WEB TECHNOLOGIES

Introduction to WWW, Web Protocols and URLs

Vinay Joshi
Department of
Computer Science and Engineering

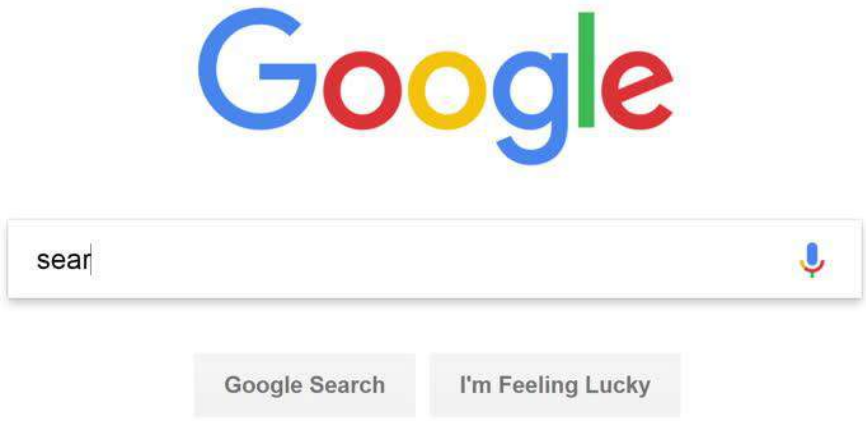
Introduction to WWW, Web Protocols and URLs

Common Terms

- Internet vs. Web
- Web Browsers
- URL
- Web Server
- DNS
- HTTP Protocol
- HTTPS

Introduction to WWW, Web Protocols and URLs

Internet vs. WWW

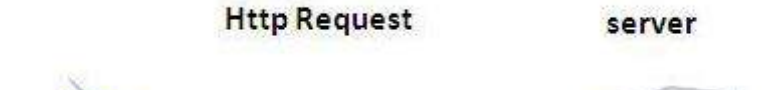


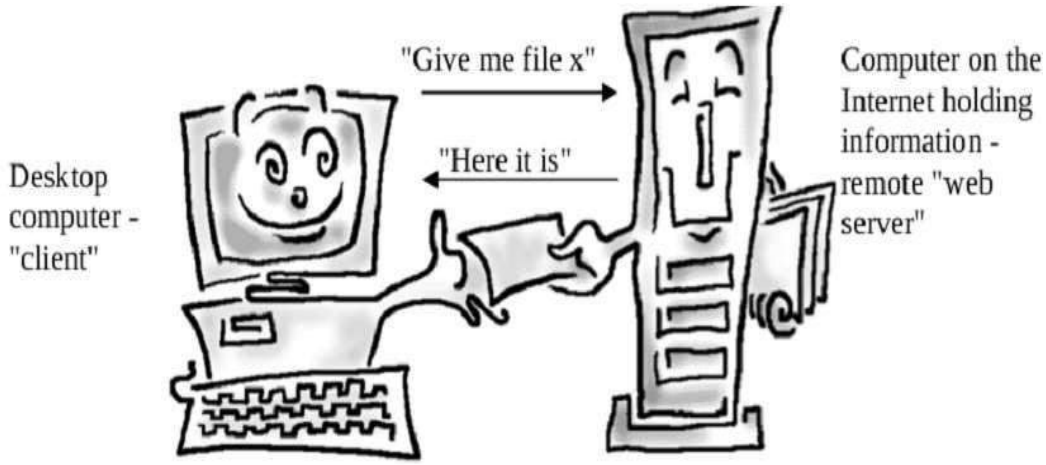
Introduction to WWW, Web Protocols and URLs

How does WWW work?

1. Client/Server Architecture

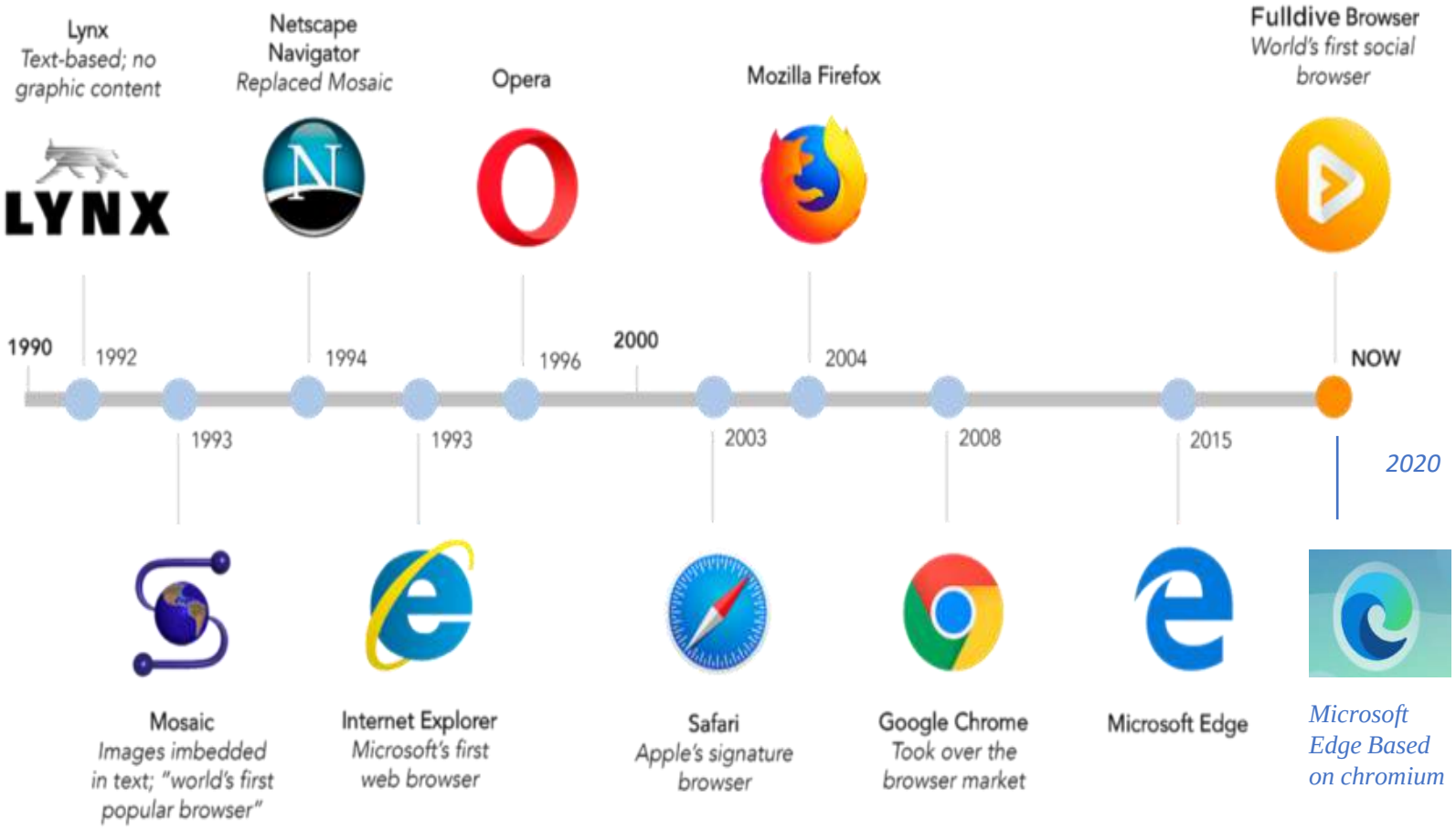
2. Request/Response Pattern





Introduction to WWW, Web Protocols and URLs

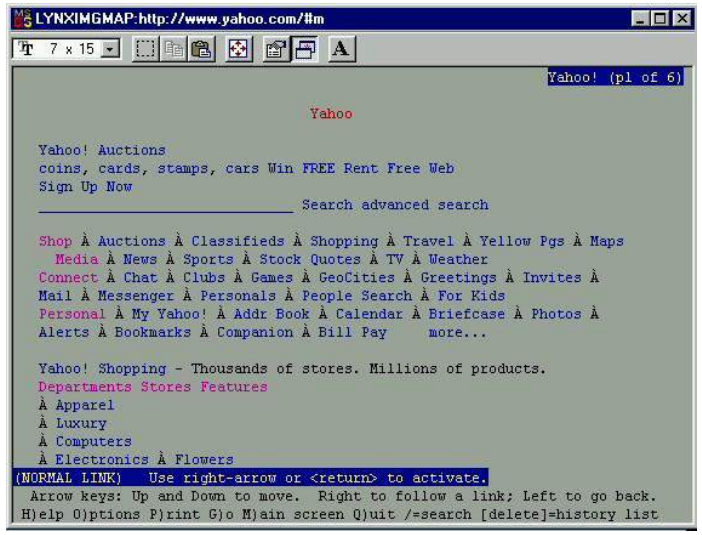
History of Web Browsers



Source:A Brief History of Browsers https://medium.com/@fulldive/a-brief-history-of-browsers-57669527c0cf

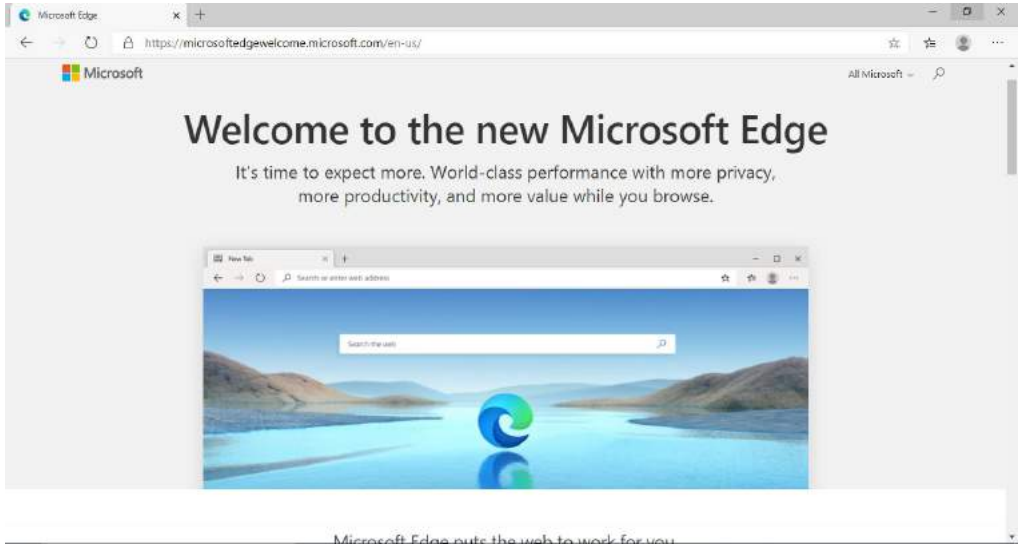
Introduction to WWW, Web Protocols and URLs

Browser Evolution



Lynx – A text based browser

Mosaic – the first graphical browser



Source: Browser Museum
http://www.donmouth.co.uk/web_design/browsermuseum/browsermuseum.html

URLs

- URL stands for Uniform Resource Locator
- General form:

  scheme:object-address

- For the http protocol, the object-address is:

  fully qualified domain name/doc path

Example:

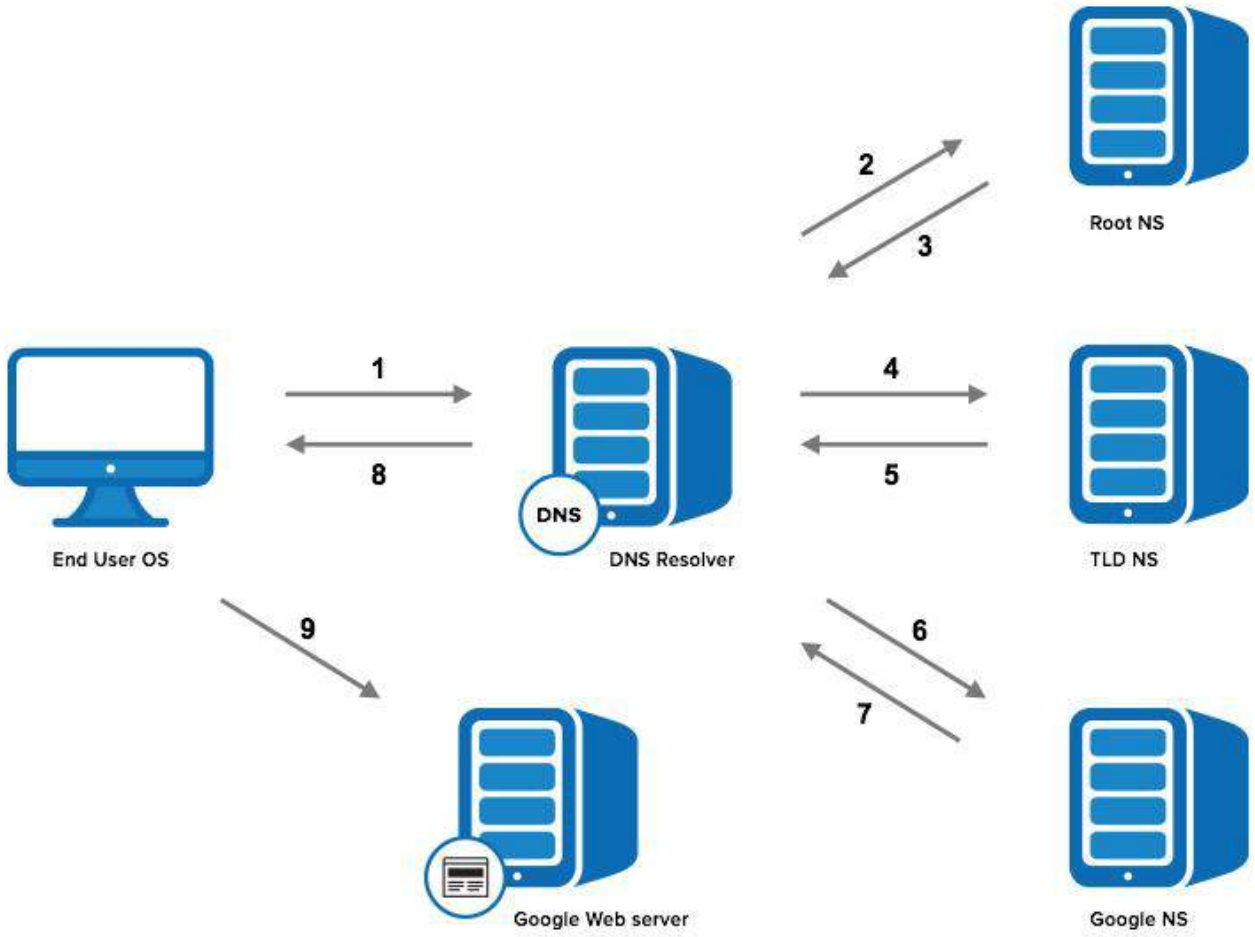
https://www.amazon.com/international-sales-offers.html

Web Servers

- General Web Server Characteristics
  - Web servers have two main directories:
    1. Document root (servable documents)
    2. Server root (server system software)
  - Document root is accessed indirectly by clients
    - Its actual location is set by the server configuration file
    - Requests are mapped to the actual location
- Popular Examples
  - Apache
  - IIS

Introduction to WWW, Web Protocols and URLs

Domain Name Service



Step 1: OS Recursive Query to DNS Resolver
Step 2: DNS Resolver Iterative Query to the Root Server
Step 3: Root Server Response
Step 4: DNS Resolver Iterative Query to the TLD Server
Step 5: TLD Server Response
Step 6: DNS Resolver Iterative Query to the Google.com NS
Step 7: Google.com NS Response
Step 8: DNS Resolver Response to OS
Step 9: Browser Starts TCP Handshake

Introduction to WWW, Web Protocols and URLs

How to get your own website?

PES UNIVERSITY ONLINE

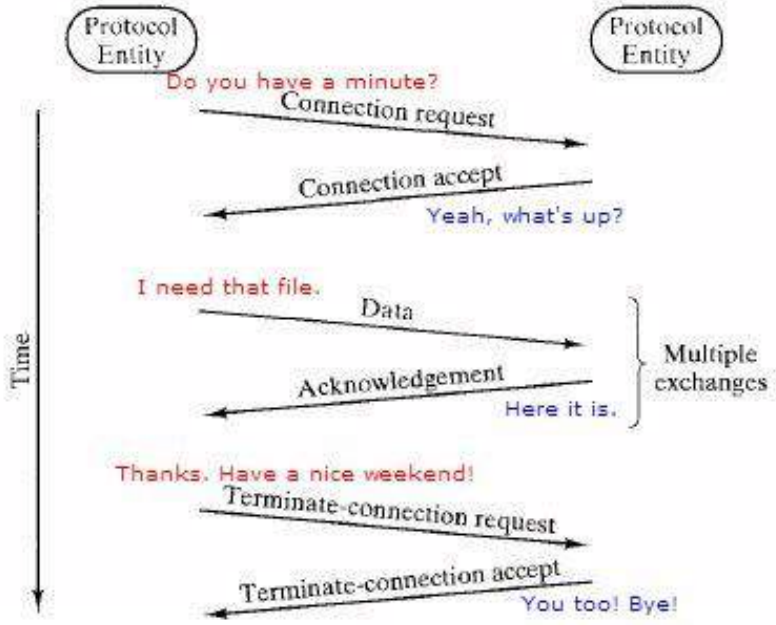
Steps:

1. Choose a domain name
2. Register a domain and sign up with web hosting
3. Set up a website using WordPress/Name cheap/Go Daddy (through web host)
4. Customize your website design and structure
5. Add pages and content to your website

Introduction to Web Protocols and HTTP

What is a Protocol?

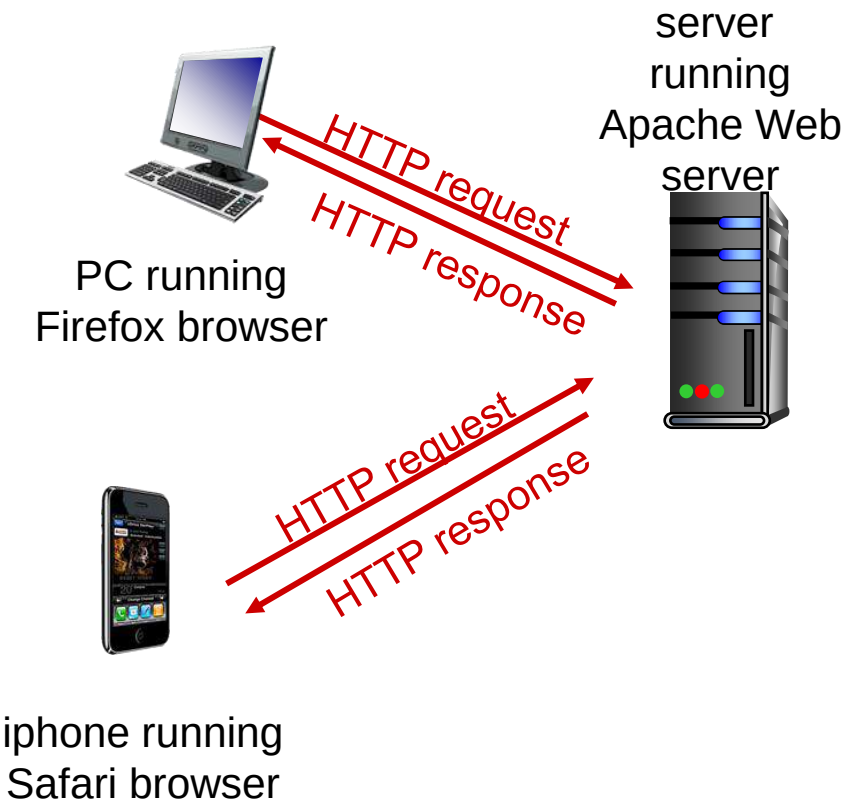
- A protocol is a set of rules and guidelines for communicating data.
- Different applications use different protocols
- The web, in particular, uses multiple protocols to communicate.
- The most important and visible protocols are HTTP and HTTPS.



HTTP Overview

HTTP: HyperText Transfer Protocol

- Application Protocol used by the Web
- Client/Server model
  - *Client:* browser that requests, receives, and "displays" Web Objects
  - *Server:* Web server sends Web Objects (using HTTP protocol) in response to requests



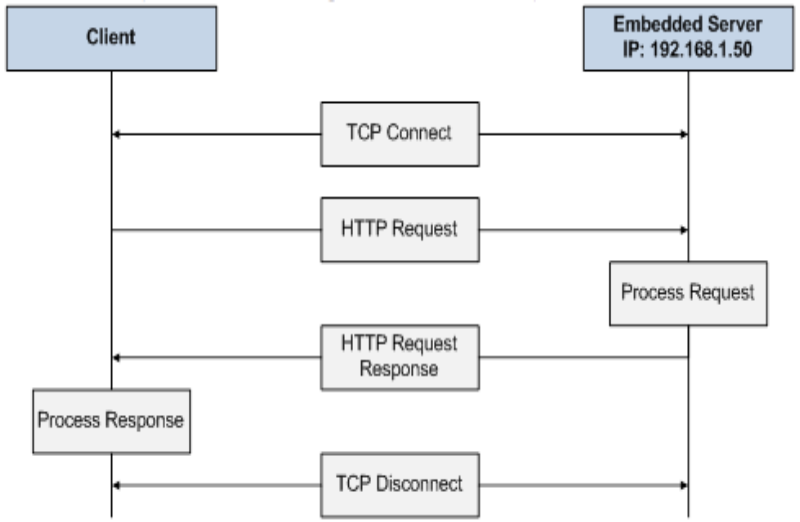
HTTP Overview…(cntd.)

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests



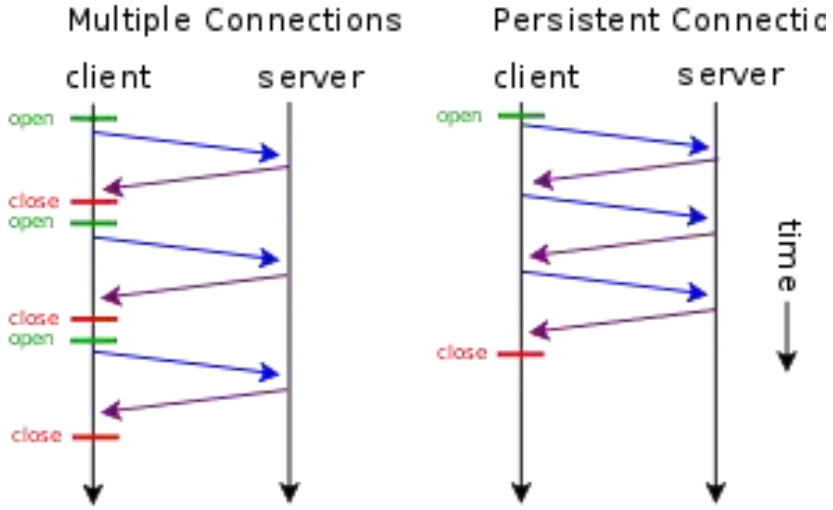
HTTP Connections

non-persistent HTTP

- at most one object sent over TCP connection
  - connection is then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client, server



HTTP Requests

- HTTP request is a *request line*, followed by zero or more *request headers*
- Request line: <method> <uri> <version>
  - <version> is HTTP version of request (HTTP/1.0 or HTTP/1.1)
  - <uri> is typically URL for proxies, URL suffix for servers.
  - <method> is either GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE.
- Request Header
- Blank line (CRLF)
- Message Body

```
GET /test.html HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01;
Windows 98)
Host: euro.ecom.cmu.edu
Connection: Keep-Alive
CRLF (\r\n)
```

HTTP Request Methods

- HTTP methods:
  - GET: Retrieve static or dynamic content
  - POST: Send content to server through request body
  - OPTIONS: Get server or file attributes
  - HEAD: Fetches only header field without any response body
  - PUT: Write a file to the server
  - DELETE: Delete a file on the server

Introduction to Web Protocols and HTTP

HTTP Response

- HTTP response is a response line followed by zero or more response headers.
- Response line:
- <version> <status code> <status msg>
  - <version> is HTTP version of the response.
  - <status code> is numeric status.
- Response headers:

  <header name>: <header data>
  - Provide additional information about response
  - Content-Type: MIME type of content in response body.
  - Content-Length: Length of content in response body.

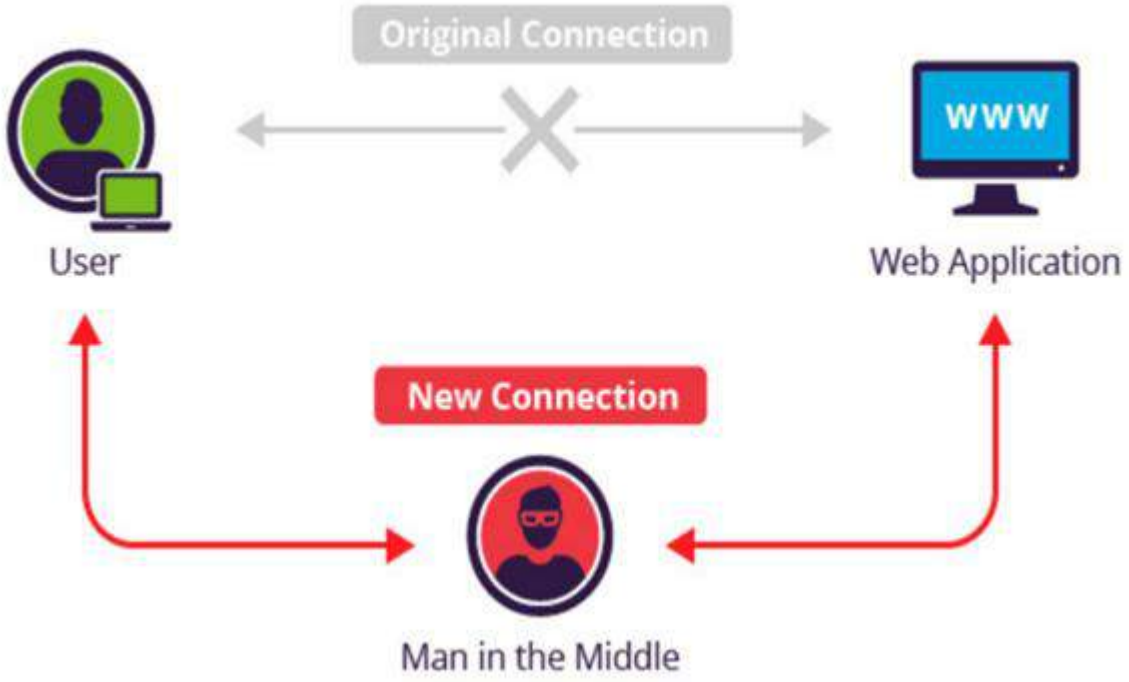
```
HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 04:02:15 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body>
<h1>Test page</h1>
</html>
```

HTTP Response : Status Codes

- Three-digit number; first digit specifies the general status
  - 1 => Informational
  - 2 => Success
  - 3 => Redirection
  - 4 => Client error
  - 5 => Server error
- <status msg> is corresponding English text.
  - 200 OK => Request was handled without error
  - 403 Forbidden => Client lacks permission to access file
  - 404 Not found => Server couldn’t find the file.

