Web Technology
Unit II- Client Side Technologies

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, Manipulating DOM,

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

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, Manipulating DOM,

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

JavaScript- Outline

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

Overview of JavaScript,

- JavaScript is one of the **3 languages** all web developers **must** 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 very powerful **client-side scripting language**.
- Javascript is a dynamic computer programming language.
-

JavaScript

- JavaScript is a front-end scripting language developed by Netscape for dynamic content
  - Lightweight, but with limited capabilities
  - Can be used as object-oriented language
- Client-side technology
  - Embedded in your HTML page
  - Interpreted by the Web browser
- Simple and flexible
- Powerful to manipulate the DOM

JavaScript Advantages

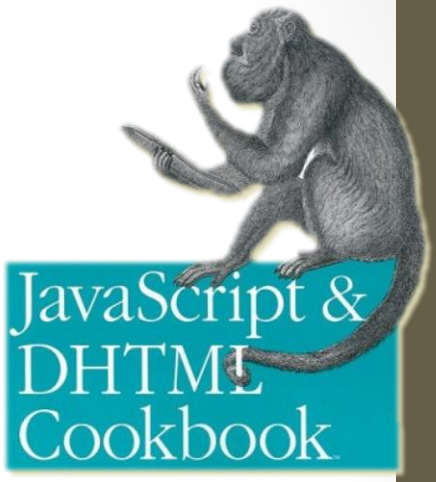
- JavaScript allows interactivity such as:
  - Implementing form validation
  - React to user actions, e.g. handle keys
  - Changing an image on moving mouse over it
  - Sections of a page appearing and disappearing
  - Content loading and changing dynamically
  - Performing complex calculations
  - 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

JavaScript Syntax

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

- **Syntax:**
  - <script language="javascript" type="text/javascript">
  - JavaScript code
  - </script>

JavaScript Syntax

- **Example:**

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

- The comment ends with a "//-->". Here "//" signifies a comment in JavaScript
- **Output**

  Hello World!

JavaScript Syntax

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

Enabling JavaScript in Browsers

- **JavaScript in Firefox**
- Open a new tab → type **about: config** in the address bar.
- Then you will find the warning dialog.
- Select **I'll be careful, I promise!**
- Then you will find the list of **configure options** in the browser.
- In the search bar, type **javascript.enabled**.
- There you will find the option to enable or disable javascript by right-clicking on the value of that option → **select toggle**.

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 code in brower

JavaScript - Placement in HTML File

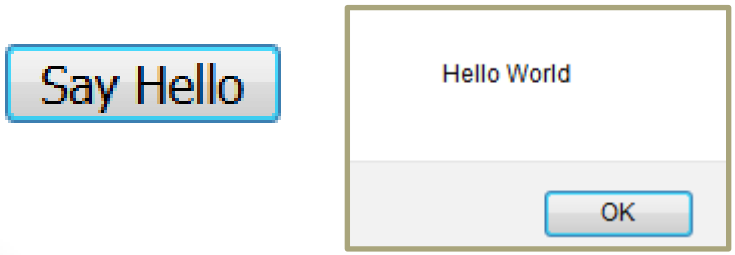
- There is a flexibility given to include JavaScript code anywhere in an HTML document.
- However the most preferred ways to include JavaScript in an HTML file are as follows −

  - Script in <head>...</head> section.
  - Script in <body>...</body> section.
  - Script in <body>...</body> and <head>...</head> sections.
  - Script in an external file and then include in <head>...</head> section.

JavaScript in <head>...</head> section

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

- **This code will produce the following results –**



JavaScript in <body>...</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 <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" onclick="sayHello()" value="Say Hello" />
</body> </html>
```

- This code will produce the following result –

JavaScript in External File

- **HTML File**

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

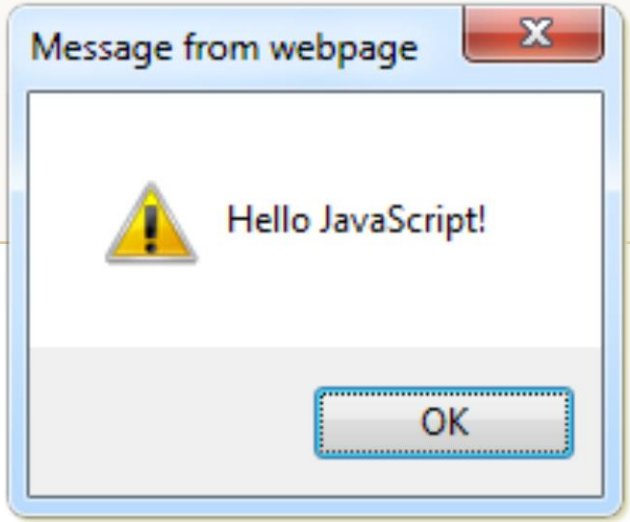
- **JavaScript File – filename.js**

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

The First Script

first-script.html

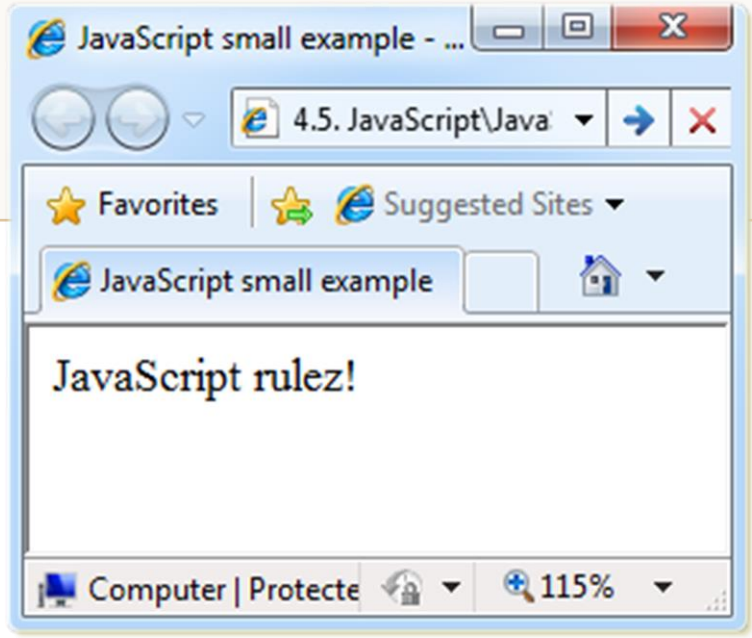
```
<html>

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

</html>
```



Another Small Example

small-example.html

```
<html>

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

</html>
```



Using JavaScript Code

- The JavaScript code can be placed in:
  - `<script>` tag in the head
  - `<script>` tag in the body – not recommended
  - External files, linked via `<script>` tag the head
    - Files usually have `.js` extension
    - Highly recommended
    - The `.js` files get cached by the browser

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

JavaScript – When is Executed?

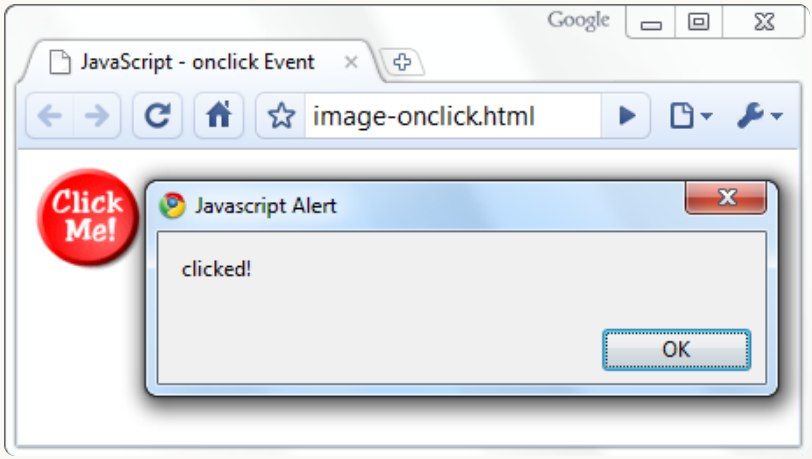
- JavaScript code is executed during the page loading or when the browser fires an event
  - All statements are executed at page loading
  - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
  - Executed when the event is fired by the browser

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

Calling a JavaScript Function from Event Handler – Example

image-onclick.html

