Part III — Angular JS

Based on lectures by Brian

Notes taken by Dexter Chua

Lent 2017-2018

These notes are not endorsed by the lecturers, and I have modified them (often significantly) after lectures. They are nowhere near accurate representations of what was actually lectured, and in particular, all errors are almost surely mine.

Contents

1 Introduction to Angular

1.1 Introduction

Angular is an open source JavaScript framework maintained by Google. We use Angular to create single page applications (SPA).

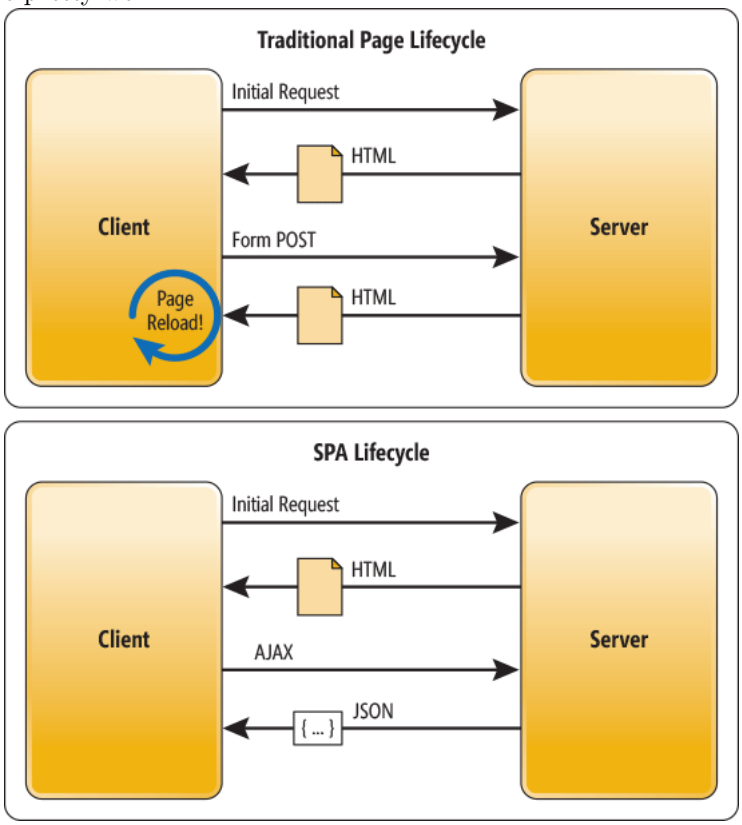
1.1.1 Single Page Application

SPA are web applications that work like a desktop application. That means upon first loading the application the user downloads all the JavaScript, HTML and CSS to render any part of the application. Any extra data needed will be fetched dynamically when the user is using the application. However a total refresh is not necessary upon most circumstances.

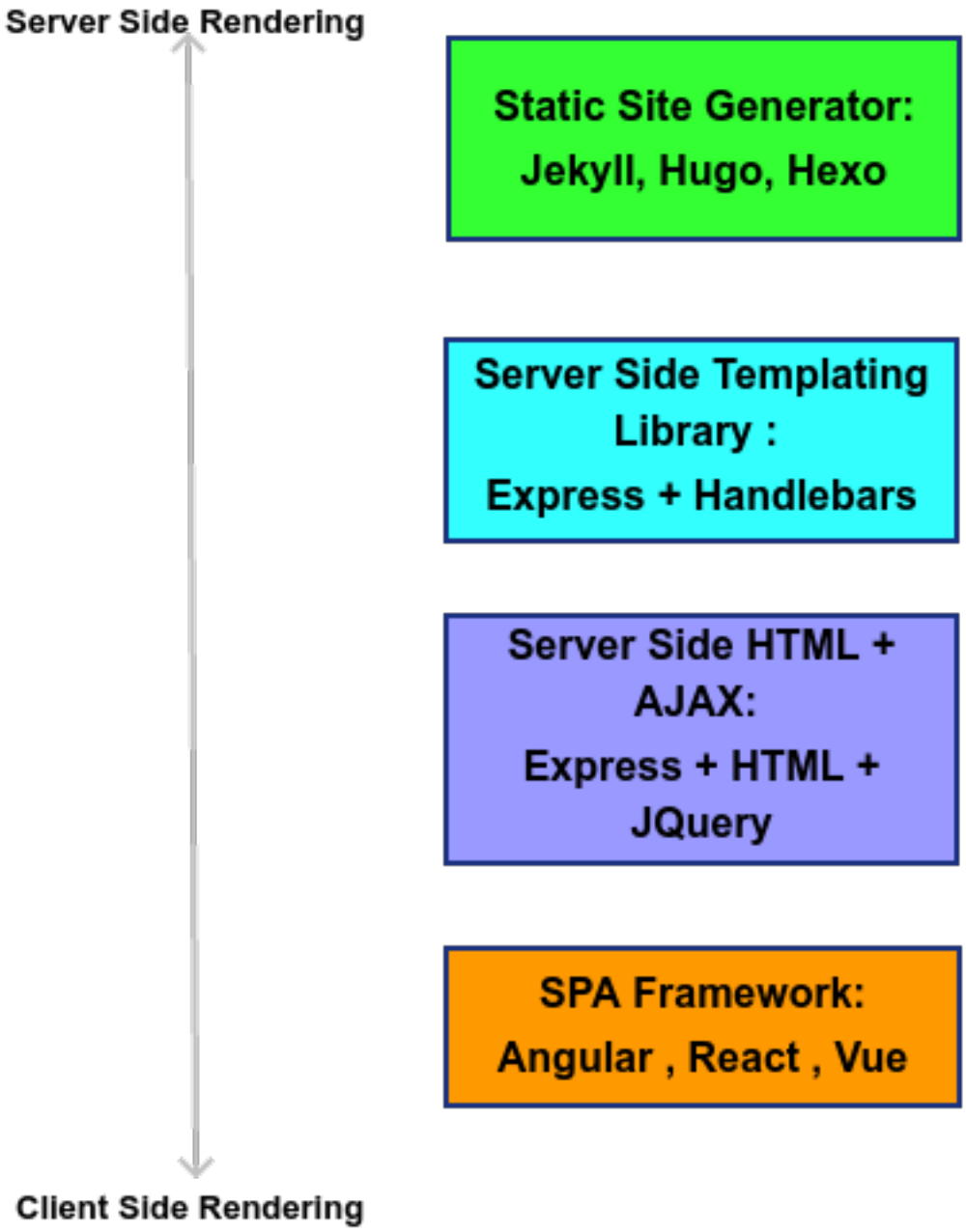
We can summarize the difference between a SPA and a normal website as follow: normal website reloads on every click. SPA loads most HTML/CSS from the beginning and loads the necessary JSON while needed.

Therefore , you can regard the applications you created before as Multiple page applications. The application we can create with Angular is thus the contrary, Single Page Application.

Here is a diagram from Microsoft Documentation that summarize the difference pretty well

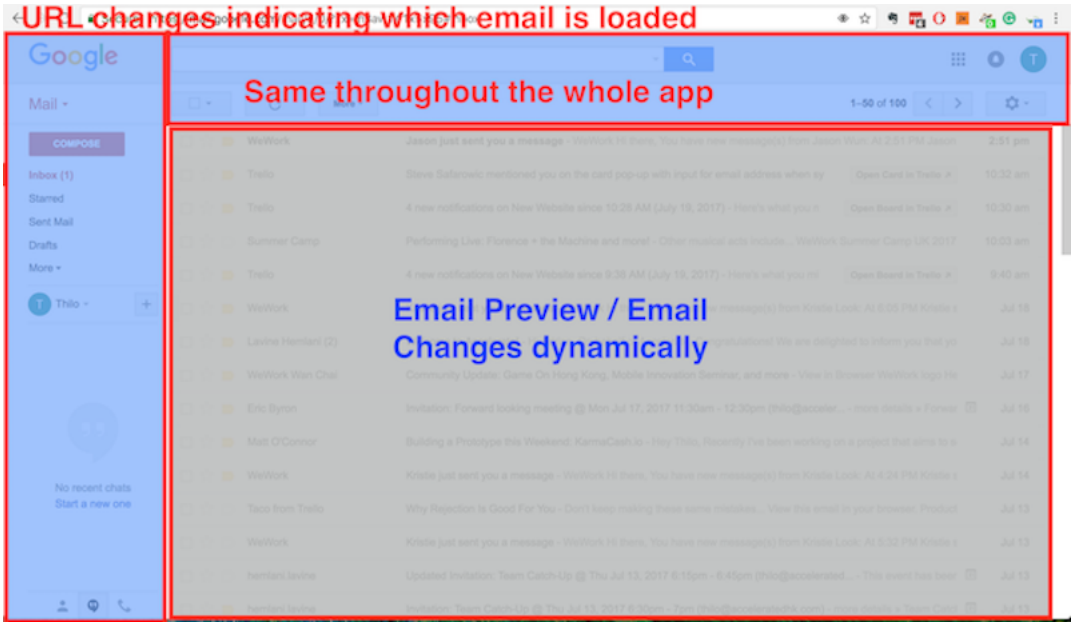


The rendering of SPA and MPA also differs. Here is a diagram showing the difference in the rendering process.



