



CS521

WEB SYSTEMS AND TECHNOLOGIES

SLIDES WEEK [1 TO 6]

MADE BY MAH JABEEN

0321 2711298

mahjabeen97869@gmail.com



Introduction to Web Systems and Technologies

TOPIC 01
MODULE 01



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



Web Systems and Technologies

TOPIC 02

Ecosystem of Web

Ecosystem of Web

A digital ecosystem is a group of connected technology resources that work together as a single system. It includes suppliers, customers, business partners, apps, third-party data services, and the technologies that link them all.

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

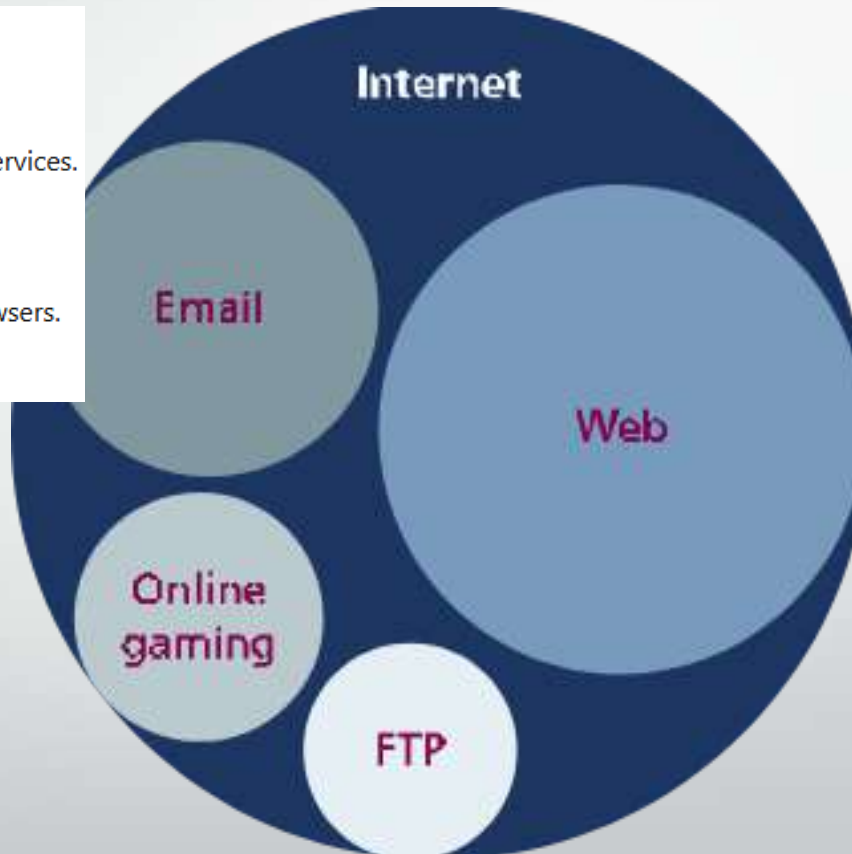
Internet vs. (World Wide) Web

Internet:

- A global network of interconnected computers and servers.
- It allows communication, data sharing, and access to various services.

World Wide Web (Web):

- A collection of websites and web pages accessed through browsers.
- It is one of the services that operates over the Internet.



Ecosystem of Web

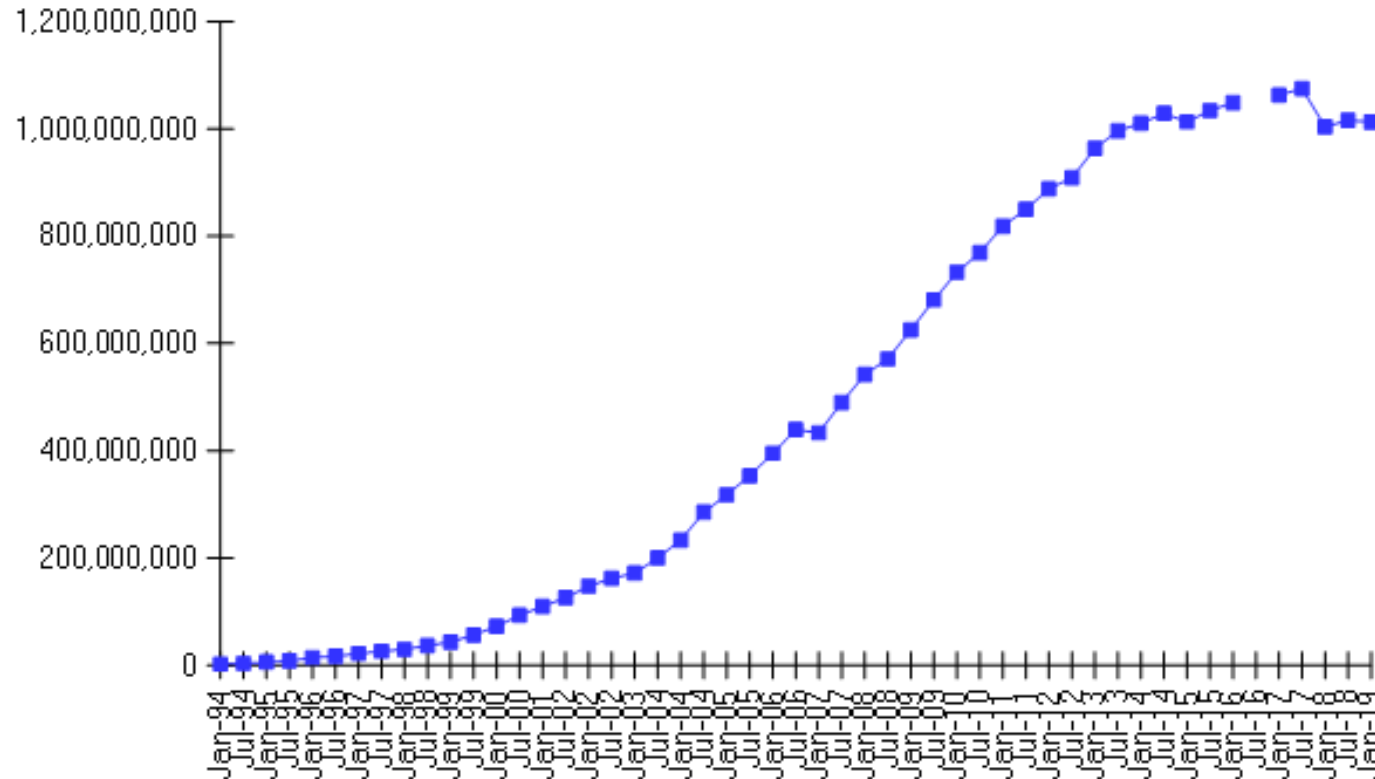
The Birth of the Web (1990)

- URLs
- HTTP
- SERVERS
- BROWSERS
- HTML

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

Desktop Applications vs Web Applications

Feature	Desktop Applications	Web Applications
Installation	Installed directly on your computer.	No installation required; accessed through a browser.
Internet Requirement	Mostly works offline.	Requires an internet connection to work.
Updates	Updates need to be manually downloaded or installed.	Automatically updated without user action.
Performance	Generally faster because it uses local resources.	May be slower, depending on internet speed and server load.
Accessibility	Only accessible from the computer it's installed on.	Accessible from any device with a browser and internet.
Platform Dependencv	Can be platform-specific (Windows, Mac, etc.).	Works across multiple platforms (Windows, Mac, Linux, etc.) as long as there's a browser.

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

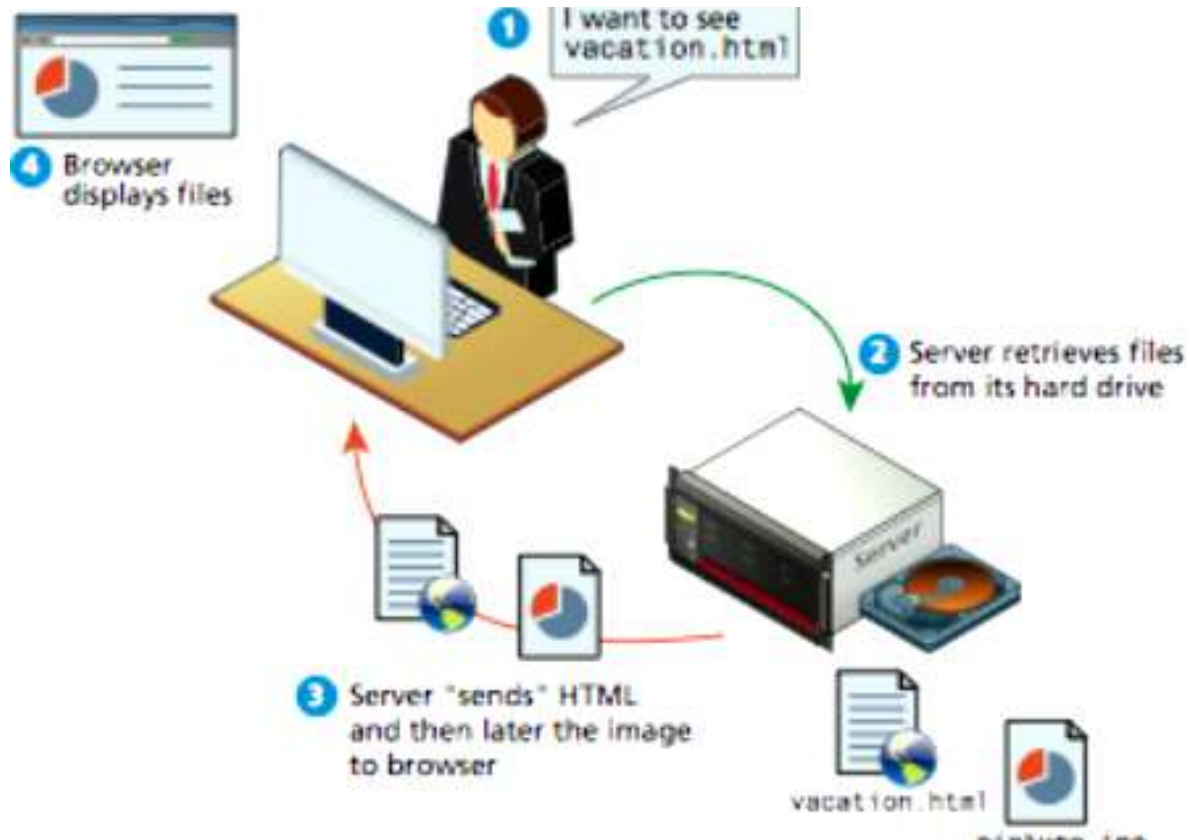
Web Systems and Technologies

Static websites contain fixed content stored on the server and delivered to the user without any server-side processing. These websites are simple and easy to host, but they lack interactivity and require manual updates. On the other hand, dynamic websites generate content on the fly using server-side processing and databases. They offer interactivity, personalized content, and dynamic features but are more complex to develop and maintain.

Static vs Dynamic Websites

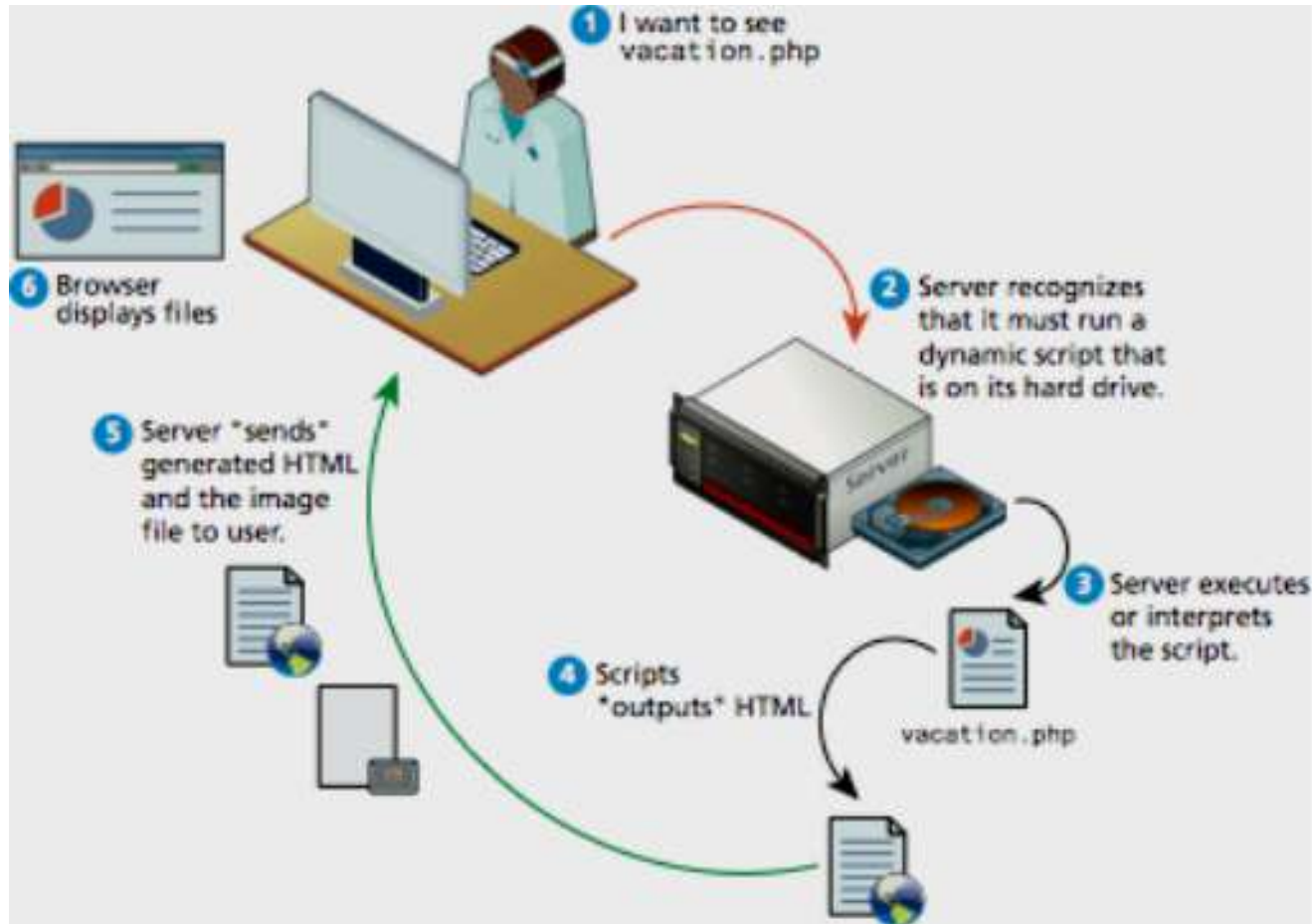
Static vs Dynamic Websites

Static websites



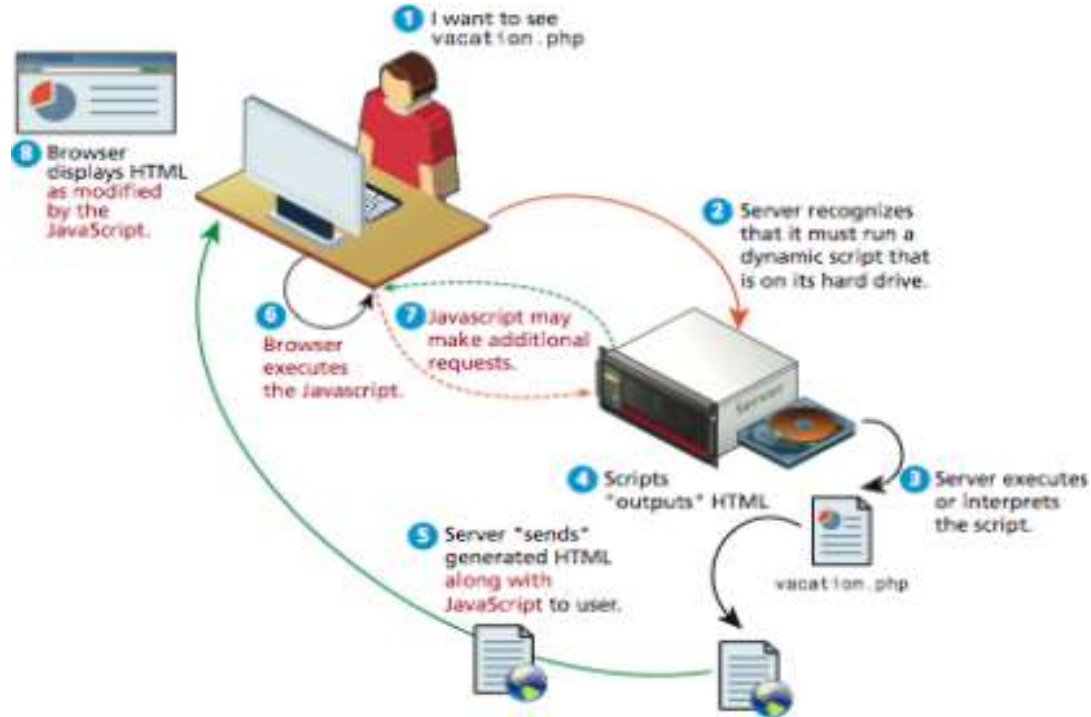
Static vs Dynamic Websites

(Traditional) Dynamic Websites



Static vs Dynamic Websites

(Modern) Dynamic Websites





Web Systems and Technologies

TOPIC 04

MODULE 04



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

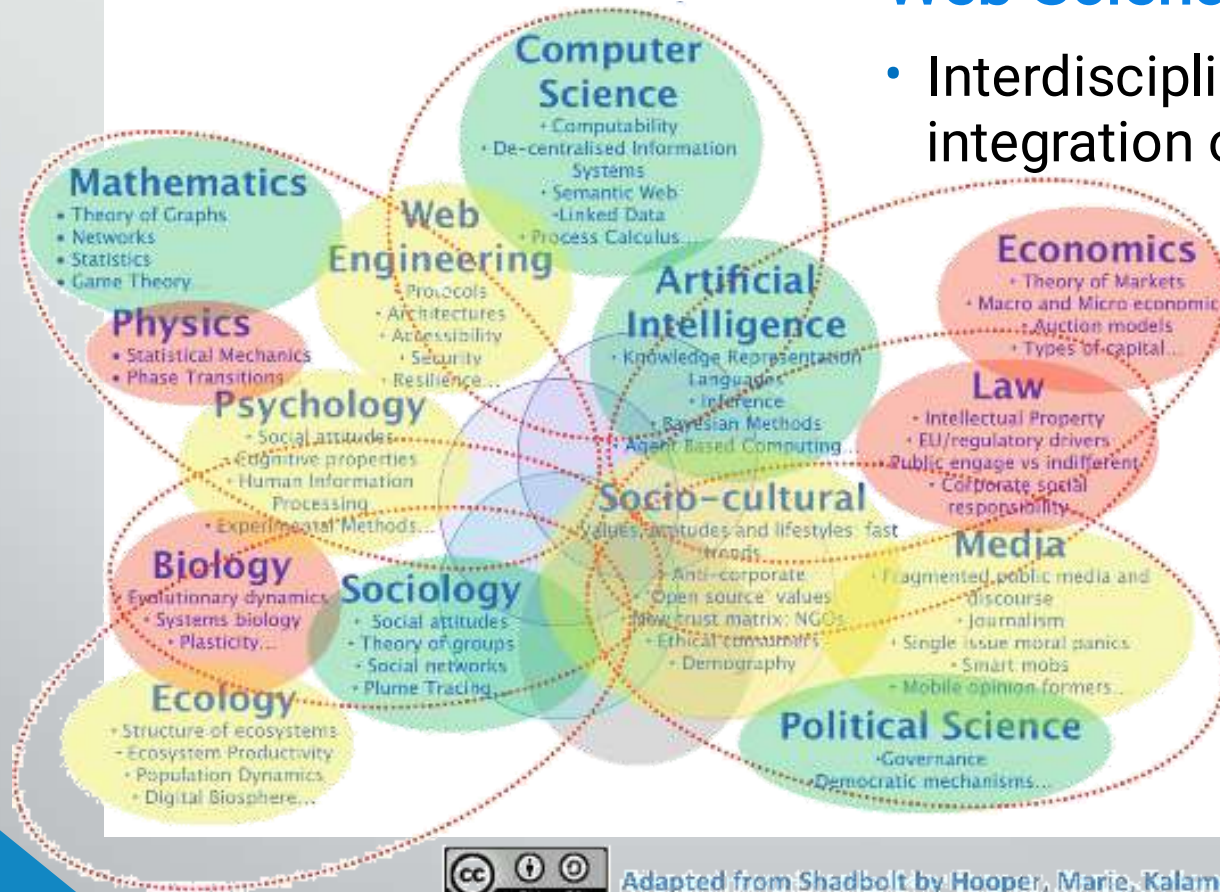
Web 3.0

- 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

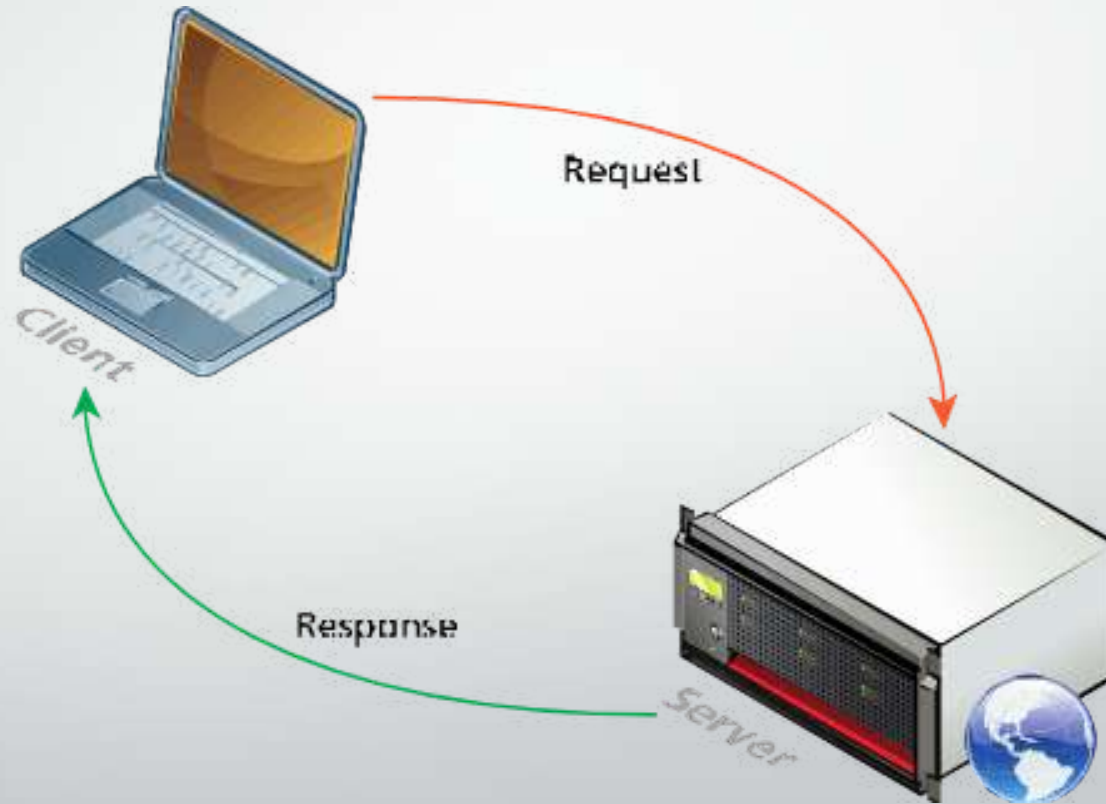


Web Systems and Technologies

Client Server Model

Client Server Model

The request-response Loop

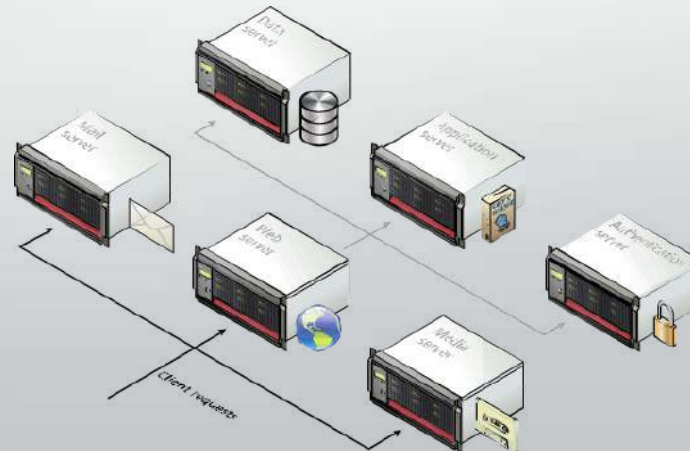


Client Server Model

Types of Servers

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

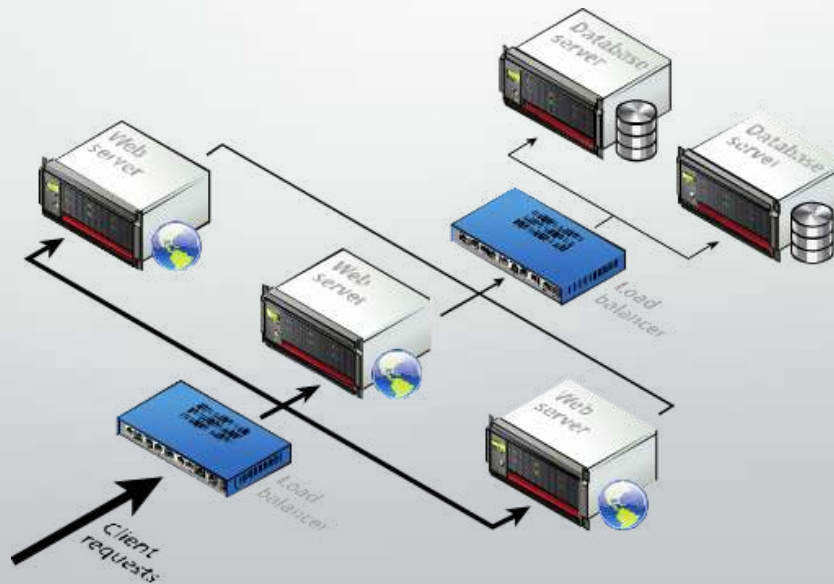
1. **Web Servers:** Handle requests from browsers and serve web pages (e.g., Apache, Nginx).
2. **Application Servers:** Run and manage web applications, processing business logic and dynamic content (e.g., Apache Tomcat, JBoss).
3. **Database Servers:** Store and manage data, handling queries from applications (e.g., MySQL, Oracle).
4. **Mail Servers:** Send, receive, and store email messages (e.g., Microsoft Exchange, Postfix).
5. **Media Servers:** Store and stream media content like videos and music (e.g., Plex, Jellyfin).
6. **Authentication Servers:** Verify user credentials before granting access to resources (e.g., RADIUS, LDAP).



