Kevin Chen

7/28/17

Angular Notes

**Introduction**

Why use AngularJS 2?

* JavaScript: needed for adding interactivity, and data collection/modification.
* Angular provides a base for building client-side apps with a specialization on data binding: can easily update content on the UI
* Angular has a component architecture, allowing you to reuse common UI elements and elements with complex functionality, as well as modularity.
* Angular dependency injection makes it easy to decouple business logic in client-side code from the view logic.
* Angular is primed with a router module that will allow you to create parent and child routes to components.

Basics of TypeScript

* Angular source code is written in TypeScript
* TypeScript is a typed superset of JavaScript that gets transpiled to plain JavaScript
  + Offers the ability to write ES2015 classes and use Modules
  + Provides strong typing for variables and function signatures.
* Using TypeScript will make your code simpler (can write classes and use decorators) and will allow you to reference the Angular source code more easily.

**Architecture Overview**

Components, Bootstrap, and the DOM

* Angular is built off of a Component tree. The starting point of an Angular app is the bootstrapping of the initial parent component.
  + It then recursively looks at the component’s html view and see if there are any nested components, running the appropriate component code on those.
* A component is used to render a portion of HTML and provide functionality to that portion. This is done through a component class in which you define application logic for the component.
* Within each component, you can specify an HTML template to get rendered.
* The template syntax easily allows you to wire up to DOM events within your template.
* Each component gets configured with a selector to associate the component class logic with the specified markup element tag.

Directives and pipes

* A component is a directive with a template.
* Directives provide functionality and can transform the DOM.
* Two types of directives:
  + Structural: modify layout by altering elements in the DOM.
  + Attribute: changes the behavior/appearance of an existing DOM element.
* A directive gets configured with a selector that Angular uses to find a match and apply the directive.
* Angular comes with a number of directives.
* The pipe is a tool that takes in some data and transforms it in some way.

Data binding

* Angular can bind data to views, and work with data in those views.
* Interpolation is the most common way of displaying data in a view template.
* Can also use directives to display data.
* The template syntax includes: interpolation, built-in directives, expressions and statements, value binding, event binding, and expression operators.
* Can create local variables using a hash to get a reference to the element.
* There is a form module to help you build HTML forms

Dependency Injection (DI)

* Provide modules with other modules it needs instead of having the modules go out and get other modules on their own.
* Allows you to write decoupled code.
* Often used in constructors: you declare types on your constructor parameters, and Angular with make sure you receive an instance of that type when the constructor runs.
* Can replace a dependency during any phase of an application code.

Services and other business logic

* Services is an implied pattern indicating a JavaScript class or function that encapsulates some logic.
* Brokering data to/from the view and adding functionality to the view should be handled in components; business logic should be put in services.
* Services can leverage DI by specifying parameter types in constructors.

Data Persistence