**Basic Type**

1. Typescript type

In typescript type is a convenient way to refer to the different properties and function that a value has.

Every value in typescript has a type

Typescript compiler uses types to analyze our code for hunting bugs and errors.

**Types in Typescript**

Typescript inherits the built-in types from javascript. Typescript types are categorized into

* Primitive types
* Object types

Primitive types

|  |  |
| --- | --- |
| **Name** | **Description** |
| string | it represents text data |
| number | It represents numeric values |
| boolean | It has true and false value |
| null | It has one value null |
| undefined | It has one value undefined |
| symbol | It represents a unique constant value |

Object types

Object types are functions, arrays, classes etc.

Purpose of types in typescript

**Generic**

Typescript generics allow us to write reusable and generalized forms of functions, classes, and interfaces.

**Generic function with multiple types**

The merge() function merges two objects with the type U and V. it combines the property of the two objects into a single object.

Type inference infers the returned value of the merge() function as intersection type of U and V which is U & V

**Benefits of typescript generics**

leverage type checks at the compile time, eliminate type casting and allow us to implement generic algorithm.

Generic constraints

Use **extends** keyword to constrain the type parameter to a specific type.

Use **extends keyof** to constrain a type that is the property of another object.

The generic function expects two objects. However it doesnt prevent us from passing a non-object

Typescript doesn’t issue any errors

Instead of working with all type, in order to denote the constraint, we use the **extends** keyword.

**Using type parameter in generic constraints**

To fix it, we ass a constraint to k to ensure that it is a key of T

**Generic class**

A generic class has a generic type parameter list in angle brackets<> that follows the name of the class

class className1<T> {}

typescript allow us to have multiple generic types in the parameter list

// multiple generic types

class className2<K, T> {}

the generic constraints are also applied to the generic types in the class

// generic constraints

type TypeA = {};

class className3<T extends TypeA> {}