|  |
| --- |
| Difference between Server-side Scripting and Client-side Scripting |

The scripts may be created in **two** ways: on the **client side** or the **server side**, where the server-side scripts are processed on a server. In contrast, client-side scripting needs browsers to execute scripts on the client system but doesn't connect with the server executing the client-side scripts.

## What is Server-side Scripting?

**Server-side scripting** is a programming technique for creating code that may run software on the server side. In other words, server-side scripting is any scripting method that may operate on a web server

E.g.

* PHP
* .NET
* Python

## What is Client-side Scripting?

**Client-side scripting** generates code that may be executed on the client end without needing server-side processing. These scripts are typically embedded into HTML text.

* JavaScript
* VBScript

What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

Why Java Script

* **Event Bases Programming**

It’ s a event based programming Language.

* **Front-End Developers**

Front-End Developers create user interfaces and visual elements for web applications using JavaScript, CSS, and HTML.

* **Back-End Developers**

Back-End Developers use JavaScript frameworks like [Node.js](https://www.codecademy.com/learn/learn-node-js?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=should_i_learn_javascript_blog) and programming languages like [SQL](https://www.codecademy.com/catalog/language/sql?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=should_i_learn_javascript_blog) to manage an application’s servers and databases.

* **Full-Stack Developers**

[Full-Stack Developers](https://www.codecademy.com/resources/blog/what-does-a-full-stack-developer-do/?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=should_i_learn_javascript_blog) are proficient with both front-end and back-end development, and they can use any mix of the tools listed above.

Benefits Of JavaScript

* **Web Applications**  
  As mentioned earlier, JavaScript frameworks like React, Vue, Angular, and Node can be used to create full-fledged web applications.  
  In fact, popular applications like Microsoft Edge, Google Maps, and Uber are also built using JavaScript.
* **Web Development**  
  From the early days, JavaScript was introduced to create web pages. It provides simple ways to add dynamic behaviors and special effects to the webpage.  
  For instance, the ability to load the content of a webpage without the need to reload the entire page, form validations, change page behaviors during button clicks, and so on.  
  With the help of JavaScript, we can even enable the user to load content into a document without them having to reload the entire page as well. Validations are one of the other important features that JavaScript provides us.  
    
  Many of the biggest tech companies make use of JavaScript to enhance the user experience even today.
* **Web Servers**  
  With the introduction of Node.js, JavaScript developers now can create web servers from the ground up and take control of them.  
    
  If you don't know, servers are everywhere in today's web development world and allow you to transfer chunks of data without buffering.  
    
  JavaScript now being able to run outside the browser, via Node.js. It gives the developers the privilege to create web servers and take control of them. With Node.js, we can create servers from the ground up, and it's all JavaScript
* **Game Development**  
  Yes, JavaScript can even create Games, and it's this versatile nature of JavaScript that makes it so much popular.  
  JavaScript can be combined with HTML5, and they together play a very important role in game development using JavaScript. There are different JavaScript libraries present that can be used to develop Games in JavaScript. Some of the famous games built using JavaScript are OpenHog, CrossCode, Tower Building, etc.
* **Mobile Applications**  
  We can even use JavaScript to create mobile applications, and one of the most popular frameworks to do the same is React Native, which allows you to build cross-platform mobile applications.

# 

# Where To add JavaScript

## The <script> Tag

In HTML, JavaScript code is inserted between <script> and </script> tags.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript in Body</h2>

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

**<script>**

**document.getElementById("demo").innerHTML = "My First JavaScript";**

**</script>**

</body>

</html>

## JavaScript Functions and Events

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

## JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

<!DOCTYPE html>  
<html>  
<head>  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
</head>  
<body>

<h2>Demo JavaScript in Head</h2>  
  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

## JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

<!DOCTYPE html>  
<html>  
<body>  
  
<h2>Demo JavaScript in Body</h2>  
  
<p id="demo">A Paragraph</p>  
  
<button type="button" onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

# JavaScript Output

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().
* o access an HTML element, JavaScript can use the document.getElementById(id) method.
* The id attribute defines the HTML element. The innerHTML property defines the HTML content:

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My First Paragraph.</p>

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

<script>

document.getElementById("demo").innerHTML = 5 + 6;

</script>

</body>

</html>

## Using document.write()

For testing purposes, it is convenient to use document.write():

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<p>Never call document.write after the document has finished loading.

It will overwrite the whole document.</p>

<script>

document.write(5 + 6);

</script>

</body>

</html>

## Using window.alert()

You can use an alert box to display data.

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>

## Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

<!DOCTYPE html>  
<html>  
<body>  
  
<script>  
console.log(5 + 6);  
</script>  
  
</body>  
</html>

## JavaScript Print

JavaScript does not have any print object or print methods.

You cannot access output devices from JavaScript.

The only exception is that you can call the window.print() method in the browser to print the content of the current window.

# JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

// How to create variables:  
var x;  
let y;  
  
// How to use variables:  
x = 5;  
y = 6;  
let z = x + y;

JavaScript Values

The JavaScript syntax defines two types of values:

* Fixed values
* Variable values

Fixed values are called **Literals**.

Variable values are called **Variables**.

## JavaScript Variables

In a programming language, **variables** are used to **store** data values.

JavaScript uses the keywords var, let and const to **declare** variables.

An **equal sign** is used to **assign values** to variables.

## When to Use JavaScript var?

Always declare JavaScript variables with var,let, orconst.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browsers, you must use var.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>In this example, x is defined as a variable.

Then, x is assigned the value of 6:</p>

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

<script>

let x;

x = 6;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

## JavaScript Identifiers

All JavaScript **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

* Names can contain letters, digits, underscores, and dollar signs.
* Names must begin with a letter.
* Names can also begin with $ and \_ (but we will not use it in this).
* Names are case sensitive (y and Y are different variables).
* Reserved words (like JavaScript keywords) cannot be used as names.

## JavaScript Operators

JavaScript uses **arithmetic operators** ( + - \* / ) to **compute** values:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Operators</h2>

<p>JavaScript uses arithmetic operators to compute values (just like algebra).</p>

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

<script>

document.getElementById("demo").innerHTML = (5 + 6) \* 10;

</script>

</body>

</html>

## JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Expressions</h2>

<p>Expressions compute to values.</p>

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

<script>

document.getElementById("demo").innerHTML = 5 \* 10;

</script>

</body>

</html>

## JavaScript Identifiers / Names

Identifiers are JavaScript names.

Identifiers are used to name variables and keywords, and functions.

The rules for legal names are the same in most programming languages.

A JavaScript name must begin with:

* A letter (A-Z or a-z)
* A dollar sign ($)
* Or an underscore (\_)

Subsequent characters may be letters, digits, underscores, or dollar signs.

## JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables lastName and lastname, are two different variables:

## JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

**Hyphens:**

first-name, last-name, master-card, inter-city.

**Underscore:**

first\_name, last\_name, master\_card, inter\_city.

**Upper Camel Case (Pascal Case):**

FirstName, LastName, MasterCard, InterCity.

**Lower Camel Case:**

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

## JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

## JavaScript Dollar Sign $

Since JavaScript treats a dollar sign as a letter, identifiers containing $ are valid variable names: