CH:-4 Introduction to JavaScript and JQuery

What is JavaScript?

- HTML and CSS concentrate on a static rendering of a page; things do not change on the page over time, or because of events.
- To do these things, we use scripting languages, which allow content to change dynamically.
- Not only this, but it is possible to interact with the user beyond what is possible with HTML.
- Scripts are programs just like any other programming language; they can execute on the client side or the server.

Advantages of client side scripting

- The web browser uses its own resources, and eases the burden on the server.
- It has fewer features than server side scripting.
- It saves network bandwidth.

Disadvantages of client side scripting

- Code is usually visible.
- Code is probably modifiable.
- Local files and databases cannot be accessed.
- User is able to disable client side scripting.

Differentiate between server side and client side scripting languages

Client-side scripting languages

- The client-side environment used to run scripts is usually a browser.
- The processing takes place on the end users computer.
- The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.
- The scripting language needs to be enabled on the client computer.
- Sometimes if a user is conscious of security risks they may switch the scripting facility off.
- When this is the case a message usually pops up to alert the user when script is attempting to run.

Server-side scripting languages

- The server-side environment that runs a scripting language is a web server.
- A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages.
- This HTML is then sent to the client browser.
- It is usually used to provide interactive web sites that interface to databases or other data stores on the server.
- This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript.
- The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

What is difference between Java script and JAVA?

- Java is a statically typed language; JavaScript is dynamic.
- Java is class-based; JavaScript is prototype-based.
- Java constructors are special functions that can only be called at object creation; JavaScript "constructors" are just standard functions.
- Java requires all non-block statements to end with a semicolon; JavaScript inserts semicolons at the ends of certain lines.
- Java uses block-based scoping; JavaScript uses function-based scoping.

Java has an implicit this scope for non-static methods, and implicit class scope; JavaScript has implicitglobal scope.

Embedded JavaScript

- JavaScript can be embedded in an HTML document.
- To embed it in HTML you must write:

```
<script type=”text/javascript”>
</script>
```

- The script tag has effect of the stopping the JavaScript being printed out as well as indentifying the code enclosed.
- The JavaScript can be placed in the head section of your HTML or the body.

```
<html>
<body>
<script type=”text/javascript”>
        document.write("<h1>This is a heading</h1>");
</script>
</body>
</html>
```

- The Scripts placed in the body section are executed as the page loads and can be used to generate the content of the page.
- As well as the body section, JavaScript can also be placed in the head part.
- The advantages of putting a script in there are that it loads before the main body.

External JavaScript

- If you want to use the same script on several pages it could be a good idea to place the code in a separate file, rather than writing it on each.
- That way if you want to update the code, or change it, you only need to do it once.
- Simply take the code you want in a separate file out of your program and save it with the extension .js.

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Topic: 1 Client side Scripting with JavaScript: Variables, conditions, Loops, Functions, Pop up Boxes

1.1 JavaScript Variables

- Variables in JavaScript behave the same as variables in most popular programming languages (C, C++, etc) do, but in JavaScript you don't have to declare variables before you use them.
- A variable's purpose is to store information so that it can be used later. A variable is a symbolic name that represents some data that you set.
- When using a variable for the first time it is not necessary to use "**var**" before the variable name.
- Variable names must begin with a letter.
- Variable names are case sensitive (y and Y are different variables).

```
var x=5;
var y=6;
var z=x+y;
```

- You can declare many variables in one statement. Just start the statement with var and separate the variables by comma:

```
var name="Doe", age=30, job="carpenter";
var name="Doe",
age=30,
job="carpenter";
```

- Variable declared without a value will have the value **undefined**.
- If you re-declare a JavaScript variable, it will not lose its value.
- The value of the variable *carname* will still have the value "Volvo" after the execution of the following two statements.

```
varcarname="Volvo";
varcarname;
```

1.2 JavaScript Operators

- Operators in JavaScript are very similar to operators that appear in other programming languages.
- The definition of an operator is a symbol that is used to perform an operation.
- Most often these operations are arithmetic (addition, subtraction, etc), but not always.

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
=	Assignment

```
<body>
<script type="text/JavaScript">
```

```
<!--
var two = 2
var ten = 10
varlinebreak = "<br />"
document.write("two plus ten = ")
var result = two + ten
document.write(result)
//-->
</script>
</body>
```

Assignment	Equivalent to
X+=Y	X=X+Y
X-=Y	X=X-Y
X*=Y	X=X*Y
X/=Y	X=X/Y
X%=Y	X=X%Y

1.3 JavaScript Array

- An array is a special variable, which can hold more than one value at a time.
- The Array object is used to store multiple values in a single variable.
- An array can be created in three ways.
- The following code creates an Array object called myCars.
    1. Regular

       ```
       varmyCars=new Array();
       myCars[0]="Saab";
       myCars[1]="Volvo";
       myCars[2]="BMW";
       ```

    2. Condensed

       ```
       varmyCars=new Array("Saab","Volvo","BMW");
       ```

    3. Literal

       ```
       varmyCars=["Saab","Volvo","BMW"];
       ```

Access an Array

- You refer to an element in an array by referring to the **index** number.
- This statement access the value of the first element in myCars.

```
var name=myCars[0];
```

- This statement modifies the first element in myCars:

```
myCars[0]="Opel";
```

1.4 JavaScript Functions

- A function is a section of code that is separate from the main program.
- It is defined once but can be invoked many times.
- A function can be passed as parameters so that they can be used and a value can be returned back.
- There are some functions already built in to JavaScript, such as the Math.cos() function, which calculates the cosine of an angle.
- An example function could be:

```
functionmultByTen(x)
{
        return x*10;
}
```

- This can then be invoked by using the function's name complete with any parameters you want to pass:

```
mysum=multByTen(3)
```

- Below is an example of JavaScript function.

```
<html><body>
<script   type="text/javascript">
var z= multXbyY(10,15);
document.write("The result is" +z);
functionmultXbyY(x,y) {
        document.write("x is " +x);
        document.write("y  is  "+y);
        return x*y;
}
</script>
</body></html>
```

1.5 JavaScript Conditions

- Conditional statements are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:
  - **if statement** - use this statement to execute some code only if a specified condition is true
  - **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
  - **if...else if ...else statement** - use this statement to select one of many blocks of code to be executed
  - **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

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

```
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.

```
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.

```
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
 }
```

Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.

```
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
}
```

The *default* Keyword

- Use the *default* keyword to specify what to do if there is no match.

Conditional Operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

```
variablename=(condition)?value1:value2
voteable=(age<18)?"Too young":"Old enough";
```

1.6 JavaScript Loops and Repetition

- Loops can execute a block of code a number of times.
- Loops are handy, if you want to run the same code over and over again, each time with a different value.

Different Kinds of Loops

- JavaScript supports different kinds of loops:
  - **for** - loops through a block of code a number of times
  - **for/in**- loops through the properties of an object
  - **while** - loops through a block of code while a specified condition is true
  - **do/while** - also loops through a block of code while a specified condition is true

The For Loop

- The for loop is often the tool you will use when you want to create a loop.
- The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3)
 {
 the code block to be executed
 }
```

- **Statement 1** is executed before the loop (the code block) starts.
- **Statement 2** defines the condition for running the loop (the code block).
- **Statement 3** is executed each time after the loop (the code block) has been executed.

```
for (var i=0; i<5; i++)
 {
 x=x + "The number is " + i + "<br>";
 }
```

The For/In Loop

- The JavaScript for/in statement loops through the properties of an object.

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

The While Loop

- The while loop loops through a block of code as long as a specified condition is true.

```
while (condition)
 {
 code block to be executed
 }
```

The Do/While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do
 {
 code block to be executed
 }
while (condition);
```

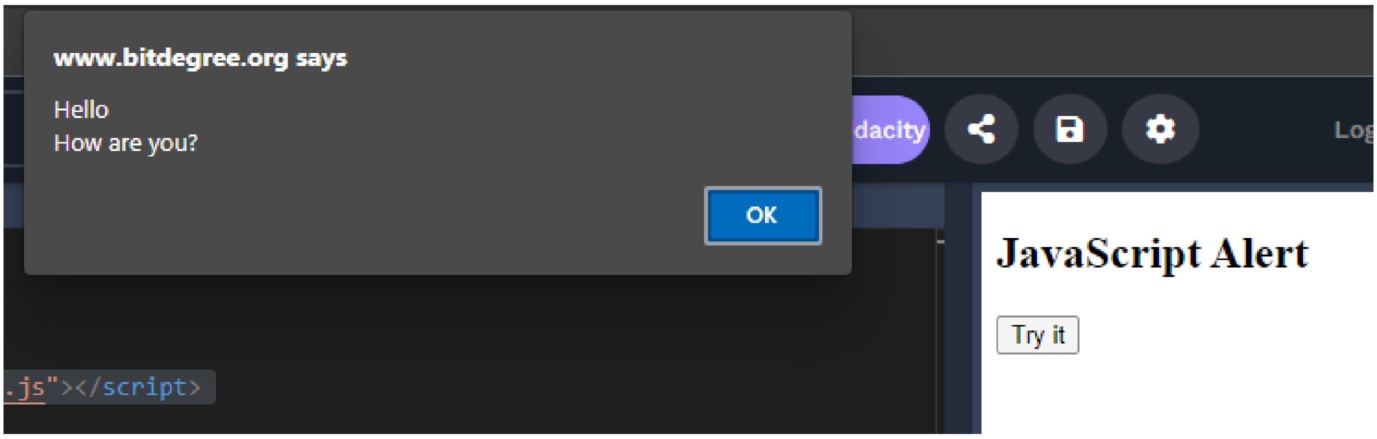
1.7 POP up box

- There are **three types** of JavaScript popup windows: **Alert**, **Confirm** and **Prompt**.
- To write **multiple lines** in a JavaScript popup window, you will need to use "\n" string.
- You **do not need** to use the window prefix when writing pop-up box methods.

Alert Box

- A JavaScript alert box is used when you need some information to reach the user. When the alert box shows up, the user will need to press the OK button to resume activity. It interrupts the user's activity, so you need to be careful when using it.
- You can write the following code to display a JavaScript alert box:
- window.alert("Alert Text!");
- The window prefix is not mandatory. You can write the code without it:
- alert("Alert Text!");
- The example below creates a pop-up alert that contains two lines of text:

```
<html>
<head>
  <script type="text/javascript" src="scripts.js"></script>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h2>JavaScript Alert</h2>
  <button onclick="firstFunction()">Try it</button>
</body>
</html>
function firstFunction() {
    alert("Hello\nHow are you?");
}
```


www.bitdegree.org says

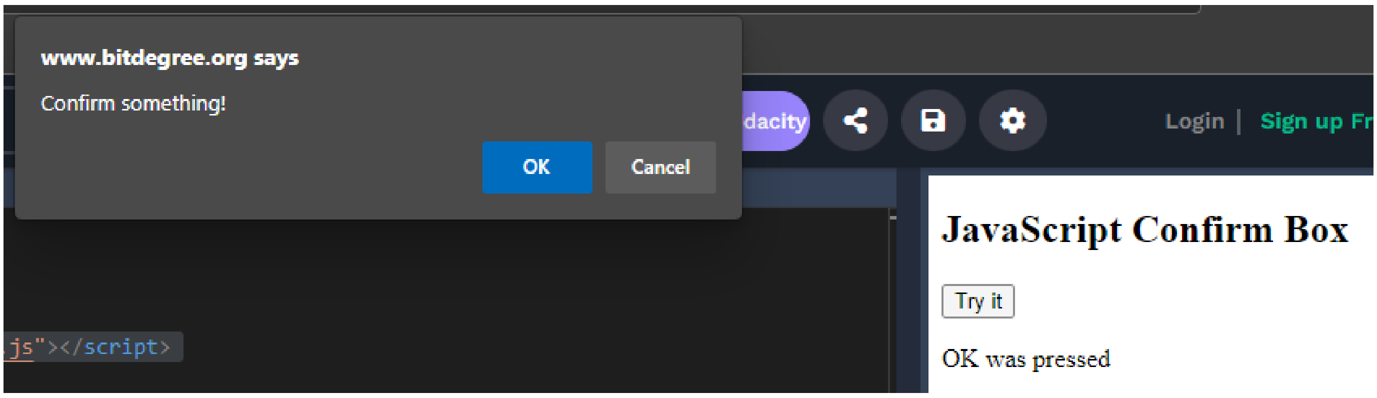
Hello
How are you?

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

Confirm Box

- A confirm pop-up box is used when you need the user to accept something. When the confirm pop-up box opens, the user has to click one of the two buttons (OK or CANCEL) to close the box. OK returns true, while CANCEL returns false.
- You can write the following code to display a confirm box: window.confirm("Confirmation Information");
- The window prefix is not mandatory here as well. You can write the code without it: confirm("Confirmation Information");
- The confirm box returns true if OK button is clicked, and returns false if CANCEL is pressed:

```
<html>
<head>
  <script type="text/javascript" src="scripts.js"></script>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h2>JavaScript Confirm Box</h2>
  <button onclick="firstFunction()">Try it</button>
  <p id="test"></p>
</body>
</html>
function firstFunction() {
   var r = confirm("Confirm something!");
   if (r == true) {
      x = "OK was pressed";
   }
   else {
      x = "Cancel was pressed";
   }
   document.getElementById("test").innerHTML = x;
}
```