```
<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

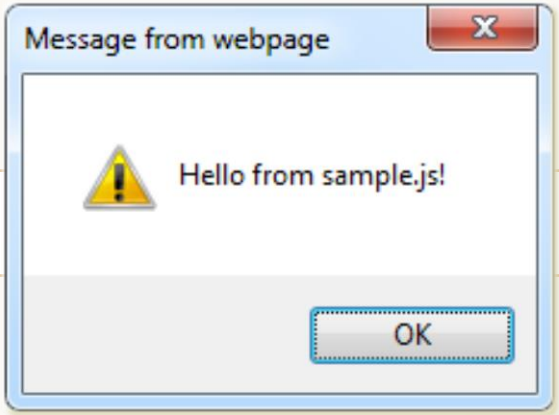
```
<html>
<head>
  <script src="sample.js" type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

The <script> tag is always empty.

- External JavaScript file:

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

sample.js



Data Types

- 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

```
var my_array = [1, 5.3, "aaa"];
```

- Associative arrays (hash tables)

```
var my_hash = {a:2, b:3, c:"text"};
```

Everything is Object

- Every variable can be considered as object
  - For example strings and arrays have member functions:

objects.html

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

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

String Operations

- The + operator joins 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
```

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 element's index in the array:

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

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, input field with default value:

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

JavaScript Variables

```
<script type="text/javascript">
<!--
 var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

Sum of Numbers – Example

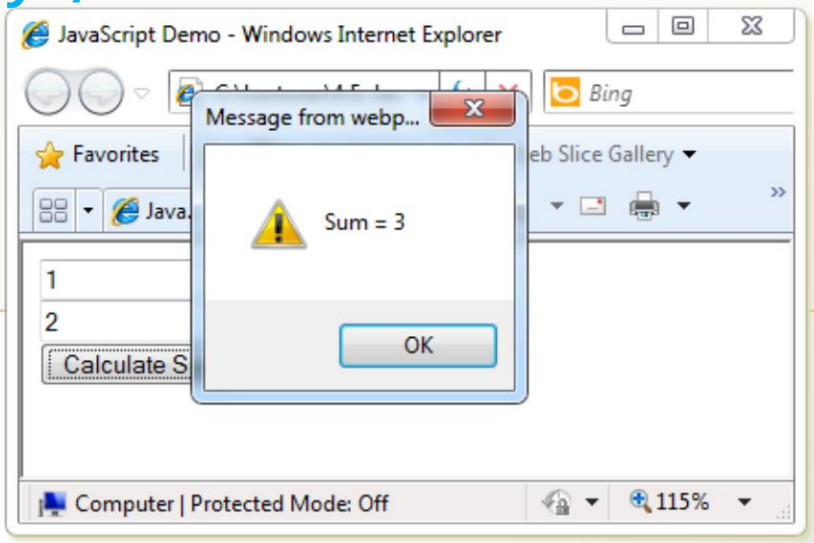
sum-of-numbers.html

```
<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;
      document.mainForm.textBoxSum.value = sum;
    }
  </script>
</head>
```

Sum of Numbers – Example (2)

sum-of-numbers.html (cont.)

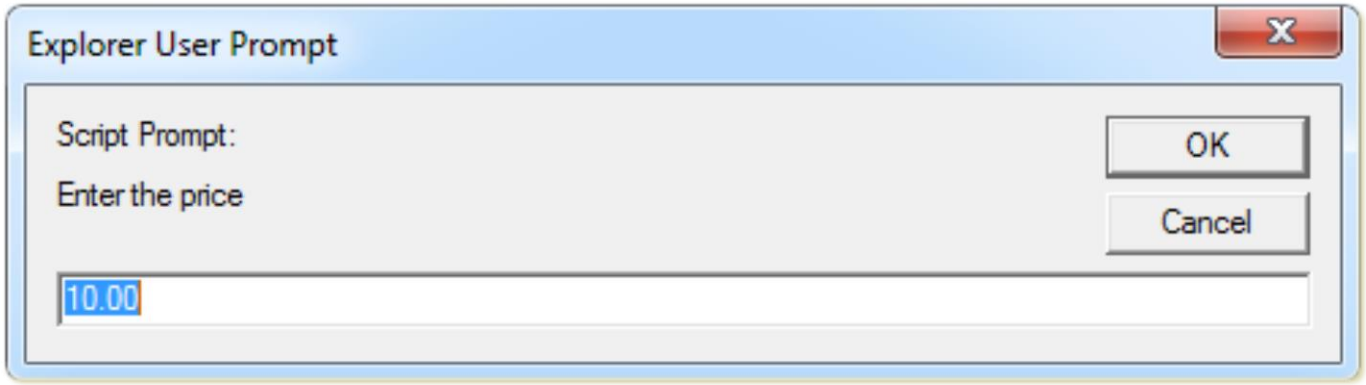
```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript: calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>

