

Curso de JavaScript Básico

Prerequisitos

- ✓ Una cuenta en Github (<https://github.com/>)

Es completamente gratis! Github es una especie de red social para desarrolladores y un repositorio de código.

- ✓ Buena actitud de aprendizaje!

¿Por qué JavaScript?

- ✓ Es el lenguaje de programación más popular de la actualidad.
- ✓ Es el único lenguaje que entienden los navegadores.
- ✓ Gracias a una plataforma llamada `Node.js` es posible ejecutar JavaScript por fuera del navegador.
- ✓ Se pueden crear todo tipo de aplicaciones con JavaScript: aplicaciones de escritorio, aplicaciones móviles, aplicaciones para la línea de comandos, backends de aplicaciones Web, entre otros.

Primeros Pasos

- ✓ Ejecutar código en repl.it .
- ✓ Ver errores de sintaxis.
- ✓ Comentarios

Tipos y operadores básicos

✓ Números

Operadores aritméticos (suma, resta, multiplicación, división, módulo, exponentes), precedencia.

✓ Cadenas de texto (strings)

Definir, imprimir, concatenar

✓ Booleanos

Expresiones booleanas, && (y), || (ó) y ! (negación).

Variables

Las variables nos permiten almacenar información temporal que podemos usar más adelante en nuestros programas.

```
var nombre = "Pedro Perez";  
var edad = 20;  
var casado = false;  
  
console.log(nombre);  
console.log(edad);  
console.log(casado);
```

Nombramiento de variables

El nombre de una variable debe comenzar con \$, _ o una letra. Después puede contener letras, dígitos, _ y \$.

```
var primerNombre;  
var $edad = 20;  
var _c4s4do = false;
```



```
var 6nombre;  
var ed&ad = 20;
```



Obtener información del usuario

En el navegador podemos usar el comando `prompt` para pedirle información al usuario.

```
var nombre = prompt("¿Cuál es tu nombre?");  
console.log("Hola" + nombre);
```


Convirtiendo un string a un número

`prompt` siempre nos devuelve un string. Si queremos convertir el string a un número debemos usar `parseInt`.

```
var edad = prompt("¿Cuál es tu edad?");  
edad = parseInt(edad)
```

Condicionales

Las condicionales nos permiten tomar diferentes caminos en la ejecución del código de acuerdo a los criterios que definamos.

La sintaxis más simple de un condicional es la siguiente:

```
if (<condición>) {  
    // código que se ejecuta si se cumple la condición  
}
```

Condicionales (else)

Existen dos atajos que te van a permitir escribir código más corto: `else` y `else if`

```
if (<condición>) {  
    // código que se ejecuta si se cumple la condición  
} else {  
    // código que se ejecuta si NO se cumple la condición  
}
```

Condicionales anidados

```
var num = 8;

if (num < 10) {
  console.log("El número es menor a 10");
} else {
  if (num > 10) {
    console.log("El número es mayor a 10");
  } else {
    console.log("El número es igual a 10");
  }
}
```

Condicionales (else if)

Existen dos atajos que te van a permitir escribir código más corto: `else` y `else if`

```
if (<condición 1>) {  
    // código que se ejecuta si la condición 1 se cumple  
} else if (<condición 2>) {  
    // código que se ejecuta si condición 1 NO se cumple  
    // pero condición 2 se cumple  
} else if (<condición 3>) {  
    // código que se ejecuta si condición 1 y condición 2  
    // NO se cumplen pero condición 3 se cumple  
} else {  
    // código si ninguna de las condiciones se cumple  
}
```

Operadores lógicos

Podemos utilizar los operadores lógicos y (&&) y ó (||) para crear condiciones más complejas.

```
var num = 15;

if (num >= 10 && num <= 20) {
  console.log("El número está entre 10 y 20");
}
```

```
var num = 8;

if (num < 10 || num > 20) {
  console.log("El número está por fuera del rango");
}
```

Operador ternario

Nos permite escribir condicionales en una sola línea. Se recomienda utilizarlo únicamente en condicionales cortos.

```
<condición> ? <si verdadero> : <si falso>
```

```
var num = 10
```

```
console.log(num > 10 ? "Mayor" : "Menor o igual")
```

Ciclos (while)

Los ciclos nos permiten repetir la ejecución de un trozo de código varias veces.

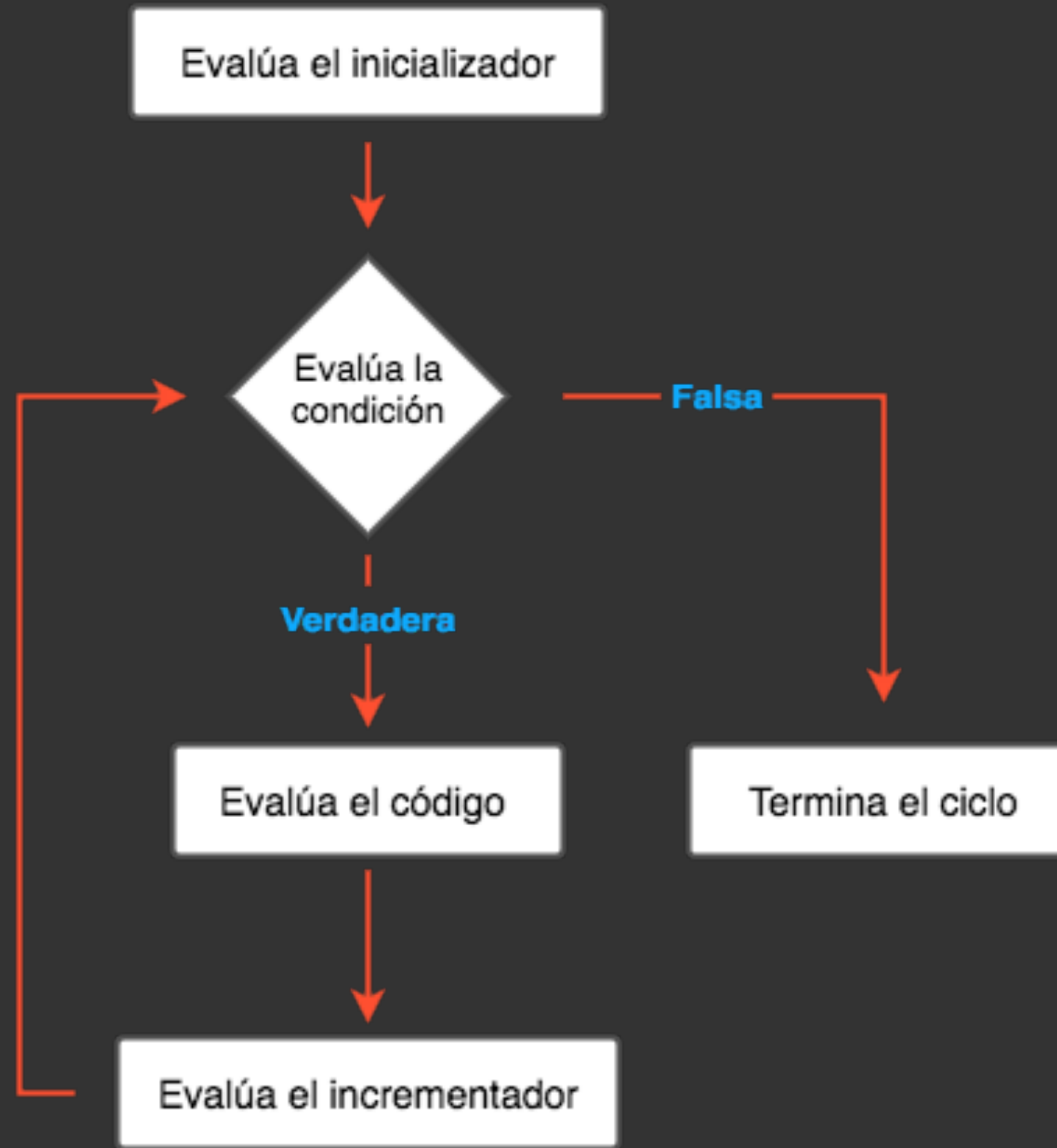
```
while (<condición>) {  
    // código que se ejecuta mientras se cumple  
    // la condición  
}
```


Ciclos (for)

`for` es un atajo para el patrón de ciclos que requiere una combinación de inicialización, condición e incremento:

```
for (<inicializador> : <condición> : <incremento>) {  
    // código que se ejecuta mientras se cumple  
    // la condición  
}
```

Ciclos (for)



Arreglos

Un arreglo es una lista ordenada de elementos. Los elementos pueden ser de cualquier tipo: números, booleanos, strings, otros arreglos u objetos

```
var array = [1, "Pedro", true, [2, "Juan"]]
```

Arreglos (operaciones básicas)

Las operaciones básicas son:

- Obtener el valor de una posición.
- Recorrer el arreglo.
- Reemplazar un elemento.
- Insertar nuevos elementos.
- Eliminar elementos.

Operaciones con strings

Las operaciones más importantes con los strings son:

- Obtener la longitud.
- Recorrer un string.
- Dividir un string.
- Unir un arreglo en un string.
- Pasar a mayúsculas o minúsculas.
- Extraer una parte del string.
- Reemplazar parte del string.
- Validar si un string contiene una subcadena específica.

Funciones

Una función es un bloque de código reutilizable que vas a poder **invocar** desde otros lugares de la aplicación para realizar alguna tarea específica.

Una función se compone de un nombre, unos argumentos, un cuerpo y, opcionalmente, un valor de retorno.

```
function hola() {  
  console.log("Hola Mundo");  
}  
  
hola()
```

Funciones (argumentos)

Las funciones pueden recibir cero o más argumentos (también conocidos como parámetros).

```
function hola(name) {  
  console.log("Hola " + name);  
}  
  
hola("Pedro")  
hola("María")
```

Funciones (retorno)

Las funciones pueden, opcionalmente, retornar un valor utilizando la palabra clave `return`:

```
function hola(name) {  
    return "Hola " + name;  
}  
  
var g1 = hola("Pedro");  
console.log(hola("María"));
```


Anatomía de una función

The diagram illustrates the components of a JavaScript function definition. It shows the code: `function calculateArea(width, height) { var area = width * height; return area; }`. Brackets and labels identify the parts: *Palabra clave* (keyword) for `function`, *Nombre* (name) for `calculateArea`, *Argumentos* (arguments) for `(width, height)`, *Cuerpo* (body) for the code inside the curly braces, and *valor de retorno* (return value) for the `return` statement.

```
function calculateArea(width, height) {  
    var area = width * height;  
    return area;  
}
```

Labels and their corresponding parts:

- Palabra clave*: `function`
- Nombre*: `calculateArea`
- Argumentos*: `(width, height)`
- Cuerpo*: `{ var area = width * height; return area; }`
- valor de retorno*: `return area;`

Funciones como cajas negras

