**built-in types - TypeScript Types**

While JavaScript comes with built-in types such as numbers, strings, booleans, null, undefined, and objects, TypeScript takes it a step further by extending this list and introducing several new types. TypeScript introduces types like 'any,' 'unknown,' 'never,' 'enum,' and 'tuple' to provide developers with a more versatile and robust type system.

**Type Annotations**

In TypeScript, type annotations are used to explicitly specify the type of a variable, function, or any other entity in your code. This helps the TypeScript compiler to understand and enforce the types you intend to use in your program. Type annotations are typically added using a colon (:) followed by the desired type.

**Using Type Annotations with Variables:**

You can explicitly annotate variable types in the following way:

let myNumberExOne: number = 42;

let myStringExOne: string = "Hello, TypeScript";

console.log("My Number Is:", myNumberExOne);

console.log("My String Is:", myStringExOne);

In this example, myNumber is explicitly annotated with the type number, and myString is annotated with the type string. This makes it clear to both developers and the TypeScript compiler what types these variables should hold. This can help catch potential type-related issues during development.

Another way to write this code is as follows:

let myNumberExTwo: number;

let myStringExTwo: string;

myNumberExTwo = 42;

myStringExTwo = "Hello, TypeScript";

console.log("My Number Is:", myNumberExTwo);

console.log("My String Is:", myStringExTwo);

In this alternative approach, you first declare the variables and then assign values to them later. TypeScript still understands the types based on the initial declarations, providing the same level of type safety and code clarity.

By using type annotations, you ensure that your code is more robust, maintainable, and less error-prone, making it easier to work on large projects and collaborate with other developers.

**dynamic type determination**

If you choose not to use type annotations in TypeScript, the language can still infer the type dynamically based on your declaration. For instance:

let greeting = "Hello, world!";

console.log(greeting);

In this example, TypeScript will automatically infer the type of the variable greeting as a string. However, while TypeScript can infer types, it's generally not recommended to rely solely on dynamic type inference, as it may lead to unintended issues and reduced code clarity. Explicitly defining types through type annotations is a best practice for writing maintainable and error-free TypeScript code.

**Any Type**

Now, regarding the any type, it's a special type in TypeScript that can represent any type of value. When a variable is assigned the any type, it essentially tells the TypeScript compiler to disable type checking for that variable, allowing it to hold values of any type.

Here's an example of using the any type:

let myVariable: any = 42;

myVariable = "Hello, TypeScript";

console.log(myVariable)

In this case, myVariable can be assigned a number or a string or any other type without any type checking.

However, it's generally advised to avoid using the any type whenever possible in TypeScript for the following reasons:

1. **Loss of Type Safety:**
2. **Reduced Code Reliability:**
3. **Maintenance Challenges:**
4. **Interferes with Tooling:**