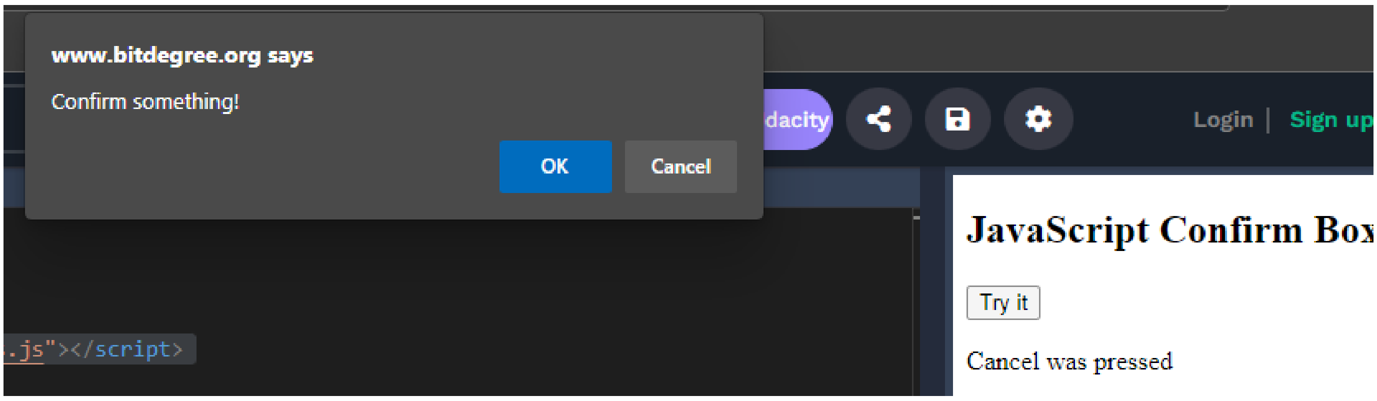

www.bitdegree.org says

Confirm something!

.js"></script>

JavaScript Confirm Box

OK was pressed


www.bitdegree.org says

Confirm something!

.js"></script>

JavaScript Confirm Box

Cancel was pressed

Prompt Box

- The prompt box is used when you need the user to insert some information before proceeding. To turn the prompt box off, the user has to click OK or CANCEL buttons.
- The confirm box returns the input data if the OK button is clicked. It returns null if CANCEL button is clicked.
- You can write the following code to display a prompt box:

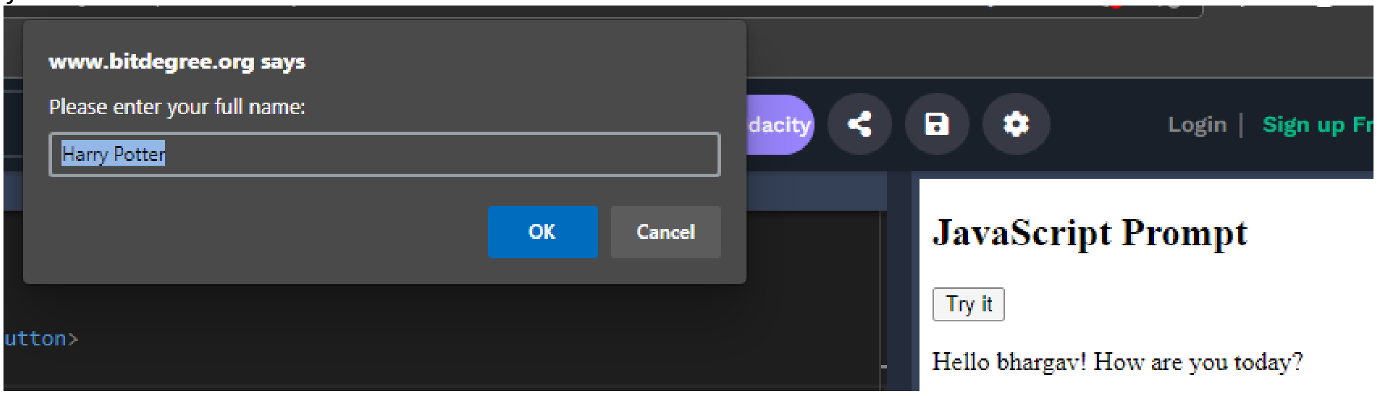
window.prompt("Information Text","Default Text");

- Just as before, the window prefix is not mandatory. You can write the code without it:

prompt("Information Text","Default Text");

- See the example below to get a clearer idea, and click **Try it Live!** to see how it works:

```
<html>
<head>
  <script type="text/javascript" src="scripts.js"></script>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h2>JavaScript Prompt</h2>
  <button onclick="firstFunction()">Try it</button>
  <p id="test"></p>
</body>
</html>
function firstFunction() {
    var txt;
    var person = prompt("Please enter your full name:", "Harry Potter");
    if (person == null || person == "") {
        txt = "User cancelled the prompt.";
    } else {
        txt = "Hello " + person + "! How are you today?"
    }
    document.getElementById("test").innerHTML = txt;
}
```

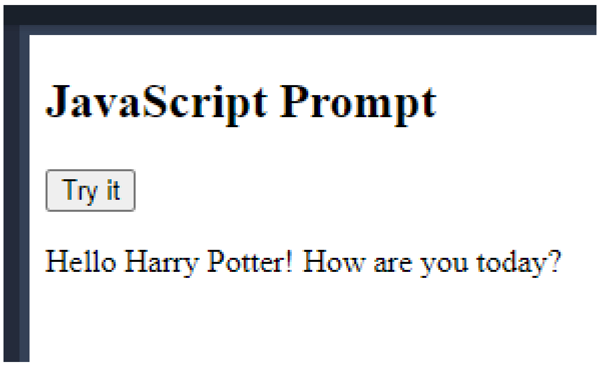

www.bitdegree.org says

Please enter your full name:

Harry Potter

JavaScript Prompt

Hello bhargav! How are you today?


JavaScript Prompt

Hello Harry Potter! How are you today?

<u>Topic: 2 Advance JavaScript : JavaScript and objects</u>

JavaScript has several built-in objects, like String, Date, Array, and more.

- An object is just a special kind of data, with **properties** and **methods**.

Accessing Object Properties

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

objectName.propertyName

- This example uses the length property of the String object to find the length of a string:

var message="Hello World!";
var x=message.length;

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();
```

Creating JavaScript Objects

- With JavaScript you can define and create your own objects.
- There are 2 different ways to create a new object:
    - Define and create a direct instance of an object.
    - Use a function to define an object, then create new object instances.

Creating a Direct Instance

- This example creates a new instance of an object, and adds four properties to it:

```
person=new Object();
person.firstname="Narendra";
person.lastname="Modi";
person.age=24;
person.eyecolor="blue";
```

User- Defined Objects/How user defined objects are created in JavaScript?

- JavaScript allows you to create your own objects.
- The first step is to use the *new* operator.

```
VarmyObj= new Object();
```

- This creates an empty object.
- This can then be used to start a new object that you can then give new properties and methods.
- In object- oriented programming such a new object is usually given a constructor to initialize values when it is first created.
- However, it is also possible to assign values when it is made with literal values.

