

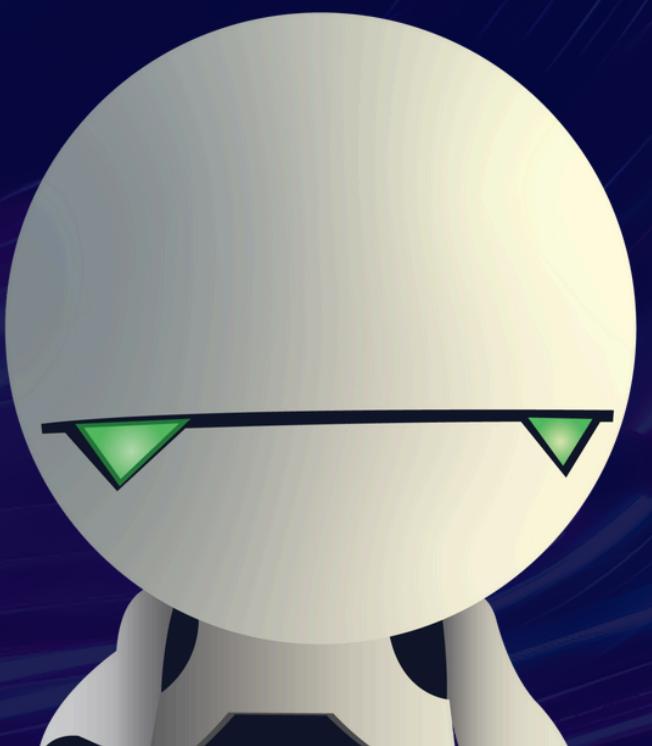
**CICLOS Y SCOPE EN
DS.**

¿QUÉ ES FOREACH?

FOREACH ES UN MÉTODO DISPONIBLE EN LOS ARRAYS EN JAVASCRIPT. ESTE MÉTODO EJECUTA UNA FUNCIÓN PROPORCIONADA UNA VEZ POR CADA ELEMENTO DEL ARRAY, EN ORDEN.

Sintaxis de forEach

```
array.forEach(callback(currentValue, index, array), thisArg);
```



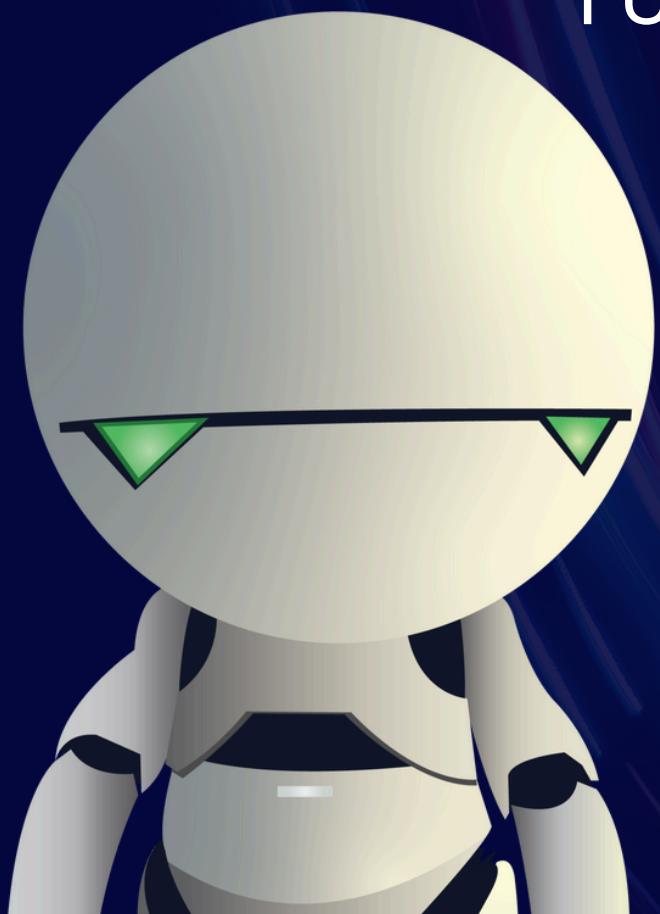
SINTAXIS DE FOREACH JAVASCRIPT

ARRAY.FOREACH (CALLBACK(CURRENTVALUE, INDEX, ARRAY), THISARG);

