**Introduction**

JavaScript is one of the 3 languages (HTML/CSS/JS) all web developers must learn. It is a high-level programming language that all modern web browsers support. It is also one of the core technologies of the web, along with HTML and CSS that you may have learned previously.

These days web pages are not the only place where JavaScript is used. Many desktop and server programs use JavaScript now. Node.js is the best known. Some databases, like MongoDB and CouchDB, also use JavaScript as their programming language.

### Syntax

JavaScript can be implemented using JavaScript statements that are placed within the <script>… </script> HTML tags in a web page. You can place the<script> tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the <head> tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

<script ...>

JavaScript code

</script>

The script tag takes two important attributes:

* **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript.
* **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to “text/javascript”.

So your JavaScript segment will look like:

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

JavaScript code

</script>

JavaScript is a **case-sensitive language**. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **variable\_name** and **Variable\_name**will convey different meanings in JavaScript.

**Placement**

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

* Script in <head>…</head> section of the HTML.
* Script in <body>…</body> section of the HTML.
* Script in <body>…</body> and <head>…</head> sections.
* Script in an external file and then include in <head>…</head> section.

Here is an example of how JavaScript code can be placed between the <head> and </head>

<html>

<head>

<script type = "text/javascript">

<!--

function sayHello() {

alert("Hello World")

}

//-->

</script>

</head>

<body>

<input type = "button" onclick = "sayHello()" value = "Say Hello" />

</body>

</html>

**Output**

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().

### Comments

Comments are used to explain code, and to make it more readable. There are 2 ways you can comment out in JavaScript:

**1. Single Line Comments:** Single line comments start with //. Any text between // and the end of the line will be ignored by JavaScript (will not be executed). This example below uses a single-line comment before each code line:

// Declare x, give it the value of 5

var x = 5;

var z = x \* 2; // Declare z, give it the value of x multiply by 2

**2. Multi-line Comments:** Multi-line comments start with /\* and end with \*/. Any text between /\* and \*/ will be ignored by JavaScript. This example below uses a multi-line comment (a comment block) to explain the code:

/\*

The code below will Declare x,

give it the value of 5

\*/

var x = 5;

### Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. Before you use a variable in a JavaScript program, you must declare it.

Variables are declared with the var keyword as follows:

<script type = "text/javascript">

<!--

var money;

var name;

//-->

</script>

You can also declare multiple variables with the same var keyword as follows:

<script type = "text/javascript">

<!--

var money, name;

//-->

</script>

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

<script type = "text/javascript">

<!--

var name = "Ali";

var money;

money = 2000.50;

//-->

</script>

**Note −** Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped language**. This means that a JavaScript variable **can hold a value of any data type**. Unlike many other languages, you don’t have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

### Data Types

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

* **Numbers**, eg. 123, 120.50 etc.
* **Strings** of text e.g. “This text string” etc.
* **Boolean** e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as object. We will cover objects in detail in a separate chapter.

**Note −** JavaScript does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values.

### Operators

Let us take a simple expression **4 + 5 is equal to 9.** Here**4 and 5** are called **operands** and **‘+’** is called the **operator**. JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

In this basic guide, we will only be covering the Arithmetic operators as that is what we will be testing you for.

##### Arithmetic Operators

Arithmetic operators perform arithmetic on numbers (literals or variables). They include:

* **Addition**: The addition operator (+) adds numbers
* **Subtraction**: The subtraction operator (-) subtracts numbers.
* **Multiplication**: The multiplication operator (\*) multiplies numbers.
* **Division**: The division operator (/) divides numbers.
* **Modulus** (Remainder) : The modulus operator (%) returns the division remainder.
* **Increment**: The increment operator (++) increments numbers.
* **Decrement**: The decrement operator (–) decrements numbers.
* **Exponentiation**: The exponentiation operator (\*\*) raises the first operand to the power of the second operand.

Some of the examples are shared as under:

// Addition

var x = 100 + 50;

// Subtraction

var z = x - y;

// Multiplication

var z = x \* y;

// Division

var z = x / y;

// Remainder

var z = x % y;

// Increment

var x = 5;

x++;

var z = x;

// Decrement

var x = 5;

x--;

var z = x;

// Exponentiation

var z = 2 \*\* 2; // result is 4

**Note:** Multiplication (\*) and division (/) have higher precedence than addition (+) and subtraction (-).

### Conditions

Very often when you write code, you want to perform different actions for different decisions. You can use **conditional statements** in your code to do this.

In JavaScript we have the following conditional statements:

* Use **if** to specify a block of code to be executed, if a specified condition is true
* Use **else** to specify a block of code to be executed, if the same condition is false
* Use **else if** to specify a new condition to test, if the first condition is false
* Use **switch** to specify many alternative blocks of code to be executed

##### The if Statement:

Use the **if statement** to specify a block of JavaScript code to be executed if a condition is true.

**Syntax:**

if (condition) {

// block of code to be executed if the condition is true

}

##### The else Statement:

Use the **else statement** to specify a block of code to be executed if the condition is false.

**Syntax:**

if (condition) {

// block of code to be executed if the condition is true

} else {

// block of code to be executed if the condition is false

}

##### The else if Statement:

Use the **else if statement** to specify a new condition if the first condition is false.

**Syntax:**

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if the condition1 is false and condition2 is true

} else {

// block of code to be executed if the condition1 is false and condition2 is false

}

##### Switch Statement:

The **switch statement** is used to perform different actions based on different conditions.

Syntax:

switch(expression) {

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

}

**This is how it works:**

1. The switch expression is evaluated once.  
2. The value of the expression is compared with the values of each case.  
3. If there is a match, the associated block of code is executed.

### Loops

Loops can execute a block of code a number of times. They are handy, if you want to run the same code over and over again, each time with a different value. JavaScript supports different kinds of loops:

* **for** – loops through a block of code a number of times
* **for/in** – loops through the properties of an object
* **for/of** – loops through the values of an iterable object
* **while** – loops through a block of code while a specified condition is true
* **do/while** – also loops through a block of code while a specified condition is true

In this basic guide, we will only be covering the for loop and while loop.

##### The For Loop:

The for loop has the following syntax:

In the following example, the code in the loop will run from 0 to 4 as per the conditions set.

for (i = 0; i < 5; i++) {

text += "The number is " + i + "

";

}

##### The While Loop:

The while loop loops through a block of code as long as a specified condition is true.

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

while (i < 10) {

text += "The number is " + i;

i++;

}

### Functions

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like alert() and write() in the earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once.

JavaScript allows us to write our own functions as well.

##### Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

The basic syntax is shown below:

<script type = "text/javascript">

<!-- function functionname(parameter-list) { statements } //-->

</script>

Let say we want to define a helloWorld function which alerts Hello World. Here is how we will do it:

<script type = "text/javascript">

function helloWorld() {

alert("Hello World"); // alert is a function which outputs hello world in browser as a popup.

}

</script>

##### Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

<html>

<head>

<script type = "text/javascript">

function sayHello() {

document.write ("Hello there!");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type = "button" onclick = "sayHello()" value = "Say Hello">

</form>

<p>Use different text in write method and then try...</p>

</body>

</html>

##### Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

We have modified our sayHello function here. Now it takes two parameters.

<script type = "text/javascript">

function sayHello(name, age) {

document.write (name + " is " + age + " years old.");

}

</script>

##### The return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

<script type = "text/javascript">

function concatenate(first, last) {

var full;

full = first + last;

return full;

}

function secondFunction() {

var result;

result = concatenate('Zara', 'Ali');

document.write (result );

}

</script>