



TS

APRENDE **TYPESCRIPT** GUÍA PRÁCTICA

A C A D E M I A X

Contenido

1	Introducción	4
1.1	Bienvenida	4
1.1.1	Libro vivo	5
1.1.2	Alcance	5
1.2	Prerequisitos	5
1.3	¿Cómo evitar bloqueos?	6
2	Primeros pasos	7
2.1	Visión general	7
2.1.1	¿Qué es y por qué debes aprenderlo?	7
2.1.2	¿En dónde se utiliza?	7
2.1.3	¿Qué trabajos puedes conseguir?	8
2.1.4	¿Cuánto puedes ganar?	8
2.1.5	¿Cuales son las preguntas más comunes?	9
2.2	Historia, evolución, y versiones	9
2.3	Características y ventajas	10
2.4	Diferencias con otros lenguajes de programación	11
2.5	Configuración	11
2.5.1	IDE	11
2.5.2	Entorno	12
2.6	Hola Mundo	14
3	Gramática	15
3.1	Sintaxis	15
3.2	Comentarios	15
4	Tipos y variables	16
4.1	Variables	16
4.2	Listas	18

Contenido

4.3	Objetos	20
4.4	Constantes	22
5	Operadores	23
5.1	Operadores de aritméticos	23
5.2	Operadores comparativos	24
5.3	Operadores lógicos	24
6	Condicionales	25
6.1	Condición única	25
6.2	Múltiples condiciones	26
7	Bucles	28
7.1	Bucle for	28
7.2	Bucle while	29
8	Funciones	30
9	Programación orientada a objetos (POO)	33
9.1	Paradigma	33
9.2	Clases	33
10	Módulos	34
11	Siguientes pasos	36
11.1	Herramientas	36
11.2	Recursos	36
11.3	¿Que viene después?	37
11.4	Preguntas de entrevista	37

1 Introducción

1.1 Bienvenida

Bienvenid@ a esta guía de Academia X en donde aprenderás TypeScript práctico.

Hola, mi nombre es Xavier Reyes Ochoa y soy el autor de esta guía. He trabajado como consultor en proyectos para Nintendo, Google, entre otros proyectos top-tier, trabajé como líder de un equipo de desarrollo por 3 años, y soy Ingeniero Ex-Amazon. En mis redes me conocen como Programador X y comparto videos semanalmente en YouTube desde diversas locaciones del mundo con el objetivo de guiar y motivar a mis estudiantes mientras trabajo haciendo lo que más me gusta: la programación.

En esta guía vas a aprender estos temas:

- Gramática
- Tipos y variables
- Operadores
- Condicionales
- Bucles
- Funciones
- Programación orientada a objetos (POO)
- Módulos

La motivación de esta guía es darte todo el conocimiento técnico que necesitas para elevar la calidad de tus proyectos y al mismo tiempo puedas perseguir metas más grandes, ya sea utilizar esta tecnología para tus pasatiempos creativos, aumentar tus oportunidades laborales, o si tienes el espíritu emprendedor, incluso crear tu propio negocio en línea. Confío en que esta guía te dará los recursos que necesitas para que tengas éxito en este campo.

Empecemos!

1 INTRODUCCIÓN

1.1.1 Libro vivo

Esta publicación fue planeada, editada, y revisada manualmente por Xavier Reyes Ochoa. La fundación del contenido fue generada parcialmente por inteligencia artificial usando ChatGPT (Mar 14 Version) de OpenAI. Puedes ver más detalles en <https://openai.com/>

Esto es a lo que llamo un trabajo **VIVO**, esto quiere decir que será actualizado en el tiempo a medida que existan cambios en la tecnología. La versión actual es 1.0.0 editada el 23 de marzo de 2023. Si tienes correcciones importantes, puedes escribirnos a nuestra sección de contacto en <https://www.academia-x.com>

1.1.2 Alcance

El objetivo de esta guía es llenar el vacío que existe sobre esta tecnología en Español siguiendo el siguiente enfoque:

- Se revizan los temas con un enfoque práctico incluyendo ejemplos y retos.
- Se evita incluir material de relleno ya que no es eficiente.
- Se evita entrar en detalle en temas simples o avanzados no-prácticos.

Si deseas profundizar en algún tema, te dejo varios recursos populares y avanzados en la lección de recursos como el estándar de esta tecnología (que puede ser difícil de leer si recién estás empezando).

1.2 Prerequisitos

Antes de aprender TypeScript, necesitas lo siguiente:

1. Un computador: cualquier computador moderno tiene las capacidades de correr este lenguaje. Te recomiendo un monitor de escritorio o laptop ya que dispositivos móviles o ipads no son cómodos para programar.

1 INTRODUCCIÓN

2. Sistema operativo: conocimiento básico de cómo utilizar el sistema operativo en el que se ejecutará TypeScript (por ejemplo, Windows, MacOS, Linux). Te recomiendo tener el sistema operativo actualizado.
3. Conocimiento básico de la línea de comandos: se utiliza para ejecutar programas en TypeScript.
4. Un editor de texto: lo necesitas para escribir código de TypeScript. Los editores de texto más populares son Visual Studio Code, Sublime Text, Atom y Notepad ++.
5. Un navegador web y conexión al internet: es útil para investigar más sobre esta tecnología cuando tengas dudas. Los navegadores más populares son Google Chrome, Mozilla Firefox, Safari y Microsoft Edge. Se recomienda tener el navegador actualizado.

Si ya tienes estos requisitos, estarás en una buena posición para comenzar a aprender TypeScript y profundizar en sus características y aplicaciones.

Si todavía no tienes conocimiento sobre algunos de estos temas, te recomiendo buscar tutoriales básicos en blogs a través de Google, ver videos en YouTube, o hacer preguntas a ChatGPT. Alternativamente, puedes empezar ya e investigar en línea a medida que encuentres bloqueos enteniendo los conceptos en esta guía.

1.3 ¿Cómo evitar bloqueos?

Para sacarle el mayor provecho a esta guía:

1. No solo leas esta guía. La práctica es esencial en este campo. Practica todos los días y no pases de lección hasta que un concepto esté 100% claro.
2. No tienes que memorizarlo todo, solo tienes que saber donde están los temas para buscarlos rápidamente cuando tengas dudas.
3. Cuando tengas preguntas usa [Google](#), [StackOverflow](#), y [ChatGPT](#)
4. Acepta que en esta carrera, mucho de tu tiempo lo vas utilizar investigando e innovando, no solo escribiendo código.

5. No tienes que aprender inglés ahora pero considera aprenderlo en un futuro porque los recursos más actualizados están en inglés y también te dará mejores oportunidades laborales.
6. Si pierdas la motivación, recuerda tus objetivos. Ninguna carrera es fácil pero ya tienes los recursos para llegar muy lejos. Te deseo lo mejor en este campo!

2 Primeros pasos

2.1 Visión general

2.1.1 ¿Qué es y por qué debes aprenderlo?

Typescript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript. Es un superconjunto de JavaScript, lo que significa que todo el código JavaScript es válido en Typescript. Typescript agrega características como tipos estáticos, clases, interfaces, decoradores y mucho más.

Debes aprender Typescript porque te permite escribir código JavaScript de forma más segura y mantenible. Los tipos estáticos te ayudan a detectar errores en tiempo de compilación, lo que significa que los errores se detectan antes de que el código se ejecute. Esto significa que el código es menos propenso a errores y más fácil de mantener. Además, Typescript es compatible con la mayoría de los frameworks de JavaScript, lo que significa que puedes usarlo para desarrollar aplicaciones web modernas.

2.1.2 ¿En dónde se utiliza?

Typescript se utiliza para desarrollar aplicaciones web y de escritorio, así como aplicaciones móviles. Está diseñado para mejorar la productividad de los desarrolladores al proporcionarles una sintaxis y un conjunto de herramientas que les permiten

2 PRIMEROS PASOS

escribir código de forma más clara y estructurada. Typescript también ofrece una forma de compilar el código a JavaScript, lo que permite a los desarrolladores escribir código en un lenguaje de alto nivel y luego compilarlo a un lenguaje de bajo nivel que puede ser ejecutado en los navegadores web.

2.1.3 ¿Qué trabajos puedes conseguir?

- Desarrollador de aplicaciones web
- Desarrollador de aplicaciones móviles
- Desarrollador de aplicaciones de escritorio
- Desarrollador de juegos
- Desarrollador de aplicaciones de realidad virtual
- Desarrollador de aplicaciones de Internet de las cosas (IoT)
- Desarrollador de aplicaciones de blockchain
- Desarrollador de aplicaciones de inteligencia artificial
- Desarrollador de aplicaciones de aprendizaje automático
- Desarrollador de aplicaciones de análisis de datos

2.1.4 ¿Cuánto puedes ganar?

