Angular js

MVC Framework

- The M in MV* stands for Model. The Model is where you store the data and state of your application.
- The V in MV* stands for View. The View is where you actually render to the user the information that you want them to see and the View is where you receive input from the user.
- The * in MV* stands for, well, something else. In many common MV* frameworks the star is either a Controller or a Presenter or a ViewModel, or even something different besides those three.
- Well, Angular uses a Controller, so some people may call Angular an MVC framework.

Forward-thinking.

- The last important attribute of Angular that we will discuss is how it is forward-thinking.
- Angular is basically supporting the future of what we will see in web technology in the coming years, and as that technology becomes more widespread, our Angular applications will already be built to take advantage of that technology.
  - What HTML would be if it had been designed for applications and not for documents? Well that's what **web components** are. Web components allow you to make truly encapsulated components and widgets for your page, encapsulating HTML, JavaScript,and CSS.
  - Another up and coming feature that will soon be supported by some browsers is **Object.observe.** This technology **lets you watch an object or a property on a JavaScript object for changes and react to those changes**. Most MV* frameworks make you stick your data into special structures and call methods whenever you want to read or write to that data. Because Angular doesn't do that, it can support Object.observe when it becomes widely available and Angular will simply benefit from the performance improvements of having things handled by faster, lower-level code

Features

- Angular handles **showing the data on the page**, which you can do using partial templates or just modify the HTML DOM that already exists.
- Angular also handles updating the data or model based on user interaction, so when a user types into a text field, that new value can automatically be copied into your model. You don't have to wait for certain events, you can just tell Angular that a certain text box owns a certain piece of data and it will keep them in sync. This feature is called **two-way binding**.
- And lastly, Angular handles routing or moving from one view to another. This is the key piece in building **single page applications** or SPAs. This way, you can completely change your view based on user interaction with your page.
- Angular will also update the URL in the browser so that the new view can be **bookmarked** for later.

Angular Features

- First,**Angular supports two-way binding**.This means the user input into form fields is instantly updated in your Angular models. That means that in most cases you don't need to watch for specific events and respond to them and then manually update your HTML. Instead,Angular will handle that for you.
- Angular also employs a technique called **dirty checking**.The net result of this is that you don't have to put your data into special structures and call getter and setter methods to read and write to your data. You can simply put your model data into plain old JavaScript objects and Angular will respond whenever your data changes, and automatically update your view.
- Lastly,Angular is **built on Dependency Injection**. This lets you encapsulate pieces of your application better and also improves testability.

Angular Components

Services ←→ Controllers ←→ Views/Directives

Angular Components

- With Angular, everything starts with the **Controller**. The Controller is the central player in an Angular application. Controllers contain both logic and state. Controllers can communicate with Views through both one-way and two-way binding.
- Next we have the **View**. Views are made up of bindings and directives. This is how Angular talks to and listens to the user.
- And the last major piece is **services**. Services give you a place to contain the real logic and state of your application. If you think about what is the essential tasks of your application, this would likely happen in your services. Complex business logic, important application state, etc.,services are the place to house all that. Also, services are the place where you will want to communicate with the server.