Commands

* ng serve

build the application

start the development server

watch the source files

rebuild the application, if any changes occurred in those files

component in angular

 display data on the screen

listen for user input

and take action based on that input.

App Module

Angular needs to know how the pieces of your application fit together and what other files and libraries the app requires. This information is called metadata.

Some of the metadata is in the @[Component](https://v8.angular.io/api/core/Component) decorators that you added to your component classes. Other critical metadata is in [@NgModule](https://v8.angular.io/guide/ngmodules) decorators.

* Two-way data binding 🡺 That means data flows from the component class out to the screen and from the screen back to the class [(ngModel)] =” ”

Reminder : we need to import  “[FormsModule](https://v8.angular.io/api/forms/FormsModule)”  @ App Module class to apply “ngModel” directive

* Angular [property binding](https://v8.angular.io/guide/template-syntax#property-binding) 🡺It's a one way data binding, to give the parent Component control over the child Component

property binding [ ] lets you use the property value in a template expression.

* [class binding](https://v8.angular.io/guide/template-syntax#class-binding) 🡺makes it easy to add and remove a CSS class conditionally.

[class.selected]="hero === selectedHero"

* interpolation syntax {{ }}. Interpolation renders a property's value as text.

Services in angular

provider is something that can create or deliver a service; in this case, it instantiates the Service class to provide the service.

Reminder: provide the service at the root level, Angular creates a single, shared instance of Service and injects into any class that asks for it.

Injector is responsible for choosing and injecting the provider where the app requires it.

five common features of Angular's template syntax:

* \*[ngFor](https://v8.angular.io/api/common/NgForOf)
* \*[ngIf](https://v8.angular.io/api/common/NgIf)
* Interpolation {{ }}
* Property binding [ ]
* Event binding ( )

**Components** : define areas of responsibility in the user interface, or UI, that let you reuse sets of UI functionality

**app-root** is the application shell. This is the first component to load and the parent of all other components. You can think of it as the base page.

**Notice** the @[**Component**](https://v8.angular.io/api/core/Component)**()**decorator. This indicates that the following class is a component. It provides metadata about the component, including its selector, templates, and styles.

**The selector** is the name you give the Angular component when it is rendered as an HTML element on the page.

**The @**[**Input**](https://v8.angular.io/api/core/Input)**()** decorator indicates that the property value passes in from the component's parent, the product list component.

Usually, used when we create a “child component” and pass to it data from “parent component”

**Routing in angular**

**The Angular Router** displays components based on the browser's URL and your defined routes

The**[ActivatedRoute](https://v8.angular.io/api/router/ActivatedRoute)** is specific to each routed component loaded by the Angular Router. It contains information about the route, its parameters, and additional data associated with the route.

**Services in angular**

service is an instance of a class that can be made available to any part of your application using Angular's [dependency injection system](https://v8.angular.io/guide/glossary#dependency-injection-di).

Services are the place where you share data between parts of your application. For the online store, the cart service is where you store your cart data and methods.

The [HttpClient](https://v8.angular.io/api/common/http/HttpClient) service is what you inject into your services to fetch data and interact with external APIs and resources.

