

UD 2

Introducción al lenguaje JavaScript

DESARROLLO WEB EN ENTORNO CLIENTE

Técnico de Grado Superior Desarrollo de Aplicaciones Web

2024-25

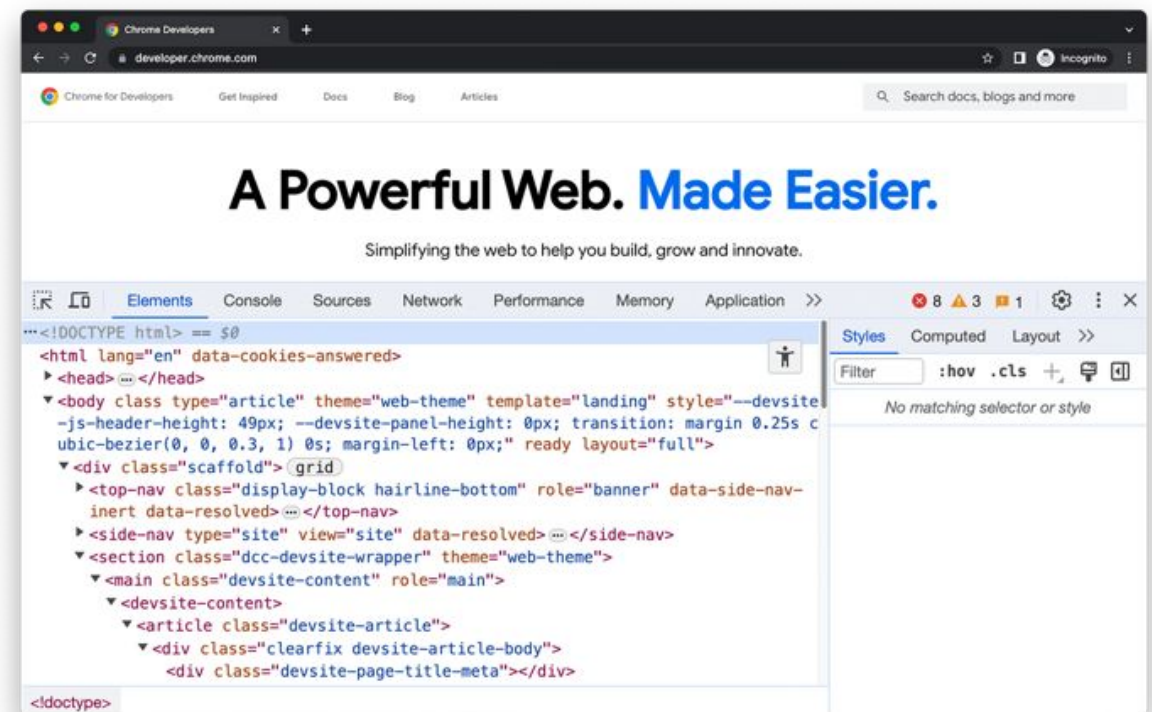
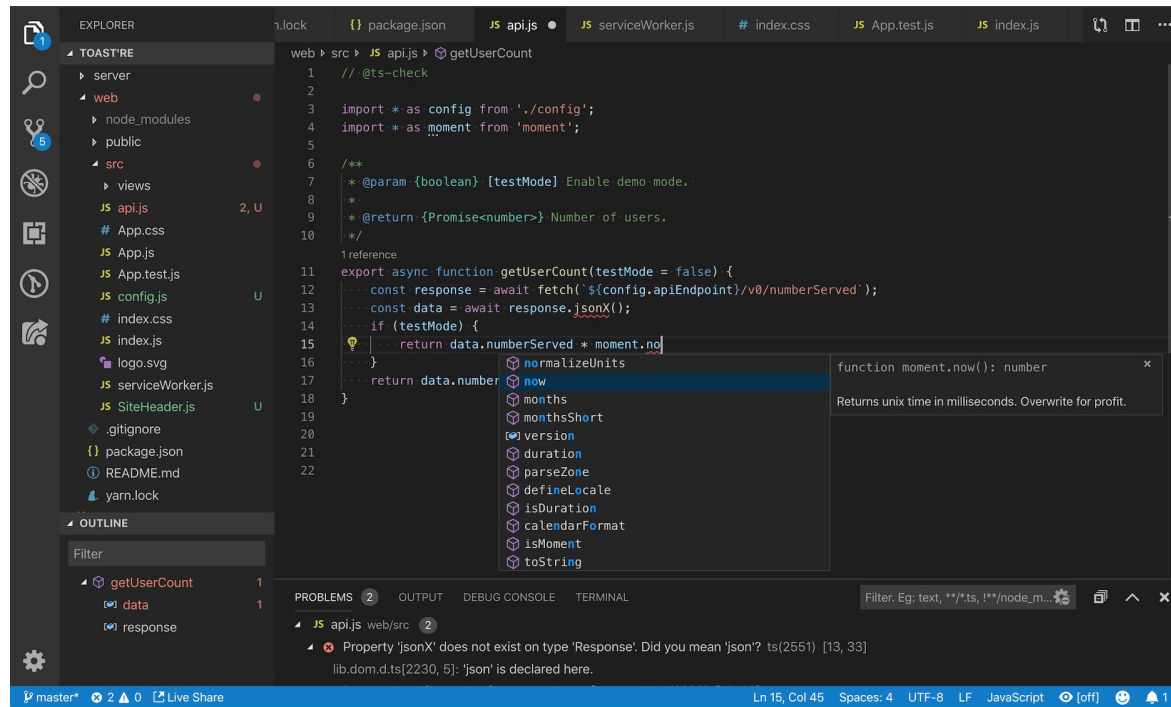


