UNIT-II
JAVASCRIPT AND JQUERY

INTRODUCTION:

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Fire fox, Chrome, Opera, and Safari.

What is JavaScript?

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

What can a JavaScript do?

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

JavaScript is Case Sensitive:

A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar". JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

WHITE SPACE

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines are equivalent:

```
name="Hege";
name = "Hege";
```

OPERATORS USED IN JAVASCRIPT:

1) JAVASCRIPT ARITHMETIC OPERATORS

Arithmetic operators are used to perform arithmetic between variables and/or values. Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=y++	x=6
--	Decrement	x=y++	x=4

2) JAVASCRIPT ASSIGNMENT OPERATORS:

Assignment operators are used to assign values to JavaScript variables. Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings:

→To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";     txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

→To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

→After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

→ *The rule is: **If you add a number and a string, the result will be a string!***

3) COMPARISON OPERATORS:

Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that **x=5**, the table below explains the comparison operators:

Operator	**Description**	**Example**
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

4) LOGICAL OPERATORS:

Logical operators are used to determine the logic between variables or values. Given that **x=6 and y=3.**

Operator	**Description**	**Example**
&&	And	(x < 10 && y > 1) is true
||	Or	(x==5 || y==5) is false
!	not	!(x==y) is true

5) CONDITIONAL OPERATORS:

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

Syntax:

```
variablename=(condition)?value1:value2
```

Example:

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President" else it will be assigned "Dear".

CONDITIONAL STATEMENTS

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
In JavaScript we have the following conditional statements:

i) **if statement** - use this statement to execute some code only if a specified condition is true
ii) **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
iii) **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
iv) **switch statement** - use this statement to select one of many blocks of code to be executed

i) IF STATEMENT

Syntax:

```
if (condition)
  {
  code to be executed if condition is true
  }
```

Example:

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

ii) IF...ELSE STATEMENT

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

Syntax:

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

Example:

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date();
var time = d.getHours();
if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

iii) IF...ELSE IF...ELSE STATEMENT:

Syntax:

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example:

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  document.write("<b>Good morning</b>");
else if (time>10 && time<16)
  document.write("<b>Good day</b>");
else
  document.write("<b>Hello World!</b>");
</script>
```

iv) SWITCH STATEMENT:

Syntax:

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

Example:

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5: document.write("Finally Friday");
            break;
case 6: document.write("Super Saturday");
            break;
case 0: document.write("Sleepy Sunday");
            break;
default: document.write("I'm looking forward to this weekend!");
}
</script>
```

JAVASCRIPT POPUP BOXES:

JavaScript has three kinds of popup boxes:

i) Alert Box
ii) Confirm Box or Message Box, and
iii) Prompt Box.

i) ALERT BOX:

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax:
alert("sometext");

Example:
```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("You are Clicked Me ☺");
}
</script>
</head>
<body>
<input type="button" onClick="show_alert()" value="Click Me ☺" />
</body>
</html>
```

ii) CONFIRM BOX OR MESSAGE BOX:

A confirm box is often used if you want the user to verify or accept something. The user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:
confirm("sometext");

Example:
```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
```

```
 {
 alert("You pressed OK!");
 }
else
 {
 alert("You pressed Cancel!");
 }
}
</script>
</head>
<body>
<input type="submit" onClick="show_confirm()" value="Clicked Me ☺" />
</body>
</html>
```

iii) PROMPT BOX

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

```
prompt("sometext","defaultvalue");
```

Example:

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
 {
 document.write("Hello " + name + "! How are you today?");
 }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

JAVASCRIPT FUNCTIONS

To keep the browser from executing a script when the page loads, you can put your script into a function. A function contains code that will be executed by an event or by a call to the function. You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file). Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax:

function *function-name*(*var1*,*var2*,...,*varX*)
{
some code;
}

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.
Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Example:

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me☺" onClick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

THE RETURN STATEMENT:

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example:
```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

THE LIFETIME OF JAVASCRIPT VARIABLES:

If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared. If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JAVASCRIPT LOOPS

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this. In JavaScript, there are two different kinds of loops:

i) **for** - loops through a block of code a specified number of times
ii) **while** - loops through a block of code while a specified condition is true

i) FOR LOOP

Syntax:
```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

Example:

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

ii) WHILE LOOP

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

Syntax:

```
while (var<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing operator.

Example:

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script> </body></html>
```

iii) DO...WHILE LOOP

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax:
```
do
{
  code to be executed
 } while (var<=endvalue);
```

Example:

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:
```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script>
</body>
</html>
```

iv) THE BREAK STATEMENT

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example:
```
<html>
<body> <script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    break;
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body></html>
```

v) THE CONTINUE STATEMENT

The continue statement will break the current loop and continue with the next value.

