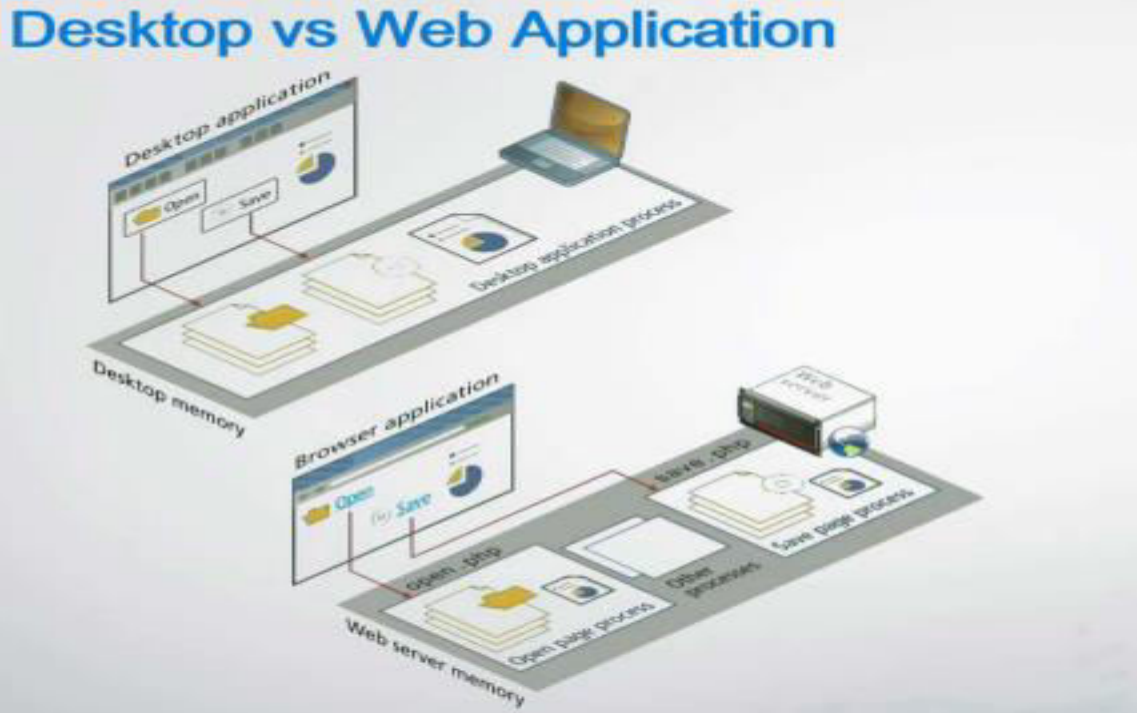
- 109 State Management on Server Side
- 110 State Management Using Query String
- 111 Cookies
- 112 Cookies - Example
- 113 Serialization
- 114 Session State
- 115 Security Principles
- 116 Security by Design
- 117 Authentication

TOPIC 109

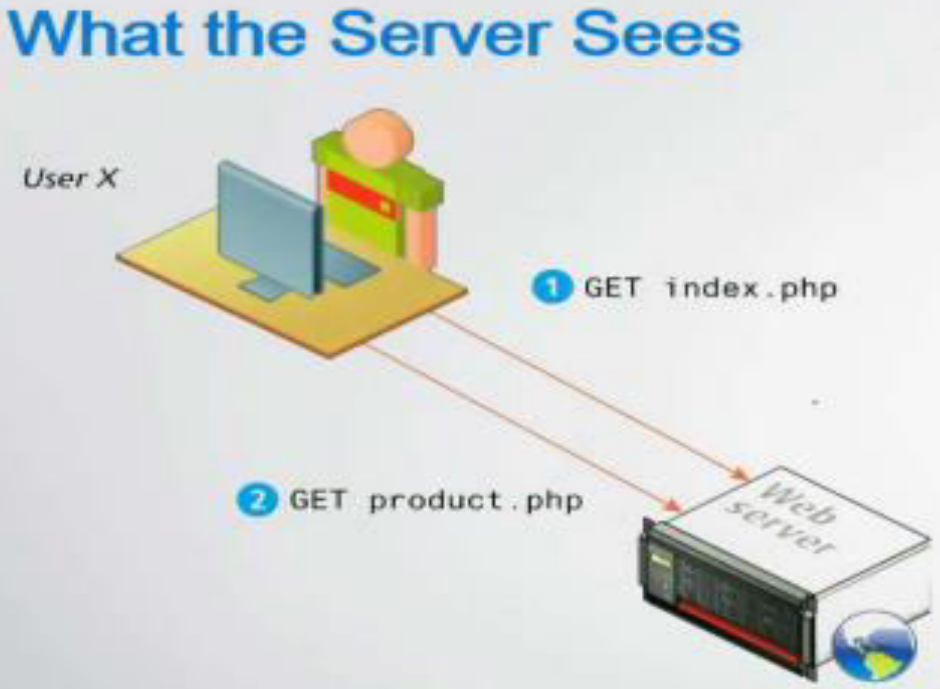
The Problem of state Management on server side

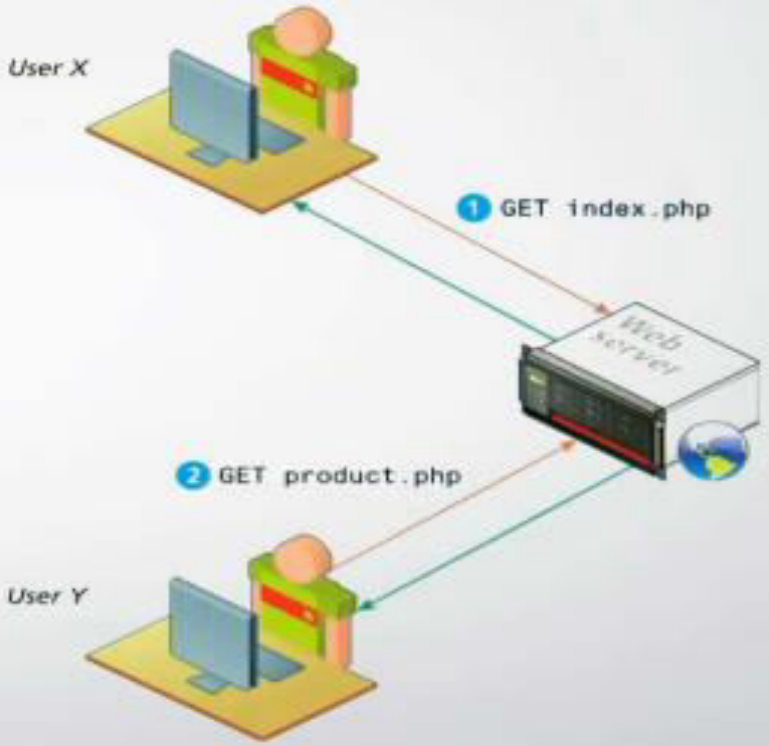
Web Systems and Technologies

Desktop vs Web Application

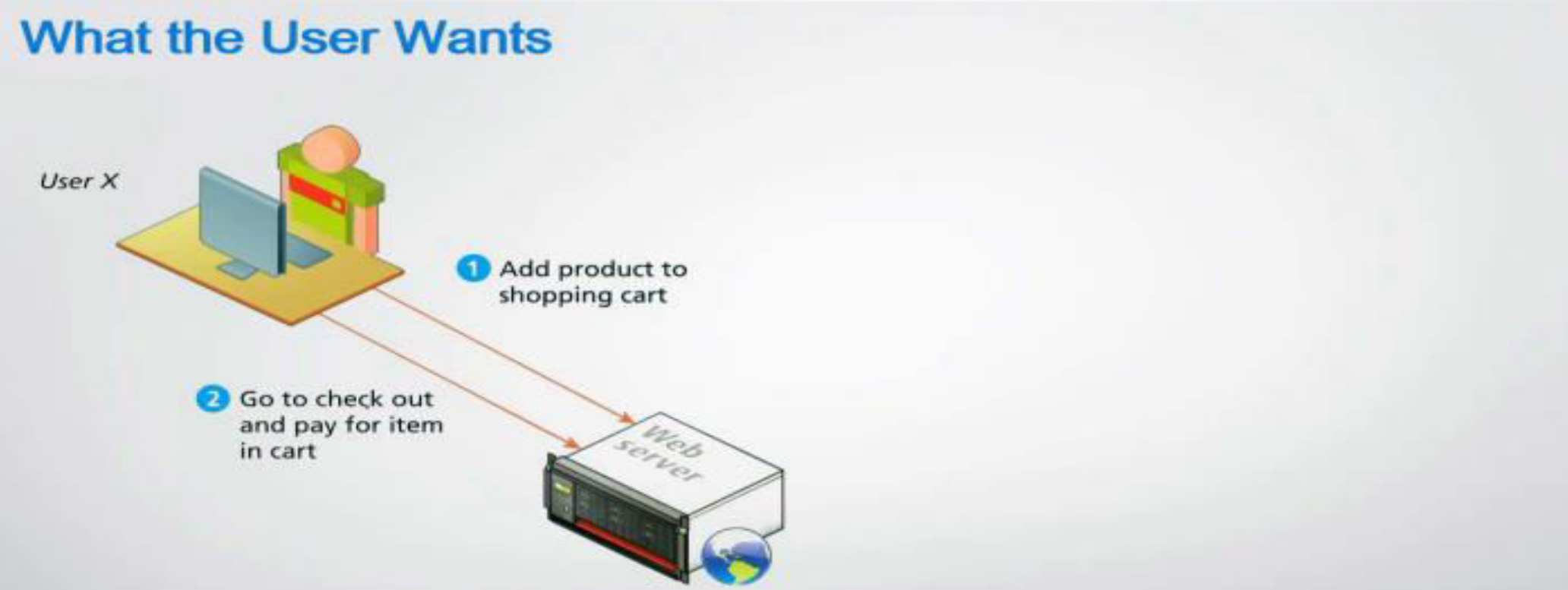


What the Sewer Sees





What the User Wants



Solution

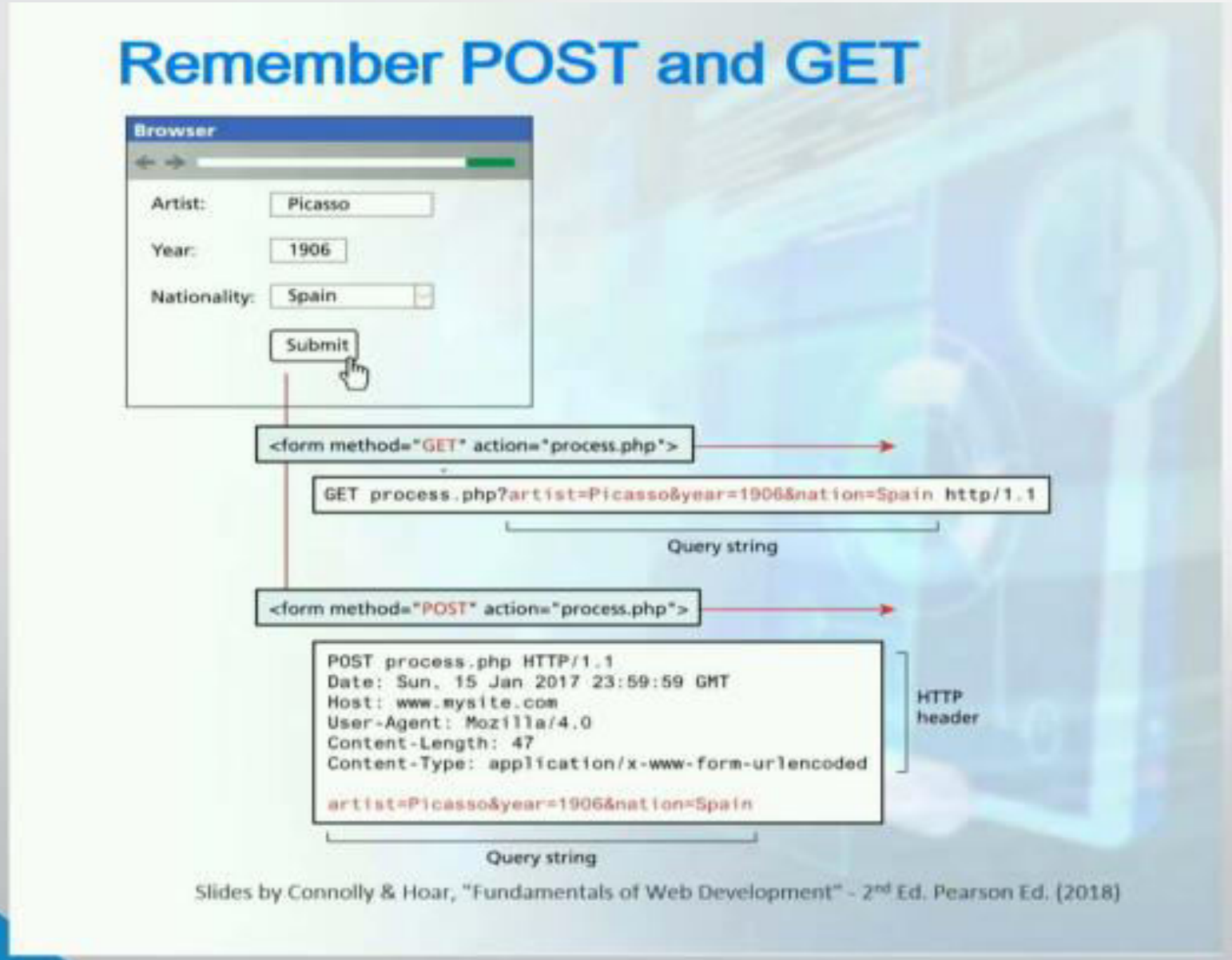
- Query Strings
- Cookies
- Sessions

Passing Information via Query Strings

TOPIC 110

Web Systems and Technologies

Remember POST and GET



Dynamic vs S atic URLs

- Dynamic URLs (i.e., query strings)
  - A pretty essential part of web application development
- Static URLs
  - The appearance of search terms within the URL does seem to improve its relative pösition.
  - Another benefit to static URLs is that users tend to prefer them.
- Can use URL rewriting to make static URLs be dynamic

Passing Information via Query Strings

URL Rewriting in Apache

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



http://www.paintingswholesaler.com/detail.asp?vcode=6umd7krr1yqi161c&title=La+Donna+Velata



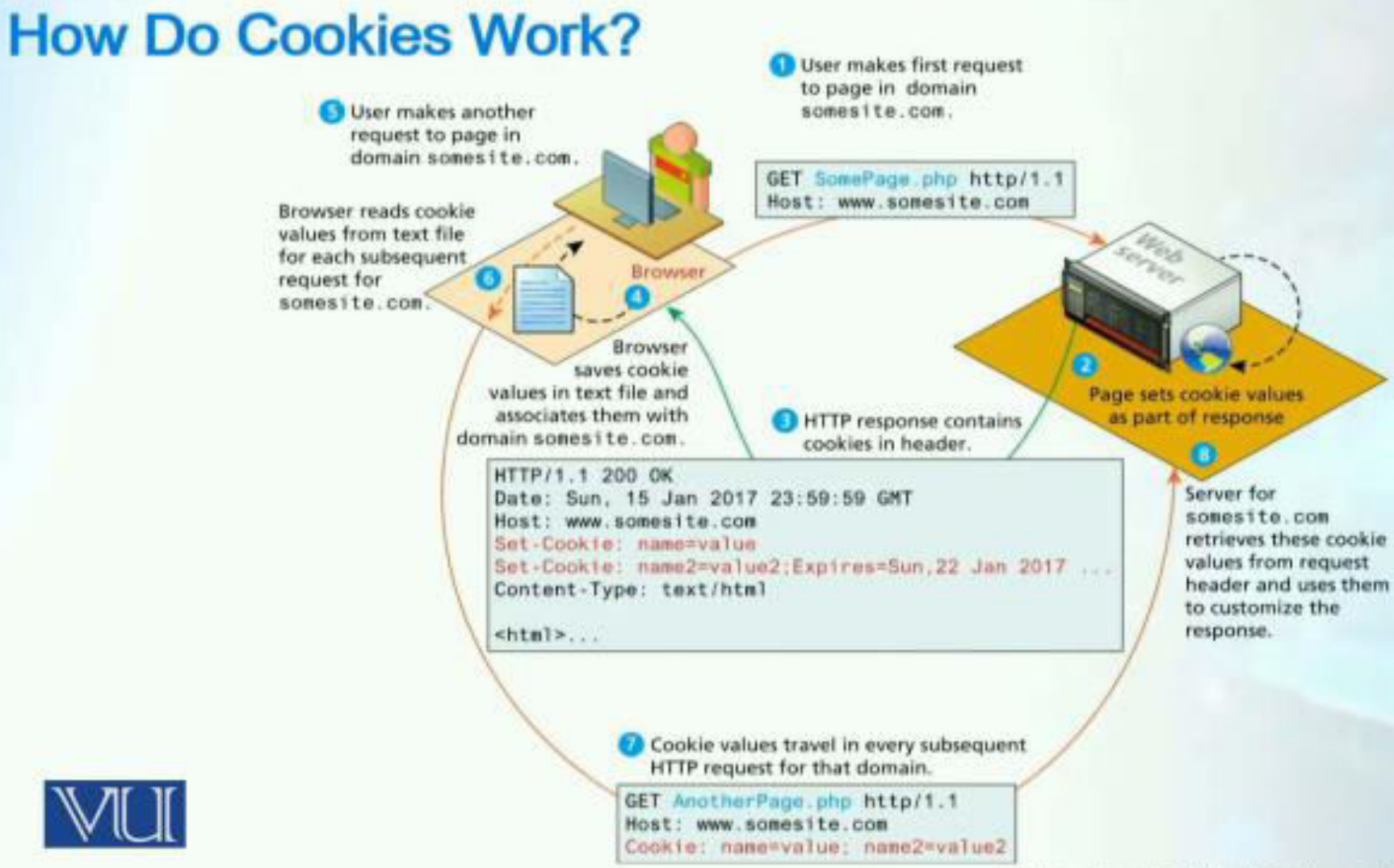
Cookies

TOPIC 111

Web Systems and Technologies

Cookies

How Do Cookies Work?



