What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

What can a JavaScript Do?

1. **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

2. **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page

3. **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.

4. **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element

5. **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

6. **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and depending on the browser - load another page specifically designed for that browser

7. **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.
Where to Put the JavaScript

1. In Head Section
   
   ```html
   <html>
   <head>
   <script type="text/javascript">
   ------
   </script>
   </head>
   </html>
   ```

2. In a Body Section
   
   ```html
   <html>
   <head>
   </head>
   <body>
   <script type="text/javascript">
   ....
   </script>
   </body>
   </html>
   ```

JavaScript Data Type


JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions. A variable can have a short name, like `x`, or a more descriptive name, like `carname`.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables. You can declare JavaScript variables with the **var statement**:

```javascript
var x;
var carname;
```
```javascript
var x=5;
var carname="Volvo";

e.g
<html>
<body>
<script type="text/javascript">
var firstname;
firstname="Hege";
document.write(firstname);
document.write("<br/>");
firstname="Tove";
document.write(firstname);
</script>
</body>
</html>

o/p
Hege
Tove

### JavaScript Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result of x</th>
<th>Result of y</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>x=y+2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>x=y-2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>x=y*2</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>x=y/2</td>
<td>2.5</td>
<td>5</td>
</tr>
<tr>
<td>%</td>
<td>Modulus (division remainder)</td>
<td>x=y%2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
<td>x=++y</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x=y++</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
<td>x=--y</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x=y--</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

### JavaScript Assignment Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Same As</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>x=y</td>
<td></td>
<td>x=5</td>
</tr>
<tr>
<td>+=</td>
<td>x+=y</td>
<td>x=x+y</td>
<td>x=15</td>
</tr>
<tr>
<td>-=</td>
<td>x-=y</td>
<td>x=x-y</td>
<td>x=5</td>
</tr>
<tr>
<td>*=</td>
<td>x*=y</td>
<td>x=x*y</td>
<td>x=50</td>
</tr>
<tr>
<td>/=</td>
<td>x/=y</td>
<td>x=x/y</td>
<td>x=2</td>
</tr>
<tr>
<td>%=</td>
<td>x%=y</td>
<td>x=x%y</td>
<td>x=0</td>
</tr>
</tbody>
</table>
Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Comparing</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>is equal to</td>
<td>x==8</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x==5</td>
<td>true</td>
</tr>
<tr>
<td>===</td>
<td>is exactly equal to (value and type)</td>
<td>x===&quot;5&quot;</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x===5</td>
<td>true</td>
</tr>
<tr>
<td>!=</td>
<td>is not equal</td>
<td>x!=8</td>
<td>true</td>
</tr>
<tr>
<td>!==</td>
<td>is not equal (neither value nor type)</td>
<td>x!==&quot;5&quot;</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x!=5</td>
<td>false</td>
</tr>
<tr>
<td>&gt;</td>
<td>is greater than</td>
<td>x&gt;8</td>
<td>false</td>
</tr>
<tr>
<td>&lt;</td>
<td>is less than</td>
<td>x&lt;8</td>
<td>true</td>
</tr>
<tr>
<td>&gt;=</td>
<td>is greater than or equal to</td>
<td>x&gt;=8</td>
<td>false</td>
</tr>
<tr>
<td>&lt;=</td>
<td>is less than or equal to</td>
<td>x&lt;=8</td>
<td>true</td>
</tr>
</tbody>
</table>

Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
<td>(x &lt; 10 &amp;&amp; y &gt; 1) is true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>not</td>
<td>!(x==y) is true</td>
</tr>
</tbody>
</table>

Conditional Operator

`variableName = (condition ? value1 : value2)`

Conditional Statements

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```plaintext
if (condition) {
    code to be executed if condition is true
}
```
If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax
if (condition)
{
    code to be executed if condition is true
} else
{
    code to be executed if condition is not true
}

If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax
if (condition1)
{
    code to be executed if condition1 is true
} else if (condition2)
{
    code to be executed if condition2 is true
} else
{
    code to be executed if neither condition1 nor condition2 is true
}

JavaScript Switch Statement

Syntax
switch(n)
{
    case 1:
        execute code block 1
        break;
    case 2:
        execute code block 2
        break;
    default:
        code to be executed if n is different from case 1 and 2
}
Loop

1. The For Loop
for (statement 1; statement 2; statement 3)
{
    the code block to be executed
}

The For/In Loop

var person={fname:"John",lname:"Doe",age:25};
for (x in person)
{
    txt=txt + person[x];
}

