BOOTCAMP

Generación T

streambe

Introducción a funciones
Parte 2

Arrow Functions

Una forma moderna y concisa de escribir funciones en JavaScript, introducida en ES6.

Sintaxis Simplificada

Menos código para funciones simples. Ideal para callbacks y funciones pequeñas.

No Modifican 'this'

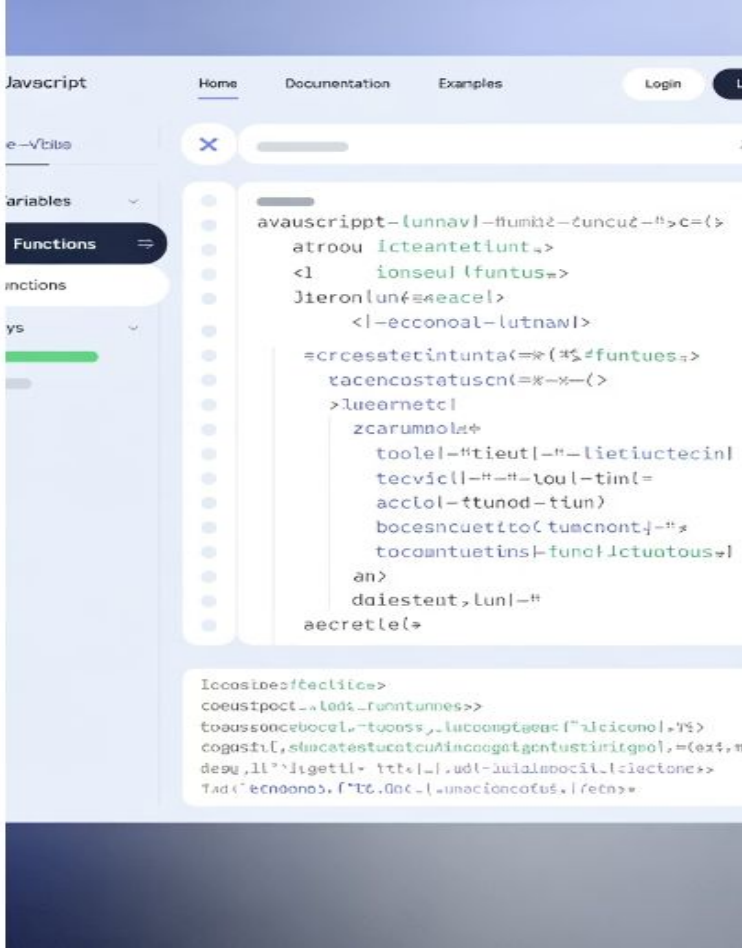
Heredan el contexto del ámbito donde se definen.

Ejemplo Básico

```
const sumar = (a, b) => a + b;
```

Sin Return Explícito

Devuelven automáticamente el resultado de la expresión en funciones de una línea.



Callback Functions

Funciones que se pasan como argumento a otras funciones y se ejecutan después de que ocurra un evento o se complete una operación.

Asincronía

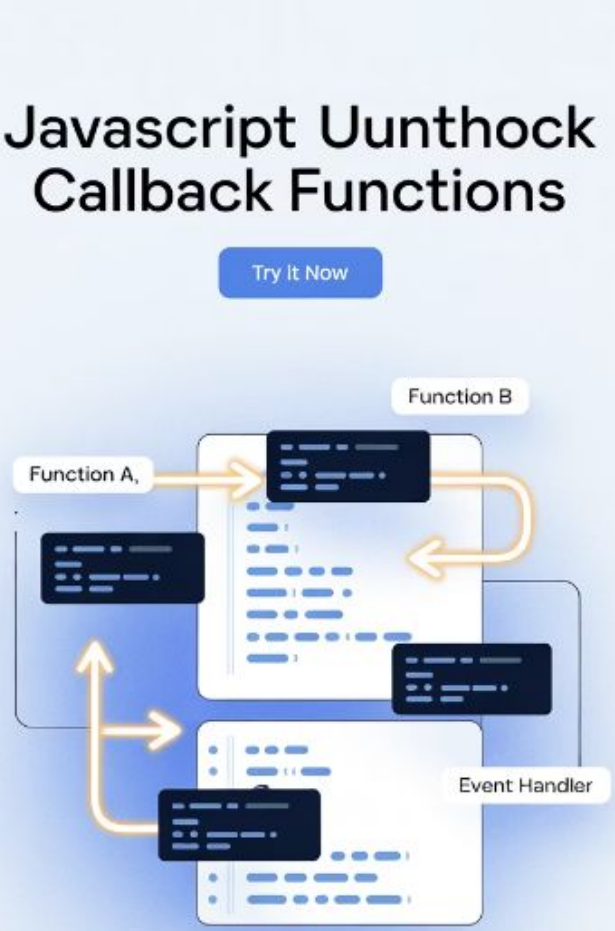
Permiten ejecutar código después de completar tareas que toman tiempo.

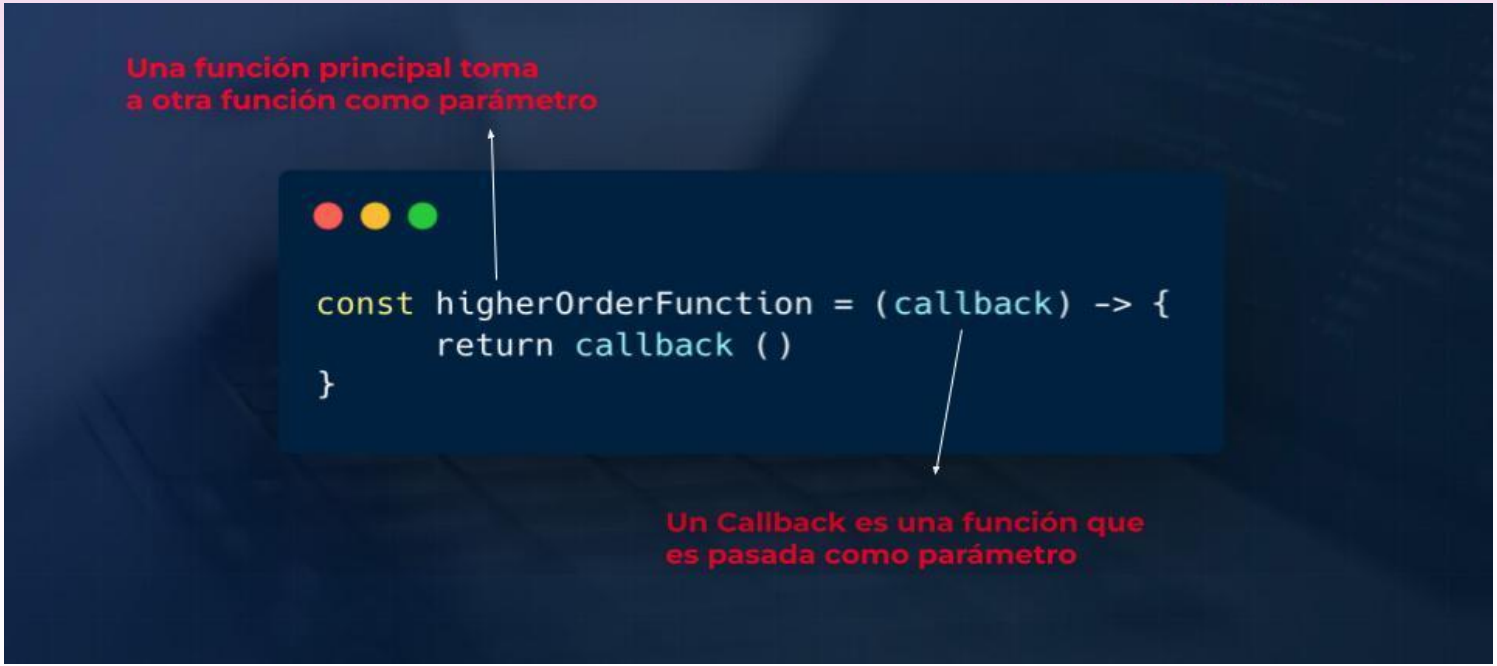
Reutilización

Facilitan la creación de código modular y reutilizable.

