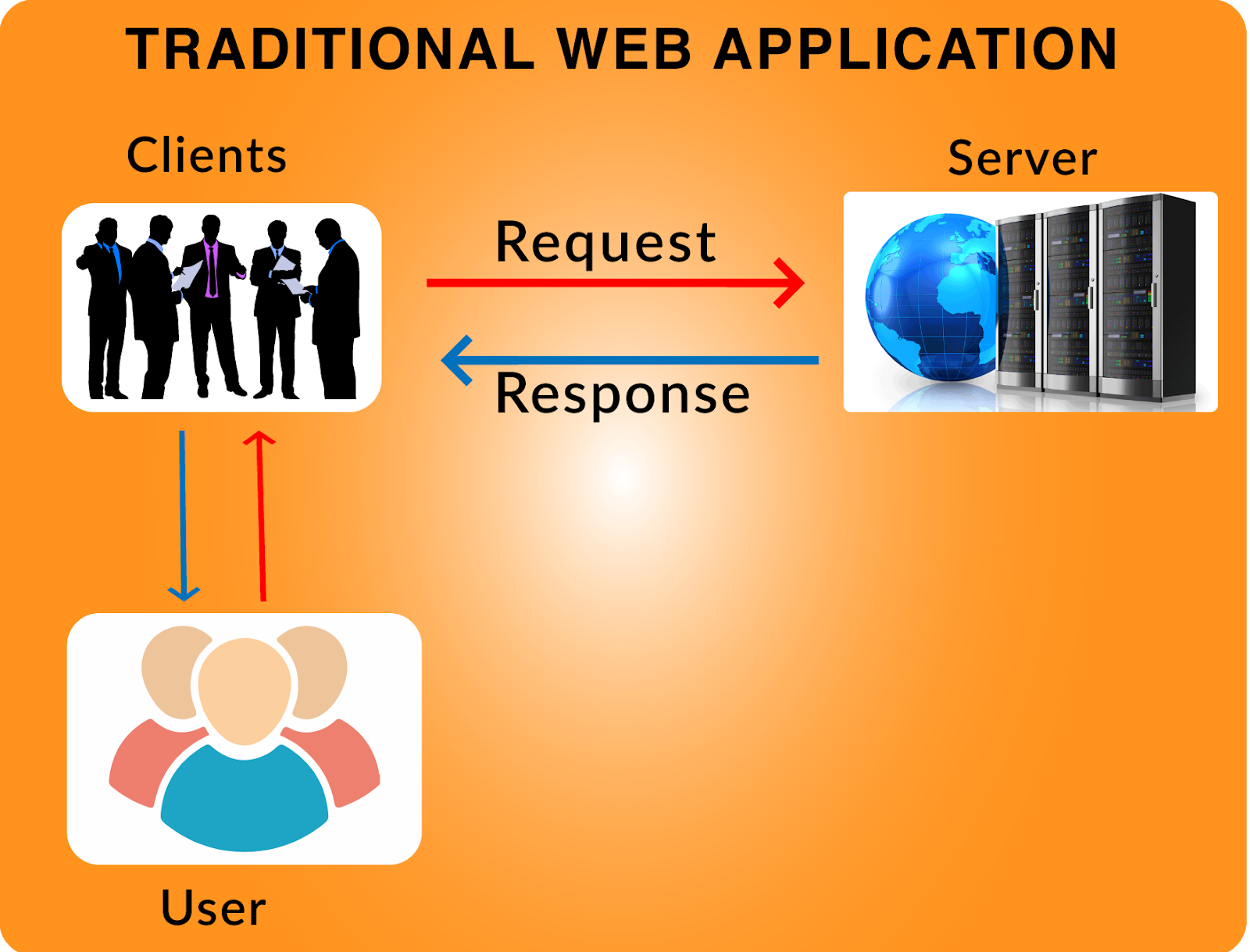
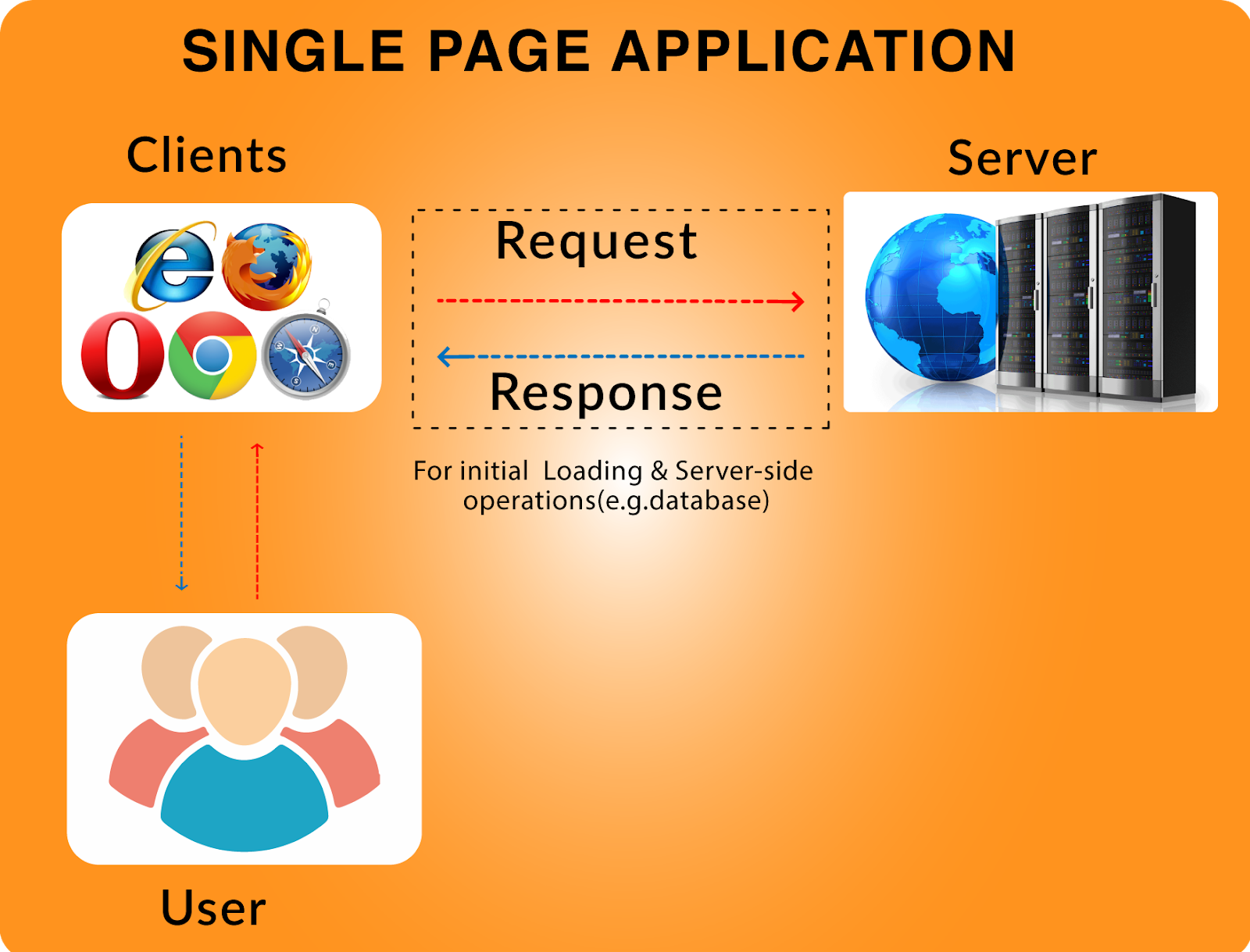
