JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

## **1) JavaScript Object by object literal**

The syntax of creating object using object literal is given below:

1. object={property1:value1,property2:value2.....propertyN:valueN}

As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

**<script>**

emp={id:102,name:"Shyam Kumar",salary:40000}

document.write(emp.id+" "+emp.name+" "+emp.salary);

**</script>**

**Output of the above example**

102 Shyam Kumar 40000

## **2) By creating instance of Object**

The syntax of creating object directly is given below:

1. var objectname=new Object();

Here, **new keyword** is used to create object.

Let’s see the example of creating object directly.

**<script>**

var emp=new Object();

emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

**</script>**

**Output of the above example**

101 Ravi 50000

## **3) By using an Object constructor**

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object.

The example of creating object by object constructor is given below.

**<script>**

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

}

e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

**</script>**

**Output of the above example**

103 Vimal Jaiswal 30000

## **Defining method in JavaScript Object**

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

The example of defining method in object is given below.

**<script>**

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

this.changeSalary=changeSalary;

function changeSalary(otherSalary){

this.salary=otherSalary;

}

}

e=new emp(103,"Sonoo Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

e.changeSalary(45000);

document.write("**<br>**"+e.id+" "+e.name+" "+e.salary);

**</script>**

[**Test it Now**](https://www.javatpoint.com/oprweb/test.jsp?filename=object4js)

#### **Output of the above example**

103 Sonoo Jaiswal 30000  
103 Sonoo Jaiswal 45000

# JavaScript Functions

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

#### **Advantage of JavaScript function**

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

## **JavaScript Function Syntax**

The syntax of declaring function is given below.

function functionName([arg1, arg2, ...argN]){

 //code to be executed

}

JavaScript Functions can have 0 or more arguments.

## **JavaScript Function Example**

Let’s see the simple example of function in JavaScript that does not has arguments.

**<script>**

function msg(){

alert("hello! this is message");

}

**</script>**

**<input** type="button" onclick="msg()" value="call function"**/>**

## **JavaScript Function Arguments**

We can call function by passing arguments. Let’s see the example of function that has one argument.

**<script>**

function getcube(number){

alert(number\*number\*number);

}

**</script>**

**<form>**

**<input** type="button" value="click" onclick="getcube(4)"**/>**

**</form>**

Top of Form

Bottom of Form

## **Function with Return Value**

We can call function that returns a value and use it in our program. Let’s see the example of function that returns value.

**<script>**

function getInfo(){

return "hello javatpoint! How r u?";

}

**</script>**

**<script>**

document.write(getInfo());

**</script>**

#### **Output of the above example**

hello javatpoint! How r u?

## **JavaScript Function Object**

In JavaScript, the purpose of **Function constructor** is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

## **Syntax**

new Function ([arg1[, arg2[, ....argn]],] functionBody)

## **Parameter**

**arg1, arg2, .... , argn** - It represents the argument used by function.

**functionBody** - It represents the function definition.

## **JavaScript Function Methods**

Let's see function methods with description.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [apply()](https://www.javatpoint.com/javascript-function-apply-method) | It is used to call a function contains this value and a single array of arguments. |
| [bind()](https://www.javatpoint.com/javascript-function-bind-method) | It is used to create a new function. |
| [call()](https://www.javatpoint.com/javascript-function-call-method) | It is used to call a function contains this value and an argument list. |
| [toString()](https://www.javatpoint.com/javascript-function-tostring-method) | It returns the result in a form of a string. |

## **JavaScript Function Object Examples**

### **Example 1**

Let's see an example to display the sum of given numbers.

**<script>**

var add=new Function("num1","num2","return num1+num2");

document.writeln(add(2,5));

**</script>**

**Output:**

7

### **Example 2**

Let's see an example to display the power of provided value.

**<script>**

var pow=new Function("num1","num2","return Math.pow(num1,num2)");

document.writeln(pow(2,3));

**</script>**

[**Test it Now**](https://www.javatpoint.com/oprweb/test.jsp?filename=JavaScriptFunctionObjectExample2)

**Output:**

8

Document Object Model

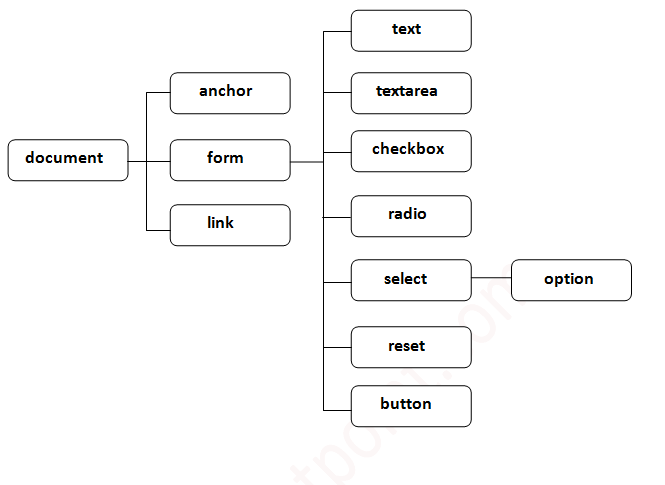
The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

it is the object of window. So

window.document

**Properties of document object**

Let's see the properties of document object that can be accessed and modified by the document object. 

## **Methods of document object**

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

|  |  |
| --- | --- |
| **Method** | **Description** |
| write("string") | writes the given string on the doucment. |
| writeln("string") | writes the given string on the doucment with newline character at the end. |
| getElementById() | returns the element having the given id value. |
| getElementsByName() | returns all the elements having the given name value. |
| getElementsByTagName() | returns all the elements having the given tag name. |
| getElementsByClassName() | returns all the elements having the given class name. |

### **Accessing field value by document object**

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.

Here, **document** is the root element that represents the html document.

**form1** is the name of the form.

**name** is the attribute name of the input text.

**value** is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

**<script** type="text/javascript"**>**

function printvalue(){

var name=document.form1.name.value;

alert("Welcome: "+name);

}

**</script>**

**<form** name="form1"**>**

Enter Name:**<input** type="text" name="name"**/>**

**<input** type="button" onclick="printvalue()" value="print name"**/>**

**</form>**

# document.getElementById() method

# The document.getElementById() method returns the element of specified id.

**<script** type="text/javascript"**>**

function getcube(){

var number=document.getElementById("number").value;

alert(number\*number\*number);

}

**</script>**

**<form>**

Enter No:**<input** type="text" id="number" name="number"**/><br/>**

**<input** type="button" value="cube" onclick="getcube()"**/>**

**</form>**

# GetElementsByClassName()

The getElementsByClassName() method is used for selecting or getting the elements through their class name value.

**<html>**

**<head>** **<h5>**DOM Methods **</h5>** **</head>**

**<body>**

**<div** class="Class"**>**

This is a simple class implementation

**</div>**

**<script** type="text/javascript"**>**

var x=document.getElementsByClassName('Class');

document.write("On calling x, it will return an arrsy-like object: **<br>**"+x);

**</script>**

**</body>**

**</html>**

### **Example of document.getElementsByName() method**

In this example, we going to count total number of genders. Here, we are using getElementsByName() method to get all the genders.

**<script** type="text/javascript"**>**

function totalelements()

{

var allgenders=document.getElementsByName("gender");

alert("Total Genders:"+allgenders.length);

}

**</script>**

**<form>**

Male:**<input** type="radio" name="gender" value="male"**>**

Female:**<input** type="radio" name="gender" value="female"**>**

**<input** type="button" onclick="totalelements()" value="Total Genders"**>**

**</form>**

### **Example of document.getElementsByTagName() method**

In this example, we going to count total number of paragraphs used in the document. To do this, we have called the document.getElementsByTagName("p") method that returns the total paragraphs.

**<script** type="text/javascript"**>**

function countpara(){

var totalpara=document.getElementsByTagName("p");

alert("total p tags are: "+totalpara.length);

}

**</script>**

**<p>**This is a pragraph**</p>**

**<p>**Here we are going to count total number of paragraphs by getElementByTagName() method.**</p>**

**<p>**Let's see the simple example**</p>**

**<button** onclick="countpara()"**>**count paragraph**</button>**

# innerHTML

In this example, we are dynamically writing the html form inside the div name having the id mylocation. We are identifing this position by calling the document.getElementById() method.

**<script** type="text/javascript" **>**

function showcommentform() {

var data="Name:**<input** type='text' name='name'**><br>**Comment:**<br><textarea** rows='5' cols='80'**></textarea>**

**<br><input** type='submit' value='Post Comment'**>**";

document.getElementById('mylocation').innerHTML=data;

}

**</script>**

**<form** name="myForm"**>**

**<input** type="button" value="comment" onclick="showcommentform()"**>**

**<div** id="mylocation"**></div>**

**</form>**

# innerText

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.

**<script** type="text/javascript" **>**

function validate() {

var msg;

if(document.myForm.userPass.value.length**>**5){

msg="good";

}

else{

msg="poor";

}

document.getElementById('mylocation').innerText=msg;

 }

**</script>**

**<form** name="myForm"**>**

**<input** type="password" value="" name="userPass" onkeyup="validate()"**>**

Strength:**<span** id="mylocation"**>**no strength**</span>**

**</form>**

# JavaScript Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When [javascript](https://www.javatpoint.com/javascript-tutorial) code is included in [HTML](https://www.javatpoint.com/html-tutorial), js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

**For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

## **Mouse events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| click | onclick | When mouse click on an element |
| mouseover | onmouseover | When the cursor of the mouse comes over the element |
| mouseout | onmouseout | When the cursor of the mouse leaves an element |
| mousedown | onmousedown | When the mouse button is pressed over the element |
| mouseup | onmouseup | When the mouse button is released over the element |
| mousemove | onmousemove | When the mouse movement takes place. |

## **Keyboard events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| Keydown & Keyup | onkeydown & onkeyup | When the user press and then release the key |

## **Form events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| focus | onfocus | When the user focuses on an element |
| submit | onsubmit | When the user submits the form |
| blur | onblur | When the focus is away from a form element |
| change | onchange | When the user modifies or changes the value of a form element |

## **Window/Document events**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| load | onload | When the browser finishes the loading of the page |
| unload | onunload | When the visitor leaves the current webpage, the browser unloads it |
| resize | onresize | When the visitor resizes the window of the browser |

## **Click Event**

**<html>**

**<head>** Javascript Events **</head>**

**<body>**

**<script** language="Javascript" type="text/Javascript"**>**

<!--

function clickevent()

{

document.write("This is JavaTpoint");

}

//--**>**

**</script>**

**<form>**

**<input** type="button" onclick="clickevent()" value="Who's this?"**/>**

**</form>**

**</body>**

**</html>**

## **MouseOver Event**

**<html>**

**<head>**

**<h1>** Javascript Events **</h1>**

**</head>**

**<body>**

**<script** language="Javascript" type="text/Javascript"**>**

<!--

function mouseoverevent()

{

alert("This is JavaTpoint");

}

//--**>**

**</script>**

**<p** onmouseover="mouseoverevent()"**>** Keep cursor over me**</p>**

**</body>**

**</html>**

## **Focus Event**

**<html>**

**<head>** Javascript Events**</head>**

**<body>**

**<h2>** Enter something here**</h2>**

**<input** type="text" id="input1" onfocus="focusevent()"**/>**

**<script>**

<!--

function focusevent()

{

document.getElementById("input1").style.background=" aqua";

}

//--**>**

**</script>**

**</body>**

**</html>**

## **Keydown Event**

**<html>**

**<head>** Javascript Events**</head>**

**<body>**

**<h2>** Enter something here**</h2>**

**<input** type="text" id="input1" onkeydown="keydownevent()"**/>**

**<script>**

<!--

function keydownevent()

{

document.getElementById("input1");

alert("Pressed a key");

}

//--**>**

**</script>**

**</body>**

**</html>**

## **Load event**

**<html>**

**<head>**Javascript Events**</head>**

**</br>**

**<body** onload="window.alert('Page successfully loaded');"**>**

**<script>**

<!--

document.write("The page is loaded successfully");

//--**>**

**</script>**

**</body>**

**</html>**

**jQuery**

* jQuery is a small and lightweight JavaScript library.
* jQuery is cross-platform.
* jQuery means "write less do more".
* jQuery simplifies AJAX call and DOM manipulation.

## **Example**

In this tutorial, you will get a lot of jQuery examples to understand the topic well. Let's see a simple jQuery example.

*File: firstjquery.html*

<!DOCTYPE html>

<html>

<head>

 <title>First 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" language="javascript">

 $(document).ready(function() {

 $("p").css("background-color", "pink");

 });

 </script>

 </head>

<body>

<p>This is first paragraph.</p>

<p>This is second paragraph.</p>

<p>This is third paragraph.</p>

</body>

</html>

Output:

This is first paragraph.

This is second paragraph.

This is third paragraph.

Example2

!DOCTYPE html**>**

**<html>**

**<head>**

**<title>**Second 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" language="javascript"**>**

 $(function() {

 $("p").css("color", "red");

 });

**</script>**

**</head>**

**<body>**

**<p>**The first paragraph is selected.**</p>**

**<p>**The second paragraph is selected.**</p>**

**<p>**The third paragraph is selected.**</p>**

**</body>**

**</html>**

The first paragraph is selected.

The second paragraph is selected.

The third paragraph is selected.

## **$(document).ready() and $()**

The code inserted between $(document).ready() is executed only once when page is ready for JavaScript code to execute.

In place of $(document).ready(), you can use shorthand notation $() only.

$(document).ready(function() {

$("p").css("color", "red");

});

$(function() {

$("p").css("color", "red");

});

# jQuery Selectors

jQuery Selectors are used to select and manipulate HTML elements. They are very important part of jQuery library.

With jQuery selectors, you can find or select HTML elements based on their id, classes, attributes, types and much more from a DOM.

In simple words, you can say that selectors are used to select one or more HTML elements using jQuery and once the element is selected then you can perform various operation on that.

All jQuery selectors start with a dollor sign and parenthesis e.g. $(). It is known as the factory function.

## **How to use Selectors**

The jQuery selectors can be used single or with the combination of other selectors. They are required at every steps while using jQuery. They are used to select the exact element that you want from your HTML document.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Selector** | **Description** |
| 1) | Name: | It selects all elements that match with the given element name. |
| 2) | #ID: | It selects a single element that matches with the given id. |
| 3) | .Class: | It selects all elements that matches with the given class. |
| 4) | Universal(\*) | It selects all elements available in a DOM. |
| 5) | Multiple Elements A,B,C | It selects the combined results of all the specified selectors A,B and C. |

