Funciones lambda (o arrow functions)

Una de las funcionalidades más importantes que se añadió en ES2015 fue la posibilidad de usar las funciones lambda (o flecha). Otros lenguajes como C#, Java, etc. también las soportan. Estas expresiones ofrecen la posibilidad de crear funciones anónimas, pero con algunas ventajas.
Vamos a ver las diferencias que tiene por creando dos funciones equivalentes (una anónima y otra lambda que hacen lo mismo):

```
let sum = function(num1, num2) {
  return num1 + num2;
}
console.log(sum(12,5)); // Imprime 17

let sum = (num1, num2) => num1 + num2;
console.log(sum(12,5)); // Imprime 17
```

Cuando declaramos una función lambda, la palabra reservada **function** no se usa. Si solo se recibe un parámetro, los paréntesis pueden ser omitidos. Después de los parametros debe ir una flecha (**=>**), y el contenido de la función.

```
let square = num => num * num;
console.log(square(3)); // Imprime 9
```

Si solo hay una instrucción dentro de la función lambda, podemos omitir las llaves '{}', y **debemos omitir** la palabra reservada **return** ya que lo hace de forma implícita (devuelve el resultado de esa instrucción). Si hay mas de una instrucción, usamos las llaves y se comporta como una función normal y por tanto, si devuelve algo, debemos usar la palabra reservada return.

```
let sumInterest = (price, percentage) => {
  let interest = price * percentage / 100;
  return price + interest;
}
console.log(sumInterest(200,15)); // Imprime 230
```

La diferencia más importante entre ambos tipos de funciones es el comportamiento de la palabra reservada **this**.