The While Loop

while (condition)
{
    code block to be executed
}

Break and continue statement is same as C language

JavaScript Functions

function functionname()
{
    some code to be executed
}

e.g

<html>
<head>

Calling a Function with Arguments

Syntax

function myFunction(var1, var2)
{
    some code
}

e.g

<html>
<head>
    <script>
    function myFunction(str)
    {
        document.write(str)
    }
    </script>
</head>

<body>
    <button onclick="myFunction("hello")">Try it</button>
</body>
</html>

Functions With a Return Value

<html>
<body>

<p>This example calls a function which performs a calculation, and returns the result:</p>
JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

1. Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

```html
<html>
<head>
<script>
function myFunction() 
{
    alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()" value="Show alert box">
</body>
</html>
```

2 Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

```html
<script>
function myFunction(a,b)
{
    return a*b;
}
document.getElementById("demo").innerHTML=myFunction(4,3);
</script>
</body>
</html>
```
If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false

```
<html>
<head>
<script>
function myFunction()
{
    var r=confirm("Press a button");
    if (r==true)
        {x="You pressed OK!";}
    else
        {x="You pressed Cancel!";}
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()" value="Show confirm box">
</body>
</html>

3. Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

```
<html>
<head>
<script>
function myFunction()
{
    var name=prompt("Please enter your name","amol");
    document.write(name);
}
</script>
</head>
<body>
</body>
```
JavaScript Object

1. JavaScript Array Object

```html
<html>
<head>
<script>
function myFunction() {

    var mycars = new Array();
    mycars[0] = "Saab";
    mycars[1] = "Volvo";
    mycars[2] = "BMW";

    for(i=0;i<mycars.length;i++) {
        document.write(mycars[i]);
    }
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()" value="Show prompt box">
</body>
</html>
```

2. String Object

```html
<html>
<head>
<script>
function myFunction() {
    var txt="Hello World!"; 
    document.write(txt.length);
}
</script>
</head>
<body>
</body>
</html>
```
3. Date Object

```html
<html>
<head>
<script>
function myFunction()
{
    var today = new Date()
    document.write(today);
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()">
</body>
</html>
```

4. Math Object

```html
<html>
<head>
<script>
function myFunction()
{
    var y=Math.sqrt(16);
    document.write(y);
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()">
</body>
</html>
```
2. Browser Object

JavaScript support a number of browser-based objects, when a webpage loads by a browser that support JavaScript, the browser creates a variety of Java Script objects. This make possible to access the web page and handle elements like HTML. There are number of browser objects for JavaScript.

It can be divided into 2
1. Document Object Model (DOM)
2. Browser Object Model (BOM)

1. The Browser Object Model (BOM)
It is collection of objects that interact with the browser window. The top level of the hierarchy is the window object, which contains the information about the window displaying document. Some of the window objects are objects themselves that describe the document and related information. The important ones are

1. Location object
2. History object
3. Document object
4. Navigator object
5. Screen object

The Window Object

The window object is supported by all browsers. It represent the browsers window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Window Properties
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td>Returns a Boolean value indicating whether a window has been closed or not</td>
</tr>
<tr>
<td>defaultStatus</td>
<td>Sets or returns the default text in the statusbar of a window</td>
</tr>
<tr>
<td>document</td>
<td>Returns the Document object for the window (See Document object)</td>
</tr>
<tr>
<td>frames</td>
<td>Returns an array of all the frames (including iframes) in the current window</td>
</tr>
<tr>
<td>history</td>
<td>Returns the History object for the window (See History object)</td>
</tr>
<tr>
<td>innerHeight</td>
<td>Sets or returns the inner height of a window's content area</td>
</tr>
<tr>
<td>innerWidth</td>
<td>Sets or returns the inner width of a window's content area</td>
</tr>
<tr>
<td>length</td>
<td>Returns the number of frames (including iframes) in a window</td>
</tr>
<tr>
<td>location</td>
<td>Returns the Location object for the window (See Location object)</td>
</tr>
<tr>
<td>name</td>
<td>Sets or returns the name of a window</td>
</tr>
<tr>
<td>navigator</td>
<td>Returns the Navigator object for the window (See Navigator object)</td>
</tr>
<tr>
<td>opener</td>
<td>Returns a reference to the window that created the window</td>
</tr>
<tr>
<td>outerHeight</td>
<td>Sets or returns the outer height of a window, including toolbars/scrollbars</td>
</tr>
<tr>
<td>outerWidth</td>
<td>Sets or returns the outer width of a window, including toolbars/scrollbars</td>
</tr>
<tr>
<td>pageXOffset</td>
<td>Returns the pixels the current document has been scrolled (horizontally) from upper left corner of the window</td>
</tr>
<tr>
<td>pageYOffset</td>
<td>Returns the pixels the current document has been scrolled (vertically) from upper left corner of the window</td>
</tr>
<tr>
<td>parent</td>
<td>Returns the parent window of the current window</td>
</tr>
<tr>
<td>screen</td>
<td>Returns the Screen object for the window (See Screen object)</td>
</tr>
<tr>
<td>screenLeft</td>
<td>Returns the x coordinate of the window relative to the screen</td>
</tr>
<tr>
<td>screenTop</td>
<td>Returns the y coordinate of the window relative to the screen</td>
</tr>
<tr>
<td>screenX</td>
<td>Returns the x coordinate of the window relative to the screen</td>
</tr>
<tr>
<td>screenY</td>
<td>Returns the y coordinate of the window relative to the screen</td>
</tr>
<tr>
<td>self</td>
<td>Returns the current window</td>
</tr>
<tr>
<td>status</td>
<td>Sets the text in the statusbar of a window</td>
</tr>
<tr>
<td>top</td>
<td>Returns the topmost browser window</td>
</tr>
</tbody>
</table>
## Window Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert()</td>
<td>Displays an alert box with a message and an OK button</td>
</tr>
<tr>
<td>blur()</td>
<td>Removes focus from the current window</td>
</tr>
<tr>
<td>clearInterval()</td>
<td>Clears a timer set with setInterval()</td>
</tr>
<tr>
<td>clearTimeout()</td>
<td>Clears a timer set with setTimeout()</td>
</tr>
<tr>
<td>close()</td>
<td>Closes the current window</td>
</tr>
<tr>
<td>confirm()</td>
<td>Displays a dialog box with a message and an OK and a Cancel button</td>
</tr>
<tr>
<td>createPopup()</td>
<td>Creates a pop-up window</td>
</tr>
<tr>
<td>focus()</td>
<td>Sets focus to the current window</td>
</tr>
<tr>
<td>moveBy()</td>
<td>Moves a window relative to its current position</td>
</tr>
<tr>
<td>moveTo()</td>
<td>Moves a window to the specified position</td>
</tr>
<tr>
<td>open()</td>
<td>Opens a new browser window</td>
</tr>
<tr>
<td>print()</td>
<td>Prints the content of the current window</td>
</tr>
<tr>
<td>prompt()</td>
<td>Displays a dialog box that prompts the visitor for input</td>
</tr>
<tr>
<td>resizeBy()</td>
<td>Resizes the window by the specified pixels</td>
</tr>
<tr>
<td>resizeTo()</td>
<td>Resizes the window to the specified width and height</td>
</tr>
<tr>
<td>scroll()</td>
<td></td>
</tr>
<tr>
<td>scrollBy()</td>
<td>Scrolls the content by the specified number of pixels</td>
</tr>
<tr>
<td>scrollTo()</td>
<td>Scrolls the content to the specified coordinates</td>
</tr>
<tr>
<td>setInterval()</td>
<td>Calls a function or evaluates an expression at specified intervals (in milliseconds)</td>
</tr>
<tr>
<td>setTimeout()</td>
<td>Calls a function or evaluates an expression after a specified number of milliseconds</td>
</tr>
</tbody>
</table>

## Screen Object

The screen object contains information about the visitor's screen.

### Screen Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>availHeight</td>
<td>Returns the height of the screen (excluding the Windows Taskbar)</td>
</tr>
<tr>
<td>availWidth</td>
<td>Returns the width of the screen (excluding the Windows Taskbar)</td>
</tr>
<tr>
<td>colorDepth</td>
<td>Returns the bit depth of the color palette for displaying images</td>
</tr>
<tr>
<td>height</td>
<td>Returns the total height of the screen</td>
</tr>
<tr>
<td>pixelDepth</td>
<td>Returns the color resolution (in bits per pixel) of the screen</td>
</tr>
<tr>
<td>width</td>
<td>Returns the total width of the screen</td>
</tr>
</tbody>
</table>

## History Object

The history object contains the URLs visited by the user (within a browser window).

The history object is part of the window object and is accessed through the window.history property.
History Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Returns the number of URLs in the history list</td>
</tr>
</tbody>
</table>

History Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>back()</td>
<td>Loads the previous URL in the history list</td>
</tr>
<tr>
<td>forward()</td>
<td>Loads the next URL in the history list</td>
</tr>
<tr>
<td>go()</td>
<td>Loads a specific URL from the history list</td>
</tr>
</tbody>
</table>

Location Object

The location object contains information about the current URL.

The location object is part of the window object and is accessed through the window.location property.

Location Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash</td>
<td>Returns the anchor portion of a URL</td>
</tr>
<tr>
<td>host</td>
<td>Returns the hostname and port of a URL</td>
</tr>
<tr>
<td>hostname</td>
<td>Returns the hostname of a URL</td>
</tr>
<tr>
<td>href</td>
<td>Returns the entire URL</td>
</tr>
<tr>
<td>pathname</td>
<td>Returns the path name of a URL</td>
</tr>
<tr>
<td>port</td>
<td>Returns the port number the server uses for a URL</td>
</tr>
<tr>
<td>protocol</td>
<td>Returns the protocol of a URL</td>
</tr>
<tr>
<td>search</td>
<td>Returns the query portion of a URL</td>
</tr>
</tbody>
</table>

Location Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign()</td>
<td>Loads a new document</td>
</tr>
<tr>
<td>reload()</td>
<td>Reloads the current document</td>
</tr>
<tr>
<td>replace()</td>
<td>Replaces the current document with a new one</td>
</tr>
</tbody>
</table>

Document Object

Each HTML document loaded into a browser window becomes a Document object.

The Document object provides access to all HTML elements in a page, from within a script.
Document Object Model

The Document Object Model (DOM) is an application programming interface for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. With the DOM model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content.

### Document Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anchors</td>
<td>Returns a collection of all the anchors in the document</td>
</tr>
<tr>
<td>applets</td>
<td>Returns a collection of all the applets in the document</td>
</tr>
<tr>
<td>body</td>
<td>Returns the body element of the document</td>
</tr>
<tr>
<td>cookie</td>
<td>Returns all name/value pairs of cookies in the document</td>
</tr>
<tr>
<td>documentMode</td>
<td>Returns the mode used by the browser to render the document</td>
</tr>
<tr>
<td>domain</td>
<td>Returns the domain name of the server that loaded the document</td>
</tr>
<tr>
<td>forms</td>
<td>Returns a collection of all the forms in the document</td>
</tr>
<tr>
<td>images</td>
<td>Returns a collection of all the images in the document</td>
</tr>
<tr>
<td>lastModified</td>
<td>Returns the date and time the document was last modified</td>
</tr>
<tr>
<td>links</td>
<td>Returns a collection of all the links in the document</td>
</tr>
<tr>
<td>readyState</td>
<td>Returns the (loading) status of the document</td>
</tr>
<tr>
<td>referrer</td>
<td>Returns the URL of the document that loaded the current document</td>
</tr>
<tr>
<td>title</td>
<td>Sets or returns the title of the document</td>
</tr>
<tr>
<td>URL</td>
<td>Returns the full URL of the document</td>
</tr>
</tbody>
</table>

### Document Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>Closes the output stream previously opened with document.open()</td>
</tr>
<tr>
<td>getElementsByName()</td>
<td>Accesses all elements with a specified name</td>
</tr>
<tr>
<td>open()</td>
<td>Opens an output stream to collect the output from document.write() or document.writeln()</td>
</tr>
<tr>
<td>write()</td>
<td>Writes HTML expressions or JavaScript code to a document</td>
</tr>
<tr>
<td>writeln()</td>
<td>Same as write(), but adds a newline character after each statement</td>
</tr>
</tbody>
</table>

The Document Object Model (DOM)
AJAX (Asynchronous JavaScript and XML)

AJAX is about updating parts of a web page, without reloading the whole page. AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

How AJAX Works

Above dig indicates that
1. The web page sends the request using javascript function.
2. The JS code makes a request to the server.
3. The server response comprises of data and not the presentation which implies that the data required by the page is provided by the server as the response, and the presentation is implemented on the data with the help of mark up language.
Every user action that normally would generate an HTTP request takes the from a Java Script call to the AJAX engine instead.

The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object (IE5 and IE6 use an ActiveXObject).

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

All modern browsers (IE7+, Firefox, Chrome, Safari, and Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```javascript
variable=new XMLHttpRequest();
```