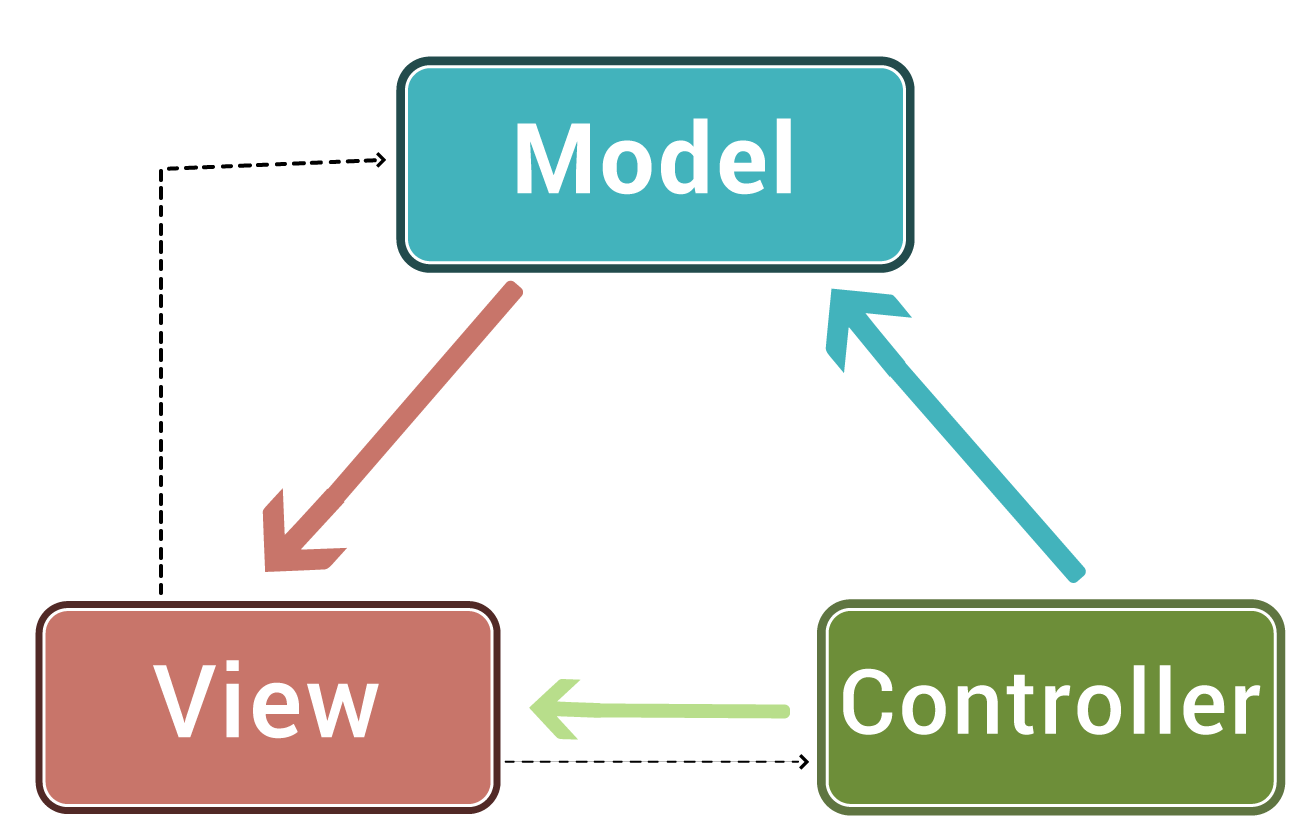
**Introduction:**