Persistent Cookie Best Practices

- Use cookies for improving user experience
  - User preferences
  - User behavior
- The website functionality should not depend upon cookies
- If using cookies for login information, back it up with a token on each login, saved on the server as well

Setting a Cookie

```
<?php
// add 1 day to the current time
expiry time
$expiryiime   time()+60*60*24;
   create a persistent cookie
$name    "Username";
$value = "Ricardo";
setcookie($name, $value, $expiryTime);
?>
```

Reading a Cookies

```
<?php
If (!isset($ COOKIE['Username'] valid cookie found


else
  echo ”The username retrieved from the
cookie is:".$ COOKIE['Username' ;

}
?>
```

Cookies – Example

TOPIC 112

Web Systems and Technologies

TOPIC 113

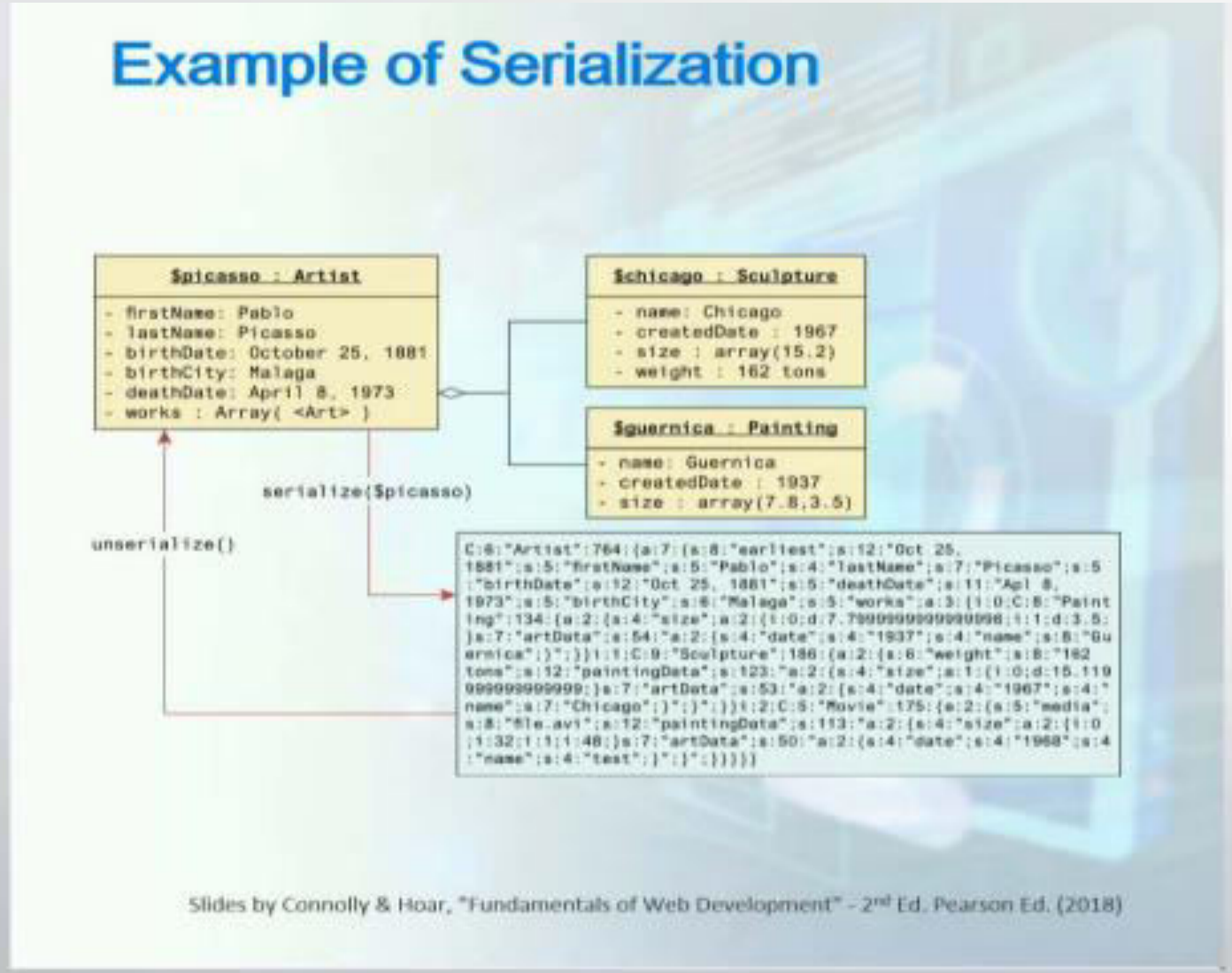
Serialization

Web Systems and Technologies

Introduction

- Serialization converts an object into binary form
- Used for storage and transmission
- Unserialization converts the serialized data back into an object
- Can be used to manage states
  - Serialize data at the end of the request
  - Unserialize at the start of the next request

Example of Serialization



Serialization in PHP

- PHP built –in data types and object are serializable
- Custom objects need to implement the serializable interface

Example

Example

```
class Artist implements Serializable {
  public function serialize() {
  return serialize(
  array("earliest" =>self::$earliestDate,
            "first" => $this->firstName,
            "last" => $this->lastName,
            "bdate" => $this->birthDate,
            "ddate" => $this->deathDate,
            "bcity" => $this->birthCity,
            "works" => $this->artworks
      )
    );
}
    ...
```

```
class Artist implements Serializable {
...
public function unserialize($data) {
  $data = unserialize($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'];
}
...
}
//for serialization
$serialized = serialize($object);

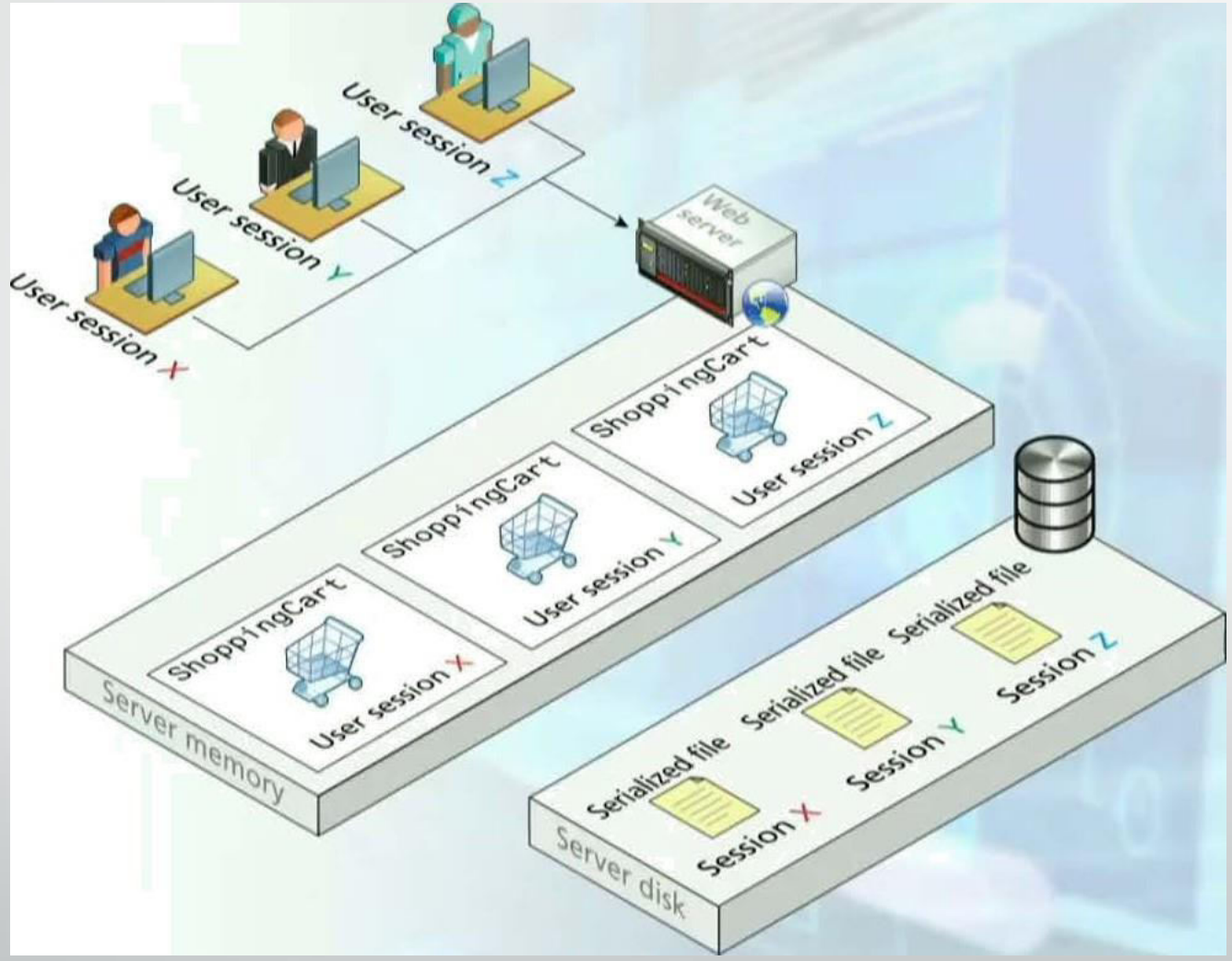
//for unserialization
$objectClone = unserialize($serialized);
```

Session State

TOPIC 114

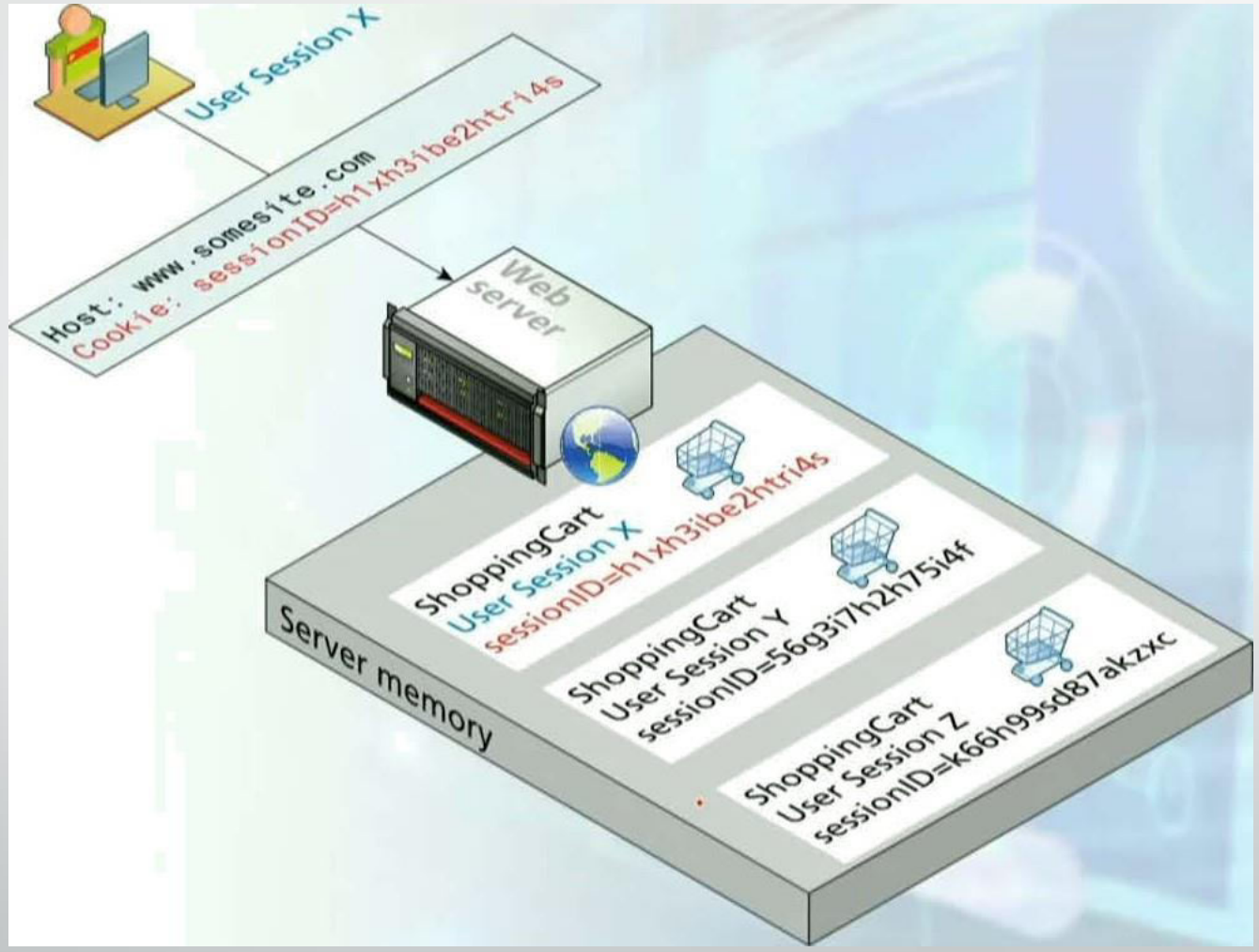
Web Systems and Technologies

How Does Session state work?

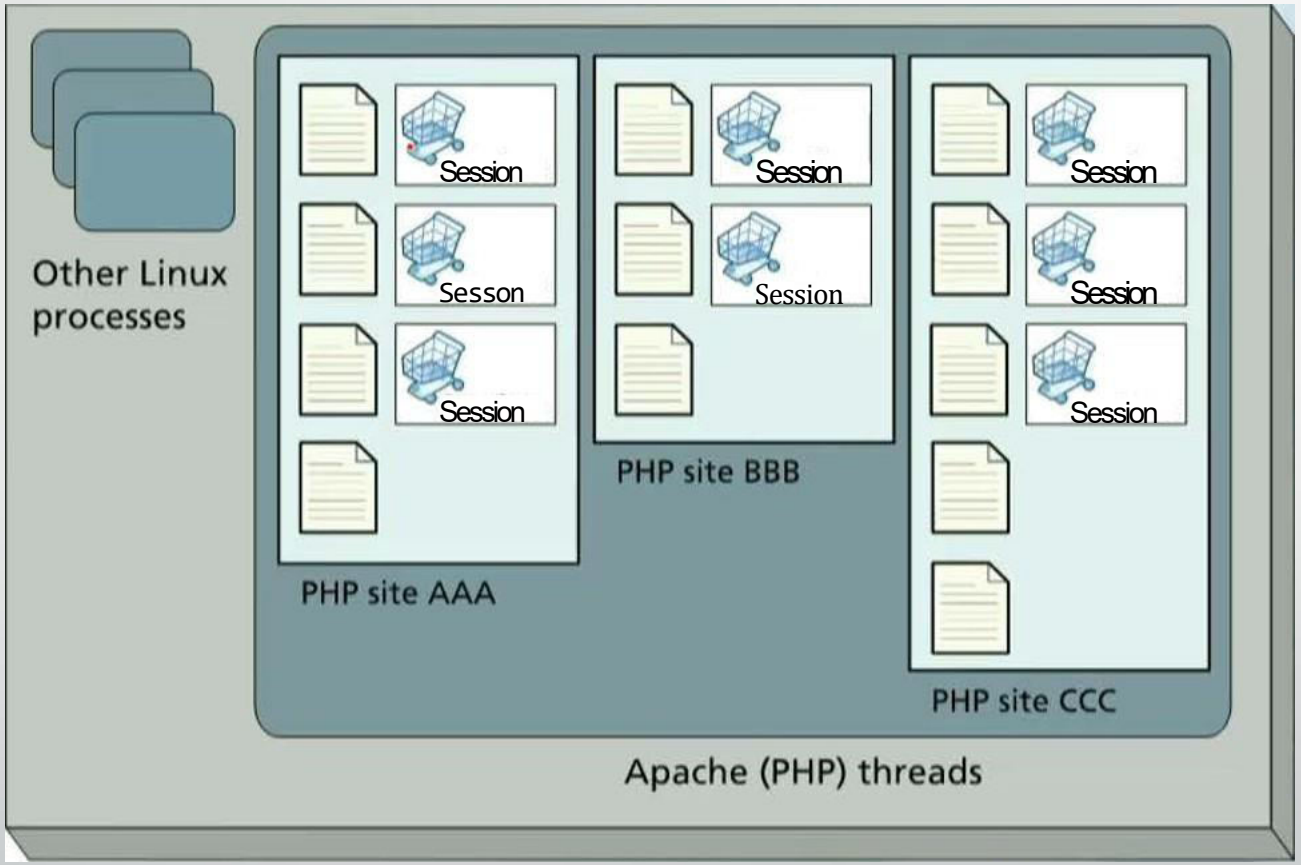


SlfTes by Com›ully & I-Huar, "Fundamentals of Web Development" - 7 * Ed. Pearson Ed, (sofa)

How does it all work



Session Storage and configuaration



Server memory

Accessing session state

Accessing session state

```
<?php
session_start();
if ( isset($_SESSION['user']) ) {
  // User is logged in
}
else {
  // No one is logged in (guest)
}
?>
```

Checking session existence

Checking session existence

```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"];
?>
```

Terminating a session

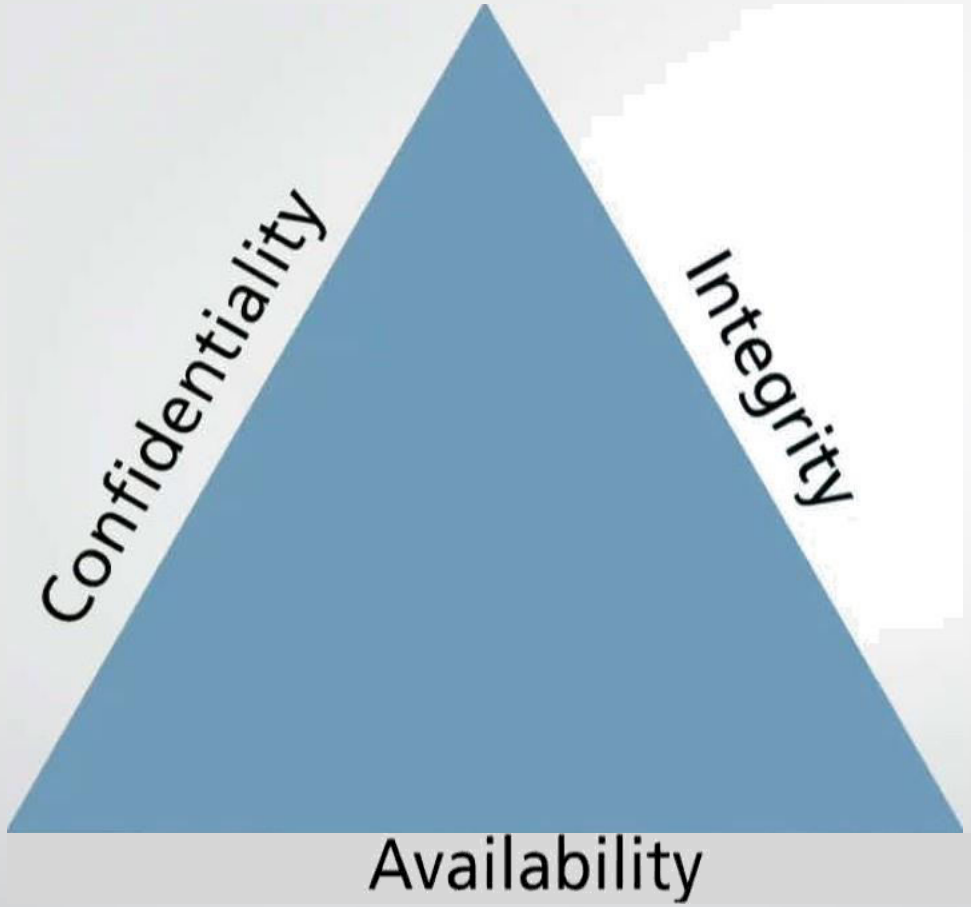
- Removing session variables
  - Session_unset()
- Destroying the session
  - session_destroy();

Security Principles

TOPIC 115

Web Systems and Technologies

Information security



Risk Assessment and Management

- Actors, Impact, Threats, and Vulnerabilities

Actors

- Internal actors are the people who work for the organization. They can be anywhere in the organization from the cashier through the IT staff, all the way to the CEO.
- External actors are the people outside of the organization. They have a wide range of intent and skill, and they are the most common source of attacks.
- Partner actors are affiliated with an organization that you partner or work with. If your partner is somehow compromised, there is a chance your data is at risk as well because quite often partners are granted some access to each other's systems (to place orders, for example).

Impact

- A loss of availability prevents users from accessing some or all of the systems.
- A loss of confidentiality includes the disclosure of confidential information to a (often malicious) third party
- A loss of integrity changes your data or prevents you from having correct data. This might manifest as an attacker hijacking a user session, perhaps placing fake orders or changing a user's home address.

Security Principles

Threats

Broadly, threats can be categorized using the STRIDE mnemonic

- **Spoofing**—Theattacker uses someone else's information to access the system.
- **Tampering**—The attacker modifies some data in nonauthorized ways.
- **Repudiation**—The attacker removes all trace of their attack, so that they cannot be held accountable for other damages done.
- **Information disclosure**—The attacker accesses data they should not be able to.
- **Denial of service** _The attacker prevents real users from accessing the systems.
- **Elevation of privilege** _ The attacker increases their privileges on the system thereby getting access to things they are not authorized to do.

Vulnerabilities

- Vulnerabilities are the security holes in your system. The top five classes of vulnerability from the Open Web Application Security Project are:
    1. Injection
    2. Broken authentication and session management
    3. Cross-site scripting
    4. Insecure deserialization
    5. Security misconfiguration

Security Policy

- Usage policy defines what systems users are permitted to use, and under what situations.
- Authentication policy controls how users are grantecl access to the systems.
- Legal policies define a wide range of things including data retention and backup policies as well as accessibility requirements (like having all public communication well organized for the blind).

Business Continuity

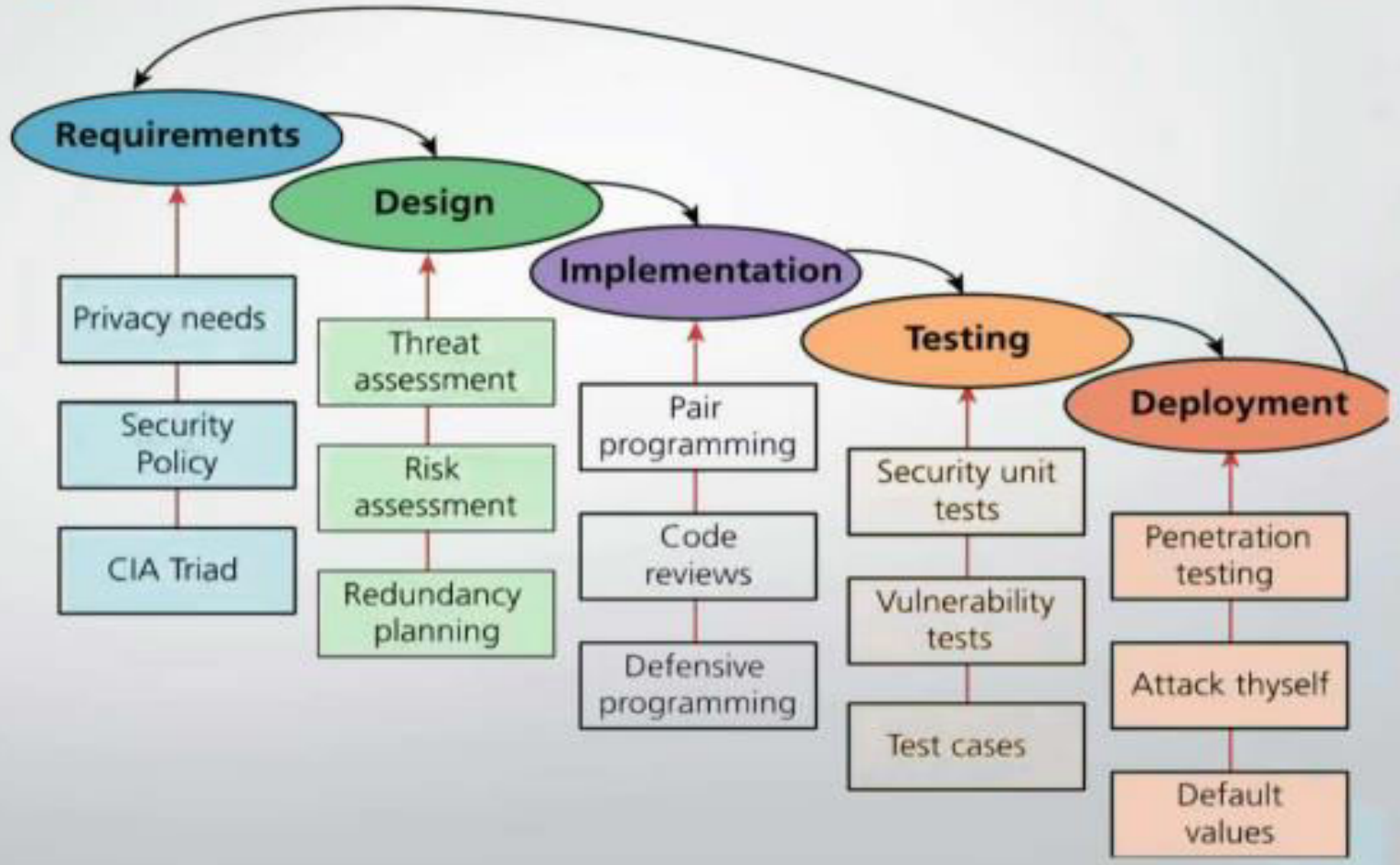
- Admin Password Management
- Backups and Redundancy
- Geographic Redundancy
- Stage Mock Events
- Auditing

TOPIC 116

Security by Design

Web Systems and Technologies

Secure by Design



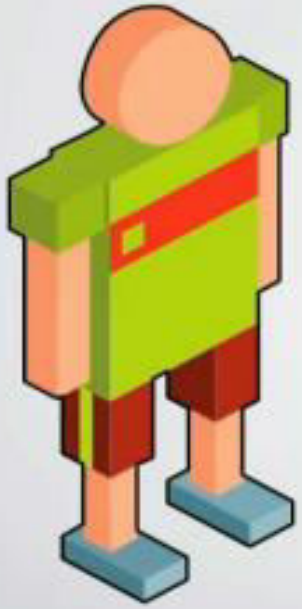
TOPIC 117

Authentication

Web Systems and Technologies

Authentication Factors

Authentication Factors



What you know (Knowledge)

Passwords, PIN, security questions, ...



What you have (Ownership)

Access card, cell phone, cryptographic FOB, ...



What you are (Inherence)

Retinas, fingerprints, DNA, walking gait, ...

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Authentication Factors

- Single-factor authentication is the weakest and most common category of authentication system where you ask for only one of the three factors.
- Multifactor authentication is where two distinct factors of authentication must pass before you are granted access.

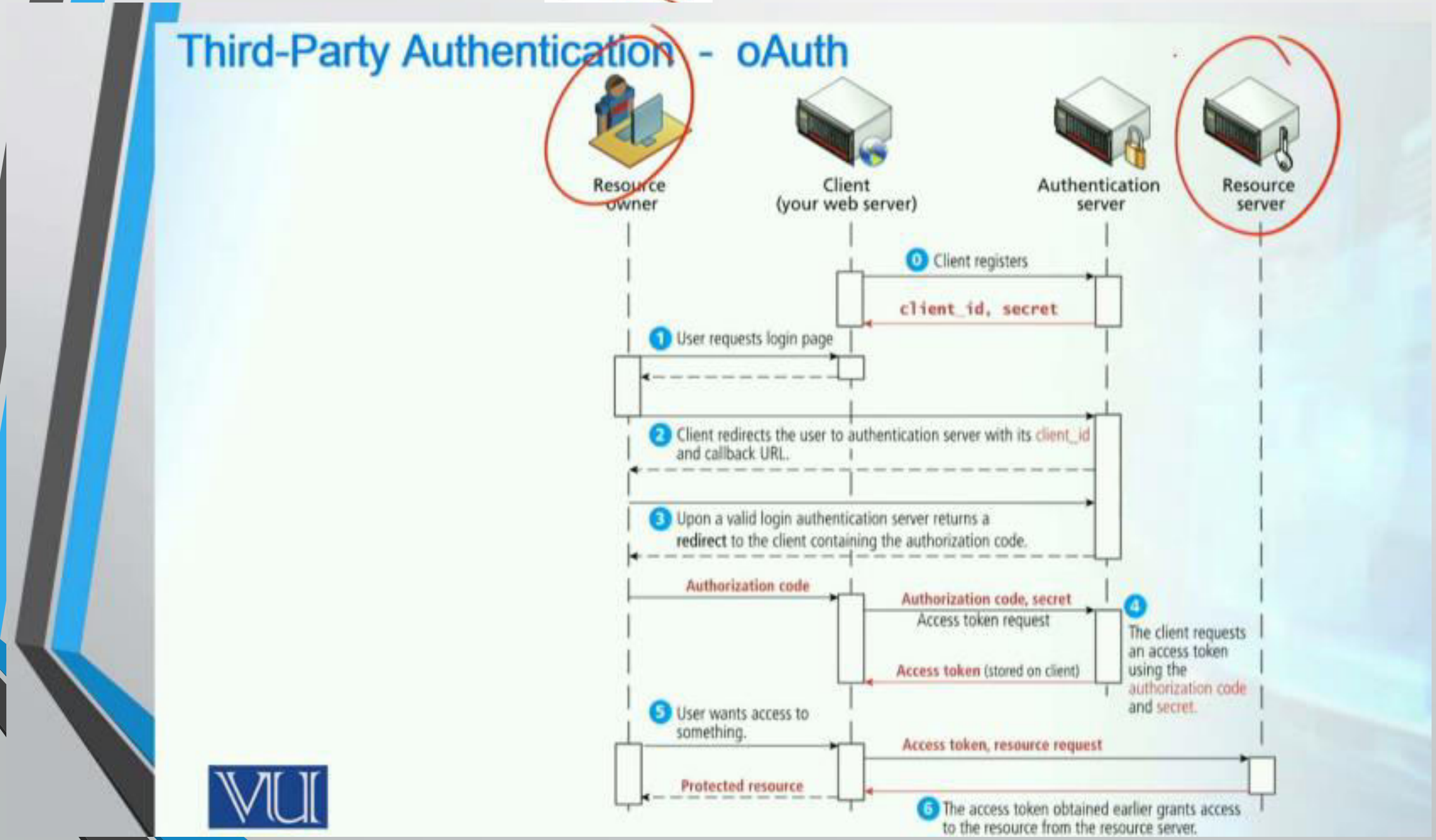
HMP Authentication

- HTTP supports several diXerent forms of authentication via the www- authenticate response header.
- HTTP Basic Authentication
- HTTP Digest Authentication
- Form-Based Authentication

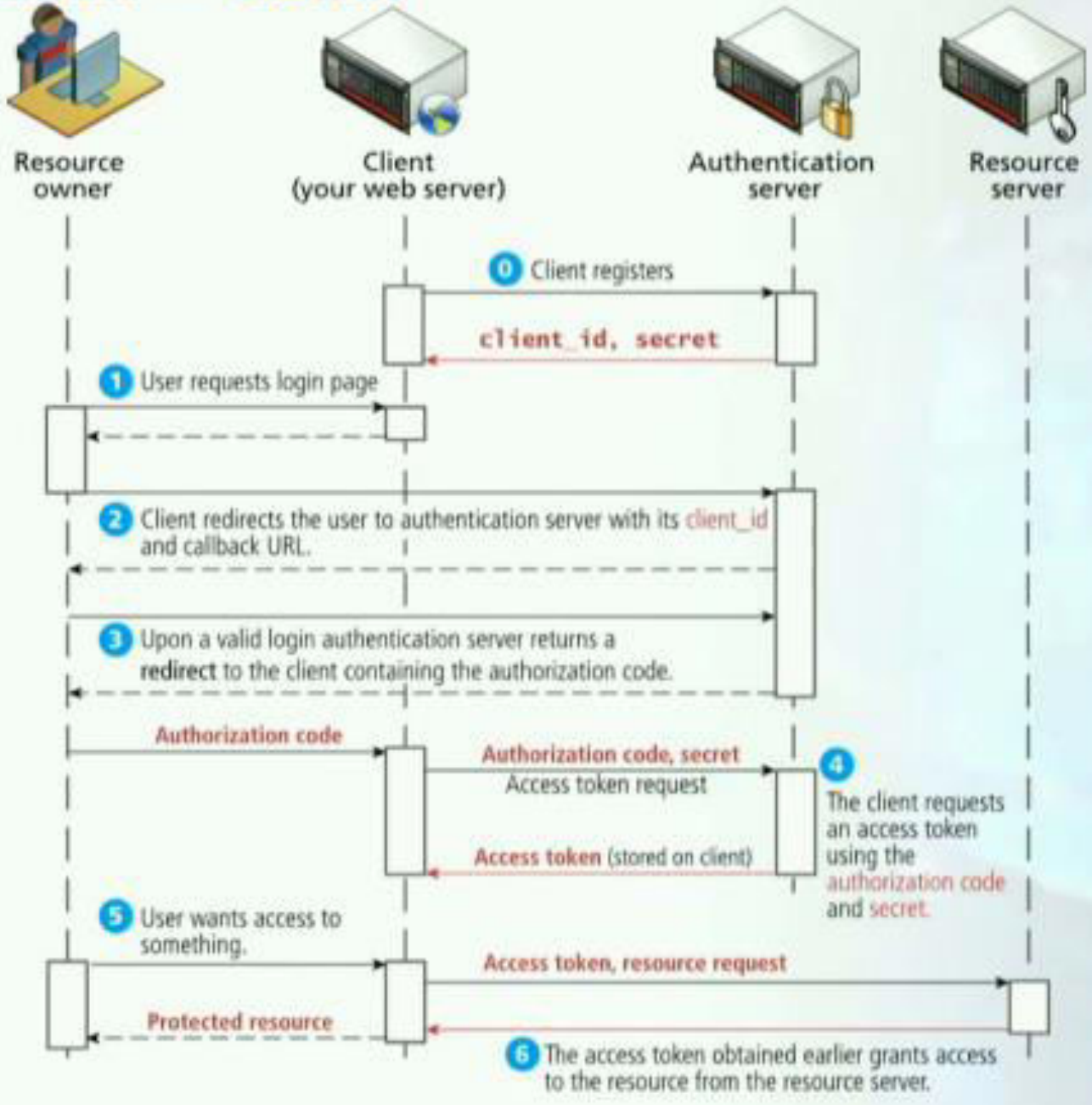
Third-Party Authentication

- Many popular services allow you to use their system to authenticate the user and provide you with enough data to manage your applicatiön

Third-Party Authenticaton - oAuth



Third-Party Authentication - oAuth



Authorization

Authorization defines what rights and privileges a user has once they are authenticated. Some principles:

- Using a separate database user for read and write privileges on a database.
- Providing each user an account where they can access their own files securely.
- Setting permissions correctly so as to not expose files to unauthorized users.
- Using Unix groups to grant users permission to access certain functionality rather than grant users admin access.
- Ensuring Apache is not running as the root account (i.e., the account that can access everythino).

Web Systems and Technologies

WEEK 14

TOPIC 118 TO 126

Web Systems and Technologies

- 118 Cryptography, Cyphers, Digital Signatures
- 119 HTTPS
- 120 XML overview, XML Parser; XML schema
- 121 XSLT and XPath
- 122 XML Processing; XML in JavaScript; XML in PHP
- 123 JSON
- 124 Overview of Web Services
- 125 Consuming Web Services in PHP
- 126 Creating a Web Service in PHP

