


Arrays

Son estructuras de datos que te permiten almacenar múltiples valores en una sola variable.

Puedes acceder a los elementos del array mediante un índice numérico y realizar diversas operaciones, como agregar, eliminar y modificar elementos. Son dinámicos, lo que significa que pueden crecer o reducir su tamaño durante la ejecución del programa, y son una herramienta fundamental en la programación JavaScript para organizar y manipular conjuntos de datos.

js


 Copiar código

```
// ## ARRAYS
let lista = ["Manzana", "Pera", 23, 25, "Hola mundo"];
console.log(lista);
lista.push("Hola"); // Agregar un elemento al final
console.log(lista);
console.log(lista[2]);
console.log(lista[4]);
console.log(lista.length);

let copia = lista.slice(0, 3);
console.log(copia);
console.log(lista);

lista.pop(); // Eliminar el último elemento
console.log(lista);
```


js

 Copiar código

```
// Ejemplo:
const prueba = [
  1, 2, "true", "Bienvenido",
  ["F", "Y", 8,
    [3, 4, "Hola", "Maria"]
  ]
];

console.log(prueba[4] [3] [3]);
```

js


 Copiar código

```
// Ejemplo:  
const arreglo = Array.of("x", "y", "Hola", "Dario", 8, 9, 0);  
console.log(arreglo);
```

Array / .fill()


Se utiliza para llenar todos los elementos de un array con un valor estático dado. Puedes especificar el valor que deseas asignar a cada elemento del array y opcionalmente indicar el índice de inicio y fin del rango en el que deseas aplicar la operación de llenado. Esto es útil cuando deseas inicializar un array con un valor predeterminado o cuando necesitas rellenar parte de un array con un valor específico.

js

 Copiar código

```
// Ejemplo 1  
const valor = Array(200).fill(0);  
console.log(valor);
```

js

 Copiar código

```
// Ejemplo 2  
const array1 = [1, 2, 3, 4];  
  
console.log(array1.fill(0, 2, 4));  
console.log(array1.fill(5, 1));  
console.log(array1.fill(6));
```

js


 Copiar código

```
// Ejemplo 3
const frutas = ["Lulo", "Fresa", "Manzana", "Papaya"];
let primerE = frutas[0];
console.log(primerE);
```

Arrays / .forEach()

Es un método que permite recorrer todos los elementos de un array y ejecutar una función para cada uno de ellos. Esta función recibe el valor de cada elemento como argumento y puede ser utilizada para realizar operaciones en cada elemento del array de manera sencilla y eficiente.


js

 Copiar código

```
const frutas = ["Lulo", "Fresa", "Manzana", "Papaya"];


// Recorriendo Ejemplo 1. Arrays ForEach
frutas.forEach(function(el, index){
  console.log(`<li id = "${index}"> ${el} </li>`);
})
```

js

 Copiar código

```
// Recorriendo Ejemplo 2. Arrays ForEach
const npruebas = [10,20,30,40,50,60];
npruebas.forEach(function(e){
  console.log(e);
})
```

js


 Copiar código

```
// Recorriendo Ejemplo 3. Arrays ForEach
const frutass = ["Lulo", "Fresa", "Manzana", "Papaya", "Melocotón", "Mora", "Piña" ];
frutass.forEach( e =>
  console.log(e));
```

Arrays / .filter()

Crea un nuevo array con elementos que cumplen un criterio definido por una función de prueba. Esta función evalúa cada elemento del array y devuelve un nuevo array con los elementos que pasan la prueba. Es una forma eficiente de filtrar elementos basados en condiciones específicas sin modificar el array original.


js

 Copiar código

```
// Ejemplo 1 - con objetos
const Empleados = [
  {
    nombre : "Carlos",
    cargo : "Web master",
    salario : "5000000",
  },
  {
    nombre : "Natalia",
    cargo : "Diseñadora",
    salario : "3000000",
  },
  {
    nombre : "Gina",
    cargo : "Gerente",
    salario : "4000000",
  },
];

const calcular = Empleados.filter(value => value.salario >= 4000000);
console.log(calcular);
```

js


 Copiar código

```
// Ejemplo trayendo según el numero de caracteres, solo funciona con strings
const frutas2 = ["Manzana", "Pera", "Ajo", "Piña", "Durazno", 123, 12, "uva"];
const res = frutas2.filter(valor => valor.length > 4);
console.log(res);
```

Arrays / find()

Se utiliza para encontrar el primer elemento en un array que cumple con cierta condición definida en una función de prueba. Permite buscar un elemento específico en un array según criterios específicos y devuelve el primer elemento que cumple con la condición o undefined si no se encuentra ningún elemento que la cumpla. Es una herramienta útil y eficiente para búsquedas en arrays.