El salario que puedes ganar usando Typescript depende de varios factores, como tu experiencia, el lugar donde trabajes y el tipo de trabajo que estés realizando. Según Glassdoor, los salarios promedio para desarrolladores de Typescript en los Estados Unidos son de \$91,945 por año.

Es importante tener en cuenta que estos son solo promedios y que el salario real que puedes ganar puede ser mayor o menor, dependiendo de los factores mencionados anteriormente. Además, siempre es una buena idea investigar y hacer preguntas sobre los salarios y las condiciones laborales antes de aceptar un trabajo.

2.1.5 ¿Cuales son las preguntas más comunes?

- ¿Qué es Typescript?
- ¿Por qué usar Typescript?
- ¿Cómo se instala Typescript?
- ¿Cómo se usa Typescript?

Al finalizar este recurso, tendrás las habilidades necesarias para responder o encontrar las respuestas a estas preguntas.

2.2 Historia, evolución, y versiones

Typescript es un lenguaje de programación de código abierto desarrollado por Microsoft para facilitar el desarrollo de aplicaciones web escalables. Fue creado por Anders Hejlsberg, el mismo creador de C#, y se lanzó por primera vez en octubre de 2012.

Typescript es un superconjunto de JavaScript que añade características como tipos estáticos, clases, interfaces, decoradores, etc. Estas características permiten una mejor organización y mantenimiento del código, lo que resulta en una mejor escalabilidad y rendimiento de la aplicación.

Desde su lanzamiento, Typescript ha experimentado una rápida adopción por parte de la comunidad de desarrolladores. Esto se debe a sus características útiles y a la facilidad con la que se puede integrar con herramientas como Visual Studio Code, Angular, React, etc.

Actualmente, Typescript está en su versión 4.0. Esta versión incluye mejoras en la sintaxis, mejoras en la inferencia de tipos, mejoras en la interoperabilidad con JavaScript, mejoras en la seguridad de tipos, entre otras.

Typescript se ha convertido en uno de los lenguajes de programación más populares para el desarrollo de aplicaciones web modernas. Esto se debe a sus características

útiles y a la facilidad con la que se puede integrar con herramientas como Visual Studio Code, Angular, React, etc.

2.3 Características y ventajas

Typescript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript. Ofrece una variedad de características y ventajas que lo hacen una excelente opción para desarrolladores de todos los niveles.

Características

- **Tipado estático:** Typescript ofrece una forma de tipado estático para JavaScript, lo que significa que los errores de tipo se detectan en tiempo de compilación. Esto permite a los desarrolladores detectar errores antes de que se produzcan en tiempo de ejecución.
- **Soporte para ES6:** Typescript admite la mayoría de las características de ES6, lo que significa que los desarrolladores pueden usar la sintaxis moderna de JavaScript sin preocuparse por la compatibilidad con los navegadores.
- **Soporte para herramientas de desarrollo:** Typescript es compatible con una variedad de herramientas de desarrollo, como Visual Studio Code, Atom y Sublime Text. Esto permite a los desarrolladores aprovechar al máximo las herramientas de desarrollo modernas.

Ventajas

- **Mejora la productividad:** Typescript permite a los desarrolladores escribir código más rápido y con menos errores. Esto aumenta la productividad y reduce el tiempo de desarrollo.
- **Mejora la legibilidad:** Typescript mejora la legibilidad del código al permitir a los desarrolladores usar nombres descriptivos para variables, funciones y clases. Esto hace que el código sea más fácil de entender y mantener.

- Mejora la escalabilidad: Typescript permite a los desarrolladores escribir código escalable que se puede reutilizar en proyectos futuros. Esto reduce el tiempo de desarrollo y ahorra dinero.

2.4 Diferencias con otros lenguajes de programación

Typescript es un superconjunto de JavaScript que añade tipado estático y algunas características de los lenguajes de programación modernos. Esto significa que los desarrolladores pueden escribir código con una sintaxis más clara y estructurada, lo que facilita la lectura y el mantenimiento del código.

Además, Typescript proporciona una mejor seguridad de tipo, lo que significa que los errores de tipo se detectan en tiempo de compilación, lo que reduce el tiempo de desarrollo y los errores en producción.

Otra diferencia importante es que Typescript es un lenguaje de programación compilado, mientras que la mayoría de los lenguajes de programación modernos son lenguajes de programación interpretados. Esto significa que el código Typescript se compila a código JavaScript antes de ser ejecutado, lo que permite una mayor velocidad de ejecución.

Finalmente, Typescript admite la programación orientada a objetos, lo que significa que los desarrolladores pueden crear clases, interfaces y herencia, lo que facilita la creación de aplicaciones más complejas. Esto es algo que no se admite en JavaScript.

2.5 Configuración

2.5.1 IDE

Los archivos de TypeScript son archivos de texto. Puedes editarlos con **editores de texto** como Notepad en Windows o Notes en MacOS pero es recomendado utilizar

2 PRIMEROS PASOS

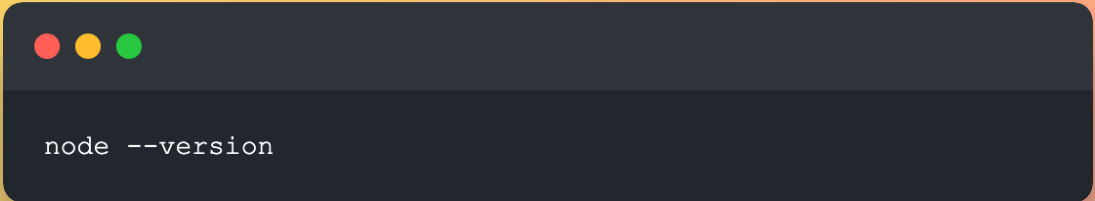
un **IDE** (Integrated Development Environment) que es una aplicación de edición de código más avanzado que le da colores a tu código para que sea más fácil de leer y tengas funciones de autocompletado, entre otras. Algunos IDEs populares son [Brackets](#), [Atom](#), [Sublime Text](#), [Vim](#), y [Visual Studio Code](#).

El editor recomendado para practicar el código que vamos a ver es Visual Studio Code (o VSCode) que puedes bajar desde <https://code.visualstudio.com/>

2.5.2 Entorno

Para usar TypeScript en tu computador, necesitas instalar Node.js y el compilador de TypeScript.

1. Instalar Node.js: Descargar e instalar Node.js desde el [sitio oficial](#).
2. Verificar la instalación: Para verificar que la instalación de Node.js se haya realizado correctamente, abre una ventana de línea de comandos y ejecuta el siguiente comando:

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal displays the command `node --version` in a light-colored monospace font.

```
node --version
```

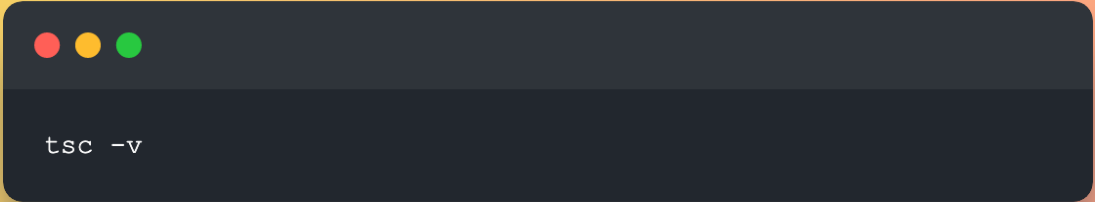
3. Instalar el compilador de TypeScript: Instalar el compilador de TypeScript usando el gestor de paquetes npm de Node.js. Abre una ventana de línea de comandos y ejecuta el siguiente comando:

2 PRIMEROS PASOS



```
npm install -g typescript
```

4. Verificar la instalación: Para verificar que la instalación de TypeScript se haya realizado correctamente, abre una ventana de línea de comandos y ejecuta el siguiente comando:



```
tsc -v
```

Esto debería mostrar la versión de TypeScript instalada.

5. Correr archivos de TypeScript: Para compilar un archivo de TypeScript, abre una ventana de línea de comandos y ejecuta el siguiente comando:



```
tsc nombre_archivo.ts
```

Esto generará un archivo JavaScript con el mismo nombre que el archivo de TypeScript. Para ejecutar el archivo JavaScript, abre una ventana de línea de comandos y ejecuta el siguiente comando:



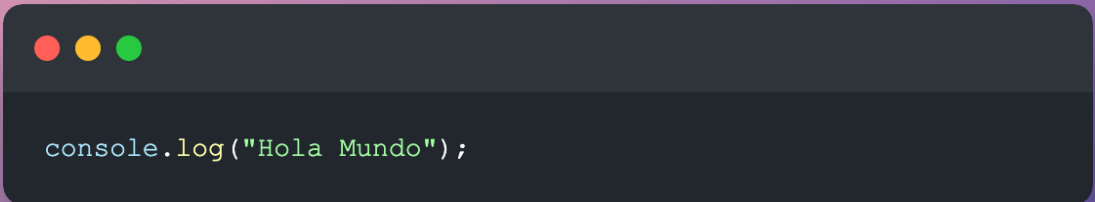
```
node nombre_archivo.js
```

2.6 Hola Mundo

“Hola Mundo” es un ejemplo clásico que se utiliza para mostrar el funcionamiento básico de un lenguaje de programación.

Ejemplo:

En este ejemplo, se imprime el texto “Hola Mundo” en la consola.



```
console.log("Hola Mundo");
```

También podrías modificar este código con tu nombre. Si es “Juan”, debería imprimir en la consola “Hola, Juan” .

Reto:

Modifica el ejemplo anterior para imprimir “Hola Universo” en la consola.

3 Gramática

3.1 Sintaxis

TypeScript es un lenguaje de programación de tipado estático basado en JavaScript. Está diseñado para proporcionar una mejor experiencia de programación al permitir a los desarrolladores escribir código con mayor claridad y precisión. La sintaxis de TypeScript es similar a la de JavaScript, pero con algunas diferencias.

Palabras clave: TypeScript tiene un conjunto de palabras clave específicas, como `class`, `interface`, `function` y `let`.

Indentación: La indentación es importante en TypeScript para mantener un código limpio y legible.

Conjunto de caracteres: TypeScript admite el conjunto de caracteres Unicode.

Sensibilidad a mayúsculas y minúsculas: TypeScript es sensible a mayúsculas y minúsculas, por lo que los nombres de variables y funciones deben escribirse de forma consistente.

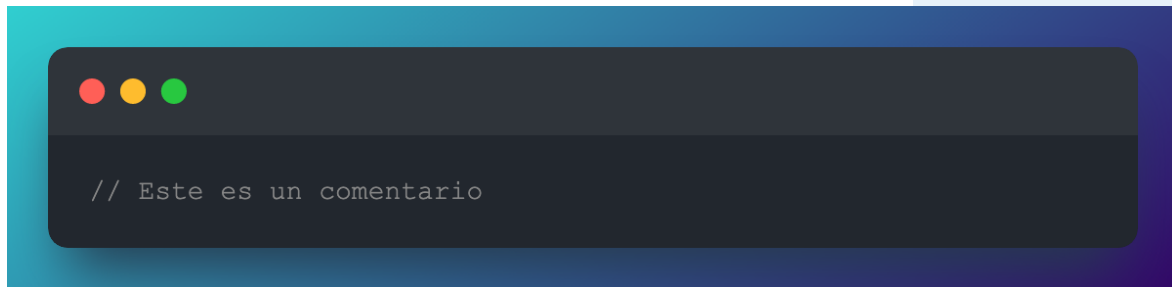
3.2 Comentarios

Los comentarios en TypeScript son líneas de texto que no son interpretadas como parte del código. Se usan para proporcionar información adicional sobre el código, como explicaciones, notas, o instrucciones para el desarrollador. Los comentarios también se pueden usar para deshabilitar código temporalmente, sin tener que eliminarlo completamente.

Ejemplo:

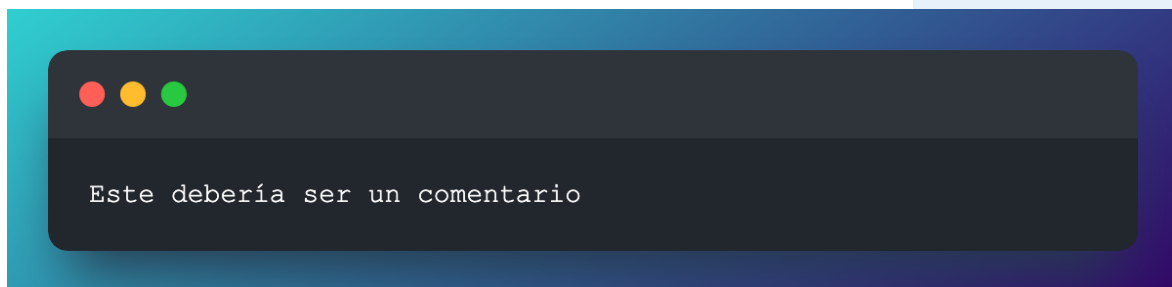
Este código es un comentario.

4 TIPOS Y VARIABLES



Reto:

Comenta esta línea para que no cause errores y se lea como comentario:



4 Tipos y variables

4.1 Variables

La manipulación de datos es una tarea fundamental de un lenguaje de programación. Para trabajar con datos necesitamos guardarlos en variables.

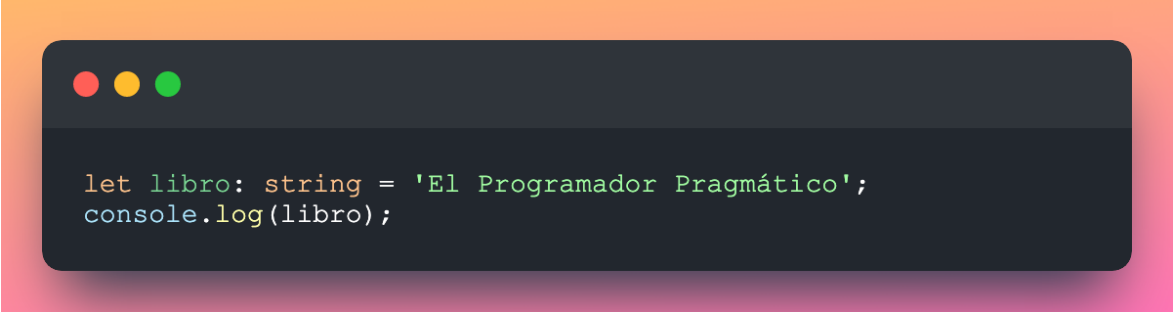
Una variable es un contenedor para almacenar datos que retiene su nombre y puede cambiar su valor a lo largo del tiempo. En los siguientes ejemplos vamos a ver varios tipos de datos que puedes guardar en variables.

Ejemplo:

Este código declara una variable llamada “libro” y le asigna el valor de una cadena


4 TIPOS Y VARIABLES

de texto que contiene el título de un libro. Luego, imprime el valor de la variable “libro” en la consola.



```
let libro: string = 'El Programador Pragmático';  
console.log(libro);
```


Texto es un tipo de dato útil para guardar información como números telefónicos y colores, entre otros. Este código asigna dos variables, una llamada telf que contiene un número de teléfono como una cadena de caracteres, y otra llamada color que contiene el color amarillo como una cadena de caracteres.



```
let telf: string = '406-234 2342';  
let color: string = 'amarillo';
```

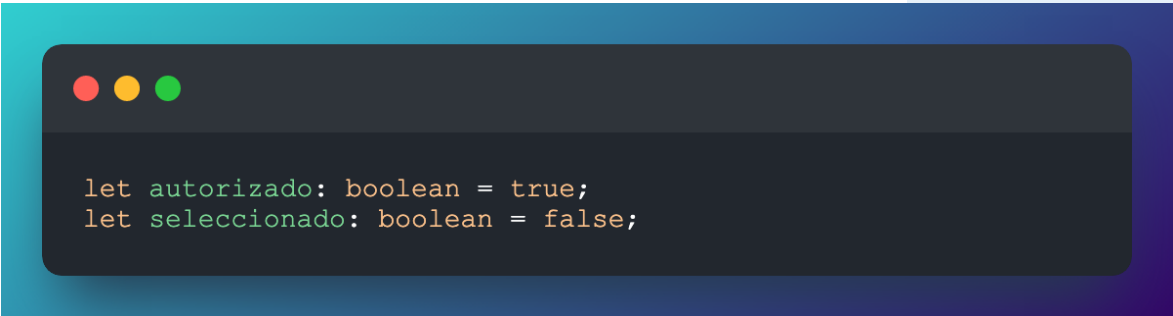
También podemos guardar datos como números enteros y decimales. Estos datos se usan para realizar operaciones matemáticas y representar valores de peso, dinero, entre otros. Este código asigna dos variables, una con un valor entero (100) y otra con un valor decimal (1.967857).

4 TIPOS Y VARIABLES



```
let entero: number = 100;
let decimal: number = 1.967857;
```

El tipo de dato booleano representa los valores de verdadero y falso. Este tipo de datos es útil, por ejemplo, para indicar si un usuario está autorizado a acceder a una app o no, entre varios usos. Este código crea una variable llamada “autorizado” que es verdadera y otra variable llamada “seleccionado” que es falsa.



```
let autorizado: boolean = true;
let seleccionado: boolean = false;
```

Es importante saber que, en el mundo del código binario, el número 1 representa verdadero y 0 representa falso.

