Method Parameters

- Every Parameter is mandatory.
- You can define optional parameters by using “null reference character [?]”.
- You can use “undefined” type to verify, value defined or not.

  Ex:

```
class Product
{
   public Details(Name:string, Price:number, Stock?:boolean){
       if(Stock==undefined){
console.log(`Name=${Name}\nPrice=${Price}`);
       } else {
console.log(`Name=${Name}\nPrice=${Price}\nStock=${Stock}`);
       }
   }
}
let tv = new Product;
tv.Details("Samsung TV",56000.55, true);
```

Note: A required parameter can’t follow after optional parameter.

Syntax:

method(**param?**:string, param:number); // invalid

- All optional parameters must be last parameters in formal list.
- A method supports maximum 1024 params.
- Method parameter can be any type
    - Array
    - Object
    - Function
    - String
    - Number
    - Boolean etc.

Ex: Object as Parameter

```
class Product
{
   public Details(product:any){
     for(var property in product) {
        console.log(`${property} : ${product[property]}`);
     }
   }
}
```

```
let tv = new Product;
tv.Details({Name: "Samsung TV", Price: 45000.55, Stock:true});
console.log(`------------------`);
let shoe = new Product;
let nike = {
    Name: "Nike Casuals",
    Price: 45000.55
}
shoe.Details(nike);
```

Ex: Array as parameter

```
class Demo
{
    PrintList(list:string[]){
        for(var item of list) {
            console.log(item);
        }
    }
}
```

```
let obj = new Demo;
obj.PrintList(new Array("TV", "Mobile"));
obj.PrintList(["Nike Casuals", "Lee Boot"]);
let fashion = ["Shirt", "Jeans"];
obj.PrintList(fashion);
```

Ex: Function Parameter

```
class Login
{
   public VerifyDetails(password:string, success:any,
failure:any){
      if(password=="admin12") {
         success();
      } else {
         failure();
      }
   }
}
let obj = new Login;
```

```
obj.VerifyDetails("admin123",function(){console.log('Login Success')}, function(){console.log('Invalid Password')});
```

Ex: Array of Objects

```
class Product
{
    public Details(products:any[]) {
        for(var item of products) {
            console.log(`${item.Name} - ${item.Price}`);
        }
    }
}
let tv = new Product;
tv.Details([{Name:"TV", Price: 45000.55}, {Name:"Mobile", Price:6000.55}]);
```

ES5 introduced “rest” parameters

- Single formal parameter can handle multiple actual values.
- It is defined by using “…paramName”
- Every method can have only one rest parameter.

- Rest parameter must be the last parameter in formal list.

Ex:

```
class Product
{
  public PrintList(...list:any){
    for(var item of list) {
       console.log(item);
    }
  }
}
let obj = new Product;
obj.PrintList("Samsung TV", "Mobile", "Nike Casual", "Lee Boot");
```

Method with Return Type

- Why we need a method to return value?
  To build an expression.
- Expression performs specified operation and returns a value.
- You can build dynamic expressions by using method with return value.

- You can use the method reference memory for storing a value.