# UNIT 2

# JAVASCRIPT

JavaScript is the programming language of the Web.

1. **HTML** to display the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

It means that a web page need no longer be static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

## An introduction to JavaScript

- JavaScript adds **behavior** to the web page where the web page is capable of responding to actions by your visitors
- JavaScript is most commonly used as a **client side scripting** language. This means that JavaScript code is written into an HTML page.
- We can use javascript to change the appearance of a webpage after it has been loaded.

# JavaScript vs. Java

- JavaScript is *not* the same as Java.
- Although the names are much alike, JavaScript is primarily a scripting language for use within HTML pages, while Java is a real programming language that does quite different things from JavaScript.
- In addition Java is much harder to learn. It was developed by Sun for use in pretty much anything that needs some computing power.
- JavaScript was developed by Brendan Eich, then working at Netscape, as a client side scripting language .

- Originally the language was called Live Script, but when it was about to the release,  Java had become immensely popular.
- At the last possible moment Netscape changed the name of its scripting language to “JavaScript”. This was done purely for marketing reasons. Worse, Eich was ordered to "make it look like Java". This has given rise to the idea that JavaScript is a "dumbed-down" version of Java

# Benefits of javascript

1.It is widely supported in web browsers

2. It gives easy access to the document objects and can manipulate most of them

3. JavaScript enhances Web pages with dynamic and interactive features.

4.could also allow the user (client) to input information and process it, might be used to validate input before it’s submitted to a remote server

4. JavaScript Improves appearance and enables form validation, calculations, special graphic and text effects, image swapping, image mapping, clocks, and more.

## The <script> Tag

- In HTML, JavaScript code must be inserted between <script> and </script> tags.

## JavaScript in <head> or <body>

- You can place any number of scripts in an HTML document.
- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

## External JavaScript

- Scripts can also be placed in external files.
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the **file extension .js**.
- To use an external script, put the name of the script file in the src (source) attribute of the <script> tag.

# Simple Script

**Example 1**

```
<html>
<head>
      <script language="javascript" >
      Document.write(“<p>javascript<p>”);
      </script>
</head>
   <body>
   </body>
</html>
```

**Output:** javascript

**Example 2 -Including external Js**

```
<html>
<head>
<script language="javascript"  src="sam.js">
</script>
</head>
<body>
</body>
</html>
```

## JavaScript Output

- JavaScript does NOT have any built-in print or display functions.

## JavaScript Display Possibilities

- JavaScript can "display" data in different ways:
- Writing into an alert box, using **window.alert()**.
- Writing into the HTML output using **document.write()**.
- Writing into an HTML element, using **innerHTML**.
- Writing into the browser console, using **console.log()**.

## using **window.alert()**

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
window.alert(5 + 6);
</script>
</body>
</html>
```

## Using document.write()

- For testing purposes, it is convenient to use **document.write()**:

```
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
document.write(5 + 6);
</script>
</body>
```

## Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

Example

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

## Example 1:

```
<html>
<head>
</head>
<body>
    <script language="javascript">
    var name;
    name= prompt("enter your name");
    document.writeln("<h1>welcome  </h1>"+name);
    document.close();
    </script>
</body>
</html>
```

**Using console.log()**

In your browser, you can use the **console.log()** method to display data.

Activate the browser console with F12, and select "Console" in the menu.

## JavaScript Variables

- In a programming language, **variables** are used to **store** data values.
- Technical name for variable is identifier
- JavaScript uses the **var** keyword to **declare** variables.
- An **equal sign** is used to **assign values** to variables.

**Declaring a variable:**

```
var myVariable;
```

**Assigning a value to a variable:**

```
Myvariable=10;
```

**Printing out a variable:**

```
document.write(myVariable); // no quotations
```

**initializing the variable:**

```
var myVariable=10;
```

## Datatypes

- Variables are used to store the data in computer memory.We can use different types of data.

**Integer**

- Is a whole number wdithout decimal places in it. can be positive or negative.

- E.g.: age cannot be in decimal places.

```
//integer

            var userage=21;
            //document.write(userage);
```

**Float** is basically a number that allowed to have decimal places. E.g. book price

```
// float        var bookprice=50.15;
            //document.write(bookprice);
```

- String: is jus  a collection of text. can be one word or it could be one sentence or it could even be a whole paragraph .string can contain spaces in between them,brackets and other symbols. with in double quotes.

```
 //string
        /*
        var username="saranya";
        document.write(username);*/
```

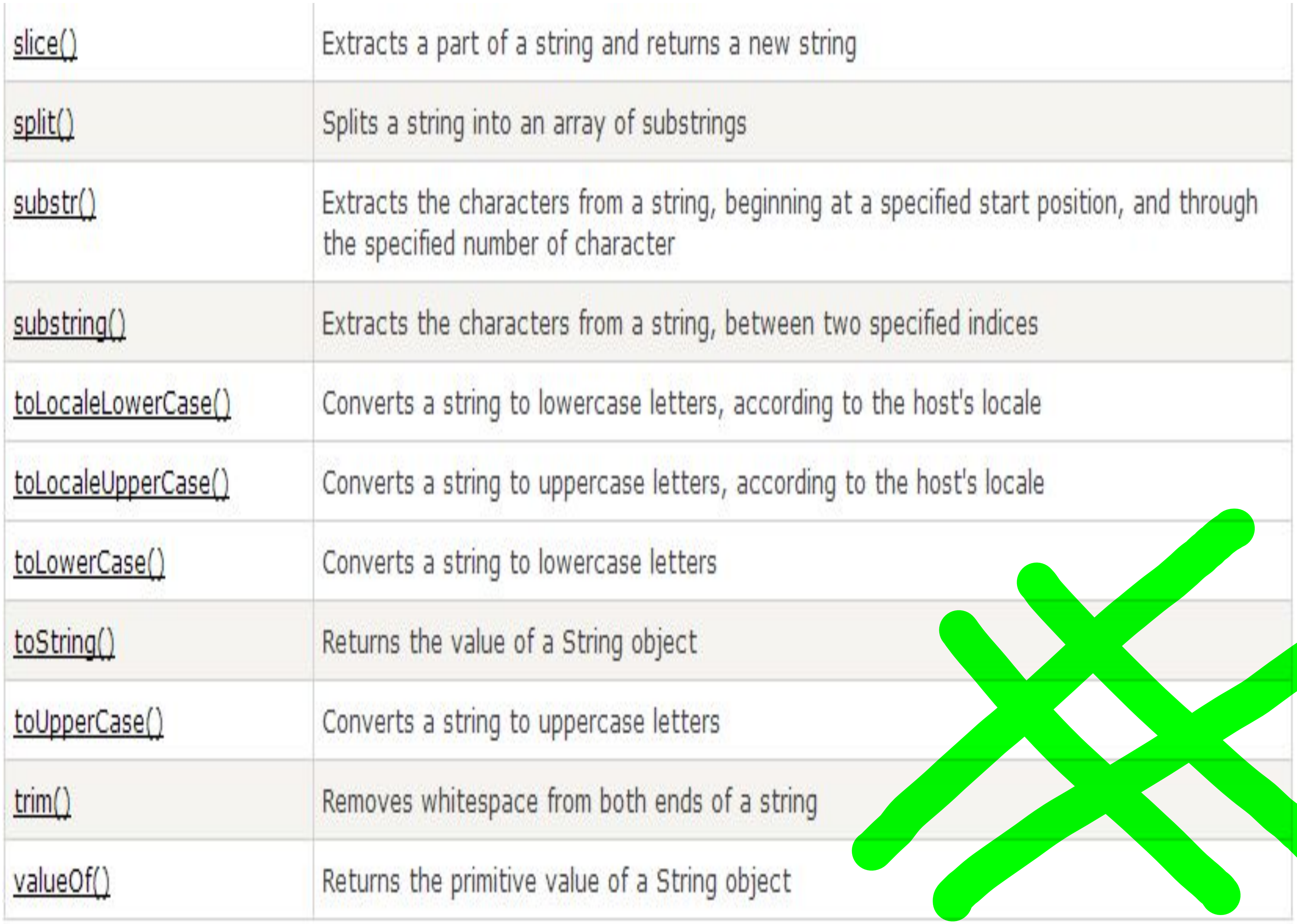
Boolean: it has only two values.it test for something and give the result as true or flase

```
// string concatenation
        var sentence1= "My name is Tom";
        var sentence2= " and i am 21 years old";
        var sentence3;
        sentence3=sentence1+sentence2;
        //document.write(sentence3);

        //string concatenation with numbers
        var age=21;
        document.write(sentence1+" and i am "+age+" years old");

        //using HTML elements inside our string
        document.write("<h1>Hello World</h1>")
        </script>
```

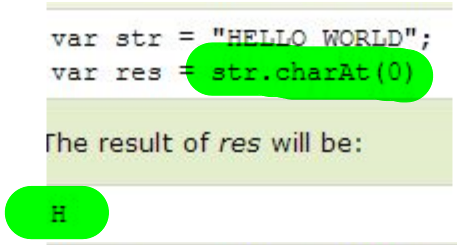
## Example: Finding Length of the String