Reto:

Crea una variable “a” con 33 como texto e imprímela a la consola.

4.2 Listas

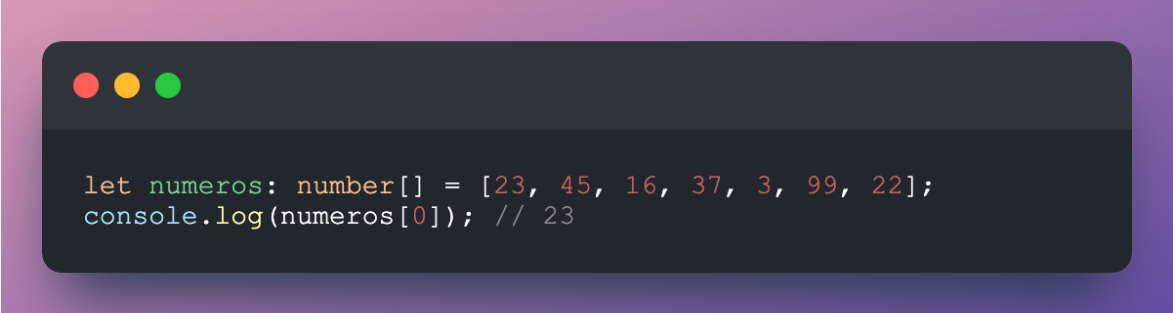
Las listas en TypeScript son una estructura de datos que nos permite almacenar una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo, desde números hasta sublistas. También los vas a encontrar con otros nombres

4 TIPOS Y VARIABLES

como arreglos, matrices, arrays, etc.

Ejemplo:

Este código está imprimiendo el primer elemento de una matriz de números a la consola. En este caso, el primer elemento es 23.



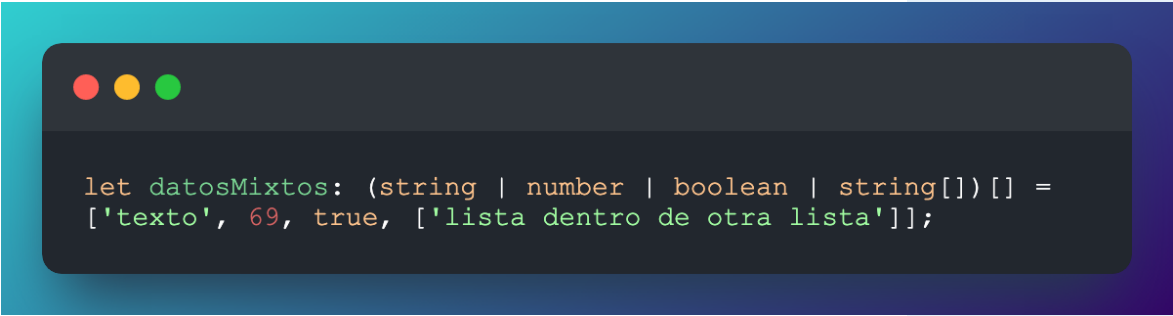
```
let numeros: number[] = [23, 45, 16, 37, 3, 99, 22];  
console.log(numeros[0]); // 23
```

También existen listas de texto. Este código crea una variable llamada “animales” que contiene una lista de elementos, en este caso, tres animales: perro, gato y tigre.



```
let animales: string[] = ['perro', 'gato', 'tigre'];
```

Y esta es una lista de datos mixtos. Este código crea una variable llamada “datosMixtos” que contiene una lista de elementos de diferentes tipos, como texto, números, booleanos y otra lista.



```
let datosMixtos: (string | number | boolean | string[])[] =  
  ['texto', 69, true, ['lista dentro de otra lista']];
```

Reto:

Crea una lista en TypeScript que contenga los nombres de los meses del año. Accede al 3er índice e imprímelo en la consola.

4.3 Objetos

Los objetos en TypeScript son una forma de almacenar y organizar datos mapeándolos de uno a uno. Estos datos se almacenan en forma de pares clave-valor, donde la clave suele ser una cadena de texto y el valor puede ser cualquier tipo de dato. También los vas a encontrar con otros nombres como mapas, diccionarios, etc.

Ejemplo:

Este código crea un objeto llamado jugadores que contiene dos pares clave-valor. El primer par clave-valor es 10: 'Messi' y el segundo es 7: 'Cristiano Ronaldo'. Luego, se imprime el valor asociado con la clave 10, que es 'Messi'.

4 TIPOS Y VARIABLES

```
let jugadores: {[key: number]: string} = {  
  10: 'Messi',  
  7: 'Cristiano Ronaldo'  
};  
console.log(jugadores[10]); // 'Messi'
```

También podemos mapear de texto a texto. Este código crea un objeto llamado “países” que contiene claves y valores. Las claves son códigos de países (EC, MX, AR) y los valores son los nombres de los países (Ecuador, México, Argentina).

```
let paises: { [key: string]: string } = {  
  EC: 'Ecuador',  
  MX: 'México',  
  AR: 'Argentina'  
};
```

E incluso podemos mapear de texto a listas de texto, entre muchas otras opciones. Este código establece un objeto llamado “emails” que contiene dos pares clave-valor. Cada clave es un nombre de persona y el valor es un arreglo de direcciones de correo electrónico asociadas con esa persona.

4 TIPOS Y VARIABLES

```
let emails: { [key: string]: string[] } = {  
  'Juan': ['juan@gmail.com'],  
  'Ricardo': ['ricardo@gmail.com', 'rick@aol.com']  
};
```

Reto:

Crea un objeto en TypeScript que represente una computadora, con sus respectivas características (marca, modelo, procesador, memoria RAM, etc).

4.4 Constantes

Las constantes son variables que no pueden ser reasignadas. Esto significa que una vez que se asigna un valor a una constante, este no puede ser cambiado.

Ejemplo:

Esta línea de código establece una constante llamada “pi” con un valor de 3.14159265359. Esta constante se puede usar para almacenar un valor numérico que no cambiará a lo largo del programa.

```
const pi: number = 3.14159265359;
```

Reto:

5 OPERADORES

Crea una constante llamada 'SALUDO' y asígnale el valor 'Hola Planeta' . Luego, imprime el valor de la constante en la consola.

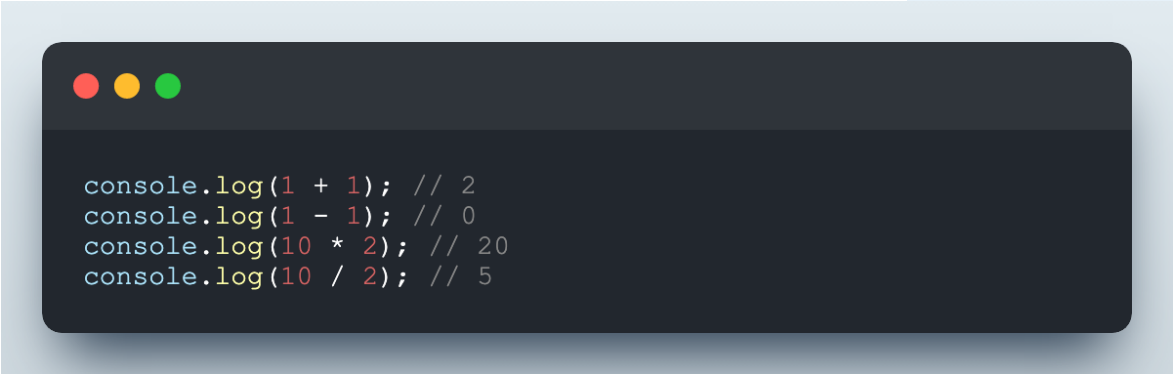
5 Operadores

5.1 Operadores de aritméticos

Los operadores aritméticos en TypeScript son usados para realizar operaciones matemáticas básicas. Estos operadores incluyen sumar, restar, multiplicar, dividir, entre otros.