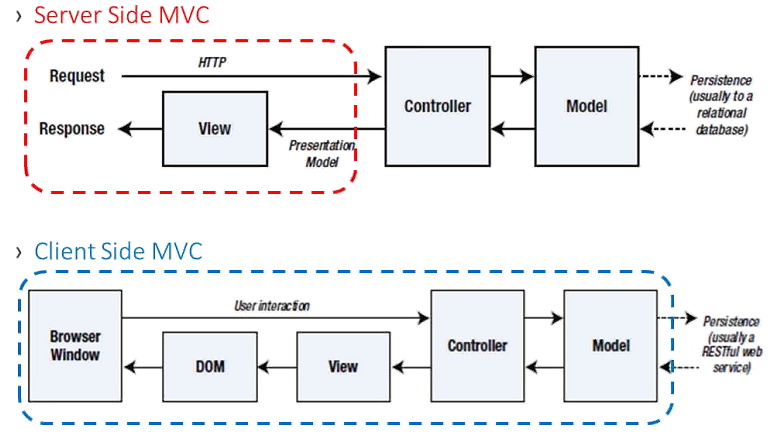
* Angular is a Client side JavaScript Framework which allows us to create Reactive Client-side Application.
* Great approach for a Single Page Application.
* By default it supports Two-way Data binding.
* By design it comes with Modular Approach, so you can reuse the code wherever you want.
* It supports many inbuilt functionalities.
* By this Angular architecture, we can easily maintainable the code.

