



Le fantome, le zombie et testacular Karma.

**Panorama des outils de tests pour
application web moderne.**

9h30 - 12h30 - Salle Seine A

Le fantome, le zombie et testacular Karma.

**Panorama des outils de tests
pour application web moderne.**

Pierre Gayvallet



@wayofspark

Jean-Laurent de Morlhon



Pierre Gayvallet



Consultant
Xebia Studio

+6 ans expérience IT

+5 ans développement web

@wayofspark

<http://blog.xebia.fr>

pgayvallet @ xebia.fr

Jean-Laurent de Morlhon



**Directeur Technique
Xebia Studio**

**+14 ans expérience IT
+7 ans pratiques agiles**

@morlhon

<http://blog.xebia.fr>

jlmorlhon @ xebia.fr

Xebia Studio



XEBIA STUDIO VALEURS L'ÉQUIPE TECHNOLOGIES NOUS REJOINDRE À PROPOS

AGILE DEVELOPMENT DONE RIGHT !

Notre métier, développer des applications de grande qualité : vos applications.



BESOIN DE DÉLIVRER VOTRE PROJET RAPIDEMENT ?

Nous aidons les entreprises à garder une longueur d'avance.

Notre agilité, notre expérience et notre connaissance des solutions de PaaS (Platform as a Service) nous permettent de démarrer votre projet en moins d'une journée.

Nous délivrons en production en continu : en quelques semaines une première version de l'application est délivrée en production et disponible pour vos utilisateurs.



TECHNOLOGIES

Nous avons une expertise reconnue dans l'écosystème Java.

Nous maîtrisons un grand nombre de technologies comme Grails, Play!, PhoneGap



L'ÉQUIPE

Nous constituons des équipes dédiées à la réalisation de votre produit.

Elles sont composées d'ingénieurs de Xebia expérimentés dont les compétences sont



CULTURE

Nous sommes passionnés par le développement de logiciel, nous avons une culture d'excellence mêlée à une geek-attitude que nous cultivons avec soin.

Pourquoi on est là !



Jasmine
The Jasmine logo is a stylized green flower with five petals and a central green dot.

Application web moderne

Navigateur



Serveur



Base



PL/SQL

ORACLE®

Le programme !

- Un mot à propos de Testing

- Taxonomie des tests
- Méthodologie
- Vocabulaire

LIVE
Phantom.js

LIVE
Zombie.js

- QUnit

LIVE
Sinon.js

LIVE
KarmaJS

LIVE
Mocha

LIVE
Chai.js

- Buster.js
- jsCover
- Plato
- Wrap-up



TESTING

I FIND YOUR LACK OF TESTS DISTURBING.