Ejemplo:

Este código realiza operaciones matemáticas básicas, imprimiendo los resultados en la consola. Estas operaciones incluyen suma, resta, multiplicación y división.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays four lines of TypeScript code using console.log to perform basic arithmetic operations.

```
console.log(1 + 1); // 2
console.log(1 - 1); // 0
console.log(10 * 2); // 20
console.log(10 / 2); // 5
```

Reto:

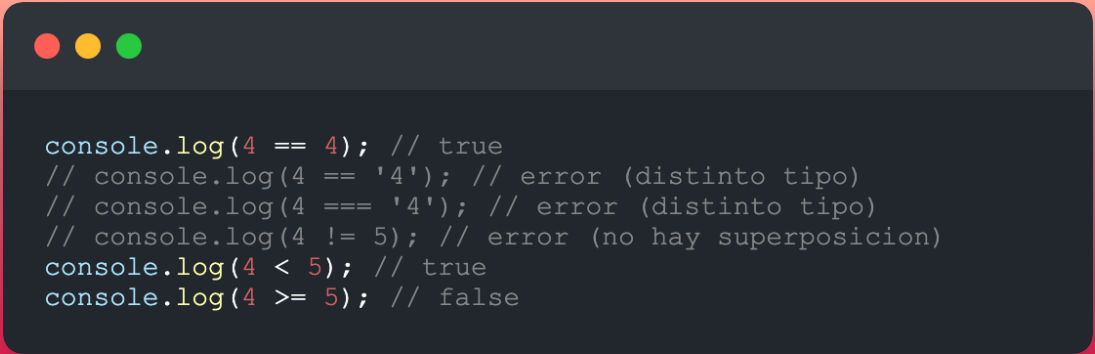
Usa los operadores aritméticos para calcular el área de un círculo con radio 5.

5.2 Operadores comparativos

Los operadores comparativos en TypeScript son usados para comparar dos valores y determinar si son iguales o diferentes. Estos operadores son útiles para determinar si dos valores son iguales, si dos valores no son iguales, si un valor es mayor o menor que otro, entre otros.

Ejemplo:

Este código muestra cómo se usan los operadores para comparar valores.



```
console.log(4 == 4); // true
// console.log(4 == '4'); // error (distinto tipo)
// console.log(4 === '4'); // error (distinto tipo)
// console.log(4 != 5); // error (no hay superposicion)
console.log(4 < 5); // true
console.log(4 >= 5); // false
```

Reto:

Escribe un programa que compare dos números 'a' y 'b' y determina si 'a' con el valor de 4 es mayor, menor o igual a 'b' con un valor de 2.

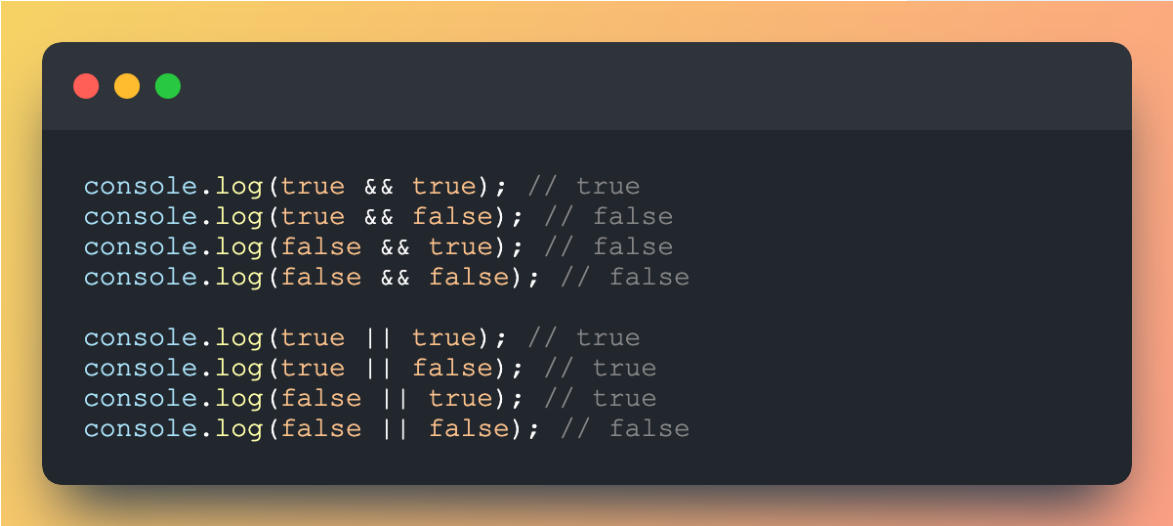
5.3 Operadores lógicos

Los operadores lógicos en TypeScript se usan para realizar comparaciones entre valores y devolver un valor booleano (verdadero o falso). Estos operadores pueden ser útiles, por ejemplo, si deseamos saber si un 'animal' es un gato y es un mamífero (al mismo tiempo).

Ejemplo:

6 CONDICIONALES

Este código muestra el uso de los operadores lógicos 'y' y 'o' para evaluar expresiones booleanas. El operador 'y' devuelve verdadero si ambas expresiones son verdaderas, de lo contrario devuelve falso. El operador 'o' devuelve verdadero si al menos una de las expresiones es verdadera, de lo contrario devuelve falso.



```
console.log(true && true); // true
console.log(true && false); // false
console.log(false && true); // false
console.log(false && false); // false

console.log(true || true); // true
console.log(true || false); // true
console.log(false || true); // true
console.log(false || false); // false
```

Reto:

Escribe una línea de código que devuelva verdadero si 'x' es mayor que 0 y 'y' es menor que 0.

6 Condicionales

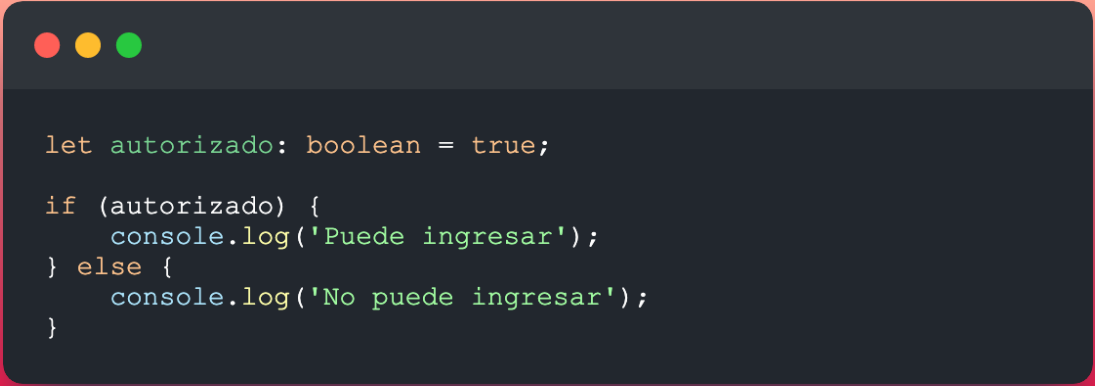
6.1 Condición única

Los condicionales son estructura de control de flujo en TypeScript, es decir, controlan el flujo del código. Permiten ejecutar una sección de código si una condición es verdadera. También permiten ejecutar otra sección de código, si la condición es falsa.

6 CONDICIONALES

Ejemplo:

Este código comprueba si una variable booleana (autorizado) es verdadera. Si es así, imprime un mensaje en la consola indicando que el usuario puede ingresar. Si no es así, imprime un mensaje indicando que el usuario no puede ingresar.



```
let autorizado: boolean = true;

if (autorizado) {
  console.log('Puede ingresar');
} else {
  console.log('No puede ingresar');
}
```

Reto:

Escribe código condicional que revise si un número 'a' es par o impar. La variable 'a' tiene el valor de 17 y debe imprimir a la consola 'Es par' si es par o 'Es impar' si es impar.

6.2 Múltiples condiciones

Adicionalmente, los condicionales permiten ejecutar varias secciones de código de acuerdo a varias condiciones.

Ejemplo:

En este caso podemos ver que el código que se ejecuta depende de varios valores. Este código comprueba si una variable llamada entero es igual a 99 o 100. Si es igual a 99, imprime "Es 99" en la consola. Si es igual a 100, imprime "Es 100" en la consola. Si no es ni 99 ni 100, imprime "No 99 ni 100" en la consola.

```
let entero: number = 100;

if (entero === 99) {
  console.log('Es 99');
} else if (entero === 100) {
  console.log('Es 100'); // ✓
} else {
  console.log('No 99 ni 100');
}
```

Este condicional es útil cuando hay una gran cantidad de posibles resultados para una expresión. Este código compara una variable (color) con diferentes valores y ejecuta una acción dependiendo del resultado de la comparación. En este caso, si la variable color es igual a 'amarillo', se imprimirá 'Advertencia' en la consola.

```
let color: string = 'amarillo';

switch (color) {
  case 'verde':
    console.log('Éxito');
    break;
  case 'amarillo':
    console.log('Advertencia'); // ✓
    break;
  default:
    console.log('Error');
    break;
}
```

7 BUCLES

Reto:

Crea un programa que evalúe una variable 'numero' y dependiendo del resultado, imprima un mensaje diferente. Si el número es mayor a 10, imprime "El número es mayor a 10" . Si el número es menor a 10, imprime "El número es menor a 10" . Si el número es igual a 10, imprime "El número es igual a 10" .

