UNIT II

CLIENT SIDE PROGRAMMING

Java Script: An introduction to JavaScript–JavaScript DOM Model-Date and Objects,- Regular Expressions- Exception Handling- Validation-Built-in objects-Event HandlingDHTML with JavaScript- JSON introduction – Syntax – Function Files – Http Request – SQL.

JAVA SCRIPT: AN INTRODUCTION TO JAVASCRIPT

History and Versions of JavaScript :

•JavaScript was initially developed by Brendan Eich as part of the Netscape 2.0 release.

•The language was called LiveScript in its early stages. But on December 4, 1995, before the final release of Netscape 2.0, the language was publicly announced as JavaScript.

•This name change was intended to link the scripting language with the rising popularity of Sun's Java programming language.

•There are also tremendous differences between JavaScript and Java.

•JavaScript became especially popular after the 1.1 release as part of Netscape's 3.0 browser. This version of JavaScript allowed web page developers to produce the familiar rollover effect: as the mouse moves over images, the images change.

•Other JavaScript versions followed as part of later Netscape browser releases. Meanwhile, Microsoft introduced the JScript Rprogramming language (the JavaScript name was owned by Sun) in July of 1996, and this language similarly went through a series of revisions as new versions of Internet Explorer were released.

•There is yet another flavor of JavaScript, known as ECMAScript.

•Soon after announcing JavaScript, Sun and Netscape approached an organization then known as the European Computer Manufacturers Association (ECMA)to produce a standard for JavaScript.

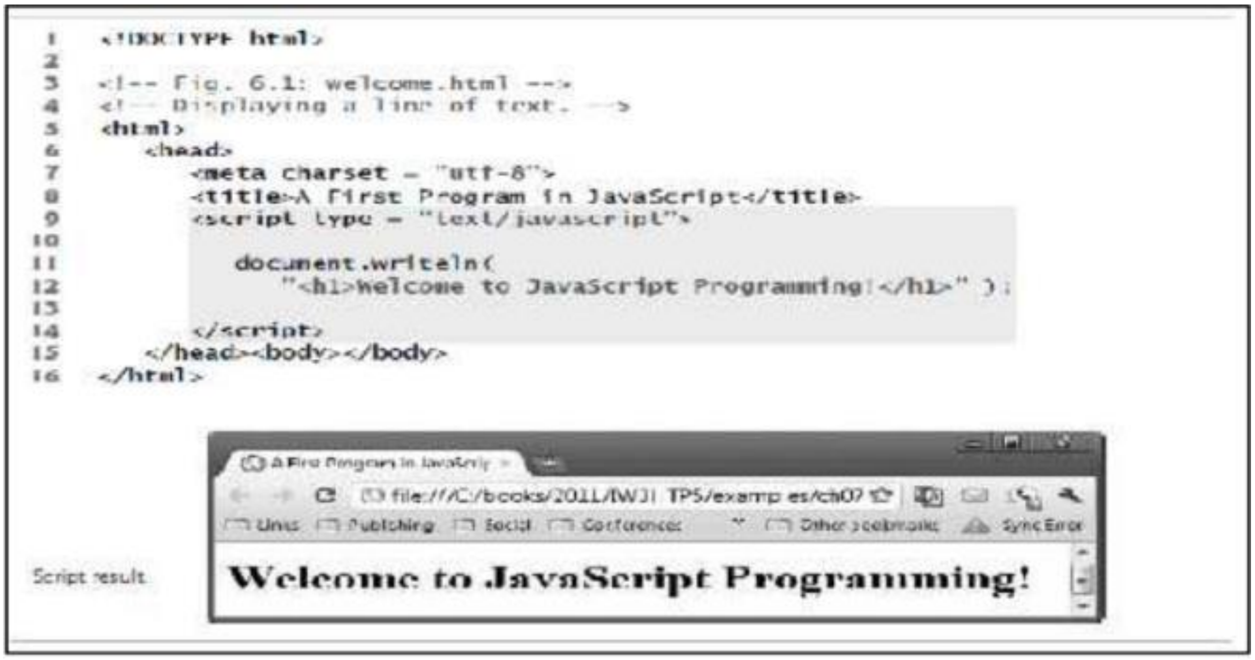
•The third edition of the ECMA-262 standard [ECMA-262] was released in 1999 and has been a significant help in bringing JavaScript and JScript closer together.

•JavaScript 1.5, the version implemented in Mozilla 1.4, conforms with ECMAScript Edition 3 (barring bugs), and JScript versions 5.5 (the version implemented in IE5.5) and beyond also appear to be compliant. Other browsers, such as Opera and Safari, also support the ECMAScript standard

First Script: Displaying a Line of Text with JavaScript in aWeb Page

•We begin with a simple script (or program) that displays the text "Welcome to JavaScriptProgramming!" in the HTML5 document.

•All major web browsers contain JavaScript interpreters, which process the commands writtenin JavaScript.



```
1   <!DOCTYPE html>
2
3   <!-- Fig. 6.1: welcome.html -->
4   <!-- Displaying a line of text. -->
5   <html>
6      <head>
7         <meta charset = "utf-8">
8         <title>A First Program in JavaScript</title>
9         <script type = "text/javascript">
10
11           document.writeln(
12              "<h1>Welcome to JavaScript Programming!</h1>" );
13
14        </script>
15     </head><body></body>
16  </html>
```

Script result
A First Program in JavaScript
file:///C:/books/2011/IW3HTP5/examples/ch07
Links
Publishing
Social
Conferences
Other bookmarks
Sync Error

```
Welcome to JavaScript Programming!
```

Lines 11–12 do the “real work” of the script, namely, displaying the phrase Welcome toJavaScript Programming! as an h1 heading in the web page. Line 6 starts the <head> section of the document. For the moment, the JavaScript code we write will appear in the <head> section.The browser interprets the contents of the <head> section first, so the JavaScript programs we write there execute before the <body> of the HTML5 document displays.