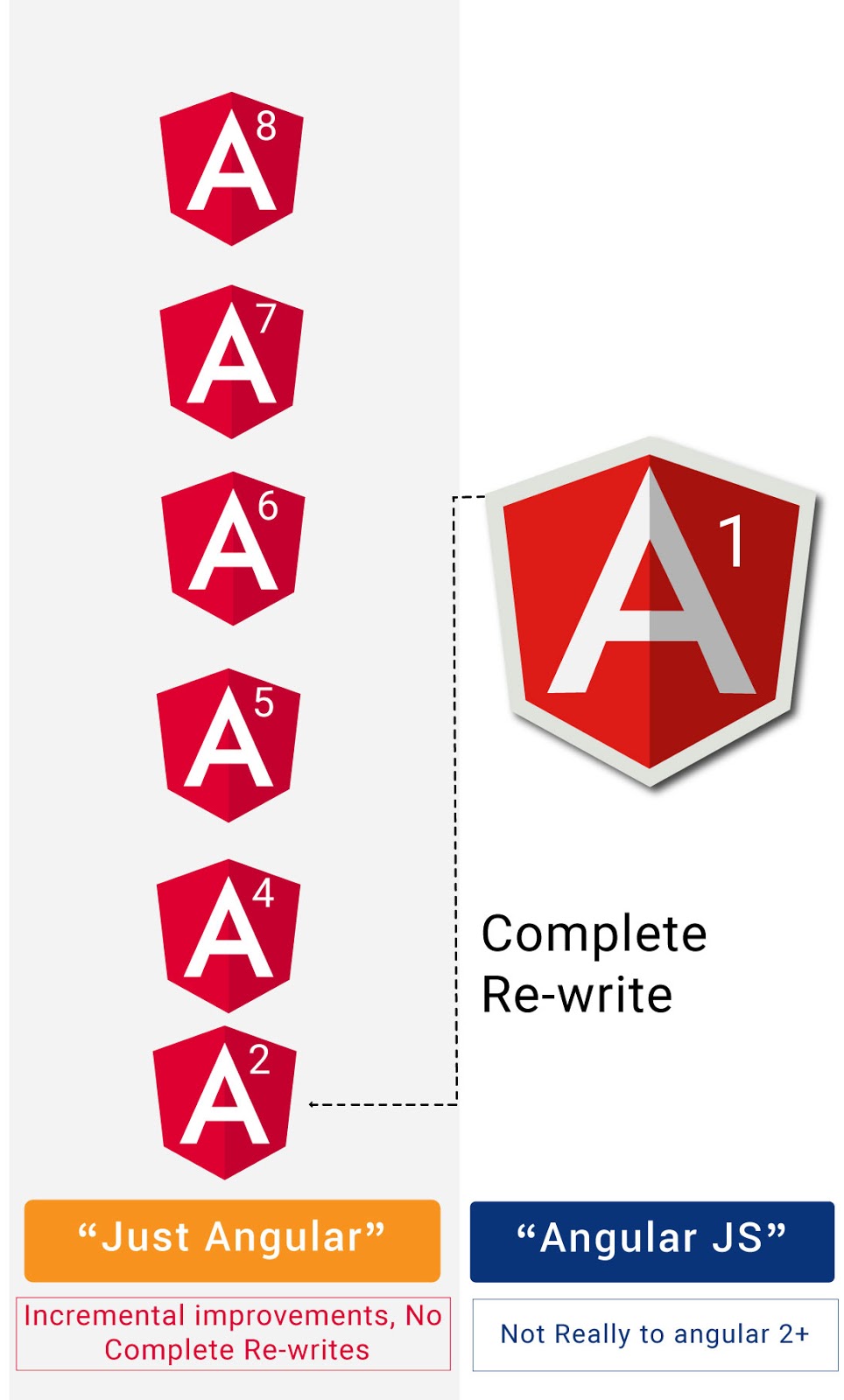




Angular has Model View Whatever structure





**Angular Versions**

**Environment Setup**

* Install the Angular CLI by using the following command.

***npm install  -g @angular/cli***

* Once you installed just use the below cmd for the verification

***ng  --version***

* It will give you the installed version details

If you installed Angular Already means, uninstall to avoid version conflicts

***npm uninstall -g*** ***angular-cli @angular/cli***

***npm cache clean --force***

* Navigate to where you want to create your Angular project by using cd cmd.
* Run the following cmd to create your 1st project.

***ng new your-project-name***

* Once your project setup, run the below cmd

***ng serve -o***

* Now you can run your project in  (By default the port is 4200)