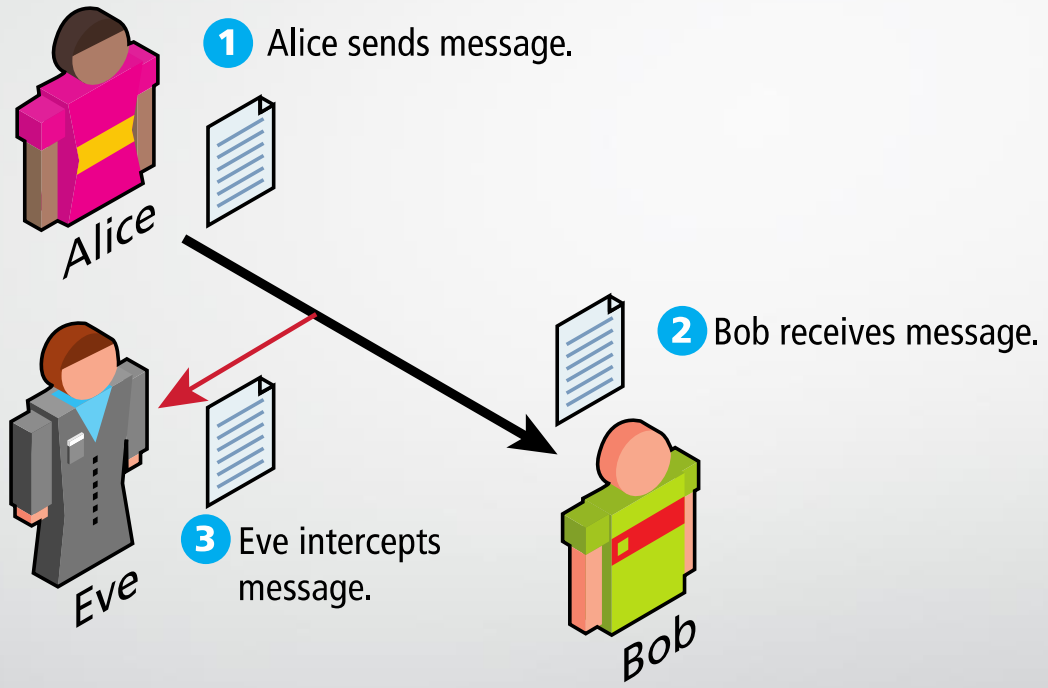
Web Systems and Technologies

TOPIC 118

Cryptography

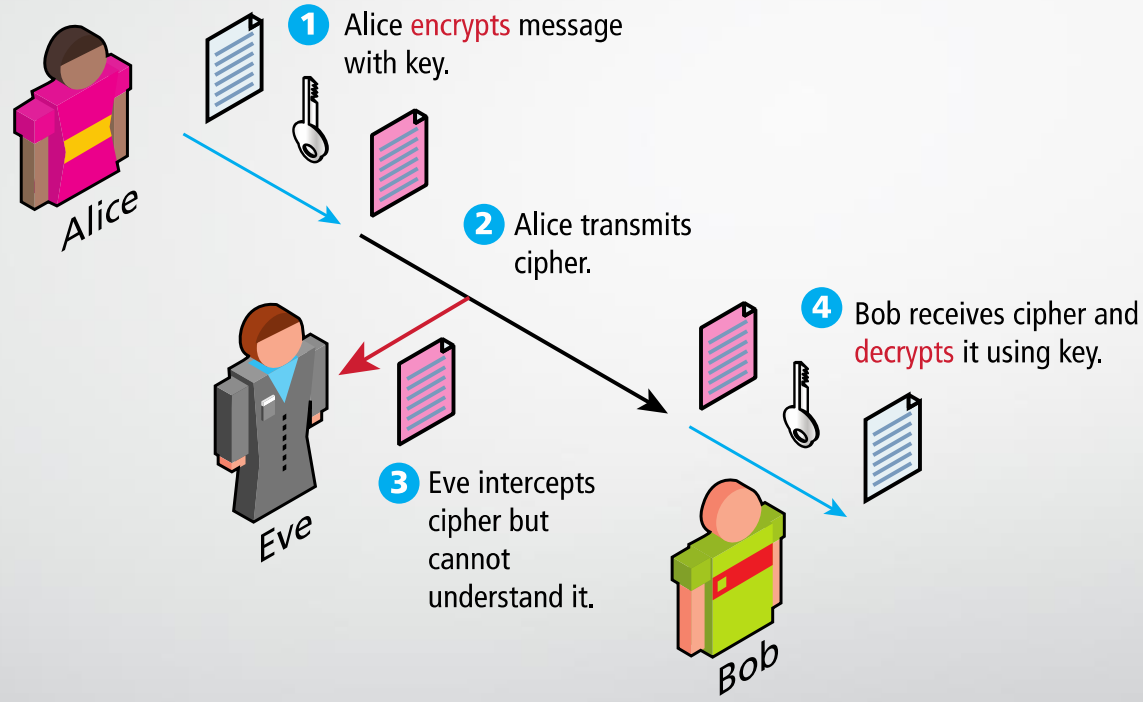
Cryptography

An Overview



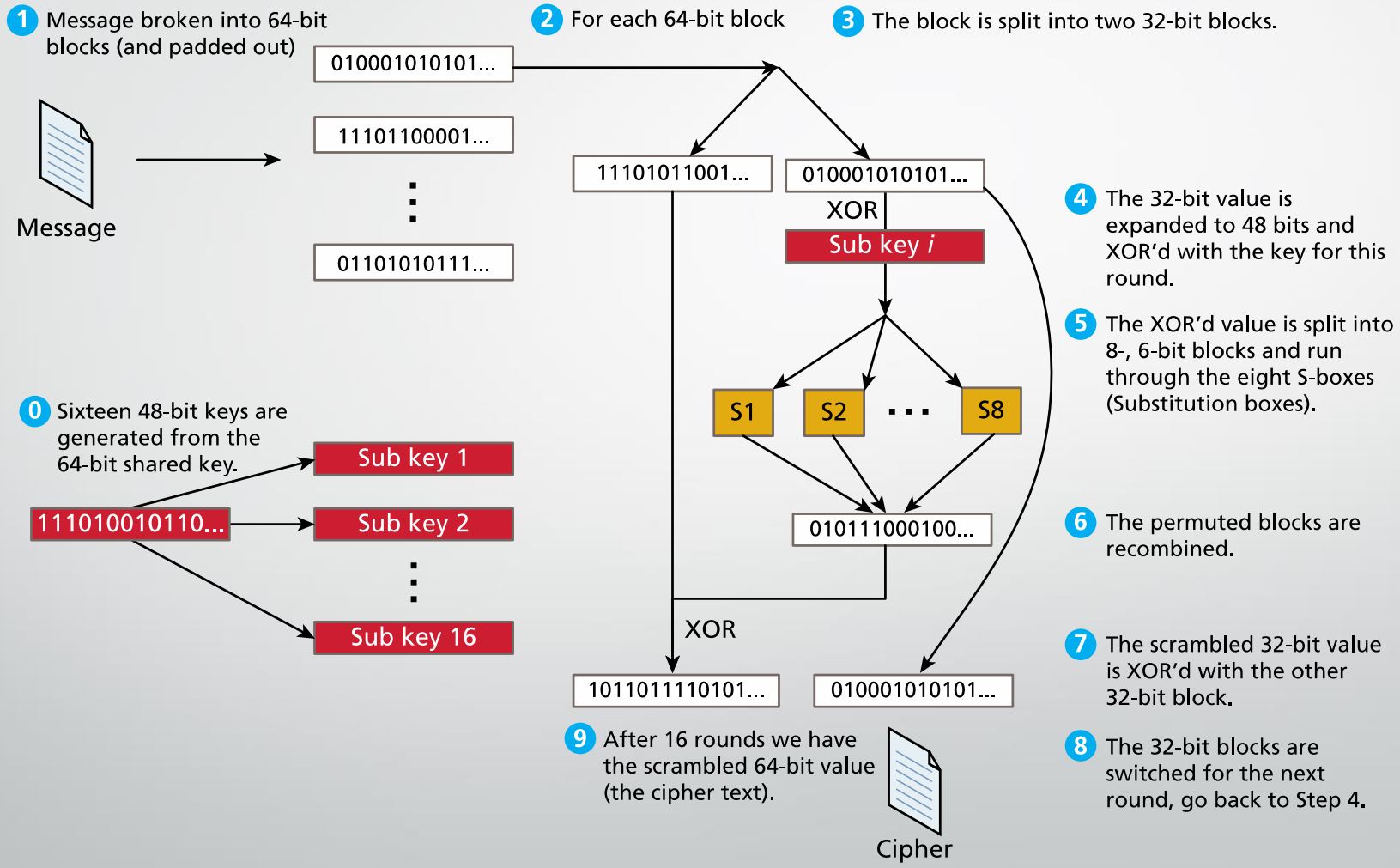
Cryptography

Symmetric Encryption



Cryptography

Modern Block Ciphers



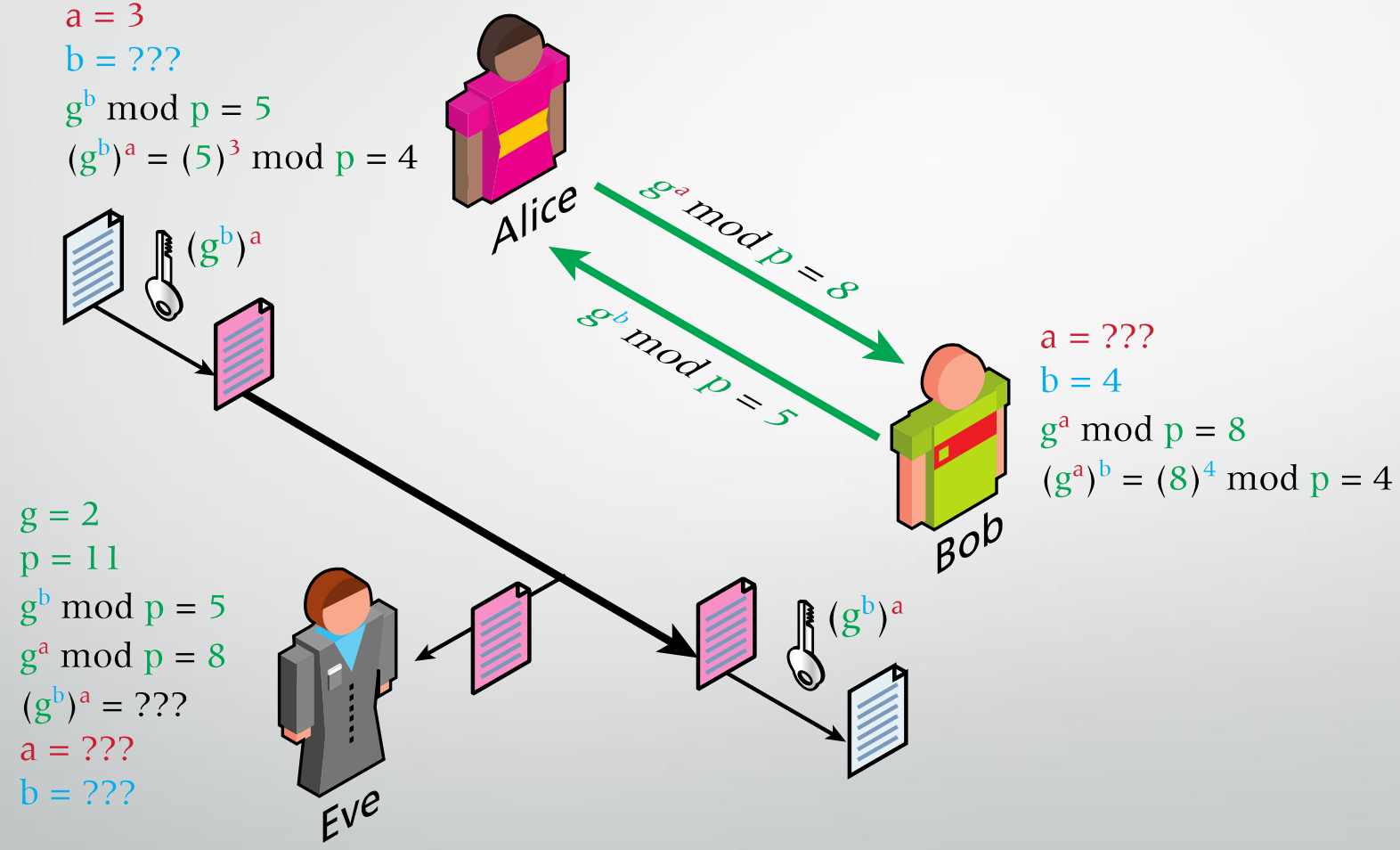
Cryptography

Public Key Cryptography

- Public key cryptography (or asymmetric cryptography) solves the problem of the secret key by using two distinct keys: a public one, widely distributed and another one, kept private.

Cryptography

Public Key Cryptography – Diffie Hellman



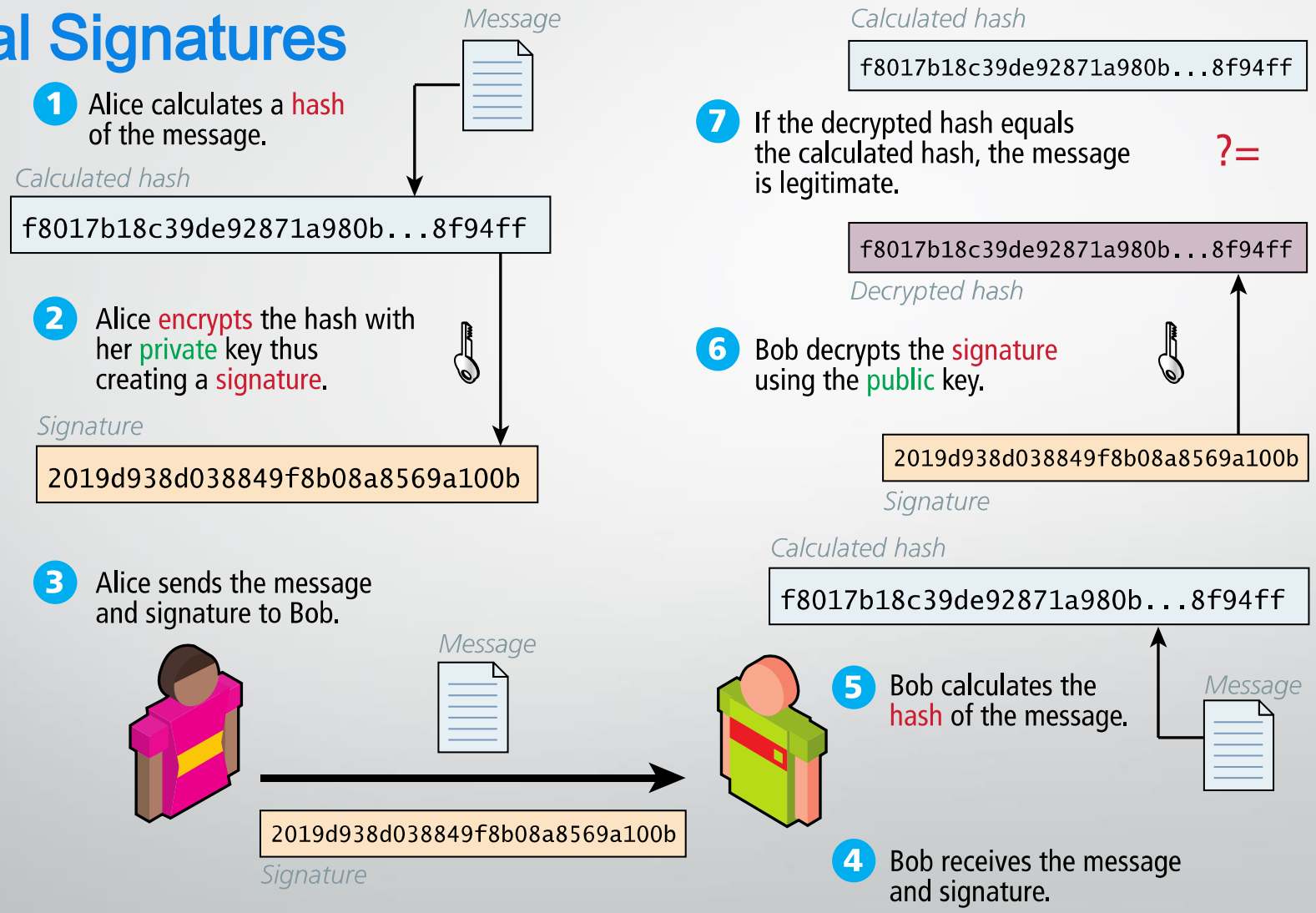
Cryptography

Digital Signatures

- A **digital signature** is a mathematically secure way of validating that a particular digital document was
  - created by the person claiming to create it (authenticity),
  - was not modified in transit (integrity), and
  - cannot be denied (nonrepudiation).

Cryptography

Digital Signatures



Web Systems and Technologies

TOPIC 119

HTTPS

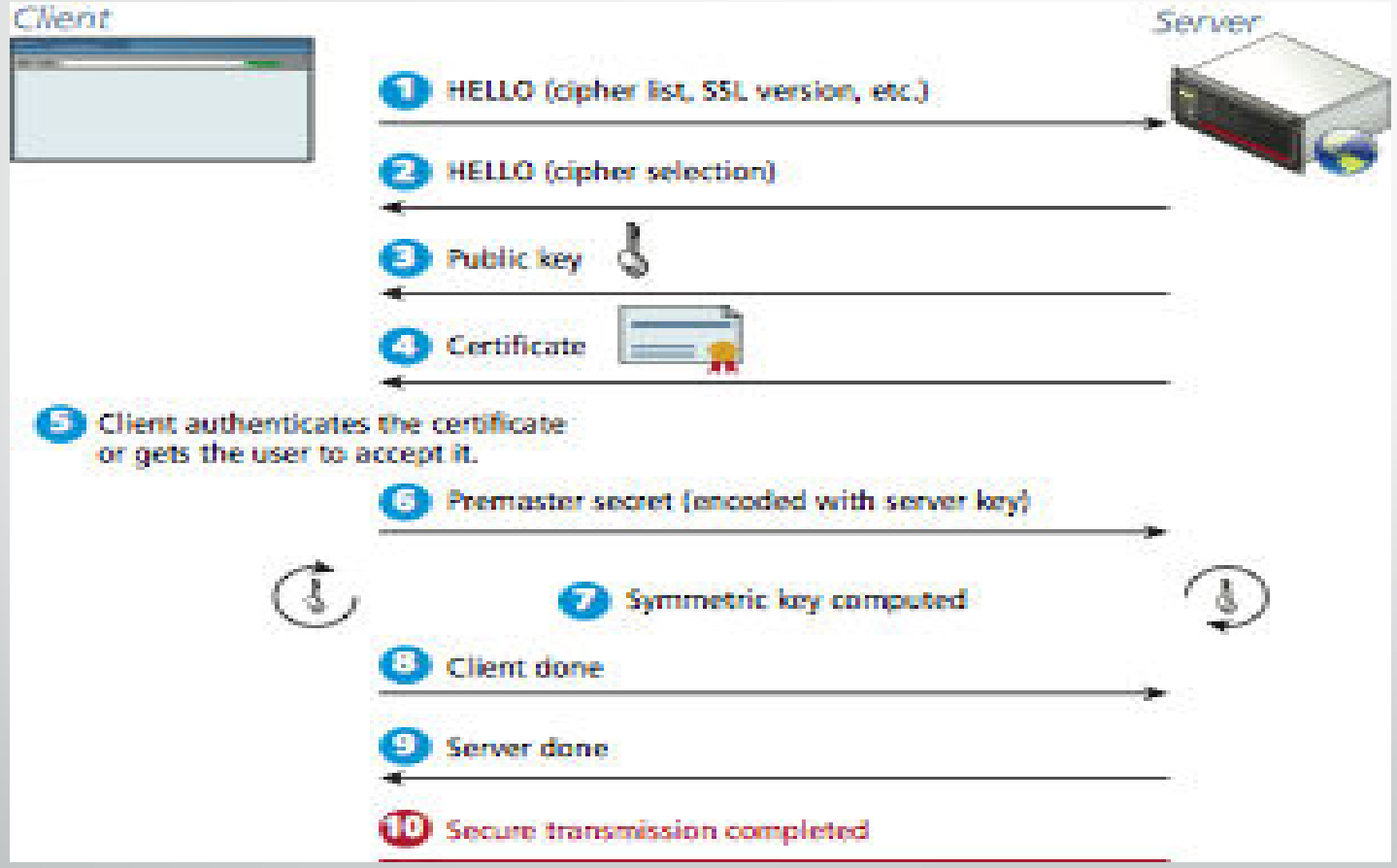
Hypertext Transfer Protocol Secure (HTTPS)

What the Browsers Say



Hypertext Transfer Protocol Secure (HTTPS)

Secure Handshakes



Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities

Plain text content

Common Name: funwebdev.com
Organization: funwebdev.com
Locality: Calgary
State: Alberta
Country: CA
Valid From: July 23, 2013
Valid To: July 23, 2014
Issuer: funwebdev.com, funwebdev.com
Key Size: 1024 bit
Serial Number: 9f6da4acd62500a0



Actual transmitted certificate

-----BEGIN CERTIFICATE-----
MIICfTCCAeYCCQCfbaSs1iUAoDANBgkqhkiG9w0BAQUFADCBgjEL
MAkGA1UEBhMCQ0ExEDAOBgNVBAgTB0FsYmVydGExEDAOBgN
VBAcTB0NhbGdhcnkxFjAUBgNVBAoTDWZ1bndlYmRldi5jb20xFjAU
BgNVBAMTDWZ1bndlYmRldi5jb20xHzAdBgkqhkiG9w0BCQEWEH
Job2FyQG10cm95YWwuY2EwHhcNMTMwNzIzMjI0NjU2WhcNMT
QwNzIzMjI0NjU2WjCBgjELMAkGA1UEBhMCQ0ExEDAOBgNVBAg
TB0FsYmVydGExEDAOBgNVBAcTB0NhbGdhcnkxFjAUBgNVBAoTD
WZ1bndlYmRldi5jb20xFjAUBgNVBAMTDWZ1bndlYmRldi5jb20xHz
AdBgkqhkiG9w0BCQEWEHJob2FyQG10cm95YWwuY2EwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMSS8uQ6ZXVW6yV
6MUcdZxdQTPfUlpXXW6DYmQMVmOEE7mjrhmj3jLDQn+FU8Qsv
IS8+GrDoyZ/5hhGBLYQLIhIcRQBULS9yNRIB7+mWOT45QycqJH/9xC
VcTwI4D//qVvAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAzOsxgr
ltLw/DZXmqcV/W8C859m43D3gbc66jaaNYu5cA+Fn2FpS7z8oYeV
m0wWXcrmIj4bIWvpp3IbhPT12+XcVfJMda4nLSb/SPyjv4yvz9jeL
Ya/c0Z1IA7v6bk1ixwZSB9E=
-----END CERTIFICATE-----

Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities

- **Certificate Authority (CA)** allows users to place their trust in the certificate since a trusted, independent third party signs it.
- The CA's primary role is to validate that the requestor of the certificate is who they claim to be, and issue and sign the certificate containing the public keys so that anyone seeing them can trust they are genuine.
- In browsers, there are many dozens of CAs trusted by default
- You can also self-sign certificates (generates warnings)

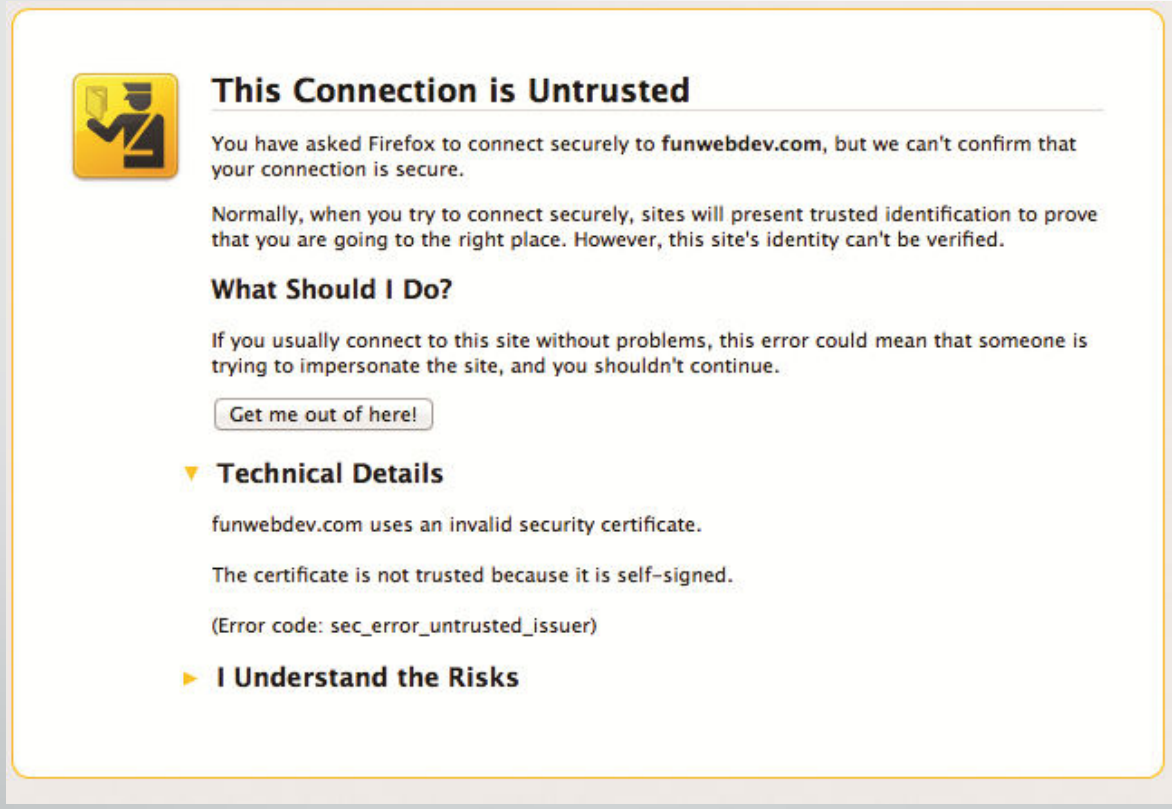
Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities



Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities



Hypertext Transfer Protocol Secure (HTTPS)

Migrating to HTTPS from HTTP

- Coordinating the migration of a website can be a complex endeavor
- Mixed Content
  - Internal links within the site.
  - External links to frameworks delivered through a CDN.
  - Any links or references generated by PHP code that might include a hardcoded http.
  - References to http within any HTML markup outside of PHP blocks.
- Redirects from old site

Web Systems and Technologies

TOPIC 120

XML Overview

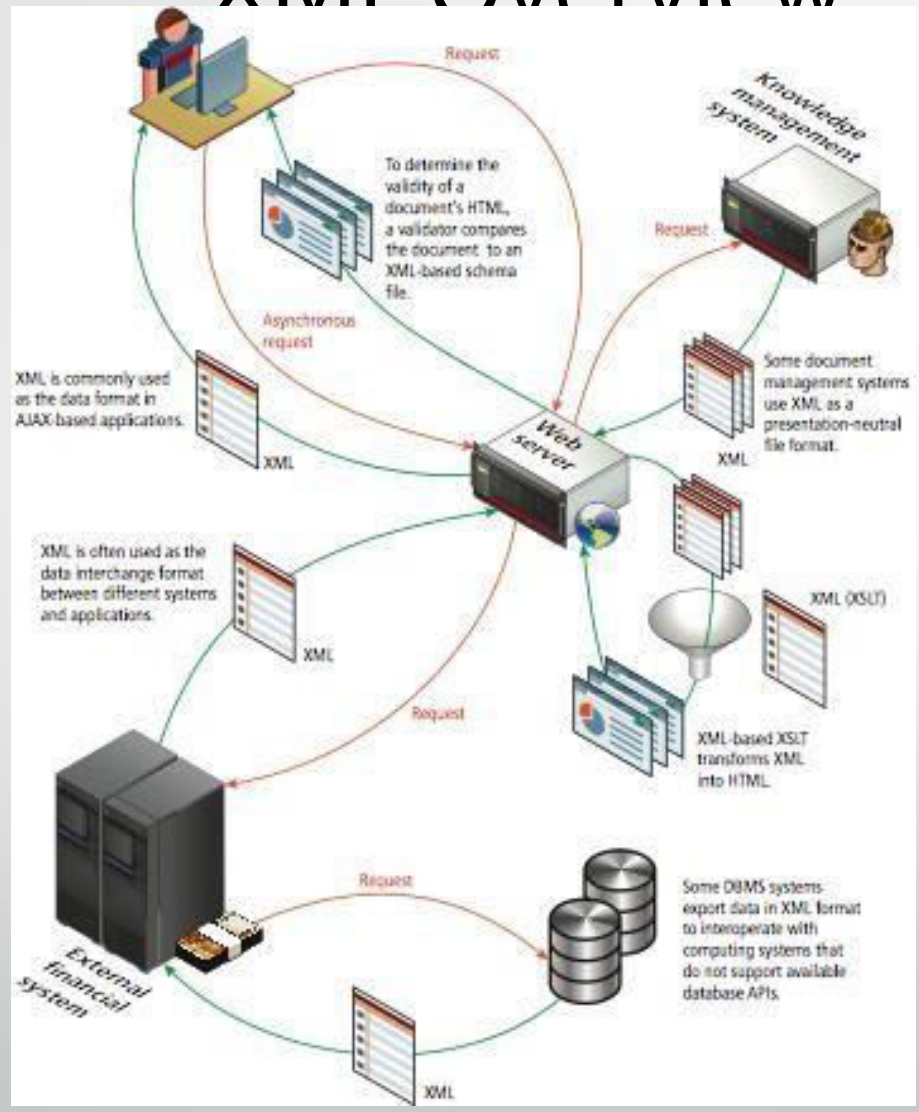
XML Overview

Introduction

- XML is a markup language, but unlike HTML, XML can be used to mark up any type of data
- One key benefit of XML data is that as plain text, it can be read and transferred between applications and different operating systems
- 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

