

Introduction to javascript

JavaScript *commonly known as JS*, is a dynamic programming language that's used for web development, in web applications, for game development, and lots more. It allows you to implement dynamic features on web pages that cannot be done with only HTML and CSS. Without JavaScript, all you would have on the web would be HTML and CSS. These alone limit you to a few web page implementations. 90% (if not more) of your webpages would be static, and you'd only have the dynamic changes like animations that CSS provides

Advantages and Disadvantages of javascript

The major advantage of using javascript is that you can use one language on the whole stack from the client side to the server side. Mobile apps are increasingly being created using js. Whereas major disadvantage (for a long time at least), is that js has been a turd of a language for professional and team development for a long time and the stink will take some time clearing. Left some of other are listed below.

Advantages:

- Speed. Client-side JavaScript is very fast because it can be run immediately within the client-side browser. Unless outside resources are required, JavaScript is unhindered by network calls to a backend server.
- Simplicity. JavaScript is relatively simple to learn and implement.
- opularity. JavaScript is used everywhere on the web.
- Interoperability. JavaScript plays nicely with other languages and can be used in a huge variety of applications.
- Server Load. Being client-side reduces the demand on the website server.
- Gives the ability to create rich interfaces.

Disadvantages:

- Client-Side Security. Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable Javascript.
- Browser Support. JavaScript is sometimes interpreted differently by different browsers. This makes it somewhat difficult to write cross-browser code.

Features of javascript

Nowadays many web-based giants are using the technology of Java Script like Facebook, YouTube, Twitter, Gmail and Google Maps, etc. The Java Script features Enterprise level pop-down menus, Opening, and Closure of new window, manipulation of HTML Layers, Interface Enhancement of HTML, Adding animation essentials within the page, adding dynamic appearance in documents of HTML and Fixing of problems related to Browsers . The followings are more features of javascript.

- Object-Centered Script Language
- Client edge Technology
- Validation of User's Input
- Else and If Statement
- Interpreter Centered
- Ability to perform In Built Function
- Light weight & dedicated
- Ability to handle events.

Program to Demonstrate javascript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JavaScript demonstration</title>
    <script>
      alert('Hello, World!');
    </script>
  </head>
  <body>
  </body>
</html>
```

output:

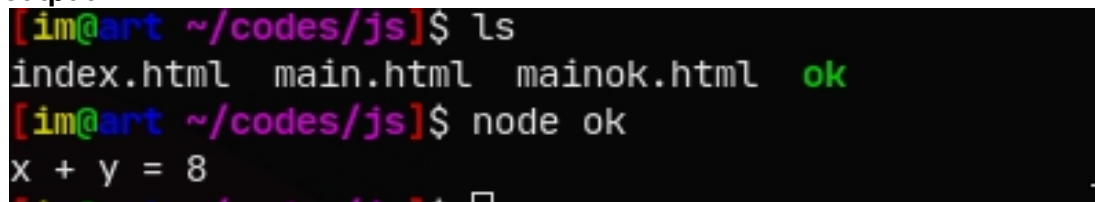


Operators

Likely to other programming languages, In JavaScript, an operator is a special symbol used to perform operations on operands (values and variables). In this below we will be demonstrating '+' with two variables x & y consisting values 4 & 2 respectively.

```
let x = 5;
let y = 3;
// addition
document.write('x + y = ', x + y);
```

output

A terminal window screenshot. The prompt is '[im@art ~/codes/js]'. The first command is 'ls', which outputs 'index.html main.html mainok.html ok'. The second command is 'node ok', which outputs 'x + y = 8'.

Functions in javascript

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs (same rules as variables). The parentheses may include parameter names separated by commas: (*parameter1*, *parameter2*, ...) The code to be executed, by the function, is placed inside curly brackets: {}

```

<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
document.getElementById("demo").innerHTML = myFunction(4, 3);
</script>
</body>
</html>

```

output:

```

[im@art ~]$ node main.js
12
[im@art ~]$ 

```

if else in javascript

```

<!DOCTYPE html>
<html>
  <body>
    <title>JS demonstration</title>
    <h2>JavaScript if else</h2>
    <script>
      // check if the number is positive or negative/zero

      const number = prompt("Enter a number: ");

      // check if number is greater than 0
      if (number > 0) {
        document.write("The number is positive");
      }
      // if number is not greater than 0
      else {
        document.write("The number is either a negative number or 0");
      }

      document.write("The if...else statement is easy");
    </script>
  </body>

```

output

```

[im@art ~/codes/js]$ node main.js
enter the number
7
The number is positive The if...else statement is easy
[im@art ~/codes/js]$ 

```

loops in javascript

There are many different kinds of loops, but they all essentially do the same thing: they repeat an action some number of times. The various loop mechanisms offer different ways to determine the start and end points of the loop. There are various situations that are more easily served by one type of loop over the others.

The following while loop iterates as long as n is less than 3:

code:

```
let n = 0;
let x = 0;
while (n < 3) {
  n++;
  x += n;
}
```

output:

```
[im@art ~/codes/js]$ ls
index.html  main.html  main.js  mainok.html  ok
[im@art ~/codes/js]$ node main.js
1
2
3
```

The following example iterates through the elements in an array until it finds the index of an element whose value is theValue:

code:

```
for (let i = 0; i < 5; i++) {
  text += "The number is " + i
  + "<br>";
}
```

output:

```
[im@art ~/codes/js]$ node main.js
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The following example shows the use cases of do while loop in javascript. The do...while statements combo defines a code block to be executed once, and repeated as long as a condition is true. The do...while is used when you want to run a code block at least one time

```
<script>
let text = "";
let i = 0;
do {text += i + "<br>";
  i++;}while (i < 5);
document.getElementById("demo").innerHTML = text;
</script>
```

output:

```
[im@art ~/codes/js]$ node main.js
1
2
3
4
5
```


switch cases in javascript

The switch expression is evaluated once. The value of the expression is compared with the values of each case. If there is a match, the associated block of code is executed. If there is no match, the default code block is executed.