```
<html>
<head>
</head>
<body>
<script language="javascript">
var name;
name= prompt("enter your name");
document.writeln("string length is "+name.length);
document.close();
</script>
</body>
</html>
```

| Method | Description |
|---|---|
| charAt() | Returns the character at the specified index |
| charCodeAt() | Returns the Unicode of the character at the specified index |
| concat() | Joins two or more strings, and returns a copy of the joined strings |
| fromCharCode() | Converts Unicode values to characters |
| indexOf() | Returns the position of the first found occurrence of a specified value in a string |
| lastIndexOf() | Returns the position of the last found occurrence of a specified value in a string |
| localeCompare() | Compares two strings in the current locale |
| match() | Searches for a match between a regular expression and a string, and returns the matches |
| replace() | Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring |
| search() | Searches for a match between a regular expression and a string, and returns the position of the match |



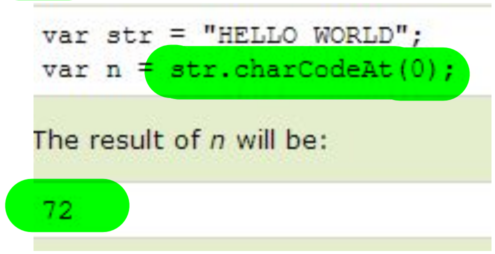
| | |
|---|---|
| slice() | Extracts a part of a string and returns a new string |
| split() | Splits a string into an array of substrings |
| substr() | Extracts the characters from a string, beginning at a specified start position, and through the specified number of character |
| substring() | Extracts the characters from a string, between two specified indices |
| toLocaleLowerCase() | Converts a string to lowercase letters, according to the host's locale |
| toLocaleUpperCase() | Converts a string to uppercase letters, according to the host's locale |
| toLowerCase() | Converts a string to lowercase letters |
| toString() | Returns the value of a String object |
| toUpperCase() | Converts a string to uppercase letters |
| trim() | Removes whitespace from both ends of a string |
| valueOf() | Returns the primitive value of a String object |



```
var str = "HELLO WORLD";
var res = str.charAt(0)
```

The result of *res* will be:

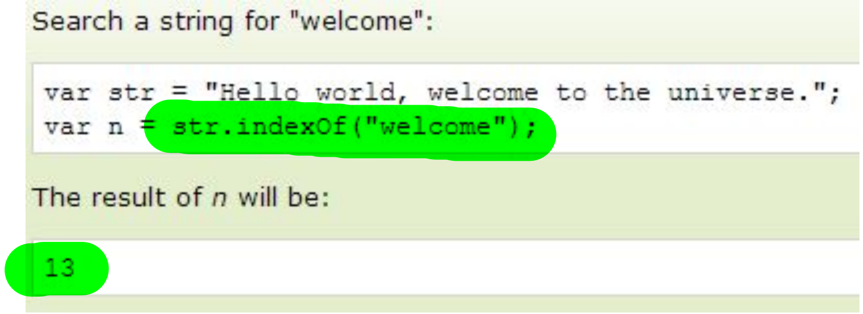
```
H
```



```
var str = "HELLO WORLD";
var n = str.charCodeAt(0);
```

The result of *n* will be:

```
72
```

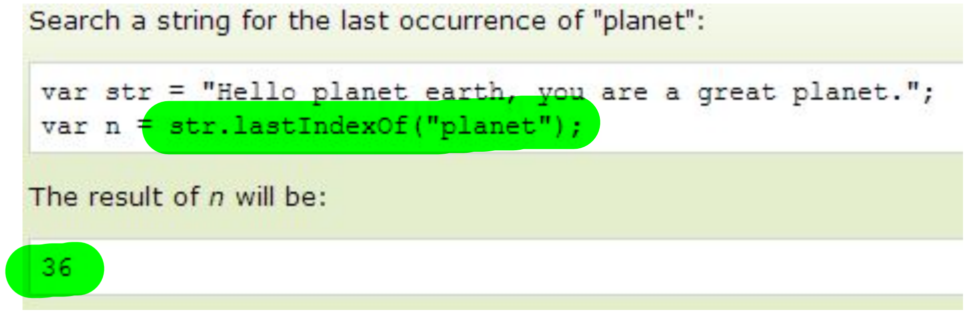


Search a string for "welcome":

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("welcome");
```

The result of *n* will be:

```
13
```

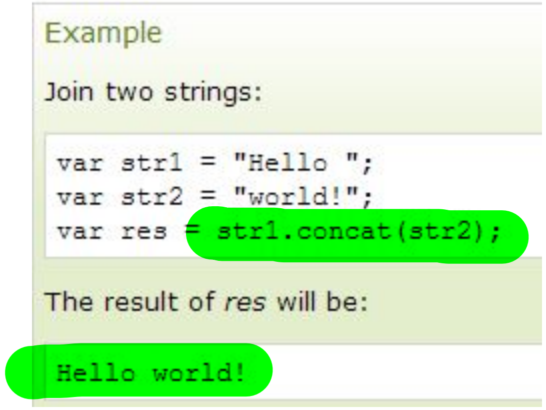


Search a string for the last occurrence of "planet":

```
var str = "Hello planet earth, you are a great planet.";
var n = str.lastIndexOf("planet");
```

The result of *n* will be:

```
36
```



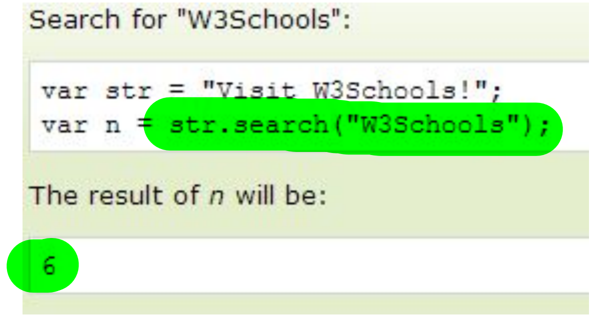
Example

Join two strings:

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);
```

The result of *res* will be:

```
Hello world!
```

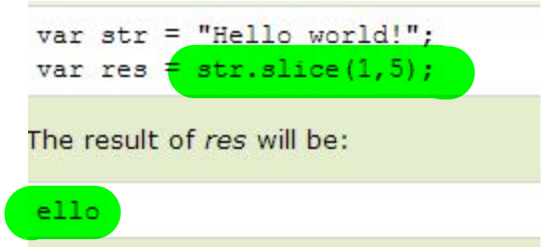


Search for "W3Schools":

```
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
```

The result of *n* will be:

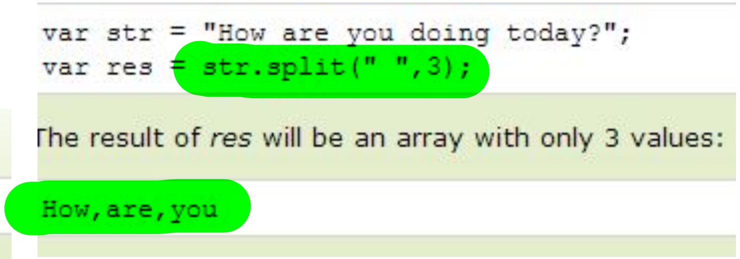
```
6
```



```
var str = "Hello world!";
var res = str.slice(1,5);
```

The result of *res* will be:

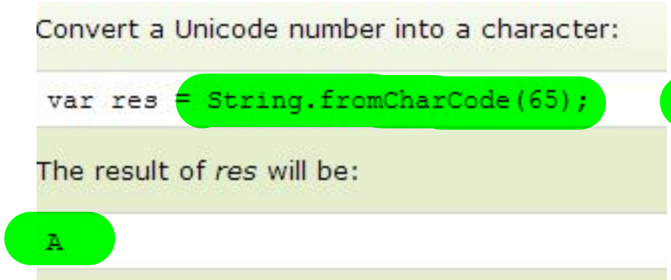
```
ello
```



```
var str = "How are you doing today?";
var res = str.split(" ",3);
```

The result of *res* will be an array with only 3 values:

```
How,are,you
```

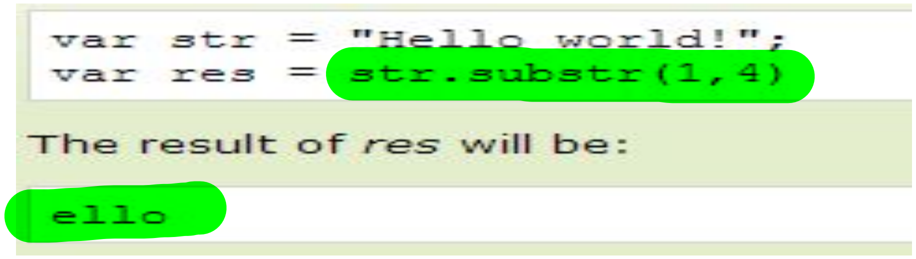


Convert a Unicode number into a character:

```
var res = String.fromCharCode(65);
```

The result of *res* will be:

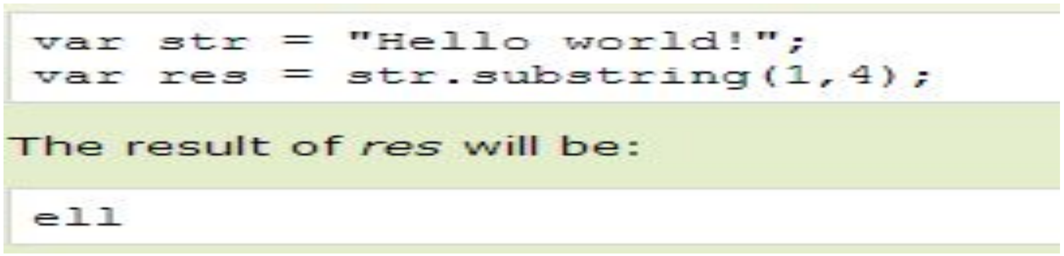
```
A
```



