**Language**: JAVASCRIPT

**what is it about?** keywords & key information

**Focus**: Vocabulary & Translation

**expected outcome**: understand how to use this tool (check with the quiz)!

**Vocabulary search**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nouns** |  | **Verbs** |  |
| Affichage | display | Accélérer | To speed up |
| Appareils de sortie | Output devices | Améliorer | To improve |
| La balise | A tag | Appartenir | To belong to |
| Le caractère suivant | Subsequent character | Cacher | To hide |
| Le chargement de la page | Page loading | Convenir / être adapté | To be suitable |
| Contenu | content | Dépanner | debugging |
| Un chemin | a path | Ecraser / écrire dessus | To overwrite |
| Un chiffre | digit | Effacer | To erase |
| Une déclaration | A statement | Empêcher | To prevent |
| Un effet secondaire | Side effect | Eviter | To avoid |
| Un évènement | An event | Fonctionner | To work |
| Un fichier | A file | Fournir | To provide |
| Une fin / un objectif | A purpose | Imprimer | To print |
| Des guillemets | quotes | Mettre en œuvre / appliquer | To apply |
| Gestion d’un erreur | Error handling | Ralentir | To slow down |
| Le même ordre | The same order | Réagir aux maj et min | Case sensitive |
| Un mot clé | Keyword | Stocker | To store |
| Un moyen | A way | Survenir | To occur |
| Un navigateur | browser |  |  |
|  |  |  |  |
| A la place de | Instead of |  |  |
| Autrement | otherwise |  |  |
| ci-dessous | bellow |  |  |
| Pratique | convenient |  |  |
| Presque | almost |  |  |
| Sans | without |  |  |

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is getElementById().

The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

document.getElementById("demo").innerHTML = "Hello JavaScript";

JavaScript accepts both double and single quotes:

JavaScript changes the value of the src (source) attribute of an <img> tag.

Changing the style of an HTML element, is a variant of changing an HTML attribute.

Hiding HTML elements can be done by changing the display style.

Showing hidden HTML elements can also be done by changing the display style:

JavaScript Where To

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

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.

**In <head>,** a JavaScript function is placed in the <head> section of an HTML page. The function is invoked (called) when a button is clicked.

**In <body>,** a JavaScript function is placed in the <body> section of an HTML page. The function is invoked (called) when a button is clicked:

Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.

Scripts can also be placed in external files:

External scripts are convenient when the same code is used in many different web pages.

JavaScript files have the file extension .js.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

<script src="myScript.js"></script>

External scripts cannot contain <script> tags.

Placing scripts in external files has some advantages:

* It separates HTML and code
* It makes HTML and JavaScript easier to read and maintain
* Cached JavaScript files can speed up page loads

An external script can be referenced in 3 different ways:

* With a full URL (a full web address)
* With a file path (like /js/)
* Without any path

**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:

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

Using document.write() after an HTML document is loaded, will **delete all existing HTML**.

The document.write() method should only be used for testing.

**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, that 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.

<!DOCTYPE html>  
<html>  
<body>  
  
<button onclick="window.print()">Print this page</button>  
  
</body>  
</html>

**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**.

In HTML, JavaScript programs are executed by the web browser.

**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".

document.getElementById("demo").innerHTML = "Hello Dolly.";

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.

