



Somos un **ecosistema** de desarrolladores de software

Estructuras de control Java Script



">Home


```
.html">Home Events</a></li>
```

Menu.html">Multiple Column Men

```
<a href="#" class="current"
```

utton-header.html">Tall But

[logo.html">Image Logo](#)



```
curl -s https://tall-logo.html
```

`f="#">Carousels`

with_slider.html">Variab

14des.html">Test1mon1

Estructuras de control

Propiedades

Las estructuras de control son herramientas fundamentales en programación que permiten alterar el flujo de ejecución de un programa. En JavaScript, hay tres tipos principales de estructuras de control:

- Estructuras de control condicionales: Permiten tomar decisiones basadas en condiciones. La instrucción `if` es un ejemplo común, donde un bloque de código se ejecutará si una condición es verdadera. También hay estructuras más complejas como `else if` y `switch`.
- Estructuras de control de bucle (ciclos): Permiten repetir un bloque de código varias veces. Los bucles más comunes son `for`, `while`, y `do-while`.
- Estructuras de control de salto: Permiten alterar el flujo normal de ejecución. Un ejemplo es la instrucción `break` para salir de un bucle, o `continue` para pasar a la siguiente iteración de un bucle.



For each

forEach es un método disponible para arrays en JavaScript que permite ejecutar una función proporcionada una vez por cada elemento del array. Es una forma más moderna y legible de iterar sobre elementos en comparación con el bucle for tradicional. La función que se pasa como argumento se ejecuta para cada elemento del array.

For each

sintaxis

```
array.forEach(function(element, index, array) {  
    // código a ejecutar para cada elemento  
});
```

element: El elemento actual del array.

index: El índice del elemento actual.

array: El array sobre el que se está iterando.

For each

Sintaxis simple, iterar sobre elementos de un array, manipulación de elementos del array

```
array.forEach(function(elemento) {  
    // Código a ejecutar para cada elemento  
});  
  
// iterar sobre elementos de un array  
let numeros = [1, 2, 3, 4, 5];  
numeros.forEach(function(numero) {  
    console.log(numero);  
});  
// Resultado: 1, 2, 3, 4, 5
```

```
//Manipulación de elementos del array  
let numeros = [1, 2, 3, 4, 5];  
let cuadrados = [];  
numeros.forEach(function(numero) {  
    cuadrados.push(numero * numero);  
});  
console.log(cuadrados);  
// Resultado: [1, 4, 9, 16, 25]
```

For each

Ejemplo

```
let fruits = ['apple', 'banana', 'orange'];

fruits.forEach(function(fruit, index) {
  console.log(`Fruit at index ${index}:
${fruit}`);
});
```

```
//Resultado//
Fruit at index 0: apple
Fruit at index 1: banana
Fruit at index 2: orange
```

For each

Arrow

```
let numbers = [1, 2, 3, 4, 5];

numbers.forEach((number, index) => {
  console.log(`Number at index ${index}:
${number}`);
});
```


For each

Con objetos

```
//for each con objetos
```

```
let person = { name: 'John', age: 30, job: 'developer' };
```

```
Object.keys(person).forEach(function(key) {  
  console.log(`${key}: ${person[key]}`);  
});
```

```
//Resultado
```

```
name: John
```

```
age: 30
```

```
job: developer
```

For each

Array con objetos

```
//Array con objetos
let students = [
  { name: 'Alice', grade: 85 },
  { name: 'Bob', grade: 92 },
  { name: 'Charlie', grade: 78 }
];

students.forEach(function(student) {
  console.log(`${student.name} scored ${student.grade}`);
});
```

Map

map es un método que se encuentra disponible en los arrays de JavaScript. Su función principal es iterar sobre cada elemento de un array y aplicar una función proporcionada a cada elemento, creando así un nuevo array con los resultados de aplicar la función a cada elemento original.



Map

sintaxis

```
//sintaxis
```

```
const nuevoArray = arrayOriginal.map(function(elemento) {  
  // Código a aplicar a cada elemento  
  return nuevoElemento;  
});
```

Map

Duplicar cada elemento de un array

```
//Duplicar cada elemento de un array  
const numeros = [1, 2, 3, 4, 5];  
const duplicados = numeros.map(function(numero) {  
  return numero * 2;  
});  
console.log(duplicados);  
// Resultado: [2, 4, 6, 8, 10]
```

Map

Convertir cadenas a mayúsculas

```
//convertir cadenas a mayúsculas  
const palabras = ['hola', 'mundo', 'javascript'];  
const mayusculas = palabras.map(function(palabra) {  
  return palabra.toUpperCase();  
});  
console.log(mayusculas);  
// Resultado: ['HOLA', 'MUNDO', 'JAVASCRIPT']
```

Map

Crear un nuevo array con objetos adicionales

```
//Crear un nuevo array de objetos con propiedades adicionales
const personas = [
  { nombre: 'Juan', edad: 25 },
  { nombre: 'María', edad: 30 },
  { nombre: 'Pedro', edad: 22 }
];

const personasConEdadDoble = personas.map(function(persona) {
  return { ...persona, edadDoble: persona.edad * 2 };
});

console.log(personasConEdadDoble);
// Resultado:
// [
//   { nombre: 'Juan', edad: 25, edadDoble: 50 },
//   { nombre: 'María', edad: 30, edadDoble: 60 },
//   { nombre: 'Pedro', edad: 22, edadDoble: 44 }
// ]
```

Map

Map con Filter



```
//map con filter  
const numeros = [1, 2, 3, 4, 5];  
const cuadradosDePares = numeros  
  .filter(numero => numero % 2 === 0)  
  .map(numero => numero * numero);  
console.log(cuadradosDePares);  
// Resultado: [4, 16]
```


Map

Map con For each



```
//map con for each  
const numeros = [1, 2, 3, 4, 5];  
const cuadrados = [];  
numeros.map(numero => cuadrados.push(numero * numero));  
console.log(cuadrados);  
// Resultado: [1, 4, 9, 16, 25]
```

Map

Map con reduce



```
//map con reduce  
const numeros = [1, 2, 3, 4, 5];  
const sumaCuadrados = numeros  
  .map(numero => numero * numero)  
  .reduce((acumulador, valorActual) => acumulador +  
valorActual, 0);  
console.log(sumaCuadrados);  
// Resultado: 55
```

</Be a
coder>