The script Element and Commenting Scripts:

•Line 9 uses the <script> tag to indicate to the browser that the text which follows is part of a script.
•The type attribute specifies the MIME type of the script as well as the scripting language used in the script—in this case, a text file written in javascript.
•In HTML5, the default MIME type for a <script> is "text/html", so we can omit the type attribute from the <script> tags.
Strings:
•Lines 11–12 instruct the browser’s JavaScript interpreter to perform an action, namely, to display in the web page the string of characters contained between the double quotation(") marks (also called a string literal).
•Individual white-space characters between words in a string are not ignored by the browser.
•Browsers ignore leading white-space characters(i.e., white space at the beginning of a string).
Using the document Object :
•Lines 11–12 use the browser’s document object, which represents the HTML5 document the browser is currently displaying.
•This object allows to specify text to display in theHTML5 document.
•The browser creates a set of objects that allow to access and manipulate every element of an HTML5 document.
•An object resides in the computer’s memory and contains information used by the script.
The term object normally implies that attributes (data) and behaviors (methods) are associated

with the object.

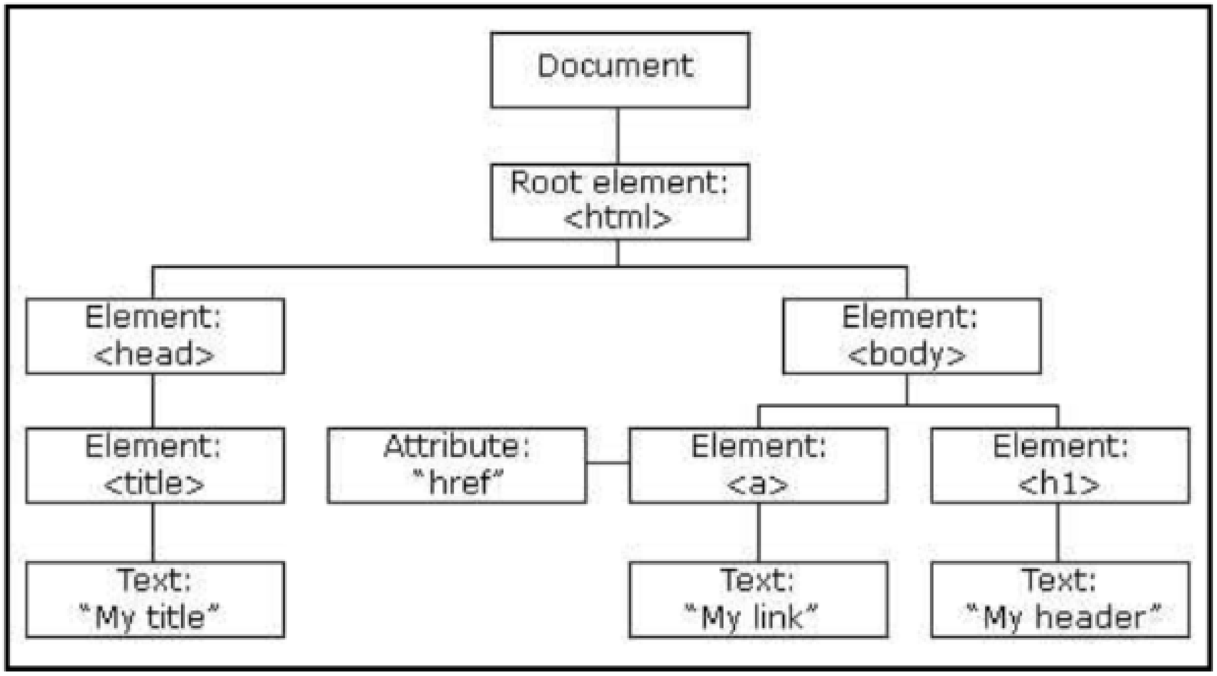
•The object’s methods use the attributes to perform useful actions for the client of the object (i.e., the script that calls the methods).
•A method may require additional information (arguments) to perform its actions; this information is enclosed in parentheses after the name of the method in the script.
•In lines 11–12, we call the document object’s writeln method to write a line of HTML5 markup in the HTML5 document.
•The parentheses following the method name writeln contain the one argument that method writeln requires.
•If the string contains HTML5 elements, the browser interprets these elements and renders them on the screen.
In this example, the browser displays the phrase Welcome to JavaScript Programming! as an h1-level
HTML5 heading, because the phrase is enclosed in an h1 element

Statements :
•The code elements in lines 11–12, including document.writeln, its argument in the parentheses(the string) and the semicolon (;), together are called a statement.
•Every statement ends with a semicolon (also known as the statement terminator)
•Line 14 indicates the end of the script. In line 15, the tags <body> and </body> specify that this HTML5 document has an empty body.

Embedding JavaScript Code into HTML5 Documents :
•JavaScript code is typically placed in a separate file, then included in the HTML5 document that uses the script.
•This makes the code more reusable, because it can be included into any HTML5 document.

JAVASCRIPT DOM MODEL

The HTML DOM (Document Object Model)
When a web page is loaded, the browser creates a Document Object Model of the page.
The HTML DOM model is constructed as a tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:
•JavaScript can change all the HTML elements in the page
•JavaScript can change all the HTML attributes in the page
•JavaScript can change all the CSS styles in the page
•JavaScript can remove existing HTML elements and attributes
•JavaScript can add new HTML elements and attributes
•JavaScript can react to all existing HTML events in the page
•JavaScript can create new HTML events in the page

What is the DOM?
The DOM is a W3C (World Wide Web Consortium) standard.
The DOM defines a standard for accessing documents:
"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
The DOM Programming Interface
The HTML DOM can be accessed with JavaScript (and with other programming languages).
In the DOM, all HTML elements are defined as objects.
The programming interface is the properties and methods of each object.
A property is a value that you can get or set (like changing the content of an HTML element).
A method is an action you can do (like add or deleting an HTML element).

The getElementById Method
The most common way to access an HTML element is to use the id of the element.
In the example above the getElementById method used id="demo" to find the element.
The innerHTML Property
The easiest way to get the content of an element is by using the innerHTML property.

The innerHTML property is useful for getting or replacing the content of HTML elements.

The HTML DOM Document

In the HTML DOM object model, the document object represents your web page.
The document object is the owner of all other objects in your web page.
If we want to access objects in an HTML page,
always start with accessing the document object

Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

Finding HTML Elements

a)Finding HTML elements by id
b)Finding HTML elements by tag name
c)Finding HTML elements by class name
d)Finding HTML elements by CSS selectors
e)Finding HTML elements by HTML object collections

Eg:
```
<!DOCTYPE html>
<html><body>
<h1>My First Page</h1><b>
<p id="demo"></p></b>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script></body></html>
```
Output:
```
Hello World!
```

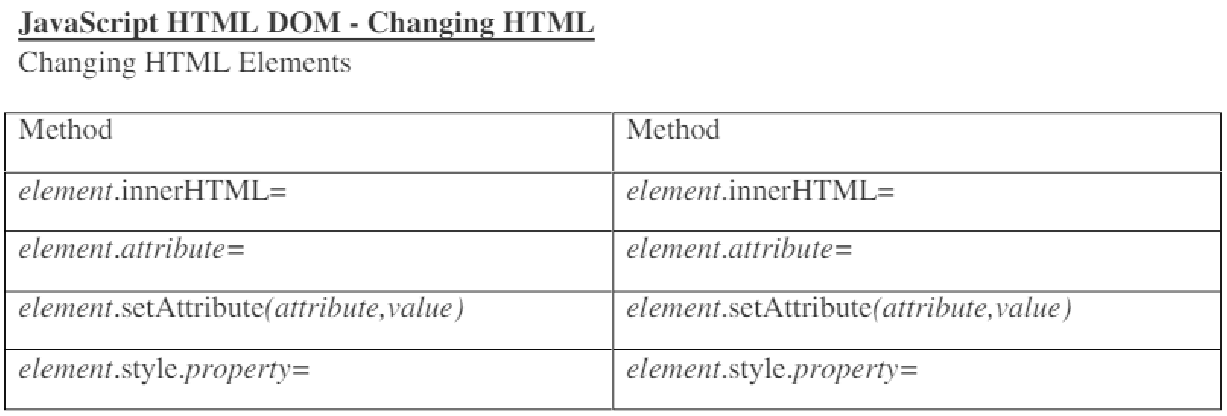
Finding HTML Elements by Tag Name

This example finds all <p> elements:

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

```
</script></body></html>
```
Output:
```
Hello World!
The DOM is very useful.
The first paragraph (index 0) is: Hello World!
```

Finding HTML Elements by Class Name
If you want to find all HTML elements with the same class name, use getElementsByClassName().This example returns a list of all elements with class="intro".
Eg:
```
<!DOCTYPE html>
<html><body>
<p  class =”a1”>Hello World program!</p>
<p class=”a2”>The DOM is very useful.</p>
<p id="demo"></p>
<script>
var  x = document.getElementsByClassName("a1");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) is: ' + x.innerHTML;
</script></body></html>
```
Output:
```
Hello World program!
The DOM is very useful.
The first paragraph (index 0) is: Hello World program!
```



JavaScript HTML DOM - Changing HTML

Changing HTML Elements

Method	Method
element.innerHTML=	*element*.innerHTML=
element.attribute=	*element.attribute*=
element.setAttribute(*attribute,value*)	*element*.setAttribute(*attribute,value*)
element.style.*property*=	*element*.style.*property*=

Changing the HTML Output Stream
JavaScript can create dynamic HTML content:
Date: Fri Aug 07 2015 09:47:04 GMT+0530 (central africa Time)
In JavaScript, document.write() can be used to write directly to the HTML output stream:
Eg:
```
<html>
<body><script>
document.write(Date());
</script></body>
```

```
</html>
```

Output:

```
Fri Aug 07 2015 09:47:58 GMT+0530 (Central africa Time)
```

Changing HTML Content

The easiest way to modify the content of an HTML element is by using he innerHTML property.
To change the content of an HTML element, use this syntax:

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

This example changes the content of a <p> element:
Eg:

```
<!DOCTYPE html>
<html>
<body><p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Output:

```
New text!
The paragraph above was changed by a script.
```

Changing the Value of an Attribute

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

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

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

```
<!DOCTYPE html>
<html>
<body>
<img id="image" src="smiley.gif" width="160" height="120">
<script>
document.getElementById("image").src = "landscape.jpg";
</script>
<p>The original image was smiley.gif, but the script changed it to landscape.jpg</p>
</body>
</html>
```

Adding and Deleting Elements

Method	Description
document.createElement()	Create an HTML element
document.removeChild()	Remove an HTML element
document.appendChild()	Add an HTML element
document.replaceChild()	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Changing HTML Style

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

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

The following example changes the style of a <p> element:

Eg:

```
<html><body>
<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontFamily = "Arial";
document.getElementById("p2").style.fontSize = "larger";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Using Events

The HTML DOM allows you to execute code when an event occurs.Events are generated by the browser when "things happen" to HTML elements:

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

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

Eg:

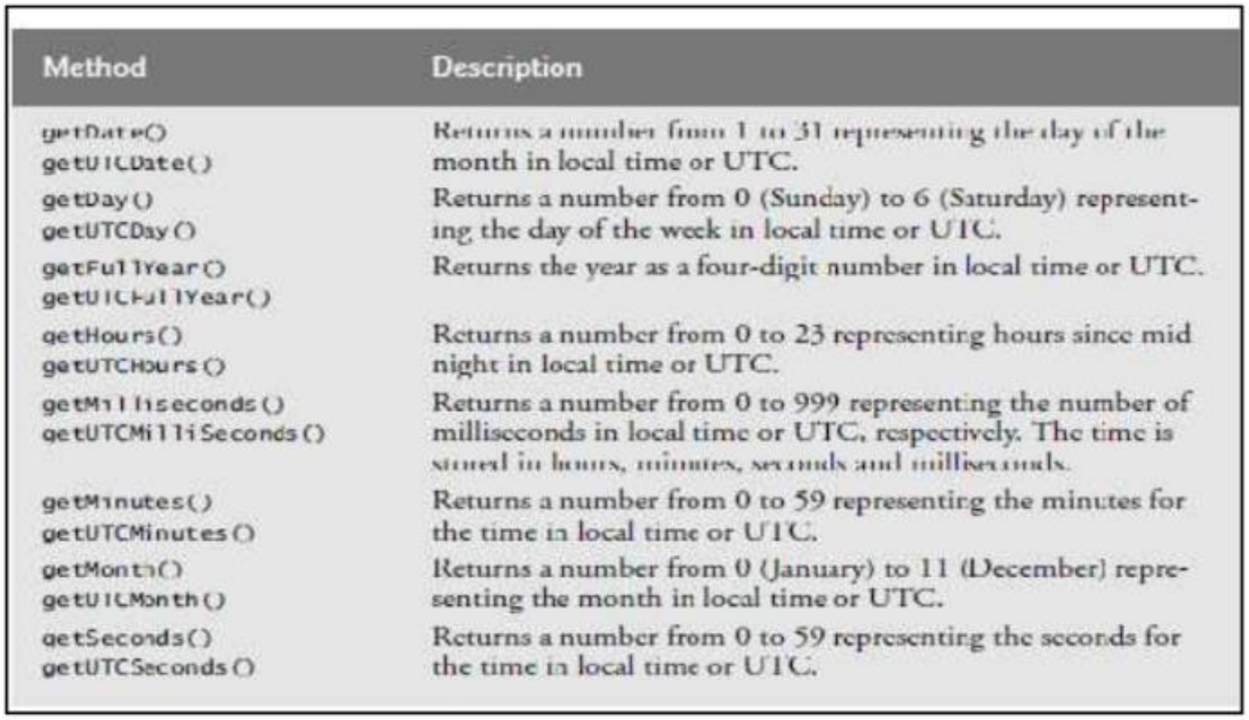
```
<html>
<body>
<h1 id="id1">My Heading 1</h1>
<button type="button" onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>
</body>
</html>
```

DATE

•JavaScript's Date object provides methods for date and time manipulations.

