Modules

- Module in programming is a set of components which you can import and use in any application.
- A module allows to maintain your contracts, templates, and application library.
- It enables reusability of component, maintainability and testability.
- TypeScript module comprises of Interfaces, Abstract Classes and Classes.
- JavaScript uses various modules systems like
    - Common JS
    - AMD [Asynchronous Module Distribution]
- TypeScript can internally handle modules as it is supported with internal module system.
- If you create a new ".ts" file then it is referred as module.
- Any component defined in a module is not accessible outside the module scope.
- You have to mark the component with "export" keyword so that it can be accessed outside the component.

Syntax:

```
export interface IName
{
}
export class className
{
```

}

- You can inject the module into any component by using “import”.

  import { interfaceName, className } from ‘./path/file’;

- JavaScript uses “require” to import any module from its module systems.

  var referenceName = require(“moduleName”);

Ex:

- Add a new folder “ModuleSystem”
- Add subfolders
  - Contracts
  - Templates
  - Components
  - App
- Go to “Contracts” folder and add

  CategoryContract.ts

  export interface ICategory
  {
  CategoryId:number;
  CategoryName:string;
  }

  ProductContract.ts

  export interface IProduct

```
{
    Name:string;
    Price:number;
    Qty:number;
    Total():number;
    Print():void;
}
```

- Go to Templates folder and add **ProductTemplate.ts**

```
import { IProduct } from '../Contracts/ProductContract';
import { ICategory } from
'../Contracts/CategoryContract';

export abstract class ProductTemplate implements
IProduct, ICategory
{
    public Name:string;
    public Price:number;
    public Qty:number;
    public CategoryName:string;
    public CategoryId:number;
    public abstract Total():number;
    public Print():void {
console.log(`Name=${this.Name}\nPrice=${this.Price}\n
Qty=${this.Qty}\nTotal=${this.Total()}\nCategory
Name=${this.CategoryName}`);
    }
```

```
}
```

- Go to Components and add

ProductComponent.ts

```
import { ProductTemplate } from
'../Templates/ProductTemplate';

export class ProductComponent extends
ProductTemplate
{
   public Name:string = "";
   public Price:number = 0;
   public Qty:number = 0;
   public CategoryId:number = 0;
   public CategoryName:string = "";
   public Total():number {
      return this.Qty * this.Price;
   }
   public Print(){
      super.Print();
   }
}
```

- Go to App folder and add

MyApp.ts

```
import { ProductComponent } from
'../Components/ProductComponent';

let tv = new ProductComponent();
console.log("---------TV Details----------");
tv.Name = "Samsung TV";
```

```
tv.Price = 34000.55;
tv.Qty = 2;
tv.CategoryId = 1;
tv.CategoryName = "Electronics";
tv.Print();
let shoe = new ProductComponent();
console.log("--------Shoe Details-----------");
shoe.Name = "Nike Casuals";
shoe.Price = 2300.44;
shoe.Qty = 1;
shoe.CategoryId=2;
shoe.CategoryName="Footwear";
shoe.Print();
```

JavaScript uses “require()”

```
var productComponent =
require("../ModuleSystem/Components/ProductComponent"
);
var obj = new productComponent.ProductComponent();
```

Namespaces