JavaScript Output

## **JavaScript Display Possibilities**

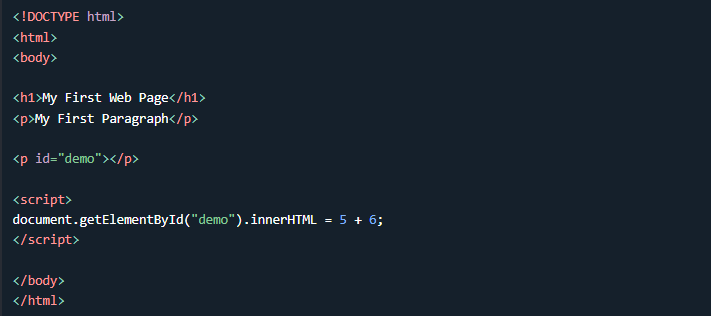
JavaScript can "display" data in different ways:

* 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().

## **Using innerHTML**

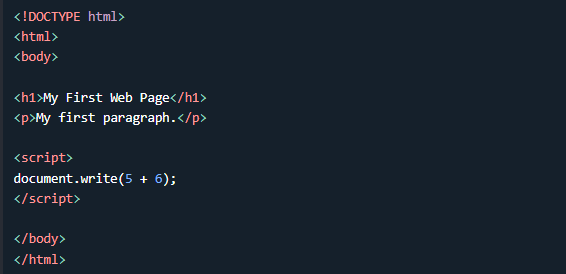
To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:



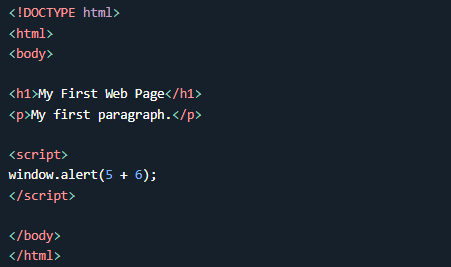
## **Using document.write()**

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



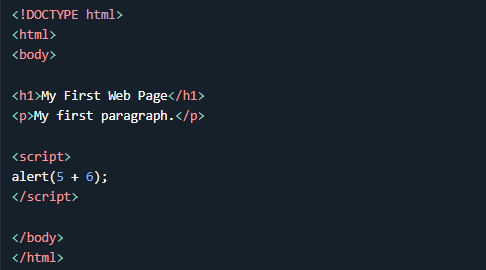
## **Using window.alert()**

You can use an alert box to display data:



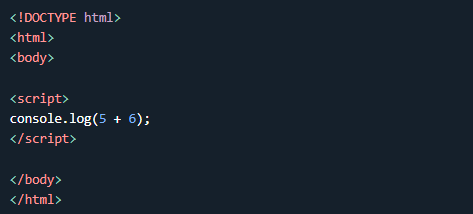
You can skip the window keyword.

In JavaScript, the window object is the global scope object. This means that variables, properties, and methods by default belong to the window object. This also means that specifying the window keyword is optional:



## **Using console.log()**

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

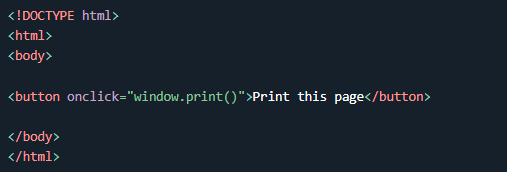


## **JavaScript Print**

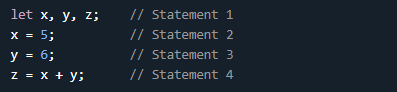
JavaScript does not have any print object or print methods.

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 Statements



## **JavaScript Programs**

A **computer program** is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called **statements**.

A **JavaScript program** is a list of programming **statements**.

## **JavaScript Statements**

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":



Most JavaScript programs contain many JavaScript statements.

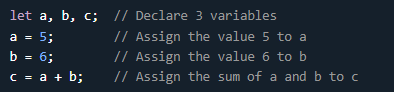
The statements are executed, one by one, in the same order as they are written.

*JavaScript programs (and JavaScript statements) are often called JavaScript code.*

## **Semicolons ;**

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:



When separated by semicolons, multiple statements on one line are allowed:



## **JavaScript White Space**

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:



A good practice is to put spaces around operators ( = + - \* / ):



## **JavaScript Line Length and Line Breaks**

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:



## **JavaScript Code Blocks**

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks is to define statements to be executed together.

One place you will find statements grouped together in blocks, is in JavaScript functions:



## **JavaScript Keywords**

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.

Our [Reserved Words Reference](https://www.w3schools.com/js/js_reserved.asp) lists all JavaScript keywords.