Client Server Model

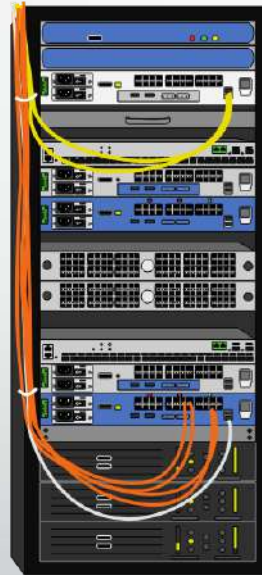
Server Farms

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



Client Server Model

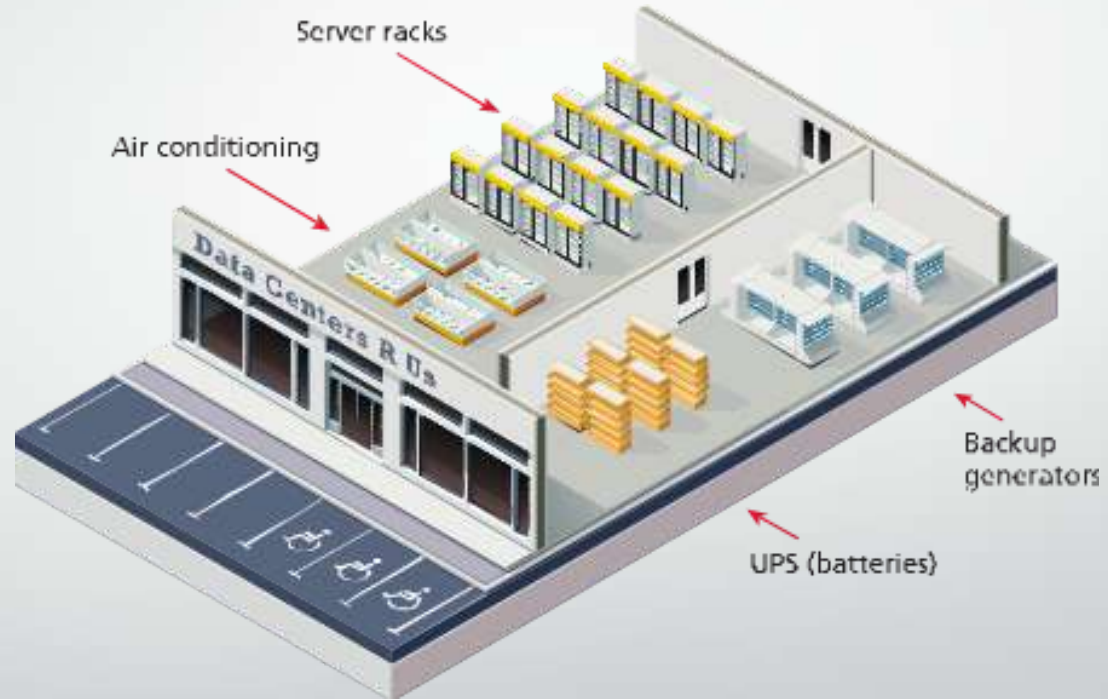
Server Racks



- Fiber channel switches
- Rack management server
- Test server
- Keyboard tray and flip-up monitor
- Patch panel
- Production web server
- Production data server
- RAID HD arrays
- Patch panel
- Production web server
- Production data server
- Batteries and UPS

Client Server Model

Data Centers



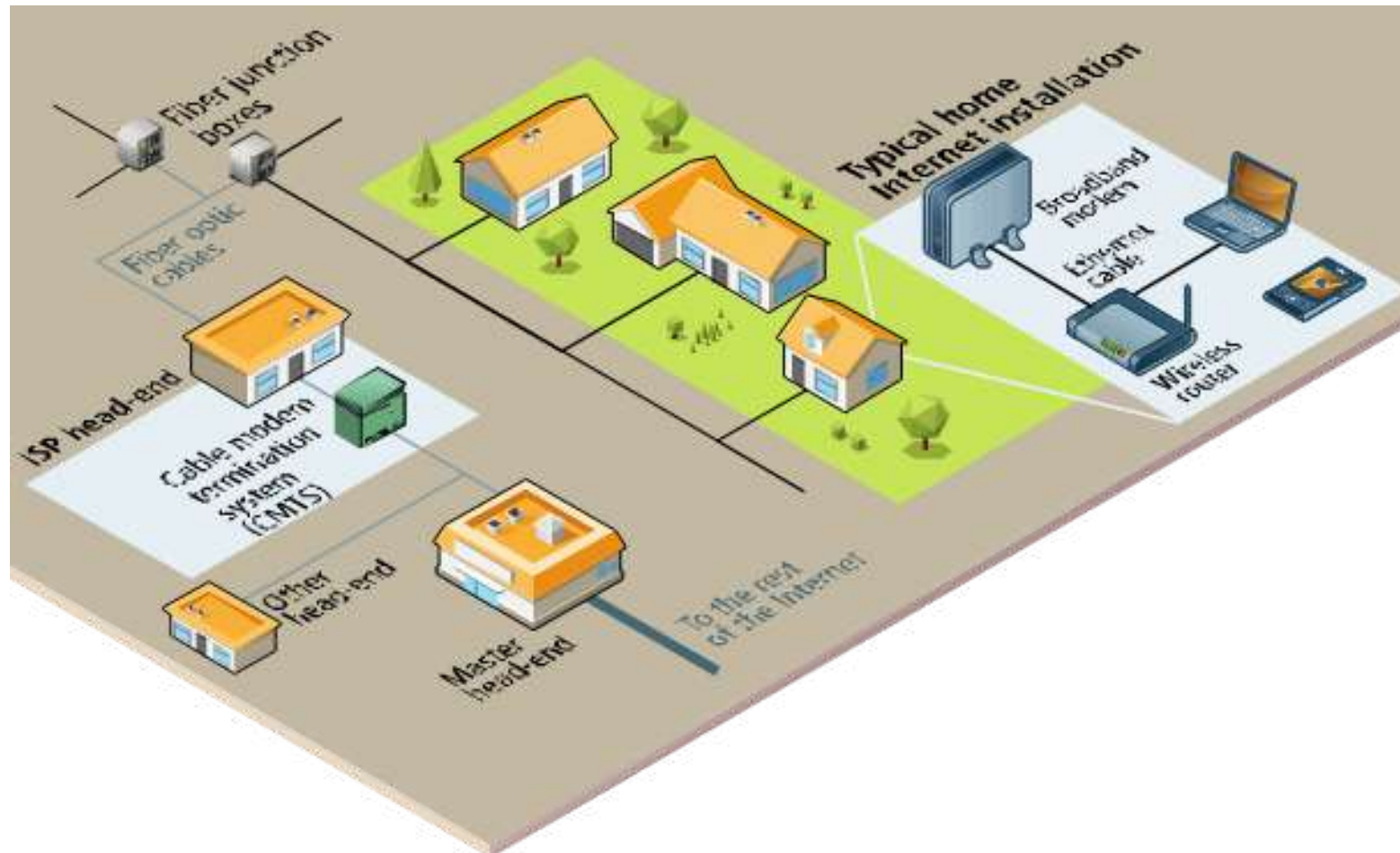


Web System and Technologies CS 521

TOPIC 06

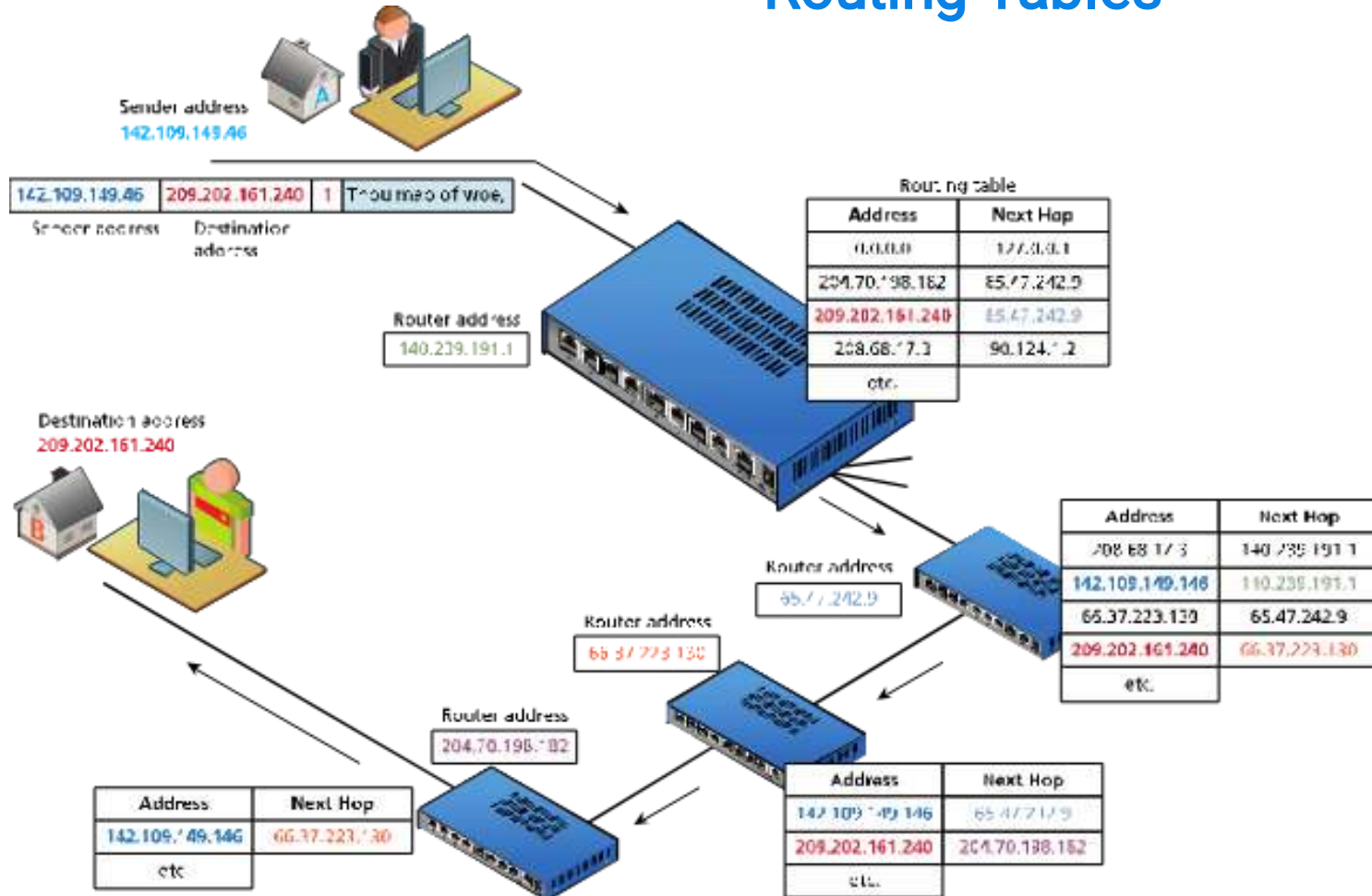
How Internet Works?

Internet from Home to ISP



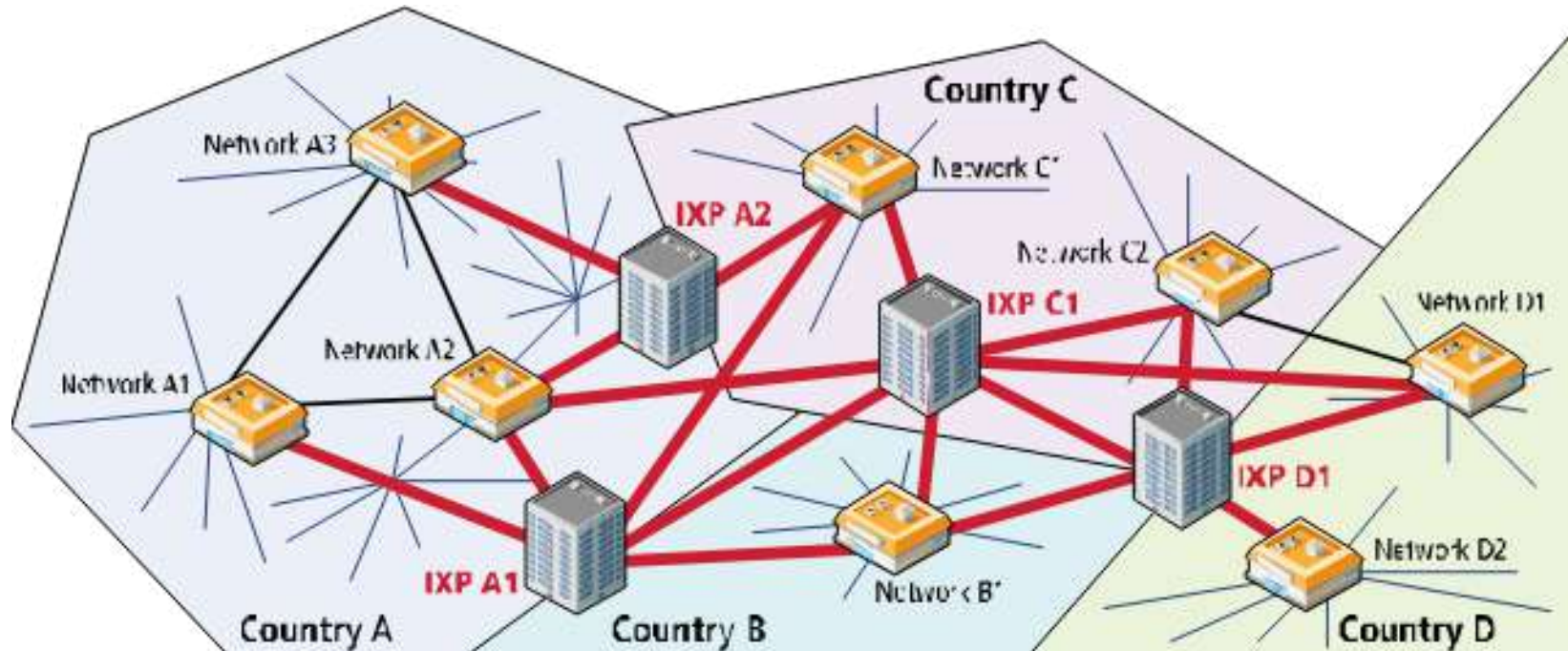
How Internet Works?

Routing Tables



How Internet Works?

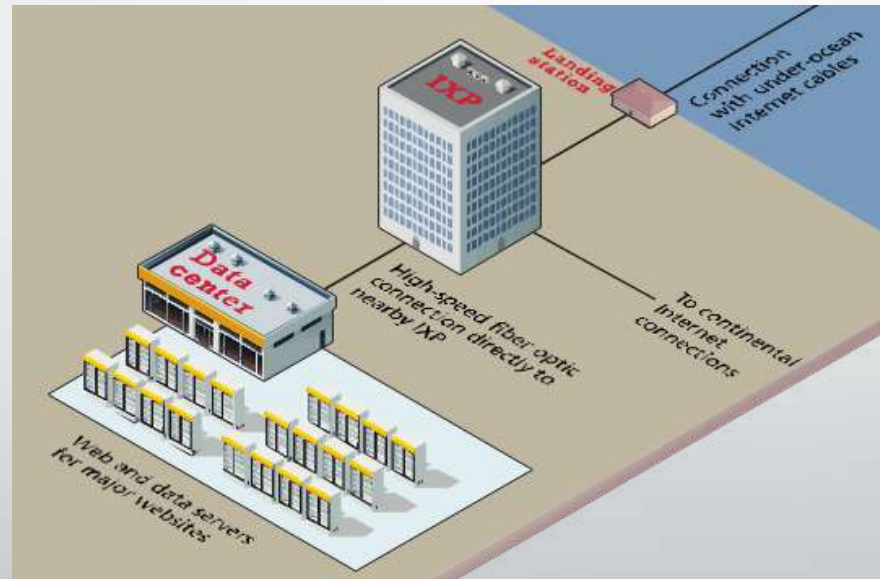
From ISP to the World



How Internet Works?

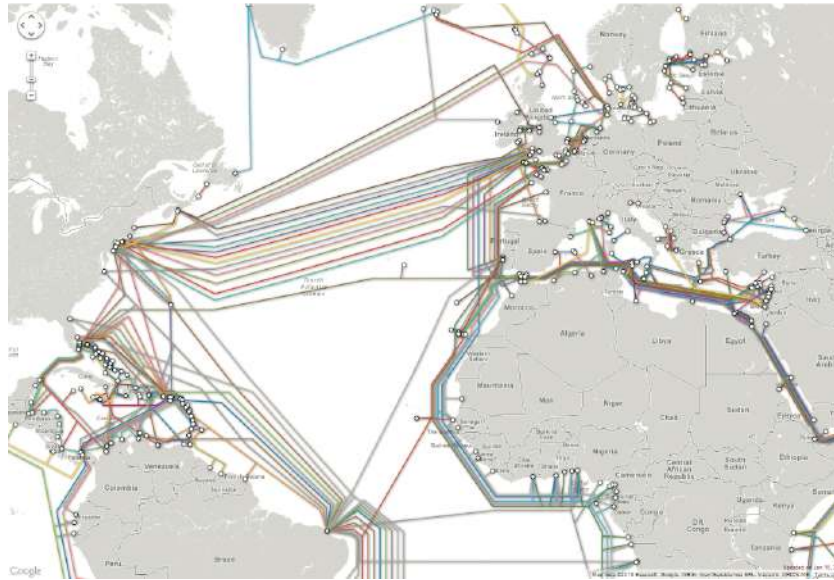
IXP

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



How Internet Works?

Under-ocean Connectivity





The End



Web System and Technologies CS 521

Topic 07

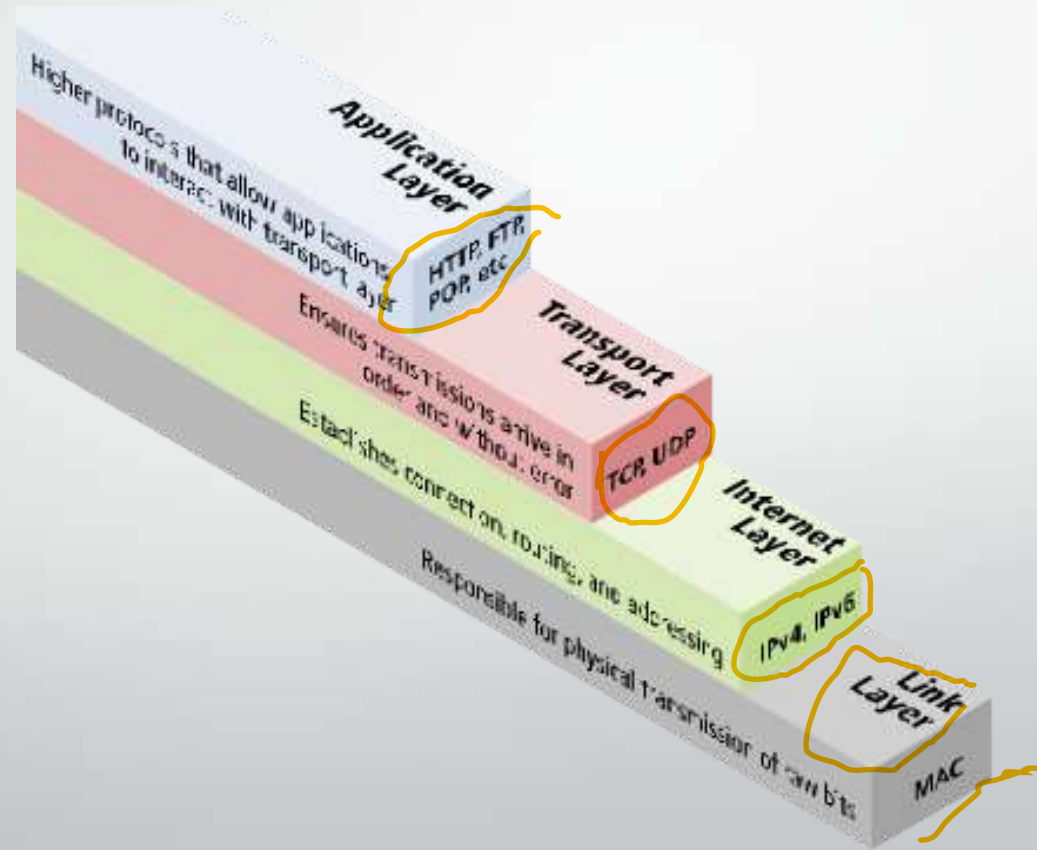
Internet Protocols

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

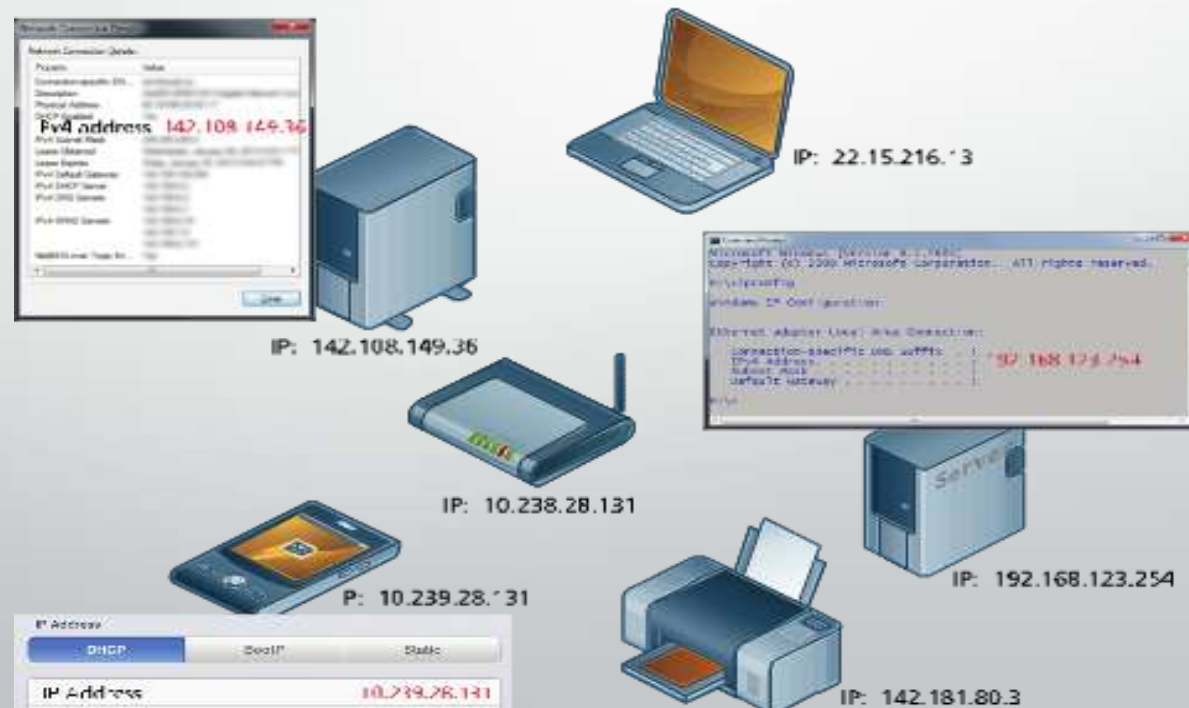
Link Layer

- 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:

IPv4
 2^{32} addresses

4-8 bit components
(32 bits)

192 . 68 . 123 . 254

IPv6
 2^{128} addresses

8 16 bit components
(128 bits)

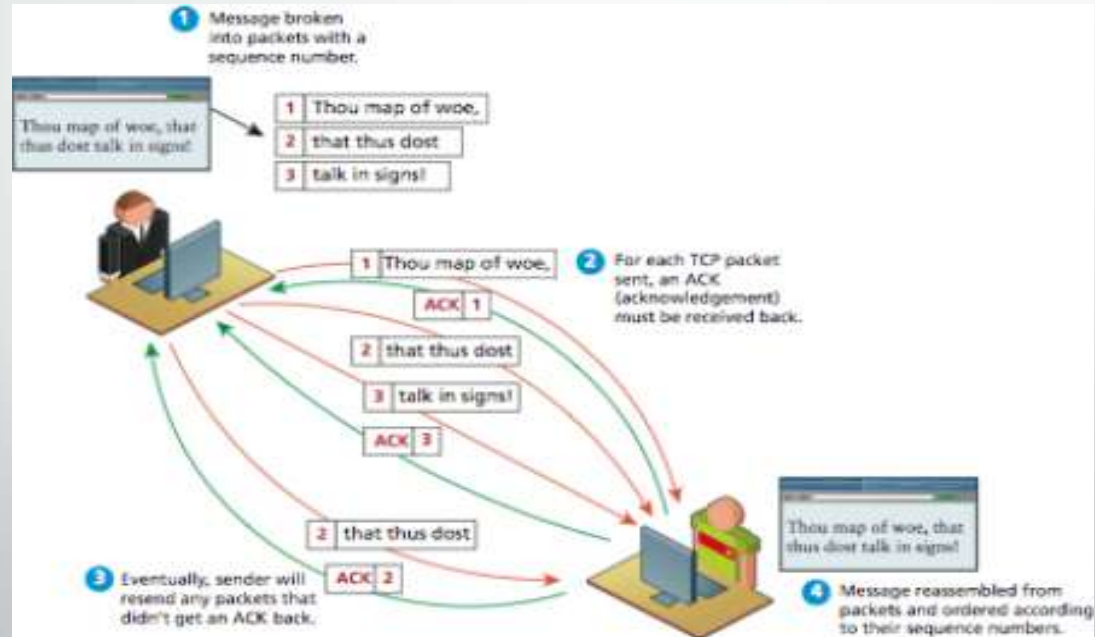
3fae : 7a10 : 4545 : 9 : 291 : e8ff : fe21 : 37ca

Internet Protocols

Transport Layer:

- Ensures transmissions arrive in order and without error

TCP is slower but ensures reliable, accurate data delivery. UDP is faster but may lose or misorder some data.



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

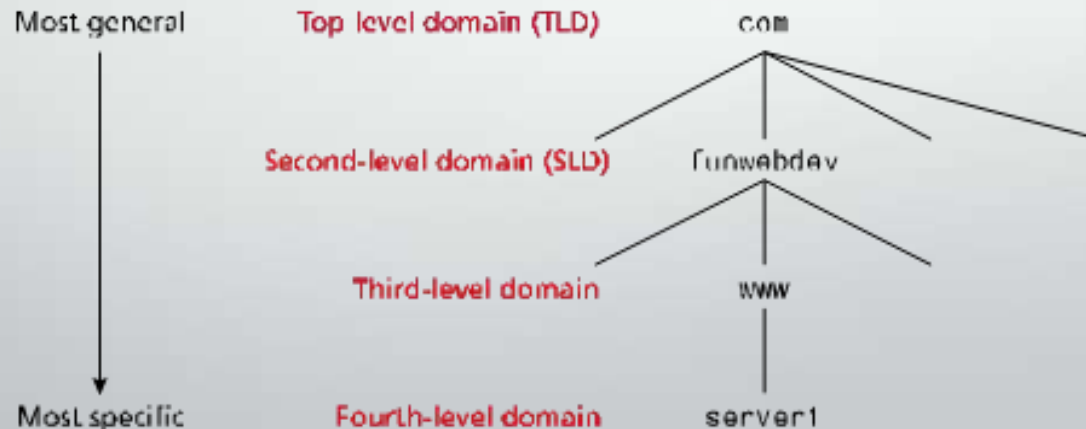
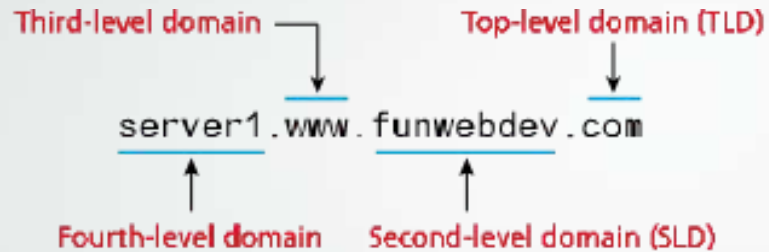


Web System and Technologies CS 521

TOPIC 08

Domain Name System

Name Levels in a Domain:



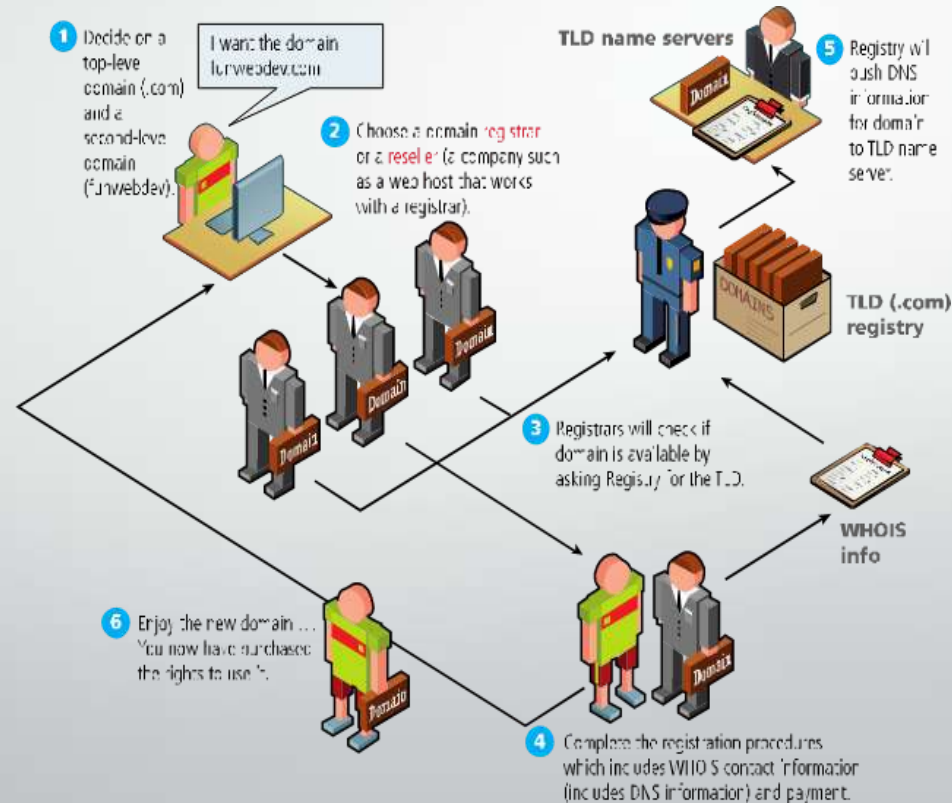
Domain Name System

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

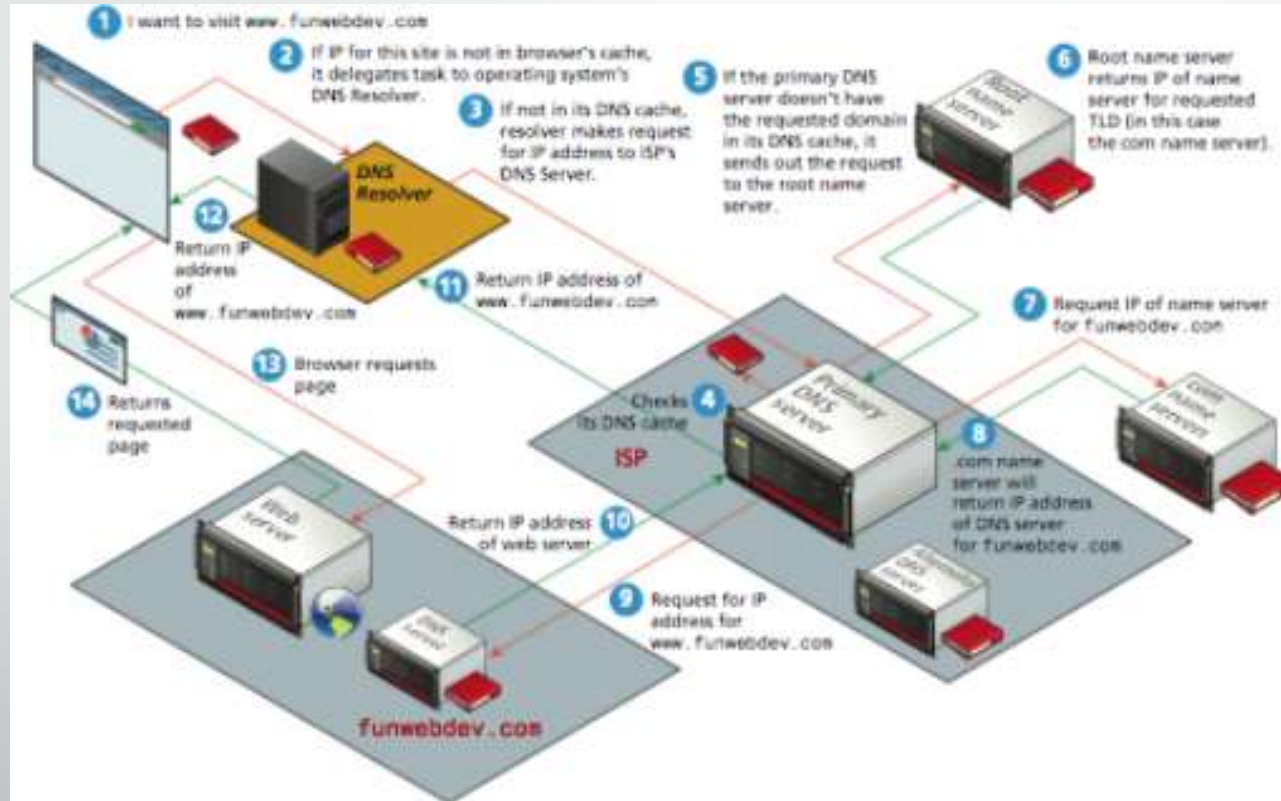
Domain Name System

Domain Registration:



Domain Name System

Address Resolution:





The End



Web System and Technologies CS 521

TOPIC 09

Uniform Resource Locators

Components of a URL:

- Protocol
- Domain
- Path
- Query String
- Fragment

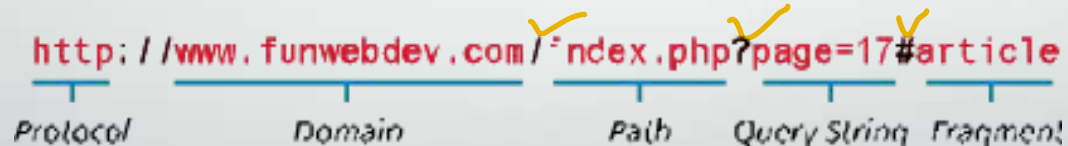
Uniform Resource Locators

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

`http://www.funwebdev.com/index.php?page=17#article`

Protocol Domain Path Query String Fragment

A diagram illustrating the components of the URL 'http://www.funwebdev.com/index.php?page=17#article'. The URL is written in red text. Below it, five blue horizontal lines with vertical tick marks at the end point to the corresponding parts of the URL: 'Protocol' points to 'http:', 'Domain' points to 'www.funwebdev.com', 'Path' points to '/index.php', 'Query String' points to '?page=17', and 'Fragment' points to '#article'. Additionally, three yellow checkmarks are placed above the 'Domain', 'Path', and 'Query String' labels.

