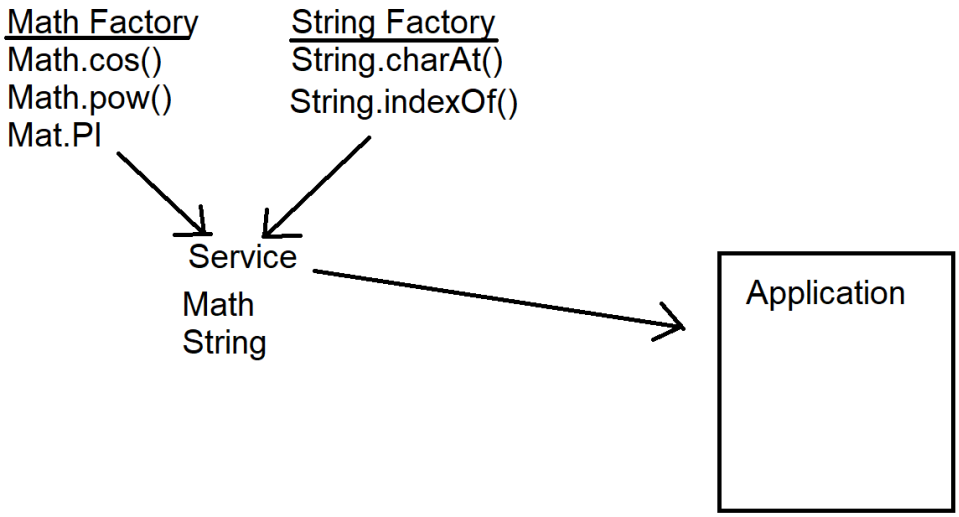
Angular Services

- Service is a pre-defined business logic which can be reused in the application by injecting into any component.
- *Service is a collection of Factories*.
- Factory is a collection of related type of functions.



- Factory uses "**Single Call Mechanism**". Every time when you want to use a function you need an object to create. [Disconnected and Discrete]
- Service uses a "**Single Ton Mechanism"**. Object is created only for the first request and the same object is used across requests. [Connected and Continuous]
- Angular Service uses "Dependency Injection" to inject a service into any component constructor. Instead of creating the service object using "new" operator.

- Dependency injection is an “Application Design Pattern”.
- Angular has its own DI framework.
- Angular uses DI framework to increase the application efficiency and modularity.
- Dependencies are services or objects that a class needs to perform its function.
- ***Dependency Injection is a coding pattern in which a class asks for dependencies from external sources rather than creating them itself.***
- It allows to share information between classes.
- In Angular, the DI framework provides declared dependencies to a class when that is instantiated.
- The DI framework lets you to supply the data to a component from an injectable service class.
- Technically **service in Angular is a class**, which comprises of set of service methods, which you can inject and use in any component.
- It enables reusability, maintainability and testability.
- Angular can’t inject any service into component until you configure an Angular Dependency Injector with provider.
- **@Injectable()** is an injector for service in Angular.
  Syntax:
  DemoService.Service.ts

```
import { Injectable } from '@angular/core';
@Injectable()
export class DemoService
{
    // set of service methods.
}
```

What is Injector?

- It is an object in Angular "Dependency-Injection" system.
- It can find a named dependency in its cache memory or create a dependency using configured **provider**.
- Injectors are created for "**NgModules"** automatically as part of the **"bootstrap"** process.
- Injector provides a **singleton instance** of a dependency and can inject the same instance into multiple components.
- The injector provides a hierarchy so that the content can be used for the parent and child components.
- We can **configure injector with different providers** that provide different implementations of the same dependency.

What is Provider?

- Provider is an object that implements on of the “Provider” interfaces.
- Provider defines how to obtain an injectable dependency associated with a DI token.
- Injector uses a provider to create a new instance of dependency for class.
- Angular registers its own providers with every injector for services.
- Angular provides different types of providers
    - ValueProvider
    - ClassProvider
    - TypeProvider
    - ConstructorProvider
    - FactoryProvider

Syntax:

```
import { Injectable } from ‘@angular/core’;
@Injectable({
  providedIn: ‘root/child’
})
export class SampleService
{
  constructor() { }
  //service methods
```

}

Injecting Services:

- You have to inject the service into any component.
- We have to make sure that service is injected into component rather than creating a new instance.
- You can tell angular to inject dependency in a components constructor by specifying a constructor parameter with the dependency type. (service type)
- Every parameter defined in a constructor is accessible only within the constructor.
- You can define an “Access Modifier to specify the scope of parameter constructor, which can public, private or protected.

Syntax:

```
import { SampleService } from ‘..sample.service’;

export class SampleComponent {

   constructor(private sampleservice: SampleService){}

}
```

Note: If a service is not defined with provider then you have to explicitly inject the service from “NgModules”.

FAQ: What is the meaning of “providedIn:root”?

- It configures a singleton pattern for service so that it can implicitly inject into any component.
- Service can be implemented without singleton by configuring in the “app.module.ts” Providers.

  providers: [ServiceName]

Ex:

- Add a new folder “services” in “app” folder
- Add a new file
  Captcha.service.ts

```
import { Injectable } from '@angular/core';

@Injectable(
 {
    providedIn: 'root'
 }
)
export class CaptchaService
{
   public GenerateCode(){
      let a = Math.random() * 10;
      let b = Math.random() * 10;
      let c = Math.random() * 10;
      let d = Math.random() * 10;
```

```
        let e = Math.random() * 10;
        let f = Math.random() * 10;
        let code = `${Math.round(a)}
${Math.round(b)} ${Math.round(c)}
${Math.round(d)} ${Math.round(e)}
${Math.round(f)}`;
        return code;
    }
}
```

- **Create a login component**
- Login.component.ts

```
import { Component, OnInit } from
'@angular/core';
import { CaptchaService } from
'../Services/captcha.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  public code;
  constructor(private captcha: CaptchaService) { }
```

```
  ngOnInit(): void {
    this.code = this.captcha.GenerateCode();
  }
  public RefreshClick() {
    this.code = this.captcha.GenerateCode();
  }
}
```

- Login.component.html

```
<div class="container-fluid" style="width: 300px;
margin: auto;">
    <h2>User Login</h2>
    <div class="form-group">
        <label>User Name</label>
        <div>
            <input type="text" class="form-control">
        </div>
    </div>
    <div class="form-group">
        <label>Password</label>
        <div>
            <input type="password" class="form-
control">
        </div>
    </div>
    <div class="form-group">
        <label>Verify Code</label>
```

```
            <div>
               {{code}}
               <button class="btn"
      (click)="RefreshClick()">
                  <span class="fa fa-sync"></span>
               </button>
            </div>
            <div>
               <input type="text" class="form-control">
            </div>
         </div>
         <div class="form-group">
            <button class="btn btn-primary btn-
      block">Login</button>
         </div>
      </div>
```

Ex:

- Generate a new service into “services” folder
  > ng g service data –skipTests

Data.service.ts

```
import { Injectable } from '@angular/core';


@Injectable({
  providedIn: 'root'
```

```
})
export class DataService {

  constructor() { }
  GetData(){
    return [
      {Name: 'JBL Speaker', Price: 4550.44},
      {Name: 'Nike Casuals', Price: 5000.44},
      {Name: 'Shirt', Price: 2300.44}
    ];
  }
}
```

Login.component.ts

```
export class LoginComponent implements OnInit {
products = [];
  constructor( private data: DataService) { }

  ngOnInit(): void {
    this.products = this.data.GetData();
  }
```

```
}
```

Login.component.html

```
<div>
      <table class="table table-hover">
         <thead>
            <tr>
               <th>Name</th>
               <th>Price</th>
            </tr>
         </thead>
         <tbody>
            <tr *ngFor="let item of products">
               <td>{{item.Name}}</td>
               <td>{{item.Price}}</td>
            </tr>
         </tbody>
      </table>
   </div>
```

Service Provider is defined with following injectors:

providedIn? : ‘root’ – The application level injector. It is providing service across all components in an application.

providedIn? : ‘platform’ – It is an Injector with singleton platform shared by all applications in workspace.

providedIn?: ‘any’ – Provides a unique instance in each lazy loading module. Other injectors are eagerly loaded.

FAQ: How to lazy load service?

A. providedIn: ‘any’

FAQ: Why a service in defined in constructor of component?

A. To configure DI [Dependency Injection]