Lecture 4: Starting the project

Olivier Liechti
TWEB

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

Agenda

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

13h00 - 13h45	45’	**Looking back at the GitHub API lab** Scaffolding, pre-processors, express.js, code review
13h45 - 14h15	30’	**Project: Environment setup (local and heroku)** Setup a database
14h15 - 15h00	45’	**Lecture** Asynchronous programming techniques & modules
15h00 - 15h30		Break
15h30 - 16h15	45’	**Project: Dissecting the project skeleton** Implement and deploy a Github front-end on Heroku
16h15 - 16h45	30’	**Project: Towards a Single Page Application** Baby steps with Angular.js
16h45 - 17h15	30’	**Project: Controlling PDF in a web app** Introduction to PDF.js
17h15 - 18h00	45’	**Project** Group work, Q&A and troubleshooting

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

1. Setup a **database** (~30')

2. Create a project **skeleton** (~45')

3. Baby steps with **Angular.js** (~30')

4. Display and control **PDF** (~30')

heig-vd

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

The Github API lab…

Framework or no framework?

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

- It is possible to write an HTTP server directly “on top” of Node.js, but the **Express.js framework** makes the job much easier.
- Of course, there is a **tradeoff**, because to be **productive**, we have to become familiar both with the core Node API **and** with the framework.
- Being able to **discover** a new framework (an more generally a new codebase) and to go through a **learning curve** is a key skill for software engineers.
- **Digging into** frameworks and analyzing their architecture and implementation is a great way to **boost your design and coding skills**.

Scaffolding (project skeleton)

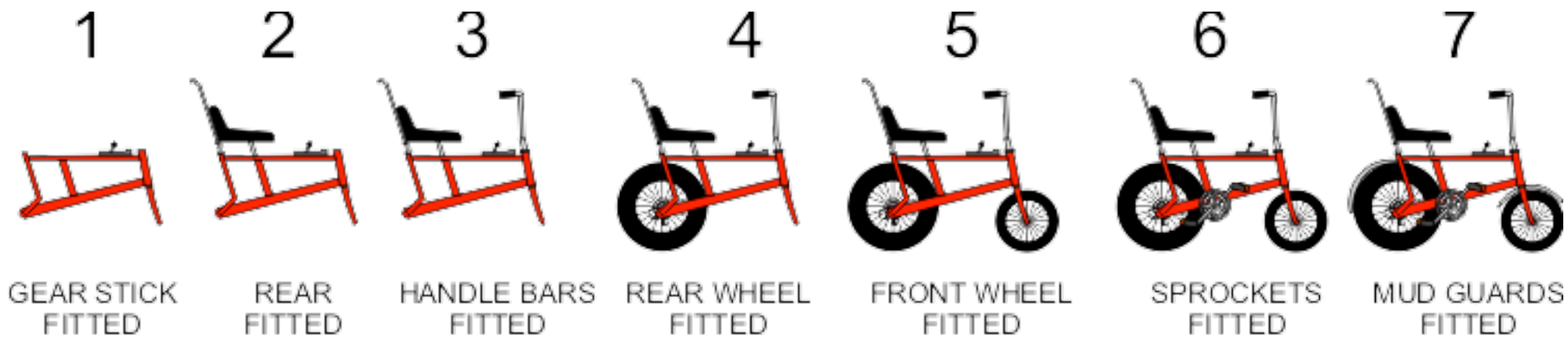
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- When using **Express.js**, we had the choice between **starting from scratch** and using the `express` command line tool to generate an **application skeleton**.
- This is a very common technique, supported by **a diversity frameworks and tools**.
- Today, we will see that the skeleton is **not limited to the source** code of the application, but can also include tools that **automate the full build, test and deployment process**.
- Like a framework, the automation infrastructure will make you **productive**... but is one more thing to **learn**.

Template engines

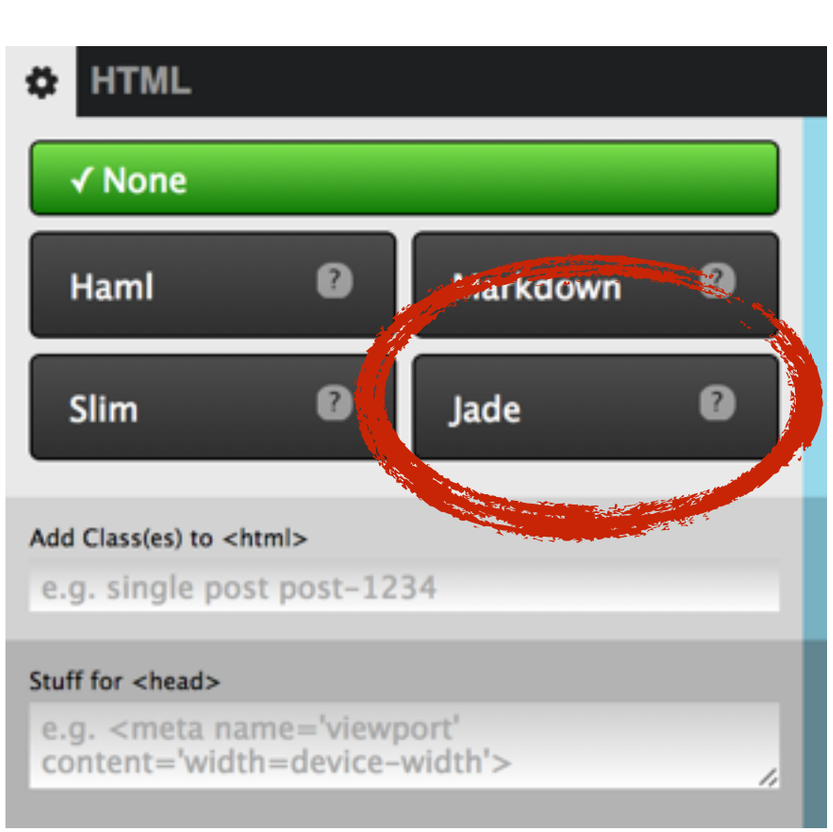
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

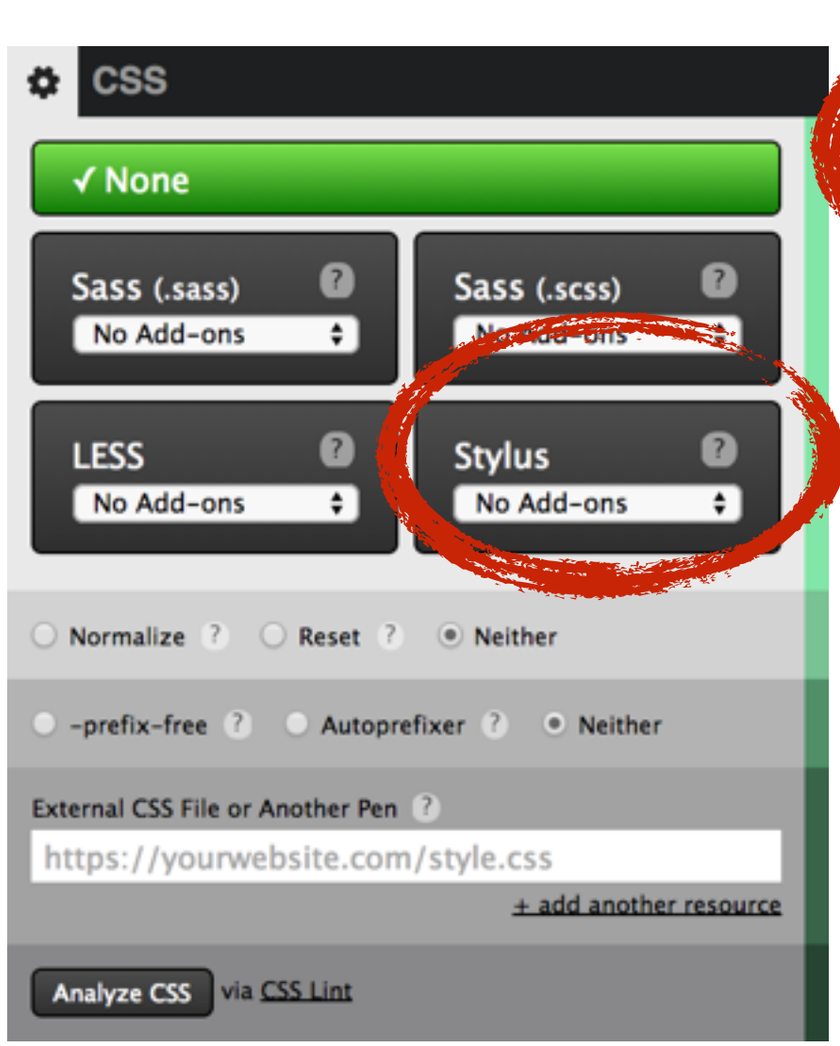
- Express.js gives us the choice to write web pages in **HTML** or to use a **server-side template engine**.
- When you use the express command line tool, **Jade** is proposed by default. However, express supports other tools (ejs, haml, etc.).
- Express also supports **CSS pre-processors**.
- This is closely related to the notion of an **asset pipeline**, where artifacts (presentation markup, styles, scripts, images, etc.) are **transformed iteratively**.

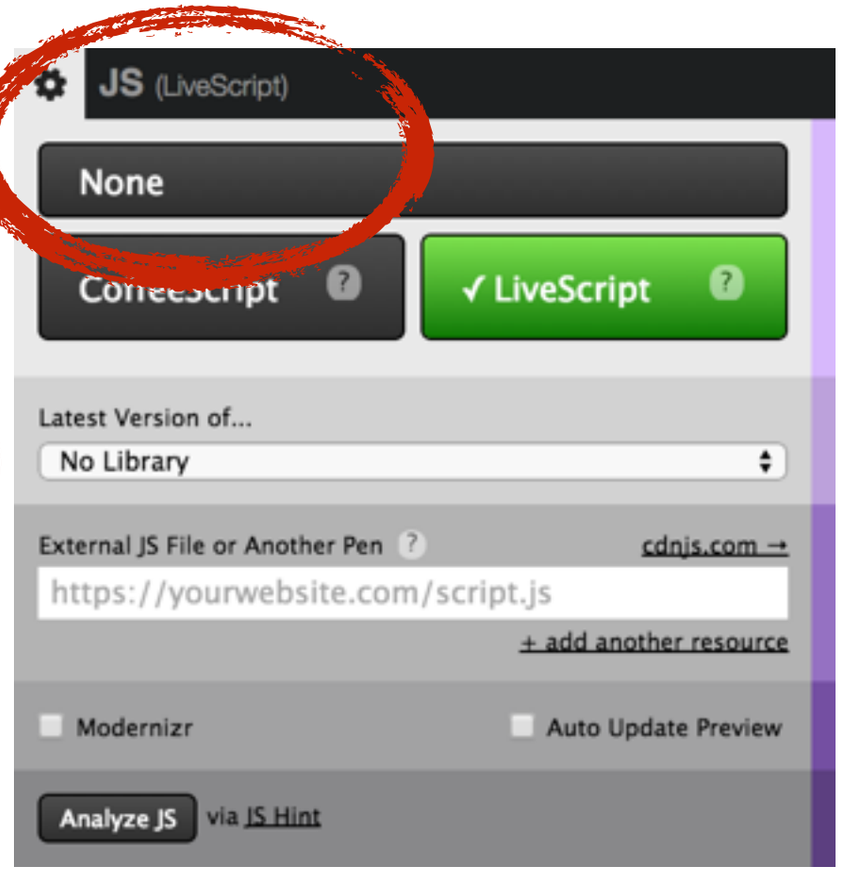


Template engines

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







The Jade Template Engine

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

```
$ express -h

  Usage: express [options] [dir]

  Options:

    -h, --help          output usage information
    -V, --version       output the version number
    -e, --ejs           add ejs engine support (defaults to jade)
    -H, --hogan         add hogan.js engine support
    -c, --css <engine>  add stylesheet <engine> support (less|stylus|compass) (defaults to plain css)
    -f, --force         force on non-empty directory
```

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
         bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
         bar(1 + 5)
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
      </p>
    </div>
  </body>
</html>
```

Template engines

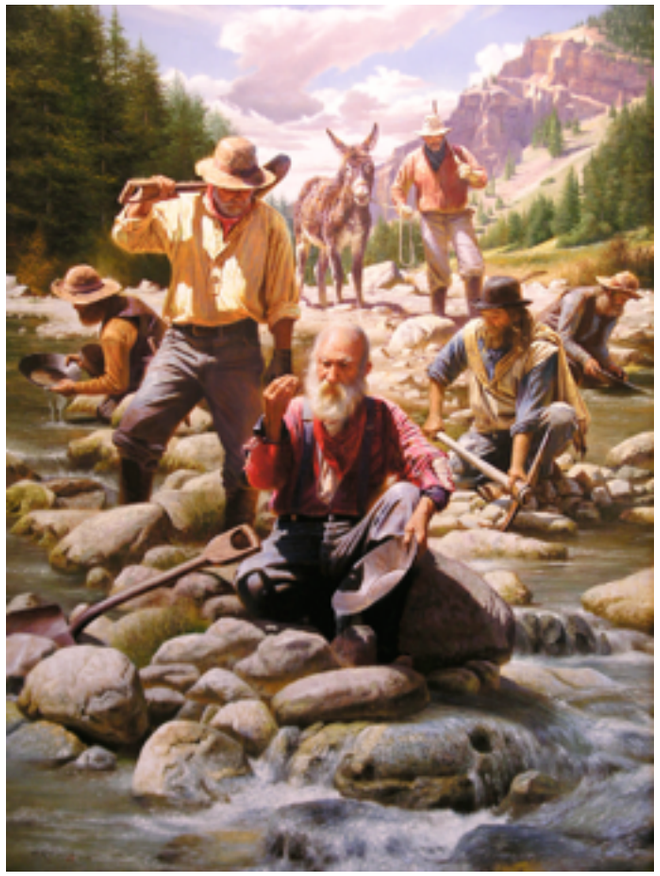
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

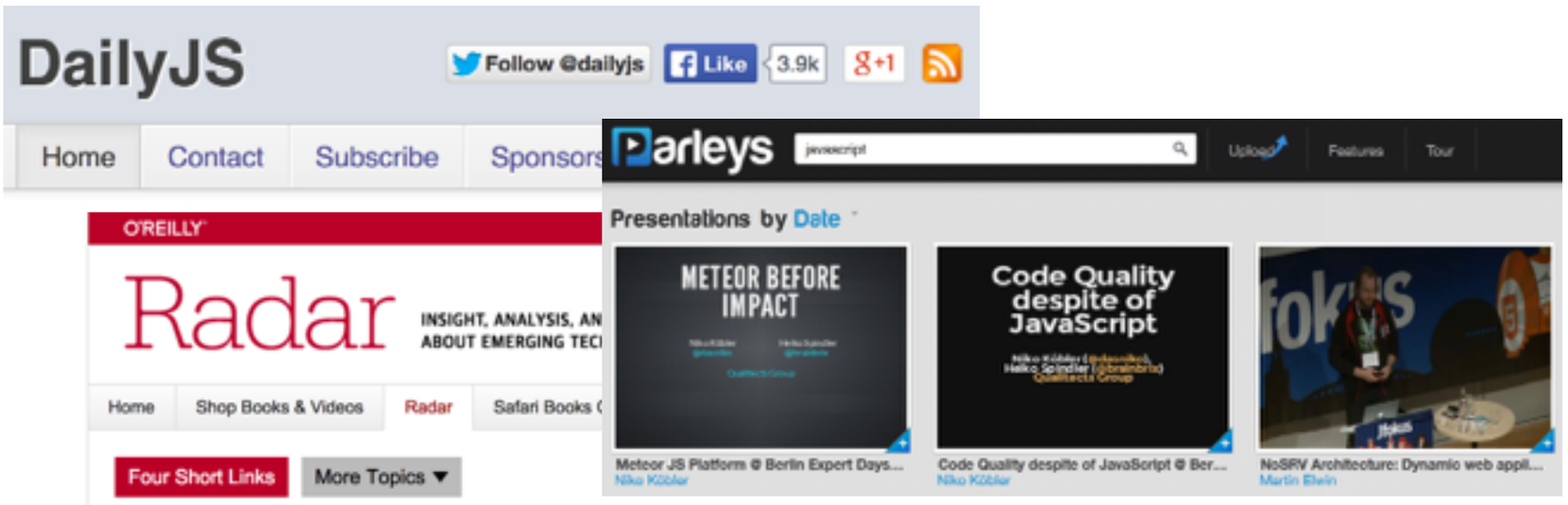
- **Template engines can be used in various architectures:**
  - When we first looked at **Handlebars**, the template engine was used on the **client side**. In this scenario, the JavaScript function builds a model (an object) and injects it into a compiled template. The resulting HTML markup fragment is injected into the DOM.
  - In the project generated by the **express command line tool**, Jade templates are used on the **server side**, at the **request processing time**. In this scenario, the client sends an HTTP request. The server builds a model and injects it into a compiled template. The result is sent back to the client.
  - In other projects, the template engine is invoked as a **pre-processor** in an asset pipeline, at the **deployment time**. In this scenario, when the project is built, the template files are transformed into HTML.
  - **Several template engines can be combined!** For instance, you can use **Jade** instead of HTML to author your pages and do the translation at deployment time (asset pipeline). But the pages can can contain **Handlebar** templates, rendered on the client side.

What did we observe during the lab...?

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

- To get information out of Github, we first had to **be aware** that Github offered an open (REST) API.
- We could have used this API directly (coding an HTTP client). But **being aware** that a npm module was already available for that purpose allowed us to save time.
- **Monitoring** information sources, evaluating new **tools** and **frameworks**, watching **conference** videos is **part of the job (and the fun!).**





TodoMVC

todomvc.com

TodoMVC

Helping you **select** an MV* framework

Download (1.3) | View on GitHub | Blog

Introduction

Developers these days are spoiled with choice when it comes to **selecting** an **MV* framework** for structuring and organizing their JavaScript web apps.

Backbone, Ember, AngularJS... the list of new and stable solutions continues to grow, but just how do you decide on which to use in a sea of so many options?

To help solve this problem, we created **TodoMVC** - a project which offers the

Examples

JavaScript | Compile-to-JS | Labs

These are examples written in pure JavaScript.

Backbone.js R	AngularJS R	Ember.js R
KnockoutJS R	Dojo R	YUI R
Agility.js R	Knockback.js R	CanJS R
Maria R	Polymer R	React R

So you need a template en ×

garann.github.io/template-chooser/

Fork me on GitHub

Template-Engine-Chooser!

▶ Is this for use on the client or the server?

▶ How much logic should it have?

▶ Does it need to be one of the very fastest?

▶ Do you need to pre-compile templates?

▶ Do you need compile-time partials?

▶ Do you want a DOM structure, or just a string?

▶ Aside from template tags, should it be the same language before and after rendering?

dom.js
github

doT.js
project (2.742k)

dust.js (LinkedIn)
github (9.3k)

EJS
project (9.8k)

Handlebars.js
project

Hogan.js
project (2.5k)

ICanHaz.js
project (5.445k)

Jade templates
github (39.687k)

JsRender

Markup.js

Fastness is based on the top three from revisions of this jsPerf.
Remember! If you don't know what it means, you probably don't care about it. ;)

Explore

GitHub, Inc. [US] https://github.com/explore

Search GitHub

Explore Gist Blog Help

wasadigi

Explore GitHub

All Showcases Trending Stars

Explore

Browse interesting projects, solving all types of interesting problems.

Policies

Emoji

Video tools

Design essentials

Alfred Hitchcock would be proud of these repositories. Get your fill of ...

6 3

Writing

GitHub repositories are places where writers can share their work with t...

Music

NoSQL databases

Icon fonts

Showcases

GitHub, Inc. [US] https://github.com/showcases

Search GitHub

Explore Gist Blog Help

wasadigi

Explore GitHub

All Showcases Trending Stars

Showcases

Browse through popular repositories organized around interesting topics.

Search Showcases

Design essentials

Whether you float to the left or float to the right, you'll find these libraries and resources for design, HTML, CSS, and fonts useful.

15 3 12 days ago

Front-end JavaScript frameworks

While the number of ways to organize JavaScript is almost infinite, here are some tools that help you build single-page applications.

9 1 12 days ago

Music

Drop the code bass with these musically themed repositories.

19 9 12 days ago

Government apps

Open source organizations

Policies

Trending JavaScript reposit ×

GitHub, Inc. [US] https://github.com/trending?l=javascript

Search GitHub

Explore Gist Blog Help

wasadigi

Explore GitHub

All Showcases Trending Stars

Trending repositories

Find what repositories the GitHub community is most excited about today.

Repositories Developers

Trending: **today**

zeman/**perfmap**

Front-end performance heatmap bookmarklet.

JavaScript • 265 stars today • Built by

★ Star

pigshell/**pigshell**

The missing shell for the web

JavaScript • 112 stars today • Built by

★ Star

a8m/**angular-filter**

Bunch of useful filters for AngularJS (with no external dependencies!)

JavaScript • 112 stars today • Built by

★ Star

All languages

Unknown languages

C++

CSS

CoffeeScript

Go

Java

JavaScript

Scala

Other: **Languages**

ProTip! Looking for most starred JavaScript repositories? Try this search

Express.js: routers and middleware

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