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

`http://www.funwebdev.com/index.php?page=17#article`

<code>http://</code>	<code>www.funwebdev.com</code>	<code>/index.php</code>	<code>?page=17</code>	<code>#article</code>
Protocol	Domain	Path	Query String	Fragment

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.

`http://www.funwebdev.com/index.php?page=17#article`

Protocol *Domain* *Path* *Query String* *Fragment*

Uniform Resource Locators

Path :

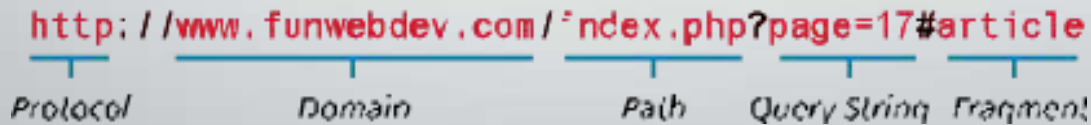
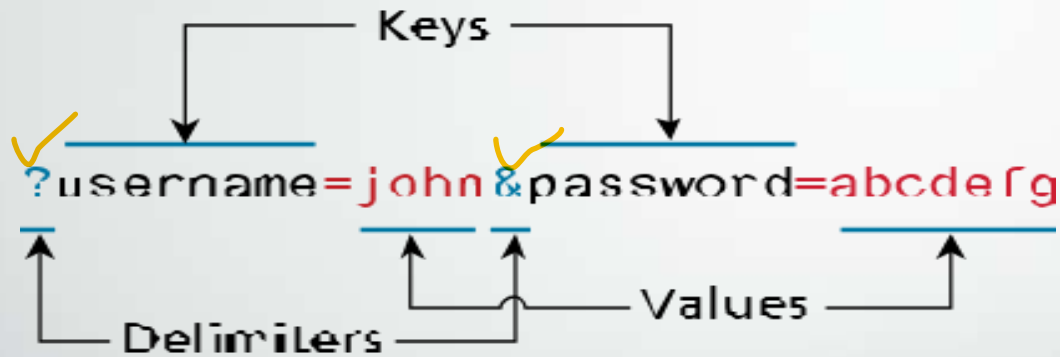
- Familiar concept to anyone who has ever used a computer file system.
- The root of a web server corresponds to a folder somewhere on that server.
 - On many Linux servers that path is /var/www/html/
 - On Windows IIS machines it is often c:/inetpub/wwwroot/
- The path is optional
 - If not provided,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

`http://www.funwebdev.com/index.php?page=17#article`

Protocol *Domain* *Path* *Query String* *Fragment*

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

`http://www.funwebdev.com/index.php?page=17#article`

Protocol *Domain* *Path* *Query String* *Fragment*



The End

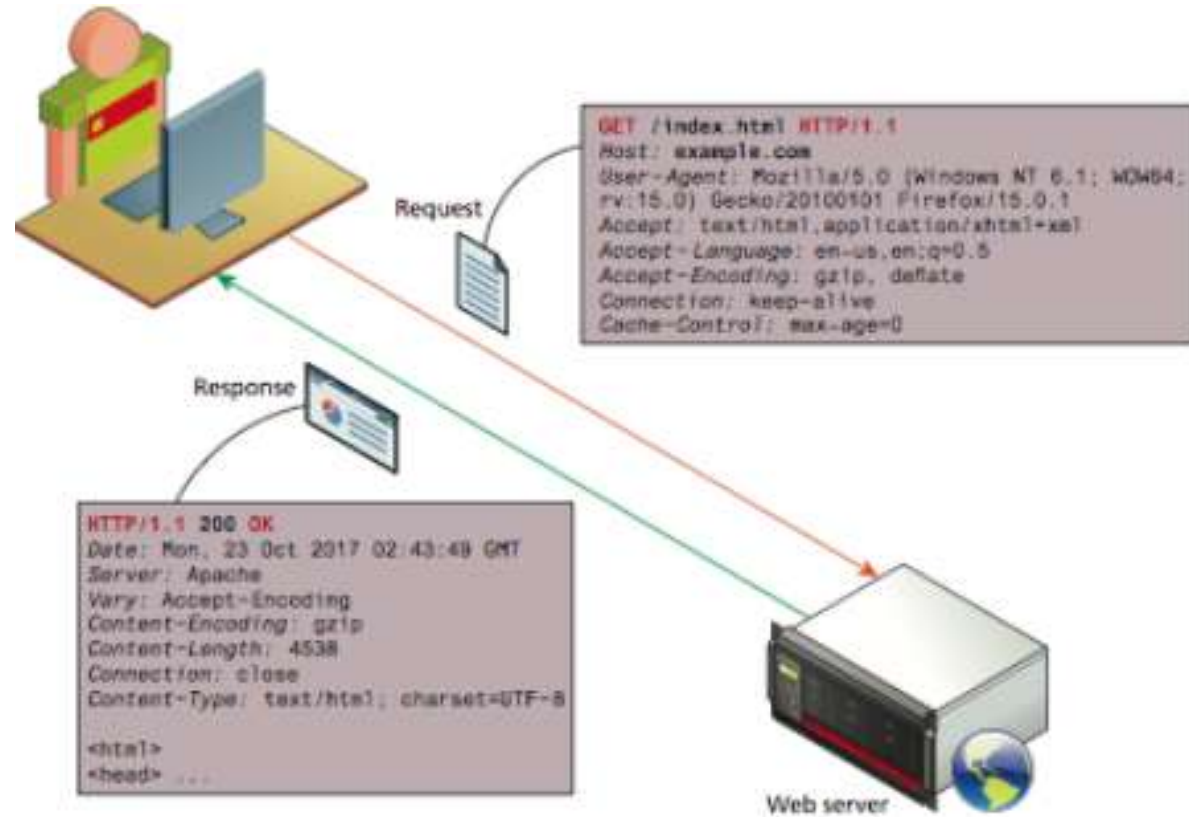


Web System and Technologies CS 521

TOPIC 10

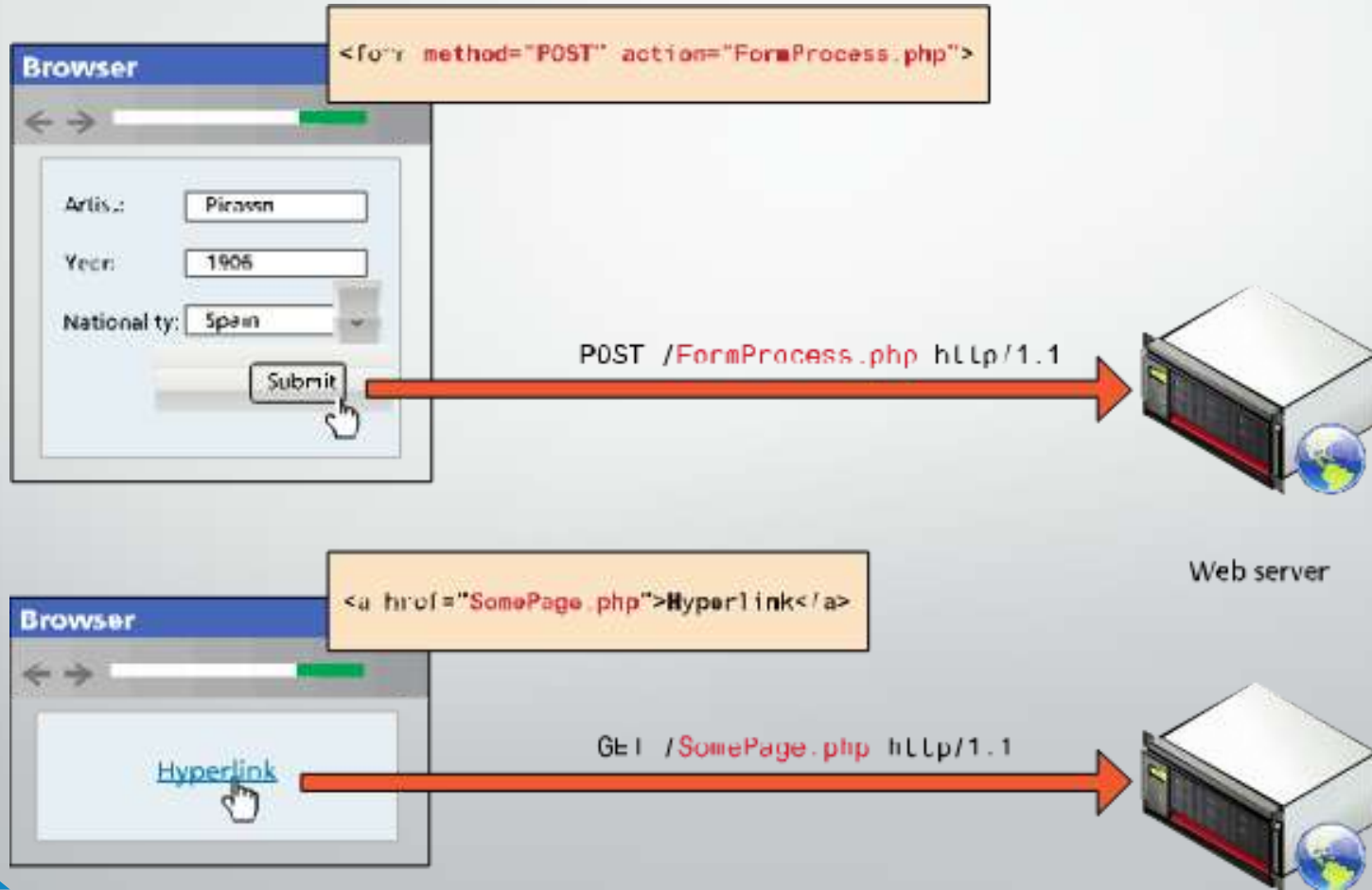
Hypertext Transfer Protocol (HTTP)

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



Web System and Technologies 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
 - Linux operating system,
 - Apache web server,
 - MySQL database, and
 - PHP 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



Web System and Technologies CS 521

TOPIC 12

Architecture of Web-based System

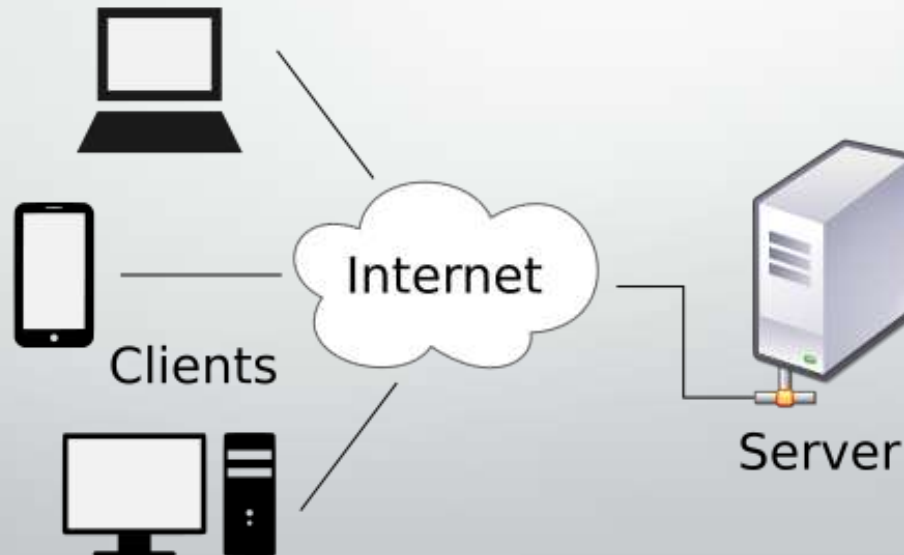
Multi-tier Architecture

- 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

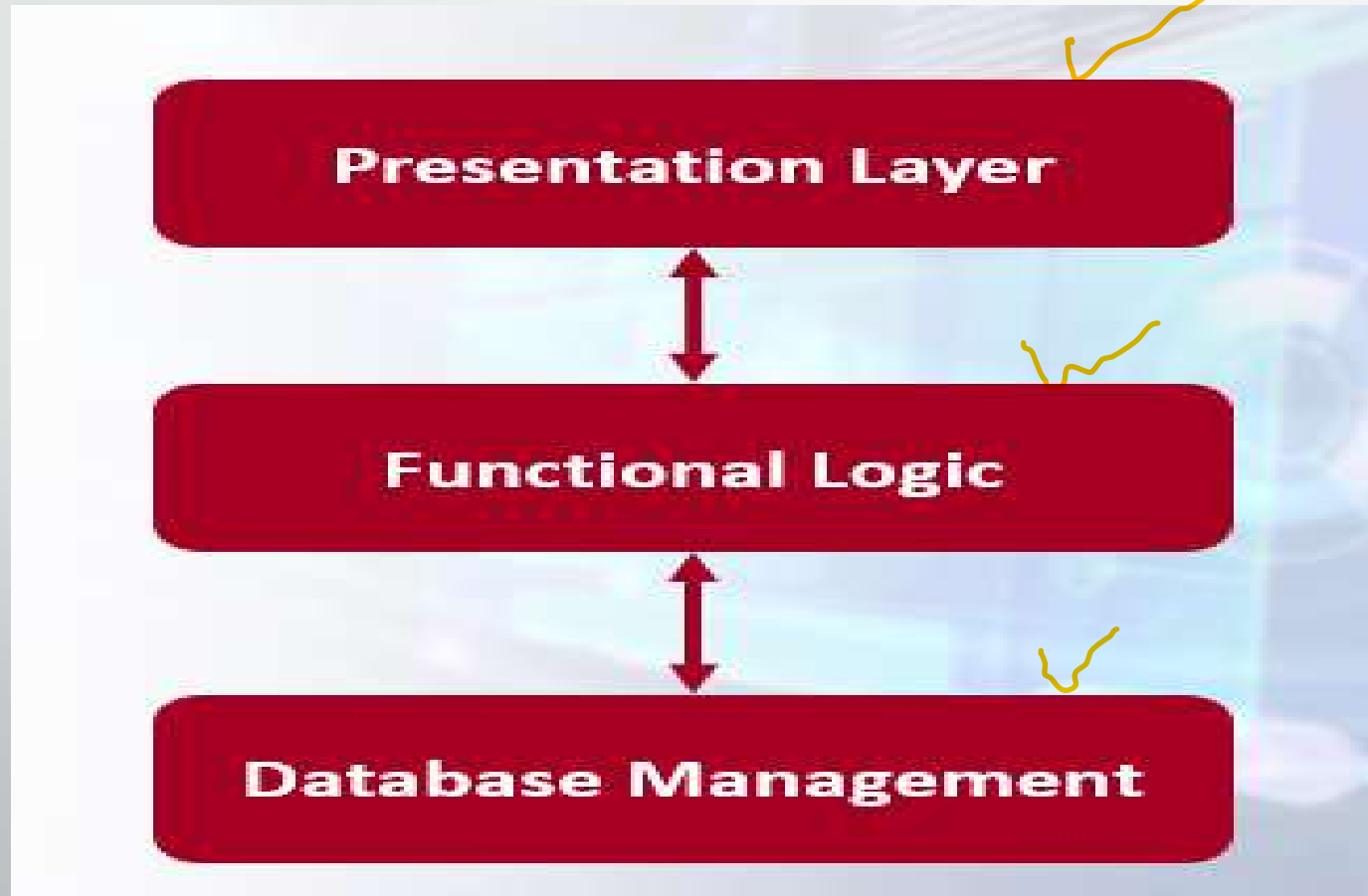


Credit: David Vignoni

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

Architecture of Web-based System

3-tier Architecture





The End



Web System and Technologies CS 521

TOPIC 13

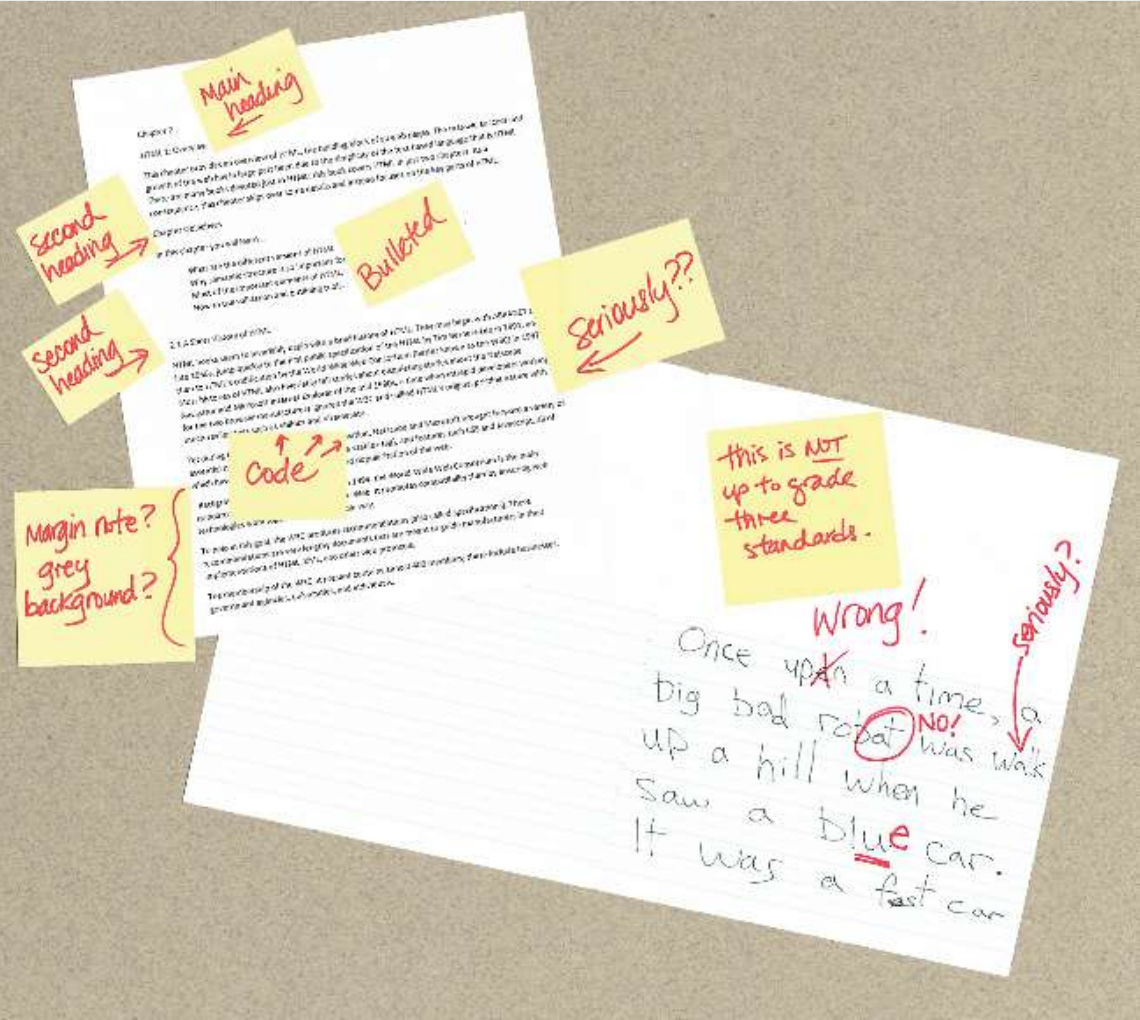
Introduction to HTML

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/>

The screenshot displays the W3C Markup Validation Service interface. The top section is titled "W3C Markup Validation Service" and includes navigation options: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" option is selected, and there is a text input field for the "Address" and a "Check" button. Below this, a paragraph explains the service's purpose: "This validator checks the markup validity of Web documents in HTML, XHTML, SVG, MathML, etc. If you wish to validate specific content and/or as HTML5, you should use the HTML5 validator, or the HTML5 and other validators and tools available. For an alternative you can also try our non-DTD-based validator." Below this is a "The W3C" section with a "VALIDATOR" button and links for "Home" and "About".

The bottom section is titled "Nu Html Checker" and shows the results of a validation. It includes a "Check Input" section with "Show" and "Check" buttons. The "Check by" dropdown is set to "relaxng". The main content area shows a list of errors, with the first one highlighted: "1. Error: An element must have an <alt> attribute, except under certain conditions. For details, consult the specification at https://www.w3.org/html/2014/07/html5-ls-changes/#attr-alt-required. From line 14, column 5 to line 14, column 40." Below the error list is a "Message Priority" section.

Two green arrows point from the text "Validator provides feedback on mark up's validity according to W3C specification" to the "Check" button and the first error message in the Nu Html Checker results.

Validator provides feedback on mark up's validity according to W3C specification

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

Nesting elements

- Correct nesting
 - Nested element must be closed first

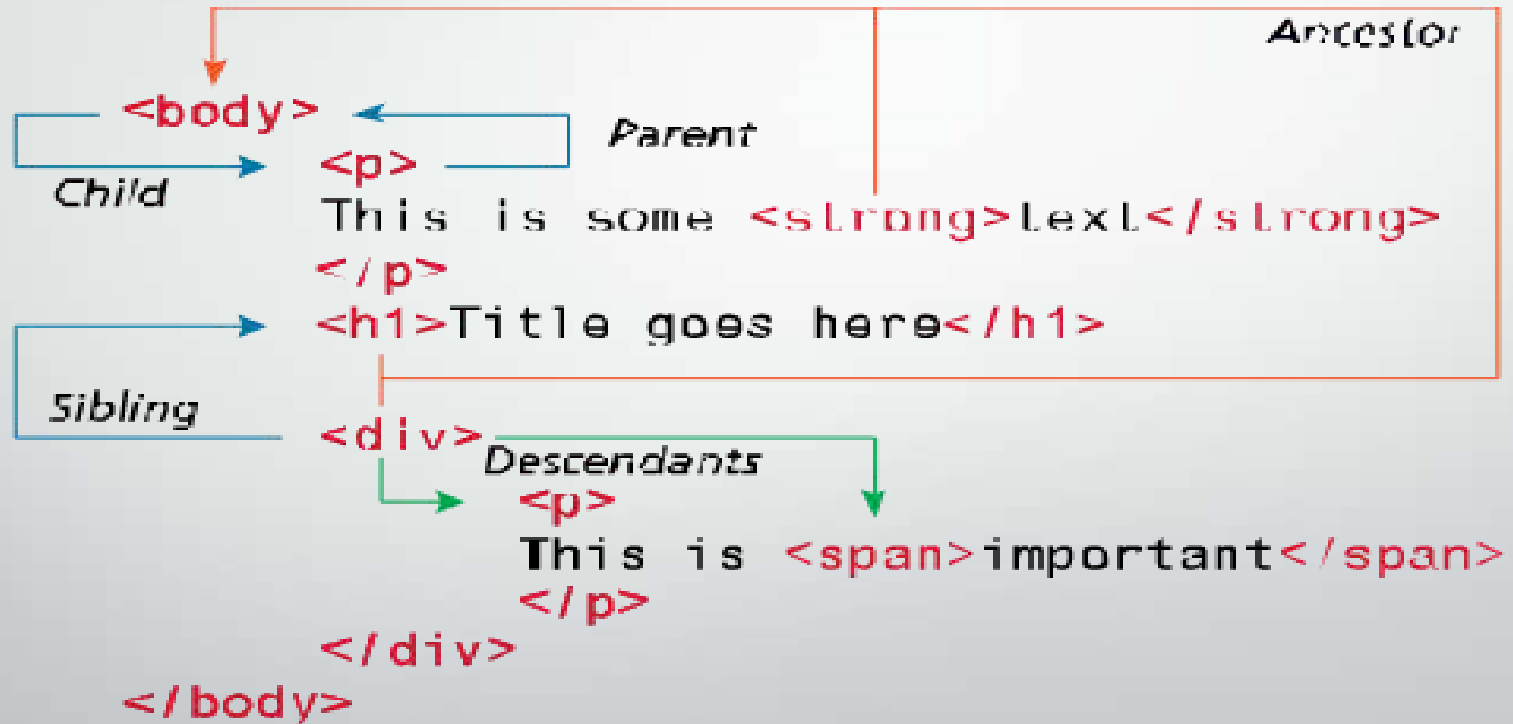
```
<h1>Share Your <strong>Travels</strong></h1>
```

- Incorrect nesting
 - Outer element is closed first

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

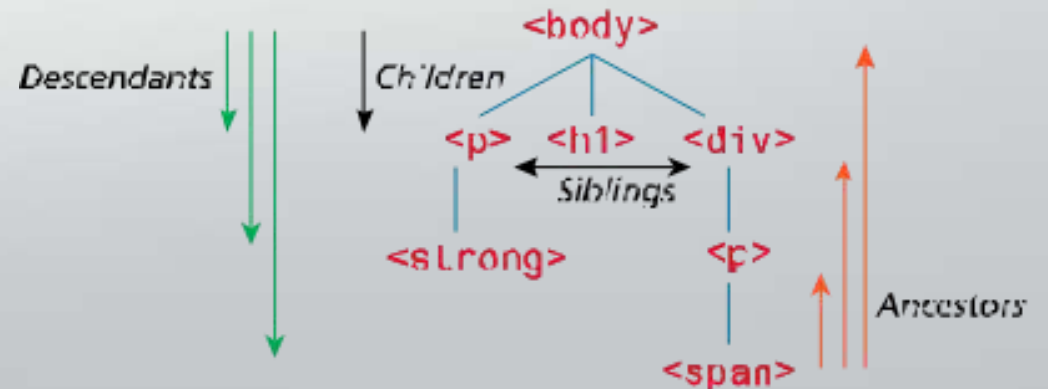
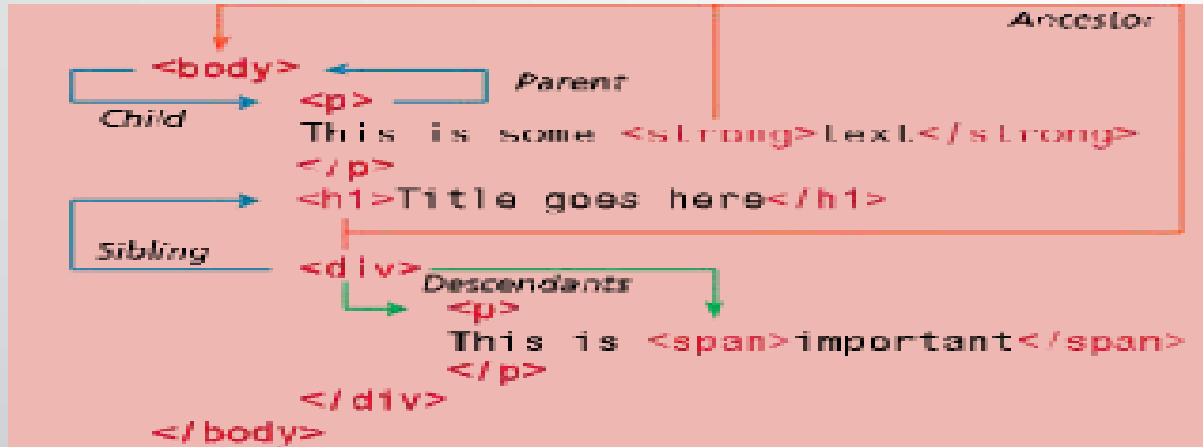
Introduction to HTML

