Introduction JavaScript

Script means small piece of code. Scripting languages are two kinds one is client-side other one is servers-side scripting. In general client-side scripting is used for verifying simple validation at client side, server-side scripting is used for database verifications. VBScript, java script and J script are examples for client-side scripting and ASP, JSP, Servlets, PHP etc. are examples of server-side scripting.

JavaScript (originally known as "LiveScript") is a scripting language that runs inside the browser to manipulate and enhance the contents of Web pages. Java Script is designed to add interactivity to HTML pages. Web pages are two types

Static web page
Dynamic webpage

- Static web page where there is no specific interaction with the client
- Dynamic web page which is having interactions with client and as well as validations can be added.

Simple HTML script is called static web page, if you add script to HTML page it is called dynamic page. Netscape navigator developed java script. Microsoft's version of JavaScript is Jscript.

- Java script code as written between <script>-----</script>tags
- All java script statements end with a semicolon
- Java script ignores white space
- Java script is case sensitive language
- Script program can save as either. Js or.html

Benefits of JavaScript

- It is widely supported by web browsers;
- It gives easy access to the document objects and can manipulate most of them.
- Java Script gives interesting animations with long download times associated with many multimedia data types;
- Web surfers don't need a special plug-in to use your scripts
- Java Script relatively secure - you can't get a virus infection directly from Java Script.
- JavaScript code resembles the code of C Language; the syntax of both the language is very close to each other. The set of tokens and constructs are same in both the language.

Problems with JavaScript

- Most scripts relay upon manipulating the elements of DOM;
- Your script does not work then your page is useless
- Because of the problems of broken scripts many web surfers disable java script support in their browsers
- Script can run slowly and complex scripts can take long time to start up

Similarities between java script and java:

1. Both java script and java having same kind of operators
2. Java script uses similar control structures of java
3. Nowadays both are used as languages for use on internet.
4. Labeled break and labeled continue both are similar
5. Both are case-sensitive languages

Difference between java script and java:

Java Script	Java
Java Script is scripting language	Java is a programming language
Java Script is object-based programming language.	Java is object –oriented programming language
Java script code is not compiled, only interpreted.	Java is compiled as well as interpreted language
Java Script is Weekly-typed language	Java is Strongly-typed language
JavaScript code is run on abrowser only.	**Java** creates applications that run in a virtual machine or browser

The syntax of the script tag isas follows:

```
<script language=""scripting language name"">
-------------------------------
</script>
```

The language attribute specifies the scripting language used in the script. Both Microsoft internet explorer and Netscape navigator use java script as the default scripting language. The script tag may be placed in either the head or the body or the body of an HTML document.

```
Ex: <script language=""javascript"">
        ----------
        ----------
        ----------
</script>
```

Comments in JavaScript:

Single line comment- //
Multi-line comment- <!-- comment -->

Operators in JavaScript:

- Arithmetic operators (+,-,*,/,%)
- Relational operators (<,>,!=,<=,>=)
- Logical operators(&&,||,!)
- Assignment operator(=)
- Increment decrement operators(++,--)
- Conditional/Ternary operator(?:)
- Bitwise operators(&,|,!)

Control structures:

- If statement
- Switch
- While
- Do-while
- For
- Break
- Continue

Control structures syntax and working as same as java language.

Variables

Variables are like storage units/place holders to hold values. A variable is a memory location to hold certain different types of data. In Javascript, A variable can store all kinds of data. It is important to know the proper syntax to which variables must conform:

- They must start with a letter or underscore ("_")
- Subsequent characters can also be digits (0-9) or letters (A-Z and/or a-z). Remember, JavaScript is case-sensitive. (That means that MyVar and myVar are two different names to JavaScript, because they have different capitalization.)
- You cannot use reserved words as variable names.
- You cannot use spaces in names.
- Names are case-sensitive.

Syntax:

var v_name = value;

Examples of legal variable names are fname, temp99, and _name.

When you declare a variable by assignment outside of a function, it is called global variable, because it is available everywhere in the document. When you declare a variable within a function, it is called local variable, because it is available only within the function. To assign a value to a variable, you use the following notation:

```
var num = 8;
var real= 4.5;
```

```
var myString = "Web Technologies";
```

Values of Variables (Data types)

JavaScript recognizes the following types of values:

- Numbers, such as 42 or 3.14159
- Boolean values, either true or false
- Strings, such as "Howdy!"
- NULL, a special keyword which refers tonothing.

Functions

Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure -a set of statements that performs a specific task when called. A function definition has these basic parts:

- The *function* keyword
- A function name
- A comma-separated list of arguments to the function in parentheses
- The statements in the function in curly braces: { }

Defining a Function

Defining the function means, name the function and specifies what to do when the function is called. You define a function within the <SCRIPT>...</SCRIPT> tags within the <HEAD> ... </HEAD> tags. While defining a function, you can also declare the variables which you will be calling in that function. Here's an example of *defining* a function:

```
function msg()
{
    window.alert("This is an alert box.");
}
```

Here's an example of a function that takes a parameter:

```
function welcome(string)
{
        window.alert("Hi"+string);
 }
```

When you call this function, you need to pass a parameter (such as the word that the user clicked on) into thefunction.

Calling a Function

Calling the function actually performs the specified actions. When you call a function, this is usually within the BODY of the HTML page, and you usually pass a parameter into the function on which the function will act.

Here's an example of calling the same function:

```
msg();
```

For the other example, this is how you may call it:

```
<inupt type="button" name="welcome" onClick="msg1()"/>
```

OBJECTS IN JAVA SCRIPT

When you load a document in your Web browser, it creates a number of JavaScript objects with properties and capabilities based on the HTML in the document and other information. These objects exist in a hierarchy that reflects the structure of the HTML page itself. The pre-defined objects that are most commonly used are the window and document objects. Some of the useful Objects are:

1. Document
2. Window
3. Browser
4. Form
5. Math
6. Date

The DocumentObject

A document is a web page that is being either displayed or created. The document has a number of properties that can be accessed by JavaScript programs and used to manipulate
the content of the page.

write or writeln

Html pages can be created on the fly using JavaScript. This is done by using the write or writeln methods of the document object.

Syntax:

```
document.write ("String");
document.writeln ("String");
```

In this document is object name and write () or writeln () are methods. Symbol period is used as connector between object name and method name. The difference between these two methods is carriage form feed character that is new line character automatically added into thedocument.

Exmaple:

```
document.write("<body>");
document.write("<h1> Hello</h1>");
```

bgcolor and fgcolor

These are used to set background and foreground(text) color to webpage. The methods accept either hexadecimal values or common names for colors.

Syntax:

```
document.bgcolor="#1f9de1";
document.fgcolor="silver";
```

anchors

The anchors property is an array of anchor names in the order in which they appear in the HTML Document. Anchors can be accessed like this:

Syntax:

```
document.anchors[0];
    :
document.anchors[n-1];
```

Links

Another array holding all links in the order in which they were appeared on the Webpage.

Forms

Another array, this one contains all of the HTML forms. By combining this array with the individual form objects each form item can be accessed.

The Window Object

The window object is used to create a new window and to control the properties of window.

Methods:

1. *open("URL","name") :* This method opens a new window which contains the document specified by URL and the new window is identified by its name.
2. *close():* this shutdowns thecurrent window.

Properties:

toolbar = [1|0] location= [1|0] menubar = [1|0] scrollbars = [1|0] status = [1|0] resizable = [1|0]

where as 1 means *on* and 0 means *off*

height=pixels, width=pixels : These properties can be used to set the window size.

The following code shows how to open a new window

```
newWin = open("first.html","newWin","status=0,toolbar=0,width=100,height=100");
```

Window object supports three types of message boxes.

1. Alert box
2. Confirm box
3. Prompt box

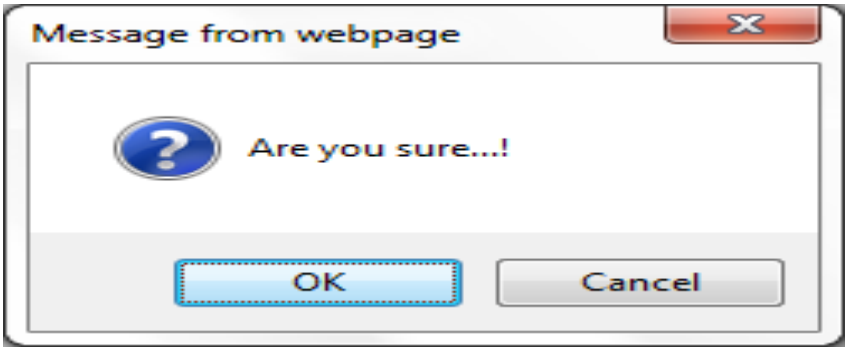
Alert box is used to display warning/error messages to user. It displays a text string with *OK* button.

Syntax: window. Alert ("Message");

Confirm Box is useful when submitting form data. This displays a window containing message with two buttons: *OK* and *Cancel.* Selecting *Cancel* will abort the any pending action, while *OK* will let the action proceed.

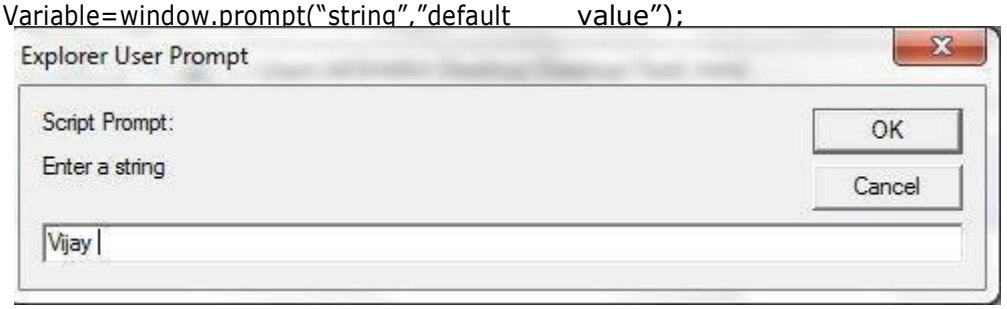
Syntax

```
window.confirm("String");
```



Prompt box used for accepting data from user through keyboard. This displays simple window that contains a prompt and a text field in which user can enter data. This method has two parameters: a text string to be used as a prompt and a string to use as the default value. If you don"t want to display a default then simply use an empty string.
Syntax

Variable=window.prompt("string","default value");



The Form Object

Two aspects of the form can be manipulated through JavaScript. First, most commonly and probably most usefully, the data that is entered onto your form can be checked at submission. Second you can actually build forms through JavaScript.
Form object supports three events to validate the form

onClick = "method()"

This can be applied to all form elements. This event is triggered when the user clicks on the element.

onSubmit = "method()"

This event can only be triggered by form itself and occurs when a form is submitted.

onReset = "method()"

This event can only be triggered by form itself and occurs when a form is reset.

Example: HTML program that applies a random background color when you click on button

```
<html>
<head>
<script language = "javascript">
        function change()
        {
                var clr = document.bgColor=parseInt(Math.random()*999999);
                document.f1.color.value=clr;
}
</script>
</head>
 <body>
```

```
<form name="f1">
        <input type="text" name="color"/>
        <input type="button" value="Cick me" onclick="change()"/>
</form> </body></html>
```

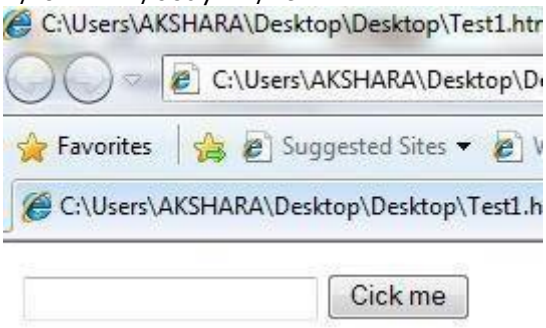


Fig.1: On first run background is white

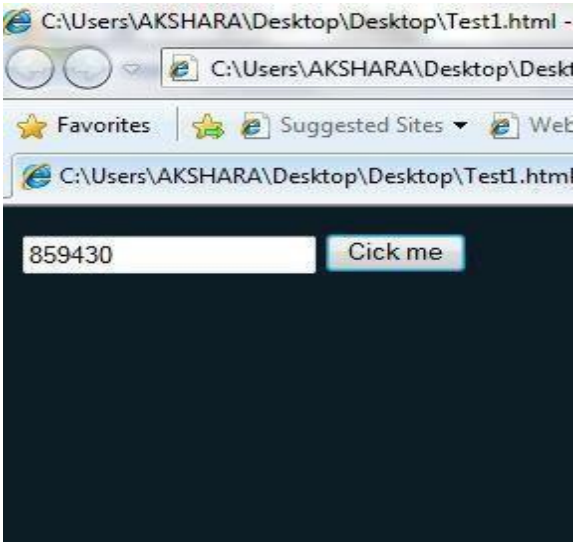


Fig.2: After clicking on button"Clickme": background changed to "black"

The browser Object

The browser is JavaScript object that can be used to know the details of browser. Some of the properties of the browser object is as follows:

Property	Description
navigator.appCodeName	It returns the internal name for the browser. For major browsers it is *Mozilla*
navigator.appName	It returns the public name of the browser – navigator or Internet Explorer
navigator.appVersion	It returns the version number, platform on which the browser is running.
navigator.userAgent	The strings appCodeName and appVersion concatenated together
navigator.plugins	An array containing details of all installed plug-ins
Navigator.mimeTypes	An array of all supported MIME Types

Example: Write javascript to display internal details of a browser (***Test.html***)

```
<script language = "javascript">
<!--
   document.writeln("Internal       Name:"+navigator.appCodeName);
   document.writeln("<br/>Public       Name:"+navigator.appName);
   document.writeln("<br/>Version:"+navigator.appVersion);
-->
</script>
```

Output:

The Math Object

The *Math* object holds all mathematical functions and values. All the functions and attributes used in complex mathematics must be accessed via this object.

Syntax:

```
Math.methodname();
Math.value;
```

Method	Description	Example
Math.abs(x)	Returns the absolute value	Math.abs(-20) is 20
Math.ceil(x)	Returns the ceil value	Math.ceil(5.8) is 6 Math.ceil(2.2) is 3
Math.floor(x)	Returns the floor value	Math.floor(5.8) is 5 Math.floor(2.2) is 2
Math.round(x)	Returns the round value, nearest integer value	Math.round(5.8) is 6 Math.round(2.2) is 2
Math.trunc(x)	Removes the decimal places it returns only integer value	Math.trunc(5.8) is 5 Math.trunc(2.2) is 2
Math.max(x,y)	Returns the maximum value	Math.max(2,3) is 3 Math.max(5,2) is 5
Math.min(x,y)	Returns the minimum value	Math.min(2,3) is 2 Math.min(5,2) is 2
Math.sqrt(x)	Returns the square root of x	Math.sqrt(4) is 2
Math.pow(a,b)	This method will compute the a^b	Math.pow(2,4) is 16
Math.sin(x)	Returns the sine value of x	Math.sin(0.0) is 0.0
Math.cos(x)	Returns cosine value of x	Math.cos(0.0) is 1.0
Math.tan(x)	Returns tangent value of x	Math.tan(0.0) is 0
Math.exp(x)	Returns exponential value i.e e^x	Math.exp(0) is 1
Math.random(x)	Generates a random number in between 0 and 1	Math.random()
Math.log(x)	Display logarithmic value	Math.log(2.7) is 1
Math.PI	Returns a ∏ value	a = Math.PI; a = 3.141592653589793