•These can be performed based on the computer's local time zone or based on World Time Standard's Coordinated Universal Time (abbreviated UTC)—formerly called Greenwich Mean Time (GMT).

•Most methods of the Date object have a local time zone and a UTC version.



Method	Description

getDate()
getUTCDate()
	Returns a number from 1 to 31 representing the day of the month in local time or UTC.

getDay()
getUTCDay()
	Returns a number from 0 (Sunday) to 6 (Saturday) representing the day of the week in local time or UTC.

getFullYear()
getUTCFullYear()
	Returns the year as a four-digit number in local time or UTC.

getHours()
getUTCHours()
	Returns a number from 0 to 23 representing hours since mid night in local time or UTC.

getMilliseconds()
getUTCMilliSeconds()
	Returns a number from 0 to 999 representing the number of milliseconds in local time or UTC, respectively. The time is stored in hours, minutes, seconds and milliseconds.

getMinutes()
getUTCMinutes()
	Returns a number from 0 to 59 representing the minutes for the time in local time or UTC.

getMonth()
getUTCMonth()
	Returns a number from 0 (January) to 11 (December) representing the month in local time or UTC.

getSeconds()
getUTCSeconds()
	Returns a number from 0 to 59 representing the seconds for the time in local time or UTC.

Method	Description
`getTime()`	Returns the number of milliseconds between January 1, 1970, and the time in the `Date` object.
`getTimezoneOffset()`	Returns the difference in minutes between the current time on the local computer and UTC (Coordinated Universal Time).

`setDate( val )`
`setUTCDate( val )`
	Sets the day of the month (1 to 31) in local time or UTC.

`setFullYear( y, m, d )`
`setUTCFullYear( y, m, d )`
	Sets the year in local time or UTC. The second and third arguments representing the month and the date are optional. If an optional argument is not specified, the current value in the `Date` object is used.

`setHours( h, m, s, ms )`
`setUTCHours( h, m, s, ms )`
	Sets the hour in local time or UTC. The second, third and fourth arguments, representing the minutes, seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the `Date` object is used.

`setMilliSeconds( ms )`
`setUTCMilliseconds( ms )`
	Sets the number of milliseconds in local time or UTC.

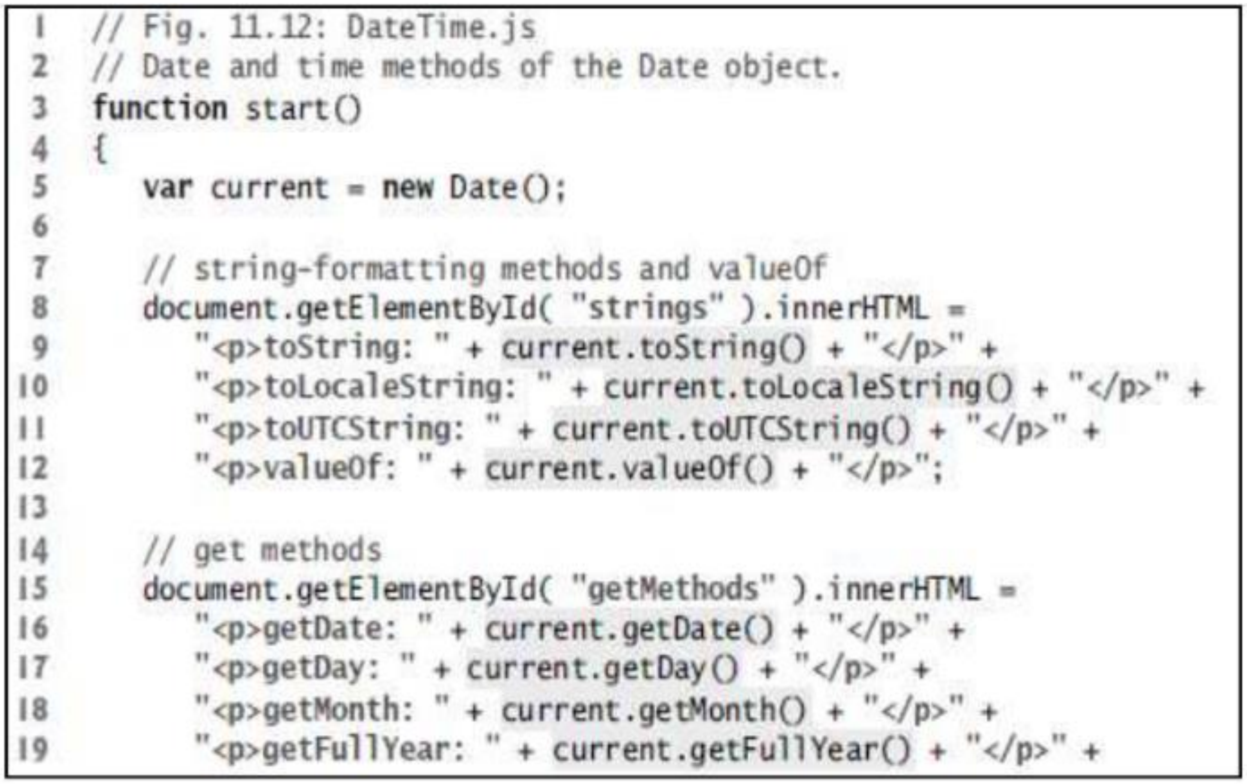
`setMinutes( m, s, ms )`
`setUTCMinutes( m, s, ms )`
	Sets the minute in local time or UTC. The second and third arguments, representing the seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the `Date` object is used.

