

Taller Práctico: ES6 Características Fundamentales

Julian F. Latorre
Bootcamp Desarrollo Web Full Stack

Introducción

Este taller práctico está diseñado para reforzar su comprensión de las características fundamentales de ES6 que hemos cubierto en nuestra sesión: **let**, **const**, funciones flecha y plantillas de cadenas. Cada ejercicio está diseñado para aplicar estos conceptos en situaciones prácticas que podrían encontrar en proyectos reales.

Instrucciones

Para cada ejercicio:

- Lean cuidadosamente el enunciado y los requisitos.
- Escriban su código.
- Prueben su código para asegurarse de que funciona como se espera.
- Si se atascan, consulten las pistas proporcionadas.

¡Buena suerte y diviértanse programando!

Ejercicio 1: Refactorización con let y const

Objetivo

Refactorizar el siguiente código para utilizar `let` y `const` apropiadamente, mejorando el alcance de las variables y la inmutabilidad donde sea posible.

Código Original

```
var nombre = "Juan";
var edad = 25;
var ciudad = "Madrid";

if (edad >= 18) {
  var mensaje = nombre + " es mayor de edad";
  console.log(mensaje);
}

for (var i = 0; i < 5; i++) {
  console.log(i);
}

console.log(i); // i es accesible fuera del bucle
```

Tu Solución

Escribe tu código refactorizado aquí:

```
// Tu código aquí
```

Pistas

- Usa `const` para variables que no se reasignan.
- Usa `let` para variables que se reasignan o tienen un alcance limitado.
- Presta atención al alcance de las variables en el bloque `if` y el bucle `for`.

Ejercicio 2: Funciones Flecha y This

Objetivo

Refactorizar el siguiente código para utilizar funciones flecha, prestando especial atención al comportamiento de `this`.

Código Original

```
var persona = {
  nombre: "Ana",
  amigos: ["Carlos", "David", "Elena"],
  imprimirAmigos: function() {
    var that = this;
    this.amigos.forEach(function(amigo) {
      console.log(that.nombre + " es amigo de " + amigo);
    });
  }
};

persona.imprimirAmigos();
```

Tu Solución

Escribe tu código refactorizado aquí:

```
// Tu código aquí
```

Pistas

- Las funciones flecha no tienen su propio `this`, lo heredan del contexto circundante.
- Puedes eliminar la variable `that` al usar una función flecha.
- Considera si `imprimirAmigos` también puede ser una función flecha.

Ejercicio 3: Plantillas de Cadenas Avanzadas

Objetivo

Crear una función que genere un resumen de un producto utilizando plantillas de cadenas. La función debe tomar un objeto producto como argumento y devolver una cadena formateada.

Código Base

```
const producto = {
  nombre: "Laptop XPS 13",
  marca: "Dell",
  precio: 1299.99,
  especificaciones: {
    procesador: "Intel Core i7",
    ram: "16GB",
    almacenamiento: "512GB SSD"
  }
};

function generarResumenProducto(producto) {
  // Tu código aquí
}

console.log(generarResumenProducto(producto));
```

Tu Solución

Completa la función `generarResumenProducto`:

```
// Tu código aquí
```

Resultado Esperado

El resultado debe ser similar a:

Producto: Laptop XPS 13 (Dell)

Precio: \$1,299.99

Especificaciones:

- Procesador: Intel Core i7
- RAM: 16GB
- Almacenamiento: 512GB SSD

Pistas

- Utiliza plantillas de cadenas con interpolación de expresiones.
- Puedes anidar plantillas de cadenas para las especificaciones.
- Considera usar el método `toFixed()` para formatear el precio.

Ejercicio 4: Proyecto Mini: Gestor de Tareas

Objetivo

Crear un pequeño gestor de tareas utilizando las características de ES6 que hemos aprendido. El gestor debe permitir añadir tareas, marcarlas como completadas y listar todas las tareas.

Estructura Base

```
// Estructura base del gestor de tareas
const gestorTareas = {
  tareas: [],

  agregarTarea(descripcion) {
    // Tu código aquí
  },

  marcarComoCompletada(indice) {
    // Tu código aquí
  },

  listarTareas() {
    // Tu código aquí
  }
};

// Código de prueba
gestorTareas.agregarTarea("Comprar leche");
gestorTareas.agregarTarea("Hacer ejercicio");
gestorTareas.agregarTarea("Estudiar ES6");

gestorTareas.listarTareas();

gestorTareas.marcarComoCompletada(1);

gestorTareas.listarTareas();
```

Tu Solución

Completa las funciones del `gestorTareas`:

```
// Tu código aquí
```

Requisitos

- Usa `const` para el objeto `gestorTareas`.
- Utiliza funciones flecha para los métodos del objeto.
- Usa `let` para variables que puedan cambiar.
- Emplea plantillas de cadenas para formatear la salida de `listarTareas`.

-
- Utiliza el método `map` de arrays junto con una función flecha para `listarTareas`.

Resultado Esperado

El resultado de ejecutar el código de prueba debería ser similar a:

Tareas:

1. ☐ Comprar leche
2. ☐ Hacer ejercicio
3. ☐ Estudiar ES6

Tareas:

1. ☐ Comprar leche
2. ☒ Hacer ejercicio
3. ☐ Estudiar ES6

Pistas

- Cada tarea puede ser un objeto con propiedades como `descripcion` y `completada`.
- Usa el método `push` para añadir tareas al array.
- En `marcarComoCompletada`, asegúrate de manejar casos donde el índice no sea válido.
- Para `listarTareas`, considera usar `forEach` o `map` junto con plantillas de cadenas.

Conclusión y Reflexión

¡Felicidades por completar este taller práctico sobre las características fundamentales de ES6! A través de estos ejercicios, has tenido la oportunidad de aplicar `let`, `const`, funciones flecha y plantillas de cadenas en contextos prácticos y realistas.





Reflexión

Tómate un momento para reflexionar sobre las siguientes preguntas:

- ¿Cómo el uso de `let` y `const` mejora la claridad y seguridad de tu código comparado con `var`?
- ¿En qué situaciones encuentras que las funciones flecha son particularmente útiles?
- ¿Cómo las plantillas de cadenas mejoran la legibilidad de tu código al trabajar con strings complejos?
- ¿Qué desafíos encontraste al refactorizar código existente para usar estas nuevas características de ES6?

Próximos Pasos

Para continuar tu aprendizaje y práctica con ES6:

-  Refactoriza un proyecto existente para incorporar estas características de ES6.
-  Explora otras características de ES6 como destructuring, spread operator, y módulos.
-  Crea un proyecto personal que utilice extensivamente estas y otras características de ES6.
-  Comparte tu código refactorizado con compañeros y discute las mejoras en legibilidad y funcionalidad.

💡 Recuerda: La práctica constante es la clave para dominar estas nuevas características y convertirte en un desarrollador JavaScript más eficiente y efectivo.