AngularJS ui-router

Overview

- In AngularJS 1.0.8 and earlier,
  the **`$routeProvider`** service was included
  - defines mappings from URL paths to routes
    defined by controllers, templates and views (**`ng-view`**)
  - does not support nested views or sibling views
- In AngularJS 1.2.0 and later,
  no route management services are included
  - **`$routeProvider`** can be downloaded separately from http://angularjs.org
    - press "Download" button, click "Extras" link, and look for **`angular-route.min.js`**
  - ui-router is a popular alternative
    - created by a team including Karsten Sperling, Nate Abele, Tim Kindberg, and others
    - supports nested views, sibling views, and more
    - download from https://github.com/angular-ui/ui-router
    - just need one file ... **`angular-ui-router.min.js`**

Setup

- In main HTML (typically `index.html`)

```
<script src="lib/angular-ui-router.min.js"></script>
```

- Add ui-router as a module dependency

```
var app = angular.module('app-name', ['ui.router']);
```

note that there is a dot instead of a hyphen

- Wherever a view is desired

```
<div ui-view>initial content</div>
```

initial content is optional

- Define states in function passed to `app.config`, that injects the `$stateProvider` and `$urlRouterProvider` services

```
app.config(function ($stateProvider, $urlRouterProvider) {
  $urlRouterProvider.otherwise('/default-path');
  $stateProvider
    .state('state-name-1', {
      state-config
    })
    ...
    .state('state-name-n',
      state-config
    });
});
```

State Configuration ...

- States are defined by a configuration object that contains a subset of the following properties (4 slides of these!)
  - **`url`**
    - string path for the state that starts with slash
    - can contain parameter names preceded by a colon or contained in curly braces; ex. **`/foo/:bar`** or **`/foo/{bar}`**
      - brace form allows specifying a regular expression that values must match; ex. **`/address/{zip:[0-9]{5}}`**
    - can contain query parameters; ex. **`/foo?bar&baz`**
    - for child states, this is relative to url of parent state
  - **`controller`** or **`controllerProvider`**
    - identifies the controller that is responsible for populating the scope used by the template
    - use **`controller`** to specify the string name of a controller
    - use **`controllerProvider`** to specify a function that can be injected with services to select and return the name of a controller or a controller function

can query parameter values be specified?

can't use capture groups

... State Configuration ...

- `template`, `templateUrl` or `templateProvider`
  - these identify an HTML snippet for rendering the state
  - use `template` to specify an HTML string
  - use `templateUrl` to specify the URL of a file containing HTML
    - typically under a directory named "`partials`"
    - can set to a function that takes `stateParams` and returns a template URL
  - use `templateProvider` to specify a function
    that can be injected with services to build and return the HTML

regardless of how
a template is specified,
it can contain directives
and binding expressions

- `views`
  - for populating multiple, named views within a single template
    - these `ui-view` attribute directives must have a value
    - ex. `<div ui-view="`*`view-name`*`"></div>`
  - some other top-level properties are ignored if this is present
    - `controller`, `controllerProvider`, `resolve`, `template`, `templateUrl` and `templateProvider`
  - value of `views` is an object where the keys are view names and
    the values are configuration objects containing properties for controllers, templates and resolve data
  - absolute view names are an advanced topic that allow targeting views in other states

`resolve` is described
on slide 7

... State Configuration ...

- **`data`**
  - attaches data with a state
  - value is an object whose properties can be accessed in controllers with **`$state.current.data.`*`property-name`***
    - must inject **`$state`** into controller to access
  - inherited by child states
- **`params`**
  - an array of parameter names or regular expressions used when the state has no URL
  - How is this useful? Where do the values come from?
- **`onEnter`** and **`onExit`**
  - functions that are called when the state is entered or exited
  - can perform state setup and teardown steps

… State Configuration

- **`resolve`**
  - value is an object
    - keys are names that can be injected into the controller
    - values are functions whose return values are injected (common case)
      or strings that are the name of a service that returns a single function
  - for values that are promises, it waits for them to be resolved
    - ex. can wait for REST services to return data (**`$http`** methods return promises)
  - obtains data before controller is rendered
- **`abstract`**
  - a state to which the UI cannot transition,
    but provides properties that are inherited by child states
  - can provide
    - base **`url`** that is prepended to child state **`urls`**
    - **`template`** that child states populate
    - **`resolve`** data that child states can inject into their controllers
    - custom **`data`** (described on the previous slide)
    - **`onEnter`** and **`onExit`** functions that run for each child state

Changing State

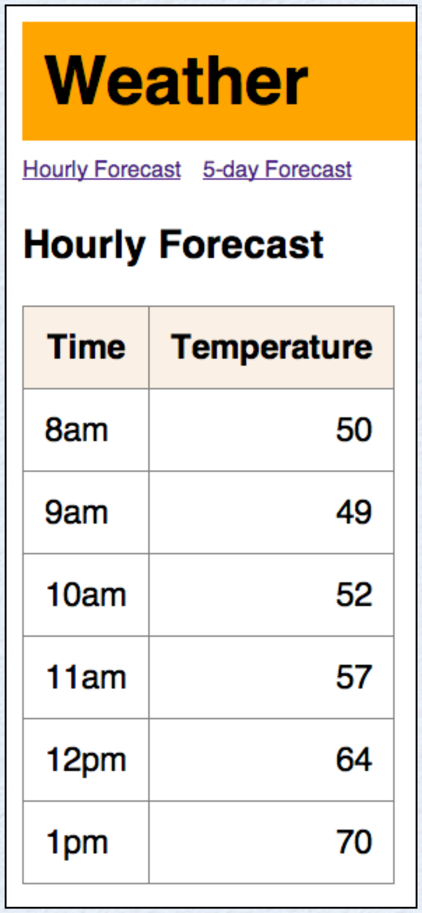
- There are three ways to change the state and thus change the UI
  - click a link with a **`ui-sref`** attribute

    ```
    <a ui-sref="state-name">link text</a>
    ```

  - call **`$state.go('state-name');`**
    - must inject **`$state`** to use
  - navigate to the URL of a state
    - typically by calling **`$location.path(url)`** or typing it into the browser address bar

Basic Example

- Demonstrates simple views that switch using **`ui-sref`** directives

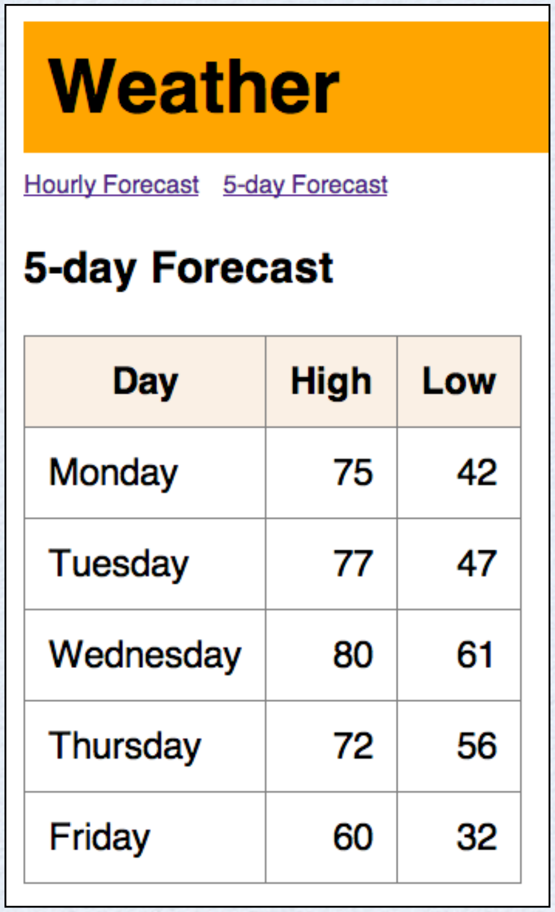


Weather

Hourly Forecast 5-day Forecast

Hourly Forecast

Time	Temperature
8am	50
9am	49
10am	52
11am	57
12pm	64
1pm	70



Weather

Hourly Forecast 5-day Forecast

5-day Forecast

Day	High	Low
Monday	75	42
Tuesday	77	47
Wednesday	80	61
Thursday	72	56
Friday	60	32

To run:
1) **`cd labs/ui-router/basic`**
2) **`grunt`**
3) browse **`localhost:3000`**

Basic Example HTML & CSS

index.html

```
<!DOCTYPE html>
<html ng-app="Weather">
  <head>
    <title>Weather</title>
    <link rel="stylesheet" href="styles/weather.css"/>
    <script src="lib/angular.min.js"></script>
    <script src="lib/angular-ui-router.min.js"></script>
    <script src="scripts/weather.js"></script>
  </head>
  <body>
    <h1>Weather</h1>
    <div id="links">
      <!-- ui-sref values are state names, not paths -->
      <a ui-sref="hourly">Hourly Forecast</a>
      <a ui-sref="daily">5-day Forecast</a>
    </div>
    <div ui-view></div>
  </body>
</html>
```

weather.css

```
body {
  font-family: sans-serif;
}

h1 {
  background-color: orange;
  padding: 10px;
  margin: 0;
}

table, th, td {
  border: solid gray 1px;
  border-collapse: collapse;
}

th {
  background-color: linen;
}

th, td {
  padding: 10px;
}

#links {
  font-size: 8pt;
  margin-top: 8px;
}

#links a {
  margin-right: 8px;
}

.number {
  text-align: right;
}
```

Basic Example Partials

daily.html

```html
<h3>5-day Forecast</h3>

<table>
  <tr>
    <th>Day</th>
    <th>High</th>
    <th>Low</th>
  </tr>
  <tr ng-repeat="dayForecast in dayForecasts">
    <td>{{dayForecast.day}}</td>
    <td class="number">{{dayForecast.high}}</td>
    <td class="number">{{dayForecast.low}}</td>
  </tr>
</table>
```

hourly.html

```html
<h3>Hourly Forecast</h3>

<table>
  <tr>
    <th>Time</th>
    <th>Temperature</th>
  </tr>
  <tr ng-repeat="hourForecast in hourForecasts">
    <td>{{hourForecast.hour}}</td>
    <td class="number">{{hourForecast.temperature}}</td>
  </tr>
</table>
```