Ejemplo

```
array.forEach(item => console.log(item));
```




Una función principal toma
a otra función como parámetro
const higherOrderFunction = (callback) -> {
return callback ()
}
Un Callback es una función que
es pasada como parámetro

Strings Como Colecciones De Caracteres

Javascript

Strings Como Colecciones De Caracteres

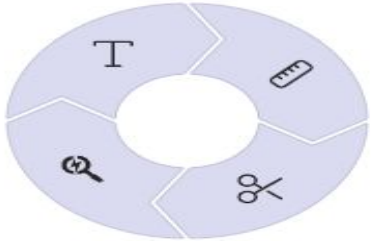
Las cadenas de texto en JavaScript son secuencias ordenadas de caracteres que pueden manipularse como colecciones.

Indexación

Cada carácter tiene una posición numérica iniciando en 0.

Iteración

Podemos recorrer cada carácter usando bucles for.



Longitud

La propiedad length devuelve el total de caracteres.

Métodos

substring(), slice() y split() permiten extraer partes específicas.

```
const texto = "JavaScript";
console.log(texto[0]);          // "J"
console.log(texto.length);      // 10
console.log(texto.slice(4));    // "Script"
```

El Índice en Strings y Arrays

El índice revela la posición exacta de cada carácter en un string o elemento en un array en JavaScript.

⚠️ **Importante:** El primer índice siempre es cero (0), no uno.

Tт	⌨	3	
Declaración	**Acceso**	**Longitud**	**Fuera de rango**
const nombre = "María";	nombre[0] // "M"	nombre.length // 5	nombre[10] // undefined

```
const texto = "JavaScript";
console.log(texto[0]);     // "J"
console.log(texto[4]);     // "S"
console.log(texto[texto.length-1]); // "t"
```

Arreglos en JavaScript

Los arreglos son colecciones ordenadas de elementos que almacenan múltiples valores en una sola variable. Funcionan como contenedores flexibles para datos relacionados.

Declaración

const frutas = ["manzana", "banana", "naranja"];

Acceso

frutas[0] // "manzana"

Métodos

push(), pop(), splice(), forEach(), map()

Propiedad Length

frutas.length // 3

Los arreglos pueden contener cualquier tipo de dato, incluso otros arreglos u objetos. Su naturaleza indexada los hace perfectos para algoritmos y manipulación de datos.

Sintaxis De Los Arreglos

La estructura base para crear y manipular arreglos en JavaScript sigue patrones consistentes y flexibles.

Declaración Literal

const colores = ["rojo", "verde", "azul"];

Añadir Elementos

colores.push("amarillo"); // Añade al final

Eliminar Elementos

colores.pop(); // Elimina el último

Modificar Elementos

colores[0] = "naranja"; // Cambia el primer elemento

Índice De Un Arreglo En JavaScript

Los arreglos usan índices numéricos para acceder a cada elemento, igual que los strings.

Acceso

miArray[2] devuelve el tercer elemento del arreglo.

Modificación

miArray[0] = "nuevo" cambia el primer elemento.

Precaución

El acceso fuera de rango retorna undefined sin error.

```
const tecnologías = ["HTML", "CSS", "JavaScript"];
console.log(tecnologías[0]); // "HTML"
console.log(tecnologías[2]); // "JavaScript"
```

Usos De Los Índices

Los índices nos permiten manipular colecciones de datos con precisión y eficiencia.

Acceso a elementos

Permite extraer valores específicos tanto en strings como en arreglos.

Modificación

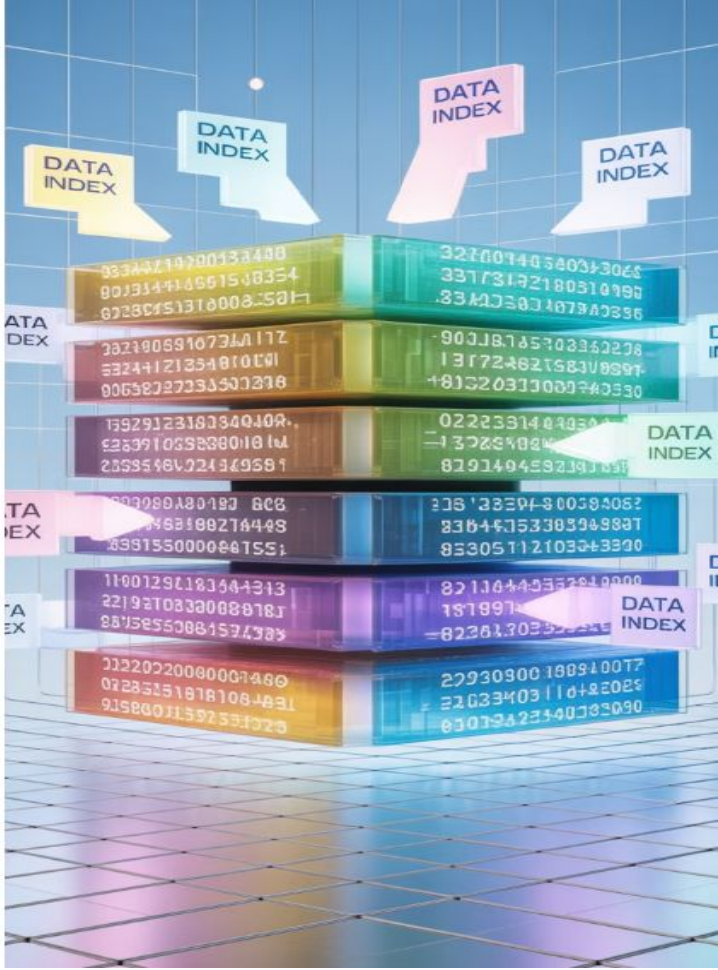
Cambia valores individuales en arreglos usando su posición numérica.

Iteración

Facilita recorrer elementos secuencialmente mediante bucles y ciclos.

Extracción de subconjuntos

Permite obtener porciones específicas usando métodos como slice().



Cómo Acceder A Los Datos De Un Arreglo

Múltiples técnicas nos permiten extraer y manipular elementos dentro de un arreglo.

{ }

Notación de corchetes

miArray[0] accede al primer elemento del arreglo.

Bucles

for, while y do...while permiten recorrer todos los elementos secuencialmente.

3

Métodos de iteración

forEach(), map(), filter() procesan elementos sin bucles tradicionales.

Desestructuración

const [primero, segundo] = miArray extrae valores a variables individuales.

Métodos De Arreglos (Array Methods)

JavaScript ofrece poderosas herramientas incorporadas para manipular arreglos sin escribir código personalizado.

Filtrado

filter() crea un nuevo arreglo con elementos que pasan una prueba.

Transformación

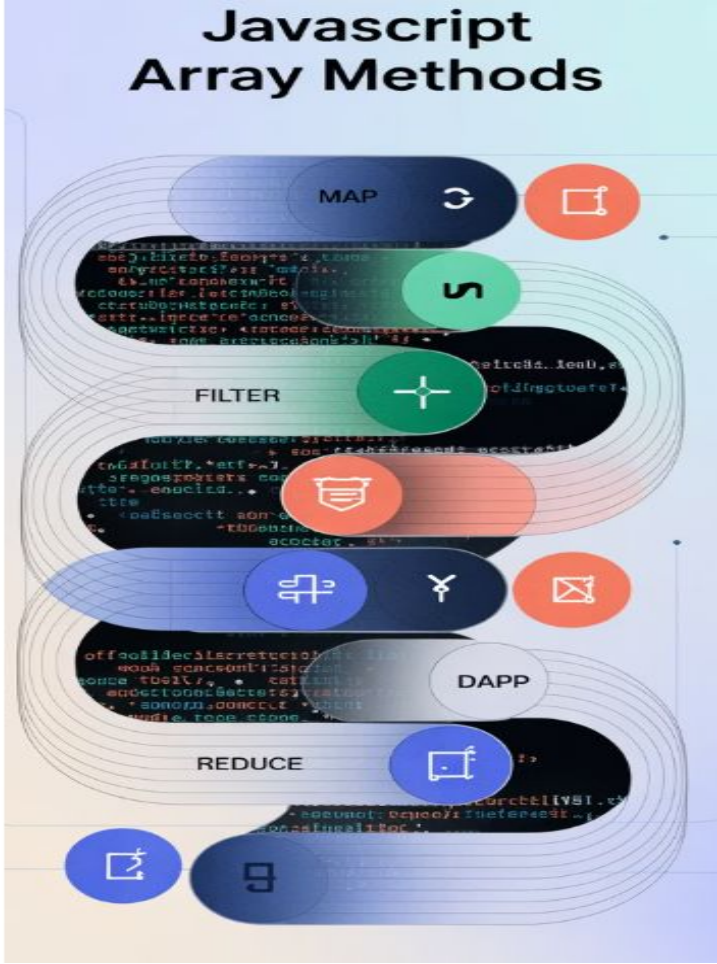
map() crea un nuevo arreglo aplicando una función a cada elemento.

