* **JavaScript**
* **It**  is a solution of client side dynamic pages.
* JavaScript is *an object-based scripting language* that is lightweight and cross-platform.
* JavaScript is not compiled but translated. The JavaScript Translator (embedded in browser) is responsible to translate the JavaScript code.

**Where JavaScript is used**

JavaScript is used to create interactive websites. It is mainly used for:

* Client-side validation
* Dynamic drop-down menus
* Displaying data and time
* Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
* Displaying clocks etc.

**Ckkjs001.html**

<h2>Welcome to JavaScript</h2>

<script>

document.write("God is great");

</script>

**Ckkjs002.html**

<h2>Welcome to JavaScript</h2>

<script type="text/javascript">

document.write("God is great");

</script>

Javascript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

The **document.write()** function is used to display dynamic content through JavaScript.

**Ckkjs003.html**

<script type="text/javascript">

  alert("God is great and it is true");

</script>

## JavaScript Example : code between the head tag

Let’s see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag. In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function\_name as given below. To call function, you need to work on event. Here we are using onclick event to call msg() function.

**Ckkjs004.html**

<html>

<head>

<script type="text/javascript">

function msg()

{

alert("Hello, God is great");

}

</script>

</head>

<body>

<p>Welcome to JavaScript</p>

<form>

<input type="button" value="click" onclick="msg()"/>

</form>

</body>

</html>

We can create external JavaScript file and embed it in many html page. It provides **code re usability** because single JavaScript file can be used in several html pages. An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let’s create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

**message.js**

function msg()

{

  alert("Hello God is great");

}

Let’s include the JavaScript file into html page. It calls the JavaScript function on button click.

**Ckkjs005.html**

<html>

<head>

<script type="text/javascript" src="message.js"></script>

</head>

<body>

<p>Welcome to JavaScript</p>

<form>

<input type="button" value="click" onclick="msg()"/>

</form>

</body>

</html>

# JavaScript Comment

* [JavaScript comments](https://www.javatpoint.com/javascript-comment)
* [Advantage of javaScript comments](https://www.javatpoint.com/javascript-comment)
* [Single-line and Multi-line comments](https://www.javatpoint.com/javascript-comment)

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code. The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

#### Advantages of JavaScript comments

There are mainly two advantages of JavaScript comments.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

## Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

## JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement. Let’s see the example of single-line comment i.e. added before the statement.

**<script>**

// It is single line comment

document.write("hello javascript");

**</script>**

Let’s see the example of single-line comment i.e. added after the statement.

**Ckkjs006.html**

<html>

<body>

<script>

var a=10;

var b=20;

var c=a+b;//It adds values of a and b variable

document.write("c = a+ b = "+c);//prints sum of 10 and 20

</script>

</body>

</html>

## JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient. It is represented by forward slash with asterisk then asterisk with forward slash. For example:

/\* your code here  \*/

It can be used before, after and middle of the statement.

**<script>**

/\* It is multi line comment.

It will not be displayed \*/

document.write("example of javascript multiline comment");

**</script>**

# Javascript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

var a=40;//holding number

var b="Rahul";//holding string

var x = 10+10; // it gives 20  
var y = "2" + 2; // it gives 22  
var z = "God" + 5; // it gives God5

## JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| String | represents sequence of characters e.g. "hello" |
| Number | represents numeric values e.g. 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents undefined value |
| Null | represents null i.e. no value at all |

## JavaScript non-primitive data types

The non-primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Object | represents instance through which we can access members |
| Array | represents group of similar values |
| RegExp | represents regular expression |

# JavaScript Variable

* [JavaScript Local variable](https://www.javatpoint.com/javascript-variable#local)
* [JavaScript Global variable](https://www.javatpoint.com/javascript-variable#gloabl)

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable. There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( $ ) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

## Correct JavaScript variables

var x = 10;

var \_value="sonoo";

## Incorrect JavaScript variables

var  123=30;

var \*aa=320;

## Example of JavaScript variable

Let’s see a simple example of JavaScript variable.