`setMonth( m, d )`
`setUTCMonth( m, d )`
	Sets the month in local time or UTC. The second argument, representing the date, is optional. If the optional argument is not specified, the current date value in the `Date` object is used.

`setSeconds( s, ms )`
`setUTCSeconds( s, ms )`
	Sets the seconds in local time or UTC. The second argument, representing the milliseconds, is optional. If this argument is not specified, the current milliseconds value in the `Date` object is used.

`setTime( ms )`	Sets the time based on its argument — the number of elapsed milliseconds since January 1, 1970.

`toLocaleString()`	Returns a string representation of the date and time in a form specific to the computer's locale. For example, September 13, 2007, at 3:42:22 PM is represented as *09/13/07 15:47:22* in the United States and *13/09/07 15:47:22* in Europe.
`toUTCString()`	Returns a string representation of the date and time in the form: *15 Sep 2007 15:47:22 UTC.*
`toString()`	Returns a string representation of the date and time in a form specific to the locale of the computer (*Mon Sep 17 15:47:22 EDT 2007* in the United States).
`valueOf()`	The time in number of milliseconds since midnight, January 1, 1970. (Same as `getTime`.)



```
1   // Fig. 11.12: DateTime.js
2   // Date and time methods of the Date object.
3   function start()
4   {
5      var current = new Date();
6
7      // string-formatting methods and valueOf
8      document.getElementById( "strings" ).innerHTML =
9         "<p>toString: " + current.toString() + "</p>" +
10        "<p>toLocaleString: " + current.toLocaleString() + "</p>" +
11        "<p>toUTCString: " + current.toUTCString() + "</p>" +
12        "<p>valueOf: " + current.valueOf() + "</p>";
13
14     // get methods
15     document.getElementById( "getMethods" ).innerHTML =
16        "<p>getDate: " + current.getDate() + "</p>" +
17        "<p>getDay: " + current.getDay() + "</p>" +
18        "<p>getMonth: " + current.getMonth() + "</p>" +
19        "<p>getFullYear: " + current.getFullYear() + "</p>" +
```

Figure: Date and time methods of the Date object.

Date-Object Constructor with No Arguments:

•Line 5, in the script creates a new Date object.
•The new operator creates the Date object. The empty parentheses indicate a call to the Date object's constructor with no arguments. A constructor is an initializer method for an object.
•The Date constructor with no arguments initializes the Date object with the local computer's current date and time.

Methods toString, toLocaleString, toUTCString and valueOf :

•Line 9-12, in the script , demonstrate the methods toString, toLocaleString, toUTCString and valueOf.
•Method valueOf returns a large integer value representing the total number of milliseconds between midnight, January 1, 1970, and the date and time stored in Date object current.

Date-Object get Methods:
•Lines 16–25 demonstrate the Date object's get methods for the local time zone.
•The method getFullYear returns the year as a four-digit number.
•The method getTimeZoneOffset returns the difference in minutes between the local time zone and UTC time.

OBJECT PROPERTIES

•An object in JavaScript is a set of properties,each of which consists of a unique name along with a value belonging to one of JavaScript’s six data types.
•Like JavaScript variables, object properties themselves do not have data types; only the values assigned to properties have data types.

```
o.prop = true;
o.prop = "true";
o.prop = 1
```

Object creation and dynamic creation and removal of properties :

1. Expressions beginning with the keyword new are used to create objects.
2. When a JavaScript statement attempts to assign a value to an object property, and the property does not exist in the object, then a property with the given name is created in the object and assigned the specified value.
3. The keyword delete can be used to remove a property from an object.

•JavaScript provides a handy called an object initializer for creating an empty object creating properties on this object, and assigning values to these properties. For example, the statement

var o2 = { p1:5+9, p2:null, testing:"This is a test" };

creates an object with three properties p1, p2, and testing and assigns these properties the values 14, null, and This is a test, respectively.
•A reference to this object is then assigned to the variable o2.
•JavaScript provides a special for-in statement that can be used to iterate through all of the property names of an object. The following JavaScript code illustrates the use of this statement:

```
var hash = new Object();
hash.kim = "85";
hash.sam = "92";
hash.lynn = "78";
for (var aName in hash) {
window.alert(aName + " is a property of hash.");
}
```

Executing this program will produce three alert boxes, each with one of the names kim, sam, or lynn.
A method in JavaScript is simply a function that has been assigned as the value of a property of an object.

Constructors :
•Every JavaScript function we declare is also automatically a constructor.
•The scripting engine that we want to call a function as a constructor by prefixing the call with the new keyword.
•JavaScript does not have a class concept exactly like that found in Java and C++, but the constructor mechanism can be used to provide somewhat similar functionality. When we construct an object in JavaScript using new, the constructed object is known as an instance of the function (object) used as the constructor.

Arrays :
•JavaScript supplies a native function object named Array that can be used to construct objects that have special array characteristics and that inherit a number of array-oriented methods.

1.Creating an Array
•One way to create an array is to use the Array constructor directly in a call with no arguments: **var ary1 = new Array();**
•Alternatively, an array can be constructed and initialized with values by supplying two or more arguments to the Array constructor: **var ary2 = new Array(4, true, "OK");**
•After this statement is executed, ary2[0] will have the Number value 4, ary2[1] the Boolean value true, and ary2[2] the String value OK.

2.**Dynamically Changing Array Length:**
•Just as we can dynamically add properties to any JavaScript object, we can add properties to an array. we could follow the code creating ary2 (which had three elements) with the statement **ary2[3] = -12.6;**
•This will create a new property named (as a String) 3 with the value -12.6. In addition, the length property of ary2 will be changed to 4.
•Elements can also be removed from a JavaScript array by decreasing the value of the length property. For example, if we next executed the statement **ary2.length = 2;**

then the properties 2 and 3 would automatically be removed from ary2.

EXCEPTION HANDLING

Javascript Exception Handling :
• Runtime errors, also called exceptions, occur during execution (after compilation/interpretation).
For example, in runtime error ,the syntax is correct, but at runtime, it is trying to call a method that does not exist.
Logical Errors :
•Logic errors can be the most difficult type of errors to track down. These errors are not the result of a syntax or runtime error. Instead, they occur when we make a mistake in the logic that drives the script and we do not get the result we expected.
•we cannot catch those errors, because it depends on the business requirement what type of logic we want to put in the program.
Try catch :
•The try statement allows to define a block of code to be tested for errors while it is being executed.
•The catch statement allows to define a block of code to be executed, if an error occurs in the try block.
•The JavaScript statements try and catch come in pairs:

Syntax:
```
try
{
Block of code to try
}
catch(err)
{
Block of code to handle errors
}
```
Eg1:
```
<html>
<body>
<p id="demo"></p>
<script>
try {
addalert("Welcome guest!");
}
catch(err) {
document.getElementById("demo").innerHTML = err.message;
}
</script>
</body>
</html>
```
Output:
```
addalert is not defined
```

Eg2:
```
<html>
<head>
<script type="text/javascript">
function myFunc()
{
var a = 100;
try
{
alert("Value of variable a is : " + a );
}
catch ( e )
{
alert("Error: " + e.description );
}
}
</script>
</head>
<body>
<p>Click the following to see the result:</p>
<form>
<input type="button" value="Click Me" onclick="myFunc();" />
</form>
</body>
</html>
```
OUTPUT:

```
Value of variable a is : 100
```

The try...catch...finally Statement :

•The latest versions of JavaScript added exception handling capabilities.
•JavaScript implements the try...catch...finally construct as well as the throw operator to handle exceptions.
we can catch programmer-generated and runtime exceptions, but we cannot catch.
JavaScript syntax errors.

Eg:2 Try catch

```
<html>
<head>
<script type="text/javascript">
function myFunc()
{
var a = 100;
try
{
alert("Value of variable a is : " + a );
}
catch ( e ) {
alert("Error: " + e.description );
}
finally {
alert("Finally block will always execute!" );
}
}
</script>
</head>
<body>
<p>Click the following to see the result:</p>
<form>
<input type="button" value="Click Me" onclick="myFunc();" />
</form>
</body>
</html>
```

JavaScript can Raise Errors:

•When an error occurs, JavaScript will normally stop, and generate an error message.
•The technical term for this is: JavaScript will raise (or throw) an exception.

The throw Statement:

•The throw statement allows to create a custom error.
•Technically we can raise (throw) an exception.
•The exception can be a JavaScript String, a Number, a Boolean or an Object:

Eg:

```
<html>
```

```
<body>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="message"></p>
<script>


function myFunction()
{
var message, x;
message = document.getElementById("message");
message.innerHTML = "";
x = document.getElementById("demo").value;
try
{
if(x == "")  throw "empty";
if(isNaN(x)) throw "not a number";
x = Number(x);
if(x < 5)    throw "too low";
if(x > 10)   throw "too high";
else
message.innerHTML = x;
}
catch(err)
{
message.innerHTML = "Input is " + err;
}
}
</script>
</body>
</html>
```

Output:

```
Please input a number between 5 and 10:
Test Input
Input is too low
```

<u>FRAMES</u>

•Frames help us to divide the browser's window into multiple rectangular regions. Each region contains separate html web page and each of them work independently.

Frameset

•A set of frames in the entire browser is known as frameset. It tells the browser how to divide browser window into frames and the web pages that each has to load.

•The following table describes the various tags used for creating frames:

Tag	Description
<frameset> </frameset>	It is replacement of the <body> tag. It doesn't contain the tags that are normally used in <body> element; instead it contains the <frame> element used to add each frame.
<frame> </frame>	Specifies the content of different frames in a web page.
<base> </base>	It is used to set the default target frame in any page that contains links whose contents are displayed in another frame.

Eg: Web page creation for xyz college using frames PROGRAM:

Home.html

```
<html>
<head>
<title>SRR Homepage</title>
<frameset rows="10%,*"  border="1">
<frame src="Frame1.html" name="f1"/>
<frameset cols="20%,*" border="1">
<frame src=" Frame2.html" name="f2"/>
<frame src="Frame3.html" name="f3"/>
</frameset>
</frameset>
</head>
</html>
```

Frame1.html

```
<html>
<body bgcolor="Red">
<center><h1><b>XYZ  Engineering College</b></h1>
</center>
</body>
</html>
```

Frame2.html

```
<html>
<body bgcolor="gray">
<a href="cse.html" target="f3">CSE<br></a>
<a href="ece.html" target="f3">ECE<br></a>
</body>
</html>
```

Frame3.html

```
<html>
<body bgcolor="Yellow"><b>
<h4>Welcome e
<body>
</html>
```

cse.html

```
<html>
<body bgcolor="green">
<center><h2>Welcome to <h1>CSE Dept</h1></h2></center>
<br><font color="red">Objective</font><br>
To encourage students for broadening their excellence in technical education and
forstering
intellectual growth.The department has been always a forerunner in academics.
<br><font color="red">Course</font><br>
The Department offers 4 years B.E. Degree in Computer Science and Engineering It is
endowed with well-equipped labs and other infrastructure needed for giving the students
the guidance and the edge to compete and deliver innovative techniques in the field of
Computer Science & Engineering. <br><font color="red">Curriculum</font><br>
The curriculum is constantly updated to keep pace with the technological developments
and is a blend of theoretical concepts and practicals in the field of Computer Science &
Engineering <br><font color="red">Faculty</font><br>
The department owns well experienced and highly qualified, dedicated staff
members.The students are motivated by their guidance not only for their academic but
also for industrial & research work.
<br><font color="red">Laboratory Facilities</font>
<ul><li>Operating System Laboratory</li>
<li>System Software Laboratory</li>
<li>Network Programming Laboratory</li>
<li>Compiler Design Laboratory</li>
<li>Case Tools Laboratory</li>
<li>Database Management System Laboratory</li>
<li>Object Oriented Programming Laboratory</li>
<li>Graphics & Multimedia Laboratory</li> </ul>
</body>
</html>
```

ece.html

```
<html>
<body bgcolor="blue">
<center><h2>Welcome to <h1>ECE Dept</h1></h2></center>
<br><font color="red">Objective</font><br>
To provide the right ambience to the students to develop their functional skills, sharpen
The cumen through the adept faculty and the substantiate laboratories, to become more
Enterprising nd meet the requirements of the industry with latest advancement.
<br><font color="red">Course</font><br>
The Department offers the 4 year B.E. degree in Electronics and Communication
Engineering. It is endowed with well-equipped labs and other infrastructure needed for
giving the students the guidance and the edge to compete and deliver innovative
techniques in the field of Electronics & Communication engineering.
<br><font color="red">Curriculum</font><br>
The curriculum is constantly updated to keep pace with the technological developments
and is a blend of theoretical concepts and practicals in the field of Electronics &
Communication engineering.
```

```
<br><font color="red">Faculty</font><br>
The department owns well experienced and highly qualified, dedicated staff
members.The students are motivated by their guidance not only for their academic but
also for industrial & research work.
<br><font color="red">Laboratory Facilities</font>
<ul><li>Digital Signal Processing Laboratory</li>
<li>Microprocessor Laboratory</li>
<li>Communication System Laboratory</li>
<li>Electronic Circuits Laboratory </li>
<li>Electronic Devices Laboratory</li>
<li>Microwave Laboratory</li>
<li>Optical & RF Laboratory
<li>VLSI Design Laboratory</li>
</ul></body></html>
```

HTML FORMS

HTML forms are used to pass data to a server.

