HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

02 - Foundations

Writing tests to learn (async) Javascript

TWEB 2017

Olivier Liechti

https://softeng-heigvd.github.io/Teaching-HEIGVD-TWEB-2017-Main/

https://t.me/joinchat/CPPWmAsLLgWdXQhoXTaNHw

https://t.me/joinchat/AAAAAEE3lWzr-jZRRMq3qg

Weekly menu

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

How are we going to spend the next 6 periods?

Goals (1)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Expand our development environment**
  - Add a testing framework
  - Make it a learning environment
  - Use it to practice with async programming (starting with the basics: callbacks)

Goals (2)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Organise / split project tasks**
  - Agree on the data to be extracted from GitHub and presented to users. Create fake data to document structure.
  - Understand how to extract this data
  - Understand how to present it

This morning

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Problem 1**
  Client side? Server side? I am so confused
- **Problem 2**
  Why should I write automated tests in Javascript and how?
- **Problem 3**
  How can I execute several asynchronous operations in sequence?

Problem

index.js

Client side? Server side?

I am so confused.

Forces

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Last week, we have seen how we can use the debugger both on the client side and on the server side.
- We have used a very basic script to run the demos.
- Your feedback: it is not easy to understand what runs on the client and on the server side.

Before Node

- The Javascript engines were in vast majority embedded in browsers.
- The browser downloads an HTML page, then a Javascript file and runs it.
- Javascript was almost always running on the client side, making AJAX requests and dynamically updating the page.

After Node.js

- We now have a new platform to write applications that run outside the browser.
- With the JDK, it is possible to use TCP and UDP to write client-server applications. With Node.js, it is exactly the same thing.
- In other words, I can implement a SMTP client, an HTTP server or a multi-player game with Node.js.

