AngularJS
and your new project

Disclaimer

- Only looking at relevant basic concepts.
- Not going into Angular 2.0 changes.
- There are many viable alternatives to Angular these days, find what best suits your team and needs!

Angular's goals

- to decouple DOM manipulation from application logic.
- to decouple the client side of an application from the server side.
- to provide structure for the journey of building an application.

General criticism

- Very opinionated (and getting more so)
- Locks you in
- Steep learning curve
- MVC
- Leaky abstraction


Two-way data binding

Angular structure

- MVC to separate presentation data and logic
  - Controllers - business logic behind view
  - Directives - Extend HTML
  - Providers (services & factories) - Sharable logic/ models
  - HTML templates - view

More concepts

- Modules - containers for parts of our app
- $ - Angular namespace
- scope - glue between models and views
- Injector - builds all the parts of the app and resolves dependencies

```
var demoApp = angular.module('demoApp', []);

demoApp.controller('TestCtrl', function ($scope, toolService) {
  $scope.tools = toolService.getTools();
});
```

```
demoApp.service('toolService', function () {
  var tools = [
    {'name': 'Ionic',
     'snippet': 'Hybrid app framework based on Angular',
     'foo': 'bar'},
    {'name': 'Auriela',
     'snippet': 'Kinda like Angular but with much simpler conventions',
     'foo': 'foo'},
    {'name': 'React.js',
     'snippet': 'Library for building UI with components in a simpler, virtual DOM',
     'foo': 'foo'},
    {'name': 'Polymer.js',
     'snippet': 'Easy way to create your own elements',
     'foo': 'foo'}
  ];

  this.getTools = function () {
      return tools;
  };
});
```

```
<html ng-app="demoApp">
  <head>
    ...
  </head>
  <body ng-controller="TestCtrl">
    <ul>
      <li ng-repeat="tool in tools" ng-class='{"current": tool.foo == "bar" }'>
        {{tool.name}}: {{tool.snippet}}
      </li>
    </ul>
    <my-first-directive></my-first-directive>
  </body>
</html>
```

Common Mistakes

Bloat the controller

- Directives should manipulate the DOM
- Try to put business logic in services
- Don’t use controller $scope as a model

Use JQuery/manipulate DOM in controller

- Should always be in a directive
- Try to forget JQuery exists - it leads to hard to maintain, non-reusable code in Angular land
- Remember to reach for directives like ng-class and ng-click

Over-use $watch

- $watch can be a go-to shortcut at the beginning
- It’s powerful and easy to abuse
- Too many watchers will give you hard-to-follow, hard-to-test code

```
<input type="text" ng-model="foo"/>
```

```
$scope.$watch('foo', function (val) {
  switch(val) {
    if(val === 'test') {
      $scope.bar = 'foo is test';
    } else {
      $scope.bar = 'I do not understand';
    }
});
```

```
<input type="text" ng-model="foo" ng-change="updateBar(foo)"/>
```

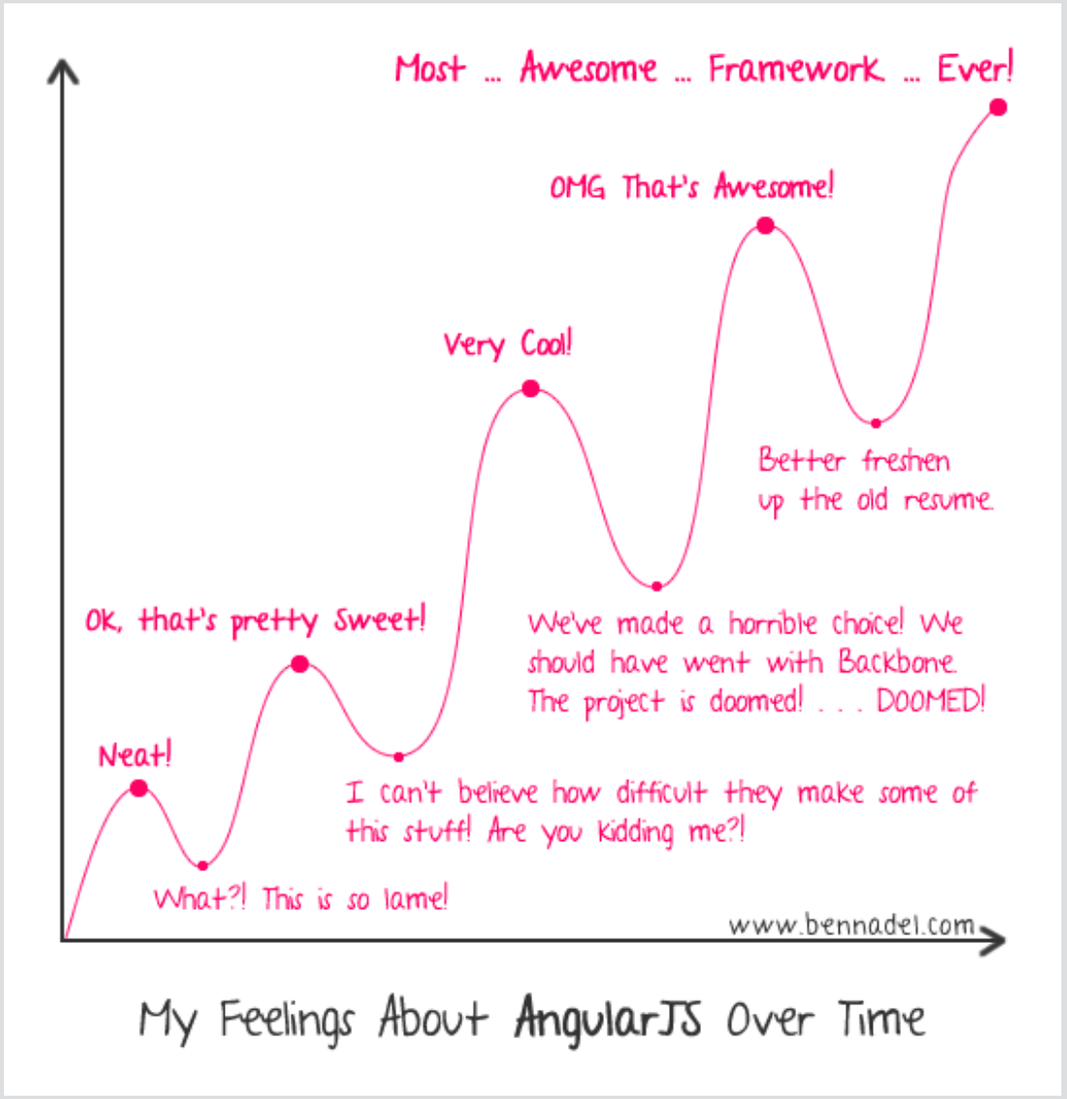
```
$scope.updateBar = function(foo) {
  if(val === 'test') {
    scope.bar = 'foo is testing me';
  } else {
    $scope.bar = 'I do not understand foo';
  }
};
```

$scope-ception

- Scopes can be tricky, especially directive scopes
- Take the time to understand:
    - scope declarations in directives
    - inheritance between parent and child scopes in Angular
    - isolated scopes

http://nathanleclaire.com/blog/2014/04/19/5-angularjs-antipatterns-and-pitfalls/

Mastering Angular takes time, and that's ok



Thanks :)