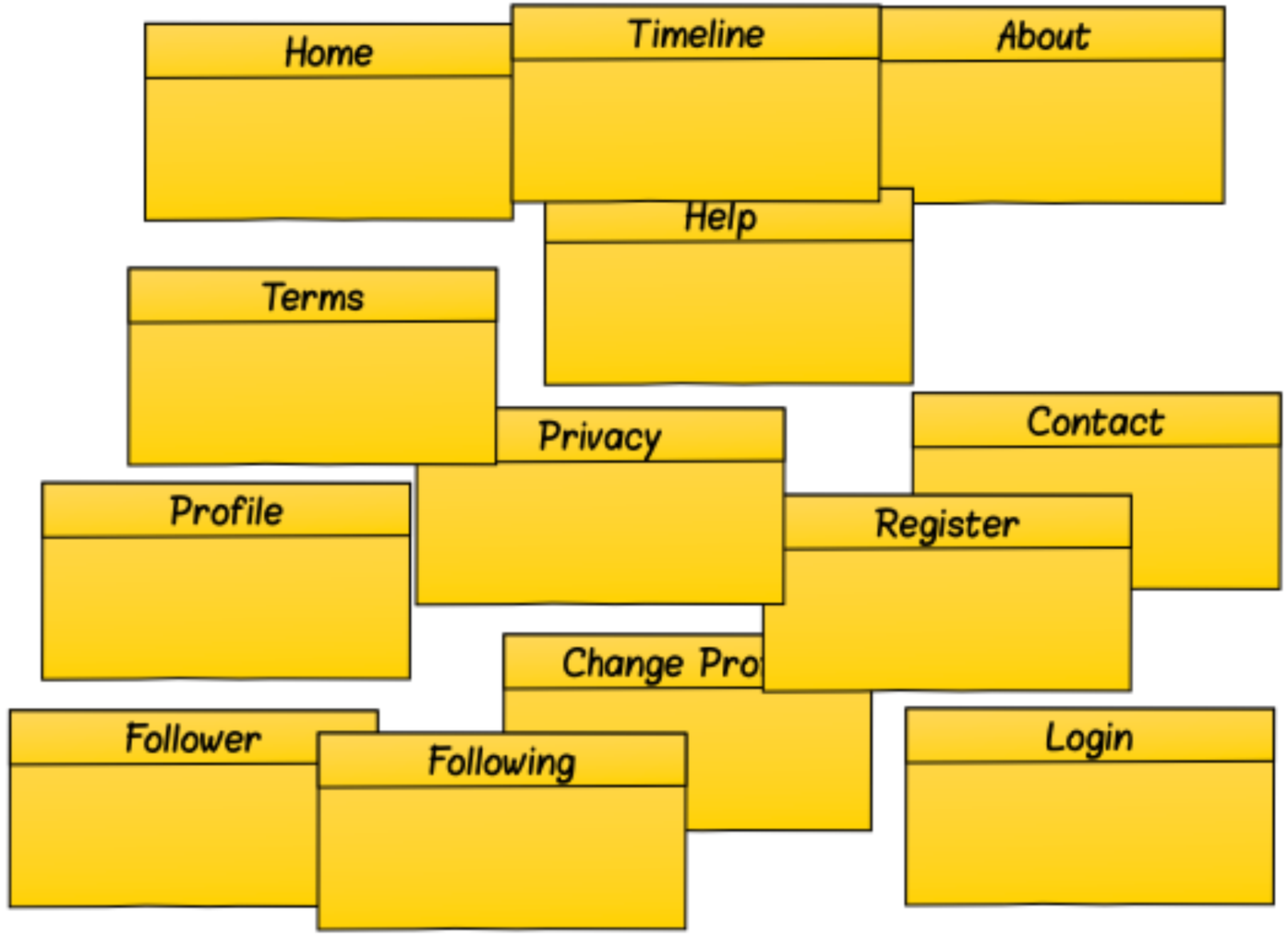
UI-ROUTER

“states” are the new “pages”