js

 Copiar código


```
const desarrollo = [
  {
    id: 1,
    lenguaje: "Javascript",
    programacion: "Web y escritorio",
  },
  {
    id: 2,
    lenguaje: "Php",
    programacion: "Web y escritorio",
  },
  {
    id: 3,
    lenguaje: "Java",
    programacion: "Web y escritorio",
  },
];
```

```
// find() recorre el arreglo y trae la primera coincidencia que encuentre.
const result = desarrollo.find(post => post.lenguaje == "Php");
console.log(result);
```

Arrays / map()

Se utiliza para crear un nuevo array transformando cada elemento de un array existente mediante una función de transformación definida. Esta función recibe cada elemento del array como argumento y devuelve el resultado de aplicar la transformación deseada a ese elemento. `.map()` es útil para modificar cada elemento de un array de manera independiente y crear un nuevo array con los resultados de estas transformaciones, sin modificar el array original. Es una herramienta poderosa para operaciones de mapeo y transformación de datos en JavaScript.


js

 Copiar código

```
// Ejemplo uno:
const programador = [
  {nombre: "Adrian", apellido: "Duran", lenguajes: "php",},
  {nombre: "Daniela", apellido: "Amaya", lenguajes: "Java y C#",},
  {nombre: "Sergio", apellido: "Montaño", lenguajes: ".net",},
  {nombre: "Duvan", apellido: "Diaz", lenguajes: "php"}
];

// Usando .map()
const estructuras = programador.map(pro =>{
  return{
    id: pro.nombre,
    lenguaje: pro.apellido,
    programacion: pro.lenguajes,
    datos: `${pro.nombre} - ${pro.lenguajes}`
  }
});
console.log(estructuras);
```

js

 Copiar código

```
// Ejemplo dos:
let numeros = [10, 20, 30, 40];
let multiplicar = 2;


let productos = numeros.map(numero => numero * multiplicar);
console.log(productos);
```

[Repasa el uso del método .map\(\) aquí](#)

Arrays / sort()


Ordena los elementos de un array alfabéticamente o numéricamente, modificando el array original. Por defecto, ordena alfabéticamente, pero puede usar una función personalizada para un criterio específico. Es una forma rápida de ordenar arrays.

js

 Copiar código

```
// Método sort(), re-ordena los elementos del array de la 'A' a la 'Z'
const miArray = [5, 12, 8, 130, 44, 78, 98];
const orden1 = miArray.sort((a, b) => a - b);
console.log(orden1);
console.log(miArray);
```

js

 Copiar código


```
// Ejemplo dos:
const miArray = [
  {id: 3, nombre: "Nidia", edad: 22},
  {id: 1, nombre: "Carl", edad: 66},
  {id: 2, nombre: "Jesus", edad: 55},
  {id: 5, nombre: "Ryan", edad: 44},
  {id: 4, nombre: "Albert", edad: 33},
];
// Ordenar por id menor a mayor:
miArray.sort((a, b) => a.id - b.id);
console.log(miArray);
/*
// Ordenar por edad menor a mayor:
miArray.sort((a, b) => a.edad - b.edad);
console.log(miArray);
*/
/*
// Ordenar por nombre menor a mayor:
miArray.sort((a, b) => a.nombre.localeCompare(b.nombre));
console.log(miArray);
*/
```

[Aprende cuando usar el método .sort\(\) aquí](#)

Arrays / toSorted()


ordena los elementos del array de la 'A' a la 'Z' sin re-ordenar el array original

js

 Copiar código

```
// Método toSorted(), ordena los elementos del array
// de la 'A' a la 'Z' sin re-ordenar el array original
const miArray2 = [5, 12, 8, 130, 44, 78, 98];
const orden2 = miArray2.toSorted((a, b) => a - b);
console.log(orden2);
console.log(miArray2);
```

js


 Copiar código

```
// Ejemplo dos:
const sorti = ["padre", "Madre", "Padre", "madre", 1, 2, 3, "hijo", "Hijo"];
sorti.sort();
console.log(sorti);
```

Tarea comparador

js

02.04.2024 S16-M2

 Copiar código

```
// función comparar
function compare(a,b){
  if(a < b ){
    return -1;
  }


  if(a > b){
    return 1;
  }

  return 0;
}

console.log(compare(2,6));
```

Usando el comparador anterior, crea un arreglo = 5 objetos(nombre,edad) y usa sort para que compare, estilo callback, es decir pasar a función el sort()

js

 Copiar código

```
// función comparar
function compare(a, b){
  if(a < b){
    return -1;
  }
  if(a > b){
    return 1;
  }
  return 0;
}

// Array de objetos
const miArray = [
  {nombre: "Nidia", edad: 22},
  {nombre: "Carl", edad: 66},
  {nombre: "Jesus", edad: 55},
  {nombre: "Ryan", edad: 44},
  {nombre: "Albert", edad: 33},
];

// Ordenar por edad de menor a mayor
miArray.sort((a, b) => compare(a.edad, b.edad));

console.log(miArray);
```

Bryan Hernández | Telento Tech DWFSV2-42 | 2024

