


Arrays / some()

Se utiliza para calcular la suma de todos los elementos en un array. Puedes pasarle un array de números como argumento y devolverá la suma total de esos números. Es una forma conveniente y rápida de obtener la suma de valores numéricos en un array sin tener que realizar un bucle manualmente.


js

 Copiar código

```
// Arrays / some()
const programador = [
  {
    id: 1,
    nombre: "Juan",
    lenguaje: "Php",
  },
  {
    id: 2,
    nombre: "Jey",
    lenguaje: "Python",
  },
  {
    id: 3,
    nombre: "Bryan",
    lenguaje: "Javascript",
  },
];

// Devuelve 'true' o 'false'
const songg = programador.some(post => post.id == 1);
console.log(songg);
```

js


 Copiar código

```
// Usando 'include' y 'some'
const ind = programador.some(posta => posta.lenguaje.includes("Php"));
console.log(ind);
```

Arrays / filter() + include()

La combinación de filter() e includes() en JavaScript te permite filtrar elementos de un array según ciertos criterios y luego verificar si un valor específico está presente en el array resultante. Esto facilita la búsqueda y el filtrado de datos en un array de manera rápida y eficiente.

js

 Copiar código

```
const Empleados = [
  {
    nombre: "Dario",
    salario: "1200000",
    aux: "750000",
    sot: "web",
  },
  {
    nombre: "Steve",
    salario: "2300000",
    aux: "300000",
    sot: "web",
  },
  {
    nombre: "Laura",
    salario: "5000000",
    aux: "1200000",
    sot: "web",
  },
];


const prueba = Empleados.filter(post => post.nombre.includes('Laura'));
console.log(prueba);
```

Arrays / every()

Se utiliza para verificar si todos los elementos de un array cumplen una condición específica definida por una función de prueba. Retorna true si todos los elementos pasan la prueba, de lo


contrario, retorna false. Es útil cuando necesitas validar si todos los elementos de un array satisfacen ciertos criterios, como por ejemplo, verificar si todos los números en un array son mayores que cierto valor.

js

 Copiar código


```
const vali = Empleados.every( post => post.sot.includes("web"));
console.log(vali);
```

js

 Copiar código

```
// Ejemplo 2
const arreglo = [35, 28, 31, 25, 29];
const test = (elemento) => elemento > 20;
let res = arreglo.every(test);
console.log(res);
```

js

 Copiar código


```
// Ejemplo 2
const arreglo1 = [35, 28, 31, 25, 29];
const copia = arreglo1.every(element => element > 20);
console.log(copia);
```

Arrays / concat()

Se usa para unir dos o más arrays en uno nuevo. Este método crea un nuevo array con los elementos de los arrays originales, sin modificar los arrays originales. Es útil cuando necesitas combinar datos de múltiples fuentes sin alterar los arrays originales.

js

03.04.2024 MOD-2


 Copiar código

```
const ejem1 = ["A", "Maria", "Julian"];
const ejem2 = ["B", "Jairo", "Roberth"];
const union = ejem1.concat(ejem2);
console.log(union);
```

Arrays / include()

Verifica si un array contiene un elemento específico, retornando true o false. Es una forma rápida y sencilla de comprobar la presencia de un valor en un array.

js


 Copiar código

```
const arreg = [35, 28, 56, 45];
console.log(arreg.includes(35));
console.log(arreg.includes(20));
```

Arrays / join()

Se utiliza para combinar los elementos de un array en una cadena de texto, separando cada elemento con un separador especificado. Esto es útil cuando necesitas representar un array como una cadena legible para humanos o cuando deseas construir una URL o una consulta SQL dinámica, por ejemplo. Por defecto, el separador es una coma, pero puedes especificar cualquier carácter o cadena para separar los elementos.

js


 Copiar código -2

```
const alf = ["a", "b", "c", "d", "e"];
console.log(alf)
console.log(alf.join());
console.log(alf.join(' '));
console.log(alf.join('-'));
console.log(alf.join(', '));
```

Arrays / reduce()

Se utiliza principalmente para reducir un array a un solo valor. Permite ejecutar una función reductora en cada elemento del array, acumulando un resultado final. Esta función toma cuatro argumentos: el acumulador, el valor actual, el índice actual y el array completo. Es especialmente útil para sumar valores, encontrar el máximo o mínimo, concatenar strings, entre otras operaciones.

js


 Copiar código

```
// Usando reduce()
const numeros = [1,2,3,4,5,6,7,8,9,10];
const num = numeros.reduce((a, b) => a + b );
console.log(num);
```

Arrays / indexOf()

Se utiliza para buscar el índice de la primera ocurrencia de un elemento dentro de un array. Si el elemento no se encuentra, devuelve -1. Es una herramienta útil para verificar la existencia de un

js


 Copiar código

```
// Usando indexOf()
const frutas = ["Manzana", "Pera", "Piña"];
const valor = frutas.indexOf('Piña');
const valor1 = frutas.indexOf('Manzana');
const valor2 = frutas.indexOf('Maracuya');
const valor3 = frutas.indexOf(3);
console.log(valor, valor1, valor2, valor3);
```

Arrays / findIndex()

Se utiliza para buscar el índice de la primera ocurrencia que cumple con una condición especificada en un array. Si encuentra un elemento que satisface la condición, devuelve el índice de ese elemento. Si no encuentra ninguna coincidencia, devuelve -1. Es útil cuando se desea encontrar la posición de un elemento en función de ciertos criterios de búsqueda dentro de un array.

js


 Copiar código

```
const frutasA = ["Manzana", "Pera", "Piña"];
const valorr = frutasA.findIndex(a => a == "Pera");
console.log(valorr);
```

Arrays / shift()

Se utiliza para eliminar el primer elemento de un array y devolver ese elemento eliminado. Después de llamar a `shift()`, el array se modifica, y su longitud se reduce en uno. Todos los demás elementos en el array se desplazan hacia una posición inferior, es decir, sus índices se reducen en uno. Este método es útil cuando se desea eliminar el primer elemento de un array y utilizar su valor en alguna operación.

js


 Copiar código

```
// Usando shift()
const frutasB = ["Manzana", "Pera", "Piña"];
frutasB.shift();
console.log(frutasB);
```

Arrays / unshift()

Se utiliza para agregar uno o más elementos al principio de un array y devuelve la nueva longitud del array. Los elementos proporcionados como argumentos son agregados en el orden en que se pasan, desplazando los elementos existentes hacia posiciones de índice superior para hacer espacio para los nuevos elementos. Es útil cuando se desea agregar elementos al inicio de un array de manera eficiente.

js


 Copiar código

```
const frutasC = ["Manzana", "Pera", "Piña"];
frutasC.unshift("Zapote");
console.log(frutasC);
```

Arrays / reverse()

Se utiliza para invertir el orden de los elementos en un array. Este método modifica el array original, alterando el orden de sus elementos de tal manera que el primer elemento pasa a ser el último y viceversa. Es útil cuando se necesita cambiar el orden de los elementos de un array de manera rápida y sencilla. Por ejemplo, si se tiene un array con datos en orden ascendente y se desea mostrarlos en orden descendente, se puede utilizar `reverse()` para lograr este efecto de manera eficiente.

js


 Copiar código

```
const frutasD = ["Manzana", "Pera", "Piña"];
console.log(frutasD);
console.log(frutasD.reverse());
console.log(frutasD);
```

Arrays / toReverse()

Cambie el orden de un array sin modificar el original

js


 Copiar código

```
// Usando toReverse()
const frutasE = ["Manzana", "Pera", "Piña"];
console.log(frutasE.toReversed());
console.log(frutasE);
```


Arrays / splice()

Permite modificar un array al agregar, eliminar o reemplazar elementos en una posición específica. Se especifica la posición de inicio y la cantidad de elementos a eliminar, con la opción de agregar nuevos elementos en esa posición. Este método altera el array original y devuelve los elementos eliminados, o un array vacío si no se eliminan elementos.


js

 Copiar código

```
// Ejemplo 1
const meses = ['Jan', 'March', 'April', 'June' ];
meses.splice(1,0,'Feb');
console.log(meses);

meses.splice(4, 1,'May');
console.log(meses);
```

js

 Copiar código

```
// Ejemplo 2
const splicee = ["Nombre", "Apellido", "Correo", "Direction" ];
splicee.splice(1, 0, "Direction");
console.log(splicee);
```