

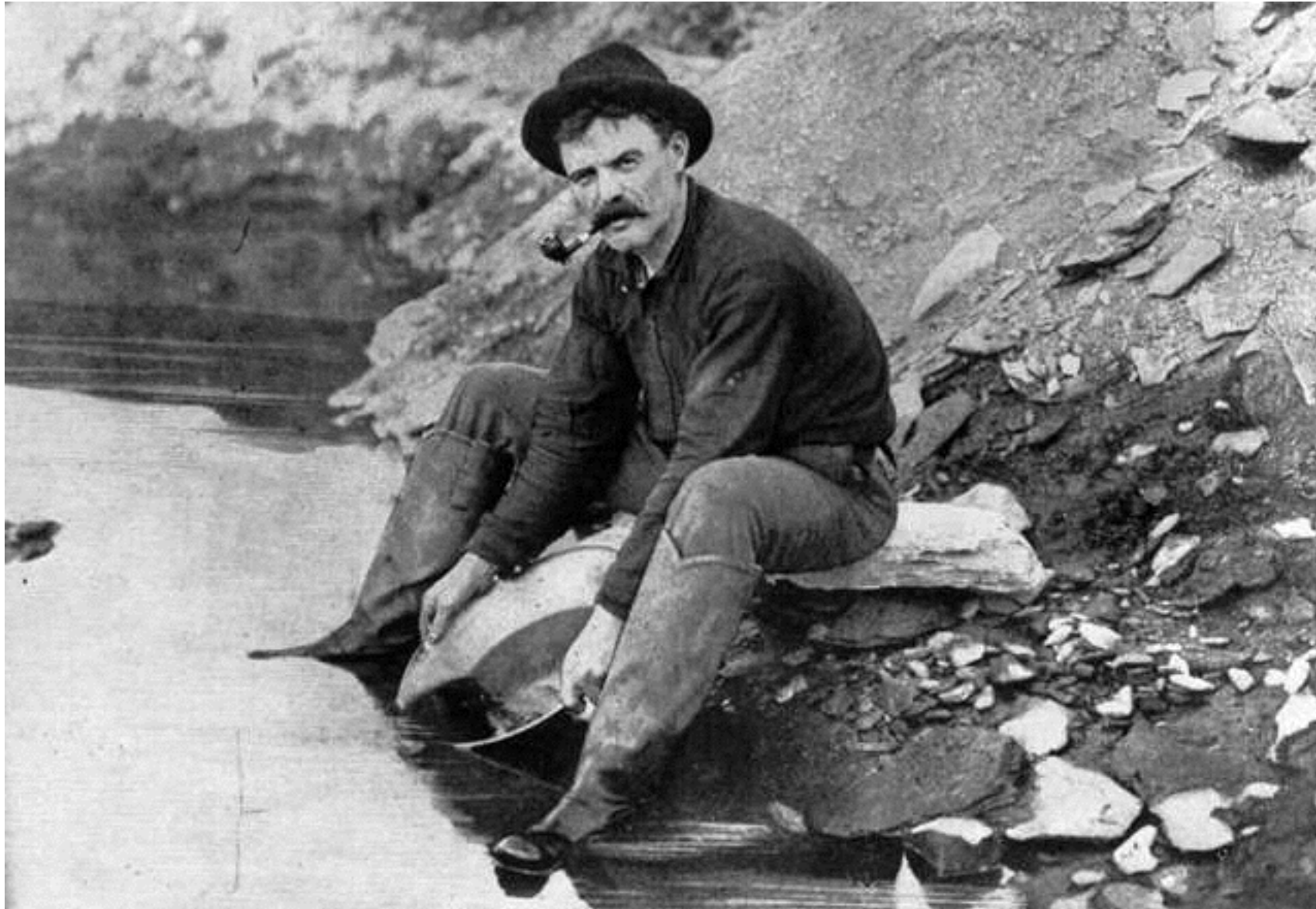
# Lecture 2: Introduction to AngularJS

---

Olivier Liechti  
TWEB

heig-vd

Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud



# GitHub Explorer - iteration 2

# Project - 2 features for the iteration

---

As a user, I can navigate between several pages (in an angular app).

As a user, I can see a graph (pie chart)

# “I can navigate between pages”

Handle **AngularJS bootstrapping** in your web app (the one you deployed on Heroku)

Create **folder structure** based on John Papa's guidelines (folder by feature)

Setup and configure **UI-Router** (define at least 2 states, define routes, templates, etc.)

Create the views and prepare the **controllers** and **services**.

Validate, update  
Heroku. Validate.

# “I can see a graph”

---

Discover and integrate an **Angular module** that provides chart capabilities (e.g. angular chart-js)

Create a **service** that provides (fake) data to be rendered in the chart. Expose the service via a controller.

Create the view that invokes the service via the **controller** and passes the data to the chart component.

Integrate this functionality with the rest of the app.

Validate, update  
Heroku. Validate.

# Today's agenda

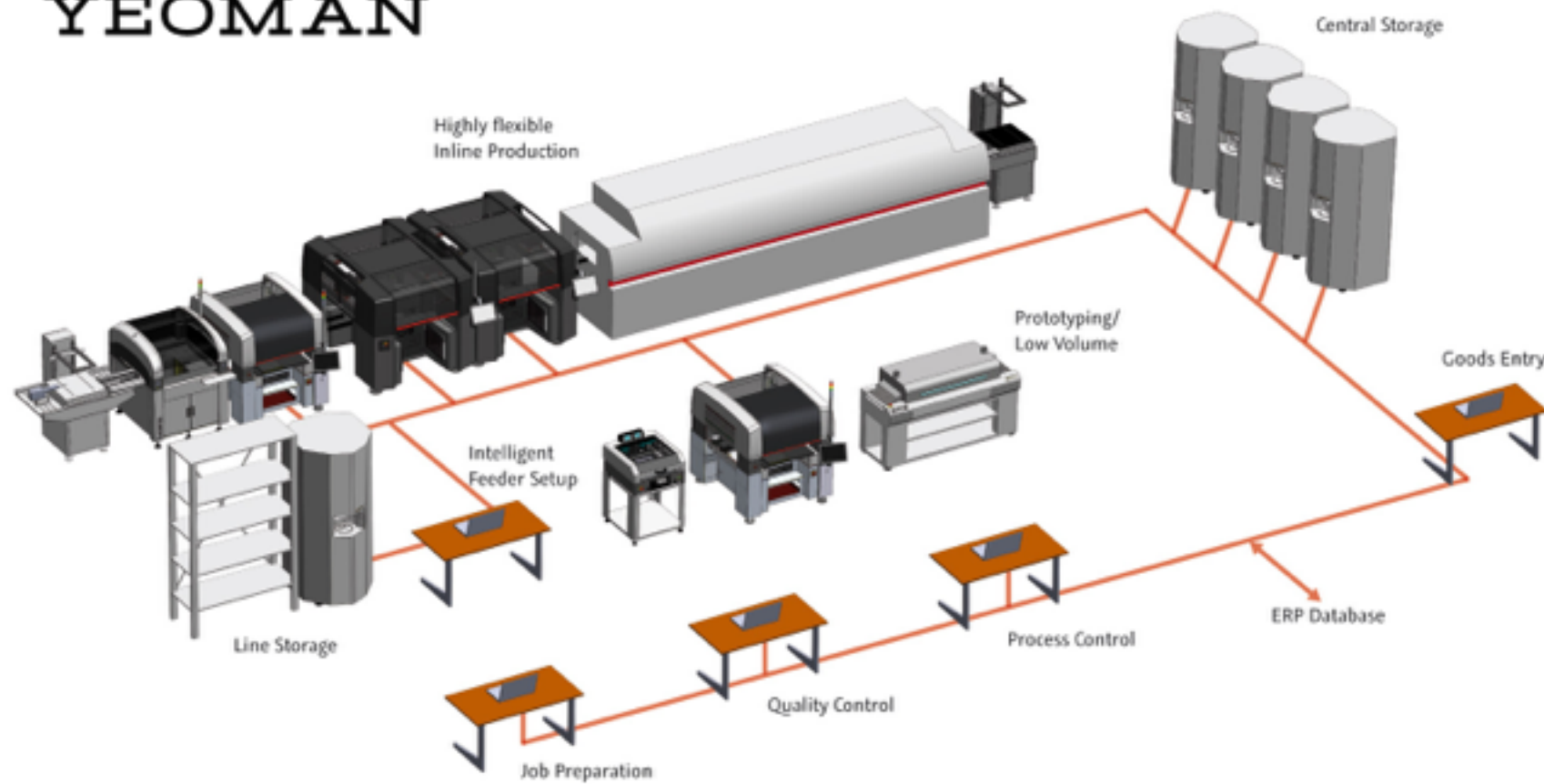
---

<b>15:45 - 16:05</b>	<b>20'</b>	<b>Intro to web tools (Yeoman)</b>
16:05 - 16:30	25'	Everybody scaffolds a project with yo
16:30 - 16:50	20'	AngularJS core concepts
16:50 - 17:29	70'	Joint exploration of the code
17:55 - 18:05	10'	Wrap-up



heig-vd

Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud



# Development tools & pipelines





# How do I **bootstrap** and **structure** my project?

- Based on the specifications, we know that we will develop components **both on the client and on the server side**. We also want to **automate the build process** for our project.
- What should we do? **Start from scratch** or use some kind of **skeleton**? What are our **options**? What are the **professional** front-end developers doing?



Paul Irish, "Delivering the goods" - Fluent 2014 Keynote

by O'Reilly • 7 months ago • 29,621 views

Fluent 2014, "Keynote With Paul Irish". About Paul Irish (Google): Paul Irish is a front-end developer who loves the web. He is on ...

HD



Fluent 2013: Paul Irish, "JavaScript Authoring Tooling"

by O'Reilly • 1 year ago • 35,810 views

<http://fluentconf.com> To view a complete archive of the Fluent 2013 tutorials and sessions, check out the All Access video ...

HD





# Meet Yeoman.

The web's scaffolding tool

yeoman.io

YEOMAN

Using Yeoman   Discovering generators   Creating a generator   Blog   Contributing

## THE WEB'S SCAFFOLDING TOOL FOR MODERN WEBAPPS

[Get started](#) and then [find a generator](#) for your webapp. Generators are available for [Angular](#), [Backbone](#), [Ember](#) and over [1000+ other projects](#). Read the [Yeoman Monthly Digest](#) for our latest picks.

One-line install using [npm](#):

```
npm install -g yo
```



# What is **Yeoman**?

- Yeoman is a **combination of tools**, which allows to you to setup a **complete, automated, efficient and reliable development workflow**.
- **Yo** is a tool for generating project skeletons (**scaffolding**). You can create and share your skeletons. **Yo generators are npm modules** and you can find one for most popular web frameworks.
- **Bower** is a tool for managing “**web dependencies**”. Not only javascript modules, but also CSS files, images, etc.
- **Grunt** is a **task runner**. It is the tool that drives your automated process, by executing a series of tasks. There are lots of grunt plugins provided by the community for all aspects of your project.



YO



GRUNT



BOWER



Ok... generators look cool. But how should I  
**pick the “right” one?**

The screenshot shows the Yeoman Generators website. The header includes the Yeoman logo and navigation links: Using Yeoman, Discovering generators, Creating a generator, Blog, and Contributing. The main heading is 'GENERATORS' with a rocket illustration. Below this, a note states: 'Your generator must have the "yeoman-generator" keyword and a repo description to be listed. Official generators are marked with 🍌'. A search bar contains the text 'express'. Below the search bar is a table of generators.

Name	Description	Author	Stars
<a href="#">angular-fullstack</a>	AngularJS with an Express server	<a href="#">Tyler Henkel</a>	1985
<a href="#">express</a>	An express generator for Yeoman, based on the express command line tool	<a href="#">petecoop</a>	180
<a href="#">mean-seed</a>	MEAN Seed / MEAN Stack (AngularJS, node.js app) - MongoDB, Express, Angular, Node, Grunt, Bower, Yo. 'Core' and 'Module' subgenerators for customization outside of that	<a href="#">Luke Madera</a>	102




## How do you **pick a generator** for your project?

- You probably **have an idea of the framework(s) you want to use** on the server and or client side (express, angular, backbone, etc.). You will use this as a first filter.
- Some of the generators are **supported by the Yeoman Team**. That is probably a good indication about the quality and support over time (evolution).
- Developers who use generators can “**star**” those they like. **Sorting by popularity** is also an interesting indication. If the community is big, you can expect issues to be reported and fixed, to see new features, etc.
- After you have identified **promising candidates**, you need to get a **first impression**. Generate and build a project with each candidate. Look at their Github repository. Do you like what you see? Do you like the documentation?
- Often, you will need to choose between “**lightweight**” and very “**rich**” generators. Lightweight generators are easier to learn and give you more control (but more work). Rich generators do a lot of things out-of-the-box but can be intimidating at first (learning curve to understand the skeleton).



Meet the **angular-fullstack** generator.

↕ Name	↕ Description	↕ Author	↕ Stars
 <a href="#">angular</a>	AngularJS	<a href="#">The Yeoman Team</a>	2617
<a href="#">angular-fullstack</a>	AngularJS with an Express server	<a href="#">Tyler Henkel</a>	1985

## AngularJS Full-Stack generator build passing

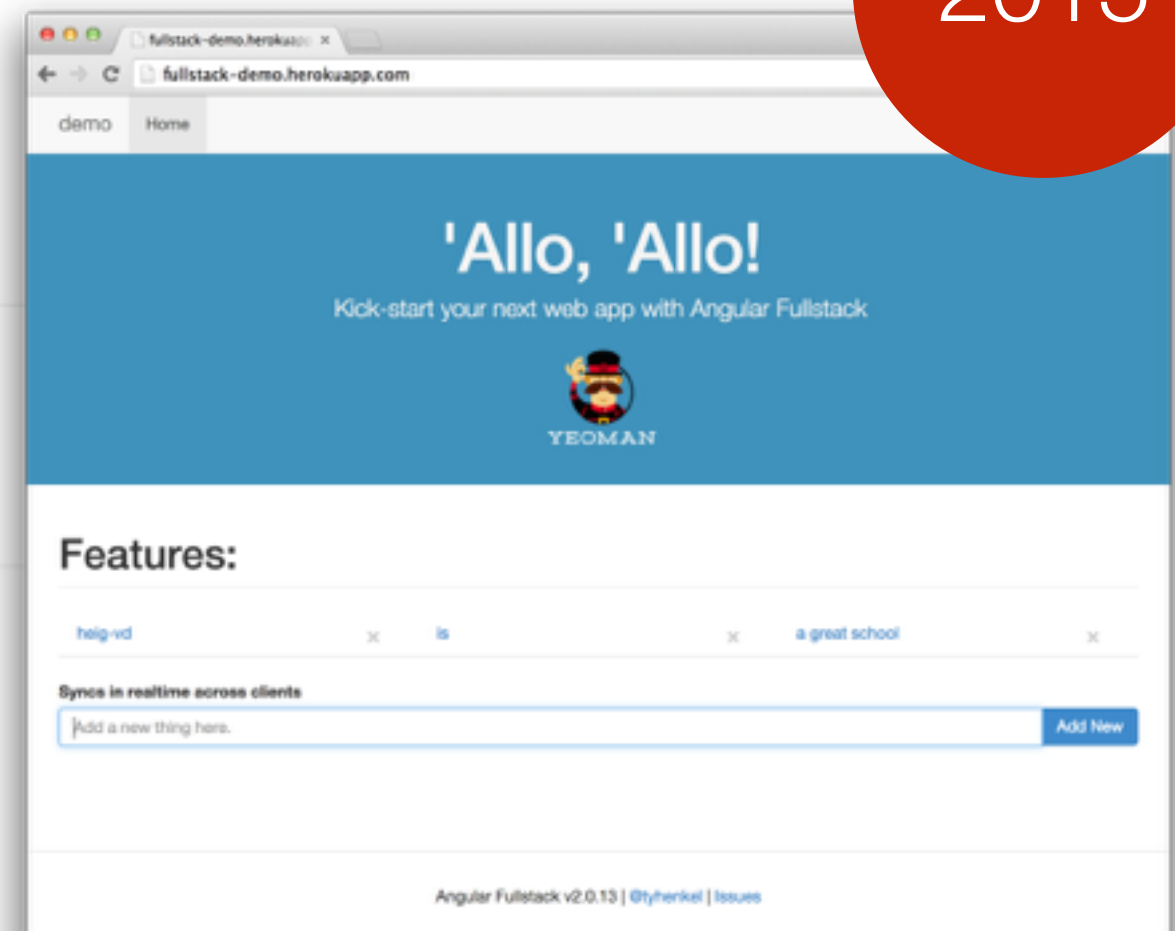
[GITTER](#) [JOIN CHAT](#)

Yeoman generator for creating MEAN stack applications, using MongoDB, Express, AngularJS, and Node - lets you quickly set up a project following best practices.

### Example project

Generated with defaults: <http://fullstack-demo.herokuapp.com/>.

Source code: <https://github.com/DaftMonk/fullstack-demo>



2015





Meet the **angular-fullstack** generator.

angular-fullstack

↕ Generator

↕ Last Updated

↕ Stars

↕ Installs

**angular-fullstack** by Andrew Koroluk

6 days ago

5403

14665

Creating MEAN stackapps, using MongoDB, Express, AngularJS, and Node

Sep 2, 2012 – Sep 26, 2016

Contributions to master, excluding merge commits

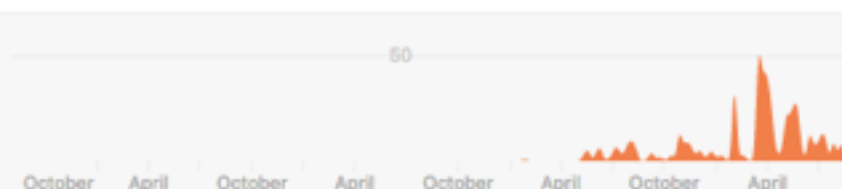
Contributions: Commits ▾



Awk34

#1

585 commits / 36,416 ++ / 25,150 --



DaftMonk

#2

437 commits / 297,707 ++ / 294,309 --







## Why and when is this generator interesting?

- It uses **Express.js** on the server side, **AngularJS** and **Socket.IO** on the client side and the glue between the frameworks.
- It has **sub-generators** to iteratively add new server-side and client-side components (new entities, new REST endpoints, new UI pages, etc.)
- The **structure of the AngularJS components** (folders and files where the UI elements are coded) follows best practices.
- It comes with support for **persistence** (with MongoDB), for **push notifications** (with Socket.IO) and for **authentication** (with Passport.js). The framework implements an interesting pattern for notifying CRUD operations to all connected users in realtime.
- Out-of-the-box, it provides a **complete and functional application** (which you can test on-line with their demo app).
- It supports **deployment on heroku** (and other cloud providers).
- The **drawback** is that the generated code and the build file is much more complex, compared to other generators. There is a much steeper learning curve and if you are starting with JavaScript, npm and Gulp, you can get lost and intimidated.



# Meet the **angm** generator.

angm

↕ **Generator**

↕ **Last Updated**

↕ **Stars**

↕ **Installs**

**angm** by [Fernando Monteiro](#)

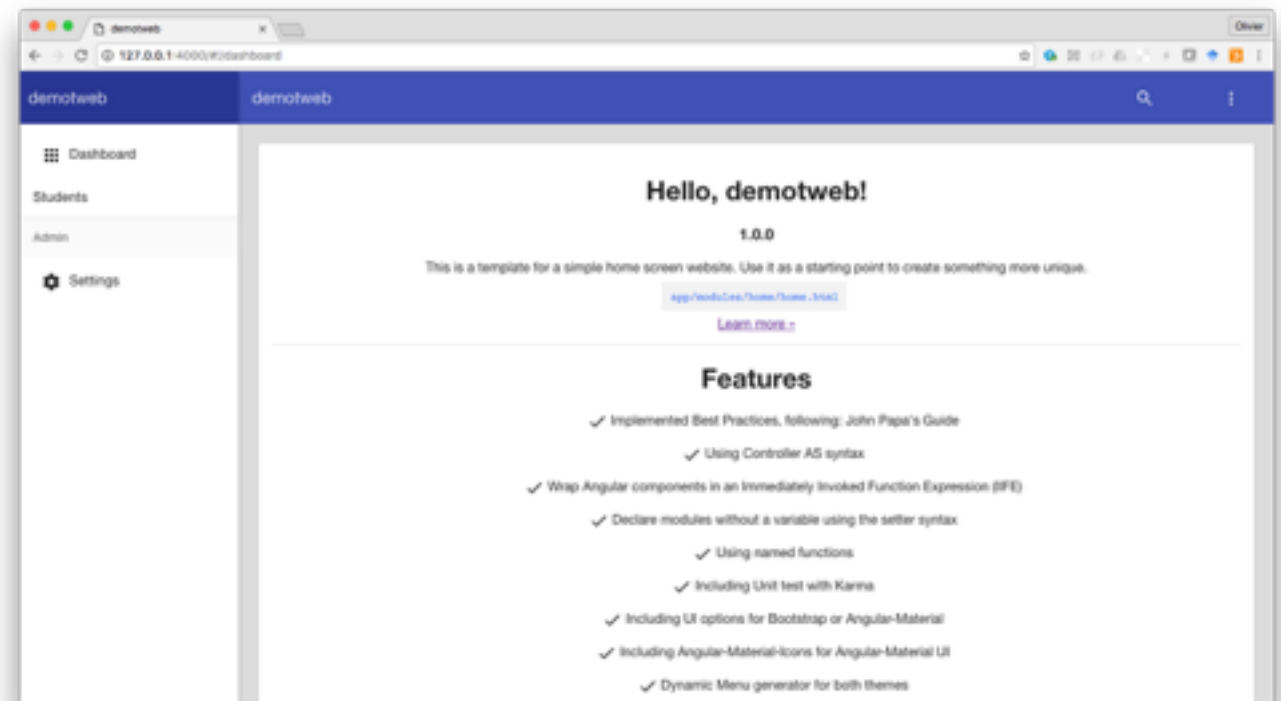
3 months ago

43

649

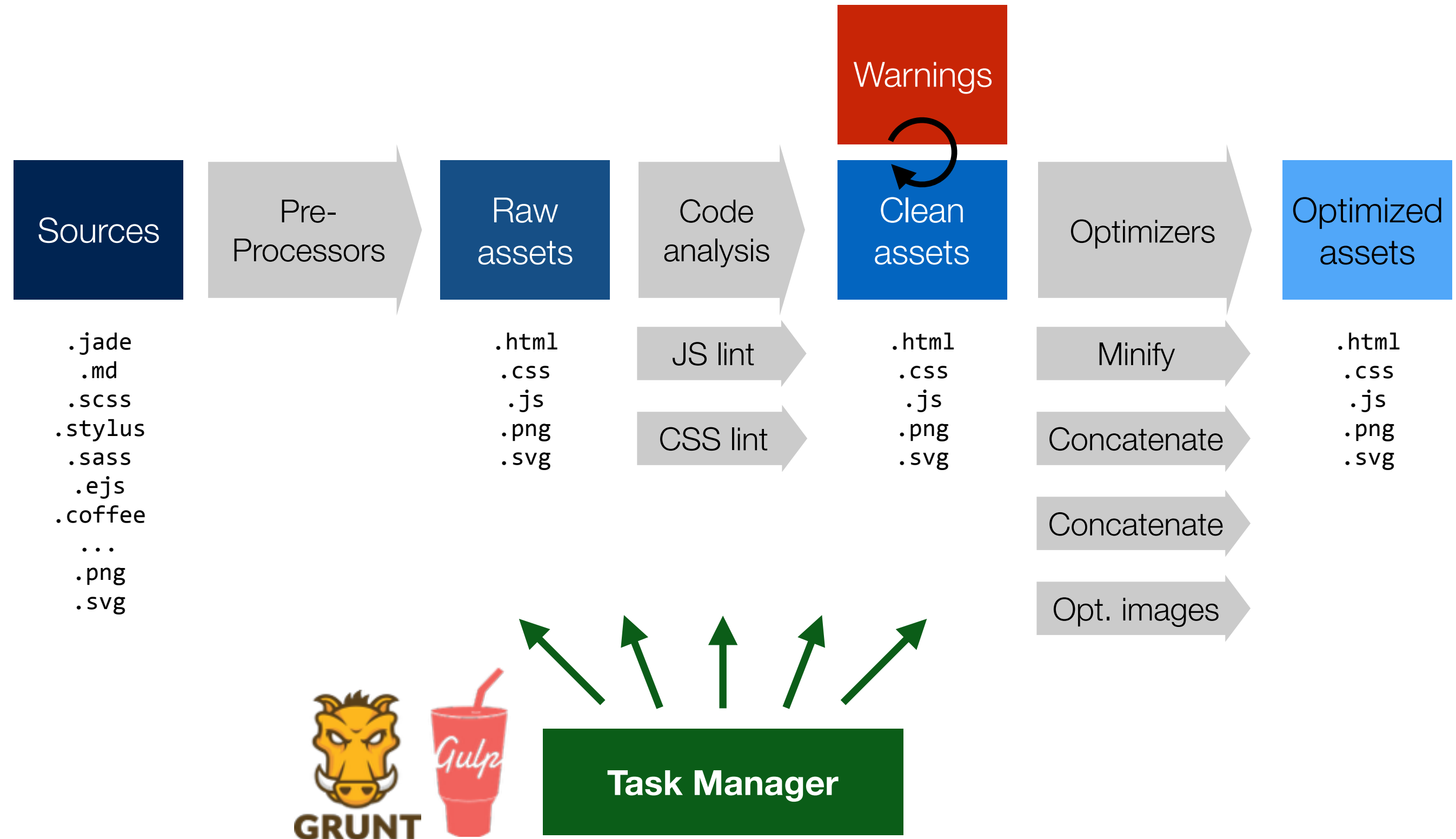
AngularJS help you getting started with a new project based on AngularJS and Angular Material to build large scaleapps

- Much simpler
- Only client-side
- Uses angular material
- **Used in the TWEB webcasts**



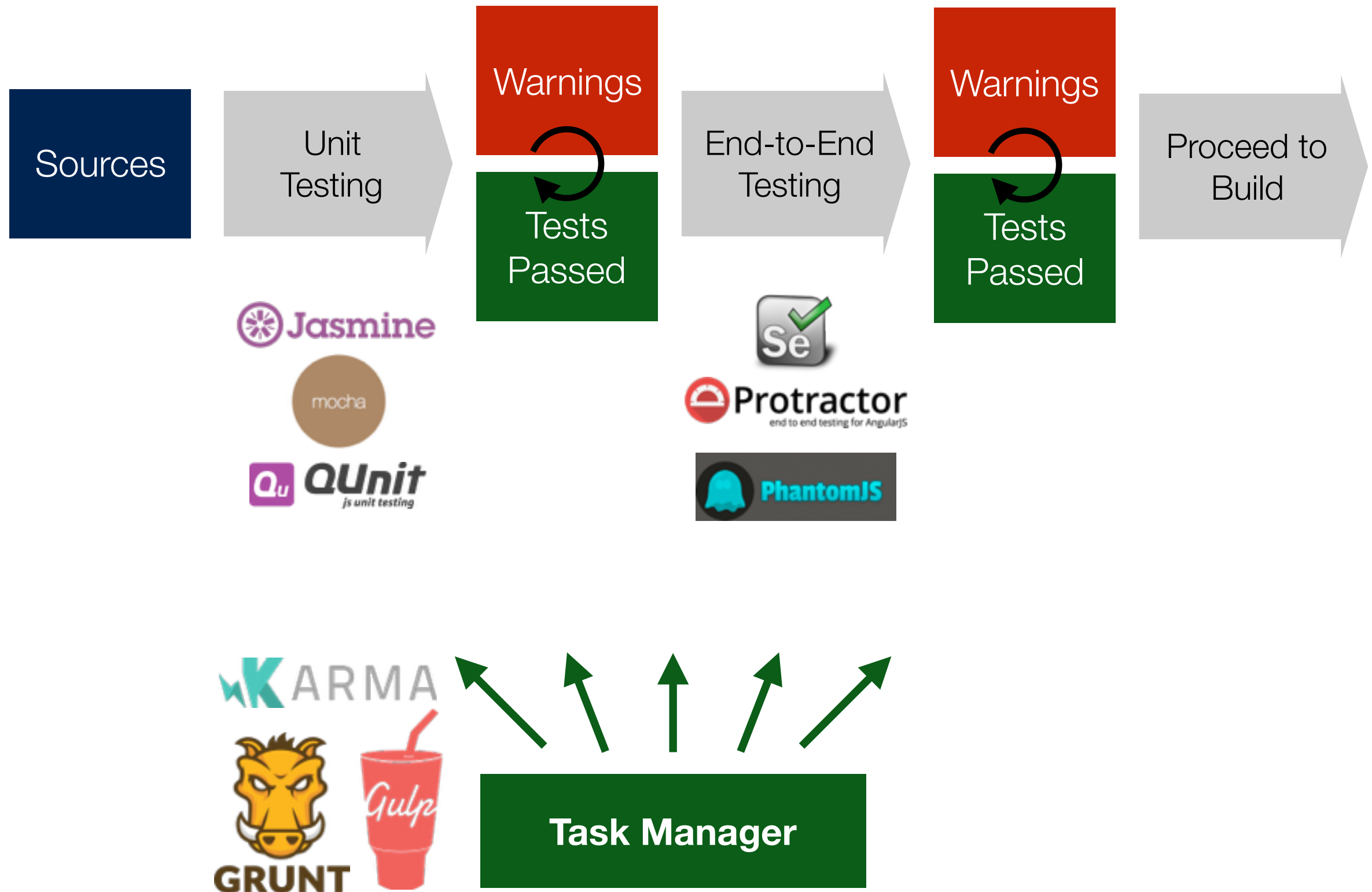


# Build Pipeline





# Test Pipeline





# Grunt



## GRUNT

The JavaScript Task Runner

→ Getting Started   Plugins   Documentation   API

### Why use a task runner?

In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it through a [Gruntfile](#), a task runner can do most of that mundane work for you—and your team—with basically zero effort.

### Why use Grunt?

The Grunt ecosystem is huge and it's growing every day. With literally hundreds of plugins to choose from, you can use Grunt to automate just about anything with a minimum of effort. If someone hasn't already built what you need, authoring and publishing your own Grunt plugin to npm is a breeze. See how to [get started](#).

### Available Grunt plugins

Many of the tasks you need are already available as Grunt Plugins, and new plugins are published every day. While the [plugin listing](#) is more complete, here's a few you may have heard of.



```
grunt
$ grunt
Running "jshint:gruntfile" (jshint) task
>> 1 file lint free.

Running "jshint:src" (jshint) task
>> 1 file lint free.

Running "jshint:test" (jshint) task
>> 1 file lint free.

Running "qunit:files" (qunit) task
Testing test/tiny-pubsub.html....OK
>> 4 assertions passed (23ms)

Running "clean:files" (clean) task
Cleaning "dist"...OK

Running "concat:dist" (concat) task
File "dist/ba-tiny-pubsub.js" created.

Running "uglify:dist" (uglify) task
File "dist/ba-tiny-pubsub.min.js" created.
Uncompressed size: 389 bytes.
Compressed size: 119 bytes gzipped (185 bytes minified).

Done, without errors.

$ _
```





# Grunt



## Plugins

This plugin listing is automatically generated from the npm module database. Officially maintained "contrib" plugins are marked with a star ★ icon.

In order for a Grunt plugin to be listed here, it must be published on [npm](#) with the [gruntplugin](#) keyword. Additionally, we recommend that you use the [gruntplugin grunt-init template](#) when creating a Grunt plugin.

Showing 1 to 24 of 24 entries (filtered from 3,638 total entries)

Search:



**Discover Dev Tools**,  
a free interactive course  
to help you master  
Chrome Dev Tools.

Ads by [Bocoup](#).

Plugin	Updated	Grunt Version	Downloads last 30 days
★ <b>contrib-jshint</b> by Grunt Team Validate files with JSHint.	6 months ago	~0.4.0	<b>538387</b>
<b>htmlhint</b> by Yanis Wang Validate html files with htmlhint.	3 months ago	~0.4.1	<b>3045</b>
★ <b>contrib-jshint-jsx</b> by Grunt Team	3 months ago	0.4.0	<b>2466</b>





# Telling grunt what to do: **Gruntfile.js**

- Let's define **two workflows**: a “test” workflow and a “default” workflow. I will be able to type “grunt test” and a “grunt” on the command line to run them.

```
// this would be run by typing "grunt test" on the command line
grunt.registerTask('test', ['jshint', 'qunit']);

// the default task can be run just by typing "grunt" on the command line
grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);
```

- The workflows use a few standard grunt plugins. Let's load them in the Gruntfile.js.

```
grunt.loadNpmTasks('grunt-contrib-uglify');
grunt.loadNpmTasks('grunt-contrib-jshint');
grunt.loadNpmTasks('grunt-contrib-qunit');
grunt.loadNpmTasks('grunt-contrib-watch');
grunt.loadNpmTasks('grunt-contrib-concat');
```

- Each grunt plugin can be configured. Here, we specify what files to lint and how.

```
jshint: {
  // define the files to lint
  files: ['gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
  // configure JSHint (documented at http://www.jshint.com/docs/)
  options: {
    // more options here if you want to override JSHint defaults
    globals: {
      jQuery: true,
      console: true,
      module: true
    }
  }
}
```



# A simple, complete Gruntfile.js

<http://gruntjs.com/sample-gruntfile>

```
module.exports = function(grunt) {

  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    concat: {
      options: {
        separator: ';'
      },
      dist: {
        src: ['src/**/*.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("dd-mm-yyyy") %> */\n'
      },
      dist: {
        files: {
          'dist/<%= pkg.name %>.min.js': ['<%= concat.dist.dest %>']
        }
      }
    },
    qunit: {
      files: ['test/**/*.html']
    },
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        // options here to override JSHint defaults
        globals: {
          jQuery: true,
          console: true,
          module: true,
          document: true
        }
      }
    },
    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint', 'qunit']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-qunit');
  grunt.loadNpmTasks('grunt-contrib-watch');
  grunt.loadNpmTasks('grunt-contrib-concat');

  grunt.registerTask('test', ['jshint', 'qunit']);
  grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

};
```

Configure the grunt plugins.

Watch is a special plugin. It allows to execute tasks when files are changed in the file system.

Load the grunt plugins.

Define the two workflows.



clean:dist	471ms	2%
concurrent:dist	1.6s	8%
wiredep:target	850ms	4%
autoprefixer:dist	260ms	1%
concat:generated	399ms	2%
ngAnnotate:dist	925ms	5%
copy:dist	4.7s	
cdnify:dist	5.4s	
cssmin:generated	596ms	3%
uglify:generated	4s	
<b>Total</b>	<b>19.8s</b>	

# Today's agenda

---

15:45 - 16:05	20'	Intro to web tools (Yeoman)
<b>16:05 - 16:30</b>	<b>25'</b>	<b>Everybody scaffolds a project with yo</b>
16:30 - 16:50	20'	AngularJS core concepts
16:50 - 17:29	70'	Joint exploration of the code
17:55 - 18:05	10'	Wrap-up

# Instructions

---

- Visit <http://yeoman.io>
- Explore the available generators
- You can try to use a couple of them, in particular:
  - **angm (the one you will see in the TWEB webcasts)**
  - angular-fullstack (a rich/complex one, which gives you the choice between Typescript and ES6). You will need a running MongoDB server.
  - express (for server-side stuff)
- Follow the installation instructions and study:
  - the code structure
  - the build pipeline structure



# Introduction



# What is AngularJS?

---

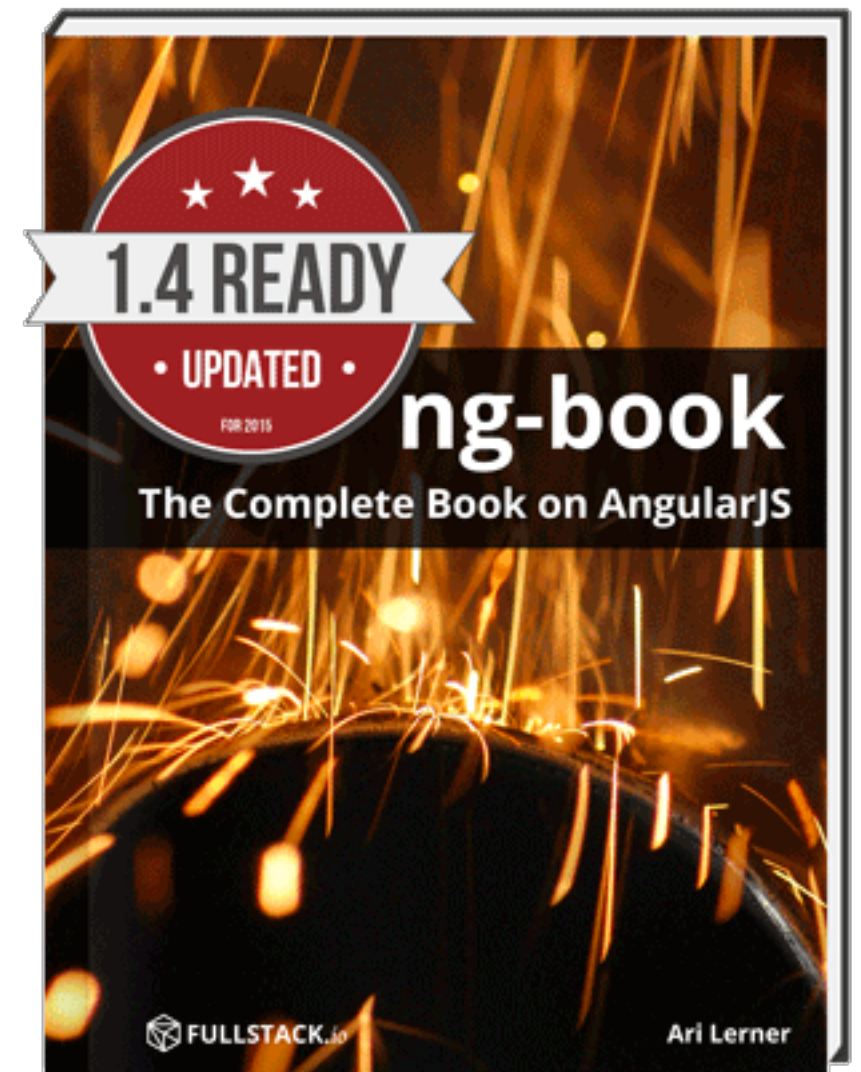
- **Client-side** JavaScript framework
- Designed to create **Single-Page Applications** (SPAs), as opposed to “server-side MVC applications”.
- Initially released in **2009**. Has become one of the most popular frameworks (see job offers!).
- **Large community** (many third-party modules, learning resources, etc.). Open source project with major contributions from **Google**.
- Current version (v1): 1.5.8
- Current version (v2): 2.0.0 final released on September 15th 2016

# What is the best way to learn AngularJS?

heig-vd

Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud

- There are **many books**. I have found this one to be particularly helpful:
  - <https://www.ng-book.com/>
- There are many sites with **tutorials and webcasts**. Here is a good example:
  - <https://egghead.io/technologies/angularjs>
- One of the most efficient ways is to study and play with **existing code**:
  - Browse through GitHub repos.
  - Use a yeoman generator
- There are often different ways to do one thing. It is important to adopt coding conventions. See:
  - <https://github.com/johnpapa/angular-styleguide>



<https://www.ng-book.com/>

# Core concepts

---

- **Bootstrapping**

- What do I need to do so that AngularJS starts doing its magic with my app?

- **Modules**

- How do I split my app into multiple libraries and how do I use libraries provided by other developers?

- **Directives**

- How can I write views/templates for AngularJS? What are those extra HTML elements (elements and attributes)?

- **Controllers**

- How do I define the data that is then rendered in the views/templates?

- **Services**

- Where do I put the business logic and how do I

- **Scopes**

- *For the time being, we will ignore this concept that used to be a cornerstone of the AngularJS architecture. In “modern” patterns, it is hidden by an alternative syntax (aka controller as).*

# How do I bootstrap AngularJS?

- 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"

# What is a Module?

- 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');
```

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**.



# What is a Directive?

- 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 (but we will not do that immediately)

## Directive components in ng

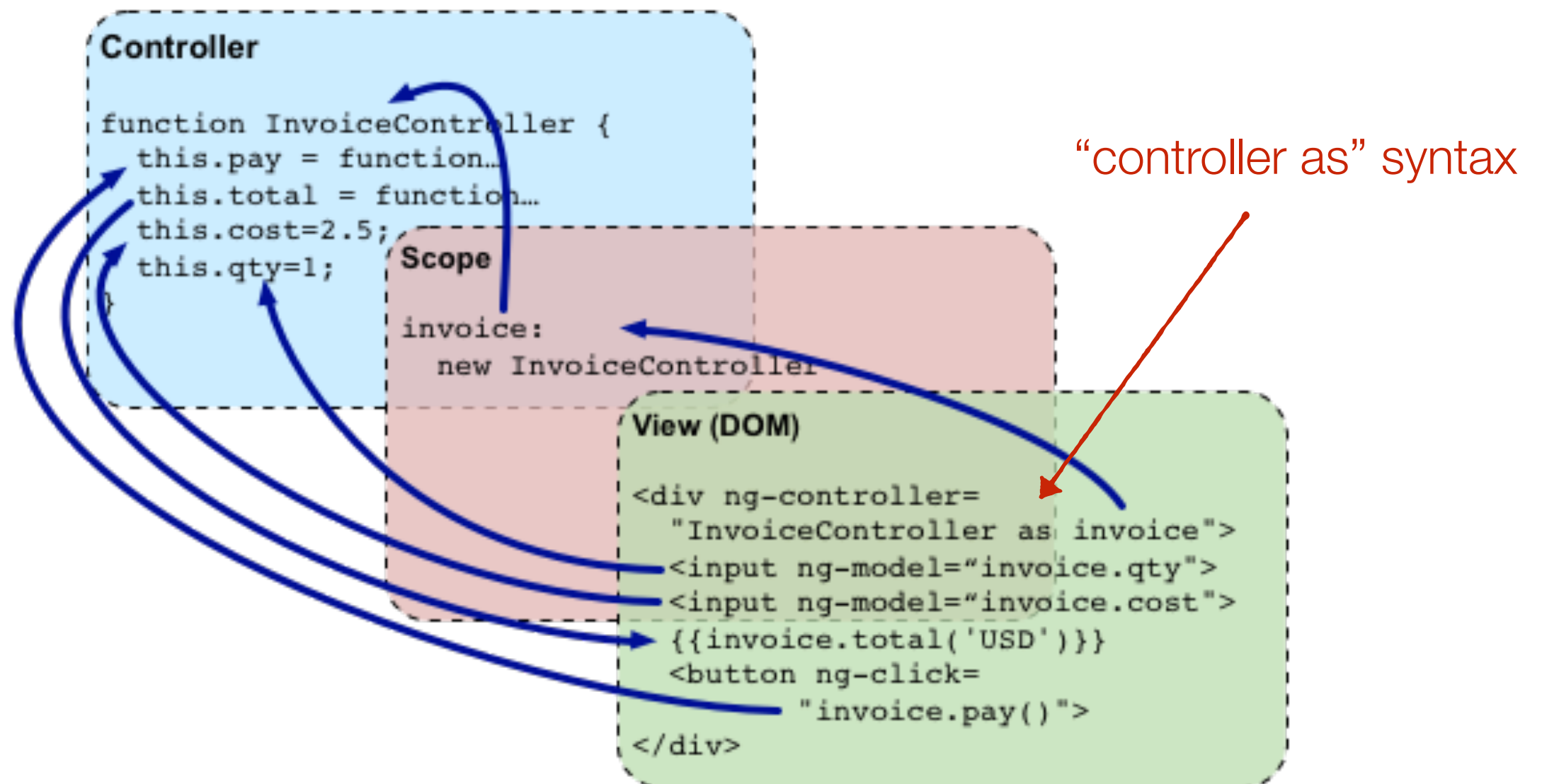
Name	Description
<code>ngJq</code>	Use this directive to force the angular.element library. This should be used to force either jqLite by leaving ng-jq blank or setting the name of the jquery variable under window (eg. jQuery).
<code>ngApp</code>	Use this directive to <b>auto-bootstrap</b> an AngularJS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

# Which directives do we use often?

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

# What is a Controller?

- An AngularJS controller is used to **provide data** and **behavior** (functions) to them views/templates.



```
angular.module('invoice1', [])  
.controller('InvoiceController', function InvoiceController() {  
  this.qty = 1;  
  this.cost = 2;  
  this.inCurr = 'EUR';  
  this.currencies = ['USD', 'EUR', 'CNY'];  
  this.usdToForeignRates = {  
    USD: 1,  
    EUR: 0.74,  
    CNY: 6.09  
  };  
});
```

```
  this.total = function total(outCurr) {  
    return this.convertCurrency(this.qty * this.cost, this.inCurr, outCurr);  
  };  
  this.convertCurrency = function convertCurrency(amount, inCurr, outCurr) {  
    return amount * this.usdToForeignRates[outCurr] / this.usdToForeignRates[inCurr];  
  };  
  this.pay = function pay() {  
    window.alert('Thanks!');  
  };  
});
```

```
div ng-app="invoice1" ng-controller="InvoiceController as invoice">  
  <b>Invoice:</b>  
  <div>  
    Quantity: <input type="number" min="0" ng-model="invoice.qty" required >  
  </div>  
  <div>  
    Costs: <input type="number" min="0" ng-model="invoice.cost" required >  
    <select ng-model="invoice.inCurr">  
      <option ng-repeat="c in invoice.currencies">{{c}}</option>  
    </select>  
  </div>  
  <div>  
    <b>Total:</b>  
    <span ng-repeat="c in invoice.currencies">  
      {{invoice.total(c) | currency:c}}  
    </span>  
    <button class="btn" ng-click="invoice.pay()">Pay</button>  
  </div>  
</div>
```

# 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**.

## service

\$anchorScroll

\$animate

\$animateCss

\$cacheFactory

\$compile

\$controller

\$document

\$exceptionHandler

\$filter

\$http

\$httpBackend

\$httpParamSerializer

\$httpParamSerializerJQLike

\$interpolate

\$interval

\$locale

\$location

\$log

\$parse

\$q

\$rootScope

\$rootScope

\$sce

\$sceDelegate

\$templateCache

\$templateRequest

\$timeout

\$window

\$xhrFactory



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>



## Welcome to GamY

232 accounts created  
433 applications managed

3327 users created by applications during the last 30 days

## Registration

Email

First name

Last name

Password

Confirm password

## Register new app

Name

Description

API Key

# users

State

Login failed!

## Welcome to GamY

232 accounts created  
433 applications managed

Logged in as olivier.ilechti@wasabi-tech.com

## Your apps

Name	Description	Api Key	# Users		
demo 1	just a test...	8ajdj\$239jwks9kk	no user	<input type="button" value="edit"/>	<input type="button" value="enabled"/>
a test app	This application was...	wkif\$209io85fs	239'229	<input type="button" value="edit"/>	<input type="button" value="disabled"/>
my photo app	A cool app that...	lofn\$2nd87ns6	1110	<input type="button" value="edit"/>	<input type="button" value="enabled"/>

## App details

Name

Description

API Key

# users

State

Logged in as olivier.ilechti@wasabi-tech.com

## Edit your account details

Email

First name

Last name

Password

Confirm password

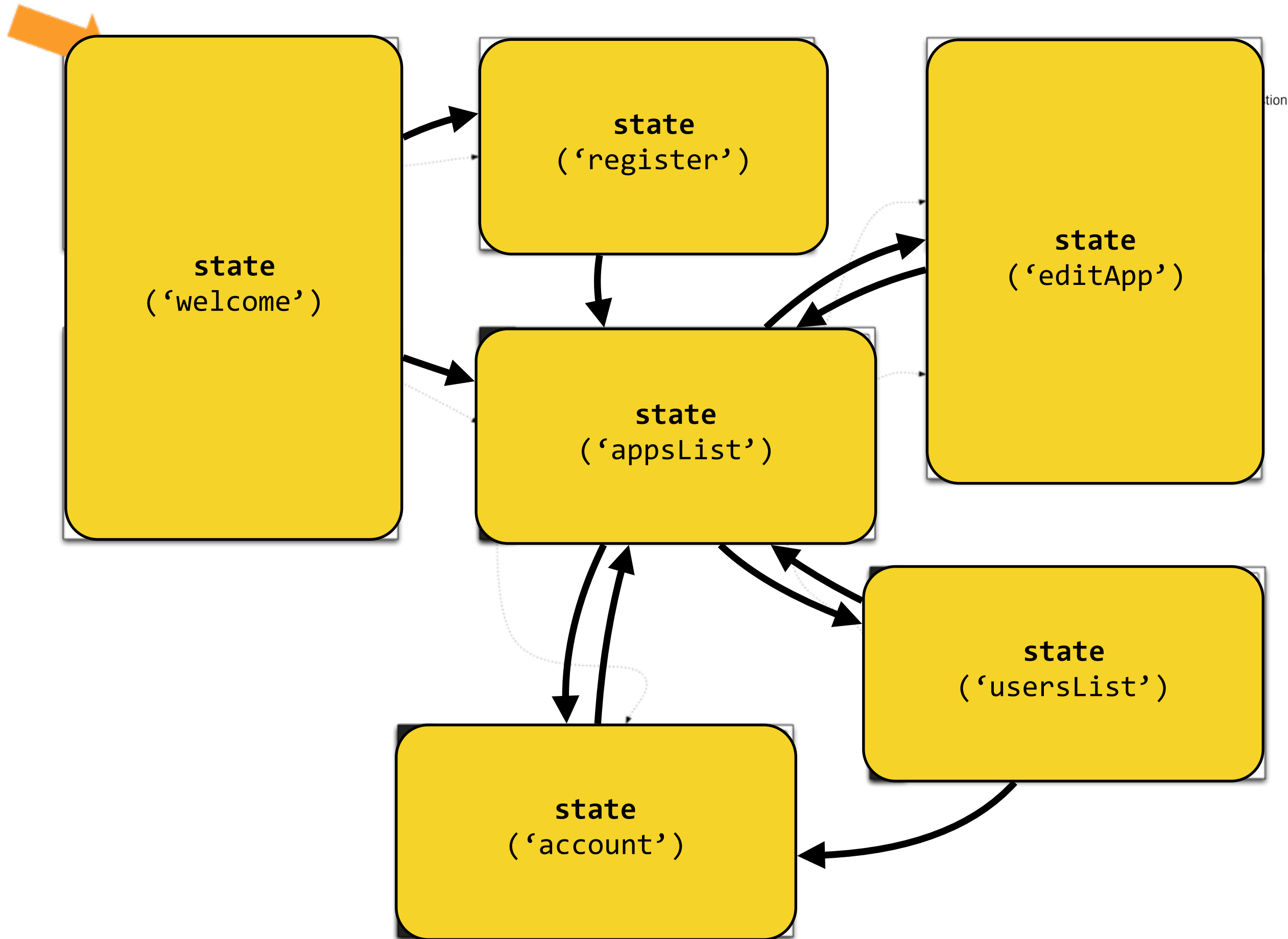
Logged in as olivier.ilechti@wasabi-tech.com

## List of users for "my photo app"

User id	Creation date
UK1928492m	20150101
KKK9494JN	20150101
KJS9KFLKAS	20150102
APK1998JS9	20150102
KENMJJK22	20150103

Page 12/229

stion



# 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).