```
var str = "Hello world!";
var res = str.substr(1,4)
```

The result of *res* will be:

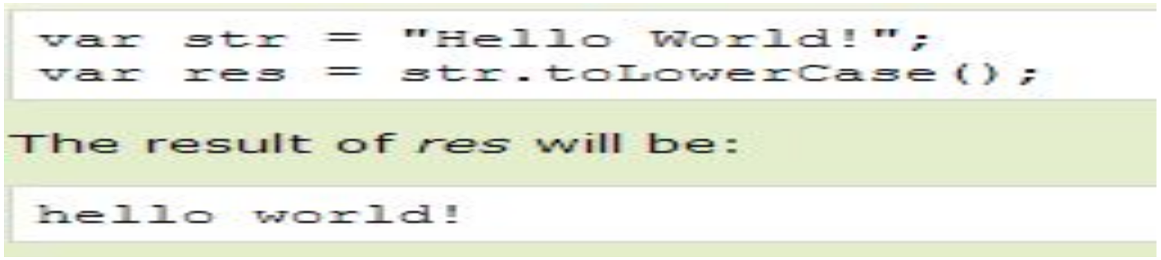
```
ello
```



```
var str = "Hello world!";
var res = str.substring(1,4);
```

The result of *res* will be:

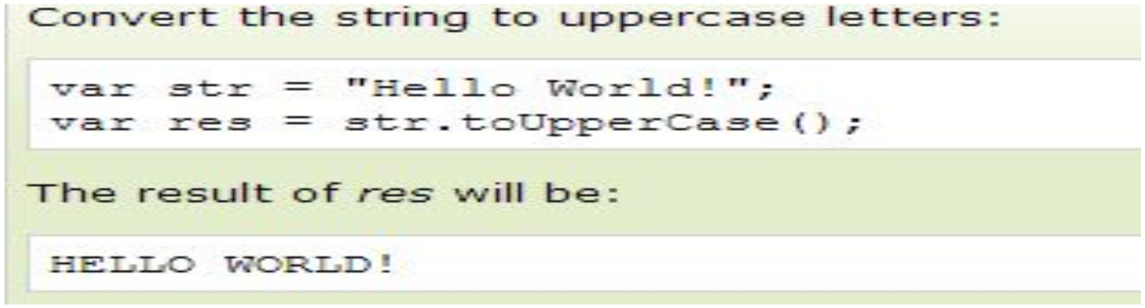
```
ell
```



```
var str = "Hello World!";
var res = str.toLowerCase();
```

The result of *res* will be:

```
hello world!
```



Convert the string to uppercase letters:

```
var str = "Hello World!";
var res = str.toUpperCase();
```

The result of *res* will be:

```
HELLO WORLD!
```

escape character( \ )

```
/*  var status="Billy said \ "javascript is the most interesting
client side scripting language\"" ;
document.write(status);
*/
```

escape character in javascript just ignore the character immediately after a slash.

- javaScript is Case Sensitive

- All JavaScript identifiers are **case sensitive**.
- The variables **lastName** and **lastname**, are two different variables.

## Functions

- A JavaScript function is a block of code designed to perform a particular task.

```
function functionname()
{
some code to be executed
}
// with arguments
function functionname(par1,par2)
{
 some code to be executed
}
```

**Example – Function Definition**

```
<script >
 function sayHello()
 {
document.write("Hello
   there");
 }
</script>
```

**Calling a Function:**

```
<script >
sayHello();
</script>
```

# Functions With Arg& Return Value

**Function Declaration**

```
<script >
 function concatenate(first,
    last)
 {
 var full;
 full = first + last;
return full;
 }
 </script>
```

**Function Calling**

```
<script >
var result;
 result = concatenate('Zara',
    'Ali');
 document.write(result );
</script>
```

# Sample program 1

```
<head>
<script>
function sum(num1,  num2)        //Function Definition
{
var s=(num1)+(num2);
document.write("addition="+s);
}
sum(3,4);                   // Function  Calling
</script>
</head>
```

## Example 2:

```
<html>
<head><script>
function sum(num1,  num2)    //Function Definition
{
var s=(num1)+(num2);
document.write("addition="+s);
} </script></head>
<body><script>
sum(3,4);   </script></body>   // Function  Calling
</html>
```

## With return type

```
<body>
<script>
function sum()        //Function Definition
{
var num1=prompt("enter 1st no");
var num2=prompt("enter 2nd no");
var s=parseInt(num1)+parseInt(num2);
return s;
}
var result=sum();          // Function  Calling
document.write(result);
</script>
</body>
```

# Calling function inside Table from HTML

```
<html>
<head>
<script >
function sayhello()
{
document.write("I am
    saying hello");
}
</script>
</head>
```

```
<body>
<script>
document.writeln("<html>
<head>HI</head><body>");
document.writeln("<h1>
welcome</h1>");
document.writeln("<table>
<tr><td>First Time");
        sayhello();
document.writeln("</td></tr>
    ");
```

```
document.writeln("<tr><td>
Second Time");
        sayhello();
document.writeln("</td>
</tr></table></body></html>");
</script>
</body>
</html>
```

**OUTPUT:**

```
HI
welcome
First Time I am saying
   hello
Second Time I am
   saying hello
```

## Scoping rules

Programming language usually impose rules called scoping, which determines how a variable can be accessed.

1. global variable
2. local variable

**JavaScript Scope**

Scope is the set of variables you have access to.

**Local JavaScript Variables**

Variables declared within a JavaScript function,
become **LOCAL** to the function.

Local variables have **local scope**: They can only be accessed within the function.

**Global JavaScript Variables**

A variable declared outside a function, becomes **GLOBAL**.

A global variable has **global scope**: All scripts and functions on a web page can access it.

**Example:**

```
<body><script>
var a=10;   G
function myFunction() {
var a=20;   L
 document.write("A value inside function is"+a);
}
 document.write("A value outside function is”
   +a+"<br>");
myFunction();</script></body>
O/P:   ?  ?
```

## JavaScript Events

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

## HTML Events

An HTML event can be something the browser does, or something a user does.

**Here are some examples of HTML events:**

An HTML web page has finished loading

An HTML input field was changed

An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

**With single quotes:**

<*some-HTML-element some-event*='***some JavaScript***'>

#

| Event | Description |
|---|---|
| Onchange | An HTML element has been changed |
| Onclick | The user clicks an HTML element |
| Onmouseover | The user moves the mouse over an HTML element |
| Onmouseout | The user moves the mouse away from an HTML element |
| Onkeydown | The user pushes a keyboard key |
| Onload | The browser has finished loading the page |

Onclick event: (when clicking the button)

```
<!DOCTYPE html>
<html lang=”en”>
        <head>
                <meta charset="utf-8"/>
                <title>javascript introduction</title>
                <script type="text/javascript">
                //alert("you have opened my web page");

                </script>
        </head>
        <body>
        <form>
        <input type="button" value="click me" onclick="alert('you clicked
me')"/>
        </form>
        </body>
</html>
```

Onmouseover event:

```
<body>
        <form>
        <input type="button" value="click me"
onmouseover="alert('you hovered over me')"/>
        </form>
        </body>
```

## JavaScript Operators

- JavaScript uses an **assignment operator** ( = ) to **assign** values to variables:
- var x = 5;
  var y = 6;
- JavaScript uses **arithmetic operators** ( + - * / )
  to **compute** values:
- (5 + 6) * 10
- JavaScript is Case Sensitive

# Objects in javascript

- "Everything" in JavaScript is an Object.
- In addition, JavaScript allows you to define your own objects.
- An object is just a special kind of data, with **properties(attributes)** and **methods**

In JavaScript almost everything is an object. Even primitive datatypes (except null and undefined) can be treated as objects.

- Booleans can be objects or primitive data treated as objects
- Numbers can be objects or primitive data treated as objects
- Strings are also objects or primitive data treated as objects
- Dates are always objects
- Maths and Regular Expressions are always objects
- Arrays are always objects
- Even functions are always objects

- Objects are variables too. But objects can contain many values.
- The values are written as **name : value** pairs (name and value separated by a colon).
- JavaScript variables can contain single values:
- **Example**
- var person = "John Doe";
- **Example**
- var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

## Accessing Object Properties

- Properties are the values associated with an object.
- The syntax for accessing the property of an object is:

  *objectName.propertyName*

**Ex:**

```
var car = {type:"Fiat", model:500,
color:"white"};
document.write(car.type);
```

**O/P : Fiat**

# Accessing Objects Methods

- Methods are the actions that can be performed on objects.
- You can call a method with the following syntax:

  *objectName.methodName()*

- This example uses the toUpperCase() method of the String object, to convert a text to uppercase:

```
var message="Hello world!";
var x=message.toUpperCase();
```

- The value of x, after execution of the code above will be:

```
HELLO WORLD!
```

## A Real Life Object. A Car:

| Object | Properties | Methods |
|---|---|---|
| | car.name = Fiat<br>car.model = 500<br>car.weight = 850kg<br>car.color = white | car.start()<br>car.drive()<br>car.brake() |

**Example:**

```
<html>
<body>
<script>
var person=new Object();
person.firstname="John";
person.lastname="Doe";
person.age=50;
document.write(person.firstname + " is " +
   person.age + " years old.");
</script>
</body>
</html>
```

## Using an Object Constructor

The examples above are limited in many situations. They only create a single object.

Sometimes we like to have an "object type" that can be used to create many objects of one type.

The standard way to create an "object type" is to use an object constructor function:

**Example**

```
function person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}
var myFather = new person("John", "Doe", 50, "blue");
var myMother = new person("Sally", "Rally", 48, "green");
```

## Date object

The Date object lets you work with dates (years, months, days, minutes, seconds, milliseconds)

- Date objects are created with new Date().
- There are four ways of instantiating a date:

  var d = new Date();
  var d = new Date(*milliseconds*);
  var d = new Date(*dateString*);
  var d = new Date(*year, month, day, hours, minutes, seconds, milliseconds*);

**JavaScript dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC). One day contains 86,400,000 millisecond.**

# Example

**Type 1**

```
<body>
<script>
var d = new Date();
document.write(d);
</script>
</body>
```

**O/P:**

```
Thu Feb 12 2015
```

**Type 2**

```
<body>
<script>
var d = new Date("October
    13 2014");
document.write(d);
</script>
</body>
```

**O/P:**

```
Mon Oct 13 2014
```

## Type 3

```
<body>
<script>
var d = new
    Date(86400000);
document.write(d);
</script>
</body>
```

**O/P:**

```
Fri Jan 02 1970 05:30:00
```

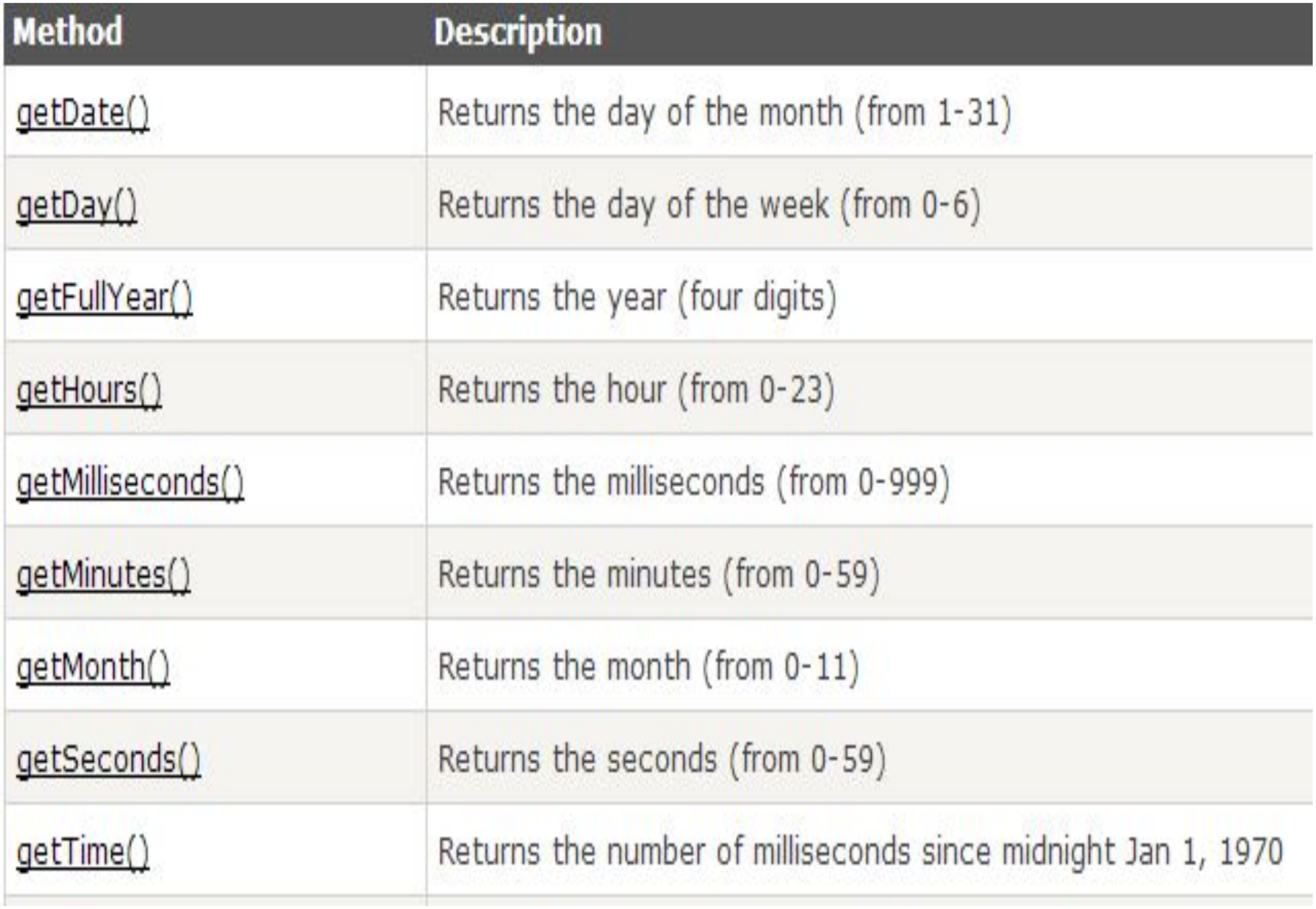
## Type 4

```
<body>
<p id="demo"></p>
<script>
var d = new
    Date(2015,1,12,11,33,3
    0,0);
document.write(d);
</script>
</body>
```

**O/P:**

```
Thu Feb 12 2015 11:33:30
```

# DATE METHODS



| Method | Description |
|---|---|
| getDate() | Returns the day of the month (from 1-31) |
| getDay() | Returns the day of the week (from 0-6) |
| getFullYear() | Returns the year (four digits) |
| getHours() | Returns the hour (from 0-23) |
| getMilliseconds() | Returns the milliseconds (from 0-999) |
| getMinutes() | Returns the minutes (from 0-59) |
| getMonth() | Returns the month (from 0-11) |
| getSeconds() | Returns the seconds (from 0-59) |
| getTime() | Returns the number of milliseconds since midnight Jan 1, 1970 |

| | |
|---|---|
| setDate() | Sets the day of the month of a date obje |
| setFullYear() | Sets the year (four digits) of a date obje |
| setHours() | Sets the hour of a date object |
| setMilliseconds() | Sets the milliseconds of a date object |
| setMinutes() | Set the minutes of a date object |
| setMonth() | Sets the month of a date object |
| setSeconds() | Sets the seconds of a date object |
| setTime() | Sets a date and time by adding or subtra midnight January 1, 1970 |

| | |
|---|---|
| toLocaleDateString() | Returns the date portion of a Date object as a string, using locale conventions |
| toLocaleTimeString() | Returns the time portion of a Date object as a string, using locale conventions |
| toLocaleString() | Converts a Date object to a string, using locale conventions |
| toString() | Converts a Date object to a string |

# Calculating number of days between two dates.

```
<HTML>
<body>
<script>
var d=new Date();
var d1=new Date("2014 1
   10");
var day=24*60*60*1000;
var no =
   (d1.getTime()-d.getTime
   ())/day;
document.write(Math.ceil
   (no));
document.write(d);
document.write(d1);
</script>
</body>
</html>
```

## Combining Date &Functions into Objects

```
html>
<head>
<script >
function objdemo()
{
myhouse=new
   house("welcome",2,4);
alert(myhouse.name+myho
   use.floors+myhouse.bed
   s+myhouse.rooms());
}
```

```
function
   house(name,floors,beds)
{
this.name=name;
this.floors=floors;
this.beds=beds;
this.rooms=frooms;
}
```

```
function frooms()
{
if(this.floors==0)
return 1;
else
{
return 2;
}
}
```

```
</script>
</head>
<body
    onLoad="objdemo()">
</body>
</html>
```

**O/P:** welcome 2 4 2

## Builtin objects

- Document object
- Window object
- Browser object
- Form object

# Document object

When an HTML document is loaded into a web browser, it becomes a **document object** .A document is a web page that is being either displayed or created.

The documents has number of properties that can be accessed by javascript program.

1. write
2. writeln
3. bgcolor
4. close()

**Example:**

- document.bgcolor=“red”;
- document.body.style.backgroundImage=

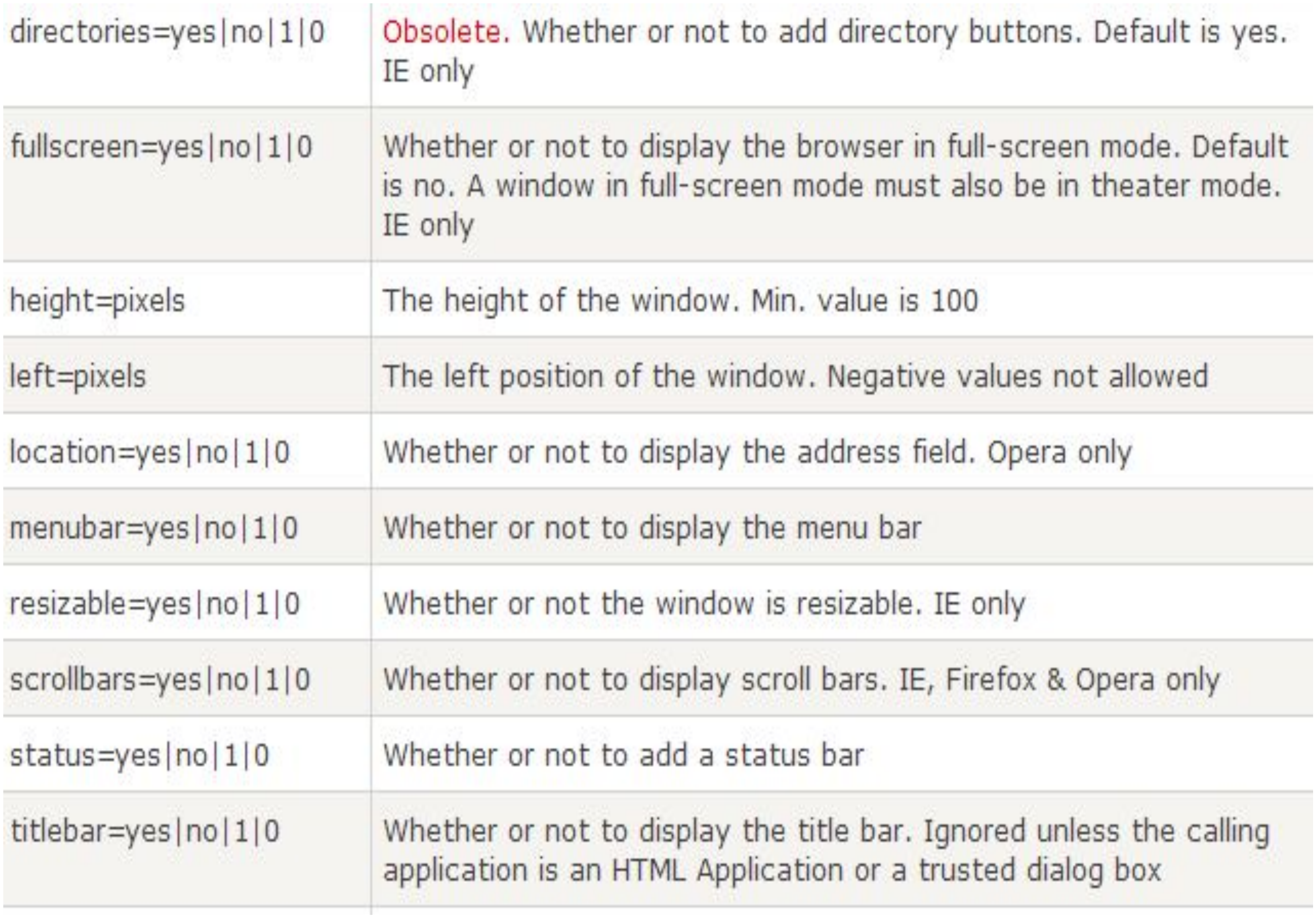
  "url('img_tree.png')";
- document.getElementById("demo");

# Window object

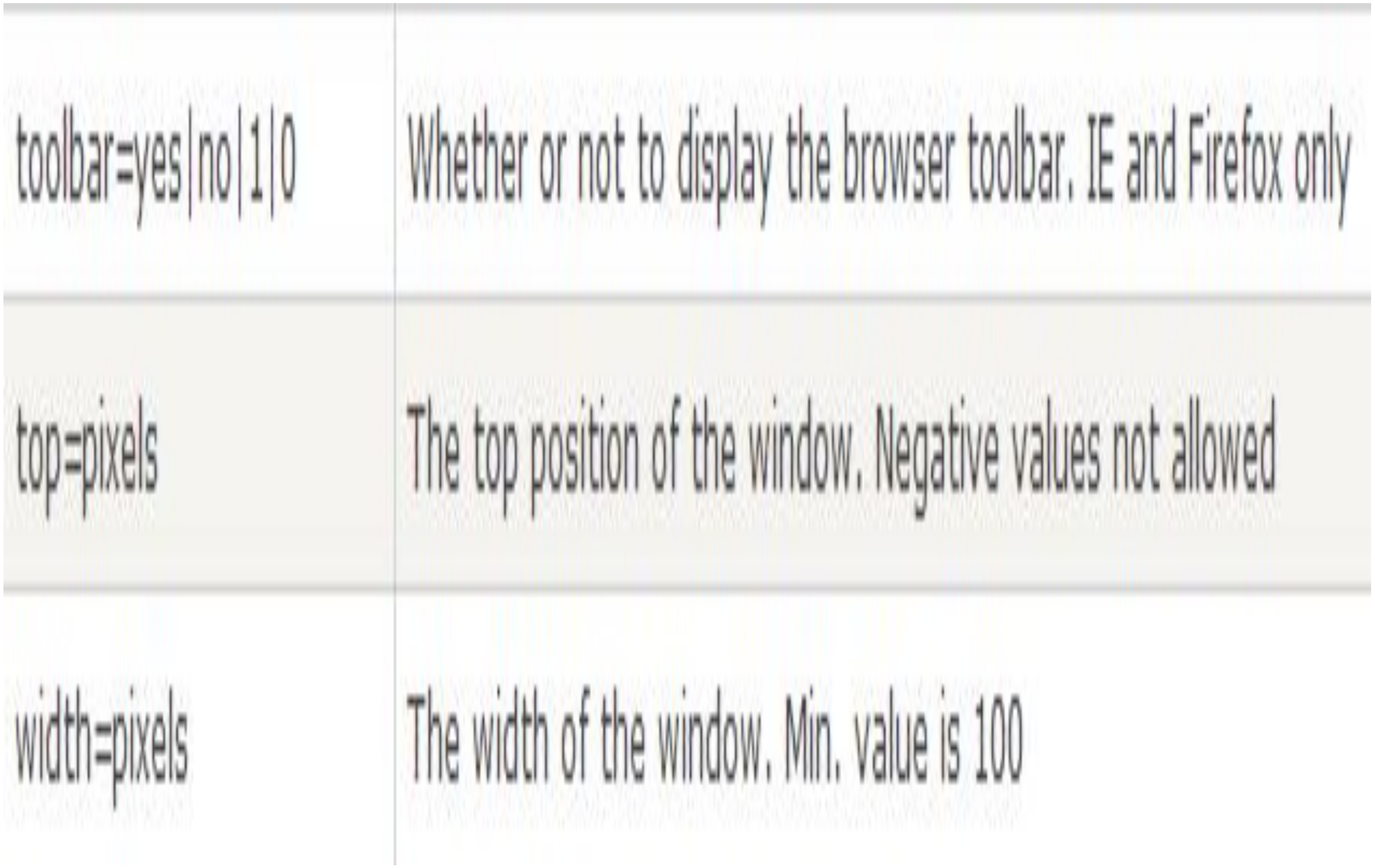
- The window object represents an open window in a browser.
- If a document contain frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

## Window object Properties

-  Open(“URL”,”name”)
-  Close()
-  toolbar=[1|0];
-  location =[1|0];
-  directories =[1|0];
-  status =[1|0];
-  Menubar =[1|0];
-  scrollbar =[1|0];
-  resizable =[1|0];
-  Width=pixels
-  Height=pixels
-  Scroll(coordinate, coordinate)



| | |
|---|---|
| directories=yes\|no\|1\|0 | Obsolete. Whether or not to add directory buttons. Default is yes. IE only |
| fullscreen=yes\|no\|1\|0 | Whether or not to display the browser in full-screen mode. Default is no. A window in full-screen mode must also be in theater mode. IE only |
| height=pixels | The height of the window. Min. value is 100 |
| left=pixels | The left position of the window. Negative values not allowed |
| location=yes\|no\|1\|0 | Whether or not to display the address field. Opera only |
| menubar=yes\|no\|1\|0 | Whether or not to display the menu bar |
| resizable=yes\|no\|1\|0 | Whether or not the window is resizable. IE only |
| scrollbars=yes\|no\|1\|0 | Whether or not to display scroll bars. IE, Firefox & Opera only |
| status=yes\|no\|1\|0 | Whether or not to add a status bar |
| titlebar=yes\|no\|1\|0 | Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box |



| | |
|---|---|
| toolbar=yes\|no\|1\|0 | Whether or not to display the browser toolbar. IE and Firefox only |
| top=pixels | The top position of the window. Negative values not allowed |
| width=pixels | The width of the window. Min. value is 100 |

**Example:**

```
window.open("http://www.w3schools.com",
"_blank", "toolbar=yes,  location=yes,
   directories=yes, status=no, menubar=yes,
   scrollbars=yes, resizable=no, copyhistory=yes,
   width=400, height=400");
```

# Form object

- The Form object represents an HTML <form> element.

**Access a Form Object:**

Access a <form> element by using
getElementById()

## Browser object Properties

-  Navigator.appVersion;

Version number of browser

-  Navigator.appName

Public name of browser

-  Navigator.appCodeName

Internal name of browser

-  Navigator.userAgent

appCodeName and appVersion are concatenated together

-  Navigator.mimeTypes

   To handle  graphics, audio, and video files via the Internet mail system.

-  Navigator.plugins

   An array containing details of all installed plug- ins.

**Example:**