The Date Object

This object is used for obtaining the date and time. In JavaScript, dates and times represent in milliseconds since 1st January 1970 UTC. JavaScript supports two time zones: UTC and local. UTC is Universal Time, also known as Greenwich Mean Time(GMT), which is standard time throughout the world. Local time is the time on your System. A JavaScript *Date* represents date from -1,000,000,000 to -1,000,000,000 days relative to 01/01/1970.

Date Object Constructors:

new Date(); Constructs an empty date object.

new Date("String"); Creates a Date object based upon the contents of a text string.

new Date(year, month, day[,hour, minute, second]); Creates a Date object based upon the numerical values for the year, month and day.

```
var dt=new Date();
document.write(dt);          // Tue Dec 23 11:23:45 UTC+0530 2015
```

Methods in Date object:

Java script date object provides several methods, they can be classified is string form, get methods and set methods. All these methods are provided in the following table.

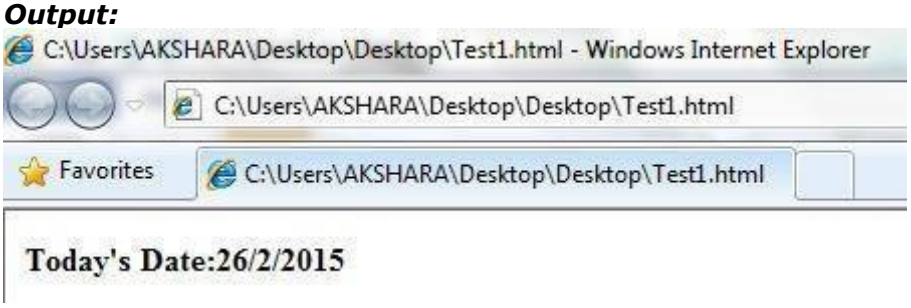
Method	Description
getDate()	Returns day of the month i.e. 1 to 31
getDay()	Returns an integer representing day of the week(0 - 6), Sunday to Saturday respectively.
getMonth()	Returns month of the year from 0 to 11, January to December respectively.

getFullYear()	Returns four-digit year number
getHours()	Returns hour field of the Date Object in 24 hours time format (0 to 23)
getMinitues()	Returns minute field of the Date Object from 0 to 59
getseconds()	Returns seconds field of the Date Object 0 to 59
setDate(v)	To set the date, day, month and full year
setDay(v)	
setMonth(v)	
setFullYear(y,m,d)	
setHours(v)	To set the hours, minutes, seconds of time
setMinutes(v)	
setSeconds(v)	
toString()	Returns the Date as a string

Example: Write javascript to display internal details of a browser (***Test.html***)

```
<script language = "javascript">
<!--
    var dt = new Date(); var day = dt.getDate();
    var month = dt.getMonth()+1; var year = dt.getFullYear();
    document.writeln("<h4>Today's Date:"+day+"/"+month+"/"+year);-
-->
</script>
```

Output:



Event Handling in Java Script

Introduction to Event Handling

- Event Handling is a software routine that processes actions, such as keystrokes and mouse movements.
- It is the receipt of an event at some event handler from an event producer and subsequent processes.

Functions of Event Handling

- Event Handling identifies where an event should be forwarded.
- It makes the forward event.
- It receives the forwarded event.
- It takes some kind of appropriate action in response, such as writing to a log, sending an error or recovery routine or sending a message.
- The event handler may ultimately forward the event to an event consumer.

Event Handlers

Category	Event Handler	Description
Window and Document events	onload	The browser has completed loading the document.
	onunload	The browser has completed unloading the document.
	onabort	The transfer of an image as been aborted.
	onerror	An error has occurred in the JavaScript program.
	onmove	The user has moved the browser window.
	onresize	The user has resized the browser window.
	onscroll	The user has moved the scrollbar.
Form events	onfocus	The user has entered an input field.
	onblur	The user has exited an input field.
	onchange	The content of an input field has changed.
	onselect	The user has selected text in an input or textarea field.
	onsubmit	A form has been submitted.
	onreset	The user has clicked the Reset button.
Keyboard and Mouse events	onkeydown	The user has begun pressing a key.
	onkeyup	The user has released a key.
	onkeypress	The user has pressed and released a key.
	onclick	The user has clicked the mouse button.
	ondblclick	The user has double-clicked the mouse button.
	onmousedown	The user has begun pressing the mouse button.
	onmouseup	The user has released the mouse button.
	onmousemove	The user has moved the mouse pointer.
	onmouseover	The user has moved the mouse over an element.
	onmouseout	The user has moved the mouse out from an element.

