TS

Let’s learn TypeScript

@abdulbasetbappy

TS

What is TypeScript?

TypeScript is a superset of JavaScript that adds static typing and other features to the language.

With TypeScript, you can catch errors at compile time instead of at runtime, making it easier to write and maintain large-scale applications.

TS

Example

Here, the **name** parameter of the greet function is declared as a **string** using a type annotation. This means that if we try to call greet with a parameter that is not a string, TypeScript will catch the error and give us a compile-time error.

index.ts

```ts
// name has a type string
function greet(name: string) {
  console.log("Hello, " + name + "!");
}

greet("Alice");
```

TS

Type Annotations

Type annotations in TypeScript allow you to specify the type of a variable, parameter, or function return value.

This can help catch errors at compile-time rather than runtime and also provides more clarity to your code.

TS

Type Annotations (Cont)

Types helps ensure that we're using the correct types and catch any potential errors early in the development process.

index.ts

```ts
let myString: string = 'Hello, world!'; // Type annotation for a string variable
let myNumber: number = 42; // Type annotation for a number variable
let myBoolean: boolean = true; // Type annotation for a boolean variable

function add(a: number, b: number): number { // Type annotations for function parameters and return value
  return a + b;
}

let myArray: number[] = [1, 2, 3]; // Type annotation for an array of numbers

let myObject: { // Type annotation for an object with specific properties and types
  name: string,
  age: number,
  isStudent: boolean
} = {
  name: 'Alice',
  age: 25,
  isStudent: true
};
```

TS

Interfaces

Interfaces in TypeScript define the shape of an object. They allow you to specify the required properties and their types, as well as optional properties, functions, and more.

index.ts

```ts
interface Person {
  firstName: string;
  lastName: string;
  age: number;
}

function greetPerson(person: Person) {
  console.log("Hello, " + person.firstName + " " + person.lastName + "!");
}

const alice = { firstName: "Alice", lastName: "Smith", age: 30 };

greetPerson(alice);
```

TS

Interfaces (Cont)

In the previous example, we defined an interface called **Person** that describes the properties of a **person** object.

We then declared a function called **greetPerson** that takes a parameter of type Person.

Finally, we created an object that conforms to the Person interface and pass it to the greetPerson function.

TS

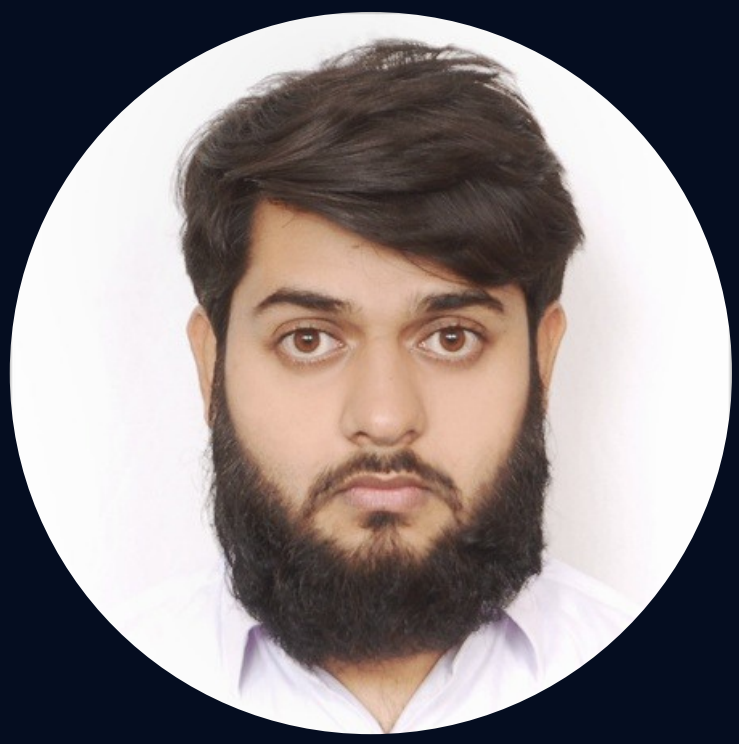
TypeScript Benefits

- **Type Safety:** It provides static type checking which helps catch errors before runtime. This can lead to fewer bugs, better maintainability, and more confidence in your code.
- **Scalability:** It is designed to scale to large codebases. Its type system makes it easier to refactor code and maintain consistency across large projects.

TS

TypeScript Benefits

- **Better Code Quality:** By catching errors earlier, TypeScript can help improve code quality. It can also encourage better coding practices, such as writing more modular and reusable code.
- **JavaScript Interoperability**: It is a superset of JavaScript, so it can easily integrate with existing JavaScript code. This makes it a good choice for gradually adopting TypeScript into existing projects.



Abdul Baset Bappy

@abdulbasetbappy

Did You Find Useful?

Share Your Thoughts