```
<script>
var x = "Version info: " + navigator.appVersion;
document.write(x);
</script>
```

**o/p:** Version info: 5.0 (Windows NT 6.1)

## DOM
## (Document Object Model)

**What is the DOM?**

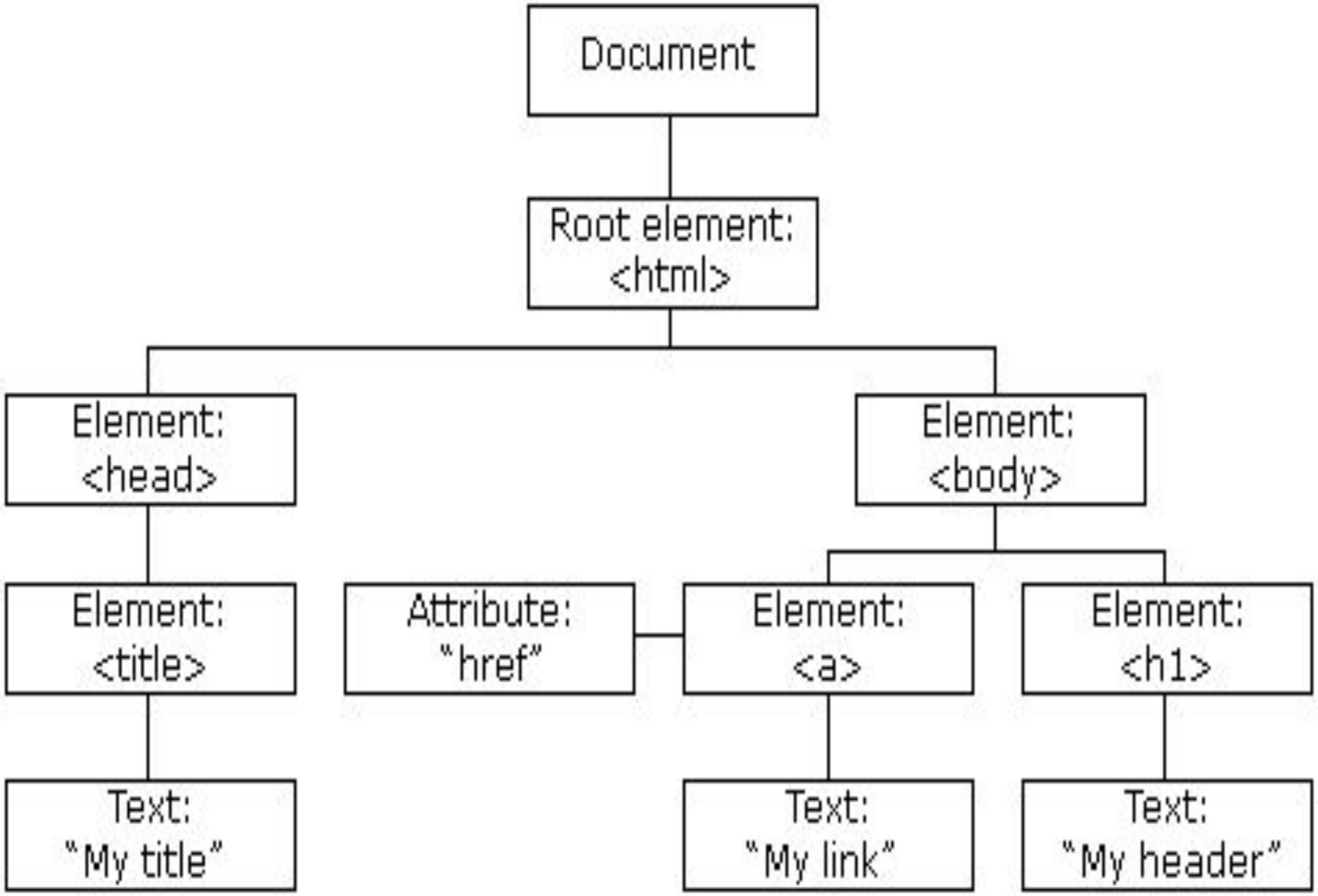
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a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

**JavaScript HTML DOM**

- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

**The HTML 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 is a standard for how to get, change, add, or delete HTML elements.**
- The **HTML DOM** model is constructed as a tree of **Objects**:


Document
Root element: <html>
Element: <head>
Element: <body>
Element: <title>
Attribute: "href"
Element: <a>
Element: <h1>
Text: "My title"
Text: "My link"
Text: "My header"

- 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

## 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 |

## Finding HTML Element by Id

### Example

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

```
<html>
<body>

<p id="demo"></p>

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

</body>
</html>
```

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

**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 by Tag Name**

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<p>The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b>
method</p>
<p id="demo"></p>
<script>
var x = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) is: ' + x[0].innerHTML;
</script>
</body>
</html>
```

**This example finds the element with id="main", and then finds all <p> elements inside "main":**

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<div id="main">
<p>The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method</p>
</div>
<p id="demo"></p>
<script>
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) inside "main" is: ' + y[0].innerHTML;
</script>
</body>
</html>
```

**Finding HTML Elements by Class Name**

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the
<b>getElementsByClassName</b> method.</p>
<p id="demo"></p>
<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>
</body>
</html>
```

## Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

**To change the content of an HTML element, use this syntax**:

document.getElementById(*id*).innerHTML = *new HTML*

*Example:*

```
<body>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
</body>
```

## Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

**document.getElementById(*id*).*attribute=new value***

**This example changes the value of the src attribute of an <img> element:**

Example

```
<body>
<img id="myImage" src="smiley.gif">
<script>
document.getElementById(‘myImage’).src = "landscape.jpg";
</script>
</body>
```

**Changing an image source.**

```
<!DOCTYPE html>
	<head>
		<meta charset="utf-8"/>
		<title>javascript introduction</title>
		<script type="text/javascript">
		function change()
		{
			var x=document.getElementById('flowers');
			x.src="lotus.jpg";
		}
		</script>
	</head>
	<body>
	<img src=flower.jpg alt="load failed" id=flowers  onclick="change();"/>
	</body>
</html>
```

```
<!DOCTYPE html>
    <head>
        <script type="text/javascript">
        var image_tracker='f';
        function change()
        {       var x=document.getElementById('flowers');
                if (image_tracker=='f'){
                x.src="lotus.jpg";
                image_tracker='l';
                }
                else{
                x.src="flower.jpg";
                image_tracker='f';
                }
        }
        </script>
    </head>
    <body>
    <img src=flower.jpg alt="load failed" id=flowers onclick="change();"/>
    </body>
</html>
```

**Using a timer**

but instead of clicking that image we make it change that image by itself over time.  For example, it would look like a  rotating add on a website. Similary most of the images or banner gets change in the website for every couple of seconds just to attract you.

**Use setTimeout() method**

```
setTimeout('code iin here', time milliseconds);
```

one second= 1000 millisecond;

```
setTimeout('change()',2000);
```

when you run this, the image change only once. Its not changing again. That's becos of setTimeout method. By using this , it once runs this code and it never happens again.

You want the timer to repeat and keep changing the image, you can use the method setInterva() method. This method takes same arguments like setTimer() method.

**setInterval(‘change()’,5000);**

if we want to be able to use more than one interval or more than one timer or to stop the timer on a web page. We actually needs to give setTimeout or srtInterval a variable name.

**var timer= setInterval(‘change()’,5000);**

if I want to stop timer and I don’t want my image to carry on repeating then what we can do is we can clear that interval using clearInterval() method.

**clearInterval( timer);** // timer is variable name

it stops the timer from actually running.

So type this in a event. Here, onclick is the event.

```
<body>
<img src=flower.jpg alt="load failed" id=flowers
                onclick=" clearInterval( timer);"/>
</body>
```

## Changing CSS

The HTML DOM allows JavaScript to change the style of HTML elements.

To change the style of an HTML element, use this syntax:

**document.getElementById(*id*).style.*property=new style***

**example**

```
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
```

## Using Events

The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements:

- –An element is clicked on
- –The page has loaded
- –Input fields are changed

This example changes the style of the HTML element with id="id1", when the user clicks a button:

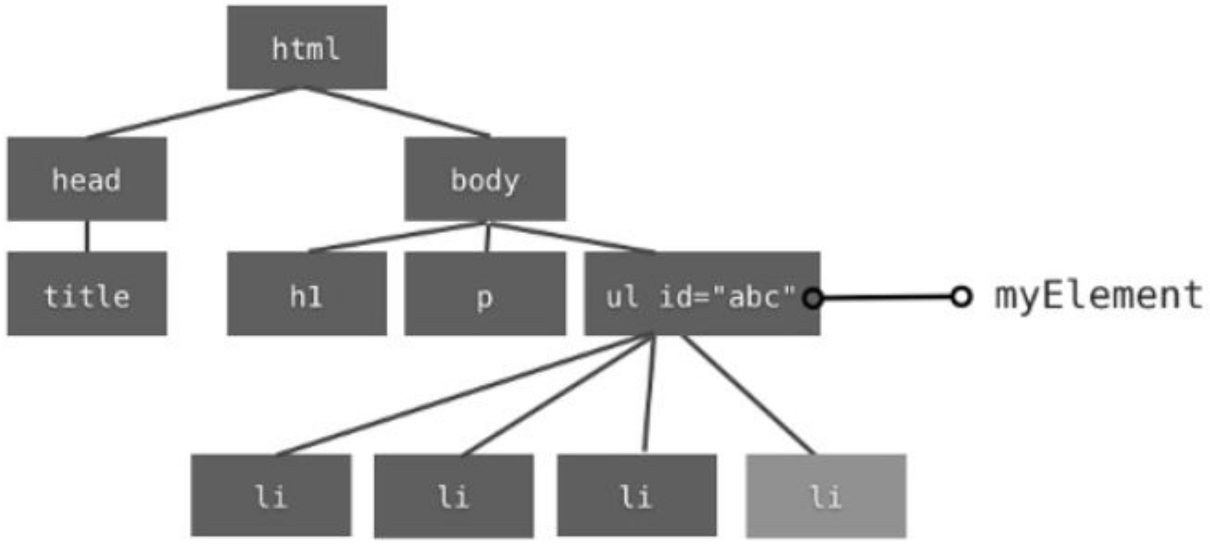
```
<body>
<h1 id="id1">My Heading 1</h1>
<button type="button"
        onclick="document.getElementById('id1').style.color = 'red'">
            Click Me!</button>
</body>
```