```
<!DOCTYPE html>
<html>
<body>
<script language=”JavaScript” type=”text/JavaScript”>
person={
                firstname: "Ketan",
                lastname: "Chavda",
                age: 24,
                eyecolor: "blue"
        }
document.write(person.firstname + " is " + person.age + " years old.");
</script>
</body>
</html>
```

How a constructor can be used to populate data in the object?

- A constructor is pre defined method that will initialize your object.
- To do this in JavaScript a function is used that is invoked through the *new* operator.
- Any properties inside the newly created object are assigned using *this* keyword, referring to the current object being created.

```
<!DOCTYPE html>
<html>
<body>
<script>
function person(firstname, lastname, age)
{
        this.firstname = firstname;
        this.lastname  =  lastname;
        this. age = age;
}
var person1=new person("Narendra","Modi",24);
document.write(person1.firstname + “ ”+ person1.lastname +” “+ person1.age);
</script>
</body>
</html>
```

- In above the function *person*becomes the constructor invoked through the *new* keyword on assignment to the *person1* variable.
- Here the values are passed as parameters to the constructor.
- Inside the constructor the *this* keyword takes on the value of the newly created object and therefore applies properties to it.

Topic: 3 DOM

Explain document object in JavaScript.

- There are lots of objects that descend from the *document* object forming a large sub-tree known as the Document Object Model (DOM), which has become standardized.
- The *document* object represents the HTML displayed in window.
- Using this object it is possible to access information about the document itself and also about the information content.
- Using this object it is possible to control certain parameters of the current document like background, foreground and link colors.

```
VarmyObj= new Object();
document.bgColor = “#9F2020”;
documet.fgcolor = “#FAF519”;
```

- It is possible to access any form information in document by using the *form[]* array.
- This contains all the *form* objects that are in the document.
- For example, a form can be constructed with :

```
<form name=”userDetails”>
<input type=”text” name= “ fname“/>
<input type=”text” name= “lname“/>
<input type=”submit” name= “ Submit“/>
```

- The form data can then be accessed with various DOM syntax constructions. The form itself can be accessed through:

  document.forms[0];

- It can also referred to by

  document.userDetails

- An individual element can then be accessed:

  document.userDetails.fname

- Below example shows the use of *document* object.

```
<html><head>
<title>Document Object Example</title>
<script type="text/javascript">
functionincrementCurrent() {
current = parseInt(document.forms["noteForm"].total.value);
document.forms["noteForm"].total.value = current + 1;
   }
</script>
</head><body>
<div id="mainDiv">
<h1>Document Object Example</h1>
<form id="noteForm">
            Current number of notes:
      <input type="text" name="total" value="0" size="3"/>
      <input type="button" value="Add a new note"
onclick="incrementCurrent()"/>
</form>
</div>
</body></html>
```

Topic: 4 Forms and Validations

Why do you need validation? Show the use of regular expression in JavaScript to validate the email address with example.

- The idea behind JavaScript form validation is to provide a method to check the user enteredinformation before they can even submit it.
- JavaScript also lets you display helpful alerts to inform the user what information they have enteredincorrectly and how they can fix it.

JavaScript Email Address validation using Regular Expression

- Using Regular expression we do checking for valid information.

```
<html>
<head>
<title>JavaScript Forms</title>
</head>
<body>
<form method=”post” name=”getinfo” onSubmit=”return processForm()”>
<input type=”text” name=”email”/>
<input type=”submit” value=”log in” name=”Login”/>
</form>
<script language=”JavaScript”
type=”text/JavaScript”>
functionprocessForm()
{
varmyform=
document.getinfo;
var check=
myform.email.value;
Document.write(test
Email(check));
}
Function testEmail(chkMail)
{
varemailpattern = “^*\\w-_\.]*[\\w-
_\.]\@[\\w]\.+[\\w]+[\\w+$”;var regex = new
RegExp(emailpattern); returnregex.test(chkMail);
}
</script>
</body>
</html>
```

- This will check for a valid email as describe.
- The testEmail() function returns true or false.
- The way it determines this end result is built on the template/pattern in the string *emailpattern.*
- This is used to work out the order of expected characters, how many times they

repeat and speciallyoccurring punctuation.

- The string in this case that is used as template is:

  “^*\\w-_\.][\\w-_\.]\@[\\w]\.+[\\w]+[\\w+$”*

- The first section is:

  ^[\\w-_\.]

- This sequence, beginning with ^, means check the first character is a word character represented by
  \\w.
- The next part is :

  **[\\w-_\.]*

- The * means that the next series of characters described can be represented many times or not atall.
- The characters themselves are the same as before; that is word characters, underscore, hyphen orperiod.

  \@[\\w]\.+
- This section begins by checking for the @ character.
- Following this should ne the word characters and then at least one ‘dot’.
- In other words it would not accept a dot straight after the @ character.
- The last part is :

  [\\w]+[\\w]$
- The first set in square brackets makes sure that there are some characters after the dot and the lastpart checks that the last character is a word character.
- In this program after the string is declared, a regular expression object is created with the pattern:

  var regex = new RegExp(emailpattern);
- The pattern can then be tested against the incoming parameter with object’s test method:

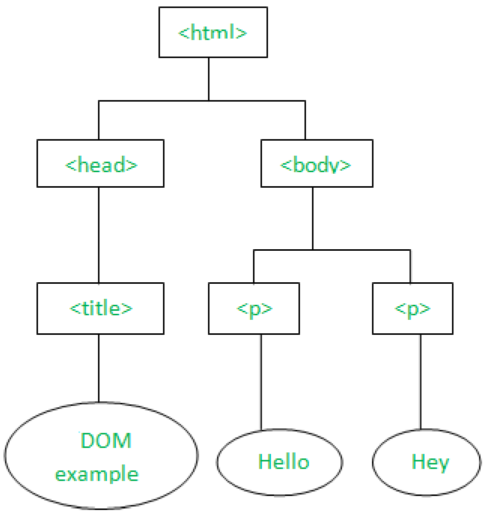
  returnregex.test(chkMail);
- This will return true or false depending on whether there is a match or not.

Topic: 5 DHTML: Combining HTML , CSS and JavaScript

- DHTML stands for Dynamic HTML. Dynamic means that the content of the web page can be customized or changed according to user inputs i.e. a page that is interactive with the user. In earlier times, HTML was used to create a static page. It only defined the structure of the content that was displayed on the page. With the help of CSS, we can beautify the HTML page by changing various properties like text size, background color etc. The HTML and CSS could manage to navigate between static pages but couldn’t do anything else. If 1000 users view a page that had their information for eg. Admit card then there was a problem because 1000 static pages for this application build to work. As the number of users increase, the problem also increases and at some point it becomes impossible to handle this problem.
- To overcome this problem, DHTML came into existence. DHTML included JavaScript along with HTML and CSS to make the page dynamic. This combo made the web pages dynamic and eliminated this problem of creating static page for each user. To integrate JavaScript into HTML, a Document Object Model(DOM) is made for the HTML document. In DOM, the document is represented as nodes and objects which are accessed by different languages

like JavaScript to manipulate the document.

```
<html>
<head>
<title>
 DOM example
</title>
</head>
<body>
<p id = "para1">Hello</p>
<p id =  “para2”>Hey</p>
</body>
</html>
```



HTML document include JavaScript:: The JavaScript document is included in our html page using the html tag. <src> tag is used to specify the source of external JavaScript file.
Following are some of the tasks that can be performed with JavaScript:
Performing html tasks
Performing CSS tasks
Handling events
Validating inputs

Topic: 6 Events and Buttons

Explain the event handling in JavaScript.

- Event handlers are attributes that force an element to "listen" for a specific event to occur.
- Event handlers all begin with the letters "on".
- There are two types of events in Javascript
  - Interactive i.g. onClick
  - Non-interactive i.g. onLoad
- The table below lists the HTML event handlers with descriptions.

Event Handler	Elements Supported	Description
Onblur	a, area, button, input, label, select, textarea	the element lost the focus
Onchange	input, select, textarea	the element value was changed
Onclick	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was clicked
Ondblclick	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was double clicked

Onfocus	a, area, button, input, label, select, textarea	the element received the focus
Onkeydown	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a key was pressed down
Onkeypress	All elements except br, font, frame, frameset,	a key was pressed and released
	head, html, iframe, isindex, meta, param, script, style, title	
Onkeyup	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a key was released
Onload	Frameset	all the frames have been loaded
Onload	Body	the document has been loaded
onmousedown	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was pressed down
onmousemove	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved within.
onmouseout	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved away
onmouseover	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer was moved onto
onmouseup	All elements except br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title	a pointer button was released
Onreset	Form	the form was reset
Onselect	input, textarea	some text was selected
Onsubmit	Form	the form was submitted
Onunload	Frameset	all the frames have been removed
Onunload	Body	the document has been removed

Buttons

- JavaScript button is one of the JavaScript element which gives effects to web pages. JavaScript buttons give a good look and feel to the website. These JavaScript buttons can be used to send data or receive data, fire click events or change the color or text, etc.
- HTML tag <button> is used in JavaScript frameworks which define a clickable button. Each time a button is being rendered onto the web page, an event is fired to perform a functionality. We shall look into creation of JavaScript buttons by using createElement() and also by using HTML tag which is used for JavaScript frameworks.

<u>syntax</u>

Using <button> HTML tag for JavaScript Buttons

```
<button onclick="sampleFunction()">Sample Text</button>
```

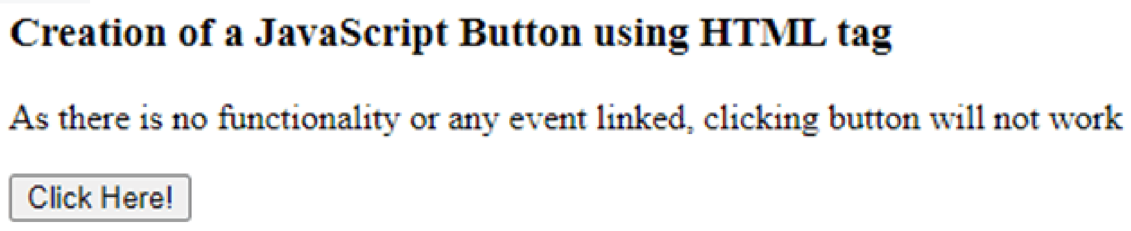
Above is the syntax mostly used in JavaScript Frameworks like ReactJs, AngularJs, etc

```
varsampleButton = document.createElement(btn);
```

Above is the Pure JavaScript Syntax used to create a JavaScript button.

```
<!DOCTYPE html>
```

```
<html>
<body>
<h3>Creation of a JavaScript Button using HTML tag</h3>
<p>As there is no functionality or any event linked, clicking button will not work</p>
<button type="button" id="btn">Click Here!</button>
<p id="samplebtn"></p>
<script>
</script>
</body>
</html>
```


Creation of a JavaScript Button using HTML tag

As there is no functionality or any event linked, clicking button will not work

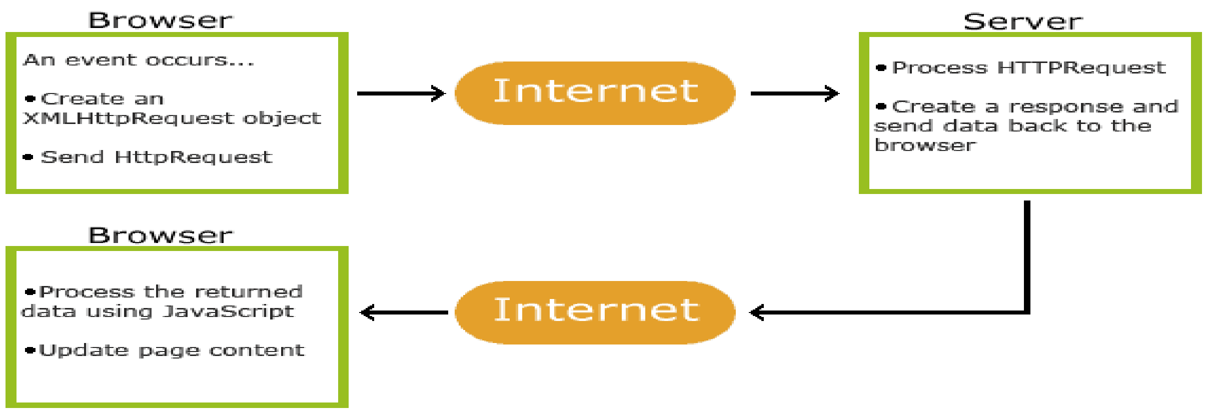
- So here in the above example, we are just creating a button using HTML <button> tag with an id. Clicking won't work as there is no event handler or functionality applicable.

MORE EXAMPLE: **JavaScript Button | Syntax and Examples of Java Script Button (educba.com)**

Topic: 7 AJAX: Introduction

- AJAX = Asynchronous JavaScript and XML.
- 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.

HOW AJAX WORKS?


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

- AJAX makes use of a browser built-in **XMLHttpRequest object** to request data from a Web Server and **HTML DOM** to display or use the data.
- **XMLHttpRequest Object** : It is an API in the form an object whose methods help in transfer of data between a web browser and a web server.
- **HTML DOM** : When a web page is loaded, the browser creates a Document Object Model of the page.

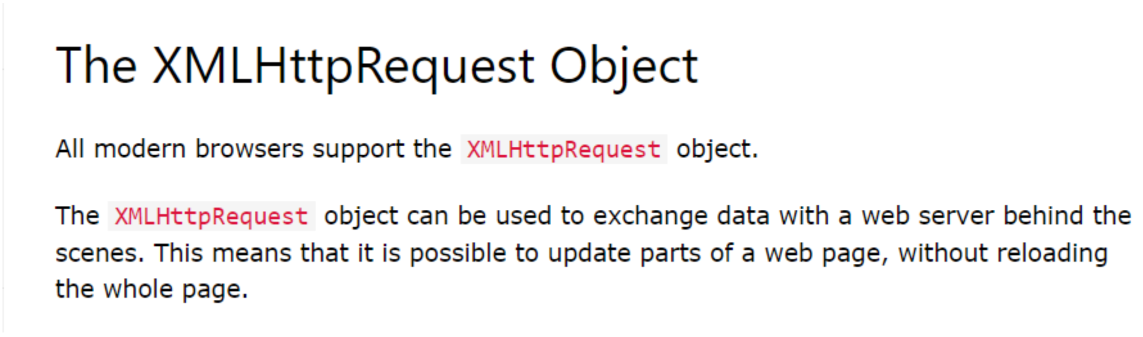
AJAX - The XMLHttpRequest Object

- An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Create an XMLHttpRequest object
2. Define a callback function
3. Open the XMLHttpRequest object
4. Send a Request to a server

Create an XMLHttpRequest object

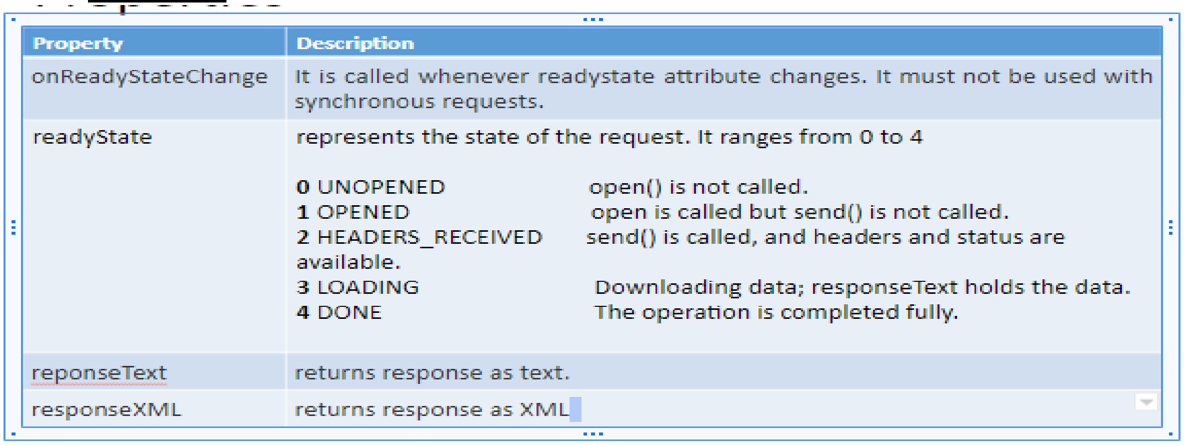


The XMLHttpRequest Object

All modern browsers support the `XMLHttpRequest` object.

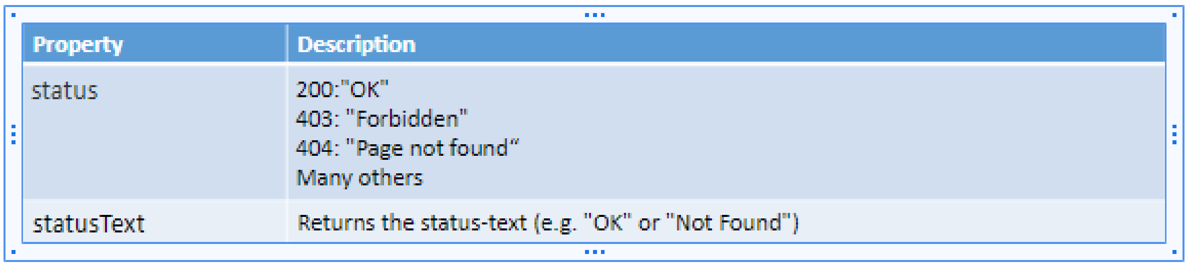
The `XMLHttpRequest` object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Properties



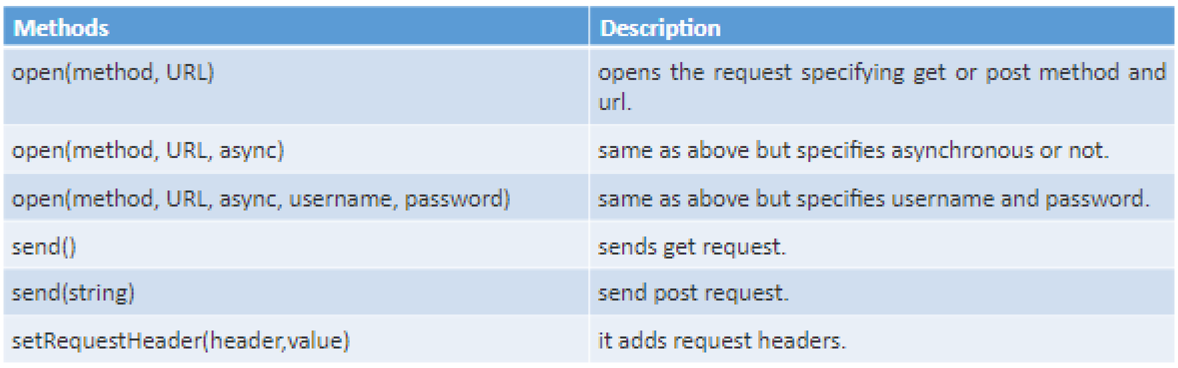
Property	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	represents the state of the request. It ranges from 0 to 4 **0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	returns response as text.
responseXML	returns response as XML

Response properties



Property	Description
status	200:"OK" 403: "Forbidden" 404: "Page not found" Many others
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Methods



Methods	Description
open(method, URL)	opens the request specifying get or post method and url.
open(method, URL, async)	same as above but specifies asynchronous or not.
open(method, URL, async, username, password)	same as above but specifies username and password.
send()	sends get request.
send(string)	send post request.
setRequestHeader(header,value)	it adds request headers.

Define a Callback Function

- There are several response attributes defined by the standard specification for the *XMLHTTPRequest()* constructor.
- *responseText:* It returns the server response as a JavaScript string.
- *responseXML:* It returns the server response as an XML DOM object.
- These attributes tell the client making the *XMLHttpRequest* important information about the status of the response.
- Some cases where dealing with non-text response types may involve some manipulation.

Create an XMLHttpRequest Object

Before performing ajax communication between client & server, an XMLHTTPRequest object is initialized.

var request = new XMLHttpRequest();

Then, the newly created request object using the open() method of XMLHTTPRequest object is instantized.

The open() method accepts two parameters--- the http request method to use & url to send the request

request.open("GET", "info.txt");

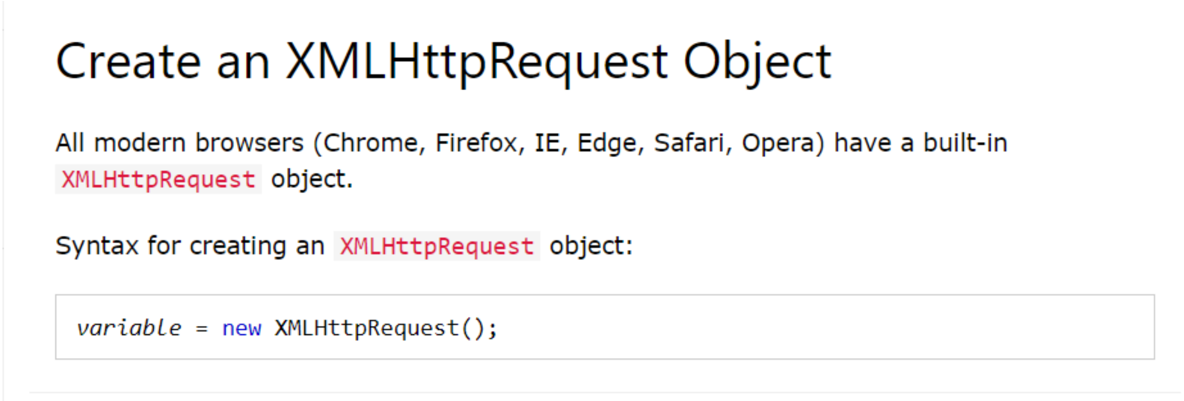
request.open("POST", "info.txt");

Finally sends the request to server using the send() method of the XMLHTTPRequest object.

request.send();

request.send(body);

Here, the *body* parameter allows us to specify the request's body. This is primarily used for HTTP POST requests.



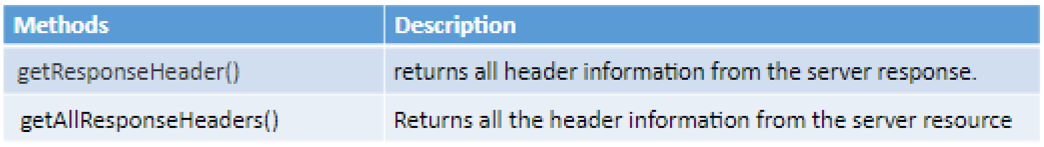
Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE, Edge, Safari, Opera) have a built-in `XMLHttpRequest` object.

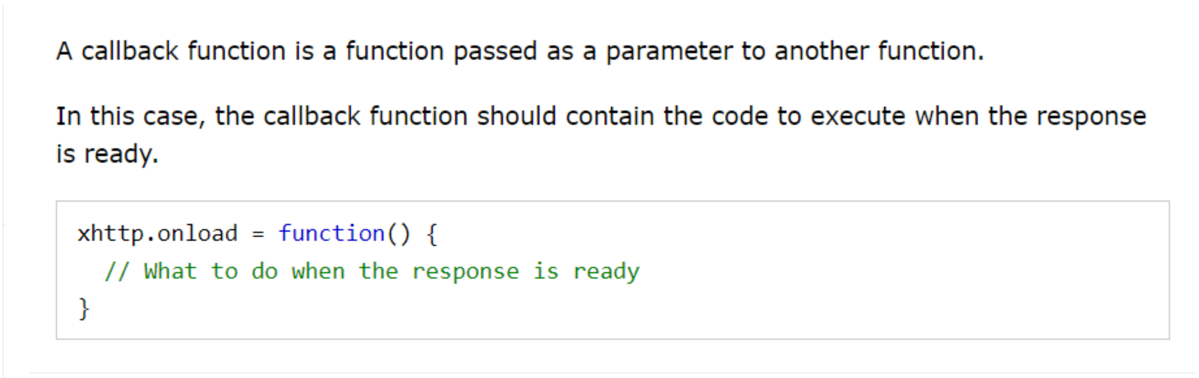
Syntax for creating an `XMLHttpRequest` object:

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

Response method



Methods	Description
getResponseHeader()	returns all header information from the server response.
getAllResponseHeaders()	Returns all the header information from the server resource

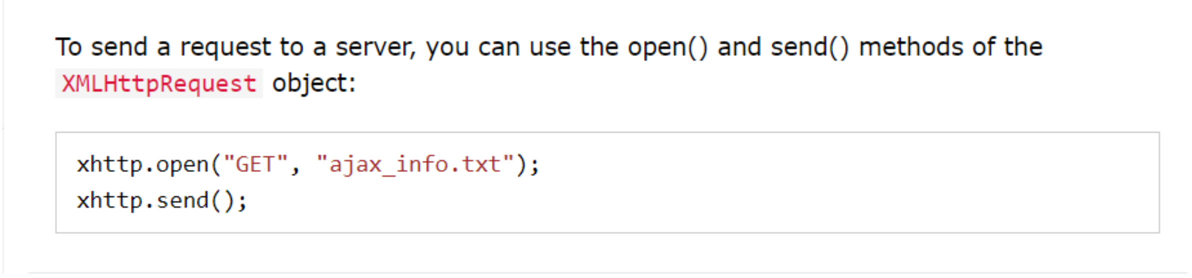


A callback function is a function passed as a parameter to another function.

In this case, the callback function should contain the code to execute when the response is ready.

```
xhttp.onload = function() {
  // What to do when the response is ready
}
```

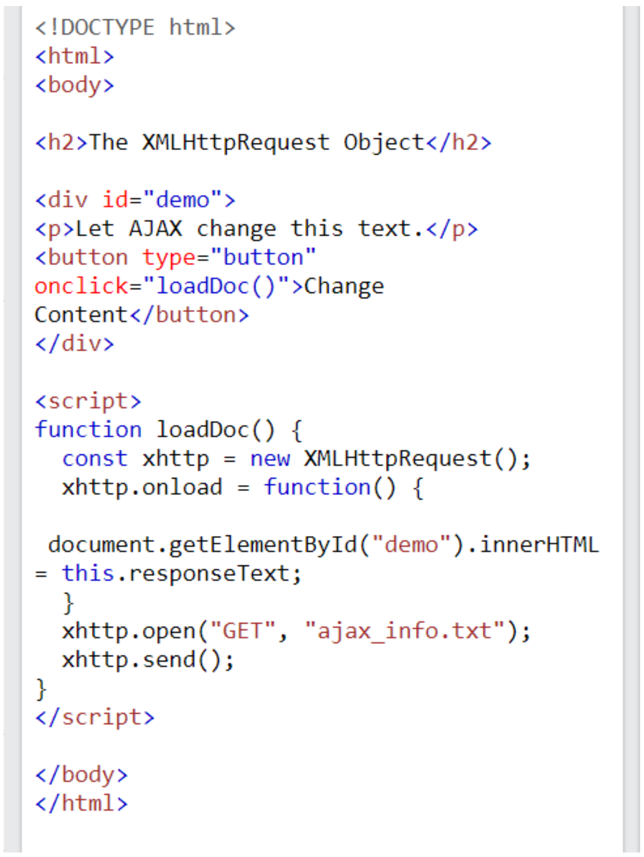
Send a Request



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