1.1.2 Gmail

One of the prominent example of a SPA is gmail. You have one page, which in its basic structure always looks the same. We can dynamically load any email but the basic structure doesnt change just like many desktop applications (like MS Office). This set the standard for modern web development. Almost all the big websites these days makes use of SPA. Just take a good look at YouTube or Facebook.

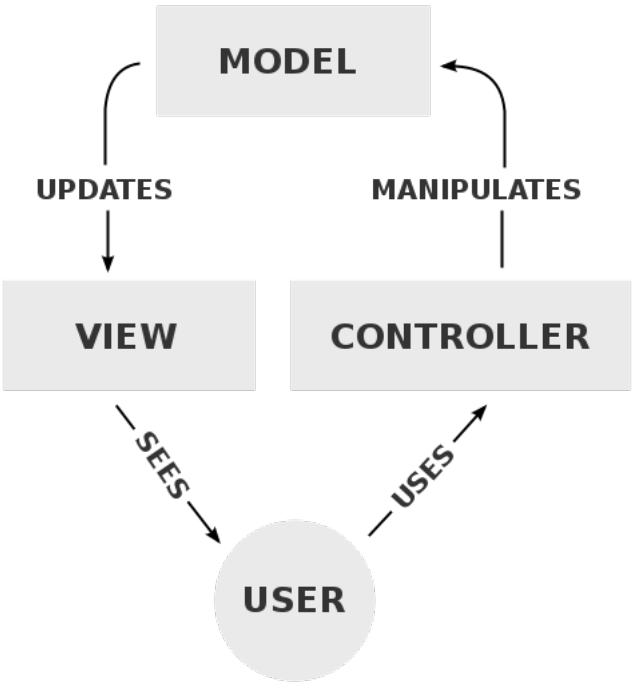


1.1.3 Different types of SPA Franework

1.1.4 MVVM vs MVS

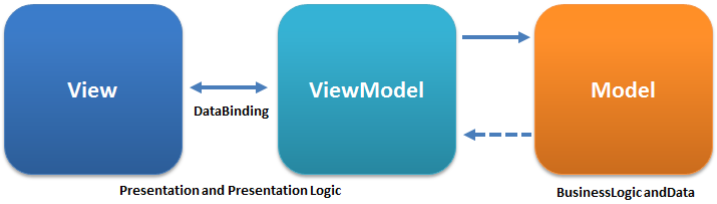
You may have heard the following two architectures. Model-View-Controller which specifies the three components in the architectures Model, View and Controller.

Here is the architecture for Model-View-Controller



In MVC, the users interacts with the controller which manipulates the model which is the data. The data is updated to view at the end.

Here is the architecture for Model-View-ViewModel



The biggest difference between MVVM and MVC lies in the reliance of Data Binding in MVVM. The data in ViewModel is bound to the view by the framework instead of updating through templates as in MVC.

For their application domains, nowadays MVC is mostly used in backend framework while MVVM is widely used in frontend framework.

1.2 Installation

The best way to create a new Angular project is through the Angular cli. So lets do exactly that.

To install the cli run:

```
npm install -g @angular/cli
```

Afterwards navigate to the folder where you save your coding projects. There we run:

```
ng new sample-app
```

As you probably figured out yourself ng is short for angular. new creates a new project and sample-app is the project name. Youll see in the terminal that angular creates a whole bunch of files and then installs a few things.

This is exactly the reason why we use the cli. Otherwise we would have to create those files manually and set up everything by ourselves, which would keep us pretty busy for a few hours (especially if we have no idea what we are doing).

Once that has finished lets go check out our app.

```
cd sample-app
ng serve
```

ng serve tells the cli to compile all the files and create a development server to serve the files from. Compiling in this process means that the Angular cli takes all the JavaScript files (our App will consist of many JS files) and bundles them together in a correct order. It will also make sure that our JS is readable by older browsers by changing it back to ES5 standards. These files are then served via a development server, the only thing special about this server is that we dont have to set it up ourselves and that it recompiles automatically every time we change something in our code.

So lets check out the project, go to http://localhost:4200

There you go your first Angular app up and running.

1.2.1 Special for Bash on Ubuntu

If you encounter error while you are installing angular cli through Bash On Ubuntu , you can try to use PowerShell to install instead. PowerShell should be installed on your Windows by default.

1.2.2 Folder structure

Lets take a better look at the application. Open everything with your editor. Youll notice a folder structure a bit similar to this:

```
.
          angular-cli.json
          e2e
          karma.conf.js
          node\_modules
          package.json
          package-lock.json
          protractor.conf.js
          README.md
          src
          tslint.json
```

e2e is a folder for end-to-end testing. We can ignore that for the moment.

node_modules you should know, but you might rightly wonder. Why are the node modules when we are developing a frontend application? As explained earlier angular actually has a built-in bundler which bundles the javascript files in the src folder for us. As a result, we can use the require() function just like what we did in the backend.

src is the folder we are actually going to working in. You might have correctly guessed that src stands for source. Here all the source files are located that we later combine into our Angular application.

package.json is the config file of the npm repository. So you can just install normal node packages like what you have been doing.