J. Mario Rodríguez
jrodper183e@g.educaand.es

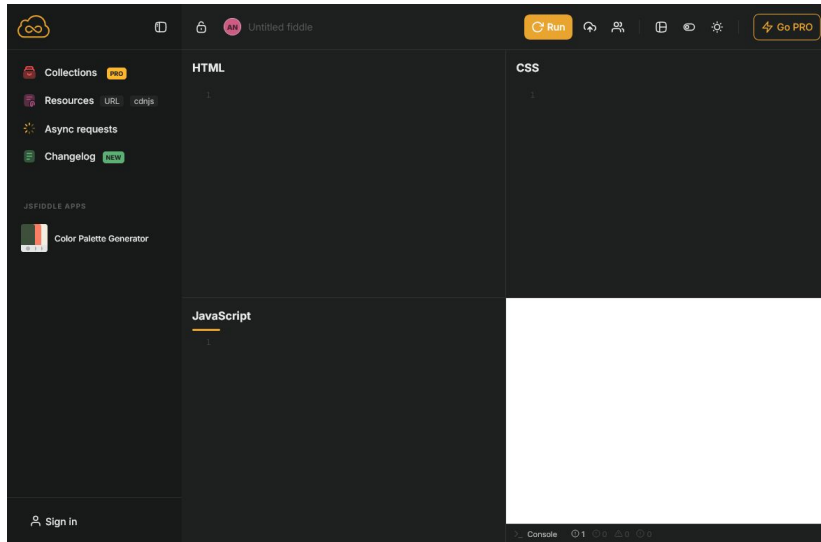
Contenidos

- Introducción a Javascript. Características.
- Variables. Ámbitos de utilización.
- Tipos de datos.
- Conversiones entre tipos de datos.
- Literales.
- Asignaciones.
- Operadores.
- Expresiones.
- Comentarios al código.
- Sentencias.
- Bloques de código.
- Decisiones.
- Bucles.
- Herramientas y entornos de desarrollo. Depuración.

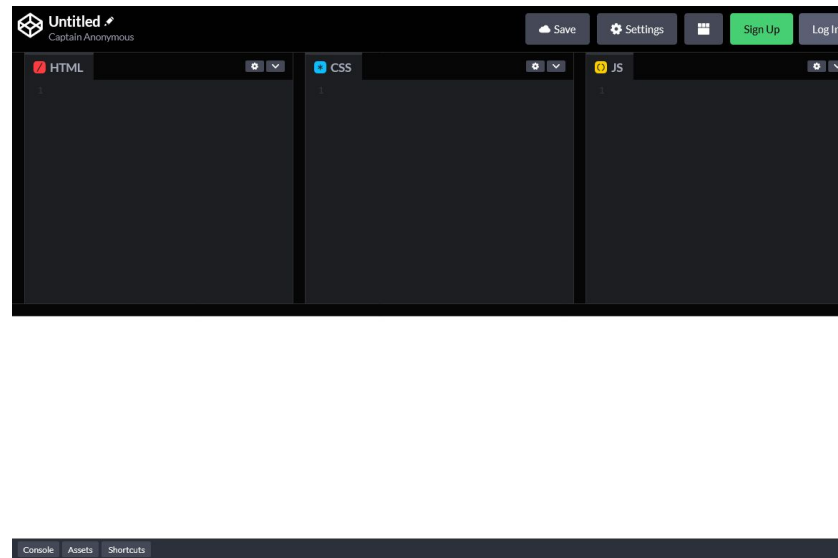
Entorno local



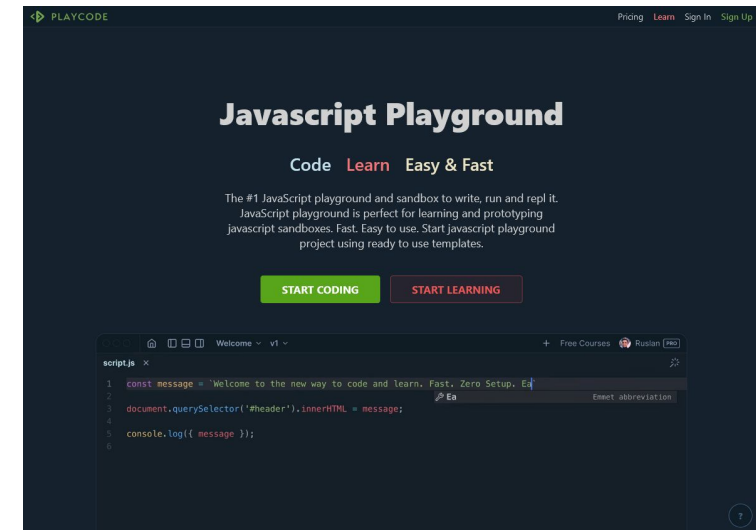
Entorno online



<https://jsfiddle.net/>



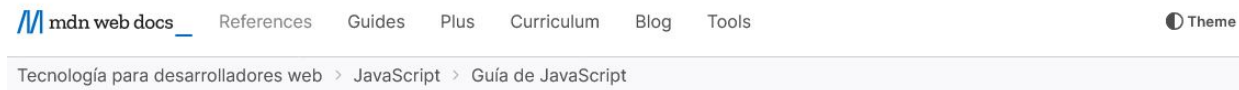
<https://codepen.io/pen/>



<https://playcode.io/>

Recursos

<https://developer.mozilla.org/es/docs/Web/JavaScript/Guide>



Filter

JavaScript

Tutorials

► Complete beginners

▼ JavaScript Guide

Introduction

Grammar and types

Control flow and error handling

Loops and iteration

Functions

Expressions and operators

Numbers and dates

Text formatting

Regular expressions

Indexed collections

Keyed collections

Working with objects

Using classes (inglés)

Using promises

Guía de JavaScript

La Guía de JavaScript te muestra cómo usar [JavaScript](#) y te brinda una perspectiva general del lenguaje. Si necesitas información exhaustiva sobre una característica del lenguaje, consulta la [Referencia de JavaScript](#).

