SERVICES

What is a service?

Ideally, a component's job is to enable the user experience and nothing more. A component should be a reusable piece of view logic.

A component can delegate certain tasks to services, such as fetching data from the server, validating user input, or logging directly to the console.

A service is just a class, decorated with the @Injectable pattern.

```
@Injectable({
  providedIn: 'root'
})
export class DashboardService {
    getNumberOfYears() {
    }
}
```

DEPENDENCY INJECTION

Components consume services; that is, you can inject a service into a component, giving the component access to that service class.

Here we are injecting a DashboardService into DashboardComponent, by defining it in the constructor.

```
export class DashboardComponent {
    widgetData;
    constructor(private dashboardService: DashboardService) {
    }
```

For this to work we need to specify **providers**. A provider is an object that tells an injector how to obtain or create a dependency.

```
@NgModule({
  providers: [DashboardService],
```

Dependency Injection lets us change a service implementation at runtime - for instance to replace a service with a mocked version during testing.

So when should we use services? The answer is everytime we can break out logic from a component.

A couple of common cases

- Logging
- Utils
- Fetching data from servers
- Validating data

EXERCISE 2

Lets break up the logic in our last exercise into a service