```
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Example



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

<h2>The XMLHttpRequest Object</h2>

<div id="demo">
<p>Let AJAX change this text.</p>
<button type="button"
onclick="loadDoc()">Change
Content</button>
</div>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {

 document.getElementById("demo").innerHTML
= this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>

</body>
</html>
```

The XMLHttpRequest Object

Let AJAX change this text.

Change Content

GET & POST METHOD

- The GET method is generally used to send small amount of data to the server. Whereas, the POST method is used to send large amount of data, such as form data.
- In GET method, the data is sent as URL parameters. But, in POST method, the data is sent to the server as a part of the HTTP request body. Data sent through POST method will not visible in the URL.

GET method

```
<html>
<body>
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>
<p id="demo"></p>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {   document.getElementById("demo").innerHTML = this.responseText; }
  xhttp.open("GET", "demo_get.asp");
  xhttp.send();
}
</script>
</body>
</html>
```

POST method

```
<html>
<body>
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>
<p id="demo"></p>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {    document.getElementById("demo").innerHTML = this.responseText; }
  xhttp.open("POST", "demo_post.asp");
  xhttp.send();
}
</script>
</body>
</html>
```

Topic: 8 AJAX Based WebApplications

jQuery ajax() Method

- The jQuery ajax() method provides core functionality of Ajax in jQuery. It sends asynchronous HTTP requests to the server.

  $.ajax(url);

  $.ajax(url, data, callback);

url: A string URL to which you want to submit or retrieve the data

- options: Configuration options for Ajax request. An options parameter can be specified using JSON format. This parameter is optional.

Send Ajax Request

The ajax() methods performs asynchronous http request and gets the data from the server.
The following example shows how to send a simple Ajax request.
Example:

```
$.ajax('/jquery/getdata', {
        success: function(data,status,xhr{
                $('p').append('data');
        })
})
```

Topic: 9 JQuery: Introduction

Javascript and Jquery

- The purpose of jQuery is to make it much easier to use JavaScript on your website. ...jQuery is a lightweight, "write less, do more", JavaScript library.

Why JQuery?

- jQuery is ideal because it can create impressive animations and interactions. jQuery is simple to understand and easy to use, which means the learning curve is small, while the possibilities are (almost) infinite.

List of important core features supported by JQuery

- **DOM manipulation** - DOM stands for Document Object Model and is a mechanism for representing and interacting with your HTML, It allows you to navigate and manipulate your documents through a programming language, which in the browser will almost always be JavaScript.
- **Event handling** - The jQuery offers an elegant way to capture a wide variety of events, such as a Input Events, Mouse Events, Click Events, Load Events, Others.
- **AJAX Support** - The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
- **Animations** - The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- **Lightweight** - The jQuery is very lightweight library.
- **Latest Technology** - The jQuery supports CSS3 selectors and basic XPath syntax.

How to use jQuery?

- There are two ways to use jQuery.
    1. **Local Installation** - You can download jQuery library on your local machine and include it in your HTML code.
    2. **CDN Based Version** - You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Local Installation

```
<html>
<head>
<title>The jQuery Example</title>
<script type = "text/javascript" src = "/jquery/jquery-2.1.3.min.js"></script>
<script type = "text/javascript">
$(document).ready(function(){ • document.write("Hello, World!"); • });
</script>
</head>
<body>
<h1>Hello</h1>
</body>
</html>
```

CDN Based Version

```
<html>
<head>
<title>The jQuery Example</title>
<script type = "text/javascript" •
src"http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type = "text/javascript">
$(document).ready(function(){ • document.write("Hello, World!"); • }); • </script>
</head>
<body>
<h1>Hello</h1>
</body>
</html>
```

DOM Manipulation Methods

- Lists down all the methods which you can use to manipulate DOM elements.
- after(content) -Insert content after each of the matched elements.
- append(content)- Append content to the inside of every matched element.
- html(val)- Set the html contents of every matched element.
- html(val) - Set the html contents of every matched element.
- More

JQuery UI Widget

- Using plug-in, we can apply behaviours to the elements. However, plug-ins lack some built-in capabilities, such as a way to associate data with its elements, expose methods, merge options with defaults, and control the plug-in's lifetime.
- Widgets Description
- Accordion
- Button
- Datepicker
- Menu
- Progressbar
- Slider
- Spinner
- Tabs
- Tooltip

JQuery - Plugins

- A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.
- There are plenty of jQuery plug-in available which you can download from repository link at http//jquery.com/plugins.

jQuery Get Method

- Sends asynchronous HTTP request to the server and retrieves the data.

Syntax:

```
$.get(url, data, callback);
```

- url: request url from which you want to retrieve the data
- data: data to be sent to the server with the request as a query string
- callback: function to be executed when request succeeds

Example

```
$.get('/data.txt',
    function (data, textStatus, jqXHR) {
        alert('status: ' + textStatus + ', data:' + data);
  })
```

jQuery Post Method

- The jQuery post() method sends asynchronous http POST request to the server to submit the data to the server and get the response.

Syntax:

```
$.post(url,data,callback,type);
```

- url: request url from which you want to submit & retrieve the data.
- data: json data to be sent to the server with request as a form data.
- callback: function to be executed when request succeeds.
- type: data type of the response content.

Example

```
$.post('/jquery/submitData',
     { myData: 'This is my data.' },
     function(data, status, jqXHR) {
            $('p').append('status: ' + status + ', data: ' + data);
      })
```

jQuery load Method

- The jQuery load() method allows HTML or text content to be loaded from a server and added into a DOM element.
- Syntax:
- $.load(url,data,callback);
- url: request url from which you want to retrieve the content
- data: JSON data to be sent with request to the server.
- callback: function to be executed when request succeeds
- Example: $('#msgDiv').load('/demo.html');
- If no element is matched by the selector then Ajax request will not be sent.
- The load() method allows us to specify a portion of the response document to be inserted into DOM element.
- This can be achieved using url parameter, by specifying selector with url separated by one or multiple space characters
- Example: $('#msgDiv').load('/demo.html #myHtmlContent');
- The load() method also allows us to specify data to be sent to the server and fetch the data.
- Example:

```
$('#msgDiv').load('getData',
    { name: 'bill' },
    function(data, status, jqXGR) {
            alert('data loaded')
      });
```