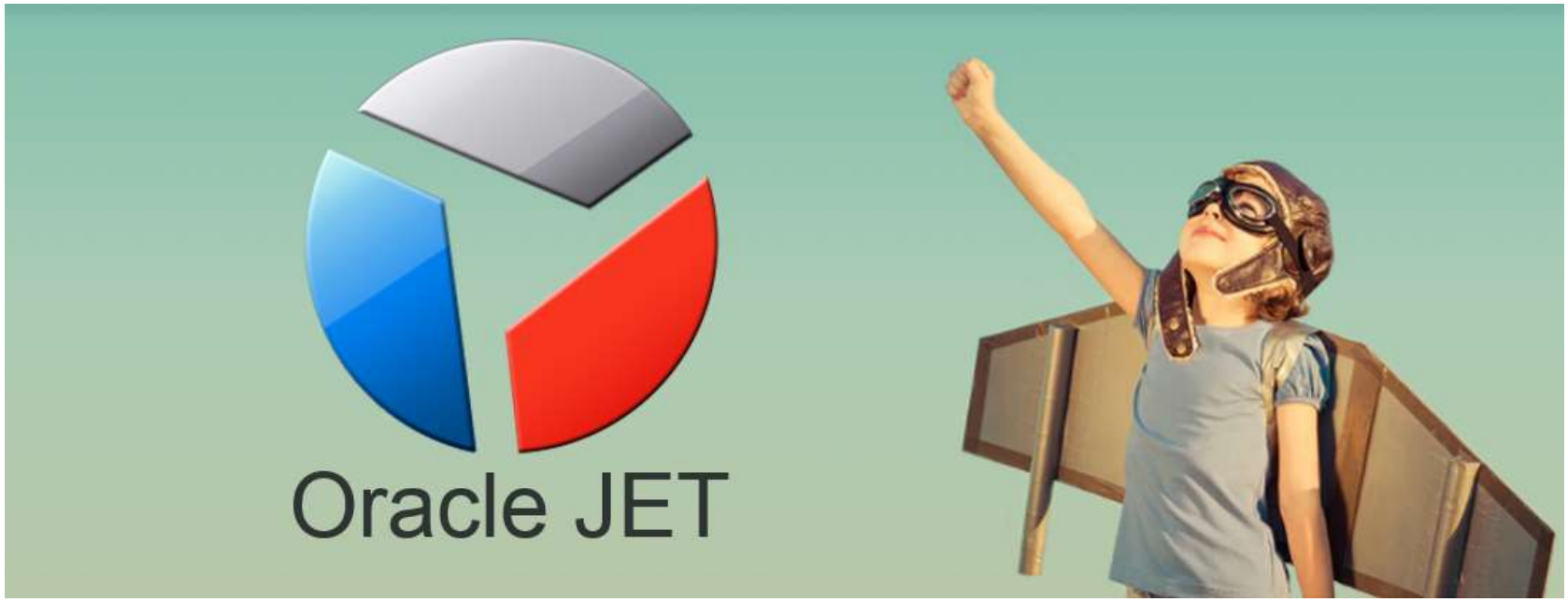


Welcome to the Live Virtual Class!

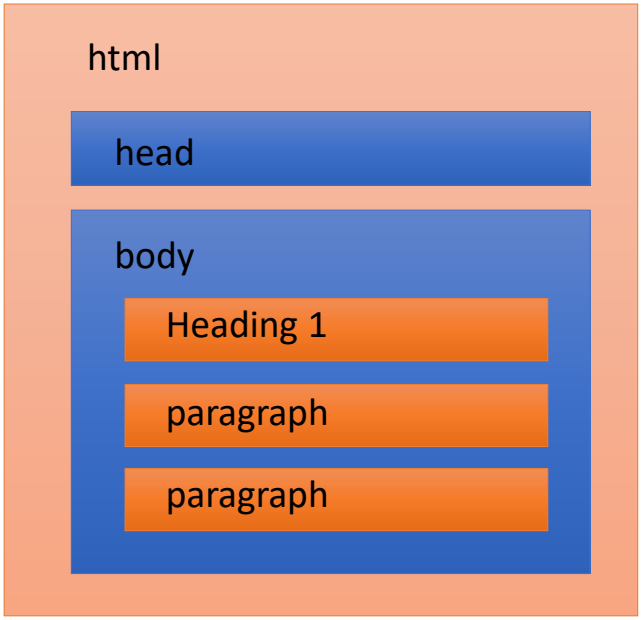
Please mention your background, experience (in HTML, CSS, JS, any other) and motivation for taking up this course in the chat box.