•A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

•The <form> tag is used to create an HTML form:

```
•<form>
.....
input elements
....</form>
```

HTML Forms - The Input Element

The most important form element is the input element.
The input element is used to select user information.
An input element can vary in many ways, depending on the type attribute. An input
element
•can be of type text field, checkbox, password, radio button, submit button, and more.
The most used input types are described below.
Text Fields
•<input type="text" /> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>
```

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters

Password Field

<input type="password" /> defines a password field:

```
•<form>
Password: <input type="password" name="pwd" />
</form>
```

Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

Checkboxes

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

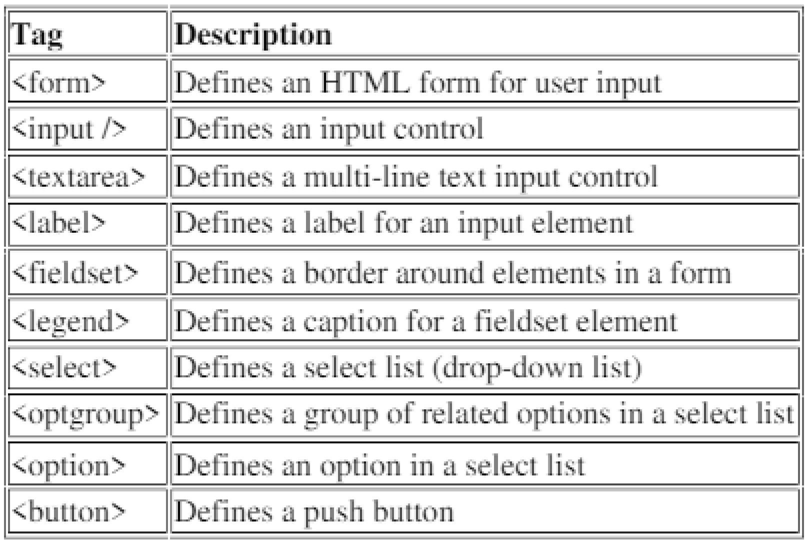
```
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>
```

Submit Button

<input type="submit" /> defines a submit button.
A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

HTML Form Tags



Tag	Description
<form>	Defines an HTML form for user input
<input />	Defines an input control
<textarea>	Defines a multi-line text input control
<label>	Defines a label for an input element
<fieldset>	Defines a border around elements in a form
<legend>	Defines a caption for a fieldset element
<select>	Defines a select list (drop-down list)
<optgroup>	Defines a group of related options in a select list
<option>	Defines an option in a select list
<button>	Defines a push button

EVENT HANDLING

Events :

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

An element is clicked on

- •The page has loaded
- •Input fields are changed

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

Eg:

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

DOM Event Handling :

The HTML DOM allows you to execute code when an event occurs.It is a script that get execute in response to those event.the process of connecting event handler to an event is called event registration.

Event Registration:

It means travelling of an event .It can be explained by two methods,

a)Event Capturing:If outer event executes first then inner executes second then it is event capturing.

b)Event bubbling:If inner executes first then outer executes second then it is event bubbling.

DOM Events :

Some of the HTML Dom Event are,

a)Mouse Events-onmousemove(),onmouseover(),onmouseout(),onmousein() etc.,

b)keyEvents-onkeyUp(),onkeydown() etc..,

c)form Event-obblur(),onChange() etc..,

d)Drag Event-onDrag(),onDraged() etc..,

e)frame Event-onLoad(),onUnload(),onresize()etc..,

Reacting to Events :

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

Examples of HTML events:

- •When a user clicks the mouse
- •When a web page has loaded
- •When an image has been loaded
- •When the mouse moves over an element
- •When an input field is changed
- •When an HTML form is submitted
- •When a user strokes a key

Eg:

```
<html>
```

```
<body>
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
</body>
</html>
```

Eg:

```
<html>
<body>
<p>Click the button to display the date.</p>
<button onclick="displayDate()">The time is?</button>
<script>
function displayDate() {
document.getElementById("demo").innerHTML = Date();
}
</script>
<p id="demo"></p>
</body>
</html
```

Assign Events Using the HTML DOM :

The HTML DOM allows you to assign events to HTML elements using JavaScript:

Eg:

```
<html>
<head>
</head>
<body>
<p>Click "Try it" to execute the displayDate() function.</p>
<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;
function displayDate() {
document.getElementById("demo").innerHTML = Date();
}
</script>
</body>
</html>
```

The onload and onunload Events :

The onload and onunload events are triggered when the user enters or leaves the page.
The onload event can be used to check the visitor's browser type and browser version, and load
The proper version of the web page based on the information.
The onload and onunload events can be used to deal with cookies.

Eg:

```
<html>
<body onload="checkCookies()">
```

```
<p id="demo"></p>
<script>
function checkCookies() {
var text = "";
if (navigator.cookieEnabled == true) {
text = "Cookies are enabled.";
} else {
text = "Cookies are not enabled.";
}
document.getElementById("demo").innerHTML = text;
}
</script>
</body>
</html>
```

The onchange Event :

The onchange event are often used in combination with validation of input fields.
Below is an example of how to use the onchange. The upperCase() function will be called when user changes the content of an input field.
Eg:

```
<html>
<head>
<script>
function myFunction() {
var x = document.getElementById("fname");
x.value = x.value.toUpperCase();
}
</script>
</head>
<body>
Enter your name: <input type="text" id="fname" onchange="myFunction()">
<p>When you leave the input field, a function is triggered which transforms the input text to
upper case.</p>
</body>
</html>
```

DHTML WITH JAVASCRIPT

DHTML-Dynamic Hypertext Markup Language
1.A page with HTML,Css,DOM and Javascript then we call that page as Dynamic HyperText Markup Language.
2.DHTML sites will be fast enough upon client side technology.
3.DHTML page can be used to play audio ,videos,background sounds and animations without the need of server

Eg: DHTML with html,CSS,javascript,DOM

