Módulo 5
Desarrollo de aplicaciones Front-End con React

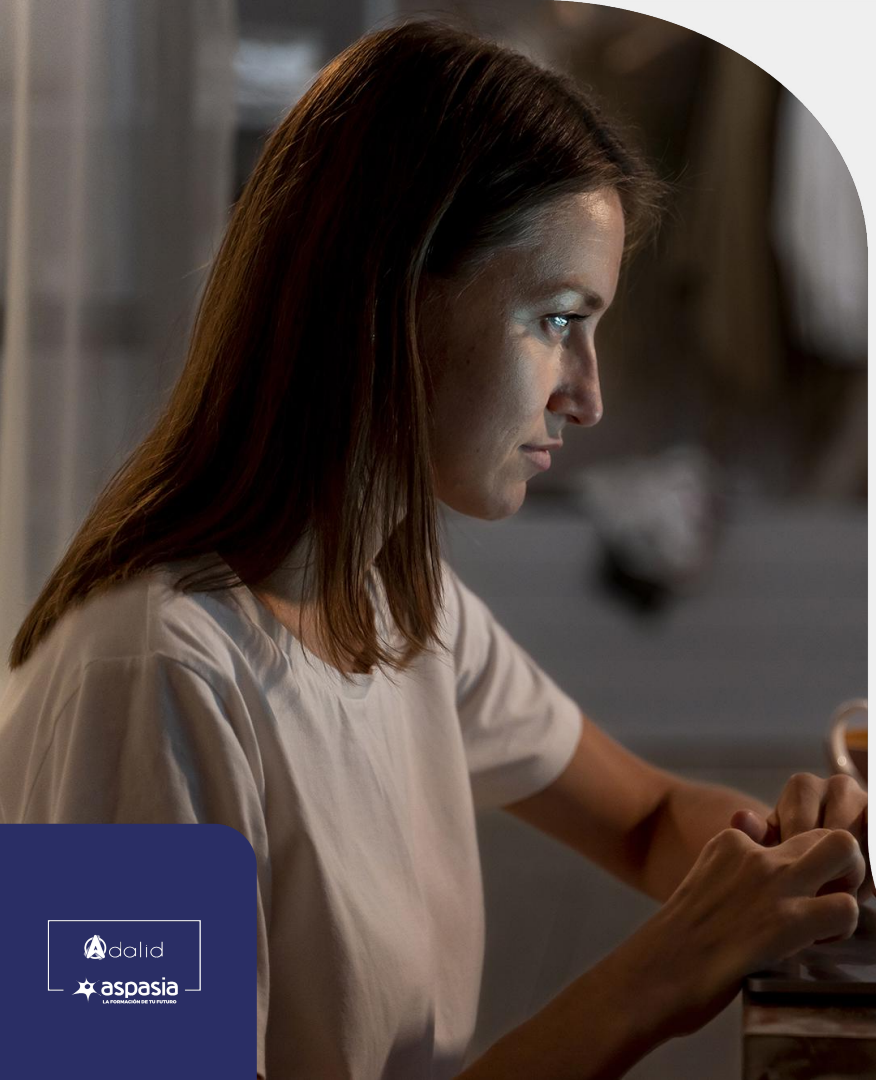
Introducción a TypeScript

Módulo 5

AE 2.1

OBJETIVOS

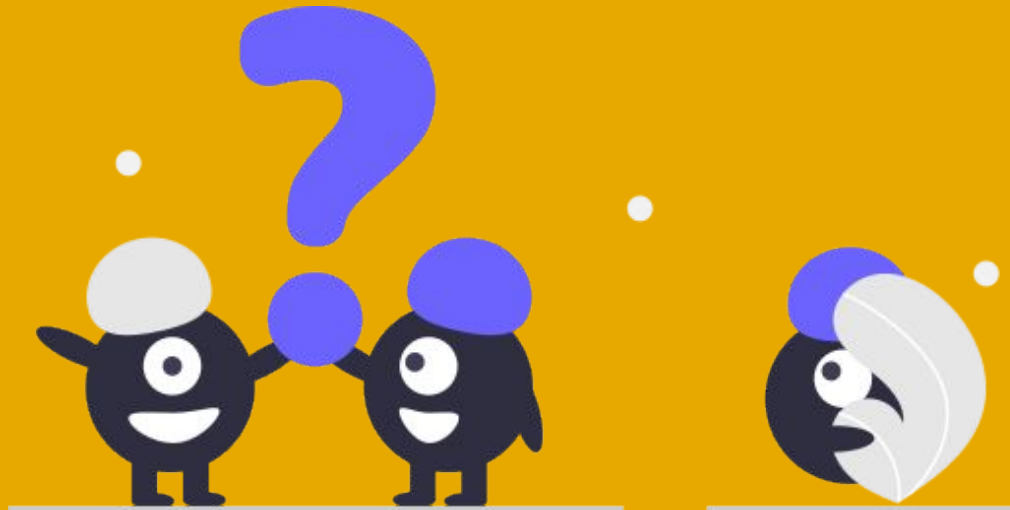
Entender qué es TypeScript, cómo usarlo en proyectos React y su ventaja frente a JavaScript. Aprender a definir y componer tipos, usar interfaces y clases, e integrar TypeScript con frameworks como Next.js y Webpack.

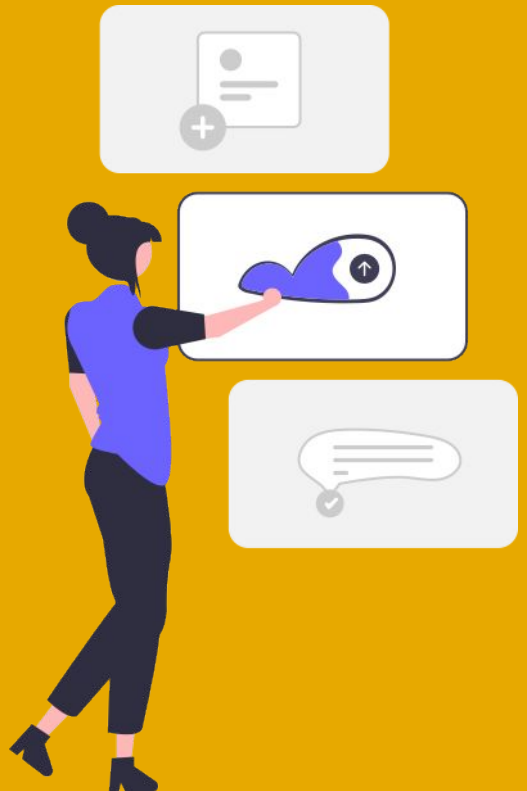


¿QUÉ VAMOS A VER?

- Introducción a TypeScript.
- Qué es TypeScript.
- Para qué se utiliza TypeScript.
- TypeScript en React.js.
- TypeScript vs. Javascript.
- TypeScript y Webpack.
- Definiendo tipos.
- Tipos por inferencia.
- Componiendo tipos.
- Sistema de tipo estructural.
- Interfaces y Clases.
- Algunos Frameworks que soportan TypeScript (Next.js, Gatsby.js).

¿Que es TypeScript?





TypeScript

Qué es TypeScript

TypeScript es un lenguaje de programación desarrollado por Microsoft que **extiende las funcionalidades de JavaScript** al incluir **tipado estático**. Esto significa que puedes declarar tipos para variables, funciones y objetos, lo que ayuda a **detectar errores en tiempo de compilación** en lugar de en tiempo de ejecución.

Qué es TypeScript

Características clave:

- **Superconjunto de JavaScript:** Todo código JavaScript válido es también válido en TypeScript.
- **Tipado estático:** Los tipos se definen explícitamente o se infieren, lo que ayuda a prevenir errores.
- **Herramientas avanzadas:** Ofrece mejor autocompletado, refactorización y navegación del código en editores como VS Code.
- **Compatibilidad:** TypeScript se compila a JavaScript, por lo que puede usarse en cualquier proyecto que soporte JavaScript.

Qué es TypeScript

Ejemplo básico:

```
let nombre: string = 'Juan'; // Variable con tipo explícito
let edad = 30; // Tipo inferido como 'number'

function saludar(nombre: string): string {
  return `Hola, ${nombre}`;
}
```

Para qué se utiliza TypeScript

- **Aplicaciones web complejas:** Reduce la complejidad al tipar estados, props y funciones en frameworks como React.
- **Desarrollo a gran escala:** Facilita el trabajo en equipo al proporcionar documentación implícita y prevenir conflictos entre módulos.
- **Refactorización:** Cambiar el nombre de variables, funciones o estructuras se vuelve más seguro.
- **Integración con librerías:** Mejora el autocompletado y la navegación al usar librerías con definiciones de tipos.

TypeScript en React.js

TypeScript en React, permite definir tipos para **props**, **estados**, **eventos** y **funciones**, lo que reduce errores y mejora la experiencia del desarrollador.

```tsx
import React from 'react';

interface ButtonProps {
  label: string;
  onClick: () => void;
}

const Button: React.FC<ButtonProps> = ({ label, onClick }) => {
  return <button onClick={onClick}>{label}</button>;
};

export default Button;
```

TypeScript vs. JavaScript

Diferencias clave:

- **Tipado:**
  - **JavaScript:** Dinámico, el tipo se determina en tiempo de ejecución.
  - **TypeScript:** Estático, el tipo se verifica en tiempo de compilación.
- **Errores:**
  - **JavaScript:** Los errores no se detectan hasta que el código se ejecuta.
  - **TypeScript:** Detecta errores antes de ejecutar el código.

TypeScript vs. JavaScript

Compatibilidad:

- Ambos lenguajes son interoperables, pero TypeScript se compila a JavaScript para ejecutarse en navegadores.

```
// JavaScript
let nombre = 'Juan';
nombre = 42; // No arroja error, pero puede causar problemas

// TypeScript
let nombre: string = 'Juan';
nombre = 42; // Error: El tipo 'number' no se puede asignar a 'string'
```

TypeScript y Webpack

TypeScript se integra con Webpack para **transcompilar código TypeScript a JavaScript**, lo que permite aprovechar las características modernas de TypeScript en proyectos con un flujo de trabajo moderno.

TypeScript y Webpack

Pasos para configurar TypeScript con Webpack:

1 - Instalación de dependencias

```
npm install typescript ts-loader
webpack webpack-cli --save-dev
```

2 - Configura **webpack.config.js** para usar **ts-loader**:

```
module.exports = {
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.ts$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
    extensions: ['.ts', '.js'],
  },
};
```

2 - Configura TypeScript, crea un archivo **tsconfig.json**:

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "strict": true
  }
}
```

Definiendo Tipos

TypeScript permite definir tipos explícitos para variables, funciones y objetos, lo que asegura que solo se usen valores válidos.

```
let nombre: string = 'Ana';
let edad: number = 25;

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

Tipos por Inferencia

TypeScript puede inferir el tipo de una variable según su valor inicial, lo que simplifica el código sin perder seguridad.

```
let mensaje = 'Hola mundo'; // Inferido como string
let cantidad = 10; // Inferido como number
```

Componiendo Tipos

Puedes combinar tipos usando operadores como union (|) y intersection (&).

```
type ID = string | number;

function imprimirID(id: ID): void {
  console.log(`El ID es: ${id}`);
}
```

Sistema de Tipo Estructural

El sistema de tipo estructural compara la forma de los objetos en lugar de sus nombres.

```
interface Persona {
  nombre: string;
  edad: number;
}

const maria = { nombre: 'Maria', edad: 30, ciudad: 'Madrid' };
const otraPersona: Persona = maria; // Aceptado porque tiene las propiedades requeridas
```

Interfaces y Clases

TypeScript permite modelar objetos usando interfaces y clases.

Ejemplo de Interfaces:

```
interface Producto {
  id: number;
  nombre: string;
  precio: number;
}

const producto: Producto = { id: 1, nombre: 'Laptop', precio: 1000 };
```

Ejemplo de Clases:

```
class Vehiculo {
  constructor(public marca: string, public modelo: string) {}

  describir(): string {
    return `${this.marca} ${this.modelo}`;
  }
}

const auto = new Vehiculo('Toyota', 'Corolla');
console.log(auto.describir());
```

Algunos Frameworks que Soportan TypeScript

Frameworks populares:

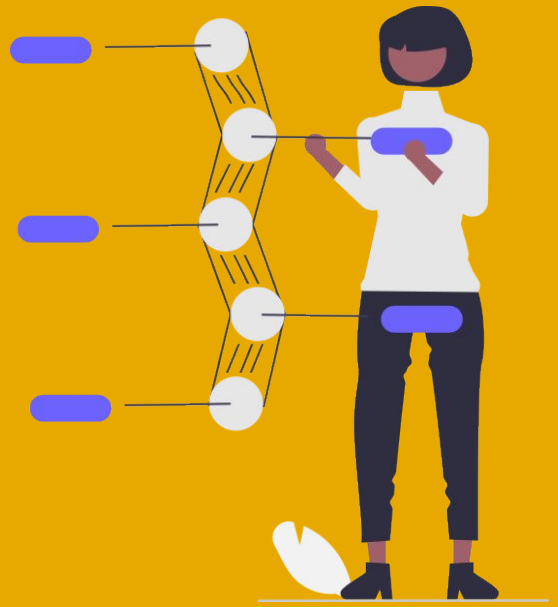
- **Next.js:** Permite construir aplicaciones de React con renderizado en servidor y soporte para TypeScript.
- **Gatsby.js:** Ideal para sitios estáticos con TypeScript.
- **Angular:** Construido con TypeScript desde el inicio, ofrece soporte nativo para sus características.

Acceso al Repositorio

En el siguiente enlace podrás acceder a un repositorio relacionado con la temática propuesta.

https://github.com/adalid-cl/ESPECIALIZACION_FRONTEND_M5_AE2

Resumen de lo aprendido 🧑‍🎓



Adalid aspasia

Resumen de lo aprendido

- TypeScript mejora JavaScript añadiendo tipado estático, lo que reduce errores y facilita el desarrollo en proyectos complejos.
- En React, permite definir tipos para props, estados y funciones, mejorando la productividad y escalabilidad del código.
- Frameworks modernos como Next.js y Gatsby.js integran TypeScript para aplicaciones más robustas y mantenibles.
- Aprendiste a usar conceptos clave como definición de tipos, inferencia, interfaces y clases, además de integrar TypeScript con herramientas como Webpack.

GRACIAS POR TU ATENCIÓN

Nos vemos en la próxima clase

Chile
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO