Event Emitters in Angular

Data flows into your component via property bindings and flows out of your component through event bindings.

If you want your component to notify his parent about something you can use the Output decorator with EventEmitter to create a custom event.

|  |
| --- |
| @Component({ |
|  | selector: 'add-todo', |
|  | template: `<input type="text" placeholder="Add todo.." [formControl]="control"> |
|  | <button (click)="add.next(control.value)">Add</button> |
|  | `, |
|  | }) |
|  | export class AddTodoComponent { |
|  | control : FormControl = new FormControl(""); |
|  | @Output() add = new EventEmitter(); |
|  | } |

We can use the Output decorator to label our property add as an event a component can fire to send data to its parent.

The parent can listen to an event like this:

<add-todo (add)="addTodo($event)"></add-todo>

Angular will subscribe to the add event and call the addTodo() method with the data when the component triggers the next() method.

# What is EventEmitter 😕

If you take a look at the source code, you are going to see something interesting.

|  |
| --- |
| export declare class EventEmitter<T> extends Subject<T> { |
|  | \_\_isAsync: boolean; |
|  | constructor(isAsync?: boolean); |
|  | emit(value?: T): void; |
|  | subscribe(generatorOrNext?: any, error?: any, complete?: any): any; |
|  | } |

The first thing you can learn from the source code is that you can pass a boolean to EventEmitter that will determine whether to send events in a synchronous or asynchronous way. ( The default is synchronous )

Because EventEmitters are Subjects ,we can use all the Rx goodness. For example, we want to emit an event only if we have a value.

@Output() add = new EventEmitter().filter(v => !!v);

That’s not all. We can also use any Subject that we want. Let’s try to use BehaviorSubject.

@Output() add = new BehaviorSubject("Awesome").filter(v => !!v);

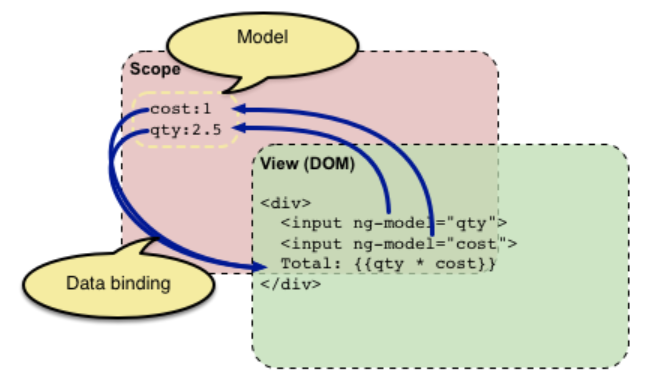
<https://netbasal.com/event-emitters-in-angular-13e84ee8d28c>

$ sign

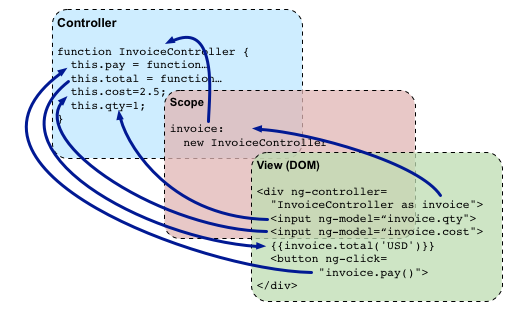
Words prefaced with a dollar sign ($) in Angular are custom objects within the Angular engine ,similarly to jQuery where $() is reference to a jQuery Object, which in turn typically references a DOM element.

/

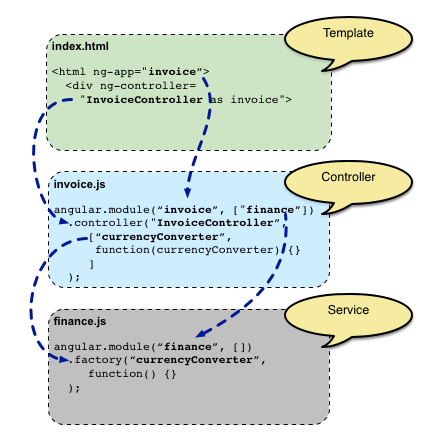




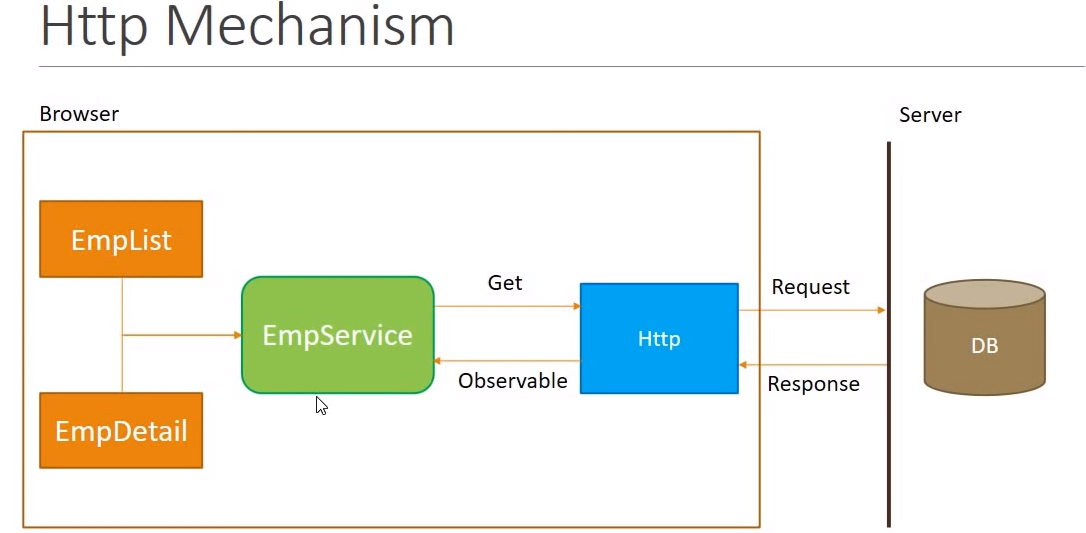


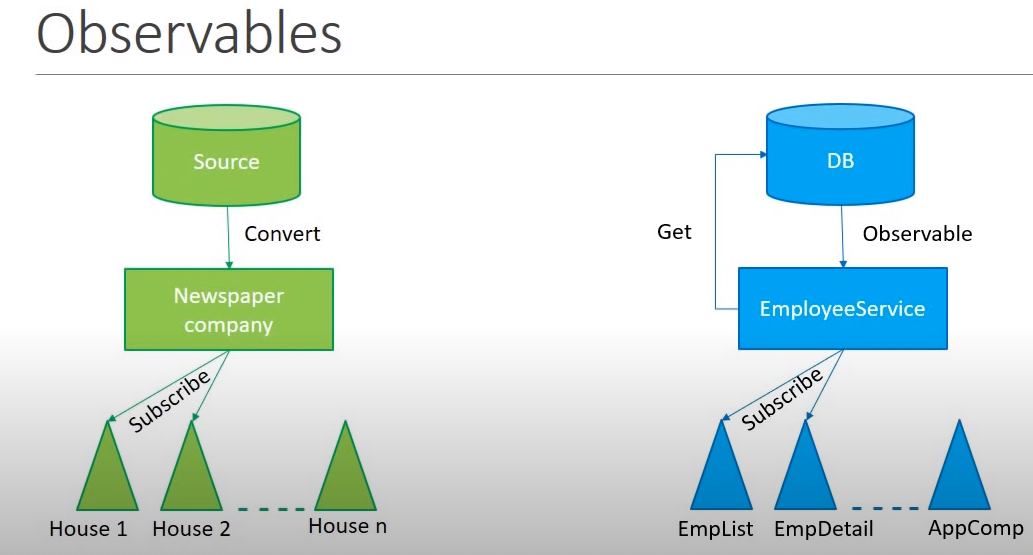




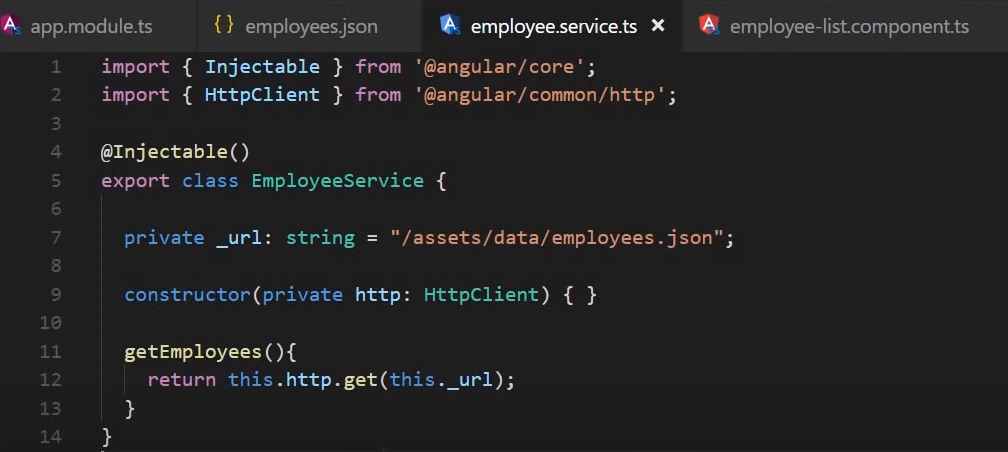


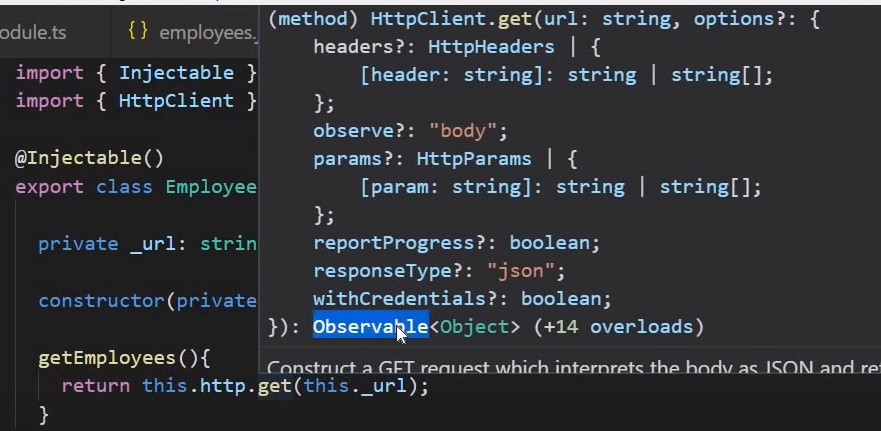
HttpClient and Observable

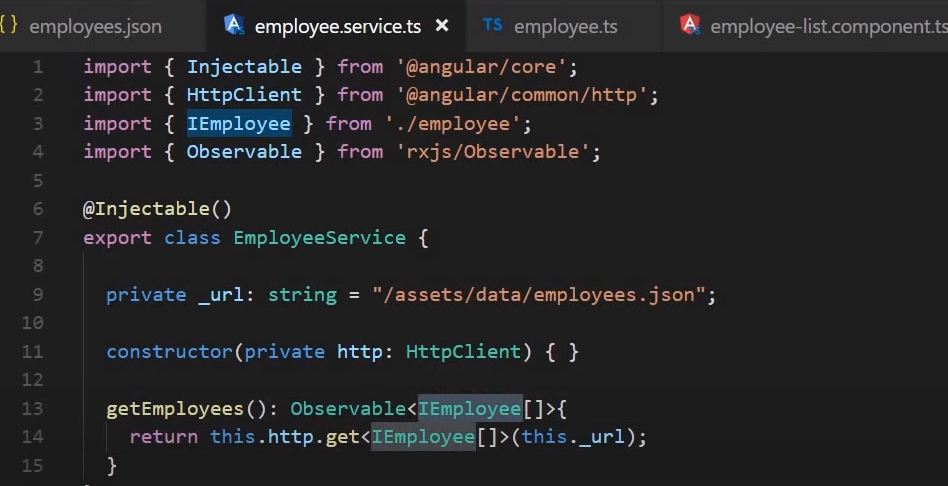


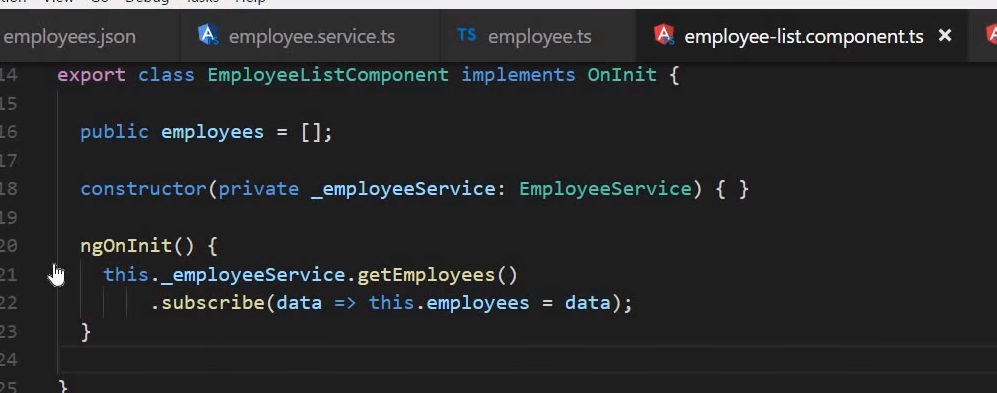




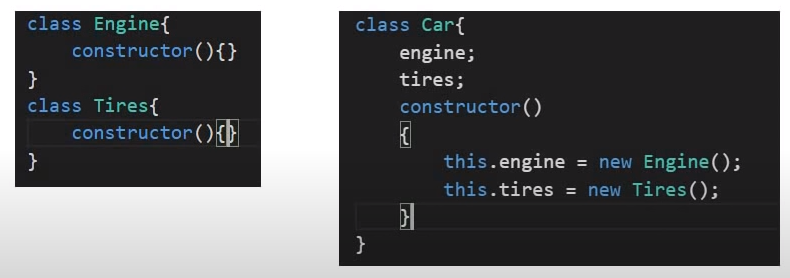




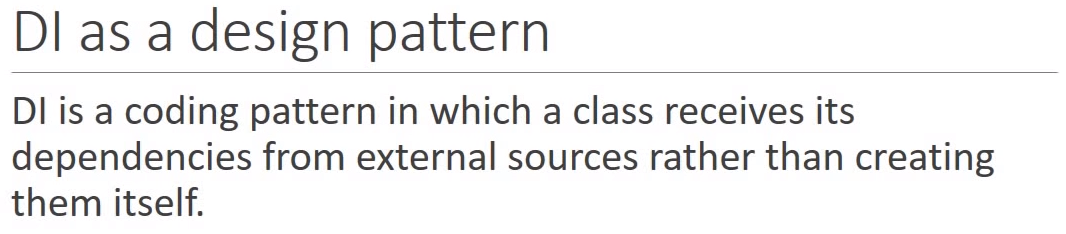




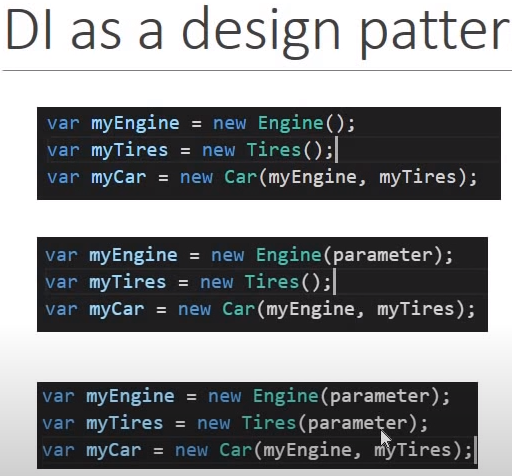
Dependency Injection

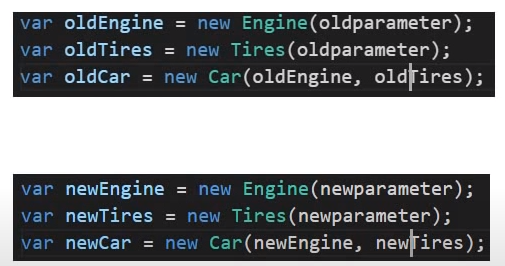


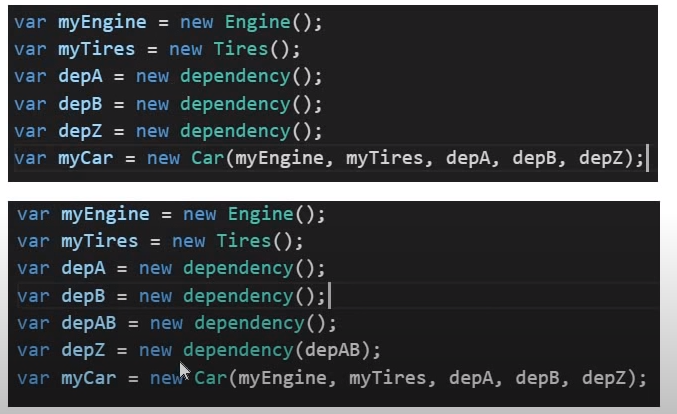


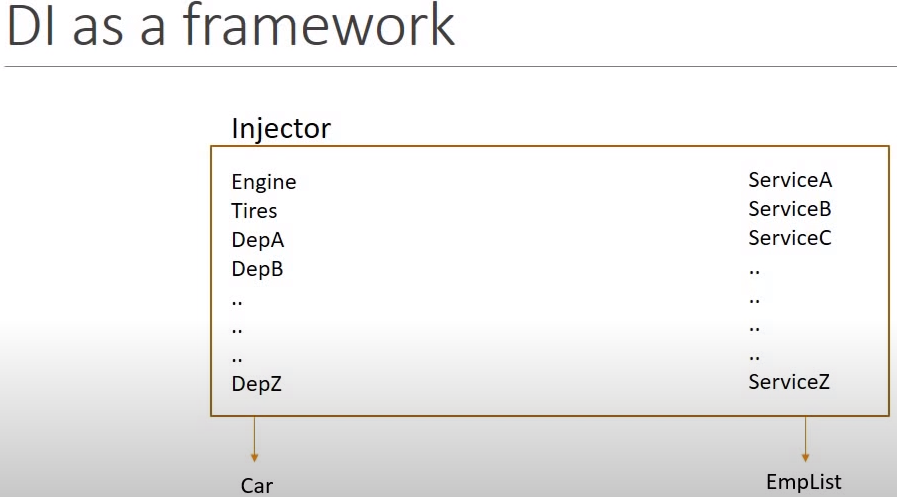




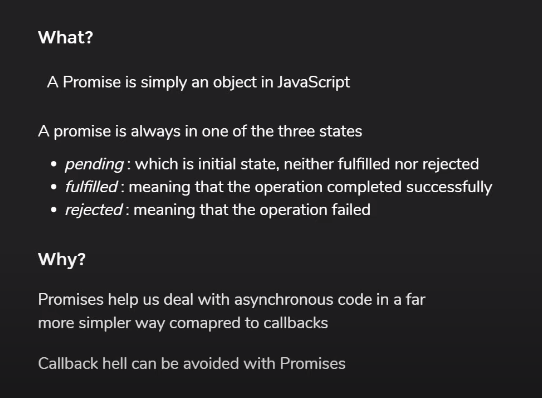


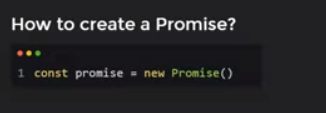


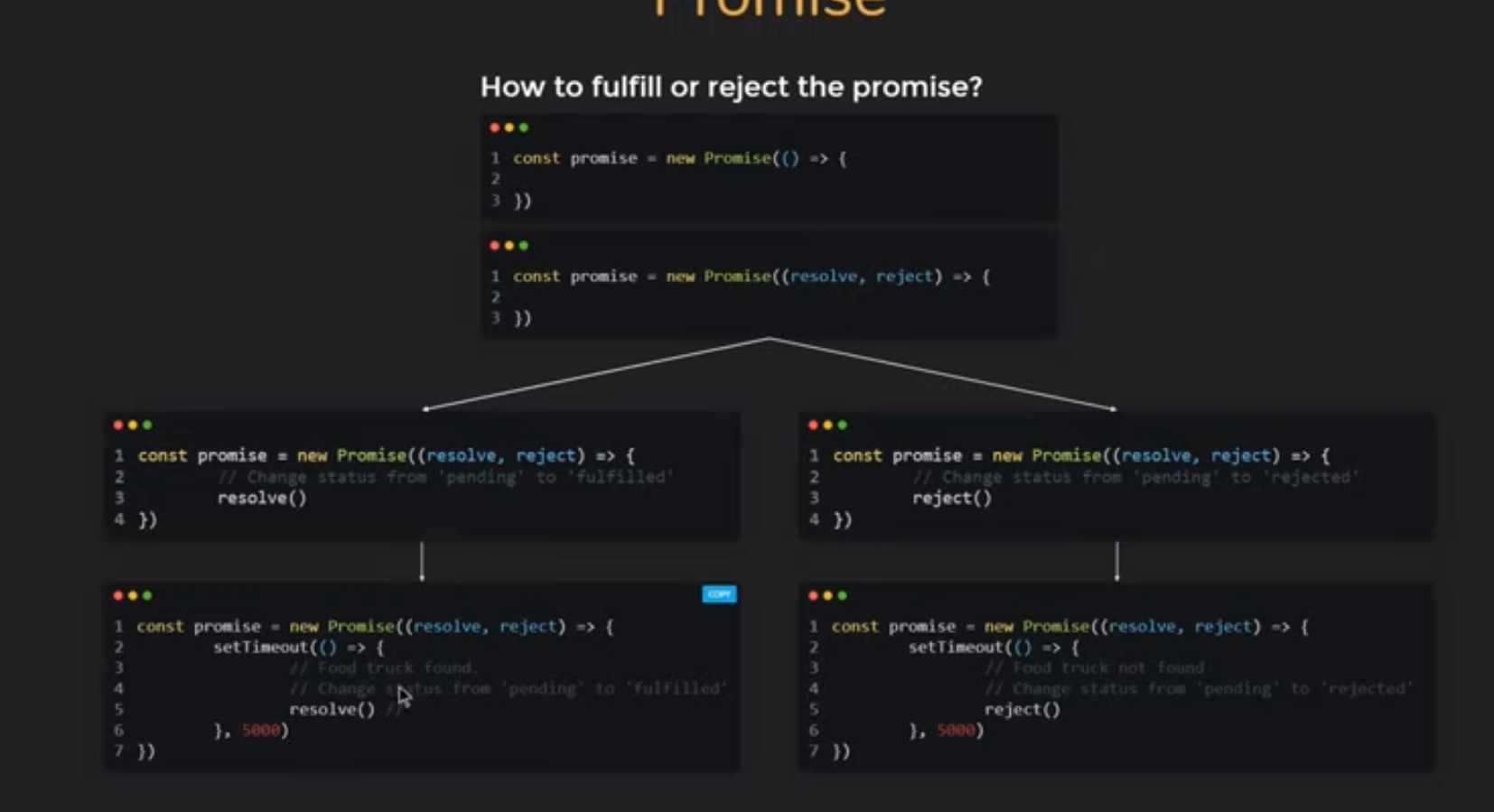


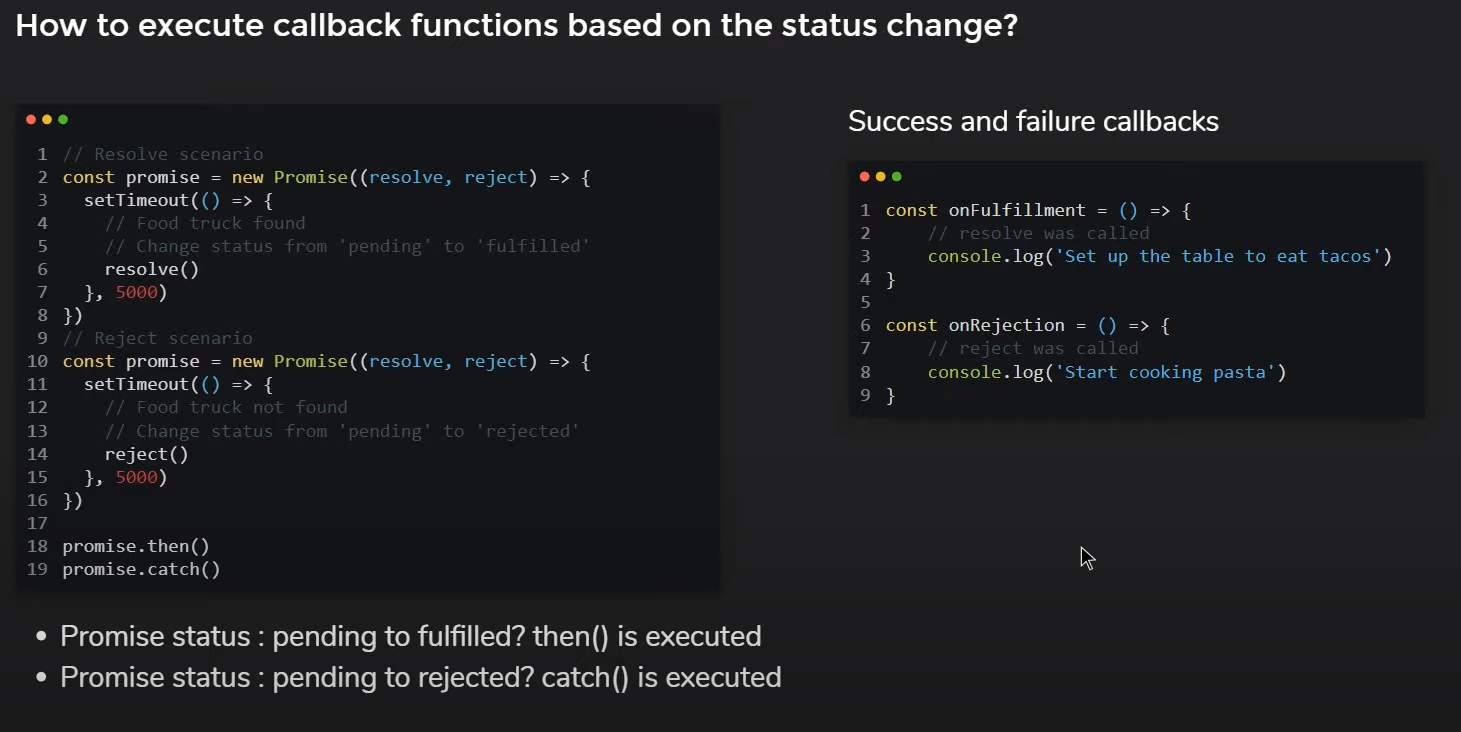


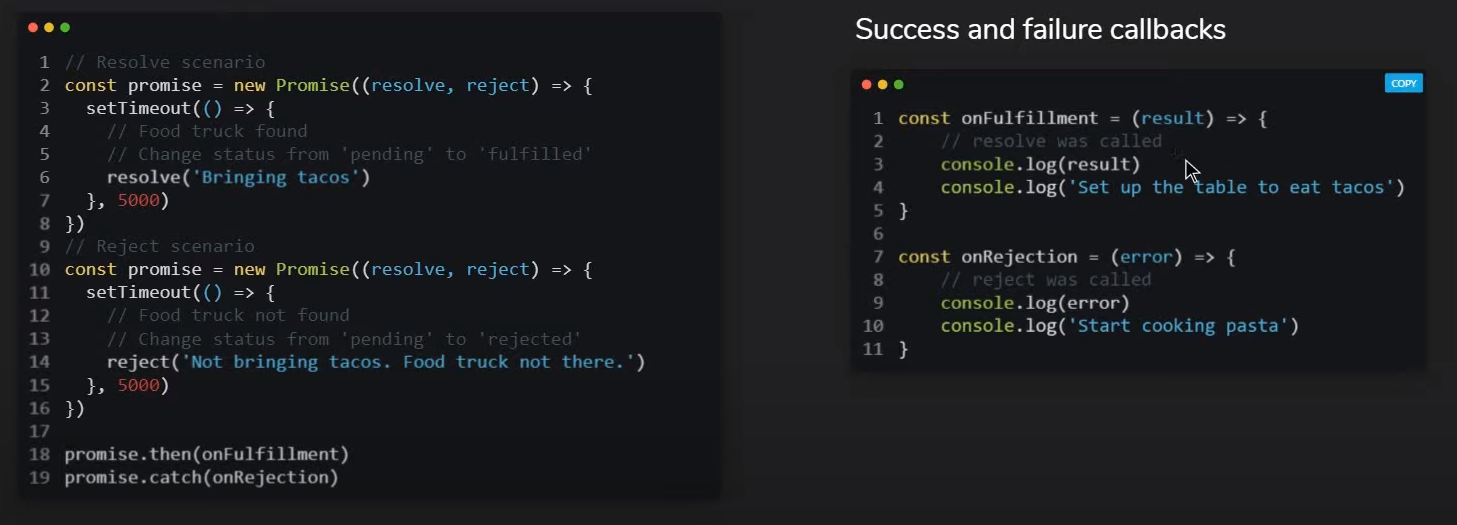
Promise











**Versioning of Packages**

Symantec versioning

**Moment 4.16.1.**

**4 here is major version number – This can break the existing functionality**

**16 here is minor version number – new feature but that new fix will not break current functionalities**

**1 is patch number –bug fix**

**NPM**

Npm gets shipped with node js so to have npm on machine you need to have node js

Node -v node version

Npm -v npm version

**Npm help**

Npm install -h : This command will tell you what you can do with npm install

Just type in npm install -h

You can even open the document

By

Npm help update --- This command will open the update command document in browser for you