```
<html>
<head>
<style type="text/css">
h6
{
color:green;
}
</style>
</head>
<body>
<p id="p1">Hello World 1!</p>
<h6 id="p2">Hello World 2 !</h6>
<p id="p3">Hello World 3!</p>
<script>
function test()
{
document.getElementById("p2").style.color = "blue";
}
</script>
<audio controls>
<source src="l.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
<button onclick="test()">click Here</button>
</body>
</html>
```

Changing HTML Style :

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

Eg:

```
<html><body>
<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontFamily = "Arial";
document.getElementById("p2").style.fontSize = "larger";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

JSON INTRODUCTION

•In 1999, JSON (JavaScript Object Notation)—a simple way to represent JavaScript objects as strings—was introduced as an alternative to XML as a data-exchange technique.

•JSON has gained acclaim due to its simple format, making objects easy to read, create and parse.

JSON INTRODUCTION , SYNTAX :

•Each JSON object is represented as a list of property names and values contained in curly braces, in the following format:

{propertyName1 : value1, propertyName2 : value2 }

•Arrays are represented in JSON with square brackets in the following format

[value0, value1, value2]

•Each value can be a string,a number, a JSON object, true, false or null.
•To appreciate the simplicity of JSON data, examine this representation of an array of address-book entries.

```
[ { first: 'Cheryl', last: 'Black' },
{ first: 'James', last: 'Blue' },
{ first: 'Mike', last: 'Brown' },
{ first: 'Meg', last: 'Gold' } ]
```

•JSON provides a straightforward way to manipulate objects in JavaScript, and many other programming languages now support this format.
•In addition to simplifying object creation, JSON allows programs to easily extract data and efficiently transmit it across the Internet.
•JSON integrates especially well with Ajax applications

JSON -FUNCTION FILES

A common use of JSON is to read data from a web server, and display the data in a web page.

JSON Example :

```
<div id="id01"></div>
<script>
function myFunction(arr) {
var out = "";
var i;
for(i = 0; i<arr.length; i++) {
out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
}
document.getElementById("id01").innerHTML = out;
}
</script>
<script src="myTutorials.js"></script>
```

Explanation for Example :

1: Create an array of objects.

•Use an array literal to declare an array of objects.
•Give each object two properties: display and url.
•Name the array myArray

```
var myArray = [
{
"display": "JavaScript Tutorial",
"url": "https://www. dietel.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www. dietel.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www.dietel.com/css/default.asp"
}
]
```

2: Create a JavaScript function to display the array

Create a function myFunction() that loops the array objects, and display the content as HTML links.

```
myFunction()
function myFunction(arr) {
var out = "";
var i;
for(i = 0; i < arr.length; i++) {
out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
}
document.getElementById("id01").innerHTML = out;
}
```

Call myFunction() with myArray as argument:

Example :

```
myFunction(myArray);
```

3: Use an array literal as the argument (instead of the array variable):

Call myFunction() with an array literal as argument Calling myFunction()

```
myFunction([
{
"display": "JavaScript Tutorial",
"url": "https://www.dietel.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www. dietel.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www. dietel.com/css/default.asp"
}
]);
```

4: Put the function in an external js file

Put the function in a file named myTutorials.js

```
myFunction([
{
"display": "JavaScript Tutorial",
"url": "https://www.dietel.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www. dietel.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www. dietel.com/css/default.asp"
}
]);
```

Add the external script to the page (instead of the function call):
Add External Script: `<script src="myTutorials.js"></script>`

<u>SQL</u>

•SQL is a database computer language designed for the retrieval and management of data in a relational database.
•SQL stands for Structured Query Language.
•SQL is a computer language for storing, manipulating and retrieving data stored in a relational database.
•SQL is the standard language for Relational Database System.
•All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
•SQL consists of many types of statements, which may be classed as sublanguages, commonly: a data query language(DQL), a data definition language (DDL), a data control language (DCL), and a data manipulation language(DML).
•The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control.

<u>RDBMS</u>

•RDBMS stands for Relational Database Management System.
•RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
•The data in RDBMS is stored in database objects called tables.
•A table is a collection of related data entries and it consists of columns and rows.
•Look at the "Customers" table:
Example :
SELECT * FROM Customers;

SQL Statements:

•Most of the actions we need to perform on a database are done with SQL statements.
•The following SQL statement selects all the records in the "Customers" table:
Example: SELECT * FROM Customers;
•Some database systems require a semicolon at the end of each SQL statement.

•Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.
Some of The Most Important SQL Commands :

- •SELECT - extracts data from a database
- •UPDATE - updates data in a database
- •DELETE - deletes data from a database
- •INSERT INTO - inserts new data into a database
- •CREATE DATABASE - creates a new database
- •ALTER DATABASE - modifies a database
- •CREATE TABLE - creates a new table
- •ALTER TABLE - modifies a table
- •DROP TABLE - deletes a table
- •CREATE INDEX - creates an index (search key)
- •DROP INDEX - deletes an index

The SQL SELECT Statement:

•The SELECT statement is used to select data from a database.

•The data returned is stored in a result table, called the result-set.

SELECT Syntax: SELECT column1,column2, ...FROM table_name;

•Here, column1, column2, ... are the field names of the table. If we want to select all the

fields available in the table, use the **following syntax**: SELECT * FROM table_name;

SELECT CustomerName, City FROM Customers;

WHERE Clause Example :

•The following SQL statement selects all the customers from the country "Mexico", in the

"Customers" table: SELECT * FROM Customers WHERE Country='Mexico';

INSERT INTO Example:

•The following SQL statement inserts a new record in the "Customers" table:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

UPDATE Table:
•The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.
UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

SQL DELETE Example:
•The following SQL statement deletes the customer "Alfred" from the "Customers" table: DELETE FROM Customers WHERE CustomerName='Alfred';

Applications of SQL :

SQL is one of the most widely used query language over the databases.

- ✓ Allows users to access data in the relational database management systems.
- ✓ Allows users to describe the data.
- ✓ Allows users to define the data in a database and manipulate that data.
- ✓ Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- ✓ Allows users to create and drop databases and tables.
- ✓ Allows users to create view, stored procedure, functions in a database.
- ✓ Allows users to set permissions on tables, procedures and views.

END