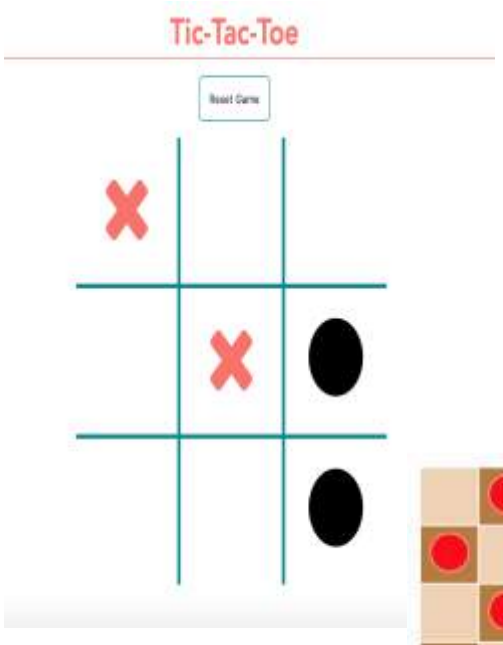
WEB TECHNOLOGIES

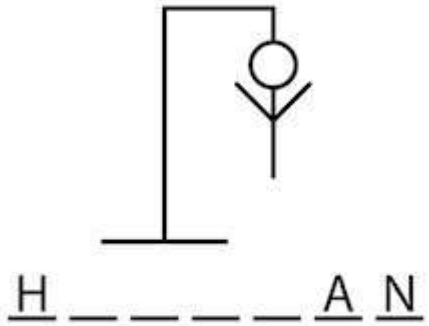
Document Object Model

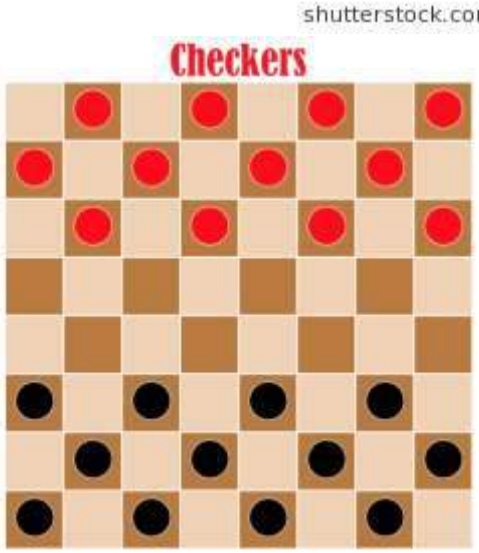
Vinay Joshi
Department of
Computer Science and Engineering

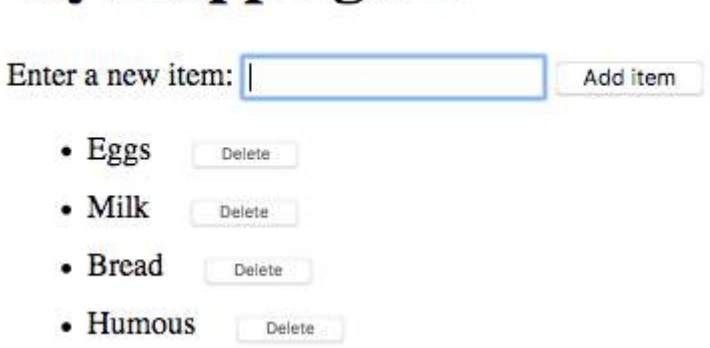
Document Object Model

How does JavaScript interact with HTML / CSS?









Document Object Model

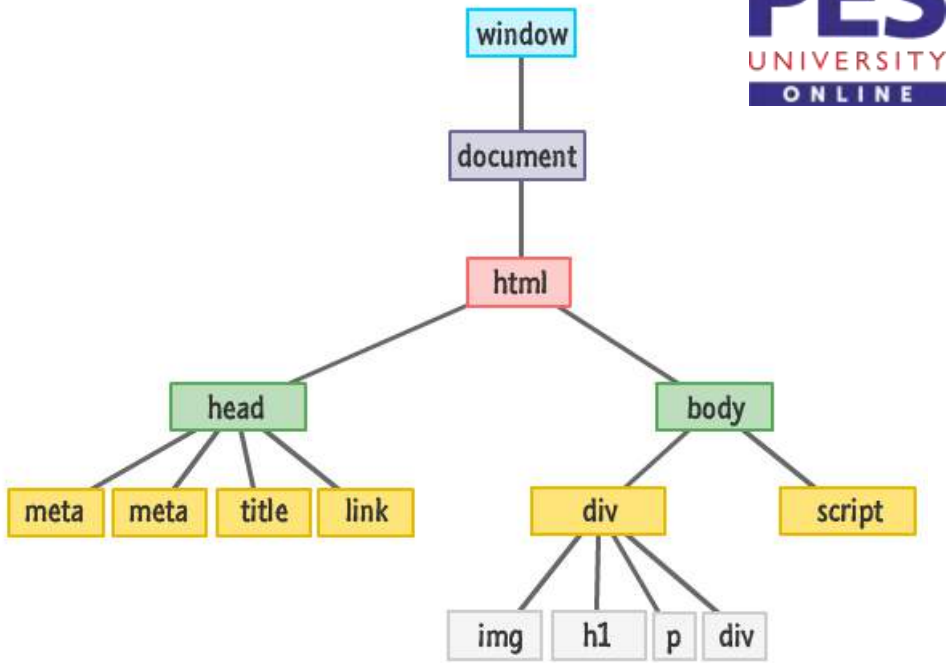
Drawbacks of using document.write()

- Document.write executed after the page has finished loading will overwrite the page, or write a new page, or not work
- Document.write practically only appending to the page

Document Object Model

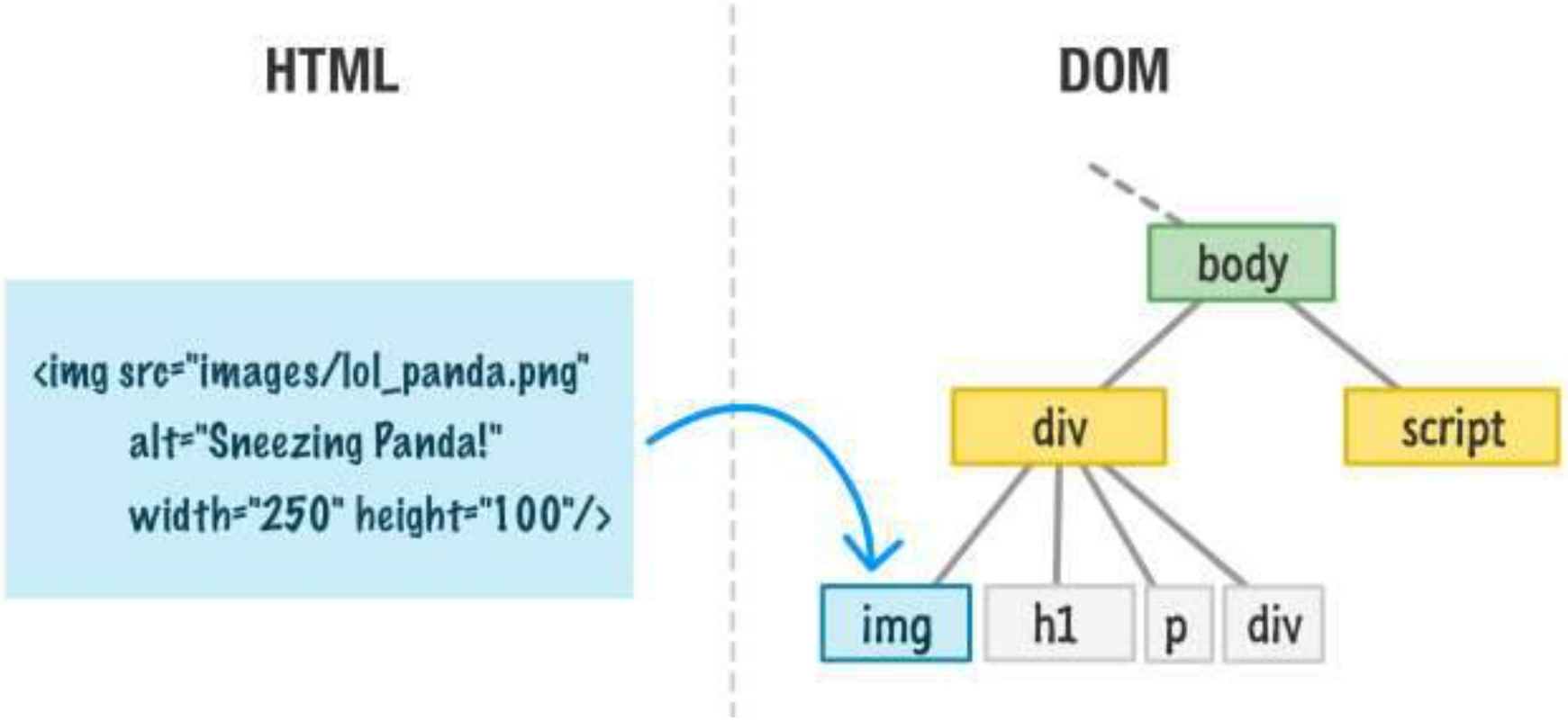
Introduction to DOM

- A Web page is a document. This document can be either displayed in the browser window or as the HTML source. But it is the same document in both cases.
- The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.