</html>
```

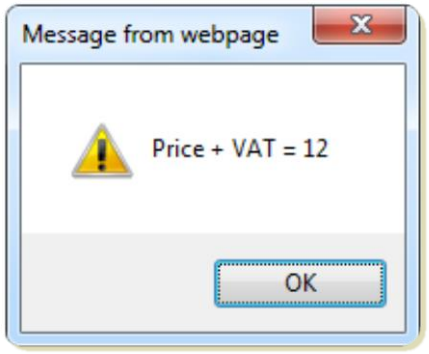


JavaScript Prompt – Example

prompt.html

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





JavaScript - Operators

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

Conditional Statement (if)

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

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

Switch Statement

- The **`switch`** statement works like in C#:

switch-statements.html

```
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

- Like in C#
  - `for` loop
  - `while` loop
  - `do … while` loop

```
var counter;
for (counter=0; counter<4; counter++)
{
  alert(counter);
}
while (counter < 5)
{
  alert(++counter);
}
```

loops.html

While-loop Example

- <html>
- <body>
- <script type="text/javascript">
- <!-- var count = 0;
- document.write("Starting Loop ");
- while (count < 10){
- document.write("Current Count : " + count + "
"); count++; }
- document.write("Loop stopped!");
- //--> </script>
- Set the variable to different value and then try... </body> </html>

Functions

- Code structure – splitting code into parts
- Data comes in, processed, result returned

```
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.

Function Arguments & Return Value

- Functions are not required to return a value
- When calling function it is not obligatory to specify all of its arguments
  - The function has access to all the arguments passed via `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));
```

functions-demo.html

JavaScript Function Syntax

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

- var x = myFunction(4, 3); // Function is called, return value will end up in x

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

Outline

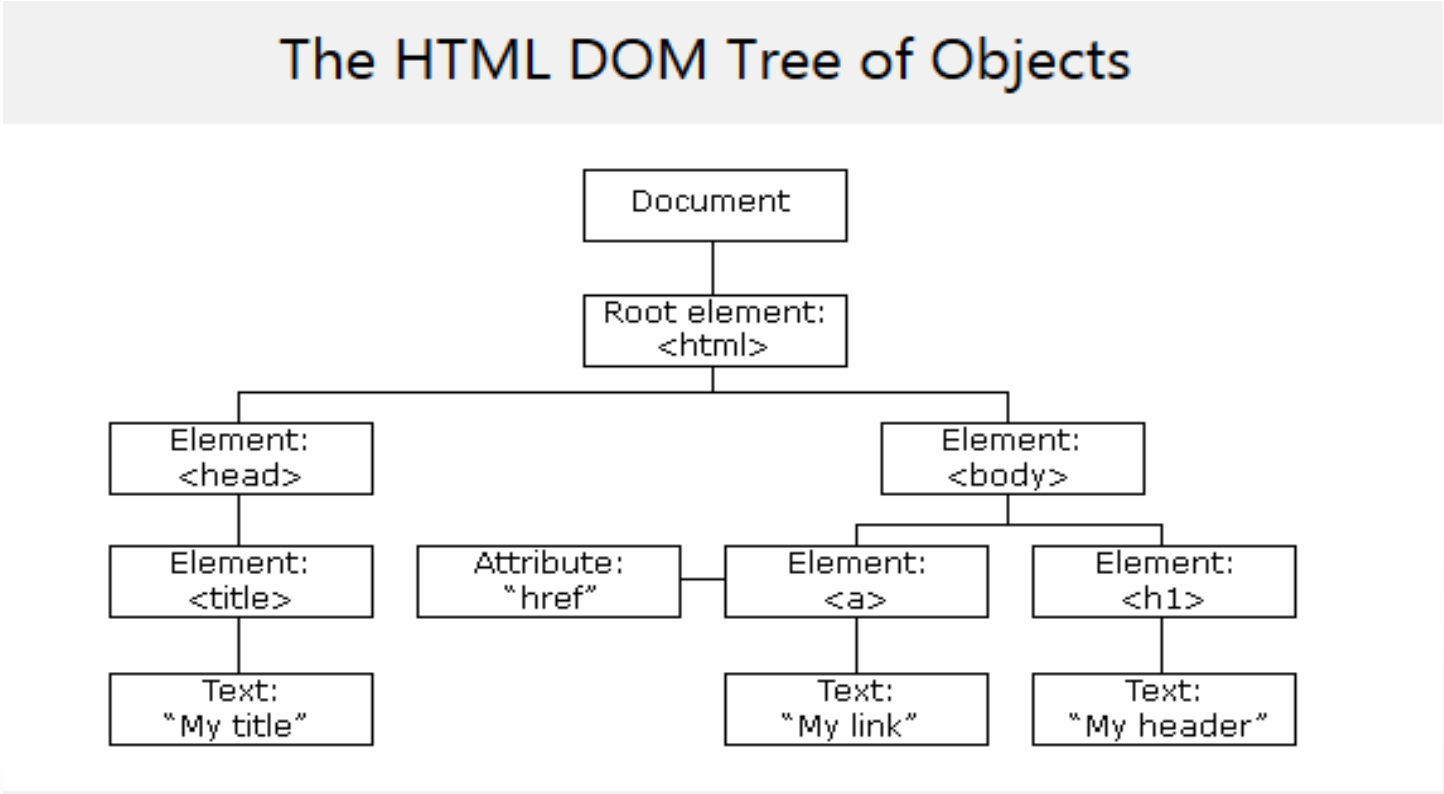
JavaScript: Overview of JavaScript, using JS in an HTML (Embedded, External), Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS,

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

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

DOM- **D**ocument **O**bject **M**ode

- 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**:



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 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

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- *"The W3C 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 Levels

- DOM Level 1
- DOM Level 2
- DOM Level 3

- For More details regarding DOM levels click on below site. https://developer.mozilla.org/fr/docs/DOM_Levels

DOM Level 1

- The DOM Level 1 specification is separated into two parts:
  1. **Core and**
  2. **HTML.**
- Core Level 1 provides a low-level set of fundamental interfaces that can represent any structured document, as well as defining extended interfaces for representing an XML document.
- HTML Level 1 provides additional, higher-level interfaces that are used with the fundamental interfaces defined in Core Level 1 to provide a more convenient view of an HTML document. Interfaces introduced in DOM1 include, among others, the Document, Node, Attr, Element, and Text interfaces.

DOM Level 2

- The DOM Level 2 specification contains six different specifications:
- The DOM2 Core,
- Views,
- Events,
- Style,
- Traversal and Range, and
- the DOM2 HTML. Most of the DOM Level 2 is supported in Mozilla.

DOM Level 2

Specification	Description
DOM Level 2 Core	helps the programs and scripts to access and update the content and structure of document dynamically
DOM Level 2 Views	allows programs and scripts to dynamically access and update the content of an HTML or XML document
DOM Level 2 Events	provides a generic event system to programs and scripts
DOM Level 2 Style	allows programs and scripts to dynamically access and update the content of style sheets
DOM Level 2 Traversal and Range	allows programs and scripts to dynamically traverse and identify a range of content in a document
DOM Level 2 HTML	allows programs and scripts to dynamically access and update the content and structure of HTML 4.01 and XHTML 1.0 documents

DOM Level 3

- The DOM Level 3 specification contains five different specifications:
- The DOM3 Core,
- Load and Save,
- Validation,
- Events, and
- XPath.

DOM Level 3

Specification	Description
DOM Level 3 Core	allows programs and scripts to dynamically access and update the content, structure, and document style
DOM Level 3 Load and Save	allows programs and scripts to dynamically load the content of an XML document into a DOM document
DOM Level 3 Validation	allows programs and scripts to dynamically update the content and structure of documents and ensures that the document remains valid

DOM Objects and their properties and methods

- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- HTML DOM **properties** are **values** (of HTML Elements) that you can set or change.
- HTML DOM methods are **actions** you can perform (on HTML Elements). A **method** is an action you can do (like add or deleting an HTML element).

DOM Example 1

- The following example changes the content (the innerHTML) of the <p> element with id="demo":
- **Example**
-

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

Output

- In the example above, getElementById is a **method**, while innerHTML is a **property**.

getElementById & innerHTML

- **The getElementById Method**
- The most common way to access an HTML element is to use the id of the element.
- In the example above the getElementById method used id="demo" to find the element.
- **The innerHTML Property**
- The easiest way to get the content of an element is by using the **innerHTML** property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

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 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

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 Events Handlers

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

Finding HTML Objects

Property	Description	DOM
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1

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.

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

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, Manipulating DOM,

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

jQuery - Introduction

- jQuery is a JavaScript Library.
- jQuery greatly simplifies JavaScript programming.
- jQuery is a lightweight
- 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
  - Utilities

Adding jQuery to Your Web Pages

- There are several ways to start using jQuery on your web site. You can:
  - Download the jQuery library from jQuery.com
  - Include jQuery from a CDN, like Google

Downloading jQuery

- There are two versions of jQuery available for downloading:
  - Production version - this is for your 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 you reference it with the HTML <script> tag (notice 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.

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 host jQuery.
- To use jQuery from Google or Microsoft, use one of 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 (query) HTML elements and perform "actions" on them.
- **syntax :**
  - **$(*selector*).*action*()**
- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the 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-Selecting elements

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, 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 parentheses: $().

The element Selector

- The jQuery element selector selects elements based on the element name.
- 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 element.
- An id should be unique within a page, so you should use the #id selector when you want to find a single, unique 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();
    });
});
```

jQuery css() Method- Changing Style

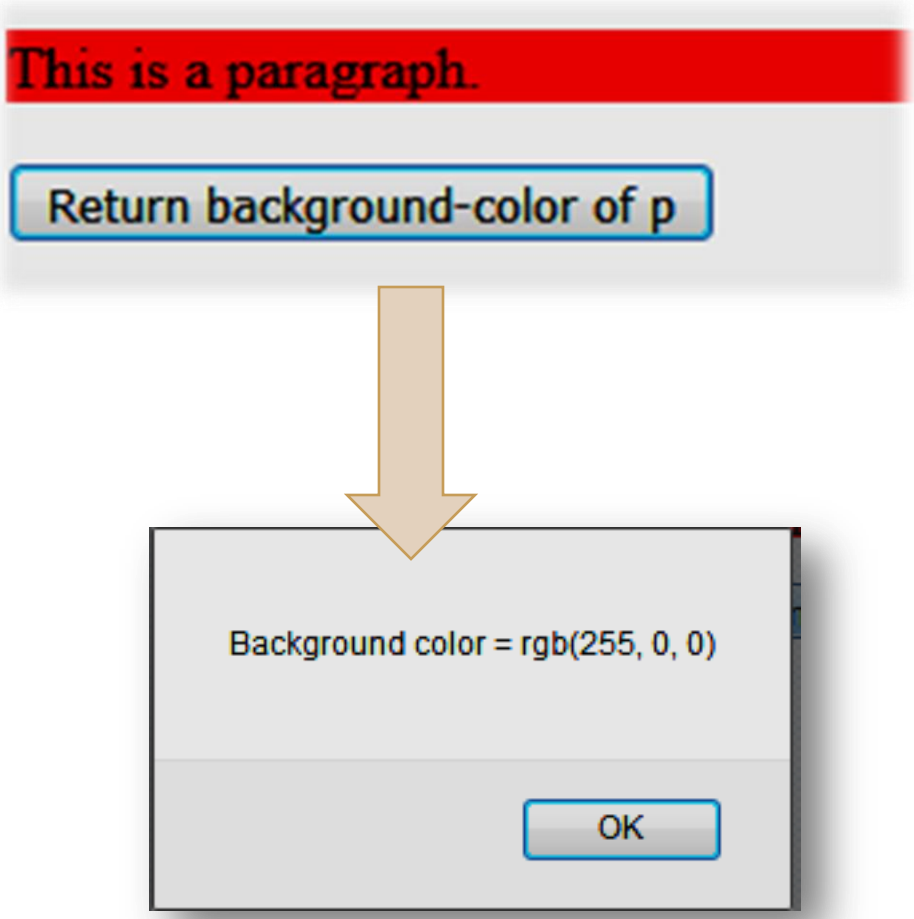
- **jQuery css() Method**
- The css() method sets or returns one or more style properties for the selected elements.
- **Return a CSS Property**
- To return the value of a specified CSS property, use the following syntax:
  - css("*propertyname*");
- **Set a CSS Property**
- To set a specified CSS property, use the following syntax:
  - css("*propertyname*","*value*");

Changing Style-
Return a CSS Property Example