Onsubmit

```
<button onsubmit="myFunction()">Double-click me</button>
<script>
function myFunction()
{
alert("xyz")
}
</script>
```

Onclick

```
<button onlclick="myFunction()">Double-click me</button>
<script>
function myFunction()
{
alert("xyz")
}
</script>
```

Double click

```
<button ondblclick="myFunction()">Double-click me</button>
<script>
function myFunction()
{
alert("xyz")
```

```
}
</script>
```

Onfocus & OnBlur:

```
<html>
  <body>
    <p>Write below:</p>
    <input type = "text" onfocus="f1(this)" onblur = "f2(this)">
    <script>
      function f1(a) {
        a.style.background = "green";
      }
      function f2(a) {
        a.style.background = "red";
      }

    </script>
  </body>
</html>
```

OnKeypress & OnKeyup:

```
<html>
      <body>
      <h1>demo on keyboard events</h1>
      <input type="text"
onkeypress="this.style.background='green'";
onkeyup="this.style.background='red'";
onkeydown="this.style.background='yellow'";>
      <input type="text"  id="t1">
   </body>
</html>
```

Mouse OVER:

```
<html>
  <body>
  <center>
      <h1>demo on mouse events</h1>
      <input type="button" value= "Click me"
       onmouseover="this.style.background='red'";
       onmouseout="this.style.background='blue'";
onmousemove="this.style.background='pink'">
   </center>
   </body>
</html>
```

Onload:

```
<html> <head>
          <script type="text/JavaScript">
          function load()
```

```
{
alert("Page Load Event");
}
function unload()
{
alert("Page unLoad Event");
}
</script>
</head>
<body onLoad="load()" onUnload="unload()">
</body>
</html>
```

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Does Not Use Predefined Tags

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML is Extensible

Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).

Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed, older version of the application can still work:

```
<note>
  <date>2015-09-01</date>
```

```
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

Old Version

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

New Version

Note

To: Tove

From: Jani

Date: 2015-09-01 08:30

Don't forget me this weekend!

XML Simplifies Things

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

Books.xml

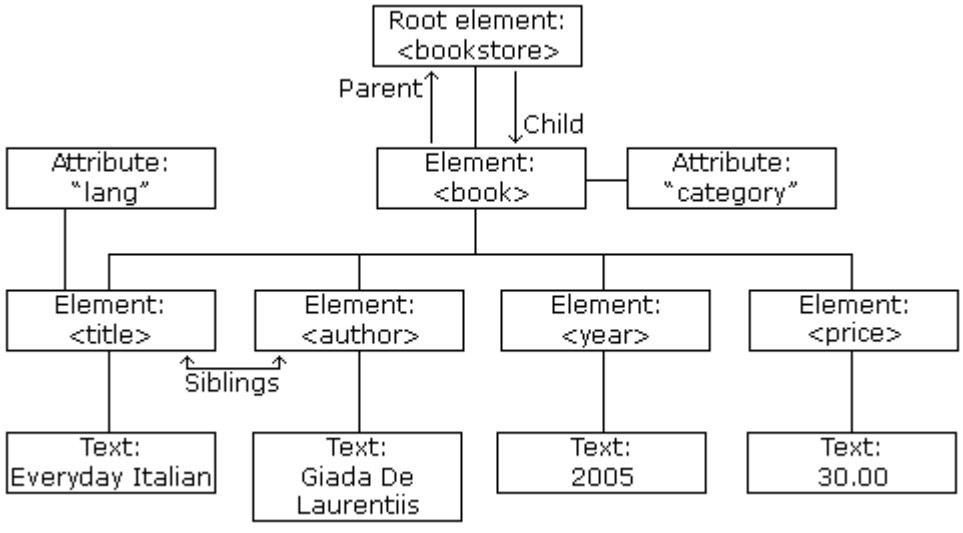
```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  <book category="web">
    <title lang="en">XQuery Kick Start</title>
```

```
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
 </book>

 <book category="web" cover="paperback">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>

</bookstore>
```

o/p:

Title	Author
Everyday Italian	Giada De Laurentiis
Harry Potter	J K. Rowling
XQuery Kick Start	James McGovern
Learning XML	Erik T. Ray

XML TREEE STRUCTURE:



XML SYNTAX:

```
<root>
 <child>
```

```
  <subchild>.....</subchild>
 </child>
</root>
```

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

Best Naming Practices

Create descriptive names, like this: <person>, <firstname>, <lastname>.

Create short and simple names, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".

Avoid ".". If you name something "first.name", some software may think that "name" is a property of the object "first".

Avoid ":". Colons are reserved for namespaces (more later).

XML Namespaces

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

```
<table>
 <tr>
  <td>Apples</td>
  <td>Bananas</td>
 </tr>
</table>
```

This XML carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

A user or an XML application will not know how to handle these differences.

Solving the Name Conflict Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

In the example above, there will be no conflict because the two <table> elements have different names.

Uniform Resource Identifier (URI)

A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource.

Default Namespaces

Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

xmlns="*namespaceURI*"

This XML carries HTML table information:

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

This XML carries information about a piece of furniture:

```
<table xmlns="https://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Naming Styles

There are no naming styles defined for XML elements. But here are some commonly used:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XSD How To?

