The most bsic types in Typescript as basic primitives in Javascript:

* bigint: 0n, 2n, -4n, …
* boolean: true or false
* null
* number:0 , 2, -4 …
* string: “helloworld”
* symbol: Symbol(), Symbol("hi"), …
* undefined

example:

1337n; // bigint true; // boolean null; // null 1337; // number "Louise"; // string Symbol("Franklin"); // Symbol undefined; // undefined

At its core, TypeScript’s type system works by:

1. Reading in your code and understanding all the types and values in existence
2. For each object, seeing what type its initial declaration indicates it may contain
3. For each object, seeing all ways it’s used later on
4. Complaining to the user if an object’s usage doesn’t match with its type

Take the following snippet, in which TypeScript is emitting a type error:

let firstName = "Whitney";

firstName.length();

// This expression is not callable.

// Type 'Number' has no call signatures

**Kinds of Errors** While writing TypeScript, the two kinds of “errors” you’ll come across most frequently are:

* Syntax: blocking TypeScript from being converted to JavaScript.
* Type: something mismatched has been detected by the type checker.

Syntax Errors:

Syntax errors are when TypeScript detects incorrect syntax that it cannot understand as code.

let let wat;

// Error: ',' expected.

Type Errors

Type errors occur when your syntax is valid but the TypeScript type checker has detected an error with the program’s types.

console.blub("Nothing is worth more than laughter.");

// Error: Property 'blub' does not exist on type 'Console'.