Document Object Model

DOM Elements are Objects



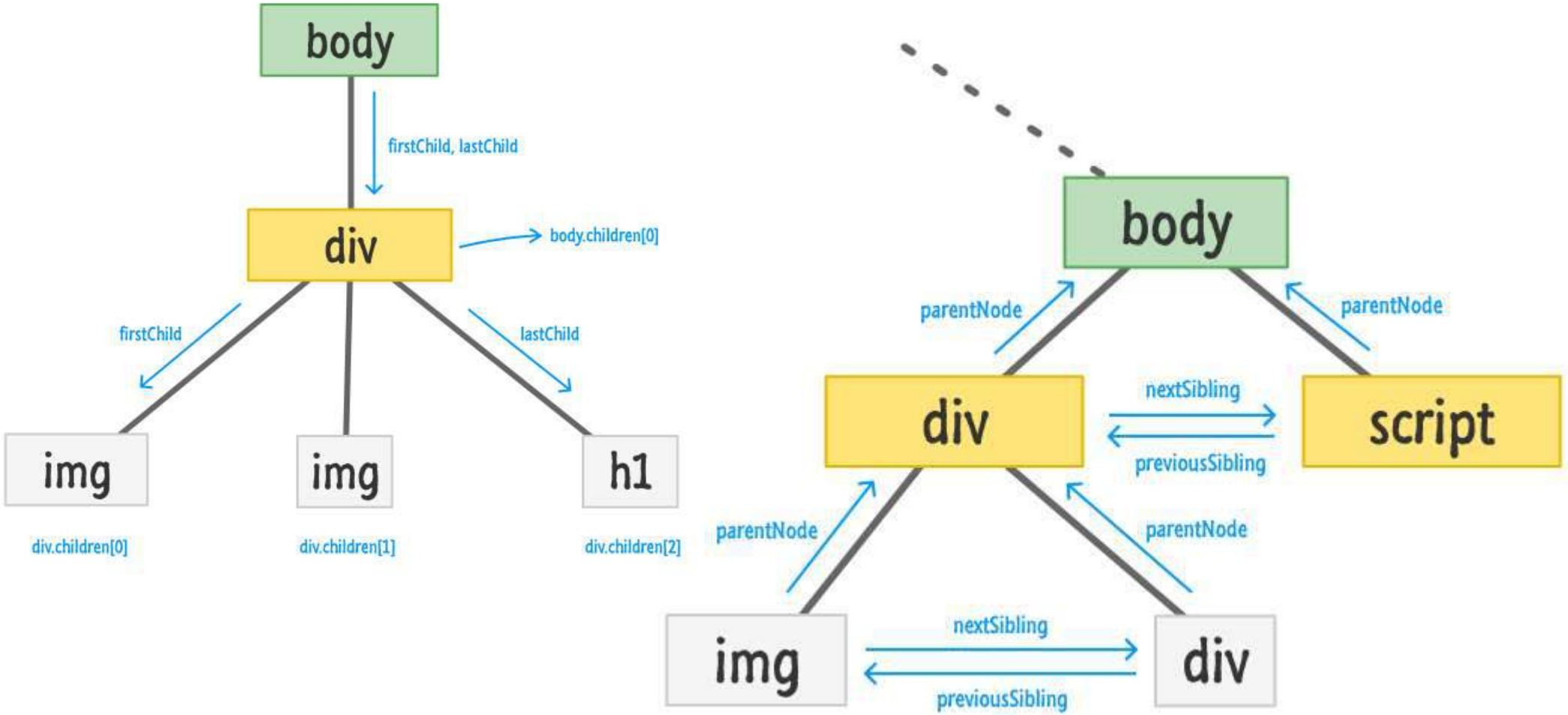
Document Object Model

Accessing Elements in DOM

Access Element By	Equivalent Selector	Method
ID	#demo	getElementByID(“demo”)
Class	.demo	getElementsByClassName(“demo”)
Tag	<tag name> like p	getElementsByTagName(“p”)
Selector (single)	Any CSS Selector	querySelector(“selector”)
Selector (all)		querySelectorAll(“selector”)

Document Object Model

Traversing the DOM



Document Object Model

Creating Element Objects

Method	Description
document.createElement()	Create a new element node using tag
document.createTextNode()	Create a new text node

Property	Description
node.textContent or node.innerText	Get or set the text content of an element node (without HTML tags)
node.innerHTML	Get or set the HTML content enclosed in the element tag

Document Object Model

Manipulating Nodes in the DOM

Method	Description
node.appendChild()	Add a node as the last child of the parent element.
node.insertBefore()	Insert a node into the parentbefore a specific sibling node
node.replaceChild()	Replace an existing node with a new node
node.removeChild()	Removes child node
node.remove()	Removes a node

* **node** here can be document.body or any existing element in the DOM

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

Events

Vinay Joshi
Department of
Computer Science and Engineering

EVENT

What are Events?

EVENT

A simple event handler Example

```
<form method="POST" action="">
    <input type="button"
           name="myButton"
           value="Click Me"
           onclick="alert('you clicked the button');">
</form>
```

EVENT

Event Handlers and Event Listeners

- Events are created by activities associated with specific HTML elements.
- The process of connecting an event handler to an event is called **registration**.
- There are two distinct approaches to event handler registration,
  - Assign element attributes
    - Inline event handlers
  - Assign handler addresses to object properties
    - Event handler properties and Event listeners

Event

Event Handler Properties

- An element can be assigned the event handler property

  element.on<event> = handler;

- It involves two parts
  - the correct event name it is to be listening for
  - the handler callback function.

For example:

div.onclick = change; or

div.onmouseover = function(){ … };

Event

Event Listeners

- An event listener watches for an event on an element.

  element.addEventListener(event, handler)

- It takes two mandatory parameters
  - the event it is to be listening for.
  - the handler callback function.

For Example:

div.addEventListener(“click”, change);

div.addEventListener(“keypress”, function(){ … });

Events

Event Sources and example events

Source	Event	Fires When…
Mouse	click	the mouse is clicked and released on an element
	dblclick	an element is clicked twice
	mousemove	every time a mouse pointer moves inside an element
	mouseover	every time a mouse pointer is placed over an element
Keyboard	keydown	when a key is pressed down
	keyup	when a key pressed is released
	keypress	when a key is pressed and released
Form	submit	a form is submitted
	reset	a form reset button is clicked
	focus	an input element is clicked and receives focus
	blur	an input element loses focus

Event

Event Object Properties

- Event object holds the context or details of the event

Property	IE5-8 Equivalent	Specifies
target	srcElement	the target of the event (most specific element).
type	-	the name of event fired (without the on prefix)
altKey / shiftKey / ctrlKey / metaKey	-	true/false to signify if Alt Key or Shift Key or Ctrl Key or Meta Key was pressed
charCode	keyCode	Unicode character code of the pressed key
key	-	Key Character Name (‘a’ or ‘F1’ or ‘CAPS LOCK’)
button	-	Returns which mouse button was pressed
clientX, clientY / offsetX, offsetY / screenX, screenY	-	the coordinates of the mouse pointer when the event triggered, relative to, the current window / target element / screen