XML documents can have a reference to a DTD or to an XML Schema.

A Simple XML Document

Look at this simple XML document called "note.xml":

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XML DTD

What is DTD

DTD stands for Document Type Definition**. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.**

The following example is a DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml"):

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

```
<email>
<to>Ravi</to>
<from>Raju</from>
<subject>Reminder Message</subject>
<body>Don't forget meet this weekend</body>
</email>
```

An XML Schema

XML Schemas are richer and more powerful than DTDs

- XML Schemas are written in XML
- XML Schemas support namespaces.
- XML Schemas Support Data Types.

The following example is an XML Schema file called "note.xsd" that defines the elements of the XML document above ("note.xml"):

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

XSD Simple elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

What is a Simple Element?

A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

However, the "only text" restriction is quite misleading. The text can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

Here are some XML elements:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

And here are the corresponding simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

XSD Restrictions/Facets

Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.

Restrictions for Datatypes

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than

	zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

No.	DTD	XSD
1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.
3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.

5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

The XMLHttpRequest Object

The XMLHttpRequest object can be used to request data from a web server.

The XMLHttpRequest object is **a developers dream**, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

Sending an XMLHttpRequest

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

<h2>Using the XMLHttpRequest Object</h2>

<div id="demo">
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</div>

<script>
function loadXMLDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "xmlhttp_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

O/P: Using the XMLHttpRequest Object

Change Content

XML Parser

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object. <html>
<body>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>

XML DOM:

The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.

<h1 id="demo">This is a Heading</h1>

```
<button type="button"
onclick="document.getElementById('demo').innerHTML = 'Hello World!'">
</button>
```

DOM Example

Look at the following XML file (books.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Node Properties DOM:

In the XML DOM, each node is an **object**.

Objects have methods and properties, that can be accessed and manipulated by JavaScript.

Three important node properties are:

- nodeName
- nodeValue
- nodeType

The nodeName Property

The nodeName property specifies the name of a node.

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always

The nodeValue Property

The nodeValue property specifies the value of a node.

- nodeValue for element nodes is undefined
- nodeValue for text nodes is the text itself

SAX (Simple API for XML)

- A SAX Parser implements SAX API. This API is an event based API and less intuitive.

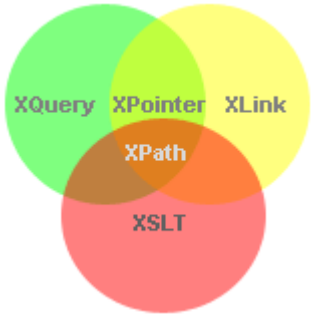
S.NO.	SAX PARSER	DOM PARSER
01.	It is called a Simple API for XML Parsing.	It is called as Document Object Model.
02.	It’s an event-based parser.	It stays in a tree structure.
03.	SAX Parser is slower than DOM Parser.	DOM Parser is faster than SAX Parser.
04.	Best for the smaller sizes of files.	Best for the larger size of files.
05.	It is suitable for making XML files in Java.	It is not good at making XML files in low memory.
06.	The internal structure can not be created by SAX Parser.	The internal structure can be created by DOM Parser.
07.	It is read-only.	It can insert or delete nodes.
08.	In the SAX parser backward navigation is not	In DOM parser backward and forward search is

S.NO.	SAX PARSER	DOM PARSER
	possible.	possible
09.	Suitable for efficient memory.	Suitable for large XML document.
10.	A small part of the XML file is only loaded in memory.	It loads whole XML documents in memory.

What is XPath?

XPath is a major element in the XSLT standard.

XPath can be used to navigate through elements and attributes in an XML document.



- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT and in XQuery
- XPath is a W3C recommendation

EXAMPLE:

```
<bookstore>

<book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="web">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
```

```
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="web">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

Displaying XML with XSLT

XSLT (eXtensible Stylesheet Language Transformations) is the recommended style sheet language for XML.

XSLT is far more sophisticated than CSS. With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

XSLT uses XPath to find information in an XML document.

```
 <breakfast_menu>

<food>
<name>Belgian Waffles</name>
<price>$5.95</price>
<description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
<calories>650</calories>
</food>

<food>
<name>Strawberry Belgian Waffles</name>
<price>$7.95</price>
<description>Light Belgian waffles covered with strawberries and whipped cream</description>
<calories>900</calories>
</food>

</breakfast_menu>
```

XSLT CODE:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>
```

```
<body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Xml code:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
```

```
  <year>1985</year>
</cd>

<cd>
  <title>Unchain my heart</title>
  <artist>Joe Cocker</artist>
  <country>USA</country>
  <company>EMI</company>
  <price>8.20</price>
  <year>1987</year>
</cd>

  </catalog>
```

What is XSLT

Before XSLT, first we should learn about XSL. XSL stands for EXtensible Stylesheet Language. It is a styling language for XML just like CSS is a styling language for HTML.

