Lecture 6: AngularJS & socket.io

Olivier Liechti
TWEB

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

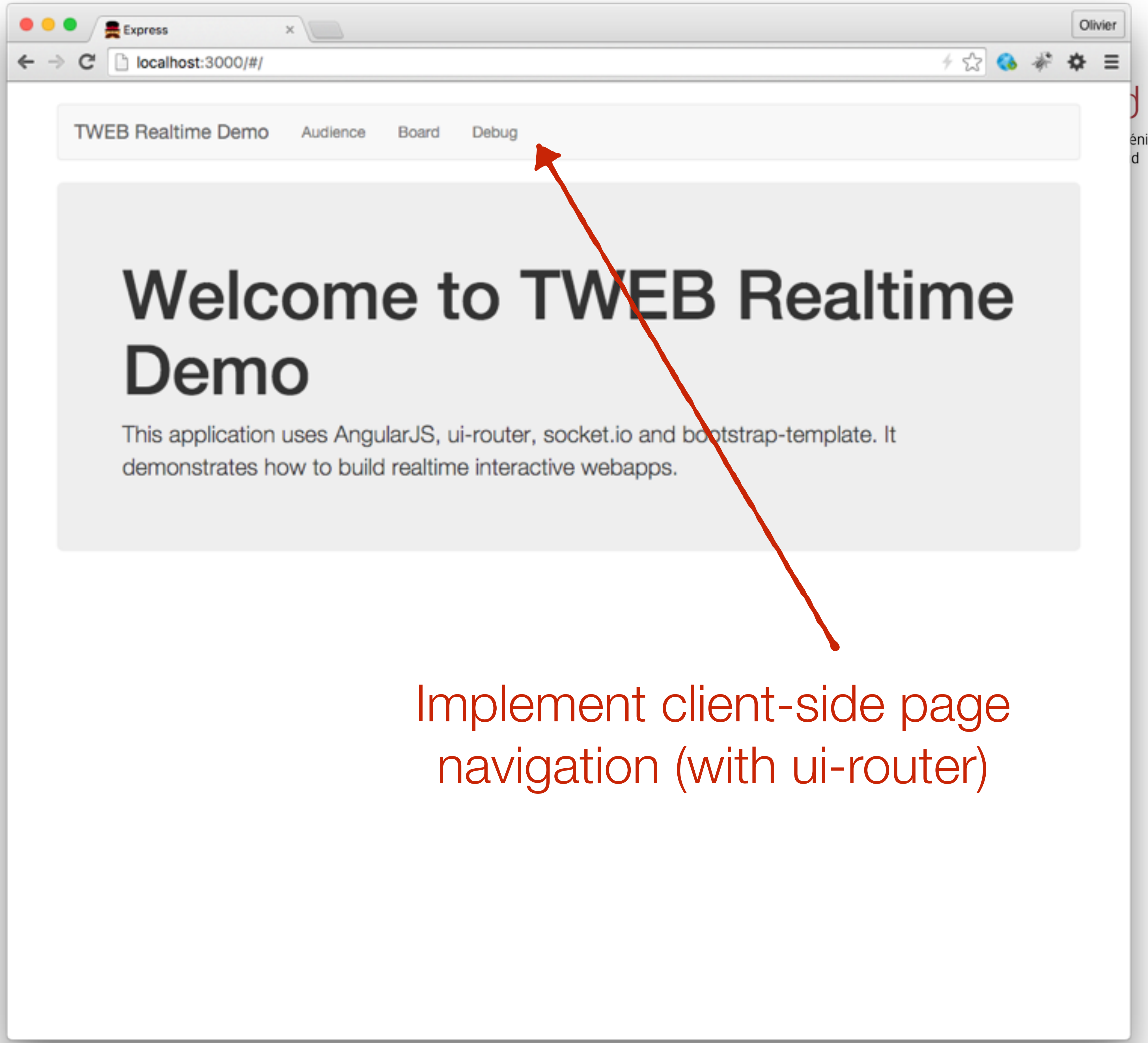
Today’s agenda

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

14h00 - 14h45	45’	**Lecture (40’)** Objectives of the day Quick AngularJS recap Visualization, socket.io and client-side navigation with AngularJS (intro)
14h45 - 16h00	75’	**Individual work (20’)** Objectives of the day
16h00 - 16h25	25’	**Review of your work** Some students will be asked to present their code. You have to submit your repo **individually**. **Hints from one implementation** GitHub repo will be provided later

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Objectives of the next 2 lectures

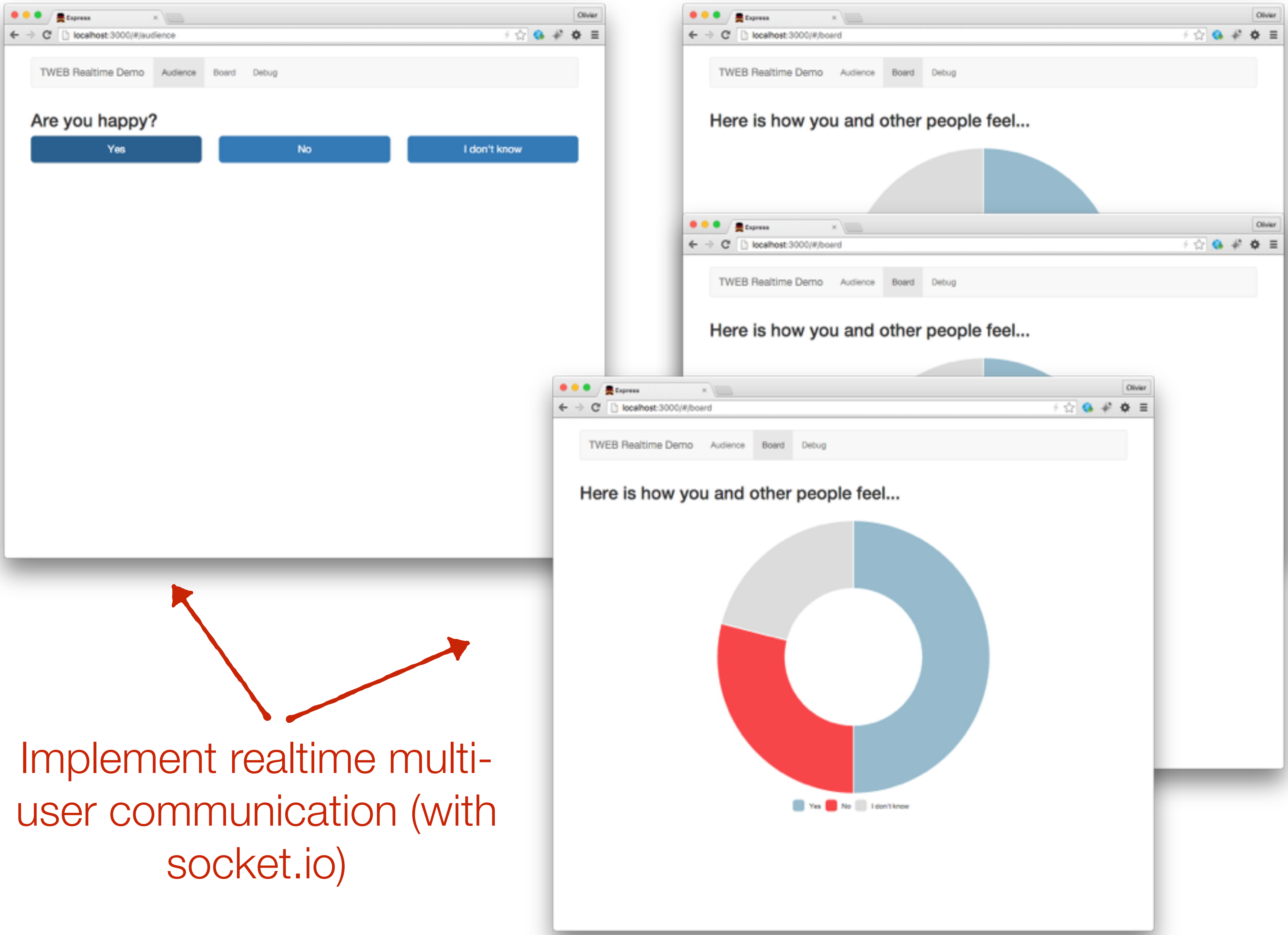


Welcome to TWEB Realtime Demo

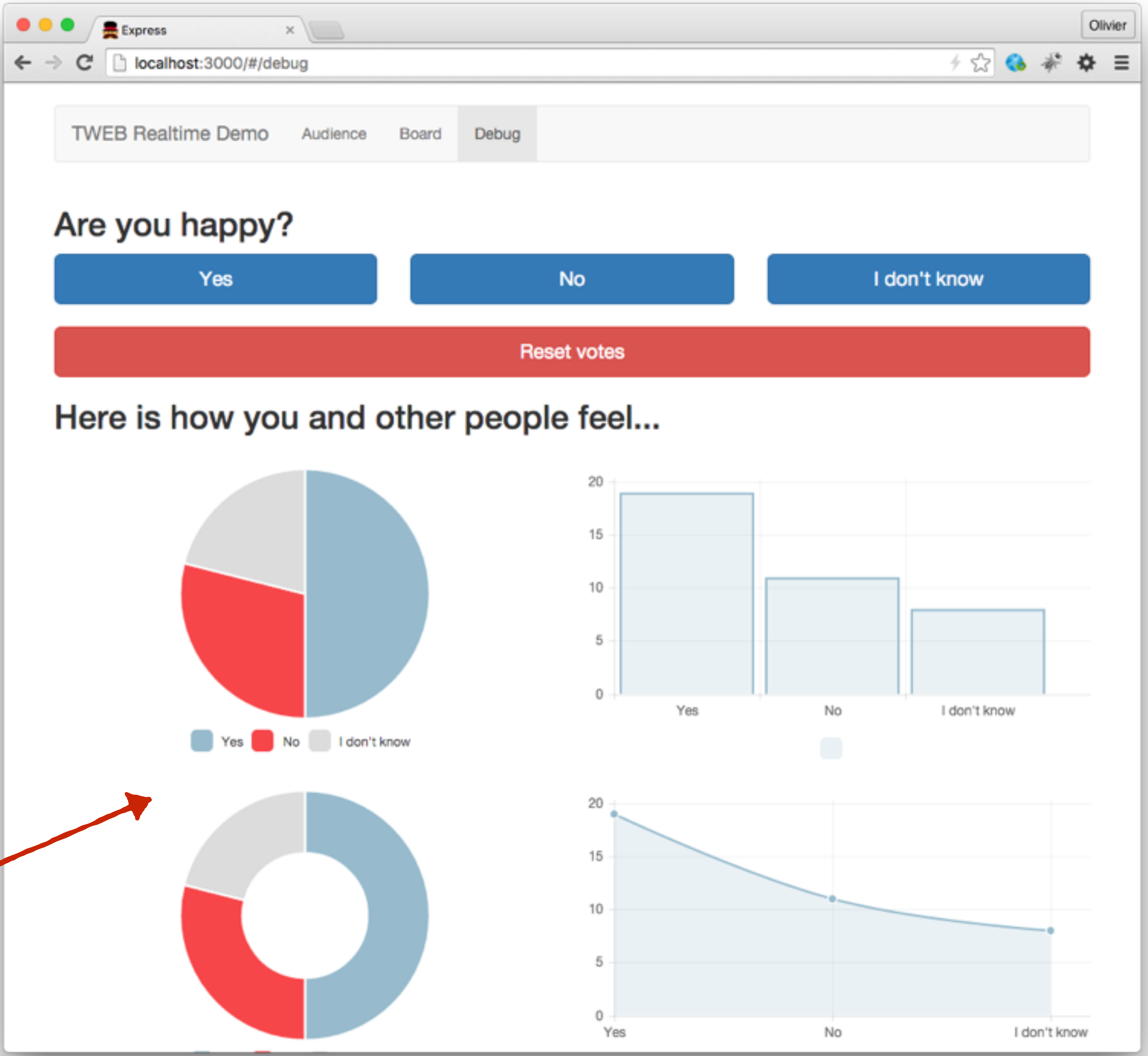
This application uses AngularJS, ui-router, socket.io and bootstrap-template. It demonstrates how to build realtime interactive webapps.

Implement client-side page navigation (with ui-router)

énierie et de Gestion
d



Implement realtime multi-user communication (with socket.io)



Are you happy?

Here is how you and other people feel...

Use charts in your AngularJS app

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

ANGULARJS
by Google

Quick recap

How do I bootstrap AngularJS?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

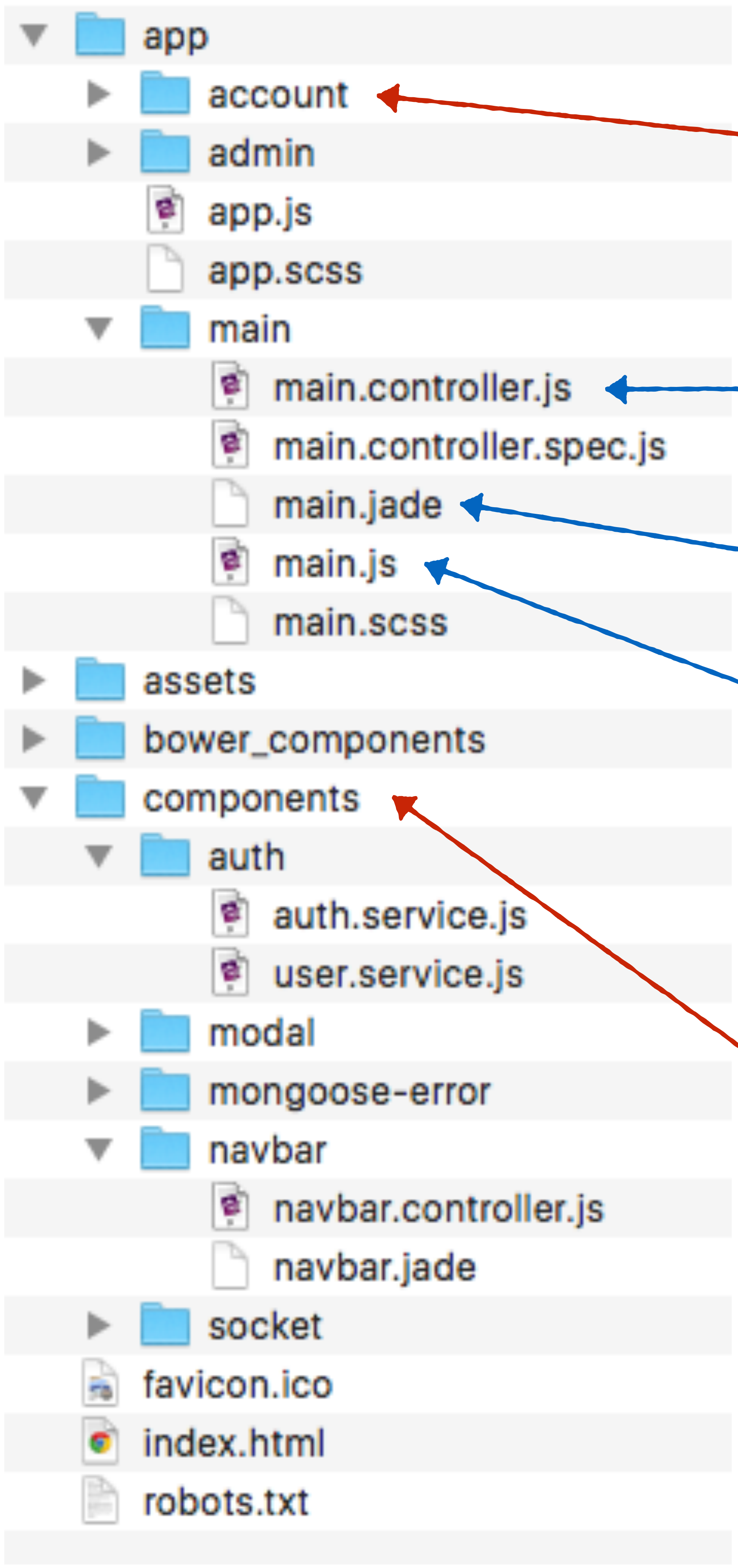
- To get started with AngularJS, you first need to **load the core framework** script. You can either use a **CDN**, download the file yourself, or use **bower**.
- You write **your code** in several scripts, which must also be loaded from index.html. In this example, all the code is in one script.

```
<html ng-app="tweb-demo-app">
   <body>
      <h1>Bootstrapping AngularJS</h1>

      <!-- load core AngularJS before any other AngularJS module -->
      <script src="js/angular.js"></script>

      <!-- load my AngularJS module -->
      <script src="js/tweb-demo-app.js"></script>
   </body>
<html>
```

Your script defines a **module** named "tweb-demo-app"



```
▼ app
  ▶ account
  ▶ admin
    app.js
    app.scss
  ▼ main
      main.controller.js
      main.controller.spec.js
      main.jade
      main.js
      main.scss
▶ assets
▶ bower_components
▼ components
  ▼ auth
      auth.service.js
      user.service.js
  ▶ modal
  ▶ mongoose-error
  ▼ navbar
      navbar.controller.js
      navbar.jade
  ▶ socket
  favicon.ico
  index.html
  robots.txt
```

The AngularJS components are organized **by feature** (in a hierarchy). This is now the best practice.

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

This file attaches a **controller** to the main module. The controller is used from the main.jade page fragment.

This file defines a **page fragment**. It contains AngularJS **directives**.

This is file is used to manage the **layout** (navigation).

The components folder contains AngularJS scripts that are **reused across features** (mostly services).

We **declare a new module** and give it a **name** ('twebApp'). Later, we will be able to lookup this module with `angular.module('twebApp')`, in other words by calling the module function without the second argument.

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
angular.module('twebApp', [
  'ngCookies',
  'ngResource',
  'ngSanitize',
  'btford.socket-io',
  'ui.router',
  'ui.bootstrap'
])
```

This will **lookup** the twebApp module.

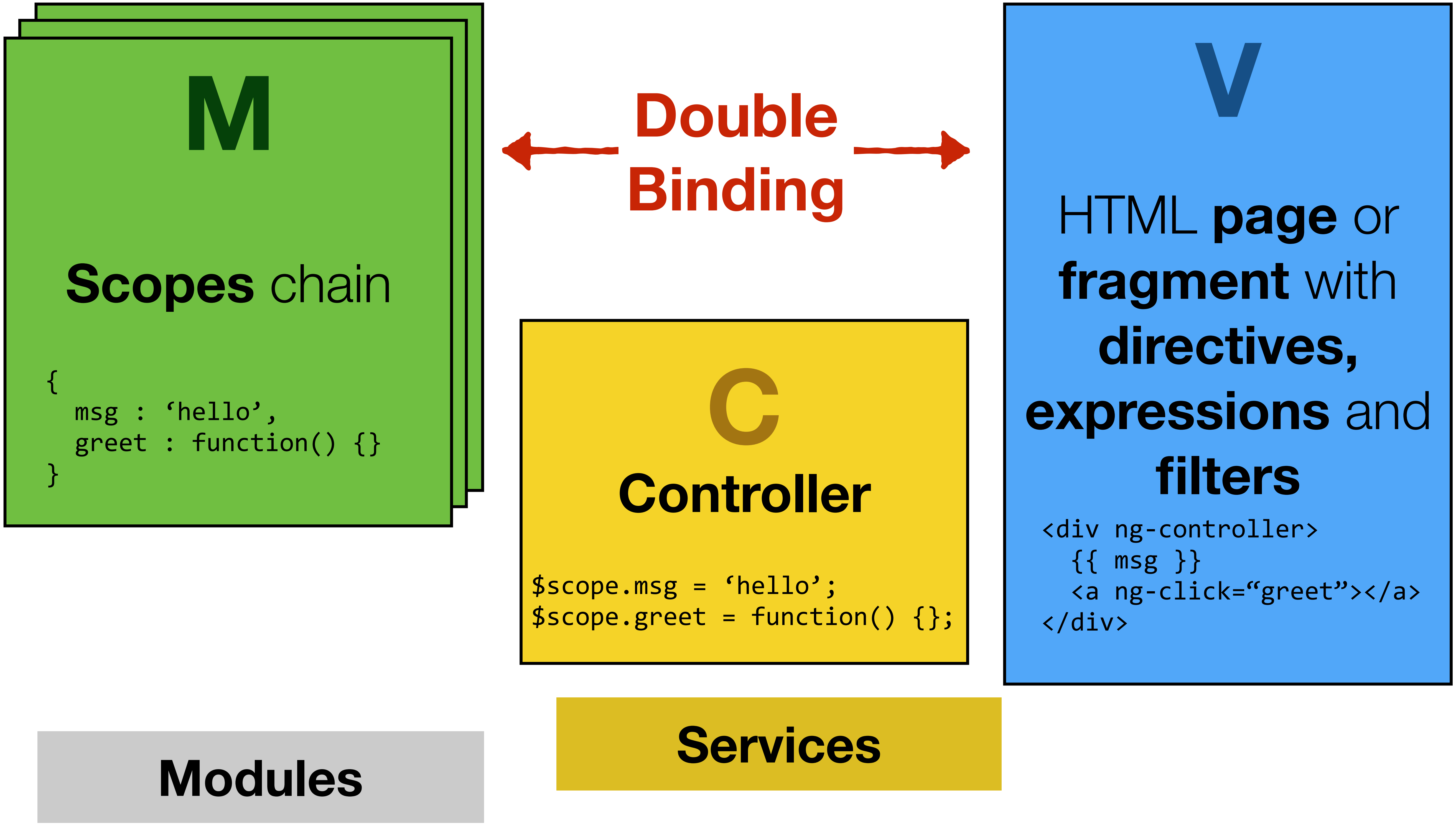
```
<body ng-app="twebApp">

  <!-- build:js({client,node_modules}) app/vendor.js -->
    <!-- bower:js -->
    <script src="bower_components/jquery/dist/jquery.js"></script>
    <script src="bower_components/angular/angular.js"></script>
    <script src="bower_components/angular-resource/angular-resource.js"></script>
    <script src="bower_components/angular-cookies/angular-cookies.js"></script>
    <script src="bower_components/angular-sanitize/angular-sanitize.js"></script>
    <script src="bower_components/angular-bootstrap/ui-bootstrap-tpls.js"></script>
    <script src="bower_components/lodash/dist/lodash.compat.js"></script>
    <script src="bower_components/angular-socket-io/socket.js"></script>
    <script src="bower_components/angular-ui-router/release/angular-ui-router.js"></script>
    <!-- endbower -->
    <script src="socket.io-client/socket.io.js"></script>
  <!-- endbuild -->
```

We **declare** that our module depends on 6 other modules (in this case, they are AngularJS and third-party modules). The corresponding *.js files must be **loaded in index.html**.

How does AngularJS implement MVC?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud


M
Scopes chain
Double Binding
V
HTML page or fragment with directives, expressions and filters
C
Controller
Modules
Services

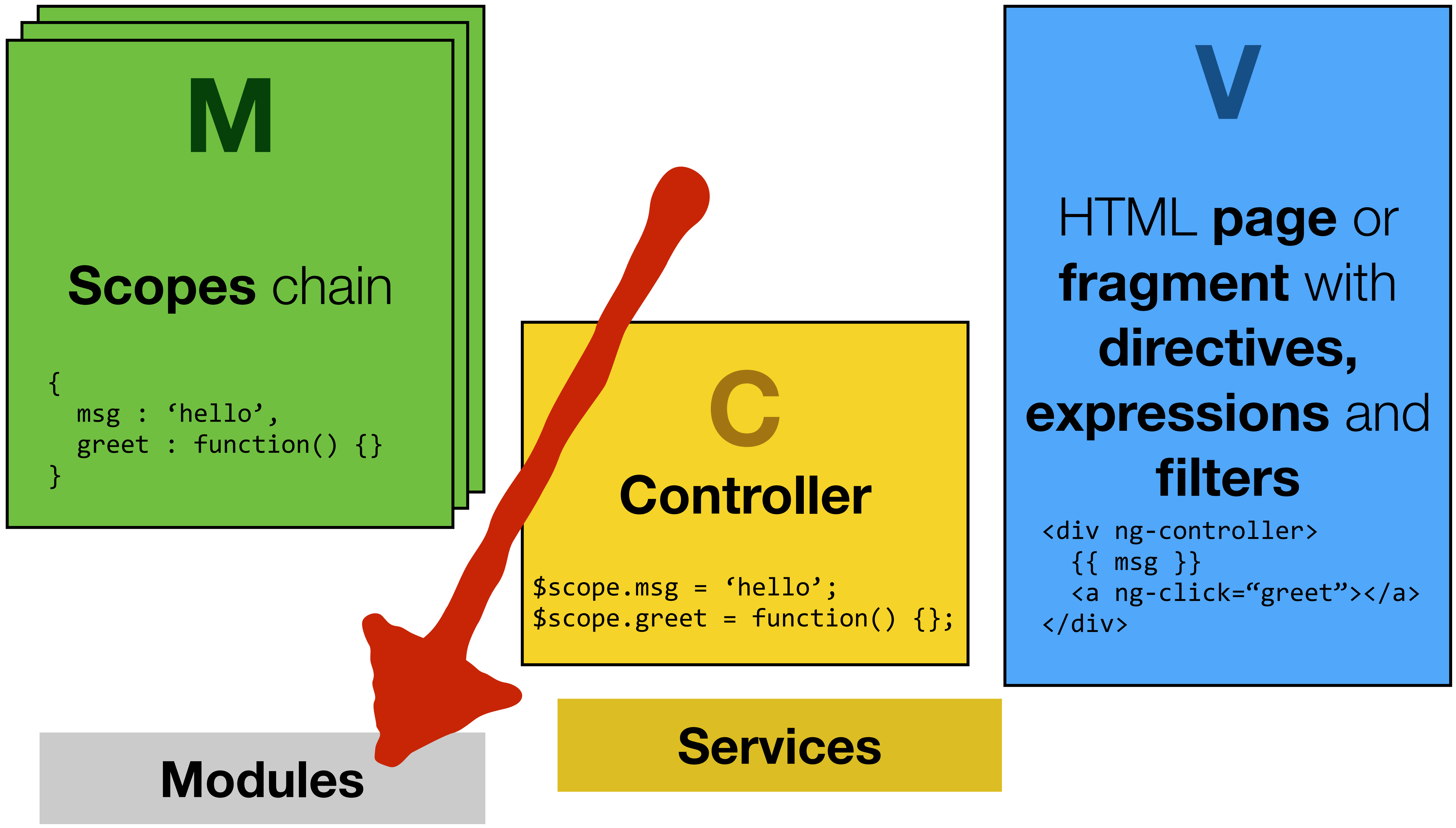
```
{
  msg : 'hello',
  greet : function() {}
}
```

```
$scope.msg = 'hello';
$scope.greet = function() {};
```

```
<div ng-controller>
  {{ msg }}
  <a ng-click="greet"></a>
</div>
```

How does AngularJS implement MVC?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud


M
Scopes chain
C
Controller
V
HTML **page** or **fragment** with **directives, expressions** and **filters**
Modules
Services

```
{
  msg : 'hello',
  greet : function() {}
}
```

```
$scope.msg = 'hello';
$scope.greet = function() {};
```

```
<div ng-controller>
  {{ msg }}
  <a ng-click="greet"></a>
</div>
```

What is a Module?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- When you develop an AngularJS application, you create **controllers**, **services**, **directives**, etc.
- At the minimum, you need to put your components in an application "**module**", which is loaded during the application **bootstrap**.
- If you have a large application, or if you want to share/reuse some of your components, it is a good idea to create **several modules**.
- You can think of modules as "**containers of components**".
- Modules can have **dependencies** on other modules.

This creates a new module, named 'tweb.users'. AngularJS will add it to its registry. The empty brackets mean that the module has no dependency on other modules.

```
angular.module('tweb.users', []);
```

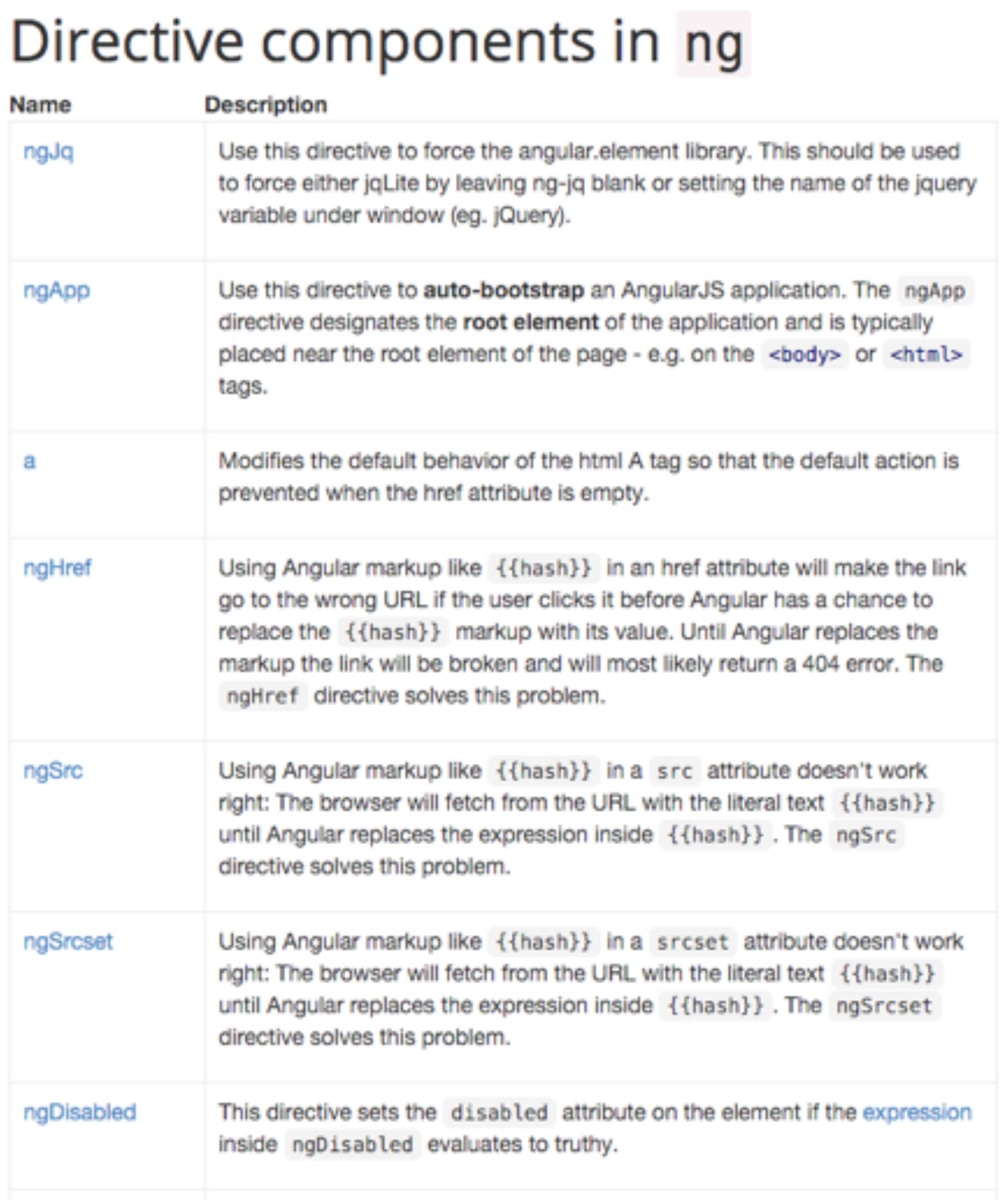
This looks up the module named 'tweb.users' in the AngularJS registry.

```
angular.module('tweb.users');
```

What is a Directive?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- An AngularJS directive is an **HTML extension** (e.g. a custom element, a custom attribute, which you include in your markup to **trigger some behavior**.
- AngularJS comes with a collection of **built-in directives**.
- **Third-party developers** have created additional directives.
- **You** can write your own directives.



Directive components in `ng`

Name	Description
ngJq	Use this directive to force the angular.element library. This should be used to force either jqLite by leaving ng-jq blank or setting the name of the jquery variable under window (eg. jQuery).
ngApp	Use this directive to **auto-bootstrap** an AngularJS application. The `ngApp` directive designates the **root element** of the application and is typically placed near the root element of the page - e.g. on the `<body>` or `<html>` tags.
a	Modifies the default behavior of the html A tag so that the default action is prevented when the href attribute is empty.
ngHref	Using Angular markup like `{{hash}}` in an href attribute will make the link go to the wrong URL if the user clicks it before Angular has a chance to replace the `{{hash}}` markup with its value. Until Angular replaces the markup the link will be broken and will most likely return a 404 error. The `ngHref` directive solves this problem.
ngSrc	Using Angular markup like `{{hash}}` in a `src` attribute doesn't work right: The browser will fetch from the URL with the literal text `{{hash}}` until Angular replaces the expression inside `{{hash}}`. The `ngSrc` directive solves this problem.
ngSrcset	Using Angular markup like `{{hash}}` in a `srcset` attribute doesn't work right: The browser will fetch from the URL with the literal text `{{hash}}` until Angular replaces the expression inside `{{hash}}`. The `ngSrcset` directive solves this problem.
ngDisabled	This directive sets the `disabled` attribute on the element if the expression inside `ngDisabled` evaluates to truthy.

https://docs.angularjs.org/api/ng/directive

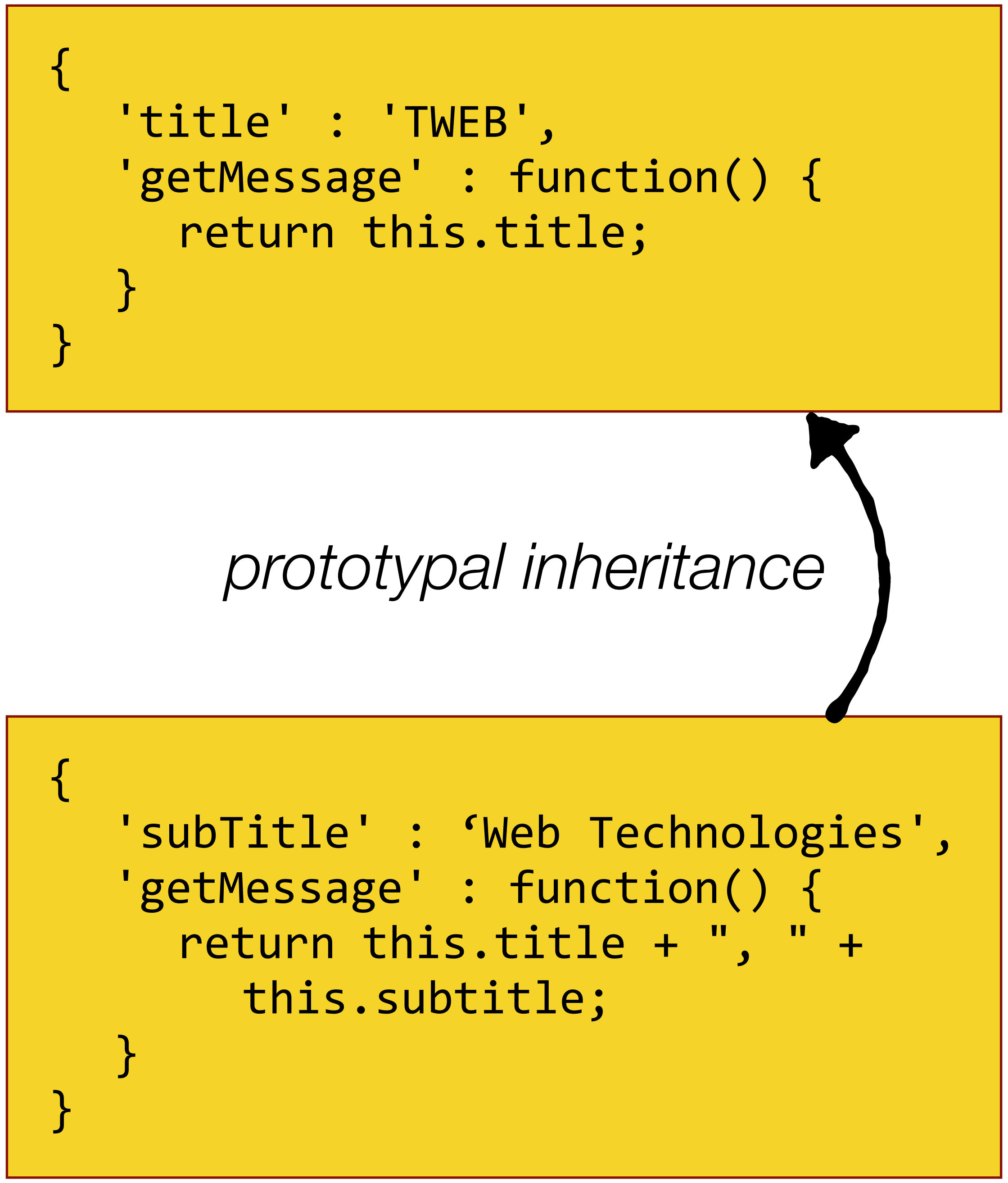
Which directives will use quickly?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

ngApp	Use this directive to **auto-bootstrap** an AngularJS application. The ngApp directive designates the root element of the application and is typically placed near the root element of the page - e.g. on the **<body>** or **<html>** tags.
ngController	The ngController directive **attaches a controller class to the view**. This is a key aspect of how angular supports the principles behind the **Model-View-Controller** design pattern.
ngModel	The ngModel directive **binds an input,select, textarea** (or custom form control) to a **property on the scope** using NgModelController, which is created and exposed by this directive.
ngRepeat	The ngRepeat directive **instantiates a template once per item from a collection**. Each template instance **gets its own scope**, where the given loop variable is set to the current collection item, and $index is set to the item index or key.
ngClick	The ngClick directive allows you to specify **custom behavior when an element is clicked**.
ngInclude	Fetches, compiles and includes an **external HTML fragment**.
ngClass	The ngClass directive allows you to **dynamically set CSS classes** on an HTML element by databinding an expression that represents all classes to be added.

What is a Scope?

- An Angular **scope** is a JavaScript **object**, created by the framework.
- It has **properties**, some of which are **functions**. The properties can be displayed in the view. The functions can be called from the view.
- Scopes are created at different levels of the **DOM** (e.g. at the level of a <DIV> node).
- Scopes are organized in a **prototypal inheritance chain**:
  - A scope often extends another scope.
  - The common ancestor of most scopes (i.e. non isolated scopes) is called **$rootScope**.



```
{
  'title' : 'TWEB',
  'getMessage' : function() {
    return this.title;
  }
}
```

```
{
  'subTitle' : ‘Web Technologies',
  'getMessage' : function() {
    return this.title + ", " +
      this.subtitle;
  }
}
```

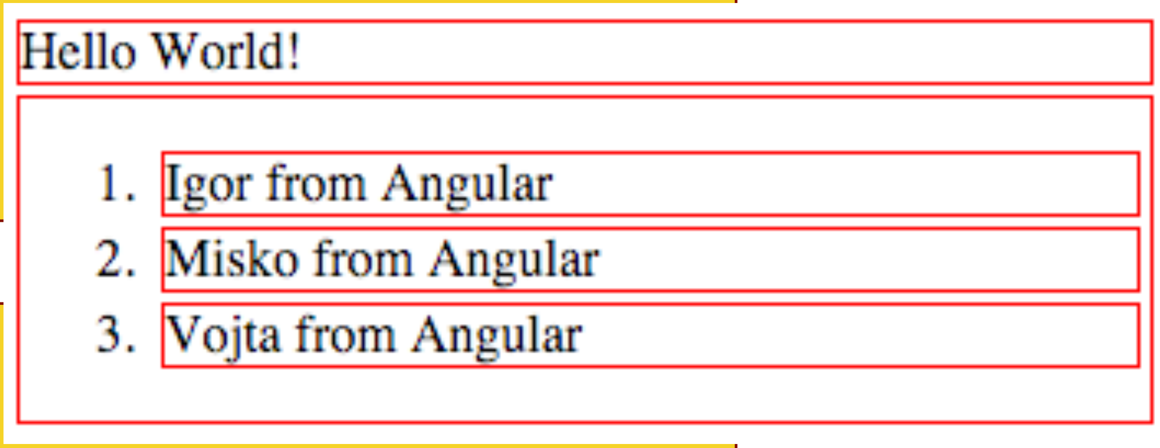
These 3 directives create a new scope

What is a Scope?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
<div class="show-scope-demo">
  <div ng-controller="GreetController">
    Hello {{name}}!
  </div>
  <div ng-controller="ListController">
    <ol>
      <li ng-repeat="name in names">{{name}} from {{department}}</li>
    </ol>
  </div>
</div>
```

The scope created by this directive inherits from the scope created by ListController, which inherits from $rootScope



Hello World!

1. Igor from Angular
2. Misko from Angular
3. Vojta from Angular

```
angular.module('scopeExample', [])

.controller('GreetController', ['$scope', '$rootScope',
function($scope, $rootScope) {
  $scope.name = 'World';
  $rootScope.department = 'Angular';
}])

.controller('ListController', ['$scope', function($scope) {
  $scope.names = ['Igor', 'Misko', 'Vojta'];
}]);
```

What is a Controller?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- An AngularJS controller is used to **initialize a scope** and to **attach behavior** (functions) to it.

This will create a new scope, which will be managed by an instance of a controller named “SpicyController”.

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

We can access the scope properties and invoke the functions.

```
<div ng-controller="SpicyController">
 <input ng-model="customSpice">
 <button ng-click="spicy('chili')">Chili</button>
 <button ng-click="spicy(customSpice)">Custom spice</button>
 <p>The food is {{spice}} spicy!</p>
</div>
```

This adds a controller named “SpicyController” to our “spicyApp2” module.

This initializes the “customSpice” and the “spice” properties (they will be available in the view).

```
var myApp = angular.module('spicyApp2', []);

myApp.controller('SpicyController', ['$scope', function($scope) {
    $scope.customSpice = "wasabi";
    $scope.spice = 'very';

    $scope.spicy = function(spice) {
        $scope.spice = spice;
    };
}]);
```

This adds function to the scope. It will be available in the view.

https://docs.angularjs.org/guide/controller

How does AngularJS implement MVC?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

M

Scopes chain

```
{
  msg : 'hello',
  greet : function() {}
}
```

C

Controller

```
$scope.msg = 'hello';
$scope.greet = function() {};
```

V

HTML **page** or **fragment** with **directives,** **expressions** and **filters**

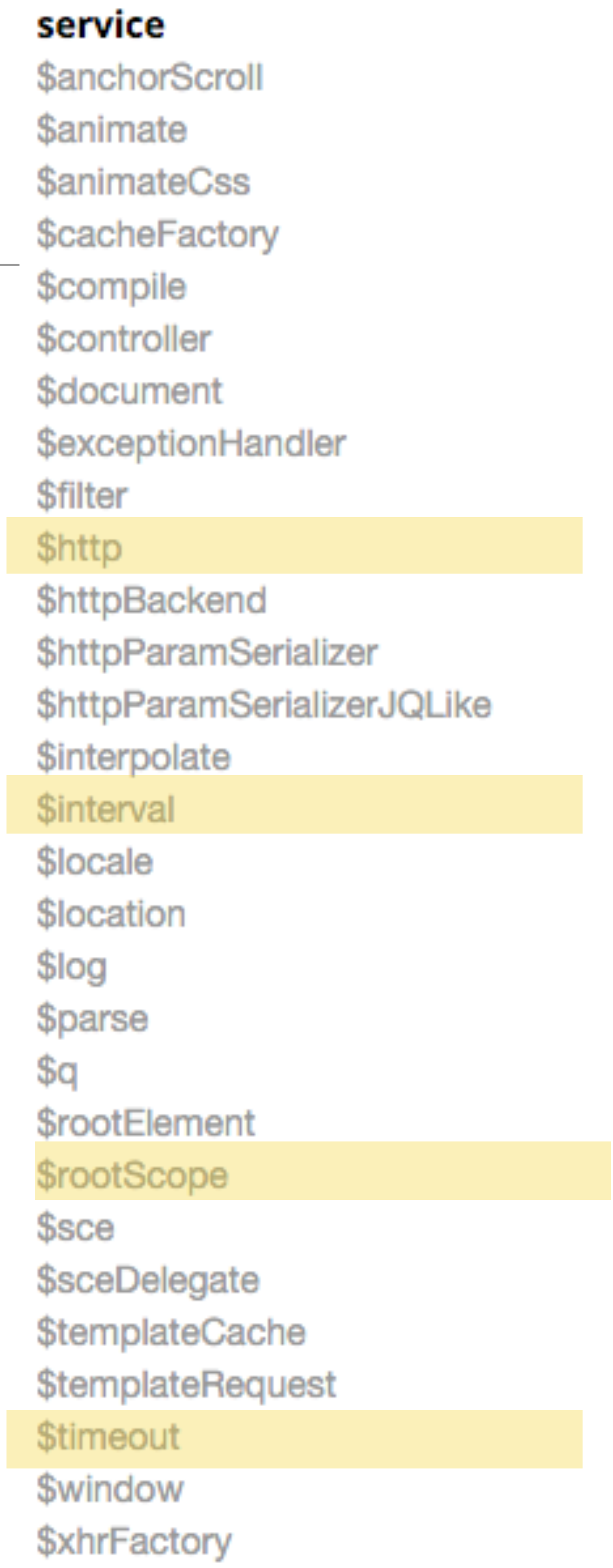
```
<div ng-controll[illegible]
  {{ msg }}
  <a ng-cl[illegible]"greet"></a>
</div>
```

Services

Modules

What is a Service?

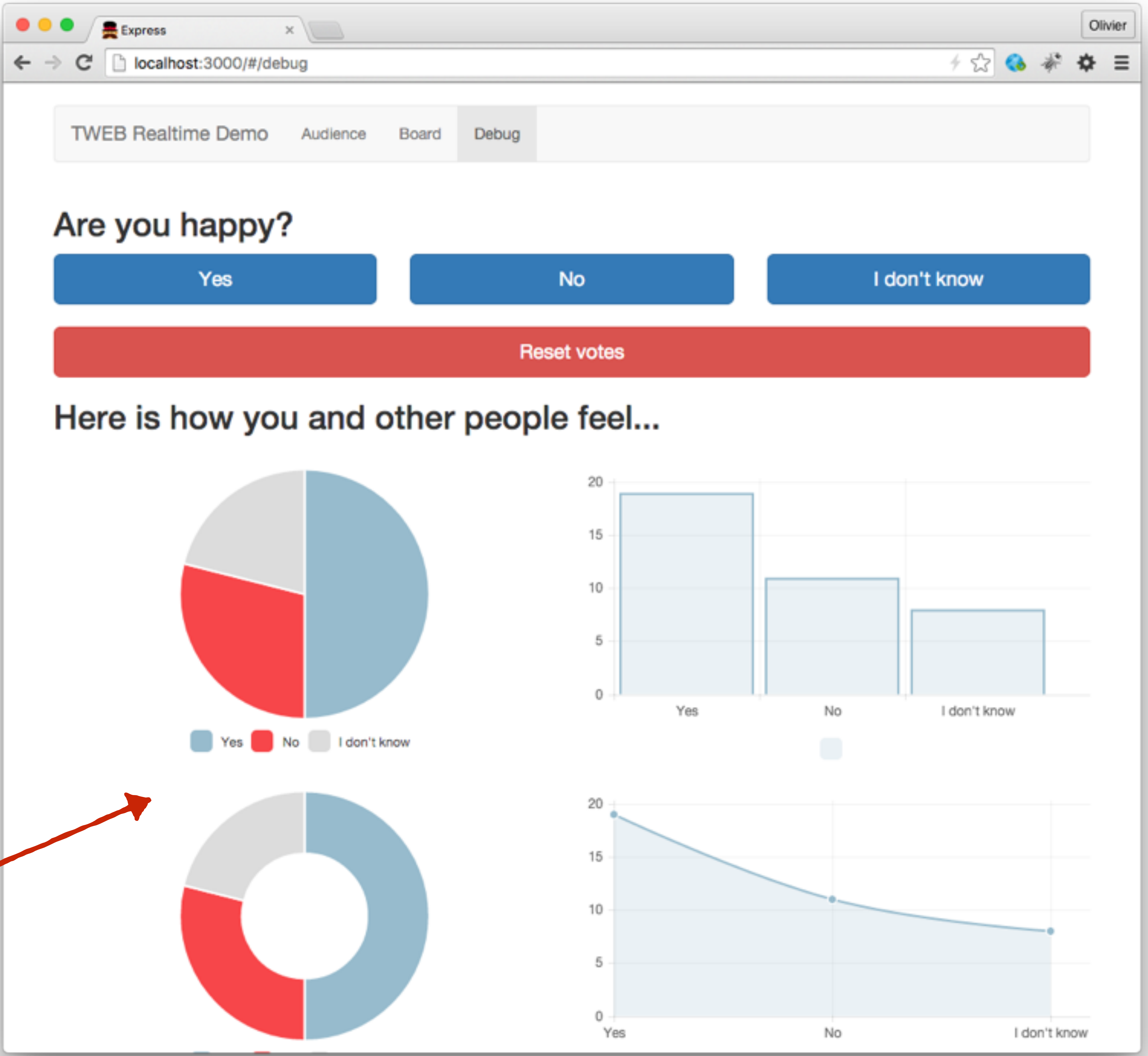
- AngularJS services are **singleton objects** that can be injected in controllers and that provide some functionality.
- It is a good practice to keep **controllers small**. For this reason, most of the complex behavior should be delegated to a service.
- A good example is the code that deals with **AJAX** requests.
- AngularJS provides a list of **built-in services**.
- You can implement **your own services**.
- **You can inject services in controllers, directives and other services. AngularJS keeps a registry of services defined by loaded modules.**



service
$anchorScroll
$animate
$animateCss
$cacheFactory
$compile
$controller
$document
$exceptionHandler
$filter
$http
$httpBackend
$httpParamSerializer
$httpParamSerializerJQLike
$interpolate
$interval
$locale
$location
$log
$parse
$q
$rootElement
$rootScope
$sce
$sceDelegate
$templateCache
$templateRequest
$timeout
$window
$xhrFactory

