Unit-II Client-side Technologies

JavaScript & DOM

Vijayendra S. Gaikwad , PICT

Vijayendra S. Gaikwad , PICT

Outline

JavaScript: Overview of JavaScript, using JS in an HTML (Embedded, External), Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS, JavaScript debugger

DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM,

JQuery: Introduction to JQuery, Loading JQuery, Selecting elements, changing styles, creating elements, appending elements, removing elements, handling events.

Introduction to JavaScript

- JavaScript is one of the **3 languages** all web developers **have to learn**:
- 1. **HTML** to define the content of web pages
- 2. **CSS** to specify the layout of web pages
- 3. **JavaScript** to program the behavior of web pages

- JavaScript is a powerful **client-side scripting language** used in web pages to dynamically add functionality, validate or auto-correct form data, communicating with server (with AJAX) and read-write HTML elements (innerHTML property, enable/disable a button)

JavaScript

- JavaScript is a front-end scripting language developed by **Brendan Eich** at **Netscape** for dynamic content.
  - Lightweight, but with limited capabilities
  - Can be used as object-oriented language
- **Client-side technology:**
  - Embedded in your HTML page or create an External ***.js*** file
  - Web browser are equipped with JavaScript interpreter
- Simple, un-typed and high-level language
- Powerful to manipulate the DocumentObjectModel
- Every browser use some DOM to make web pages accessible via JS

Vijayendra S. Gaikwad, PICT

JavaScript Advantages

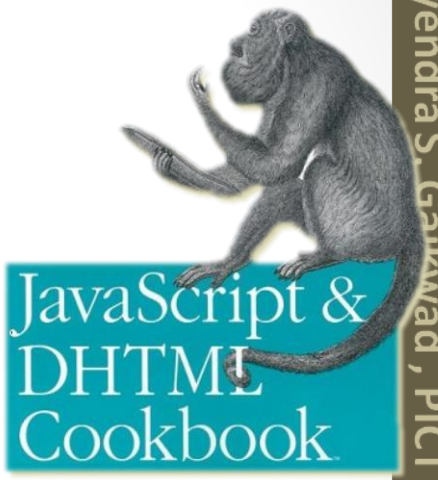
- JavaScript **allows interactivity** such as:
  - Implementing form validation
  - React to user actions, e.g. highlight paragraphs
  - Changing an image on moving mouse over it
  - Sections of a page appearing and disappearing
  - Content loading and changing dynamically
  - Performing complex calculations
  - Custom HTML controls, e.g. scrollable table
  - Implementing AJAX functionality

What Can JavaScript Do?

- Can handle events
- Can read and write HTML elements
- Can validate form data
- **Can access / modify browser cookies**
- **Can detect the user's browser and OS**
- Can be used as object-oriented language
- Can handle exceptions
- Can perform asynchronous server calls (AJAX)

The JavaScript Syntax

```
if (pop < 10)
{
     map.graphics.add(features[i].setSymbol(onePopSymbol));
}
else if (pop >= 10 && pop < 95)
{
     map.graphics.add(features[i].setSymbol(twoPopSymbol));
}
else if (pop >= 95 && pop < 365)
{
     map.graphics.add(features[i].setSymbol(threePopSymbol));
}
else if (pop >= 365 && pop < 1100)
{
     map.graphics.add(features[i].setSymbol(fourPopSymbol));
}
else
{
     map.graphics.add(features[i].setSymbol(fivePopSymbol));
}
```



JAVA SCRIPT

Vijayendra S. Gaikwad, PICT

Vijayendra S. Gaikwad, PICT

JavaScript Syntax

- JavaScript can be implemented using **<script>.....</script>** tags in a HTML web page.
- Place the **<script>** tags, within the **<head>** tags.

- **Syntax:**

```
<script  type="text/javascript">
        // JavaScript code
</script>
```

JavaScript - Placement in HTML File

- There is flexibility to include JavaScript code anywhere in an HTML document.
- However the most preferred ways to include JavaScript in an HTML file are as follows −
  - Script in <head>...</head> section.
  - Script in <body>...</body> section.
  - Script in <body>...</body> and <head>...</head> sections.
  - Script in an external file is usually included in <head>...</head> section only.

JavaScript Editor and Extension

Use Notepad to write the code

Save the document using .html (if embedded JavaScript)

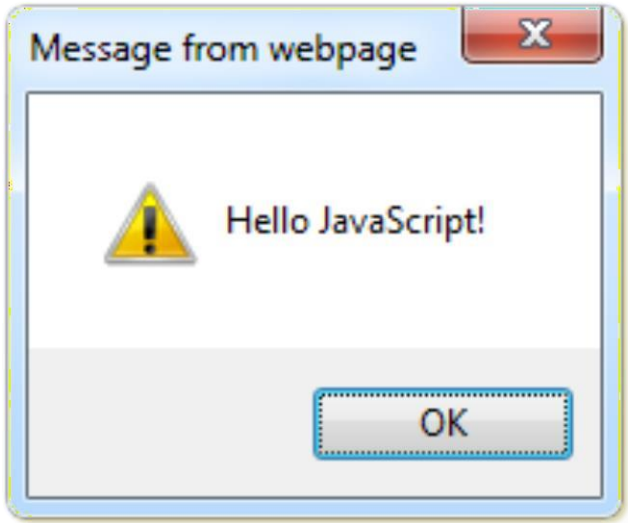
Save document with .js (if external JavaScript)

Run the target .html file using browser

The First Script

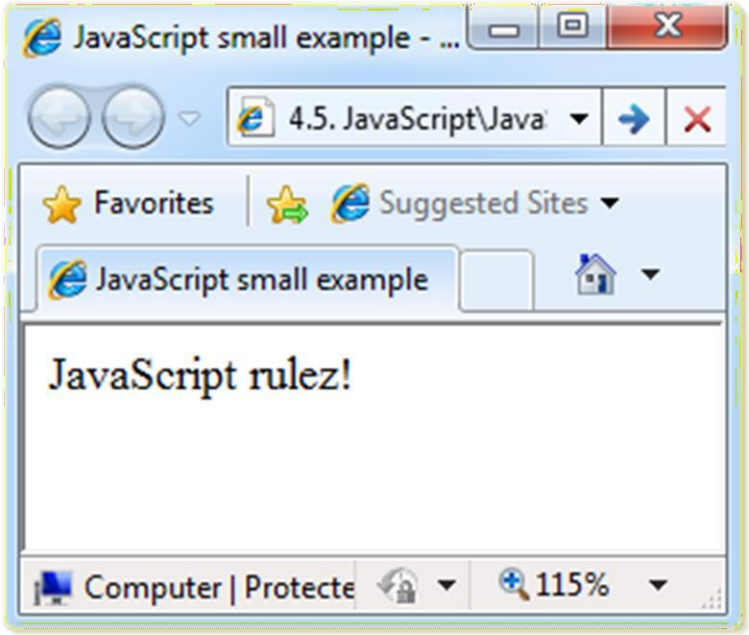
first-script.html

```
<html>

<body>
  <script type="text/javascript">
     alert('Hello JavaScript!');
  </script>
</body>

</html>
```



Example: document.write( )

```
<html>

<body>
  <script type="text/javascript">
     document.write('JavaScript rulez!');
  </script>
</body>

</html>
```



JavaScript Comments

- **Example:**

```
<html>
 <body>
<script type="text/javascript">
    // document.write("Hello World!")
document.write("Introducing JavaScript.....")
 </script>
</body>
 </html>
```

- **Output:** Introducing JavaScript…..

- *Comments are made in JavaScript by adding // before a single line, or /* before and */ after multiple lines.*

JavaScript in <body> section

```
<html>
 <head> </head>
 <body>
<script type="text/javascript">
    document.write("Hello World")
</script>
<p>This is web page body </p>
 </body>
</html>
```

- This code will produce the following results –

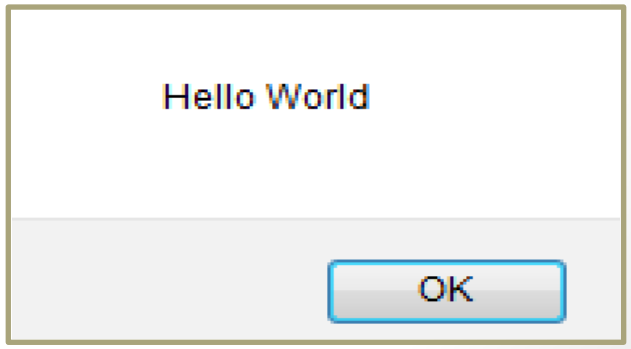
  Hello World

  This is web page body

JavaScript in <head> section

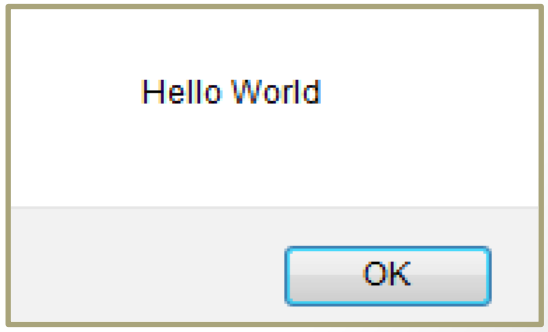
```
<html>
<head>
<script type="text/javascript">
 function sayHello( )
{
  alert("Hello World")
}
 </script>
</head>
<body>
<input type="button" value="Say Hello" onclick="sayHello( )" />
</body>
</html>
```

- This code will produce the following result –



JavaScript in <body> and <head>

```
<html>
<head>
<script type="text/javascript">
  function sayHello()
  { alert("Hello World") }
 </script>
</head>
<body>
<script type="text/javascript">
 document.write("Hello World")
</script>
<input type="button" value="Say Hello" onclick="sayHello()" />
</body> </html>
```



JavaScript in External File

- **HTML File**

```
<html>
<head>
 <script type="text/javascript" src="filename.js" >
</script>
</head>
 <body> ..... </body>
</html>
```

- **JavaScript File – filename.js**

```
function sayHello()
 {
   alert("Hello World")
 }
```

Using JavaScript files

- External files are linked via `<script>` tag in the <head>

Tag:

- Files have `.js` extension
- The `.js` files get cached by the browser

```
<script src="scripts.js" type="text/javscript">
   //code placed here will not be executed!
</script>
```

- **Do not use internal scripting along with external files in same `<script>` tag**

Vijayendra S. Gaikwad, PICT

JavaScript –When it is Executed ?

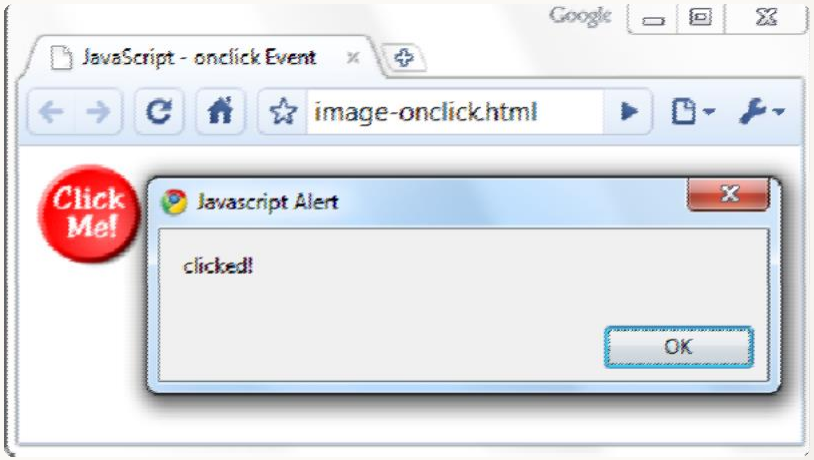
- **JavaScript code is executed during the page loading or when the browser fires an event.**
  - All statements are executed at the time of page loading.
  - Some statements just define functions that can be called later.

- Function calls or code can be attached as "event handlers" via different tag attributes
- Executed when the event is fired by the browser

```
<img src="logo.gif" onclick="alert('clicked!')" />
```

Calling a JavaScript Function from an Event Handler attribute

```
<html>
<head>
<script type="text/javascript">
  function test (message)
 {
   alert(message);
  }
</script>
</head>

<body>
  <img src="logo.gif"
    onclick="test('clicked!')" />
</body>
</html>
```



Using External Script Files

- Using external script files:

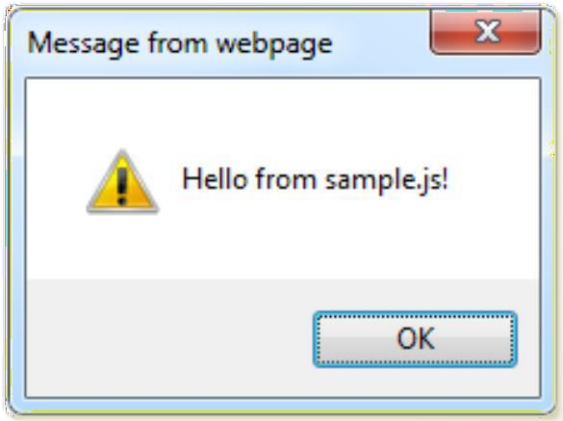
external-JavaScript.html

```
<html>
<head>
<script src="sample.js" type="text/javascript">
</script>
</head>
<body>
  <input type="button" onclick="sample()" value="Call
    JavaScript  function from sample.js" />
</body>
</html>
```

The <script> tag is always empty.

- External JavaScript file: `sample.js`

```
function sample() {
  alert('Hello from sample.js!')
}
```



JavaScript Syntax

- The JavaScript syntax is similar to Java and C#:
  - Operators (+, *, =, !=, &&, ++, …)
  - Variables (*typeless*)
  - Conditional statements (if, else)
  - Loops (for, while)
  - Arrays (my_array[])
  - Associative arrays (my_array['abc'])
  - Functions (*can return value*)

Data Types & Variables

- JavaScript data types:
  - Numbers (integer, floating-point)
  - Boolean (true / false)

- String type – string of characters

```
var myName = "You can use both single or
double  quotes for strings";
```

- Arrays

Everything is Object

- In JavaScript, every variable can be **considered as object**

For example **strings and arrays have member functions**:

```
var test = "some string";
alert(test[7]); // shows letter 'r'
alert(test.charAt(5)); // shows letter 's'
alert("test".charAt(1)); //shows letter 'e'
alert("test".substring(1,3)); //shows 'es'
```

```
var arr = [1,3,4];
alert (arr.length); // shows 3
arr.push(7); // appends 7 to end of array
alert (arr[3]); // shows 7
```


Vijayendra S. Gaikwad, PICT

Arrays Operations and Properties

- Declaring new empty array:

```
var arr = new Array();
```

- Declaring an array holding few elements:

```
var arr = [1, 2, 3, 4, 5];
```

- Appending an element & getting the last element:

```
arr.push(3);
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

- Finding any element's index in the array:

```
arr.indexOf(1);
```


Vijayendra S. Gaikwad, PICT

JavaScript Variables

```
<script type="text/javascript">
var name = "Anand";
var money;
money = 2000.50;
</script>
```

Variables are Objects

JavaScript variables usually contain single values:

let person = "John Doe";

JavaScript variables can also contain many values:

let person = {firstName: "John", lastName: "Doe", age: 50, eyeColor: "blue"};

4 Ways to Declare a JavaScript Variable:

- Using ***var***
- Using ***let***
- Using ***const***

When to Use JavaScript var?

- Always declare JavaScript variables with **var, let** or **const**.
- The ***var*** keyword is used in all JavaScript code from 1995 to 2015.
- The ***let*** and ***const*** keywords were added to JavaScript in 2015.

- If you want your code to run in older browser, you must use ***var***.

String Operations

- The **+** operator is used to join strings:

```
string1 = "fat ";
string2 = "cats";
alert(string1 + string2);  // fat cats
```

- What is "9" + 9?

```
alert("9" + 9);  // 99
```

- Converting string to number:

```
alert(parseInt("9") + 9);  // 18
```

Standard Popup Boxes

- **Alert box with text and [OK] button**
  - Just a message shown in a dialog box:

```
alert("Some text here");
```

- **Confirmation box**
  - Contains text, [OK] button and [Cancel] button:

```
confirm("Are you sure?");
```

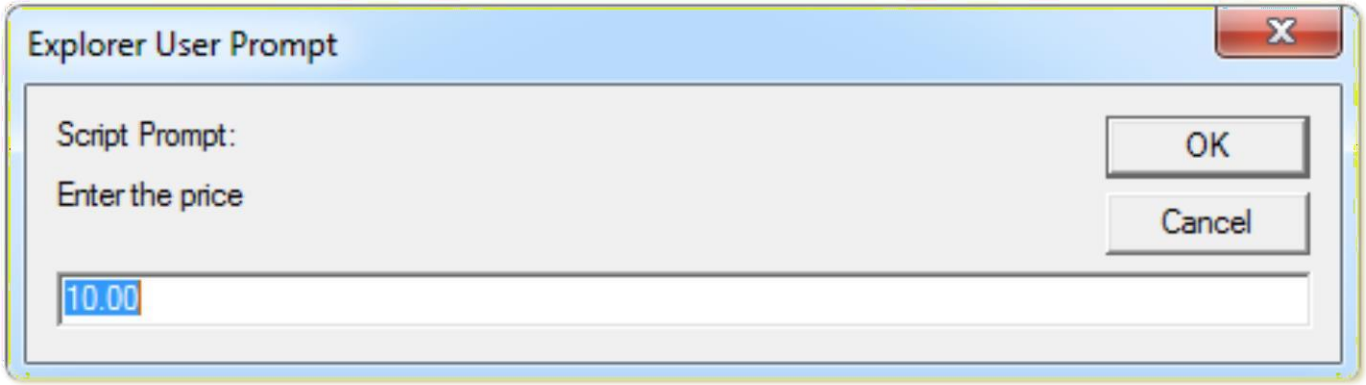
- **Prompt box**
  - Contains text and an input field with some default value:

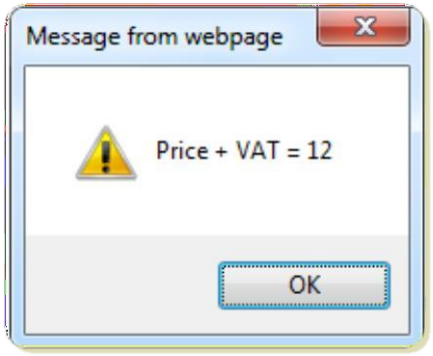
```
prompt ("enter amount", 10);
```


Vijayendra S. Gaikwad, PICT

JavaScript Prompt – Example

```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.2);
```





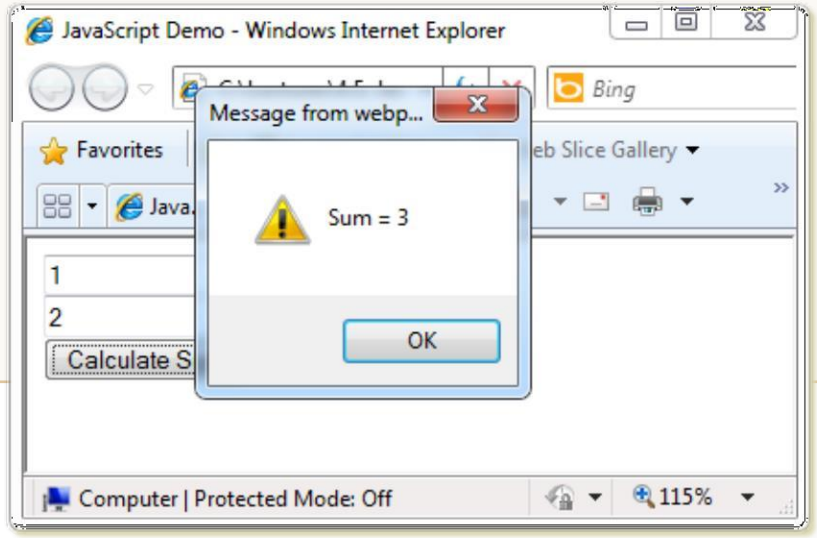
Sum of Numbers – Example

```
<html>

<head>
  <title>JavaScript Demo</title>
     <script type="text/javascript">
       function calcSum() {
        value1 =
            parseInt(document.mainForm.textBox1.value);
        value2 =
            parseInt(document.mainForm.textBox2.value);
        sum = value1 + value2;
        alert('Sum='+ sum);
            }
     </script>
</head>
```


Vijayendra S. Gaikwad, PICT

sum-of-numbers.html (cont.)

```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Calculate Sum"
      onclick="javascript:calcSum()" />
  </form>
</body>

</html>
```



JavaScript - Operators

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Symbol	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal
!=	Not equal

Conditional Statement (if)

```
unitPrice = 1.30;
if (quantity > 100) {
   unitPrice = 1.20;
}
```

Switch Statement

- The **`switch`** statement:

```
switch (variable) {
  case 1:
    // do something
    break;
  case 'a':
    // do something else
    break;
  case 3.14:
    // another code
    break;
  default:
   // something completely different
}
```

Switch case Example

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var grade='A';
            document.write("Entering switch block<br />");
            switch (grade)
            {
               case 'A': document.write("Good job<br />");
               break;

               case 'B': document.write("Pretty good<br />");
               break;

               case 'C': document.write("Passed<br />");
               break;

               case 'D': document.write("Not so good<br />");
               break;

               case 'F': document.write("Failed<br />");
               break;

               default:  document.write("Unknown grade<br />")
            }
            document.write("Exiting switch block");
         //-->
      </script>

      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Loops

- for loop
- while loop
- do … while loop

```
var counter;
for (counter=0; counter<4; counter++)
{
  alert(counter);
}

while (counter < 5)
{
  alert(++counter);
}
```


Vijayendra S. Gaikwad, PICT

While-loop Example

```
<html>
<body>
<script type="text/javascript">
var count = 0;
document.write("Starting Loop…. ");
while (count < 10) {
document.write("Current Count : " + count + "<br />");
   count++;
}
document.write("Loop stopped!");
</script>
</body> </html>
```

Functions: Code structure – splitting code into parts

- Data comes in, processed, result returned
- A JavaScript function is defined with the **function** keyword

```
function average(a, b, c)
{
    var total;
    total = a+b+c;
    return total/3;
}
```

Parameters come in here.

Declaring variables is optional. Type is never declared.

Value returned here.

JavaScript Function Syntax

- function *name*(*parameter1, parameter2, parameter3*)
  {
  code to be executed
  }

// Function is called, return value will be stored in variable *x*

- var x = myFunction(4, 3);
  document.getElementById("demoID").innerHTML = x;

  function myFunction(a, b)
  {
  return a * b; // Function returns the product of a & b
  }

Function Arguments & Return Value

- Functions are not required to return a value
- While declaring functions, it is not obligatory to specify all of its arguments
- The function has access to all the arguments passed to it via the **`arguments`** array

```
function sum() {
  var sum = 0;
  for (var i = 0; i < arguments.length; i ++)
    sum += parseInt(arguments[i]);
  return sum;
}
alert(sum(1, 2, 4));
```

Function Invocation

•The code inside the function will execute when "**something**" **invokes** (calls) the function:

-When an event occurs (when a user clicks a button)
-When it is explicitly called from JavaScript code
- Automatically (self invoked):

```
(function () {
        var x = "Hello!!";
})();
```

After the function has been initialized, it will be immediately invoked and executed.

For example, if we have a web page in which we want to **attach event listeners to DOM elements** and other initialization work, then self-invoking functions would be the best tool to make this arrangement when the page loads at first

Vijayendra S. Gaikwad , PICT

JavaScript Debugging

- All modern browsers have a built-in JavaScript debugger.
- We can use ***console.log()*** to display JavaScript values for debugging :

```
<!DOCTYPE html>
<html>
<body>
<p>Activate debugging in your browser (Chrome, IE, Firefox) with F12, and select "Console" in the debugger menu.</p>
<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>
</body>
</html>
```


Vijayendra S. Gaikwad , PICT

The debugger keyword

- The debugger keyword stops the execution of JavaScript and calls (if available) the debugging function.
- This has the same function as setting a breakpoint in the debugger.
- If no debugging is available, the debugger statement has no effect.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo" > </p>
<p>With the debugger turned on, the code below should stop executing before it executes the third line.</p>
<script>
let x = 15 * 5;
debugger;
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```


Vijayendra S. Gaikwad , PICT

Outline

JavaScript: Overview of JavaScript, using JS in an HTML (Embedded, External), Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS, JavaScript debugger

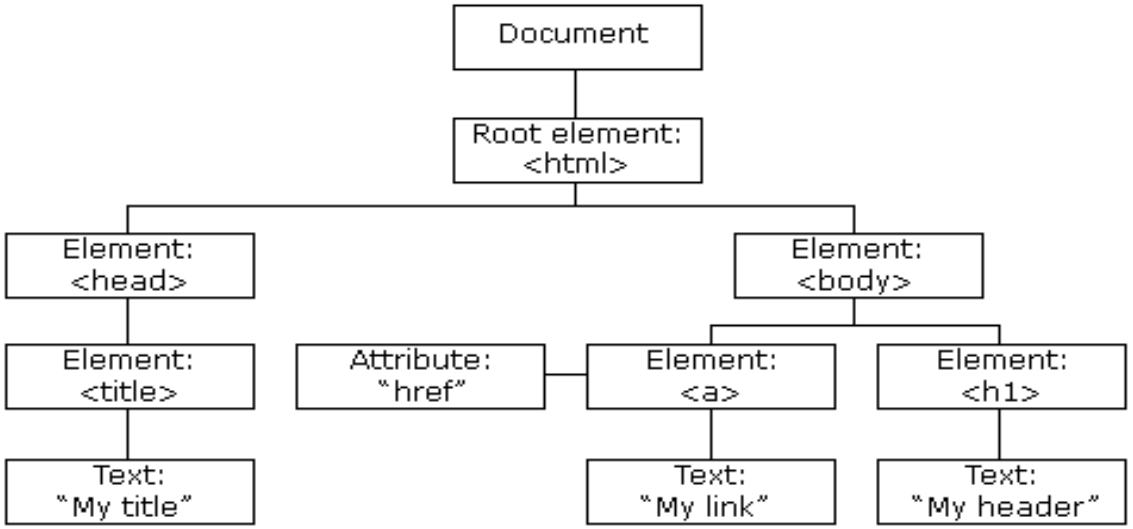
DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM,

JQuery: Introduction to JQuery, Loading JQuery, Selecting elements, changing styles, creating elements, appending elements, removing elements, handling events.

DOM- Document Object Model

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects** (i.e. HTML tags):
- Nested tags are called children of enclosing tags

The HTML DOM Tree of Objects



Vijayendra S. Gaikwad , PICT

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard for accessing documents:
- *"The Document Object Model (DOM) is a platform and **language-neutral interface** that allows programs and scripts to **dynamically access and update the content, structure and style of a document**."*

- The W3C DOM standard is separated into 3 different parts:
  1. Core DOM - standard model for all document types
  2. XML DOM - standard model for XML documents
  3. HTML DOM - standard model for HTML documents

DOM Objects and their properties and methods

- In the DOM, all HTML elements are defined as **objects**.
- The programming interface consists of the properties and methods associated with each DOM object.
- HTML DOM **properties are attributes** of HTML elements that we can set or change.
- HTML DOM **methods are actions** (like add or deleting an HTML element) we can perform on HTML elements.

Vijayendra S. Gaikwad , PICT

Benefits of DOM to JavaScript

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
  - JavaScript **can change all the HTML elements** in the page
  - JavaScript **can change all the HTML elements' attributes** in the page
  - JavaScript can change all the **CSS styles** in the page
  - JavaScript **can remove existing HTML elements** and **attributes**
  - JavaScript **can add new HTML elements and attributes**
  - JavaScript can **react to all existing HTML events** in the page
  - JavaScript can **create new HTML events** in the page

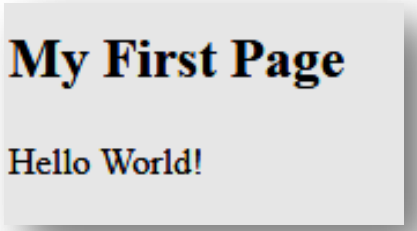
DOM Example 1

- The following example changes the content (i.e. innerHTML property) of the element with id="demo" :

- **Example**
- <html>
  <body>
  <h1>My First Page</h1>

  <script>
  document.getElementById("demo").innerHTML = "Hello World!";
  </script>
  </body>
  </html>

Output



- In the example above, getElementById () is a **method**, while innerHTML is a **property/ attribute**.

Vijayendra S. Gaikwad , PICT

getElementById & innerHTML

- The **getElementById** Method:
- The most common way to access an HTML element is to use the *id* attribute of the element.
- In the example, the getElementById method used id="demo" to find the **target element**.

- The **innerHTML** Property:
- The easiest way for **getting or replacing the content of an element** is by using the **innerHTML** property.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = *new_html_content*	Change the inner HTML (content) of an Element
element.attribute = new_value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new_style*	Change the style of an HTML element

Vijayendra S. Gaikwad , PICT

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Event Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler & its code to an ***onclick*** event

Changing HTML Content

- **Example**

```
<html> <body>
<h2>JavaScript can Change HTML</h2>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
<p>The paragraph above was changed by a script.</p>
</body> </html>
```

- **Output**

JavaScript can Change HTML

New text!

The paragraph above was changed by a script.

Vijayendra S. Gaikwad , PICT

Changing the Value of an Attribute

- **Example**

```
<html><body>
<img id="image" src="smiley.gif" width="160" height="120">
<script>
document.getElementById("image").src = "landscape.jpg";
</script>
<p>The original image was smiley.gif, but the script changed it to landscape.jpg
</p>
</body></html>
```

- **Output**

The original image was smiley.gif, but the script changed it to landscape.jpg

, PICT

Sample DOM manipulations using JavaScript

- Access and change text content of an HTML element
- Access and change the text color & background color of text content of an HTML element:

```
document.getElementsByTagName("LI")[1].style.color="red";
```

- Access element by class:

```
<div class="example"> First div element with class as "example".</div>

var x = document.getElementsByClassName("example");
 x[0].innerHTML = "Hello World!";+
```

- Access the URI of HTML document:

```
var x = document.documentURI;
document.getElementById("demo").innerHTML = x;
```

continued..

- Create an element through javascript:

```
var btn = document.createElement("BUTTON");
 btn.innerHTML = "CLICK ME";
 document.body.appendChild(btn);
```

- Create an attribute for any element:

```
 var li= document.getElementsByTagName("LI")[0];
 var attri = document.createAttribute("class");
 attri.value = "democlass";
 li.setAttributeNode(attri);
```

```
<style>
.democlass {
 color: red;
}
</style>
```

continued…

Create the *onmouseover* attribute to simulate hover effect:

```
<p id="demo">Hover over this paragraph to see the change...</p>

<script>
(function() {

  var p= document.getElementById("demo");
  var attri= document.createAttribute("onmouseover");
  attri.value="myFunction()";
  p.setAttributeNode(attri);

})();

function myFunction()
{
  document.getElementById("demo").innerHTML="See..!! I changed....";
}
</script>
```

continued…

Give focus to an element:

```
<input type="text" id= "textfield1" >

document.getElementById(“textfield1").focus();
```

Assignment 2:

Use *document.images* and *document.scripts* and print the sources of the images and external scripts used in an HTML document. (dynamically create an unordered list of these sources)

Outline

JavaScript: Overview of JavaScript, using JS in an HTML (Embedded, External), Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS,

DOM: DOM levels, DOM Objects and their properties and methods, modifying element style, the document tree, DOM event handling

JQuery: Introduction to JQuery, Loading JQuery, Selecting elements, changing styles, creating elements, appending elements, removing elements, handling events, AngularJS overview.

jQuery - Introduction

- jQuery is a JavaScript Library.
- jQuery greatly simplifies JavaScript programming.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
- The jQuery library contains the following features:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX

Vijayendra S. Gaikwad , PICT

Adding jQuery to YourWeb Pages

- Following are the ways to start using jQuery on your Website:
  - Download the jQuery library from jQuery.com
  - Include jQuery from a CDN, like GoogleCDN

Downloading jQuery

- There are two versions of jQuery available for downloading:
  - Production version - this is **for live website** because it has been minified and compressed
  - Development version - this is **for testing and development**
- Both versions can be downloaded from jQuery.com.
- The jQuery library is a **single JavaScript file** and we provide reference to it within the HTML <script> tag

  (note that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

- **Tip:** Place the downloaded file in the same directory as the pages where you wish to use it.

Vijayendra S. Gaikwad , PICT

jQuery CDN

- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- Both Google and Microsoft are hosting jQuery.
- jQuery from Google or Microsoft can be used in the following:
- **Google CDN:**

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
</head>
```

jQuery Syntax

- **With jQuery you select (find and query) the HTML elements and perform certain "actions" on them.**
- **Syntax :**
  - **$(*selector*).*action*()**
- $ sign is to define/access jQuery library
- (*selector*) is to query or find the target HTML elements
- *action*() specifies any jQuery action to be performed on the selected element(s)
- **Examples:**
  - $(this).hide() - hides the current element.
  - $("p").hide() - hides all <p> elements.
  - $(".test").hide() - hides all elements with class="test".
  - $("#test").hide() - hides the element with id="test".

jQuery Selectors- for selecting the target elements

- jQuery selectors allow you to select and further manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) the target HTML elements **based on their name, id, class type, attributes, values of attributes** and much more.
- It's based on the existing **CSS selectors** and in addition, it has some own custom selectors.
- All selectors in jQuery start with the dollar sign and are included in parentheses: **$().**

Vijayendra S. Gaikwad , PICT

Vijayendra S. Gaikwad , PICT

The element Selector

- This jQuery element selector **selects elements based on the HTML element's name.**
- With element selector you can select all <p> elements on a page like this:

```
$("p")
```

- **Example:**

  When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){
   $("button").click(function(){
  $("p").hide();
});
});
```

The #id Selector

- The jQuery #id selector **uses the id attribute of an HTML tag to find the specific HTML element**.
- An *id* is unique at least within a page, so you should use the #id selector when you want to find a single, unique HTML element.
- To find an element with a specific id, write a hash character *(#)*, followed by the *id* of the HTML element:

```
$("#test")
```

- **Example**
- When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```


Vijayendra S. Gaikwad , PICT

jQuery css() Method-for changing Style

jQuery css() Method :

- The css() method sets or returns one or more style properties for the selected HTML elements.

Returns a CSS Property

- To return the value of a specified CSS property, use the following syntax:
  - css("*propertyname*");

Sets a CSS Property

- To set a specified CSS property, use the following syntax:
  - css("*propertyname*","*value*");

Vijayendra S. Gaikwad , PICT

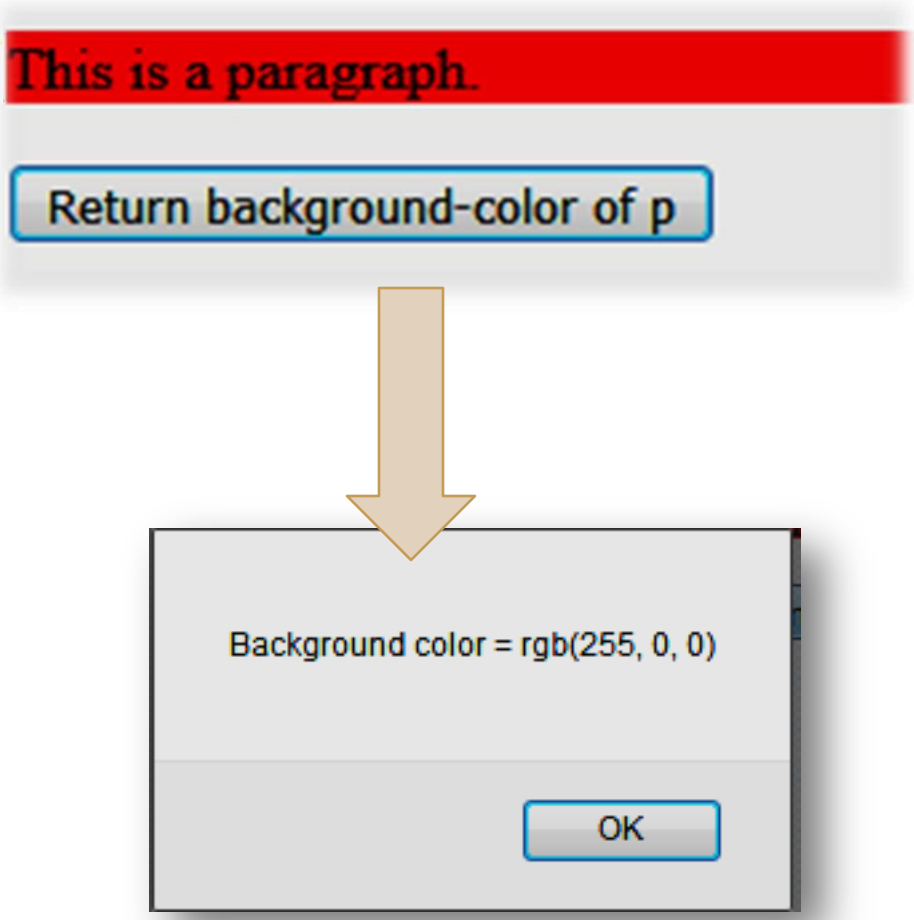
Changing Style-

Example to Return a CSS Property

