Módulo 5
Preguntas y Respuestas

por Karina Hidalgo

¿Qué es TypeScript y para qué se utiliza?

TypeScript es un lenguaje de programación desarrollado por Microsoft que extiende JavaScript añadiendo tipado estático. Se utiliza para detectar errores en tiempo de desarrollo, mejorar la legibilidad del código y facilitar el mantenimiento de proyectos, especialmente en aplicaciones grandes.

¿Cuáles son las principales diferencias entre TypeScript y JavaScript?

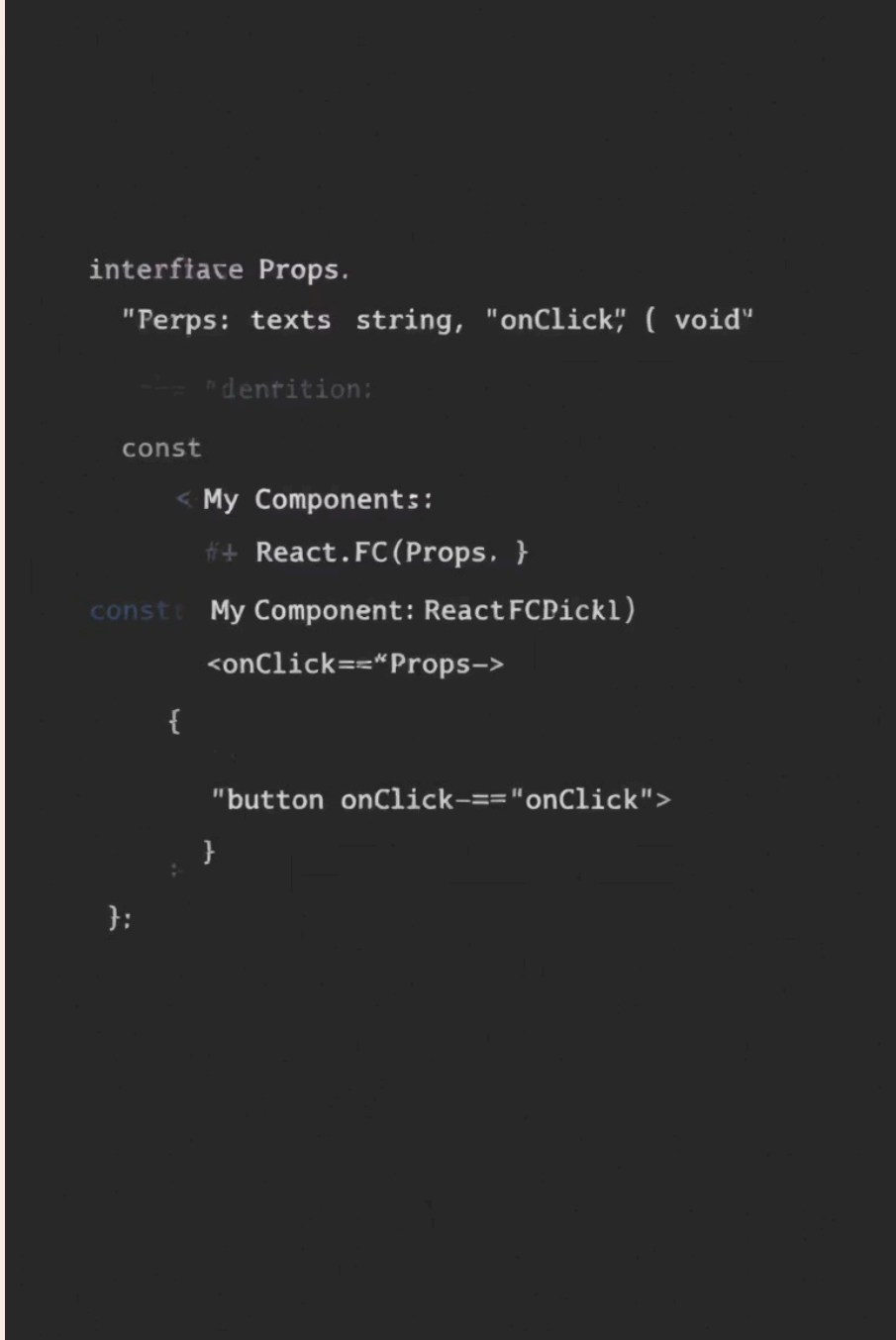
La siguiente tabla muestra algunas diferencias clave entre TypeScript y JavaScript:

Característica	JavaScript	TypeScript
Tipado	Dinámico	Estático (opcional)
Verificación de errores	En ejecución	En compilación
Autocompletado	Limitado	Avanzado
Soporte en IDEs	Básico	Excelente
Compilación	Nativo	Compila a JavaScript

¿Por qué es útil TypeScript en React?

Porque permite definir con precisión los tipos de props y state, reduciendo errores en el manejo de datos. Mejora el autocompletado en editores como VS Code, y ayuda a construir componentes más robustos y seguros.

El sistema de tipos de TypeScript es un conjunto de reglas que define qué tipo de datos puede usar cada variable, función o propiedad. Este sistema detecta errores antes de ejecutar el código, lo que mejora la calidad del software.



Ventajas de usar TypeScript en ReactJS

Previene errores antes de ejecutar el código

Mejora la productividad gracias al autocompletado

Hace el código más legible y mantenible

Facilita el trabajo en equipo

Documenta el código implícitamente con tipos

Ejercicio Práctico: Definiendo Tipos e Inferencia

A continuación, se presenta una función que utiliza tipado explícito e inferencia en TypeScript:

```
function sumar(a: number, b: number): number {
 return a + b;
}

let resultado = sumar(5, 10); // TypeScript infiere que 'resultado' es de tipo number
```

Definición de Interfaces y Clases en TypeScript

Ejemplo de una interfaz **Doctor** y una clase **DoctorImpl** que implementa esa interfaz:

```
interface Doctor {
 nombre: string;
 especialidad: string;
 obtenerInformacion(): string;
 actualizarEspecialidad(nuevaEspecialidad: string): void;
}

class DoctorImpl implements Doctor {
 constructor(public nombre: string, public especialidad: string) {}

 obtenerInformacion(): string {
   return `${this.nombre} - ${this.especialidad}`;
 }

 actualizarEspecialidad(nuevaEspecialidad: string): void {
   this.especialidad = nuevaEspecialidad;
 }
}
```

TypeScript y ReactJS: Implementación Básica en un Componente

A continuación, se muestra un componente funcional en React usando TypeScript con props tipadas:

```
import React from 'react';

interface DoctorProps {
 nombre: string;
 especialidad: string;
}

const TarjetaDoctor: React.FC <DoctorProps> = ({ nombre, especialidad }) => {

 return (
```

{nombre}

```
Especialidad: {especialidad}

 ); }; export default TarjetaDoctor;
```