# Week 2

coursera.org/learn/single-page-web-apps-with-angularjs/discussions/weeks/2/threads/UrqphApsEee6PA6lXFBgGA

**Short Review On Week 2**:

1) Variables starting with **$$** are internal to AngularJS and so you should never interact with them directly.

2) **$digest()** doesn't handle exceptions; **$apply()** handles both exceptions and $digest() internally; **$timeout()** is the AngularJS version for the JavaScript's setTimeout() and therefore no need to use $apply() or $digest().

3) **1-way binding**: {{ myValue }} is a regular interpolation with curly braces where *myValue* can only be updated from the controller to the browser; **2-way binding**: ng-model="myValue" is used in an <input> tag where *myValue* can be updated from the browser (user entry) to the controller ($scope.myValue) and vice-versa; **1-time binding**: {{ :: myValue }} where *myValue* is updated only once (initialized) when digest cycle kicks in - a **$watcher** is applied at initialization but also promptly removed once cycle is completed. Other watchers ($watcher) remain for whole life-cycle of each 1-way binding and 2-way binding (double curly braces sets up watcher; ng-model sets up watcher).

4) **ng-repeat** directive exposes **$index**; Filtered **ng-repeat** narrows items matching to specific entry (search):

```
<input type="text" ng-model="search">

<ul>

    <li ng-repeat="item in list | filter: search"> {{ item }} </li>

</ul>
```

5) Custom services instantiated with **.service** method are singleton (design pattern with single instance). Lazily Instantiated: only created if a component depends on this service.

6) Design pattern Factory - ***.factory****("factoryName", FactoryFunction)* - produces any type of object or function, including dynamic customized services. An injected factory function ("factoryName") refers to a function (or an object literal with function) that creates something.

7) **.provider** is the most flexible method for creating services in AngularJS. It creates dynamically configurable factories by declaring them before the application starts. AngularJS expects the provider to have a **$get** property attached to its instance to return a factory function with new created service. Also inside the service provider you have a config object ( *this.config = {...}* ) with defaults for later overwriting values when you configure the entire application.

8) **.config** is a special function that runs before services, factories and controllers are even created (and therefore those cannot be injected to it). A service provider (on the other hand) can be injected to a config function by attaching "Provider" to the end of its service name - *.provider("**serviceName**", ServiceProvider) / .config(Config) / Config.$inject = ["**serviceNameProvider**"];* - because providers are configured/declared before everything else. The injected service provider instance can then have its config properties redefined or reconfigured for a certain application.

9) **ng-if** / **ng-show** / **ng-hide**: Inside a div tag, the AngularJS's property/directive **ng-if** takes a condition which when evaluated to ***true*** the entire div is shown, but if ***false*** then the entire div is completely removed from the dom tree (commented out); **ng-show** doesn't remove the div but hides it using special class (ng-hide); Finally, **ng-hide** does the opposite to ng-show by reversing its condition's evaluation:  ... 