```
<html><head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
<script>
$(document).ready(function(){
   $("button").click(function(){
      alert("Background color is- " + $("p").css("background-color"));
   });
});
</script>
</head><body>
<p style="background-color:#ff0000">This is a paragraph.</p>
<button>Return background-color of p</button>
</body></html>
```

Changing Style-
Return a CSS Property Example o/p

Output of Previous Code

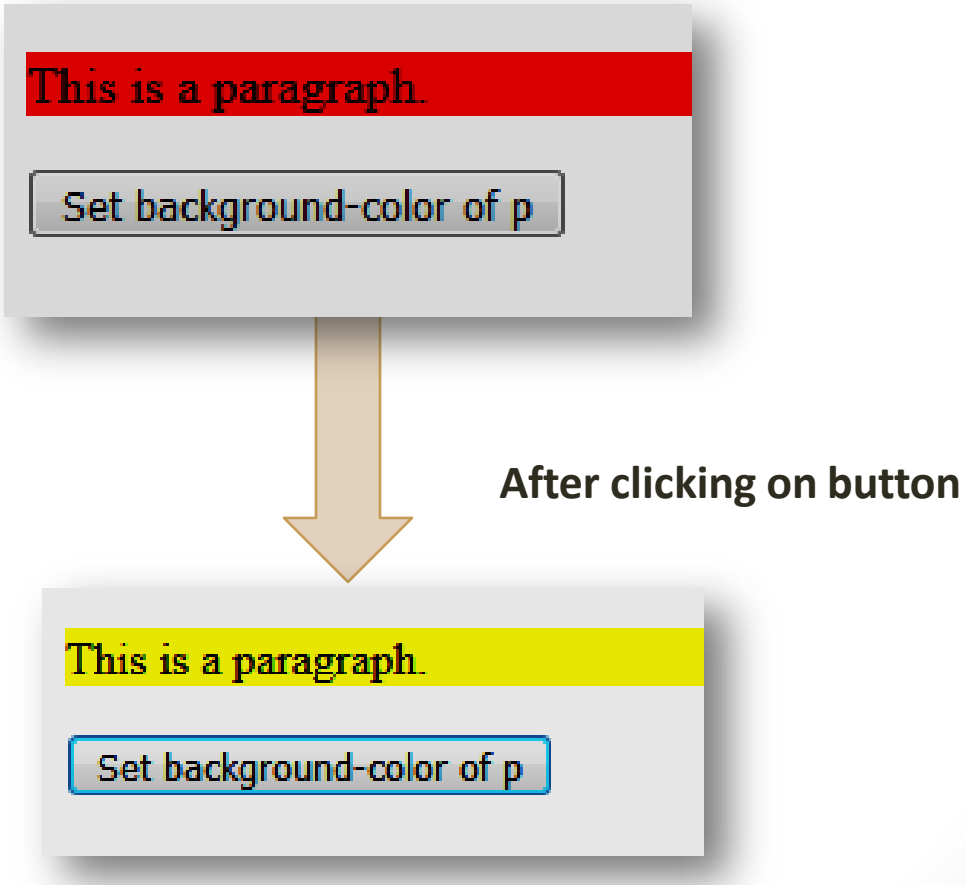


Changing Style-
Set a CSS Property Example

```
<html><head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").css("background-color", "yellow");
  });
});
</script>
</head>
<body>
<p style="background-color:#ff0000">This is a paragraph.</p>
<button>Set background-color of p</button>
</body></html>
```

Changing Style-
Set a CSS Property Example o/p

Output of Previous Code

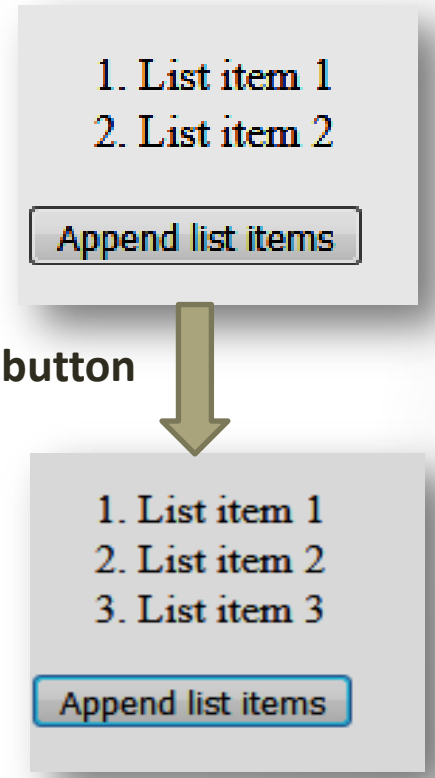


jQuery –Add Elements

- The jQuery methods that are used to add new content:
  - **append() -** Inserts content at the end of the selected elements
  - **prepend() -** Inserts content at the beginning of the selected elements
  - **after() -** Inserts content after the selected element
  - **before() -** Inserts content before the selected element

jQuery append( ) Method- Example

```
<html><head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
   $("#btn1").click(function(){
      $("ol").append("<li> New List item  </li>");
   }); });
   </script>
</head>
<body>
<ol>
  <li>List item 1</li>
  <li>List item 2</li>
</ol>
<button id="btn1"> Append list items </button>
</body></html>
```



Vijayendra S. Gaikwad , PICT

jQuery - Remove Elements

Remove Elements/Content :

- To remove elements and content, there are mainly two jQuery methods:
  - **remove() -** Removes the selected element, its contents and also all the child elements
  - **empty() –** Removes the contents from the selected element & also the child elements (but the selected element remains)

jQuery remove( ) Method- Example

```
<html > <head>
<script src="https://code.jquery.com/jquery-1.10.2.js">
   </script>
</head>
<body>
<p> Hello <button>I will disappear..</button></p>
<br>
<button id="b1">Remove </button>
<script>
$( " #b1" ).click(function() {
$( "p" ).remove();
});
</script>
</body></html>
```

jQuery Quiz link:

https://forms.office.com/r/JntqnShZHq