- XML Usage



XML Overview

Well-Formed XML

- Element names are composed of any of the valid characters
- Element names can’t start with a number
- There must be a single-root element
- 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

XML Overview

Well-Formed XML Simplified Example

```
<?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>
</art>
```

XML Overview

Valid XML

- A valid XML document is one that is well formed and whose element and content conform to the rules of either its document type definition (DTD) or its schema
- DTDs tell 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

XML Overview

Example Document Type Definition (DTD)

```
<?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>
</art>
```

```
<?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)>
]>
```

XML Overview

- Example XML Schema

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema 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:attribute type="xs:integer" name="id"/>
    <xs:sequence>
      <xs:element type="xs:string" name="title"/>
      <xs:element name="artist">
        <xs:complexType>
          <xs:choice>
            <xs:element type="xs:string" name="name"/>
            <xs:element type="xs:string" name="nationality"/>
          </xs:choice>
        </xs:complexType>

          </xs:element>
```

```
<?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>
</art>
```

XML Overview

- Example XML Schema

```
            <xs:element type="xs:short"
            name="year"/>
            <xs:element type="xs:string"
            name="medium"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

```
<?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>
</art>
```

Web Systems and Technologies

TOPIC 121

XPath and XSLT

XPath and XSLT

XPath

- XML Path Language
- Used for accessing elements and attributes in an XML document
- Widely used in JavaScript for accessing certain elements/attributes of HTML documents

XPath and XSLT

XPath – Expressions

Expression	Description	
nodename	All nodes with the name "nodename"	painting
/	From root	/art/painting
//	Anywhere in the document	
.	Current node	
..	Parent node	
@	Attribute	
[]	Predicate	

XPath and XSLT

```
art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

XPath – Examples

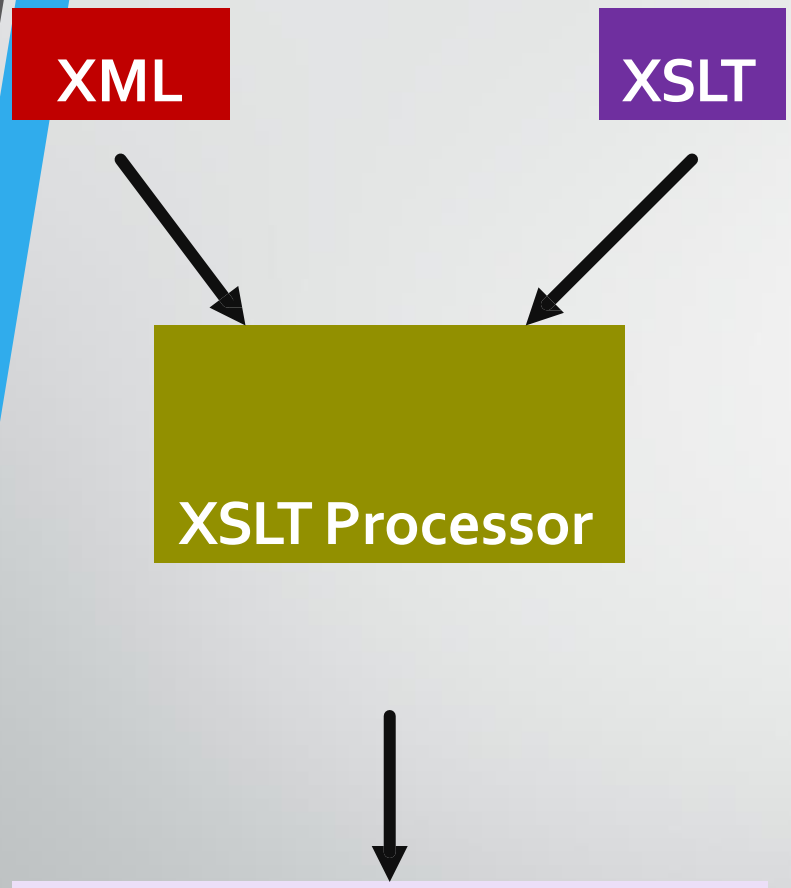
- /art
  - Selects element "art"
- //year
  - Selects element "year"
- //name/..
  - Selects element "artist"
- //painting/@id
  - Selects attribute "id" in painting
- //painting[year>1800]
  - Selects "painting" elements with year > 1800

XPath and XSLT

XSLT

- eXtensible Stylesheet Language Transformation (XSLT)
- Used for transforming an XML document into another form
- Defined as an XML document
- Uses XPath for selecting elements and built-in functions for processing/transforming them

XPath and XSLT



XSLT – Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/art">
  <h1>Paintings</h1>
  <ul>
    <xsl:for-each select="painting">
      <li>
        <xsl:value-of select="title"/> by
        <xsl:value-of select="artist/name"/>
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>

</xsl:stylesheet>
```

Web Systems and Technologies

TOPIC 122

XML Processing

XML Processing

XML Processing in JavaScript

- The in-memory approach, which involves reading the entire XML file into memory
- The event or pull approach, which lets you pull in just a few elements or lines at a time

XML Processing

XML Processing in JavaScript

```
var xmlhttp = new XMLHttpRequest()
xmlhttp.open("GET","art.xml",false);
xmlhttp.send();
var xmlDoc = xmlhttp.responseXML;
var paintings = xmlDoc.getElementsByTagName("painting");
```

XML Processing

XML Processing in jQuery

```
var art = '<?xml ...';

// create the DOM object
var xmlDoc = $.parseXML(art);

// convert DOM object to jQuery object var xml =
$(xmlDoc);

// find all the painting elements  var
paintings = xml.find("painting");
```

XML Processing

XML Processing in PHP

- Can process XML using
  - SimpleXML
  - XMLReader

XML Processing

SimpleXML

```
<?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>';
}
```

XML Processing

XPath with SimpleXML

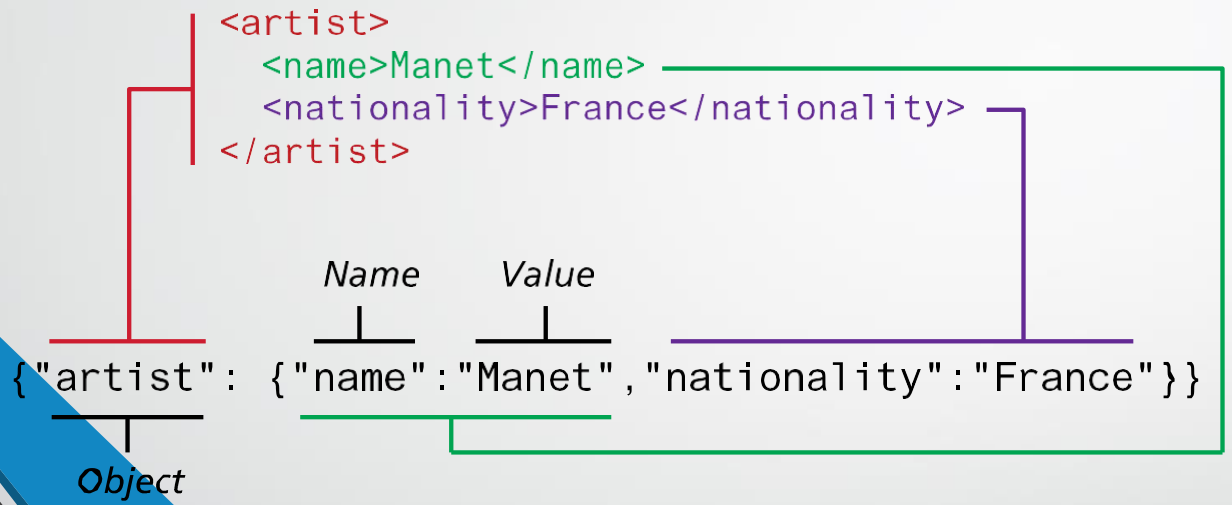
```
$art = simplexml_load_file($filename);
$titles = $art->xpath('/art/painting/title');
foreach ($titles as $t) {
  echo $t . '<br/>';
}
$names = $art->xpath('/art/painting[year>1800]/artist/name');
foreach ($names as $n) {
  echo $n . '<br/>';
}
```

Web Systems and Technologies

TOPIC 123: JSON

JSON

Sample JSON



JSON

Using JSON in JavaScript

```
var text =  '{"artist": {
             "name":"Manet",
             "nationality":"France"
                    }
                  }';
var a = JSON.parse(text);  alert(a.artist.nationality);
//or a['artist']['nationality']
```

JSON

Using JSON in PHP

```
<?php
// convert JSON string into PHP object
$text =  '{"artist": {
              "name":"Manet",  "nationality":"France"
                  }
                }';
$anObject = json_decode($text);
// check for parse errors
if (json_last_error() == JSON_ERROR_NONE)
{
  echo $anObject->artist->nationality;
}

?>
```

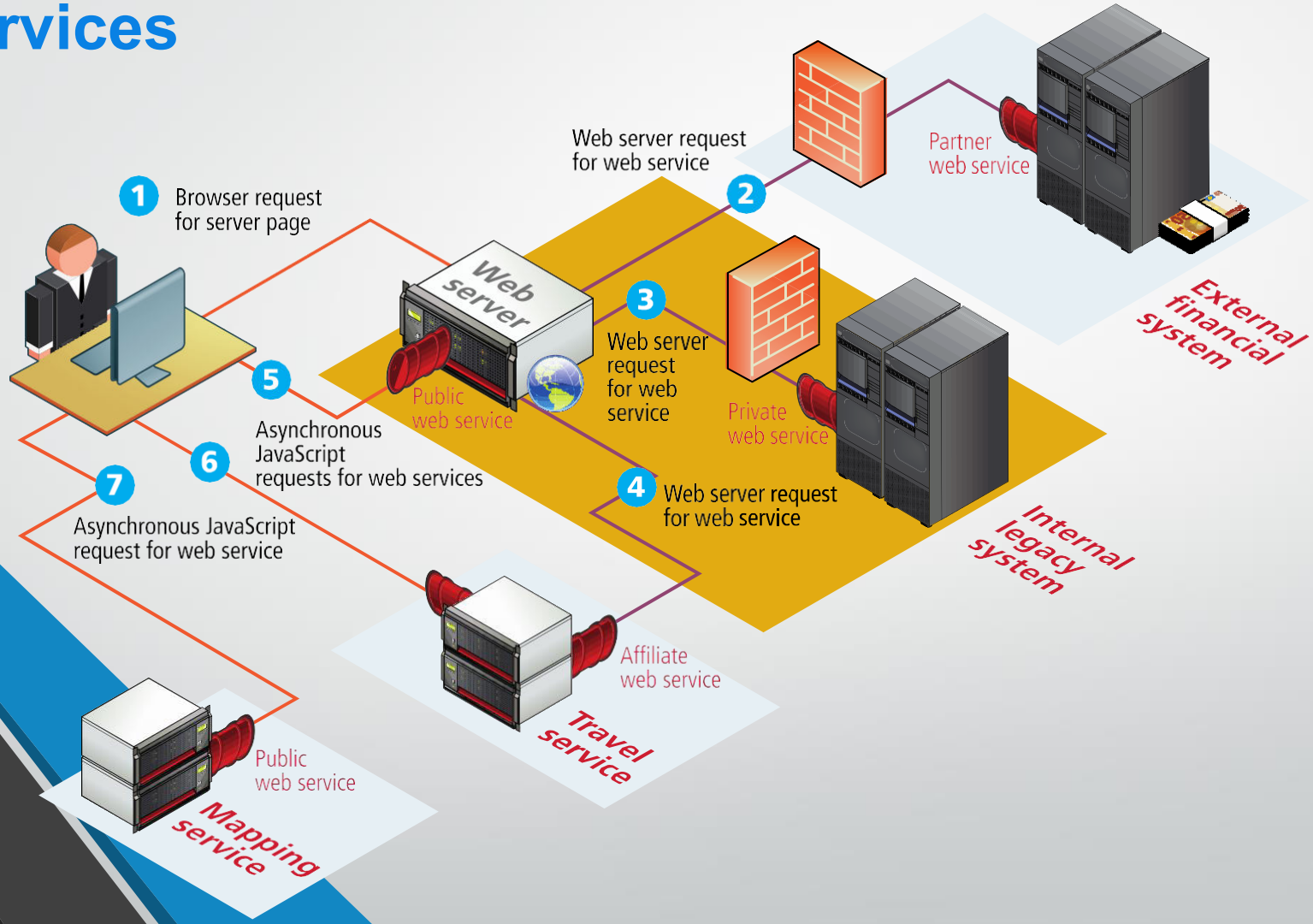
Web Systems and Technologies

TOPIC 124

Overview of Web Services

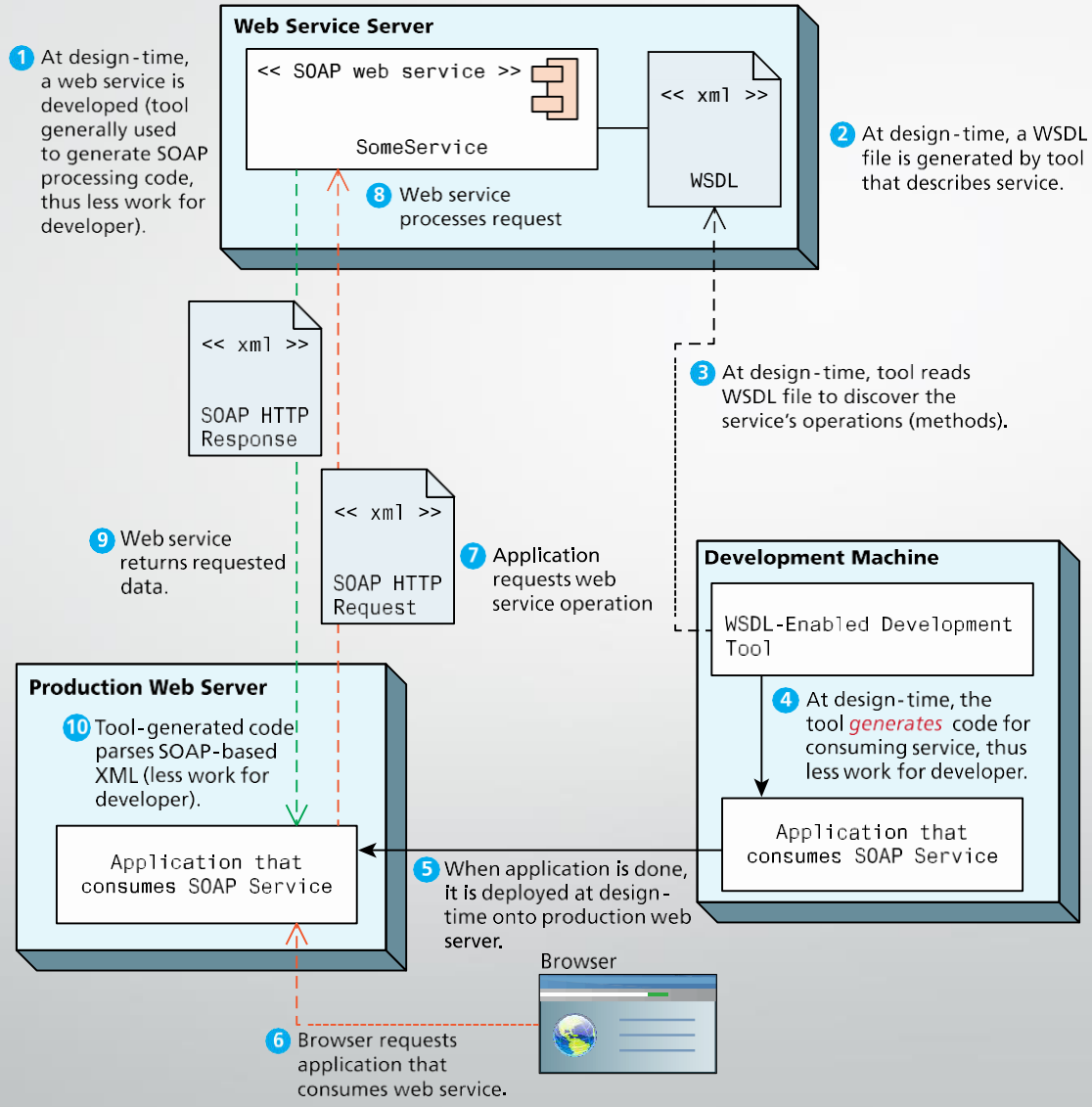
Overview of Web Services

Web Services



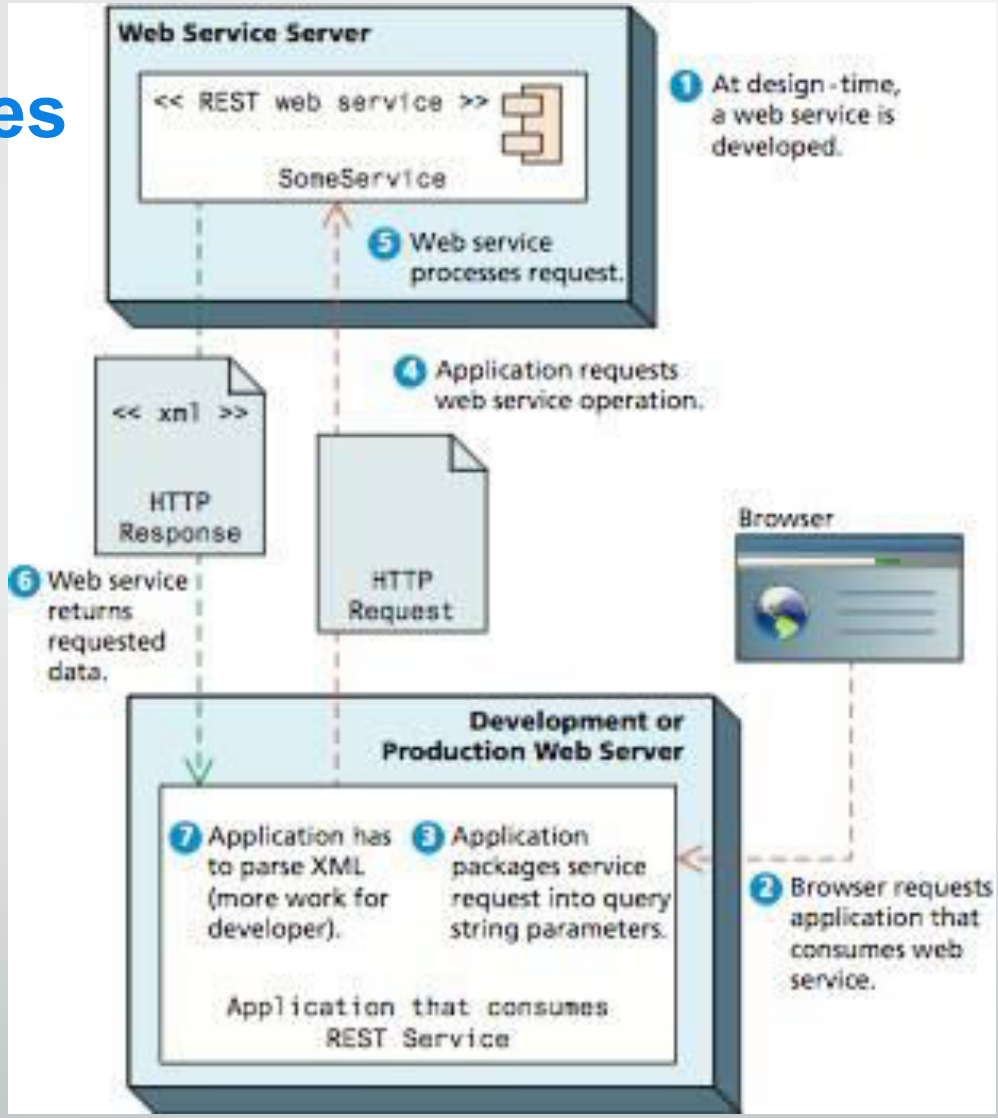
Overview of Web Services

SOAP Services



Overview of Web Services

REST Services



/authors/1

Get Post
Put Delete

Overview of Web Services

Identifying and Authenticating Service Requests

- 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.
- API Keys
- https://dev.virtualearth.net/REST/v1/Locations?o=json&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&key=**[BING API KEY HERE]**