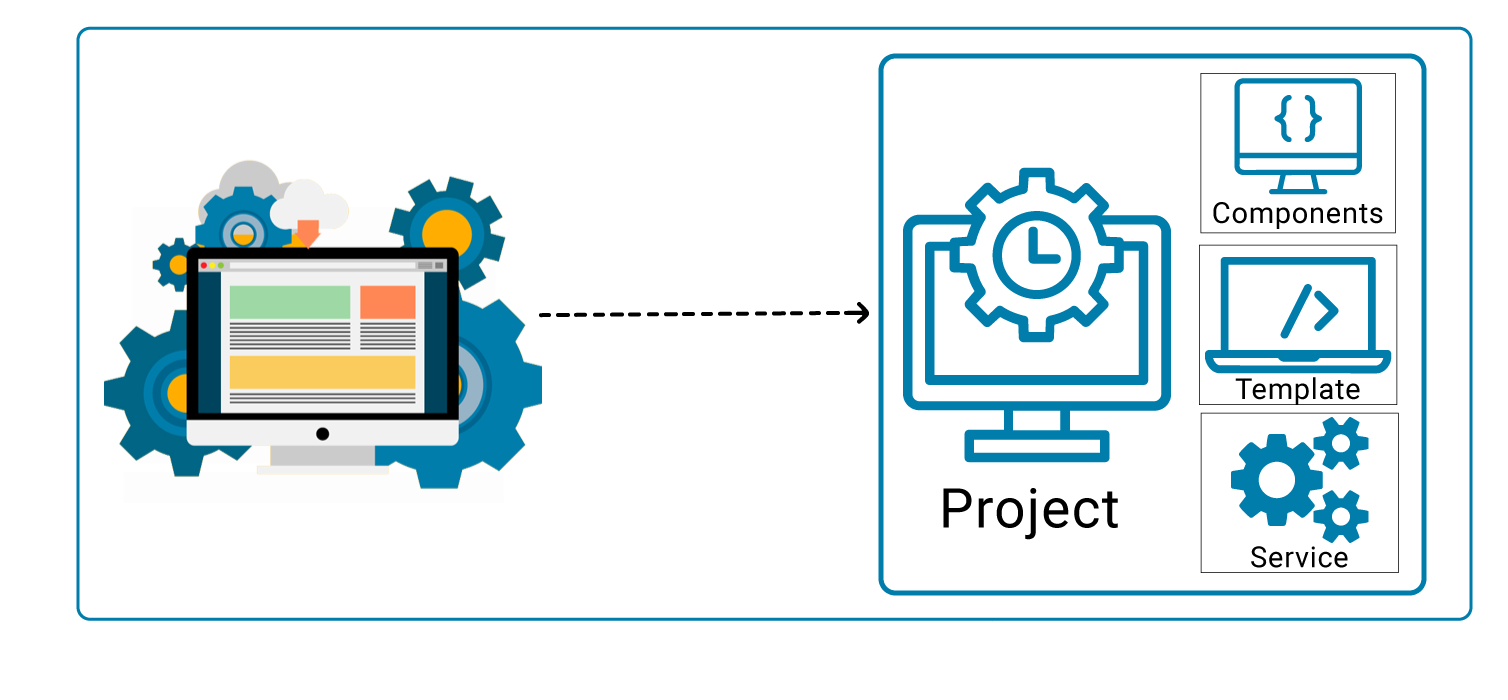
[***http://localhost:4200/***](http://localhost:4200/)

* If you want to change the port, run the following command

***ng serve --port 4201 -o***

First let's understand why should we use Angular CLI and what problems it solves.

CLI stands for Command Line Interface.



1.Create a separate application folder and add the package definition file ( ie. package.json) and other configuration files.

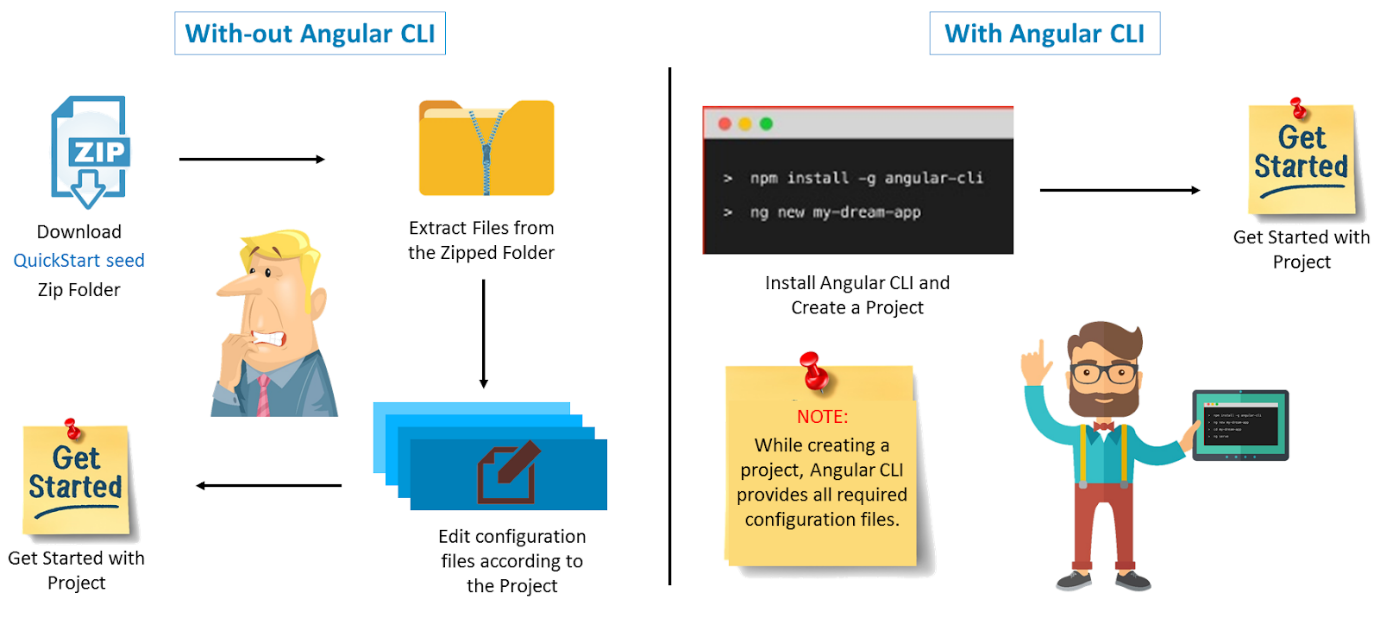
2. Install the packages using NPM

3. Setup the environment.

4. Provides required files from angular program.

5. Create index.html which hosts our application.

6. Reduces the development effort



**Features of Angular**

1. **High performance**

It gives high performance and You can get 10\* performance.

**2. Single language for both platforms**

You can build native mobile apps with strategies using Ionic Framework, NativeScript, and React Native.

**3. Code Generation**

Angular giving you all the benefits of hand-written code with the productivity of a framework by using Angular CLI.

**4. Universal**

You can use any technology with Angular for serving the application like node.js, .NET, PHP and other servers.

**5. Code splitting**

Angular apps load quickly with the new Component Router, which delivers automatic code-splitting, so users only load code required to render the view they request.

**6. Templates**

Quickly create UI views with simple and powerful template syntax.

**7.  Angular CLI**

Command line tools: You can easily and quickly start building components, adding components, testing them, and then, instantly deploy them using Angular CLI.

**Structure of Angular Project**

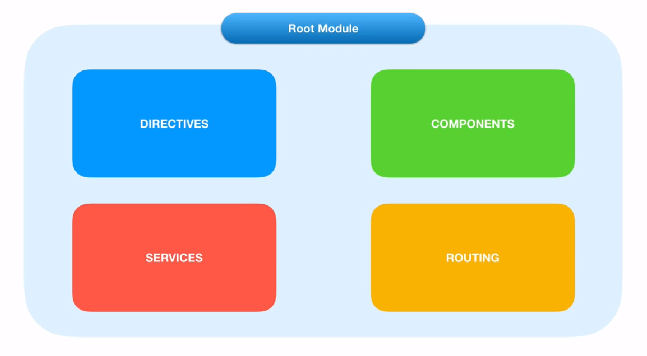
* **e2e.** stands for end to end testing. It is the way to ensure correct working of components together to provide desired functionality.
* **node\_modules** is the place where our all third party library resides, which can be used for development purpose.
* **Src** folder contains the actual source code for development. As you can see in the picture, src folder has 3 subfolders: App, Assets, Environment.
* **App** folder is the prominent part of Angular. It has all the components and modules of the application.
* **Assets** folder is the place where we can store our images and icons.
* **Environment** folder has two files: environment.prod.ts stores configuration for production environment and environment.ts stores configuration for development environment.
* **favicon.ico**. is the icon file which is displayed on the browser when you run the application.
* **index.html** is the first html file that is loaded when your application is run on browser.
* **Main.ts** is doorway for our application. It is the typescript file. Here we can bootstrap(load) our main module using methods.
* **Pollyfills.ts** is the scripts which eliminates the need to set up everything. In the other words, it makes our application compatible with different browsers. It bridges the gap between our Angular app and browser.
* **style.css** is the file where global styles for our application resides.
* **test.ts** is used for testing purpose.
* **tsconfig.app.json** has the root files and the compiler options.
* **.editorconfig** is used to define standard and consistence coding patterns for team development purpose.
* **.gitignore** is used when exporting your files and folders to github.
* **karma.conf.js** is the file which is used for unit test.
* **package.json** is the file which holds metadata for projects such as project name, version and handle dependencies of the project.
* **tsonfig.json** file has setting to convert your typescript code into javascript code to make the browser understand our code.
* **tslint.json** file checks your code for errors. It can be customised according to your own rules.

**Main Building Blocks of Angular**

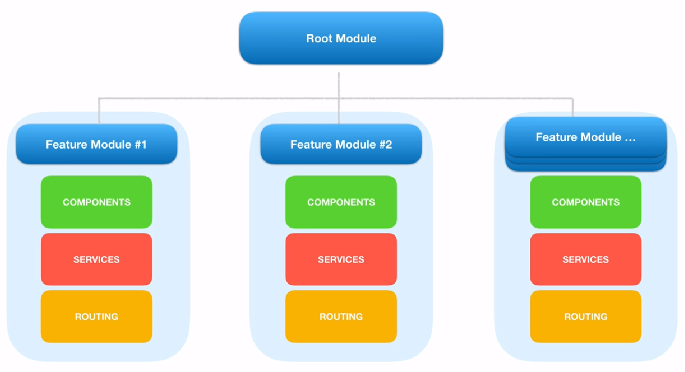
1. Modules
2. Components
3. Templates
4. Metadata
5. Data binding
6. Directives
7. Services
8. Dependency Injection.

**Modules**

* Bundle of functionalities of our App.
* Every Angular app contains at least one Angular module, i.e. the **root module**.
* Generally, it is named **as AppModule**.
* We can create multiple Modules if needed.
* Any angular module is a class with **@NgModule** decorator.
* Encapsulation of different similar functionalities.



**Importing other modules in Root Module**



**Module Decorators:**

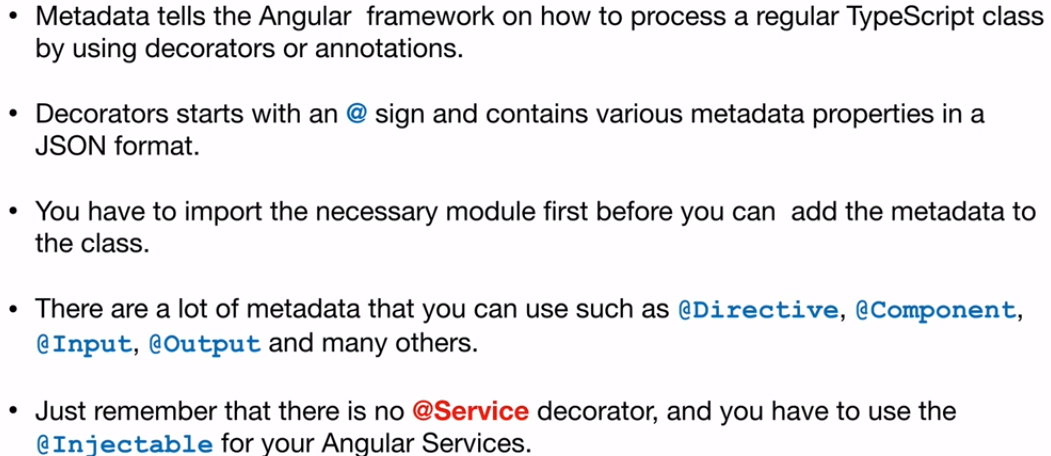
NgModule is a decorator function that takes metadata object whose properties describe the module.

Decorators are basically used for attaching metadata to classes so that it knows the configuration of those classes and how they should work.

The properties are,

* **declarations**: The classes that are related to views and it belongs to this module.
* **imports**: Modules whose classes are needed by the component of this module.
* **providers**: Services present in one of the modules which is to be used in the other modules or components. Once a service is included in the providers it becomes accessible in all parts of that application.
* **exports**: The classes that should be accessible to the components of other modules. (A root module generally doesn’t export it’s class because as root module is the one which imports other modules  & components to use them.)
* **bootstrap**: The *root component* which is the main view of the application. This root module only has this property and it indicates the component that is to be bootstrapped.

**Angular Metadata**

****

**Angular libraries**

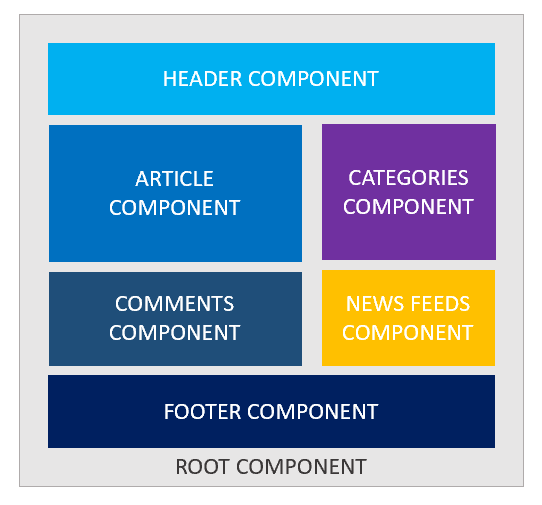
* Angular gives us a collection of JavaScript modules (library modules) which provide various functionalities.
* Each Angular library has **@angular** prefix, like @angular/core, @angular/compiler, @angular/compiler-cli, @angular/http, @angular/router.
* You can install them using the **npm** package manager and import parts of them with JavaScript import statements.

|  |
| --- |
| import { Component } from '@angular/core'; |

* In the above example, Angular’s Component decorator is imported from the @angular/core library.

