Introduction to Angular Framework

### Introduction to Angular **Framework**, History & Overview

### Environment Setup

### Angular CLI, Installing Angular CLI

### NPM commands & package.json

### Bootstrapping Angular App, Components, AppModule

### Project Setup, Editor Environments

### First Angular App & Directory Structure

### Angular Fundamentals, Building Blocks

### MetaData

## Introduction to Angular Framework, History & Overview

**What is Angular?**

It is TS based framework which allows you to create Single Page Application.

Angular is a **TypeScript-based open-source** front-end platform that makes it easy to build applications with in web/mobile/desktop.

The major features of this framework such as declarative templates, dependency injection, end to end tooling, and many more other features are used to ease the development.

**Angular JS Vs Angular**

| **AngularJS** | **Angular** |
| --- | --- |
| It is based on **MVC architecture** | This is based on **Service/Controller** |
| This uses use J**avaScript** to build the application | Introduced the **typescript** to write the app. |
| Based **on controllers** concept. | This is a **component** based UI approach |
| **Not a mobile friendly** framework. | Developed considering **mobile platform.** |
| Difficulty in SEO friendly application development | Ease to create SEO friendly applications |

**SPA?**

Use Case: Traditional Approach of Web Experience

* the constant full page reloads
* also due to the network back and forth trips to the server to fetch all that HTML.

**Solution is SPA?**

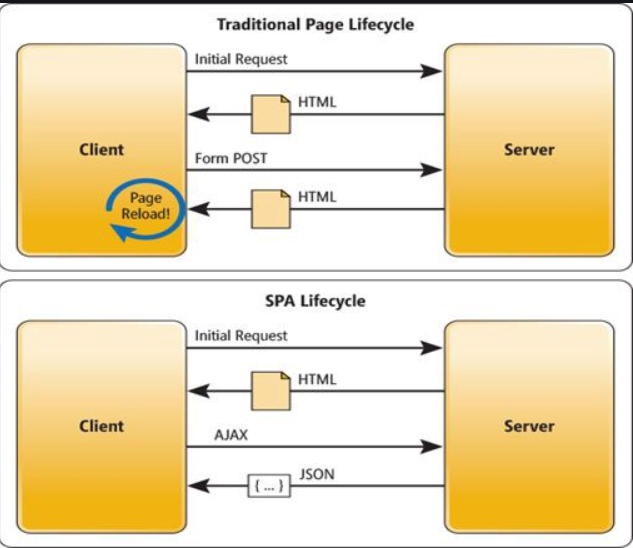
SPA is a website design approach where each new page's content is served not from loading new HTML pages but generated dynamically through JavaScript's **ability to manipulate the DOM elements on the existing page itself.**

**SPA Approach:** Allows the user to continue consuming and interacting with the page while new elements are being updated or fetched, and can result in much faster interactions and content reloading.

**Advantages of SPA?**

* much-improved experience to the user
* Feel faster because less bandwidth is being used, and no full page refreshes are occurring as the user navigates through the application. only data gets sent over the wire as a JSON payload or some other format. But no HTML or CSS gets sent anymore over the wire after the application is running.
* The application will be much easier to deploy in production, at least certainly the client part: all we need is a static server to serve a minimum of 3 files: our single page index.html, a CSS bundle, and a Javascript bundle.
* The frontend part of the application is very simple to version in production, allowing for simplified deployment and rollbacks to previous version of the frontend if needed.
* Ideal as a base for future mobile app development.
* The key point to understand how single page applications work is the following: instead of converting data to HTML on the server and then send it over the wire, in a SPA we have now moved that conversion process from the server to the client. The conversion happens last second on the client side, which allow us to give a much improved user experience to the user.

Describe SPA

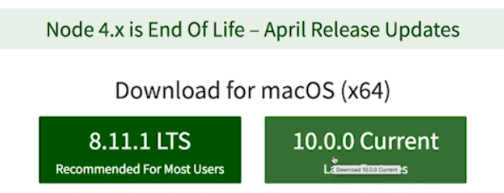
****

## Environment Setup

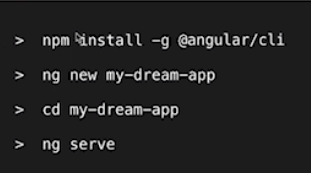
## Angular CLI, Installing Angular CLI

**Setup for Angular**

1. Install Node.js - Node bundles and Optimize our project and uses npm to mange dependencies.



##### Install Angular CLI

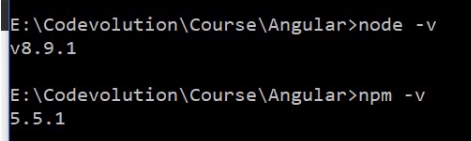


@latest will install latest version of angular

##### Verify

Check version of NodeJS > node -v

Check Angular Version> npm -v



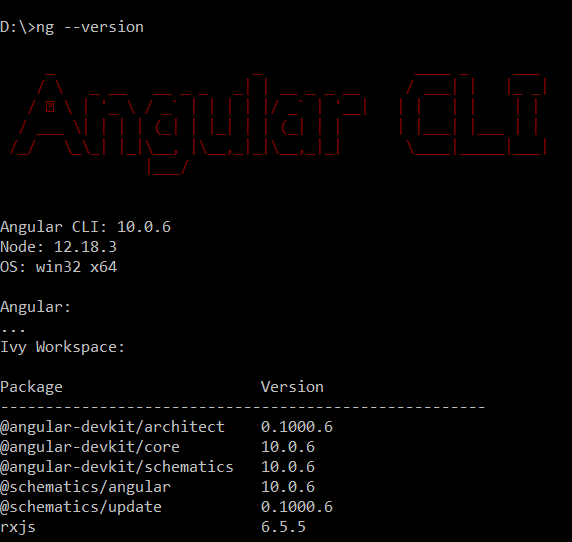
Check path of npm > npm config get prefix



Check Angular version> **ng --version**

To Update Angular : ng update @angular/cli @angular/core

##### Check version of Angular



## NPM commands & package.json

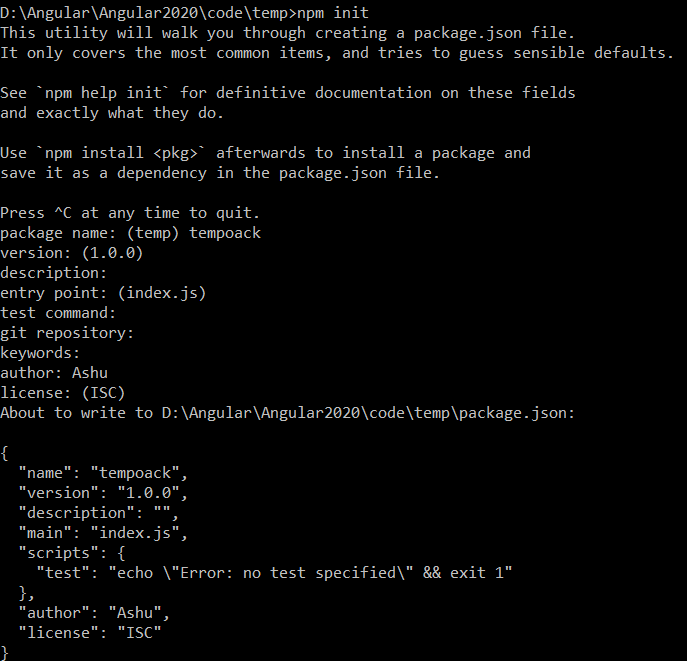
<https://medium.com/beginners-guide-to-mobile-web-development/introduction-to-npm-and-basic-npm-commands-18aa16f69f6b>

1. Npm –v
2. Node –v

The package.json is the project manifest file. Using package.json you can manage dependencies and write scripts.

**Install npm packages:**

|  |  |
| --- | --- |
| Locally | Globally |
| A locally installed package can be accessed only on the folder you’ve downloaded it. | A globally installed packages works anywhere on the machine. To install global packages you’ve to use -g flag. |
| The **node\_modules** is the folder in **which our local packages** are installed. There will be a new file named **package-lock.json**. This file contains the exact version of the package, unlike **package.json**which contains the semantic version(which we will be learning later). |  |
|  |  |
|  |  |

**How to create package.json file?**

**How to install dependencies from package.json?**

>npm install

**How to list all installed package?**

**>**npm list

**List all the packages including its dependencies of all packages.**

>npm list depth <number>

## Project Setup, Editor Environments

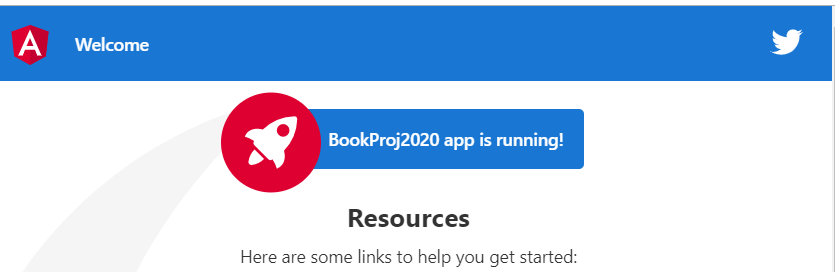
Step1: Create new Project

* ng new bookproj

Step 2: Run it

* ng serve

Step 3: Test on UI



## First Angular App & Directory Structure

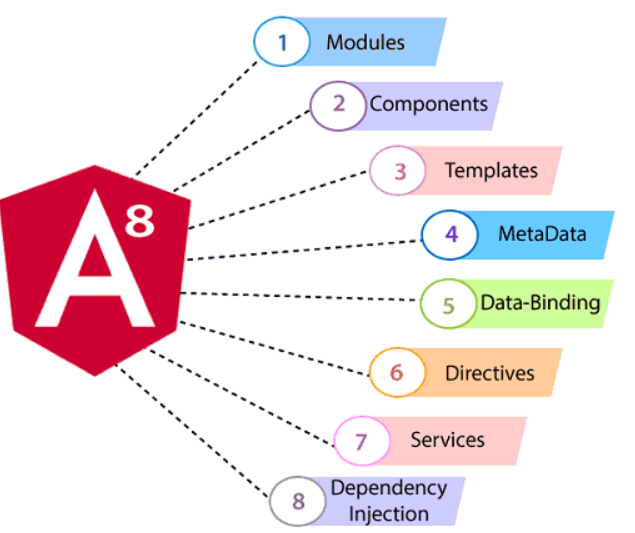
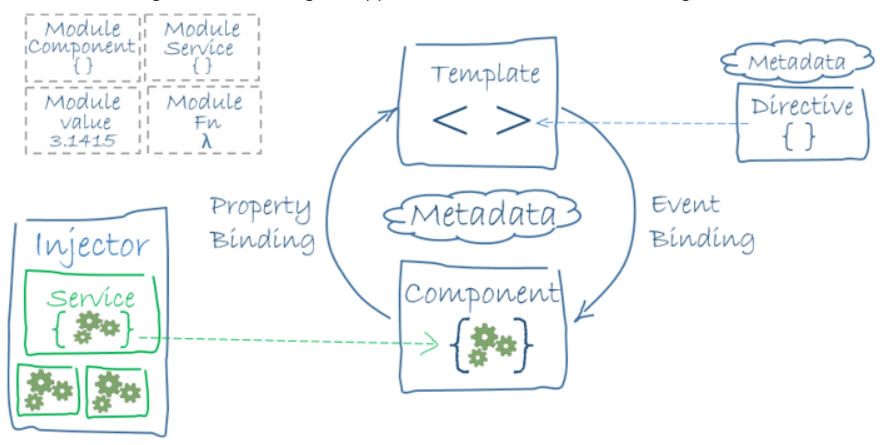
Directory Structure

|  |  |
| --- | --- |
| Src | Source code |
| Node\_Modules | All dependencies |
| Angular.json | <https://angular.io/guide/workspace-config>   provides workspace-wide and project-specific configuration defaults for build and development tools provided by the Angular CLI. |
| Karma.conf.js | Client side testing |

## Bootstrapping Angular App, Components, AppModule

Main.ts-> app.module.ts -> AppComponent

## Angular Fundamentals, Building Blocks

Architecture

**Building Blocks of Angular**

|  |  |  |
| --- | --- | --- |
| 1. | **Component** | To Control HTML View  Eg: Nav bar One Component. Form is another component. Footbar is another component. |
| 2. | **Modules** | Module is set of angular basic building blocks like component, directives, services etc. Each piece of code is called as "module" which perform a single task.   1. Eg: ***NgModules*,** which provide a compilation context for *components*. |
| 3. | **Templates** | Views of an Angular application. Provides connect to application data and DOM. |
| 4. | **Services** | Used to create components which can be shared across the entire application. Service providers can be injected into components as dependencies |
| 5. | **Metadata** | Used to add more data to an Angular class. Metadata is used to decorate a class so that it can configure the expected behavior of the class. |
| 6. | **Directives** | Used for expanding the functionality of HTML elements. |
| 7. | **Routing** | provides a service that facilitates developers to define a navigation path among the different application. |

## MetaData

Metadata is used to decorate a class so that it can configure the expected behavior of the class. The metadata is represented by decorators

|  |  |  |
| --- | --- | --- |
| **Class Decorators** | @Component and @NgModule |  |
| **Property decorators** | @Input and @Output |  |
| **Method decorators** | @HostListener |  |
| **Parameter decorators** | @Inject, Optional |  |