Capítulos

Esta guía se divide en varios capítulos:

- [Introducción](#)
 - [Acerca de esta guía](#)
 - [Acerca de JavaScript](#)
 - [JavaScript y Java](#)
 - [ECMAScript](#)
 - [Herramientas](#)
 - [Hola, Mundo](#)
- [Gramática y tipos](#)
 - [Sintaxis básica y comentarios](#)
 - [Declaración de variables](#)
 - [Ámbito de variables](#)
 - [Elevación de variables \(hoisting\)](#)
 - [Estructuras y tipos de datos](#)

<https://uniwebsidad.com/libros/javascript>

uniwebsidad

INICIO LIBROS TUTORIALES EVENTOS FORO BUSCAR

Introducción a JavaScript

Escrito por Javier Eguiluz

Capítulo 1. Introducción

- 1.2. Breve historia
- 1.3. Especificaciones oficiales
- 1.4. Cómo incluir JavaScript en documentos XHTML
- 1.5. Etiqueta noscript
- 1.6. Glosario básico
- 1.7. Sintaxis
- 1.8. Posibilidades y limitaciones
- 1.9. JavaScript y navegadores
- 1.10. JavaScript en otros entornos

Capítulo 2. El primer script

Capítulo 3. Programación básica

- 3.1. Variables
- 3.2. Tipos de variables
- 3.3. Operadores
- 3.4. Estructuras de control de flujo
- 3.5. Funciones y propiedades básicas de JavaScript

Capítulo 4. Programación avanzada

- 4.1. Funciones
- 4.2. Ámbito de las variables
- 4.3. Sentencias break y continue
- 4.4. Otras estructuras de control

Capítulo 5. DOM

- 5.1. Árbol de nodos
- 5.2. Tipos de nodos
- 5.3. Acceso directo a los nodos
- 5.4. Creación y eliminación de nodos
- 5.5. Acceso directo a los atributos XHTML
- 5.6. Ejercicios sobre DOM

Capítulo 6. Eventos

- 6.1. Modelos de eventos
- 6.2. Modelo básico de eventos
- 6.3. Obteniendo información del evento (objeto event)

Capítulo 7. Formularios

- 7.1. Propiedades básicas de formularios y elementos
- 7.2. Utilidades básicas para formularios
- 7.3. Validación

Capítulo 8. Otras utilidades

- 8.1. Relojes, contadores e intervalos de tiempo
- 8.2. Calendario
- 8.3. Tooltip
- 8.4. Menú desplegable
- 8.5. Galerías de imágenes (Lightbox)

Capítulo 9. Detección y corrección de errores

- 9.1. Corrección de errores con Internet Explorer
- 9.2. Corrección de errores con Firefox
- 9.3. Corrección de errores con Opera

Capítulo 10. Recursos útiles

Capítulo 11. Ejercicios resueltos

- 11.2. Ejercicio 2
- 11.3. Ejercicio 3
- 11.4. Ejercicio 4
- 11.5. Ejercicio 5
- 11.6. Ejercicio 6
- 11.7. Ejercicio 7
- 11.8. Ejercicio 8
- 11.9. Ejercicio 9
- 11.10. Ejercicio 10
- 11.11. Ejercicio 11
- 11.12. Ejercicio 12
- 11.13. Ejercicio 13
- 11.14. Ejercicio 14
- 11.15. Ejercicio 15
- 11.16. Ejercicio 16
- 11.17. Ejercicio 17
- 11.18. Ejercicio 18
- 11.19. Ejercicio 19

Recursos

<https://www.w3schools.com/js/>

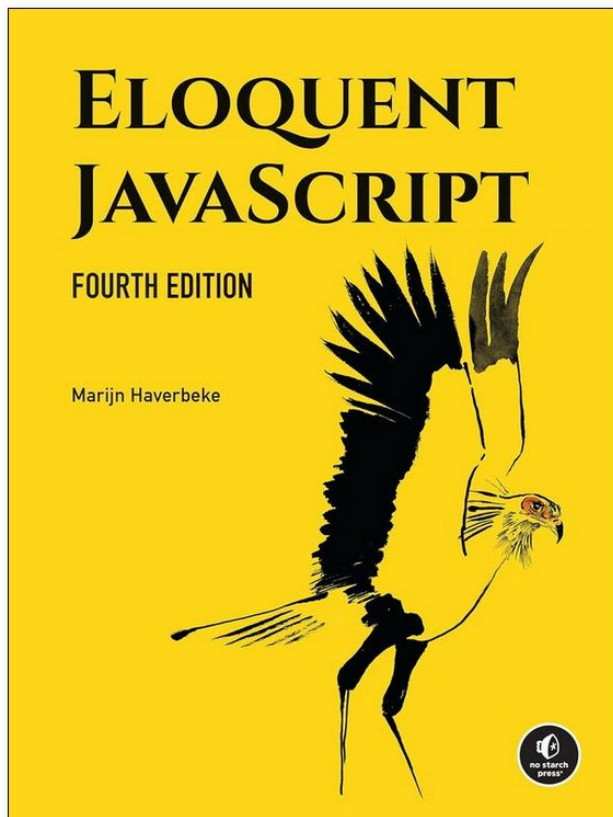
The screenshot shows the W3Schools website with the JavaScript Tutorial section highlighted. The left sidebar lists various topics under 'JS Tutorial', with 'JS HOME' selected. The main content area is titled 'JavaScript Tutorial' and includes a 'Home' button. Below this, there is a green box with text about JavaScript's popularity and a 'Start learning JavaScript now »' button. Further down, the 'Examples in Each Chapter' section is visible, featuring an 'Example' box with the title 'My First JavaScript' and a 'Try it Yourself »' button.

<https://javascript.info/js>

The screenshot shows the JavaScript.info website. The header includes the site name 'JAVASCRIPT.INFO' and a 'Buy EPUB/PDF' button. The main content area is titled 'The JavaScript language' and includes a brief introduction. Below this, there is a list of 14 chapters, with the first chapter 'An introduction' highlighted. The left sidebar lists various topics under 'Sibling chapters', including 'The JavaScript language', 'Browser: Document, Events, Interfaces', 'Frames and windows', 'Binary data, files', 'Network requests', 'Storing data in the browser', 'Animation', 'Web components', and 'Regular expressions'.

Recursos

<https://eloquentjavascript.net/>




ELOQUENT JAVASCRIPT
4TH EDITION (2024)


CONTENTS

- Introduction
- 1. Values, Types, and Operators
- 2. Program Structure
- 3. Functions
- 4. Data Structures: Objects and Arrays
- 5. Higher-order Functions
- 6. The Secret Life of Objects
- 7. Project: A Robot
- 8. Bugs and Errors
- 9. Regular Expressions
- 10. Modules
- 11. Asynchronous Programming
- 12. Project: A Programming Language
- 13. JavaScript and the Browser
- 14. The Document Object Model
- 15. Handling Events
- 16. Project: A Platform Game
- 17. Drawing on Canvas
- 18. HTTP and Forms
- 19. Project: A Pixel Art Editor
- 20. Node.js
- 21. Project: Skill-Sharing Website

<https://www.theodinproject.com/>

 THE ODIN PROJECT

[All Paths](#) [About](#) [Community](#) [Support us](#) [Sign in](#)



JavaScript Course

Overview

Make your websites dynamic and interactive with JavaScript! You'll create features and stand-alone applications. This module includes projects where you will learn how to manipulate the DOM, use object-oriented programming principles, and fetch real-world data using APIs.

Introduction

☐ How This Course Will Work

☐ A Quick Review

Organizing Your Javascript Code

☐ Organizing Your Javascript Code Introduction

☐ Objects and Object Constructors

☒ Project: Library

☐ Factory Functions and the Module Pattern

☒ Project: Tic Tac Toe

☐ Classes

☐ ES6 Modules

☐ npm

☐ Webpack

☒ Project: Restaurant Page

☐ Revisiting Webpack

☐ JSON

☐ OOP Principles

☒ Project: Todo List

JavaScript



[About Ecma](#) ▾ [Publications and s](#)

[Back to the list](#)

ECMA-262

ECMAScript® 2024 language specification

15th edition, June 2024

This Standard defines the ECMAScript 2024 general-purpose programming language.

Características

- Es un lenguaje **interpretado**, no compilado
- Se ejecuta en el **lado cliente**, en un navegador web
- Es un lenguaje que **acepta orientación a objetos**
podemos crear e instanciar objetos y usar objetos predefinidos del lenguaje pero... está basado en **prototipos**
por debajo un objeto es un prototipo; podemos crear objetos sin instanciarlos haciendo copias del prototipo
- Es un lenguaje **débilmente tipado** y con **tipificación dinámica**
no se indica el tipo de una variable al declararla e incluso puede cambiarse

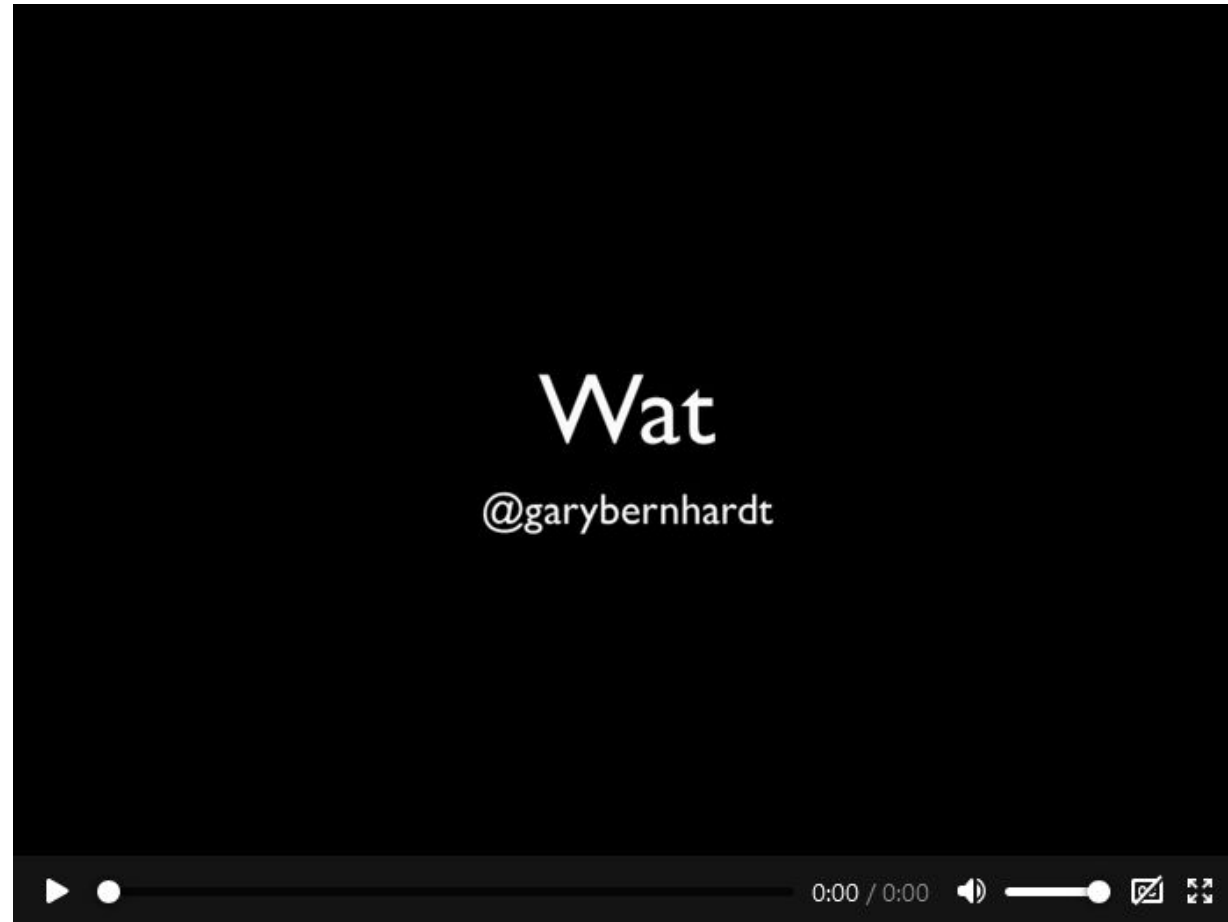
¿Siempre?

Características

Algunas cosas que JavaScript no puede hacer

- Acceder al sistema de ficheros del cliente
- Capturar datos de un servidor
 - puede solicitar datos; el servidor responderá (o no) a la petición
- Modificar las preferencias del navegador
- Enviar e-mails de forma invisible
- Crear ventanas sin que el usuario sea consciente
- ...

Características



<https://www.destroyallsoftware.com/talks/wat>

entrarán en una nueva dimensión de energía y creatividad.

Filter

instalación de software básico

¿Cuál será la apariencia de tu sitio Web?

Manejo de archivos

Conceptos básicos de HTML

Sin embargo, sentirse cómodo con JavaScript es un poco más difícil que sentirse cómodo con HTML y CSS. Deberás comenzar poco a poco y continuar trabajando en pasos pequeños y consistentes. Para comenzar, mostraremos cómo añadir JavaScript básico a tu página, creando un «¡Hola Mundo!» de ejemplo ([el estándar en los ejemplos básicos de programación](#)).

En este artículo

¿Qué es JavaScript realmente?

Ejemplo «¡Hola Mundo!»

Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria Almacenamiento

Filtrar salida Errores Advertencias Registros Información Depurar CSS XHR Peticiones

```
>> console.log("Curso DAW!");
```

Curso DAW! [debugger eval code:1:9](#)

< undefined

>>

Utiliza

console, alert, confirm, prompt

Hola, Mundo en el navegador

HTML

```
<script src="scripts/main.js"></script>
```

Recomendación: ponerla al final del `<body>` para que no se detenga el renderizado de la página mientras se descarga y se ejecuta el código.

También se puede utilizar en el `<head>` usando los atributos **async** o **defer**

JS


```
const miTitulo = document.querySelector("h1");  
miTitulo.textContent = "¡Hola mundo!";
```

Variables

var: la definición original

let: con alcance de bloque

const: no se puede reasignar



Recomendación:
usar let y const

<https://es.javascript.info/var>

```
if (true) {  
    var a = 5;  
    let b = 6;  
}  
  
console.log(a);  
console.log(b);
```

Variables

Declara tres variables: `nombre`, `edad`, y `ciudad`. Asigna tus propios valores y usa `let` o `const` de manera correcta. Muestra estos valores en la consola usando `console.log`.

```
let nombre = "María";  
const edad = 28;  
let ciudad = "Barcelona";  
console.log(nombre, edad, ciudad);
```


Variables

Ámbito de una variable

- Global (fuera de una función)
- Local (en una función)

```
console.log(miVariable); // undefined
var miVariable = "Hola, mundo!";
console.log(miVariable); // "Hola, mundo!"
```

Ocurre también con
let y const?

```
var miVariable; // La declaración es "elevada" al principio
console.log(miVariable); // undefined, porque no ha sido inicializada aún
miVariable = "Hola, mundo!"; // Ahora se inicializa
console.log(miVariable); // "Hola, mundo!"
```

Variable hoisting:

Las variables pueden hacer referencia a una variable declarada más tarde.
Las variables son “elevadas” a la parte superior de la función su ámbito.

Y con function?

```
saludar(); // "¡Hola!"
function saludar() {
  console.log("¡Hola!");
}
```

Las variables que no se han inicializado todavía devolverán el valor `undefined`.

Tipos de datos

Primitivos

number, string, boolean, undefined, null, symbol, bigint

Referencia

objetos, arrays, funciones

Recomendación:
mantener consistencia en
los tipos

```
let numero = 42;  
let texto = "Hola";  
let esMayor = true;  
let persona = { nombre: "Ana", edad: 25 };
```

Tipos de datos

typeof

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof>

```
console.log(typeof 42);  
// Expected output: "number"
```

```
console.log(typeof 'blubber');  
// Expected output: "string"
```

```
console.log(typeof true);  
// Expected output: "boolean"
```

```
console.log(typeof undeclaredVariable);  
// Expected output: "undefined"
```

Conversiones entre tipos de datos

automáticas (coerción)

explícitas (manual)

```
let num = "5";  
let suma = num + 10; // Coerción implícita, suma se convierte en "510"  
let numeroReal = Number(num); // Conversión explícita, devuelve 5
```

Conversiones entre tipos de datos

Dado un número almacenado en una variable como cadena ("10"), conviértelo en un número, súmale 5 y muestra el resultado. Luego convierte el resultado nuevamente a cadena.

```
let cadena = "10";  
let numero = Number(cadena);  
numero += 5;  
let resultadoFinal = String(numero);  
console.log(resultadoFinal); // "15"
```

Literales

son los valores que se les puede asignar a una variable

Arrays

Booleanos

Números coma flotante

Enteros

Objetos

Expresiones Regulares

Cadenas

Asignaciones

El operador de asignación (=) se utiliza para asignar un valor a una variable.

La operación de asignación evalúa el valor asignado.

Es posible encadenar el operador de asignación para asignar un solo valor a múltiples variables

```
let x = 2;  
const y = 3;  
  
console.log(x);  
// Expected output: 2  
  
console.log((x = y + 1)); // 3 + 1  
// Expected output: 4  
  
console.log((x = x * y)); // 4 * 3  
// Expected output: 12
```


Operadores

- Operadores de Asignación
- Operadores de Comparación
- Operadores Aritméticos
- Operadores sobre Bits
- Operadores Lógicos
- Operadores con cadenas de texto
- Operador ternario
- Operador coma
- Operador delete
- Operador typeof
- Operador void
- Operador in
- Operador instanceof

Operadores

<code>x += y</code>	<code>x = x + y</code>	Suma al valor actual de x el valor de y para asignarlo a x.
<code>x -= y</code>	<code>x = x - y</code>	Subtrae al valor actual de x el valor de y para asignarlo a x.
<code>x *= y</code>	<code>x = x * y</code>	Multiplica al valor actual de x el valor de y para asignarlo a x.
<code>x /= y</code>	<code>x = x / y</code>	Divide al valor actual de x el valor de y para asignarlo a x.
<code>x %= y</code>	<code>x = x % y</code>	Calcula el resto de x dividido entre y para asignarlo a x.
<code>x **= y</code>	<code>x = x ** y</code>	Calcula el exponente de x elevado a y para asignarlo a x.
<code>x <<= y</code>	<code>x = x << y</code>	Realiza un desplazamiento de bits a izquierda sobre x de y para asignarlo a x.
<code>x >>= y</code>	<code>x = x >> y</code>	Realiza un desplazamiento de bits a derecha sobre x de y para asignarlo a x.
<code>x >>>= y</code>	<code>x = x >>> y</code>	Realiza un desplazamiento de bits a derecha rellenando con ceros sobre x de y para asignarlo a x.
<code>x &= y</code>	<code>x = x & y</code>	Realiza un AND binario de x con y para asignarlo a x.
<code>x ^= y</code>	<code>x = x ^ y</code>	Realiza un XOR binario de x con y para asignarlo a x.
<code>x = y</code>	<code>x = x y</code>	Realiza un OR binario de x con y para asignarlo a x.
<code>x &&= y</code>	<code>x && (x = y)</code>	Calcula un AND lógico de x e y para asignarlo a x.
<code>x = y</code>	<code>x (x = y)</code>	Calcula un OR lógico de x e y para asignarlo a x.
<code>x ??= y</code>	<code>x ?? (x = y)</code>	Realiza una anulación lógica de x e y para asignarlo a x.

Expresiones

combinación de valores, variables y operadores que devuelven un valor

Comentarios

Recomendación:
comentarios para explicar "por
qué", no "qué". El código bien
escrito ya debería ser claro en
cuanto al "qué"

Comentario de línea: `// comentario`
Comentario de bloque: `/* comentario */`

Sentencias

cualquier instrucción ejecutable (declaración, asignación, llamada a función, etc.)

Bloques de código

Agrupación de sentencias entre {}

```
{  
  let y = 10;  
  console.log(y);  
}
```

Decisiones

https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/conditionals

```
if (condición) {  
    // Código a ejecutar si la condición es verdadera  
} else {  
    // Código a ejecutar si la condición es falsa  
}
```

¿Existe switch?
¿Cuál es su sintaxis?

```
let resultado = (condición) ? valorSiVerdadero : valorSiFalso;
```

Decisiones

Reto 1 decisiones:

Escribe una función que reciba la edad de una persona y determine si puede votar o no (mayor de 18 años). Si tiene 18 o más, debe imprimir "Puedes votar", de lo contrario "No puedes votar". Agrega también un caso especial para edades negativas que indique "Edad no válida".

Reto 2 decisiones:

Escribe un programa que reciba un número del 1 al 7 y devuelva el día de la semana correspondiente. Por ejemplo, si el usuario introduce 1, debe devolver "Lunes".

Bucles

https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/Looping_code

```
for (let i = 0; i < 10; i++) {  
    console.log(i);  
}
```

for ...in
itera sobre las propiedades enumerables de un objeto

```
for (let propiedad in objeto) {  
    // código a ejecutar en cada iteración  
}
```

```
let colores = ["Rojo", "Verde", "Azul"];  
  
for (let indice in colores) {  
    console.log(indice + ": " + colores[indice]);  
}
```

```
let i = 0;  
while (i < 10) {  
    console.log(i);  
    i++;  
}
```

```
let i = 0;  
do {  
    console.log(i);  
    i++;  
} while (i < 10);
```


Bucles

Reto 1 bucles:

Escribe un programa que imprima en la consola los números del 1 al 50 utilizando un bucle for.

Reto 2 bucles:

Crea un programa que reciba un número y determine si es primo o no. Para esto, deberás usar un bucle for para verificar si el número solo es divisible por 1 y por sí mismo.

Reto 3 bucles:

Escribe un programa que imprima una tabla de multiplicar (del 1 al 10) de un número introducido por el usuario. Usa un bucle while.

Funciones

```
function nombreDeLaFuncion(parámetros) {  
  // Cuerpo de la función  
  // Código que se ejecuta cuando la función es invocada  
}
```

```
nombreDeLaFuncion(argumentos);
```

¿Existen también métodos?

```
let nombre = "Carlos"; // Variable de ámbito global  
  
function saludar() {  
  let saludo = "Hola"; // Variable de ámbito local  
  console.log(saludo + ", " + nombre);  
}  
  
saludar(); // Hola, Carlos  
console.log(saludo); // Error: saludo no está definido
```

Funciones

Filter

▶ JavaScript first steps

▼ JavaScript building blocks

JavaScript building blocks

Making decisions in your code — conditionals

Looping code

Functions — reusable blocks of code

Build your own function

Function return values

Introduction to events

Event bubbling

Image gallery

▶ Introducing JavaScript objects

▶ Asynchronous JavaScript

▶ Client-side web APIs

Web forms — Working with user data

▶ Web form building blocks

Functions — reusable blocks of code

[Previous](#)

[Overview: JavaScript building blocks](#)

[Next](#)

Another essential concept in coding is **functions**, which allow you to store a piece of code that does a single task inside a defined block, and then call that code whenever you need it using a single short command — rather than having to type out the same code multiple times. In this article we'll explore fundamental concepts behind functions such as basic syntax, how to invoke and define them, scope, and parameters.

Prerequisites:

A basic understanding of HTML, CSS, and [JavaScript first steps](#).

Objective:

To understand the fundamental concepts behind JavaScript functions.

Where do I find functions?

In JavaScript, you'll find functions everywhere. In fact, we've been using functions all the way through the course so far; we've just not been talking about them very much. Now is the time, however, for us to start talking about functions explicitly, and really exploring their syntax.

Pretty much anytime you make use of a JavaScript structure that features a pair of parentheses — `()` — and you're **not** using a common built-in language structure like a [for loop](#), [while or do...while loop](#), or [if...else statement](#), you are making use of a function.

Cadenas y funciones con cadenas

Filter

Standard built-in objects

String

▼ Constructor

[String\(\)](#) constructor

▼ Static methods

[String.fromCharCode\(\)](#)

[String.fromCodePoint\(\)](#)

[String.raw\(\)](#)

▼ Instance methods

[String.prototype.anchor\(\)](#)

[String.prototype.at\(\)](#)

[String.prototype.big\(\)](#)

[String.prototype.blink\(\)](#)

[String.prototype.bold\(\)](#)

[String.prototype.charAt\(\)](#)

[String.prototype.charCodeAt\(\)](#)

String

✓ Baseline Widely available



The `String` object is used to represent and manipulate a sequence of characters.