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

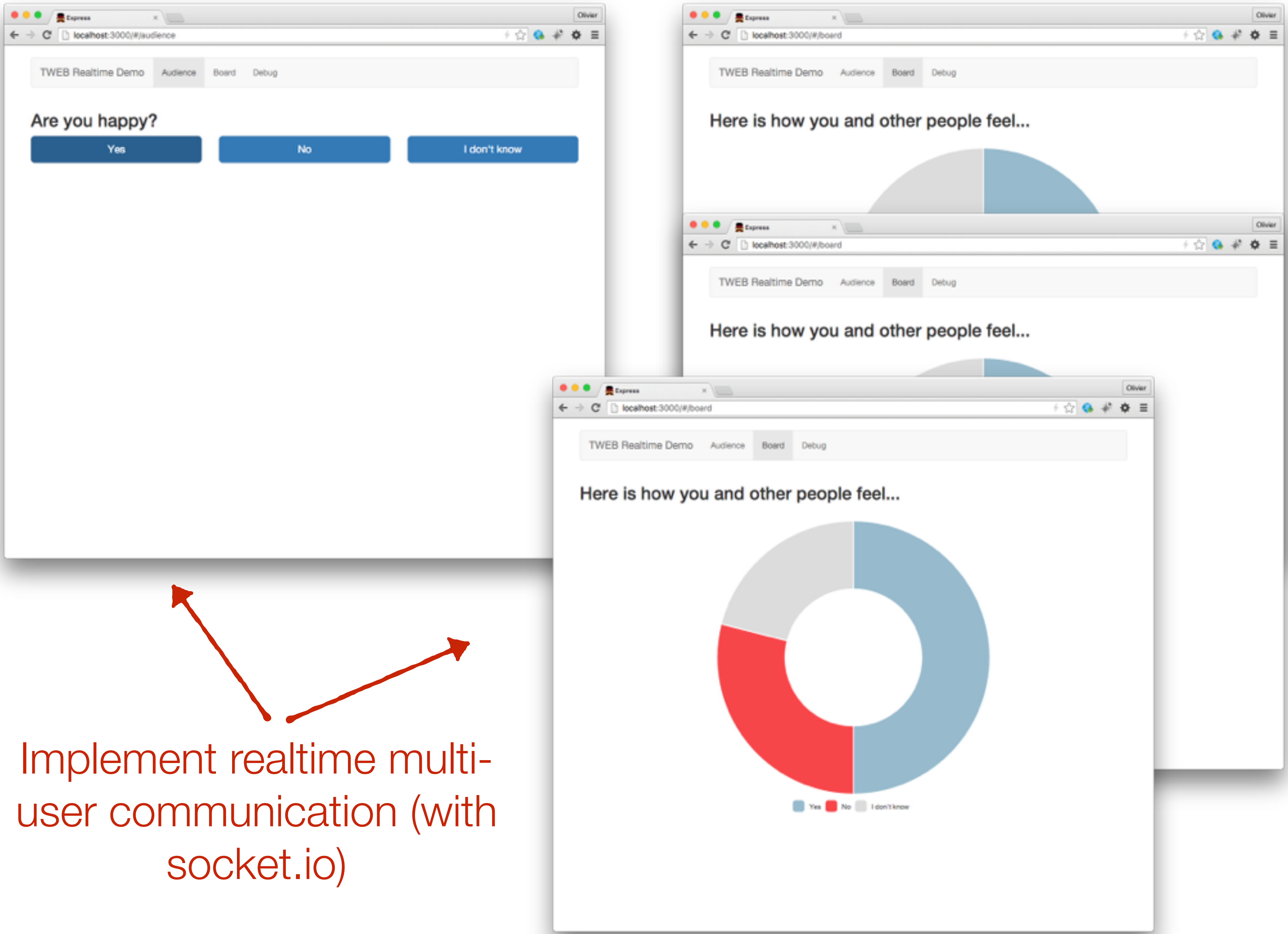
Back to the objectives of the day



Are you happy?

Here is how you and other people feel...

Use charts in your AngularJS app

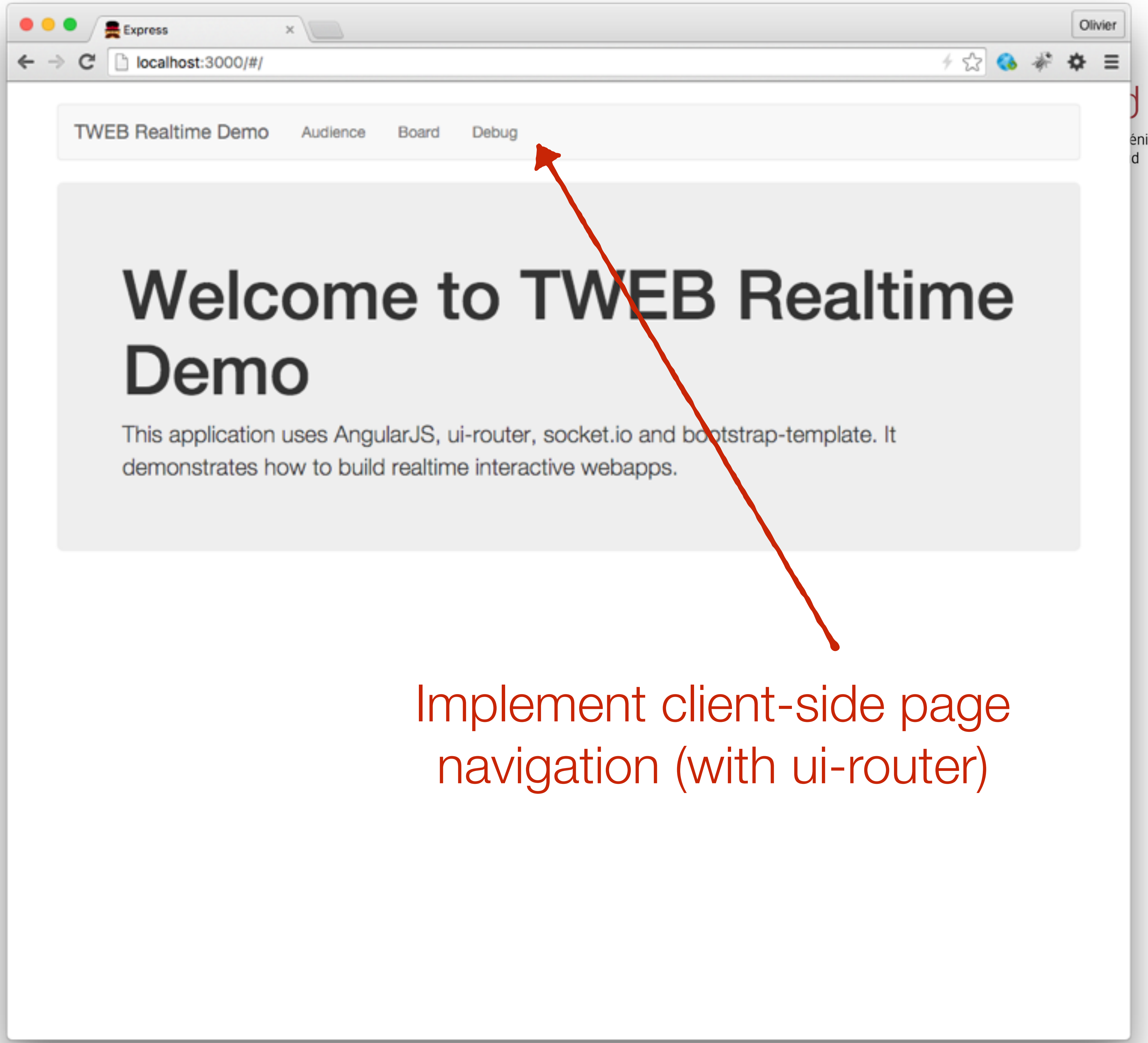

Are you happy?

Here is how you and other people feel...

Here is how you and other people feel...

Here is how you and other people feel...

Implement realtime multi-user communication (with socket.io)



Welcome to TWEB Realtime Demo

This application uses AngularJS, ui-router, socket.io and bootstrap-template. It demonstrates how to build realtime interactive webapps.

Implement client-side page navigation (with ui-router)

énierie et de Gestion
d

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Chart.js and Angular Chart

Overview

- **Chart.js** is one of the many JavaScript **visualization libraries**. You don't need AngularJS to use it.
- **angular-chart.js** is a set of custom **AngularJS directives**. It makes it very easy to add graphs in your app, based on the data attached to your AngularJS **$scopes**.
- This integration is **really easy**.
- It is also interesting to **study the angular-chart.js** source code to see how they have implemented the directives (if you want to develop your own directives).
- **What you need to remember**: how to create an AngularJS module for your app, how to specify that it has a dependency on angular-chart.js, how to create a controller and attach state to a scope.

Howto

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
<body ng-app="app" id="top">
  <div ng-controller="DoughnutCtrl">

     <canvas id="doughnut" class="chart chart-doughnut"
                   chart-data="data" chart-labels="labels">
     </canvas>

   </div>
</body>
```

the chart data is provided by the scope managed by the controller

```
angular.module("app", ["chart.js"])

.controller("DoughnutCtrl", function ($scope) {
  $scope.labels = ["Download Sales", "In-Store Sales", "Mail-Order Sales"];
  $scope.data = [300, 500, 100];
});
```

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

socket.io

Overview: server push

- In the "**traditional**" Web model, clients send requests to servers to **pull data**.
- In "**modern**" web applications:
    - the server often want to **push data** to clients
    - there is often a need to support **realtime communication** between different pages and/or users (chat, games, collaborative editors, etc.)
- **Various solutions** have been proposed for that purpose, for a very long time:
    - **polling** via AJAX (poor efficiency and high latency)
    - **long polling** (client sends a request, server does not respond immediately and only when it has an event to send to the client)
    - streaming over **persistent HTTP connections**
- The **WebSocket protocol and API** is now the recommended solution and is well supported by browsers (also on mobile).

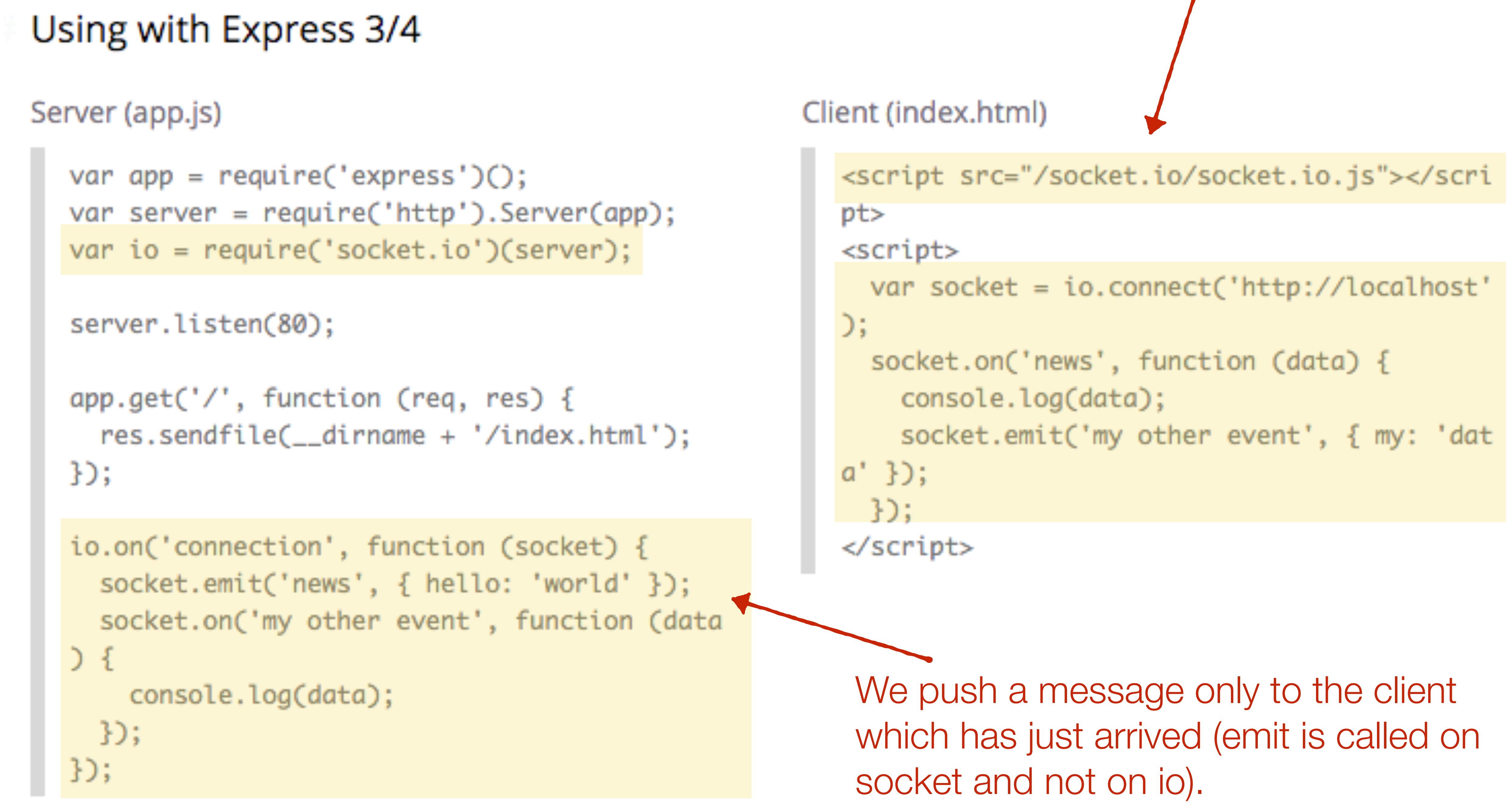
Overview: socket.io

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **socket.io** is a JavaScript layer which makes it **very easy** to use the WebSocket protocol in your applications.
- **socket.io** also has the ability to **fallback** on alternative, older push mechanisms if WebSocket is not supported.
- **socket.io** adds functionality on top of the WebSocket API:
  - broadcasting to other connected clients
  - Error handling (reconnections)
  - Namespaces (think “chat rooms”)
  - etc.

Quick start

The npm module in the back-end will serve this script - no need to copy it in your public folder



Using with Express 3/4

Server (app.js)

```
var app = require('express')();
var server = require('http').Server(app);
var io = require('socket.io')(server);

server.listen(80);

app.get('/', function (req, res) {
  res.sendfile(__dirname + '/index.html');
});

io.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data
) {
    console.log(data);
  });
});
```

Client (index.html)

```
<script src="/socket.io/socket.io.js"></scri
pt>
<script>
  var socket = io.connect('http://localhost'
);
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'dat
a' });
  });
</script>
```

We push a message only to the client which has just arrived (emit is called on socket and not on io).

Messaging patterns

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
socket.emit('welcome', { hello: 'only to the person who just connected' });
socket.broadcast.emit('welcome', { hello: 'to all except the person who just connected' });
socketio.emit('welcome', { hello: 'to absolutely everybody' });
```

When you receive a message on the socket, you can:
- send a response, only to the sender of this message
- forward a notification to all other clients
- send message to all clients (including the sender of this message)

You can also use **namespaces** and **rooms** to organize the flow of your messages (see http://socket.io/docs/rooms-and-namespaces/)

Using socket.io with AngularJS

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- There are **different third-party AngularJS modules** that provide an integration with socket.io.
- If you look at their code, you will see that it is quite brief. One thing that they do is to make sure that the **events** that happen on the socket.io side are well integrated with the AngularJS **page rendering process**.
- This makes it easy to use **bindings** between the socket.io service, the scopes and the views.
- One module that works well is **angular-socket-io** (https://github.com/btford/angular-socket-io)

angular-socket-io

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
angular.module('myApp', [‘btford.socket-io'])

  .factory('mySocket', function (socketFactory) {
    return socketFactory();
  })

  .controller('MyCtrl', function (mySocket) {
  // ...
  });
```

btford.socket-io module gives a service called “**socketFactory**”. We use it to create our service that we call “**mySocket**”.

We can then inject **mySocket** in our controllers and services. This service exposes the **socket.io API**.

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

angular-ui / ui-router

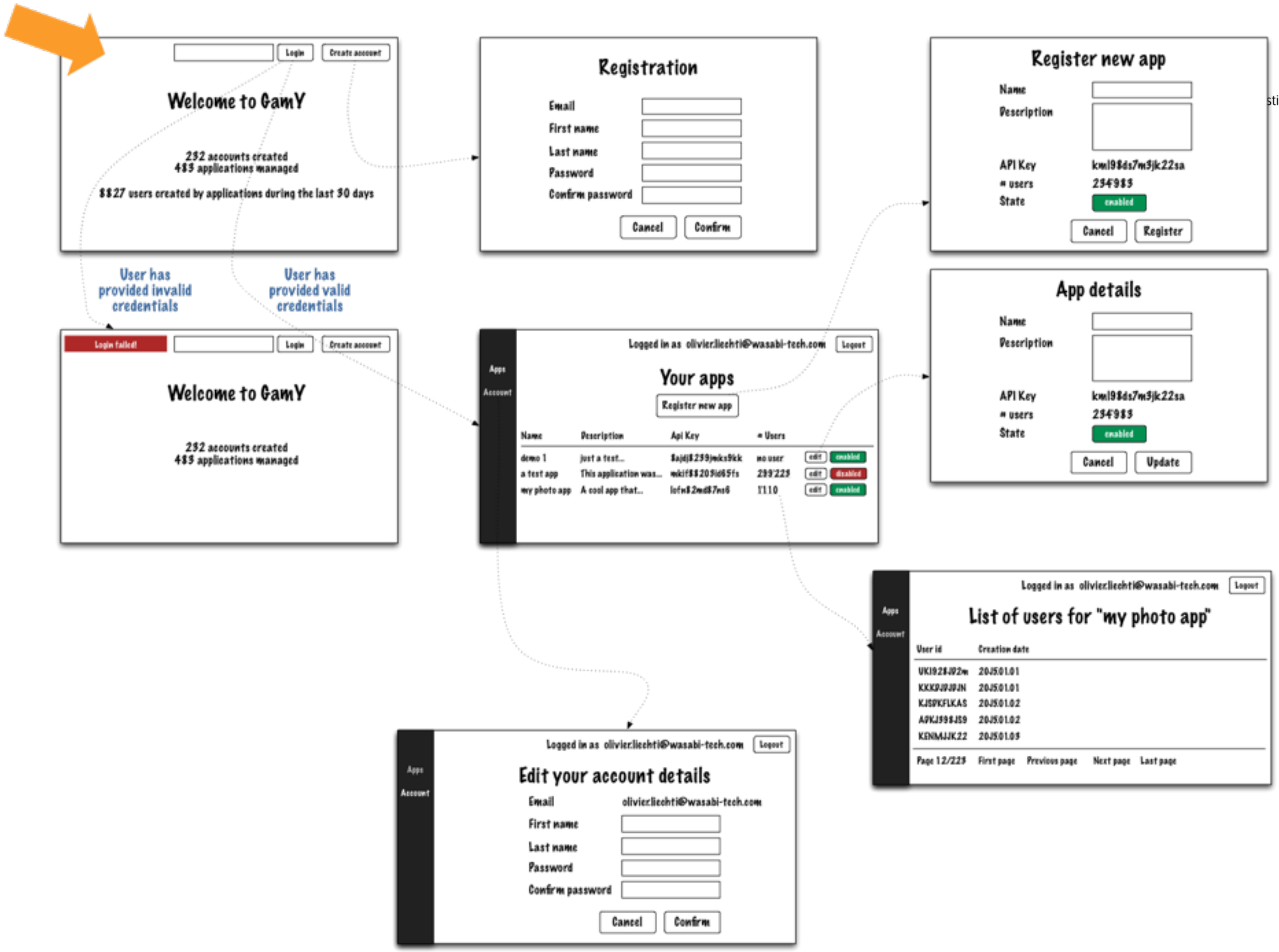
ui-router is a very popular alternative to the ngRoute service provided by AngularJS

AngularUI Router is a **routing framework** for AngularJS, which allows you to organize the parts of your interface into a state machine. Unlike the $route service in the Angular ngRoute module, which is organized around URL routes, **UI-Router is organized around states, which may optionally have routes, as well as other behavior, attached**.

States are bound to named, nested and parallel views, allowing you to powerfully manage your application's interface.

https://github.com/angular-ui/ui-router

https://github.com/angular-ui/ui-router/wiki/Quick-Reference

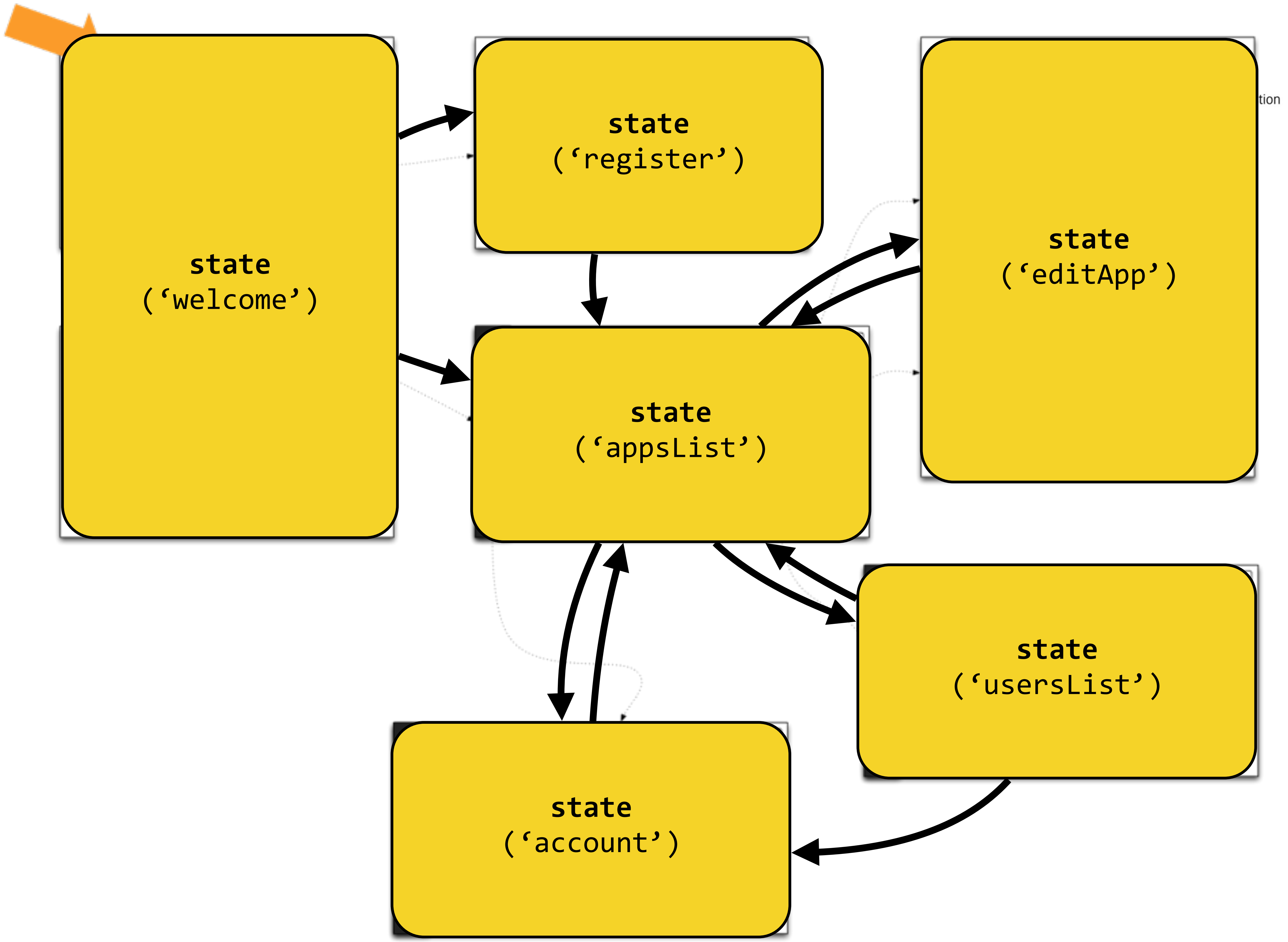


Name	Description	Api Key	# Users		
demo 1	just a test...	8ajdj8239jmks9kk	no user	edit	enabled
a test app	This application was...	mkif88209id65fs	299'223	edit	disabled
my photo app	A cool app that...	lofw82md87ms6	1'110	edit	enabled

User id	Creation date
UKI928J92m	2015.01.01
KKKDJJJJN	2015.01.01
KJSDKFLKAS	2015.01.02
ADKJ998J89	2015.01.02
KENMJJK22	2015.01.03

Page 12/223 First page Previous page Next page Last page

stion


state ('welcome')
state ('register')
state ('editApp')
state ('appsList')
state ('usersList')
state ('account')
tion

Basic example

```
angular.module("myApp", ["ui.router"])

.config(function( $stateProvider ) {

    $stateProvider.state('welcome', {
      templateUrl: 'partials/welcome.html',
      url: '/welcome'
    });

    $stateProvider.state('about', {
      templateUrl: 'partials/about.html',
      url: '/about'
    });

});
```

This function is executed when the myApp module is **loaded**. We can configure the $stateProvider service (provided by ui.router).

A state has a **name** (about) and a **config object** with quite a few properties. Here, we only define the **page fragment** that will be injected in the **ui-view** element and the url that will be displayed in the **navigation bar** when the state is active.

```
<body ng-controller="MainCtrl">
   <a ui-sref="welcome">Home</a> | <a ui-sref="about">About</a>
  <section ui-view></section>
</body>
```

Learning how to use ui-router

- **Information is provided by the authors of the module**:
    - In a **short tutorial**: http://angular-ui.github.io/ui-router/
    - On the **GitHub wiki**: https://github.com/angular-ui/ui-router/wiki
    - In the **API reference**: http://angular-ui.github.io/ui-router/site/#/api/ui.router
    - In a **sample application**: http://angular-ui.github.io/ui-router/sample/#/ (source: https://github.com/angular-ui/ui-router/tree/master/sample)
- The **angular-fullstack generator** uses ui-router (well, it gives you the choice when you generate your skeleton).

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Your turn

Proposed approach (1)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **Create a new project express project:**
    - I would start with the **express** yeoman generator (lightweight and will be faster than to remove stuff from a skeleton generated with angular-fullstack).
- **Setup AngularJS:**
    - Put the required JavaScript files (including the one containing your module and components) at the right place. Load them from your index.html. Make sure that the whole thing works (validate binding between scope and view).
- **Integrate the Graph.js:**
    - This will allow you to validate that you can declared and load a dependency from your application module.
    - Provide fake data in your controller (or even better, for practice, create a small service that you inject in your controller and that provides the fake data)

Proposed approach (2)

- **Integrate socket.io in your express back-end:**
  - Figure out how to pass the server object to the socket.io module and where to put your code.
  - **Validate the setup** by with a **very basic client-side script** (don’t worry about integration with AngularJS initially). You should validate that the server receives a notification when a client loads the page (and thus establishes a WebSocket connection). You should also validate bi-directional communication.
- **Integrate socket.io in your AngularJS front-end:**
  - Use the **btford.socket-io** module and validate that the exchange of messages works (don’t worry about votes just yet).

Proposed approach (3)

- **Integrate ui-router and implement your UI**
  - Define the three states and the corresponding layouts.
  - Validate that you can switch from one state to the other with **ui-sref**
- **Connect your UI to socket.io**
  - When the “Vote” buttons are clicked, emit a message (also work on the server side to update the state of the poll and to broadcast a notification to all users).
  - When a “vote update” notification arrives on the socket, make sure that the graphs are updated.

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Submit your GitHub repo URL
in Cyberlearn until 4 PM

Activity 1: JSON-P

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **Start by reading and analyzing in details the following article:**
    - http://schock.net/articles/2013/02/05/how-jsonp-really-works-examples/
- **Prepare answers to the following questions:**
    - What is the **problem** addressed by JSON-P? Illustrate with a concrete example.
    - What needs to be done on the **client side** in order to implement JSON-P (explain what happens at the **lowest level** and how libraries can help)?
    - What needs to be done on the **server side** in order to implement JSON-P?
- **You have 15 minutes to prepare yourself. I will ask a few students to present their solutions.**
- **Other helpful resources:**
    - https://developer.github.com/v3/#json-p-callbacks
    - https://johnnywey.wordpress.com/2012/05/20/jsonp-how-does-it-work/
    - http://www.uitrick.com/javascript/jsonp-and-its-usages/

Activity 2: CORS

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Start by reading and analyzing in details the following article:**
  - http://www.eriwen.com/javascript/how-to-cors/
  - Prepare answers to the following questions:
  - What is the **problem** addressed by CORS? Illustrate with a concrete example.
  - What needs to be done on the **client side** in order to implement CORS?
  - What needs to be done on the **server side** in order to implement CORS?
  - Illustrate the process with a sequence diagram.
- **You have 15 minutes to prepare yourself. I will ask a few students to present their solutions.**
- **Other helpful resources:**
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
  - http://www.w3.org/TR/cors/#introduction
  - http://www.html5rocks.com/en/tutorials/cors/