Basic Example JavaScript ...

```
var app = angular.module('Weather', ['ui.router']);

app.factory('weatherSvc', function () {
  var svc = {};

  svc.getHourlyForecasts = function () {
    var forecasts = [];
    forecasts.push({hour: '8am', temperature: 50});
    ...
    return forecasts;
  };

  svc.getDailyForecasts = function () {
    var forecasts = [];
    forecasts.push({day: 'Monday', high: 75, low: 42});
    ...
    return forecasts;
  };

  return svc;
});

app.controller('WeatherCtrl', function ($scope, weatherSvc) {
  $scope.hourForecasts = weatherSvc.getHourlyForecasts();
  $scope.dayForecasts = weatherSvc.getDailyForecasts();
});
```

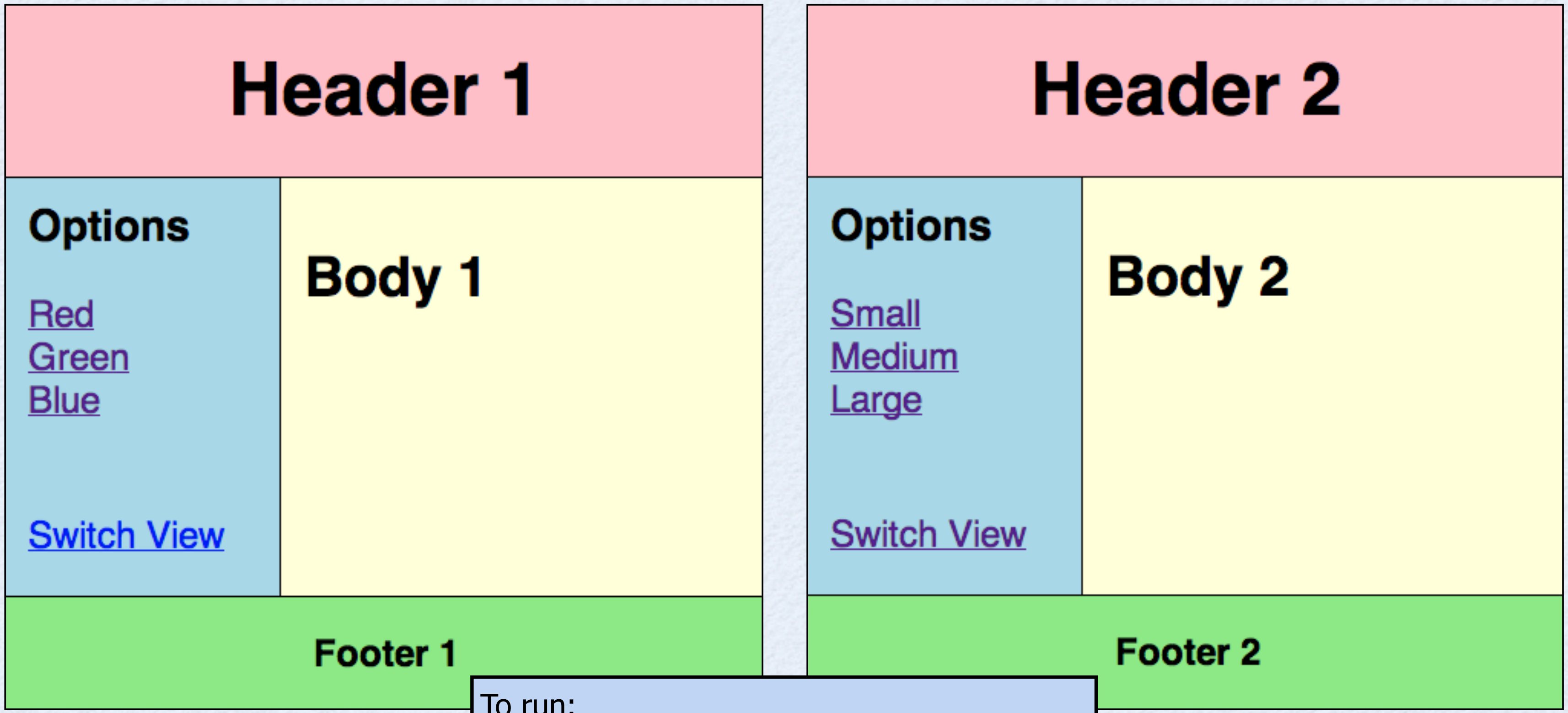
a real app would call REST services to obtain data rather than returning dummy data

... Basic Example JavaScript

```
app.config(function ($stateProvider, $urlRouterProvider) {
  $urlRouterProvider.otherwise('/daily');

  $stateProvider
    .state('hourly', {
      url: '/hourly',
      controller: 'WeatherCtrl',
      templateUrl: 'partials/hourly.html'
    })
    .state('daily', {
      url: '/daily',
      controller: 'WeatherCtrl',
      templateUrl: 'partials/daily.html'
    });
});
```

Sibling Views

- A template can contain more than one **`ui-view`** directive if they are named



To run:
1) **`cd labs/ui-router/sibling-views`**
2) **`grunt`**
3) browse **`localhost:3000`**

Sibling HTML

```
<html ng-app="SiblingViews">
  <head>
    <title>AngularJS Sibling Views</title>
    <link rel="stylesheet" href="styles/sibling.css"/>
    <script src="lib/jquery-1.10.1.min.js"></script>
    <script src="lib/angular.min.js"></script>
    <script src="lib/angular-ui-router.min.js"></script>
    <script src="scripts/sibling.js"></script>
  </head>
  <body>
    <header ui-view="header"></header>
    <nav ui-view="nav"></nav>
    <section ui-view="body"></section>
    <footer ui-view="footer"></footer>
  </body>
</html>
```

multiple, named views

Sibling CSS

```
/* Could use LESS to eliminate reduncancy. */

body {
  font-family: sans-serif;
  margin: 0;
}

footer {
  background-color: LightGreen;
  text-align: center;
  position: absolute;
  bottom: 0;
  height: 50px;
  width: 100%
}

footer > h4 {
  line-height: 50px; /* footer height */
  margin: 0;
  text-align: center;
  vertical-align: middle;
}

header {
  background-color: pink;
  height: 75px;
}

header > h1 {
  line-height: 75px; /* header height */
  margin: 0;
  text-align: center;
  vertical-align: middle;
}
```

```
nav {
  background-color: LightBlue;
  padding: 10px;
  position: absolute;
  top: 75px; /* header height */
  bottom: 50px; /* footer height */
  left: 0;
  width: 120px;
}

nav > div {
  margin-top: 40px;
}

nav > h3 {
  margin-top: 0;
}

section {
  background-color: LightYellow;
  padding: 10px;
  position: absolute;
  top: 75px; /* header height */
  bottom: 50px; /* footer height */
  left: 120px; /* nav width */
  right: 0;
}
```

Sibling Partials

header*.html

```html
<h1>Header 1</h1>
```

```html
<h1>Header 2</h1>
```

nav1.html

```html
<h3>Options</h3>

<a href="" ng-click="changeColor('red')">Red</a><br/>
<a href="" ng-click="changeColor('green')">Green</a><br/>
<a href="" ng-click="changeColor('blue')">Blue</a>

<div>
  <a ui-sref='second'>Switch View</a>
</div>
```

nav2.html

```html
<h3>Options</h3>

<a href="" ng-click="changeFontSize('12pt')">Small</a><br/>
<a href="" ng-click="changeFontSize('18pt')">Medium</a><br/>
<a href="" ng-click="changeFontSize('24pt')">Large</a>

<div>
  <a ui-sref='first'>Switch View</a>
</div>
```

body*.html

```html
<h1>Body 1</h1>
```

```html
<h1>Body 2</h1>
```

footer*.html

```html
<h4>Footer 1</h4>
```

```html
<h4>Footer 2</h4>
```

Sibling JavaScript ...

sibling.js

```
(function () {
  'use strict';

  var app = angular.module('SiblingViews', ['ui.router']);

  app.controller('SiblingCtrl', function ($scope) {
    $scope.changeColor = function (colorName) {
      $('section').css('color', colorName);
    };

    $scope.changeFontSize = function (size) {
      $('section').css('font-size', size);
    };
  });

  app.config(function ($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/first');
    $stateProvider
      ... code snippets on next slide go here ...
  });
})();
```

... Sibling JavaScript

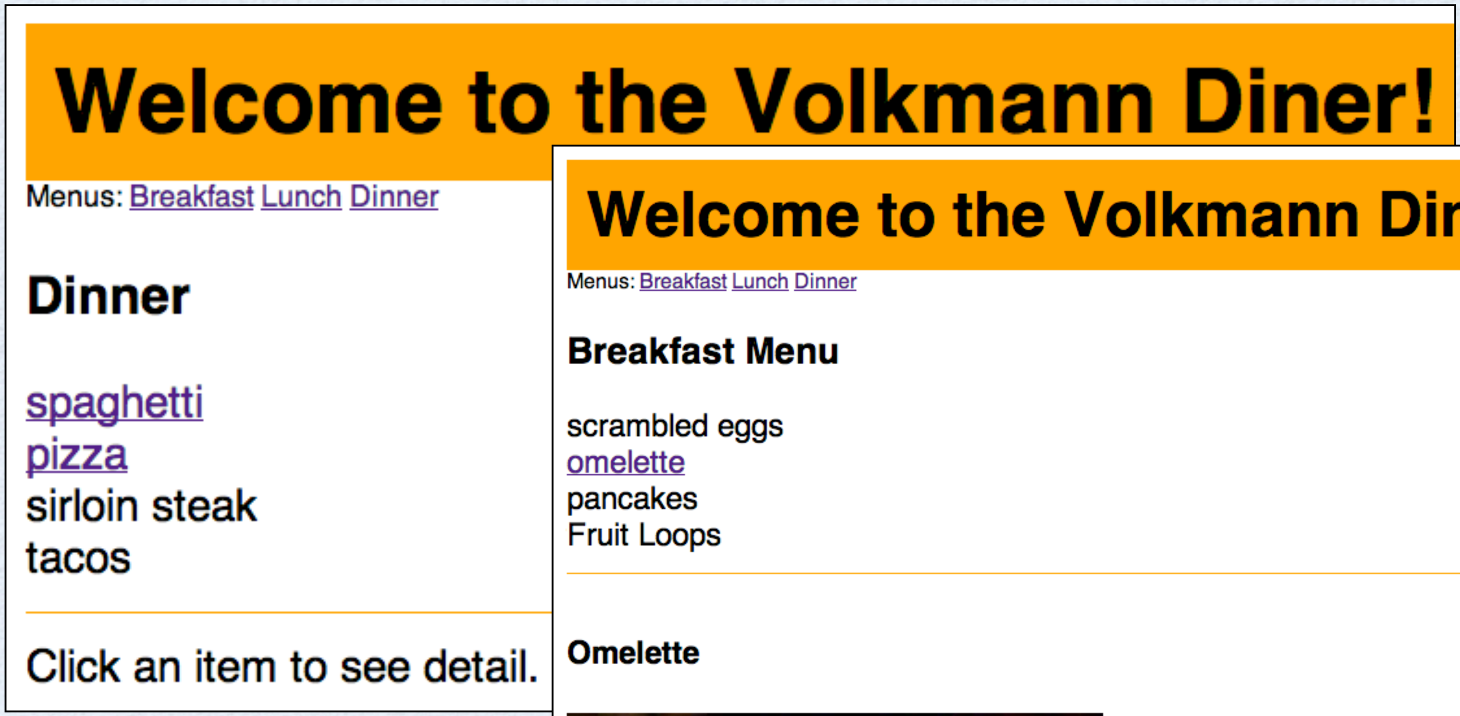
```
.state('first', {
  url: '/first',
  views: {
    header: {
      templateUrl: 'partials/header1.html'
    },
    nav: {
      controller: 'SiblingCtrl',
      templateUrl: 'partials/nav1.html'
    },
    body: {
      templateUrl: 'partials/body1.html'
    },
    footer: {
      templateUrl: 'partials/footer1.html'
    }
  }
})
```

```
.state('second', {
  url: '/second',
  views: {
    header: {
      templateUrl: 'partials/header2.html'
    },
    nav: {
      controller: 'SiblingCtrl',
      templateUrl: 'partials/nav2.html'
    },
    body: {
      templateUrl: 'partials/body2.html'
    },
    footer: {
      templateUrl: 'partials/footer2.html'
    }
  }
});
```

Nested Views ...

- Specified by defining a state whose name contains a period
  - ***'parent-name.child-name'***
  - navigating to the URL of a child view renders that and its parent (if not already rendered)
  - when defining a child state, the child **`url`** property is relative to the parent **`url`** property
    - ex. if parent state **`url`** is **`'/team'`** and child state url is **`'/player'`** then the full URL is **`/team/player`**
    - can be parameterized; ex. **`'/:name'`**
- Child states must be defined after their parent state
  - if not, will get “TypeError: Cannot read property 'navigable' of undefined” with no indication of which state definition caused the error
- Parameterized URLs
  - values are obtained using the **`$stateParams`** service
    - in properties on that object

… Nested Views



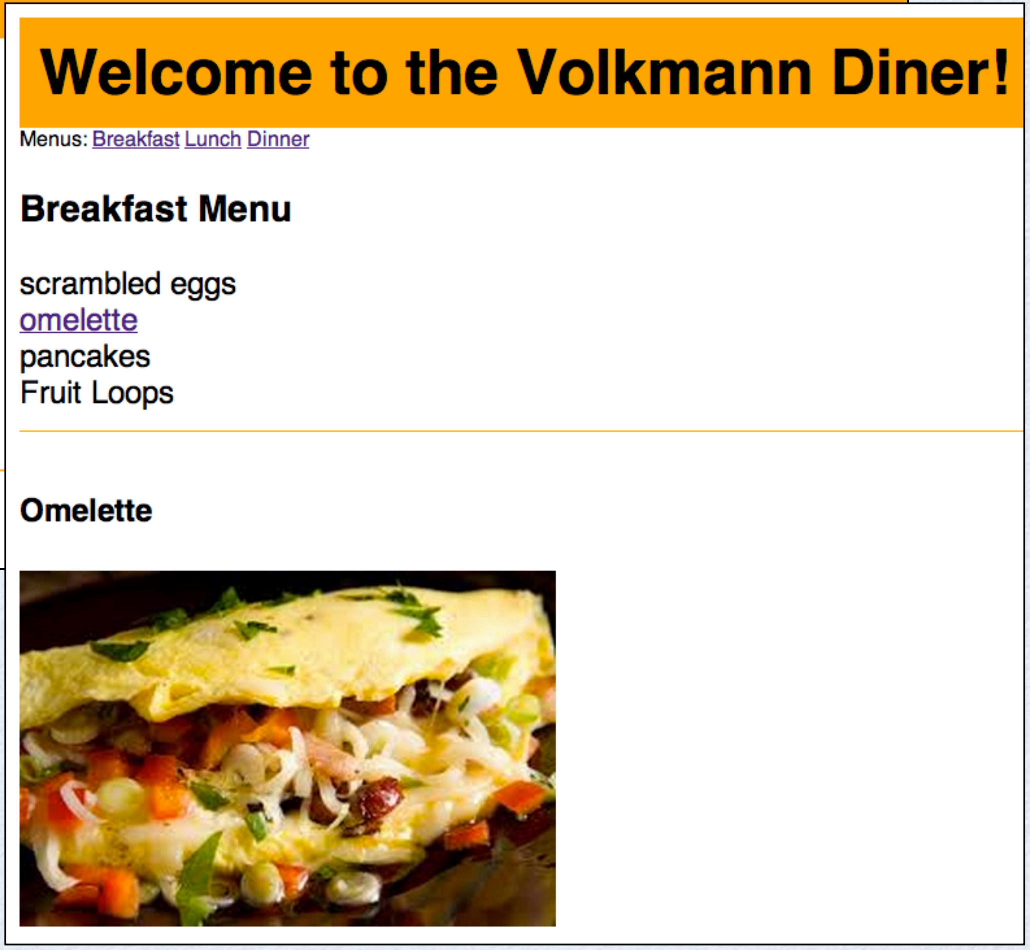
Welcome to the Volkmann Diner!