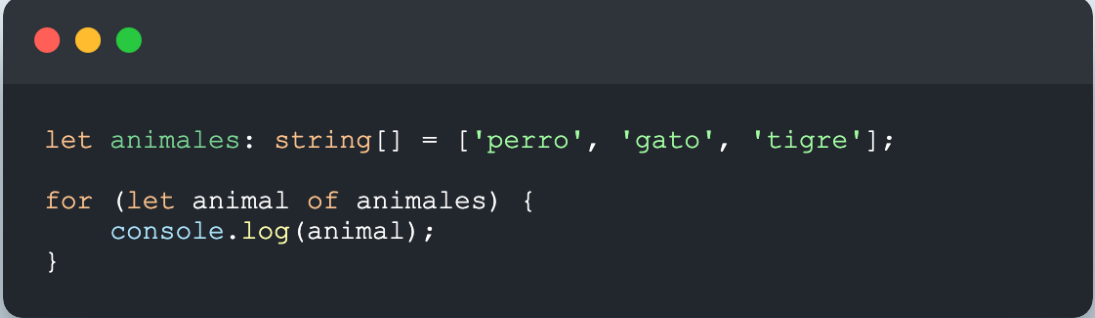
7 Bucles

7.1 Bucle for

Los bucles son otra estructura de control de flujo que se utilizan para iterar sobre una secuencia de elementos. También se los llama loops o ciclos.

Ejemplo:

Este código itera sobre una lista de animales e imprime cada elemento de la lista en la consola.



```
let animales: string[] = ['perro', 'gato', 'tigre'];

for (let animal of animales) {
  console.log(animal);
}
```

Reto:

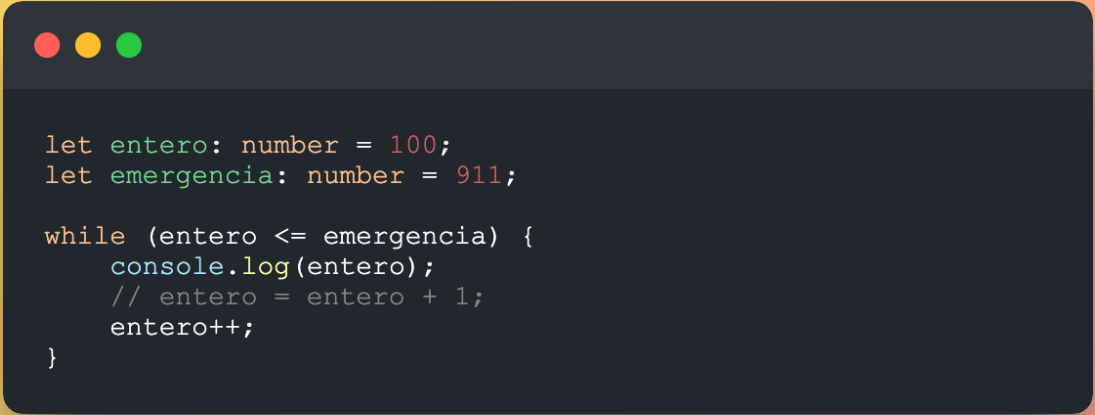
Escribe un bucle que imprima los números del 1 al 10 en orden ascendente.

7.2 Bucle while

while es un tipo de bucle en TypeScript que se usa para ejecutar una serie de instrucciones mientras se cumpla una condición. Esta condición se evalúa antes de cada iteración del bucle.

Ejemplo:

Este código establece un bucle while que imprime los números enteros desde 100 hasta 911 en la consola. El bucle while se ejecutará mientras la variable entero sea menor o igual a la variable emergencia. Cada vez que el bucle se ejecuta, la variable entero se incrementa en 1.



```
let entero: number = 100;
let emergencia: number = 911;

while (entero <= emergencia) {
  console.log(entero);
  // entero = entero + 1;
  entero++;
}
```

Ten cuidado de no incluir una condición para parar el bucle. Esto podría seguir corriendo tu código indefinidamente hasta congelar tu computador.

Reto:

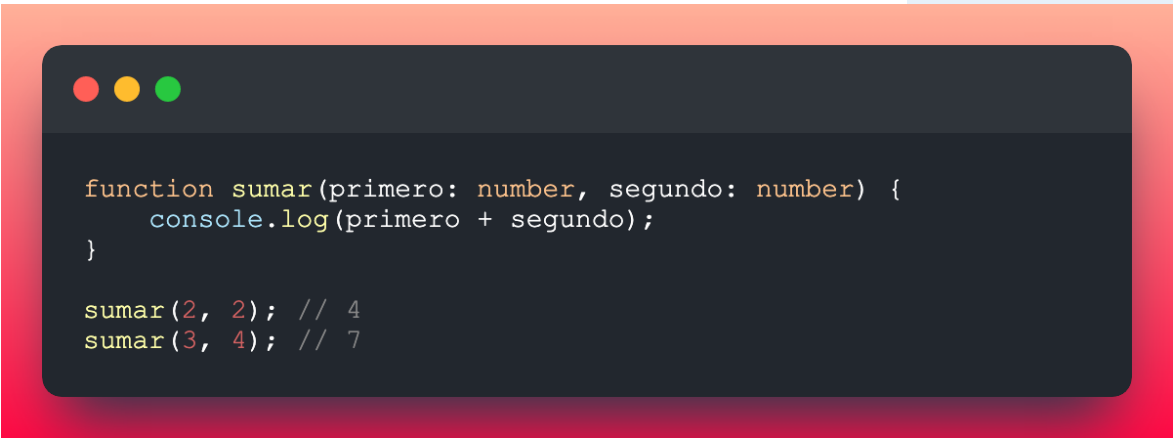
Crea un bucle while que imprima los números del 1 al 10 en la consola.

8 Funciones

En TypeScript, una función es un bloque de código diseñado para realizar una tarea específica y se puede reutilizar en varias partes el código. Las funciones se pueden definir para realizar cualquier tarea, desde realizar cálculos hasta mostrar mensajes en la pantalla.

Ejemplo:

Esta función toma dos argumentos (primero y segundo) de forma dinámica y los suma. Luego, imprime el resultado en la consola. Esta función se llama dos veces con diferentes argumentos para mostrar los resultados de la suma.



```
function sumar(primero: number, segundo: number) {  
    console.log(primero + segundo);  
}  
  
sumar(2, 2); // 4  
sumar(3, 4); // 7
```

También puedes crear funciones que retornen un valor. Esto significa que una vez que se ejecuta una función, el valor devuelto se puede usar en otra parte del código. Esta función toma dos argumentos (primero y segundo) y los multiplica para devolver el resultado. En este caso, el resultado es 6.

```
function multiplicar(primer: number, segundo: number):  
number {  
    return primero * segundo;  
}  
  
let resultado: number = multiplicar(3, 2);  
console.log(resultado); // 6
```

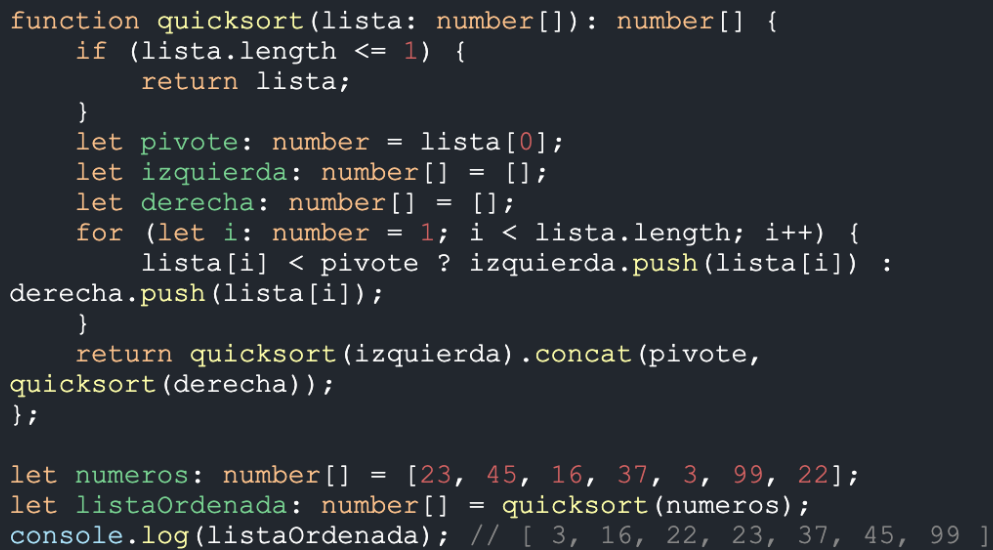
Esta función imprime el primer elemento de una lista. En este caso, la lista es una lista de animales, y la función imprime el primer elemento de la lista, que es 'perro'. Funciones como esta son útiles para evitar la repetición de código y para ahorrar tiempo.

```
function imprimirPrimerElemento(lista: string[]) {  
    console.log(lista[0]);  
}  
  
let animales: string[] = ['perro', 'gato', 'tigre'];  
imprimirPrimerElemento(animales); // 'perro'
```