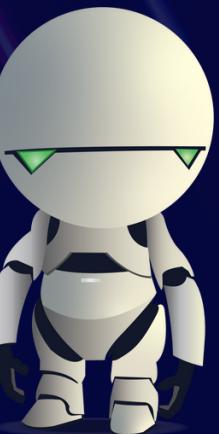
CALLBACK: FUNCIÓN A EJECUTAR EN CADA ELEMENTO DEL ARRAY. TIENE TRES PARÁMETROS:

- CURRENTVALUE: EL VALOR DEL ELEMENTO ACTUAL DEL ARRAY.
- INDEX (OPCIONAL): EL ÍNDICE DEL ELEMENTO ACTUAL DEL ARRAY.
- ARRAY (OPCIONAL): EL ARRAY SOBRE EL QUE SE ESTÁ ITERANDO.

THISARG (OPCIONAL): VALOR A USAR COMO THIS CUANDO SE EJECUTA LA FUNCIÓN CALLBACK.



EJEMPLO DETALLADO



```
CONST NUMBERS = [10, 20, 30, 40, 50];
```

```
NUMBERS.FOREACH(FUNCTION(CURRENTVALUE, INDEX, ARRAY) {  
    CONSOLE.LOG('VALOR ACTUAL:', CURRENTVALUE);  
    CONSOLE.LOG('ÍNDICE ACTUAL:', INDEX);  
    CONSOLE.LOG('ARRAY COMPLETO:', ARRAY);  
    CONSOLE.LOG('-----');  
});
```

EXPLICACIÓN DEL CALLBACK

1. CURRENT VALUE: ES EL VALOR DEL ELEMENTO ACTUAL DEL ARRAY EN LA ITERACIÓN.
2. INDEX (OPCIONAL): ES EL ÍNDICE DEL ELEMENTO ACTUAL DEL ARRAY EN LA ITERACIÓN.
3. ARRAY (OPCIONAL): ES EL ARRAY COMPLETO SOBRE EL CUAL ESTAMOS ITERANDO.

EJEMPLO DETALLADO

SUPONGAMOS QUE TENEMOS UN OBJETO CON UN VALOR Y QUEREMOS MULTIPLICAR CADA NÚMERO DEL ARRAY POR ESE VALOR.

```
CONST MULTIPLIER = {  
    VALUE: 2  
};
```

```
CONST NUMBERS = [10, 20, 30, 40, 50];
```

```
NUMBERS.FOREACH(FUNCTION(CURRENTVALUE, INDEX, ARRAY) {  
    CONSOLE.LOG('VALOR ACTUAL:', CURRENTVALUE);  
    CONSOLE.LOG('MULTIPLICADO POR:', THIS.VALUE);  
    CONSOLE.LOG('RESULTADO:', CURRENTVALUE * THIS.VALUE);  
    CONSOLE.LOG('-----');  
}, MULTIPLIER);
```

EXPLICACIÓN DEL EJEMPLO CON THISARG

- MULTIPLIER: ES UN OBJETO QUE CONTIENE UN VALOR VALUE QUE UTILIZAREMOS PARA MULTIPLICAR CADA NÚMERO DEL ARRAY.
- THIS.VALUE: DENTRO DEL CALLBACK, THIS SE REFIERE AL OBJETO MULTIPLIER DEBIDO A QUE LO PASAMOS COMO THISARG.



DIFERENCIAS CLAVE DE FOREACH

1. No tiene retorno: forEach no devuelve un valor. Si necesitas un nuevo array basado en el original, considera usar métodos como map.
2. No puede ser interrumpido: No se puede usar break, continue o return dentro de un forEach para terminar o saltar iteraciones. Si necesitas esta funcionalidad, es mejor usar un bucle for.
3. Más funcional: forEach se adhiere más al estilo de programación funcional, pasando una función callback para ser ejecutada en cada elemento del array

INTRODUCCIÓN AL SCOPE

El scope (alcance) se refiere al contexto en el cual se define una variable o una función. Define la accesibilidad y la vida útil de esas variables y funciones.



TIPOS DE SCOPE EN JAVASCRIPT

Global Scope

Las variables y funciones definidas fuera de cualquier función o bloque tienen un alcance global y pueden ser accedidas desde cualquier parte del código.

```
var globalVar = "Soy una variable global";
```

```
function showGlobalVar() {  
  console.log(globalVar); // Puede acceder a globalVar  
}
```

```
showGlobalVar(); // "Soy una variable global"
```

LOCAL SCOPE (FUNCIONAL)

Las variables definidas dentro de una función tienen un alcance local y solo pueden ser accedidas desde dentro de esa función.

```
function localScopeExample() {  
  var localVar = "Soy una variable local";  
  console.log(localVar); // Puede acceder a localVar  
}
```

```
localScopeExample(); // "Soy una variable local"  
console.log(localVar); // Error: localVar no está definida
```

BLOCK SCOPE

Con la introducción de `let` y `const` en ES6, se añadió el concepto de scope de bloque. Las variables definidas con `let` o `const` dentro de un bloque (`{}`) solo son accesibles dentro de ese.

```
{  
  let blockVar = "Soy una variable de bloque";  
  console.log(blockVar); // Puede acceder a blockVar  
}  
  
console.log(blockVar); // Error: blockVar no está definida
```

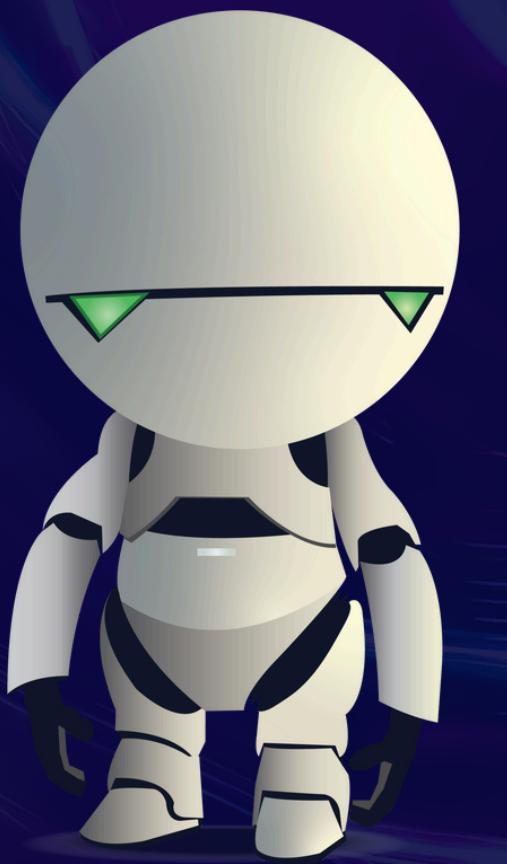


SCOPE ANIDADO

El scope en JavaScript es jerárquico. Las funciones pueden anidarse dentro de otras funciones, creando scopes anidados. Las funciones internas tienen acceso a las variables definidas en sus funciones externas.

```
function outerFunction() {  
    var outerVar = "Variable externa";  
  
    function innerFunction() {  
        var innerVar = "Variable interna";  
        console.log(outerVar); // Puede acceder a outerVar  
        console.log(innerVar); // Puede acceder a innerVar  
    }  
  
    innerFunction();  
    console.log(innerVar); // Error: innerVar no está definida  
}  
  
outerFunction();
```

PREGUNTAS EN VIVO



RETO DE HOY

Crear una aplicación web que permita a los usuarios gestionar una lista de compras de frutas. Los usuarios podrán agregar, eliminar y editar elementos en la lista. El proyecto utilizará arrays para almacenar y manipular los elementos de la lista.

REQUISITOS

1. HTML:

- UN BOTÓN PARA AGREGAR EL ELEMENTO A LA LISTA.
- UNA LISTA (``) PARA MOSTRAR LOS ELEMENTOS

2. CSS:

- LA APLICACIÓN DEBE SER VISUALMENTE ATRACTIVA.

3. JAVASCRIPT:

- UTILIZAR UN ARRAY PARA ALMACENAR LOS ELEMENTOS DE LA LISTA.
- PERMITIR AGREGAR ELEMENTOS AL ARRAY Y ACTUALIZAR LA VISTA.
- PERMITIR ELIMINAR ELEMENTOS DEL ARRAY Y ACTUALIZAR LA VISTA.

FELIZ noche