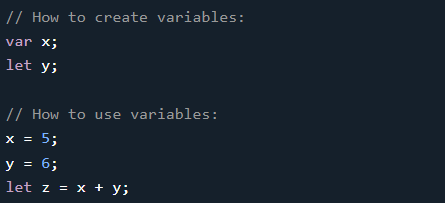
Here is a list of some of the keywords you will learn about in this tutorial:

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| var | Declares a variable |
| let | Declares a block variable |
| const | Declares a block constant |
| if | Marks a block of statements to be executed on a condition |
| switch | Marks a block of statements to be executed in different cases |
| for | Marks a block of statements to be executed in a loop |
| function | Declares a function |
| return | Exits a function |
| try | Implements error handling to a block of statements |

*JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.*

JavaScript Syntax

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



## **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 Literals**

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

**

2. **Strings** are text, written within double or single quotes:

**

# JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

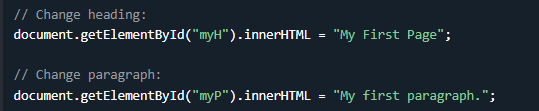
JavaScript comments can also be used to prevent execution, when testing alternative code.

## **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 uses a single-line comment before each code line:

**

This example uses a single line comment at the end of each line to explain the code:

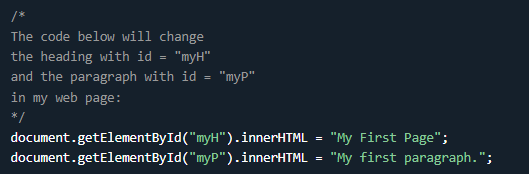
**

## **Multi-line Comments**

Multi-line comments start with /\* and end with \*/.

Any text between /\* and \*/ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

**

## **Using Comments to Prevent Execution**

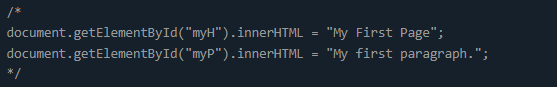
Using comments to prevent execution of code is suitable for code testing.

Adding // in front of a code line changes the code lines from an executable line to a comment.

This example uses // to prevent execution of one of the code lines:

**

This example uses a comment block to prevent execution of multiple lines:



# JavaScript Variables

### **4 Ways to Declare a JavaScript Variable:**

* Using var
* Using let
* Using const
* Using nothing

## **What are Variables?**

Variables are containers for storing data (storing data values).

In this example, x, y, and z, are variables, declared with the var keyword:



In this example, x, y, and z, are variables, declared with the let keyword:



In this example, x, y, and z, are undeclared variables:



From all the examples above, you can guess:

* x stores the value 5
* y stores the value 6
* z stores the value 11

## **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.

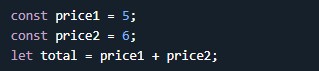
If you want your code to run in older browsers, you must use var.

## **When to Use JavaScript const?**

If you want a general rule: always declare variables with const.

If you think the value of the variable can change, use let.

In this example, price1, price2, and total, are variables:



The two variables price1 and price2 are declared with the const keyword.

These are constant values and cannot be changed.

The variable total is declared with the let keyword.

This is a value that can be changed.

## **Just Like Algebra**

Just like in algebra, variables hold values:



Just like in algebra, variables are used in expressions:



From the example above, you can guess that the total is calculated to be 11.

## **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 tutorial).
* Names are case sensitive (y and Y are different variables).
* Reserved words (like JavaScript keywords) cannot be used as names.

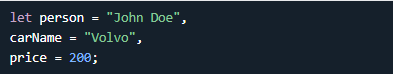
## **One Statement, Many Variables**

You can declare many variables in one statement.

Start the statement with let and separate the variables by **comma**:

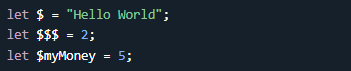


A declaration can span multiple lines:



## **JavaScript Dollar Sign $**

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

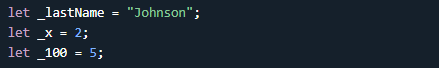


Using the dollar sign is not very common in JavaScript, but professional programmers often use it as an alias for the main function in a JavaScript library.

In the JavaScript library jQuery, for instance, the main function $ is used to select HTML elements. In jQuery $("p"); means "select all p elements".

## **JavaScript Underscore (\_)**

Since JavaScript treats underscore as a letter, identifiers containing \_ are valid variable names:



Using the underscore is not very common in JavaScript, but a convention among professional programmers is to use it as an alias for "private (hidden)" variables.