**<script>**

var x = 10;

var y = 20;

var z=x+y;

document.write(z);

**</script>**

## JavaScript local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

**<script>**

function abc()

{

var x=10;//local variable

}

**</script>**

Or,

**<script>**

If(10**<13**)

{

var y=20;//JavaScript local variable

}

**</script>**

## JavaScript global variable

A **JavaScript global variable** is declared outside the function or declared with window object. It can be accessed from any function. Let’s see the simple example of global variable in JavaScript. A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

**Ckkjs007.html**

<html>

<body>

<script>

var data=200;//gloabal variable

function a()

{

document.writeln("In a : "+data);

}

function b()

{

document.writeln("<br/>In b : "+data);

}

a();//calling JavaScript function

b();

</script>

</body>

</html>

**Declaring JavaScript global variable within function**

To declare JavaScript global variables inside function, you need to use **window object**. For example:

window.value=90;

Now it can be declared inside any function and can be accessed from any function. For example:

**Ckkjs008.html**

<html>

<body>

<script>

function m()

{

window.value=100;//declaring global variable by window object

}

function n()

{

alert("GlobAL variable declared in m, used in n = "+window.value); //accessing global variable from other function

}

m();

n();

</script>

</body>

</html>

## Internals of global variable in JavaScript

When you declare a variable outside the function, it is added in the window object internally. You can access it through window object also. For example:

**Ckkjs009.html**

<html>

<body>

<script>

var value=50;

function a()

{

alert(window.value);//accessing global variable

}

a();

</script>

</body>

</html>

# JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

## JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition | 10+20 = 30 |
| - | Subtraction | 20-10 = 10 |
| \* | Multiplication | 10\*20 = 200 |
| / | Division | 20/10 = 2 |
| % | Modulus (Remainder) | 20%10 = 0 |
| ++ | Increment | var a=10; a++; Now a = 11 |
| -- | Decrement | var a=10; a--; Now a = 9 |

## JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Is equal to | 10==20 = false |
| === | Identical (equal and of same type) | 10==20 = false |
| != | Not equal to | 10!=20 = true |
| !== | Not Identical | 20!==20 = false |
| > | Greater than | 20>10 = true |
| >= | Greater than or equal to | 20>=10 = true |
| < | Less than | 20<10 = false |
| <= | Less than or equal to | 20<=10 = false |

## JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | Bitwise AND | (10==20 & 20==33) = false |
| | | Bitwise OR | (10==20 | 20==33) = false |
| ^ | Bitwise XOR | (10==20 ^ 20==33) = false |
| ~ | Bitwise NOT | (~10) = -10 |
| << | Bitwise Left Shift | (10<<2) = 40 |
| >> | Bitwise Right Shift | (10>>2) = 2 |
| >>> | Bitwise Right Shift with Zero | (10>>>2) = 2 |

## JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Logical AND | (10==20 && 20==33) = false |
| || | Logical OR | (10==20 || 20==33) = false |
| ! | Logical Not | !(10==20) = true |

## JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assign | 10+10 = 20 |
| += | Add and assign | var a=10; a+=20; Now a = 30 |
| -= | Subtract and assign | var a=20; a-=10; Now a = 10 |
| \*= | Multiply and assign | var a=10; a\*=20; Now a = 200 |
| /= | Divide and assign | var a=10; a/=2; Now a = 5 |
| %= | Modulus and assign | var a=10; a%=2; Now a = 0 |

## JavaScript Special Operators

The following operators are known as JavaScript special operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| (?:) | Conditional Operator returns value based on the condition. It is like if-else. |
| , | Comma Operator allows multiple expressions to be evaluated as single statement. |
| Delete | Delete Operator deletes a property from the object. |
| In | In Operator checks if object has the given property |
| instanceof | checks if the object is an instance of given type |
| New | creates an instance (object) |
| typeof | checks the type of object. |
| Void | it discards the expression's return value. |
| Yield | checks what is returned in a generator by the generator's iterator. |

# JavaScript If-else

The **JavaScript if-else statement** is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

### JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

if(expression){

//content to be evaluated

}

Let’s see the simple example of if statement in javascript.

