Angular Introduction: What It Is, and Why You Should Use It

- A general overview of a very popular and widely used client-side framework
- This Angular introduction is mostly aimed at newcomer developers
- A JavaScript framework is a kind of buzzword these days

Why Do I Need a Framework?

- A JavaScript (or client-side) framework is a technology providing us the right tools to build a web application while also defining how it should be designed and how the code should be organized.
- Most JS frameworks these days are **opinionated**, meaning they have their own philosophy of how the web app should be built and you may need to spend some time to learn the core concepts.
- Other solutions, like **Backbone**, do not instruct developers on how they should craft the project, thus some people even call such technologies simply **libraries**, rather than frameworks.
- JavaScript frameworks emerged not that long ago.
  - Websites were built with poorly structured JS code (in many cases, powered by jQuery).
  - Client-side UIs have become more and more complex, and JavaScript lost its reputation as a “toy” language.
  - Modern websites rely heavily on JS and the need to properly organize (and test!) the code has arisen.
  - Client-side frameworks have become popular and nowadays there are at least dozen of them.

Angular Introduction: What Angular IS

- AngularJS used to be the “golden child” among JavaScript frameworks
- Introduced by Google corporation in 2012.
  - It was built with the **Model-View-Controller** concept in mind, though authors of the framework often called it **“Model-View-*”** or even **“Model-View-Whatever”**.
- The framework, written in pure JavaScript, was intended to decouple an application’s logic from DOM manipulation, and aimed at dynamic page updates.
- It wasn’t very intrusive: you could have only a part of the page controlled by AngularJS.
- Specifically, a concept of data binding was introduced that meant automatic updates of the view whenever the model (data) changed, and vice versa.
-

The idea of **directives** was presented, which allowed inventing your own HTML tags, brought to life by `JavaScript` .

> For example, you may write:

```
<calendar></calendar>
```

- This is a custom tag that will be processed by AngularJS and turned to a full-fledged calendar as instructed by the underlying code.
- Another quite important thing was **Dependency Injection**, which allowed application components to be wired together in a way that facilitated reusable and testable code.
- More to AngularJS, but we're not going to discuss it
- AngularJS became popular very quickly and received a lot of traction.
- Its maintainers decided to take another step further and proceeded to develop a new version which was initially named **Angular 2** (later, simply Angular without the "JS" part).
  - It's no coincidence the framework received a new name: actually, it was *fully re-written and redesigned*, while many concepts were reconsidered.
- The first stable release of Angular 2 was published in **2016**, and since then AngularJS started to lose its popularity
- One of the main features of Angular 2 was the ability to develop for multiple platforms: **web**, **mobile**, and **native desktop**
- To make things even more complex, by the end of 2016, **Angular 4** was released. "So, where is version 3?", you might wonder. I was asking the same question, as it **appears that version 3 was never published at all!**
- As explained in the official blog post, maintainers decided to stick with the semantic versioning since Angular 2.

> Following this principle, changing the major version (for example, "2.x.x" becomes "3.x.x") means that some breaking changes were introduced. The problem is that the Angular Router component was already on version 3. Therefore, to fix this misalignment it was decided to skip Angular 3 altogether.

Angular Introduction: the Advantages of Angular

- Why Angular?
  - It's supported on various platforms (web, mobile, desktop native), it's powerful, modern, has a nice ecosystem:

    -
      ```
      Angular presents you not only the tools but also design patterns to build your
      project in a maintainable way. When an Angular application is crafted properly, you
      don't end up with a tangle of classes and methods that are hard to modify and even
      harder to test. The code is structured conveniently and you won't need to spend much
      time in order to understand what is going on.
      ```

    -
      ```
      It's JavaScript, but better. Angular is built with TypeScript, which in turn relies
      on JS ES6. You don't need to learn a totally new language, but you still receive
      features like static typing, interfaces, classes, namespaces, decorators etc.
      ```

    -
      ```
      No need to reinvent the bicycle. With Angular, you already have lots of tools to
      start crafting the application right away. You have directives to give HTML elements
      dynamic behavior. You can power up the forms using FormControl and introduce various
      ```

validation rules. You may easily send asynchronous HTTP requests of various types. You can set up routing with little hassle. And there are many more goodies that Angular can offer us!

- Components are decoupled. Angular strived to remove tight coupling between various components of the application. Injection happens in NodeJS-style and you may replace various components with ease.

- All DOM manipulation happens where it should happen. With Angular, you don’t tightly couple presentation and the application’s logic making your markup much cleaner and simpler.

- Testing is at the heart. Angular is meant to be thoroughly tested and it supports both unit and end-to-end testing with tools like Jasmine and Protractor.

- Angular is mobile and desktop-ready, meaning you have one framework for multiple platforms.

- Angular is actively maintained and has a large community and ecosystem. You can find lots of materials on this framework as well as many useful third-party tools.

Angular Introduction: Angular’s Complexities

- Angular is quite a big and complex framework with its own philosophy, which can be challenging for newcomers to understand and get used to.
- Learning the framework’s concepts is not the only task, however; on top of this, you also have to be comfortable with a handful of additional technologies:
  - It’s recommended to code Angular apps in TypeScript, so you must understand it. It is possible to write the code with modern JavaScript (ES6), though I rarely see people doing this.
  - TypeScript is a superset of JavaScript, so you’ll need to be comfortable with it as well.
  - It’s a good idea to get the grasp of the Angular CLI to speed up the development process even further.
  - Node’s package manager npm is used extensively to install Angular itself and other components, so you’ll need to be comfortable with that as well.
  - Learning how to set up a task runner like Gulp or Grunt can come in really handy, as there can be lots of things to be done before the application is actually deployed to production.
  - Using minifiers (like UglifyJS) and bundlers (like Webpack) is also very common these days.
  - While developing the app, it’s vital to be able to debug the code, so you should know how to work with debugging tools like Augury.
  - Of course, it’s very important to test Angular applications, which can become very complex. One of the most popular testing tools out there are called Jasmine (which is a framework for testing) and Protractor (which is used for end-to-end testing).

One last thing worth mentioning is that sometimes using Angular for an app may be overkill. If you have a small or medium-sized project without any complex user interfaces and interactions, it may be a much better idea to stick with plain old JavaScript. Therefore, it’s very important to assess all the requirements

Tools to Help Development

- Augury Browser extension
- VSCode: Angular Essentials -> John Papa