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:

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

- 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:

```
beforeEachProviders(() => {
  let someService = { getData: () => [] };
  // using `useValue` instead of `useClass`
  return [ provide(SomeSvc, {useValue: someService}) ];
});
```

- You can also use a factory as a provider.

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

```
let myServiceFactory = (dx: DepX, dy: DepY) => {
  return new MyService(dx, dy.value);
}

// provider definition object.
let myServiceDefinition = {
   useFactory: myServiceFactory,
   deps: [DepX, DepY]
};

// create provider and bootstrap
let myServiceProvider = provide(MyService, myServiceDefinition);
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:

```
var myObj = {};

bootstrap(AppComponent, [
  provide('coolObjToken', {useValue: myObj})
]);

// and you can inject it to a component

import {Inject} from 'angular2/core'
constructor(dx: DepX, @Inject('coolObjToken') config)
```