Services and Providers

- A service is nothing more than a class in Angular 2. It remains nothing more than a class until we register it with the Angular injector.
- When you bootstrap your app, Angular creates an injector on the fly that can inject services and other dependencies throughout the app.
- You can register the service or the dependencies during when bootstrapping the app or when defining a component.

- If you have a class called `MyService`, you can register it with the Injector and then you can inject it everywhere:

```
1  bootstrap(App, [MyService]); // second param is an array of prov
```

- Providers is a way to specify what services are available inside the component in a hierarchical fashion.
- A provider can be a class, a value or a factory.
- Providers create the instances of the things that we ask the injector to inject.
- `[SomeService];` is short for `[provide(SomeService, {useClass:SomeService})];` where the first param is the token, and the second is the definition object.

- A simple object can be passed to the Injector to create a Value Provider:

```
1  beforeEachProviders(() => {
2    let someService = { getData: () => [] };
3    return [ provide(SomeSvc, {useValue: someService}) ]; // using
4  });
```

- You can also use a factory as a provider.

- You can use a factory function that creates a properly configured Service:

```
1  let myServiceFactory = (dx: DepX, dy: DepY) => {
2      return new MyService(dx, dy.value);
3  }
4
5  // provider definition object.
6  let myServiceDefinition = {
7      useFactory: myServiceFactory,
8      deps: [DepX, DepY]
9  };
10
11  // create provider and bootstrap
12  let myServiceProvider = provide(MyService, myServiceDefinition);
13  bootstrap(AppComponent, [myServiceProvider, DepX, DepY]);
```

- Defining object dependencies is simple. You can make a plain JavaScript object available for injection using a string-based token and the `@Inject` decorator:

```
1  var myObj = {};
2
3  bootstrap(AppComponent, [
4      provide('coolObjToken', {useValue: myObj})
5  ]);
6
7  // and you can inject it to a component
8
9  import {Inject} from 'angular2/core'
10 constructor(dx: DepX, @Inject('coolObjToken') config)
```