JavaScript puede cambiar el contenido HTML

Uno de los muchos métodos HTML de JavaScript es `getElementById()`.

El siguiente ejemplo "busca" un elemento HTML (con id="demo") y cambia el contenido del elemento (innerHTML) a "Hello JavaScript":

Ejemplo

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

JavaScript in <head>

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

The function is invoked (called) when a button is clicked:

Example

```
<!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.

The function is invoked (called) when a button is clicked:

Example

```
<!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>
```

External JavaScript

Scripts can also be placed in external files:

External file: myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical 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:

Example

```
<script src="myScript.js"></script>
```

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()`.

Example

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

<h1>My First Web Page</h1>
<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()`:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>
</html>
```

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
alert(5 + 6);
</script>

</body>
</html>
```

You will learn more about debugging in a later chapter.

Example

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

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 Variables

4 Ways to Declare a JavaScript Variable:

- Using `var`
- Using `let`
- Using `const`
- Using nothing

What are Variables?

Example

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

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.

Strings are written inside double or single quotes. Numbers are written without quotes.

If you put a number in quotes, it will be treated as a text string.

Example

```
const pi = 3.14;
let person = "John Doe";
let answer = 'Yes I am!';
```

Declaring a JavaScript Variable

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the `var` or the `let` keyword:

Signo de dólar JavaScript $

Dado que JavaScript trata un signo de dólar como una letra, los identificadores que contienen $ son nombres de variables válidos:

Ejemplo

```
let $ = "Hello World";
let $$$ = 2;
let $myMoney = 5;
```

Usar el signo de dólar no es muy común en JavaScript, pero los programadores profesionales a menudo lo usan como un alias para la función principal en una biblioteca de JavaScript.

En la biblioteca JavaScript jQuery, por ejemplo, la función principal `$` se usa para seleccionar elementos HTML. En jQuery `$("p");` significa "seleccionar todos los elementos p".

JavaScript Underscore (_)

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

Example

```
let _lastName = "Johnson";
let _x = 2;
let _100 = 5;
```

Usar el guión bajo no es muy común en JavaScript, pero una convención entre los programadores profesionales es usarlo como un alias para las variables "privadas (ocultas)".

Example

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

Example

```
// You can create a constant array:
const cars = ["Saab", "Volvo", "BMW"];

// You can change an element:
cars[0] = "Toyota";

// You can add an element:
cars.push("Audi");
```

You can change the properties of a constant object:

Example

```
// You can create a const object:
const car = {type:"Fiat", model:"500", color:"white"};

// You can change a property:
car.color = "red";

// You can add a property:
car.owner = "Johnson";
```

But you can NOT reassign the object:

Example

```
const car = {type:"Fiat", model:"500", color:"white"};

car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR
```

Redeclaring a JavaScript `var` variable is allowed anywhere in a program:

Example

```
var x = 2;      // Allowed
var x = 3;      // Allowed
x = 4;          // Allowed
```

Redeclaring an existing `var` or `let` variable to `const`, in the same scope, is not allowed:

Example

```
var x = 2;      // Allowed
const x = 2;    // Not allowed

{
let x = 2;      // Allowed
const x = 2;    // Not allowed
}

{
const x = 2;    // Allowed
const x = 2;    // Not allowed
}
```

Reassigning an existing `const` variable, in the same scope, is not allowed:

Example

```
const x = 2;      // Allowed
x = 2;            // Not allowed
var x = 2;        // Not allowed
let x = 2;        // Not allowed
const x = 2;      // Not allowed

{
  const x = 2;    // Allowed
  x = 2;          // Not allowed
  var x = 2;      // Not allowed
  let x = 2;      // Not allowed
  const x = 2;    // Not allowed
}
```

Redeclaring a variable with `const`, in another scope, or in another block, is allowed:

JavaScript:

```
let x = 16 + 4 + "Volvo";
```

Result:

```
20Volvo
```

Const Hoisting

Variables defined with `var` are **hoisted** to the top and can be initialized at any time.

Meaning: You can use the variable before it is declared:

Example

This is OK:

```
carName = "Volvo";
var carName;
```

What Can JavaScript Do?

JavaScript can change HTML attribute values.

In this case JavaScript changes the value of the src (source) attribute of an image.

Turn on the light Turn off the light

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

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attribute values.</p>

<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the
light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the
light</button>

</body>
</html>
```

Types of JavaScript Operators

There are different types of JavaScript operators:

- Aritmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Conditional Operators
- Type Operators

JavaScript Arithmetic Operators

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

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Assignment

```
let x = 10;
x += 5;
```

Try it Yourself »

Adding JavaScript Strings

The `+` operator can also be used to add (concatenate) strings.

Example

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

The result of text3 will be:

```
John Doe
```

Try it Yourself »

The `+=` assignment operator can also be used to add (concatenate) strings:

Example

```
let text1 = "What a very ";
text1 += "nice day";
```

The result of text1 will be:

```
What a very nice day
```

Try it Yourself »

Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

Example

```
let x = 5 + 5;
let y = "5" + 5;
let z = "Hello" + 5;
```

The result of *x*, *y*, and *z* will be:

```
10
55
Hello5
```

Try it Yourself »

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Shift Assignment Operators

Operator	Example	Same As
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y

Logical Assignment Operators

Operator	Example	Same As
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y

The += Operator

The `+=` assignment operator adds a value to a variable.

Addition Assignment

```
let x = 10;
x += 5;
```

Try it Yourself »

The -= Operator

The `-=` assignment operator subtracts a value from a variable.

Subtraction Assignment

El operador *=

El `*=` operador de asignación multiplica una variable.

Asignación de multiplicación

```
let x = 10;
x *= 5;
```

El operador /=

La `/=` asignación divide una variable.

Asignación de división

```
let x = 10;
x /= 5;
```

El operador %=

El `%=` operador de asignación asigna un resto a una variable.

Asignación restante

```
let x = 10;
x %= 5;
```

El operador <<=

El `<<=` operador de asignación desplaza a la izquierda una variable.

Asignación de desplazamiento a la izquierda

```
let x = -100;
x <<= 5;
```

El operador &=

El `&=` operador de asignación hace AND de una variable.

Asignación AND bit a bit

```
let x = 10;
x &= 5;
```

El operador !=

El `!=` operador de asignación OR de una variable.

Asignación OR bit a bit

```
let x = 10;
x != 5;
```

JavaScript Data Types

‹ Previous

JavaScript variables can hold different data types: numbers, strings, objects and more:

```
let length = 16;                               // Number
let lastName = "Johnson";                      // String
let x = {firstName:"John", lastName:"Doe"};    // Object
```

The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

```
let x = 16 + "Volvo";
```

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

```
let x = "16" + "Volvo";
```

When adding a number and a string, JavaScript will treat the number as a string.

Example

```
let x = 16 + "Volvo";
```

Try it Yourself »

Example

```
let x = "Volvo" + 16;
```

Try it Yourself »

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

JavaScript:

```
let x = 16 + 4 + "Volvo";
```

Result:

```
20Volvo
```

Try it Yourself »

JavaScript:

```
let x = "Volvo" + 16 + 4;
```

Result:

```
Volvo164
```

Try it Yourself »

In the first example, JavaScript treats 16 and 4 as numbers, until it reaches "Volvo".

In the second example, since the first operand is a string, all operands are treated as strings.

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

Try it Yourself »

JavaScript Strings

A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

Example

```
let carName1 = "Volvo XC60";   // Using double quotes
let carName2 = 'Volvo XC60';   // Using single quotes
```

Try it Yourself »

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

Try it Yourself »

JavaScript Booleans

Booleans can only have two values: `true` or `false`.

Example

```
let x = 5;
let y = 5;
let z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

Try it Yourself »

Booleans are often used in conditional testing.

You will learn more about conditional testing later in this tutorial.

JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called `cars`, containing three items (car names):

Example

```
const cars = ["Saab", "Volvo", "BMW"];
```

Try it Yourself »

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

You will learn more about **arrays** later in this tutorial.

JavaScript Objects

JavaScript objects are written with curly braces `{}`.

Object properties are written as name:value pairs, separated by commas.

Example

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Try it Yourself »

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

You will learn more about **objects** later in this tutorial.

The typeof Operator

You can use the JavaScript `typeof` operator to find the type of a JavaScript variable.

The `typeof` operator returns the type of a variable or an expression:

Example

```
typeof ""             // Returns "string"
typeof "John"         // Returns "string"
typeof "John Doe"     // Returns "string"
```

Try it Yourself »

Example

```
typeof 0              // Returns "number"
typeof 314            // Returns "number"
typeof 3.14           // Returns "number"
typeof (3)            // Returns "number"
typeof (3 + 4)        // Returns "number"
```

Try it Yourself »

Undefined

In JavaScript, a variable without a value, has the value `undefined`. The type is also `undefined`.

Example

```
let car;    // Value is undefined, type is undefined
```

Try it Yourself »

Any variable can be emptied, by setting the value to `undefined`. The type will also be `undefined`.

Example

```
car = undefined;    // Value is undefined, type is undefined
```

Try it Yourself »

Empty Values

An empty value has nothing to do with `undefined`.

An empty string has both a legal value and a type.

Example

```
let car = "";    // The value is "", the typeof is "string"
```

Try it Yourself »

The else Statement

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

```
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
}
```

Example

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The result of greeting will be:

```
Good day
```

Try it Yourself »

EJERCICIOS

Entrada de datos por teclado

1. Confeccionar un programa que permita cargar el nombre de un usuario y su mail por teclado. Mostrar posteriormente los datos en la página HTML.

Estructuras secuenciales de programación.

2. Realizar la carga del radio de una circunferencia, mostrar por pantalla el área de este.
3. Escribir un programa en el cual se ingresen las edades de 4 personas, calcular la suma del mayor y del menor junto al producto del tercero con el segundo.
4. Solicitar por teclado ingresar los lados de un cuadrado y mostrar por pantalla el perímetro.
5. Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente. Mostrar lo que debe abonar el comprador (Ingresar por teclado un precio sin decimales, es decir un entero.)

```
let usuario, email;
usuario = prompt('Nombre de usuario');
email = prompt('Email');

alert('El usuario ' + usuario + ' tiene el email ' + email)


document.write('el email ')
document.write(email)
document.write(' pertenece a ')
document.write(usuario)
```

```
let radio, area;
radio = prompt('Teclea el radio de la circunferencia ');
area = 3.14 * radio * radio;

/*console.log(El area de la circunferencia es );*/

document.write('El area de la circunferencia es ');
document.write(area);
```

```
let edad1, edad2, edad3, edad4, mayor, menor, suma, producto;
edad1 = prompt('Teclea la edad de la persona 1 ');
edad2 = prompt('Teclea la edad de la persona 2 ');
edad3 = prompt('Teclea la edad de la persona 3 ');
edad4 = prompt('Teclea la edad de la persona 4 ');

if (edad1 >= edad2) {mayor = edad1}
else
   { let mayor = edad2}

alert('mayor '+ mayor);

if (edad3 >= mayor) {
    let mayor = edad3}

if (edad4 >= mayor) {
    let mayor = edad4}

alert('El mayor es ' + mayor)
```

```
let edad1, edad2, edad3, edad4, mayor, menor, suma, producto;
edad1 = prompt('Teclea la edad de la persona 1 ');
edad2 = prompt('Teclea la edad de la persona 2 ');

//if (edad1 > edad2) {mayor = edad1}
//else {mayor = edad2}
//alert('1El mayor es ' + mayor)

if (parseInt(edad1) > parseInt(edad2)) {
    let mayor = parseInt(edad1);

  } else {
    let mayor = parseInt(edad2);

    alert('El mayor es ' + mayor);
  }

alert('Despues del if el mayor es ' + mayor);
```