Example:

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

vi) JAVASCRIPT FOR...IN STATEMENT

The **for...in** statement loops through the elements of an array or through the properties of an object.

Syntax:

```
for (variable in object)
  {
  code to be executed
  }
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example:

Use the for...in statement to loop through an array:

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
```

```
for (x in mycars)
  {
  document.write(mycars[x] + "<br />");
  }
</script>
</body>
</html>
```

JAVASCRIPT EVENTS:

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

ONLOAD AND ONUNLOAD:

The onLoad and onUnload events are triggered when the user enters or leaves the page. The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

ONFOCUS, ONBLUR AND ONCHANGE:

The onFocus, onBlur and onChange events are often used in combination with validation of form fields. Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onChange="checkEmail()">
```

ONSUBMIT:

The onSubmit event is used to validate ALL form fields before submitting it. Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

ONMOUSEOVER AND ONMOUSEOUT:

onMouseOver and onMouseOut are often used to create "animated" buttons. Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver event');return false"><img src="w3s.gif" alt="W3Schools" /></a>
```

JAVASCRIPT TRY...CATCH STATEMENT

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax:

```
try
  {
  //Run some code here
  }
catch(err)
  {
  //Handle errors here
  }
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Examples:

The example below is supposed to alert "Welcome guest!" when the button is clicked. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs. The catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

Example:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try   {
  adddlert("Welcome guest!");
  }
catch(err) {
  txt="There was an error on this page.\n\n";
  txt+="Click OK to continue viewing this page,\n";
  txt+="or Cancel to return to the home page.\n\n";
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/";
    }
}
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

THE THROW STATEMENT

The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

Throw(exception)

The exception can be a string, integer, Boolean or an object. Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example:

The example below determines the value of a variable called x. If the value of x is higher than 10, lower than 0, or not a number, we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
 {
 if(x>10)
   {
   throw "Err1";
   }
 else if(x<0)
   {
   throw "Err2";
   }
 else if(isNaN(x))
   {
   throw "Err3";      }    }
catch(er)
 {
 if(er=="Err1")
   {
   alert("Error! The value is too high");
   }
 if(er=="Err2")
   {
   alert("Error! The value is too low");
   }
 if(er=="Err3")
   {
   alert("Error! The value is not a number");
```

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

JAVASCRIPT SPECIAL CHARACTERS

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: **We are the so-called**

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north. Here is another example:

```
document.write ("You \& I are singing!");
```

The example above will produce the following output:

```
You & I are singing!
```

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	**Outputs**
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	Tab
\b	backspace
\f	form feed

JAVASCRIPT OBJECTS INTRODUCTION:

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

OBJECT ORIENTED PROGRAMMING:

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types. However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail. Note that an object is just a special kind of data. An object has properties and methods.

Properties:

Properties are the values associated with an object. In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

The output of the code above will be: 12

Methods:

Methods are the actions that can be performed on objects. In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

```
HELLO WORLD!
```

STRING OBJECT:

The String object is used to manipulate a stored piece of text.

Examples of use:

The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";
document.write(txt.length);
```

The code above will result in the following output: 12

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";
document.write(txt.toUpperCase());
```

The code above will result in the following output: HELLO WORLD!

String Object Examples:

1. **Return The Length of A String**

```
<html>
<body>
<script type="text/javascript">
var txt = "Hello World!";
document.write(txt.length);
</script>
</body>
</html>
```

Output: 12

2. **Style Strings**

```
<html>
<body>
<script type="text/javascript">
var txt = "Hello World!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");
document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");
document.write("<p>Fontcolor: " + txt.fontcolor("green") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(6) + "</p>");
document.write("<p>Subscript: " + txt.sub() + "</p>");
document.write("<p>Superscript: " + txt.sup() + "</p>");
document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>");
document.write("<p>Blink: " + txt.blink() + " (does not work in IE, Chrome, or
Safari)</p>");
</script>
</body>
</html>
```

Output:

Big: Hello World!
Small: Hello World!
Bold: **Hello World!**

Italic: *Hello World!*
Fixed: `Hello World!`
Strike: ~~Hello World!~~
Fontcolor: Hello World!
Fontsize: Hello World!
Subscript: $_{\text{Hello World!}}$
Superscript: $^{\text{Hello World!}}$
Link: [Hello World!]
Blink: Hello World! (does not work in IE, Chrome, or Safari)

3. **Return the Position of the First Occurrence of A Text in A String - indexof():**

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!";
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("WORLD") + "<br />");
document.write(str.indexOf("world"));
</script>
</body>
</html>
```

Output: 10 -1 6

4. **Search for A Text in A String and Return the Text If Found - Match():**

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!";
document.write(str.match("world") + "<br />");
document.write(str.match("World") + "<br />");
document.write(str.match("worlld") + "<br />");
document.write(str.match("world!"));
</script>
</body>
</html>
```

Output: world null null world!

5. **Replace Characters in A String - Replace() :**

```
<html>
<body>
<script type="text/javascript">
var str="Visit Microsoft!";
document.write(str.replace("Microsoft","W3Schools"));
```

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

Output: Visit W3Schools!

6. **Convert a String to Lowercase Letters:**

```
<html>
<body>
<script type="text/javascript">
var str="Hello World!";
document.write(str.toLowerCase());
</script>
 </body>
</html>
```

Output: hello world!

JAVASCRIPT DATE OBJECT

The Date object is used to work with dates and times.

➔ Create a Date Object

The Date object is used to work with dates and times. Date objects are created with the Date() constructor. There are four ways of instantiating a date:

```
New Date() // current date and time
new Date(milliseconds) //milliseconds since 1970/01/01
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

Most parameters above are optional. Not specifying causes 0 to be passed in. Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time. All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.
Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00")
d2 = new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

➔ Set Dates

We can easily manipulate the date by using the methods available for the Date object. In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

➔ **Compare Two Dates**

The Date object is also used to compare two dates. The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();

if (myDate>today)
  {
  alert("Today is before 14th January 2010");
  }
else
  {
  alert("Today is after 14th January 2010");
  }
```

DATE OBJECT EXAMPLES:

1. **Use Date() to Return Today's Date And Time**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d);
</script>
</body>
</html>
```

Output: Wed Jan 12 2011 14:38:08 GMT+0530 (India Standard Time)

2. **Use getTime() to Calculate the Years Since 1970**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d.getTime() + " milliseconds since 1970/01/01");
</script>
```

```
</body>
</html>
```
Output: 1294823298285 milliseconds since 1970/01/01

3. **Use setFullYear() to Set a Specific Date**

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
d.setFullYear(1992,10,3);
document.write(d);
</script>
</body>
</html>
```
Output: Tue Nov 03 1992 14:40:15 GMT+0530 (India Standard Time)

4. **Use toUTCString() to Convert Today's Date (according to UTC) to A String**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write("Original form: ");
document.write(d + "<br />");
document.write("To string (universal time): ");
document.write(d.toUTCString());
</script>
</body>
</html>
```
Output:
Original form: Wed Jan 12 2011 14:38:17 GMT+0530 (India Standard Time)
To string (universal time): Wed, 12 Jan 2011 09:08:17 GMT

5. **Use getDay() and An Array to Write a Weekday, and Not Just A Number**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
```

```
weekday[6]="Saturday";
document.write("Today is " + weekday[d.getDay()]);
</script>
</body>
</html>
```

Output: Today is Wednesday

6. **Display a Clock**

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Output: 14:42:13

JAVASCRIPT ARRAY OBJECT

The Array object is used to store multiple values in a single variable. An array is a special variable, which can hold more than one value, at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array! An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own ID so that it can be easily accessed.

➔ Create an Array

An array can be defined in three ways. The following code creates an Array object called myCars:

1st Method:

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";       // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

2nd Method:

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed array
```

3rd Method:

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

➔ Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0. The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

➔ Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

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

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output: Opel

Array Object Examples:

1. **Program for array concatenation**

```
<html>
<body>
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);
</script>
</body>
</html>
```

2. **Program for array concatenation with multiple arrays.**

```
<html>
<body>
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var brothers = ["Stale", "Kai Jim", "Borge"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(brothers, children);
document.write(family);
</script>
</body>
 </html>
```

3. **Program for array join operation.**

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));
</script>
</body> </html>
```

4. **Program for array pop.**

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.pop() + "<br />");
document.write(fruits);
</script>
</body>
 </html>
```

5. **Program for array push.**

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("Kiwi") + "<br />");
document.write(fruits.push("Lemon","Pineapple") + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

6. **Program for array reverse.**

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.reverse());
</script>
</body>
</html>
```

7. **Program for array shift.**

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.shift() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.shift() + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

8. Program for array slice.

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.slice(0,1) + "<br />");
document.write(fruits.slice(1) + "<br />");
document.write(fruits.slice(-2) + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

9. Program for array sort().

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());
</script>
</body>
</html>
```

10. Program for array toString().

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.toString());
</script>
</body>
</html>
```

JAVASCRIPT MATH OBJECT

The Math object allows you to perform mathematical tasks. The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Mathematical Constants:

JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E. You may reference these constants from your JavaScript like this:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Mathematical Methods:

In addition to the mathematical constants that can be accessed from the Math object there are also several methods available. The following example uses the round() method of the Math object to round a number to the nearest integer:

```
document.write(Math.round(4.7));
```

Output: 5

The following example uses the random() method of the Math object to return a random number between 0 and 1:

```
document.write(Math.random());
```

Output: 0.9306157949324372

The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

```
document.write(Math.floor(Math.random()*11));
```

Output: 6

MATH OBJECT EXAMPLES

1. **Use round() to Round a Number:**

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));
</script>
</body>
```

```
</html>
```

2. **Use random() to Return a Random Number Between 0 and 1**:

```
<html>
<body>
<script type="text/javascript">
//return a random number between 0 and 1
document.write(Math.random() + "<br />");
//return a random integer between 0 and 10
document.write(Math.floor(Math.random()*11));
</script>
</body>
</html>
```

3. **Use max() to return the Number With the Highest Value of Two Specified Numbers:**

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(5,10) + "<br />");
document.write(Math.max(0,150,30,20,38) + "<br />");
document.write(Math.max(-5,10) + "<br />");
document.write(Math.max(-5,-10) + "<br />");
document.write(Math.max(1.5,2.5));
</script>
</body>
</html>
```

4. **Use min() to Return the Number With the Lowest Value of Two Specified Numbers:**

```
<html>
<body>
<script type="text/javascript">
document.write(Math.min(5,10) + "<br />");
document.write(Math.min(0,150,30,20,38) + "<br />");
document.write(Math.min(-5,10) + "<br />");
document.write(Math.min(-5,-10) + "<br />");
document.write(Math.min(1.5,2.5));
</script>
</body>
</html>
```

5. **Convert Celsius to Fahrenheit:**

```
<html>
<head>
<script type="text/javascript">
function convert(degree)
```

```
{
if (degree=="C")
 {
 F=document.getElementById("c").value * 9 / 5 + 32;
 document.getElementById("f").value=Math.round(F);
 }
else
 {
 C=(document.getElementById("f").value -32) * 5 / 9;
 document.getElementById("c").value=Math.round(C);
 }
}
</script>
</head>
<body>
<p></p><b>Insert a number into one of the input fields
below:</b></p>
<form>
<input id="c" name="c" onkeyup="convert('C')"> degrees
Celsius<br />
equals<br />
<input id="f" name="f" onkeyup="convert('F')"> degrees
Fahrenheit
</form>
<p>Note that the <b>Math.round()</b> method is used, so that the
result will be returned as an integer.</p>
</body>
</html>
```

JAVASCRIPT BOOLEAN OBJECT:

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object

JAVASCRIPT WINDOW OBJECT:

The window object represents an open window in a browser.

Note: There is no public standard that applies to the Window object, but all major browsers support it.

Window Object Properties:

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

Window Object Methods:

Method	**Description**
alert()	Displays an alert box with a message and an OK button
blur()	Removes focus from the current window
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
createPopup()	Creates a pop-up window
focus()	Sets focus to the current window
open()	Opens a new browser window
print()	Prints the content of the current window

prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height

Window Object Examples

1. **Display an alert box:**

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

2. **Display a prompt box:**

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt
box" />
</body>
</html>
```

3. **Display a confirm box, and alert what the visitor clicked:**

```
<html>
<head>
```

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

4. **Create a pop-up window Open a new window when clicking on a button:**

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
window.open("http://kishor.ucoz.com");
}
</script>
</head>
<body>
<form>
<input type=button value="Open Window" onclick="open_win()">
</form>
</body>
</html>
```

5. **Open a new window and control its appearance:**

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
```

```
window.open("http://www.w3schools.com","_blank","toolbar=yes,
location=yes, directories=no, status=no, menubar=yes, scrollbars=yes,
resizable=no, copyhistory=yes, width=400, height=400");
}
</script>
</head>
<body>
<form>
<input type="button" value="Open Window" onclick="open_win()">
</form>
</body>
</html>
```

6. **Open multiple new windows:**

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
window.open("http://www.microsoft.com/");
window.open("http://kishor.ucoz.com");
}
</script>
</head>
<body>
<form>
<input type=button value="Open Windows" onclick="open_win()">
</form>
</body>
</html>
```

7. **Close the new window:**

```
<html>
<head>
<script type="text/javascript">
function openWin()
{
myWindow=window.open("","","width=200,height=100");
myWindow.document.write("<p>This is 'myWindow'</p>");
}
function closeWin()
{
myWindow.close();
}
</script>  </head>
<body>
```

```
<input type="button" value="Open 'myWindow'" onclick="openWin()" />
<input type="button" value="Close 'myWindow'" onclick="closeWin()"/>
</body>
</html>
```

8. Print the current page:

```
<html>
<head>
<script type="text/javascript">
function printpage()
{
window.print();
}
</script>
</head>
<body>
<input type="button" value="Print this page" onclick="printpage()" />
</body>
</html>
```

9. A simple timing:

```
<html>
<head>
<script type="text/javascript">
function timeText()
{
var t1=setTimeout("document.getElementById('txt').value='2
seconds!'",2000);
var t2=setTimeout("document.getElementById('txt').value='4
seconds!'",4000);
var t3=setTimeout("document.getElementById('txt').value='6
seconds!'",6000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Display timed text!" onclick="timeText()"/>
<input type="text" id="txt" />
</form>
<p>Click the button above. The input field will tell you when two, four,
and six seconds have passed…..</p>
</body>
</html>
```

JQUERY INTRODUCTION:

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: **Write less, do more**. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:

- **DOM manipulation:** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
- **Event handling:** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- **AJAX Support:** The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
- **Animations:** The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- **Lightweight:** The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- **Cross Browser Support:** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology:** The jQuery supports CSS3 selectors and basic XPath syntax.

Installation of jQuery:

There are two ways to use jQuery.

- **Local Installation** − You can download jQuery library on your local machine and include it in your HTML code.
- **CDN Based Version** − You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Local Installation:

- Go to the https://jquery.com/download/ to download the latest version available.
- Now, insert downloaded jquery-2.1.3.min.js file in a directory of your website, e.g. C:/your-website-directory/jquery/jquery-2.1.3.min.js.

Example:

Now, you can include jQuery library in your HTML file as follows:

```
<html>
   <head>
     <title>The jQuery Example</title>
     <script type="text/javascript"src="/jquery/jquery-
     2.1.3.min.js"></script>
     <script type="text/javascript">
     $(document).ready(function()
                {
```

```
          document.write("Hello, World!");
        });
      </script>
    </head>
    <body>
      <h1>Hello</h1>
    </body>
</html>
```

This will produce the following result –

```
      Hello, World!
```

CDN Based Version:

You can include jQuery library into your HTML code directly from Content Delivery Network (CDN). Google and Microsoft provides content deliver for the latest version. We are using Google CDN version of the library.

Example:

Now, you can include jQuery library in your HTML file as follows:

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

This will produce the following result –

```
Hello, World!
```

Calling jQuery Library Functions:

If you want to an event work on your page, you should call it inside the $(document).ready() function. Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

To do this, we register a ready event for the document as follows:

```
$(document).ready(function() {
        //write the stuff when DOM is ready
});
```

To call upon any jQuery library function, use HTML script tags as shown below:

```
<html>
<head>
     <title>The jQuery Example</title>
     <script type="text/javascript" src="/jquery/jquery
     1.3.2.min.js"></script>
     <script type="text/javascript" language="javascript">
     $(document).ready(function({
       $("div").click(function() {
          Alert("Hello world!");
       });
     });
     </script>
 </head>
 <body>
     <div id="newdiv">
          Click on this to see a dialogue box.
     </div>
 </body>
 </html>
```

Creating and Executing Custom Scripts:

It is better to write our custom code in custom JavaScript file: **custom.js**, as follows:

```
/* Filename: custom.js */
$(document).ready(function() {
   $("div").click(function() {
      alert("Hello world!");
   });
});
```

Now we can include custom.js file in our HTML file as follows:

```
 <html>
 <head>
 <title>The jQuery Example</title>
 <script type="text/javascript" src="/jquery/jquery-
    1.3.2.min.js"></script>
 <script type="text/javascript" src="/jquery/custom.js"></script>
 </head>
 <body>
 <div id="newdiv">
 Click on this to see a dialogue box.
 </div>
 </body>
</html>
```

This will produce the following result:

```
 Click on this to see a dialogue box.
```

jQuery Selectors:

The jQuery library binds the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).

A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.

The $() Factory Function:

All type of selectors available in jQuery, always start with the dollar sign and parentheses: $(). The factory function $() makes use of the following three building blocks while selecting elements in a given document:

Selector	Description
Tag Name	Represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.
Tag ID	Represents a tag available with the given ID in the DOM. For example $('#some- id') selects the single element in the document that has an ID of some-id.
Tag Class	Represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.

All the above items can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

NOTE: The factory function $() is a synonym of jQuery() function. So in case you are using any other JavaScript library where $ sign is conflicting with something else then you can replace $ sign by jQuery name and you can use function jQuery() instead of $().

Example:

Following is a simple example which makes use of Tag Selector. This would select all the elements with a tag name **p**.

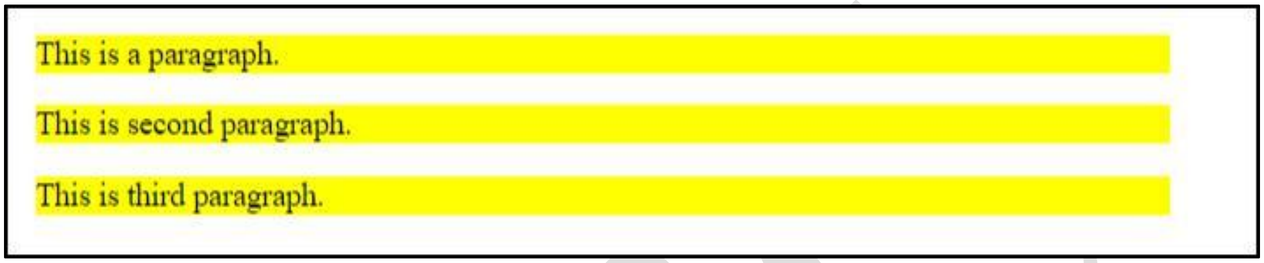
```
<html>

<head>

<title>the title</title>
   <script type="text/javascript" src="/jquery/jquery-
   1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      var pars = $("p");
      for( i=0; i<pars.length; i++ ){
         alert("Found paragraph: " + pars[i].innerHTML);
      }
   });
   </script>
</head>
<body>
```

```
    <div>
      <p class="myclass">This is a paragraph.</p>
      <p id="myid">This is second paragraph.</p>
      <p>This is third paragraph.</p>
    </div>
  </body>
</html>
```

This will produce the the following result:



This is a paragraph.

This is second paragraph.

This is third paragraph.

Using of Selectors:

The selectors are very useful and would be required at every step while using jQuery. They get the exact element that you want from your HTML document.

Following table lists down few basic selectors and explains them with examples.

Selector	Description
Name	Selects all elements which match with the given element **Name**.
#ID	Selects a single element which matches with the given **ID**.
.Class	Selects all elements which matches with the given **Class**.
Universal (*)	Selects all elements available in a DOM.
Multiple Elements E, F,G	Selects the combined results of all the specified selectors **E, F** or **G**.

jQuery – Element Name Selector:

The element selector selects all the elements that have a tag name of T.

Syntax:

Here is the simple syntax to use this selector –

```
$('tagname')
```

Parameters:

Here is the description of all the parameters used by this selector –

- **tagname** – Any standard HTML tag name like div, p, em, img, li etc.

Returns:

Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example:

- **$('p')** – Selects all elements with a tag name of **p** in the document.
- **$('div')** – Selects all elements with a tag name of **div** in the document.

Following example would select all the divisions and will apply yellow color to their background

```
 <html>
    <head>
       <title>The Selecter Example</title>
       <script type="text/javascript"
            src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquer
            y.min.js">
       </script>
       <script type="text/javascript" language="javascript">
       $(document).ready(function() {
            /* This would select all the divisions */
            $("div").css("background-color", "yellow");
         });
       </script>
    </head>
    <body>
       <div class="big" id="div1">
         <p>This is first division of the DOM.</p> </div>
       <div class="medium" id="div2">
         <p>This is second division of the DOM.</p> </div>
       <div class="small" id="div3">
         <p>This is third division of the DOM</p>
        </div>
    </body>
</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery – Element ID Selector:

Description:

The element ID selector selects a single element with the given id attribute.

Syntax:

Here is the simple syntax to use this selector –

Parameters:

Here is the description of all the parameters used by this selector –

- **`Elementid:`** This would be an element ID. If the id contains any special characters like periods or colons you have to escape those characters with backslashes.

Returns:

Like any other jQuery selector, this selector also returns an array filled with the found element.

Example:

- **$('#myid')** – Selects a single element with the given id myid.
- **$('div#yourid')** – Selects a single division with the given id yourid.

Following example would select second division and will apply yellow color to its background as below:

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
>
      </script>
      <script type="text/javascript" language="javascript">
      $(document).ready(function() {
           /* This would select second division only*/
           $("#div2").css("background-color", "yellow");
        });
      </script>
   </head>
   <body>
      <div class="big" id="div1">
           <p>This is first division of the DOM.</p>
      </div>
      <div class="medium" id="div2">
        <p>This is second division of the DOM.</p>
       </div>
      <div class="small" id="div3">
        <p>This is third division of the DOM.</p>
      </div>
   </body>
</html>
```

This will produce the following result:

```
This is first division of the DOM.
This is second division of the DOM.
This is third division of the DOM.
```

jQuery - Element Class Selector:

Description:

The element class selector selects all the elements which match with the given class of the elements.

Syntax:

Here is the simple syntax to use this selector:

$('.classid')

Parameters:
Here is the description of all the parameters used by this selector –

- **classid** − This is class ID available in the document.

Returns:
Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example:

- **$('.big')** − Selects all the elements with the given class ID big.
- **$('p.small')** − Selects all the paragraphs with the given class ID small.
- **$('.big.small')** − Selects all the elements with a class of big and small.

Following example would select all divisions with class .big and will apply yellow color to its background.

```
 <html>
   <head>
     <title>The Selecter Example</title>
     <script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
 >
     </script>
     <script type="text/javascript" language="javascript">
       $(document).ready(function() {
         /* This would select second division only*/
         $(".big").css("background-color", "yellow");
       });
     </script>
   </head>
   <body>
     <div class="big" id="div1">
       <p>This is first division of the DOM.</p> </div>
      <div class="medium" id="div2">
       <p>This is second division of the DOM.</p> </div>
     <div class="small" id="div3">
       <p>This is third division of the DOM</p> </div>
   </body>
</html>
```

This will produce the following result:

```
This is first division of the DOM.
This is second division of the DOM.
This is third division of the DOM
```

jQuery - Universal Selector:

Description:
The universal selector selects all the elements available in the document.

Syntax:
Here is the simple syntax to use this selector –

```
$('*')
```

Parameters:
Here is the description of all the parameters used by this selector

- ***** – A symbolic star.

Returns:
Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example:

- **$('*')** selects all the elements available in the document.

Following example would select all the elements and will apply yellow color to their background. Try to understand that this selector will select every element including head, body etc.

```
 <html>
    <head>
       <title>The Selecter Example</title>
       <script type="text/javascript"
 src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
 >
       </script>
       <script type="text/javascript" language="javascript">
          $(document).ready(function() {
             /* This would select all the elements */
             $("*").css("background-color", "yellow");
          });
       </script>
    </head>
    <body>
       <div class="big" id="div1">
          <p>This is first division of the DOM.</p> </div>
       <div class="medium" id="div2">
             <p>This is second division of the DOM.</p>
       </div>
       <div class="small" id="div3">
          <p>This is third division of the DOM</p> </div>
    </body>
</html>
```

This will produce the following result:

```
This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM
```

jQuery – Multiple Elements Selector:

Description:
This Multiple Elements selector selects the combined results of all the specified selectors E, F or G. You can specify any number of selectors to combine into a single result. Here order of the DOM elements in the jQuery object aren't necessarily identical.

Syntax:
Here is the simple syntax to use this selector −
`$('E, F, G,....')`

Parameters:
Here is the description of all the parameters used by this selector –

- E − Any valid selector
- F − Any valid selector
- G − Any valid selector

Returns:
Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example:

- **$('div, p')** − selects all the elements matched by **div** or **p**.
- **$('p strong, .myclass')** − selects all elements matched by **strong** that are descendants of an element matched by **p** as well as all elements that have a class of **myclass**.
- **$('p strong, #myid')** − selects a single element matched by **strong** that is descendant of an element matched by **p** as well as element whose id is **myid**.

Following example would select elements with class ID **big** and element with ID **div3** and will apply yellow color to its background –

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"
>
      </script>
      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $(".big, #div3").css("background-color", "yellow");
         });
      </script>
   </head>
   <body>
      <div class="big" id="div1">
         <p>This is first division of the DOM.</p> </div>
       <div class="medium" id="div2">
         <p>This is second division of the

      DOM.</p> </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p> </div>
```