HTTP Secure (HTTPS)

- A common security attack
- Need to encrypt data to save it from such attacks



THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

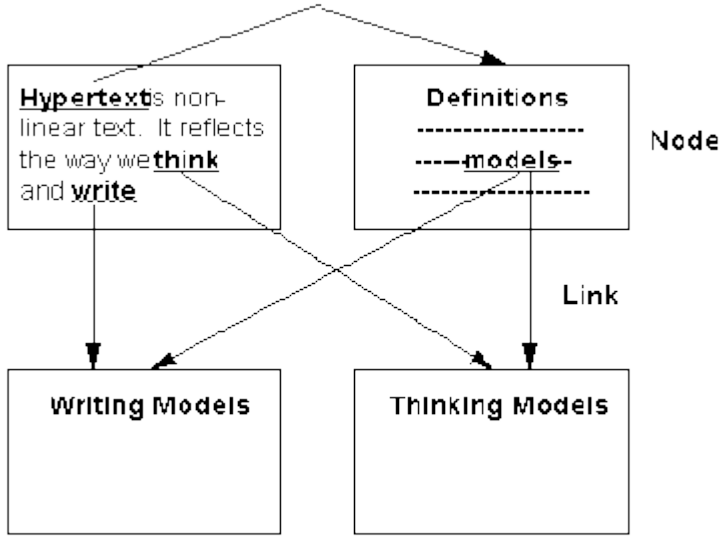
WEB TECHNOLOGIES

HTML - Basic Markups

Vinay Joshi
Department of
Computer Science and Engineering

HTML – Hyper Text Markup Language

Hypertext - cross-referencing /linking between related sections of text and associated graphic material



Markup - define elements within a document using tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recipe PUBLIC "-//Happy-Monkey//DTD RecipeBook//EN"
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

    <title>Peanut-butter On A Spoon</title>

    <ingredientlist>
        <ingredient>Peanut-butter</ingredient>
    </ingredientlist>

    <preparation>
        Stick a spoon in a jar of peanut-butter,
        scoop and pull out a big glob of peanut-butter.
    </preparation>

</recipe>
```

Basic Syntax

- Elements are defined by tags (markers)
  - Tag format:
    - Opening tag: **<tag_name>**
    - Closing tag: **</tag_name>**

    Syntax:

    <tag_name>

    Content…

    </tag_name>
- Not all tags have content
  - If a tag has no content, its form is **<tag_name … />**
- The container and its content together are called an *element*

HTML – Basic Markups

Comments

- An HTML comment begins with **<! – –** and the comment closes with **– –>**.
- HTML comments are visible to anyone that views the page source code, but are not rendered when the HTML document is rendered by a browser.

```
You will be able to see this text.

<!-- You will not be able to see this text. -->

You can even comment out things in <!-- the middle of --> a sentence.

<!--
Or you can
comment out
a large number of lines.
-->

<div class="example-class">
Another thing you can do is put comments after closing tags, to help you find
where a particular element ends. <br>
(This can be helpful if you have a lot of nested elements.)
</div> <!-- /.example-class -->
```

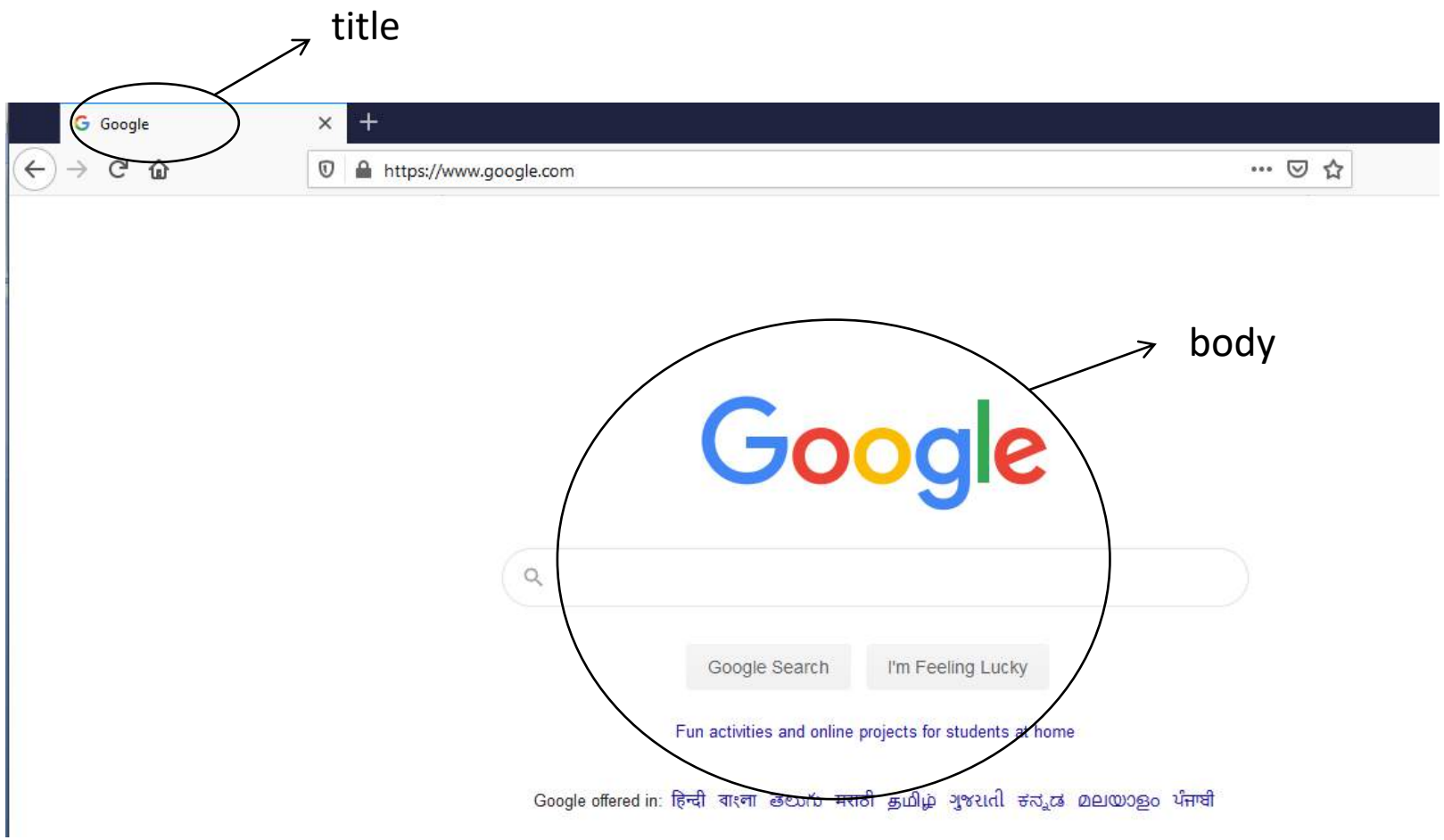
HTML – Basic Markups

Document Structure

```
<html>
  <head>
      <title>… </title>
  </head>
  <body>
      …
  </body>
</html>
```

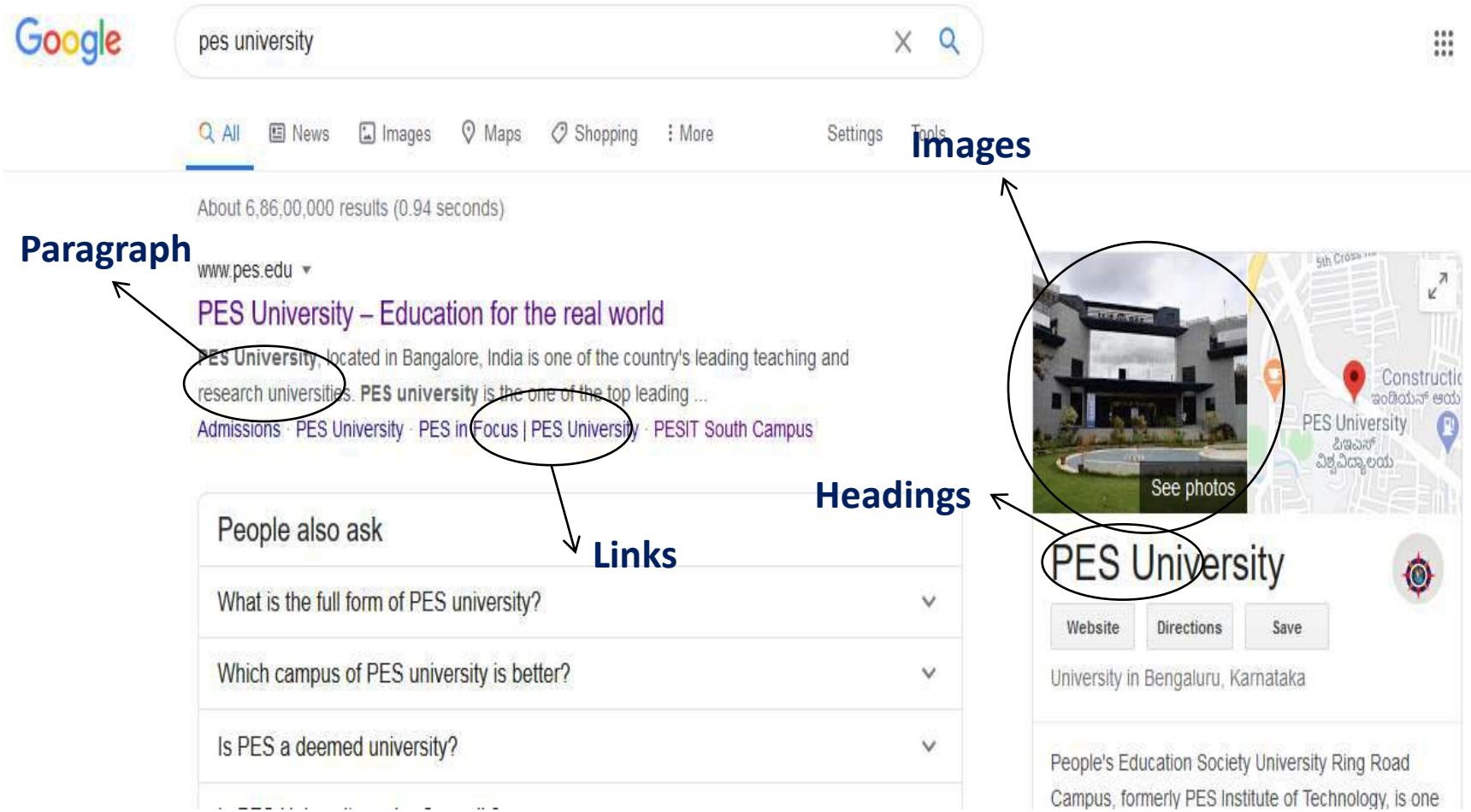
HTML – Basic Markups

Document Structure



HTML – Basic Markups

Common Tags / Elements



HTML – Basic Markups

Common Tags / Elements

- Text Content
  - Paragraphs
  - Headings
  - Lists
  - Tables
- Images
- Links

HTML – Basic Markups

Paragraph

- Text is normally placed in paragraph elements
- Paragraph Elements:
  - The <p> tag breaks the current line and inserts a blank line - the new line gets the beginning of the content of the paragraph
  - The browser puts as many words of the paragraph's content as will fit in each line

Example:

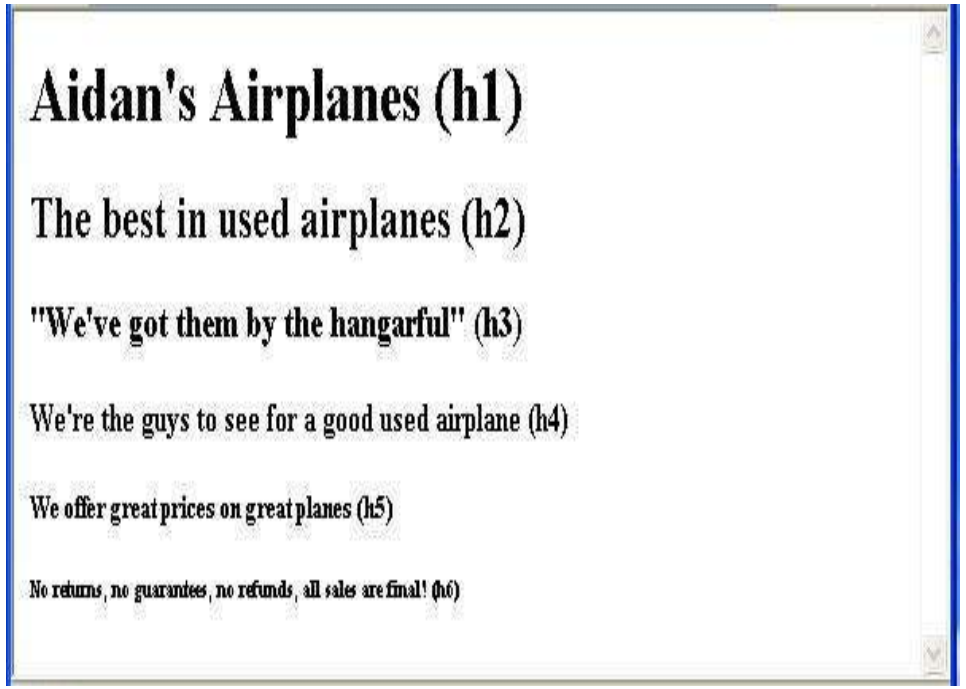
```
<p>
   A paragraph always starts on a new line, and browsers automatically add some
   white space (a margin) before and after a paragraph.
</p>
```

HTML – Basic Markups

Headings

Example:

```
<body>
    <h1> Aidan's Airplanes </h1>
    <h2> The best in used airplanes </h2>
    <h3> "We've got them by the hangarful" </h3>
    <h4> We're the guys to see for a good used airplane /h4> <!– Regular text size -->
    <h5> We offer great prices on great planes </h5>
    <h6> No returns, no guarantees, no refunds </h6>
</body>
```



Formatting tags

- Superscripts and subscripts
  - Subscripts with <sub>
  - Superscripts with <sup>

Example: x₂³

Display: x_2^3

- Inline versus block elements
- Block elements CANNOT be nested in inline elements

HTML – Basic Markups

Special Characters

- Character Entities:

Char.	*Entity*	*Meaning*
&	&amp;	Ampersand
<	<	Less than
>	>	Greater than
”	"	Double quote
’	'	Single quote
(space)		Non-breaking space

- Horizontal rules:

  <hr /> draws a line across the display, after a line break

HTML – Basic Markups

Images

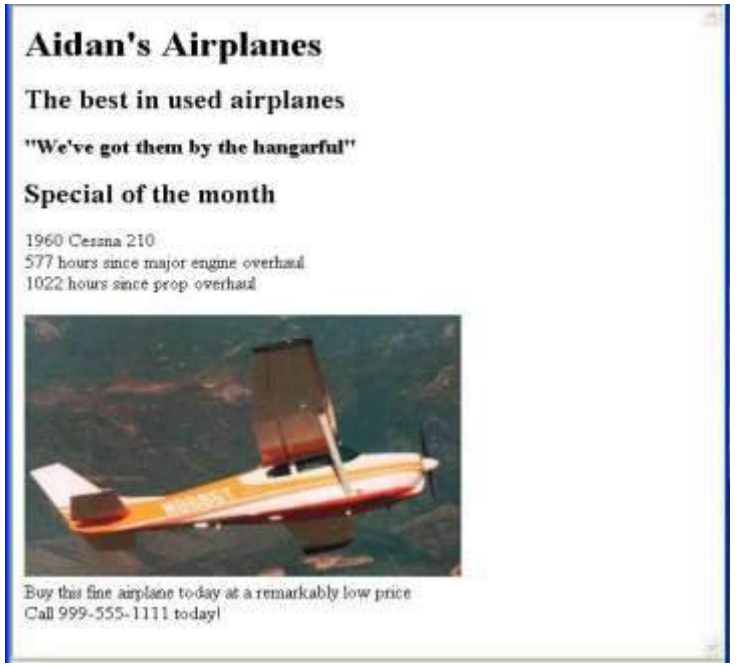
- GIF (Graphic Interchange Format) =8-bit color (256 different colors)
- JPEG (Joint Photographic Experts Group) =24-bit color (16 million different colors)
- Portable Network Graphics (PNG)
- Images are inserted into a document with the <img /> tag with the src attribute
  - The alt attribute is required by XHTML

- The <img> tag has 30 different attributes, including width and height (in pixels)

HTML – Basic Markups

Images…(cntd.)

```
<html>
 <head> <title> Images </title>
 </head>
 <body>
<p>
<img src = "c210new.jpg"  alt = "Picture of a Cessna 210"/>
     <br />
     Buy this fine airplane today at a
     remarkably low price <br />
     Call 999-555-1111 today!
 </p>
 </body>
</html>
```



HTML – Basic Markups

Hypertext Links

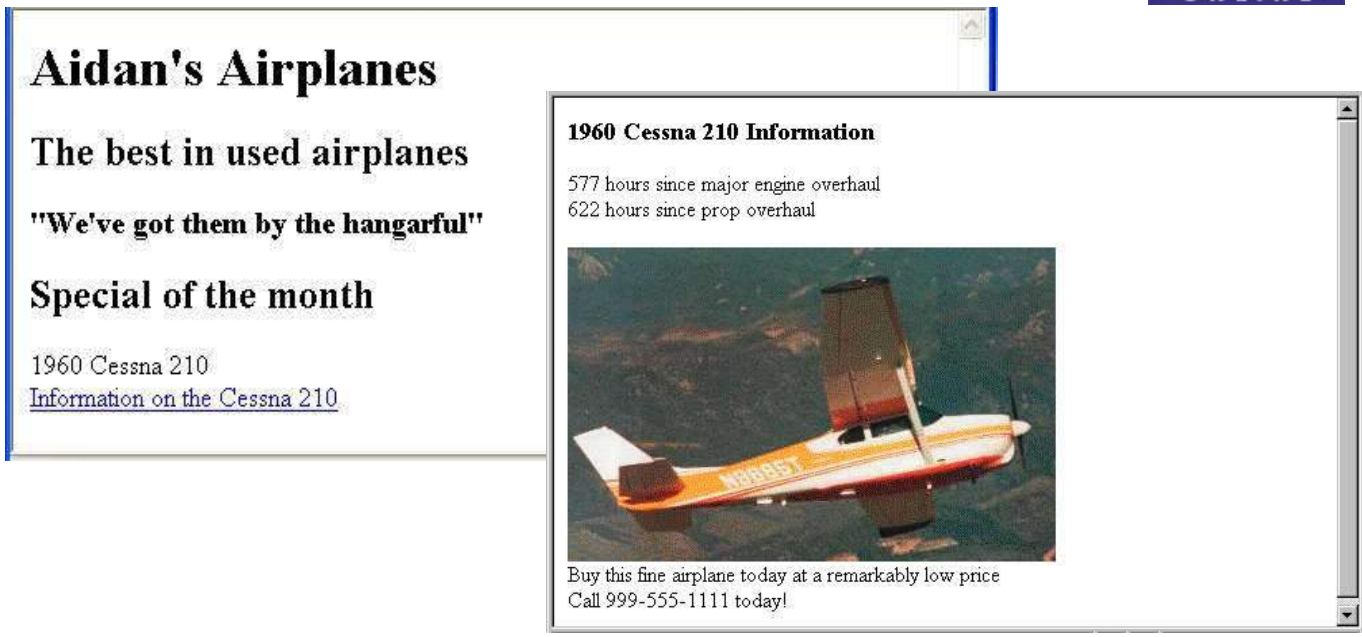
- Hypertext is the essence of the Web!
- A link is specified with the href attribute of <a> (the anchor tag)
  - The content of <a> is the visual link in the document
  - If the target is a whole document (not the one in which the link appears), the target need not be specified in the target document as being the target
- Note: Relative addressing of targets is easier to maintain and more portable than absolute addressing

HTML – Basic Markups

Hypertext links…(cntd.)

Example:

```
<html>
 <head> <title> Links </title>  </head>
 <body>
  …
  <h2> Special of the month </h2>
  <p> 1960 Cessna 210 <br />
  <a href = "C210data.html"> Information on the Cessna 210
  </a>
  </p>
 </body>
</html>
```



HTML – Basic Markups

Lists

Used to present list of information in well formed and semantic way.

- Three types of Lists:
  - ❑Unordered list
  - ❑Ordered list
  - ❑Definition List

HTML – Basic Markups

Unordered Lists

- The list is the content of the <ul> tag
- List elements are the content of the <li> tag

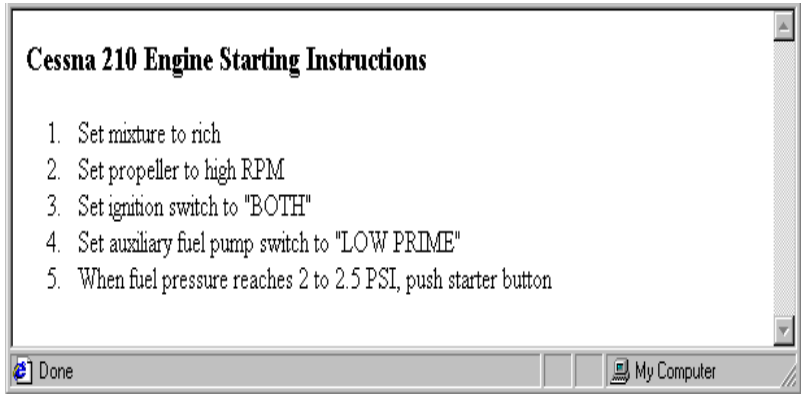
```
<h3> Some Common Single-Engine Aircraft </h3>
 <ul>
   <li> Cessna Skyhawk </li>
   <li> Beechcraft Bonanza </li>
   <li> Piper Cherokee </li>
 </ul>
```

HTML – Basic Markups

Ordered Lists

- The list is the content of the <ol> tag
- Each item in the display is preceded by a sequence value

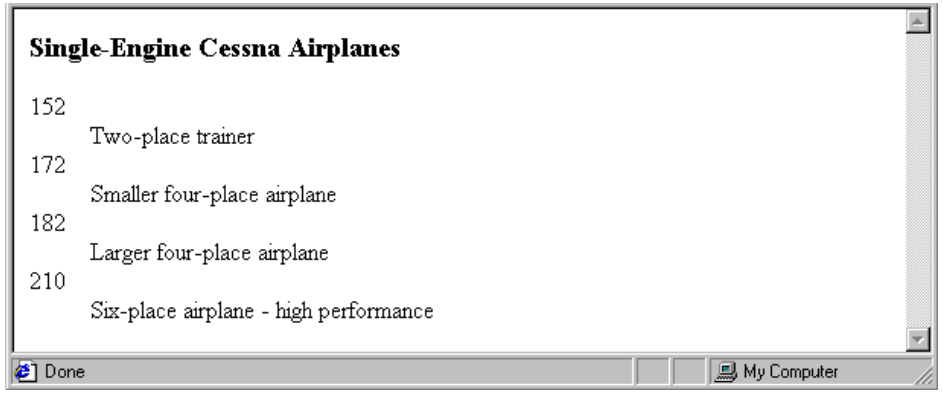
```
<h3> Cessna 210 Engine Starting Instructions</h3>
<ol>
  <li> Set mixture to rich </li>
  <li> Set propeller to high RPM </li>
  <li> Set ignition switch to "BOTH" </li>
  <li> Set auxiliary fuel pump switch to
       "LOW PRIME" </li>
  <li> When fuel pressure reaches 2 to 2.5
       PSI, push starter button </li>
</ol>
```



HTML – Basic Markups

Definition Lists

- List is the content of the <dl> tag
- Terms being defined are the content of the <dt> tag
- The definitions themselves are the content of the <dd> tag




```
<h3> Single-Engine Cessna Airplanes </h3>
<dl >
  <dt> 152 </dt>
  <dd> Two-place trainer </dd>
  <dt> 172 </dt>
  <dd> Smaller four-place airplane </dd>
</dl>
```

HTML – Basic Markups

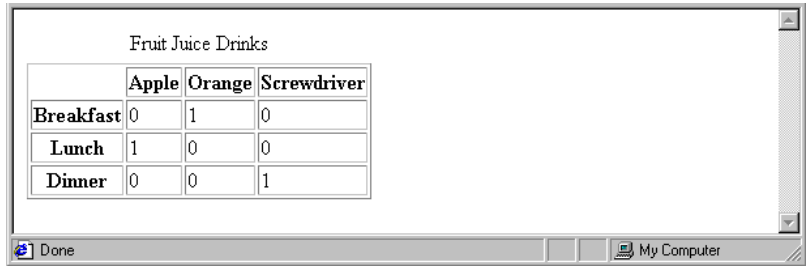
Tables

- A matrix of cells, each possibly having content
- The cells can include almost any element
- Some cells have row or column labels and some have data
- A table is specified as the content of a <table> tag
- A border attribute in the <table> tag specifies a border between the cells
- Tables are given titles with the <caption> tag, which can immediately follow <table>

HTML – Basic Markups

Tables…(cntd.)

- Each row of a table is specified as the content of a <tr> tag
- The row headings are specified as the content of a <th> tag
- The contents of a data cell is specified as the content of a <td> tag.




```
<table border = "border">
  <caption> Fruit Juice Drinks </caption>
   <tr>
    <th> </th>
    <th> Apple </th>
    <th> Orange </th>
    <th> Screwdriver </th>
   </tr>
   <tr>
    <th> Breakfast </th>
    <td> 0 </td>
    <td> 1 </td>
    <td> 0 </td>
   </tr>
   </table>
```

HTML – Basic Markups

Tables : Column Span

```
<tr>
  <th colspan = "3"> Fruit Juice Drinks
   </th>
</tr>
<tr>
  <th> Orange </th>
  <th> Apple </th>
  <th> Screwdriver </th>
</tr>
```

HTML – Basic Markups

Tables : Row Span

- If the rows have labels and there is a spanning column label, the upper left corner must be made larger, using rowspan

```
<table border = "border">
  <tr>
    <td rowspan = "2"> </td>
    <th colspan = "3"> Fruit Juice Drinks
    </th>
  </tr>
  <tr>
    <th> Apple </th>
    <th> Orange </th>
</tr>
        …
</table>
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

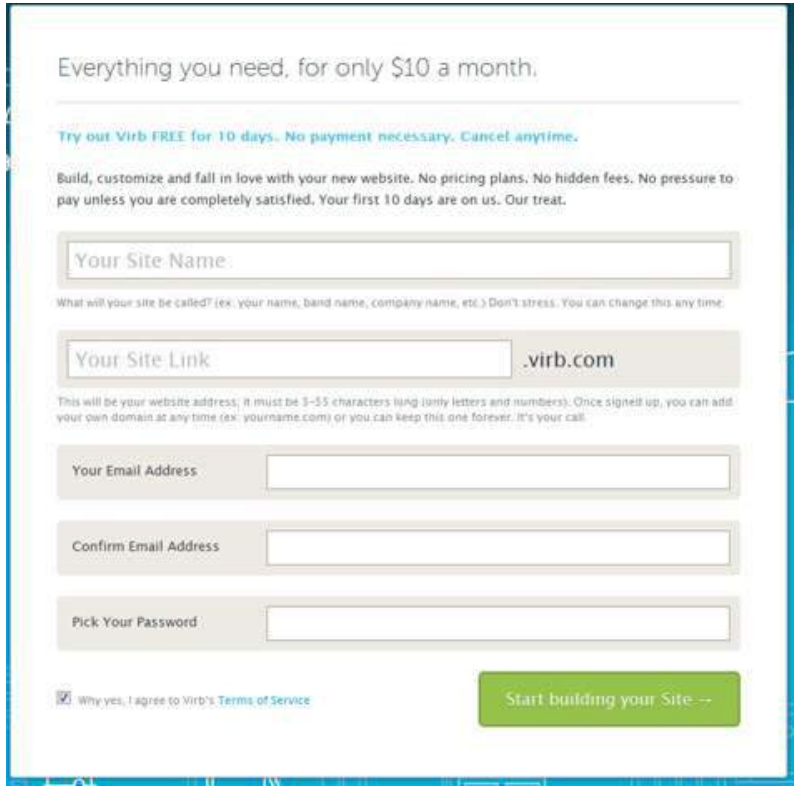
WEB TECHNOLOGIES

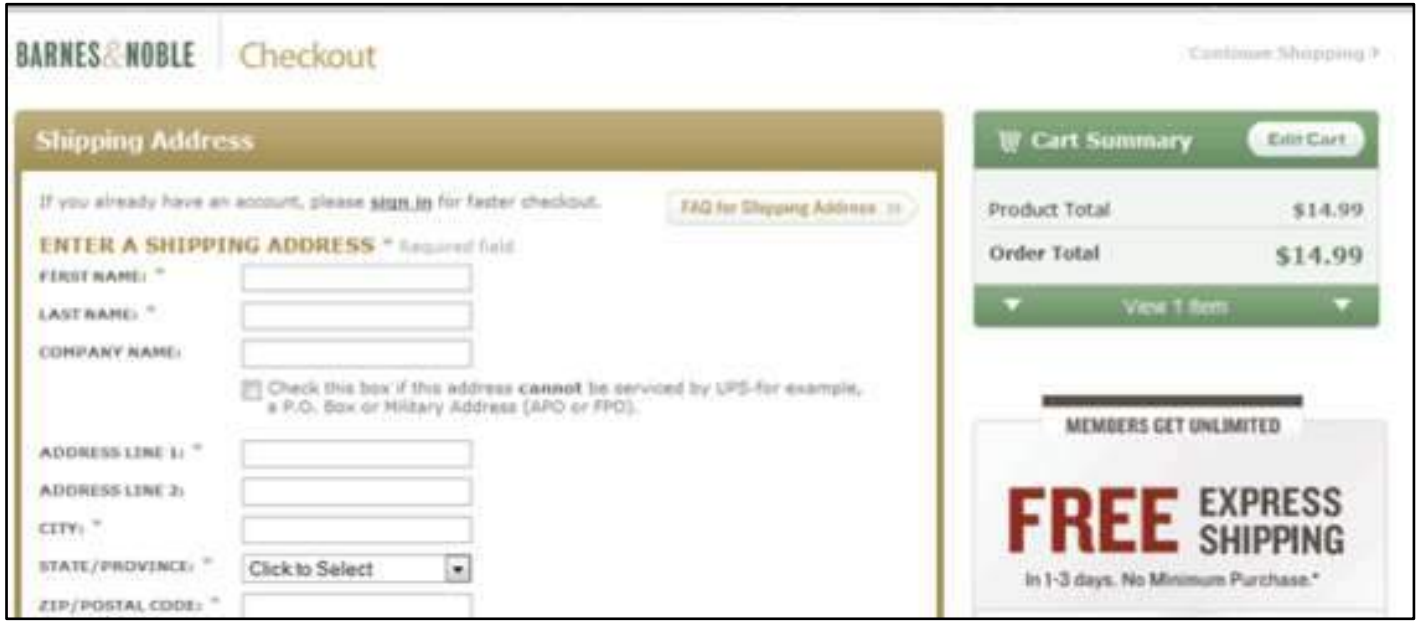
HTML – Forms

Vinay Joshi
Department of
Computer Science and Engineering

HTML – Forms

Introduction





HTML – Forms

Introduction

- A form is a way to send information from a browser to a server
- All the components of a form appear as the content of <form> tag
  - The components are called *widgets* (e.g., text boxes, radio buttons and checkboxes)

HTML – Forms

Attributes

Syntax:

```
<form list_of_attributes_and_values>
    <input element>
    …
</form>
```

Important attributes of the <form> tag

- Method
- Action
- Target

Example:

```
<form method="post" action="survey.php" target="_blank">
    <input type="text">
    …
</form>
```

HTML – Forms

Input widgets

- Input widget can be any of the following types
  - Text
  - Textarea
  - Button
  - Checkboxes
  - Radio Buttons
  - Dropdown list
  - Hidden

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

HTML 5

Vinay Joshi
Department of
Computer Science and Engineering

HTML5

Features

HTML 5

Forms – New Input Widgets

- HTML5 specifications introduced new Input types and properties

New Input Types

- email: email address
- number: spinbox
- range: slider
- url: web addresses
- color: color pickers
- search: search boxes
- date: date
- time: time
- file: input file selection

New Input properties

- placeholder
- required
- pattern
- autofocus

HTML 5

Semantic Elements

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

CSS – Cascading Style Sheet and Selectors

Vinay Joshi
Department of
Computer Science and Engineering

CSS – Cascading Style Sheets and Selectors

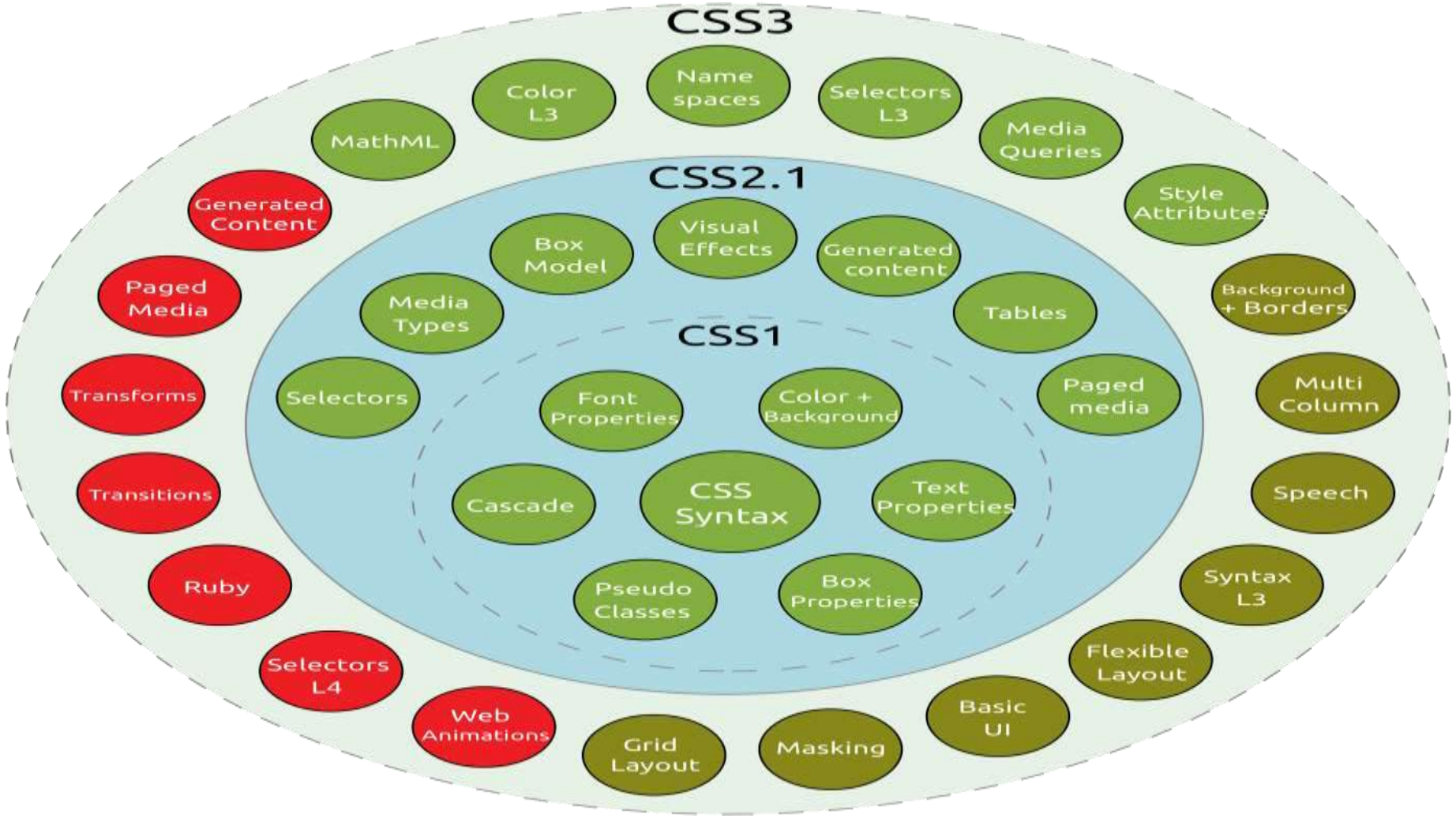
Introduction

HTML without CSS

HTML with CSS

CSS – Cascading Style Sheets and Selectors

Introduction



CSS – Cascading Style Sheets and Selectors

Three ways to include CSS

1) Inline Style - CSS is placed directly into the XHTML element.

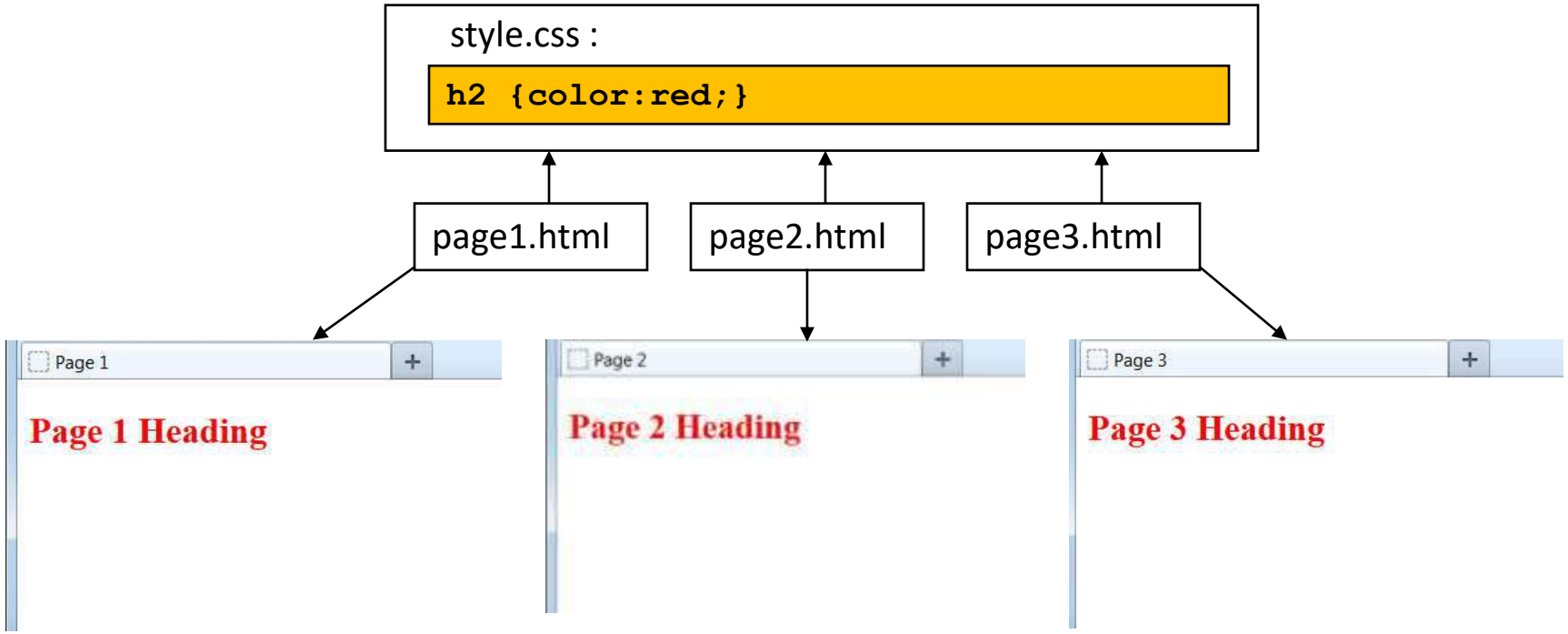
2) Internal Style Sheet - CSS is placed into a separate area within the <head> section of a web page.

3) External Style Sheet - CSS is placed into a separate file and "connected" to a web page.

CSS – Cascading Style Sheets and Selectors

Benefits of External Style Sheets

The real power of using an external style sheet is that multiple web pages on our site can link to the same style sheet:



Styles declared in an external style sheet will affect all matching elements on all web pages that link to the style sheet. By editing the external style sheet, we can make site-wide changes (even to hundreds of pages) instantly.

Internal vs. External Style Sheets

Internal Style Sheets

- are appropriate for very small sites, especially those that have just a single page.
- might also make sense when each page of a site needs to have a completely different look.

External Style Sheets:

- are better for multi-page websites that need to have a uniform look and feel to all pages.
- make for faster-loading sites (less redundant code).
- allow designers to make site-wide changes quickly and easily.

External style sheets create the furthest separation between content and presentation. Hence, the best option when creating a new site.

CSS – Cascading Style Sheets and Selectors

Conflicts

- Same formatting rules can be defined in all three locations at the same time.
- For example, a paragraph element could contain an inline style (color:red) but the internal style sheet (color:blue) and the external style sheet (color:green) give conflicting instructions to the web browser.
- Web browsers need a consistent way of "settling" this disagreement.

Cascading in CSS

- We use the term cascading because there is an established order of priority to resolve these formatting conflicts:
  1) Inline style (highest priority)
  2) Internal style sheet (second priority)
  3) External style sheet (third priority)
  4) Web browser default (only if not defined elsewhere)

For each XHTML element, the browser will combine all the styles defined at different levels. For all conflicts, it will use the above priority system to determine which format to display on the page.

In the prior example, the paragraph would display as red, because the inline style "outranks" all the others.