## **Different jQuery Selectors**

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Description** |
| \* | $("\*") | It is used to select all elements. |
| #id | $("#firstname") | It will select the element with id="firstname" |
| .class | $(".primary") | It will select all elements with class="primary" |
| class,.class | $(".primary,.secondary") | It will select all elements with the class "primary" or "secondary" |
| element | $("p") | It will select all p elements. |
| el1,el2,el3 | $("h1,div,p") | It will select all h1, div, and p elements. |
| :first | $("p:first") | This will select the first p element |
| :last | $("p:last") | This will select he last p element |
| :even | $("tr:even") | This will select all even tr elements |
| :odd | $("tr:odd") | This will select all odd tr elements |
| :first-child | $("p:first-child") | It will select all p elements that are the first child of their parent |
| :first-of-type | $("p:first-of-type") | It will select all p elements that are the first p element of their parent |
| :last-child | $("p:last-child") | It will select all p elements that are the last child of their parent |
| :last-of-type | $("p:last-of-type") | It will select all p elements that are the last p element of their parent |
| :nth-child(n) | $("p:nth-child(2)") | This will select all p elements that are the 2nd child of their parent |
| :nth-last-child(n) | $("p:nth-last-child(2)") | This will select all p elements that are the 2nd child of their parent, counting from the last child |
| :nth-of-type(n) | $("p:nth-of-type(2)") | It will select all p elements that are the 2nd p element of their parent |
| :nth-last-of-type(n) | $("p:nth-last-of-type(2)") | This will select all p elements that are the 2nd p element of their parent, counting from the last child |
| :only-child | $("p:only-child") | It will select all p elements that are the only child of their parent |
| :only-of-type | $("p:only-of-type") | It will select all p elements that are the only child, of its type, of their parent |
| parent > child | $("div > p") | It will select all p elements that are a direct child of a div element |
| parent descendant | $("div p") | It will select all p elements that are descendants of a div element |
| element + next | $("div + p") | It selects the p element that are next to each div elements |
| element ~ siblings | $("div ~ p") | It selects all p elements that are siblings of a div element |
| :eq(index) | $("ul li:eq(3)") | It will select the fourth element in a list (index starts at 0) |
| :gt(no) | $("ul li:gt(3)") | Select the list elements with an index greater than 3 |
| :lt(no) | $("ul li:lt(3)") | Select the list elements with an index less than 3 |
| :not(selector) | $("input:not(:empty)") | Select all input elements that are not empty |
| :header | $(":header") | Select all header elements h1, h2 ... |
| :animated | $(":animated") | Select all animated elements |
| :focus | $(":focus") | Select the element that currently has focus |
| :contains(text) | $(":contains('Hello')") | Select all elements which contains the text "Hello" |
| :has(selector) | $("div:has(p)") | Select all div elements that have a p element |
| :empty | $(":empty") | Select all elements that are empty |
| :parent | $(":parent") | Select all elements that are a parent of another element |
| :hidden | $("p:hidden") | Select all hidden p elements |
| :visible | $("table:visible") | Select all visible tables |
| :root | $(":root") | It will select the document's root element |
| :lang(language) | $("p:lang(de)") | Select all p elements with a lang attribute value starting with "de" |
| [attribute] | $("[href]") | Select all elements with a href attribute |
| [attribute=value] | $("[href='default.htm']") | Select all elements with a href attribute value equal to "default.htm" |
| [attribute!=value] | $("[href!='default.htm']") | It will select all elements with a href attribute value not equal to "default.htm" |
| [attribute$=value] | $("[href$='.jpg']") | It will select all elements with a href attribute value ending with ".jpg" |
| [attribute|=value] | $("[title|='Tomorrow']") | Select all elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen |
| [attribute^=value] | $("[title^='Tom']") | Select all elements with a title attribute value starting with "Tom" |
| [attribute~=value] | $("[title~='hello']") | Select all elements with a title attribute value containing the specific word "hello" |
| [attribute\*=value] | $("[title\*='hello']") | Select all elements with a title attribute value containing the word "hello" |
| :input | $(":input") | It will select all input elements |
| :text | $(":text") | It will select all input elements with type="text" |
| :password | $(":password") | It will select all input elements with type="password" |
| :radio | $(":radio") | It will select all input elements with type="radio" |
| :checkbox | $(":checkbox") | Itwill select all input elements with type="checkbox" |
| :submit | $(":submit") | It will select all input elements with type="submit" |
| :reset | $(":reset") | It will select all input elements with type="reset" |
| :button | $(":button") | It will select all input elements with type="button" |
| :image | $(":image") | It will select all input elements with type="image" |
| :file | $(":file") | It will select all input elements with type="file" |
| :enabled | $(":enabled") | Select all enabled input elements |
| :disabled | $(":disabled") | It will select all disabled input elements |
| :selected | $(":selected") | It will select all selected input elements |
| :checked | $(":checked") | It will select all checked input elements |