In this example, The `getDay()` method returns the weekday as a number between 0 and 6 (Sunday=0, Monday=1, Tuesday=2 ..)

```
switch (new Date().getDay()) {  
  case 0:  
    day = "Sunday";  
    break;  
  case 1:  
    day = "Monday";  
    break;  
  case 2:  
    day = "Tuesday";  
    break;  
  case 3:  
    day = "Wednesday";  
    break;  
  case 4:  
    day = "Thursday";  
    break;  
  case 5:  
    day = "Friday";  
    break;  
  case 6:  
    day = "Saturday";  
}
```

Output:



```
[im@art ~/codes/js]$ ls  
day.js index.html main.html main.js mainok.html ok  
[im@art ~/codes/js]$ node day.js  
Today is monday  
[im@art ~/codes/js]$
```

javascript objects

In JavaScript, an object is an unordered collection of key-value pairs. Each key-value pair is called a property. The key of a property can be a string. And the value of a property can be any value, e.g., a string, a number, an array, and even a function. JavaScript provides you with many ways to create an object.

JavaScript provides you with many ways to create an object. The most commonly used one is to use the object literal notation.

The following example creates an empty object using the object literal notation:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

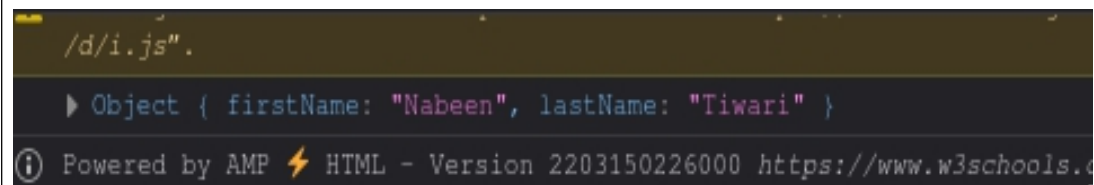
<script>
let person = {
    firstName: 'Nabin',
    lastName: 'Tiwari'
};

person.firstName = 'Nabeen';

    console.log(person);
</script>

</body>
```

output:



```
/d/i.js".
▶ Object { firstName: "Nabeen", lastName: "Tiwari" }
ⓘ Powered by AMP ⚡ HTML - Version 2203150226000 https://www.w3schools.com/
```

some predefined objects are listed below.

Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive. You can define a property by assigning it a value. For example, let's create an object named myCar and give it properties named make, model, and year as follows:

```
const myCar = new Object();
myCar.make = 'Ford';
myCar.model = 'Syangja';
myCar.year = 2004;
```

```
const myCar = {
    make: 'Ford',
    model: 'Syangja',
    year: 1969
};
```

Object.create method

```
const person = {
    isHuman: false,
    printIntroduction: function() {
        console.log('My name is ${this.name}. Am I human? ${this.isHuman}');
    }
};
const me = Object.create(person);
me.name = 'Nabin'; // "name" is a property set on "me", but not on "person"
me.isHuman = true; // inherited properties can be overwritten
me.printIntroduction();
```

Output:

```
[im@art ~/codes/js]$ ls
day.js  human.js  index.html  main.html  main.js  mainok.html  ok
[im@art ~/codes/js]$ node human.js
My name is Nabin. Am I human? true
[im@art ~/codes/js]$
```

Event handling in javascript

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events. we can manipulate this to work as our needings . The below examples consists of onmouseover and onclick event handling in js.

onclick example:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript addEventListener()</h2>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").addEventListener("click", click);

function click() {
    alert("clicked");
}
</script>

</body>
</html>
```

Output:

clicked

OK

In above example when we click the button appearing with try it . it displays the alert message as clicked.

onmouse event:

code:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  border: 1px solid black;
  margin: 10px;
  float: left;
  padding: 30px;
  text-align: center;
  background-color: lightgray;
}

p {
  background-color: white;
}
</style>
</head>
<body>

<h3>This example demonstrates the difference between onmousemove,
onmouseenter and onmouseover.</h3>

<div onmousemove="myMoveFunction()">
  <p>onmousemove: <br> <span id="demo">Mouse over me!</span></p>
</div>
<script>
var x = 0;
var y = 0;
var z = 0;

function myMoveFunction() {
  document.getElementById("demo").innerHTML = z+=1;
}
</script>

</body>
</html>
```

Output:



Form Validation in javascript

In this form validation example first its asks for the user to input her/his name and password and it redirects to my github profile.

```
<html>
<body>
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
    alert("Name can't be blank");
    return false;
}else if(password.length<6){
    alert("Password must be at least 6 characters long.");
    return false;
}
}
</script>
<body>
<form name="myform" method="post" action="https://github.com/iyamnabeen"
onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```

output:

