

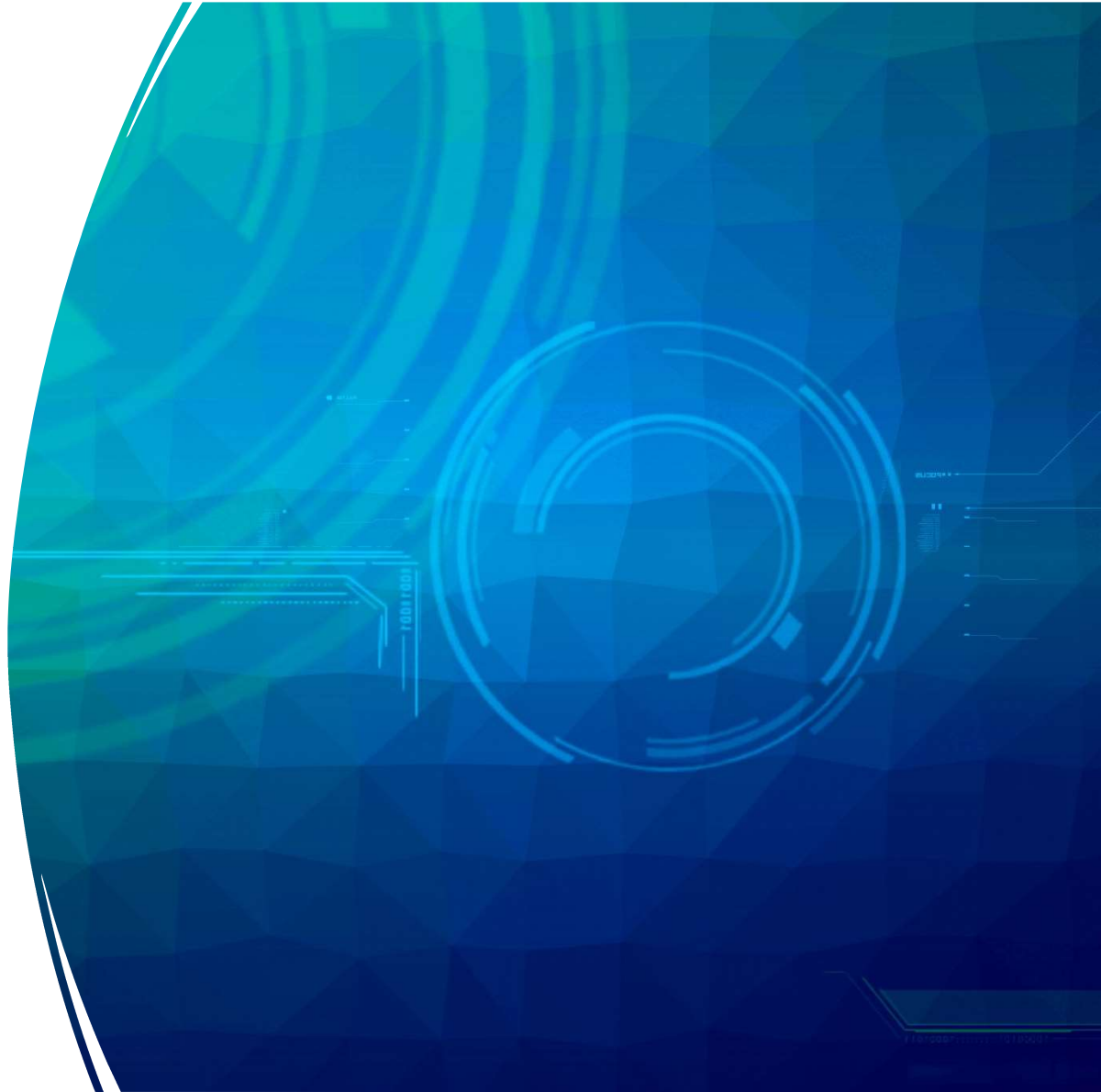


# JAVASCRIPT

# ¿Qué es JavaScript?

---

- JavaScript es un lenguaje de programación de alto nivel que se ejecuta en el lado del cliente (navegador) y permite interactuar con los elementos de una página web. Fue creado originalmente por Brendan Eich en 1995 y desde entonces ha evolucionado significativamente.



# Características principales de JavaScript:

---



**INTERACTIVIDAD EN TIEMPO REAL:**  
JAVASCRIPT PERMITE QUE LAS PÁGINAS WEB RESPONDAN A LAS ACCIONES DEL USUARIO EN TIEMPO REAL, SIN NECESIDAD DE RECARGAR LA PÁGINA COMPLETA. ESTO SE LOGRA A TRAVÉS DE LA MANIPULACIÓN DEL DOM (MODELO DE OBJETOS DEL DOCUMENTO).



**VERSATILIDAD:** JAVASCRIPT ES UN LENGUAJE MUY VERSÁTIL QUE SE PUEDE UTILIZAR TANTO EN EL LADO DEL CLIENTE COMO EN EL LADO DEL SERVIDOR. EN EL LADO DEL CLIENTE, SE EJECUTA EN EL NAVEGADOR WEB Y SE UTILIZA PARA CREAR EFECTOS VISUALES, VALIDAR FORMULARIOS, INTERACTUAR CON APIS, ENTRE OTROS. EN EL LADO DEL SERVIDOR, SE UTILIZA NODE.JS PARA CREAR APLICACIONES WEB Y SERVICIOS.



**FACILIDAD DE APRENDIZAJE:**  
JAVASCRIPT ES RELATIVAMENTE FÁCIL DE APRENDER PARA AQUELLOS QUE YA TIENEN CONOCIMIENTOS DE PROGRAMACIÓN. TIENE UNA SINTAXIS SIMILAR A OTROS LENGUAJES COMO C O JAVA, LO QUE FACILITA SU ADOPCIÓN.



**AMPLIA COMUNIDAD Y RECURSOS:**  
JAVASCRIPT CUENTA CON UNA COMUNIDAD ACTIVA DE DESARROLLADORES EN TODO EL MUNDO, LO QUE SIGNIFICA QUE HAY UNA GRAN CANTIDAD DE RECURSOS, BIBLIOTECAS Y FRAMEWORKS DISPONIBLES PARA FACILITAR EL DESARROLLO DE APLICACIONES.

```
// Definimos dos variables
var numero1 = 5;
var numero2 = 10;

// Sumamos los números
var suma = numero1 + numero2;

// Mostramos el resultado por consola
console.log(suma);
```

## Ejemplo

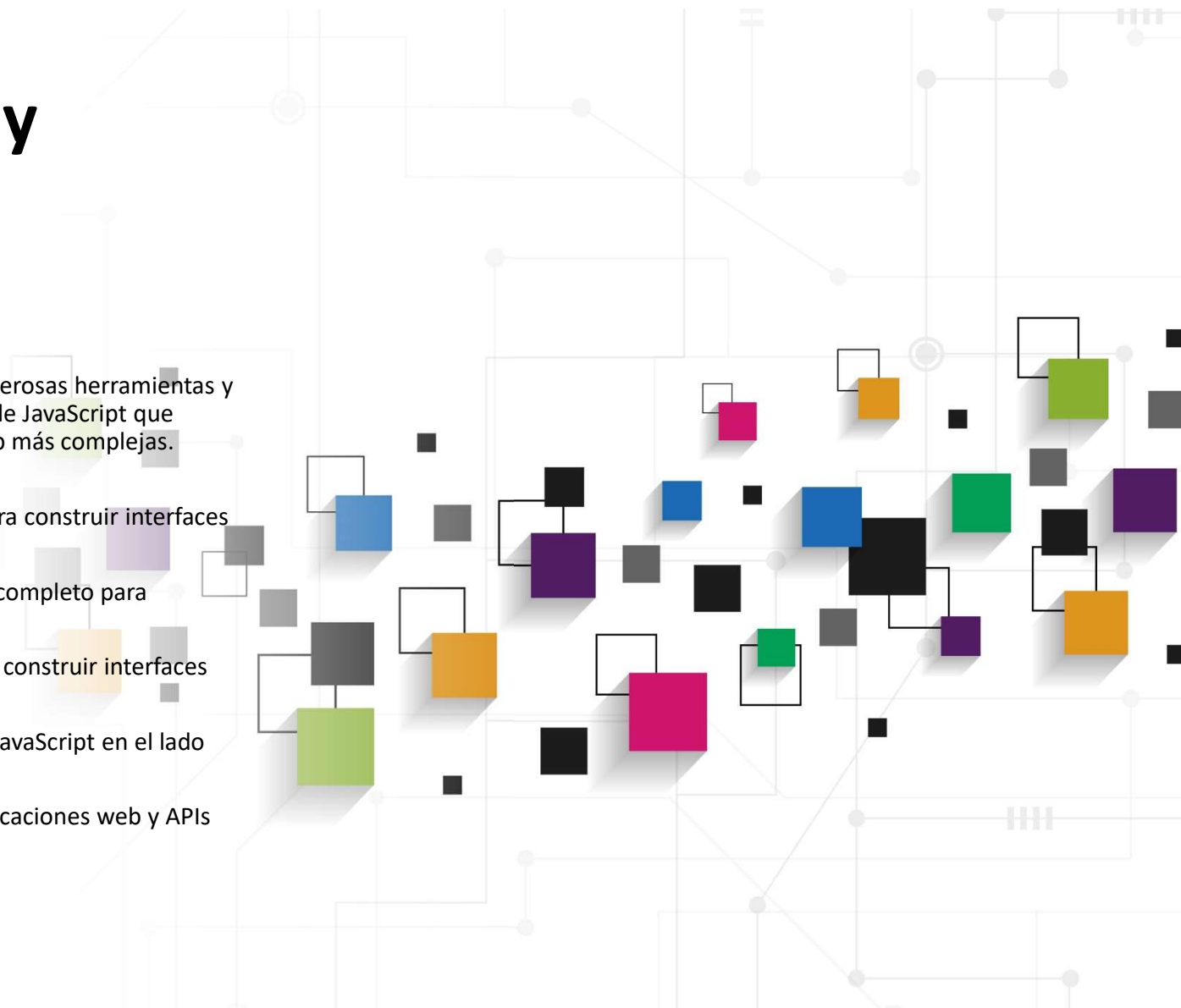
- Aquí tienes un ejemplo básico de código JavaScript para sumar dos números:



# Herramientas y frameworks populares:

Además del lenguaje básico, existen numerosas herramientas y frameworks populares en el ecosistema de JavaScript que facilitan el desarrollo de aplicaciones web más complejas. Algunos de los más utilizados son:

- **React:** una biblioteca de JavaScript para construir interfaces de usuario.
- **Angular:** un framework de JavaScript completo para construir aplicaciones web.
- **Vue.js:** un framework progresivo para construir interfaces de usuario.
- **Node.js:** un entorno de ejecución de JavaScript en el lado del servidor.
- **Express:** un framework para crear aplicaciones web y APIs en Node.js.



# JavaScript:

- JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos.
- Es un lenguaje de programación dinámico, lo que significa que no se requiere especificar el tipo de datos de una variable al declararla.
- JavaScript se ejecuta directamente en el navegador web y es compatible con todos los principales navegadores.
- Es un lenguaje flexible y permite un desarrollo rápido debido a su sintaxis sencilla y concisa.
- JavaScript tiene una amplia gama de bibliotecas y frameworks disponibles, como React, Angular y Vue.js, que facilitan el desarrollo de aplicaciones web.

# TypeScript

- TypeScript es un superconjunto de JavaScript, lo que significa que todo código JavaScript válido es también código TypeScript válido.
- A diferencia de JavaScript, TypeScript es un lenguaje de programación de tipado estático. Esto significa que se deben declarar los tipos de datos de las variables.
- TypeScript permite añadir anotaciones de tipo a las variables, parámetros de función y valores de retorno, lo que facilita el desarrollo y la detección de errores antes de la ejecución del código.
- TypeScript requiere un proceso de compilación para generar código JavaScript que pueda ser ejecutado en los navegadores.
- Añade características adicionales a JavaScript, como interfaces, clases, herencia, módulos, entre otros. Esto ayuda a mejorar la estructura y organización del código, especialmente en proyectos más grandes.
- TypeScript está diseñado para proyectos escalables y se utiliza ampliamente en aplicaciones de gran envergadura.

# Ventajas de TypeScript sobre JavaScript:



**Detección de errores temprana:** El tipado estático de TypeScript permite detectar errores de tipo durante la etapa de desarrollo, lo que ayuda a evitar errores comunes antes de ejecutar el código.



**Mantenibilidad y escalabilidad:** TypeScript facilita el mantenimiento y la evolución de proyectos a largo plazo, especialmente en aplicaciones complejas, gracias a su soporte para clases, interfaces y otras características de programación orientada a objetos.

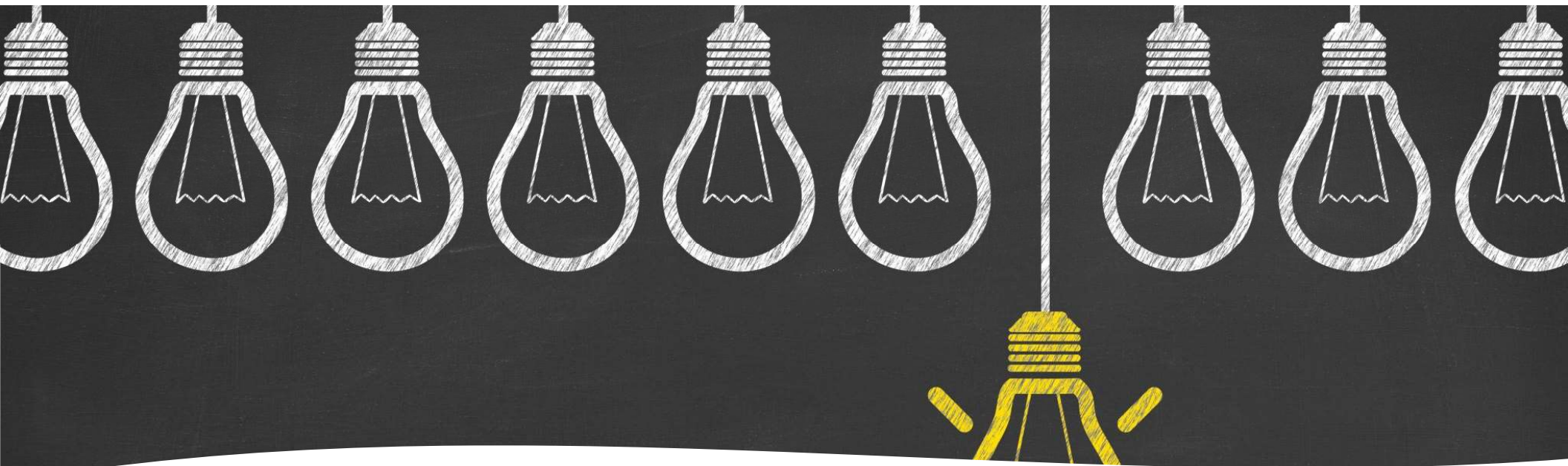


**Mejor herramienta de desarrollo:** TypeScript proporciona una experiencia de desarrollo más rica, con autocompletado, sugerencias de errores y refactorización de código más avanzada en comparación con JavaScript puro.



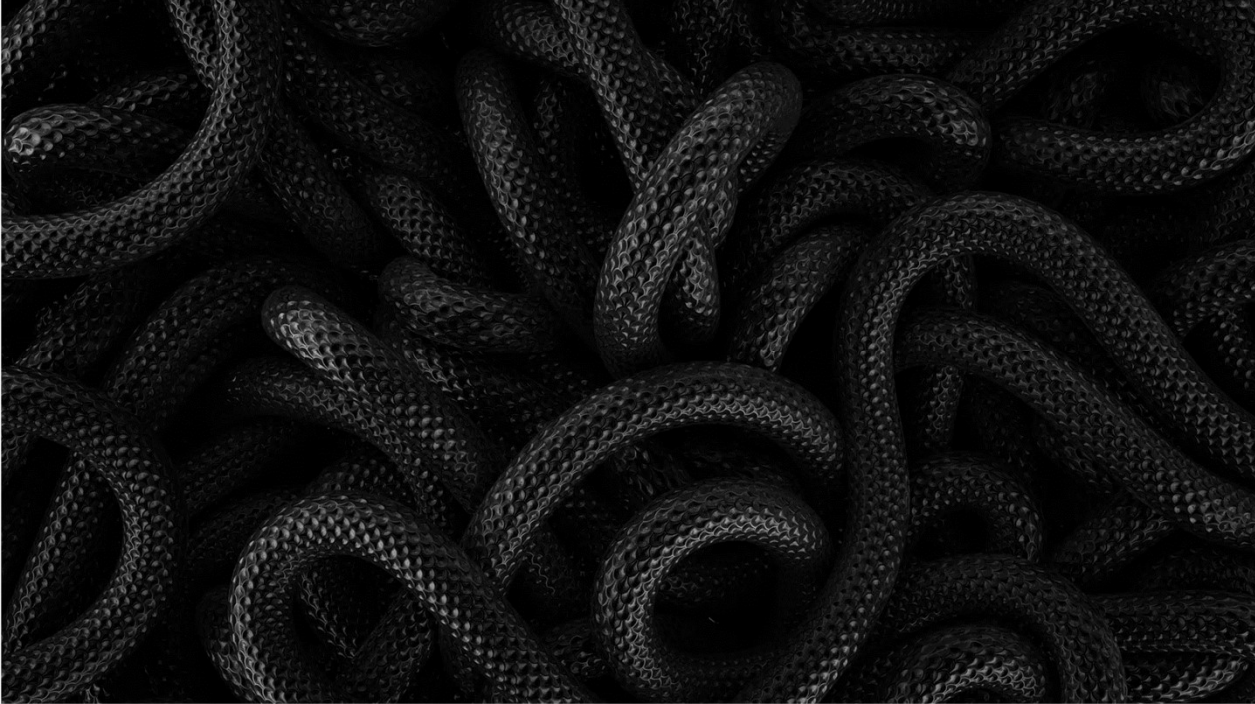
**Compatibilidad con JavaScript:** Dado que TypeScript es un superconjunto de JavaScript, se puede utilizar en proyectos existentes, ya que todo el código JavaScript válido también es válido en TypeScript.





