ES6 (ECMAScript 6): Características Fundamentales

Julian F. Latorre

18 de octubre de 2024

Contenido

Introducción a ES6

- ES6 (ECMAScript 2015): Una actualización significativa de JavaScript
- Introduce numerosas mejoras y nuevas funcionalidades
- Transforma la forma de escribir código JavaScript moderno
- Mejora la legibilidad, expresividad y robustez del código

let

- Introduce alcance de bloque
- No hay elevación (hoisting)
- Temporal Dead Zone (TDZ)

```
let x = 10;
if (true) {
    let x = 20; // Nueva variable x
    console.log(x); // Imprime 20
}
console.log(x); // Imprime 10
```

const

- Para declarar constantes
- Alcance de bloque
- Debe ser inicializada
- No permite reasignación
- No implica inmutabilidad para objetos y arrays

```
const PI = 3.14159;
PI = 3; // Genera un TypeError

const persona = {nombre: "Juan"};
persona.nombre = "Pedro"; // Válido
persona = {nombre: "María"}; // Error
```

Funciones Flecha (Arrow Functions)

- Sintaxis concisa para escribir funciones
- No tienen su propio `this`
- No pueden ser usadas como constructores
- No tienen objeto `arguments`

```
// Función tradicional
function suma(a, b) {
    return a + b;
}

// Función flecha equivalente
const suma = (a, b) => a + b;
```

Ejemplos de Funciones Flecha

```
// Sin parámetros
const saludar = () => console.log("Hola");

// Un solo parámetro
const cuadrado = x => x * x;

// Múltiples parámetros
const multiplicar = (a, b) => a * b;

// Cuerpo de función con múltiples líneas
const esPar = (numero) => {
    if (numero % 2 === 0) {
        return true;
    }
    return false;
};
```

Plantillas de Cadenas (Template Strings)

- Interpolación de expresiones
- Cadenas multilínea
- Expresiones complejas

```
const nombre = "María";
console.log(`Hola, ${nombre}!`);

const mensaje = `
    Esto es una cadena
    que abarca múltiples
    líneas sin necesidad de \\n
`;
```

Ejemplos Avanzados de Plantillas de Cadenas

```
// Interpolación con expresiones
const a = 5;
const b = 10;
console.log(`La suma de ${a} y ${b} es ${a + b}`);

// Uso con funciones
const resaltar = texto => `**${texto}**`;
console.log(`Este es un ${resaltar('texto importante')
    }`);
```

Conclusión

- `let` y `const`: Mejor control del alcance y prevención de errores
- Funciones flecha: Sintaxis concisa, útil para callbacks
- Plantillas de cadenas: Creación de cadenas complejas y legibles
- Estas características son fundamentales para el JavaScript moderno
- Practiquen regularmente estos conceptos

Recursos Adicionales

- MDN Web Docs: `https://developer.mozilla.org/es/docs/Web/JavaScript`
- Exploring ES6 por Axel Rauschmayer: `https://exploringjs.com/es6/`
- JavaScript.info: `https://javascript.info/`

¡Sigan practicando y experimentando!