Selectors

- Primary Selectors
- Nested Selectors
- Multiple Selectors
- Pseudo Selectors

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

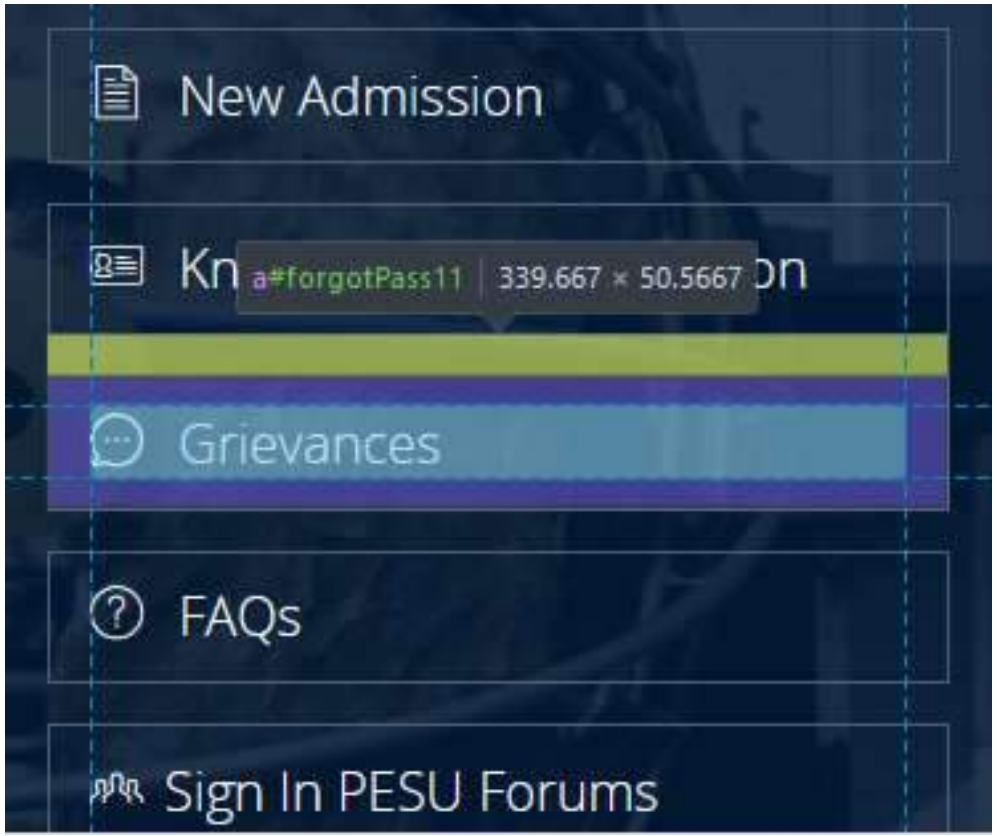
WEB TECHNOLOGIES

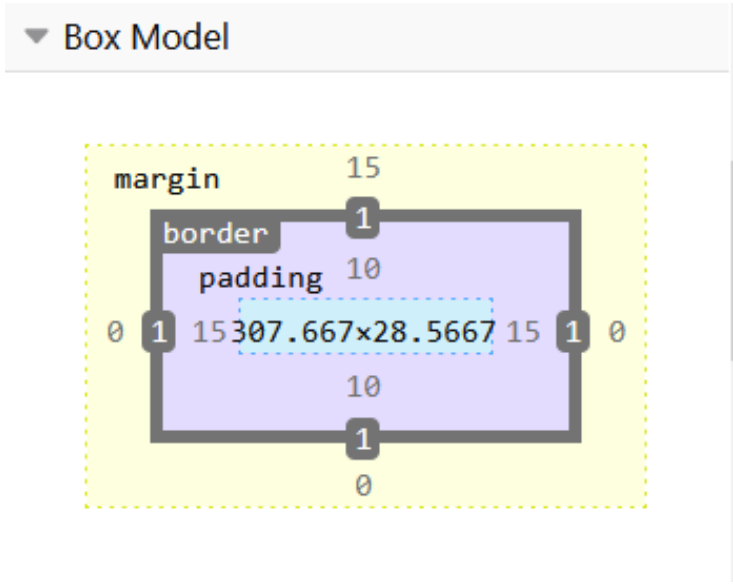
CSS – Box Model and Position Property

Vinay Joshi
Department of
Computer Science and Engineering

CSS – Box Model and Position Property

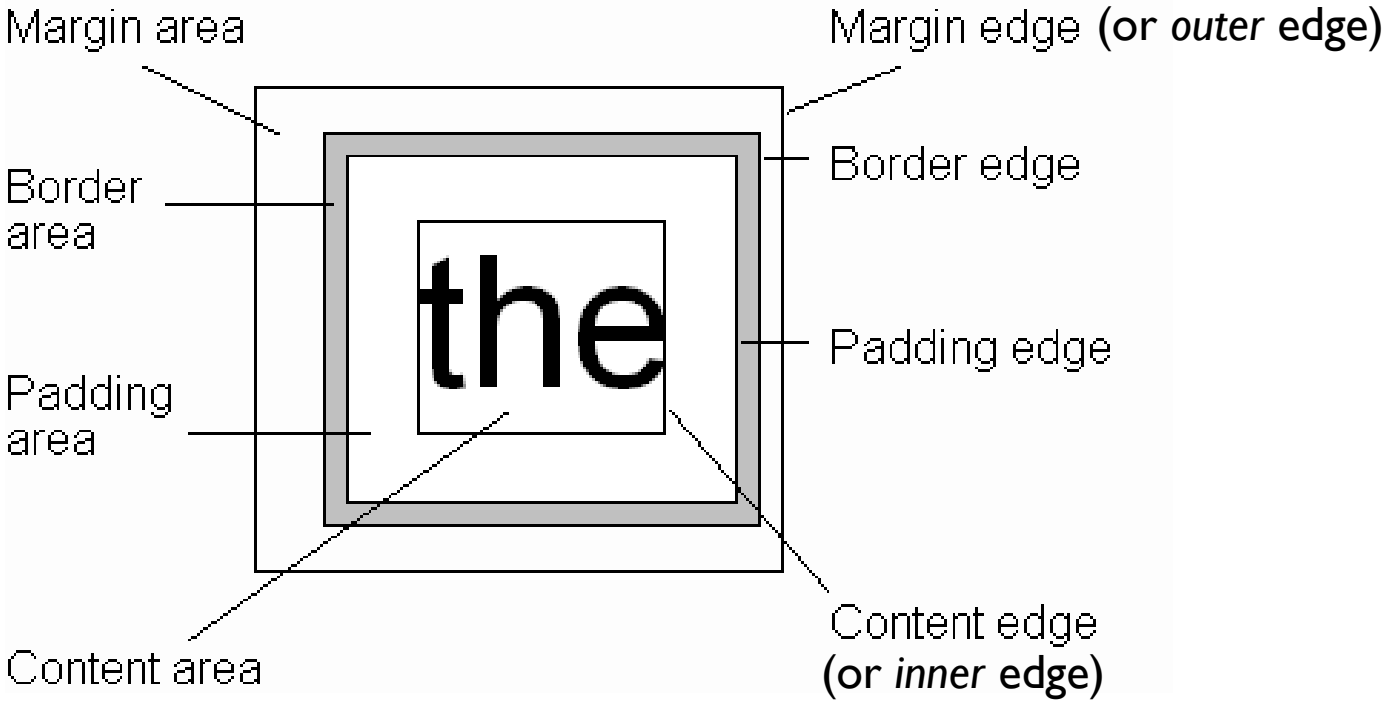
Box Model





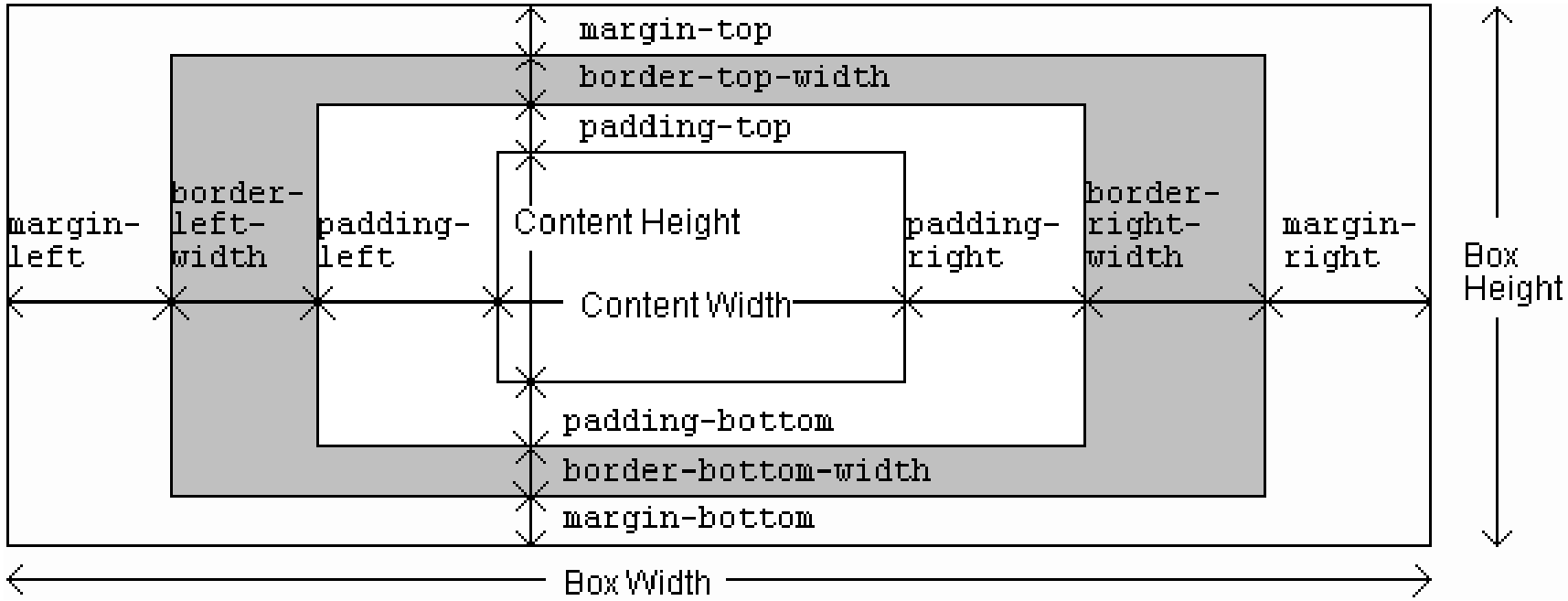
Box Model

- Every rendered element occupies a box:



CSS – Box Model and Position Property

Box Model - Components



CSS – Box Model and Position Property

Position Property

- By default, the browser determines the positioning of each element
- CSS introduced the **position** property and a capability to control how and where page elements are displayed
- Position property values:
  - Absolute
  - Relative
  - Fixed
  - Sticky

Element support for styling

- There are two elements used commonly to style specific parts of a webpage
  -
    - To apply style to a part of a paragraph
  -
    - To apply style to a set of elements or paragraphs

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

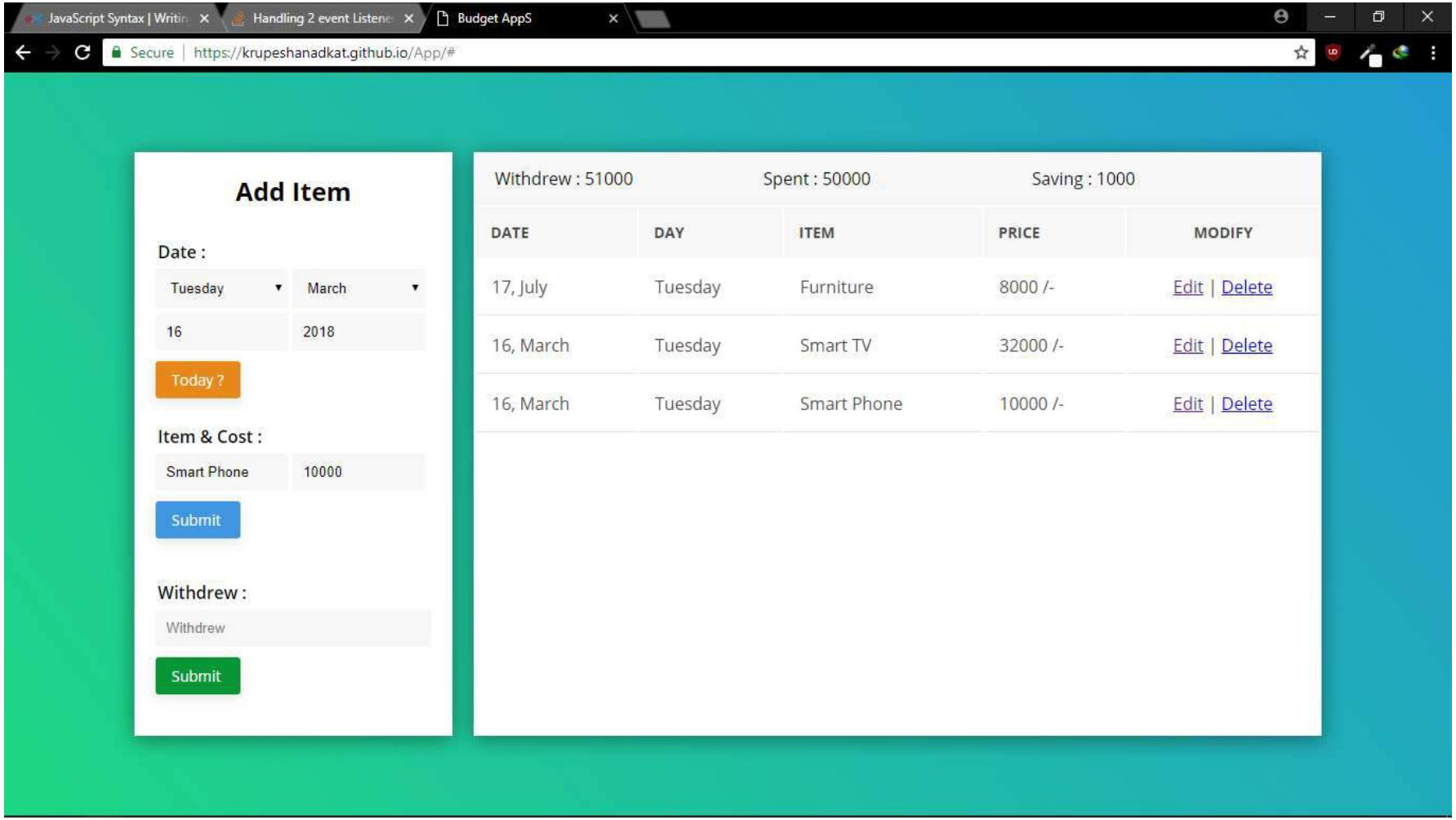
WEB TECHNOLOGIES

JavaScript - Basics

Vinay Joshi
Department of
Computer Science and Engineering

JavaScript - Basics

Introduction to JavaScript



Introduction to JavaScript

- Client Side Scripting Language
- Originally, LiveScript in NetScape Browser
- JavaScript programs are run by an interpreter that comes bundled with the user's web browser
- Now the language has evolved with additional Server Side Scripting capabilities (like in Node.JS)

JavaScript Code

JavaScript can be embedded into documents by using the SCRIPT tag

```
<script type="text/javascript">
        document.write("Hello World!");
</script>
```

It can also included from a external script file as follows:

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

Comments in JavaScript

-Single line comment : **//**

-Multiline comments : **/* … */**

JavaScript Code - Debugging

PES UNIVERSITY ONLINE

Debugging JavaScript Errors

- When you're learning or using JavaScript, it's important to be able to track typing or other coding errors.