Ordenamiento

sort() organiza los elementos según criterios personalizados o predeterminados.

Reducción

reduce() combina todos los elementos en un único valor resultante.



push() Y pop() En Arrays

Los métodos push() y pop() permiten manipular arrays como estructuras de pila (LIFO).

push()

- Añade elementos al final del array
- Modifica el array original
- Retorna la nueva longitud del array

```
const frutas = ["manzana"]; frutas.push("naranja"); // frutas = ["manzana", "naranja"]
```

pop()

- Elimina el último elemento del array
- Modifica el array original
- Retorna el elemento eliminado

```
const último = frutas.pop(); // último = "naranja", frutas = ["manzana"]
```

Funcionamiento Tipo Pila

Estos métodos implementan el concepto "último en entrar, primero en salir" (LIFO).

Ideal para manejar historial, deshacer acciones o gestionar procesos secuenciales.

Uso Práctico

- Gestión de navegación por pestañas
- Historial de acciones del usuario
- Procesamiento de datos en orden específico

unshift() y shift() En Arrays

Los métodos unshift() y shift() manipulan arrays como estructuras de cola (FIFO), operando desde el inicio del array.

+

unshift()

- Añade elementos al principio del array
- Modifica el array original
- Retorna la nueva longitud del array

const frutas = ["plátano"]; frutas.unshift("fresa"); // frutas = ["fresa", "plátano"]

⊟

shift()

- Elimina el primer elemento del array
- Modifica el array original
- Retorna el elemento eliminado

const primero = frutas.shift(); // primero = "fresa", frutas = ["plátano"]

Funcionamiento Tipo Cola

Estos métodos implementan el concepto "primero en entrar, primero en salir" (FIFO).

Complementan a push() y pop() para manipulación completa de arrays.

4

Uso Práctico

- Gestión de colas de procesamiento
- Implementación de buffers circulares
- Manipulación de listas de espera

El Método indexOf() en JavaScript

Búsqueda de Elementos

Encuentra la primera posición de un elemento dentro de un array o string.

Retorno de Índice

Devuelve el índice (número) donde se encuentra el elemento buscado.

Elemento No Encontrado

Retorna -1 si el elemento no existe en el array o string.

Sintaxis Simple

array.indexOf(elemento) o string.indexOf(substring)

El método indexOf() es fundamental para verificar la existencia y ubicación de elementos. Funciona tanto en arrays como en strings, permitiendo búsquedas eficientes.

```
const frutas = ["manzana", "pera", "naranja"];
const posicion = frutas.indexOf("pera"); // posicion = 1

const texto = "JavaScript";
const indice = texto.indexOf("Script"); // indice = 4
```

Hands on!

streambe GENERACIÓN T

Muchas gracias.

streambe GENERACIÓN T

Nuestras Redes

www.generaciont.org
www.streambe.com
www.instagram.com/generaciont_ar
www.tiktok.com/@generaciont
generaciont@generaciont.org
Cel: 11 61331747

GENERACIÓN T

streambe