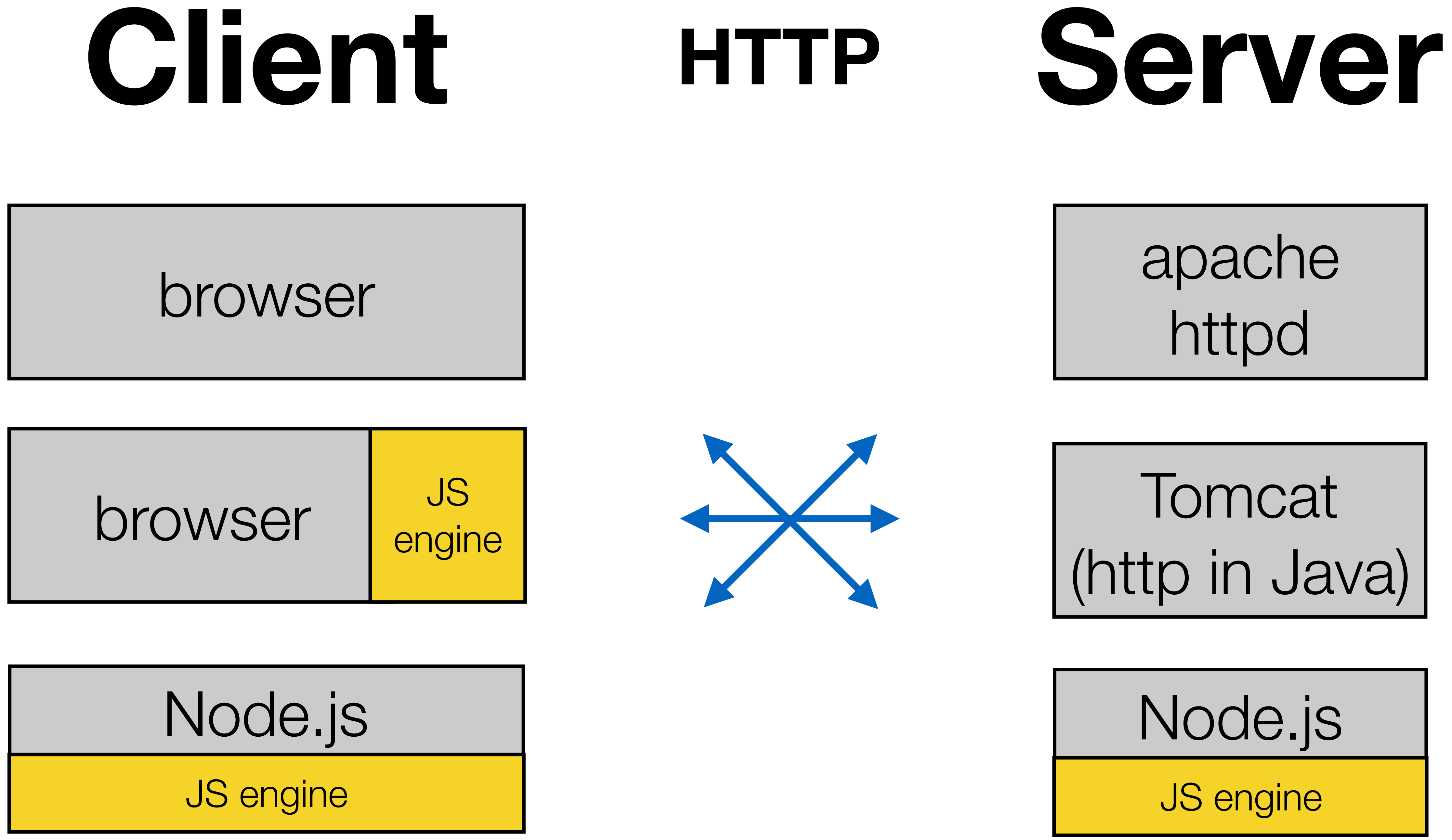
Analogy

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Node.js is similar to the Java SE JDK

```
java -jar http-server.jar
java -jar http-client.jar
java -jar cmd-line-tool.jar
```

```
node http-server.js
node http-client.js
node cmd-line-tool.js
```



e.g. a cmd line tool that sends HTTP requests to crawl a website
e.g. a cmd line tool that sends HTTP requests to fetch data from GitHub

index.js

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- You write scripts that will always run on the server side.
- You write other scripts that will always run in a browser.
- Many developers use “index.js” as a name for either client-side or server-side scripts.
- Tip: if you are confused, use explicit names such as server.js, ajax-client.js.

Problem

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Why should I write automated tests in Javascript and how?

Forces

- Automated testing is important for quality and continuous delivery.
- Writing tests is also an approach to design and document software (TDD).
- As with other topics, there are so many choices in the JS ecosystem that I don't know where to start.

Solution

- Select a testing framework: **mocha.js**
- Select an assertion library: **chai.js**
- Write tests to get familiar and document features of the Javascript language

```
const chai = require('chai');

chai.should();

describe('Javascript objects', () => {
  it('is possible to create an object with a literal', () => {
    const student = {
      firstName: 'Olivier',
      lastName: 'Liechti',
    };
    student.should.be.an('object');
    student.should.have.a.property('firstName', 'Olivier');
  });
  it('is possible to create an object from a prototype', () => {
    const father = {
      firstName: 'Olivier',
      lastName: 'Liechti',
    };
    const son = Object.create(father);
    son.firstName = 'Sacha';
    son.should.be.an('object');
    son.should.have.a.property('firstName', 'Sacha');
    son.should.have.a.property('lastName', 'Liechti');
  });
});
```

```
$ mocha test/objects.js


  Javascript objects
    ✓ is possible to create an object with a literal
    ✓ is possible to create an object from a prototype


  2 passing (8ms)
```

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
├── src
│   ├── application.js
│   ├── client-async.js
│   ├── client.js
│   └── crawler-github-pullrequests.js
└── test
    ├── application.js
    ├── chai-config.js
    ├── client-async.js
    ├── client.js
    └── crawler-github-pullrequests.js
```

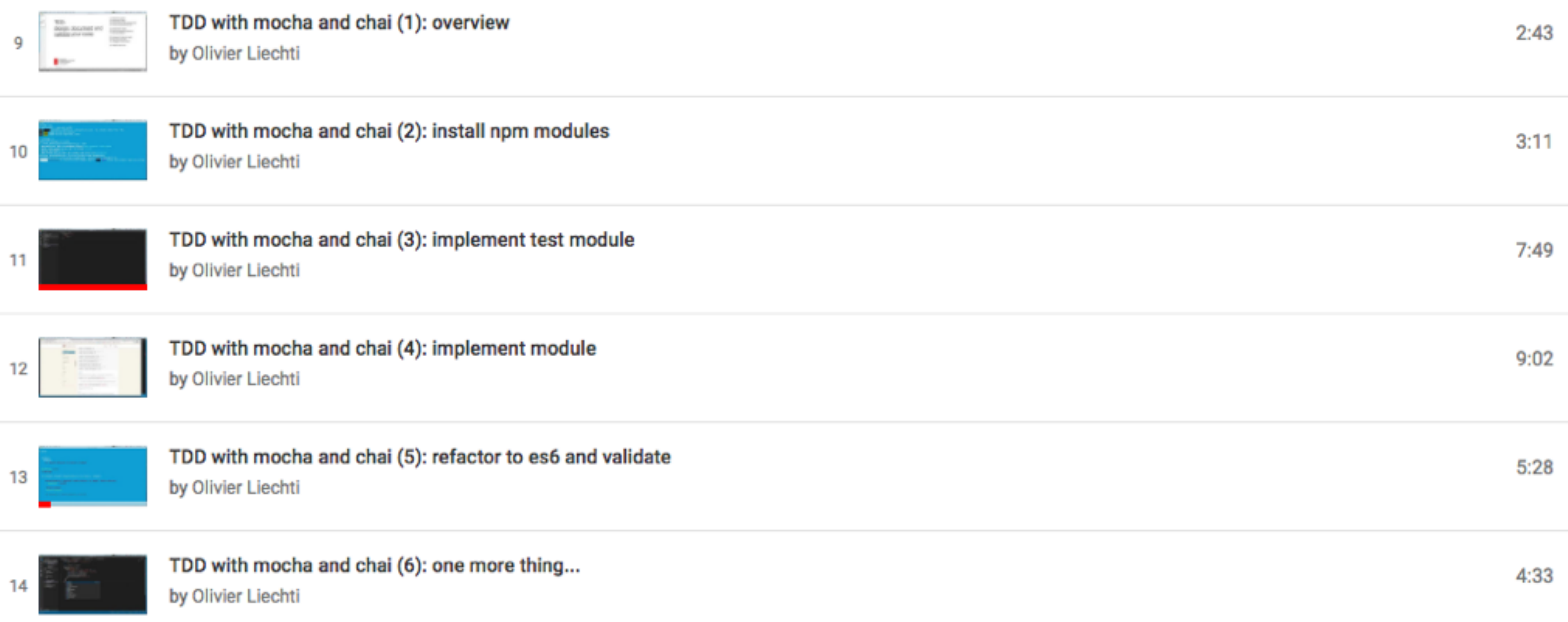
In practice

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Create a JS project with npm
- Install and configure ESLint
- Install mocha.js
- Install chai.js
- Write first tests

Webcast

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



9	TDD with mocha and chai (1): overview by Olivier Liechti	2:43
10	TDD with mocha and chai (2): install npm modules by Olivier Liechti	3:11
11	TDD with mocha and chai (3): implement test module by Olivier Liechti	7:49
12	TDD with mocha and chai (4): implement module by Olivier Liechti	9:02
13	TDD with mocha and chai (5): refactor to es6 and validate by Olivier Liechti	5:28
14	TDD with mocha and chai (6): one more thing... by Olivier Liechti	4:33

References

- mocha.js
  https://mochajs.org
- chai.js
  http://chaijs.com

Problem

How can I execute several asynchronous operations in sequence?

Forces

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Javascript engines provide a single-thread programming model (event loop).
- I/O operations are performed asynchronously. Developers provide callback functions.
- Things get complicated when several async operations need to be coordinated (serial, parallel execution).

Callbacks

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
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).

Callbacks

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
$(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

- 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.

1st 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

Fix: callback

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
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

Sequence

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- 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
```

FAIL

Sequence

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- 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"

Parallel

- 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) {
 });
 async2(function(r2) {
 });
 async3(function(r3) {
 });

  done();