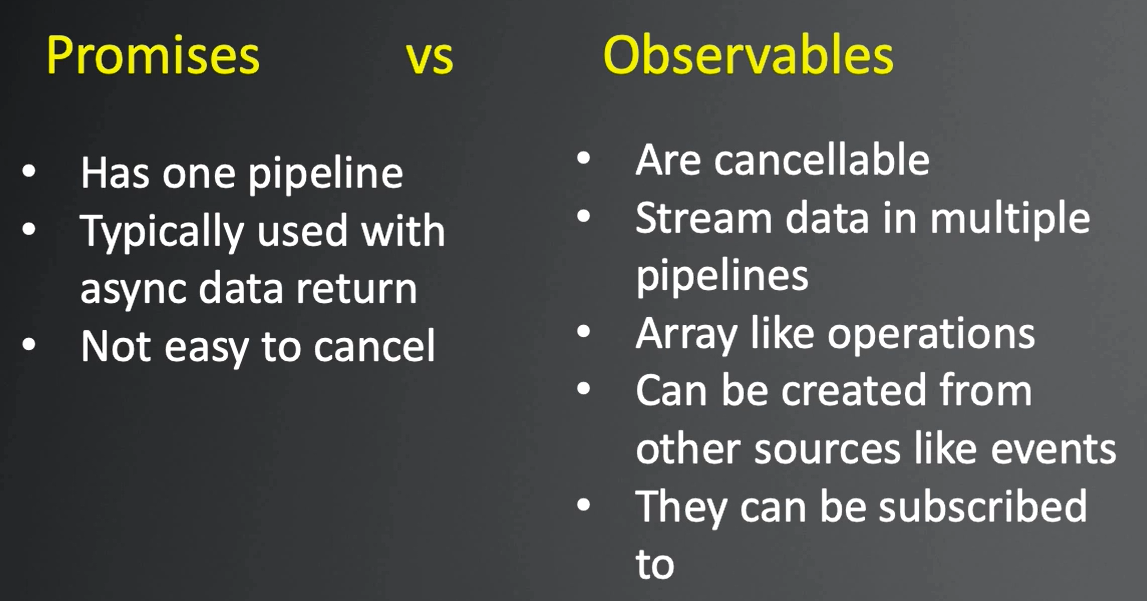
Observables

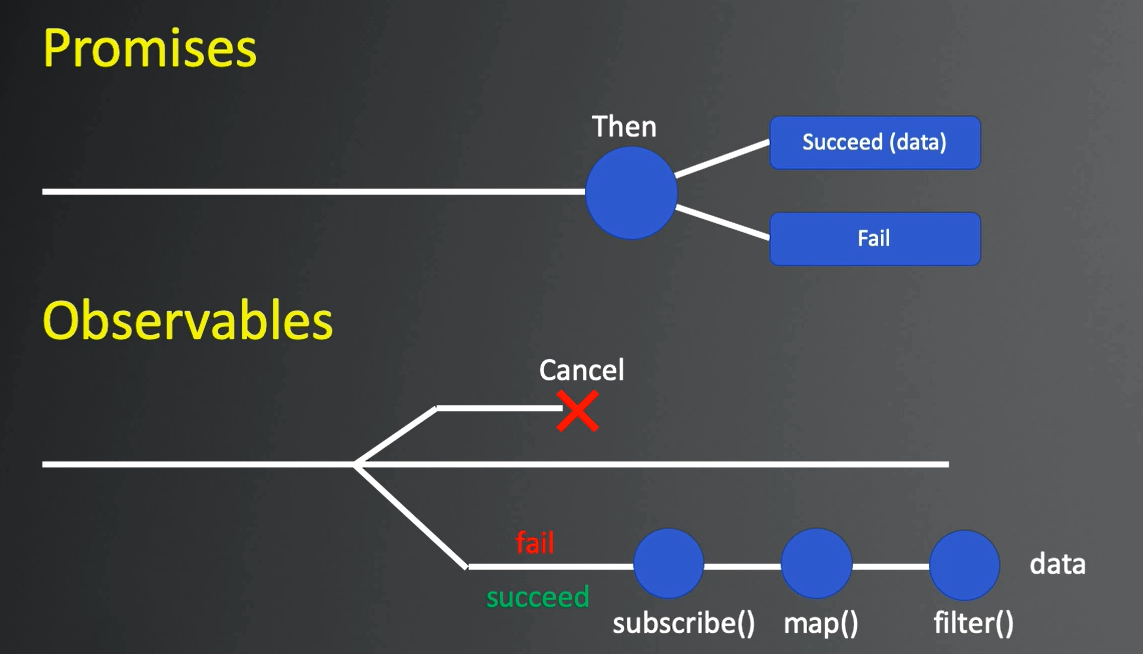
Observable 🡺 a sequence of items that arrived asynchronously over time

We can do various things with the observable [subscribe it, map it, filter data, shape data format ]

Promises 🡺 javascript is synchronously by its nature, stop the execution of the script and wait for the data

When we receive promise back we didn’t do something with the result





RxJS 🡺 helper library that angular uses to work with observable

* Reactive extensions for javascript
* Utility library for working with observables
* Uses the pipe() method to chain RxJS operators together

Typescript advantages

Strong typed 🡺 avoid big old messy javascript file (spagitti code) that doesn’t protect us against typos then have to spend a significant amount of time trying to figure out why some thing doesn’t work

Object oriented

Better intellisense

Access modifier 🡺 just developer friendly tools that can prevent our selves from doing something in a class we don’t really want to

Future javascript features 🡺 one of the things that angular uses to describe a class as an component is decorators

Catches silly mistakes in development 🡺 it’s going to catch typos as long as we are using its strong typing feature and it will prevent us from making silly mistakes

Tsconfig.json 🡺 for compiler ant its options

Usage of interface in angular

Instead of declare property with any[] @ component 🡺 use interface to avoid mistakes in template where we use the property and benefit form intellesanse typescript option

Note: use in type safety

File and