```
$ express --css stylus

   create : .
   create : ./package.json
   create : ./app.js
   create : ./public
   create : ./public/javascripts
   create : ./public/images
   create : ./public/stylesheets
   create : ./public/stylesheets/style.styl
   create : ./routes
   create : ./routes/index.js
   create : ./routes/users.js
   create : ./views
   create : ./views/index.jade
   create : ./views/layout.jade
   create : ./views/error.jade
   create : ./bin
   create : ./bin/www

   install dependencies:
     $ cd . && npm install

   run the app:
     $ DEBUG=express ./bin/www
```

The are functions (routers)...
... passed here.

```
var express = require('express');
var path = require('path');
var favicon = require('static-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var routes = require('./routes/index');
var users = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(favicon());
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded());
app.use(cookieParser());
app.use(require('stylus').middleware(path.join(__dirname,
'public')));
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);
app.use('/users', users);

/// catch 404 and forward to error handler
app.use(function(req, res, next) {
    var err = new Error('Not Found');
    err.status = 404;
    next(err);
});
```

These are **middleware** components (combined as a **chain of filters**).
Some of them are defined in **third-party modules** (see package.json)

Express.js vs Java EE

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

- Using **Express.js** is very similar to implementing the MVC pattern with **Java EE** technologies.
- We can attach logic to URL patterns. With Java EE, this is done with **servlets**. With Express.js, this is done with **routes**.
- We can build **pipelines**, in which components successively process a client request (for authentication, logging, etc.). With Java EE, this is done with **servlet filters**. With Express.js, this is done with **middleware**.
- We have a **request** and a **response** objects.
- We can send a response directly from the controller, or delegate the work to a view (true MVC). With Express.js, this is done by using `res.send()`, respectively `res.render()`.
- Note: the built-in **static** middleware is very useful (it takes care of **serving static content**). Other middleware components are available in the Connect framework (e.g. for dealing with cookies).

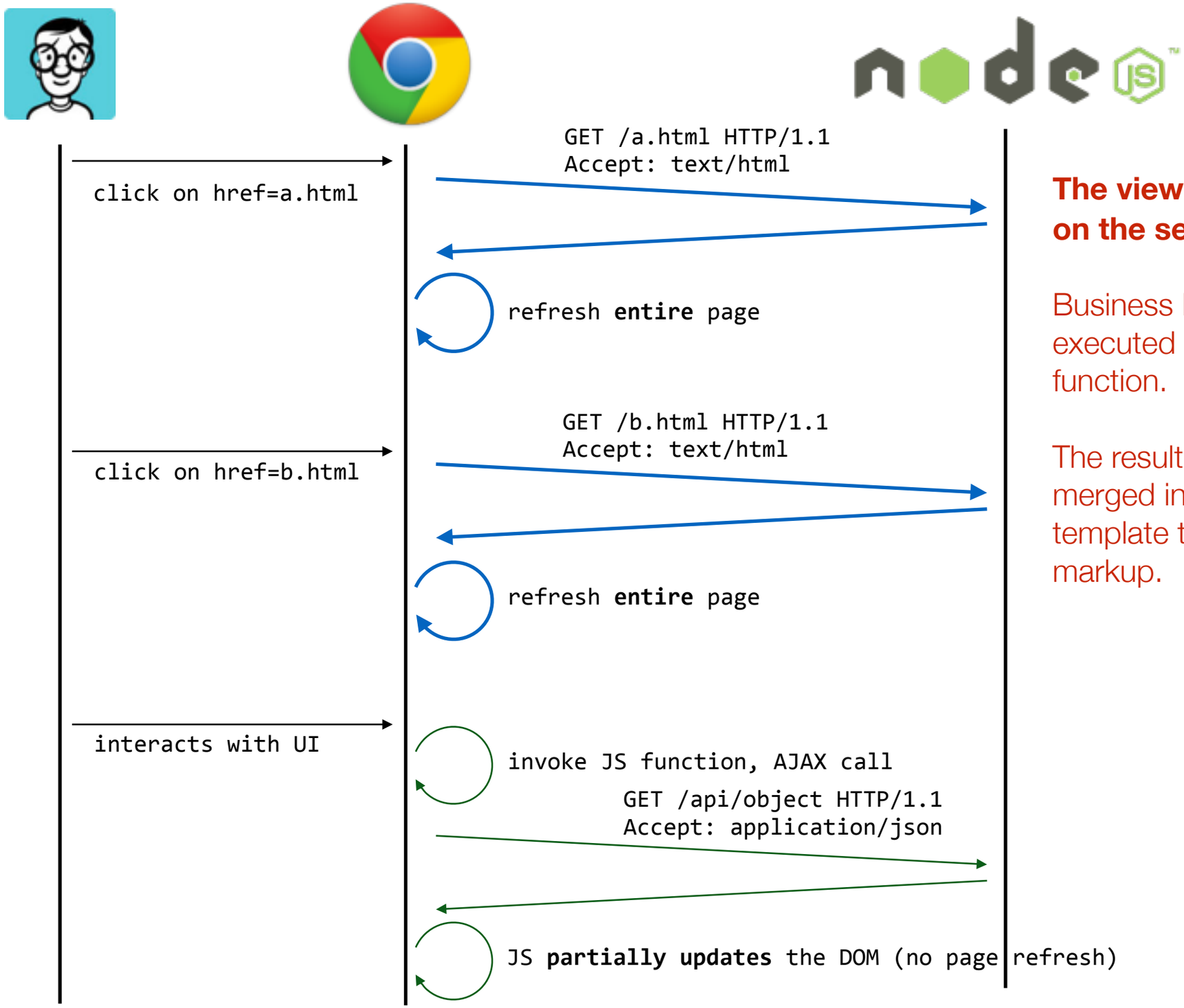
Multi-page application architecture

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

- Although we only had one page in our web app, we assumed a **server-centric, multi-page application architecture**:
  - **Several pages** (controller+view) are created on the server side. They are attached to URL end-points via the **Express.js router**.
  - **Navigation** between pages is implemented on the server side. When the user clicks on a link in a page, a “normal” request (i.e. not an AJAX request) is sent to the server and the response is rendered by the browser as a whole new page.
- **Within one page**, Javascript is used to make AJAX requests in order to fetch data and partially update the DOM (this is what should happen when you click on the “Search” button).
- But in the previous case, **we cannot speak about “page navigation”**, because the user stays on the same page and the context stays the same.

Multi-page application architecture

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



The view is fully built on the sever side:

Business logic is executed in a JavaScript function.

The resulting model is merged into a view template to obtain HTML markup.

Single-page application architecture

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

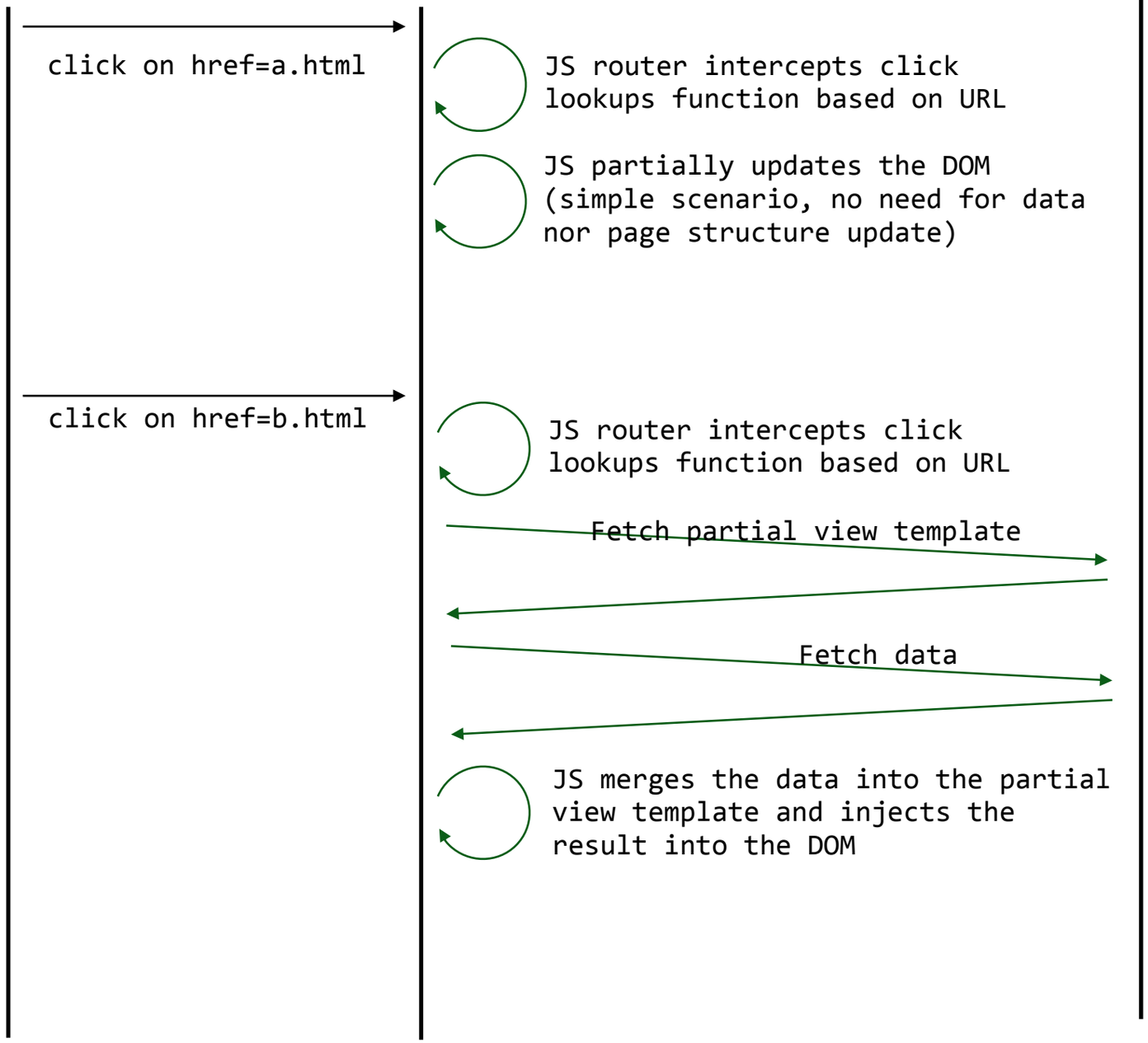
- There is a big trend towards “single-page applications”, where some of the responsibilities are moved from the server to the client side.
  - The client **initially fetches a single** “shell” page, which provides a rendering context and loads application modules (scripts, markup partials, stylesheets, etc.).
  - **When the user clicks on hyperlinks**, the browser does not (immediately) send an HTTP request to fetch a new page. Instead, the event is caught and processed by a JavaScript router on the client side.
- **Routing is done on the client side**. The JavaScript router (typically provided by an application framework) looks at the target URL and decides which JavaScript function needs to be invoked. This function can update the DOM, sometimes in drastic manners (giving the impression that we move from an “Customers List” page to a “Customer Details” page).

http://smalljs.org/client-side-routing/page/
http://visionmedia.github.io/page.js/

Single-page application architecture

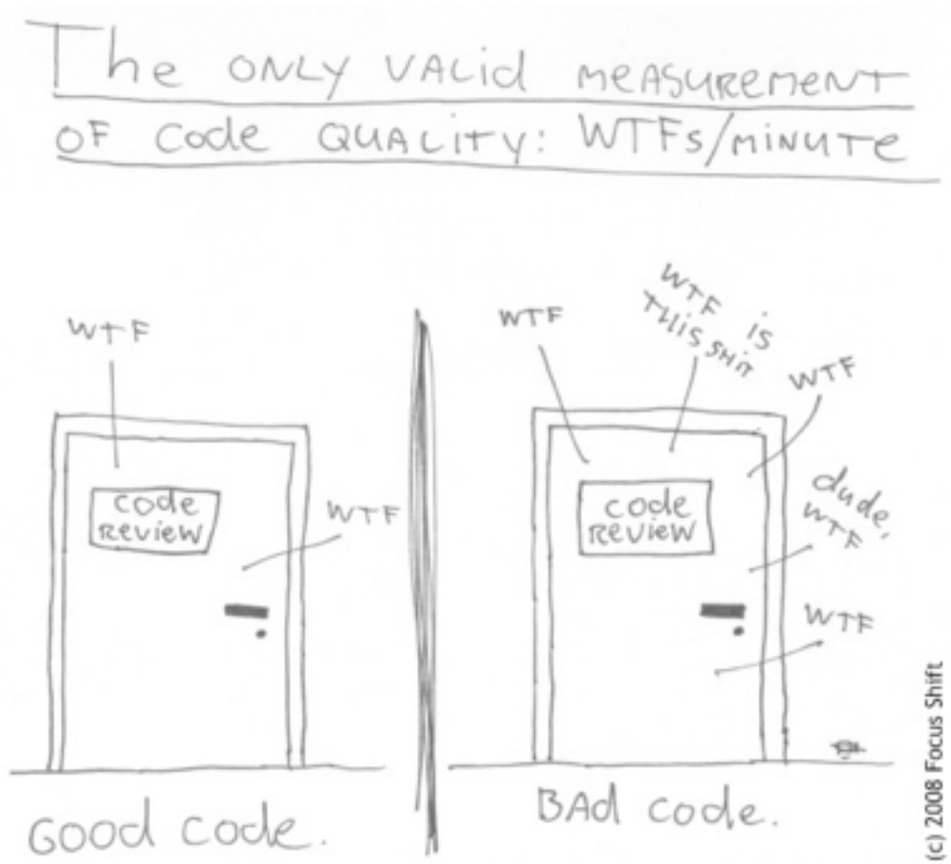
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

The client side handles the navigation logic and builds the views.

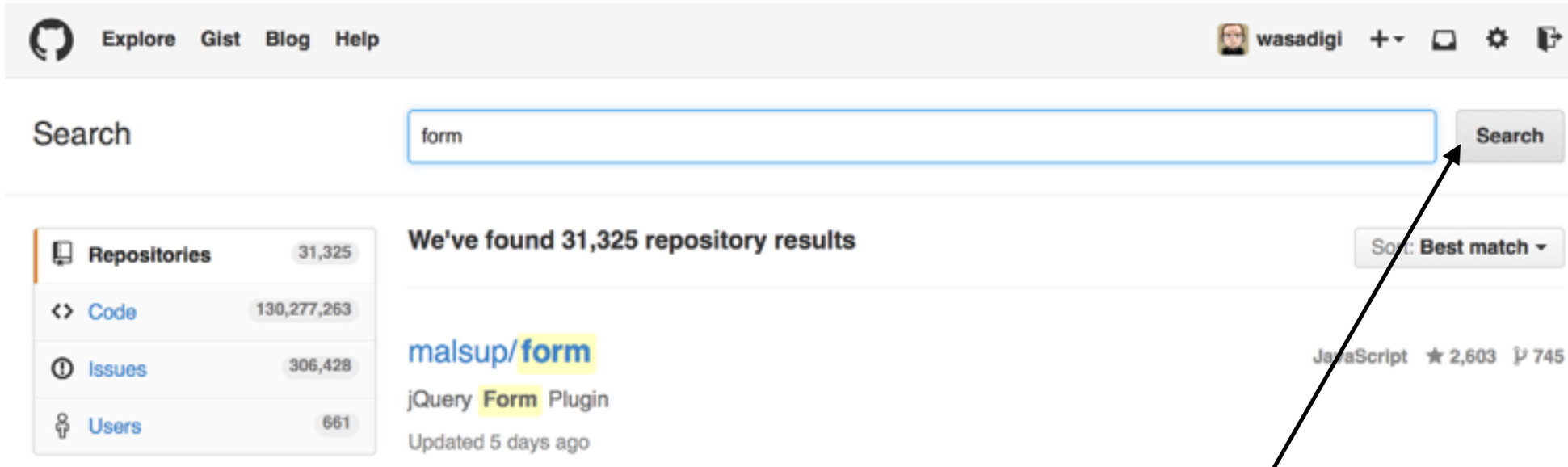


The server side is dumb.

It only serves assets (HTML, Jade, JavaScript, image files).


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
BAD code.
(c) 2008 Focus Shift

Code review



AJAX or no AJAX?

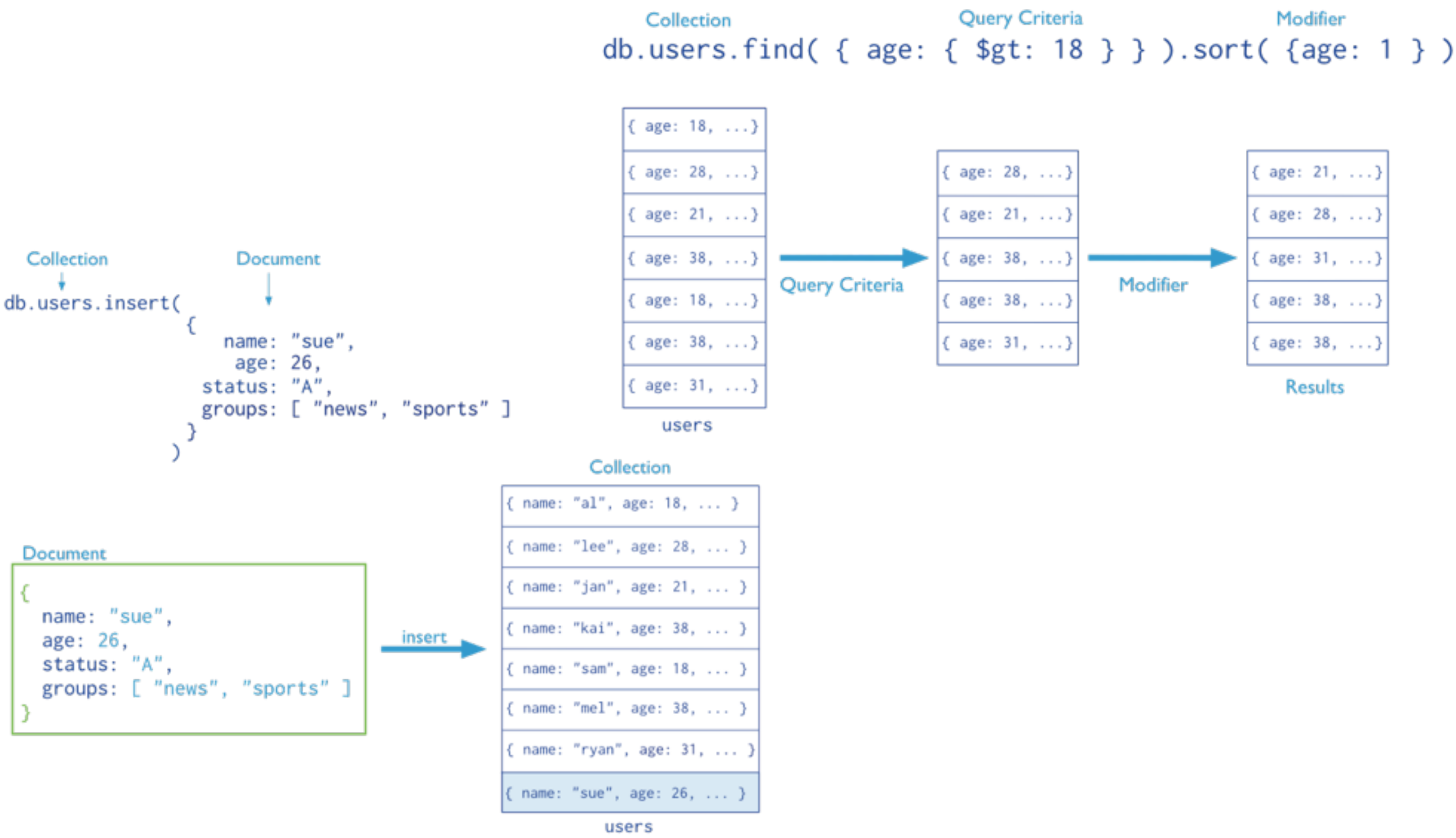
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

1. Setup a database

Let's install a NoSQL database (**MongoDB**)

- We will look at persistence and **study coding techniques later** (in a couple of lectures).
- However, for the initial project configuration to work, we already need to install MongoDB, **both locally and on heroku**.
- MongoDB is a **document-oriented database**. It allows you to store JSON objects. It provides you with a special query language.

Let's install a NoSQL database (**MongoDB**)



http://docs.mongodb.org/manual/core/crud-introduction/

Let's install a NoSQL database (**MongoDB**)

- For your **local environment**, follow the instructions given at http://docs.mongodb.org/manual/installation/.
- At the end of the process, **make sure that your database is running** and that you can connect to it with the command line tool.
- For **heroku**, we will use a **free add-on** offered by a company (MongoHQ), which is provides a **hosted MongoDB service**.
- Before activating the add-on, even if the service is free, we need to **add credit card information** to our heroku account.
- Login on heroku, go to "**Manage Account**" and "**Billing**"
- Enter the **credit card details** provided in the class.

Let’s install a NoSQL database (**MongoDB**)

Let’s allocate **30 minutes** to do the setup.

Start with heroku, then do the local setup.

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

Asynchronous Programming

Asynchronous programming techniques

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

- We have already seen that JavaScript relies on **asynchronous programming**:
  - The JS engine is **single-threaded**. For this reason, IO operations **have to be non-blocking**.
  - An event loop is used both in the browser and on the server (node.js):
    - As the program executes, events are added to a queue. Every event has an associate callback function.
    - A dispatcher takes the next event in the queue and invokes the callback function (on the single thread).
    - When the callback function returns, the dispatcher takes the next event in the queue, and continues forever (it's an event **loop**).

Asynchronous Programming Techniques

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

```
setTimeout( function() {
   console.log(“the callback has been invoked”);
}, 2000);
```

An event will be added to the queue in 2000 ms. In other words, the function passed as the first argument will be invoked in 2 seconds or more (the thread might be busy when the event is posted...).

```
fs.readFile('/etc/passwd', function (err, data)
{
  if (err) throw err;
  console.log(data);
});
```

An event will be added when the file has been fully read (in a non-blocking way). When the event is taken out of the queue, the callback function has access to the file content (data).

Asynchronous Programming Techniques

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


```
$(document).mousemove(function(event){
  $("span").text(event.pageX + ", " +
event.pageY);
});
```

An event will be added to the queue whenever the mouse moves. In each case, the callback function has access to the event attributes (coordinates, key states, etc.).


```
$.get( "ajax/test.html", function( data ) {
$( ".result" ).html( data );
alert( "Load was performed." );
});
```

An event will be added when the AJAX request has been processed, i.e. when a response has been received. The callback function has access to the payload.

Beyond simple callbacks...

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

- The principle of passing a callback function when invoking an asynchronous operation is pretty straightforward.
- Things get more tricky as soon as you want to **coordinate multiple tasks**. Consider this simple example...

Do this first...

... when done, do this.

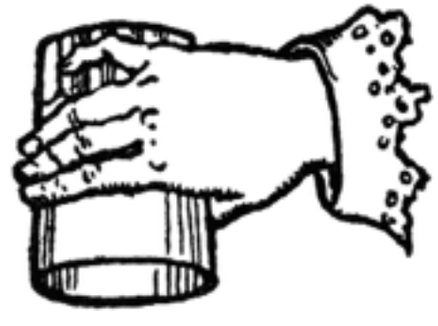
Beyond simple callbacks...

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

- A first attempt...

```
var milkAvailable = false;

function milkCow() {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
  }, 2000);
}

milkCow();
console.log("Can I drink my milk? (" + milkAvailable + ")");
```

FAIL



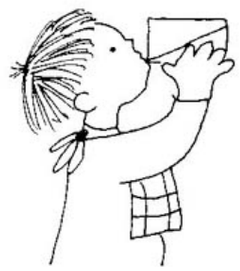
Beyond simple callbacks...

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

- Fixing the issue with a callback...

```
var milkAvailable = false;

function milkCow(done) {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
    done();
  }, 2000);
}

milkCow(function() {
  console.log("Can I drink my milk? (" + milkAvailable + ")");
});
```

SUCCESS



Beyond simple callbacks...

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

- Ok... but what happens when I have **more than 2 tasks** that I want to execute in sequence?
- Let's say we want to have the sequence B, C, D, X, Y, Z, E, F, where X, Y and Z are asynchronous tasks.

```
function f() {
   syncB();
   syncC();
   syncD();
   asyncX();
   asyncY();
   asyncZ();
   syncE();
   syncF();
}
```

```
B result available
C result available
D result available
E result available
Z result available
Y result available
F result available
X result available
```

Beyond simple callbacks...

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

- Ok... but what happens when I have **more than 2 tasks** that I want to execute in sequence?
- Let's say we want to have the sequence B, C, D, X, Y, Z, E, F, where X, Y and Z are asynchronous tasks.

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX(function() {
    asyncY(function() {
      asyncZ(function() {
        syncE();
        syncF();
      });
    });
  });
}
```

```
B result available
C result available
D result available
X result available
Y result available
Z result available
E result available
F result available
```

But welcome to the "callback hell" aka "callback pyramid"

Beyond simple callbacks...

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

- Now, let's imagine that we have 3 asynchronous tasks. We want to invoke them in parallel and **wait until all of them complete**.
- Typical use case: you want to send several AJAX requests (to get different data models) and update your DOM once you have received all responses.

```
function f(done) {

 async1(function(r1) {
    reportResult(r1);
  });
  async2(function(r2) {
    reportResult(r2);
  });
  async3(function(r3) {
    reportResult(r3);
  });

  done();

}
```

Double fail: not only do I invoke done() to early, but also I don't have any result to send back...

Beyond simple callbacks...

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

- Now, let's imagine that we have 3 asynchronous tasks. We want to invoke them in parallel and **wait until all of them complete**.
- Typical use case: you want to send several AJAX requests (to get different data models) and update your DOM once you have received all responses.

```
function f(done) {
  var numberOfPendingTasks = 3;
  var results = [];

  function reportResult(result) {
    result.push(result);
    numberOfPendingTasks -= 1;
    if (numberOfPendingTasks === 0) {
      done(null, results);
    }
  }
  async1(function(r1) {
    reportResult(r1);
  });
  async2(function(r2) {
    reportResult(r2);
  });
  async3(function(r3) {
    reportResult(r3);
  });
}
```

When this reaches 0, I know that all the tasks have completed. I can invoke the "done" callback function that I received from the client. I can pass the array of results to the function.

When a task completes, it invokes this function and passes its result. The result is added to the array and the number of pending tasks is decremented.

The three tasks are asynchronous, so they pass their own callback functions and receive a result when the operation completes.

Beyond simple callbacks...

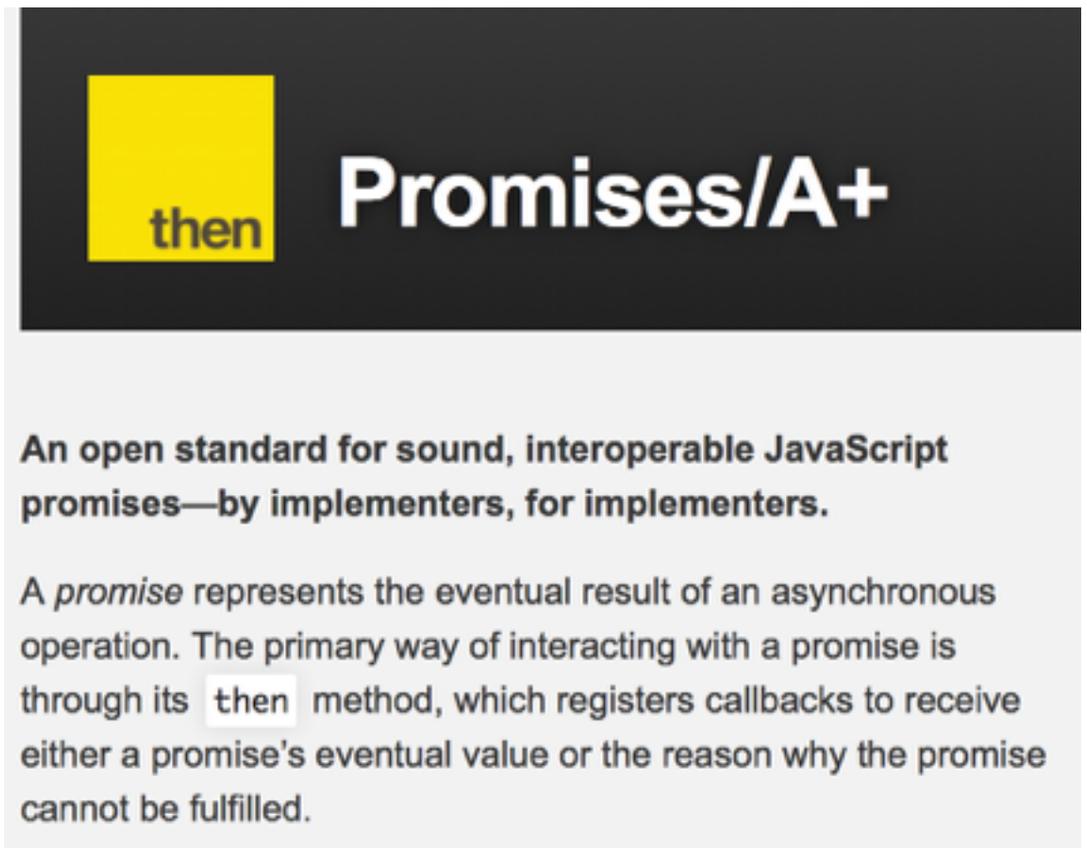
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Give me my Threads back!

Async libs to the rescue

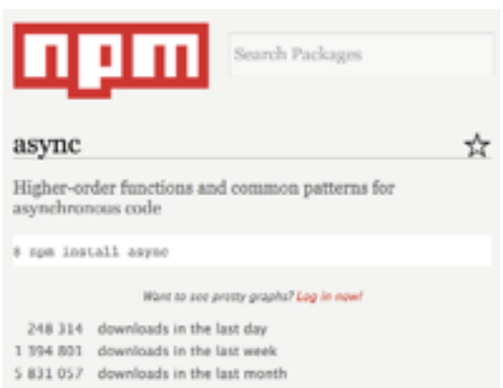
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- Different approaches and libraries have been proposed to make it easier to write asynchronous code.



https://github.com/promises-aplus/promises-spec

http://documentup.com/kriskowal/q/

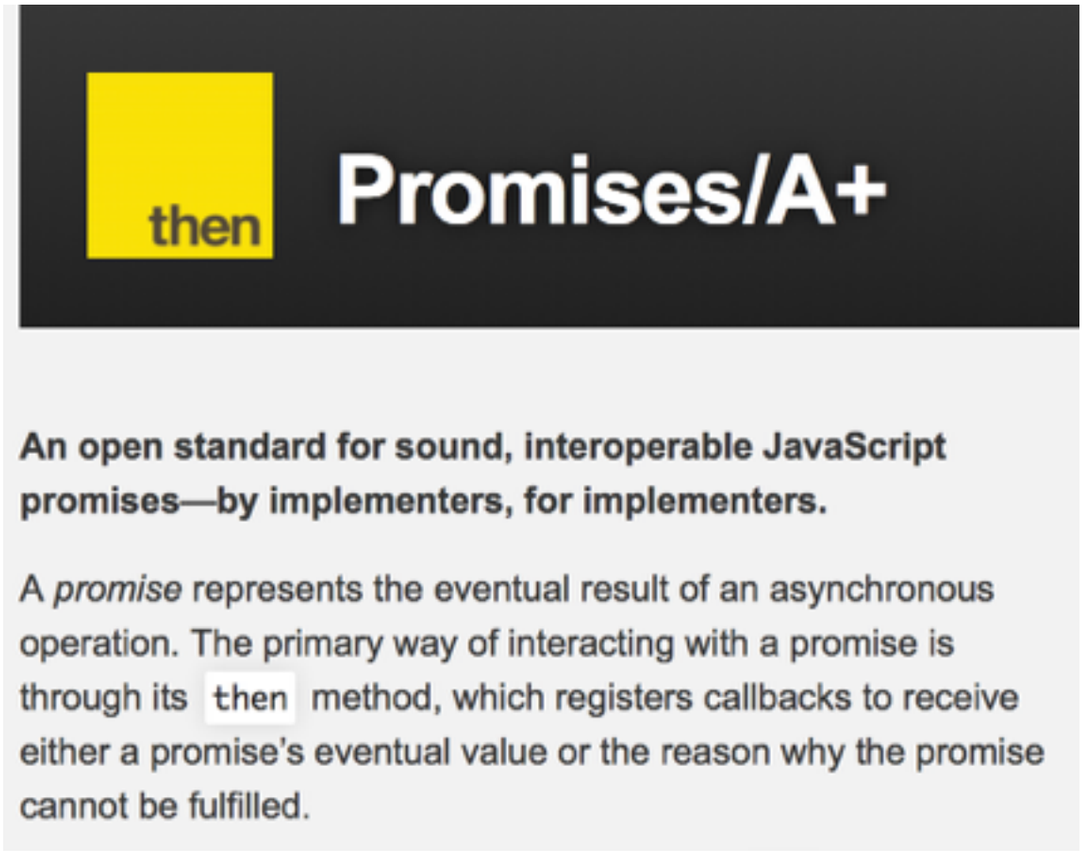


jQuery
write less. do more.
Deffered objects

Promises

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

- "A **promise** must be in **one of three states**: pending, fulfilled, or rejected.
- When **pending**, a promise:
  - may transition to either the fulfilled or rejected state.
- When **fulfilled**, a promise:
  - **must not transition** to any other state.
  - must have a **value**, which must not change.
- When **rejected**, a promise:
  - **must not transition** to any other state.
  - must have a **reason**, which must not change."

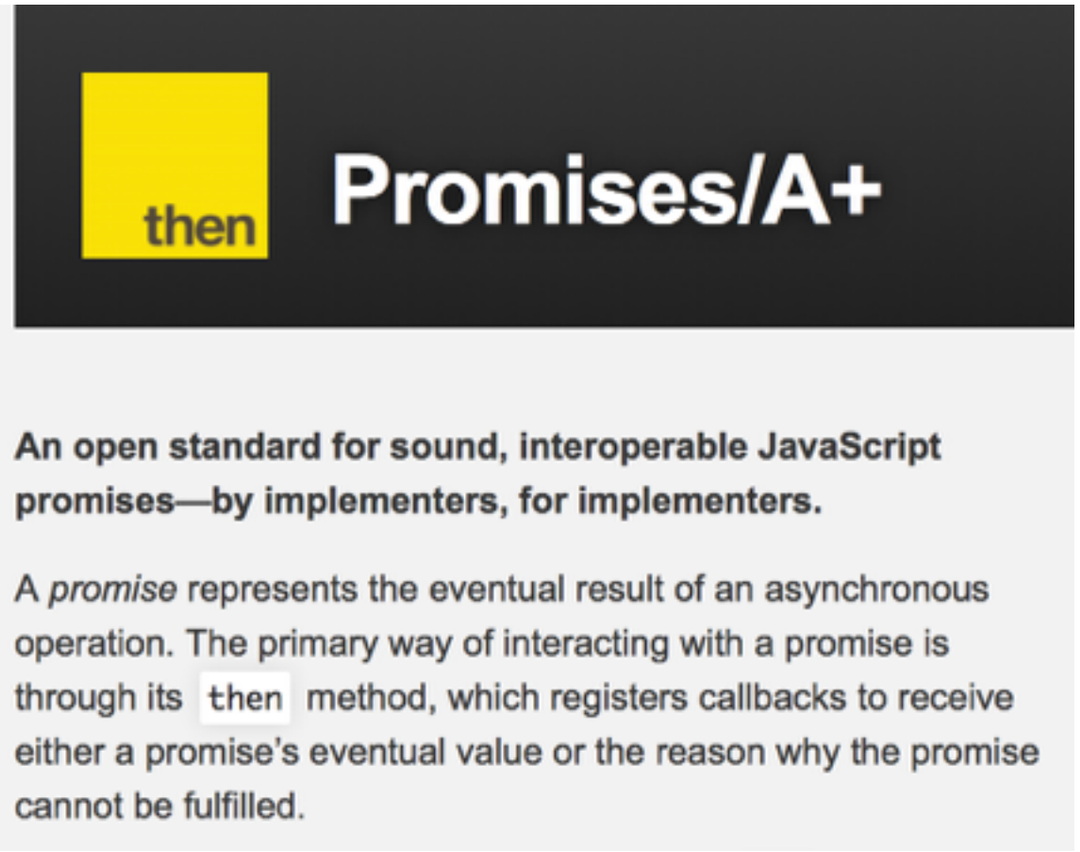


https://github.com/promises-aplus/promises-spec

Promises

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

- **“A promise must provide a `then` method to access its current or eventual value or reason.**
- A promise's then method accepts two arguments:
  - `promise.then(onFullfilled, onRejected)`
- If `onFulfilled` is a function:
  - it must be called after promise is fulfilled, with promise's value as its first argument.
  - it must not be called before promise is fulfilled.
  - it must not be called more than once.
- If `onRejected` is a function,
  - it must be called after promise is rejected, with promise's reason as its first argument.
  - it must not be called before promise is rejected.
  - it must not be called more than once”



https://github.com/promises-aplus/promises-spec

Promises

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

- **“`then` must return a promise [3.3].**
  - `promise2 = promise1.then(onFulfilled, onRejected);`
- If either `onFulfilled` or `onRejected` returns a value x, run the Promise Resolution Procedure [[Resolve]](promise2, x).
- If either `onFulfilled` or `onRejected` throws an exception e, promise2 must be rejected with e as the reason.
- If `onFulfilled` is not a function and promise1 is fulfilled, promise2 must be fulfilled with the same value as promise1.
- If `onRejected` is not a function and promise1 is rejected, promise2 must be rejected with the same reason as promise1.”

https://github.com/promises-aplus/promises-spec

Example 1: Deferred objects in JQuery

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

```
var d1 = new $.Deferred();
var d2 = new $.Deferred();
$.when( d1, d2 ).done(function ( v1, v2 ) {
  console.log( v1 ); // "Fish"
  console.log( v2 ); // "Pizza"
});
d1.resolve( "Fish" );
d2.resolve( "Pizza" );
```

*"a **promise** represents a value that is not yet known*
*a **deferred** represents work that is not yet finished"*

http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt1-theory-and-semantics

```
$.when( $.ajax( "/page1.php" ), $.ajax( "/page2.php" ) )
  .then( myFunc, myFailure );
```

```
$.when( $.ajax( "/page1.php" ), $.ajax( "/page2.php" ) )
  .done( function( a1, a2 ) {
  // a1 and a2 are arguments resolved for the page1 and page2 ajax requests, respectively.
  // Each argument is an array with the following structure: [ data, statusText, jqXHR ]
    var data = a1[ 0 ] + a2[ 0 ]; // a1[ 0 ] = "Whip", a2[ 0 ] = " It"
    if ( /Whip It/.test( data ) ) {
    alert( "We got what we came for!" );
  }
});
```

Example 2: the Q library

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

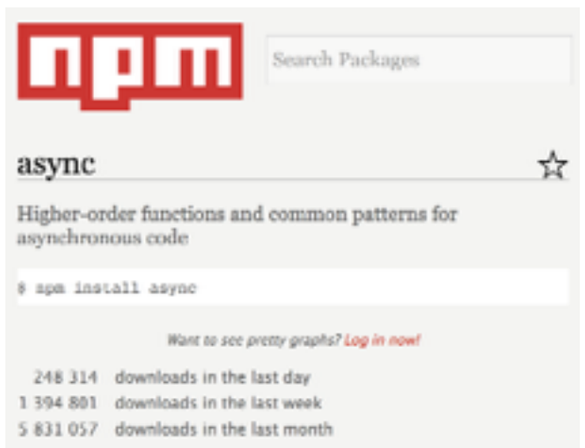
```
function doReadFile() {
  var deferred = Q.defer();
  FS.readFile("foo.txt", "utf-8", function (error, text) {
    if (error) {
      deferred.reject(new Error(error));
    } else {
      deferred.resolve(text);
    }
  });
  return deferred.promise;
}

doReadFile().then() {
  console.log(“task done”);
}
```

http://documentup.com/kriskowal/q/

Example 3: the async.js library

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

```
async.map(['file1','file2','file3'], fs.stat, function(err, results){
    // results is now an array of stats for each file
});

async.filter(['file1','file2','file3'], fs.exists, function(results){
    // results now equals an array of the existing files
});

async.parallel([
    function(){ ... },
    function(){ ... }
], callback);

async.series([
    function(){ ... },
    function(){ ... }
]);
```



Collections

- each
- eachSeries
- eachLimit
- map
- mapSeries
- mapLimit
- filter
- filterSeries
- reject
- rejectSeries
- reduce
- reduceRight
- detect
- detectSeries
- sortBy
- some
- every
- concat
- concatSeries

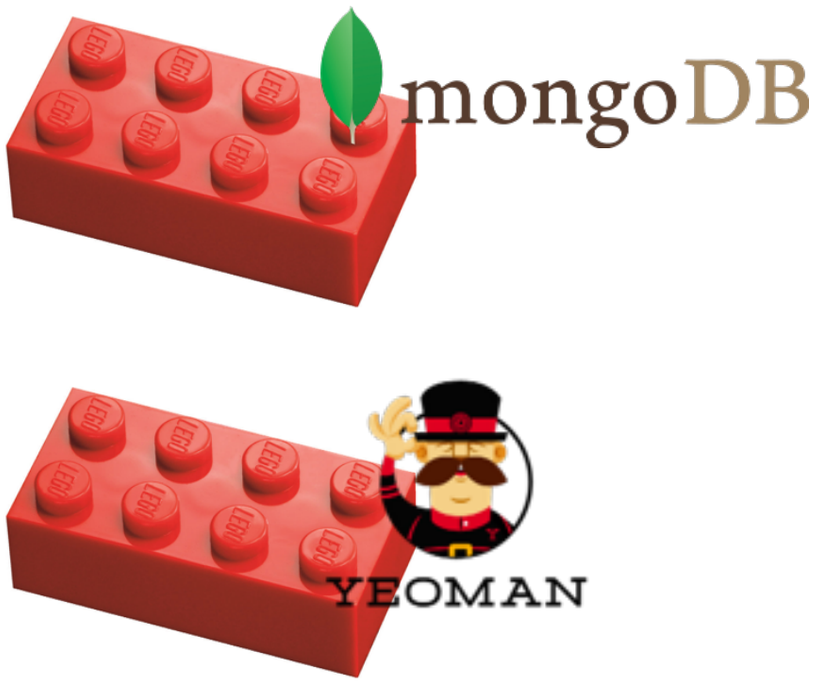
Control Flow

- series
- parallel
- parallelLimit
- whilst
- doWhilst
- until
- doUntil
- forever
- waterfall
- compose
- seq
- applyEach
- applyEachSeries
- queue
- priorityQueue
- cargo
- auto
- retry
- iterator
- apply
- nextTick
- times
- timesSeries

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

Project Building Blocks

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



1. Setup a **database** (~30')

2. Create a project **skeleton** (~30')

3. Baby steps with **Angular.js** (~30')

4. Display and control **PDF** (~30')

heig-vd

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

2. Create a project skeleton

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?

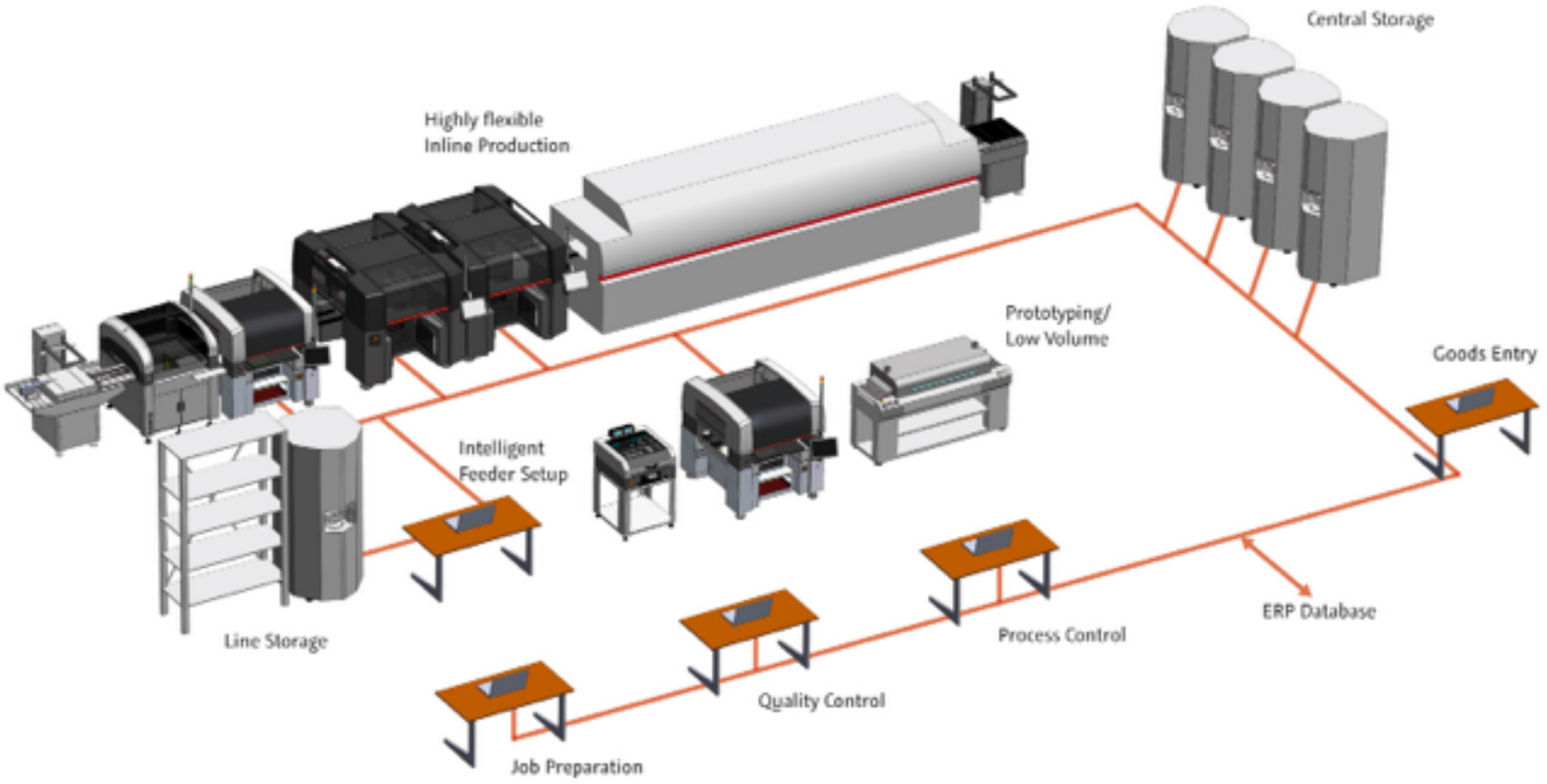
https://www.youtube.com/watch?v=vDbbz-BdyYc

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

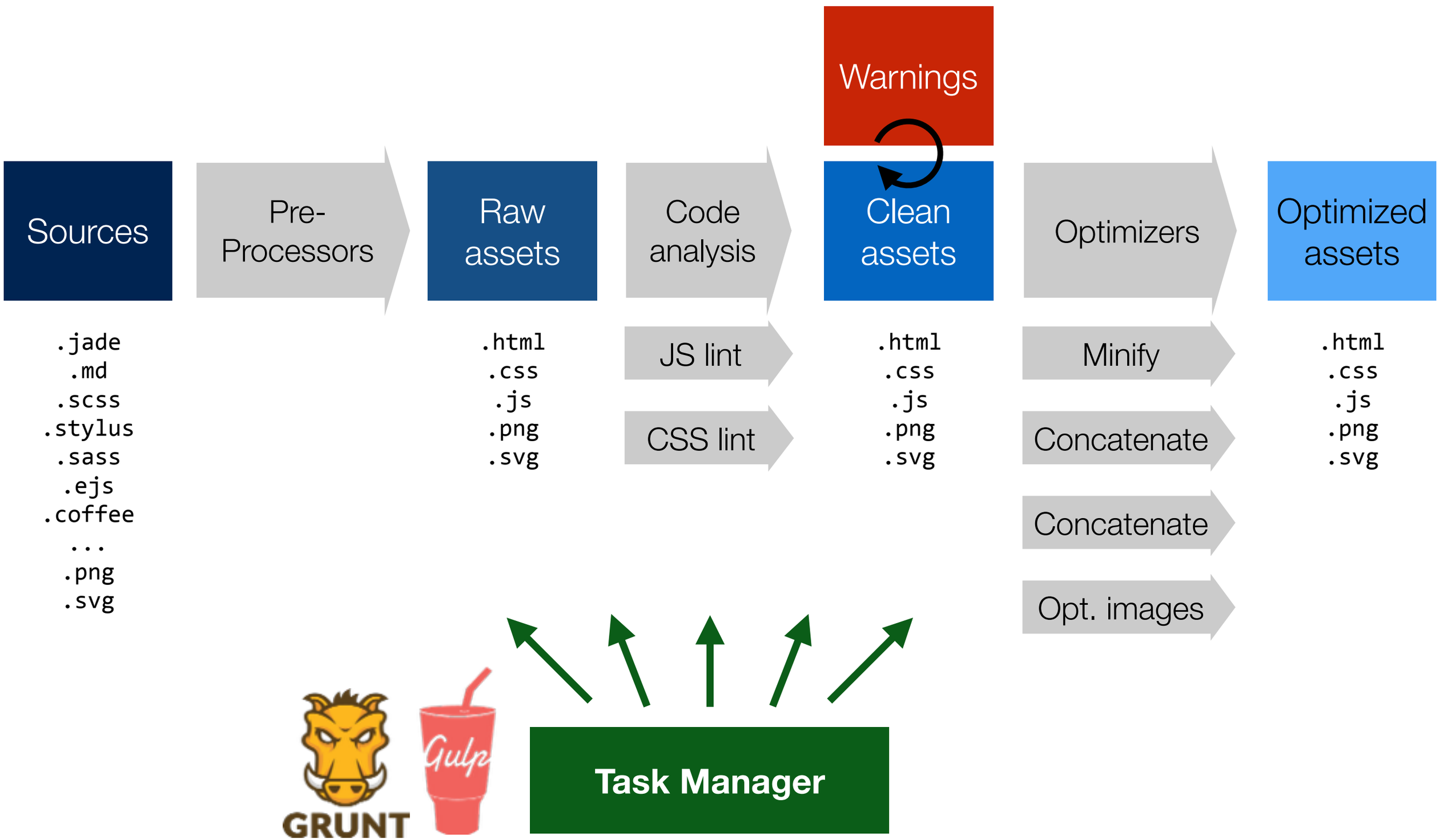
heig-vd

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

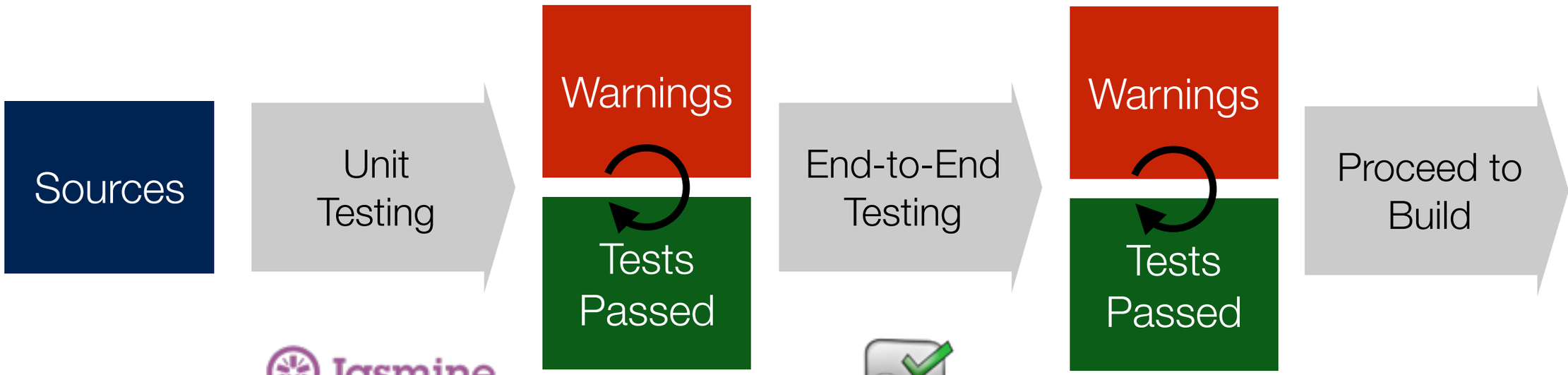


Development Pipeline

Build Pipeline



Test Pipeline



Jasmine

mocha

QUnit
js unit testing

Protractor
end to end testing for AngularJS

PhantomJS

KARMA

GRUNT

Gulp

Task Manager

Grunt

Getting Started · Plugins · Documentation · API

GRUNT

The JavaScript Task Runner

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

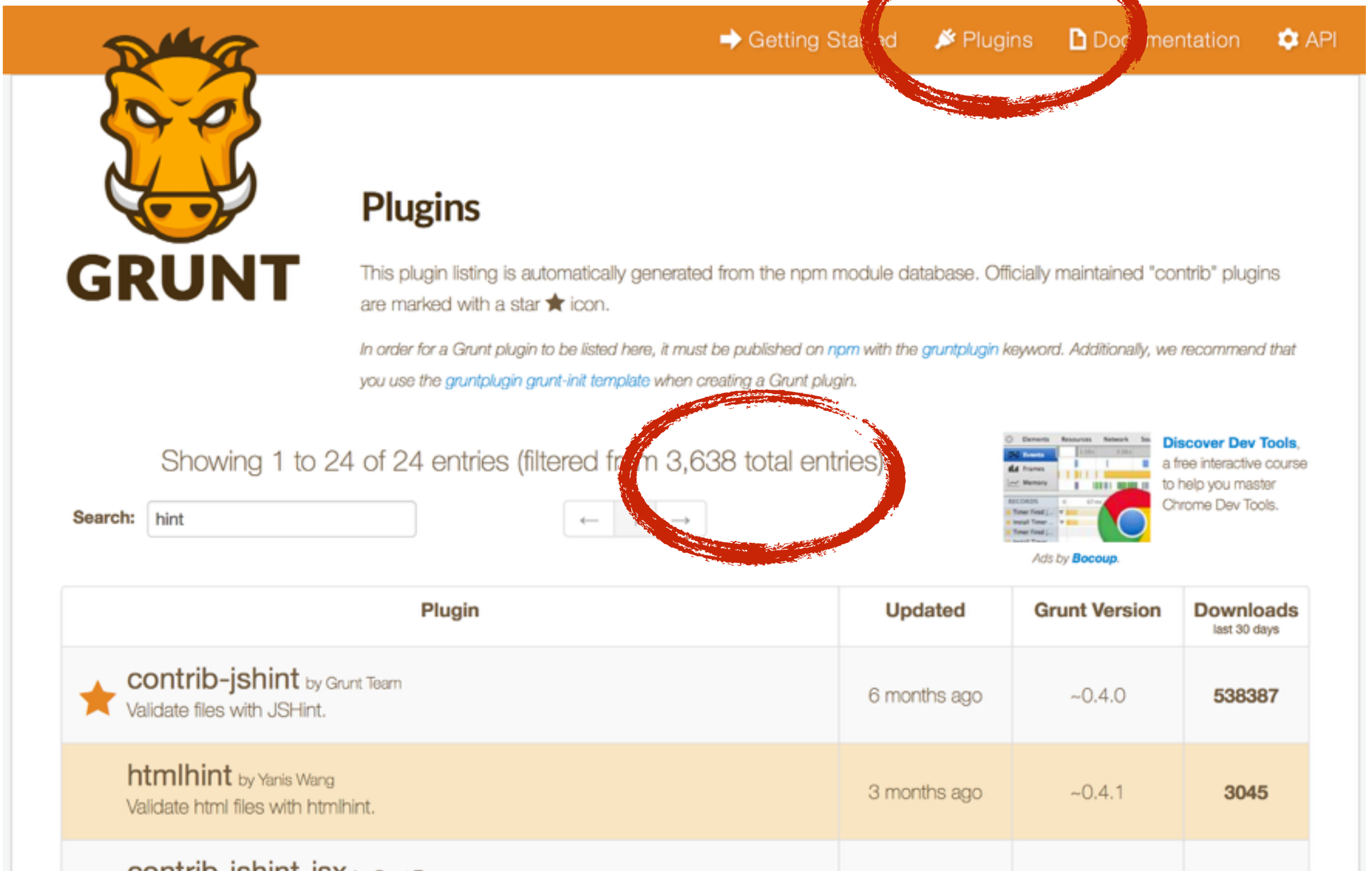
Many of the tasks you need are already available as Grunt Plugins, and new plugins are published every day. While the plugin listing is more complete, here's a few you may have heard of.

CoffeeScript · handlebars · jade (Node Template Engine) · JSHint · less · RequireJS · Sass · stylus

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


Getting Started
Plugins
Documentation
API
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: hint
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
contrib-jshint by Grunt Team
Validate files with JSHint.
6 months ago
~0.4.0
538387
htmlhint by Yanis Wang
Validate html files with htmlhint.
3 months ago
~0.4.1
3045
contrib-jshint-jsx by Grunt Team

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

```
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']);

};
```

http://gruntjs.com/sample-gruntfile

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.

Executing a more complex workflow...

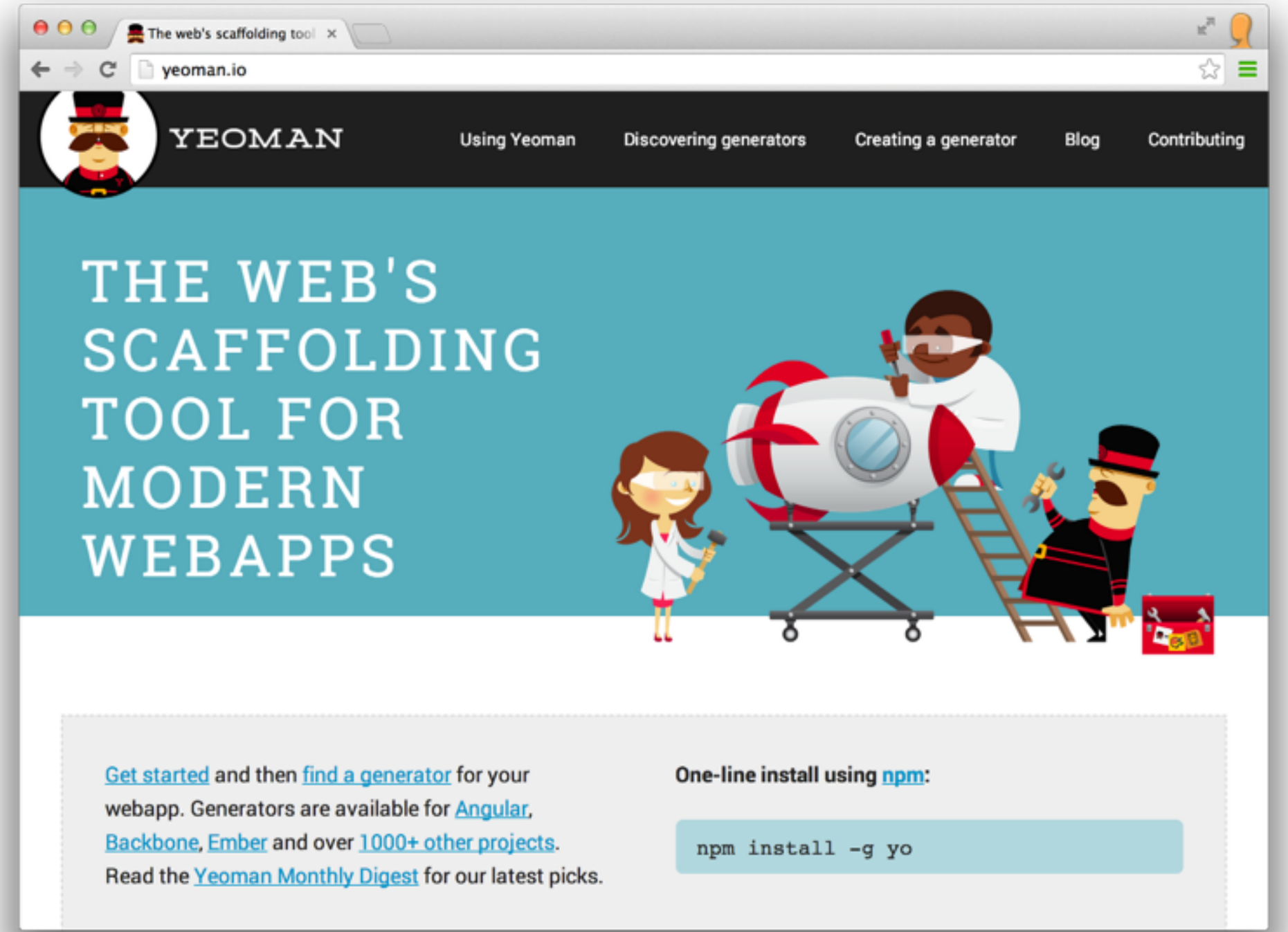
```
MacBook-Pro-de-admin:demo admin$ grunt build
Running "clean:dist" (clean) task
Cleaning .tmp...OK
Cleaning dist/package.json...OK
Cleaning dist/public...OK
Cleaning dist/server...OK

Running "injector:stylus" (injector) task
Missing option `template`, using `dest` as template instead
Injecting styl files (5 files)
>> Nothing changed

Running "concurrent:dist" (concurrent) task

    Running "stylus:server" (stylus) task
    File .tmp/app/app.css created.

    Done, without errors.


    Execution Time (2014-10-13 04:48:31 UTC)
    loading tasks  121ms  26%
    stylus:server  343ms  74%
    Total 464ms

    Running "jade:compile" (jade) task
    File .tmp/app/about/about.html created.
    File .tmp/app/account/login/login.html created.
    File .tmp/app/account/settings/settings.html created.
    File .tmp/app/account/signup/signup.html created.
    File .tmp/app/admin/admin.html created.
    File .tmp/app/main/main.html created.
    File .tmp/components/modal/modal.html created.
    File .tmp/components/navbar/navbar.html created.

    Done, without errors.


    Execution Time (2014-10-13 04:48:31 UTC)
    loading tasks  121ms  20%
    jade:compile   497ms  80%
    Total 619ms

    Running "svgmin:dist" (svgmin) task
    Total saved: 0 B

    Done, without errors.


    Execution Time (2014-10-13 04:48:31 UTC)
    loading tasks  476ms  67%
    svgmin:dist    234ms  33%
    Total 710ms

    Running "imagemin:dist" (imagemin) task
    ✔ client/assets/images/yeoman.png (saved 3.73 kB - 30%)
    Minified 1 image (saved 3.73 kB)

    Done, without errors.


    Execution Time (2014-10-13 04:48:31 UTC)
    loading tasks     1s  83%
    imagemin:dist  207ms  17%
    Total 1.2s

Running "injector:scripts" (injector) task
Missing option `template`, using `dest` as template instead
Injecting js files (16 files)
>> Nothing changed

Running "injector:stylus" (injector) task
Missing option `template`, using `dest` as template instead
Injecting styl files (5 files)
>> Nothing changed
```

```
Running "injector:css" (injector) task
Missing option `template`, using `dest` as template instead
>> Nothing changed

Running "wiredep:target" (wiredep) task

Running "useminPrepare:html" (useminPrepare) task
Going through client/index.html to update the config
Looking for build script HTML comment blocks

Configuration is now:

  concat:
  { generated:
   { files:
      [ { dest: '.tmp/concat/app/vendor.css', src: [] },
        { dest: '.tmp/concat/app/app.css',
          src: [ '{.tmp,client}/app/app.css' ] },
        { dest: '.tmp/concat/app/vendor.js',
          src:
           [ '{client,node_modules}/bower_components/jquery/dist/jquery.js',
             '{client,node_modules}/bower_components/angular/angular.js',
             '{client,node_modules}/bower_components/angular-resource/angular-resource.js',
             '{client,node_modules}/bower_components/angular-cookies/angular-cookies.js',
             '{client,node_modules}/bower_components/angular-sanitize/angular-sanitize.js',
             '{client,node_modules}/bower_components/angular-bootstrap/ui-bootstrap-tpls.js',
             '{client,node_modules}/bower_components/lodash/dist/lodash.compat.js',
             '{client,node_modules}/bower_components/angular-socket-io/socket.js',
             '{client,node_modules}/bower_components/angular-ui-router/release/angular-ui-router.js',
             '{client,node_modules}/socket.io-client/socket.io.js' ] },
        { dest: '.tmp/concat/app/app.js',
          src:
           [ '{.tmp,client}/app/app.js',
             '{.tmp,client}/app/about/about.controller.js',
             '{.tmp,client}/app/about/about.js',
             '{.tmp,client}/app/account/account.js',
             '{.tmp,client}/app/account/login/login.controller.js',
             '{.tmp,client}/app/account/settings/settings.controller.js',
             '{.tmp,client}/app/account/signup/signup.controller.js',
             '{.tmp,client}/app/admin/admin.controller.js',
             '{.tmp,client}/app/admin/admin.js',
             '{.tmp,client}/app/main/main.controller.js',
             '{.tmp,client}/app/main/main.js',
             '{.tmp,client}/components/auth/auth.service.js',
             '{.tmp,client}/components/auth/user.service.js',
             '{.tmp,client}/components/modal/modal.service.js',
             '{.tmp,client}/components/mongoose-error/mongoose-error.directive.js',
             '{.tmp,client}/components/navbar/navbar.controller.js',
             '{.tmp,client}/components/socket/socket.service.js' ] } ] } }

  uglify:
  { generated:
   { files:
      [ { dest: 'dist/public/app/vendor.js',
          src: [ '.tmp/concat/app/vendor.js' ] },
        { dest: 'dist/public/app/app.js',
          src: [ '.tmp/concat/app/app.js' ] } ] } }

  cssmin:
  { generated:
   { files:
      [ { dest: 'dist/public/app/vendor.css',
          src: [ '.tmp/concat/app/vendor.css' ] },
        { dest: 'dist/public/app/app.css',
          src: [ '.tmp/concat/app/app.css' ] } ] } }

Running "autoprefixer:dist" (autoprefixer) task
File .tmp/app/app.css created.

Running "ngtemplates:main" (ngtemplates) task
>> No templates found
File .tmp/templates.js created.
Added .tmp/templates.js to <!-- build:js app/app.js -

Running "ngtemplates:tmp" (ngtemplates) task
File .tmp/tmp-templates.js created.
Added .tmp/tmp-templates.js to <!-- build:js app/app.
```

```
Running "concat:generated" (concat) task
File .tmp/concat/app/vendor.css created.
File .tmp/concat/app/app.css created.
File .tmp/concat/app/vendor.js created.
File .tmp/concat/app/app.js created.

Running "ngAnnotate:dist" (ngAnnotate) task
>> 2 files successfully generated.

Running "copy:dist" (copy) task
Created 68 directories, copied 356 files

Running "cdnify:dist" (cdnify) task
Going through dist/public/index.html to update script refs
✔ bower_components/jquery/dist/jquery.js changed to //ajax.googleapis.com/ajax/libs/jquery/1
✔ bower_components/angular/angular.js changed to //ajax.googleapis.com/ajax/libs/angularjs/1
✔ bower_components/angular-cookies/angular-cookies.js changed to //ajax.googleapis.com/ajax/
✔ bower_components/angular-resource/angular-resource.js changed to //ajax.googleapis.com/aja
✔ bower_components/angular-sanitize/angular-sanitize.js changed to //ajax.googleapis.com/aja

Running "cssmin:generated" (cssmin) task
>> Destination not written because minified CSS was empty.
File dist/public/app/app.css created: 146.58 kB → 120.4 kB

Running "uglify:generated" (uglify) task
File dist/public/app/vendor.js created: 1.8 MB → 398.01 kB
File dist/public/app/app.js created: 26.05 kB → 16.02 kB

Running "rev:dist" (rev) task
dist/public/app/app.js >> 4e83113a.app.js
dist/public/app/vendor.js >> b05ecff2.vendor.js
dist/public/app/app.css >> 32a41561.app.css
dist/public/assets/images/yeoman.png >> d535427a.yeoman.png

Running "usemin:html" (usemin) task

Processing as HTML - dist/public/index.html
Update the HTML to reference our concat/min/revved script files
<script src="app/vendor.js" changed to <script src="app/b05ecff2.vendor.js"
<script src="app/app.js" changed to <script src="app/4e83113a.app.js"
Update the HTML with the new css filenames
<link rel="stylesheet" href="app/app.css" changed to <link rel="stylesheet" href="app/32a415
Update the HTML with the new img filenames
Update the HTML with data-main tags
Update the HTML with data-* tags
Update the HTML with background imgs, case there is some inline style
Update the HTML with anchors images
Update the HTML with reference in input

Running "usemin:css" (usemin) task

Processing as CSS - dist/public/app/32a41561.app.css
Update the CSS to reference our revved images

Running "usemin:js" (usemin) task
```

```
Execution Time (2014-10-13 04:48:30 UTC)
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  24%
cdnify:dist          5.4s  27%
cssmin:generated    596ms   3%
uglify:generated       4s  20%
Total 19.8s
```

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**?

Generators | Yeoman

yeoman.io/generators/

YEOMAN

Using Yeoman | Discovering generators | Creating a generator | Blog | Contributing

GENERATORS

Your generator must have the "yeoman-generator" keyword and a repo description to be listed. Official generators are marked with

express

Name	Description	Author	Stars
angular-fullstack	AngularJS with an Express server	Tyler Henkel	1985
express	An express generator for Yeoman, based on the express command line tool	petecoop	180
mean-seed	MEAN Seed / MEAN Stack (AngularJS, node.js app) - MongoDB, Express, Angular, Node, Grunt, Bower, Yo. 'Core' and 'Module' subgenerators for	Luke Madera	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
angular	AngularJS	The Yeoman Team	2617
angular-fullstack	AngularJS with an Express server	Tyler Henkel	1985

AngularJS Full-Stack generator build passing

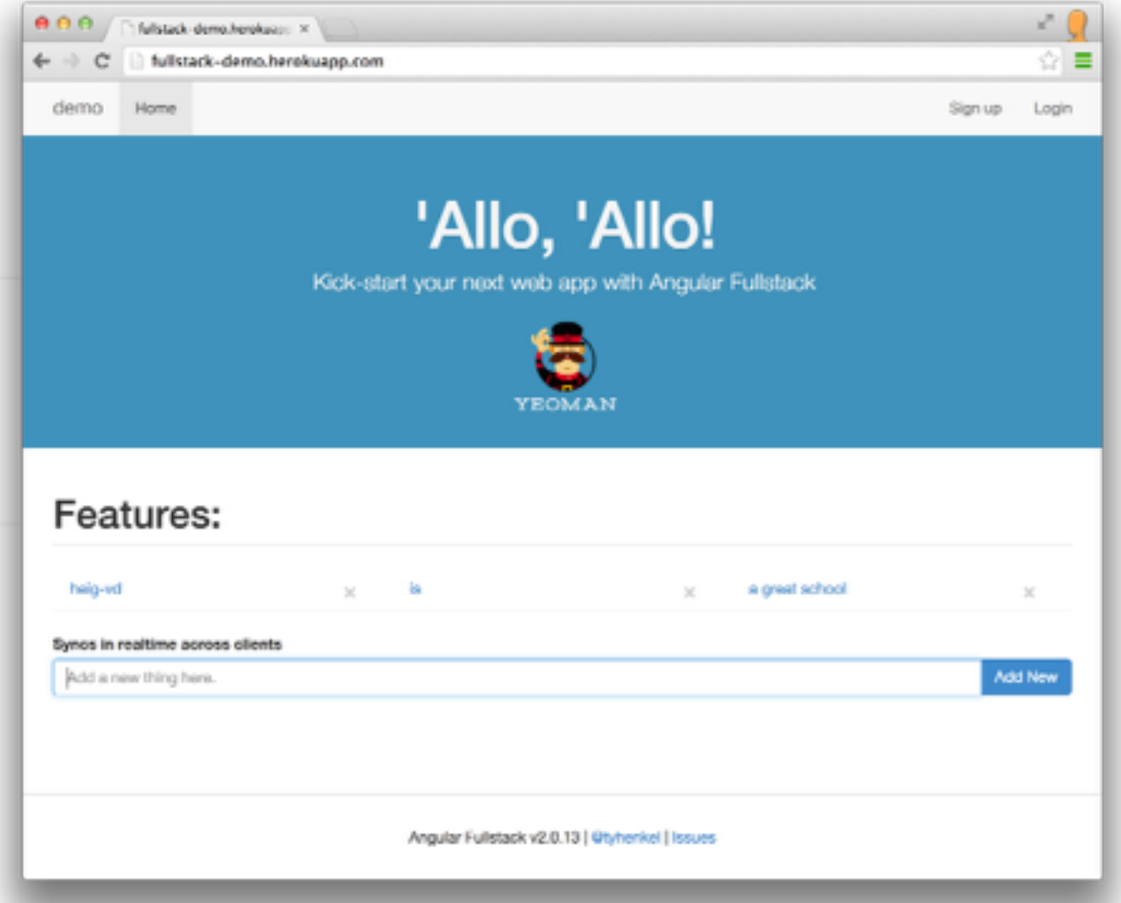
GITTER JOIN CHAT →

> 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



How and **why** did I select this generator?

- I knew that I wanted to use **Express.js on the server side**.
- I knew that we will study the **Angular.js** client-side framework later in the semester, but I was **not sure whether it was a good idea to use it** for the first project (because it is one more thing to use).
- I knew that we need to deal with **persistence** (with MongoDB), with **push notifications** (with Socket.io) and with **authentication**. All of these can be added to an existing project, but if they are part of the skeleton, it can be interesting.
- I identified **between 5 and 10 candidates** (with some generators offering different skeleton types). I invested a bit more than one day to evaluate them (creating skeletons, building the projects, skimming through the repo and source code).
- In the end, I decided to **go for a "feature-rich" skeleton**, rather than a "lightweight" one. Some features of the angular-fullstack generator are really compelling (pre-integrated frameworks, heroku integration, full workflow). It is also the 2nd most popular generator.
- Based on my experience, I **believe** that investing the time to go through the learning curve will allow me to save time. **But I have to be ready to deal with the complexity of the skeleton and to spend time experimenting, digging into the documentation and code, etc.**

Setting up your **project structure**

- Install **yo** (see http://yeoman.io/learning/index.html)
- Install the **angular-fullstack generator** and run it a first time to see what it looks like. Don't do it in your project folder, you will generate the real skeleton after playing with the generator a couple of times (https://github.com/DaftMonk/generator-angular-fullstack)

Installing yo and some generators

First, you'll need to install `yo` and other required tools:

```
$ npm install -g yo
```

Usage

Install `generator-angular-fullstack`:

```
npm install -g generator-angular-fullstack
```

Make a new directory, and `cd` into it:

```
mkdir my-new-project && cd $_
```

Run `yo angular-fullstack`, optionally passing an app name:

```
yo angular-fullstack [app-name]
```

Run `grunt` for building, `grunt serve` for preview, and `grunt serve:dist` for a preview of the built app.

Which options did I use for my skeleton?

- JavaScript
- **Jade**
- **Stylus**
- **uiRouter**
- include Bootstrap: Yes
- include UI Bootstrap: Yes
- MongoDB with Mongoose: Yes
- Scaffold authentication: Yes (select the 3 oAuth strategies)
- use socket.io: Yes

Generate your project skeleton

Let's allocate **30 minutes** to do the setup.

Do it twice.

Do a **first test** in a **temporary** directory. Do a "`grunt build`" and a "`grunt serve`" and test your app.

If everything is ok, generate the skeleton in your **local git repo**. Build and test your app locally. **Commit and push to GitHub**.

Follow the **heroku-specific** instructions carefully: https://github.com/DaftMonk/generator-angular-fullstack#heroku

So... what can I find in this skeleton?

```
$ ls -1G
Gruntfile.js
bower.json
client
dist
e2e
karma.conf.js
node_modules
package.json
protractor.conf.js
server
```

- `Gruntfile.js` → The **task runner** configuration. What happens when you build.
- `bower.json` → The “web **dependencies**”
- `client` → The **client-side assets**, including you **angular.js** componets.
- `dist` → The **output of the build process**. What you will deploy to heroku.
- `e2e` → End-to-end automated tests.
- `karma.conf.js` → **Test runner** configuration.
- `node_modules` → The **npm** modules used by your app **or** by the build process.
- `package.json` → The **dependencies** and **script** configuration.
- `protractor.conf.js` → **Test** configuration.
- `server` → **THe server-side assets**, on top of express.js.

```
$ du -sh *
 20K   Gruntfile.js
4,0K   bower.json
7,7M   client
8,2M   dist
 16K   e2e
4,0K   karma.conf.js
249M   node_modules
4,0K   package.json
4,0K   protractor.conf.js
152K   server
```

So... what can I find in this skeleton?

```
$ cd server/
$ ls -1G
api
app.js
auth
components
config
routes.js
views
```

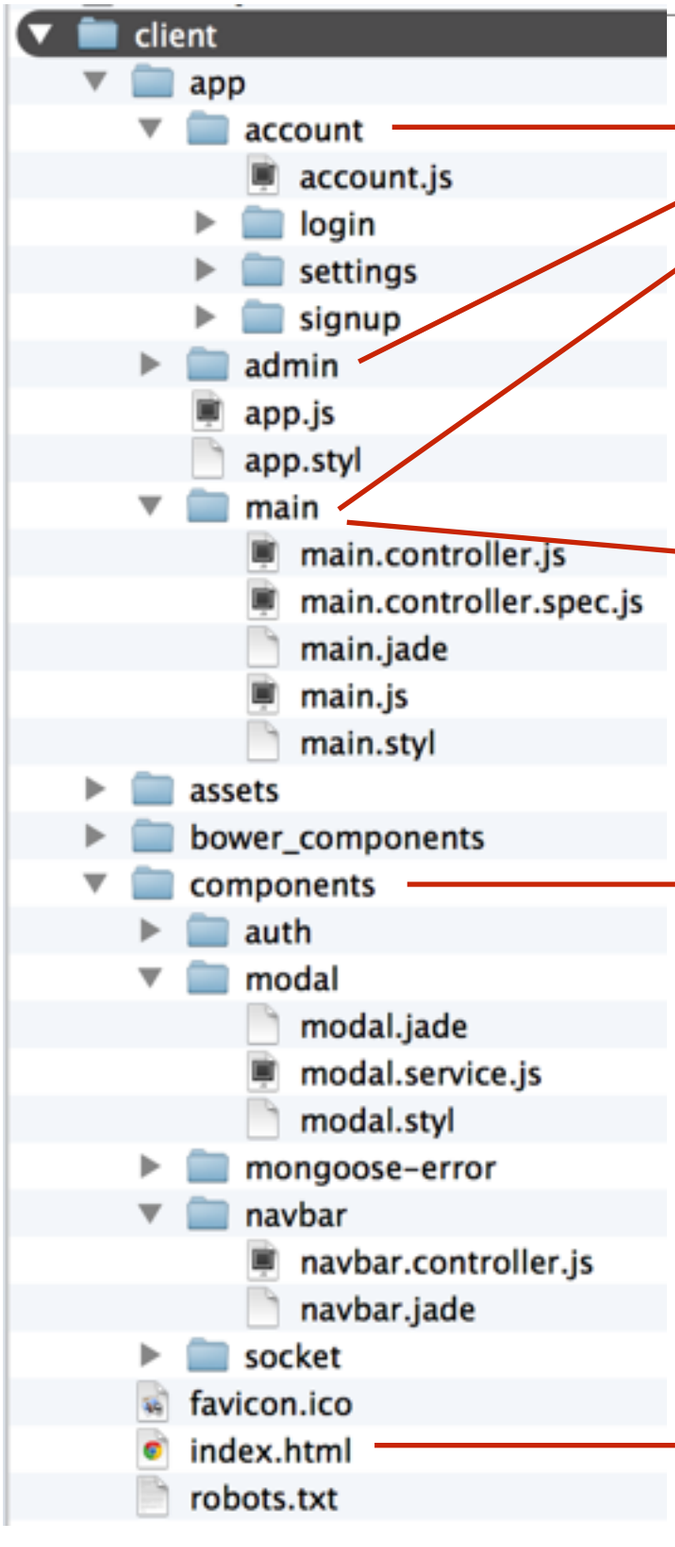
The **REST API handlers** that give access to data (users, etc.). This is where we will do most of the extension work (define new endpoints and interact with MongoDB).

The skeleton has already integrated the **passport.js** module.

Test runner configuration.

Since we have a single-page app, don't expect too much here.

So... what can I find in this skeleton?



With **angular.js**, it is a good practice to have a **folder** for every "**portion**" of the application.

In each of these folders, you will find:
view components (e.g. jade and stylus)
angular controller components (main.controller.js)
angular configuration scripts (main.js)
files for automated tests (main.controller.spec.js

In the components folder, you will find elements that are reused **across the application**, such as:
angular services (and supporting files)
angular controllers that are used in headers/footers

This is the starting point for the single-page application. It loads the scripts and bootsraps angular.js. Important: it is **re-generated** at each build (dependencies)

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

3. Baby steps with Angular.js

AngularJS — Superheroic J

https://angularjs.org

ANGULARJS | Home | Learn | Develop | Discuss | Search

ANGULARJS by Google

HTML enhanced for web apps!

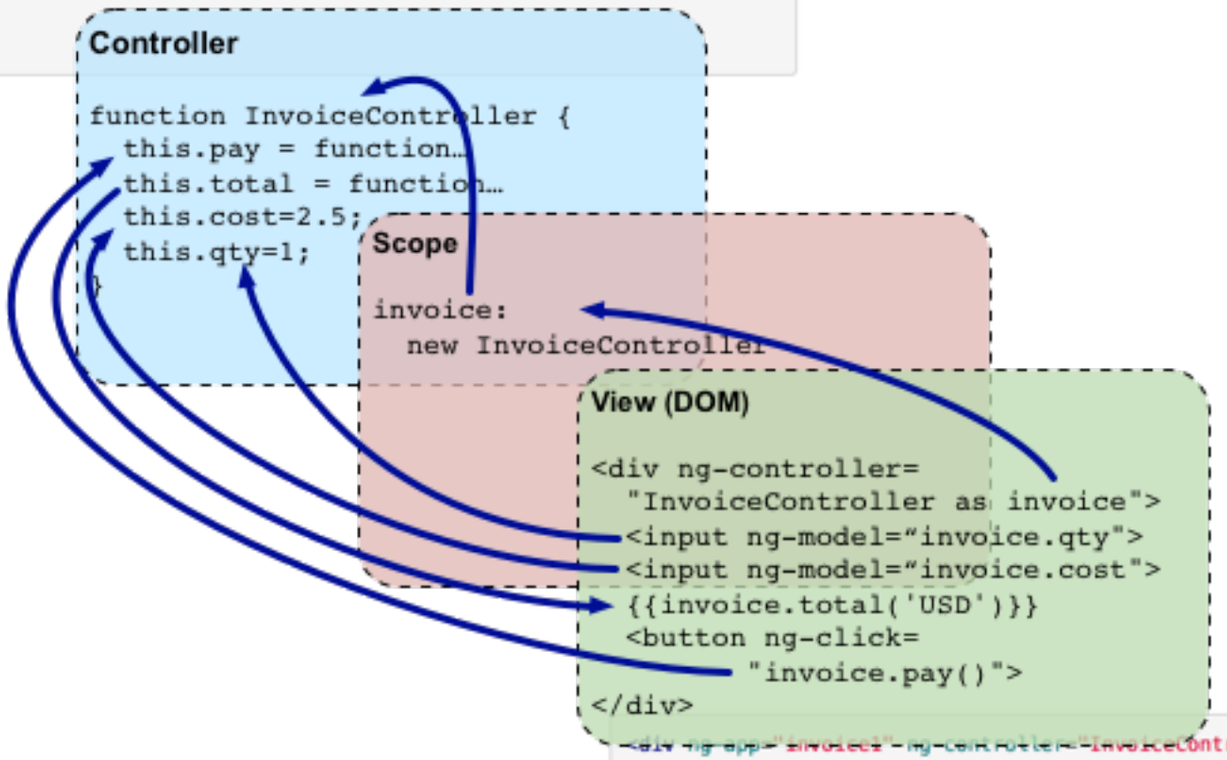
View on GitHub

Download (1.2.26 / 1.3.0-rc.5)

Follow +AngularJS on | +1 +88472 | Follow @angularjs 49.2K followers | Tweet 4,933

Learn Angular in your browser for free!

```
angular.module('invoice1', [])
  .controller('InvoiceController', function() {
    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!");
    };
  });
```



Invoice:
Quantity: 12
Costs: 34 EUR
Total: USD551.35 EUR408.00 CNY3,357.73 Pay

```
<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>
```

http://embed.plnkr.co/AM7KtWcwpM3Ej3ykZl69/preview
https://docs.angularjs.org/guide/concepts

A first look at Angular.js... The navigation bar.

The **controller** (navbar.controller.js)

```
'use strict';

angular.module('demoApp')
  .controller('NavbarCtrl', function ($scope, $location, Auth) {
    $scope.menu = [{
      'title': 'Home',
      'link': '/'
    }];

    $scope.isCollapsed = true;
    $scope.isLoggedIn = Auth.isLoggedIn;
    $scope.isAdmin = Auth.isAdmin;
    $scope.getCurrentUser = Auth.getCurrentUser;

    $scope.logout = function() {
      Auth.logout();
      $location.path('/login');
    };

    $scope.isActive = function(route) {
      return route === $location.path();
    };
  });
```

The **view** (navbar.jade)

```
div.navbar.navbar-default.navbar-static-top(ng-controller='NavbarCtrl')
  div.container
    div.navbar-header
      button.navbar-toggle(type='button', ng-click='isCollapsed = !isCollapsed')
        span.sr-only Toggle navigation
        span.icon-bar
        span.icon-bar
        span.icon-bar
      a.navbar-brand(href='/') demo

  div#navbar-main.navbar-collapse.collapse(collapse='isCollapsed')
    ul.nav.navbar-nav
      li(ng-repeat='item in menu', ng-class='{active: isActive(item.link)}')
        a(ng-href='{{item.link}}') {{item.title}}

      li(ng-show='isAdmin()', ng-class='{active: isActive("/admin")}')
        a(href='/admin') Admin

    ul.nav.navbar-nav.navbar-right
      li(ng-hide='isLoggedIn()', ng-class='{active: isActive("/signup")}')
        a(href='/signup') Sign up

      li(ng-hide='isLoggedIn()', ng-class='{active: isActive("/login")}')
        a(href='/login') Login

      li(ng-show='isLoggedIn()')
        p.navbar-text Hello {{ getCurrentUser().name }}

      li(ng-show='isLoggedIn()', ng-class='{active: isActive("/settings")}')
        a(href='/settings')
          span.glyphicon.glyphicon-cog

      li(ng-show='isLoggedIn()', ng-class='{active: isActive("/logout")}')
        a(href='', ng-click='logout()') Logout
```

With **$scope**, we can expose data and functions to our view.

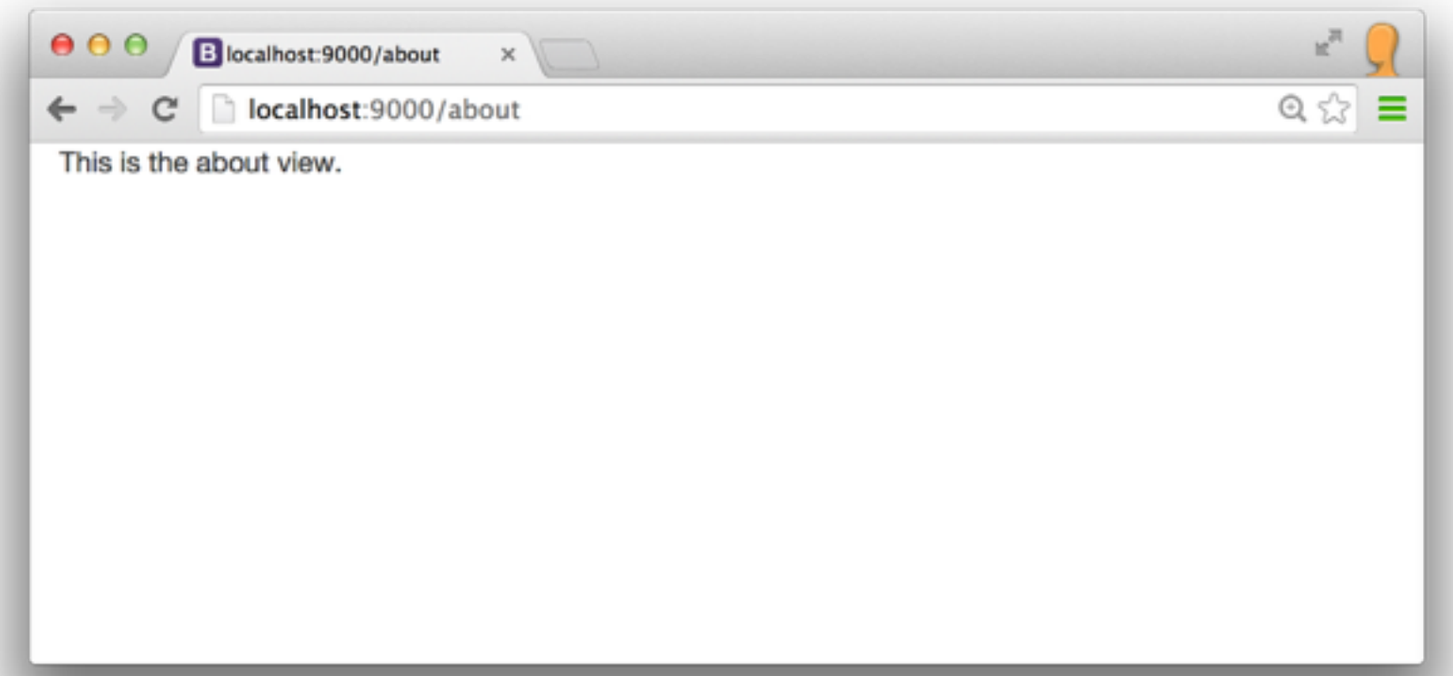
In the view, we can use **directives** (e.g. ng-repeat) and **variables** (e.g. {{item.link}}) to refer to the scope.

The binding is two-ways.

Ok... let's add our own page

```
$ yo angular-fullstack:route about
? Where would you like to create this route? client/app/
? What will the url of your route be? /about
   create client/app/about/about.js
   create client/app/about/about.controller.js
   create client/app/about/about.controller.spec.js
   create client/app/about/about.jade
   create client/app/about/about.styl
```

```
$ grunt serve
```



Ok... let's add our own "about" page

```
$ yo angular-fullstack:route about
? Where would you like to create this route? client/app/
? What will the url of your route be? /about
   create client/app/about/about.js
   create client/app/about/about.controller.js
   create client/app/about/about.controller.spec.js
   create client/app/about/about.jade
   create client/app/about/about.styl
```

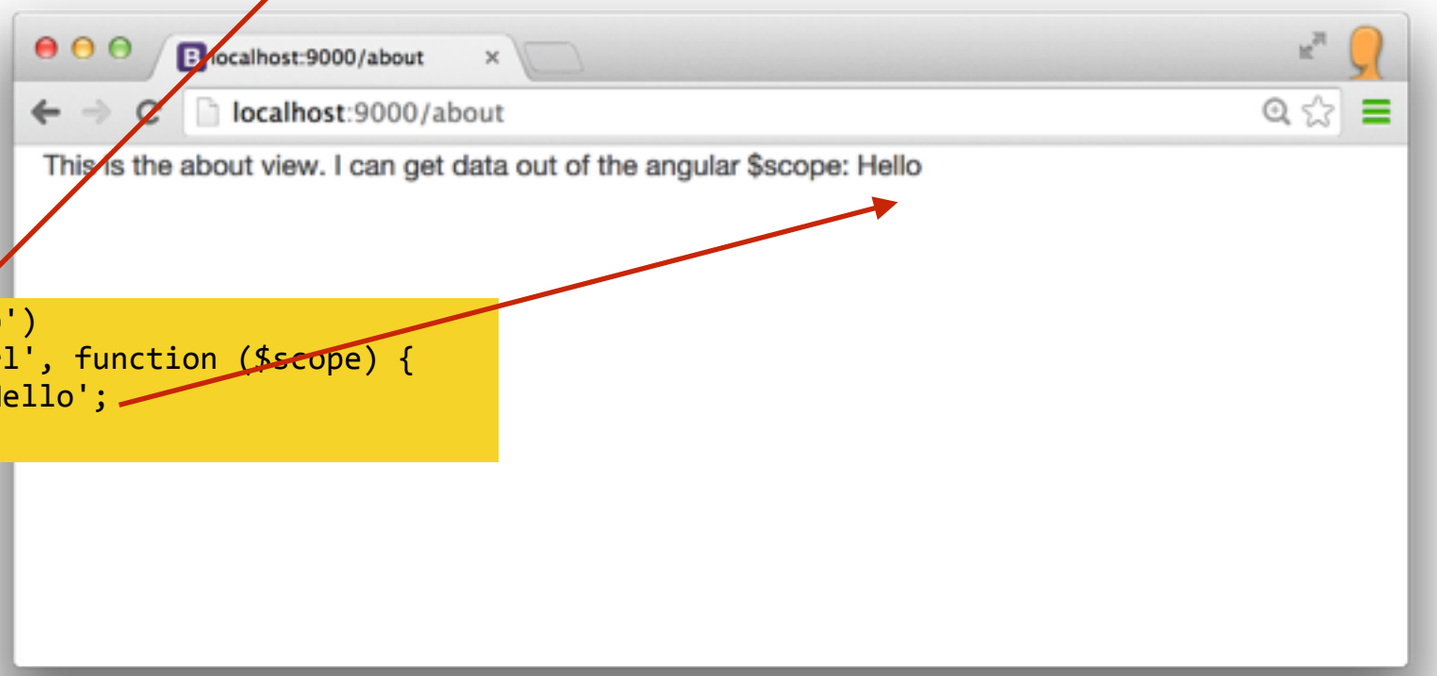
```
$ grunt serve
```


```
angular.module('demoApp')
  .controller('AboutCtrl', function ($scope) {
    $scope.message = 'Hello';
  });
```

```
.col-md-12
    | This is the about view.
```

Let's update the **about.jade** file (and observe **live reload**)

```
.col-md-12
    | This is the about view. I can get data
    | out of the angular $scope: {{message}}
```

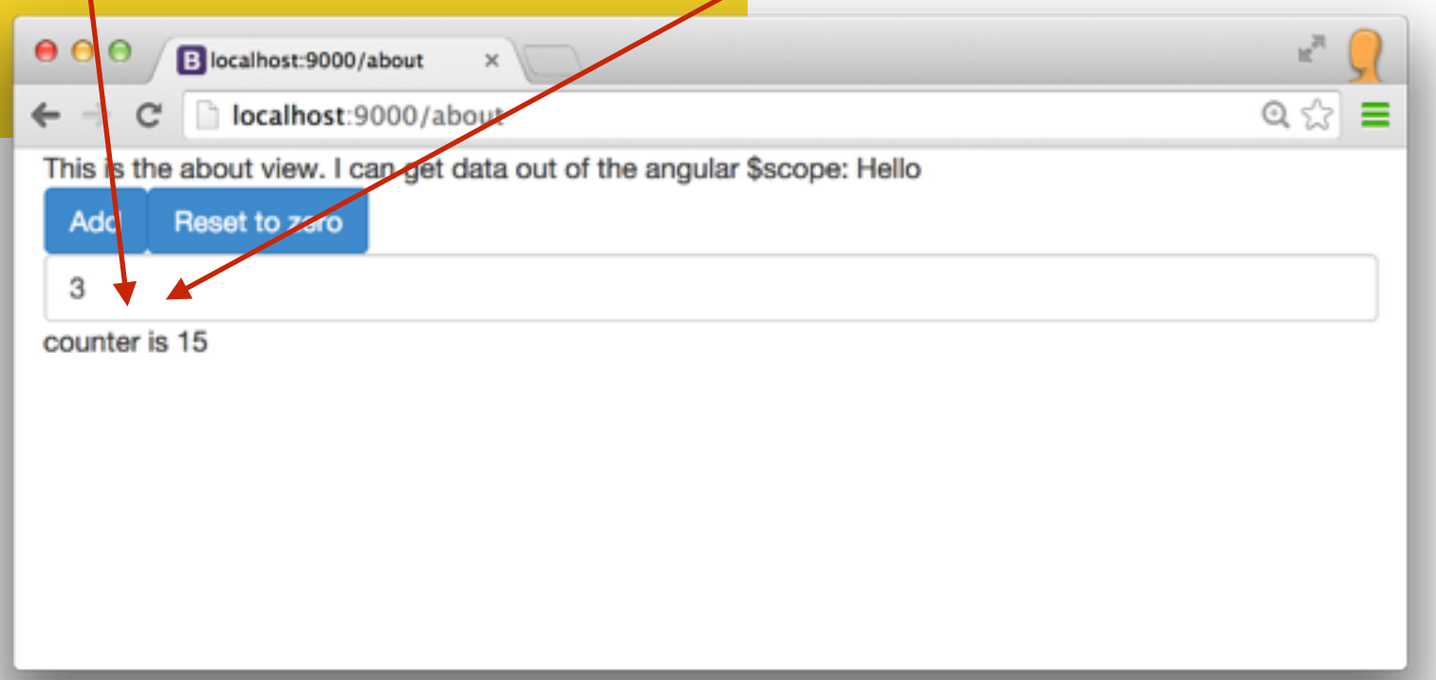



```
angular.module('demoApp')
  .controller('AboutCtrl', function ($scope) {
    $scope.message = 'Hello';
  });
```

Let's play with **two-ways binding**...

```
'use strict';

angular.module('demoApp')
  .controller('AboutCtrl', function ($scope) {
    $scope.message = 'Hello';

    $scope.counter = 0;
    $scope.incParam = 1;
    $scope.increment = function() {
      $scope.counter = $scope.counter+$scope.incParam;
    };
    $scope.reset = function() {
      $scope.counter = 0;
    };
  });
```

```
.col-md-12
    | This is the about view. I can get data
    | out of the angular $scope: {{message}}
    .p
      .btn.btn-primary(ng-click='increment()') Add
      .btn.btn-primary(ng-click='reset()') Reset to zero
      input.form-control(ng-model='incParam', type='number')
    .p
      The counter is {{counter}}
```

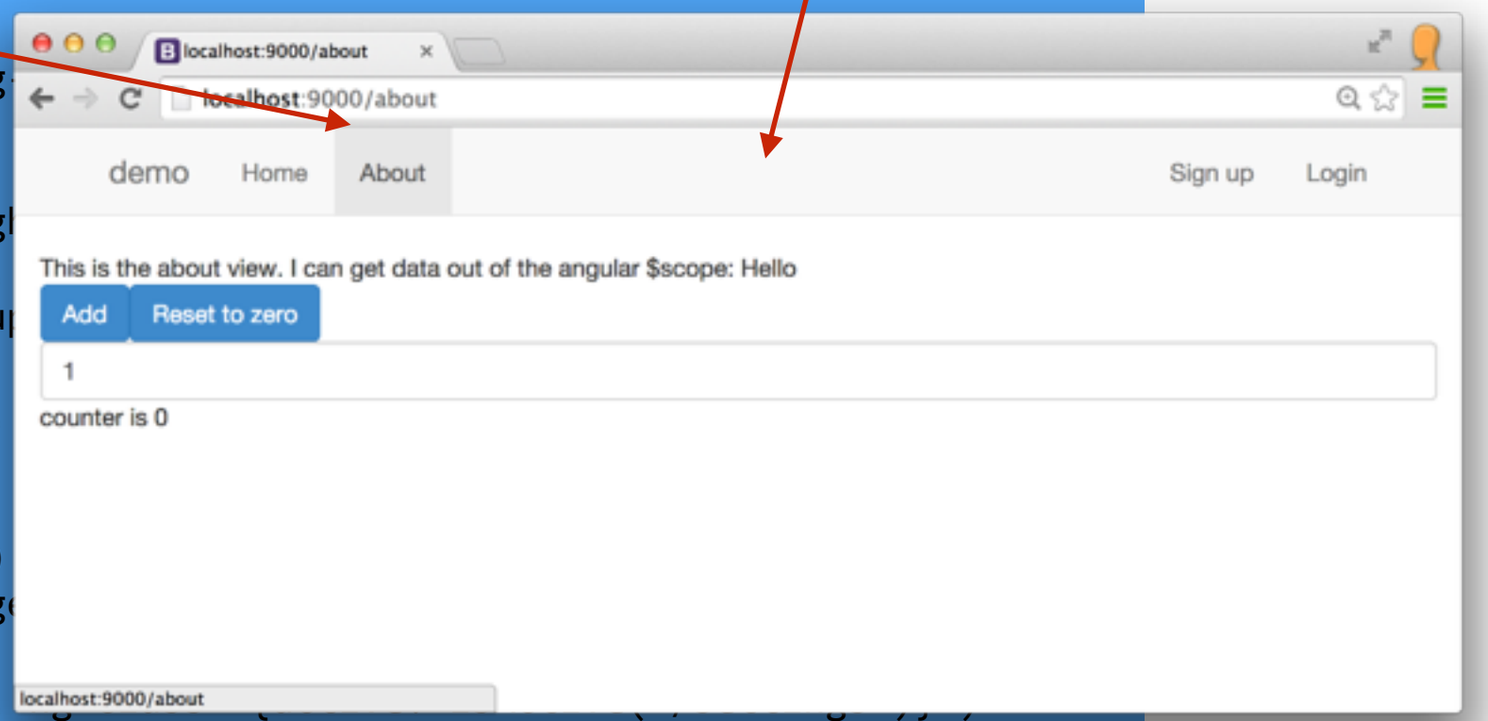


And let's integrate our page in the **site layout** (**navigation bar**)

```
div.navbar.navbar-default.navbar-static-top(ng-controller='NavbarCtrl')
  div.container
    div.navbar-header
      button.navbar-toggle(type='button', ng-click=
        span.sr-only Toggle navigation
        span.icon-bar
        span.icon-bar
        span.icon-bar
      a.navbar-brand(href='/') demo

    div#navbar-main.navbar-collapse.collapse(collaps
      ul.nav.navbar-nav
        li(ng-repeat='item in menu', ng-class='{active: isActive(item.link)}')
          a(ng-href='{{item.link}}') {{item.title}}

        li(ng-class='{active: isActive("/about")}')
          a(href='/about') About

        li(ng-show='isAdmin()', ng-
          a(href='/admin') Admin

      ul.nav.navbar-nav.navbar-righ
        li(ng-hide='isLoggedIn()',
          a(href='/signup') Sign up

        li(ng-hide='isLoggedIn()',
          a(href='/login') Login

        li(ng-show='isLoggedIn()')
          p.navbar-text Hello {{ ge

        li(ng-show='isLoggedIn()',
          a(href='/settings')
```

```
div(ng-include='"components/navbar/navbar.html"')

.col-md-12
    | This is the about view. I can get data
    | out of the angular $scope: {{message}}
    .p
      .btn.btn-primary(ng-click='increment()') Add
      .btn.btn-primary(ng-click='reset()') Reset to zero
      input.form-control(ng-model='incParam', type='number')
    .p
      The counter is {{counter}}
```

client/components/navbar/navbar.jade



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

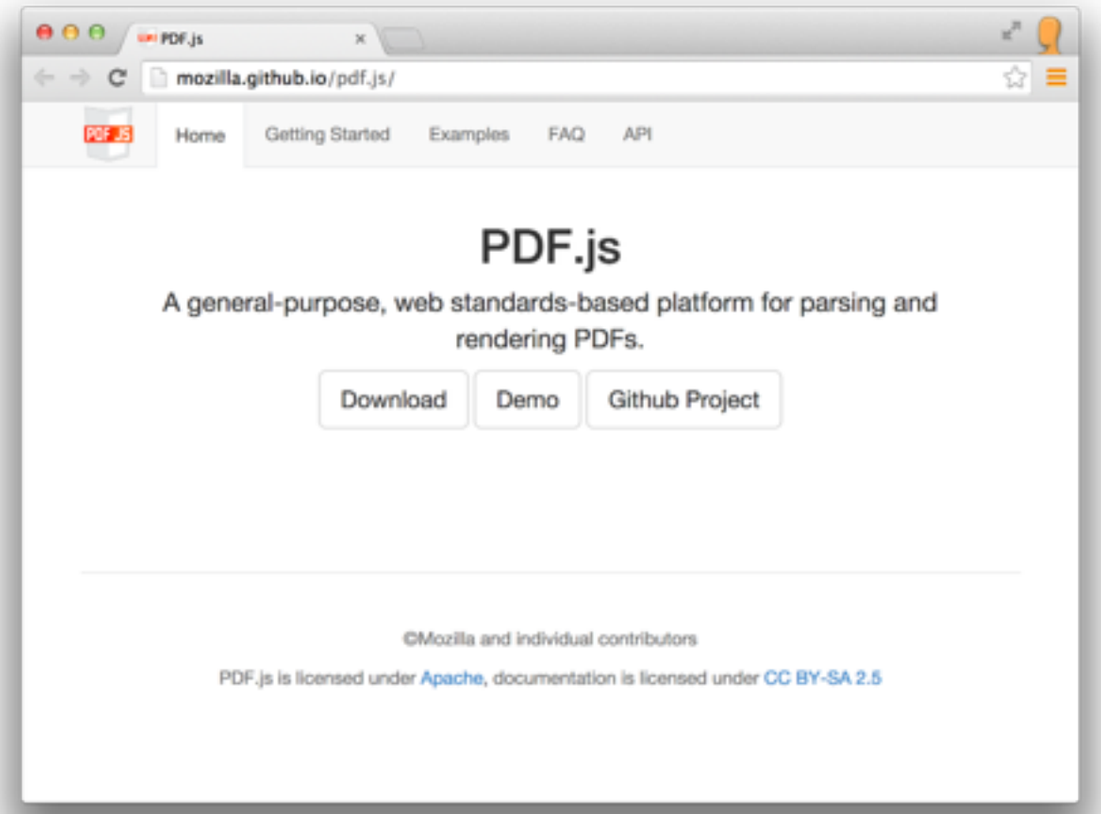
4. Display and control PDF

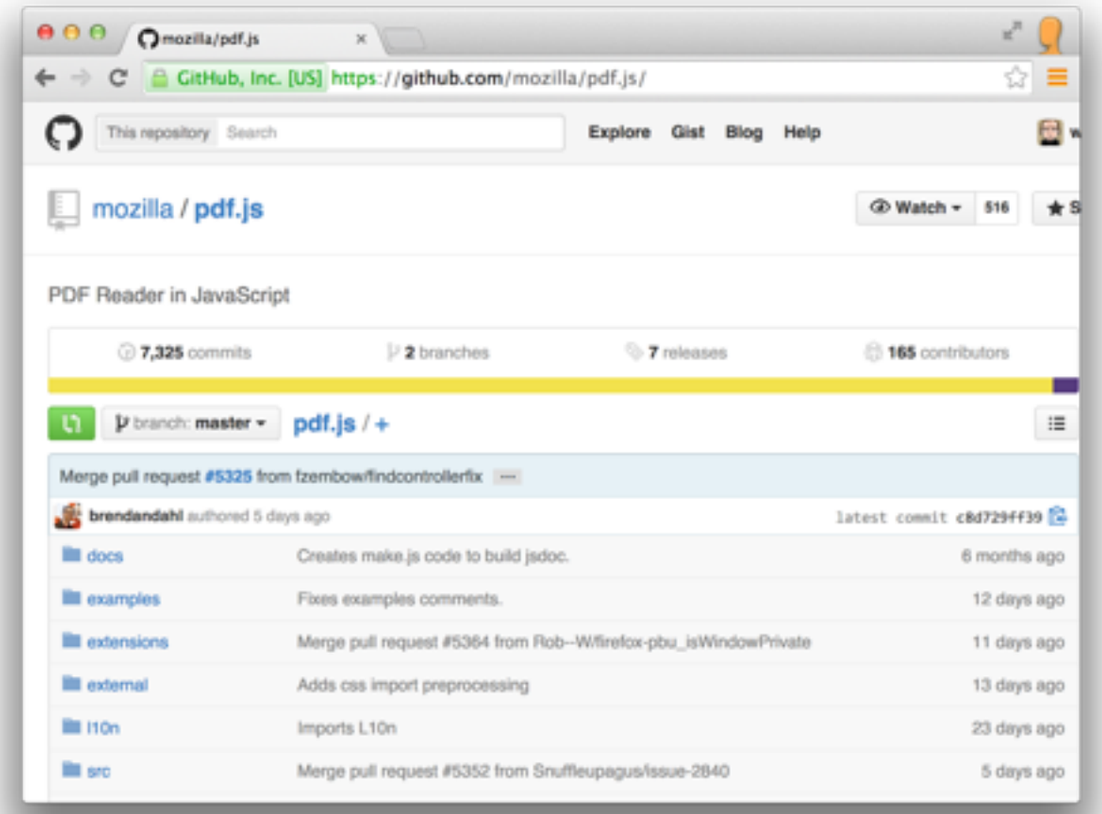
Using **PDF** in the project

- **From the project specifications, we know that:**
  - **PDF seems like a practical format**. It is easy to generate (from Keynote, Powerpoint, etc.). Probably a better choice than requiring the presenters to generate HTML content.
  - We want **display slides within a web page** (so that we can display the chat area next to it).
  - We want the presenter to be able to **control the slideshow**. At the minimum, he should be able to **move to the previous/next slide**. Being able to jump to any particular slide would be nice.
- **Questions:**
  - How can we embed a PDF document in a Web page?
  - How can we control the PDF viewer from JavaScript?

Using **PDF** in the project (1)

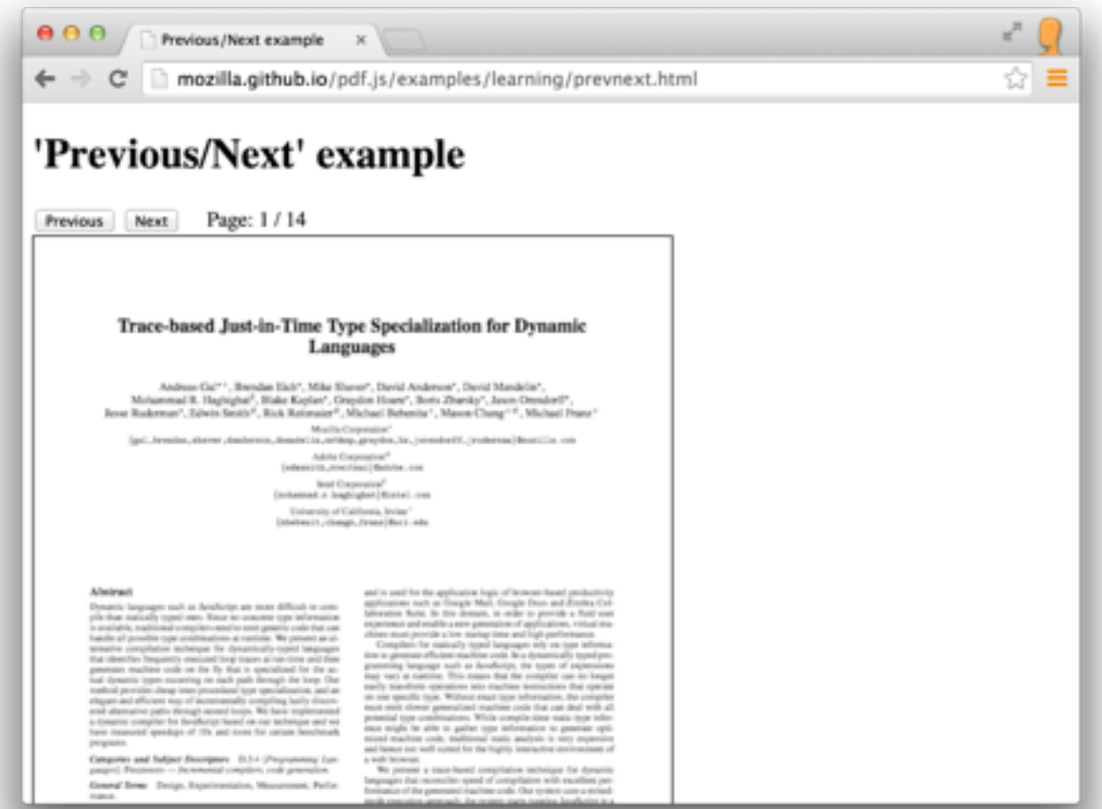
- **Through an investigation process, we have learned that:**
  - There is a **Mozilla project called “PDF.js”** that seems very promising. It is advertised as a “platform for parsing and rendering PDFs”.
  - They have a **Github repo**





Using **PDF** in the project (2)

- **Through an investigation process, we have learned that:**
  - There is an example that shows how to embed a borderless viewer and how to control it from Javascript. Less than 100 LOC.

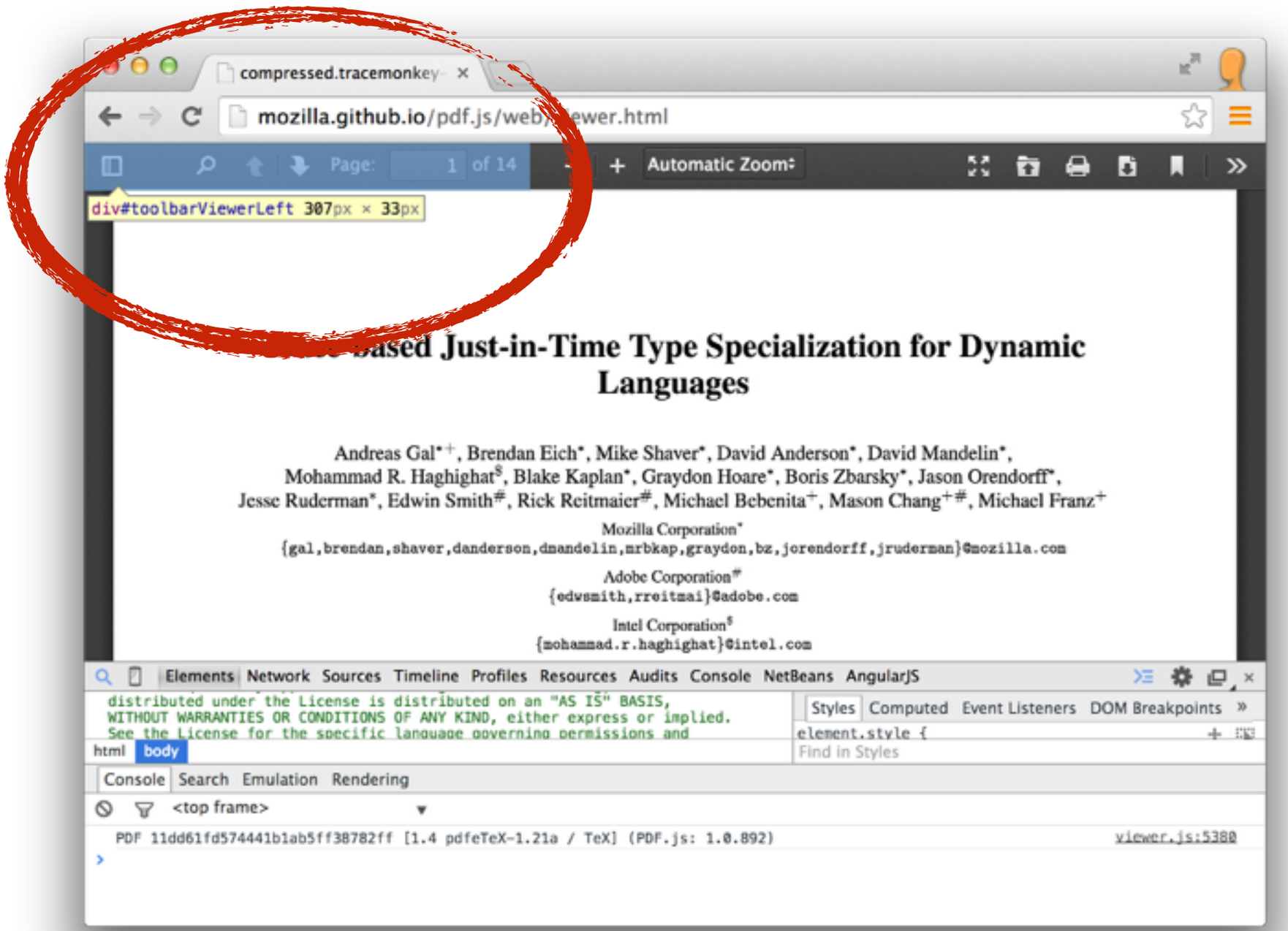



```
function renderPage(num) {
  pageRendering = true;
  // Using promise to fetch the page
  pdfDoc.getPage(num).then(function(page) {
    var viewport = page.getViewport(scale);
    canvas.height = viewport.height;
    canvas.width = viewport.width;

    // Render PDF page into canvas context
    var renderContext = {
      canvasContext: ctx,
      viewport: viewport
    };
    var renderTask = page.render(renderContext);
    // Wait for rendering to finish
    renderTask.promise.then(function () {
      pageRendering = false;
      if (pageNumPending !== null) {
        // New page rendering is pending
        renderPage(pageNumPending);
        pageNumPending = null;
      }
    });
  });
```

```
/**
 * Asynchronously downloads PDF.
 */
PDFJS.getDocument(url).then(function (pdfDoc_) {
  pdfDoc = pdfDoc_;
  document.getElementById('page_count').textContent = pdfDoc.numPages;

  // Initial/first page rendering
  renderPage(pageNum);
});
```

Using **PDF** in the project

Were you aware that when you open a PDF in Firefox,
it actually rendered by a JavaScript component?



Experiment with PDF.js

Start by experimenting with the framework in a **standalone project.** You will integrate PDF.js into the project later, when you have understood how it works.

A good starting point is the **Github repo**. There is a directory that contains examples:
https://github.com/mozilla/pdf.js/tree/master/examples

The **simple example with the previous / next buttons** is here:
https://github.com/mozilla/pdf.js/blob/master/examples/learning/prevnext.html

MUST READ for the Tests

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

- **Understanding the Node.js Event Loop**
  - http://strongloop.com/strongblog/node-js-event-loop/
- **Mixu's Node book: What is Node.js? (chapter 2)**
  - http://book.mixu.net/node/ch2.html
- **Node.js Explained, video**
  - http://kunkle.org/talks/

How do I **render PDF** in my Web app?

- **How do I use the full height?**

  http://webdesign.about.com/od/csstutorials/f/set-css-height-100-percent.htm

- **How do I get rid of “gutters” (space between cols)**

  http://stackoverflow.com/questions/18202240/bootstrap-3-grid-with-no-gap

- **How do I scroll to the end of my chat div?**

  http://plnkr.co/edit/wxTyp7PpyxJOHSlUumVC?p=preview
  https://github.com/Luegg/angularjs-scroll-glue