Finalmente, puedes ver otro ejemplo de una función bastante compleja. Esta función implementa el algoritmo de ordenamiento QuickSort para ordenar una lista de elementos. El algoritmo comienza tomando un elemento de la lista como pivote y luego coloca los elementos menores que el pivote a su izquierda y los elementos mayores a su derecha. Luego, se aplica el mismo algoritmo a las sublistas izquierda y derecha hasta que la lista esté ordenada. No tienes que entender todo lo que hace

8 FUNCIONES

internamente pero te dará una idea de la complejidad que pueden tener programas como apps con miles de funciones.



```
function quicksort(lista: number[]): number[] {
  if (lista.length <= 1) {
    return lista;
  }
  let pivote: number = lista[0];
  let izquierda: number[] = [];
  let derecha: number[] = [];
  for (let i: number = 1; i < lista.length; i++) {
    lista[i] < pivote ? izquierda.push(lista[i]) :
    derecha.push(lista[i]);
  }
  return quicksort(izquierda).concat(pivote,
  quicksort(derecha));
};

let numeros: number[] = [23, 45, 16, 37, 3, 99, 22];
let listaOrdenada: number[] = quicksort(numeros);
console.log(listaOrdenada); // [ 3, 16, 22, 23, 37, 45, 99 ]
```

Reto:

Escribe una función que tome dos números como argumentos y devuelva el mayor de los dos.

9 Programación orientada a objetos (POO)

9.1 Paradigma

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la creación de objetos que contienen datos y funcionalidades. Estos objetos se relacionan entre sí para formar una estructura de datos compleja. Los lenguajes de programación orientados a objetos permiten que los programadores puedan crear objetos y usarlos para construir aplicaciones.

9.2 Clases

Las clases en TypeScript son una forma de definir objetos creando una plantilla. Se pueden usar para crear objetos con propiedades y métodos similares. Las propiedades son valores que pertenecen al objeto y los métodos son funciones que pertenecen al objeto.

Ejemplo:

En este ejemplo creamos una clase llamada 'Lenguaje' que inicializamos con las propiedades 'nombre' y 'año'. Adicionalmente creamos el método 'descripción' que imprime un texto usando las propiedades previas. Con esa clase podemos crear una instancia para el lenguaje 'HTML' y otra para 'CSS'. Ahora podemos crear, en teoría, un número infinito de lenguajes sin reescribir los objetos manualmente.

```
class Lenguaje {
  constructor(nombre: string, año: number) {
    this.nombre = nombre;
    this.año = año;
  }
  descripcion() {
    console.log(`${this.nombre} fue creado en
    ${this.año}`);
  }
}

let html = new Lenguaje('HTML', 1993);
let css = new Lenguaje('CSS', 1996);
html.descripcion(); // 'HTML fue creado en 1993'
css.descripcion(); // 'CSS fue creado en 1996'
```

Reto:

Crea una clase llamada Auto que tenga propiedades como marca, modelo y año. Luego, crea un par de instancias de esta clase.

10 Módulos

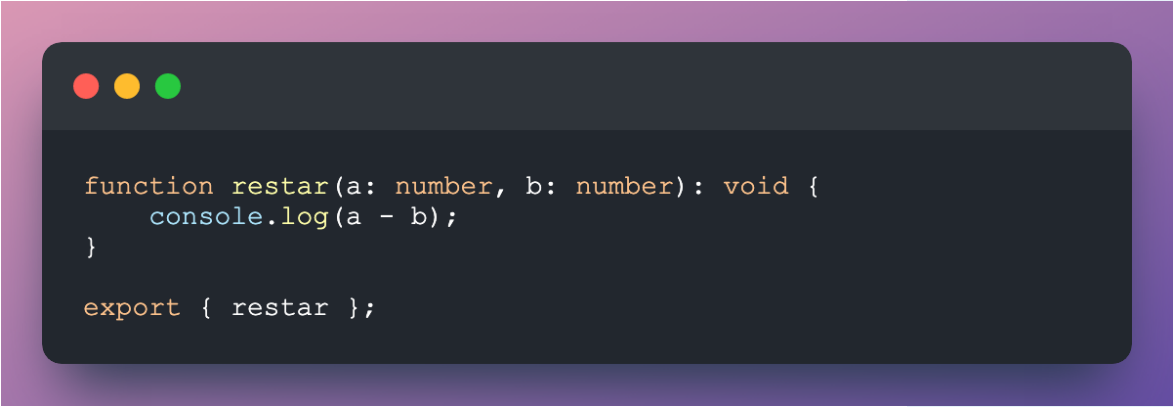
Los módulos son una forma de incluir código externo de otro archivo. Esto permite a los desarrolladores reutilizar código y ahorrar tiempo al escribir código.

Ejemplo:

Este archivo define una función que toma dos argumentos (a y b) y luego imprime el resultado de la resta de a y b en la consola. El código exporta la función para que pueda ser usada en otros archivos. Exportar es una forma de compartir código


10 MÓDULOS

entre archivos. Esto significa que un archivo puede exportar variables, funciones y objetos para que otros archivos los puedan usar.



```
function restar(a: number, b: number): void {  
    console.log(a - b);  
}  
  
export { restar };
```

Y en otro archivo podemos importar y utilizar esta función. Este código importa una función desde el archivo externo y luego la usa para restar 10 menos 2, resultando en 8.



```
import { restar } from './restar';  
  
restar(10, 2); // 8
```

Reto:

Crea un archivo llamado que contenga una función para calcular el área de un rectángulo. Luego, importa la función en otro archivo y usala para calcular el área de un rectángulo con lados de 3 y 4.

11 Siguientes pasos

11.1 Herramientas

1. **Compilador de Typescript:** Es una herramienta que se utiliza para compilar código Typescript a JavaScript. Esto le permite a los desarrolladores escribir código en Typescript y luego compilarlo para que pueda ser ejecutado en navegadores web.
2. **TypeScript Playground:** Esta herramienta es una aplicación web que permite a los desarrolladores escribir y ejecutar código Typescript directamente en el navegador. Esto le permite a los desarrolladores probar y depurar su código sin tener que compilarlo primero.
3. **TypeScript Language Service:** Esta herramienta es una extensión de Visual Studio Code que proporciona una variedad de características para ayudar a los desarrolladores a escribir código Typescript. Estas características incluyen la autocompleción de código, la verificación de errores, la refactorización de código y la depuración.
4. **TypeScript Node:** Esta herramienta es una biblioteca de Node.js que permite a los desarrolladores escribir código Typescript para aplicaciones Node.js. Esto le permite a los desarrolladores escribir código en Typescript y luego compilarlo para que pueda ser ejecutado en un entorno Node.js.

11.2 Recursos

1. [Typescript Documentation](#): Esta es la documentación oficial de Typescript, que incluye una guía de inicio, tutoriales, ejemplos y referencias.
2. [TypeScript Deep Dive](#): Esta es una guía profunda de Typescript escrita por Basarat Ali Syed. Esta guía cubre todos los aspectos de Typescript desde la sintaxis básica hasta los conceptos avanzados.

3. [Typescript Playground](#): Esta es una herramienta en línea para probar código de Typescript. Esta herramienta es útil para probar pequeños fragmentos de código sin tener que configurar un proyecto completo.
4. [Typescript Handbook](#): Esta es una guía de referencia para los tipos básicos de Typescript. Esta guía es útil para los desarrolladores que desean aprender los conceptos básicos de Typescript.
5. [Typescript Wiki](#): Esta es la wiki oficial de Typescript, que contiene una gran cantidad de información sobre cómo usar Typescript. Esta wiki es una excelente fuente de información para los desarrolladores que desean aprender más sobre Typescript.

11.3 ¿Que viene después?

1. Empezar a programar: Una vez que haya configurado TypeScript, puede empezar a escribir código usando TypeScript.
2. Compilar y ejecutar el código: Después de escribir el código, puede compilarlo y ejecutarlo usando el comando tsc.
3. Depurar el código: Si hay algún problema con el código, puede depurarlo usando herramientas como el depurador de TypeScript.
4. Publicar el código: Una vez que el código esté listo para ser publicado, puede publicarlo en un servidor web o en un repositorio de código.
5. Crear proyectos cada vez más avanzados para tu portafolio, seguir aprendiendo, y aplicar a trabajos relacionados a esta tecnología.