## En resumen

- JavaScript es un lenguaje flexible y ampliamente utilizado en el desarrollo web, mientras que TypeScript agrega características de tipado estático y una serie de beneficios adicionales para proyectos más grandes y complejos. La elección entre JavaScript y TypeScript depende de los requisitos del proyecto y las preferencias del equipo de desarrollo.

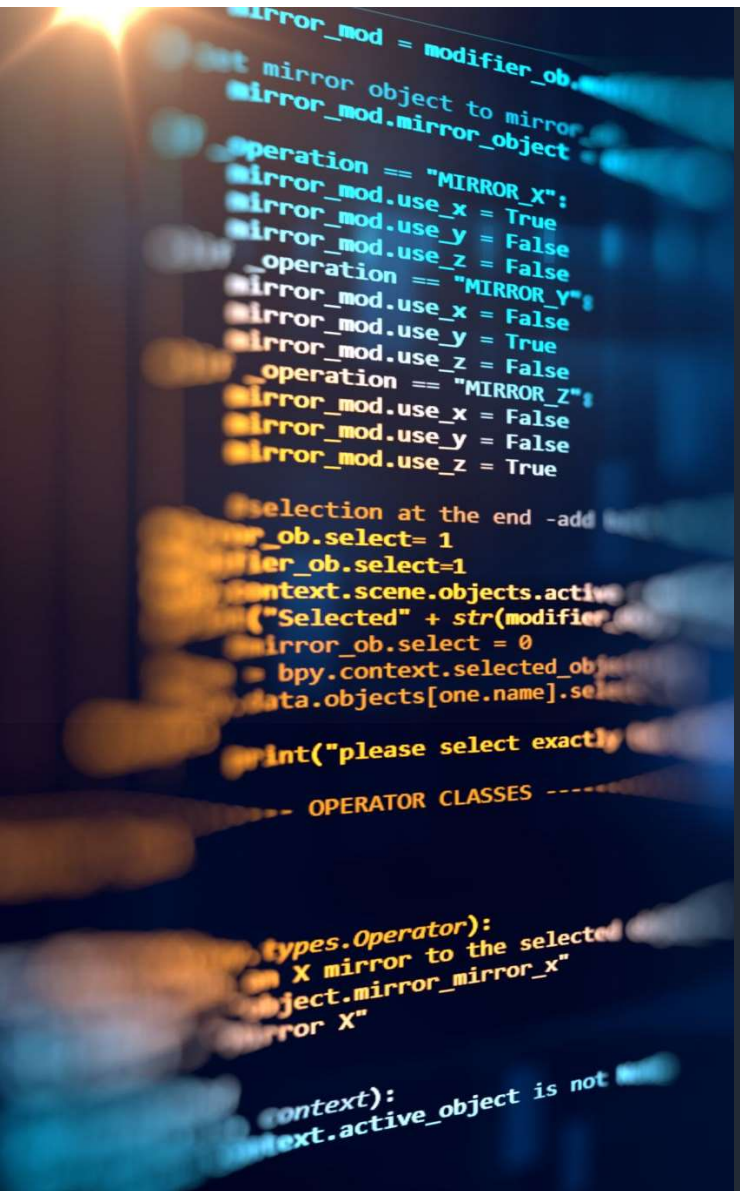


NODE JS



- Node.js es un entorno de tiempo de ejecución (runtime) de JavaScript basado en el motor V8 de Chrome. A diferencia de JavaScript en el navegador, que se ejecuta en el lado del cliente, Node.js permite ejecutar JavaScript en el lado del servidor.





Node.js se ha vuelto muy popular en el desarrollo web, ya que ofrece varias ventajas:

1. **JavaScript en el lado del servidor:** Con Node.js, puedes utilizar JavaScript tanto en el lado del cliente como en el lado del servidor, lo que permite tener un lenguaje de programación unificado en todo el stack de desarrollo.
2. **Event-driven y basado en el modelo de I/O sin bloqueo:** Node.js utiliza un modelo de programación basado en eventos y en I/O sin bloqueo (non-blocking I/O), lo que significa que puede manejar múltiples solicitudes simultáneamente sin bloquear el hilo principal de ejecución. Esto lo hace muy eficiente en términos de rendimiento y escalabilidad, especialmente para aplicaciones web en tiempo real o con muchas solicitudes concurrentes.
3. **Amplio ecosistema de módulos y paquetes:** Node.js cuenta con un gestor de paquetes llamado npm (Node Package Manager), que permite instalar y gestionar fácilmente una amplia variedad de módulos y paquetes de terceros. Existe un vasto ecosistema de módulos disponibles en el repositorio de npm, lo que facilita el desarrollo de aplicaciones con funcionalidades adicionales.
4. **Desarrollo rápido y productivo:** Node.js proporciona una experiencia de desarrollo rápida y productiva debido a su naturaleza liviana y su capacidad para manejar múltiples tareas en paralelo. Además, gracias a la gran cantidad de módulos disponibles, es posible reutilizar y compartir código de manera sencilla.
5. **Escalabilidad y rendimiento:** Node.js es conocido por su escalabilidad y rendimiento. Gracias a su modelo sin bloqueo y su capacidad para manejar múltiples solicitudes de manera concurrente, es especialmente adecuado para aplicaciones web que necesitan manejar una gran cantidad de usuarios concurrentes y ofrecer respuestas en tiempo real.



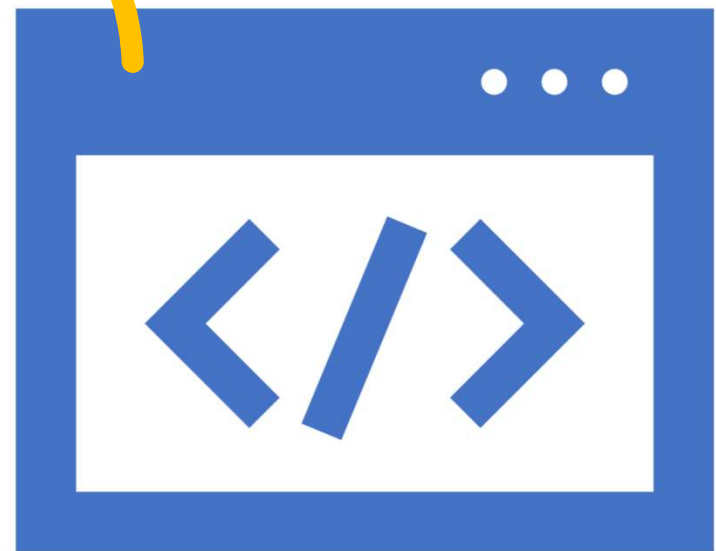
ANGULAR

---



# Concepto



- Angular es un framework de desarrollo de aplicaciones web de código abierto, basado en TypeScript, creado por Google. Proporciona un conjunto de herramientas y características que facilitan la creación de aplicaciones web robustas, escalables y de alto rendimiento.



## A continuación, te presento algunas características principales de Angular:

- **Componentes y Directivas:** Angular utiliza una arquitectura basada en componentes, donde los componentes son bloques de construcción fundamentales de la aplicación. Cada componente representa una parte específica de la interfaz de usuario y tiene su propio código, plantilla y estilos asociados. Las directivas se utilizan para agregar comportamientos adicionales a los componentes.
- **Binding de datos:** Angular ofrece un poderoso sistema de binding de datos que permite mantener sincronizados los datos y la interfaz de usuario. Puedes enlazar propiedades y eventos entre el código TypeScript y las plantillas HTML, lo que facilita la manipulación y actualización de datos de forma automática.
- **Inyección de dependencias:** Angular proporciona un mecanismo de inyección de dependencias que permite gestionar las dependencias de los componentes de manera eficiente. Esto facilita la reutilización de código, mejora la modularidad y simplifica las pruebas unitarias.
- **Enrutamiento:** Angular cuenta con un enrutador incorporado que permite la navegación entre diferentes componentes y vistas dentro de una aplicación. Puedes definir rutas y asociar componentes a esas rutas, lo que facilita la construcción de aplicaciones de varias páginas.



- 
- **Pruebas unitarias y E2E:** Angular incluye herramientas y utilidades para facilitar las pruebas unitarias y las pruebas end-to-end (E2E) de tus aplicaciones. Esto ayuda a garantizar la calidad del código y la funcionalidad de la aplicación.
  - **Gestión del estado:** Angular proporciona un sistema de gestión del estado llamado "NgRx" que se basa en el patrón Redux. Con NgRx, puedes gestionar de manera eficiente el estado de la aplicación, facilitando la manipulación y actualización de datos en diferentes componentes.
  - **Amplia comunidad y ecosistema:** Angular cuenta con una gran comunidad de desarrolladores y una amplia variedad de recursos, bibliotecas y herramientas disponibles. Además, Google proporciona soporte y actualizaciones regulares para el framework, lo que garantiza su mantenimiento y evolución continua.
- 



# Arquitectura de angular

1. **Componentes:** Los componentes son unidades funcionales y visuales de la aplicación. Representan partes específicas de la interfaz de usuario y encapsulan su lógica, plantilla y estilos asociados. Cada componente tiene su propio ciclo de vida y puede comunicarse con otros componentes a través de eventos, propiedades y servicios.
2. **Plantillas HTML:** Las plantillas HTML definen la estructura y presentación visual de los componentes. Utilizan una sintaxis especial que permite enlazar datos y eventos del componente.
3. **Servicios:** Los servicios en Angular se utilizan para compartir lógica y datos entre componentes. Proporcionan funcionalidades comunes y se pueden inyectar en los componentes que los necesitan. Los servicios pueden realizar llamadas a APIs, gestionar el estado de la aplicación, manejar la autenticación, entre otras tareas.
4. **Inyección de dependencias:** Angular utiliza el patrón de inyección de dependencias para gestionar las dependencias de los componentes y servicios. Esto permite una mayor modularidad y facilita la reutilización de código. El framework se encarga de crear instancias de los servicios y proporcionarlos automáticamente a los componentes que los necesitan.

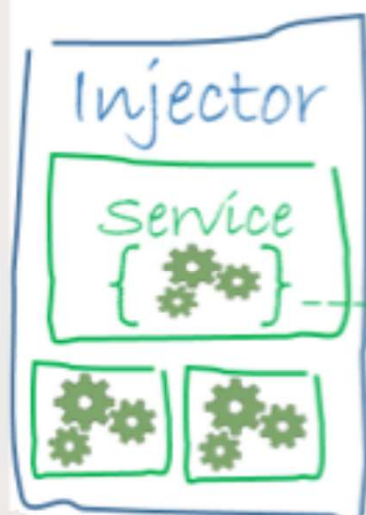


# Arquitectura de angular

1. **Módulos:** Los módulos en Angular se utilizan para organizar la aplicación en unidades funcionales y lógicas. Un módulo agrupa componentes, servicios y otros recursos relacionados. Además, se utiliza para configurar las dependencias y la configuración global de la aplicación.
2. **Enrutamiento:** El enrutamiento en Angular permite la navegación entre diferentes vistas y componentes dentro de la aplicación. Se configuran rutas que corresponden a diferentes URLs y se asocian a componentes específicos. El enrutador de Angular se encarga de cargar y mostrar los componentes adecuados según la URL actual.
3. **Gestión del estado:** Para aplicaciones más grandes y complejas, Angular recomienda el uso de patrones de gestión del estado, como NgRx (basado en el patrón Redux). NgRx proporciona una manera predecible y centralizada de manejar el estado de la aplicación, lo que facilita la manipulación y actualización de datos en diferentes componentes.

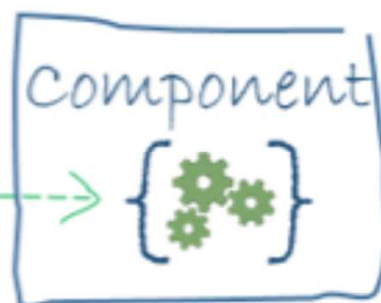
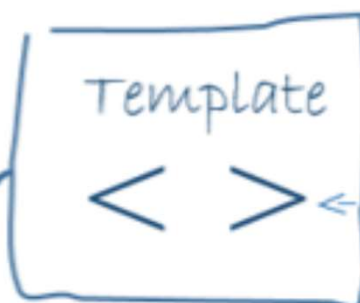