Other files are mainly set up files, we dont need to worry about at this moment.

1.2.3 The src folder

The src folder contains all the source file of this angular project including JS/CSS/HTML. The file structure of your src folder should be as follow,

```
.
        app
        assets
        environments
        favicon.ico
        index.html
        main.ts
        polyfills.ts
        styles.css
        test.ts
        tsconfig.json
```

As you can see, we now have some files ended with *.ts instead. They are the typescript files. Angular makes heavy use of typescript. We will cover more typescript in the next section.

There are several files and folder which are interesting to talk about:

app: The folder that contains all the typescript files of your application. It also includes the template.

assets: The assets folder which stores the images and videos of your application

environment: The environment of this application. Angular by default distinguish between production environment or non-production environment.

index.html: The single page that you client would actually load. if you read the content of it, it is only a empty body with a Loading... message , because everythingelse is loaded by javascript.

main.ts: The entry point of your application. It loads all of the component from app and start rendering your application.

styles.css: The global css that styles is going to applied to all elements in your application. Note. Angular applications usually have many css files because each Angular component can have its own css file, where the css is localised and affecting elements within that component only - more on this later.

test.ts: The entry point of your test case. When you run npm test which is basically ng test.

1.2.4 Plugin

If you are using visual studio code, we recommend you to install this angular plugin to make your developement experience better.

You will probably find Augury helpful for your development, it helps you with debugging your code and will make development run a lot smoother.

1.3 TypeScript

When we have been saying that the Angular cli takes our JavaScript files and combines them into one, we didnt talk the whole truth there. While it is true that the compiler takes some JavaScript files and combines them together, we ourselves dont develop in JavaScript.

We write our Angular logic in TypeScript (TS). TS is a superset of javascript and is going to be compiled to javascript to run in a browser.

1.3.1 Installation

ng-cli already installs the required compiler for you. So you only have to install the editor support to start coding typescript.

If you are using an editor that supports Typescript by default, you can start coding typescript right away. Otherwise you have to install the supporting packages specifically.

If you want to try out typescript outside your angular project, you can also install the typescript compiler by the following command

```
npm install -g typescript
```

You can then compile the typescript file with the following command

```
  tsc example.ts
```

You can also see real-time compiling in the typescript Playground page

1.3.2 Syntax

The syntax of Typescript is very similar to javascript with the exception of type declaration. You can always refer to the documentation in Typescript official website here

Here is a piece of javascript code. It is also a piece of valid typescript code.

```
function Greeter(greeting) {
    this.greeting = greeting;
}

Greeter.prototype.greet = function() {
    return "Hello, " + this.greeting;
}
let greeting = new Greeter("world");
let greeting2 = new Greeter(1234);
```

You can see the property greeting could be an integer or a string. But for typescript , you can add type to constraint the type of the parameter

```
function Greeter(greeting: string){
    this.greeting = greeting;
}

Greeter.prototype.greet = function() {
    return "Hello, " + this.greeting;
}

let greeter = new Greeter("world");
// Now this one fail!!
let greeter2 = new Greeter(1234)
```

The following error is shown

```
test.ts(11,32): error TS2345: Argument of type '1234' is not
    assignable to parameter of type 'string'.
```

The compiler tsc does the type-checking and discover what is the mismatch in the type. The ability of having type-checking in compiling phase instead of runtime is a property of static programming language to which typescript belongs.

1.4 TS node

2 Components

2.1 Components

2.2 creating component

2.3 Data Binding

2.4 Templates and Models

2.5 Breaking down Components

3 Template

3.1 Directives

3.2 Pipes

4 Forms

4.1 Template Driven Form

4.2 Model Driven Form

5 Assignment 1

6 Service

6.1 Services

6.2 Injectors

7 Routing

8 Http

8.1 HTTP

8.2 Authentication

9 Other Topics

9.1 Testing

9.2 Lifecycle Hooks

9.3 Full Stack Development Environment

9.4 JSON Web Token

10 Assignment 2

11 Introduction to Ionic

11.1 Installing

12 Ionic Components

12.1 Pages

12.2 Styling

13 Ionic Testing

13.1 Depoly on IOS

13.2 Depoly on Android

14 Ionic Native

15 Assignment 3

16 Extra Topics

16.1 TCP/IP Model

16.2 Angular Depolyment

16.3 OO Design Principles and Clean Code