**Ckkjs010.html**

<html>

<body>

<script>

var a=20;

if(a>10)

{

document.write("value of a is greater than 10");

}

</script>

</body>

</html>

### JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

if(expression){

//content to be evaluated if condition is true

}

else{

//content to be evaluated if condition is false

}

Let’s see the example of if-else statement in JavaScript to find out the even or odd number.

### Ckkjs011.html

### <html>

### <body>

### <script>

### var a=20;

### if(a%2==0)

### {

### document.write("a is even number");

### }

### else

### {

### document.write("a is odd number");

### }

### </script>

### </body>

### </html>

### JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

if(expression1){

//content to be evaluated if expression1 is true

}

else if(expression2){

//content to be evaluated if expression2 is true

}

else if(expression3){

//content to be evaluated if expression3 is true

}

else{

//content to be evaluated if no expression is true

}

Let’s see the simple example of if else if statement in javascript.

### Ckkjs012.html

# <html>

# <body>

# <script>

# var a=20;

# if(a==10)

# {

# document.write("a is equal to 10");

# }

# else if(a==15)

# {

# document.write("a is equal to 15");

# }

# else if(a==20)

# {

# document.write("a is equal to 20");

# }

# else

# {

# document.write("a is not equal to 10, 15 or 20");

# }

# </script>

# </body>

# </html>

# JavaScript Switch

The **JavaScript switch statement** is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if..else..if because it can be used with numbers, characters etc. The signature of JavaScript switch statement is given below.

switch(expression)

{

case value1:

 code to be executed;

 break;

case value2:

 code to be executed;

 break;

......

default:

 code to be executed if above values are not matched;

}

Let’s see the simple example of switch statement in javascript.

### Ckkjs013.html

<html>

<body>

<script>

var grade='B';

var result;

switch(grade)

{

case 'A':

result="A Grade";

break;

case 'B':

result="B Grade";

break;

case 'C':

result="C Grade";

break;

default:

result="No Grade";

}

document.write(result);

</script>

</body>

</html>

The switch statement is fall-through i.e. all the cases will be evaluated if you don't use break statement

Let’s understand the behaviour of switch statement in JavaScript.

<script>

var grade='B';

var result;

switch(grade)

{

case 'A':

result="A Grade";

break;

case 'B':

result="B Grade";

break;

case 'C':

result="C Grade";

break;

default:

result="No Grade";

}

document.write(result);

</script>

#### Output of the above example

B Grade

# JavaScript Loops

The **JavaScript loops** are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array. There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

## JavaScript For loop

The **JavaScript for loop** iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

for (initialization; condition; increment)

{

    code to be executed

}

Let’s see the simple example of for loop in javascript.

### Ckkjs014.html

## <html>

## <body>

## <script>

## for (i=1; i<=5; i++)

## {

## document.write(i + "<br/>")

## }

## </script>

## </body>

## </html>

## JavaScript while loop

The **JavaScript while loop** iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

while (condition)

{

    code to be executed

}

Let’s see the simple example of while loop in javascript

**Ckkjs015.html**

<html>

<body>

<script>

var i=1;

while (i<=15)

{

document.write(i + "<br/>");

i++;

}

## </script>

## </body>

## </html>

## JavaScript do while loop

The **JavaScript do while loop** iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

do

{

    code to be executed

}while (condition);

Let’s see the simple example of do while loop in javascript.

**Ckkjs016.html**

## <html>

## <body>

## <script>

## var i=1;

## do

## {

## document.write(i + "<br/>");

## i++;

## }while (i<=10);

## </script>

## </body>

## </html>

# 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.

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

**Ckkjs017.html**

## <html>

## <body>

## <script>

## function msg()

## {

## alert("hello, God is great");

## }

## </script>

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

## </body>

## </html>

## JavaScript Function Arguments

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

**Ckkjs018.html**

<html>

<body>

<script>

function getcube(number)

{

alert(number\*number\*number);

}

</script>

<form>

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

</form>

</body>

</html>

**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.

**Ckkjs019.html**

<html>

<body>

<script>

function getInfo()

{

return "hello God is great, How r u?";

}

</script>

<script>

document.write(getInfo());

</script>

</body>

</html>