Nesting elements



Introduction to HTML

Tree of Nested Elements





The End



Web System and Technologies CS 521

TOPIC 14

Common HTML Tags

A Simple Example

1

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



Common HTML Tags

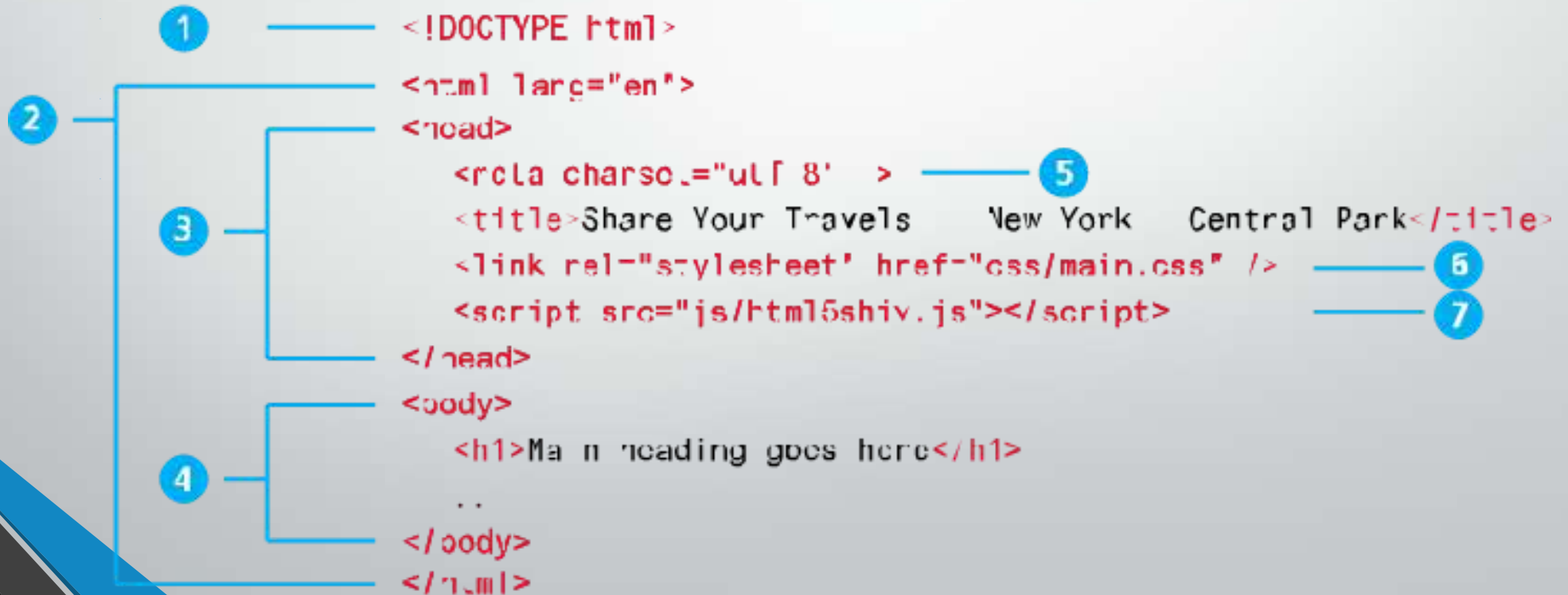
DOCTYPE

- 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

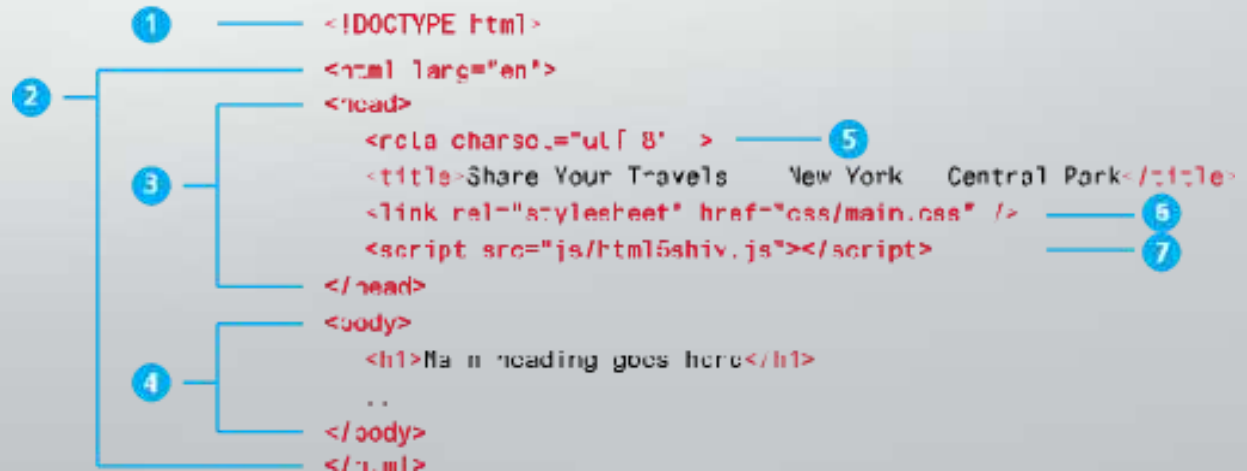
Another Example



Common HTML Tags

Head and Body Tags

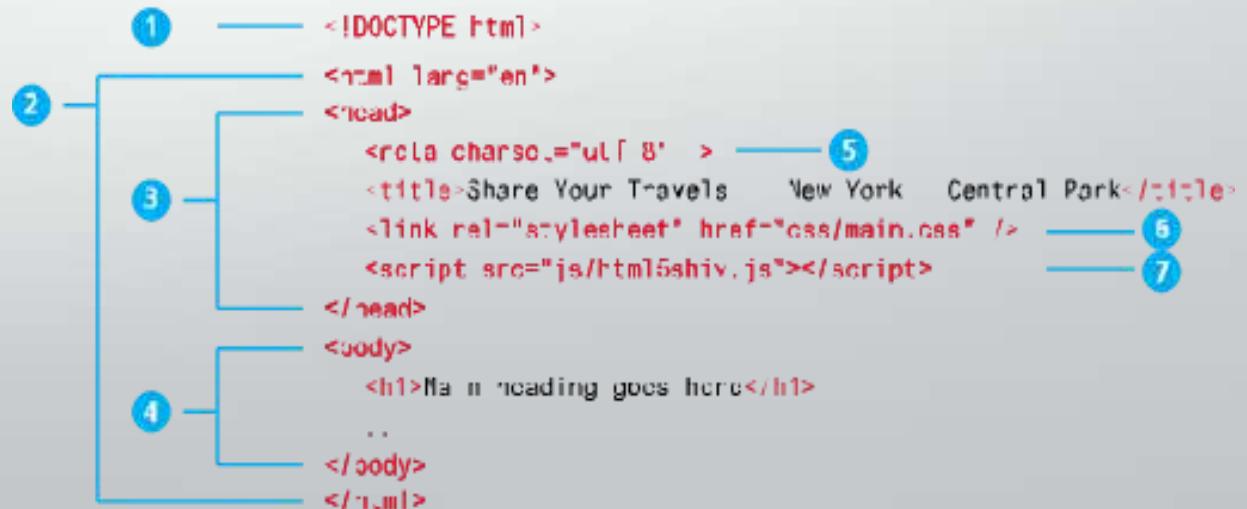
- HTML5 does not require the use of the `<html>` , `<head>` , and `<body>` elements (most developers still use them)
- `<html>` contains all the other HTML elements in the document (Item 2)
- `<head>` contains descriptive elements about the document, such (title, style sheets, JavaScript files etc.)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



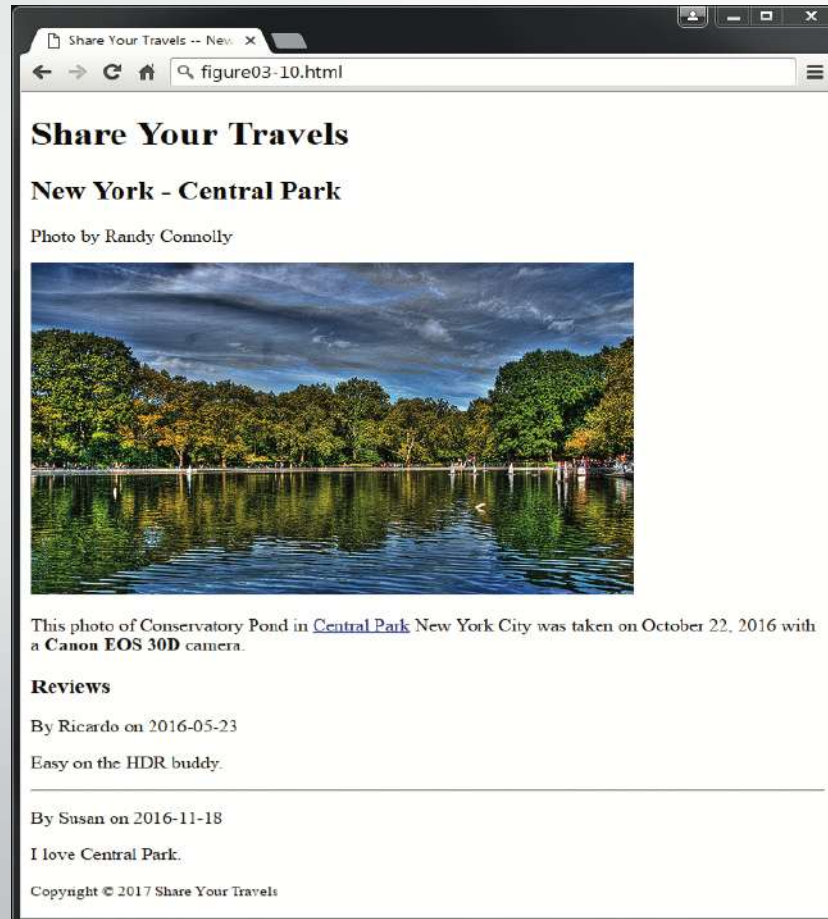
Common HTML Tags

Another Example

```
<body>
  1 | <h1>Share Your Travels</h1>
  2 | <h2>New York - Central Park</h2>
  2 | <p>Photo by Randy Connolly</p>
  2 | <p>This photo of Conservatory Pond in
    |   <a href="http://www.centralpark.com/">Central Park</a> 3
    |   New York City was taken on October 22, 2016 with a
    |   <strong>Canon EOS 30D</strong> camera.
    | </p> 4
  5 | 
    |
    | <h3>Reviews</h3>
  6 | <div> 7
    |   <p>By Ricardo on <time>2016-05-23</time></p>
    |   <p>Easy on the HDR buddy.</p>
    | </div>
  8 | <hr>
    | <div>
    |   <p>By Susan on <time>2016-11-18</time></p>
    |   <p>I love Central Park.</p>
    | </div> 9
    |
    | <p><small>Copyright &copy; 2017 Share Your Travels</small></p>
 10 | </body>
```

Common HTML Tags

Rendered Document



Common HTML Tags

HTML is for structure!

- CSS is used for styling

The image displays four browser windows, each showing a different CSS style for the heading 'Share Your Travels'. The windows are titled 'listing02-03.html', 'listing02-04.html', 'listing02-05.html', and 'listing02-06.html'. Each window shows the rendered heading and the corresponding CSS code used to style it.

listing02-03.html: Shows the heading 'Share Your Travels' in a standard black font. The CSS code is: `h1 { margin: 0 0 0 50px; color: helvetica, sans-serif; font-style`

listing02-04.html: Shows the heading 'Share Your Travels' in a red, italicized font. The CSS code is: `h1 { margin: 0 0 0 0; color: #cc6633; font: 200% arial, helvetica, sans-serif; background-color: #F0EDC7; border: 2px solid green; padding: 5px 10px; }`

listing02-05.html: Shows the heading 'Share Your Travels' in a red font, centered on a yellow background with a green border. The CSS code is: `h1 { margin: 0 0 0 0; color: #cc6633; font: 200% arial, helvetica, sans-serif; background-color: #F0EDC7; border: 2px solid green; padding: 5px 10px; }`

listing02-06.html: Shows the heading 'SHARE YOUR TRAVELS' in a large, red, dripping font, centered on a background with diagonal stripes. The CSS code is: `h1 { margin: 0 0 0 0; padding: 20px; text-align: center; color: #A61C07; font-family: 'Nosifer', cursive; font-size: 60pt; line-height: 54pt; background: url(images/header-background.jpg) repeat-x; height: 120px; }`

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)

```
<body>  
1 | <h1>Share Your Travels</h1>  
  | <h2>New York - Central Park</h2>
```

Common HTML Tags

Paragraphs and Divisions

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



```
<body>  
  <h1>Share Your Travels</h1>  
  <h2>New York - Central Park</h2>
```

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
 - <div>
 - <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 tooltip when user moves mouse over image.

```

```


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).

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

Destination

Label (text)

Links

Examples

Link to external site

`Central Park`

Link to resource on external site

`Central Park`

Link to another page on same site as this page

`Home`

Link to another place on the same page

`Go to Top of Document`

...

``

Defines anchor for a link to another place on same page

Links

Examples

Link to specific place on another page

`Reviews for product X`

Link to email

`Someone`

Link to JavaScript function

`See This`

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

`Call toll free (800) 922-0579`

Links

Relative Referencing

Relative Link Type	Example
Same directory	<code></code>
Child Directory	<code></code>
Grandchild/Descendant Directory	<code></code>
Parent/Ancessor Directory	<code></code> <code></code>
Sibling Directory	<code></code>
Root Reference	<code></code>

Web Systems and Technologies

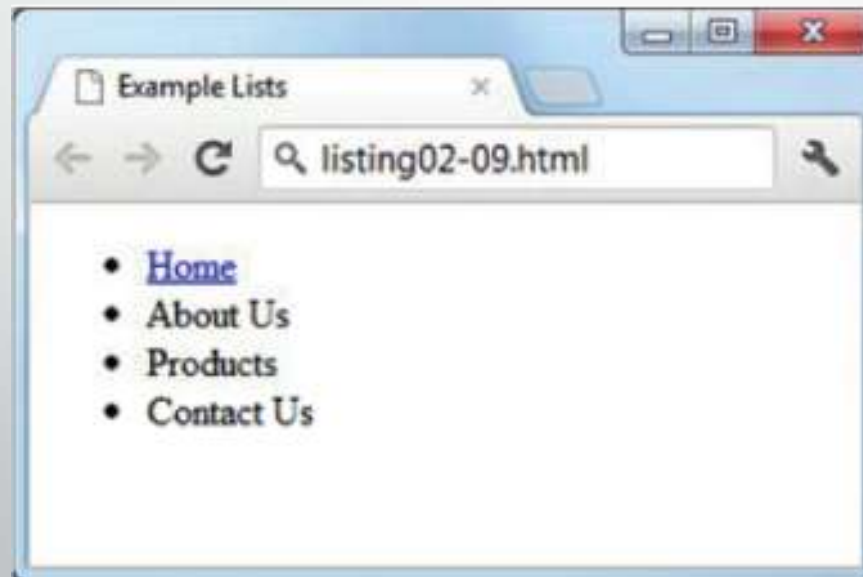
TOPIC 16

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

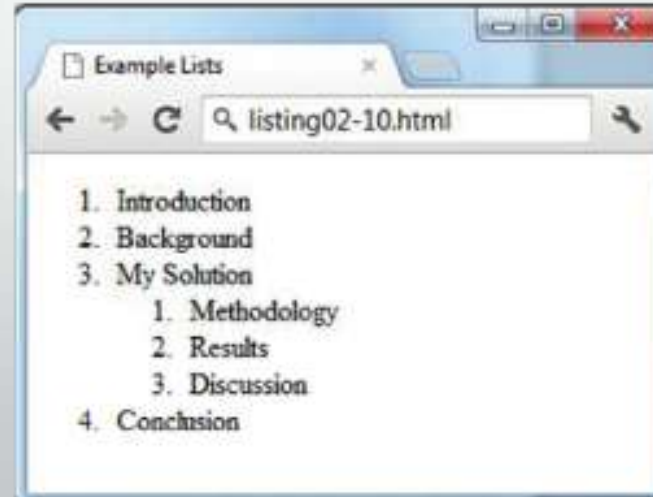
```
<ol>  
  <li>Introduction</li>  
  <li>Background</li>  
  <li>My Solution</li>  
  <li>Conclusion</li>  
</ol>
```

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



Web Systems and Technologies

TOPIC 17

Semantics in HTML 5

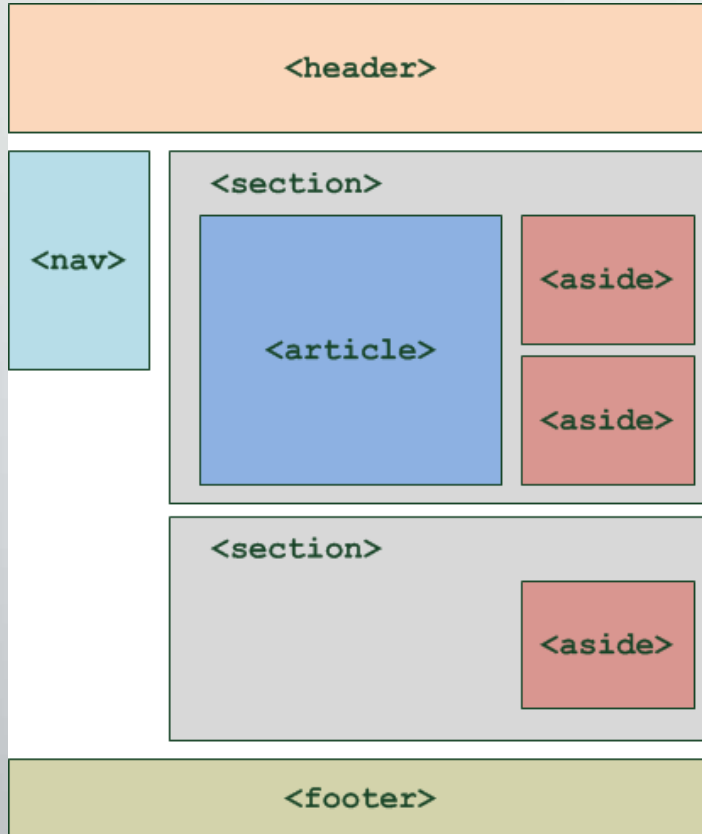
Semantics in HTML 5

Semantic Markup

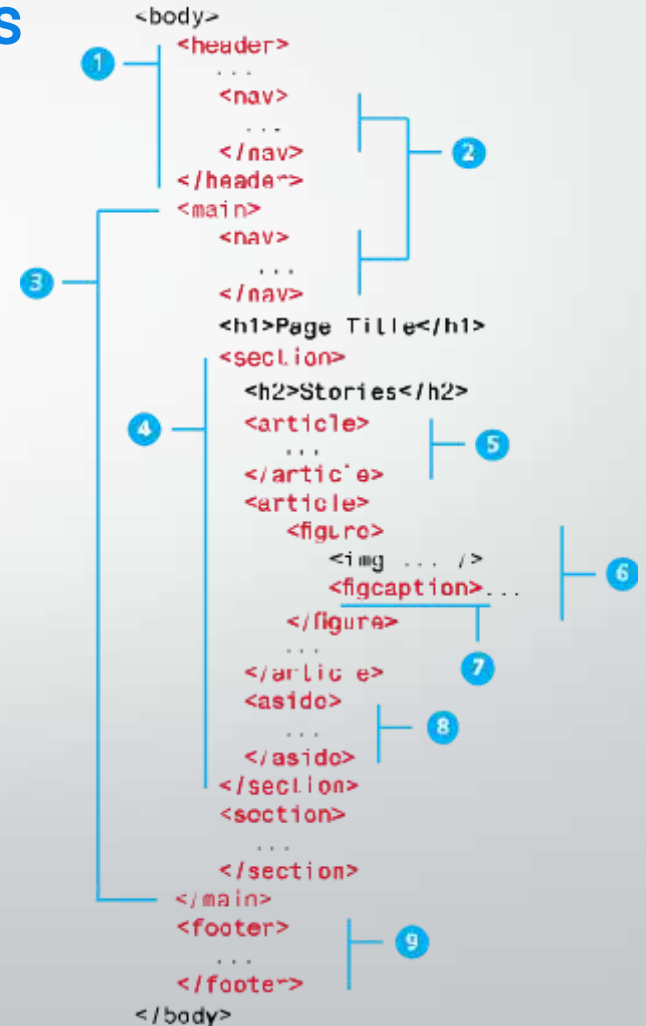
- Semantic tags are used for organizing contents and labeling contents (adding meaning)
- Do not change the structure of the document
- Make HTML code
 - maintainable
 - accessible

Semantics in HTML 5

Semantic tags



Jandi soon, CC BY-SA 3.0



Semantics in HTML 5

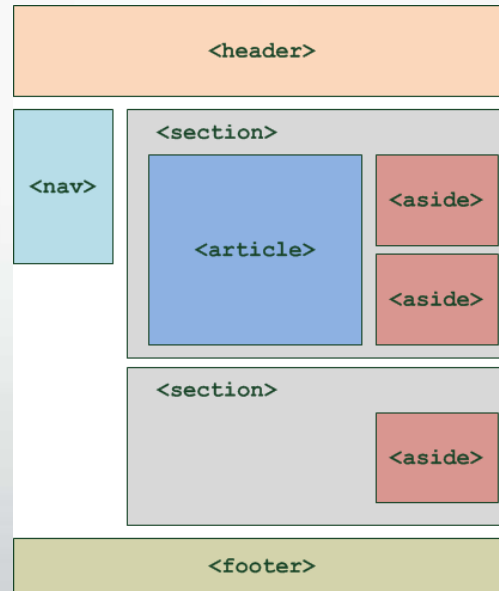
Header and Footer

1

```
<header>  
...  
<nav>  
...  
</nav>  
</header>
```

```
<footer>  
...  
</foote~>
```

9



Semantics in HTML 5

Header

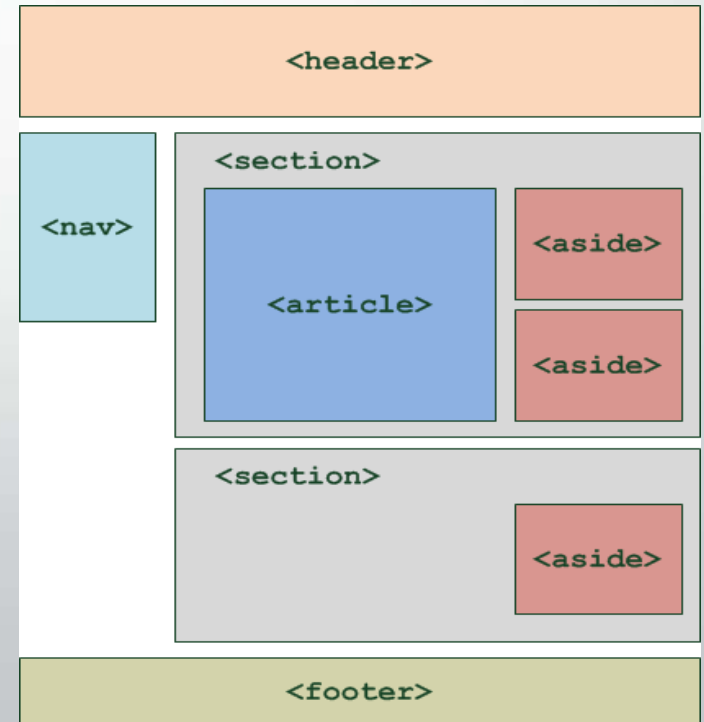
- Intended to usually contain the section's heading (an h1– h6 element), but not required
- The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.

```
<header>
  
  <h1>Fundamentals of Web Development</h1>
  ...
</header>
```

Semantics in HTML 5

Navigation (nav)

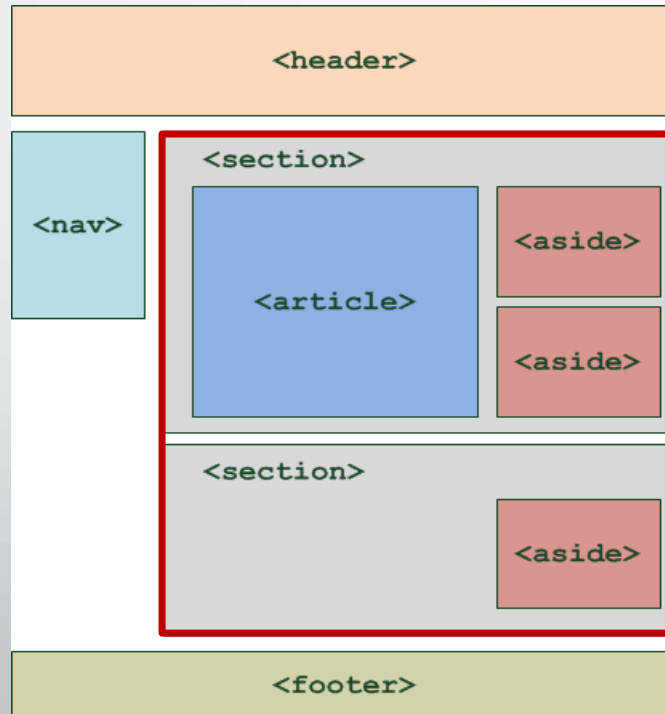
```
<header>
  
  <h1>Fundamentals of Web Development</h1>
  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="about.html">About Us</a></li>
      <li><a href="browse.html">Browse</a></li>
    </ul>
  </nav>
</header>
```



Semantics in HTML 5

Main

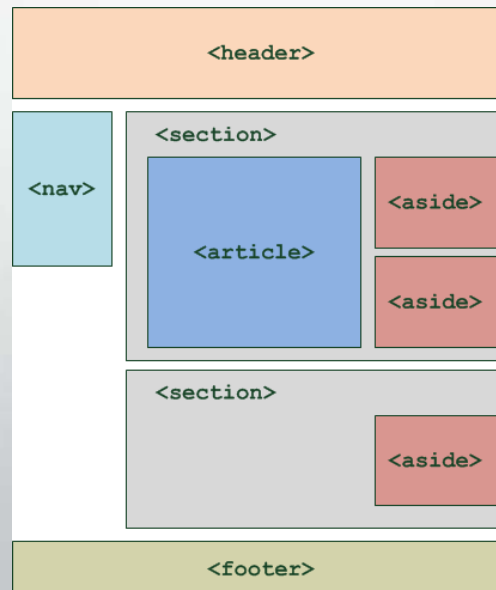
- `<main>` is meant to contain the main unique content of the document.



Semantics in HTML 5

Section and Article

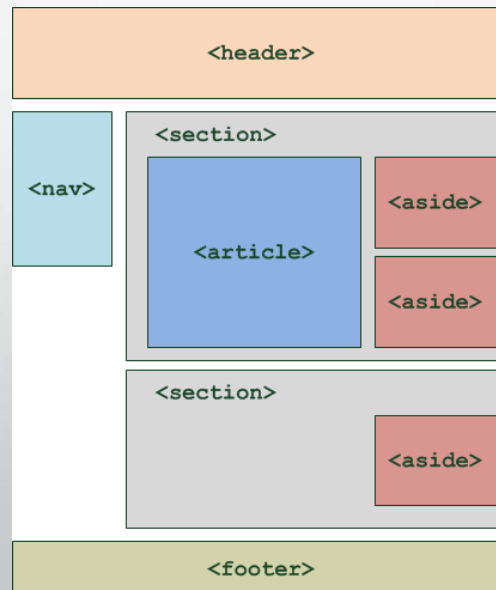
- `<section>` is a much broader element, while the
- `<article>` element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page



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



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



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



```
<body>
  <h2>Girl with a Pearl Earring</h2>
  <details>
    <summary>Image</summary>
    <br>
    <p>Museum: Royal Picture Gallery Mauritshuis ...</p>
  </details>
  <details>
    <summary>Artist</summary>
    <p><strong>Jan Vermeer</strong> was a Dutch ...</p>
  </details>
  <details>
    <summary>Information</summary>
    <p>
      Date: 1665<br>
      Medium: Oil on Canvas
    </p>
  </details>
</body>
```



Web Systems and Technologies

TOPIC 18

Tools Insight

Tools Insights

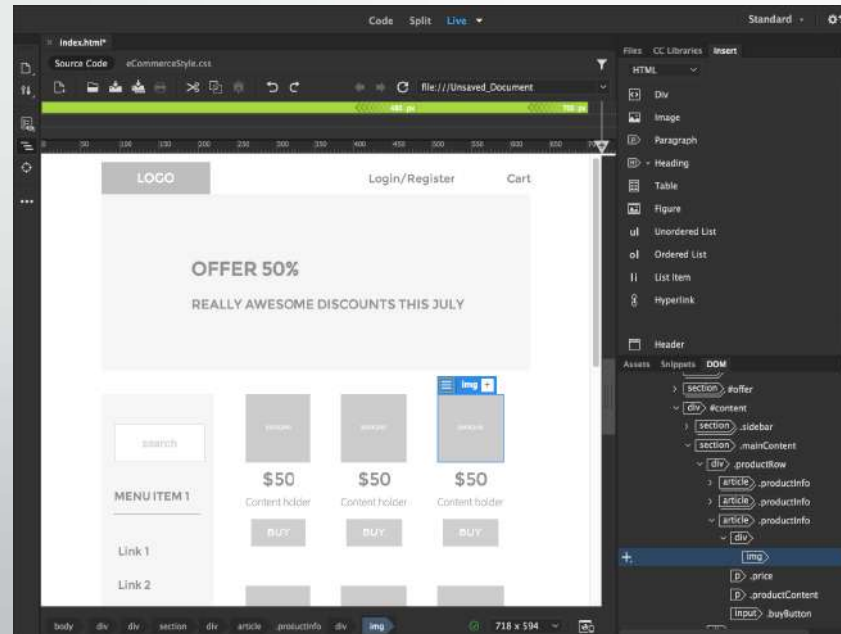
A variety of tools available for web development

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

Tools Insights

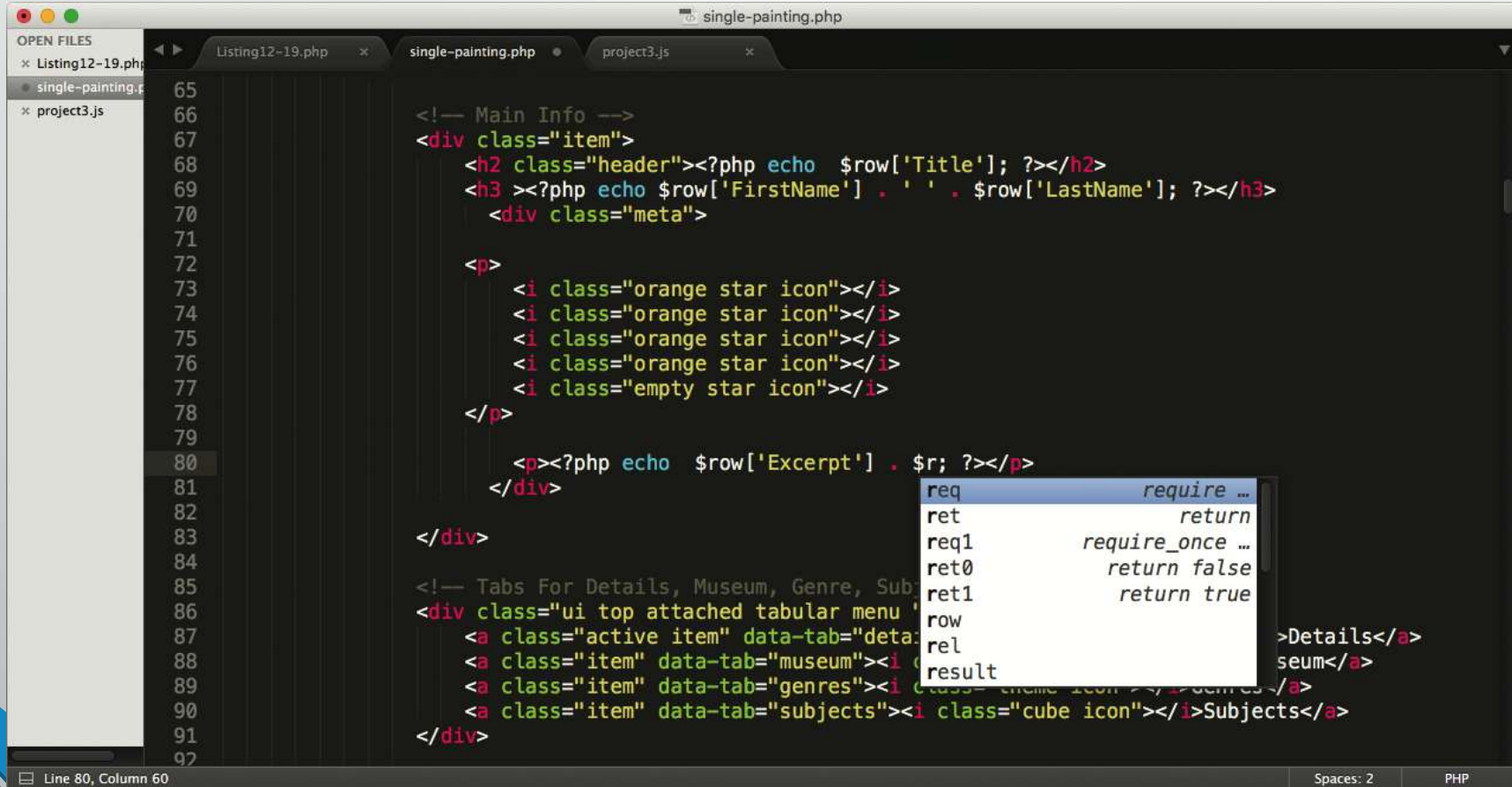
WYSIWYG Editors

- Dreamweaver
- Google Web Designer
- Seamonkey composer



Tools Insights

Code editors



```
65
66
67 <!-- Main Info -->
68 <div class="item">
69   <h2 class="header"><?php echo $row['Title']; ?></h2>
70   <h3 ><?php echo $row['FirstName'] . ' ' . $row['LastName']; ?></h3>
71   <div class="meta">
72
73     <p>
74       <i class="orange star icon"></i>
75       <i class="orange star icon"></i>
76       <i class="orange star icon"></i>
77       <i class="orange star icon"></i>
78       <i class="empty star icon"></i>
79     </p>
80     <p><?php echo $row['Excerpt'] . $r; ?></p>
81   </div>
82
83 </div>
84
85 <!-- Tabs For Details, Museum, Genre, Subj
86 <div class="ui top attached tabular menu"
87   <a class="active item" data-tab="details" href="#">Details</a>
88   <a class="item" data-tab="museum" href="#">Museum</a>
89   <a class="item" data-tab="genres" href="#">Genres</a>
90   <a class="item" data-tab="subjects" href="#">Subjects</a>
91 </div>
92
93
```

req require ...
ret return
req1 require_once ...
ret0 return false
ret1 return true
row
rel
result

Line 80, Column 60 Spaces: 2 PHP

Tools Insights

Cloud-based Environments

The screenshot shows a web editor interface for Codeanywhere, Inc. The browser address bar displays `https://codeanywhere.com/editor/`. The editor has a dark theme and a file explorer on the left showing a project structure with folders like `vendor`, `webroot`, `css`, `images`, `includes`, `js`, and `lib`. The main editor area shows HTML code for a navigation menu. A tooltip is visible over the code, displaying `alt` and `accesskey`. The status bar at the bottom shows the current file path `Chapter17-project2 > webroot > detail.html`, the zoom level `46.67`, the tab size `Tab Size: 2`, and the file type `HTML`.

```
23 *      <i class="user icon"></i>
24      Account
25 *      <i class="dropdown icon"></i>
26 *      <div class="menu">
27 *          <a class="item"><i class="sign in icon"></i> Login</a>
28 *          <a class="item"><i class="edit icon"></i> Edit Profile</a>
29 *          <a class="item"><i class="globe icon"></i> Choose Language</a>
30 *          <a class="item"><i class="settings icon"></i> Account Settings</a>
31 *      </div>
32 *      </div>
33 *      <a class=" item">
34 *          <i class="heartbeat icon"></i> Favorites
35 *      </a>
36 *      <a class=" item">
37 *          <i class="shop icon"></i> Cart
38 *      </a>
39 *      </nav>
40 *      </div>
41 *      </div>
42 *
43 *      <div class="ui attached stackable borderless huge menu" >
44 *          <div class="ui container">
45 *              <h2 class="header item">
46 *                  
48 *              <a class="item">
49 *                  <i class="home icon"></i> Home
50 *              </a>
51 *              <a class="item">
52 *                  <i class="mail icon"></i> About Us
53 *              </a>
54 *              <a class="item">
55 *                  <i class="home icon"></i> Blog
```

Tools Insights

Code playgrounds

The screenshot shows a CodePen playground with the following code:

```
HTML
<html>
  <head>
    <link rel="stylesheet" href="https://www.dreamstime.com/func.../font-awesome.min.css" />
    <link rel="stylesheet" href="css/alert.css" />
    <script type="text/javascript" src="js/alert.js" />
  </head>
  <body>
    <div class="alert info">
      </div>
  </body>
</html>
```

```
CSS
.alert {
  margin: 20px 0px;
  padding: 12px;
}
.info {
  color: #105863;
  background-color: #FFD3DF;
}
.success {
  color: #4FF7A5;
  background-color: #105863;
}
.warning {
  color: #105863;
}
```

```
JS
window.$(function() {
  // Load all the functions
  our: {
    showAlert: function() {
      document.querySelector("#main div.alert");
    }
  }
  // Loop through them and add a class
  $(".alert").fadeToggle(1000);
  for (var i=0; i < $(".alert").length; i++) {
    $(".alert")[i].fadeOut(1000);
  }
});
```

The rendered output shows three panels with placeholder text:

- Alert panel (orange background)
- Info panel (teal background)
- Success panel (orange background)

At the bottom, there is a navigation bar with options: [Full Screen](#), [Console](#), [Assets](#), [Embed](#), [Comments](#), [Delete](#), [Share](#), [Export](#). The status bar indicates "Last saved less than a minute ago".



Web Systems and Technologies

TOPIC 19

Introduction to CSS

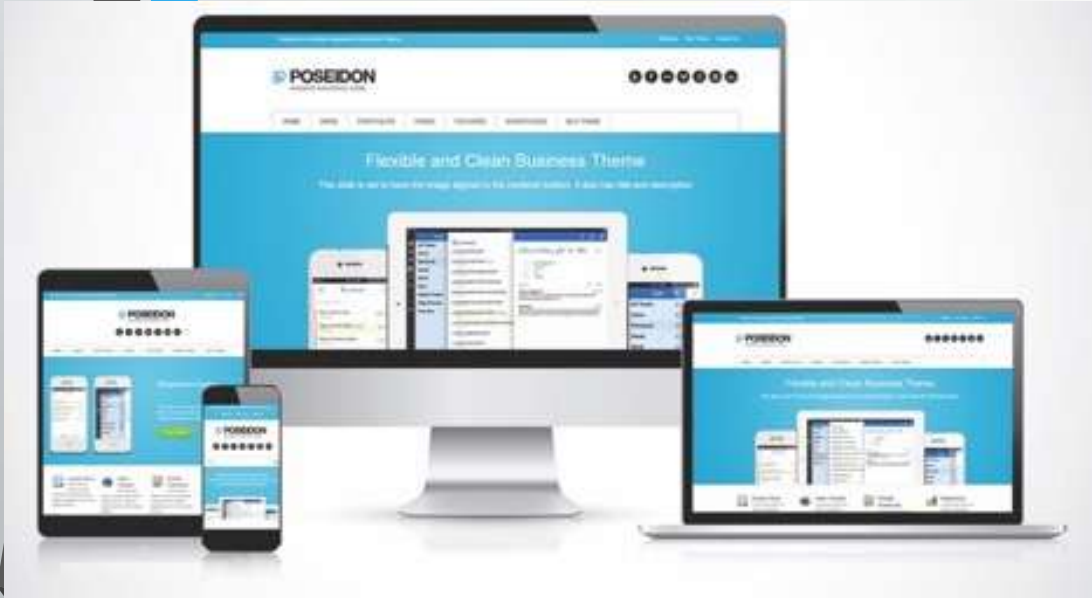
Introduction to CSS

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)

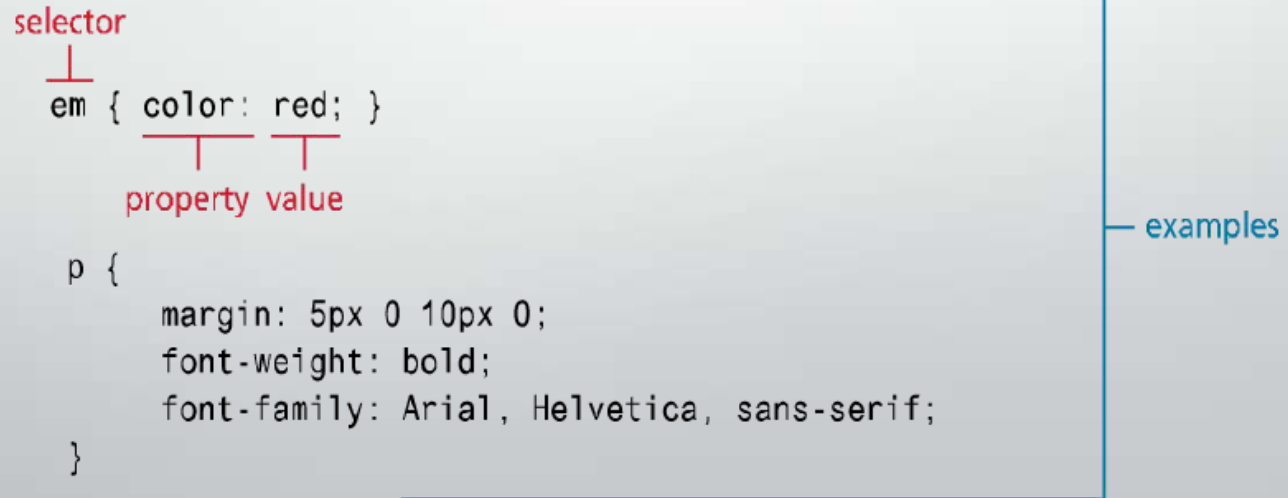
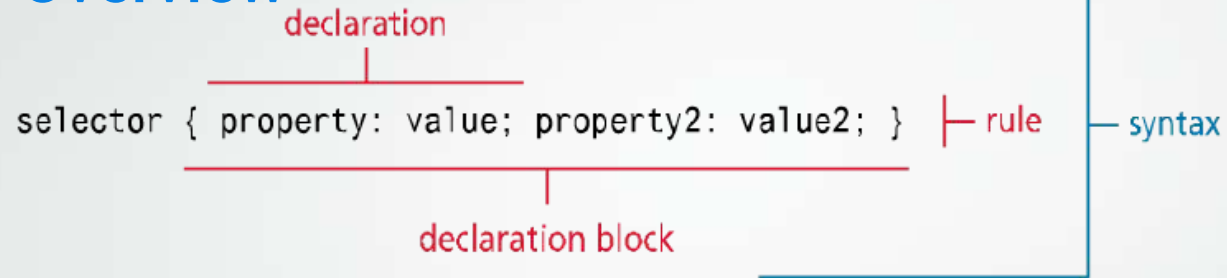
CSS Syntax

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



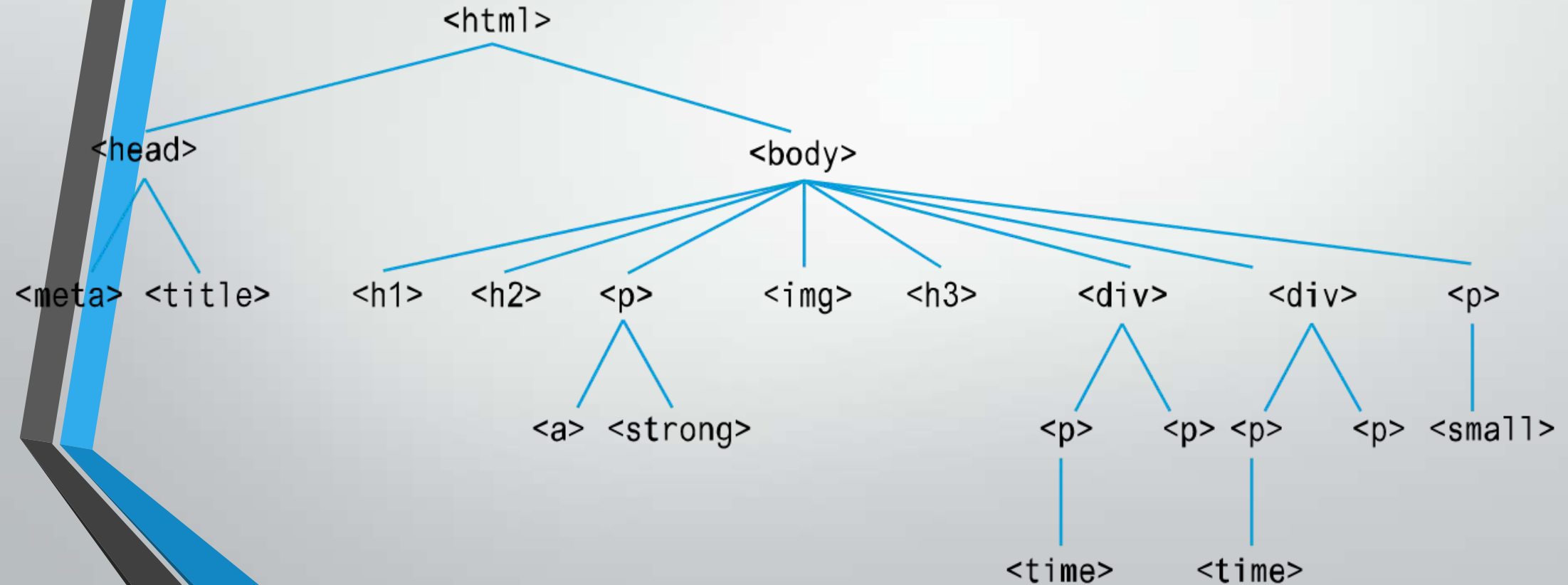
Web Systems and Technologies

TOPIC 23

**CSS Selectors
(Element, Class, ID)**

Selectors

HTML DOM Tree



Selectors

Element Selectors

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

selector



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

property value

```
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/>
</body>
```



Web System and Technologies 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>
  
  <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>
    
    
    
  </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.

Checks if a specific attribute exists.

Exact match: attribute has a specific value.

Contains word: attribute value matches at least one word in a list. Use in class, title, or data-*

Starts with: attribute value begins with a specified string. Use in URLs or IDs

Contains substring: attribute value contains a specified substring.

Ends with: attribute value ends with a specified string.



The End



Web System and Technologies 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 For specific portion
- A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships For all line or statement

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



Web System and Technologies CS 521

TOPIC 26

CSS Selectors (Contextual)

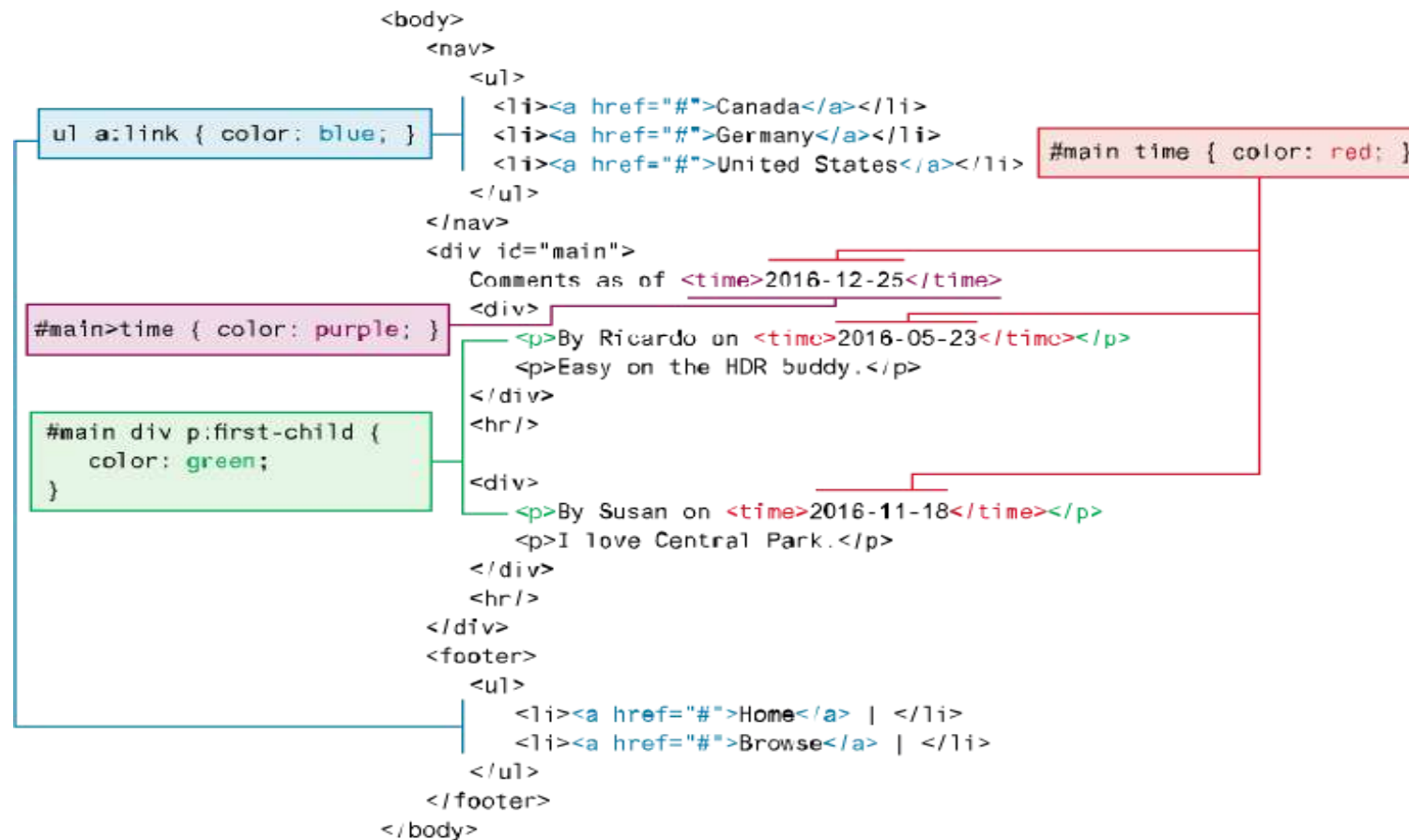
```
<div>
  <article>
    <p>...</p>
  </article>
  <p>...</p>
  <p>...</p>
</div>
```

```
<div>
  <article>
    <p>...</p>
  </article>
  <p>...</p>
  <p>...</p>
</div>
```

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	div p Selects a <p> element that is contained somewhere within a <div> element.
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 <div> element.
Adjacent Sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element	h3+p Selects the first <p> after any <h3>.
General Sibling	A specified element that shares the same parent as the specified element	h3~p Selects all the <p> elements that share the same parent as the <h3>.

```
<div>
  <article>
    <p>...</p>
  </article>
  <p>...</p>
  <p>...</p>
</div>
```

CSS Selectors (Contextual)





The End



Web System and Technologies 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

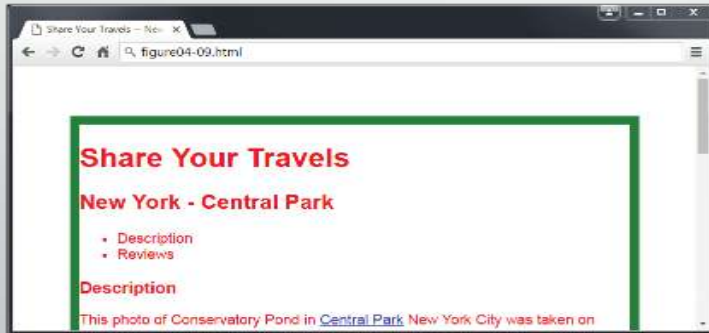
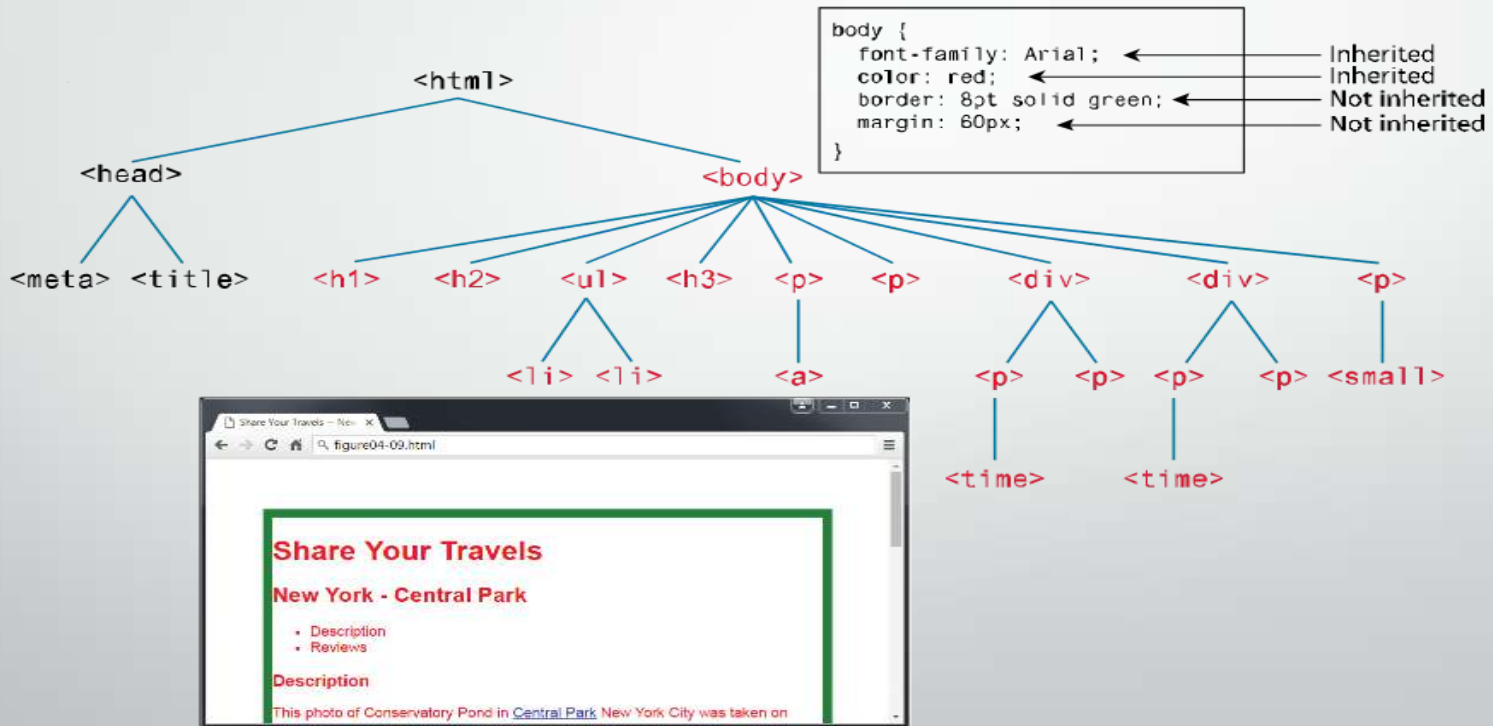
The Cascade: How Styles Interact

Inheritance:

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

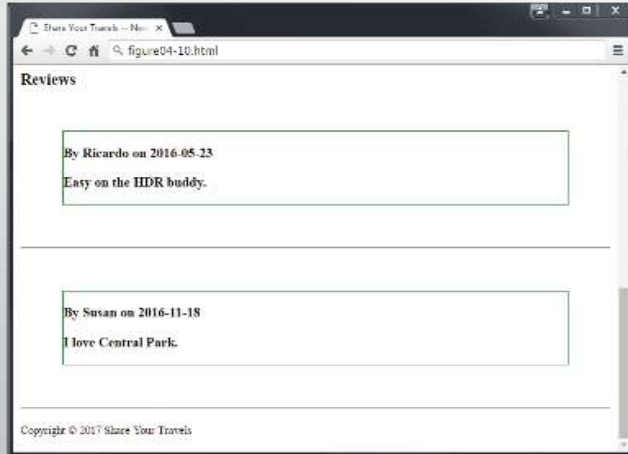
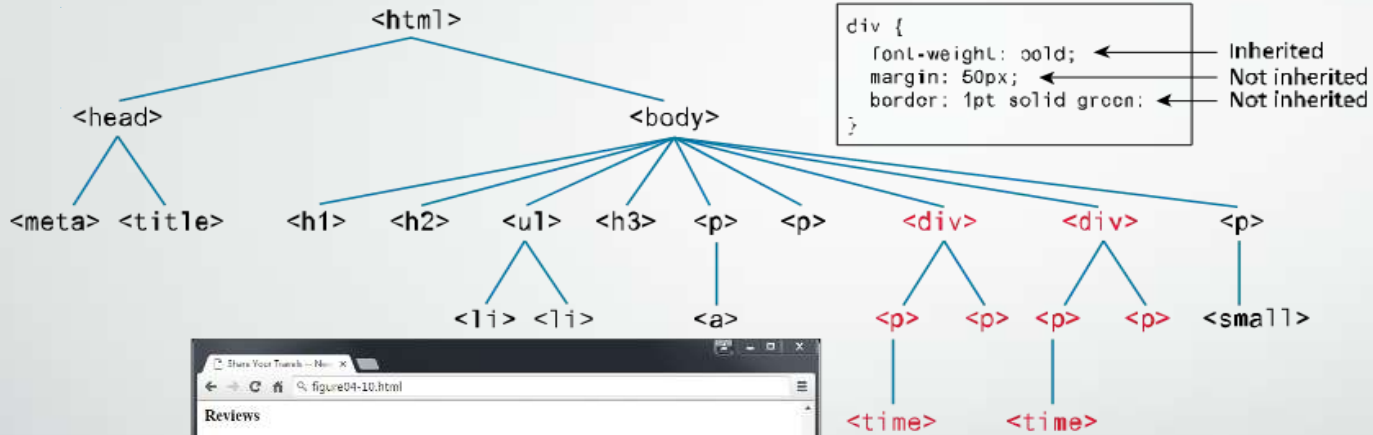
The Cascade: How Styles Interact

Inheritance:



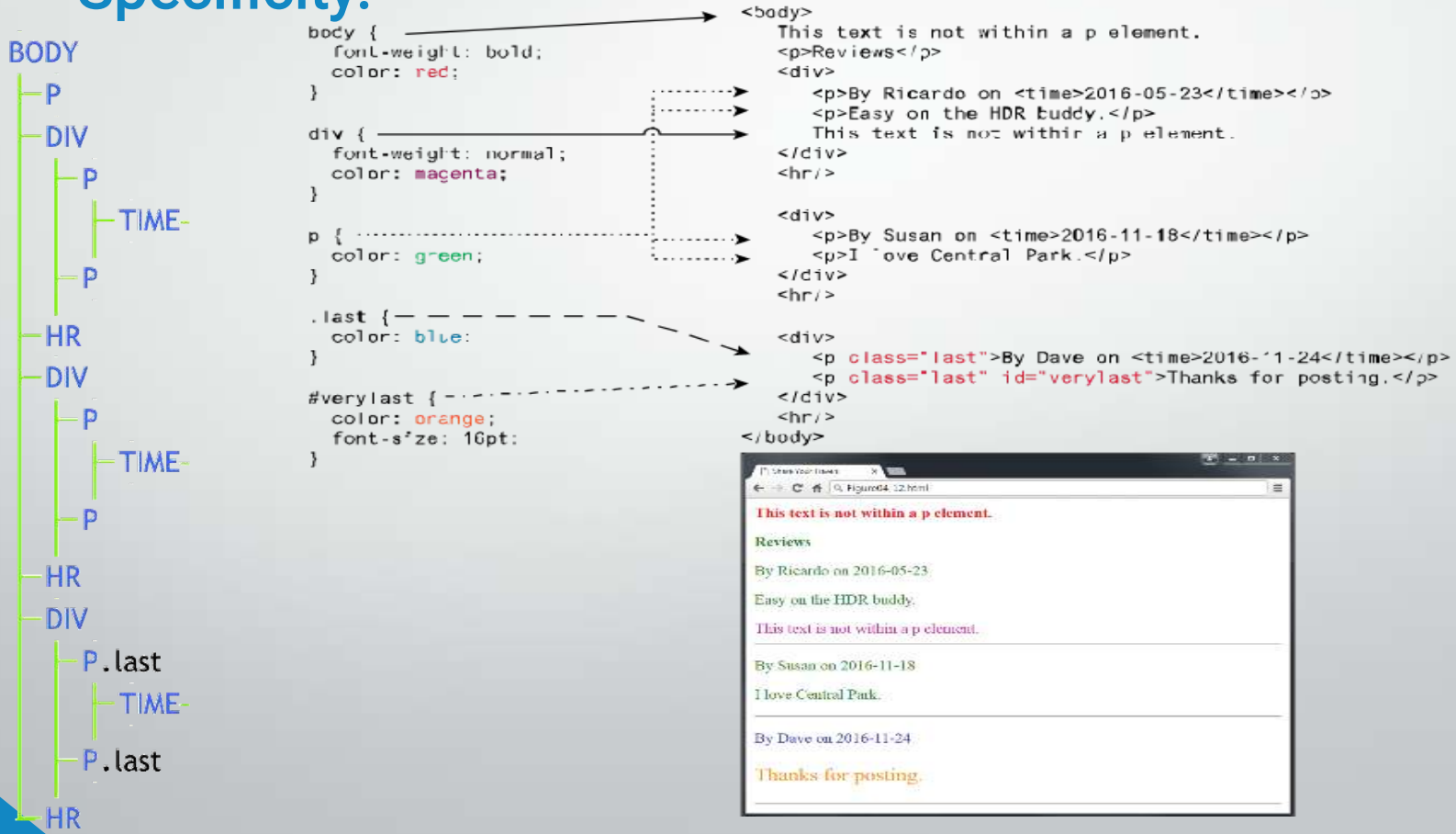
The Cascade: How Styles Interact

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;
}

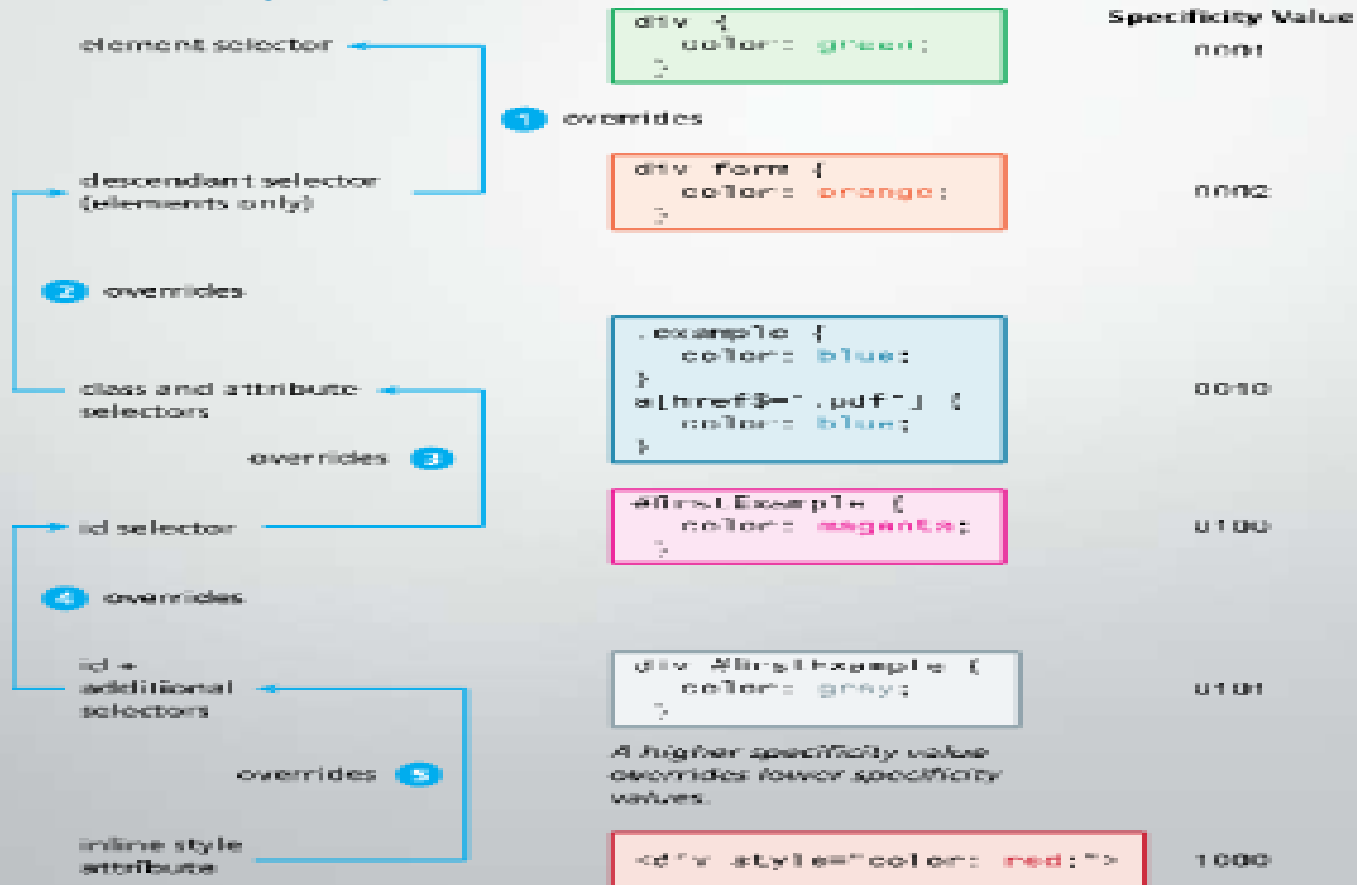
#verylast {
  color: orange;
  font-size: 10pt;
}
```

```
<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>
```



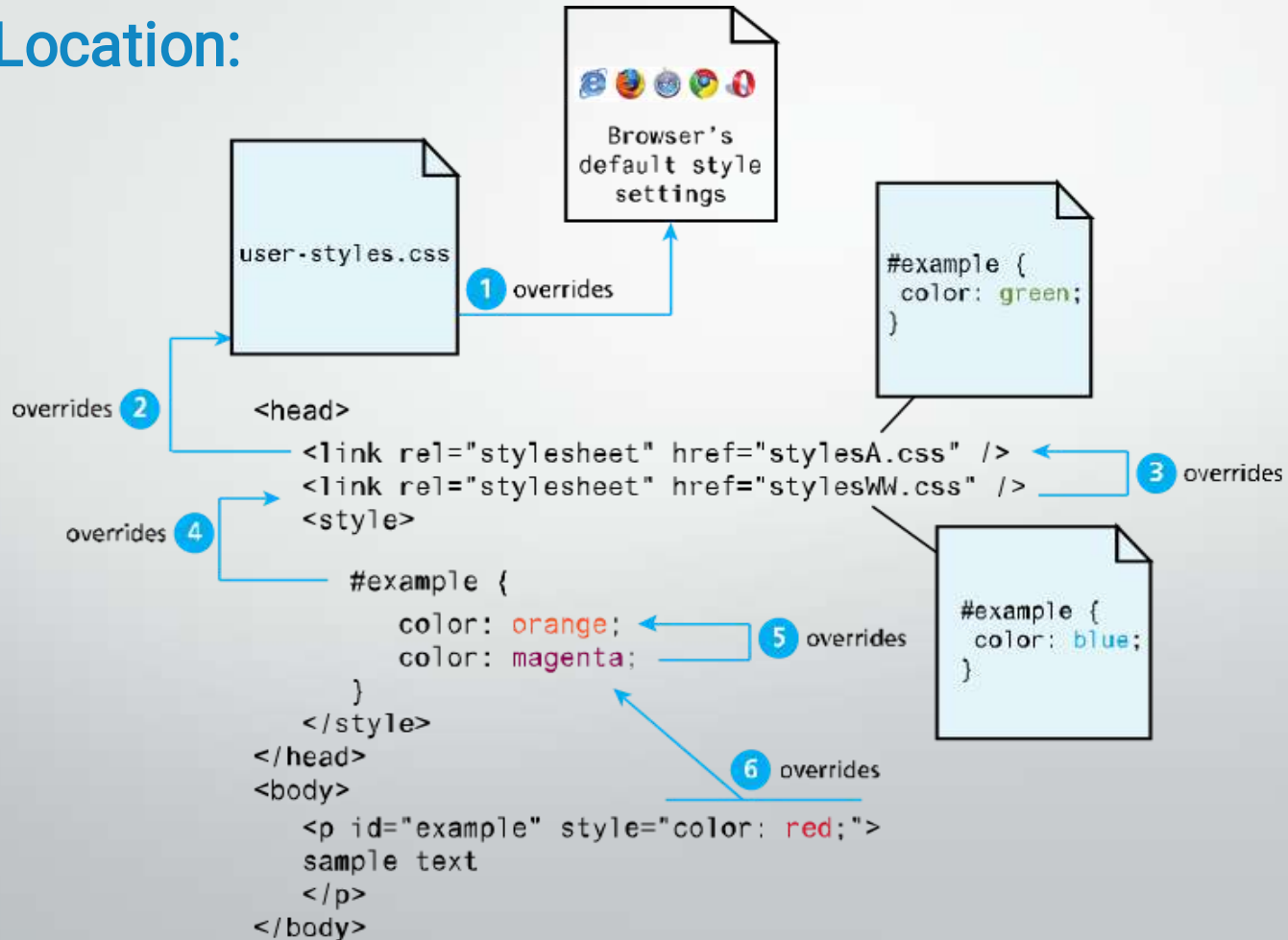
The Cascade: How Styles Interact

Specificity Algorithm:



The Cascade: How Styles Interact

Location:





The End

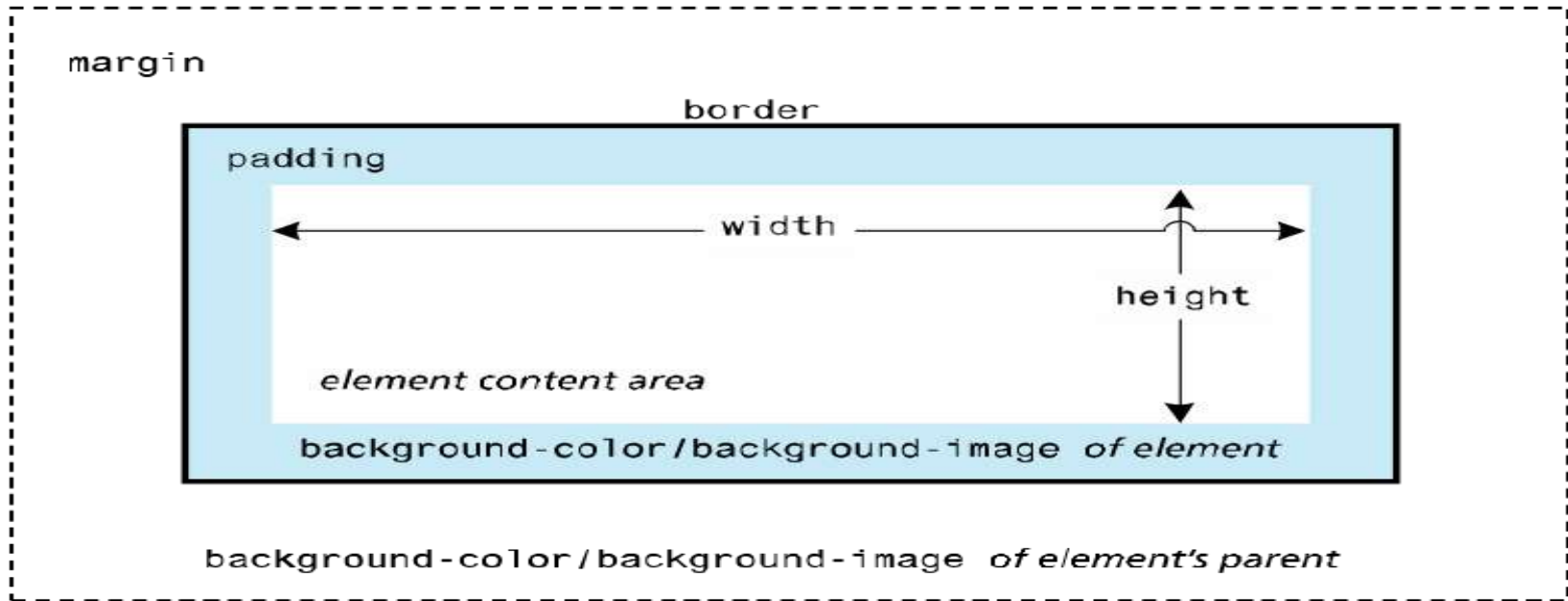


TOPIC 28 WITH VOICE NOTE

Web System and Technologies CS 521

The Box Model

Overview:



The Box Model

Background:

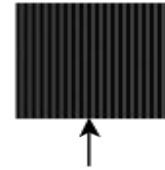
- Background
- Background-attachment
- Background-color
- Background-image
- Background-position
- Background-repeat
- Background-size

The Box Model

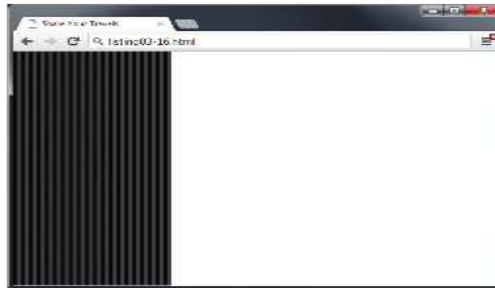
BackGround:



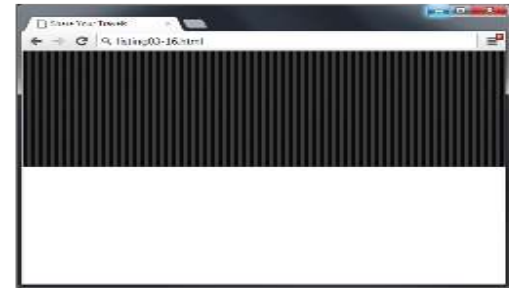
```
background-image: url(../images/backgrounds/body-background-tile.gif);  
background-repeat: repeat;
```



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



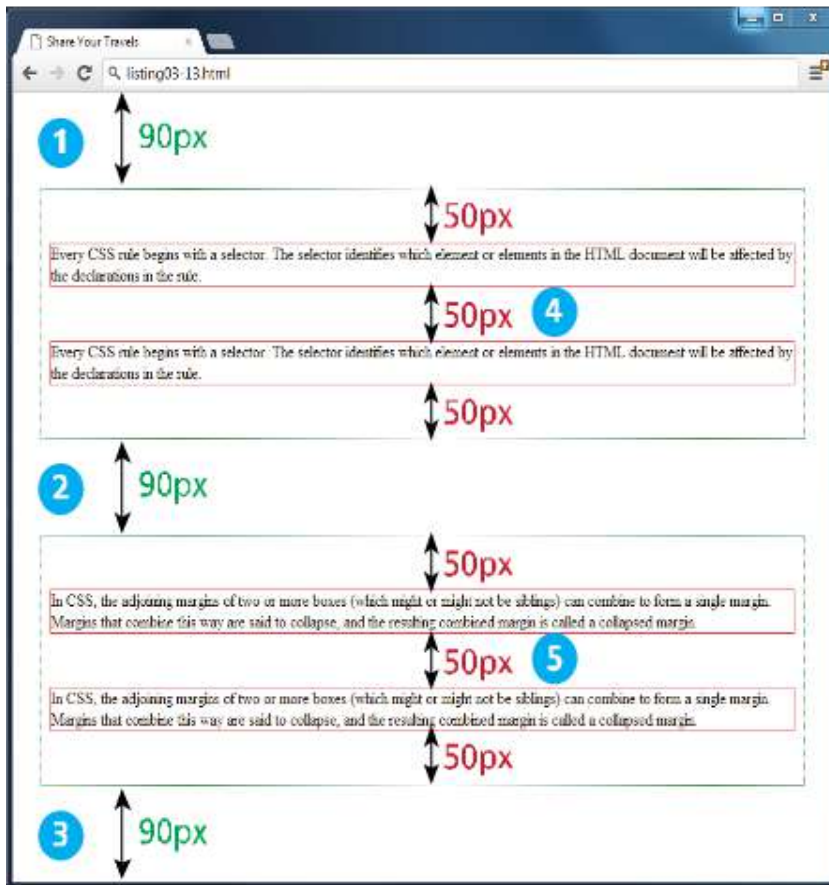
```
background-repeat: repeat-y;
```



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

The Box Model

Collapsing Margins:

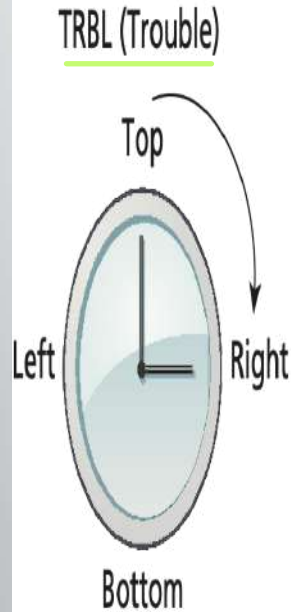


```
<div>
  <p>Every CSS rule ...</p>
  <p>Every CSS rule ...</p>
</div>
<div>
  <p>In CSS, the adjoining ... </p>
  <p>In CSS, the adjoining ... </p>
</div>
```

```
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



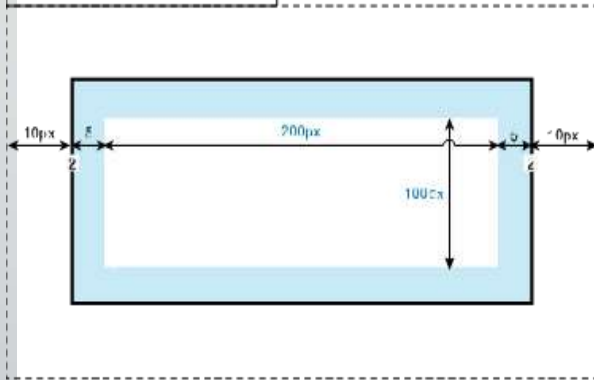
Web System and Technologies CS 521

TOPIC 29

The Box Model

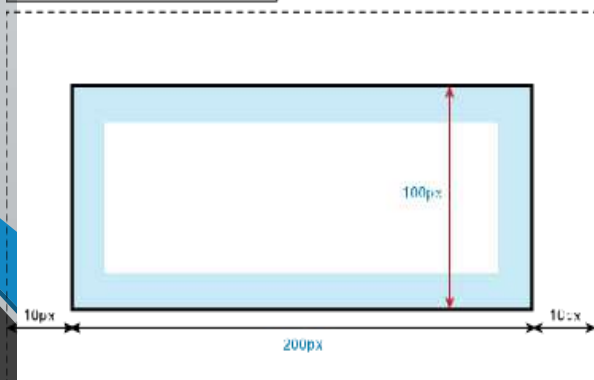
```
div {  
  box-sizing: content-box;  
  width: 200px;  
  height: 100px;  
  padding: 5px;  
  margin: 10px;  
  border: solid 2pt black;  
}
```

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

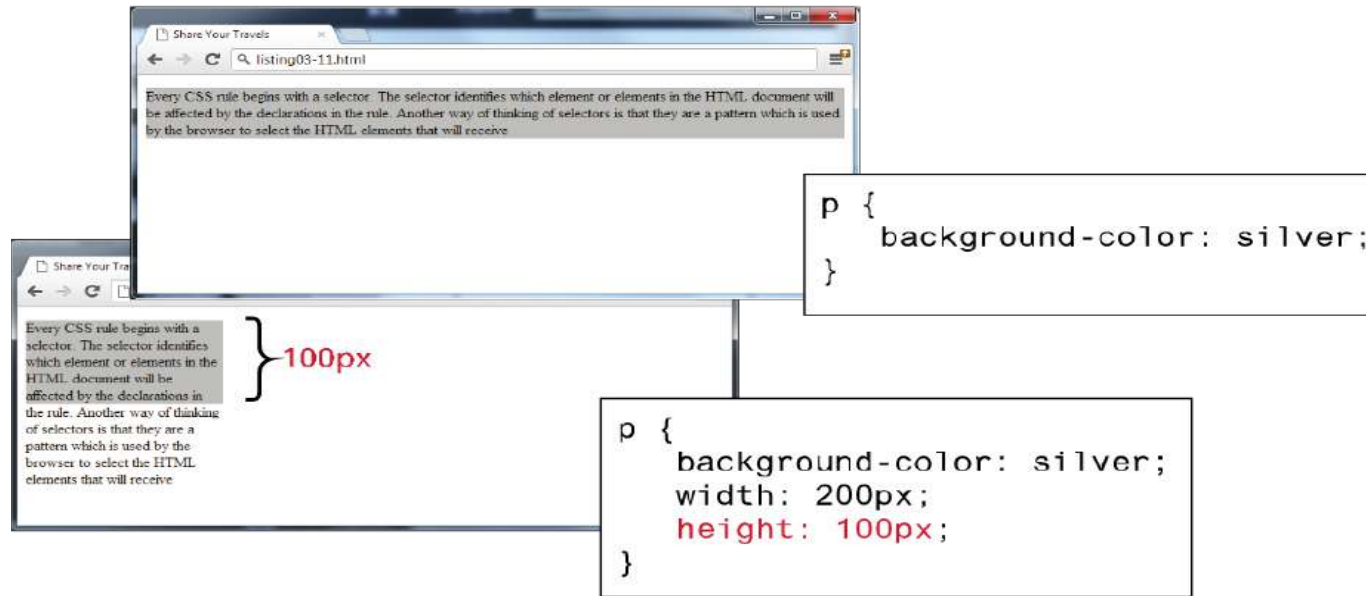


Box Dimensions:

- 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

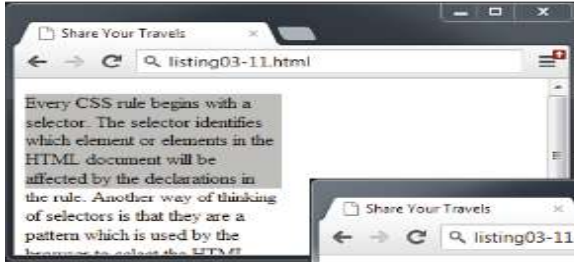
Limitation of Height Property:



The Box Model

OverFlow Property:

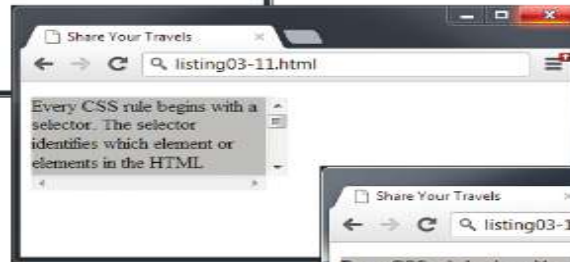
`overflow: visible;`



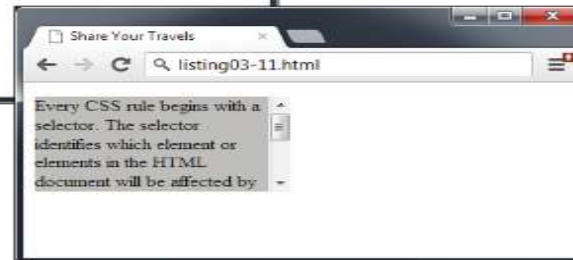
`overflow: hidden;`



`overflow: scroll;`



`overflow: auto;`



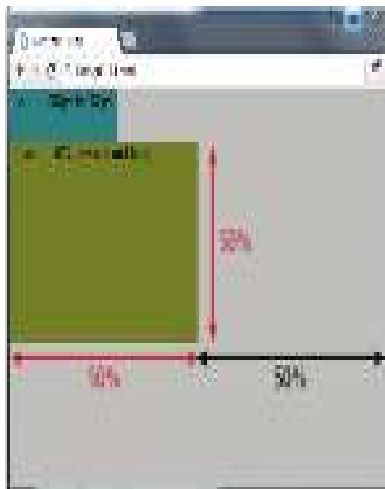
The Box Model

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;
}
</style>
```



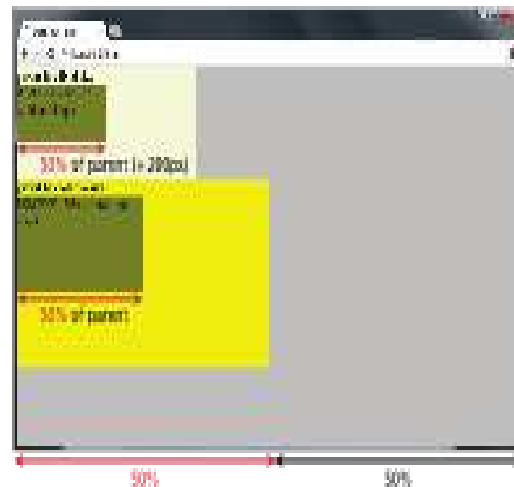
```
<body>
<div class="pixels">
Pixels - 200px by 50 px
</div>
<div class="percent">
Percent - 50% of width and height
</div>
</body>
```



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



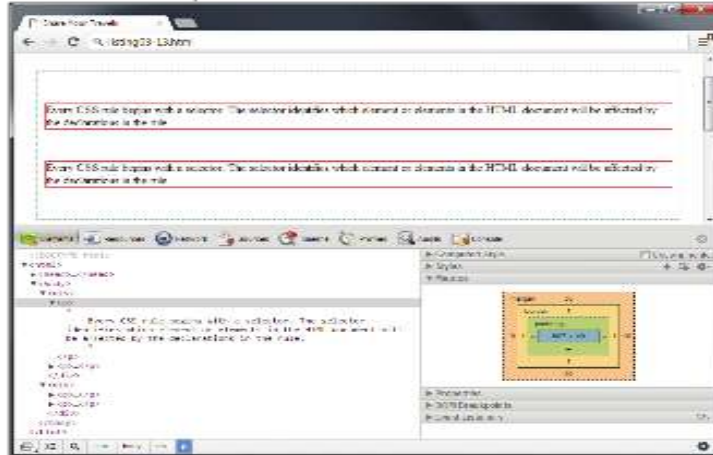
```
<body>
<div class="parentFixed">
<strong>parent has fixed size</strong>
<div class="percent">
PERCENT - 50% of width and height
</div>
</div>
<div class="parentRelative">
<strong>parent has relative size</strong>
<div class="percent">
PERCENT - 50% of width and height
</div>
</div>
</body>
```



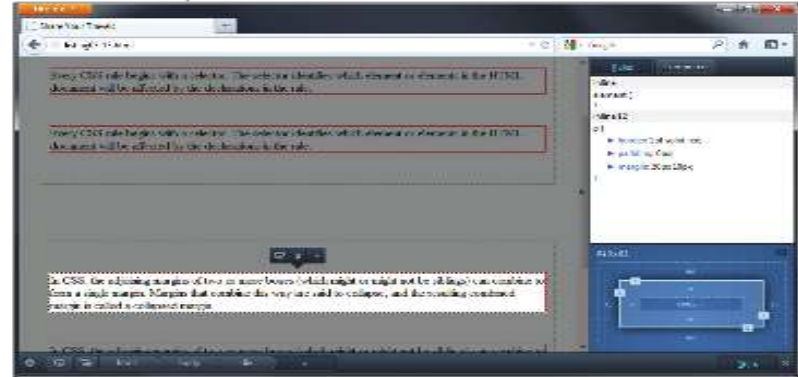
The Box Model

Inspecting CSS using Modern Browsers:

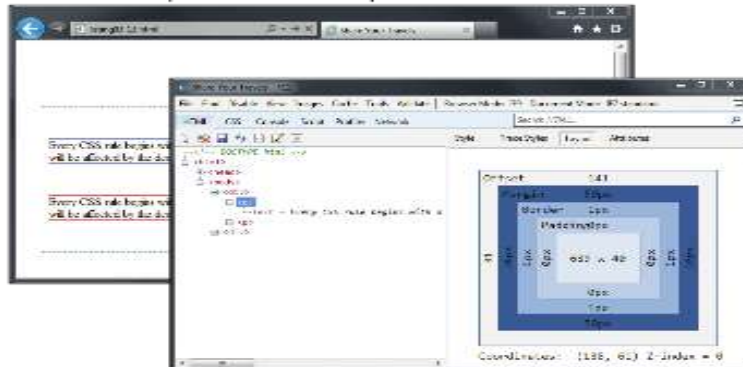
Chrome – Inspect Element



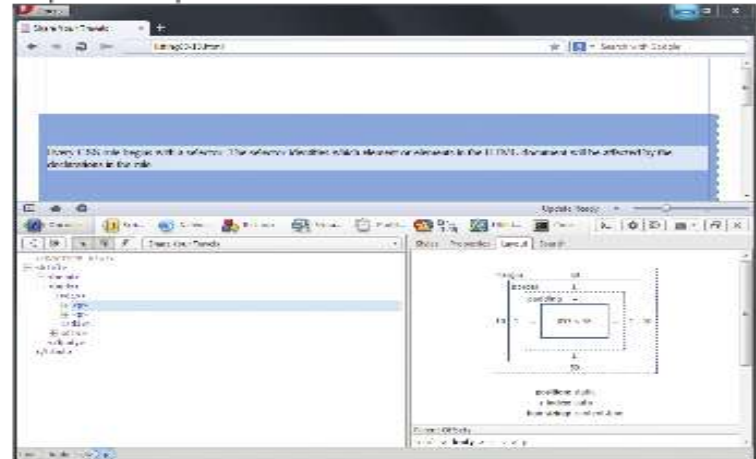
Firefox – Inspect



Internet Explorer – Developer Tools



Opera – Inspect Element





The End



TOPIC 30 WITH VOICE NOTE

Web System and Technologies CS 521

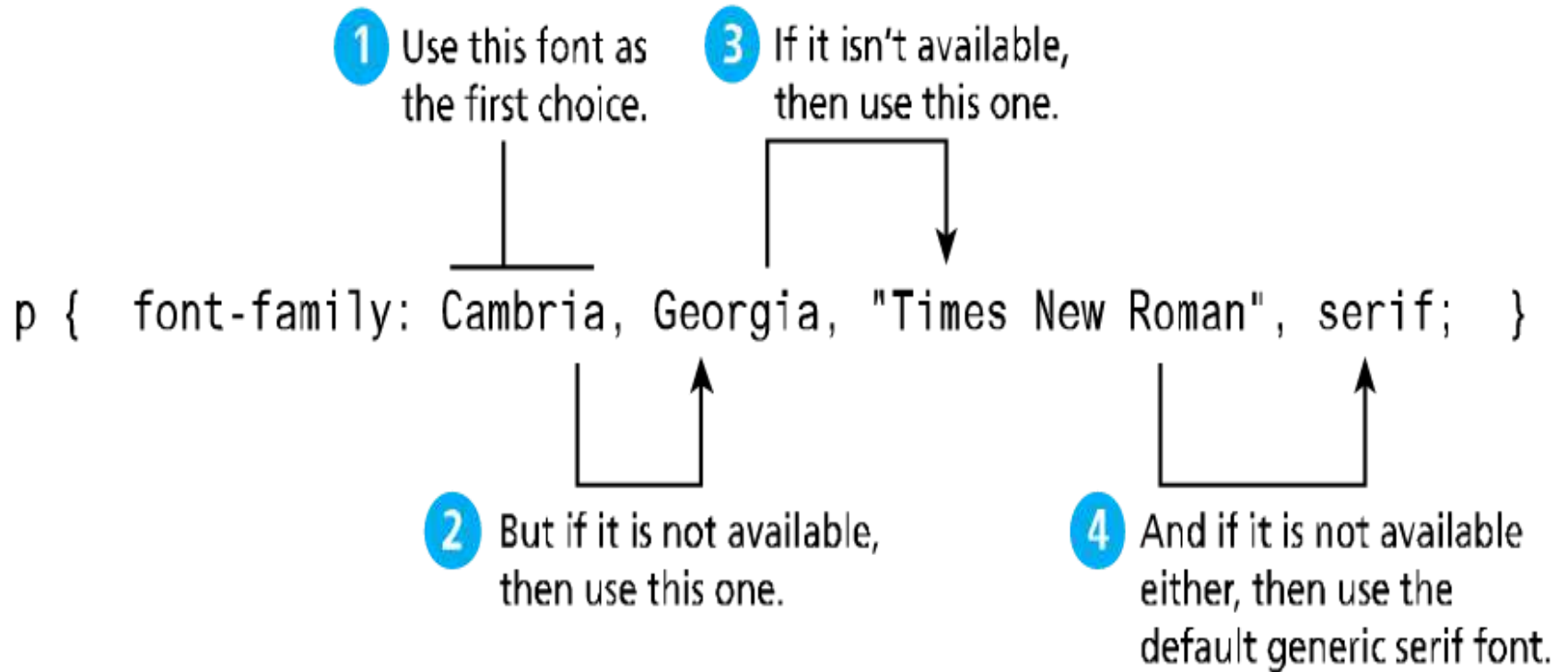
CSS Text Styling

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.


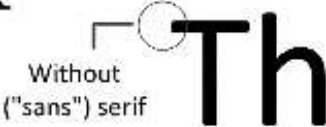
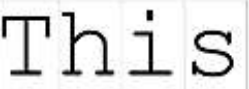
CSS Text Styling

Specifying the Font Family:



CSS Text Styling

Different Font Families:

	Generic Font-Family Name		
This	serif		
This	sans-serif		
This	monospace		In a monospace font, each letter has the same width.
This	cursive		
This	fantasy		In a regular, proportionally-spaced font, each letter has a variable width.

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

CSS Text Styling

Font Sizes:

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

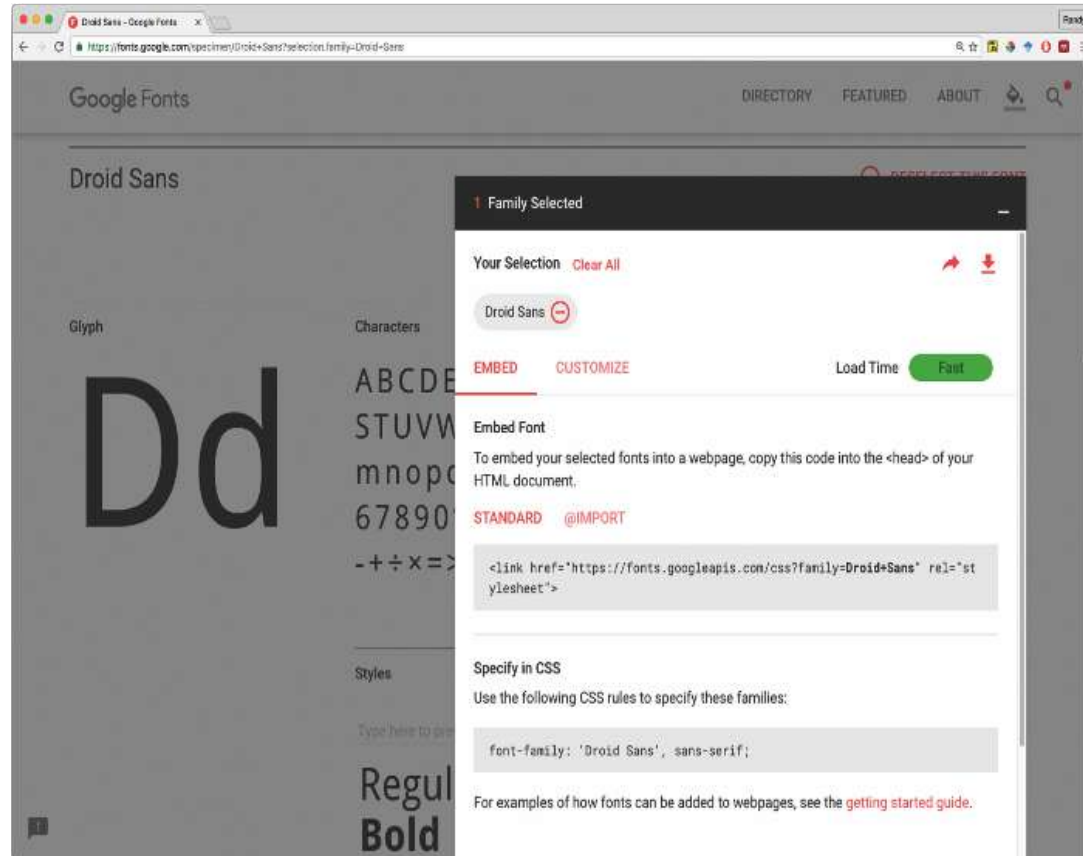
/ using 16px scale */*

```
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

Embedded Fonts:



The screenshot shows the Google Fonts website interface for the Droid Sans font family. The main content area displays the font name "Droid Sans" and a large "Dd" glyph. To the right, there are sections for "Characters" (showing "ABCDE", "STUVW", "mnop", "67890", and "- + ÷ × = >") and "Styles" (showing "Regul" and "Bold"). A modal window titled "1 Family Selected" is open, showing "Your Selection" as "Droid Sans" with a "Clear All" link. Below this, there are "EMBED" and "CUSTOMIZE" buttons, and a "Load Time" indicator showing "Fast". The "EMBED" section includes instructions: "To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document." and provides the following code snippet:

```
<link href="https://fonts.googleapis.com/css?family=Droid+Sans" rel="stylesheet">
```

 The "Specify in CSS" section provides the CSS rule:

```
font-family: 'Droid Sans', sans-serif;
```

 and a link to a "getting started guide".

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:



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

Diagram labels for the first code line:
- `20px` (x offset)
- `20px` (y offset)
- `10px` (blur size)
- `rgba(0,0,0,0.5)` (shadow color)

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

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



Web System and Technologies CS 521

TOPIC 31

Introducing Tables

Basic Table Structure:

The image displays four overlapping browser windows, each illustrating a different table structure:

- Top Left Window:** A pricing table with columns for 'Free', 'Basic', and 'Premium' plans. It lists features like 'Upload Space', 'Daily Uploads', 'Total Uploads', 'Social Sharing', and 'Analytics'.
- Top Middle Window:** An 'Artist Directory' table with columns for 'Artist', 'Work Details', 'Year', and 'Home'. It shows entries for 'Jacques-Louis David' and his works 'The Death of Marat' and 'The Intervention of the Sabine Women'.
- Bottom Left Window:** A 'Paintings' table with columns for 'Title', 'Artist', 'Year', and 'Genre'. It lists various artworks such as 'Death of Marat', 'Lictors Bearing to Brutus the Bodies of His Sons', and 'Liberty Leading the People'.
- Bottom Right Window:** A calendar table for 'October 2014', showing days of the week and dates from 1 to 31.

Introducing Tables

Basic Table Structure:

```
<table>
```

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

```
</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>
```



A screenshot of a web browser window titled "Chapter 5" with the address bar showing "figure05-03.html". The browser displays the rendered HTML table from the code block above. The table has five columns: Title, Artist, Year, Width, and Height. The first row contains the headers, and the following two rows contain the data for "The Death of Marat" and "Burial at Ornans".

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

Introducing Tables

Adding Headings:

```
<table>
```

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

```
</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

Spanning Columns:

Title	Artist	Year	Size (width x height)	
The Death of Marat	Jacques-Louis David	1793	162cm	128cm
Burial at Ornans	Gustave Courbet	1849	314cm	663cm

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	Title	Year
Jacques-Louis David	The Death of Marat	1793
	The Intervention of the Sabine Women	1799
	Napoleon Crossing the Alps	1800

```
<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>
  </tr>
  <tr>
    <td>The Intervention of the Sabine Women</td>
    <td>1799</td>
  </tr>
  <tr>
    <td>Napoleon Crossing the Alps</td>
    <td>1800</td>
  </tr>
</table>
```

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



The End



Web Systems and Technologies

TOPIC 32

Introducing Tables (2)

Introducing Tables

- Using Tables for Layout



```
<table>
<tr>
<td>

</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></td>
<td></td>
</tr>
<tr>
<td></td>
<td></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>
```



The screenshot shows a browser window titled 'Chapter 5' with the address bar containing 'figure05-06.html'. The page content is a table with the following data:

19th Century French Paintings		
Title	Artist	Year
The Death of Marat	Jacques-Louis David	1793
Burial at Ornans	Gustave Courbet	1849
Total Number of Paintings		2



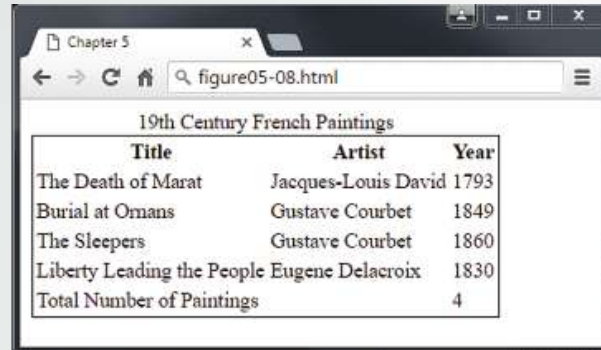
Web Systems and Technologies

TOPIC 33

Styling Tables

Styling Tables

- Table Borders

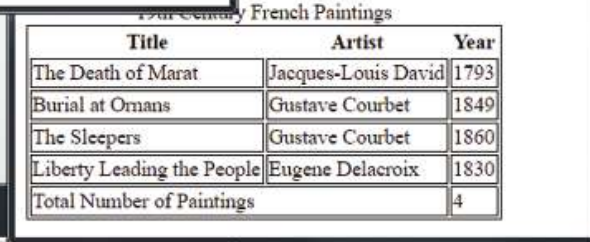


Chapter 5
figure05-08.html

19th Century French Paintings

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

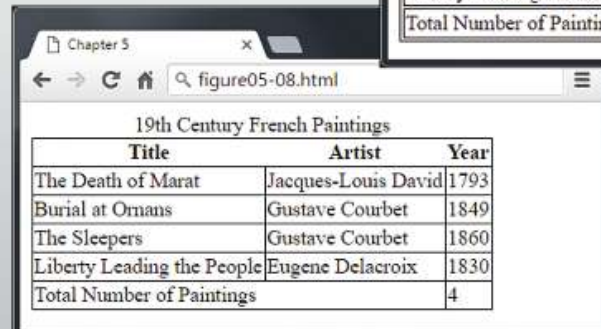
```
table {  
  border: solid 1pt black;  
}
```



19th Century French Paintings

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

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



Chapter 5
figure05-08.html

19th Century French Paintings

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

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

Styling Tables

- Table Borders

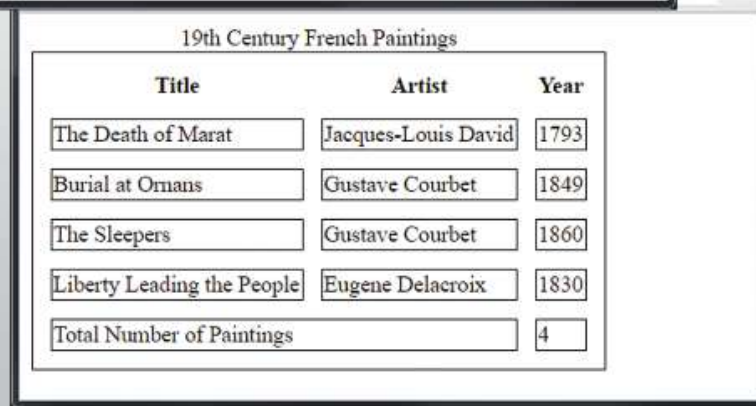


Chapter 5
figure05-08.html

19th Century French Paintings

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

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



19th Century French Paintings

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

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

Styling Tables

- Boxes



Chapter 5
figure05-09.html

19TH CENTURY FRENCH PAINTINGS

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

```
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



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

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

Styling Tables

- Boxes

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

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

Styling Tables

- Hover & Zebras

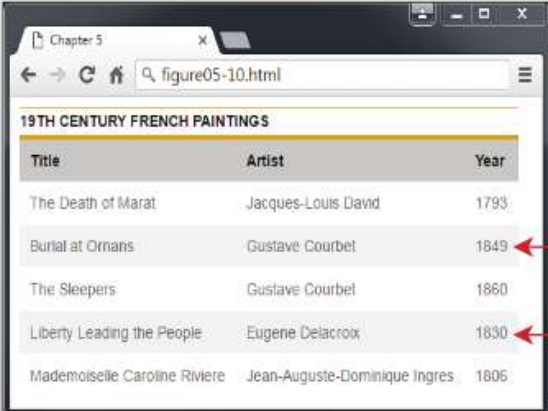


Chapter 5
figure05-10.html

19TH CENTURY FRENCH PAINTINGS

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

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



Chapter 5
figure05-10.html

19TH CENTURY FRENCH PAINTINGS

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

```
tbody tr:nth-child(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

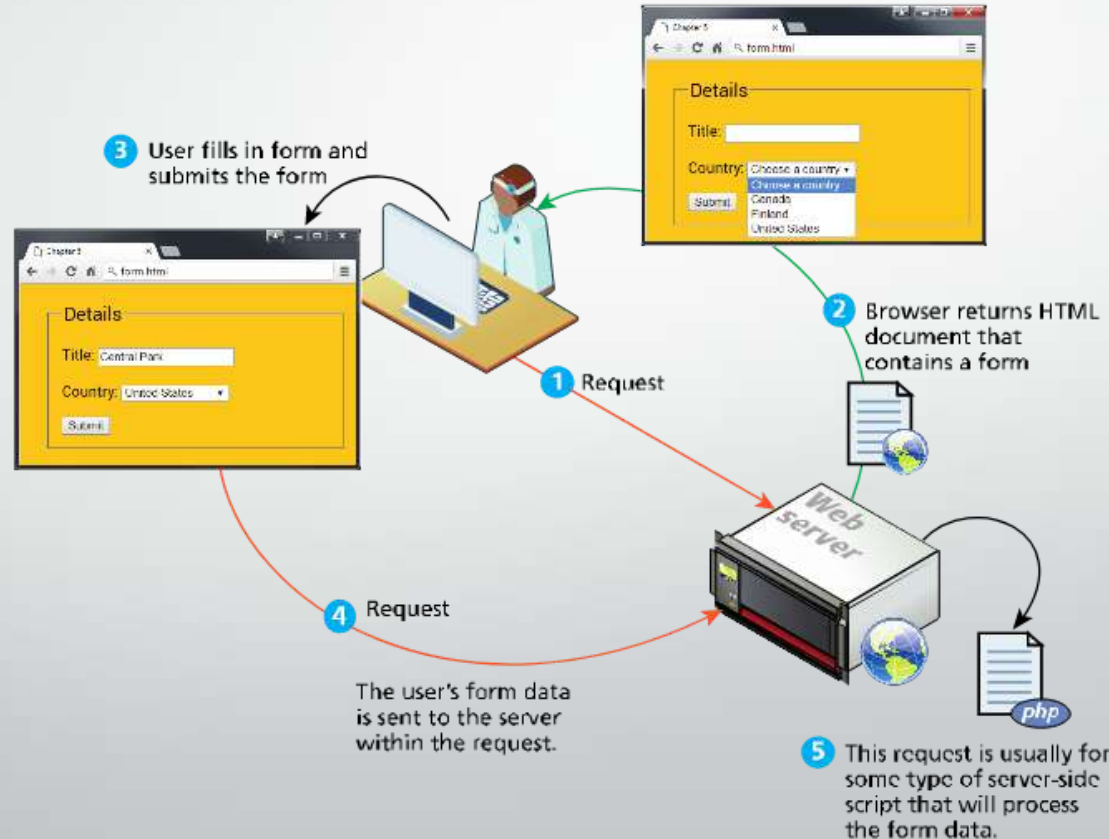
- Form Structure

The screenshot shows a web browser window with the address bar displaying 'form.html'. The page content is highlighted in yellow and contains a form with the following elements: a legend titled 'Details', a text input field labeled 'Title:', a dropdown menu labeled 'Country:' with a list of options including 'Choose a country', 'Canada', 'Finland', and 'United States', and a 'Submit' button.

```
<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

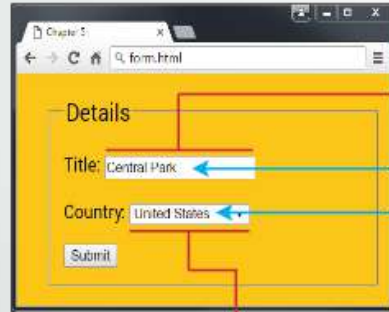
- How Forms Work



Introducing Forms

- Query Strings

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



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

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

Introducing Forms

- GET and POST



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

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

query string

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

```
POST /process.php http/1.1
Date: Sun, 21 May 2017 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47
```

HTTP Header

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

query string

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

Text:

textarea

TextArea: TextArea:

password

Password: Password:

search

Search: Search:

email

Email:

Please enter a valid email address

Email:

Please enter an email address

url

url:

Please enter a URL

tel

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]" />
```

Postal:

Postal:

! Please match the requested format.

Search City:

- Paris
- Prague

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

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

Form Control Elements

- Choice Controls (`<select>`)

Select:

```
<select name="choices">
```

```
<option>First</option>
```

```
<option selected>Second</option>
```

```
<option>Third</option>
```

```
</select>
```

Select:
First
Second
Third

Select:
First
Second
Third
Fourth

```
<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>
```

Cities:
North America
Calgary
Los Angeles
Europe
London
Paris
Prague

Form Control Elements

- Choice Controls `<select>` using value attribute

Select:

| | |
|--------|---|
| Second | ▾ |
| First | |
| Second | |
| Third | |

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

?choices=Second

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

?choices=2

Form Control Elements

- Button Controls (radio)

Continent:

- North America
- South America
- Asia

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

Form Control Elements

- Button Controls (checkbox)

I accept the software license

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

Where would you like to visit?

- Canada
- France
- Germany

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

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

Form Control Elements

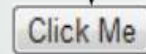
- Button Controls (checkbox)

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



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

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



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



```
<button>  
  <a href="email.html">  
      
    Email  
  </a>  
</button>
```

```
<button type="submit" >  
    
  Edit  
</button>
```



Web Systems and Technologies

TOPIC 36

Form Control Elements (2)

Form Control Elements

- Specialized Controls (file upload)

Upload a travel photo
 No file chosen



Upload a travel photo
 IMG_0020.JPG

```
<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)

Rate this photo:
2

Grumpy ————— Ecstatic

```
<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
```

Rate this photo:

Grumpy 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

Date:



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

Time:

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

DateTime:

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

DateTime Local:

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

Form Control Elements

- Date and Time Controls

Month:

March, 2013

March, 2013

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

This month Clear

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

Week:

2013-W10

March 2013

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

Today

```
<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>
```

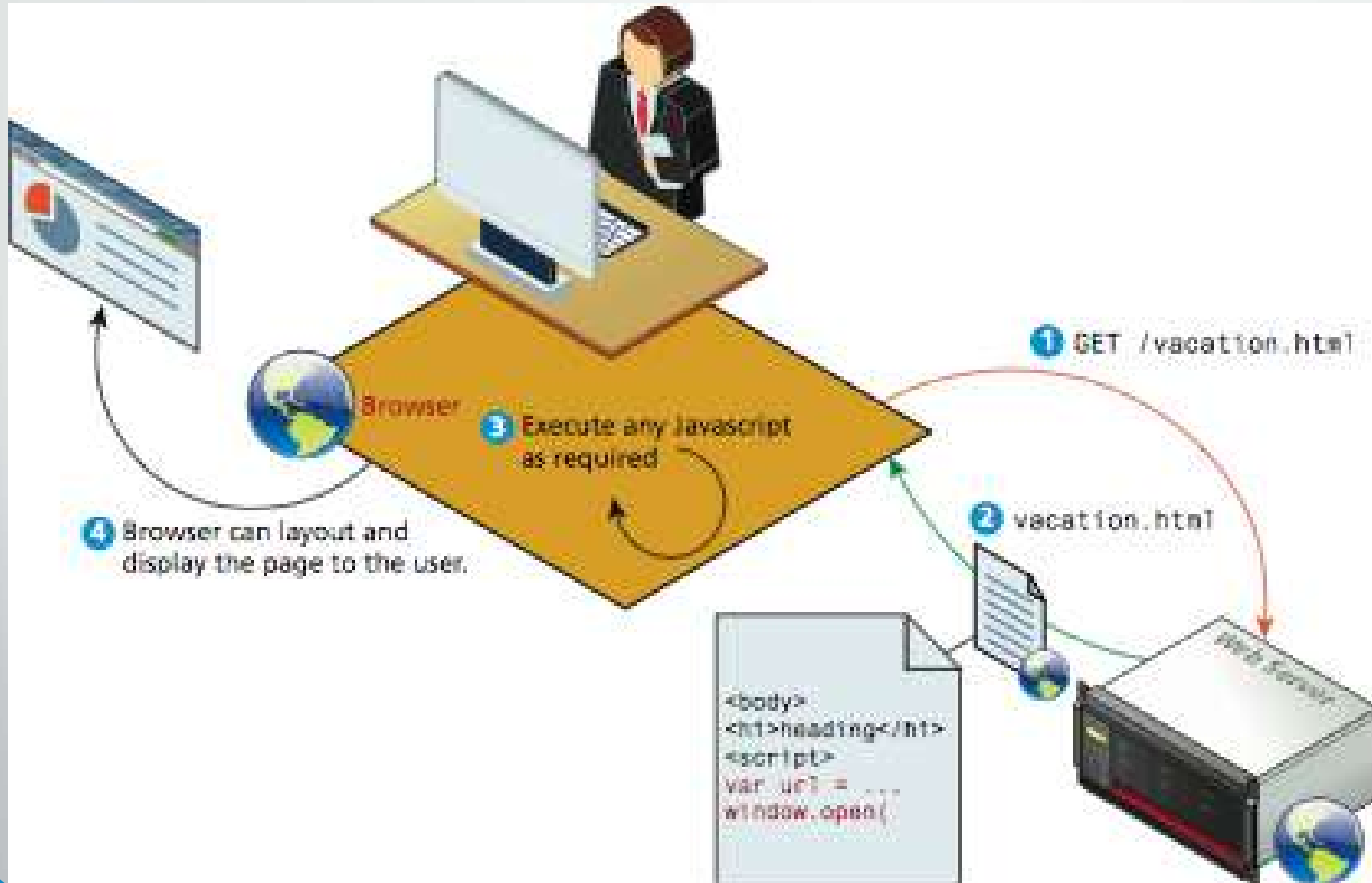


Web System and Technologies CS 521

WEEK 05 TOPIC 37 TO 45

Introduction to JavaScript

Client Side Scripting



Introduction to JavaScript

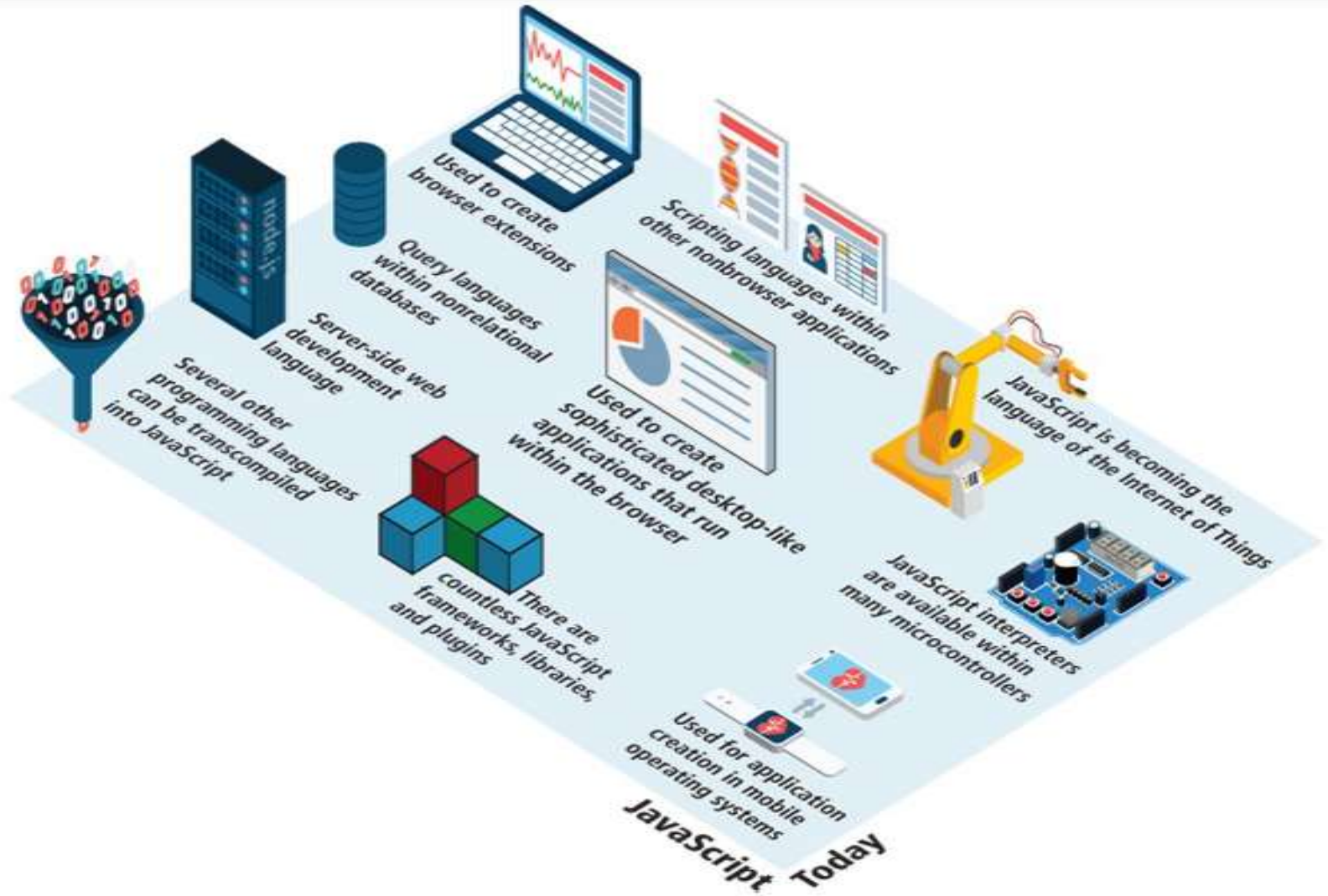
JavaScript History

- 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





Web System and Technologies CS 521

Writing and Storing the Script

Inline JavaScript

- Inline JavaScript refers to the practice of including JavaScript 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

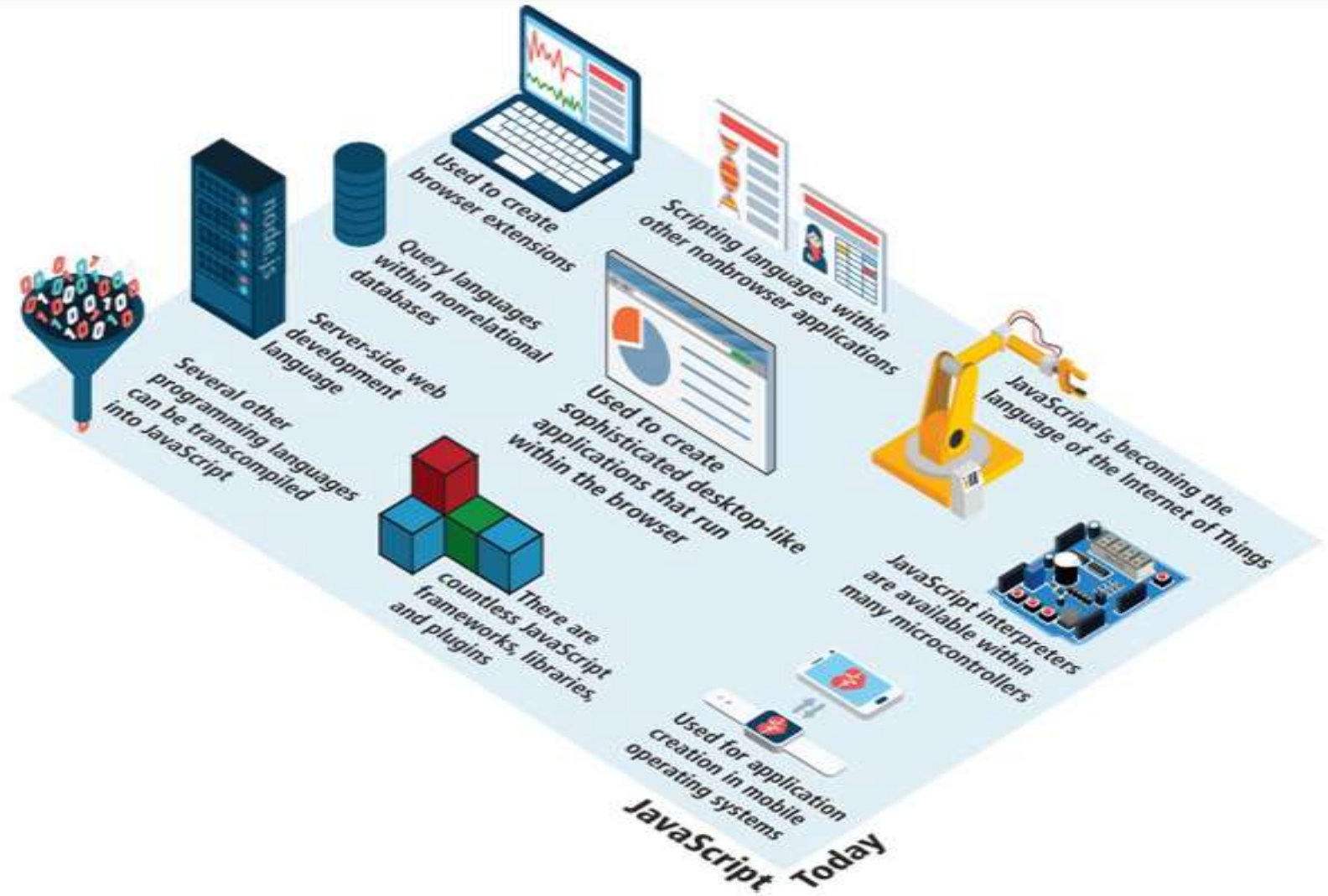
JavaScript History

- 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



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

Defines a variable named `abc`

```
var abc;
```

Each line of JavaScript should be terminated with a semicolon

```
var def = 0;
```

A variable named `def` is defined and initialized to `0`

```
def = 4 ;
```

`def` is assigned the value of `4`

Notice that whitespace is unimportant

```
def =  
"hello" ;
```

`def` is assigned the value of `"hello"`

Notice that a line of JavaScript can span multiple lines

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)

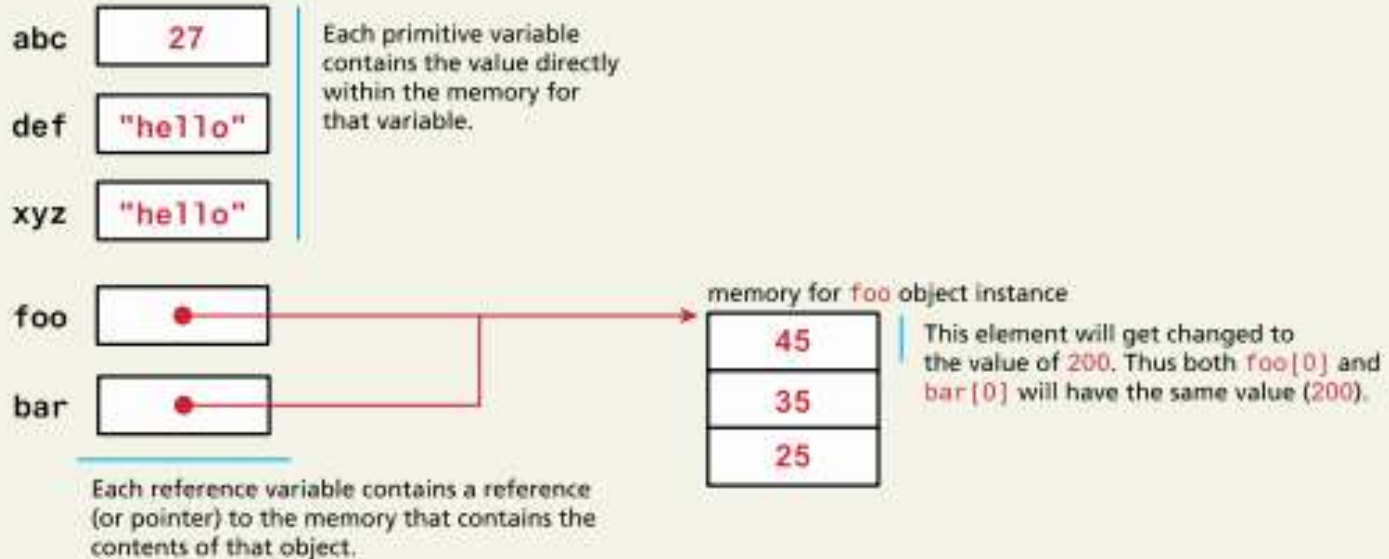
```
var xyz = def;  
var bar = foo;
```

these new variables differ in important ways
(see below)

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

changes value of the first element of array

Memory representation



JavaScript Output



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

The screenshot shows a browser window with the address bar displaying 'file:///T:/Comp...write-test.html'. The page content is 'here in the head in the body' followed by a large 'Heading'. The Browser Inspector is open, showing the DOM tree with the following structure:

```
<html>
  <head>
    <script></script>
  </head>
  <body>
    here in the head
    <meta charset="UTF-8"></meta>
    <link href="styles.css"></link>
    <script></script>
    in the body
    <h1></h1>
  </body>
</html>
```

The console shows the rendered HTML content, including the 'here in the head' text and the 'Heading' element.

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)

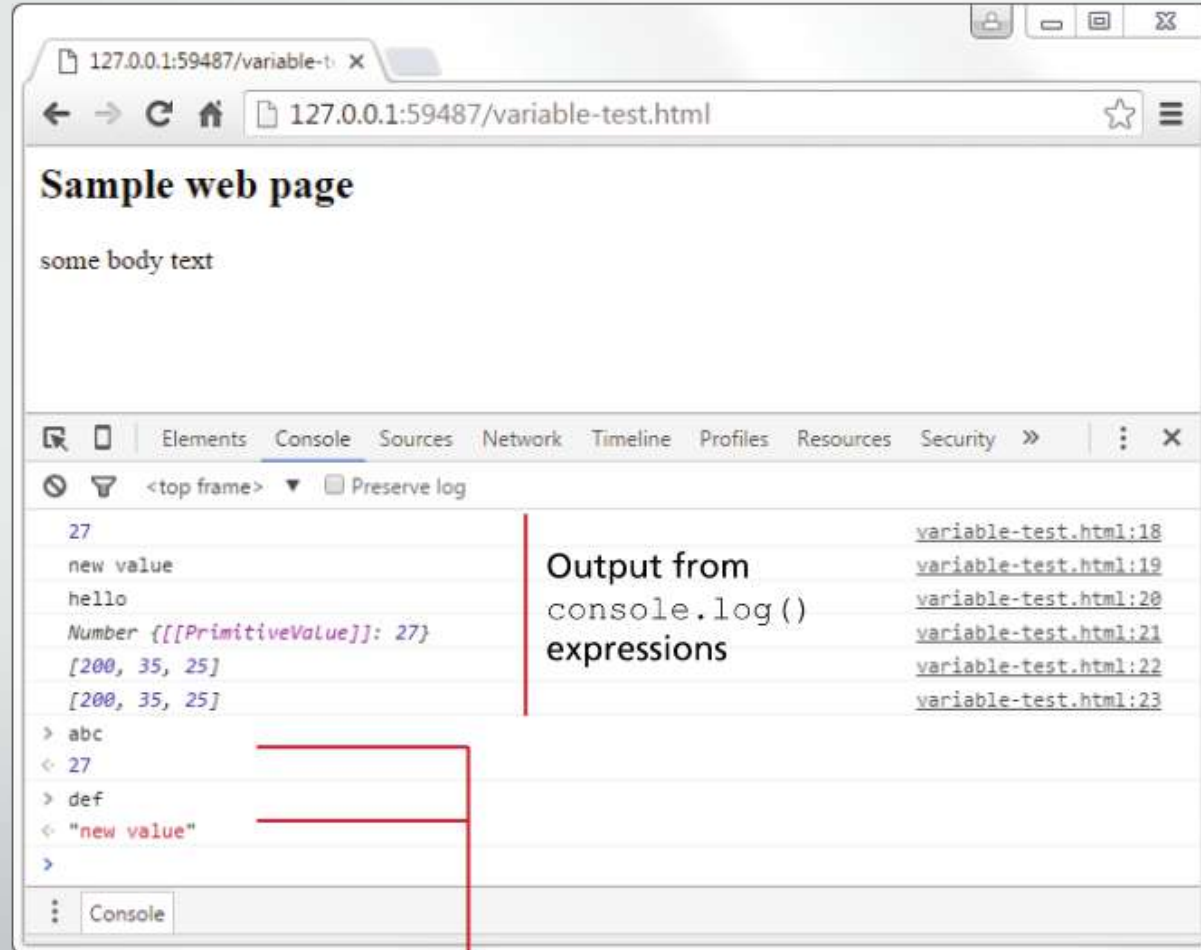
The appearance of this line will shift the following write() calls to the <body>

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



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";
```

<u> </u>	<u> </u>	<u> </u>
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

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

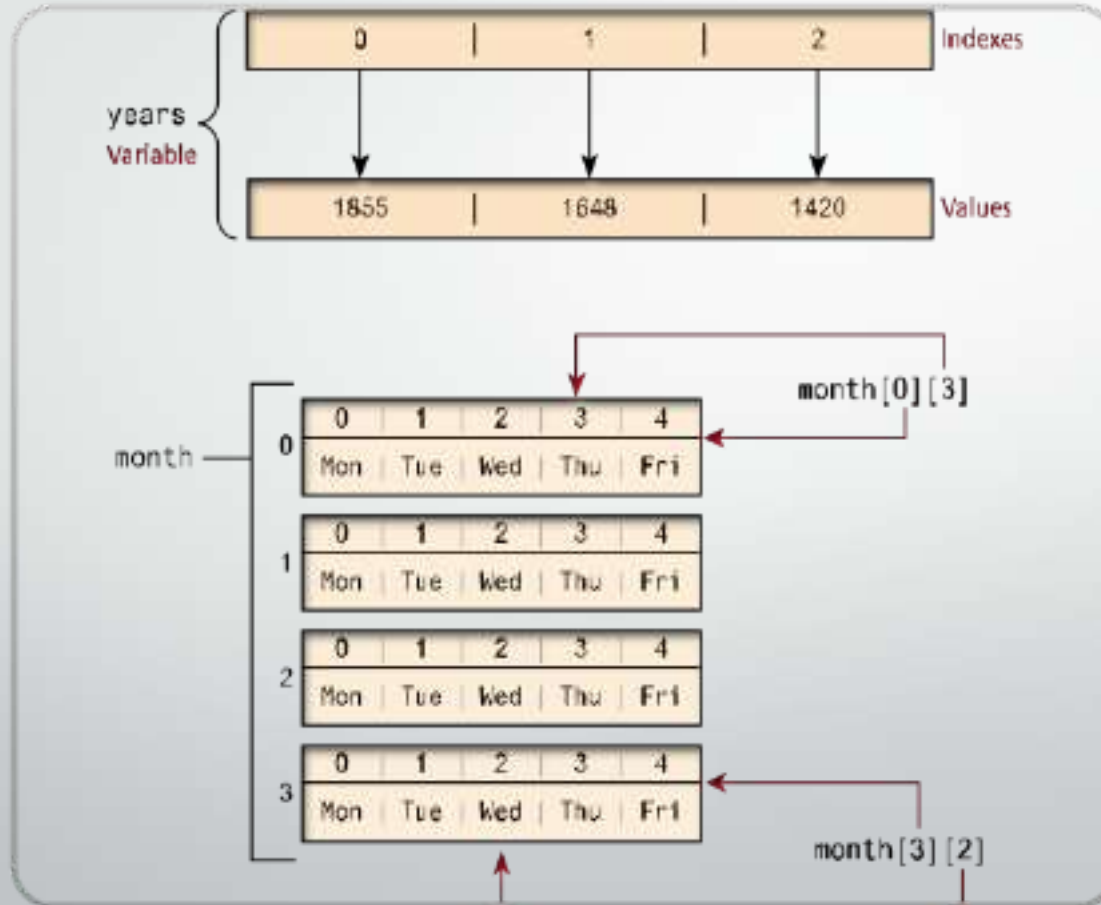
.push()

.pop()

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

Arrays

- Arrays Illustrated



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

```
      initialization   condition   post-loop operation  
      _____     _____     _____  
for (var i = 0; i < 10; i++) {  
    // do something with i  
    // ...  
}
```

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);  
})  
}
```

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

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
```



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 Elements



Functions (2)

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

```
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;  
  };  
}
```

BUT

Variable assignment is **not** hoisted

THUS

The value of the calculateTax variable here is **undefined**



Callback Functions

Callback Functions

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

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

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

1 Passing the `calcTax()` function object as a parameter

We can say that `calcTax` variable here is a **callback function**

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

Callback Functions

```
var temp = calculateTotal( 50, 2,
```

Passing an **anonymous function definition**
as a callback function parameter

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

```
);
```



Functions - Objects

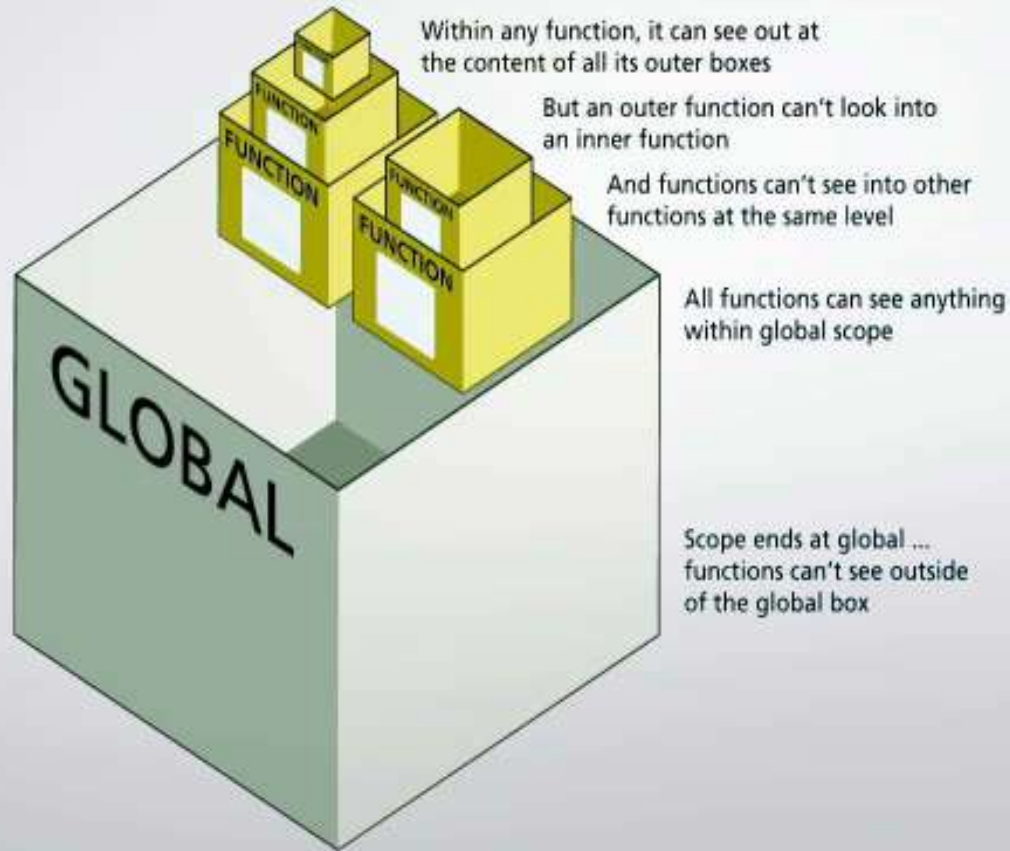
Objects and Functions Together

```
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 in JavaScript

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



Anything declared inside this block is global and accessible everywhere in this block

global variable `c` is defined
global function `outer()` is called

```
1 var c = 0;  
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() {
```

```
5 console.log(a);
```

✓ allowed

outputs 5

```
6 var b = 23;
```

```
7 c = 37;
```

✓ allowed

```
}
```

local (outer) variable `a` is accessed
local (inner) variable `b` is defined
global variable `c` is changed

```
3 var a = 5;
```

```
4 inner();
```

```
8 console.log(c);
```

✓ allowed

outputs 37

```
9 console.log(b);
```

✗ not allowed

generates error or
outputs `undefined`

local (outer) variable `a` is defined
local function `inner()` is called
global variable `c` is accessed
undefined variable `b` is accessed

```
}
```

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

1 A brand new empty object is created and given the name `cust`

```
var cust = new Customer("Sue", "123 Somewhere", "Calgary");
```

2 Then the function is called

Note: it is a coding convention to capitalize the first letter of a constructor function

```
function Customer(name, address, city) {  
  this.name = name;  
  this.address = address;  
  this.city = city;  
}
```

3 `this.name = name;`

The new empty object is set as the context for `this`. Thus, the new empty object gains these property values.

4 Since there is no return, the function will end with the (no longer empty) new object being assigned to the `cust` variable



Exception Handling

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

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')
 - toString(): converts a date object to a string in human readable form

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

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))
```



Object Prototypes

Introduction

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

Methods Get Duplicated



x1: Die

```
this.color = "red";  
this.faces = [1,2,3,4,5,6];
```

```
this.randomRoll = function() {  
  var randNum = ...;  
  return faces[randNum-1];  
};
```

A function expression is an object whose content is the definition of the function ...



x2: Die

```
this.color = "green";  
this.faces = [1,2,3,4,5,6];
```

```
this.randomRoll = function() {  
  var randNum = ...;  
  return faces[randNum-1];  
};
```

so each instance will contain that same content ...



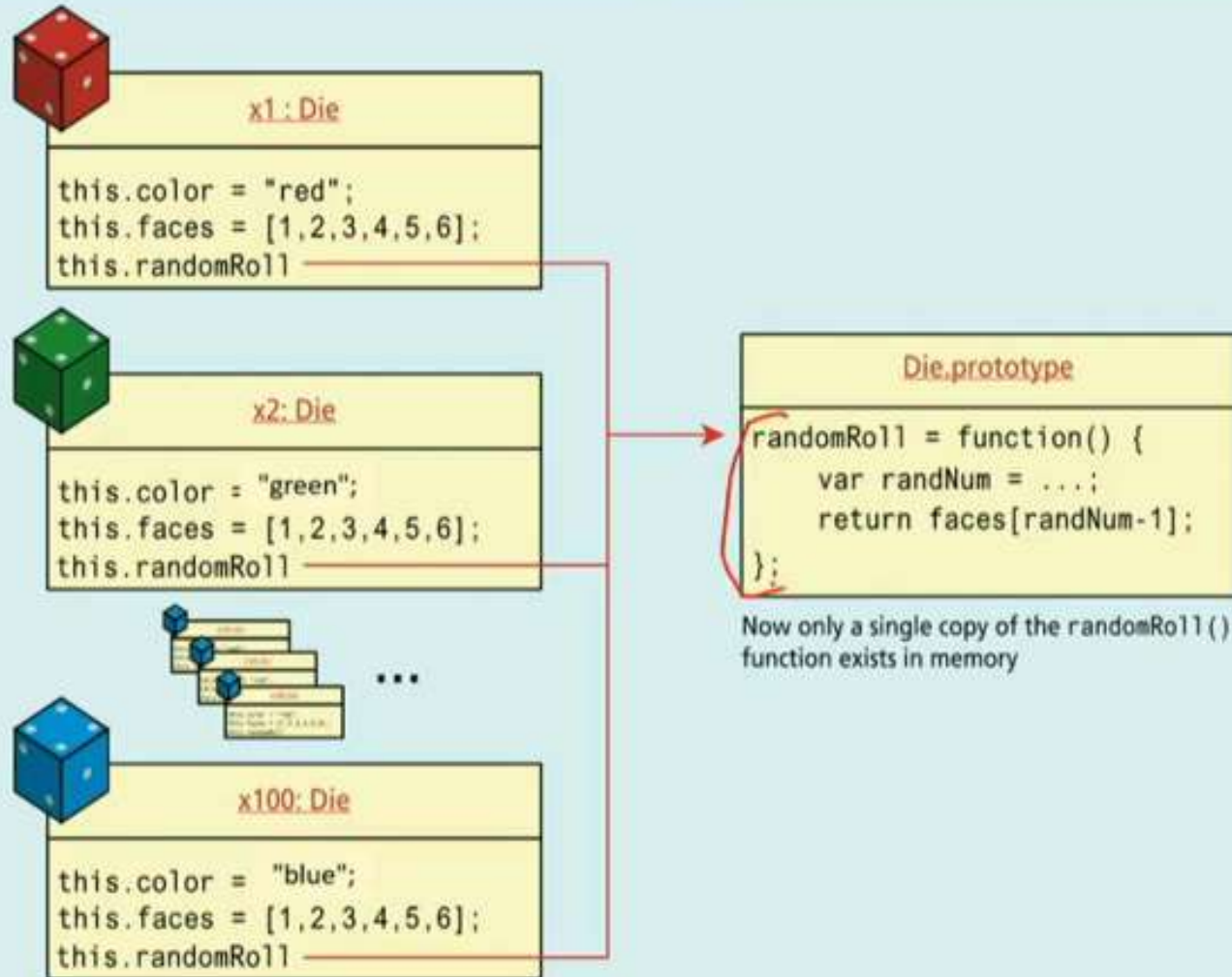
x100: Die

```
this.color = "blue";  
this.faces = [1,2,3,4,5,6];
```

```
this.randomRoll = function() {  
  var randNum = ...;  
  return faces[randNum-1];  
};
```

which is incredibly memory inefficient when there are many instances of that object

Using Prototype reduces Duplication at Runtime



Now only a single copy of the `randomRoll()` function exists in memory

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");
```

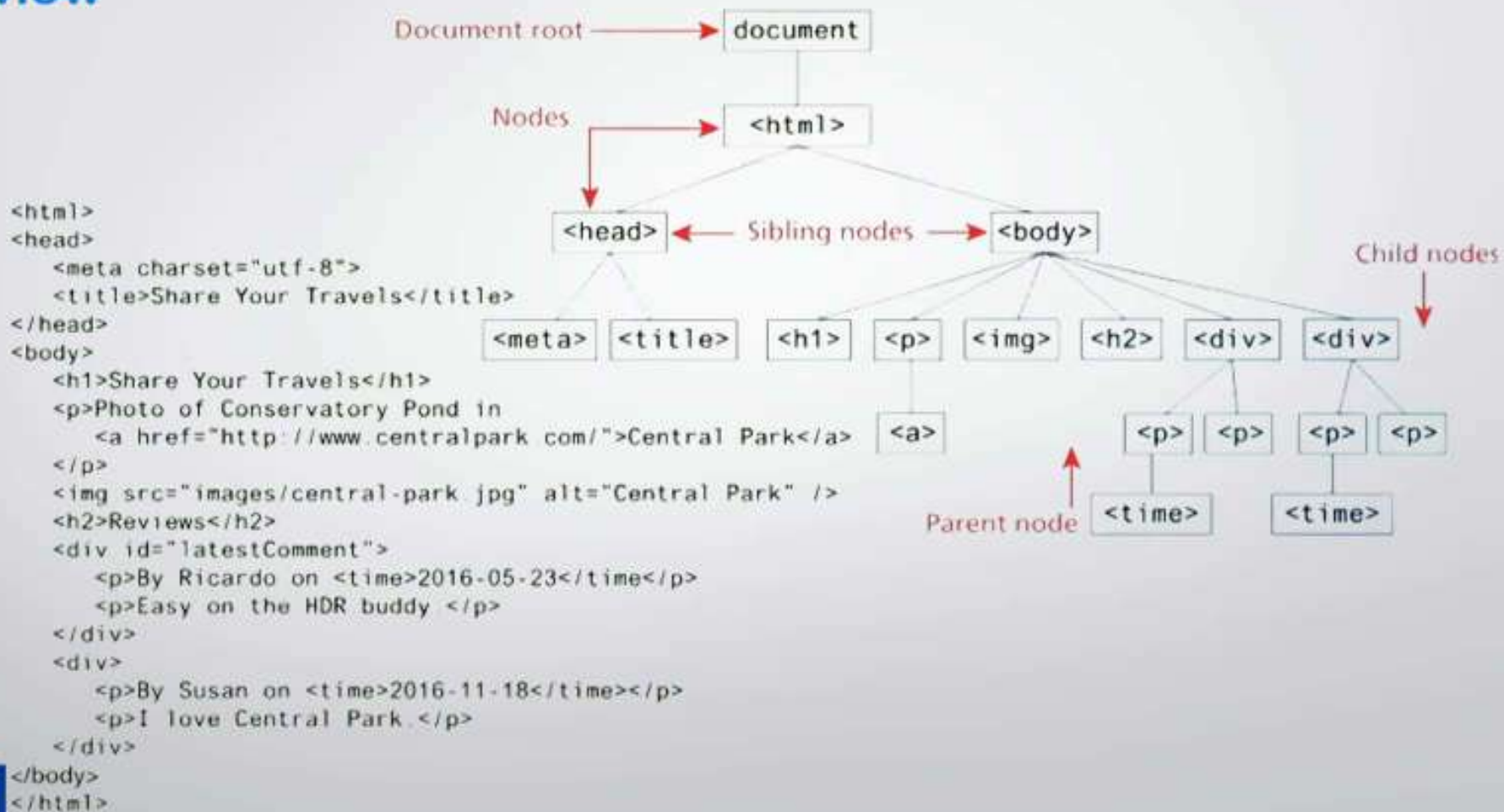


The Document Object Model (DOM)

Introduction

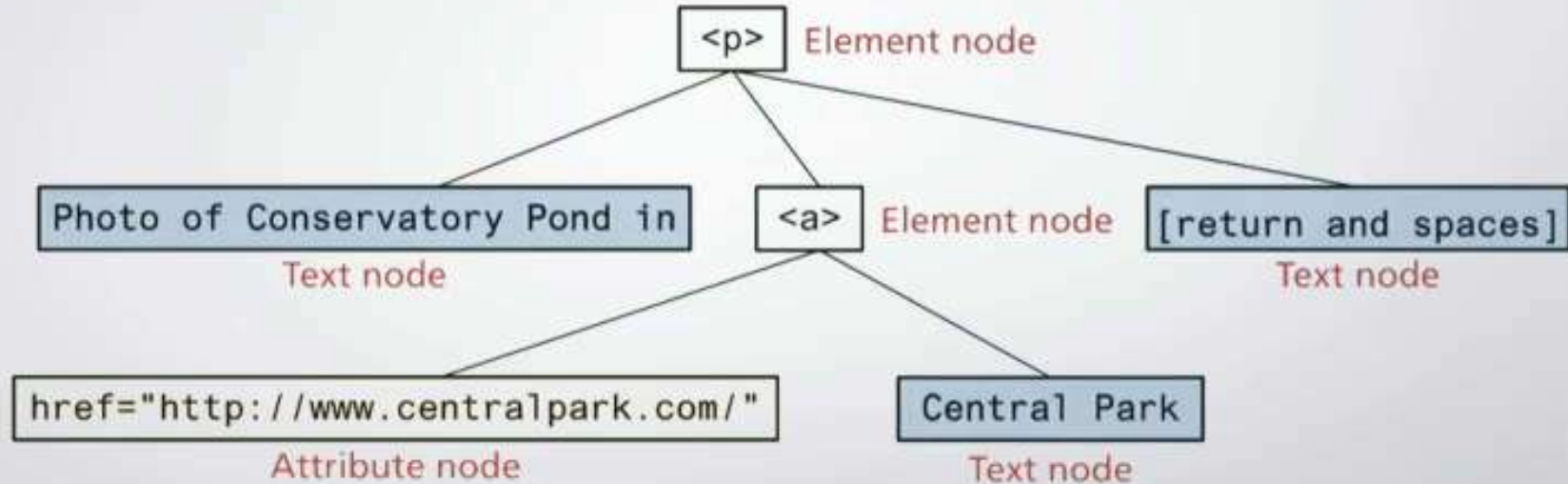
- Any HTML document can be represented in form of a tree called Document Object Model
- Available as a data structure in JavaScript
- Helps in accessing and manipulating HTML elements through JavaScript

Overview



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

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.getElementsByClassName("comment");
```

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



Query Selector

querySelectorAll("nav ul a:link")

```
<body>
  <nav>
    <ul>
      <li><a href="#">Canada</a></li>
      <li><a href="#">Germany</a></li>
      <li><a href="#">United States</a></li>
    </ul>
  </nav>
```

querySelectorAll("#main div time")

querySelector("#main>time")

```
<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>
```

querySelector("footer")

```
<footer>
  <ul>
    <li><a href="#">Home</a> | </li>
    <li><a href="#">Browse</a> | </li>
  </ul>
</footer>
```

</body>

Slide material provided by Connolly & Hoar, "Fundamentals of Web Development" -

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