## CREATING DOM CONTENT

1. **create** the element
2. **add** it to the document

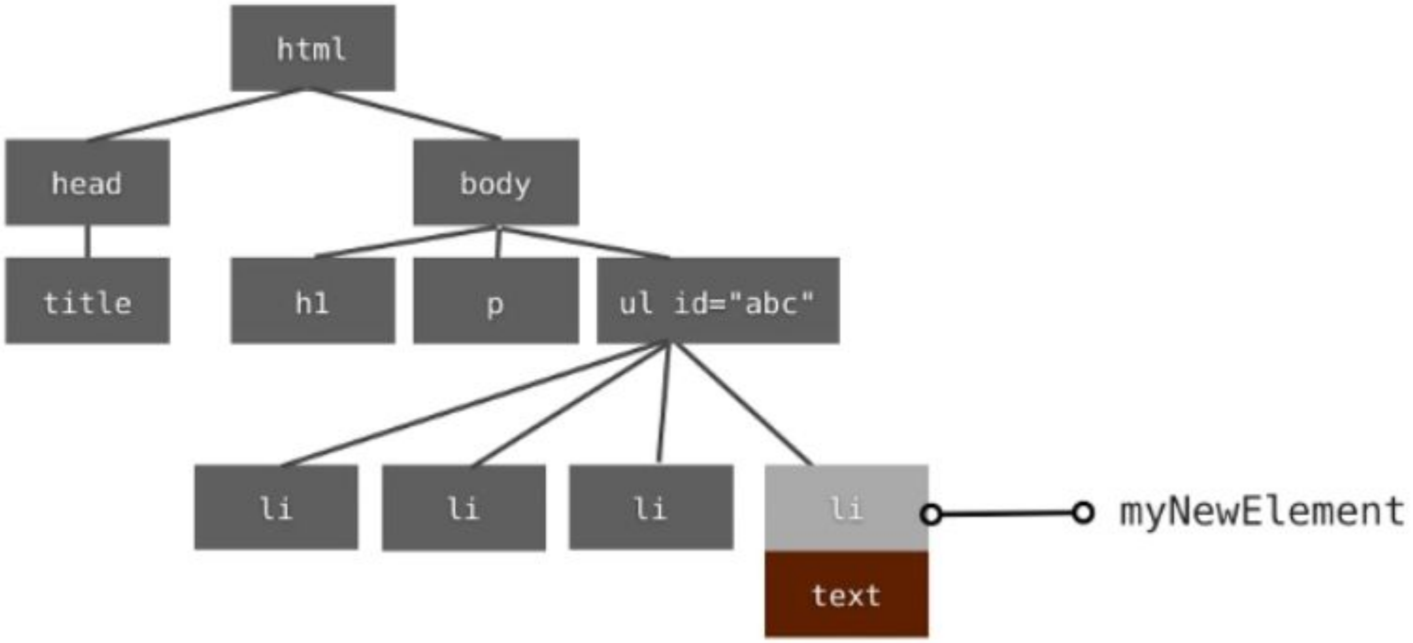
## CREATING ELEMENTS

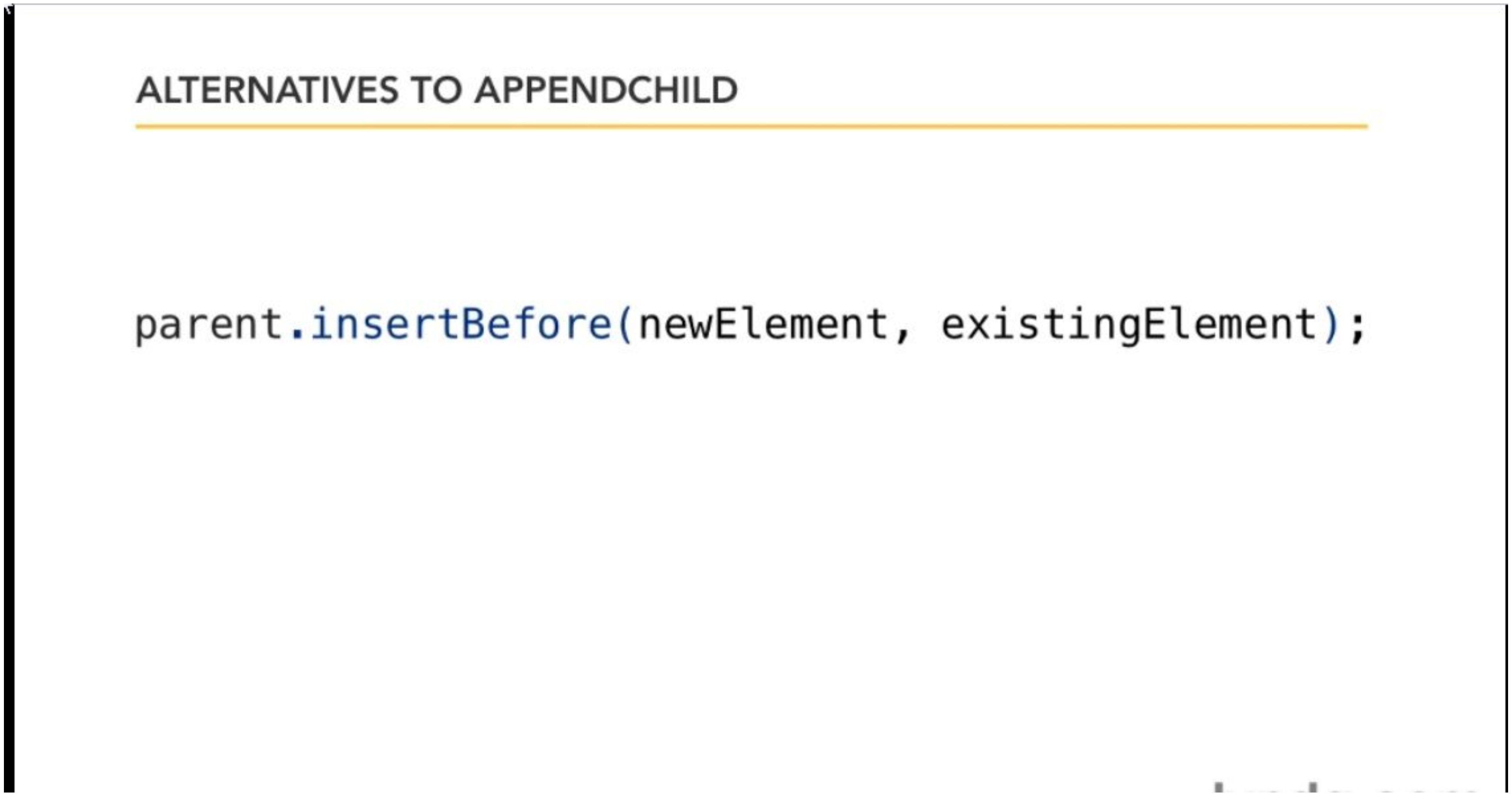
```
var myNewElement = document.createElement("li");
myElement.appendChild(myNewElement);
myNewElement.innerHTML = "New item text";
```



## CREATING TEXT NODES

```
var myText = document.createTextNode("New list item text");
myNewElement.appendChild(myText);
```