XSLT stands for XSL Transformation. It is used to transform XML documents into other formats (like transforming XML into HTML).

What is XSL

In HTML documents, tags are predefined but in XML documents, tags are not predefined. World Wide Web Consortium (W3C) developed XSL to understand and style an XML document, which can act as XML based Stylesheet Language.

An XSL document specifies how a browser should render an XML document.

Main parts of XSL Document

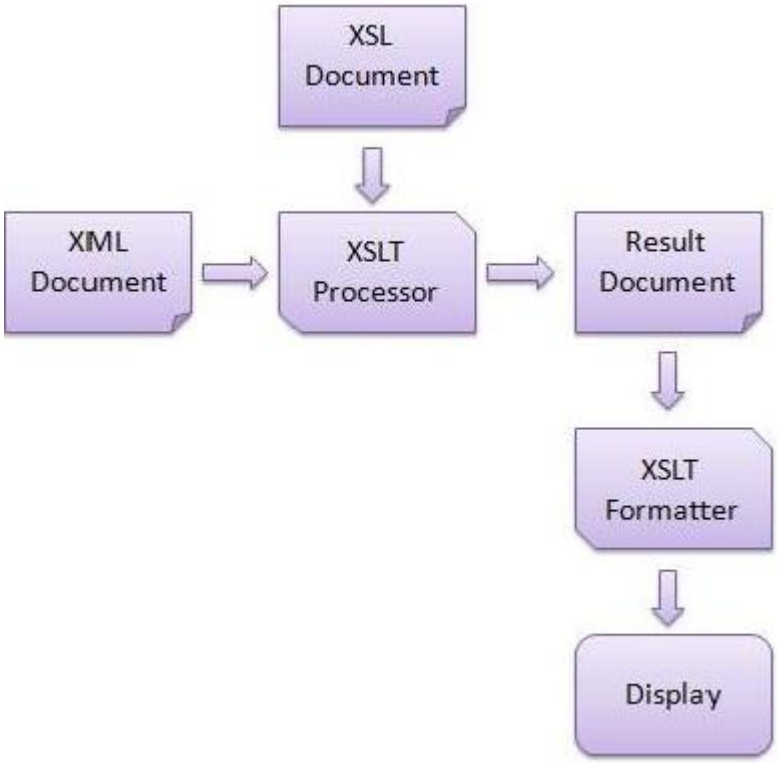
- **XSLT:** It is a language for transforming XML documents into various other types of documents.
- **XPath:** It is a language for navigating in XML documents.
- **XQuery:** It is a language for querying XML documents.
- **XSL-FO:** It is a language for formatting XML documents.

How XSLT Works

The XSLT stylesheet is written in XML format. It is used to define the transformation rules to be applied on the target XML document. The XSLT processor takes the XSLT stylesheet and applies the

transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. At the end it is used by XSLT formatter to generate the actual output and displayed on the end-user.

IMAGE REPRESENTATION:



Advantages of XSLT:

- XSLT is template based. So it is more resilient to changes in documents than low level DOM and SAX.
- By using XML and XSLT, the application UI script will look clean and will be easier to maintain.
- XSLT templates are based on XPath pattern which is very powerful in terms of performance to process the XML document.
- XSLT can be used as a validation language as it uses tree-pattern-matching approach.
- You can change the output simply modifying the transformations in XSL files.

Some examples: to represent student data USING XML(mid question)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
  <gender>Male</gender>
 </student>
 <student>
  <name>Daryl Dixon </name>
  <age>33</age>
  <subject>Science</subject>
  <gender>Male</gender>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
  <gender>Female</gender>
 </student>
</students>
```

AJAX

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data

AJAX Example Explained

HTML Page

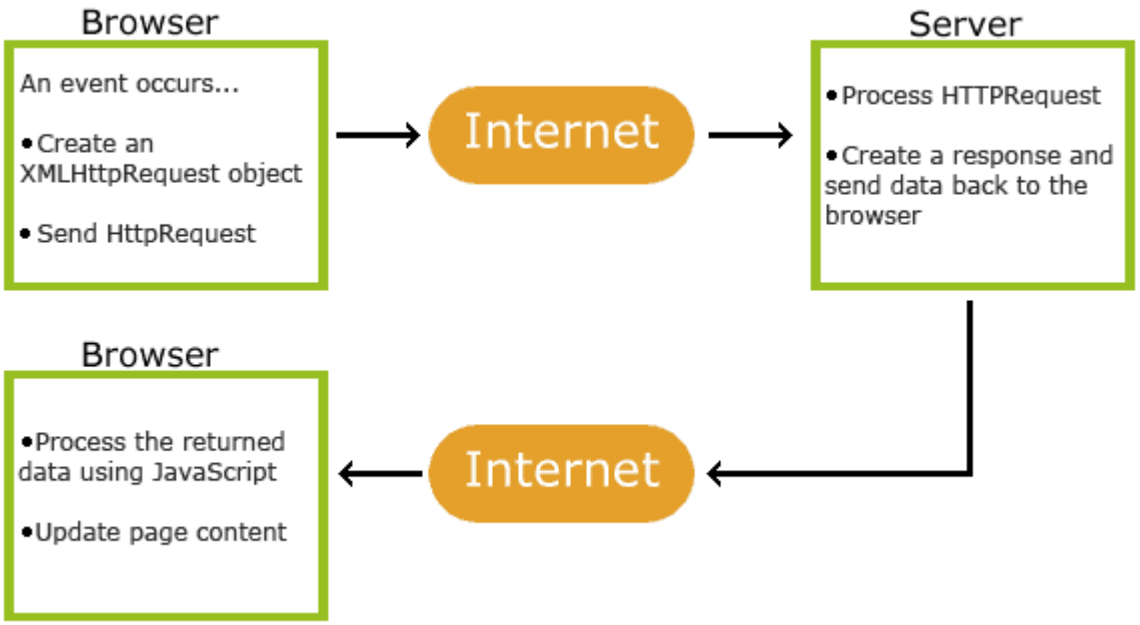
```
<!DOCTYPE html>
```