11.4 Preguntas de entrevista

1. ¿Qué es TypeScript?

- TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript. Es un superconjunto de JavaScript y agrega características de programación orientada a objetos, como clases, interfaces y tipos estáticos.

2. ¿Cómo se diferencia TypeScript de JavaScript?

- TypeScript es un superconjunto de JavaScript, lo que significa que todo el código JavaScript es válido en TypeScript. TypeScript agrega características de programación orientada a objetos, como clases, interfaces y tipos estáticos, que no están disponibles en JavaScript.

3. ¿Qué ventajas ofrece TypeScript?

- TypeScript ofrece una variedad de ventajas, como una mejor legibilidad del código, una mayor productividad, una mejor escalabilidad y una mejor mantenibilidad. También ofrece una mejor seguridad y una mejor depuración de errores.

4. ¿Qué es una interfaz en TypeScript?

- Una interfaz en TypeScript es una forma de definir un contrato entre una clase y sus clientes. Una interfaz define los métodos, propiedades y eventos que una clase debe implementar para satisfacer el contrato.

5. ¿Cómo se usa TypeScript en un proyecto?

- Para usar TypeScript en un proyecto, primero debe instalar el compilador TypeScript. Luego, debe crear un archivo de configuración TypeScript (tsconfig.json) para especificar las opciones de compilación. Finalmente, debe compilar el código TypeScript usando el compilador.

6. ¿Qué es una clase en TypeScript?

- Una clase en TypeScript es una plantilla para crear objetos. Una clase define los atributos y comportamientos de un objeto, como sus propiedades, métodos y eventos.

7. ¿Qué es una función en TypeScript?

- Una función en TypeScript es un bloque de código que se puede ejecutar para realizar una tarea específica. Las funciones pueden aceptar parámetros y devolver un resultado.

8. ¿Qué es un tipo en TypeScript?

- Un tipo en TypeScript es una etiqueta que se usa para especificar el tipo de datos de una variable. Los tipos en TypeScript incluyen cadenas, números, booleanos, objetos, matrices y muchos más.

9. ¿Qué es una variable en TypeScript?

- Una variable en TypeScript es un contenedor para almacenar datos. Una variable se puede usar para almacenar un valor, como un número o una cadena, o una referencia a un objeto.

10. ¿Qué es una declaración de tipo en TypeScript?

- Una declaración de tipo en TypeScript es una forma de especificar el tipo de datos de una variable. Esto se puede hacer usando la palabra clave “tipo” seguida del nombre del tipo. Por ejemplo, para declarar una variable como un número, se usaría la declaración “tipo numérico” .

11. ¿Qué es una función flecha en TypeScript?

- Una función flecha en TypeScript es una forma abreviada de escribir una función. Las funciones flecha se escriben usando la sintaxis “=>” en lugar de la sintaxis “function” .

12. ¿Qué es una clase abstracta en TypeScript?

- Una clase abstracta en TypeScript es una clase que no se puede instanciar directamente. Las clases abstractas se usan para definir una interfaz para una jerarquía de clases.

13. ¿Qué es una enumeración en TypeScript?

- Una enumeración en TypeScript es una forma de definir un conjunto de constantes. Las enumeraciones se usan para definir un conjunto de valores que una variable puede tomar.

14. ¿Qué es una función genérica en TypeScript?

- Una función genérica en TypeScript es una función que se puede usar con diferentes tipos de datos. Las funciones genéricas se escriben usando la sintaxis “tipo ” , donde T es el nombre del tipo de datos.

15. ¿Qué es una interfaz genérica en TypeScript?

- Una interfaz genérica en TypeScript es una interfaz que se puede usar con diferentes tipos de datos. Las interfaces genéricas se escriben usando la sintaxis “tipo ” , donde T es el nombre del tipo de datos.

16. ¿Qué es una clase de plantilla en TypeScript?

- Una clase de plantilla en TypeScript es una clase que se puede usar con diferentes tipos de datos. Las clases de plantilla se escriben usando la sintaxis “tipo ” , donde T es el nombre del tipo de datos.

17. ¿Qué es una función de flecha genérica en TypeScript?

- Una función de flecha genérica en TypeScript es una función flecha que se puede usar con diferentes tipos de datos. Las funciones de flecha genéricas se escriben usando la sintaxis “tipo ” , donde T es el nombre del tipo de datos.

18. ¿Qué es una declaración de importación en TypeScript?

- Una declaración de importación en TypeScript es una forma de importar código de otro archivo TypeScript. Esto se hace usando la palabra clave “import” seguida del nombre del archivo.

19. ¿Qué es una declaración de exportación en TypeScript?

11 SIGUIENTES PASOS

- Una declaración de exportación en TypeScript es una forma de exportar código a otro archivo TypeScript. Esto se hace usando la palabra clave “export” seguida del nombre del archivo.
20. ¿Qué es una declaración de módulo en TypeScript?
- Una declaración de módulo en TypeScript es una forma de organizar el código en módulos. Esto se hace usando la palabra clave “module” seguida del nombre del módulo.
21. ¿Qué es una declaración de tipo de alias en TypeScript?
- Una declaración de tipo de alias en TypeScript es una forma de crear un alias para un tipo de datos. Esto se hace usando la palabra clave “type” seguida del nombre del alias.
22. ¿Qué es una declaración de tipo de interfaz en TypeScript?
- Una declaración de tipo de interfaz en TypeScript es una forma de definir una interfaz. Esto se hace usando la palabra clave “interface” seguida del nombre de la interfaz.
23. ¿Qué es una declaración de tipo de clase en TypeScript?
- Una declaración de tipo de clase en TypeScript es una forma de definir una clase. Esto se hace usando la palabra clave “class” seguida del nombre de la clase.
24. ¿Qué es una declaración de tipo de función en TypeScript?
- Una declaración de tipo de función en TypeScript es una forma de definir una función. Esto se hace usando la palabra clave “function” seguida del nombre de la función.
25. ¿Qué es una declaración de tipo de enumeración en TypeScript?

11 SIGUIENTES PASOS

- Una declaración de tipo de enumeración en TypeScript es una forma de definir una enumeración. Esto se hace usando la palabra clave “enum” seguida del nombre de la enumeración.
26. ¿Qué es una declaración de tipo de genérico en TypeScript?
- Una declaración de tipo de genérico en TypeScript es una forma de definir un tipo genérico. Esto se hace usando la palabra clave “type” seguida del nombre del tipo genérico.
27. ¿Qué es una declaración de tipo de flecha en TypeScript?
- Una declaración de tipo de flecha en TypeScript es una forma de definir una función flecha. Esto se hace usando la palabra clave “arrow” seguida del nombre de la función flecha.
28. ¿Qué es una declaración de tipo de plantilla en TypeScript?
- Una declaración de tipo de plantilla en TypeScript es una forma de definir una clase de plantilla. Esto se hace usando la palabra clave “template” seguida del nombre de la clase de plantilla.
29. ¿Qué es una declaración de tipo de módulo en TypeScript?
- Una declaración de tipo de módulo en TypeScript es una forma de definir un módulo. Esto se hace usando la palabra clave “module” seguida del nombre del módulo.
30. ¿Qué es una declaración de tipo de objeto en TypeScript?
- Una declaración de tipo de objeto en TypeScript es una forma de definir un objeto. Esto se hace usando la palabra clave “object” seguida del nombre del objeto.
31. ¿Qué es una declaración de tipo de matriz en TypeScript?

11 SIGUIENTES PASOS

- Una declaración de tipo de matriz en TypeScript es una forma de definir una matriz. Esto se hace usando la palabra clave “array” seguida del nombre de la matriz.
32. ¿Qué es una declaración de tipo de tupla en TypeScript?
- Una declaración de tipo de tupla en TypeScript es una forma de definir una tupla. Esto se hace usando la palabra clave “tuple” seguida del nombre de la tupla.
33. ¿Qué es una declaración de tipo de literal en TypeScript?
- Una declaración de tipo de literal en TypeScript es una forma de definir un literal. Esto se hace usando la palabra clave “literal” seguida del nombre del literal.
34. ¿Qué es una declaración de tipo de unión en TypeScript?
- Una declaración de tipo de unión en TypeScript es una forma de definir una unión. Esto se hace usando la palabra clave “union” seguida del nombre de la unión.
35. ¿Qué es una declaración de tipo de intersección en TypeScript?
- Una declaración de tipo de intersección en TypeScript es una forma de definir una intersección. Esto se hace usando la palabra clave “intersection” seguida del nombre de la intersección.
36. ¿Qué es una declaración de tipo de restricción en TypeScript?
- Una declaración de tipo de restricción en TypeScript es una forma de definir una restricci