Programación Imperativa

Módulo 4. JS Avanzado

Sesión 17: Avanzado con funciones

Septiembre 15 de 2021

Arrow functions

```
// forma clásica
function sumar(a, b) {
 return a + b;
}
console.log(sumar(2, 4));

// ES6 arrow function
let sumar = (a, b) => a + b;
// Si el código es una sola línea que contiene lo que se desea retornar, no hacen falta las llaves, ni el return
console.log(sumar(2, 4));


let elDoble = num => num*2;
// Si la función recibe solo un parámetro, no requiere el paréntesis

let saludo = () => 'Hola Mundo!'
// Si la función no recibe parámetros, el paréntesis es requerido.
```

Callback

Es una función que se pasa como parámetro de otra función.

Callback anónimo

```
setTimeout( function() {
        console.log('Hola Mundo!')
}, 1000)
```

Callback definido

```
let miCallback = () => console.log('Hola mundo!');
setTimeout(miCallback, 1000);
```

Sincrónico

No es necesario tener la función definida desde antes, para pasarla como callback.

Let saludoOtro = saludar3((nombre => { return nombre }, ‘Paula’}

Sesión 18 (repaso). Sincrónico.

Require vs import.

Sesión 18. Arrays y más arrays

Septiembre 16 de 2021.

Métodos de Arrays avanzados

.map(), .filter(), .reduce(), .forEach()

.map()
Recorre el array y devuelve un nuevo array modificado.

```
let numeros = [2, 4, 6];
let elDoble = numeros.map(function(num){
	return num * 2;
});
```

.filter()
Recorre el array y filtra los elementos según una condición que exista en el callback.

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
	return edad > 18;
});
```

.reduce()
Recorre el array y devuelve un único valor

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
	return acum + num;
});
```

.forEach()
Itera sobre un array, pero no retorna nada.

```
var paises = ['Argentina', 'Cuba', 'Perú'];
paises.forEach(function(pais){
	console.log(pais);
});
```

.slice()
Devuelve (extrae) una copia de una parte del array dentro de un array

(subarray)

```
let numeros = [3, 4, 5, 6, 7];
let subArray = numeros.slice(0, 3);
```

.splice()
Sirve para remover y/o agregar elementos de un array.
Recibe 3 parámetros:
- inicio: el índice del primer elemento
- cant (opcional): indica la cantidad de elementos a eliminar
- items (opcional): indica los elementos que se agregan, desde inicio.

```
let numeros = [3, 4, 5, 6, 7];
numeros.splice(0, 0, 2);
console.log(numeros); // [2,3,4,5,6,7]

numeros.splice(1, 2);
console.log(numeros); // [2,5,6,7]
```

.sort()
Sirve para ordenar los elementos de un array.
Recibe un callback como parámetro (opcional) que especifica el modo de ordenamiento. Si es omitido, el array es ordenado con el valor de string (Unicode).
Convierte a string a cada elemento.

```
let numeros = [10, 3, 4, 52, 6, 7];
numeros.sort();
console.log(numeros); // Imprime [10, 4, 52, 7, 8, 9]. Ordena como string.


function compare(a, b){
	return a-b;
}
numeros.sort(compare);
console.log(numeros); // [4, 7, 8, 9, 10, 52]
```

.find()
devuelve el valor del primer elemento de un array que cumple con una función especificada (callback).

```
let criptos = [
	{nombre: 'Bitcoin', simbolo: 'BTC'},
	{nombre: 'Ethereum', simbolo: 'ETH'},
	{nombre: 'Cardano', simbolo: 'ADA'}
];
```

```
function esBitcoin(criptos) {
	return criptos.nombre === 'Bitcoin';
}

console.log(criptos.find(esBitcoin)); // {nombre: 'Bitcoin', simbolo: 'BTC'}
```

Otra forma:

```
let res = criptos.find(e => e.nombre === 'Bitcoin');
console.log(res);
```

Sincrónico

El método reduce requiere un segundo parámetro, que es igual al valor inicial de acum. Si no se especifica, se toma el primer elemento del array.

La diferencia entre map y forEach, es que mientras map retorna un array, forEach no retorna nada.