**Components**

* A component controls one or more section on the screen called a **view**.
* For example, if we build shopping cart Application, we can have components like App Component (the bootstrapped component), list products, product description, add to cart, update cart, etc.,



* Component fetch and update data from **services**. Transforms the DOM using **Directives** and Redirecting the user to another component by using **Routing**.
* Inside the component, you define a component’s **presentation** logic i.e. how does it support the view—inside a class.
* Every app has a main component which is bootstrapped inside the main module, i.e AppComponent.

|  |
| --- |
| import { Component } from '@angular/core';   @Component({ selector:'app-root', templateUrl:'./app.component.html', styleUrls: ['./app.component.css'] })   export class AppComponent{ title = 'app works!'; } |

**Metadata:**

Metadata tells Angular how to process a class.

|  |
| --- |
| import { Component } from '@angular/core';   @Component({ selector:'app-root', templateUrl:'./app.component.html', styleUrls: ['./app.component.css'] })  ng generate component componentName |

* Here is the ***@Component*** decorator, which identifies the class immediately below it as a component class.
* The ***@Component*** decorator takes the required configuration object which Angular needs to create and present the component and its view.

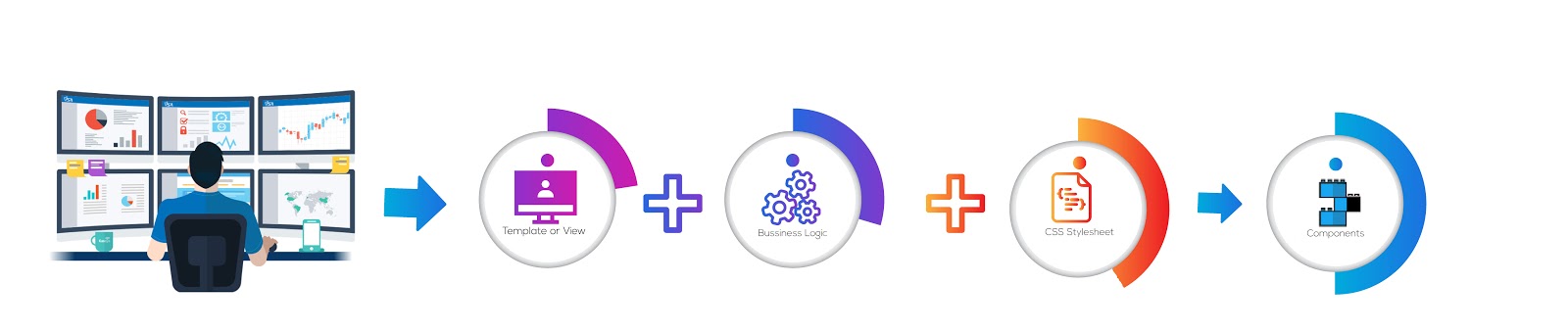
The most important configurations of @Component decorator are,

* **selector**: Selector tells Angular to create and insert an instance of this component where it finds ***<product-desc>*** tag. For example, if an app’s HTML contains ***<product-desc></product-desc>***, then Angular inserts an instance of the Product Description view between those tags.
* **templateUrl**: It contains the path of this component’s HTML template.
* **styleUrls**: We can specify the styes which is related to current component.

The template, metadata, and component together describe a view.

**An Angular Component in Action**

* app.component.css
* app.component.html
* app.component.spec.ts
* app.component.ts
* app.module.ts
* **A CSS file**: where we define all the styles for that component. These styles will only be scoped to this component and will not leak to the outside.
* **An HTML file**: contains the markup to render in the DOM.
* **A spec file**: includes the unit tests.
* **A TypeScript file**: where we define the state (the data to display) and behavior (logic) of our component.



**Note:**

* Angular Server serves the index.html and renders the application view in browser.
* The index.html contains all the JS, styles, compilation files which is required for run the application.
* The index.html holds the root component (AppComponent) which bootstrapped from the root model and  get loads very 1st.
* The index.html should have only one component that is root component by the directive **<app-root></app-root> (template directive).**
* The selector property value is used to create and attach the instance of class in the innerHtml of the element.
* The root component having multiple nested child components.

**Create Component through CLI**

|  |
| --- |
| ng generate component componentName  ng g c componentName |