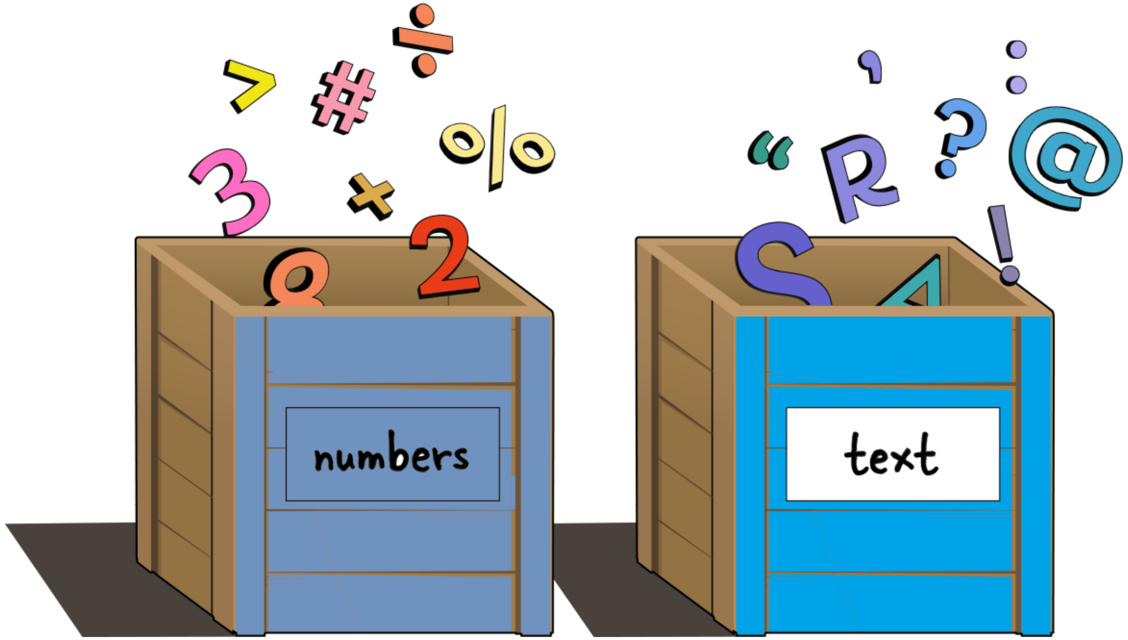
Typescript DataTypes

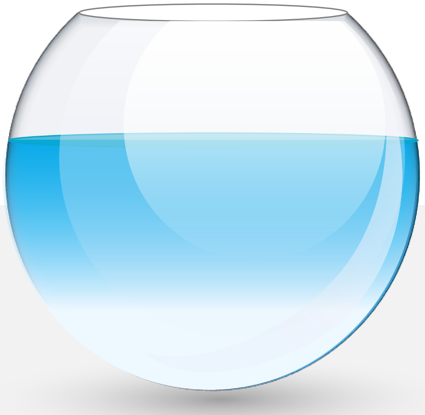
Chapter-2



Aitrich

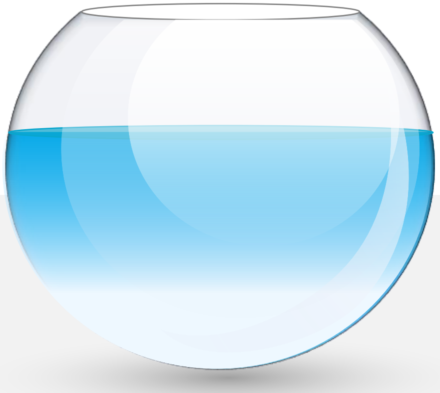
DataTypes

A data type is a classification of a variable representing the type of data it can hold



Typescript Data Types

Built-in types
User-defined types



Typescript Assign a type

Explicit
Implicit

Built-in Types

- ✓ Number
- ✓ String
- ✓ Boolean
- ✓ void
- ✓ null
- ✓ undefined
- ✓ Any

Number & String

Number

It consists of whole numbers and floating point values, it is represented by the keyword number.

String

The values in a string are surrounded by single quotation marks or double quotation marks. It is represented by the keyword string.

Number

```
let age:number=1;

let amount:number=510.25
```

```
let name1:string='remya';

let name2:string="raju";

console.log(name1);

console.log(name2);
```

Boolean & Void

Boolean

Boolean represents the true or false values. It is represented by the keyword boolean..

Void

The void type is used when there is no data, it is used when functions return no value. It is represented by the void keyword

Boolean

```
let a:boolean=true;
let b:boolean=false;
console.log(a);
console.log(b);
```

Void

```
function greet(): void {
  console.log("Hello, world!");
}
```

Null & undefined

Null

Null refers to the absence of any object value. It means nothing or no value. It is similar to the void type but we have to define it explicitly. The null keyword is used to define the null type in typescript.

undefined

Denotes all uninitialized variables in typescript. Assigning a value to an undefined data type is of no use.

Null

```
let myValue: string | null = null;
```

undefined

```
let myVariable: number; // automatically
initialized with 'undefined'
```

Aitrich

Any

any

In TypeScript, the any type is a special type that is used to represent values of any type.

any

```
let a: any;
let b: any;
a="aitrich";
b=1;
console.log(a);
console.log(b);
```

User-Defined Types

- ✓ Array
- ✓ Tuple
- ✓ Interface
- ✓ class
- ✓ Functions
- ✓ Enums

Array & Tuple

Array

Array is a collection of elements of a similar data type. TypeScript supports working with arrays of values.

Tuple

It is a data type that includes two sets of values of different data types.

Arrays can be Written in

```
var list : number [] = [1,2,3]
var list : Array<number>=[1,2,3]
var list : any [] =[1, true , “free”]
```

Tuple can be Written in

```
var x : [string , number ]
x=[“hello” , 10]
```

Aitrich

Interface

Interface

Interfaces are a way to define contracts for object shapes..

Interface

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

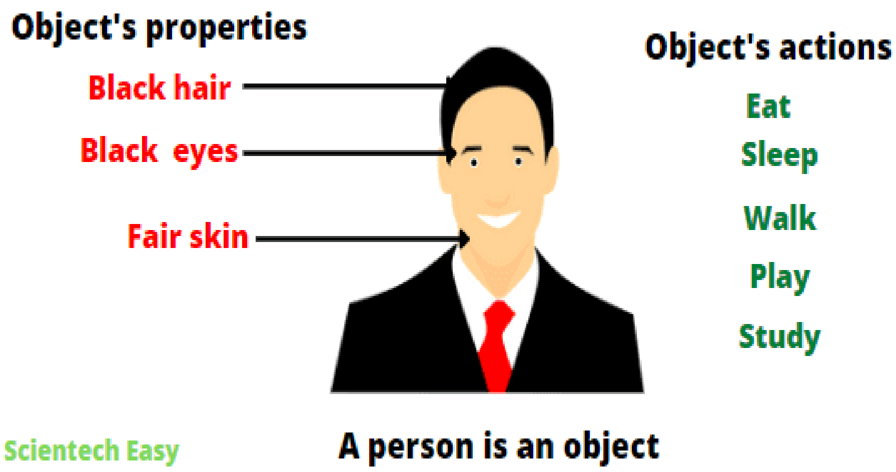
Interface

```
const person: Person = {
  firstName: "John",
  lastName: "Doe",
  age: 25,
};
```

class

class

Classes allow you to create objects with both properties and methods.



A person is an object

```
class Person {
  private name: string;

  public constructor(name: string) {
    this.name = name;
  }

  public getName(): string {
    return this.name;
  }
}

const person = new Person("Jane");
console.log(person.getName());
```

Enum

Enum

Enums provide a way to define a set of named constant values.

Enum

```
enum Color {
  Red = "RED",
  Green = "GREEN",
  Blue = "BLUE",
}

const selectedColor: Color =
Color.Green;.
```

Questions???