```
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```

Function loadDoc()

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

How AJAX Works



- 1. An event occurs in a web page (the page is loaded, a button is clicked)
- 2. An XMLHttpRequest object is created by JavaScript
- 3. The XMLHttpRequest object sends a request to a web server
- 4. The server processes the request
- 5. The server sends a response back to the web page
- 6. The response is read by JavaScript
- 7. Proper action (like page update) is performed by JavaScript

Integrating PHP and AJAX:

```
<html>
```

```
<head>
<script>
function showHint(str) {
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  } else {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("txtHint").innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET", "gethint.php?q=" + str, true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form action="">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

O/P:

Start typing a name in the input field below:

First name:
Suggestions:

XML Web Services
WSDL

- WSDL stands for Web Services Description Language
- WSDL is an XML-based language for describing Web services.
- WSDL is a W3C recommendation

SOAP

- SOAP stands for Simple Object Access Protocol
- SOAP is an XML based protocol for accessing Web Services.
- SOAP is based on XML
- SOAP is a W3C recommendation

RDF

- RDF stands for Resource Description Framework

- RDF is a framework for describing resources on the web
- RDF is written in XML
- RDF is a W3C Recommendation

RSS

- RSS stands for Really Simple Syndication
- RSS allows you to syndicate your site content
- RSS defines an easy way to share and view headlines and content
- RSS files can be automatically updated
- RSS allows personalized views for different sites
- RSS is written in XML

XML WSDL:

- WSDL stands for Web Services Description Language
- WSDL is used to describe web services
- WSDL is written in XML

WSDL Documents

An WSDL document describes a web service. It specifies the location of the service, and the methods of the service, using these major elements:

Element	Description
<types>	Defines the (XML Schema) data types used by the web service
<message>	Defines the data elements for each operation
<portType>	Describes the operations that can be performed and the messages involved.
<binding>	Defines the protocol and data format for each port type

The main structure of a WSDL document looks like this:

```
<definitions>

<types>
  data type definitions.....
</types>

<message>
  definition of the data being communicated....
</message>

<portType>
  set of operations.....
</portType>

<binding>
  protocol and data format specification....
</binding>

</definitions>
```

WSDL Example

This is a simplified fraction of a WSDL document:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

The <portType> Element

The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.

The request-response type is the most common operation type, but WSDL defines four types:

Type	Definition
One-way	The operation can receive a message but will not return a response

Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

---

WSDL One-Way Operation

A one-way operation example:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

WSDL Request-Response Operation

A request-response operation example:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

WSDL Binding to SOAP

WSDL bindings defines the message format and protocol details for a web service.

A request-response operation example:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

The **binding** element has two attributes - name and type.

The **soap:binding** element has two attributes - style and transport.

XML SOAP:

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- Soap is based on XML

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

---

Skeleton SOAP Message

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

The SOAP Envelope Element

The required SOAP Envelope element is the root element of a SOAP message. This element defines the XML document as a SOAP message.

Example

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

The SOAP Body Element

The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.

Immediate child elements of the SOAP Body element may be namespace-qualified.

Example

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

The example above requests the price of apples. Note that the m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.

A SOAP response could look something like this:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

A SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

The SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

UDDI:

- UDDI stands for Universal Description, Discovery and Integration.
- UDDI is a directory for storing information about web services.
- UDDI is a directory of web service interfaces described by WSDL.
- UDDI communicates via SOAP.
- UDDI is built into the Microsoft .NET platform.
- UDDI is a specification for a distributed registry of web services.
- UDDI is a platform-independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.

A business or a company can register three types of information into a UDDI registry. This information is contained in three elements of UDDI.

These three elements are −

- White Pages,
- Yellow Pages, and
- Green Pages.

White Pages

White pages contain −

- Basic information about the company and its business.
- Basic contact information including business name, address, contact phone number, etc.

Yellow Pages

- Yellow pages contain more details about the company.

Green Pages

Green pages contains technical information about a web service. A green page allows someone to bind to a Web service after it's been found. It includes −

- The various interfaces
- The URL locations
- Discovery information and similar data required to find and run the Web service.