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

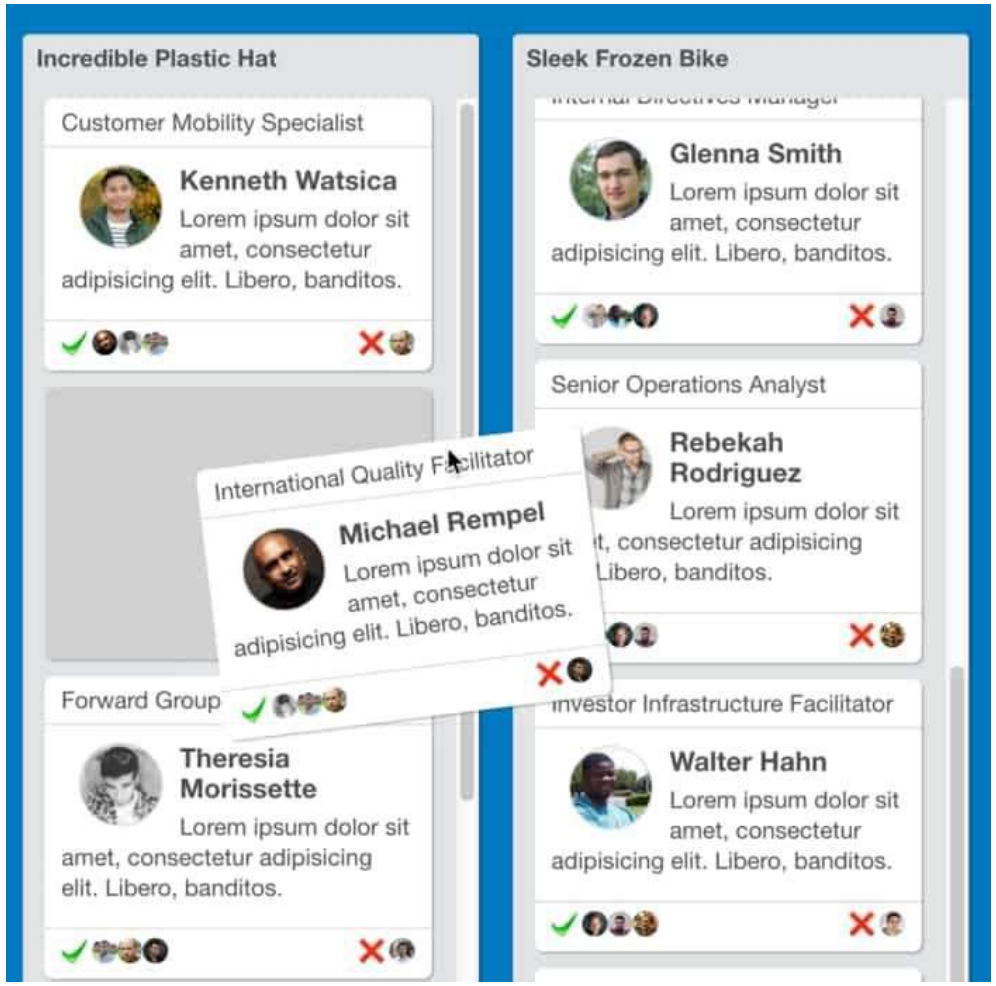
+91 80 2672 6622

WEB TECHNOLOGIES

Event Handling

Vinay Joshi
Department of
Computer Science and Engineering

Event Handling
Event Propagation



Event Handling

Event Flow

There are three phases in which an event can propagate to handlers defined in parent elements:

- Capturing phase
- Target phase
- Bubbling phase

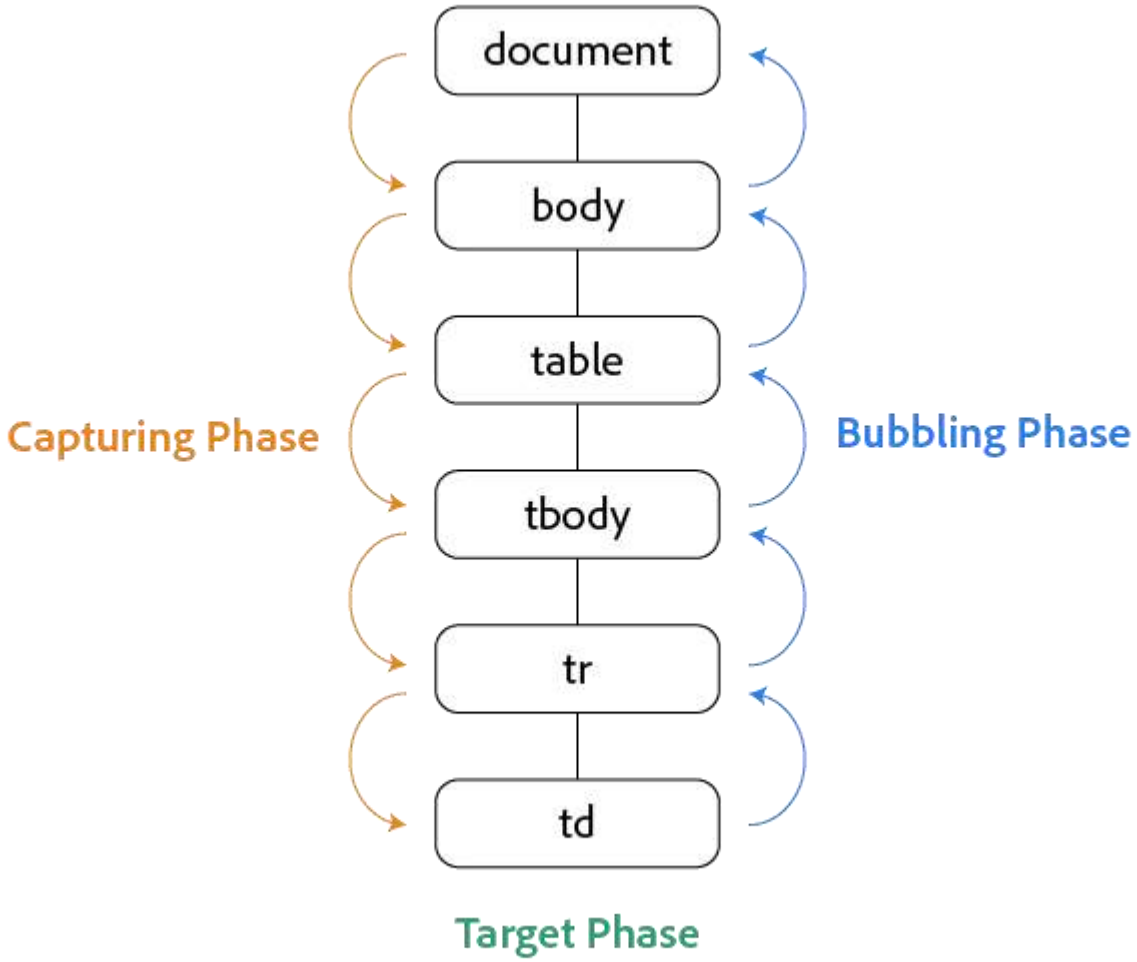
elem.addEventListener("event", func_ref, **flag**);

flag = true :=> Handler registered for Capturing phase

flag = false:=> Handler registered for Bubbling phase (default)

Event Flow

Event Capturing



Event Handling
Event Object Properties

- Event object has properties and methods related to Bubbling and Capturing

Property/ Method	IE5-8 Equivalent	Purpose
cancelBubble	-	A historical alias to stopPropagation(). Setting its value to **true** before returning prevents propagation
eventPhase	-	Specifies which phase of the event flow is being processed
cancelable	Not supported	Indicates whether you can cancel the default behaviour of an element
preventDefault()	returnValue	It cancels the default behavior of the event (if possible)
stopPropogation()	cancelBubble	It stops any further bubbling/ capturing of the event.

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

HTML 5 – Audio, Video and Progress elements

Vinay Joshi
Department of
Computer Science and Engineering

HTML5 - Audio

Video elements

- A standard approach for browser to play audio without the use of plug-in

Syntax

```
<audio controls="controls">
    <source src="song.ogg" type="audio/ogg" />
    <source src="song.mp3" type="audio/mp3" />
    Your browser does not support the audio element.
</audio>
```

HTML5 - Audio

Audio element properties

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready.
controls	controls	Specifies that controls will be displayed, such as a play button.
loop	loop	Specifies that the audio will start playing again (looping) when it reaches the end
preload	preload	Specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	*url*	Specifies the URL of the audio to play

HTML5 - Video

Video elements

- A standard approach for browser to play video without the use of plug-in

Syntax

```
<video width="320" height="240" controls="controls">
    <source src="movie.ogg" type="video/ogg" />
    <source src="movie.mp4" type="video/mp4" />
    <source src="movie.webm" type="video/webm" />
    Your browser does not support the video tag.
</video>
```

HTML5 - Video

Video element properties

Attribute	Value	Description
audio	muted	Defining the default state of the the audio. Currently, only "muted" is allowed
autoplay	autoplay	If present, then the video will start playing as soon as it is ready
controls	controls	If present, controls will be displayed, such as a play button
height	*pixels*	Sets the height of the video player
loop	loop	If present, the video will start over again, every time it is finished
poster	*url*	Specifies the URL of an image representing the video
preload	preload	If present, the video will be loaded at page load, and ready to run. Ignored if "autoplay" is present
src	*url*	The URL of the video to play
width	*pixels*	Sets the width of the video player

HTML5 - Video

Progress elements

- The <progress> tag represents the completion progress of a task.
- Always add the <label> tag for describing the task!
- Use JavaScript to manipulate the value of the progress bar

Syntax:

```
<label for="file">Downloading progress:</label>
<progress id="file" value="32" max="100"> 32% </progress>
```

Attribute	Value	Description
max	*number*	Specifies how much work the task requires in total. Default value is 1
value	*number*	Specifies how much of the task has been completed

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

HTML 5 – Canvas & SVG

Vinay Joshi
Department of
Computer Science and Engineering

HTML5 - Canvas

Canvas element

- Uses JavaScript to draw graphics on a web page
- A rectangular area, and you control every pixel of it

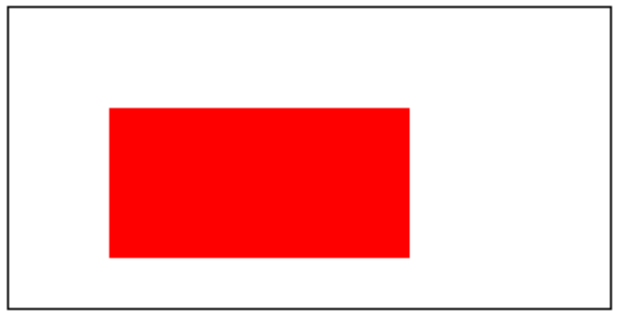
Syntax

```
<canvas id="myCanvas" width="200" height="100">
    Canvas is not supported
</canvas>
```

Canvas – context object

- The canvas element has no drawing abilities of its own. All drawing must be done inside a JavaScript using the context object

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctx=c.getContext("2d");
    ctx.fillStyle="#FF0000";
    ctx.fillRect(50,50,150,75);
</script>
```



HTML5 - Canvas

Canvas – context methods

Method	Description
fillRect(x, y, width, height)	Draws a filled rectangle
strokeRect(x, y, width, height)	Draws a rectangular outline
clearRect(x, y, width, height)	Clears the specified rectangular area, making it fully transparent
moveTo(x, y)	Moves the pen to the coordinates specified by x and y
lineTo(x, y)	Draws a line from the current drawing position to the position specified by x and y
arc(x, y, r, sAngle, eAngle, anticlockwise)	Draws an arc centered at (x, y) with radius r starting at sAngle and ending at eAngle going anticlockwise (defaulting to clockwise).
arcTo(x1, y1, x2, y2, radius)	Draws an arc with the given control points and radius, connected to the previous point by a straight line

HTML5 - Canvas

Canvas – context methods (cntd.)

Method	Description
createLinearGradient(x1, y1, x2, y2)	Creates a linear gradient object with a starting point of (x1, y1) and an end point of (x2, y2).
createRadialGradient(x1, y1, r1, x2, y2, r2)	Creates a radial gradient. The parameters represent two circles, one with its center at (x1, y1) and a radius of r1, and the other with its center at (x2, y2) with a radius of r2.
fillText(text, x, y [, maxWidth])	Fills a given text at the given (x,y) position. Optionally with a maximum width to draw.
strokeText(text, x, y [, maxWidth])	Strokes a given text at the given (x,y) position. Optionally with a maximum width to draw.
drawImage(image, x, y [,width, height])	Draws the CanvasImageSource specified by the image parameter at the coordinates (x, y) with optional width and height

HTML5 – SVG

SVG Element

- SVG stands for Scalable Vector Graphics.
- SVG defines vector-based graphics using HTML elements
- SVG graphics do NOT lose any quality if they are zoomed or resized

```
<svg width="100" height="100">
     <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4"
    fill="yellow" />
</svg>
```

HTML5 – SVG

SVG – Predefined Shape Element

- <rect width="300" height="100" style = "fill:rgb(0,0,255); stroke-width:3; stroke:rgb(0,0,0)" />
- <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
- <ellipse cx="200" cy="80" rx="100" ry="50“ style = "fill:yellow; stroke:purple; stroke-width:2" />
- <polygon points="200,10 250,190 160,210" style = "fill:lime; stroke:purple; stroke-width:1" />
- <text x="0" y="15" fill="red" transform="rotate(30 20,40)">I love SVG</text>

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

HTML 5 – Geo Location & Web Workers

Vinay Joshi
Department of
Computer Science and Engineering

HTML5 – Geo Location

Introduction

- Enables your web application to obtain the geographical position of your website visitors
- The user has to accept to share their location
- Accessed via JavaScript, through the **navigator.geolocation** object

HTML5 – Geo Location

Methods

- navigator.geolocation object allows you to access geo location through two primary functions:
  - getCurrentPosition()
    - Returns the location of the visitor as a one-time snapshot
  - watchPosition()
    - returns the location of the visitor every time the location changes
- Both functions take the following parameters:
  - Success callback function
  - Error callback function (optional)
  - Geo location options object (optional)

HTML5 – Geo Location

Associated Objects

- Success callback function receives **position** object with these read only properties
  - double latitude
  - double longitude
  - double accuracy
  - double altitude
  - double altitudeAccuracy
  - double heading (direction)
  - double speed

- Error callback function receives **error** object with these two properties
  - short code
    - 1, meaning PERMISSION_DENIED
    - 2, meaning POSSITION_UNAVAILABLE
    - 3, meaning TIMEOUT
  - DOMString message

- **Options** object (third parameter to getCurrentPosition or watchPosition)
  - enableHighAccuracy // true or false
  - timeout // milliseconds
  - maximumAge // milliseconds

HTML5 – Web Worker

Introduction

- A web worker is essentially a thread executing a JavaScript file
- Makes it possible to execute a JavaScript file asynchronously and autonomously
- Helps achieve multi threading in your web applications

- To create a web worker:

```
var worker = new Worker(“myasync.js");
```

- Parameter is the URL of the JavaScript file to execute

HTML5 – Web Worker

Restrictions

- A web worker does not have access to the DOM of the page that creates the web worker.
- Here is a list of what a web worker can do:
  - Listen for messages, using the onmessage event listener function.
  - Send messages via the postMessage() function.
  - Send AJAX requests using the XMLHttpRequest.
  - Create timers using the setTimeout() and sendInterval() functions.

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

jQuery – JavaScript Library

Vinay Joshi
Department of
Computer Science and Engineering

jQuery

Introduction

- Simplifies the interaction between HTML and JavaScript
- Lightweight : 19KB in size (Minified and Gzipped)
- CSS1 to CSS3 Complaint
- Cross Browser
  - (IE 6.0+, FF 2+, Safari 3.0+, Opera 9.0+, Chrome)

jQuery

Including the Library

- Download jQuery from http://docs.jquery.com/Downloading_jQuery
- Include the library in your web page

```
<head>
<script src="path/to/jquery-3.5.1.min.js"></script>
</head>
```

OR

- Include from a CDN (Content Delivery Network)

```
<head>
<script src = "https://code.jquery.com/jquery-3.5.1.min.js"> </script>
</head>
```

jQuery

Write less code, to do more

Vanilla JavaScript:

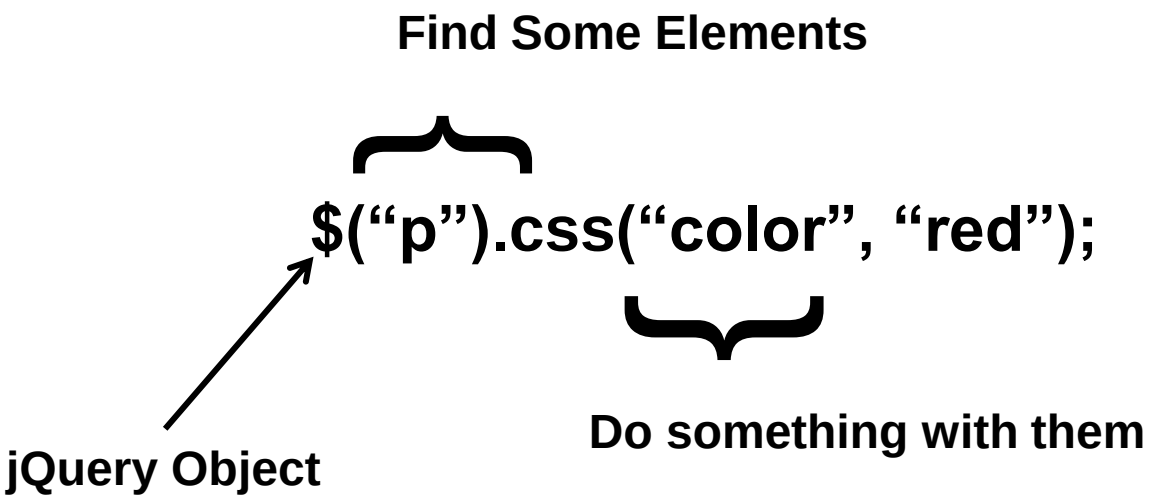
```
let paras = document.querySelectorAll(“p”)
for (let i=0; i<paras.length; i++)
      paras[i].style.color = “red”
```

jQuery:

```
$(“p”).css(“color”, “red”);
```

jQuery

Statement Structure



jQuery

Selecting Elements – using CSS Selectors

Select By	Example
ID	$(“#header”)
Class	$(“.updated”)
Tag Name	$(“table”)
Combination	$(“table.user-list”) or $(“#footer ul.menu li”)
Basic Filters	:first, :last, :even, :odd
Content Filters	:empty , :contains(text), :has(selector)
Attribute Filters	[attribute], [attribute=value], [attribute!=value]
Forms	:input, :text, :submit, :password, :enabled, :checked

jQuery

The Action Part

Action	Example
DOM Manipulation	before(), after(), append(), appendTo()
Attributes	addClass(), css(), attr(), html(), val(), text()
Events	click(), on(), bind(), unbind(), live()
Effects	hide(), fadeOut(), toggle(), animate()
AJAX	load(), get(), ajax(), post(), getJSON()

jQuery
DOM Manipulation Methods

- append() – adds a set of elements to the end of the children
  - **$("pelem").append($c1[, c2, …])**
    // $c1, $c2, … will be appended to child elements of $pelem
  - Similar methods : appendTo, prepend
- after() - adds a set of elements after the specified element
  - **$("elem").after($e1[, $e2,…])**
    //$e1, $e2 will be added after $elem under the same parent
  - Similar methods : insertAfter, before, insertBefore

jQuery
Attribute Methods

- Attribute Methods like css(), attr(), val(), html(), text() can be used for both setting and getting attributes
- Based on whether one or two arguments were passed

- Example:
  - For Setting
    $(“p:last”).css(“color”, “green”);
  - For Getting
    let pcolor = $(“p:last”).css(“color”)

jQuery
Chaining Methods

- Most jQuery methods return jQuery object
- You can chain them together to perform multiple operations on the same elements

  $(“#deleted”).addClass(“red”).fadeOut(“slow”);

- This will not work as val() returns a string

  $(“:button”).val(“Click Me”).click(function(){…})

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

jQuery – Events and Effects

Vinay Joshi
Department of
Computer Science and Engineering

jQuery – Events and Effects

The Action Part

Action	Example
DOM Manipulation	before(), after(), append(), appendTo()
Attributes	addClass(), css(), attr(), html(), val()
Events	click(), bind(), unbind(), live()
Effects	hide(), fadeOut(), toggle(), animate()
AJAX	load(), get(), ajax(), getJSON()

jQuery – Events

Event Methods

- Register an event handler for an element object passing a function reference to the event method

  $(“span#message”).click(function(event){…});

  OR

  $(“span#message”).on(“click”, function(event) {…});

jQuery – Events

Manually triggering Event

- Like with attribute methods, the event methods have a different meaning without arguments

- Without the function reference argument, the event methods are treated like a manual firing of event

  $(“span#message”).click();

jQuery – Events

this and ready

- Special event to register for handler once the document has been loaded completely is ready

```
$(document).ready(function(){
        …
});
```

- within any event handler function **this** element refers to the element for which the handler is called

```
$( "p" ).click(function() {
        var htmlString = $( this ).html();
        …
}
```

jQuery – Effects

Effect Methods

- Lot of animation/styling effects can be accomplished using the effects methods like hide, show, toggle, fadein, fadeout etc.

```
$(“a#show-cart”).click(function(){
    $(“#cart”).slideToggle(“slow”);
});
```

- Or, use the animate method to build custom animations

```
$("#showdown").click(function(){
    $("#my-div").animate({
        width: "70%",
        opacity: 0.4,
        fontSize: "3em“
    }, 1200 );
});
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

Callback and Promises

Vinay Joshi
Department of
Computer Science and Engineering

Callbacks

Introduction

- As seen in setInterval, setTimeout and addEventListener, a function accepts a function reference as an argument
- They will be called asynchronously based on timer or other events
- These function references are called **Callback functions**

- Example:
  div.addEventListener(“keypress”, **function(){ … }**);

Promises

Introduction

- A promise is used to handle the asynchronous result of an operation.
- With Promises, we can defer execution of a code block until an async request is completed.
- The Promise object is created using the new keyword and contains the promise; this is an executor function which has a **resolve** and a **reject** callback
- Essentially, a promise is a returned object to which you attach callbacks, instead of passing callbacks into a function.

```
const promise = new Promise(function(resolve, reject) {
  // promise description
});
```

Promises

Example


```
var weather;
const date = new Promise(
    function(resolve, reject) {
        weather = true; //usually a API call
        if (weather) {
                const dateDetails = {
                        name: 'Cubana Restaurant',
                        location: '55th Street',
                        table: 5
                };
                resolve(dateDetails)
        } else {
                reject(new Error('Bad weather'))
        }
    }
);
```

```
date
.then(function(done) {
    console.log('We are going on a date!')
    console.log(done)
    })
.catch(function(error) {
    console.log(error.message)
    })
```

Callback and Promises

Comparison

- Callbacks and Promises are not the same

- Callbacks are function passed to another function as a reference
- Chaining of Callbacks can be clumsy and lead to **Callback Hell**

- Promises use Callbacks and more elegant than Callbacks
- Chaining of Promises is supported

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

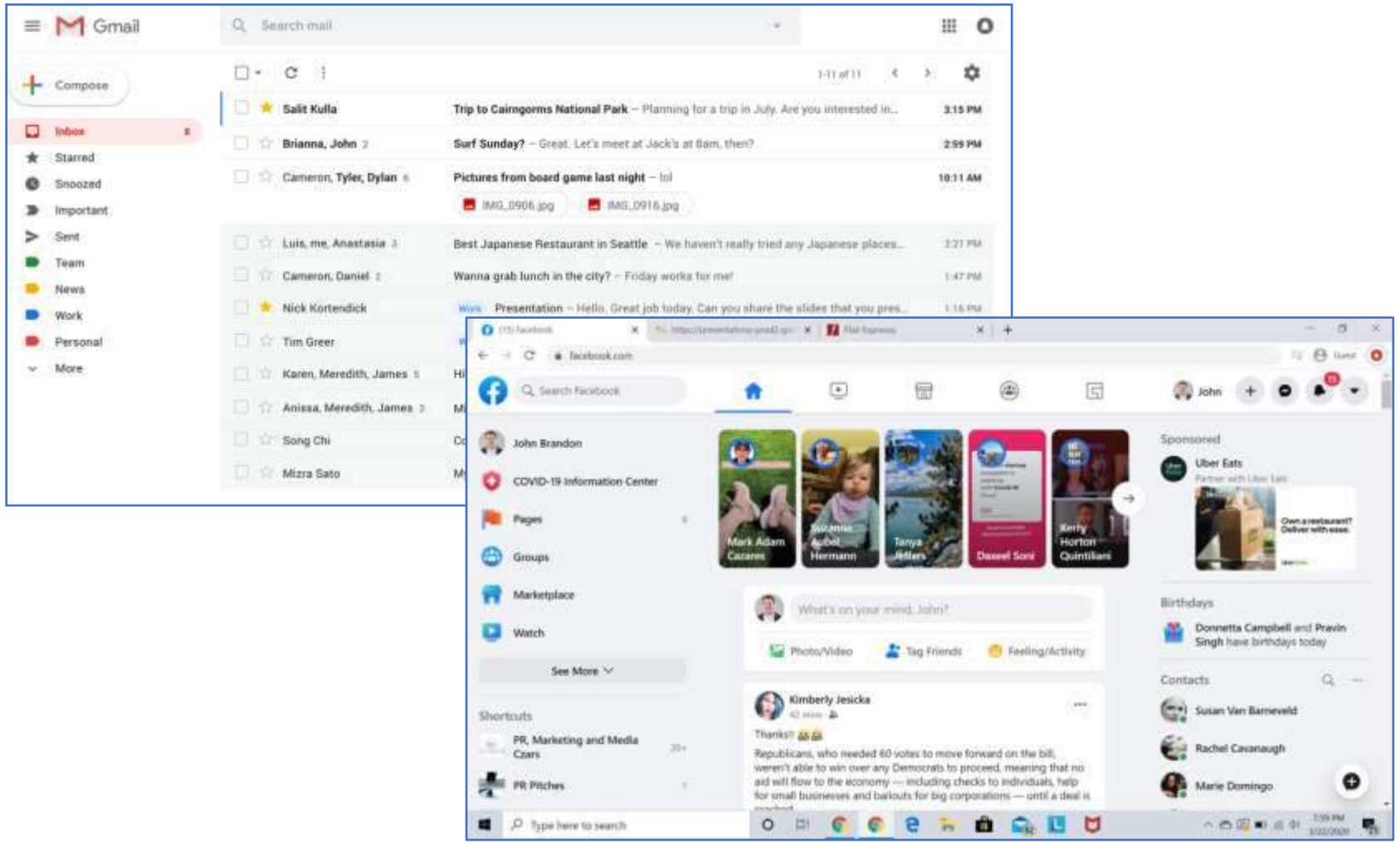
WEB TECHNOLOGIES

Single Page Applications & Asynchronous Communication

Vinay Joshi
Department of
Computer Science and Engineering

Single Page Applications

Introduction



Single Page Applications

Introduction

- Instead of the default method of the browser loading entire new pages, a single-page application (SPA) interacts with the web browser by dynamically rewriting the current web page with new data from the web server
- Resources are dynamically loaded and added to the page as necessary, usually in response to user actions
- The page does not reload at any point in the process, nor does it transfer control to another page

- Can be built using
  - AJAX
  - Frameworks like ReactJS, AngularJS

Asynchronous Communication

AJAX

- Traditional web applications, upon request from user like clicking a link or submitting a form, a new page is loaded with requested resources
- Asynchronous applications, upon user action, updates a part of the page without reloading the entire page
- Approaches include
  - Setting src property of iFrame or img element
  - A more elegant and complete approach is use of XHR or XMLHttpRequest object

- First create an XHR object using

  var xhr = new XMLHttpRequest();

Asynchronous Communication

XHR object properties and methods

Properties / Methods	Description
open(*method, url [, asynchronous])*	Initializes the request in preparation for sending to the server. The *method parameter is the HTTP method to use,* for example "GET" or "POST". The *url is the relative or absolute URL the* request will be sent to. The optional *asynchronous parameter* indicates whether *send() returns immediately or after the* request is complete (default is true, meaning it does not wait for response to come back)
onreadystatechange	Function to call whenever the readyState changes
send([*body])*	*Initiates the request to the server. The body parameter* should contain the body of the request, i.e., a string containing *fieldname=value&fieldname2=value2… for POSTs* or a null value for GET request

Asynchronous Communication

XHR object properties and methods

Properties / Methods	Description
readyState	Integer indicating the state of the request, either: 0 (uninitialized) 1 (loading) 2 (response headers received) 3 (some response body received) 4 (request complete)
status	HTTP status code returned by the server (e.g., 200, 404, etc.)
responseText	Full response from the server as a string (responseType property is set to “text” – default)
responseXML	A Document object representing the server’s response parsed as an XML document (responseType property is set to “document”)
response	Any other type of response received ((responseType property is set to “blob” or “json”)

Asynchronous Communication

XMLHttpRequest – Code Example

```
let xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", filepath, true);
xmlhttp.onreadystatechange=handler;
[xmlhttp.responseType=“json”|“document” |“blob”] // default text
xmlhttp.send(null);

function handler() {
      if(this.readyState == 4  && this.status == 200) {
            // use this.response (json/blob) or this.responseText (text) or
            // this.responseXML (document) to update a part of the page
      }
}
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622

WEB TECHNOLOGIES

jQuery AJAX and fetch() methods

Vinay Joshi
Department of
Computer Science and Engineering

jQuery AJAX methods

Introduction

- jQuery provides methods that use XMLHttpRequest internally to make AJAX requests

- The methods are
  - $.ajax
  - $.get
  - $.post
  - $(“elem”).load

- In a single method call, the entire functionality of making an AJAX call using XMLHttpRequest and updating the page can be achieved

jQuery AJAX methods

$.ajax

- Syntax
  - $.ajax({name:value, name:value, ... })
- Example
  - $.ajax({url: "demo_test.txt", success: function(result){
    $("#div1").html(result);
    }});
  - $.ajax('/jquery/submitData', {
    type: 'POST', // http method
    data: { myData: 'This is my data.' }, // data to submit
    success: function (data, status, xhr) { // success callback function
    $('p').append('status: ' + status + ', data: ' + data);
    },
    error: function (jqXhr, textStatus, errorMessage) {
    $('p').append('Error' + errorMessage);
    }
    });

jQuery AJAX methods

$.get

- Syntax
  - $.get(url, [data],[callback]);
- Example
  - $.get('/jquery/getjsondata', // url
    {name:'Steve'}, // request parameters
    function (data, textStatus, jqXHR) { // success callback function
    $('p').append(data.firstName);
    }
    );
- Other Variants
  - $.getJSON(url, [data],[callback]);
  - $.getScript(url, [data],[callback]);

jQuery AJAX methods

$.post

- Syntax
  - $.post(url,[data],[callback],[type]);
- Example
  - $.post('/jquery/submitJSONData', // url
    { myData: 'This is my data.' }, // data to be submitted
    function(data, status, jqXHR) { // success callback function
    $('p').append('status: ' + status + ', data: ' + data);
    },
    “json” //response type
    );
- Internally uses $.ajax with method=“post”

jQuery AJAX methods

$.load

- Allows HTML or text content to be loaded from a server and added into an existing DOM element
- Syntax
  - $(‘selector’).load(url,[data],[callback],[type]);
- Example
  - ```
    $('#msgDiv').load('getData',                    // url
                      { name: 'bill' },             // data
                      function(data, status, jqXGR) {   // success callback function
                          console.log('data loaded')
                      }
    );
    ```
    ```

fetch() AJAX method

Introduction

- Starts the process of fetching a resource from the network
- Returns a promise which is fulfilled once the response is available
- The promise resolves to the Response object representing the response to your request
- The promise does not reject on HTTP errors — it only rejects on network errors. You must use **then** handler to check for HTTP errors.

- Syntax

```
const fetchResponsePromise = fetch(resource [, init])
```

fetch() AJAX method

Code Example


```
const mydiv = document.querySelector('.my-div');

fetch(‘resp.html')              // returns a promise that resolves on response object
.then(function(response) {
    return response.text();  // returns a promise that resolves on text response
})
.then(function(text) {
    mydiv.innerHTML= text;
});
```

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu

+91 80 2672 6622