Menus: Breakfast Lunch Dinner

Dinner

spaghetti
pizza
sirloin steak
tacos

Click an item to see detail.



Welcome to the Volkmann Diner!

Menus: Breakfast Lunch Dinner

Breakfast Menu

scrambled eggs
omelette
pancakes
Fruit Loops

Omelette

To run:
1) `cd labs/ui-router/nested-views`
2) `grunt`
3) browse `localhost:3000`

Nested HTML

```html
<!DOCTYPE html>
<html ng-app="Diner">
  <head>
    <title>AngularJS ui-router sibling view demo</title>
    <link rel="stylesheet" href="styles/diner.css"/>
    <script src="lib/angular.min.js"></script>
    <script src="lib/angular-ui-router.min.js"></script>
    <script src="scripts/diner.js"></script>
  </head>
  <body ng-controller="DinerCtrl">
    <h1>Welcome to the {{name}} Diner!</h1>

    <div id="menus">
      Menus:
      <!-- ui-sref values are state names, not paths -->
      <a ui-sref="breakfast">Breakfast</a>
      <a ui-sref="lunch">Lunch</a>
      <a ui-sref="dinner">Dinner</a>
    </div>

    <div id="menu" ui-view></div>
  </body>
</html>
```

Nested CSS

diner.css

```css
body {
  font-family: sans-serif;
}

h1 {
  background-color: orange;
  padding: 10px;
  margin: 0;
}

#item {
  border-top: solid orange 1px;
  margin-top: 10px;
  padding-top: 10px;
}

#menus {
  font-size: 8pt;
}
```

Nested Partials ...

breakfast.html

```
<h3>Breakfast Menu</h3>

<div>scrambled eggs</div>
<a ui-sref="breakfast.omelette">omelette</a><br/>
<div>pancakes</div>
<div>Fruit Loops</div>

<div id="item" ui-view>Click an item to see detail.</div>
```

lunch.html

```
<h3>Lunch Menu</h3>

<a ui-sref="lunch.pizza">pizza</a><br/>
<div>salad</div>
<div>stir fry</div>
<div>sub sandwich</div>

<div id="item" ui-view>Click an item to see detail.</div>
```

dinner.html

```
<h3>Dinner</h3>

<a ui-sref="dinner.spaghetti">spaghetti</a><br/>
<a ui-sref="dinner.pizza">pizza</a><br/>
<div>sirloin steak</div>
<div>tacos</div>

<div id="item" ui-view>Click an item to see detail.</div>
```

... Nested Partials

breakfast.omelette.html

```
<h4>Omelette</h4>
<img src="images/omelette.jpg"/>
```

lunch.pizza.html

```
<h4>Lunch Pizza</h4>
<img src="images/pizza.jpg"/>
```

dinner.pizza.html

```
<h4>Dinner Pizza</h4>
<img src="images/pizza.jpg"/>
```

dinner.spaghetti.html

```
<h4>Spaghetti</h4>
<img src="images/spaghetti.jpg"/>
```

Nested JavaScript ...

diner.js

```
(function () {
  'use strict';

  var app = angular.module('Diner', ['ui.router']);

  app.controller('DinerCtrl', function ($scope) {
    $scope.name = 'Volkmann';
  });

  app.controller('MealCtrl', function ($scope, $rootScope, $state, $timeout) {
    // This demonstrate changing state from code.
    // It changes to the "lunch" state after two seconds.
    // To use it, specify this as the controller for one or more of the states.
    $timeout(function () {
      $state.go('lunch');
    }, 2000);
  });
```