Description

Strings are useful for holding data that can be represented in text form. Some of the most-used operations on strings are to check their [length](#), to build and concatenate them using the [+ and](#) [string operators](#), checking for the existence or location of substrings with the [indexOf\(\)](#) method, extracting substrings with the [substring\(\)](#) method.

Creating strings

Strings can be created as primitives, from string literals, or as objects, using the [String\(\)](#) constructor:

```
JS
const string1 = "A string primitive";
const string2 = 'Also a string primitive';
const string3 = `Yet another string primitive`;
```

Useful string methods

[Previous](#)

[Overview: JavaScript first steps](#)

[Next](#)

Now that we've looked at the very basics of strings, let's move up a gear and start thinking about what useful operations we can do on strings with built-in methods, such as finding the length of a text string, joining and splitting strings, substituting one character in a string for another, and more.

Prerequisites:	A basic understanding of HTML and CSS, an understanding of what JavaScript is.
Objective:	To understand that strings are objects, and learn how to use some of the basic methods available on those objects to manipulate strings.

Strings as objects

Most values can be used as if they are objects in JavaScript. When you create a string, for example by using

```
JS
const string = "This is my string";
```

Arrays

mdn web docs

References

Guides

Plus

Curriculum

Blog

Tools

Theme

Guides > JavaScript — Dynamic client-side scripting > JavaScript first steps > Arrays

Filter

JavaScript

Storing the information you need — Variables

Basic math in JavaScript — numbers and operators

Handling text — strings in JavaScript

Useful string methods

Arrays

Silly story generator

▶ JavaScript building blocks

▶ Introducing JavaScript objects

▶ Asynchronous JavaScript

▶ Client-side web APIs

Web forms — Working with user data

▶ Web form building blocks

▶ Advanced web form techniques

Accessibility — Make the web usable by everyone

▶ Accessibility guides

Performance — Making websites

Arrays

[Previous](#)[Overview: JavaScript first steps](#)[Next](#)

In the final article of this module, we'll look at arrays — a neat way of storing a list of data items under a single variable name. Here we look at why this is useful, then explore how to create an array, retrieve, add, and remove items stored in an array, and more besides.

Prerequisites:	A basic understanding of HTML and CSS, an understanding of what JavaScript is.
Objective:	To understand what arrays are and how to manipulate them in JavaScript.

What is an array?

Arrays are generally described as "list-like objects"; they are basically single objects that contain multiple values stored in a list. Array objects can be stored in variables and dealt with in much the same way as any other type of value, the difference being that we can access each value inside the list individually, and do super useful and efficient things with the list, like loop through it and do the same thing to every value. Maybe we've got a series of product items and their prices stored in an array, and we want to loop through them all and print them out on an invoice, while totaling all the prices together and printing out the total price at the bottom.

If we didn't have arrays, we'd have to store every item in a separate variable, then call the code that does the printing and adding separately for each item. This would be much longer to write out, less efficient, and more error-prone. If we had 10 items to add to the invoice it would already be annoying, but what about 100 items, or 1000? We'll return to this example later on in the article.

Depuración

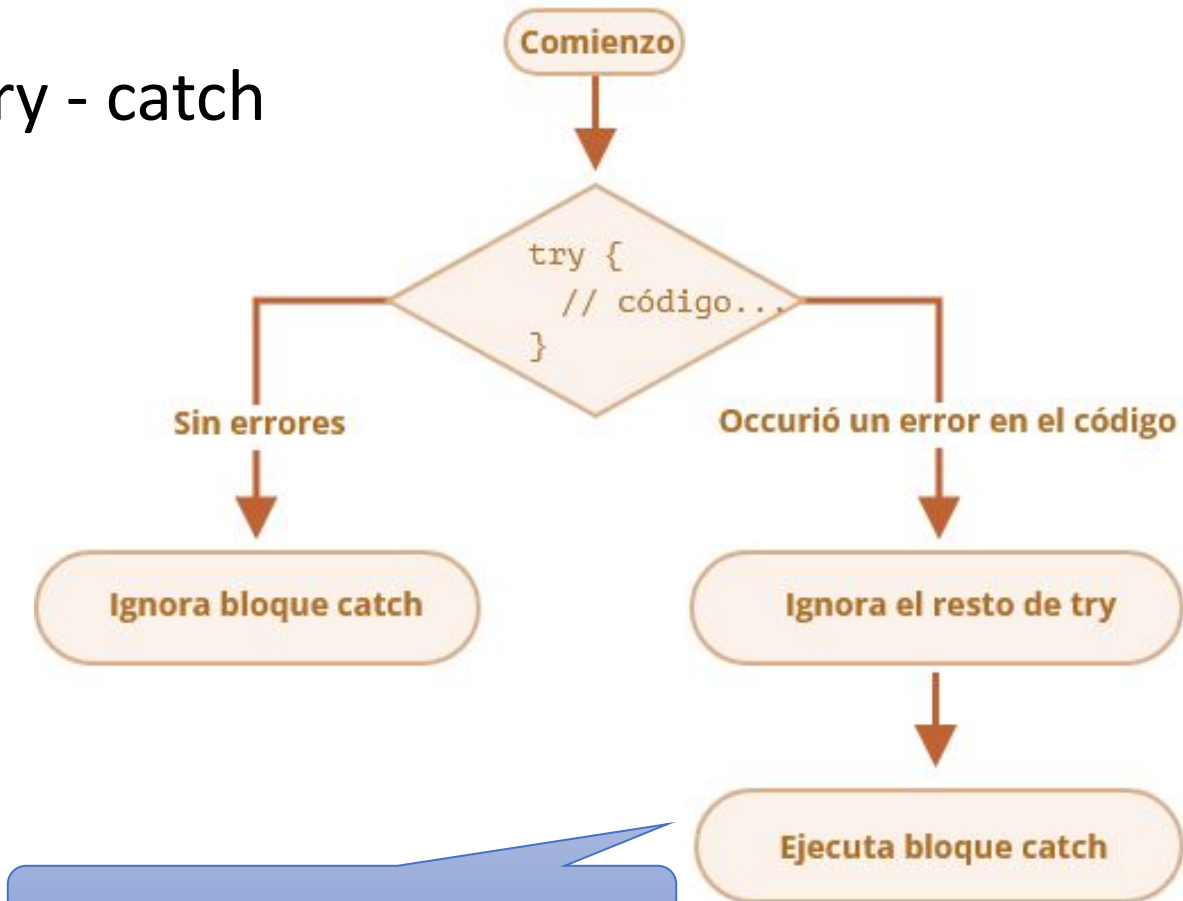
```
let x = 10;  
console.log("El valor de x es:", x);
```

```
let a = 10;  
let b = 5;  
if a > b {  
    console.log("A es mayor que B")  
}
```

¿Cuál es el error?

