REST APIs with Javascript

Olivier Liechti & Laurent Prévost
COMEM Web Services

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

Use this to create and build the project

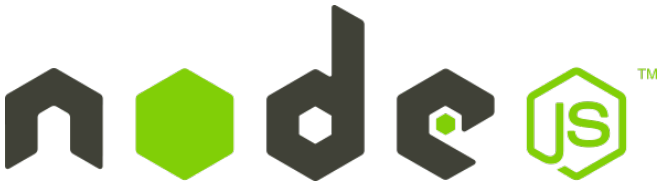
Use this to implement `/api/endpoints`

express
web application framework for node

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

mongoose

Use this interact with mongoDB



Use this because it's cool

~= maven

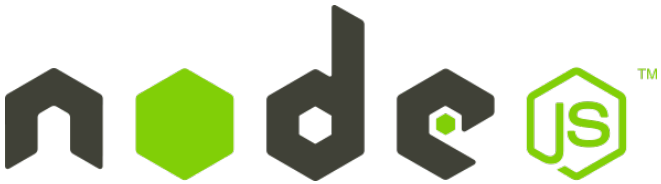
~= JAX-RS

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

express
web application
framework for
node

mongoose

~= Spring Data



~= Java EE

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

JavaScript 101

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

Douglas Crockford: JavaScript: The Good Parts

https://www.youtube.com/watch?v=_DKkVvOt6dk

JavaScript is built on some very good ideas and a few very bad ones.

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

JavaScript is an important language because it is the language of the web browser. Its association with the browser makes it one of the most popular programming languages in the world. **At the same time, it is one of the most despised programming languages in the world**. […]

Most people in that situation **don't even bother to learn JavaScript first**, and then they are surprised when JavaScript turns out to have significant differences from the some other language they would rather be using, and that those differences matter.

The amazing thing about JavaScript is that it is possible to get work done with it without knowing much about the language, or even knowing much about programming. It is a language with enormous expressive power. It is even better when you know what you're doing. **Programming is difficult business. It should never be undertaken in ignorance.**

Rule #1
JavaScript defines 6 types

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

```
var aNumber = 3.12;
var aBoolean = true;
var aString = "HEIG-VD";
var anObject = {
  aProperty: null
};

// t is true for all of these:
var t;
t = typeof aNumber === "number";
t = typeof aBoolean === "boolean";
t = typeof aString === "string";
t = typeof anObject === "object";
t = typeof anObject.aProperty ===
"object";
t = typeof anObject.foobar ===
"undefined";
```

http://codepen.io/wasadigi/full/tJFoy/

- The 6 types are:
  - number
  - boolean
  - string
  - object
  - undefined
  - null
- `null` is a type, but `typeof null === object`.
- JavaScript is a dynamic language: when you declare a variable, you don't specify a type (and the type can change over time).

Rule #2
There are 2 scopes for variables: the (evil) global scope and the function scope

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

```
var aVariableInGlobalScope;

function myFunction() {
  var aVariableInFunctionScope;
      anotherVariableInGlobalScope;
}

function myFunction2() {
  for (i=0; i<10; i++) {
    //i is in global scope!
  }
  for (var j=0; j<10; j++) {
    //j is in function scope!
  }
}
```

- A variable declared within a function is **not accessible** outside this function.
- Unless using **strict mode**, it is not mandatory to declare variables (beware of typos…)
- Two scripts loaded from the same HTML page share the same global scope (beware of **conflicts**…).
- There is **no block scope**.

http://codepen.io/wasadigi/full/sucyK

Rule #3
Objects are dynamic bags of properties

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

```
// let's create an object
var person = {
  firstName: 'olivier',
  lastName: 'liechti'
};

// we can dynamically add properties
person.gender = 'male';
person['zip'] = 1446;

// and delete them
delete person.zip;

for (var key in person) {
  console.log(key + " : " +
person[key]);
};
```

- There are different ways to **access properties** of an object.
- JavaScript is **dynamic**: it is possible to **add** and **remove** properties to an object at any time.
- Every object has a different list of properties (**no class**).

http://codepen.io/wasadigi/full/cmjGA

Rule #4
The language has no support for classes
There are 3 ways to create objects

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

```
// create an object with a literal
var person = {
  firstName: 'olivier',
  lastName: 'liechti'
};

// create an object with a prototype
var child = Object.create(person);

// create an object with a constructor
var child = new Person('olivier',
'liechti');
```

- **`class`** is a reserved word in JavaScript, but it is not used in the current version of the language (reserved for the future).
- A **constructor** is function like any other (uppercase is a coding convention).
- It is the use of the **`new`** keyword that triggers the object creation process.

Rule #5
Every object inherits from a prototype object

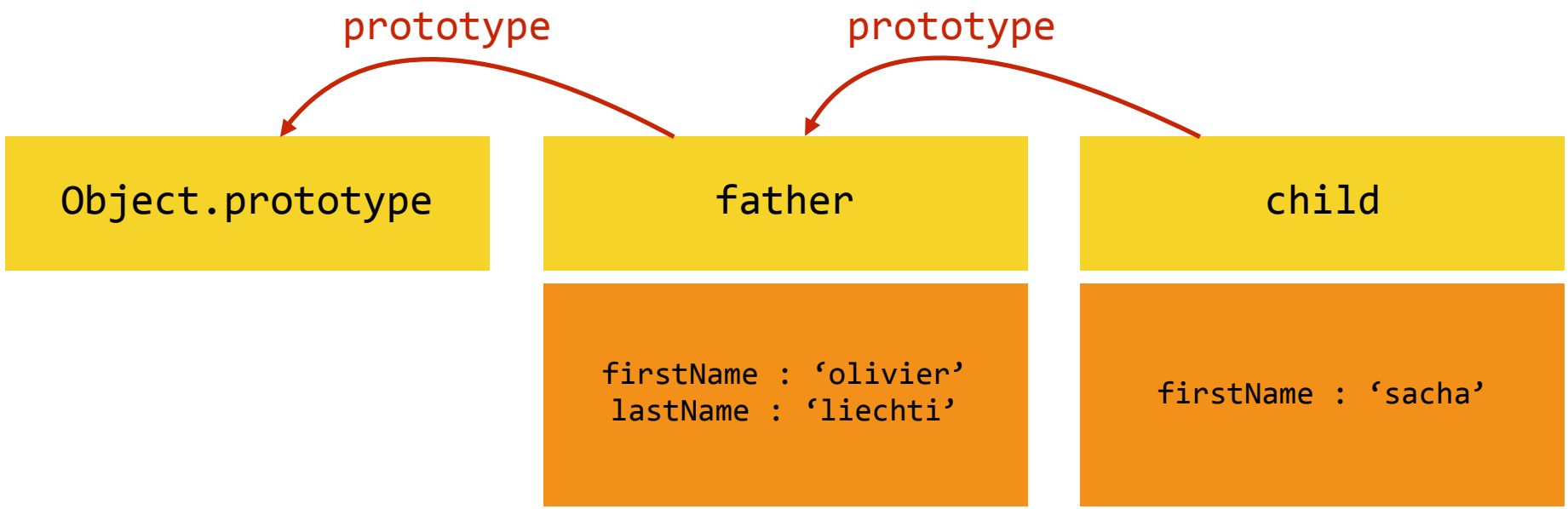
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
var person = {
   firstName: "olivier",
   lastName: "liechti"
};
// person's prototype is Object.prototype

var father = {};
var child = Object.create(father);
// child's prototype is father

function Person(fn, ln) {
  this.firstName = fn;
  this.lastName = ln;
}
var john = new Person("John", "Doe");
// john's prototype is Person.prototype
```

http://codepen.io/wasadigi/full/nklil

Rule #5
Every object inherits from a prototype object

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




```
console.log(child.lastName);
// prints 'liechti' on the
console
```

- Every object inherits from a prototype object. **It inherits and can override its properties**, including its methods.
- Objects created with object literals inherit from **Object.prototype**.
- When you access the property of an object, JavaScript **looks up the prototype chain** until it finds an ancestor that has a value for this property.

Rule #6
With patterns, it is possible to implement class-like data structures

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

```
function Person(fn, ln) {
  var privateVar;
  this.firstName = fn;
  this.lastName = ln;
  this.badGreet = function() {
    console.log("Hi " + this.firstName);
  };
};

Person.prototype.greet = function() {
  console.log("Hey " + this.firstName);
};

var p1 = new Person("olivier", "liechti");

p1.badGreet();
p1.greet();
```

http://codepen.io/wasadigi/full/FeiAd/

- **`badGreet`** is a property that will be replicated for every object created with the Person constructor:
  - poor memory management
  - not possible to alter behavior of all instances at once
- **`greet`** is a property that will be shared by all instances (because it will be looked up along the object inheritance chain).
- **`privateVar`** is not accessible outside of the constructor.
- **`fistName`** is publicly accessible (no encapsulation).

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

Node.js

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

"Node.js is a **platform** built on Chrome's JavaScript runtime for easily building **fast**, **scalable network applications**.

not only!

Node.js uses an **event-driven**, **non-blocking I/O model** that makes it lightweight and efficient, perfect for **data-intensive real-time applications that run across distributed devices**."

Google

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

Trends

Topics

Subscribe

Node.js
Software

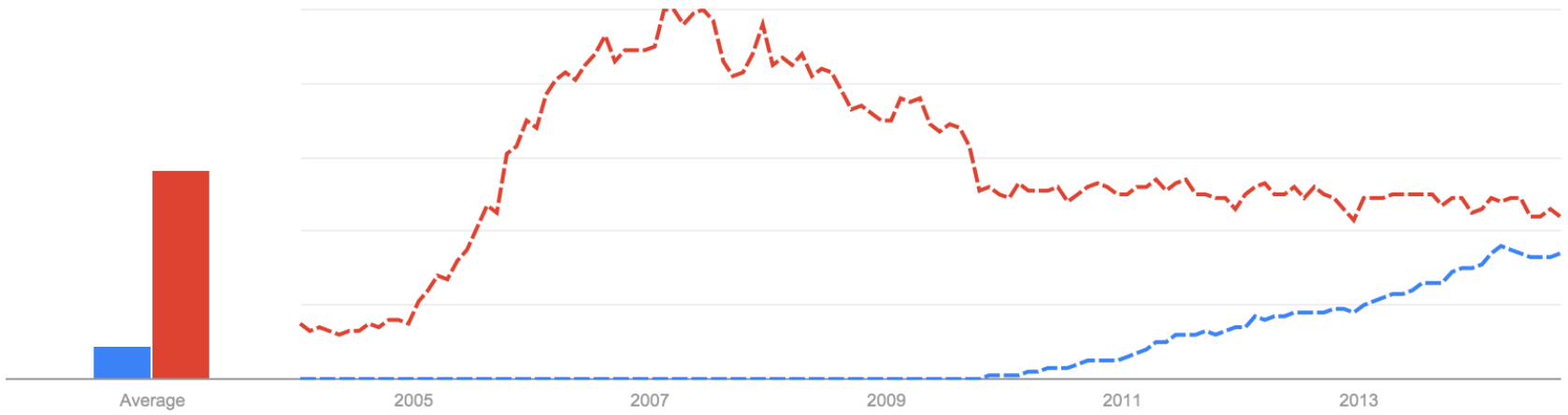
Ruby on Rails
Web Framework

+ Add term

Beta: Measuring search interest in *topics* is a beta feature which quickly provides accurate measurements of overall search interest. To measure search interest for a specific *query*, select the "search term" option.

Interest over time

News headlines Forecast




Job Trends from Indeed.com
node js
Percentage of Matching Job Postings
0.06
0.04
0.02
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14


Job Trends from Indeed.com
ruby on rails
Percentage of Matching Job Postings
0.15
0.1
0.05
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14

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

Single-threaded Asynchronous

"I am not leaving until you are done..."
vs
"Wake me up when you have result"

Let's look at a **first example**

```
var fs = require("fs");

/**
 * Simple function to test the synchronous readFileSync function provided by
 * Node.js
 * @param {string} filename - the name of the file we want to read in the test
 */
function testSyncRead(filename) {
  console.log("I am about to read a file: " + filename);
  var data = fs.readFileSync(filename);
  console.log("I have read " + data.length + " bytes (synchronously).");
  console.log("I am done.");
}

// We get the file name from the argument passed on the command line
var filename = process.argv[2];

console.log("\nTesting the synchronous call");
testSyncRead(filename);
```

We use a standard **Node module** for accessing the file system

fs.readFileSync is synchronous: it blocks the main thread until the data is available.

*Synchronous functions are easier to use, but they have **severe** performance implications!!*

Let's look at a **first example**

```
var fs = require("fs");

/**
 * Simple function to test the synchronous readFileSync function provided by
 * Node.js
 * @param {string} filename - the name of the file we want to read in the test
 */
function testSyncRead(filename) {
  console.log("I am about to read a file: " + filename);
  var data = fs.readFileSync(filename);
  console.log("I have read " + data.length + " bytes (synchronously).");
  console.log("I am done.");
}

// We get the file name from the argument passed on the command line
var filename = process.argv[2];

console.log("\nTesting the synchronous call");
testSyncRead(filename);
```

We use a standard **Node module** for accessing the file system

fs.readFileSync is synchronous: it blocks the main thread until the data is available.

```
$ node sample2.js medium.txt

Testing the synchronous call
I am about to read a file: medium.txt
I have read 1024 bytes (synchronously).
I am done.
```

*Synchronous functions are easier to use, but they have **severe** performance implications!!*

Let's look at a **second example**

```
var fs = require("fs");

/**
 * Simple function to test the asynchronous readFile function provided by Node.js
 * @param {string} filename - the name of the file we want to read in the test
 */
function testAsyncRead(filename) {
  console.log("I am about to read a file: " + filename);

  fs.readFile(filename, function (err, data) {
    console.log("Nodes just told me that I have read the file.");
  });

  console.log("I am done. Am I?");
}
// We get the file name from the argument passed on the command line
var filename = process.argv[2];

console.log("\nTesting the asynchronous call");
testAsyncRead(filename);
```

fs.readFile is asynchronous: it does not block the main thread until the data is available.

We must provide a **callback function**, which Node.js will invoke when the data is available.

Problems can happen when an (asynchronous) function is called.

*Node.js developers **have to** learn the asynchronous programming style.*

Let's look at a **second example**

```
var fs = require("fs");

/**
 * Simple function to test the asynchronous readFile function provided by Node.js
 * @param {string} filename - the name of the file we want to read in the test
 */
function testAsyncRead(filename) {
  console.log("I am about to read a file: " + filename);

  fs.readFile(filename, function (err, data) {
    console.log("Nodes just told me that I have read the file.");
  });

  console.log("I am done. Am I?");
}
// We get the file name from the argument passed on the command line
var filename = process.argv[2];

console.log("\nTesting the asynchronous call");
testAsyncRead(filename);
```

```
$ node sample2.js medium.txt

Testing the asynchronous call
I am about to read a file: medium.txt
I am done. Am I?
Nodes just told me that I have read the file.
```

fs.readFile is asynchronous: it does not block the main thread until the data is available.

We must provide a **callback function**, which Node.js will invoke when the data is available.

Problems can happen when an (asynchronous) function is called.

*Node.js developers **have to** learn the asynchronous programming style.*

Node.js v0.10.32 Manual & Documentation

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

- Assertion Testing
- Buffer
- C/C++ Addons
- Child Processes
- Cluster
- Console
- Crypto
- Debugger
- DNS
- Domain
- Events
- File System
- Globals
- HTTP
- HTTPS
- Modules
- Net
- OS
- Path
- Process
- Punycode
- Query Strings
- Readline
- REPL
- Stream
- String Decoder
- Timers
- TLS/SSL
- TTY
- UDP/Datagram
- URL
- Utilities
- VM
- ZLIB

Let's look at a **third example**

```
/*global require */

var http = require("http");

/**
 * This function starts a http daemon on port 9000. It also
 * registers a callback handler, to handle incoming HTTP
 * requests (a simple message is sent back to clients).
 */
function runHttpServer() {
  var daemon = http.createServer();

  daemon.on("request", function (req, res) {
    console.log("A request has arrived: URL=" + req.url);
    res.writeHead(200, {
      'Content-Type': 'text/plain'
    });
    res.end('Hello World\n');
  });

  console.log("Starting http daemon...");
  daemon.listen(9000);
}

runHttpServer();
```

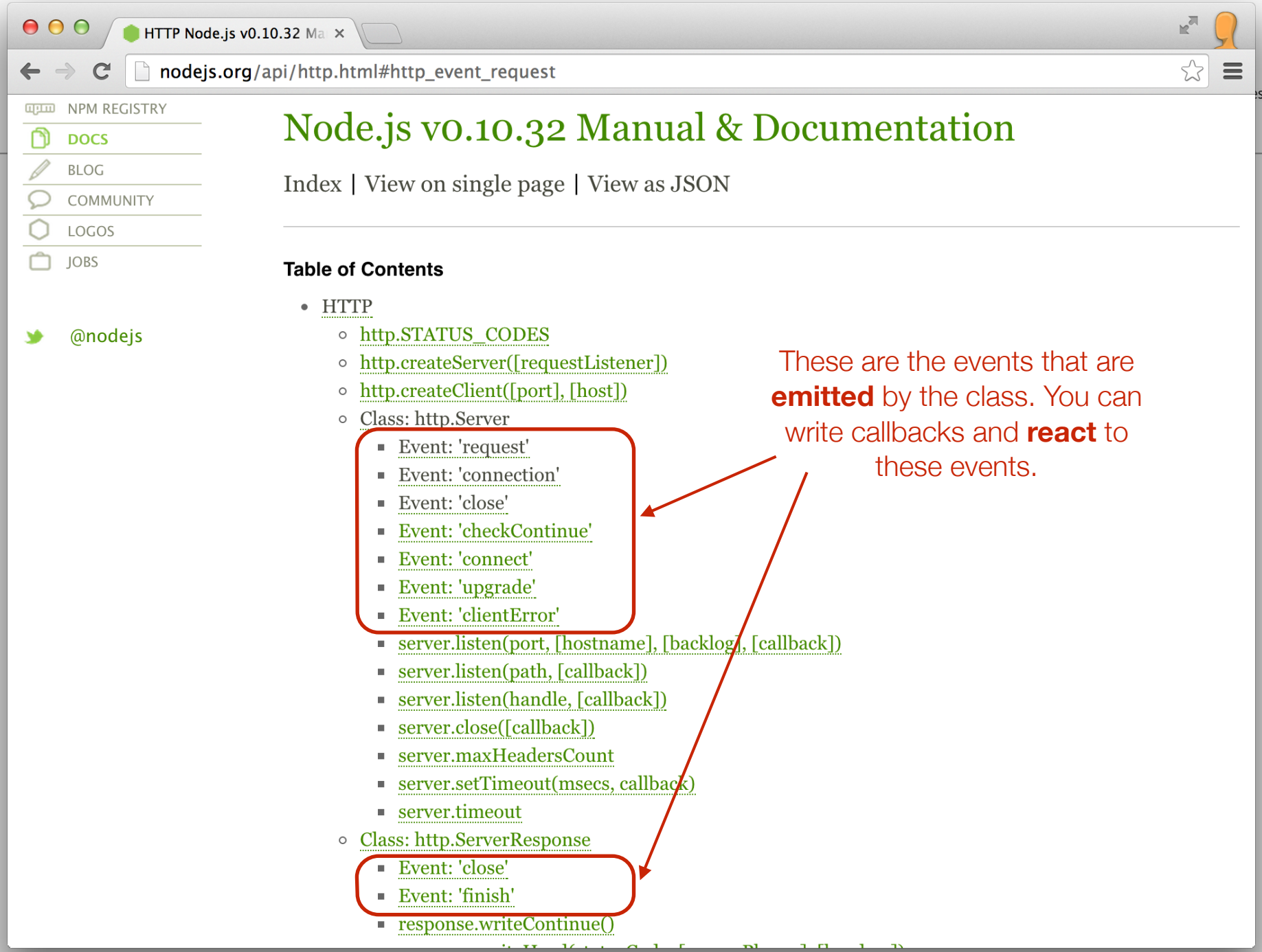
We use a standard **Node module** that takes care of the HTTP protocol.

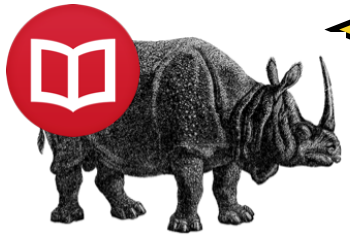
Node can provide us with a **ready-to-use** server.

We can attach **event handlers** to the server. Node will notify us asynchronously, and give us access to the request and response.

We can **send back** data to the client.

We have wired everything, let's **welcome** clients!


HTTP Node.js v0.10.32 Ma
nodejs.org/api/http.html#http_event_request
NPM REGISTRY
DOCS
BLOG
COMMUNITY
LOGOS
JOBS
@nodejs
Node.js v0.10.32 Manual & Documentation
Index | View on single page | View as JSON
Table of Contents
HTTP
http.STATUS_CODES
http.createServer([requestListener])
http.createClient([port], [host])
Class: http.Server
Event: 'request'
Event: 'connection'
Event: 'close'
Event: 'checkContinue'
Event: 'connect'
Event: 'upgrade'
Event: 'clientError'
server.listen(port, [hostname], [backlog], [callback])
server.listen(path, [callback])
server.listen(handle, [callback])
server.close([callback])
server.maxHeadersCount
server.setTimeout(msecs, callback)
server.timeout
Class: http.ServerResponse
Event: 'close'
Event: 'finish'
response.writeContinue()
These are the events that are emitted by the class. You can write callbacks and react to these events.



How does Node.js use an **event loop** to offer an asynchronous programming model?

```
on('request', function(req, res) { // my code});
on('data', function(data) { // my code});
```

Callback functions that **you** have written and registered

‘request’ **event**

‘request’ **event**

‘data’ **event**

‘request’ **event**

Queue of events that have been **emitted**

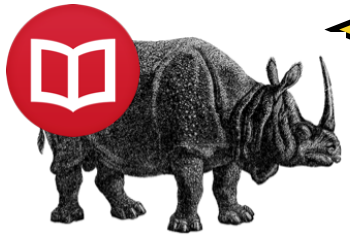
Event Loop

get the next event in the queue; invoke the registered callbacks in sequence; delegate I/O operations to the Node platform

All the code that **you** write runs on a **single thread**

The **long-running tasks** (I/Os) are executed by Node in parallel; Node emits events to report progress (which triggers your callbacks).

Another pattern is to provide a callback to node when invoking an asynchronous function.



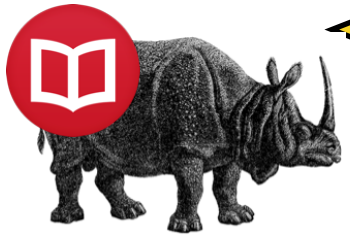
What is **npm**?

*”**npm** is the **package manager** for the Node JavaScript platform. It puts **modules** in place so that node can find them, and manages **dependency** conflicts intelligently.*

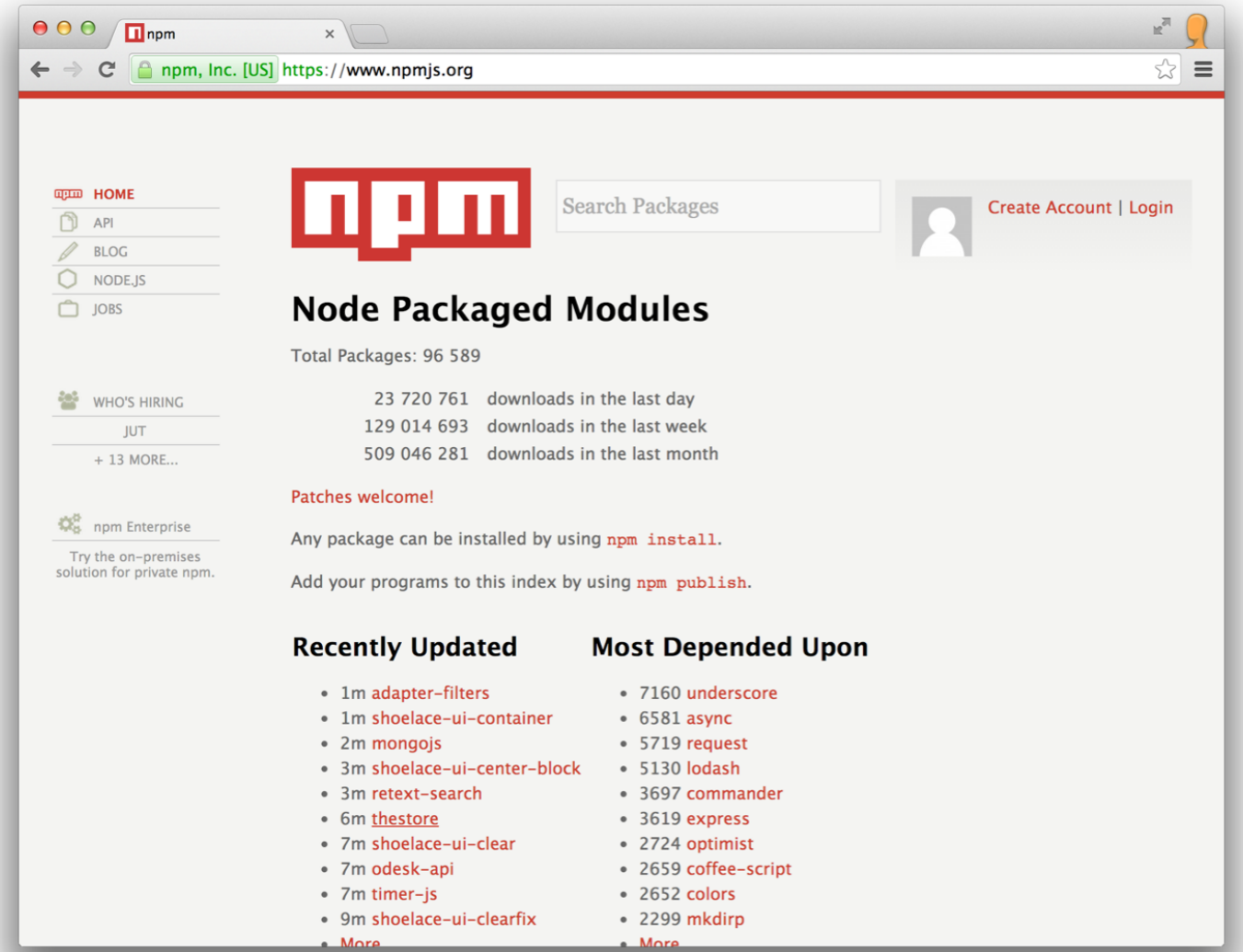
*It is extremely configurable to support a wide variety of use cases. Most commonly, it is used to **publish**, **discover**, **install**, and **develop** node programs.”*

https://www.npmjs.org/doc/cli/npm.html

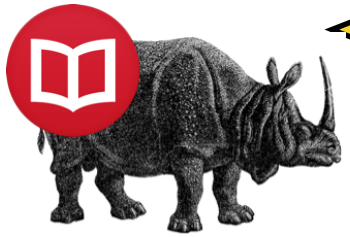
You **have to** read this:
https://www.npmjs.org/doc/misc/npm-faq.html



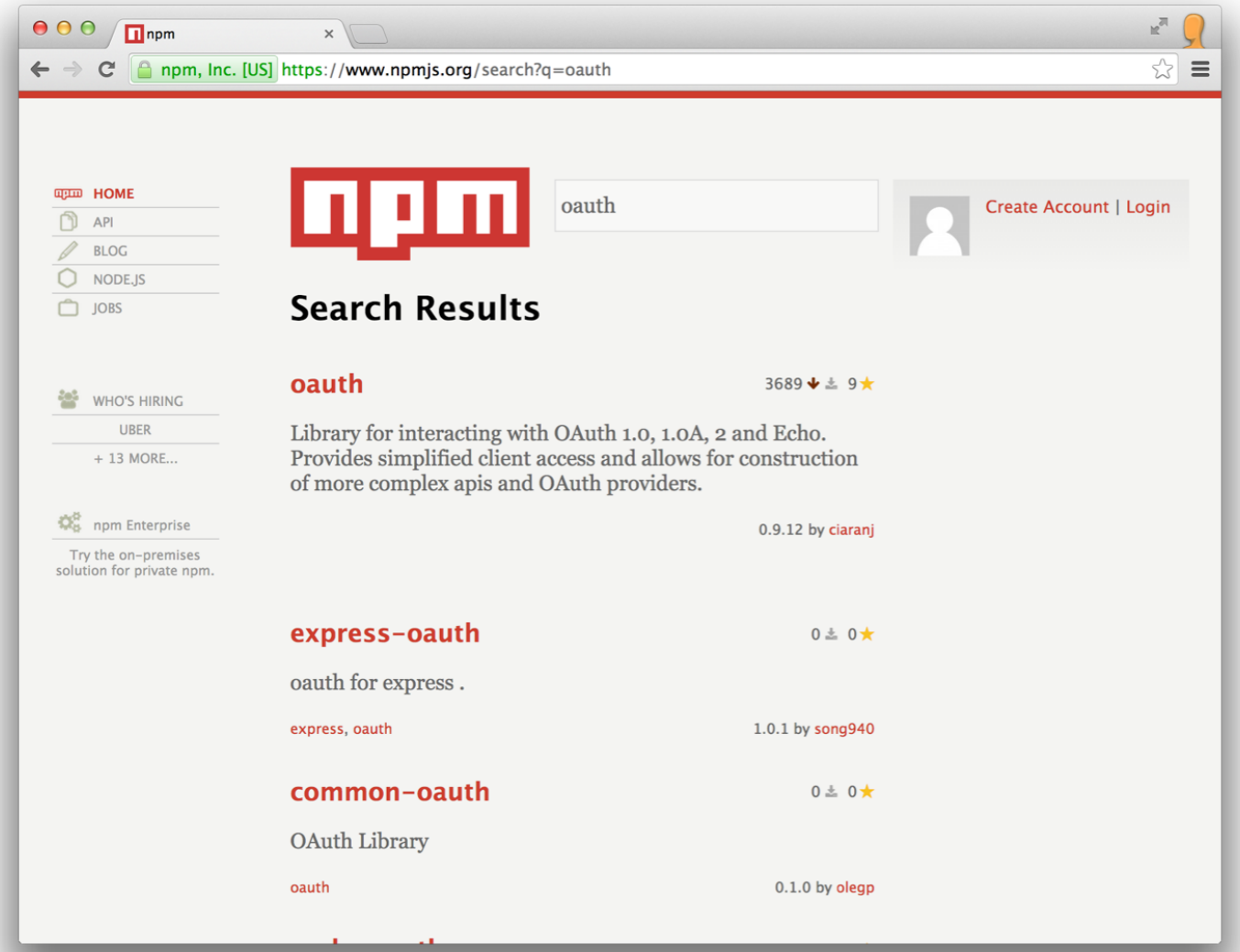
npm is a set of command line tools that work together with the **node registry**



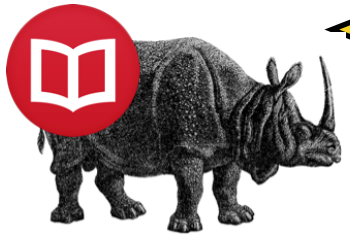
https://www.npmjs.org



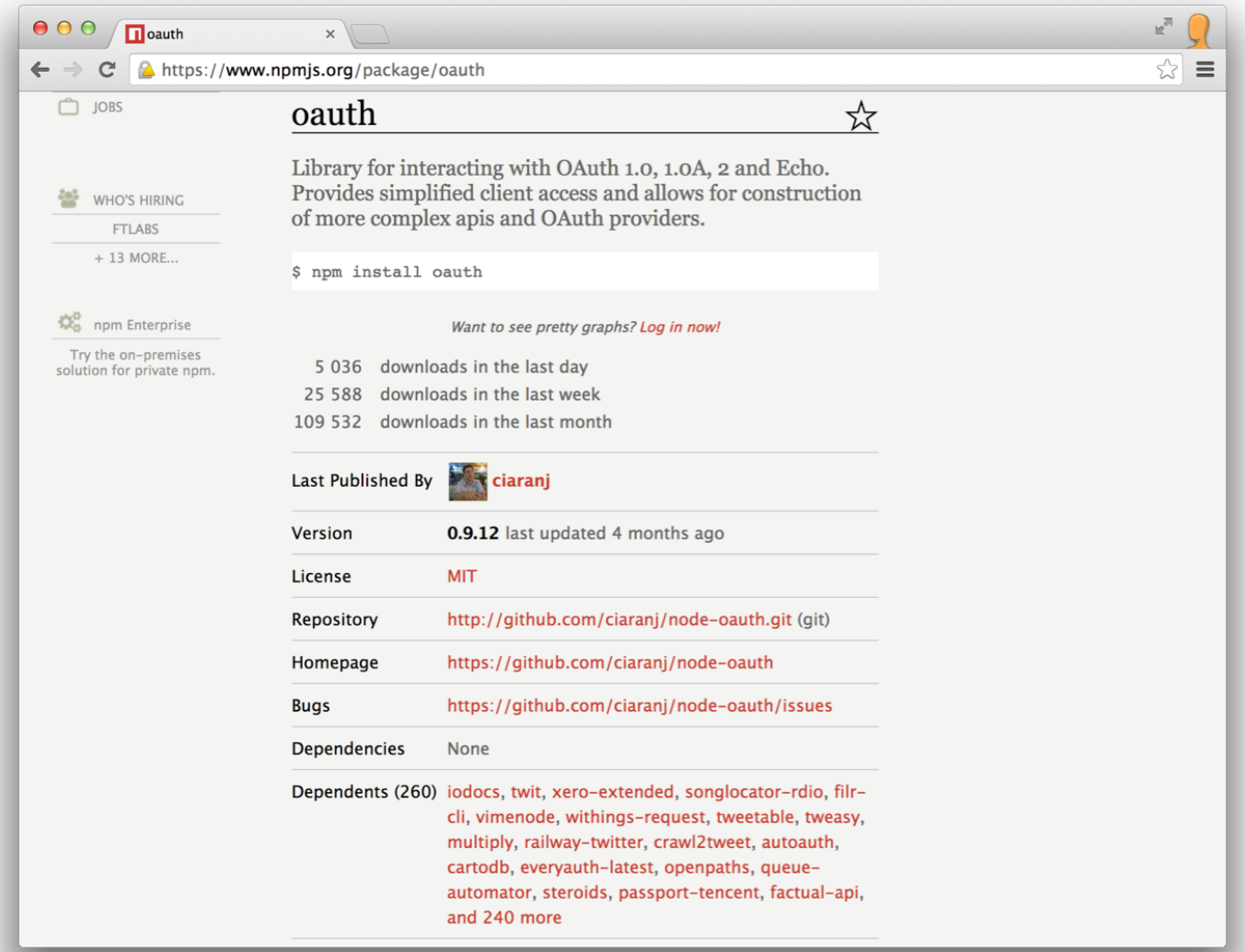
npm is a set of command line tools that work together with the **node registry**



https://www.npmjs.org



npm is a set of command line tools that work together with the **node registry**



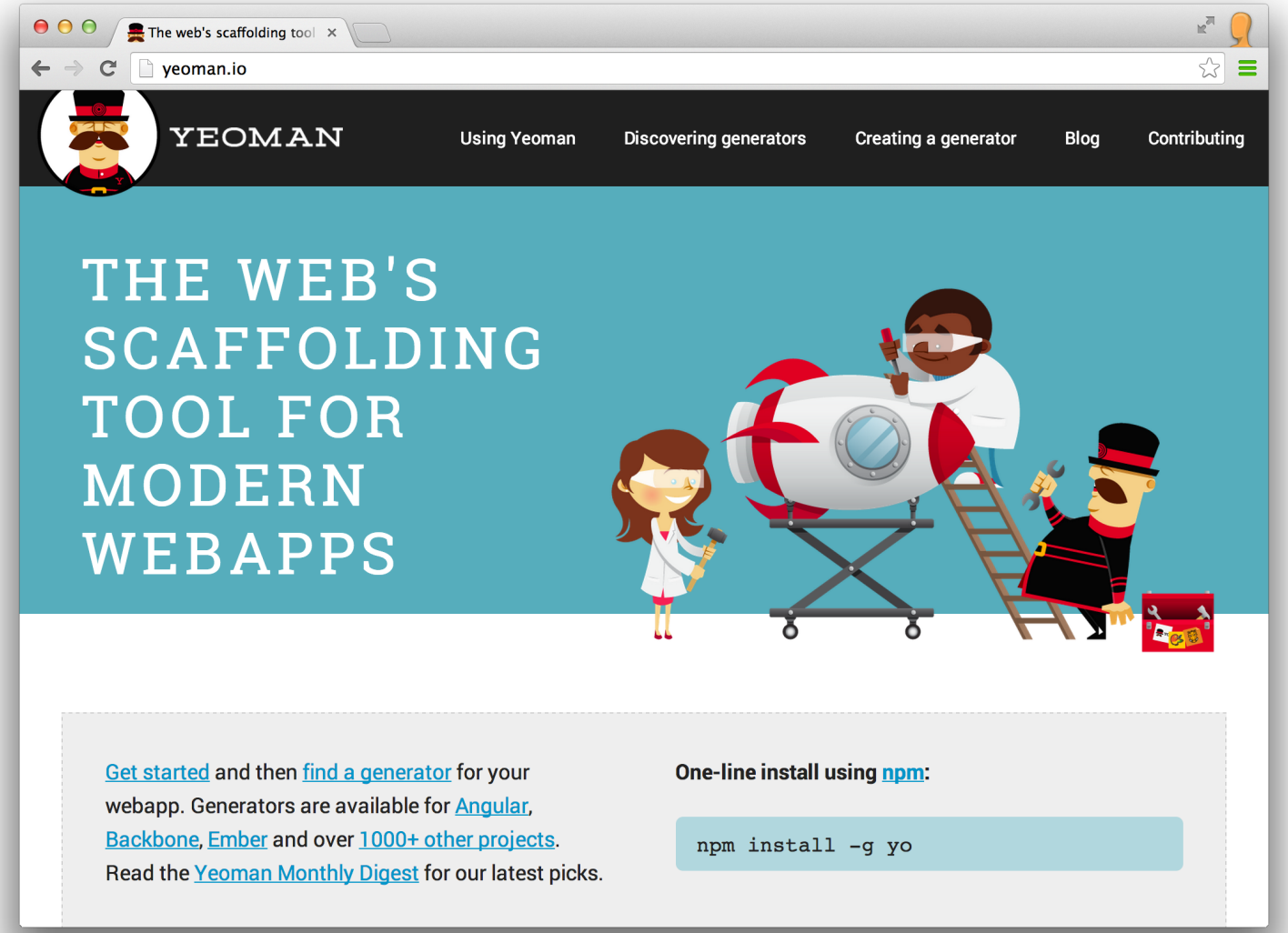
https://www.npmjs.org

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

Yeoman

What is Yeoman?

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



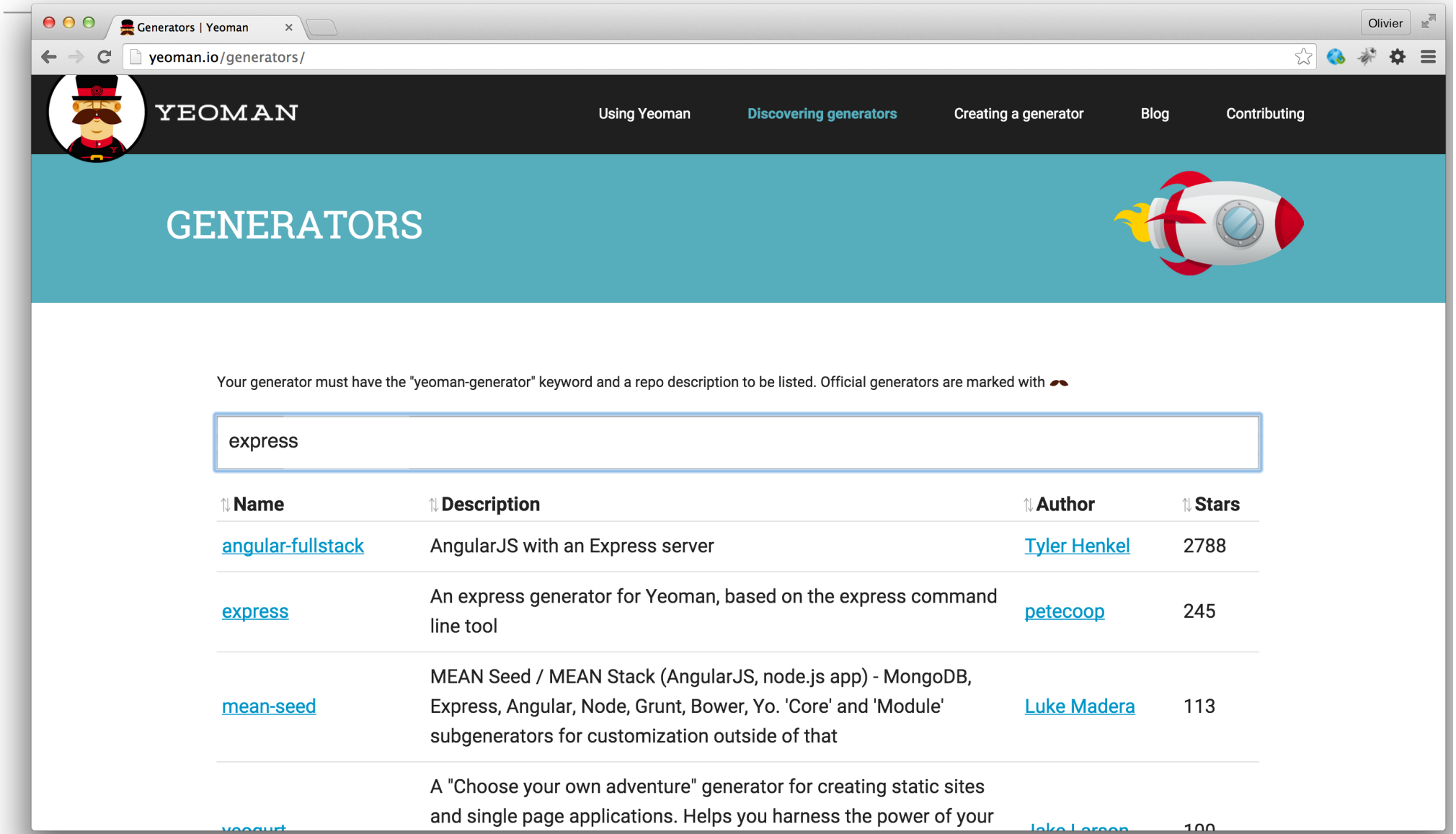
What is Yeoman?

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

- 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.

What is Yeoman?

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



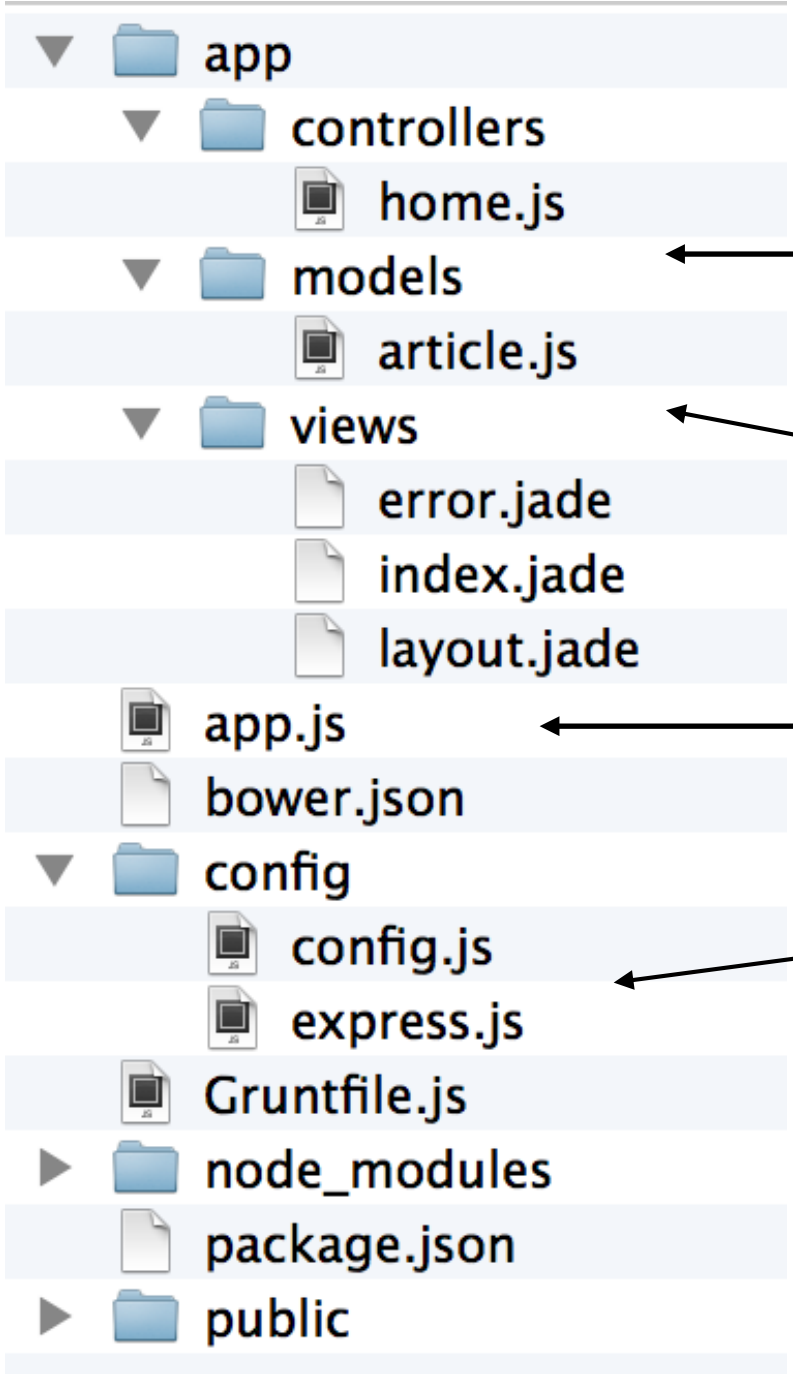
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

express
web application
framework for
node

Express.js

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

```
MacBook-Pro-de-admin:express admin$ yo express
? Select a version to install: MVC
? Select a view engine to use: Jade
? Select a database to use: MongoDB
   create .bowerrc
   create .gitignore
   create bower.json
   create Gruntfile.js
   create app.js
   create app/controllers/home.js
   create app/models/article.js
   create config/config.js
   create config/express.js
   create package.json
   create app/views/error.jade
   create app/views/index.jade
   create app/views/layout.jade


I'm all done. Running bower install & npm install for you to install
the required dependencies. If this fails, try running the command
yourself.
```



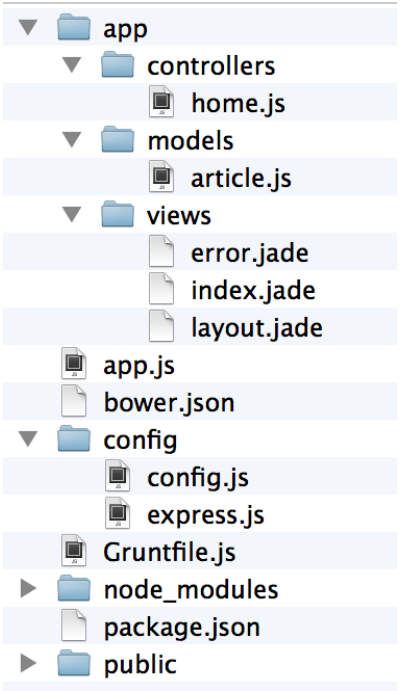
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

controller =~ resource

data access (mongoose)

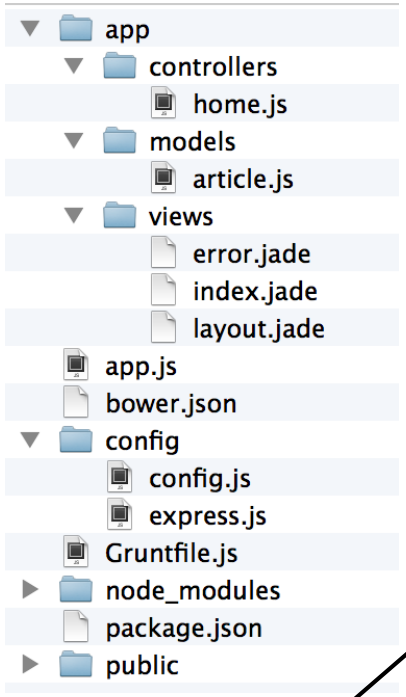
entry point

configuration




```
var express = require('express'),
  config = require('./config/config'),
  glob = require('glob'),
  mongoose = require('mongoose');

mongoose.connect(config.db);
var db = mongoose.connection;
db.on('error', function () {
  throw new Error('Can't connect to db at ' + config.db);
});

var models = glob.sync(config.root + '/app/models/*.js');
models.forEach(function (model) {
  require(model);
});
var app = express();

require('./config/express')(app, config);

app.listen(config.port);
```

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



./app/controllers/joke.js

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

```
var express = require('express');
var router = express.Router();

module.exports = function (app) {
  app.use('/jokes/', router);
};

router.get('/', function (req, res, next) {
    var jokes = [
        {
            "author" : "olivier",
            "text" : "What's the difference between
a dead baby and..."
        },
        {
            "author" : "sacha",
            "text" : "Knock! Knock! Who's there?"
        }
    ];
    res.json(jokes);
});
```

routing

payload

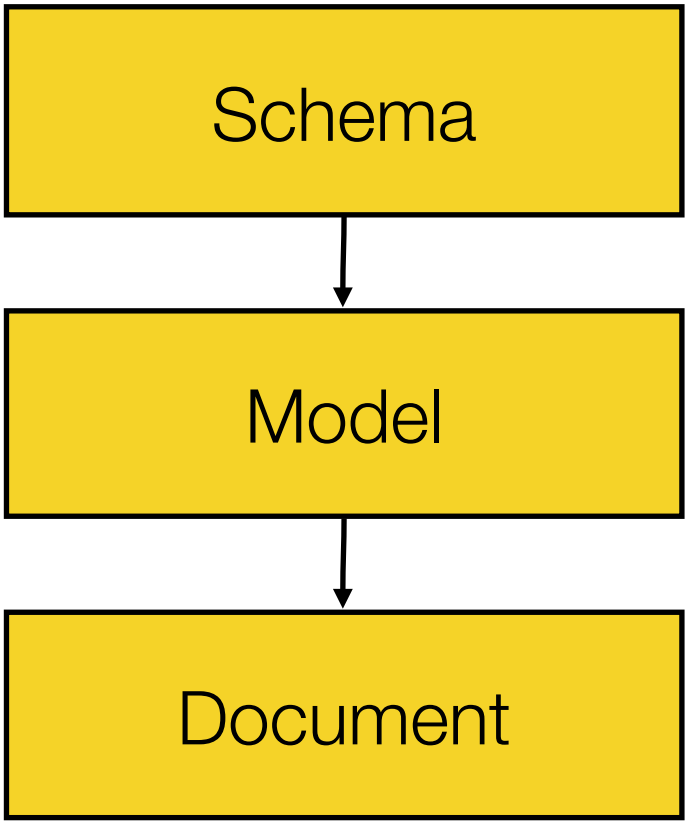
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Mongoose

mongoose: an ODM for MongoDB

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

*"Mongoose provides a **straight-forward**, schema-based solution to **modeling** your application data and includes built-in type **casting**, **validation**, **query** building, business logic hooks and more, out of the box."*



"Everything in Mongoose starts with a Schema. Each **schema maps to a MongoDB collection** and defines the shape of the documents within that collection."

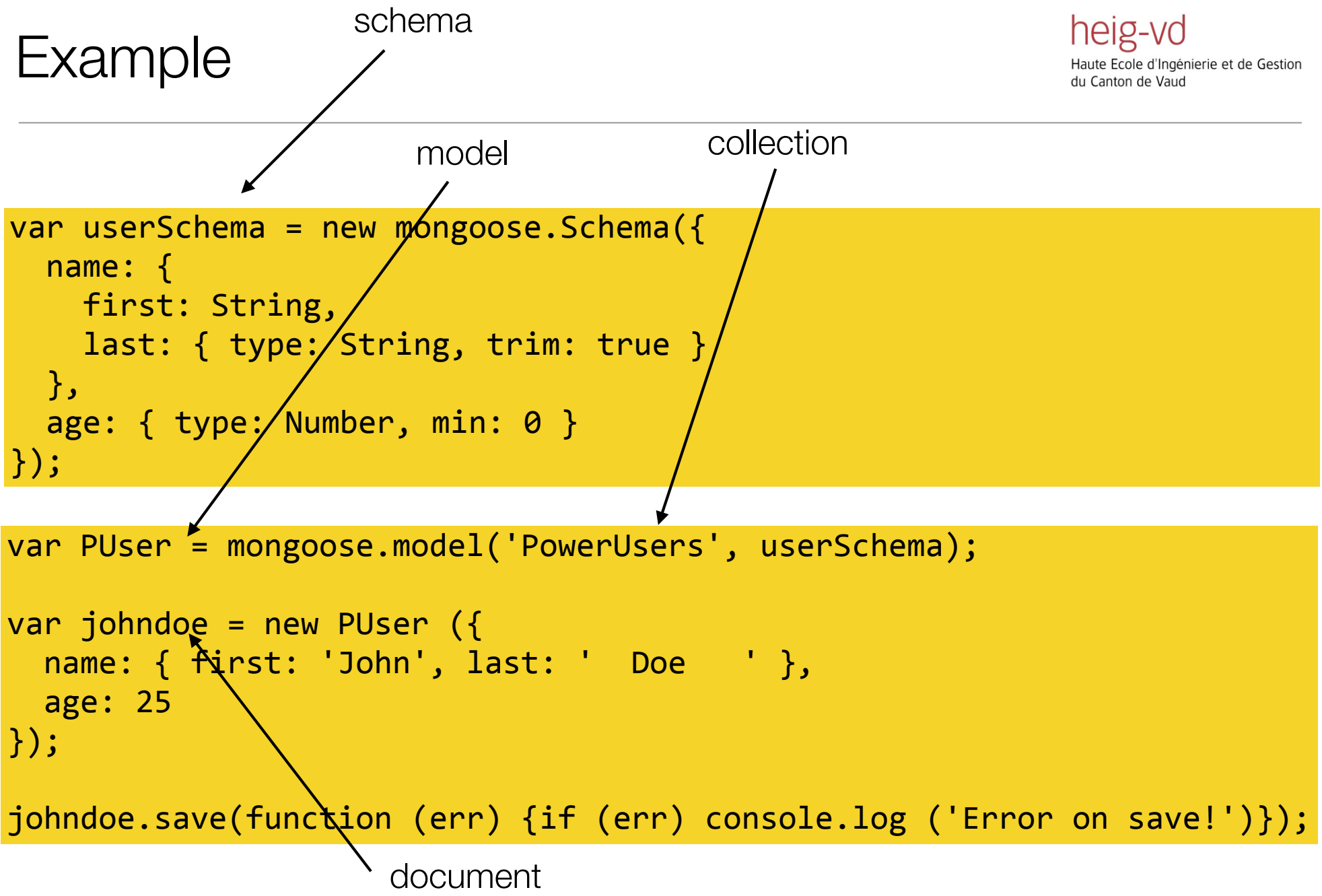
"**Models are fancy constructors** compiled from our Schema definitions."

"Mongoose documents represent a **one-to-one mapping** to documents as stored in MongoDB. Each document is an **instance of its Model**."

http://mongoosejs.com/docs/guide.html

Example

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



https://devcenter.heroku.com/articles/nodejs-mongoose

Example: query

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

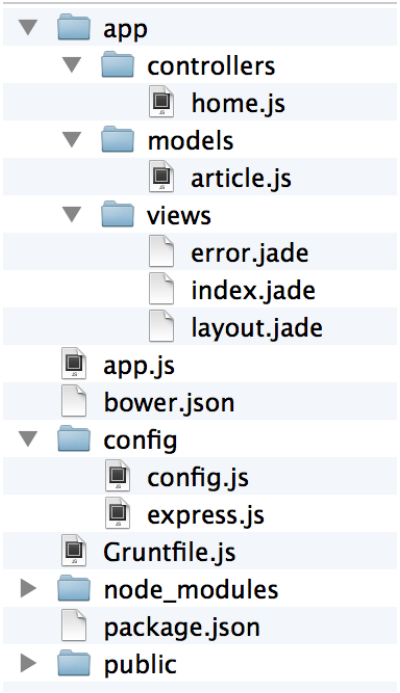
we can chain conditions

```
Person
.find({ occupation: /host/ })
.where('name.last').equals('Ghost')
.where('age').gt(17).lt(66)
.where('likes').in(['vaporizing', 'talking'])
.limit(10)
.sort('-occupation')
.select('name occupation')
.exec(callback);
```

we are interested in only some of the fields

we only want to get at most 10 documents

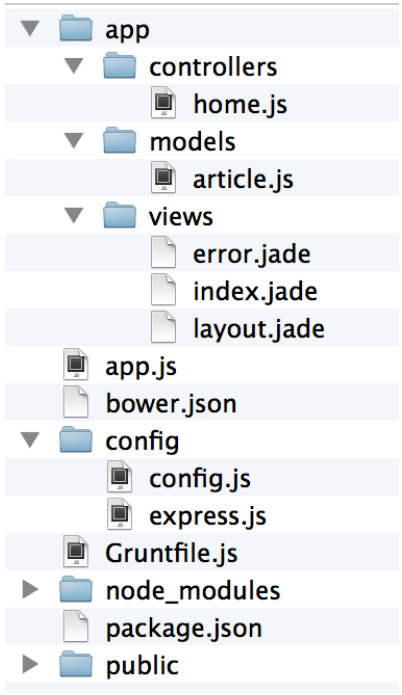
http://mongoosejs.com/docs/queries.html



./app/models/action.js

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

```
// Example model

var mongoose = require('mongoose'),
  Schema = mongoose.Schema;

var ActionSchema = new Schema({
  type: String,
  timestamp: Date,
  user: String
});

mongoose.model('Action', ActionSchema);
```



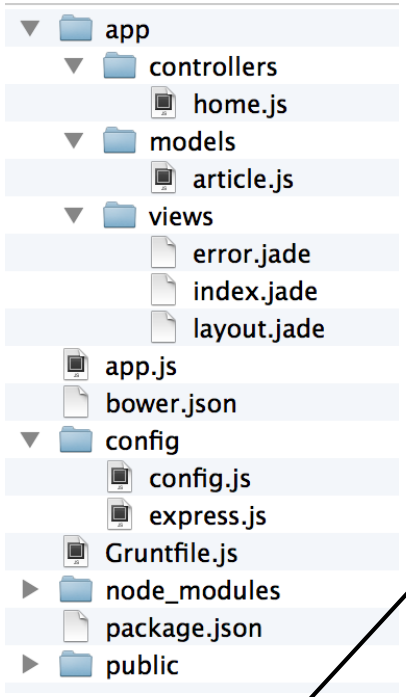
./app/controllers/actions.js

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

```
var express = require('express');
var router = express.Router();
var mongoose = require('mongoose');
var Action = mongoose.model('Action');

module.exports = function (app) {
  app.use('/actions/', router);
};

router.post('/', function (req, res, next) {
    console.log(req.body);
    var payload = req.body;
    payload.timestamp = new Date();
    var action = new Action(payload);
    action.save( function (err ) {
        if (err) {
            console.log("Unable to save action!");
            console.log(err);
            res.status(500).end();
        }
    });
    res.status(202).end();
});
```

POST /actions/

./app/controllers/actions.js



GET /actions?
type=xxx&user=xxx

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

```
router.get('/', function (req, res, next) {
    var type = req.query.type;
    var user = req.query.user;
    var query = {};
    if ( type !== undefined) {
        query.type = type;
    }
    if ( user !== undefined) {
        query.user = user;
    }

    Action.find(query, function (err, actions)  {
        if (err) {
            console.log("Unable to save action!");
            console.log(err);
            res.status(500).end();
        }
        res.json(actions);
    })
});
```