HTML

Introduction

- HTML is the standard markup language for creating Web pages.
- HTML stands for Hyper Text Markup Language
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements are represented by tags
- Browsers use tags to render the content of the page

Structure




```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Hyper Text Mark-up Language</title>
</head>

<body>
    <h1>Learning HTML</h1>
    <p style="color:red">Build your website using HTML</p>
    <p style="color:green">It's the standard mark-up language for web pages</p>
    <p style="color:blue">You will enjoy it!</p>
</body>

</html>
```

Tags

Attribute

Value

<h1> </h1>

Opening and Closing Tags

DOM

- The Document Object Model (DOM) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document.
- The DOM represents a document with a logical tree
- It’s important to be aware of this representation while we study the HTML

Views

Learning HTML

Build your website using HTML

It's the standard mark-up langauge for web pages

You will enjoy it!

Rendered View

```
DOCTYPE: html
HTML lang="en"
  HEAD
    #text:
    TITLE
      #text: Hyper Text Mark-up Language
    #text:
  #text:
  BODY
    #text:
    H1
      #text: Learning HTML
    #text:
    P style="color:red"
      #text: Build your website using HTML
    #text:
    P style="color:green"
      #text: It's the standard mark-up langauge for web pages
    #text:
    P style="color:blue"
      #text: You will enjoy it!
    #text:
```

DOM View

https://software.hixie.ch/utilities/js/live-dom-viewer/
Or install DOM inspection plug-ins for your browser

CSS3

Cascading Style Sheets - Importance

- Let's do a simple experiment…
- Add **pendule**/**web developer toolbar** extensions to your Chrome browser before you start the experiment
- Load www.facebook.com or any other website of your choice
- Try disabling all the CSS styles from the web page you are viewing
- What did you learn??

CSS describes how the HTML elements should be displayed

03_tables.html visit this

Fundamentals

- CSS Syntax
- CSS Comment

```
/* This is a comment */
```

- CSS selectors are used to select the HTML elements to be styled

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all elements and all <p> elements

Ways to Add Styles

- Inline CSS
  - Can be used to apply a unique style to a single element

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Ways to Add Styles

- Internal CSS
  - Can be used if one single HTML page has a unique style

```
<!DOCTYPE html>
<html>
<head>
    <style>

    body {
          background-
    color: linen;
    }
    h1 {
          color: maroon;
          margin-left: 40px;
    }

    </style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Way to Add Styles

- An eternal .css file is maintained
- Every HTML file should reference it

```html
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

style.css

```css
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
```

JavaScript

Javascript Practice

- https://www.w3schools.com/js/js_exercises.asp

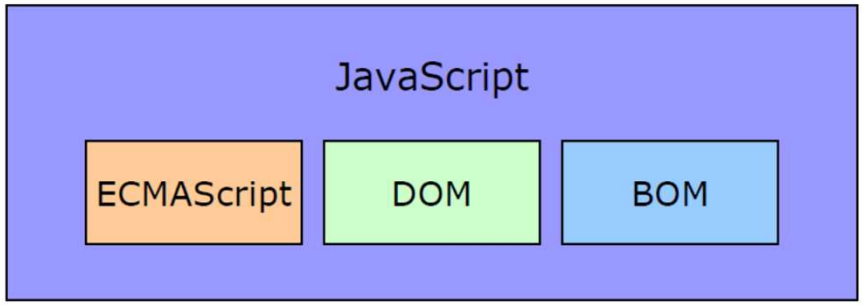
Javascript

- JavaScript is the programming language for HTML and the Web.
- Can be written with in the HTML code in <head> and <body>
- The best practice is to write it in a separate file.
- Placing Javascript in external files has the following advantages:
  - It separates HTML and code
  - It makes HTML and JavaScript easier to read and maintain
  - Cached JavaScript files can speed up page loads
- Reference to external Javascript file

```
<script src="script.js"></script>
```

Javascript Implementations

- The complete Javascript implementations are made of three part



ECMA Script

- It is simple a description, defining all properties, methods and objects of a scripting language
  - Syntax
  - Types
  - Statements
  - Keywords
  - Reserved Words
  - Operators
  - Objects
- Each browser has its own implementation of the ECMA Script interface, which is then extended to contain DOM and BOM