Control de errores

try - catch



¿Existe el bloque finally?

```
try {  
    alert('Inicio de ejecuciones try'); // (1) <--  
    lalala; // error, variable no está definida!  
    alert('Fin de try (nunca alcanzado)'); // (2)  
} catch (err) {  
    alert(`¡Un error ha ocurrido!`); // (3) <--  
}
```

`console.error(error);`

Otros entornos de desarrollo



Practicar: **piedra, papel, tijera**

Dos jugadores (usuario y la máquina) eligen entre "piedra", "papel" o "tijera".

Se determina un ganador según las reglas:

Piedra vence a Tijeras.

Tijeras vence a Papel.

Papel vence a Piedra.

Si ambos eligen lo mismo, es un empate.

Vamos a implementar un juego donde el usuario juega contra la máquina.

Practicar: piedra, papel, tijera

Ayuda:

The screenshot shows the MDN Web Docs page for the `Math.floor()` method. The page layout includes a top navigation bar with links to 'References', 'Guides', 'Plus', 'Curriculum', 'Blog', and 'Tools'. A search bar and a 'Sign up for free' button are also present. The main content area is titled 'Math.floor()' and features a green banner indicating 'Baseline Widely available' with browser compatibility icons. Below this, a description states that the method always rounds down and returns the largest integer less than or equal to a given number. A 'Try it' section contains a JavaScript demo with the following code:

```
JavaScript Demo: Math.floor()
1 console.log(Math.floor(5.95));
2 // Expected output: 5
3
4 console.log(Math.floor(5.05));
5 // Expected output: 5
6
7 console.log(Math.floor(5));
8 // Expected output: 5
9
10 console.log(Math.floor(-5.05));
11 // Expected output: -6
12
```

On the left side of the page, a sidebar lists various mathematical methods, including `Math.acosh()`, `Math.asin()`, `Math.asinh()`, `Math.atan()`, `Math.atan2()`, `Math.atanh()`, `Math.cbrt()`, `Math.ceil()`, `Math.clz32()`, `Math.cos()`, `Math.cosh()`, and `Math.exp()`.

Practicar: piedra, papel, tijera

¿Lo complicamos más?

- Puntuación: Haz que el programa guarde cuántas veces ha ganado el usuario y cuántas veces ha ganado la computadora. Muestra las puntuaciones al final del juego.
- Validación de entrada: Asegúrate de que el usuario pueda introducir tanto letras minúsculas como mayúsculas (por ejemplo, que "Piedra" o "PIEDRA" funcionen).
- Dos jugadores: Modifica el juego para que permita que dos usuarios jueguen entre sí en lugar de uno contra la máquina.
- Piedra, papel, tijera, lagarto, Spock:

<https://www.youtube.com/watch?v=iVVbWilPGf0>



Practicar

Generador de passwords

Descripción

En este reto se trata de que crees un script para generar passwords.

Las passwords deben tener una longitud comprendida entre 8 y 15 caracteres. Estos caracteres deberán tener al menos una letra mayúscula, otra minúscula, un dígito y un carácter especial

Los caracteres especiales serán uno de estos:

! @ # \$ % & * ? + = - /

Cuando tengas tu solución mira la que te propongo. ¡No tienen que ser iguales!

<https://www.aulascript.com/retos/gen-passwords.html>

Practicar

Retos: comprobar cierres emparejados

Descripción

Se trata de encontrar una función que reciba una expresión literal conteniendo signos de agrupamiento como paréntesis, corchetes y llaves. La función debe analizar el literal y comprobar que todos los signos están correctamente emparejados y anidados.

La función devolverá -1 si todo es correcto y un número indicando la posición en la que ha encontrado el error de anidamiento o cierre.

Por ejemplo:

```
simbEquilibrados("[1+x+3*(y-5)])" -> da -1
```

```
simbEquilibrados("[1+x)") -> da 4
```

```
simbEquilibrados(")1+x") -> da 0
```

https://www.aulascript.com/retos/signos_cierres.html

Practicar

¿Cuál es la salida de los siguientes fragmentos de código?

```
let x = 5;
```

```
let y = '5';
```

```
console.log(x == y);
```

```
console.log(x === y);
```



```
function test() {  
    console.log(a);  
    var a = 10;  
    console.log(a);  
}
```

```
test();
```

```
setTimeout(() => {  
    console.log("First");  
}, 1000);  
  
console.log("Second");
```

```
for (var i = 0; i < 3; i++) {  
    setTimeout(() => {  
        console.log(i);  
    }, 1000);  
}
```

```
const a = 10;  
let b = 20;  
  
{  
  const a = 30;  
  let b = 40;  
  console.log(a); // línea 1  
  console.log(b); // línea 2  
}  
  
console.log(a); // línea 3  
console.log(b); // línea 4
```

```
const arr = [1, 2, 3];  
const result = arr.map(num => {  
  if (num % 2 === 0) {  
    return num * 2;  
  }  
});  
  
console.log(result);
```

```
let fruits = ['apple', 'banana', 'orange'];  
let result = fruits.slice(1, 2);  
console.log(result);
```

```
const arr1 = [1, 2, 3];  
const arr2 = [...arr1, 4, 5];  
console.log(arr2);
```

```
try {  
    console.log("Start");  
    throw new Error("Something went wrong");  
} catch (e) {  
    console.log(e.message);  
} finally {  
    console.log("End");  
}
```