Browser	How to access JavaScript error messages
Apple Safari	Select Safari → Preferences → Advanced → "Show Developer menu in menu bar." You may prefer to use the Firebug Lite JavaScript module, which many people find easier to use.
Google Chrome	Press Ctrl-Shift-J on a PC, or Command-Shift-J on a Mac.
Mozilla Firefox	Press Ctrl-Shift-J on a PC, or Command-Shift-J on a Mac.
Microsoft Internet Explorer & Edge	Press F12 to call up the DevTools Console.
Opera	Select Tools → Advanced → Error Console.

JavaScript Code – Using Semicolon

- JavaScript generally automatically inserts semicolons at the end of line

  x += 10 => x += 10;

- However, when you wish to place more than one statement on a line, you must separate them with semicolons, like this:

  x += 10; y -= 5; z = 0

- When a statement spans across multiple lines, JavaScript will not raise error if the next line has a valid symbol/literal/token

  return a

  + b

JavaScript Code – Variable Declaration

- variable name starts with a-z, A-Z, $, or _
- then a-z, A-Z, 0-9, $, _
- Variable names are case-sensitive.

Count and count are two different variables

- Variable can be declared using
  - let (block scope)
  - var (function or global scope)
  - const (block scope)
  - use without declaring (global scope)

Keyword	Scope	Can be reassigned	Can be redeclared
var	Function	Yes	Yes
let	block	Yes	No
const	block	No	No

JavaScript Code – Datatypes

- JS is loosely typed or dynamic typed
- Primitive Datatypes
  - number
  - string
  - boolean
  - null
  - undefined
- Non-Primitive Datatypes (used with new keyword)
  - Object
  - Date
  - Array
  - Number
  - String
  - Boolean

JavaScript Code – Operators and Constructs

JavaScript has most of the operators we're used to from C/Java

- Arithmetic (+, - , *, /, %)
- Assignment (=, +=, -=, *=/=, %=, ++, --)
- Logical (&&, ||, !)
- Comparison (<, >, <=, >=, ==, ===,!=,!==)

Note: + also does concatenation if one of the operands is string

- Constructs: if, else, while, for, switch, case

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

JavaScript – Arrays and Functions

Vinay Joshi
Department of
Computer Science and Engineering

JavaScript – Arrays

Array

- Arrays are lists of elements indexed by a numerical value starting with 0 to (length of the array – 1)
- Arrays can be created using
  - The new Array method
    - let arr = new Array(100) – creates an array of 100 elements
    - let arr = new Array(10, 20) – creates an array of 2 elements
  - Literal arrays using square brackets
    - var alist = [1, “ii”, “gamma”, “4”];

Array Length

- Array length property can be modified at runtime
- Hence, the length property does not necessarily indicate the number of defined values in the array

```
const arr = [1, 2];
console.log(arr);
// [ 1, 2 ]

arr.length = 7;
// set array length to 7
console.log(arr);
// [ 1, 2, <5 empty items> ]
```

```
console.log(arr[4]);
//undefined

for (i in arr)
    console.log(typeof i + i));
// string 0
// string 1
```

JavaScript - Functions

What are Functions?

- Whenever you have a relatively complex piece of code that is likely to be reused, you have a candidate for a function.
- The general syntax for a function is:

```
function function_name([parameter [, ...]])
{
        statements
        //optional return statement
}
```

- The general syntax for calling a function is:

```
[retval =] function_name([argument [,…]])
```

JavaScript - Functions

Parameters and Arguments

- Argument list and parameter list mismatch does not give errors.
- Parameter that is not passed a value in arguments list is treated as **undefined**
- To access additional arguments, use the **arguments** object or the **args** array to access the values passed.

```
function func() {
   // access passed values
   // using arguments object
}
```

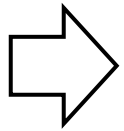
```
function func(…args) {
        // access passed values
        // using args array
}
```

JavaScript - Functions

Hoisting – Variables and Functions

- Hoisting is JavaScript's default behavior of moving all variable and function declarations to the top of the current scope (to the top of the current <script> or the current function).
- Only declarations are hoisted not initializations
- Variables and constants declared with **let** or **const** are not hoisted!

```
num = 6;
console.log(num);
var num = 8;
console.log(num);
```



```
var num;
num = 6;
console.log(num);
num = 8;
console.log(num);
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

JavaScript – Arrays and Functions

Vinay Joshi
Department of
Computer Science and Engineering

JavaScript – Arrays

Array

- Arrays are lists of elements indexed by a numerical value starting with 0 to (length of the array – 1)
- Arrays can be created using
  - The new Array method
    - let arr = new Array(100) – creates an array of 100 elements
    - let arr = new Array(10, 20) – creates an array of 2 elements
  - Literal arrays using square brackets
    - var alist = [1, “ii”, “gamma”, “4”];

Array Length

- Array length property can be modified at runtime
- Hence, the length property does not necessarily indicate the number of defined values in the array

```
const arr = [1, 2];
console.log(arr);
// [ 1, 2 ]

arr.length = 7;
// set array length to 7
console.log(arr);
// [ 1, 2, <5 empty items> ]
```

```
console.log(arr[4]);
//undefined

for (i in arr)
    console.log(typeof i + i));
// string 0
// string 1
```

JavaScript - Functions

What are Functions?

- Whenever you have a relatively complex piece of code that is likely to be reused, you have a candidate for a function.
- The general syntax for a function is:

```
function function_name([parameter [, ...]])
{
        statements
        //optional return statement
}
```

- The general syntax for calling a function is:

```
[retval =] function_name([argument [,…]])
```

JavaScript - Functions

Parameters and Arguments

- Argument list and parameter list mismatch does not give errors.
- Parameter that is not passed a value in arguments list is treated as **undefined**
- To access additional arguments, use the **arguments** object or the **args** array to access the values passed.

```
function func() {
   // access passed values
   // using arguments object
}
```

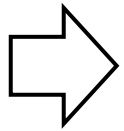
```
function func(…args) {
        // access passed values
        // using args array
}
```

JavaScript - Functions

Hoisting – Variables and Functions

- Hoisting is JavaScript's default behavior of moving all variable and function declarations to the top of the current scope (to the top of the current <script> or the current function).
- Only declarations are hoisted not initializations
- Variables and constants declared with **let** or **const** are not hoisted!

```
num = 6;
console.log(num);
var num = 8;
console.log(num);
```



```
var num;
num = 6;
console.log(num);
num = 8;
console.log(num);
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

JavaScript – Built-in Objects

Vinay Joshi
Department of
Computer Science and Engineering

JavaScript – Built-in Objects

Global Objects supported

- Number
- String
- Array
- Date
- Math
- window
- document

JavaScript – Built-in Objects

Number Object

- Properties
  - MAX_VALUE
  - MIN_VALUE
  - NaN
  - POSITIVE_INFINITY
  - NEGATIVE_INFINITY
- Operations resulting in errors return NaN
  - Use isNaN(a) to test if a is NaN
- toString method converts a number to string

JavaScript – Built-in Objects

String Object

Method	Description
charAt(*index*)	Returns a string containing the character at the specified *index*. If there is no character at the *index*, charAt returns an empty string. The first character is located at *index* 0.
charCodeAt(*index*)	Returns the Unicode value of the character at the specified *index*. If there is no character at the *index*, charCodeAt returns NaN (Not a Number).
concat(*string*)	Concatenates its argument to the end of the string that invokes the method. The string invoking this method is not modified; instead a new String is returned. This method is the same as adding two strings with the string concatenation operator + (e.g., s1.concat(s2) is the same as s1 + s2).
fromCharCode(*value1, value2,)*	Converts a list of Unicode values into a string containing the corresponding characters.
indexOf(*substring, index)*	Searches for the first occurrence of *substring* starting from position *index* in the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from index 0 in the source string.
lastIndexOf(*substring, index)*	Searches for the last occurrence of *substring* starting from position *index* and searching toward the beginning of the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from the end of the source string.

JavaScript – Built-in Objects

String Object

Method	Description
slice(*start, end)*	Returns a string containing the portion of the string from index *start* through index *end*. If the *end* index is not specified, the method returns a string from the *start* index to the end of the source string. A negative *end* index specifies an offset from the end of the string starting from a position one past the end of the last character (so –1 indicates the last character position in the string).
split(*string)*	Splits the source string into an array of strings (tokens) where its *string* argument specifies the delimiter (i.e., the characters that indicate the end of each token in the source string).
substr (*start, length)*	Returns a string containing *length* characters starting from index *start* in the source string. If *length* is not specified, a string containing characters from *start* to the end of the source string is returned.
substring(*start, end)*	Returns a string containing the characters from index *start* up to but not including index *end* in the source string.
toLowerCase()	Returns a string in which all uppercase letters are converted to lowercase letters. Non-letter characters are not changed.
toUpperCase()	Returns a string in which all lowercase letters are converted to uppercase letters. Non-letter characters are not changed.
toString()	Returns the same string as the source string.
valueOf()	Returns the same string as the source string.

JavaScript – Built-in Objects

Array Methods

Method	Description
push	Add to the end of array
pop	Remove from the end of array
shift	Remove from the front of array
unshift	Add to the front of array
join	return a string with array elements
indexOf	return the index of array
sort	sort an array in ascending order by default
concat	concatenate two arrays
slice	returns a subset if of the array

JavaScript – Built-in Objects

Array – Sort

- *arr*.sort(*[compareFunction]*)
- If compareFunction is not specified,array is sorted as strings in ascending order
- Arr = [1, 2, 11, 12, 22]
- Console.log(arr.sort())
- // 1, 11, 12, 2, 22
- compareFunction takes two parameters say a and b and returns
  - 1 if a > b
  - 0 if a = b
  - -1 if a < b
- To reverse the order modify the condition for returning 1, 0 and -1

```
var numbers = [4, 2, 5, 1, 3];
numbers.sort(function(a, b) { return b - a; });
console.log(numbers);
// [5,4,3,2,1]
```

JavaScript – Built-in Objects

Date Methods

Method	Description
toLocaleString	A string of the Date information
(get/set)Date	The day of the month
(get/set)Month	The month in the range of 0 to 11
(get/set)Day	The day of the week in the range of 0 to 6
(get/set)FullYear	The year
(get/set)Time	The number of milliseconds since January 1, 1970
(get/set)Hours	The number of the hour in the range of 0 to 23
(get/set)Minutes	The number of the minute in the range of 0 to 59
(get/set)Seconds	The number of the second in the range of 0 to 59
(get/set)Milliseconds	The number of the millisecond in the range of 0 to 999

JavaScript – Built-in Objects

Math Methods

Provides **static** mathematical constants and functions

Constant /Method	Description
Math.E	Euler's Constant (Approx. 2.718)
Math.PI	Value of PI (Approx. 3.1416)
Math.SQRT2	Square root of 2 (Approx. 1.414)
Math.abs(x)	Returns the absolute value of x
Math.ceil(x)	Returns the smallest integer greater than or equal to x
Math.floor(x)	Returns the largest integer less than or equal to x.
Math.max([x[, y[, …]]])	Returns the largest of zero or more numbers.
Math.min([x[, y[, …]]])	Returns the smallest of zero or more numbers.
Math.pow(x, y)	Returns base x to the exponent power y (that is, x^y).
Math.random()	Returns a random number between 0 and 1

JavaScript – Built-in Objects

Window – Properties and Methods

- Global object containing global variables and functions declared in the page. For example, *var x;* can also be accessed as *window.x*

Properties /Method	Description
location	object containing location details like href, path etc.
history	object containing the browser history
localStorage	object containing a local cache for storing user info
innerHeight, innerWidth	dimensions of the display area of the browser
alert(text)	method to display a dialog box with message
prompt(text,default)	method to seek input from user, returns string
confirm(text)	method to show a confirmation dialog
setInterval, clearInterval	start/stop performing action repeatedly after an interval
setTimeout, clearTimeout	start/stop performing action once after a timeout period

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

JavaScript Object Inheritance

Vinay Joshi
Department of
Computer Science and Engineering

JavaScript Object Inheritance

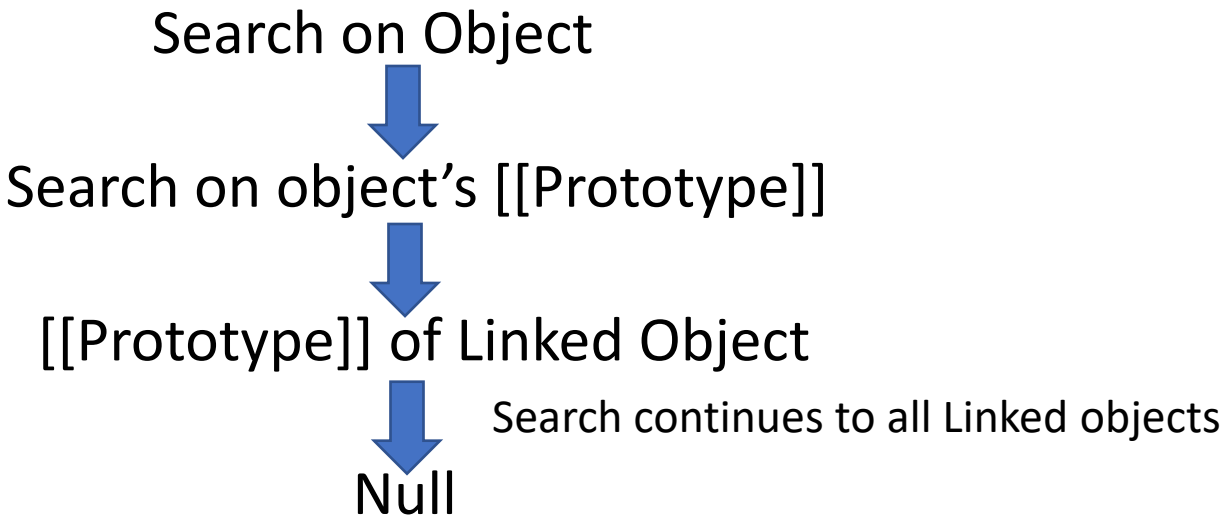
Prototype Property - Accessing Properties and Methods

- JavaScript is a prototype based language-object properties and methods can be shared through generalized objects that have the ability to be cloned and extended.
- All objects in JavaScript descend from the parent Object constructor.
- Every object in JavaScript has an internal property called [[Prototype]]
- Built in objects also has Prototype property
- Adding new properties and methods to the prototype property is better compared to adding directly to the constructor.

JavaScript Object Inheritance

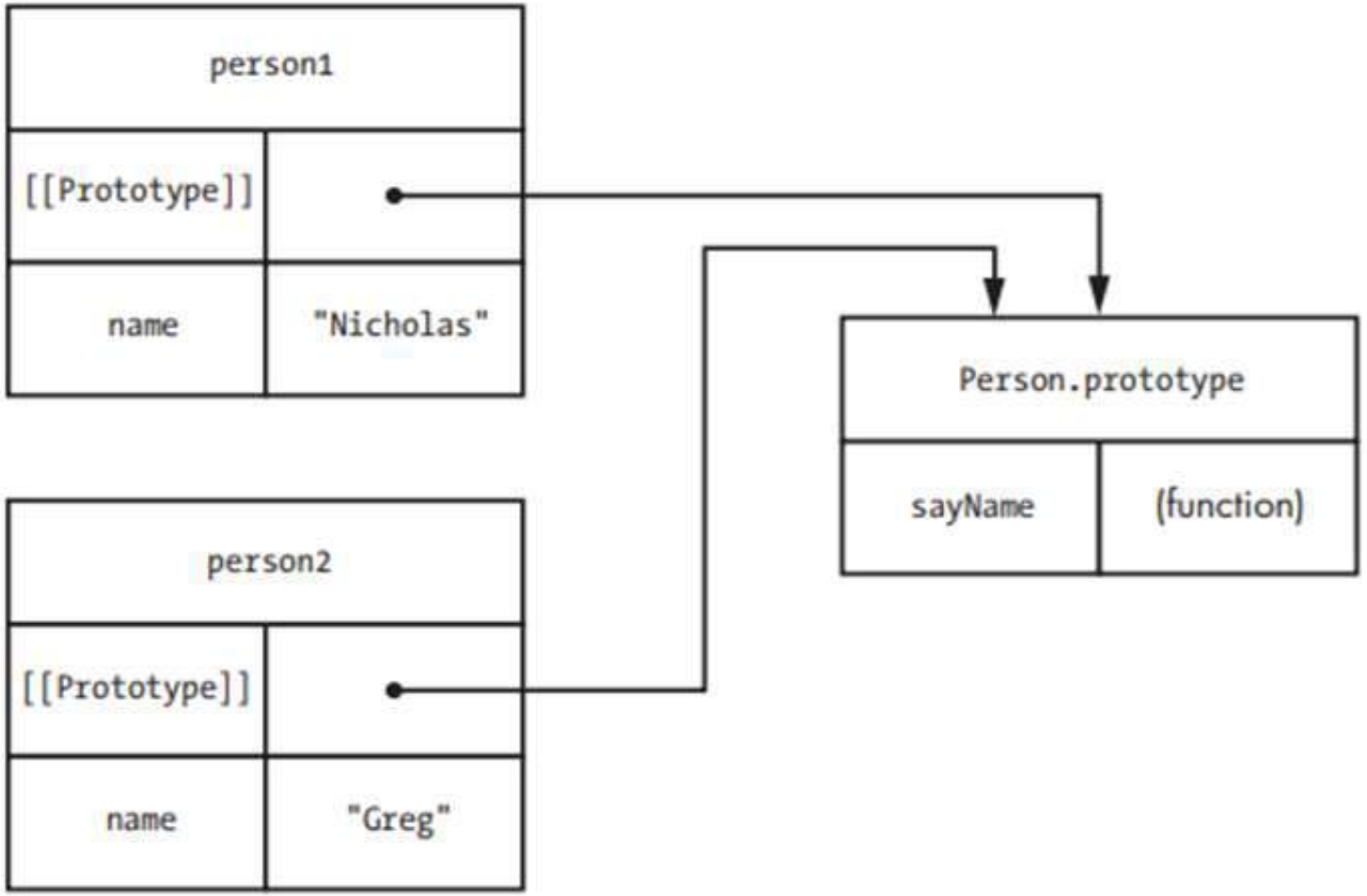
Prototype Property

- All objects in JavaScript inherit from at least one other object.
- The object being inherited from is known as the prototype, and the inherited properties can be found in the prototype object of the constructor.
- Access properties and methods outside the objects



JavaScript Object Inheritance

Prototype Properties



JavaScript Object Inheritance

Prototype Chain

```
// Let's create an object car from function
vehicle with its own properties name and year:
let Vehicle = function () {
    this.name = "Hyndai";
    this.year = 2019;
}
let car = new Vehicle();
// add properties in vehicle function's prototype
vehicle.prototype.year = 2020;
vehicle.prototype.color = "blue";
```

car.[[Prototype]] = {year,color}
car.[[Prototype]].[[Prototype]] => Object.prototype
car.[[Prototype]].[[Prototype]].[[Prototype]] =>null

JavaScript Object Inheritance

Prototype Chain - Questions

- console.log(car.name)?
- console.log(car.year)? Prototype Shadowing happens here
- console.log(car.color)?
- console.log(car.model)?

JavaScript Object Inheritance

Object.create() method

- Object.create() method is used to create a new object with the

specified prototype object and properties.

- Returns a new object with the specified prototype object.
- Used for implementing inheritance.

```
Object.create(prototype_object, propertiesObject)
```

prototypeObject: Newly created object's prototype object. It has to be an object or null.

propertiesObject: Properties of the new object. (Data Descriptors and Access Descriptors)

JavaScript Object Inheritance

Constructor method

- Define the base object constructor
- Define the derived object constructor which in turn calls the base object constructor
- Assign the prototype of derived object as the base object
- Assign the prototype.constructor to derived object constructor

This will enable the derived object to access the properties and methods of the base object.

THANK YOU

Vinay Joshi
Department of
Computer Science and Engineering
vinayj@pes.edu

WEB TECHNOLOGIES

JavaScript Objects

Vinay Joshi
Department of
Computer Science and Engineering

Javascript Objects

Basic Object Oriented Concepts

- An object in JavaScript is a reference data type.
- An object can be compared to any real world entities.
- An object is an unordered list of properties consisting of a name (always a string) and a value. When the value of a property is a function, it is called a method.

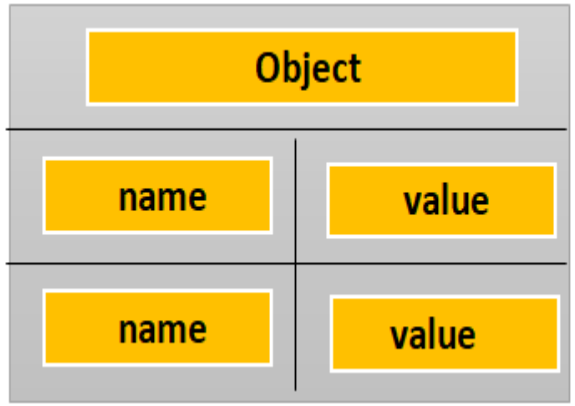


Fig 1:Object Structure

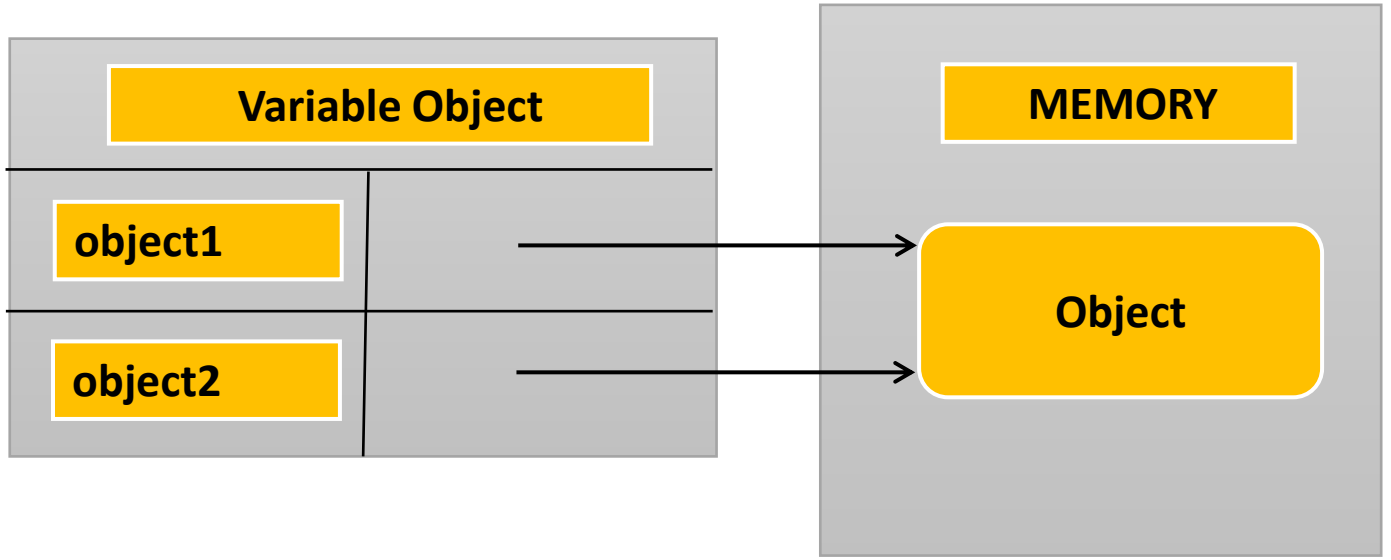


Fig 2:Object Reference

```
Var object1= new  Object();
Var object2 = object1;
```

Javascript Objects

Object Literals using {key : value}

```
var emp2 = {
    "firstName": "Aruna",
    "lastName": "Srinivasan",
    "showName": function () {
        alert(this.firstName + " " + this.lastName);
    }
};
emp2.showName();
```

- A variant of object literal syntax
- Specifies object members and their values inside the curly brackets as key-value pairs
- A member and its value are delimited using colon (:) character.

Javascript Objects

Creating an object using function Constructor

```
function Employee(fname,lname) {
    this.firstName = fname;
    this.lastName = lname;
    this.showName = function () {
        alert(this.firstName + " " + this.lastName);
    };
}
var emp5 = new Employee("Aruna", "Srinivasan");
emp5.showName();
```

- JavaScript functions are objects themselves. It can be called as a Constructor Function.
- Once the function constructor is created you create object of that function using new keyword.
- The function name uses camel case convention

Javascript Objects

Creating an object using anonymous function

```
var emp6 = new function () {
    this.firstName = "Aruna";
    this.lastName = "Srinivasan";
    this.showName = function () {
        alert(this.firstName + " " + this.lastName);
    };
}
emp6.showName();
```

- Two steps are performed here
  - Firstly, it creates an anonymous constructor function
  - Secondly, it calls new on it to create its object.

Javascript Objects
Prototype property

- Prototype property of an object holds the structure of that object
- It is shared by all object instances created using that constructor
- This can be used to modify/add properties to all instances after they have been created
- Methods should be added to prototype since only one copy of the method is created. If methods are added to constructor, each object instance holds its own copy of the method.

Employee.prototype.showName = function(){…}

Javascript Objects

Creating an object using EcmaScript 6 Class Keyword

```
class Employee {
    constructor(fname, lname)
      {  this.firstName = fname;
        this.lastName = lname;   }
        showName()
        {   alert(this.firstName + " " + this.lastName); }
}
var emp7 = new Employee("Aruna", "Srinivasan");
emp7.showName();
emp7.firstName = "Sanvi";
emp7.showName();
```

- Class is defined .
- Then the objects of the class are created.
- It internally uses the constructor/prototype based approach of creating objects

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering
vinayj@pes.edu