En este artículo vamos a ver qué son y cómo utilizar las **Funciones Flecha** o Arrow Functions de **JavaScript**, una nueva funcionalidad introducida con el estándar **ES6** (ECMAScript 6).

Contenidos

Qué son las Funciones Flecha

Las **Funciones Flecha** o Arrow Functions son una nueva sintaxis para definir funciones anónimas en JavaScript de un modo más conciso. Al ser una **sintaxis abreviada**, nos ahorrará bastante tiempo, además de simplificar el **ámbito de la función**. Es una de las características más utilizadas de **ES6**.

Actualmente, la mayor parte de los navegadores soportan ya las Funciones Flecha, por lo que podemos utilizarlas sin peligro alguno. Eso sí, no están soportadas por ninguna versión de Internet Explorer.

Para definir una Función Flecha utilizamos el símbolo «=>», cuya forma nos recuerda a la de una flecha doble. No es necesario escribir ni la palabra ***function*** ni las llaves de apertura y de cierre de función, siendo un caso parecido a las *Lambdas* de lenguajes como Python o C#. Las funciones flecha pueden definirse en **una sola línea**.

Sintaxis de las Funciones Flecha

Podemos **definir las funciones flecha** de muchas formas. En este apartado vamos a ver las más utilizadas, así como las **diferencias entre una Función Flecha ES6 y una función normal**.

Vamos a definir una función de ejemplo muy sencilla que sume dos números.

```
// ES5
var sumaEs5 = function(x, y) {
    return x + y;
};

// ES6
const sumaEs6 = (x, y) => { return x + y };

console.log(sumaEs6(2, 4)); // 6
```

Tal y como vemos, con al función flecha nos ahorramos la palabra ***function***, y además podemos escribir la función en **una sola línea** manteniendo las buenas prácticas recomendadas.

Sintaxis con un parámetro

En caso de utilizar **un solo parámetro**, podemos **obviar los paréntesis** que rodean al parámetro. Como ejemplo, una función que calcule el cuadrado de un número:

```
// ES5
var cuadradoEs5 = function(x) {
    return x * x;
};

// ES6
const cuadradoEs6 = (x) => { return x * x };

console.log(cuadradoEs6(3)); // 9
```

Sintaxis sin parámetros

En caso de **no utilizar ningún parámetro**, tendremos que **escribir igualmente los paréntesis**. Como ejemplo, una función que haga un *alert*:

```
// ES5
var holaEs5 = function() {
    alert('Hola');
};

// ES6
const holaEs6 = () => { alert('Hola'); };

holaEs6();
```

Sintaxis de expresión literal

Las funciones flecha también pueden devolver un objeto en su forma de **expresión literal**. Los objetos se definen entre llaves, algo que entraría en conflicto con la definición de las funciones flecha. Es por ello que para devolver un literal tendremos que **rodear a la función de paréntesis**. Es decir, que además de las llaves, tendremos que colocar un paréntesis de apertura y otro de cierre. Como ejemplo, esta función que transforma dos parámetros en un objeto:

```
// ES5
var agregarIdEs5 = function(id, nombre) {
    return {
        id: id,
        nombre: nombre
    }
};

// ES6
const agregarIdEs6 = (id, nombre) => ({ id: id, nombre: nombr

console.log(agregarIdEs6(1, "Edu")); // Object {id: 1, nombre
```

Ahora ya sabes cómo se definen las Funciones Flecha, por lo que a