Object Type

- Object into computer programming was introduced in early 1960's by **Alan Kay.**
- Object can keep relative data and functionality under one reference name.
- Data is stored in properties and functionality in functions.
- Object is a set of properties and functions encapsulated into one component.
- TypeScript objects are defined in memory by using "**object**" type.
- TypeScript early versions use "any" type for object.

  Syntax:
  ```
  let product:object = {
     property: data,
     method: function() { }
   }
  ```
- The members of an object are accessible with in the object by using "**this**" keyword.

  ```
  this.property;
  this.method();
  ```

- The members of an object are accessible outside the object by using the object name.

  ```
  product.property;
  product.method();
  ```

- Property in object can handle any type of data i.e Primitive or Non-Primitive type.
- You can reuse an object and its members for different requirements.

Ex:

```
let product:object = {
    Name:"Samsung TV",
    Price: 45000.44,
    InStock:true,
    Qty:2,
    ShippedTo: ["Delhi","Hyd"],
    Total: function(){
        return this.Qty * this.Price;
    },
    Print: function(){
        console.log(`Name=${this.Name}\nPrice=${this.Price}\nQty=${this.Qty}\nTotal=${this.
Total()}\nInStock=${(this.InStock==true)?"Available":"Out of Stock"}\nShipped To=${this.Shi
ppedTo.toString()}`);
    }
}
console.log(`-------TV Details--------`);
product.Print();
console.log(`-------Shoe Details------`);
product.Name = "Nike Casuals";
product.Price = 4200.30;
product.Qty = 2;
product.InStock =false;
product.ShippedTo = ["Chennai","Mumbai"];
product.Print();
```

Issues with Object:

- Object Contains default keys, which can collide with your own keys.
- Properties are not referred as keys. They have a key by default configured.
- Keys must be a string type.
- The number of items in an object must be determined manually.
- It is hard for reflection.
- Iterating over an object requires obtaining its keys and values by using “for..in”.

  Ex:
  ```
  let product:any = {
    Name: "Samsung TV",
    Price: 45000.55
  }
  for(var property in product) {
      console.log(`${property} : ${product[property]}`);
  }
  ```

- Not optimized for frequent additions and removals of keys and value pairs.
- Explicitly you have use special operators and function for adding or removing items.

Ex:

```
let product:any = {

  Name: "Samsung TV",

  Price: 45000.55

}

delete product.Price;

if(product.Price==undefined) {

    console.log(`Name=${product.Name}`);

} else {

console.log(`Name=${product.Name}\nPrice=${product.Price}`);

}
```

If you are looking for a Key / Value pair of collection then go with “Map” and “Set” introduced into JavaScript from ES6.

Array of Objects

[JSON – Type]

[{ }, { }, { }]