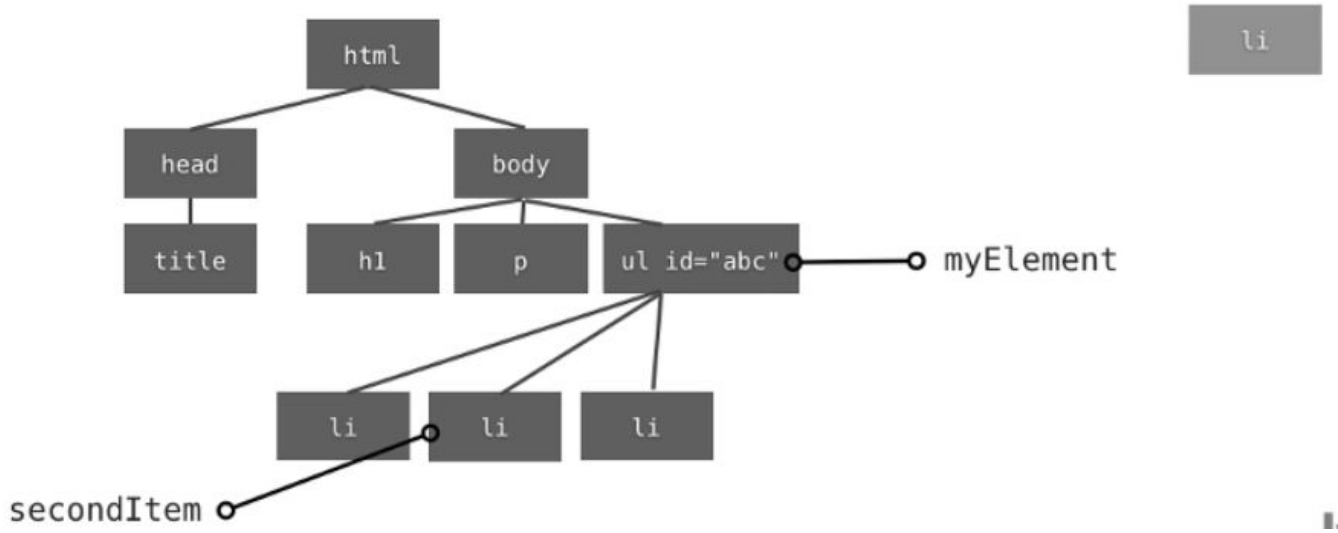


## ALTERNATIVES TO APPENDCHILD

```
parent.insertBefore(newElement, existingElement);
```

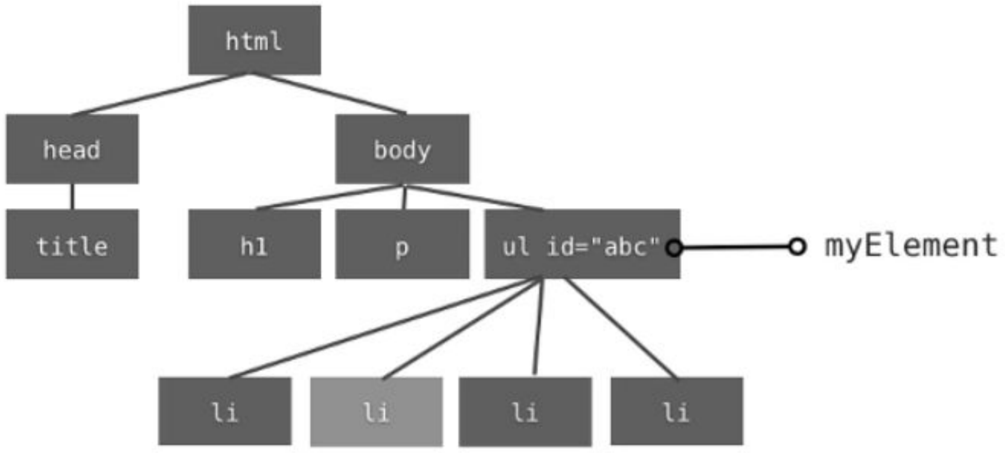
## ALTERNATIVES TO APPENDCHILD

```
var myNewElement = document.createElement("li");
var secondItem = myElement.getElementsByTagName("li")[1];
myElement.insertBefore(myNewElement, secondItem);
```



## ALTERNATIVES TO APPENDCHILD

```
var myNewElement = document.createElement("li");
var secondItem = myElement.getElementsByTagName("li")[1];
myElement.insertBefore(myNewElement, secondItem);
```



**JavaScript Events**

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

**HTML Events**

An HTML event can be something the browser does, or something a user does.

**Here are some examples of HTML events:**

An HTML web page has finished loading

An HTML input field was changed

An HTML button was clicked

Often, when events happen, you may want to do something.

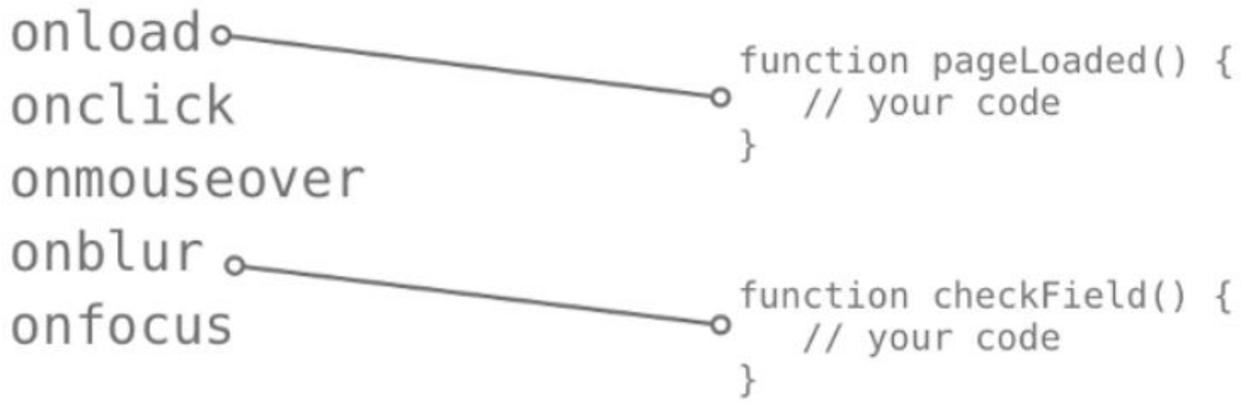
JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

**With single quotes:**

<*some-HTML-element some-event*=**'*some JavaScript*'**>

## EVENT NAMES



```
onload
onclick
onmouseover
onblur
onfocus
```

```
function pageLoaded() {
   // your code
}
```

```
function checkField() {
   // your code
}
```

## HANDLING EVENTS: METHOD 1

```
<button onclick="alert('Hello, world');">
   Run Some JavaScript
</button>
```

Onmouseover event:

```
<body>
    <form>
    <input type="button" value="click me"
    onmouseover="alert('you hovered over me')"/>
    </form>
    </body>
```

## HANDLING EVENTS: METHOD 2

```
element.event =
```

```
window.onload
```

```
nameField.onblur
```

## HANDLING EVENTS: METHOD 2

```
myelement.onclick = function() {
   // your event handler code
   // ...
   // ...
};
```

## Exception handling

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors(throw exception explicitly based on our own defined condition).

Errors Will Happen!

When the JavaScript engine is executing JavaScript code, different errors can occur:

- It can be syntax errors, typically coding errors or typos made by the programmer.
- It can be misspelled or missing features in the language (maybe due to browser differences).
- It can be errors due to wrong input, from a user, or from an Internet server.
- And, of course, it can be many other unforeseeable things.

JavaScript Throws Errors

- When an error occurs, when something goes wrong, the JavaScript engine will normally stop, and generate an error message.
- The technical term for this is: JavaScript will **throw** an error.

# JavaScript - Try and Catch

**Syntax**

```
try
    {
    //statements that may cause an exception
  }
catch(e)
    {
    //Handle errors here
    }
```

## Example 2: TRY CATCH THROW

```
<html>
<head>
<script >
function checkeligibility(age)
{
     try {
if(age<16){
throw("Not Eligible for voting"
    );
              }
         }
catch (e ) {
    document.write(e) ;
           }
 }
</script></head>
<body>
<script>
 checkeligibility(13);
</script>
</body>
</html>
```

## Example 3: TRY CATCH THROW WITH EVENT

```
<html>
<head><script>
function message()
{
try {
var a=prompt("1s no");
var b=prompt("2nd no");
if (b==0)
throw "pls enter valid no";
else
document.write(a/b);
  }
catch(err)
  {
    alert(err);
  }
}
</script></head>
<body>
<input type="button"
    value="View message"
    onclick="message()" />
</body></html>
```

## jQuery

- **jQuery** is a lightweight, "write less, do more", **JavaScript library**.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- 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

- There are lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.
- Many of the biggest companies on the Web use jQuery, such as:

  Google

  Microsoft

  IBM

  Netflix

**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

## jQuery Syntax

- With jQuery you select (query) HTML elements and perform "actions" on them.
- Basic 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"

## The Document Ready Event

- It is good to put all jQuery methods inside a document ready event:

```
$(document).ready(function(){

   // jQuery methods go here...

});
```

- This is to **prevent any jQuery code from running before the document is finished loading (is ready).**

- It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.
- Here are **some examples** of actions that can fail if methods are run before the document is fully loaded:
  - Trying to hide an element that is not created yet
  - Trying to get the size of an image that is not loaded yet

## jQuery Selectors

- jQuery selectors **allow you to select and manipulate HTML element(s).**
- jQuery selectors are used to **"find"** (or select) **HTML elements based on their id, classes, types, attributes, values of attributes** and much more.
- **All selectors in jQuery start with the dollar sign and parentheses: $().**
- **The element Selector :** $("p")
- **The #id Selector :** $("#test")
- **The .class Selector :** $(".test")

| | |
|---|---|
| $("*") | Selects all elements |
| $(this) | Selects the current HTML element |
| $("p.intro") | Selects all <p> elements with class="intro" |
| $("p:first") | Selects the first <p> element |
| $(":button") | Selects all <button> elements and <input> elements of type="button" |

## jQuery Syntax For Event Methods

- To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

- The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

## Commonly Used jQuery Event Methods

- $(document).ready()
- click()
- dblclick()
- mouseenter()
- mouseleave()
- hover()
- focus()
- blur()

## jQuery Effects

- **Hide, Show, Toggle, Slide, Fade, and Animate.**
- With jQuery, **you can hide and show HTML elements with the hide() and show() methods:**

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

- The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

```
$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

Thank You