}
```

FAIL FAIL

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

Parallel

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- 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) {
    results.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.

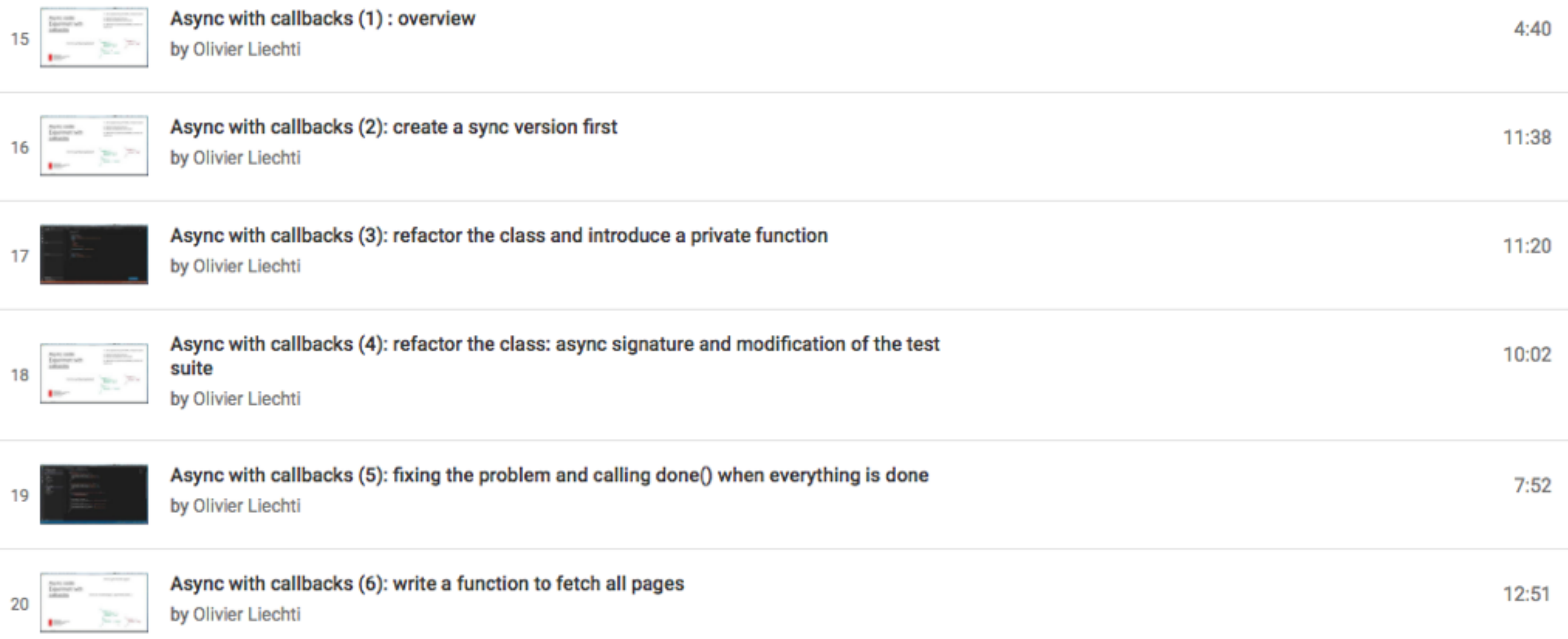
The three tasks are asynchronous, so they pass their own callback functions and receive a result when the operation completes.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Over the years, libraries and language features have been proposed to make async programming easier. We will review some of them during the semester (promises, rxjs, etc.)

Webcasts

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



15	Async with callbacks (1) : overview by Olivier Liechti	4:40
16	Async with callbacks (2): create a sync version first by Olivier Liechti	11:38
17	Async with callbacks (3): refactor the class and introduce a private function by Olivier Liechti	11:20
18	Async with callbacks (4): refactor the class: async signature and modification of the test suite by Olivier Liechti	10:02
19	Async with callbacks (5): fixing the problem and calling done() when everything is done by Olivier Liechti	7:52
20	Async with callbacks (6): write a function to fetch all pages by Olivier Liechti	12:51