Old versions of Internet Explorer (IE5 and IE6) uses an ActiveX Object:

```javascript
variable=new ActiveXObject("Microsoft.XMLHTTP");
if (window.XMLHttpRequest)
{
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
}
else
{
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
```

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```javascript
xmlhttp.open("GET","ajax_info.php",true);
xmlhttp.send();
```
Method | Description
--- | ---
open(*method, url, async*) | Specifies the type of request, the URL, and if the request should be handled asynchronously or not.

- **method**: the type of request: GET or POST
- **url**: the location of the file on the server
- **async**: true (asynchronous) or false (synchronous)

send(*string*) | Sends the request off to the server.

- **string**: Only used for POST requests

Server Response

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseText</td>
<td>get the response data as a string</td>
</tr>
<tr>
<td>responseXML</td>
<td>get the response data as XML data</td>
</tr>
</tbody>
</table>

The onreadystatechange event

When a request to a server is sent, we want to perform some actions based on the response. The onreadystatechange event is triggered every time the readyState changes. The readyState property holds the status of the XMLHttpRequest.
Three important properties of the XMLHttpRequest object:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onreadystatechange</td>
<td>Stores a function (or the name of a function) to be called automatically each time the readyState property changes</td>
</tr>
</tbody>
</table>
| readyState        | Holds the status of the XMLHttpRequest. Changes from 0 to 4:  
|                   | 0: request not initialized  
|                   | 1: server connection established  
|                   | 2: request received  
|                   | 3: processing request  
|                   | 4: request finished and response is ready |
| status 200:       | "OK"  
|                   | 404: Page not found |

```javascript
xmlhttp.onreadystatechange=function()
{
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
  }
}
```

Eg 1.

```html
<html>
<head>
<script>
function showHint(str)
{
  if (str.length===0)
  {
    document.getElementById("txtHint").innerHTML="";
  }

```
if (window.XMLHttpRequest)
{
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
}
else
{
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}

xmlhttp.onreadystatechange=function()
{
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
        document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
}

xmlhttp.open("GET","data.php?q="+str,true);
xmlhttp.send();

</script>
</head>
<body>

Start typing a name in the input field below:<b></b>

<form method="get">
First name: <input type="text" onkeyup="showHint(this.value)">
</form>

Suggestions: <span id="txtHint"></span>

</body>
</html>

**data.php**

```php
<?php
    $q=$_GET['q'];
    $len=strlen($q);
    if ($len > 0)
    {
        echo "length is ".$len;
    }
    else
```
To change the contains

```html
<html>
<head>
<script>
function loadXMLDoc()
{
if (window.XMLHttpRequest)
// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlobot.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlobot.open("GET","data4.php",true);
xmlobot.send();
}
</script>
</head>
<body>
<form>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</form>
<div id="txtHint"></div>
</body>
</html>

data4.php
<?php

echo "<h1>hello</h1>";

?>
<html>
<head>
<script>
function showUser(str)
{
  if (str=="")
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
  else
  {// code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function()
  {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
      document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","data2.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
<option value="" selected>Select a Rollno:</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
</select>
<br>
<div id="txtHint">student info will be listed here.</div>
</form>
</body>
</html>
```php
<?php
$q=$_GET["q"]; if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("tydb", $con);
$sql="SELECT * FROM student WHERE sno=".$q."";
$result = mysql_query($sql);

while($row = mysql_fetch_array($result)) {
    echo "<tr>
    <td>" . $row['sno'] . "</td>
    <td>" . $row['sname'] . "</td>
    <td>" . $row['per'] . "</td>
    </tr>";
}
```

```html
AJAX and XML

```javascript
function showCD(str) {
    if (str=="") {
        document.getElementById("txtHint").innerHTML="";
        return;
    }
    if (window.XMLHttpRequest)
```
data3.php

<?php
Sq=$_GET["q"];  

$xmlDoc = new DOMDocument();
$xmlDoc->load("/var/www/html/cd.xml");

$x=$xmlDoc->getElementsByTagName('ARTIST');
for ($i=0; $i<=$x->length-1; $i++) {
{
//Process only element nodes
if ($x->item($i)->nodeType==1)
{
    if ($x->item($i)->childNodes->item(0)->nodeValue == $q)
    {
        $y=($x->item($i)->parentNode);
    }
}
} }

$cd=($y->childNodes);

for ($i=0;$i<$cd->length;$i++)
{
    //Process only element nodes
    if ($cd->item($i)->nodeType==1)
    {
        echo("<b>" . $cd->item($i)->nodeName . ":" .</b>");
        echo($cd->item($i)->childNodes->item(0)->nodeValue);
        echo("<br>");
    }
} ?>