DOM

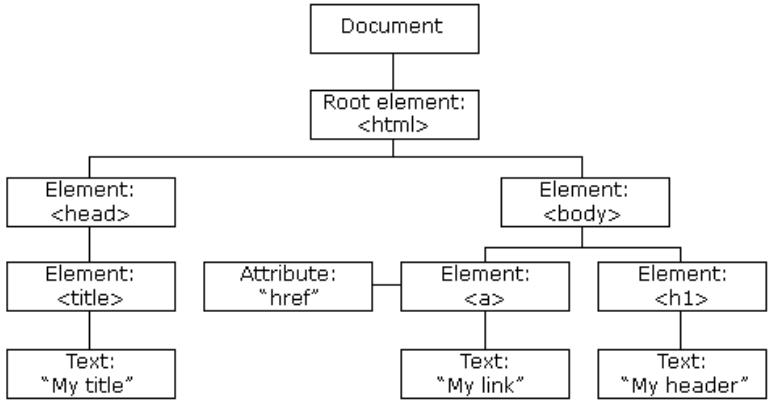
- The DOM describes methods and interfaces for working with the content of the web page
- DOM is a tree based, language independent API for HTML and XML
- Document object is the only object that belongs to both DOM and BOM
- Some functions defined are:
  - `getElementsByTagName(), getElementsByName(), getElementById()`
- All attributes are included in HTML DOM elements as properties
  - `oImg.src = “picture.jpg”;`
  - `oDiv.className = “footer”;  // cf.class → className`

BOM

- The BOM describes methods and interfaces for interacting with the browser
- Each browser has its own implementations
- The window object represents the entire browser window:
  - Objects
    - Document : anchors, forms, images, links, location
    - Frames, history, navigator, screen
  - Methods
    - `moveBy(), moveTo(), resizeBy(), resizeTo()`
    - `open(), close(), alert(), confirm(), input()`
    - `setTimeOut(), clearTimeOut(), setInterval(), clearInterval()`
  - Properties
    - `screenX, screenY, status, defaultStatus, etc`

HTML DOM

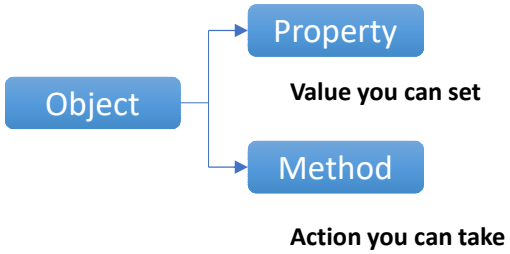
- The browser creates the Document Object Model when the webpage loads
- It is created as a tree of objects
- With the DOM, JS can do the following:
  - Change all the HTML elements in the page
  - Change all the HTML attributes in the page
  - Change all the CSS styles in the page
  - Remove existing HTML elements and attributes
  - Add new HTML elements and attributes
  - React to all existing HTML events in the page
  - Create new HTML events in the page



The HTML DOM is a standard for how to get, change, add, or delete HTML elements

HTML DOM

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements




```
<html>
<body>

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

</body>
</html>
```

Document Object

- Document Object represents the webpage
- All objects are accessed through the document object
- Use the `document` keyword to access the document object

https://www.w3schools.com/js/js_htmldom_document.asp

Finding HTML Elements

- To manipulate HTML elements, first it should be accessed
- Finding HTML elements by id

```
var myElement = document.getElementById("intro");
```

- Finding HTML elements by tag name

```
var x = document.getElementsByTagName("p");
```

- Finding HTML elements by class name

```
var x = document.getElementsByClassName("intro");
```

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

Finding HTML Elements

- Finding HTML elements by CSS selectors

```
var x = document.querySelectorAll("p.intro");
```

- Finding HTML elements by HTML object collections

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Finds the form element id = ‘frm1’

Manipulating HTML Content

- Use **innerHTML** property to modify the content of an HTML element

```
document.getElementById("p1").innerHTML = "Game of Thrones";

var element = document.getElementById("id01");
element.innerHTML = "Game of Thrones";
```

- HTML attributes can be changed using the **attribute**

```
<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
```

Manipulating CSS Content

- CSS content can be modified using the style property

```
document.getElementById("p2").style.color = "blue";
```

Demo and Labs

Events

- Javascript DOM functions can be used to react to HTML events

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

displayDate is JS function

- Other events:
  - **onload**, **onunload**
  - **onchange**
  - **onmouseover**, **onmouseout**
  - **onmousedown**, **onmouseup**

Event Listener

- The **`addEventListener()`** method attaches an event handler to the specified element
- Does not override existing event handlers
- Many even handlers can be added
- The event handler can be removed using **`removeEventListener()`**

```
<button id="myBtn">Try it</button>

<script>
document.getElementById("myBtn").addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

Any DOM event
https://www.w3schools.com/jsref/dom_obj_event.asp

Navigation

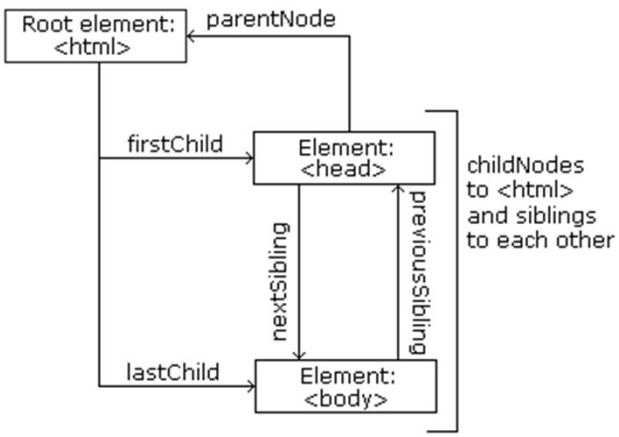
- The DOM tree can be navigated using the node relationships
- The nodes in the node tree have a hierarchical relationship
  - The terms parent, child, and sibling are used to describe the relationships.
  - In a node tree, the top node is called the root (or root node)
  - Every node has exactly one parent, except the root (which has no parent)
  - A node can have a number of children
  - Siblings (brothers or sisters) are nodes with the same parent

```
<html>

  <head>
    <title id=‘demo’>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



Navigating Between Nodes

- The following node properties can be used to navigate between nodes with JavaScript:
  - **parentNode**
  - **childNodes[nodenumber]**
  - **firstChild**
  - **lastChild**
  - **nextSibling**
  - **previousSibling**

```
var myTitle = document.getElementById("demo").innerHTML;
var myTitle = document.getElementById("demo").firstChild.nodeValue;
var myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

All these access **DOM Tutorial** in text tag

Navigating Between Nodes

- The full document can be accessed using the following properties
  - **`document.body`** - The body of the document
  - **`document.documentElement`** - The full document
- The node related properties are as follows
  - **`nodeValue`** property specifies the value of a node
  - **`nodeType`** property is read only. It returns the type of a node
    ELEMENT_NODE, TEXT_NODE, DOCUMENT_NODE, COMMENT_NODE …
  - **`nodeName`** property specifies the name of a node
    Same as tag name, attribute name, #text, #document respectively

Creating and Removing Nodes

- To add a new element you need to create the element node and then use **appendChild()** or **insertBefore()** to add the element

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

Creating Nodes

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(para, child);
</script>
```

Removing a Child Node

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

Replacing Nodes

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p"); // <p></p>
var node = document.createTextNode("This is new."); // This is new
para.appendChild(node); // <p>This is new</p>



var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para, child);
</script>
```

Collections and Node Lists

- An **HTMLCollection** object is an array-like list of HTML elements
- The **length** property defines the number of elements

```
var myCollection = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML = myCollection.length;
```

- A **NodeList** object is a list of nodes extracted from a document

```
var myNodelist = document.querySelectorAll("p");
var i;
for (i = 0; i < myNodelist.length; i++) {
  myNodelist[i].style.backgroundColor = "red";
}
```

Returns a node list

Demo

jQuery

https://api.jquery.com/
https://www.tutorialsteacher.com/jquery/jquery-tutorials
https://www.tutorialrepublic.com/jquery-examples.php

https://www.educba.com/javascript-vs-jquery/

Please, stop torturing yourself with **document.getElementByWhatever** and start using **$('[CSS_SELECTOR]').doSomething()**

jQuery

- jQuery is a JavaScript Library
- It's light weight and works with "**write less, do more**" philosophy
- It contains following features:
  - **Unobtrusive** HTML/DOM, CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities

jQuery

- Basic Syntax: **$(*selector*).*action*()**
  - **Example:** `$(this).hide()` – hides the current element
  - `$("p").hide()` – hides all <p> element

- Including jQuery in the project
  - Download and refer to it

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

  - Use the CDN

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

JS Vs jQuery

JavaScript

```
var myElement = document.getElementById("id01");

var myElements = document.getElementsByTagName("p");

var myElements = document.getElementsByClassName("intro");

var myElements = document.querySelectorAll("p.intro");
```

jQuery

```
var myElement = $("#id01");

var myElements = $("p");

var myElements = $(".intro");

var myElements = $("p.intro");
```

jQuery for HTML Manipulation

Setting Values

```
$("#test1").text("Hello world!");
$("#test2").html("<b>Hello world!</b>");
$("#test3").val("Hello World");
$("#w").attr("href", "/jquery")
```

Getting Values

```
var x = $("#test1").text();
var x = $("#test2").html();
var x = $("#test3").val();
var x = $("#w").attr("href")
```

```
<p><a href="" title="" id="w">jQuery</a></p>
```

```
$("button").click(function(){
  $("#w").attr({
    "href" : "/jquery/",
    "title" : "jQuery Tutorial"
  });
});
```

The jQuery ready()

- The .ready() method offers a way to run JavaScript code as soon as the page's Document Object Model (DOM) becomes safe to manipulate.
- This will often be a good time to perform tasks that are needed before the user views or interacts with the page, for example to add event handlers and initialize plugins.

