## What are services?

Services solve one big problem: They prevent us from copying logic over and over. Instead, they centralize business logic.

Also, they are very useful everywhere in our application.

That is because they can be easily requested via Dependency Injection.

They are also very useful if you want to use the same instance of a class everywhere in your class.

## Services are just classes

After all, services are just classes. Other than components, services may only contain logic. They should be completely separated from the view (anything visual). They also should only fulfill one purpose, following the single responsibility principle.

## Use Cases

The most common use case is I/O (Input/Output). To get more specific: HTTP requests. Generally, all HTTP requests in Angular are wrapped by a service. Why? Because with the help of Dependency Injection, our code stays highly maintainable. Here is an example:

Imagine you changed the route of your REST-Endpoint. Imagine you called that route in a billion different places. Good luck finding and replacing them all!

By wrapping our Http calls in a service, we know exactly where the change has to be made. And it is only one line affected.

## Singleton services

If a service is a singleton, there is only one instance of that service for the whole app.

With singleton services there is an alternative way of providing the service. Instead of using the providers-array of the @NgModule, we can tell the @Injectable where the service should be provided.

This is done by passing the provideIn option to the @Injectable decorator:



In this case, the service is provided in "root". That means it is provided application-wide. It can also be provided in a specific module.