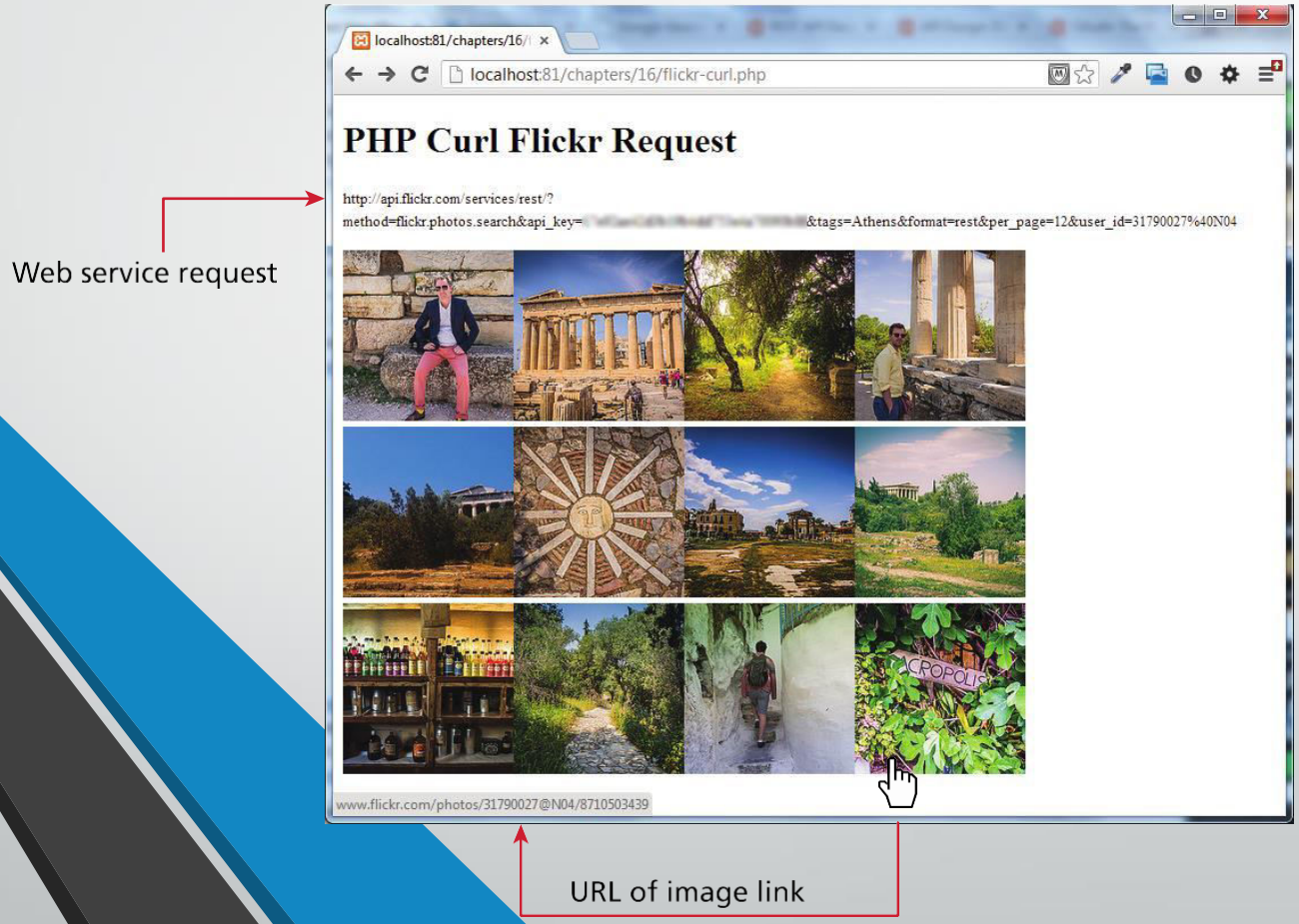
Web Systems and Technologies

TOPIC 125:

Consuming Web Services in PHP

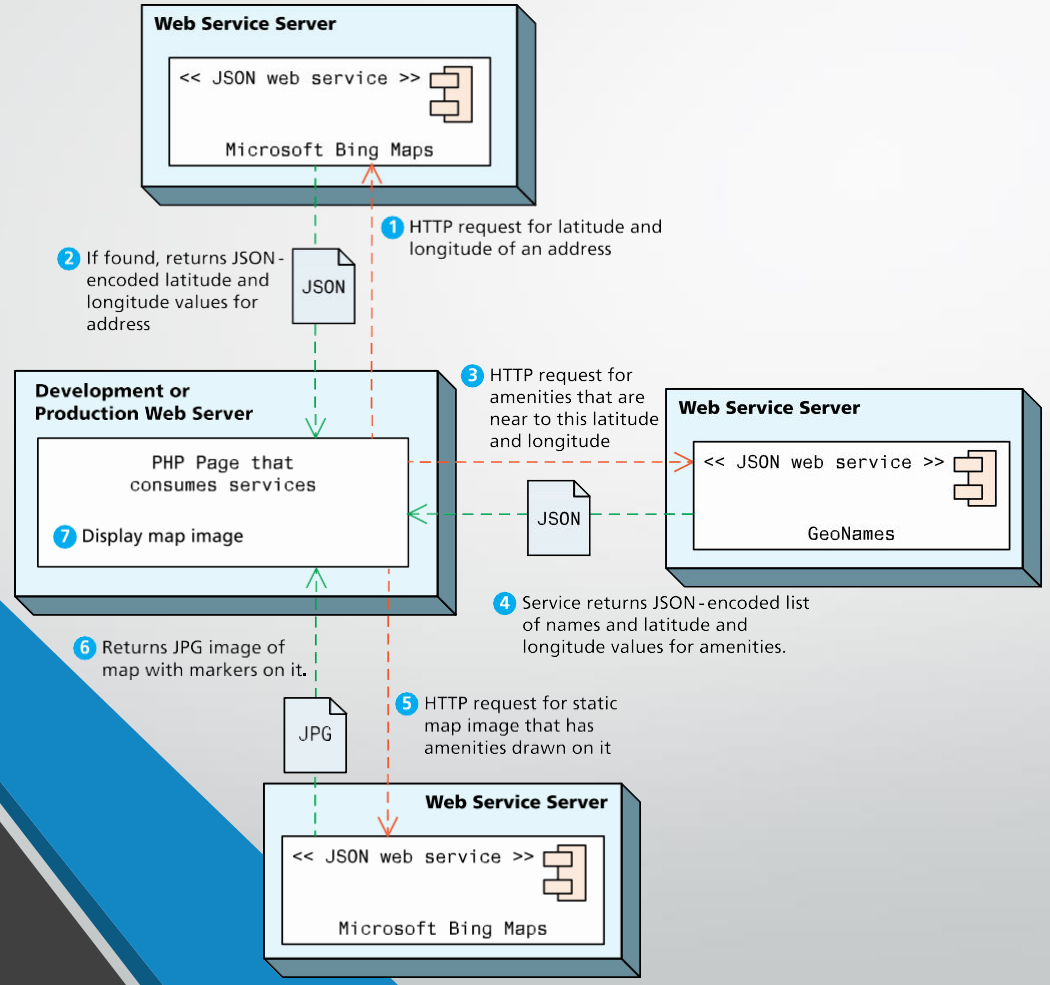
Consuming Web Services in PHP

Consuming an XML Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service

URL of service request for static road map image

Zoom level (between 1 and 21)

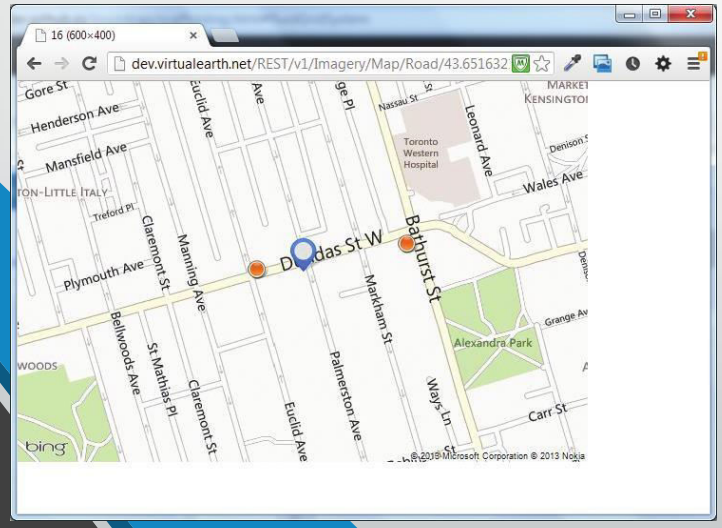
```
http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/43.65163,-79.40853/16?
key=[your api key]
&mapSize=600,400
&pp=43.65163,-79.40853;66;
&pp=43.65208,-79.40618;34;
&pp=43.65166,-79.40958;34;
```

Width and height of map in pixels

Location (latitude and longitude) of center of map

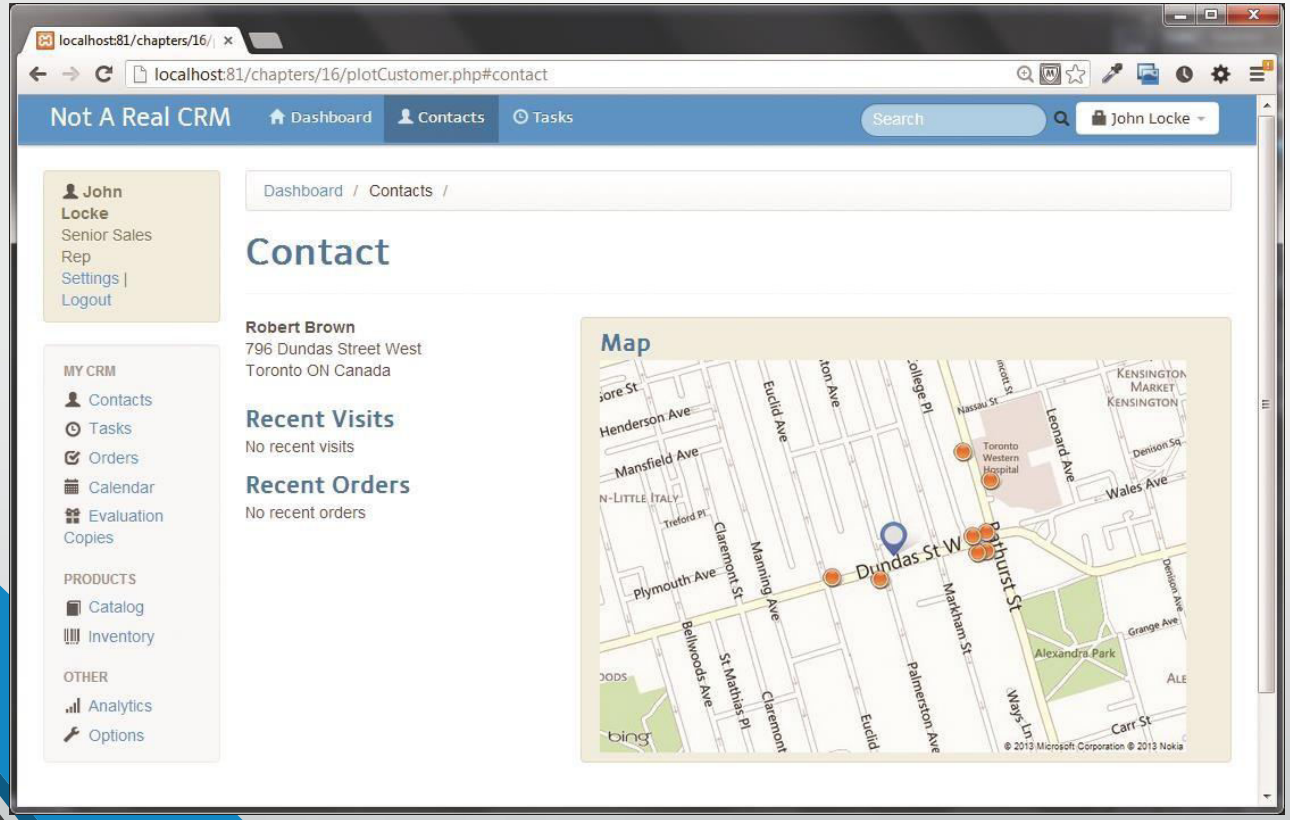
Location of marker (marker 66 = blue circle)

Location of other markers (amenities) with marker 34 = orange circle



Consuming Web Services in PHP

Consuming a JSON Web Service



Web Systems and Technologies

TOPIC 126

Creating Web Services

Creating Web Services

Creating a JSON Web Service

- Consider the URL and format of requests
- Tell the browser to expect JSON rather than HTML
  - Header('Content-Type: application/json');
- Use json_encode() to format.
- Implement *JsonSerializable*

Creating Web Services

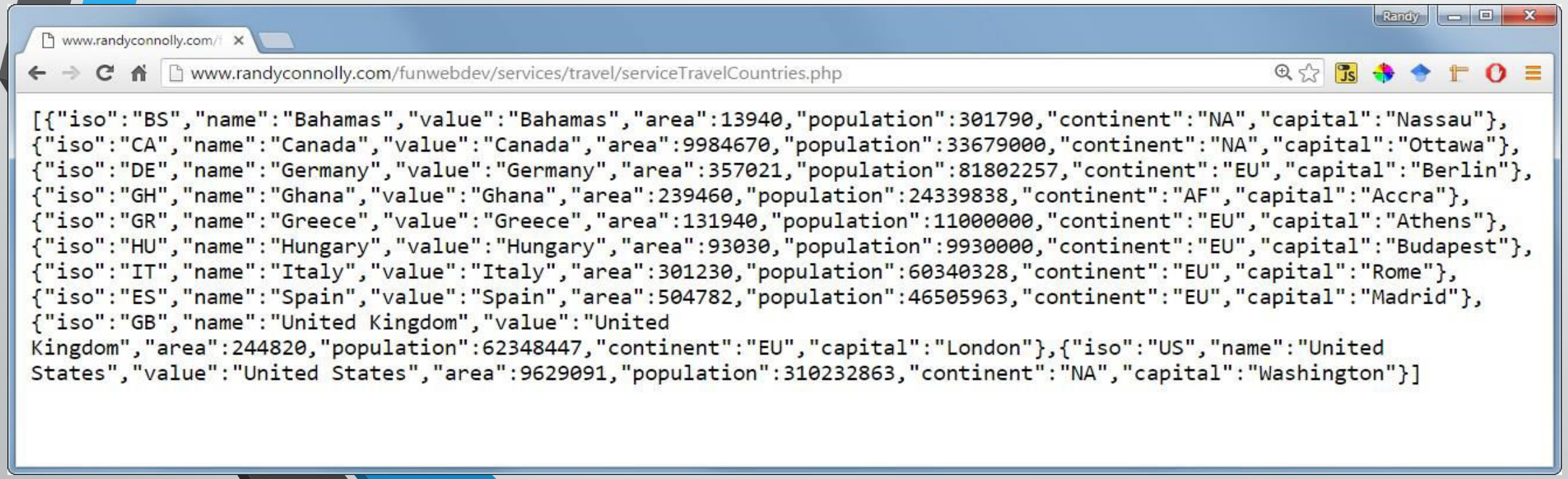
Creating a JSON Web Service

```
class Country extends DomainObject implements JsonSerializable
{
...
  /*
  This method is called by the json_encode() function that is part of PHP
  */
  public function jsonSerialize() { return
['iso'  =>  $this->ISO,  'name' => $this-
>CountryName,  'value' => $this->CountryName,
'area'  =>  $this->Area,  'population' =>
$this->Population,  'continent' => $this-
>Continent,  'capital' => $this->Capital

]; } }
```

Creating Web Services

Creating a JSON Web Service



Web Systems and Technologies

WEEK 15

TOPIC 127 TO 135

Web Systems and Technologies

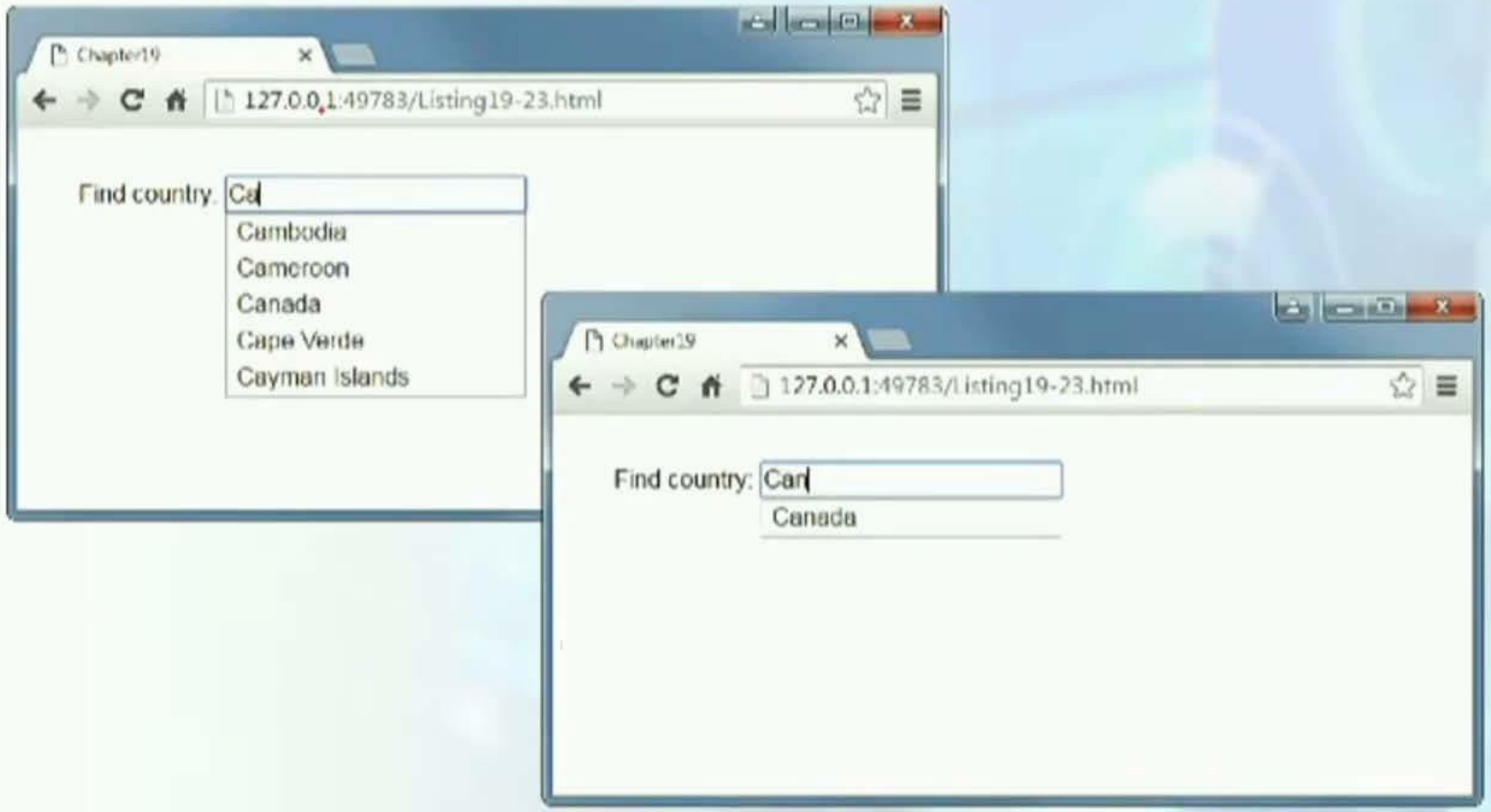
- 127-Web Services Using AJAX
- 128-JavaScript Frameworks
- 129-Node.JS
- 130-Simple Node.JS Program
- 131-Express.js, Routing in Express
- 132-Node.JS MYSQL
- 133-Introduction to Angular
- 134-Introduction to Web Sockets
- 135-Web Server Hosting

Interacting with Web Services using AJAX

TOPIC 127

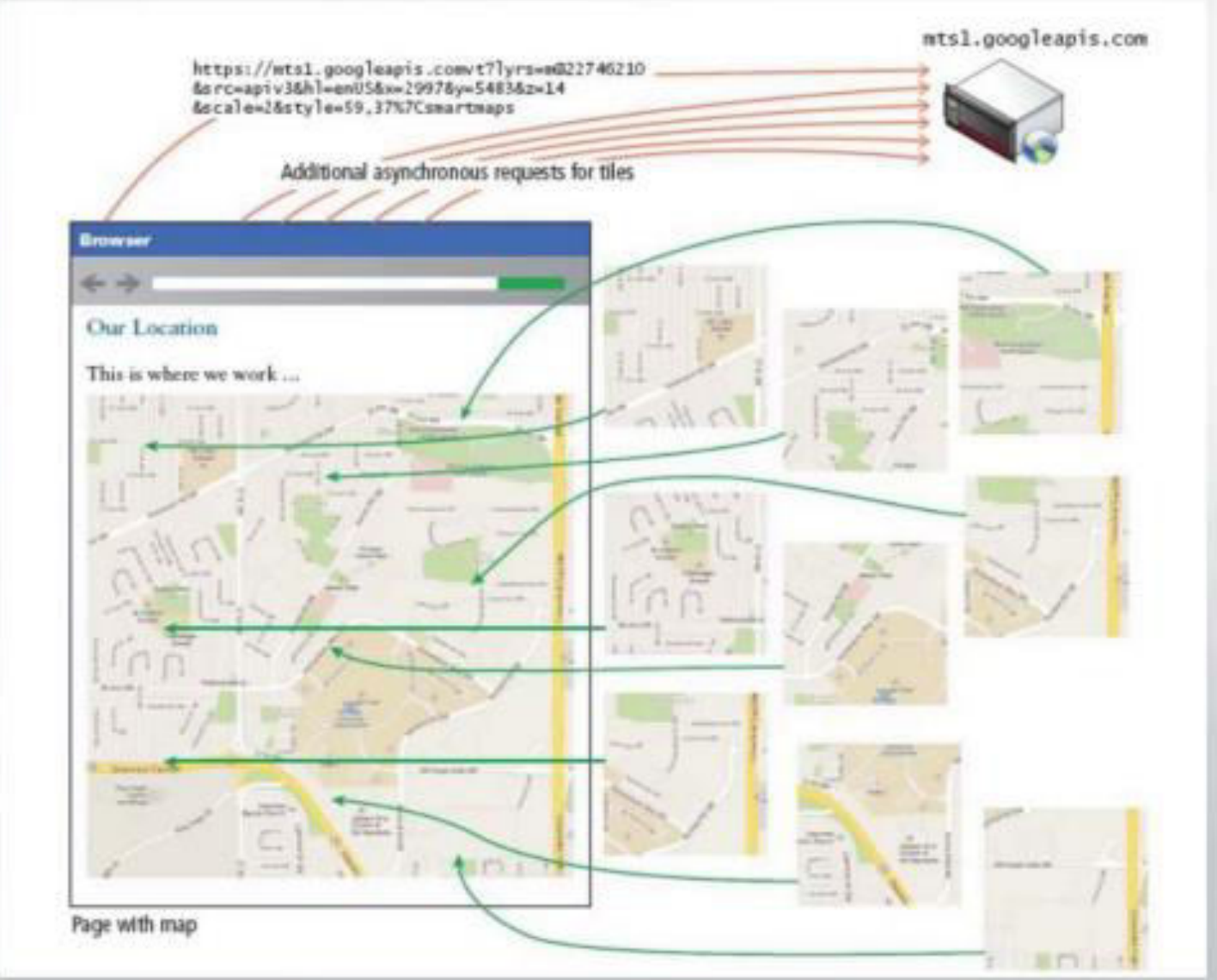
Web Systems and Technologies

Consuming Your Own Service



Using Google maps

Using Google Maps



Using Google Maps

```
<style>
#map {
  height: 500px;
  width: 600px
}
</style>

<script
src="https://maps.googleapis.com/maps/a
pi/js?key=YOUR_API_KEY" async></script>
```

Using Google Maps

```
<script>
  $(function() {
  // hard-coded latitude and longitude for demonstration purposes
  var ourLatLong = {lat: 51.011179 , lng: -114.132866 };
  var ourMap = new google.maps.Map(document.getElementById('map'), {
    center: ourLatLong,
    scrollwheel: false,
    zoom: 14
    });});
</script>
</head>
<body>
  <h2>Our Location</h2>
  <h3>This is where we work ... </h3>
  <div id="map"></div>
</body>
</html>
```

TOPIC 128

JavaScript
Frameworks

Web Systems and Technologies

JavaScript Frame work

JavaScript Frame work

JavaScript framework provide structure for developing JavaScript applications

Popular frameworks

- Angular
- Ember
- Vue
- React
- Express

JavaScript Front-End Frameworks

- **Ember** forces developers to adopt a known and well-regarded approach to structuring and implementing a web application. It uses a variant of the MVC pattern
- **Angular** has many similarities to Ember (i.e., models, templates, and routing), and has the added advantage of being partially maintained by Google.
- **React** is developed by Facebook. Unlike Ember and Angular, React is not a complete MVC-like framework; instead it focuses on the view.

JavaScript for Backend Node.js

- Alternative to LAMP stack
- MEAN stack

  MongoDB-Express-Angular-Node.js
- MERN stack

  MongoDB-Express-React-Node.js

Node.js

TOPIC 129

Web Systems and Technologies

Introduction

- Started in 2009
- Runs Javascript outside the browser
- Uses Chrome V8 Javascript engine
- Javascript used to generate HTML code
- Uses non-blocking I/O
- Often used for server-side web Development

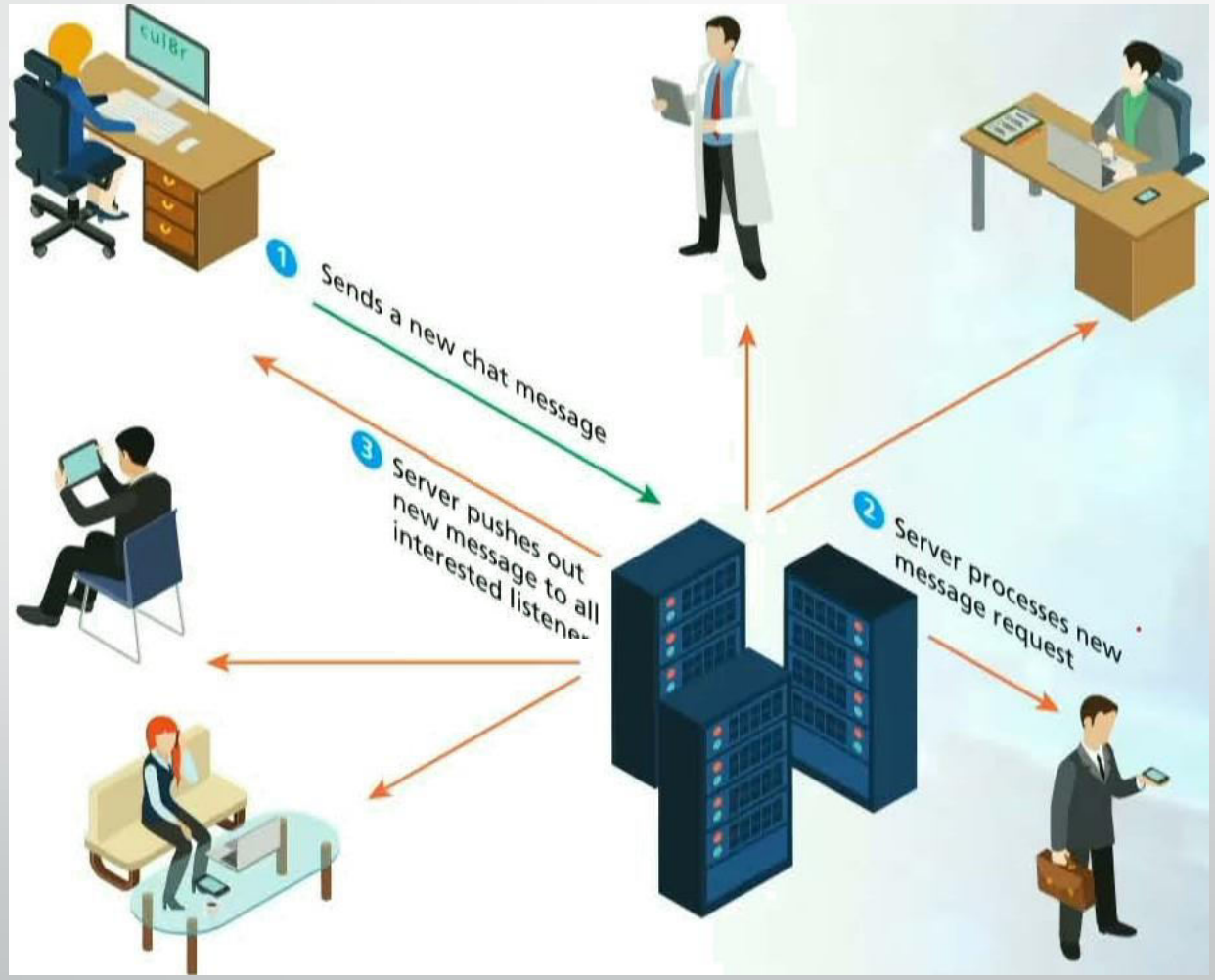
    Also used for non-web development,

Introduction

- LinkedIn switched from Ruby on Rails to Node.js
  - Up to 20 times faster
  - Servers reduced from 30 to 3
- PayPal switched from Java to Node.js
  - Developed in less time, with less developers
  - 33% less lines of code
  - 40% less files
  - Doubled performance (requests/second), with less processors
  - Fast response time, 35% less response time

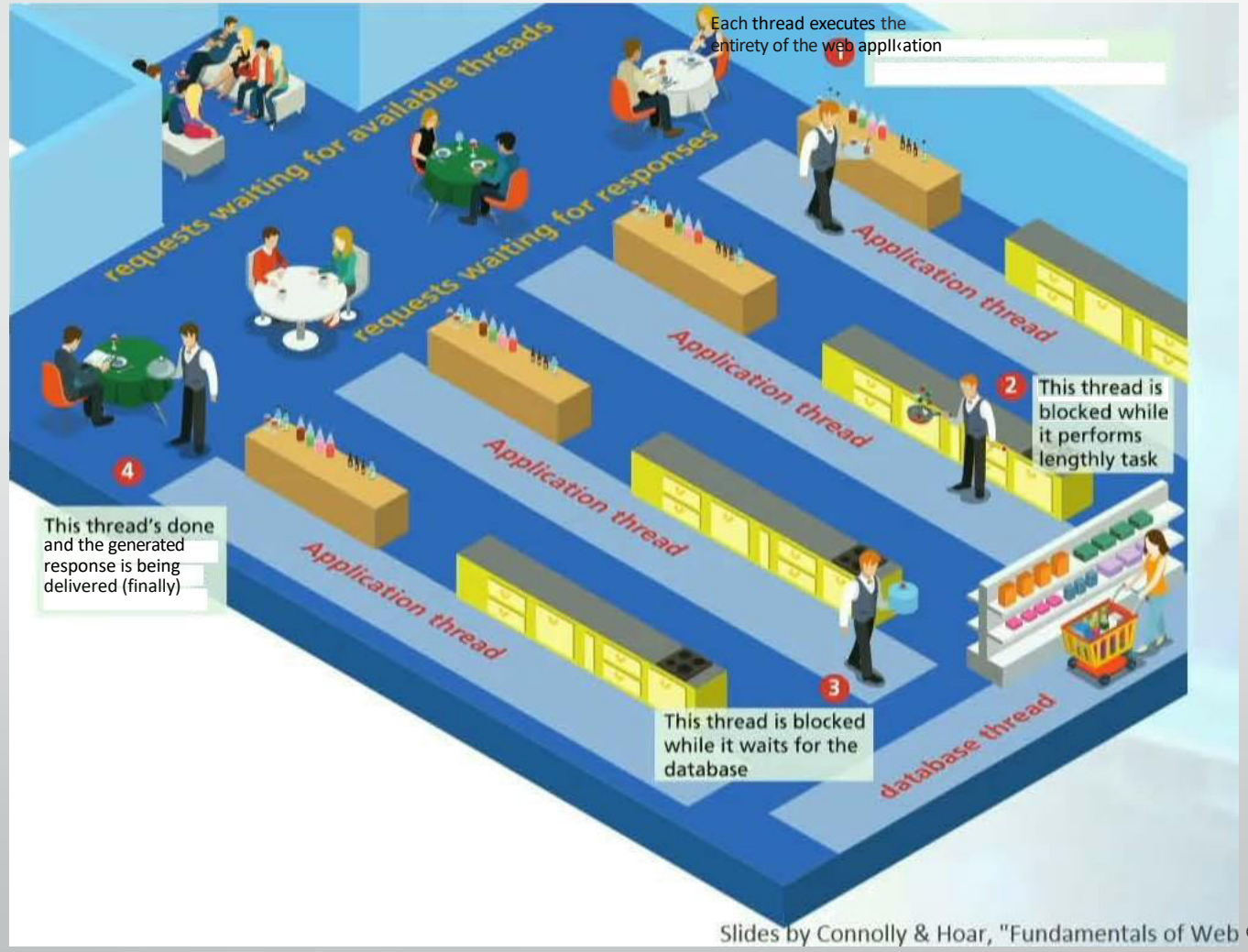
Node.js

Push based web application

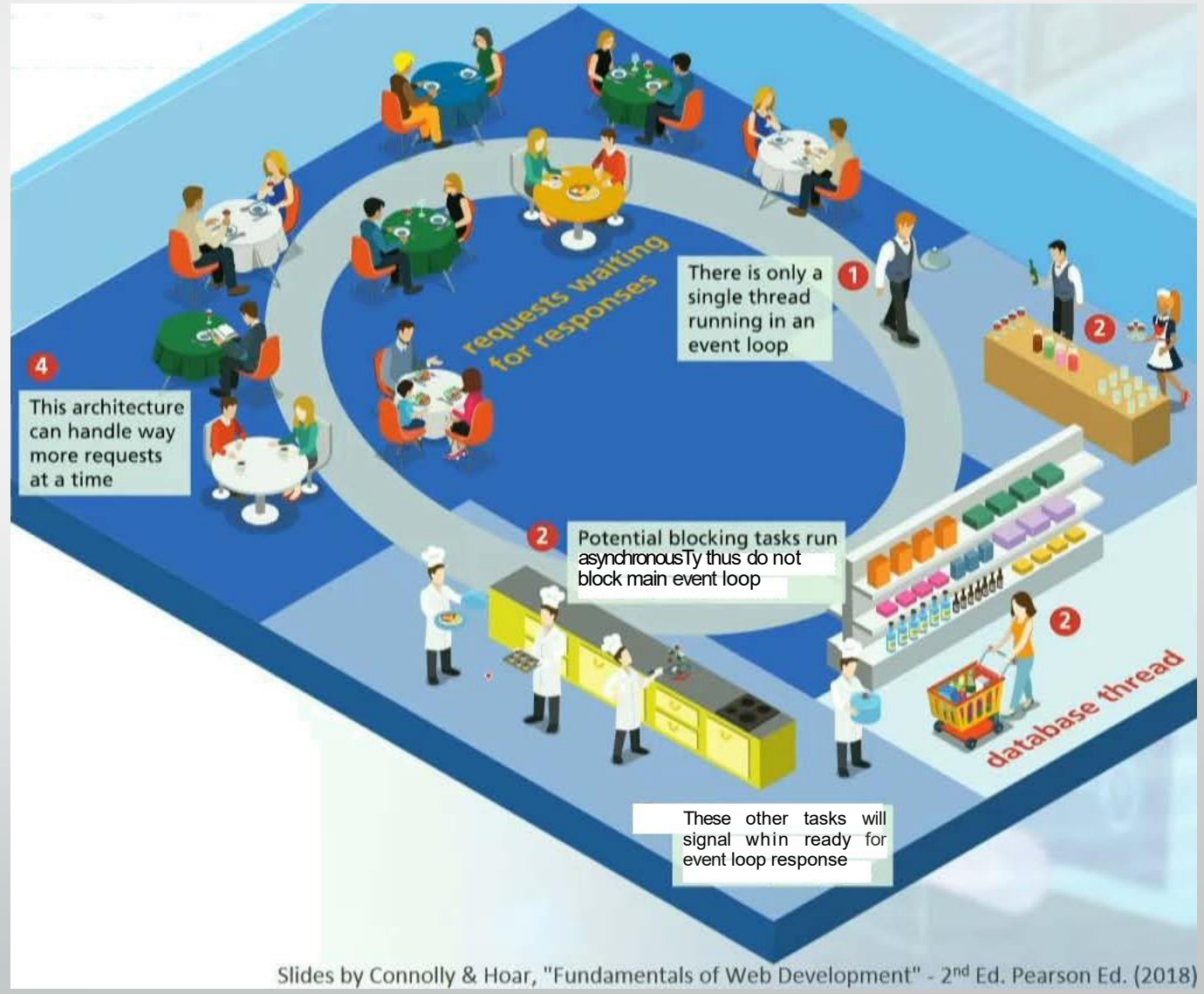


Slit1es by CDnnDlly & I3oar, "Fundamental.s of Web Development" - 2'* Ed. Pearson Ed. (2018)

Blocking Thread-based Architecture (Traditional)



Node.js Single-Thread Architecture



Node Package Manager (NPM)

- Package manager for JavaScript
- Used for installing JavaScript libraries and frameworks
- Can handle dependencies
- Usage

```
npm install
npm install packagename
npm install packagename@version
npm update
npm list
```

Node.js – Example

TOPIC 130

Web Systems and Technologies

Working with Node.js

```
// Load the http module to create an HTTP server
var http = require('http');

// Configure HTTP server to respond with Hello World to all requests var server =
http.createServer(function (request, response) {


        response.writeHead(200, ("Content-Type": "text/plain"));
        response.write("Hello this is our first node.js application\n"); response.end( );

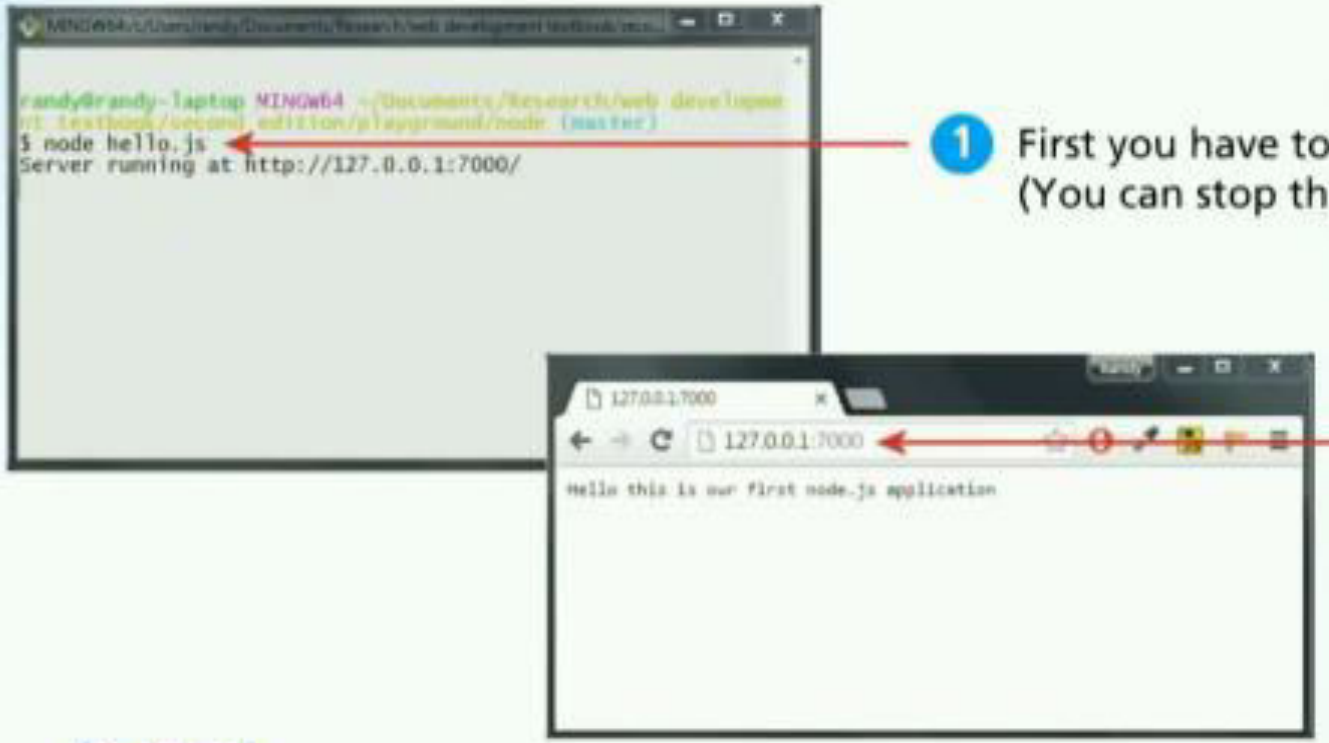
)};


   Listen on port 7000 on localhost server.listen(7000,
”localhost");

   display a message on the terminal

   Console.log(“server running at http://127.0.0.1:7000/”);
```

Working with Node.js



1. First you have to run the program via node command (You can stop the program via Ctrl-C)

2. Then use browser to request URL and port

Note: every time you make a change to your Node source file, you will have to stop the program and re-run it.

Example

- Writing to a file

  data is written in the file 'message.txt'

  When writing is complete, anonymous function is "called back"

```
const fs = require('fs')

fs.writeFile('message.txt',
  'Hello World!', function (err) {
  if (err) {
    console.error(err);
    process.exit(1);
  } else {
    console.log('Writing is done.');
  }
})
```

Express.js

TOPIC 131

Web Systems and Technologies

Introduction

Minimalist web application framework for Node.js

- Handles
  - Routes
  - Requests/Responses
  - Views

Example

```
const express = require('express')
const app = express()
const port = 7000

app.get('/:name', function(req, res) {
  res.send('Hello '.req.params.name)
})

app.listen(port, function (){
 console.log("http://localhost:".${port})
})
```

TOPIC 132

MySQL with Node.js

Web Systems and Technologies

Introduction

- Mostly Node.js is used with NoSQL databases like MongoDB
- Can be used with traditional DBMS like mysql

Example

```
Var require('mysql');
var sql ='SELECT * FROM books';



Var con = mysql.createConnection({
Host :"localhost",
user:"root",
password:"",

});


con.connect(function(err)  {
     If (err) throw err; console.log("Connected!" ,
     con.query(sql, function (err, result

       If (err) throw err; console.log("Result:" + result);

       });

       });
```

Angular

TOPIC 134

Web Systems and Technologies

Introduction

- Angular is a popular browser-based, open-source JavaScript MVC framework (Google driven)
- It is the A' in the MEAN stack, independent of the other components of the stack, and can be used without any of them
- Angular allows developers to write in TypeScript, JavaScript, or Dart
- AngularJS uses JavaScript
- Well suited for single page applications

Angular

Creating Simple AngularJS Application

Angular

Creating a Simple AngularJS Application

A directive for designating the root AngularJS element

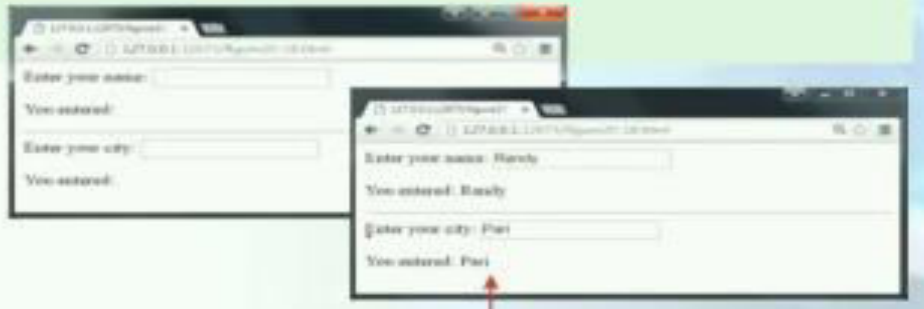
```
<html ng-app>
<head>
<title>Chapter 20</title>
<script src="https://code.angularjs.org/1.5.0/angular.min.js" >
</script>
</head>
<body>
  Enter your name: <input type="text" ng-model="name" />
  <p>You entered: {{ name }}  </p>

  <hr>
  Enter your city: <input type="text" ng-model="city" />
  <p>You entered: {{ city }}  </p>
</body>
</html>
```

A template

A directive for saving the field value in the Model

A data binding expression



Appears as user types into textbox

A Controller

Angular

A Controller

Now this directive is specifying the *module* used in the application

```
<html ng-app="demo">
...   This element is going to use a controller to get its data
<body ng-controller="myController">
    <div id="search">                Save the user's input in a model property named search
      City Search: <input type="text" ng-model="search" />
    </div>
    <table>  A directive to loop through a collection named cities (which is defined in the controller)
      <tr ng-repeat="city in cities | filter:search | orderBy: 'name'">
        <td>{{city.name }}</td>
        <td>{{city.country}}</td>
      </tr>
    </table>
</body>
</html>
```

Data bind to values in the collection

Uses *filters* to alter how this element works. In this example, the `filter` filter and the `orderBy` filter are used to modify how the `ng-repeat` works. Here the search refers to data item in the model.

A module is an AngularJS container for the different components used in the application

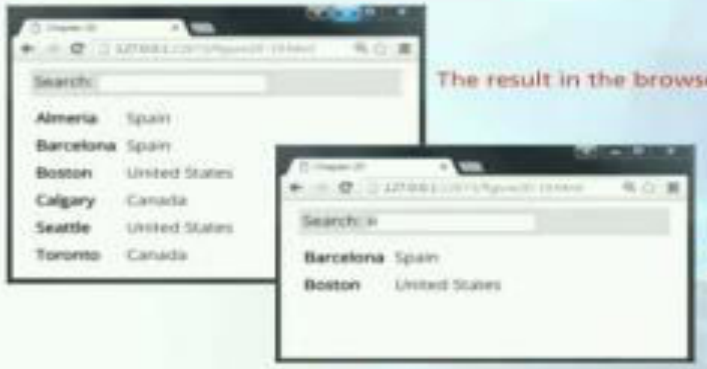
```
var myapp = angular.module('demo',[]);
```

Add a controller to the module named myController

The $scope variable is passed (injected into) the controller by AngularJS

```
myapp.controller('myController', function ($scope) {
    $scope.cities = [{name: 'Calgary', country: 'Canada'},
                     {name: 'Toronto', country: 'Canada'},
                     {name: 'Boston', country: 'United States'},
                     {name: 'Seattle', country: 'United States'},
                     {name: 'Almeria', country: 'Spain'},
                     {name: 'Barcelona', country: 'Spain'}];
});
```

The $scope variable is used to store the model (data). Here we are defining an array of object literals named `cities`



The result in the browser (notice the sort order)

The `filter` filter alters the displayed cities based on the current value of the Search text field.

Getting Data from a Web Service

```
var myapp = angular.module('demo',[ ]),
  //notice that our callback function is passed the $http parameter.
  //parameters are injected by the AngularJS system "behind-the-scenes"
myapp.controller('myController',function (6scope, $http)

  //retrieve country data from web service
  Var url  =' http://randyoonnolly.com/funwebdev/services/travel/countries.php†continent=EU;'

$http.get(url).then(function (response)
$scope.countries= response.data
});
});
```

TOPIC 134

Web sockets

Web Systems and Technologies

Introduction

- Web socket is a full duplex communication protocol
- Enables interaction between web browser and web server

Server Side Example (Node.js)

Server Side Example (Node.js)

```
const express = require('express');
const app = express();
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server);

io.on('connection', (socket) => {
  console.log('A user connected');});

io.on('message', function(message) {
  console.log('Message from client: '+message);
  io.emit('Message received '+message});

server.listen(7000, () => {
console.log('localhost:7000');});
```

Client Side Example (JavaScript)

Client Side Example (JavaScript)

```
var client = new WebSocket("http://localhost:7000");

//sending message on socket open
client.onopen = function() {
  client.send("Message sent to the server");
}

//receiving message
client.onmessage = function (event) {
  console.log(event.data);
}

//close connection
client.close();
```

TOPIC 135

Web server Hosting

Web Systems and Technologies

Hosting Types

- Shared Hosting
  - Simple
  - Virtualized
- Dedicated
- Collocated
- In-house
- Cloud

Web Systems and Technologies

WEEK 16

TOPIC 136 TO 144

Web Systems and Technologies

- 136 Domain and Name Server Administration
- 137Search Engines
- 138Page Rank
- 139Search Engine Optimization
- 140Understanding Bad SEO
- 141Introduction to Online Social Networks
- 142Monetization with Ads
- 143Web Analytics
- 144 Google Analytics

Web Systems and Technologies

TOPIC 136

Domain and Name Server Administration

Domain and Name Server Administration

Introduction

- Domain Name System (DNS) is used to resolve an IP address of a website
- Domain name registrars register website
  - GoDaddy
  - TuCows
  - Network Solutions

Domain and Name Server Administration

DNS

- Registration
  - WHOIS
    - Use whois command
  - Private registration
- Updating name servers
  - Takes up to 48 hours
  - Check status using dig command
  - https://toolbox.googleapps.com/apps/dig
- Reverse DNS

Domain and Name Server Administration

DNS Record Types

- Mapping (A/AAA)
- Mail Exchange (MX)
- Authoritative (NS/SOA)
- Validation (TXT/SPF)

Web Systems and Technologies

TOPIC 137

Search Engines

Search Engines

Introduction

- Before the search engines
  - Emails, message boards, websites
  - Web directories (vlib, yahoo, dmoz)
- Web Crawlers
- Google

Search Engines

Search Engine Overview

- Components of a search engine
  - Web Crawler, URL submission, …
  - Indexes
  - Query Server

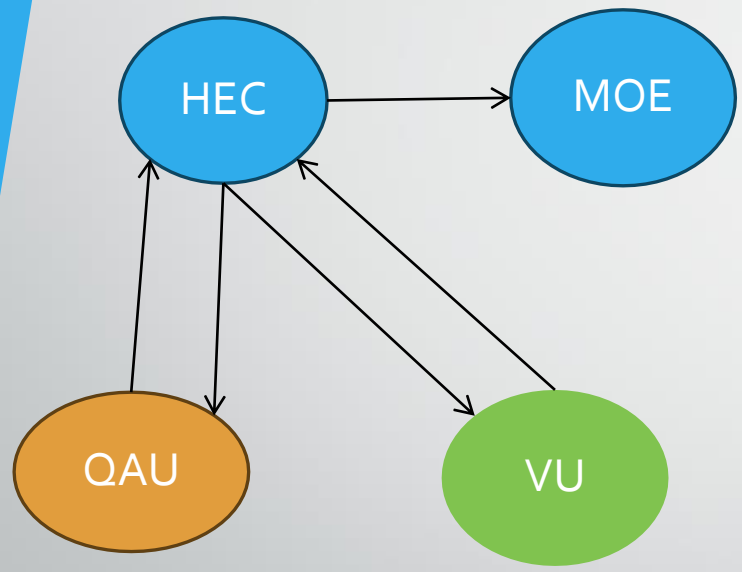
Web Systems and Technologies

TOPIC 138

PageRank

PageRank

- PageRank



- An algorithm for ranking websites
- Models WWW as a graph, example:
  - On VU website

    HEC
  - On QAU website

    HEC
  - On HEC's website

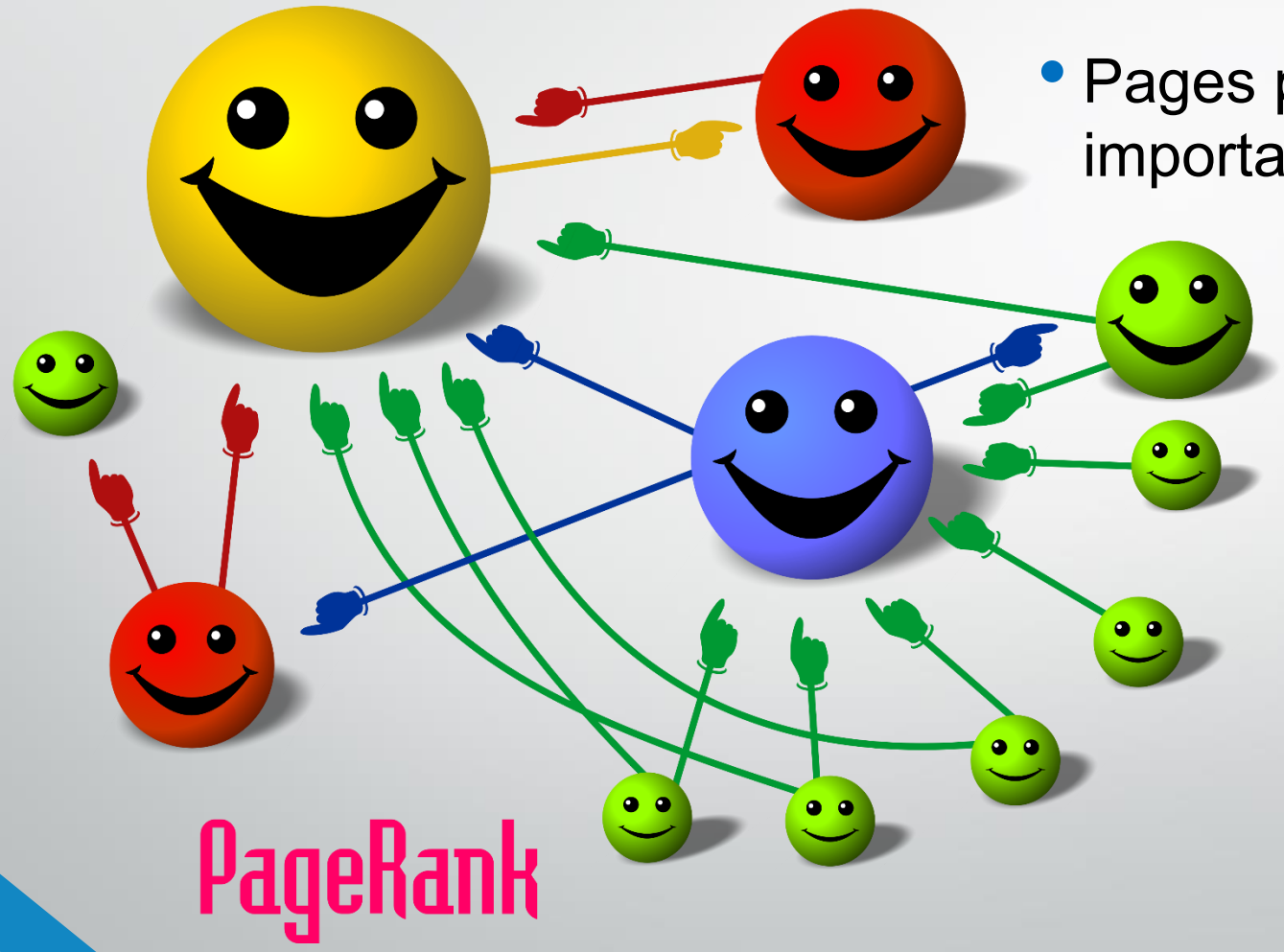
    VU

    QAU

    MOE

PageRank

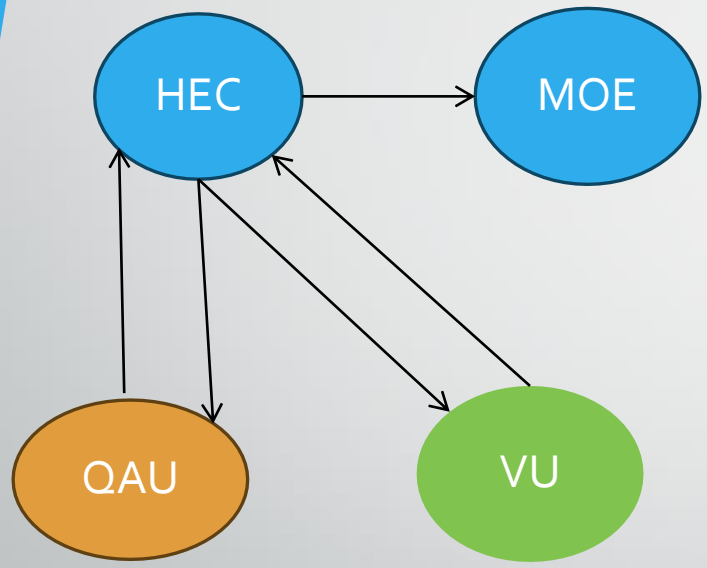
PageRank



- Pages pointed by important pages are important

PageRank

How PageRank Works



	HEC	VU	QAU	MOE
Iter 0	¼	¼	¼	1/4
Iter 1	¼+ ¼=1/2	1/12	1/12	1/12
Iter 2				
…				

Web Systems and Technologies

TOPIC 139

Search Engine Optimization

Search Engine Optimization

Introduction

- Search Engine Optimization(SEO) is a way to increase the rank of a website
  - White hat SEO
  - Black hat SEO

Search Engine Optimization

White-Hat SEO Techniques

- Title and Meta-data
  - Title, description, robot

/users/id/123

- Designing URLs
  - Descriptive paths and file names
- Website Design
  - Layout, readability, JavaScript, structure
- Sitemaps
- Anchors and Images
  - Anchor text, image name, alt
- Contents
  - Original, fresh, UGCs

Web Systems and Technologies

TOPIC 140

Black Hat SEO

Search Engines

Introduction

- Try to exploit search engine rankings
- Should not be used

Search Engines

Black-Hat SEO Techniques

- Content spamming
  - Keyword stuffing, hidden content, paid links, doorway pages
- Link spam
  - Hidden links, comment spam, link farms, and pyramids, google bombing
- Google bowling
- Cloaking
- Duplicate Contents

Web Systems and Technologies

TOPIC 141

Online Social Networks

Online social Networks

Introduction

- Social networks
- Free usage
- User profile
- Monetization
- Extensive API support

Online social Networks

Social Media Presence

- Home pages on social media platforms
- Link and logos
- Using URL shorteners

Online social Networks

Facebook Plugins

- Register as a developer
- Facebook Markup Language(FBML)
- Like and recommend buttons

```
<div class="fb-like"
    data-href="https://example.com/page.html"
    data-width="200"
    data-layout="standard"  data-action="like"
    data-size="small"
    data-share="true">
</div>
```

- More plugins

  https://devolopers.facebook.com/docs/plugins

Online social Networks

Twitter Widgets

- Twitter
- Widgets
  - Tweet this button

```
<a class="twitter-share-button"
href=https://twitter.com/intent/tweet?text=Hello World!">Tweet</a>
<script async
src=https://platform.twitter.com/widgets.js  charset="utf-8"></script>
```

- More widgets

  https://publish.twitter.com

Web Systems and Technologies

TOPIC 142

Monetization with Ads

Monetization with Ads

Introduction

- Ads on a website can generate revenue
- How web ads work?
- Ad Networks

Monetization with Ads

Ad Types

- Ad types
  - Graphics
  - Text
  - Dynamic

Monetization with Ads

Creating Ads

- Register for a Google AdSense (or other account)
- Get code through AdSense account
- Place it on your website

Monetization with Ads

Web Ads Economy

- Ad Commodities
  - Views
  - Clicks
  - Actions
- Web Commodity markets
  - Cost Per Click(CPC)
  - Cost Per Mile (CPM)
  - Cost Per Action(CPA)
  - ClickThrough Rate (CTR)
- Auctions

Web Systems and Technologies

TOPIC 143

Web Analytics

Web Analytics

Introduction

- Web analytics provide us insights about website usage
  - Bandwidth usage
  - Referrers
  - Popular domains/page
  - Search terms used for landing
  - User behavior
  - Demographics

Web Analytics

Web Metrics

- Page Views
- Unique page views
- Average visit duration
- Bounce rate

Web Analytics

Internal Analytics

- IP address
- User agent analysis
- Frequency analysis
- Can be provided by hosting company

Web Analytics

Third Party Analytics

- Google, Bing, …
- Dashboards
- Flow analysis
- In-page analytics

Web Systems and Technologies

TOPIC 144

Google Analytics-Demo

Prepared by DL...
May You All Stay Success in your Whole Life, Ameen! Summ Ameen... Just Remember us in your Deep Prayers. JazakAllah...