Apuntes clase 5

Funciones que llaman a otras funciones, funciones anonimas

```
function llamoAOtra(funcion){
        funcion();
}

function meLlaman() {
        console.log("Hola");
}

llamoAOtra(meLlaman); // Notar la ausencia de parentesis en meLlaman
llamoAOtra(function(){ console.log("Hola 2") });
```

Output:

```
Hola
Hola 2
```

Funciones flecha

Mismo ejemplo que antes pero escrito con funciones flecha.

```
//    nombre       (parametros)
const llamoAOtra = (funcion) => {
        funcion();
}

const meLlaman = () => {
        console.log("Hola, flecha");
}

llamoAOtra(meLlaman); // Notar la ausencia de parentesis en meLlaman
llamoAOtra(() => { console.log("Hola, flecha 2") });

// Estas dos funciones son iguales
const sum = (a, b) => { return a + b }
const sum = (a, b) => a + b
// Si la funcion solo ocupa una linea,
// podemos omitir los parentes y
```

```
// esta retornará el valor que
// resulte de esta unica linea.
```

Output:

```
Hola, flecha
Hola, flecha 2
```

forEach, map, filter, reduce

- `forEach`: recibe una funcion como parametro y la ejecuta por cada elemento del arreglo.
- `map`: recibe una funcion como parametro y la ejecuta por cada elemento del arreglo, asignandole a ese elemento lo que retorne la funcion.
- `filter`: recibe una funcion como parametro y la ejecuta por cada elemento del arreglo. Si esta devuelve true, ese elemento permanece en el nuevo arreglo, sino lo elimina.
- `reduce`: recibe una funcion como parametro y la ejecuta por cada elemento del arreglo. Retorna un unico valor.

> Nota: ninguna de estas funciones modifica el arreglo original.

```
let arr = [1, 2, 3];

arr.forEach((e) => console.log(e));
let arrX2 = arr.map((n) => n * 2); // [2,4,6]
let filteredArr = arr.filter((n) => n >= 2); // [2,3]
let arrSum = arr.reduce((sum, n) => sum + n, 0); // 1+2+3 = 6
let arrX2X2 = arrX2.map((n) => n * 2).forEach((n) => console.log(n));
```

Output:

```
1
2
3
4
8
12
```