‘WEB 1.0’ WEBSITE


Home
Timeline
About
Help
Terms
Privacy
Contact
Profile
Register
Change Pro
Follower
Following
Login

Server

index.html

GET /

User clicks on link . . .

index.html

```
1  <!DOCTYPE html>
2  <html>
3  <title>Cool Website</title>
4  <head>
5  <script src="/jquery.min.js"></script>
6  <link href="/stylesheets/style.css"></link>
7  </head>
8  <body>
9    <nav>
10     <li><a href="/">Home</a></li>
11     <li><a href="/about">About</a></li>
12     <li><a href="/contact">Contact</a></li>
13   </nav>
14   <h1>Damn, they don't make websites like this anymore</h1>
15 </body>
16 </html>
```

Another server request

AND ANOTHER!

CLICK!

Server

about.html

GET /about

About.html

```
<!DOCTYPE html>
<html>
<title>Cool Website</title>
<head>
<script src="/jquery.min.js"></script>
<link href="/stylesheets/style.css"></link>
</head>
<body>
  <nav>
    <li><a href="/">Home</a></li>
    <li><a href="/about">About</a></li>
    <li><a href="/contact">Contact</a></li>
  </nav>
  <h1>About Me: I love giraffes</h1>
</body>
</html>
```

Yes, we are making these requests AGAIN.

Every time we navigate to a new page its a brand new event.

WEB 1.0 (NOT SPA)

- Views stored and rendered on the server, served up as HTML.
- When user goes to a new page, the browser navigates in totality, navigating, refreshing and retrieving a brand new HTML page.
- Each page, since it is a new page, retrieves stylesheets, script files, etc.

SINGLE PAGE APPLICATION (SPA)

Server

index.html

GET /

User clicks on link . . .

index.html

```
1  ...
2  <body ng-app="app" ng-controller="NavController">
3    <nav>
4      <li><a ng-click="requestBody('home')">Home</a></li>
5      <li><a ng-click="requestBody('about')">About</a></li>
6      <li><a ng-click="requestBody('contact')">Contact</a></li>
7    </nav>
8    <h1>{{bodyContent}}</h1>
9    <script>
10     app.controller('NavController', function ($scope, $http) {
11       $scope.bodyContent = "Home content";
12       $scope.requestBody = function (reqString) {
13         $http.get('/api/getBody/' + reqString)
14           .then(function (res) {
15             $scope.requestBody = res.data;
16           })
17       };
18     });
19   </script>
20 </body>
21 ...
```

Server

```
{
  data: “This is the about page”
}
```

GET /api/getBody/about

handles XHR response

index.html

{ data: “...” }

```
1  ...
2  <body ng-app="app" ng-controller="NavController">
3    <nav>
4      <li><a ng-click="requestBody('home')">Home</a></li>
5      <li><a ng-click="requestBody('about')">About</a></li>
6      <li><a ng-click="requestBody('contact')">Contact</a></li>
7    </nav>
8    <h1>{{bodyContent}}</h1>
9    <script>
10     app.controller('NavController', function ($scope, $http) {
11       $scope.bodyContent = "Home content";
12       $scope.requestBody = function (reqString) {
13         $http.get('/api/getBody/' + reqString)
14           .then(function (res) {
15             $scope.bodyContent = res.data;
16           })
17       };
18     });
19   </script>
20 </body>
21 ...
```

SINGLE PAGE APPLICATIONS

- On page change, a new page is not loaded. The front-end application replaces elements on existing DOM to update view.

- AJAX plays a big part to fill in data that would normally be served up by the server (think swig).

WE LOST SOME GOOD STUFF!

- **Browser History API** allows for control of URL and back/forward button, but this doesn't work if every change is handled with an **ng-click**

- **Search Engine Optimization** (SEO) - relies on classic routing!

- **Modularity** - routes are a nice abstraction for separating independent pages and defining parent/child relationships.

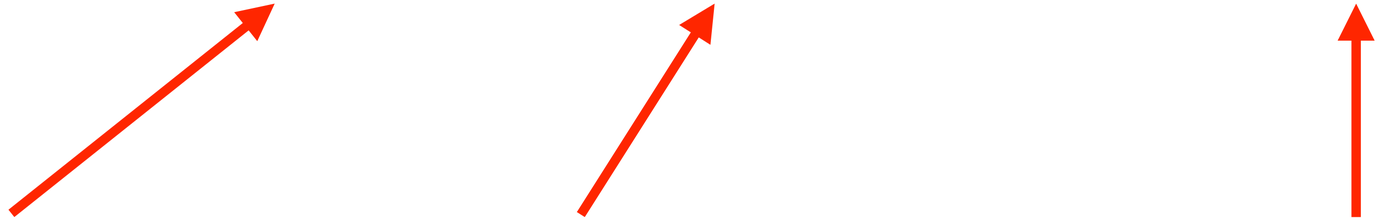
  / /settings /articles /articles/:articleID


UI

WHAT IS UI-ROUTER?

- An Angular-specific tool for management of different views in a single page application.
- Ties into URL and history to allow for easy navigation to and between different parts of your application.
- Easily integrates nesting of views.

STATE = URL + VIEW + CONTROLLER



localhost:3000/about

about.html

```
app.controller('AboutController', function ($scope) {
    // controls the state of the associated view
})
```

GETTING STARTED

1.
```
# in project root
npm install --save angular-ui-router
```

2.
```
<!-- in index.html -->
...
<script src="/route/for/angular-ui-router.js"></script>
...
```

3.
```
// in main app script
var theApp = angular.module('kittens', ['ui.router']);
```

CONFIGURING A STATE

```
// main app script
theApp.config(function ($stateProvider) {
  // registers a 'home' state for the url '/'
  $stateProvider.state('home', {
    url: '/',
    template: '<p>Best landing page ever</p>'
  });
});
```

```
<!-- index.html -->
<html>
  <head>...</head>
  <body>
    <div>I am common to all state views</div>
    <ui-view></ui-view>
  </body>
</html>
```

when a user visits the ‘/‘ route, view loads as…

```
<html>
  <head>...</head>
  <body>
    <div>I am common to all state views</div>
    <ui-view>
      <p>Best landing page ever</p>
    </ui-view>
  </body>
</html>
```

TWO STATES

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('home', {
    url: '/',
    template: '<p>Best landing page ever</p>'
  });
});

theApp.config(function ($stateProvider) {
  $stateProvider.state('contact', {
    url: '/about',
    template: '<p>Just shout really loudly</p>'
  });
});
```

```
<!-- index.html -->
...
<div>
  <p>I am common to all state views</p>
  <div>
    <!-- ui-sref allows us to link to another state -->
    <a ui-sref="home"></a>
    <a ui-sref="contact"></a>
  </div>
</div>
<ui-view></ui-view>
...
```

user clicks first link

```
...
<ui-view>
  <p>Best landing page ever</p>
</ui-view>
...
```

user clicks second link

```
...
<ui-view>
  <p>Just shout really loudly</p>
</ui-view>
...
```

STATE CONTROLLER

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('contact', {
    url: '/about',
    template: '<p>Just shout {{ adjective }} loudly</p>',
    controller: function ($scope) {
      $scope.adjective = 'really';
    }
  });
});
```

DYNAMIC STATE TRANSITION

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('contact', {
    url: '/about',
    template: '<p>Just shout really loudly</p>',
    controller: function ($state) {
      setTimeout(function () {
        // $state.go allows us to trigger state change from javascript
        $state.go('home');
      }, 3000);
    }
  });
});
```

TEMPLATE URL

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('gallery', {
    url: '/kittens',
    template: '<div ng-repeat="kitten in kittens"><p>{{ kitten.name }}</p></div>',
    controller: function ($scope, KittenFactory) {
      KittenFactory.fetchAll(function (kittens) {
        $scope.kittens = kittens;
      });
    }
  });
});
```

```
<!-- kitten-gallery.html -->
<div ng-repeat="kitten in kittens">
  <p>{{ kitten.name }}</p>
</div>
```

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('gallery', {
    url: '/kittens',
    templateUrl: '/route/for/kitten-gallery.html',
    controller: function ($scope, KittenFactory) {
      KittenFactory.fetchAll(function (kittens) {
        $scope.kittens = kittens;
      });
    }
  });
});
```

PARAMETERIZED STATES

register state

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('detail', {
    // specifying a state parameter 'kittenId'
    url: '/kittens/:kittenId',
    template: '<img ng-src="kitten.photoURL">',
    controller: function ($scope, KittenFactory, $stateParams) {
      var theId = $stateParams.kittenId;
      KittenFactory.fetchById(theId, function (theKitten) {
        $scope.kitten = theKitten;
      });
    }
  });
});
```

create link to state in html

```
<a ui-sref="detail({kittenId: someKitten.id})"></a>
```

transition to state in javascript

```
$state.go('detail', {kittenId: someKitten.id});
```

“PROBLEM”

currently at /kittens route

```
<html>
  <head>...</head>
  <body>
    <div>I am common to all state views</div>
    <ui-view>
      <div ng-repeat="kitten in kittens">
        <p>
          <a ui-sref="detail({kittenId: kitten.id})">
            {{ kitten.name }}
          </a>
        </p>
      </div>
    </ui-view>
  </body>
</html>
```

when user clicks on a kitten

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('detail', {
    url: '/kittens/:kittenId',
    template: '<img ng-src="kitten.photoURL">',
    controller: function ($scope, KittenFactory, $stateParams) {
      var theId = $stateParams.kittenId;
      KittenFactory.fetchById(theId, function (theKitten) {
        $scope.kitten = theKitten;
      });
    }
  });
});
```

```
<html>
  <head>...</head>
  <body>
    <div>I am common to all state views</div>
    <ui-view>
      <img ng-src="kitten.photoURL">
    </ui-view>
  </body>
</html>
```

Kitten list gets replaced by single image…

Instead how could we show the image off to the right?

CHILD STATES

```
...
<ui-view>
  <div ng-repeat="kitten in kittens">
    <p>
      <a ui-sref="detail({kittenId: kitten.id})">
        {{ kitten.name }}
      </a>
    </p>
  </div>
</ui-view>
<ui-view style="position:fixed; right:0;">
... <ui-view></ui-view>
  </div>
</ui-view>
...
```

¡nested ui view!

CHILD STATES

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('gallery', {...});
});
```

now detail is a child state of gallery

```
theApp.config(function ($stateProvider) {
  $stateProvider.state('gallery.detail', {...});
});
```

“SOLUTION”

currently at /kittens route

```
...
<ui-view>
  <div ng-repeat="kitten in kittens">
    <p>
      <a ui-sref=“gallery.detail({kittenId: kitten.id})”>
        {{ kitten.name }}
      </a>
    </p>
  </div>
  <div style="position:fixed; right:0;">
    <ui-view></ui-view>
  </div>
</ui-view>
...
```

when user clicks on a kitten

```
theApp.config(function ($stateProvider) {
  $stateProvider.state(‘gallery.detail', {
    url: '/:kittenId',
    template: '<img ng-src="kitten.photoURL">',
    controller: function ($scope, KittenFactory, $stateParams) {
      var theId = $stateParams.kittenId;
      KittenFactory.fetchById(theId, function (theKitten) {
        $scope.kitten = theKitten;
      });
    }
  });
});
```

STATES

- state = URL + view + controller
- states must be registered during in app.config
- state views “fill” the ui-view directive
- ui-sref is a directive that creates links from states
- $state.go is a method that can trigger transition to a state
- states can be parameterized
- child states “nest” into parent’s ui-view directive
- all of this is FRONTEND ONLY

ANGULAR SUMMARY

Defined for a URL