Module Component { }	Module Service { }
Module value 3.1415	Module Fn $\lambda$

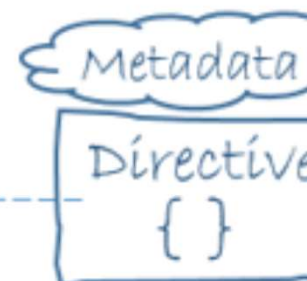


Property Binding

Metadata



Event Binding



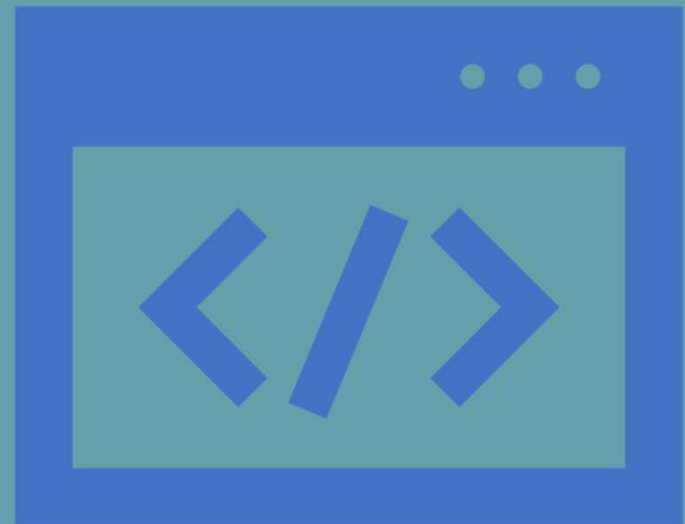
TypeORM





# Algunas características y beneficios de TypeORM son:

1. **Soporte multiplataforma:** TypeORM es compatible con una variedad de bases de datos relacionales, como MySQL, PostgreSQL, SQLite, Microsoft SQL Server y Oracle. Esto permite a los desarrolladores elegir la base de datos que mejor se adapte a sus necesidades.
2. **Mapeo objeto-relacional:** TypeORM facilita la comunicación entre la base de datos y los objetos de la aplicación mediante el mapeo de tablas y columnas de la base de datos a clases y propiedades de TypeScript. Esto permite trabajar con las entidades de la base de datos utilizando objetos familiares en lugar de escribir consultas SQL.
3. **Creación y migración de esquemas:** TypeORM proporciona herramientas para crear y modificar esquemas de base de datos de forma programática. También admite migraciones de esquemas, lo que facilita la actualización y mantenimiento de la estructura de la base de datos a medida que la aplicación evoluciona.





**Consultas y relaciones:** TypeORM ofrece una sintaxis de consulta potente y expresiva para recuperar y manipular datos en la base de datos. También admite relaciones entre entidades, lo que facilita la navegación y el manejo de relaciones complejas en la base de datos.



**Soporte para transacciones y caching:** TypeORM permite trabajar con transacciones en la base de datos, lo que garantiza la integridad y la consistencia de los datos. También admite caching de consultas y resultados, lo que puede mejorar el rendimiento de la aplicación al reducir la carga en la base de datos.



**Integración con Express y otros frameworks:** TypeORM se integra bien con el framework web Express.js, lo que permite construir aplicaciones web de forma sencilla. También ofrece soporte para otros frameworks y librerías populares de Node.js.



Database

- MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) muy popular y ampliamente utilizado. Es de código abierto y está basado en el modelo de datos relacional, lo que significa que almacena datos en tablas estructuradas con filas y columnas.





**Escalabilidad y rendimiento:** MySQL está diseñado para manejar grandes volúmenes de datos y es altamente escalable. Puede manejar múltiples conexiones y consultas concurrentes, lo que lo hace adecuado para aplicaciones con alto rendimiento y carga de trabajo.

**Lenguaje SQL:** MySQL utiliza el lenguaje SQL (Structured Query Language) como su lenguaje de consulta principal. SQL proporciona una sintaxis estándar para interactuar con la base de datos, lo que permite realizar operaciones como inserción, actualización, eliminación y consulta de datos.

**Soporte multiplataforma:** MySQL es compatible con una amplia gama de plataformas, incluyendo Windows, macOS y varias distribuciones de Linux. Esto facilita la portabilidad de las aplicaciones que utilizan MySQL como base de datos.

**Alta disponibilidad y replicación:** MySQL ofrece opciones para alta disponibilidad y replicación de datos. Mediante la configuración de réplicas, puedes crear copias de tus datos y distribuir la carga entre varios servidores, lo que mejora el rendimiento y proporciona redundancia en caso de fallos.

**Seguridad:** MySQL ofrece varias características de seguridad para proteger tus datos. Esto incluye autenticación basada en contraseñas, cifrado de datos en tránsito y en reposo, y control de acceso granular a nivel de usuario y privilegios.

**Integración con otros lenguajes y tecnologías:** MySQL es compatible con una amplia gama de lenguajes de programación y tecnologías. Existen controladores (drivers) disponibles para muchos lenguajes, como PHP, Python, Java, Node.js, entre otros, lo que facilita la integración con diferentes aplicaciones y sistemas.

**Herramientas de administración:** MySQL ofrece varias herramientas de administración, como MySQL Workbench y línea de comandos, que permiten administrar y monitorear la base de datos, realizar copias de seguridad, optimizar el rendimiento y realizar tareas de mantenimiento.



Gracias