Funciones flecha

Hay otra sintaxis muy simple y concisa para crear funciones, que a menudo es mejor que las Expresiones de funciones.
Se llama "funciones de flecha", porque se ve así:

```
let func = (arg1, arg2, ..., argN) => expression;
```

Esto crea una función func que acepta los parámetros arg1..argN, luego evalúa la expression del lado derecho mediante su uso y devuelve su resultado.
En otras palabras, es la versión más corta de:

```
let func = function(arg1, arg2, ..., argN) {
  return expression;
};
```

Veamos un ejemplo concreto:

```
let sum = (a, b) => a + b;

/* Esta función de flecha es una forma más corta de:

let sum = function(a, b) {
  return a + b;
};
*/

alert( sum(1, 2) ); // 3
```

Como puedes ver, (a, b) => a + b significa una función que acepta dos argumentos llamados a y b. Tras la ejecución, evalúa la expresión a + b y devuelve el resultado.

Si solo tenemos un argumento, se pueden omitir paréntesis alrededor de los parámetros, lo que lo hace aún más corto.
Por ejemplo:

```
let double = n => n * 2;
// Más o menos lo mismo que: let double = function(n) { return n * 2 }

alert( double(3) ); // 6
```

Si no hay parámetros, los paréntesis estarán vacíos; pero deben estar presentes:

```
let sayHi = () => alert("¡Hola!");

sayHi();
```

Las funciones de flecha se pueden usar de la misma manera que las expresiones de función.
Por ejemplo, para crear dinámicamente una función:

```
let age = prompt("What is your age?", 18);

let welcome = (age < 18) ?
  () => alert('¡Hola!') :
  () => alert("¡Saludos!");

welcome();
```

Las funciones de flecha pueden parecer desconocidas y poco legibles al principio, pero eso cambia rápidamente a medida que los ojos se acostumbran a la estructura.
Son muy convenientes para acciones simples de una línea, cuando somos demasiado flojos para escribir muchas palabras.

Funciones de flecha multilinea

Las funciones de flecha que estuvimos viendo eran muy simples. Toman los parámetros a la izquierda de =>, los evalúan y devuelven la expresión del lado derecho.
A veces necesitamos una función más compleja, con múltiples expresiones o sentencias. En ese caso debemos encerrarlos entre llaves. La diferencia principal es que las llaves necesitan usar un return para devolver un valor (tal como lo hacen las funciones comunes).
Como esto:

```
let sum = (a, b) => {  // la llave abre una función multilínea
  let result = a + b;
  return result; // si usamos llaves, entonces necesitamos un "return" explícito
};

alert( sum(1, 2) ); // 3
```

Resumen

Las funciones de flecha son útiles para acciones simples, especialmente las de una sola línea. Vienen en dos variantes:

1. Sin llaves: (...args) => expression – el lado derecho es una expresión: la función la evalúa y devuelve el resultado. Pueden omitirse los paréntesis si solo hay un argumento, por ejemplo n => n*2.
2. Con llaves: (...args) => { body } – las llaves nos permiten escribir varias declaraciones dentro de la función, pero necesitamos un return explícito para devolver algo.