```
<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 = " + $("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

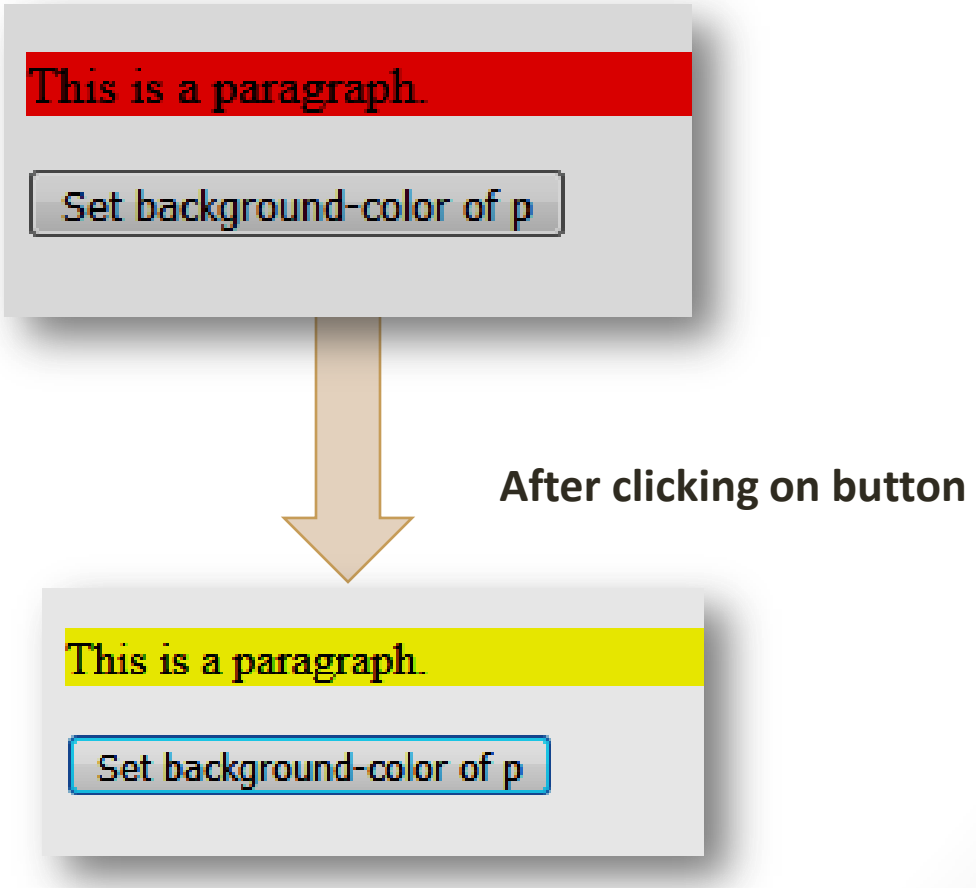


Changing Style-
Set a CSS Property Example

```
<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

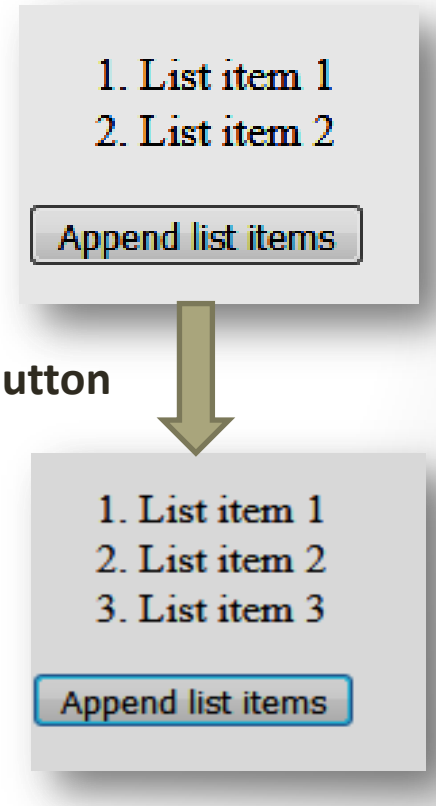


jQuery - Add Elements

- We will look at four 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 elements
  - **before() -** Inserts content before the selected elements

jQuery append() Method- Example

```
<html><head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></scrip
t>
<script>
$(document).ready(function(){
   $("#btn1").click(function(){
      $("ol").append("<li>  List item 3 </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>
```

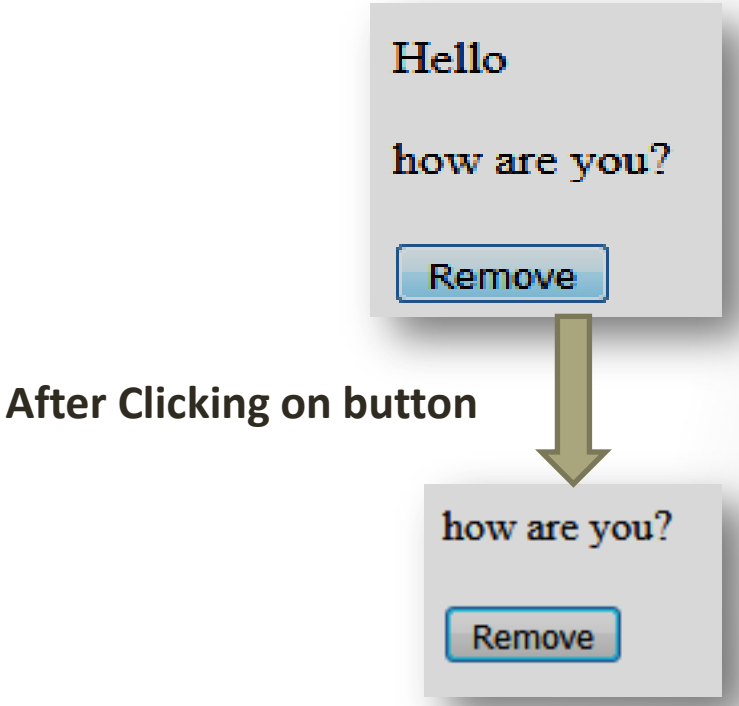


jQuery - Remove Elements

- **Remove Elements/Content**
- To remove elements and content, there are mainly two jQuery methods:
  - **remove() -** Removes the selected element (and its child elements)
  - **empty() -** Removes the child elements from the selected element

jQuery remove() Method- Example

- <html > <head>
- <script src="https://code.jquery.com/jquery-1.10.2.js"> </script>
- </head>
- <body>
- ** Hello **
- **how are you?**
- **<button>Remove </button>**
- **<script>**
- **$("button").click(function() {**
- **$("p").remove();**
- **});**
- **</script>**
- </body></html>



References

- *https://www.slideshare.net/rakhithota/js-ppt*
- *https://www.tutorialspoint.com/javascript/javascript_quick_guide.htm*
- *https://www.w3schools.com/js*
- *https://www.w3schools.com/js/js_htmldom.asp*
- https://developer.mozilla.org/fr/docs/DOM_Levels
- https://codescracker.com/js/js-dom-levels.htm
- https://www.w3schools.com/jquery/jquery_get_started.asp
- https://www.w3schools.com/jquery/jquery_css.asp
- https://www.w3schools.com/jquery/jquery_dom_add.asp