**Take latest version of npm globally**

Npm install npm@latest -g

Do this with administrative privileges.

**Package.json**

Manage dependencies

--- package and its version

Scripts

--- Initial Build

**What npm init does??**

It actually creates the package.json in your project. So when you have create new project. Then you can use npm init to create the package.json and it have some default code in it.

When you do npm init it will ask you multiple question for creating a file . If you want to skip those question you can simply write npm init --yes, this will create the package with default settings.

You can set some default in package file. Watch this video [npm Tutorial for Beginners - 5 - package json Defaults](https://www.youtube.com/watch?v=r0CJwuCP2ck)

**Installing packages**

1. **When you are the one adding dependencies in the project**

Means you are installing a dependency and you need to have entry in the package.json file for that dependency

Now if you do

Moment is javacript librabry for datetime, using for an example

Npm install moment

This command will install the dependencies on your machine in node\_modules folder

But this will not make entry in package.json file for that you need to run command with save

i.e.

Npm install moment – save

Install/i is alias

--save/ -S capital S alias

**Existing project**

create node\_module and install all dependencies on machine

npm install

**What is difference between devdependencies and dependencies**

There are some package which you require only for development and not for production. Those are the ones which come under devdependencies.

To add in devdependencies we need to add -dev

Npm install moment --save-dev

Uninstall

To remove the dependency for node\_module and save that in package.json?

Npm uninstall moment --save-dev

Remove the dependency from node\_module and save that in package.json in devdependencies

**Install Globally**

**What is the need to install globally?**

There are some packages (example grunt and gulp) need to be used from command line. We need to install them globally so there binaries endup in path environment variables.

So to install globally

Npm install moment -g

The entry will **not come** in local package.json

**Where the global node\_module resides on machine?**

C/user/username/AppData/Roaming/npm/node\_modules

To uninstall

Npm uninstall moment -g

**Un/uninstall/remove/rm all does the same thing. These are different aliases.**

**Npm list** to list all packages – and its further dependencies

**Npm list –depth 0 (it Is double dash before depth)**

List with no depth, no hierarchy

To list global packages

Npm list –global true–depth 0

**Version**

We can update install any version but on top of that the important part is we can install combinations also as per need.

Check this video if you really wants to play with versions

<https://www.youtube.com/watch?v=ggc4mnfyBg4>

**“lodash”: “^4.14.1”**

**Here what is the meaning of ^?**

This tells npm that will installing package in exiting project take the keep the major version as same but take the latest version of minor and patch. Npm install by default does the same.

**“lodash”: “~4.14.1”**

**Here what is the meaning of ~?**

This tells npm that will installing package in exiting project take the keep the major version and minor version same but take the latest version of patch.

**“lodash”: “\*”**

Latest version of lodash in market will be installed.

**“lodash”: “4.14.1”**

This will install the exact version

**Update particular package version and save in package.json**

Npm undate lodash --save

Just update dev dependencies

Npm update –dev –save-dev

To update everything

Npm update

Globally

Npm update -g

Particular package globally

Npm update -g gulp

**In your project you have installed multiple packages but you are not using all of them. So to just delete the packages which you are not using in your project, the command is prune**

Npm prune

To see all shortcut **https://docs.npmjs.com/misc/config**

Npm scripts

“scripts”:{

“start”: “echo Starting... && ng serve”

}

Run npm start will execute echo Starting... && ng serve

Npm is node package module

And package.json has all the information which npm will use

Means if I say npm start it means whatever written in package.json file in opposite to start will run.

Usually will write ng serve which will serve the website to you

Sometimes you will see that dotnet run application is not running in that case think about it

Comment the npm start command in startup file and run the command in console to start the angular application

And then run you dot net code

This way the application will run.