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>
```

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()	It is used to call a function contains this value and a single array of arguments.
bind()	It is used to create a new function.
call()	It is used to call a function contains this value and an argument list.
toString()	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>
```

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>
```

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.

Creating Objects in JavaScript

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:

```
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>
```

2) By creating instance of Object

The syntax of creating object directly is given below:

```
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>
```

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);
```

10. **</script>**

JavaScript Array

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

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

---

1) JavaScript array literal

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

1. var arrayname=[value1,value2.....valueN];

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
1. <script>
2. var emp=["Sonoo","Vimal","Ratan"];
3. for (i=0;i<emp.length;i++){
4. document.write(emp[i] + "<br/>");
5. }
6. </script>
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

1. var arrayname=new Array();

Here, **new keyword** is used to create instance of array.

Let's see the example of creating array directly.

```
1. <script>
2. var i;
```

```
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

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

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

---

1) By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

```
var stringname="string value";
```

Let’s see the simple example of creating string literal.

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

```
var stringname=new String("string literal");
```

Here, **new keyword** is used to create instance of string.

Let’s see the example of creating string in JavaScript by new keyword.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

JavaScript String Methods

Let's see the list of JavaScript string methods with examples.

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.

match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.

1) JavaScript String charAt(index) Method

The JavaScript String charAt() method returns the character at the given index.

```
<script>
1. var str="javascript";
2. document.write(str.charAt(2));
3. </script>
```

2) JavaScript String concat(str) Method

The JavaScript String concat(str) method concatenates or joins two strings.

```
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

3) JavaScript String indexOf(str) Method

The JavaScript String indexOf(str) method returns the index position of the given string.

```
<script>
var s1="javascript from javaschool indexof";
var n=s1.indexOf("from");
document.write(n);
</script>
```

4) JavaScript String lastIndexOf(str) Method

The JavaScript String lastIndexOf(str) method returns the last index position of the given string.

```
<script>
var s1="javascript from javaschool indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

5) JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters.

```
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
```

6) JavaScript String toUpperCase() Method

The JavaScript String toUpperCase() method returns the given string in uppercase letters.

```
<script>
var s1="JavaScript toUpperCase Example";
var s2=s1.toUpperCase();
document.write(s2);
</script>
```

7) JavaScript String slice(beginIndex, endIndex) Method

The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

```
<script>
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write(s2);
</script>
```

8) JavaScript String trim() Method

The JavaScript String trim() method removes leading and trailing whitespaces from the string.

```
<script>
var s1="    javascript trim    ";
var s2=s1.trim();
document.write(s2);
</script>
```

JavaScript Date Object

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)

3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Methods

Let's see the list of JavaScript date methods with their description.

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of

	universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal

	time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

JavaScript Math

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

JavaScript Math Methods

Let's see the list of JavaScript Math methods with description.

Methods	Description

abs()	It returns the absolute value of the given number.
acos()	It returns the arccosine of the given number in radians.
asin()	It returns the arcsine of the given number in radians.
atan()	It returns the arc-tangent of the given number in radians.
cbrt()	It returns the cube root of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.
cos()	It returns the cosine of the given number.
cosh()	It returns the hyperbolic cosine of the given number.
exp()	It returns the exponential form of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
hypot()	It returns square root of sum of the squares of given numbers.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sinh()	It returns the hyperbolic sine of the given number.
sqrt()	It returns the square root of the given number

tan()	It returns the tangent of the given number.
tanh()	It returns the hyperbolic tangent of the given number.
trunc()	It returns an integer part of the given number.

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 code is included in HTML, 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

Let's discuss some examples over events and their handlers.

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function clickevent()
    {
       document.write("This is Javascript");
    }
    //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```