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

This will produce the following result:

This is first division of the DOM.
This is second division of the DOM.
This is third division of the DOM

jQuery Selector Examples:

Syntax	Description
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first <li> element of the first <ul>
$("ul li:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

jQuery DOM:

JQuery provides methods to manipulate DOM in efficient way. You do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.

JQuery provides methods such as .attr(), .html(), and .val() which act as getters, retrieving information from DOM elements for later use.

Content Manipulation:

The **html( )** method gets the html contents (innerHTML) of the first matched element.

Here is the syntax for the method −

selector.html( )

Example:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src =
"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               var content = $(this).html();
               $("#result").text( content );
            });
         });
      </script>

      <style>
         #division{ margin:10px;padding:12px; border:2px solid #666;
width:60px;}
      </style>
   </head>

   <body>
      <p>Click on the square below:</p>
      <span id = "result"> </span>

      <div id = "division" style = "background-color:blue;">
         This is Blue Square!!
      </div>
   </body>
</html>
```

text(val):
The **text(val)** method sets the combined text contents of all matched elements.
Syntax:

```
                              selector.text( val )
```

Example:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).text( "<h1>Click another square</h1>");
            });
         });
      </script>

      <style>
         .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>
   </head>

   <body>
      <p>Click on any square below to see the result:</p>

      <div class = "div" style = "background-color:blue;"></div>
      <div class = "div" style = "background-color:green;"></div>
      <div class = "div" style = "background-color:red;"></div>
   </body>
</html>
```

jQuery Events:
jQuery is tailor-made to respond to events in an HTML page. All the different visitors' actions that a web page can respond to are called events. An event represents the precise moment when something happens.
Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
Click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Syntax:
In jQuery, most DOM events have an equivalent jQuery method. To assign a click event to all paragraphs on a page, you can do this:

```
$(selector).click();
```

Sometime you need to define what should happen when the event fires, then you must pass a function to the event:

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

Commonly used jQuery Events:
$(document).ready():
The `$(document).ready()` method allows us to execute a function when the document is fully loaded.
Example:

```
$(document).ready(function(){
        alert("Hello World!");
});
```

click():
The click() method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element.

Example:

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

jQuery Attributes:

Some of the most basic components we can manipulate when it comes to DOM elements are the properties and attributes assigned to those elements.

Most of these attributes are available through JavaScript as DOM node properties. Some of the more common properties are −

- className
- tagName
- id
- href
- title
- rel
- src

Consider the following HTML markup for an image element –

<img id = "imageid" src = "image.gif" alt = "Image" class = "myclass" title = "This is an image"/>

In this element's markup, the tag name is img, and the markup for id, src, alt, class, and title represents the element's attributes, each of which consists of a name and a value. jQuery gives us the means to easily manipulate an element's attributes and gives us access to the element so that we can also change its properties.

Get Attribute Value:

The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

Example:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            var title = $("em").attr("title");
            $("#divid").text(title);
         });
      </script>
   </head>
   <body>
      <div>
         <em title = "Bold and Brave">This is first paragraph.</em>
         <p id = "myid">This is second paragraph.</p>
         <div id = "divid"></div>
      </div>
   </body>
</html>
```

This will produce following result –
This is first paragraph.
This is second paragraph.
Bold and Brave

Set Attribute Value:
The **attr(name, value)** method can be used to set the named attribute onto all elements in the wrapped set using the passed value.
Example:
```
<html>
   <head>
      <title>The jQuery Example</title>
      <base href="https://www.tutorialspoint.com" />
      <script type = "text/javascript"
         src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("#myimg").attr("src", "/jquery/images/jquery.jpg");
         });
      </script>
   </head>
   <body>
      <div>
         <img id = "myimg" src = "/images/jquery.jpg" alt = "Sample image" />
      </div>
   </body>
</html>
```