03. ARROW FUNCTIONS JAVASCRIPT

IES ESTACIÓ CURS 2021- 2022

Índex

1. Què és una Arrow Function en JavaScript?

- Les funcions de fletxa, o arrow functions són una nova manera de definir funcions i hi ha diferents variants en la sintaxi.
- Quan declarem una arrow functions, la paraula reservada **function** no s'usa.
- Si només es rep un paràmetre, els parèntesis poden ser omesos.
- Després dels paràmetres ha d'anar una fletxa (=>), i el contingut de la funció.

2. Funció d'un sol paràmetre.

- **Funció d'un sol paràmetre.**
  - En crear una arrow function d'un sol paràmetre no és necessari escriure els parèntesis, tampoc és necessari escriure les claus, això es pot quan la funció és d'una sola línia i retorna un valor.

```
1  //Antes
2  var saludo = function (nombre) {
3    return 'Hola ' + nombre;
4  };
5  console.log( saludo('Jonathan') ); //Imprime Hola Jonathan
6
7  //Ahora
8  let saludo = nombre => `Hola ${nombre}`;
9  console.log( saludo('Jonathan') ); //Imprime Hola Jonathan
```

3. Funció de diversos paràmetres.

- **Funció de diversos paràmetres.**
  - Quan la funció tinga més d'un paràmetre és necessari embolicar el nom d'aquests entre parèntesis.

```
1 //Antes
2 var sumar = function (a, b) {
3   return a + b;
4 };
5 console.log( sumar(10, 9) ); //Imprime 19
6
7 //Ahora
8 let sumar = (a, b) => a + b;
9 console.log( sumar(10, 9) ); //Imprime 19
```

4. Funció sense paràmetres.

- **Funció sense paràmetres.**
  - Quan la funció no reba paràmetres també són necessaris els parèntesis.

```
1  //Antes
2  var saludo = function () {
3    return 'Hola a tod@s';
4  };
5  console.log( saludo() ); //Imprime Hola a tod@s
6
7  //Ahora
8  let saludo = () => `Hola a tod@s`;
9  console.log( saludo() ); //Imprime Hola a tod@s
```

5. Funció amb cos.

- **Funció amb cos.**
  - Quan la funció té més d'una línia (o no retorna cap valor) és necessari utilitzar les claus.

```
1  //Antes
2  var fecha = new Date(),
3    hora = fecha.getHours();
4
5  var saludo = function (hr) {
6    if (hr <= 5) {
7      return 'No me jodas!!!';
8    } else if(hr >= 6 && hr <= 11) {
9      return 'Buenos días!!!';
10   } else if(hr >= 12 && hr <= 18) {
11     return 'Buenas tardes!!!';
12   } else {
13     return 'Buenas noches!!!';
14   }
15 };
16
17 console.log( saludo(hora) ); //Imprime el saludo dependiendo la hora del día
18
```

5. Funció amb cos.

```
18
19 //Ahora
20 let fecha = new Date(),
21   hora = fecha.getHours();
22
23 let saludo = (hr) => {
24   if (hr <= 5) {
25     return 'No me jodas!!!';
26   } else if(hr >= 6 && hr <= 11) {
27     return 'Buenos días!!!';
28   } else if(hr >= 12 && hr <= 18) {
29     return 'Buenas tardes!!!';
30   } else {
31     return 'Buenas noches!!!';
32   }
33 };
34
35 console.log( saludo(hora) ); //Imprime el saludo dependiendo la hora del día
36
```

5. Funció amb cos.

```
37  //Antes
38  var numeros = [1, 2, 3, 4];
39
40  numeros.forEach(function (num) {
41    console.log(num); //Imprime el número en turno
42    console.log(num * 10); //Imprime el número en turno por 10
43  });
44
45  //Ahora
46  let numeros = [1, 2, 3, 4];
47
48  numeros.forEach((num) => {
49    console.log(num); //Imprime el número en turno
50    console.log(num * 10); //Imprime el número en turno por 10
51  });
```

6. Context Lèxic de this.

- Les arrow function tenen la capacitat de capturar l'objecte **this** del context on la arrow s'executa i així utilitzar-lo dins del seu bloc de sentències.

```
1  //El problema de `this` Antes
2  function Persona(nombre) {
3    //El constructor Persona() define `this` como una instancia de él mismo
4    this.nombre = nombre;
5    this.edad = 0;
6
7    setInterval(function () {
8      //La función anónima define `this` como una instancia de ella misma
9      this.edad++;
10   }, 1000);
11 }
12
13 var jon = new Persona('Jonathan');
14 console.log(jon); //Imprime la edad en 0 por cada segundo que pasa
15
```

6. Context Lèxic de this.

```
16 //La solución al problema de `this` Antes
17 function Persona(nombre) {
18   //Se declara una variable self (algunos prefieren that) para guardar el `this` del co
19   var self = this;
20
21   self.nombre = nombre;
22   self.edad = 0;
23
24   setInterval(function () {
25     //La función anónima define su propio `this` pero el valor que aumenta es edad del
26     self.edad++;
27   }, 1000);
28 }
29
30 var jon = new Persona('Jonathan');
31 console.log(jon); //Imprime el valor de edad más uno por cada segundo que pasa
32
```

6. Context Lèxic de this.

```
33 //La solución al problema de `this` Ahora
34 function Persona(nombre) {
35   //El constructor Persona() define `this` como una instancia de él mismo
36   this.nombre = nombre;
37   this.edad = 0;
38
39   setInterval(() => {
40     //`this` hace referencia al objeto Persona()
41     this.edad++;
42   }, 1000);
43 }
44
45 const jon = new Persona('Jonathan');
46 console.log(jon); //Imprime el valor de edad más uno por cada segundo que pasa
47 console.log(jon.edad); //Imprime la edad
```