```
$( document ).ready(function() {
    // Handler for .ready() called.
  });
```

Adding Elements

- Adding of elements can be done using
  - **append()** and **prepend()**
  - **before()** and **after()**

```
function appendText() {
  var txt1 = "<p>Text.</p>";                // Create element with HTML
  var txt2 = $("<p></p>").text("Text.");    // Create with jQuery
  var txt3 = document.createElement("p");   // Create with DOM
  txt3.innerHTML = "Text.";
  $("body").append(txt1, txt2, txt3);       // Append the new elements
}
```

Removing Elements

- The empty() method removes the child elements

```
$("#div1").empty();
```

- The remove() filter the elements to be removed

```
$("p").remove(".test, .demo");
```

jQuery for CSS Manipulation

- jQuery has several methods for CSS manipulation. We will look at the following methods:
  - `addClass()` - Adds one or more classes to the selected elements
  - `removeClass()` - Removes one or more classes from the selected elements
  - `toggleClass()` - Toggles between adding/removing
  - `css()` - Sets or returns the style attribute

```html
<div id="div1">This is some text.</div>
<div id="div2">This is some text.</div>
```

```style
<style>
.important {
  font-weight: bold;
  font-size: xx-large;
}

.blue {
  color: blue;
}
</style>
```

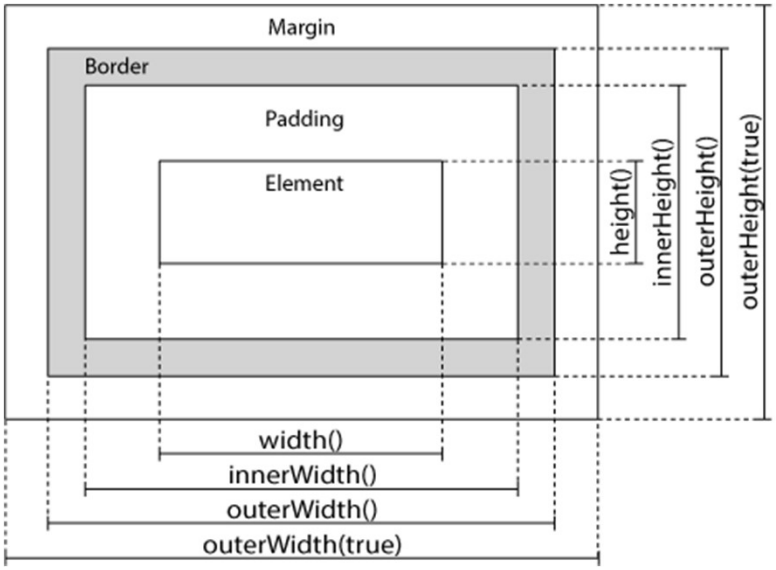
```script
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").addClass("important blue");
  });
});
</script>
```

Using css()

```
$("p").css("background-color", "yellow");
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

jQuery Dimensions

- jQuery has several important methods for working with dimensions:
  - `width()`
  - `height()`
  - `innerWidth()`
  - `innerHeight()`
  - `outerWidth()`
  - `outerHeight()`



jQuery Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

The $(document).ready() method allows us to execute a function when the document is fully loaded.

```
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
```

jQuery Traversing

- Traversing allow you to move through the HTML elements

```
$("span").parent();
$("span").parents();
$("span").parents("ul");
$("span").parentsUntil("div");
```

```
$("div").children();
$("div").children("p.first");
$("div").find("*");
$("h2").siblings("p");
$("h2").next();
$("h2").nextAll();
$("h2").nextUntil("h6");
```

The prev(), prevAll() and prevUntil() work with reverse functionality

```
$("div").first();
$("div").last();
$("p").eq(1);
$("p").filter(".intro");
$("p").not(".intro");
```

```
<script>
$(document).ready(function(){
  $("h2").next().css({"color": "red", "border": "2px solid red"});
});
</script>
```

Demo and Labs