... Nested JavaScript

diner.js

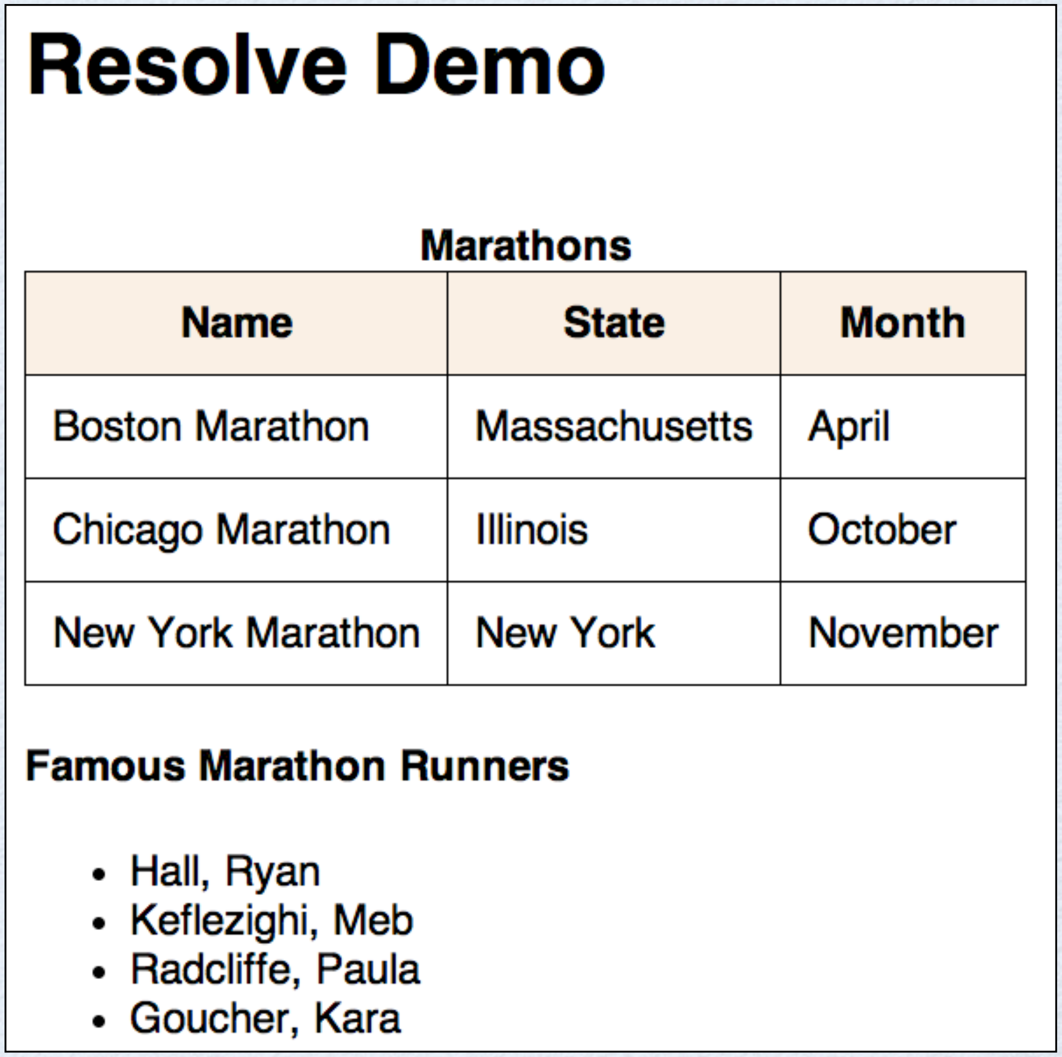
```
  app.config(function ($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/dinner');

    $stateProvider
      .state('breakfast', {
        url: '/breakfast',
        templateUrl: 'partials/breakfast.html'
      })
      .state('breakfast.omelette', {
        url: '/omelette',
        templateUrl: 'partials/breakfast.omelette.html'
      })
      .state('lunch', {
        url: '/lunch',
        templateUrl: 'partials/lunch.html'
      })
      .state('lunch.pizza', {
        url: '/pizza',
        templateUrl: 'partials/lunch.pizza.html'
      })
      .state('dinner', {
        url: '/dinner',
        templateUrl: 'partials/dinner.html'
      })
      .state('dinner.pizza', {
        url: '/pizza',
        templateUrl: 'partials/dinner.pizza.html'
      })
      .state('dinner.spaghetti', {
        url: '/spaghetti',
        templateUrl: 'partials/dinner.spaghetti.html'
      });
  });
})();
```

Resolve

- Can load data before view is rendered
  - rather that having the page update in a haphazard fashion
- Can wait for multiple “requests” to be resolved

To run:
1) **`cd labs/ui-router/sibling-views`**
2) **`grunt`**
3) browse **`localhost:3000`**

Note jumpy population of page.
Modify **`marathons.js`** to use **`GoodCtrl`** instead of **`BadCtrl`** and run again.



Resolve Demo

Marathons

Name	State	Month
Boston Marathon	Massachusetts	April
Chicago Marathon	Illinois	October
New York Marathon	New York	November

Famous Marathon Runners

- Hall, Ryan
- Keflezighi, Meb
- Radcliffe, Paula
- Goucher, Kara

Resolve HTML & CSS

index.html

```html
<!DOCTYPE html>
<html ng-app="Marathons">
  <head>
    <title>AngularJS ui-router resolve demo</title>
    <link rel="stylesheet" href="styles/marathons.css"/>
    <script src="lib/angular.min.js"></script>
    <script src="lib/angular-ui-router.min.js"></script>
    <script src="scripts/marathons.js"></script>
  </head>
  <body>
    <h1>Resolve Demo</h1>
    <div ui-view>The view is loading.</div>
  </body>
</html>
```

marathons.css

```css
body {
  font-family: sans-serif;
}

table, th, td {
  border: solid black 1px;
  border-collapse: collapse;
  border-spacing: 0;
  padding: 10px;
}

table > caption {
  font-weight: bold;
  margin-top: 20px;
}

table th {
  background-color: linen;
}
```

Resolve Partial

marathons.html

```
<table>
  <caption>Marathons</caption>
  <tr>
    <th>Name</th>
    <th>State</th>
    <th>Month</th>
  </tr>
  <tr ng-repeat="marathon in marathons">
    <td>{{marathon.name}}</td>
    <td>{{marathon.state}}</td>
    <td>{{marathon.month}}</td>
  </tr>
</table>

<h4>Famous Marathon Runners</h4>
<ul>
  <li ng-repeat="runner in runners">
    {{runner.lastName}}, {{runner.firstName}}
  </li>
</ul>
```

Resolve JavaScript ...

marathons.js

```
(function () {
  'use strict';

  var app = angular.module('Marathons', ['ui.router']);

  // This uses the $q service to simulate the delay of HTTP requests
  // and returning a promise.
  app.factory('marathonSvc', function ($q, $timeout) {
    var svc = {};
```

... Resolve JavaScript ...

marathons.js

```
  svc.getMarathons = function () {
    var dfr = $q.defer();
    $timeout(function () {
      dfr.resolve([
        {name: 'Boston Marathon', month: 'April', state: 'Massachusetts'},
        {name: 'Chicago Marathon', month: 'October', state: 'Illinois'},
        {name: 'New York Marathon', month: 'November', state: 'New York'}
      ]);
    }, 1500);
    return dfr.promise;
  };

  svc.getRunners = function () {
    var dfr = $q.defer();
    $timeout(function () {
      dfr.resolve([
        {firstName: 'Ryan', lastName: 'Hall'},
        {firstName: 'Meb', lastName: 'Keflezighi'},
        {firstName: 'Paula', lastName: 'Radcliffe'},
        {firstName: 'Kara', lastName: 'Goucher'},
      ]);
    }, 1000);
    return dfr.promise;
  };

  return svc;
});
```

... Resolve JavaScript ...

```
app.controller('BadCtrl', function ($scope, marathonSvc) {
  // Must deal with the promise that is returned manually (calling then).
  marathonSvc.getMarathons().then(
    function (marathons) { $scope.marathons = marathons; },
    function (err) { alert(err); });
  marathonSvc.getRunners().then(
    function (runners) { $scope.runners = runners; },
    function (err) { alert(err); });
});

app.controller('GoodCtrl', function ($scope, marathons, runners) {
  $scope.marathons = marathons;
  $scope.runners = runners;
});

app.config(function ($stateProvider, $urlRouterProvider) {
  $urlRouterProvider.otherwise('/marathons');
```

... Resolve JavaScript ...

```
    $stateProvider
      .state('marathons', {
        url: '/marathons',
        templateUrl: 'partials/marathons.html',

        // With BadCtrl, table caption and headings
        // are visible before data is loaded.
        controller: 'BadCtrl'
      });
  });
})();
```

```
    $stateProvider
      .state('marathons', {
        url: '/marathons',
        templateUrl: 'partials/marathons.html',

        // With GoodCtrl, table caption and headings
        // are not visible until data is loaded.
        controller: 'GoodCtrl',
        resolve: {
          // Can wait for any number of properties to be resolved.
          // Waits for promises to be resolved before
          // injecting into controller (don't need to call then).
          marathons: function (marathonSvc) {
            return marathonSvc.getMarathons();
          },
          runners: function (marathonSvc) {
            return marathonSvc.getRunners();
          }
        }
      });
  });
})();
```

State Change Events

- ui-router emits these events on **`$rootScope`**
  - to listen for them, `$rootScope.$on('event-name', function (params) { ... });`
- **`$stateChangeStart`**
  - arguments are **`event`**, **`toState`**, **`toParams`**, **`fromState`** and **`fromParams`**
  - state names are in **`toState.name`** and **`fromState.name`**
  - to prevent transition, call **`event.preventDefault()`**
- **`$stateChangeSuccess`**
  - emitted when state transition is completed
  - arguments are the same as for **`$stateChangeStart`**
- **`$stateChangeError`**
  - arguments are the same as for **`$stateChangeStart`** plus **`error`** argument at end
- **`$stateNotFound`**
  - argument is object that has properties **`to`** (the state name), **`toParams`** and **`options`**

State Changes

```
...

app.factory('stateMonitorSvc', function ($rootScope) {
  $rootScope.$on('$stateChangeStart',
    function (event, toState, toParams, fromState, fromParams) {
      console.log('changing state from', fromState.name, 'to', toState.name);
    });

  $rootScope.$on('$stateChangeSuccess',
    function (event, toState, toParams, fromState, fromParams) {
      console.log('changed state from', fromState.name, 'to', toState.name);
    });

  $rootScope.$on('$stateNotFound', function (unfoundState) {
    console.log('tried to change to state', unfoundState.to,
      'but that state is not known');
  });

  $rootScope.$on('$stateChangeError',
    function (event, toState, toParams, fromState, fromParams, error) {
      console.log(error, 'changing state from',
        fromState.name, 'to', toState.name);
    });
});

...
```

View Load Events

- ui-router emits these events on **`$rootScope`**
  - to listen for them, `$rootScope.$on('event-name', function (params) { ... });`
- **`$viewContentLoading`**
  - when a view begins loading and the DOM is not yet rendered
  - arguments are **`event`** and **`viewConfig`** which contains all the state configuration properties and the view name in **`targetView`**
- **`$viewContentLoaded`**
  - after a view has been loaded and the DOM is rendered
  - argument is **`event`**