**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:

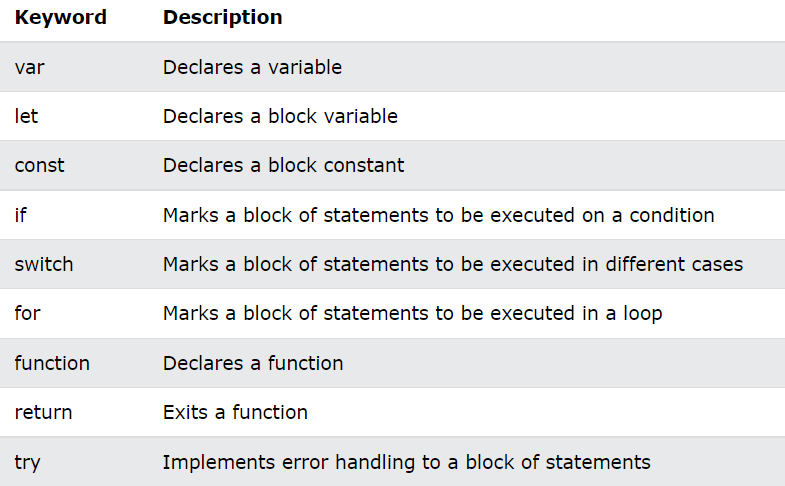
function myFunction() {  
  document.getElementById("demo1").innerHTML = "Hello Dolly!";  
  document.getElementById("demo2").innerHTML = "How are you?";  
}

**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:



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

The two most important syntax rules for fixed values are:

|  |  |
| --- | --- |
| 1. **Numbers** are written with or without decimals:  10.50 1001 | 2. **Strings** are text, written within double or single quotes:  "John Doe" 'John Doe' |

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.

JavaScript uses **arithmetic operators** ( + - \* / ) to **compute** values. (5 + 6) \* 10

JavaScript uses an **assignment operator** ( = ) to **assign** values to variables:

let x, y;  
x = 5;  
y = 6;

**JavaScript Comments**

Not all JavaScript statements are "executed".

Code after double slashes // or between /\* and \*/ is treated as a **comment**.

Comments are ignored, and will not be executed:

let x = 5;   // I will be executed  
// x = 6;   I will NOT be executed

**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.

**Note**

Numbers are not allowed as the first character in names.

This way JavaScript can easily distinguish identifiers from numbers.

**JavaScript is Case Sensitive**

All JavaScript identifiers are **case sensitive**.

The variables lastName and lastname, are two different variables:

**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 start with //.

Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

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

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

It is most common to use single line comments.  
Block comments are often used for formal documentation.

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:

//document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";

**JavaScript Variables**

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

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 browser, you must use var.

**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

**Note :** JavaScript identifiers are case-sensitive.

**JavaScript Best Practices**

Minimize the use of global variables.

This includes all data types, objects, and functions.

Global variables and functions can be overwritten by other scripts.

Use local variables instead, and learn how to use [closures](https://www.w3schools.com/js/js_function_closures.asp).

**Always Declare Local Variables**

All variables used in a function should be declared as **local** variables.

Local variables **must** be declared with the var keyword or the let keyword,or the const keyword, otherwise they will become global variables.

Strict mode does not allow undeclared variables.

**Declarations on Top**

It is a good coding practice to put all declarations at the top of each script or function.

This will:

* Give cleaner code
* Provide a single place to look for local variables
* Make it easier to avoid unwanted (implied) global variables
* Reduce the possibility of unwanted re-declarations

## Initialize Variables

It is a good coding practice to initialize variables when you declare them.

This will:

* Give cleaner code
* Provide a single place to initialize variables
* Avoid undefined values

## Don't Use new Object()

* Use "" instead of new String()
* Use 0 instead of new Number()
* Use false instead of new Boolean()
* Use {} instead of new Object()
* Use [] instead of new Array()
* Use /()/ instead of new RegExp()
* Use function (){} instead of new Function()

## Beware of Automatic Type Conversions

* JavaScript is loosely typed.
* A variable can contain all data types.
* A variable can change its data type:

## Avoid Number, String, and Boolean as Objects

Always treat numbers, strings, or booleans as primitive values. Not as objects.

Declaring these types as objects, slows down execution speed, and produces nasty side effects.

## Avoid Using eval()

The eval() function is used to run text as code. In almost all cases, it should not be necessary to use it.

Because it allows arbitrary code to be run, it also represents a security problem.

***Source : https://www.w3schools.com/js/js\_intro.asp***



Now this is your go! Test your knowledge with the quiz!