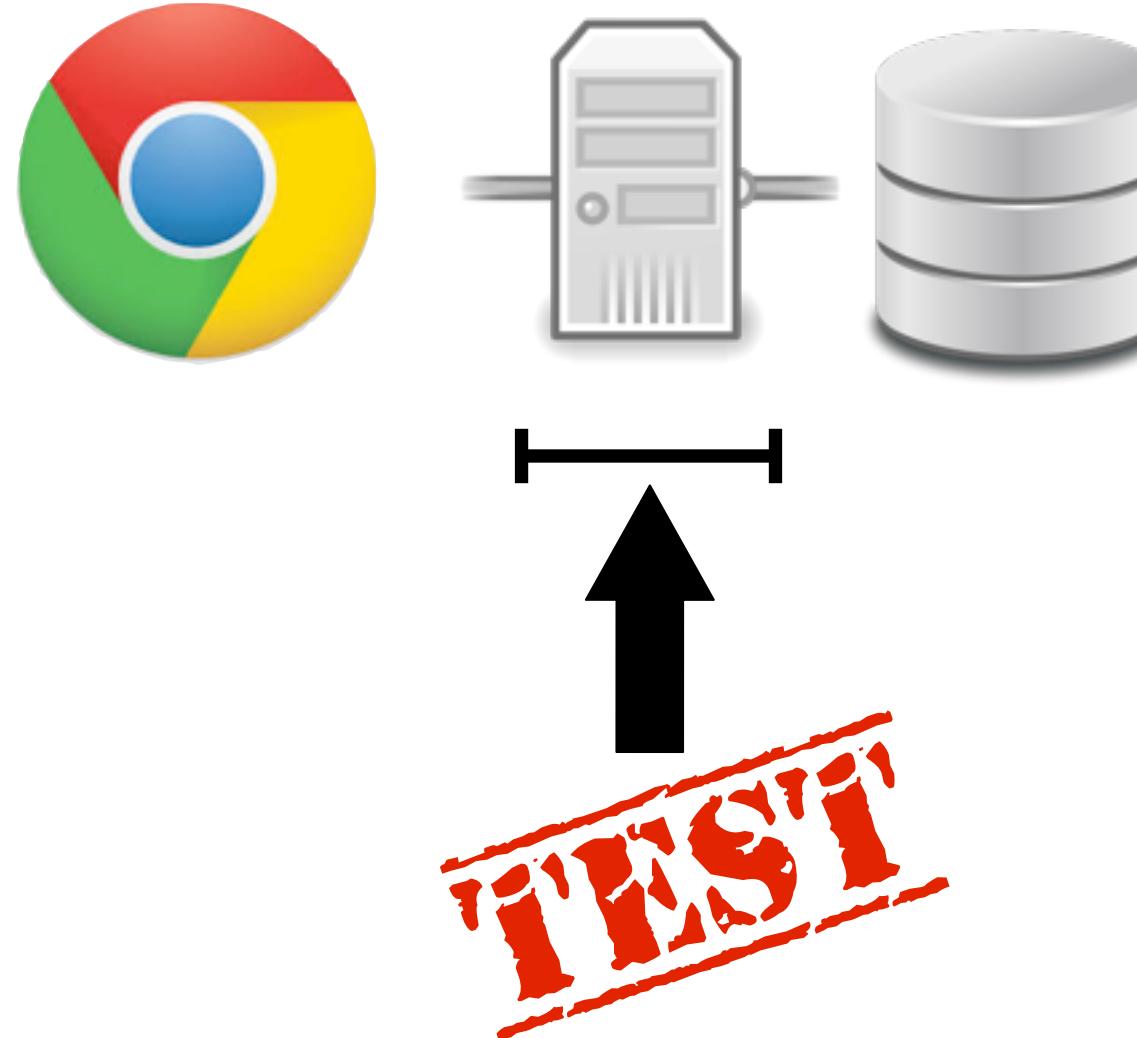
Testing

1. Taxonomie des tests
2. Méthodologie de testing
3. Vocabulaire

Taxonomie des tests

- Tests d'acceptance
- Tests d'intégration
- Tests unitaires

Tests unitaires



- Test technique
- Ultra rapide (- 10 ms)
- Le code testé est isolé du reste de l'application

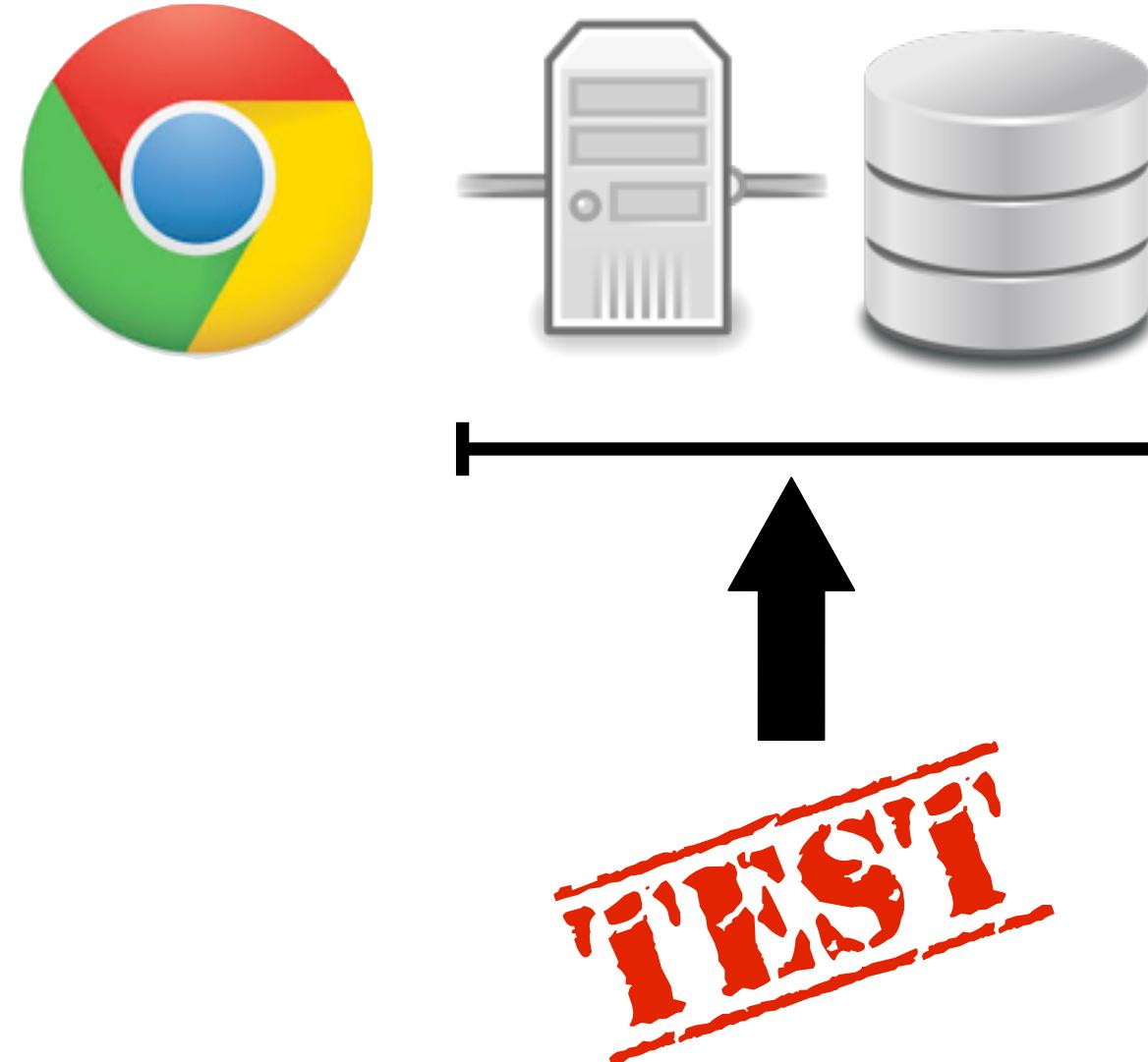
Exemple test unitaire

```
@Test
public void should_find_winning_bids_when_there_is_option_bids() {
    Bid bid = new Bid();
    bid.setStatus(OPTION);

    ExpiredBids expiredBids = new ExpiredBids();
    expiredBids.add(bid);

    assertThat(expiredBids.isAnyWinning()).isTrue();
}
```

Tests d'intégration

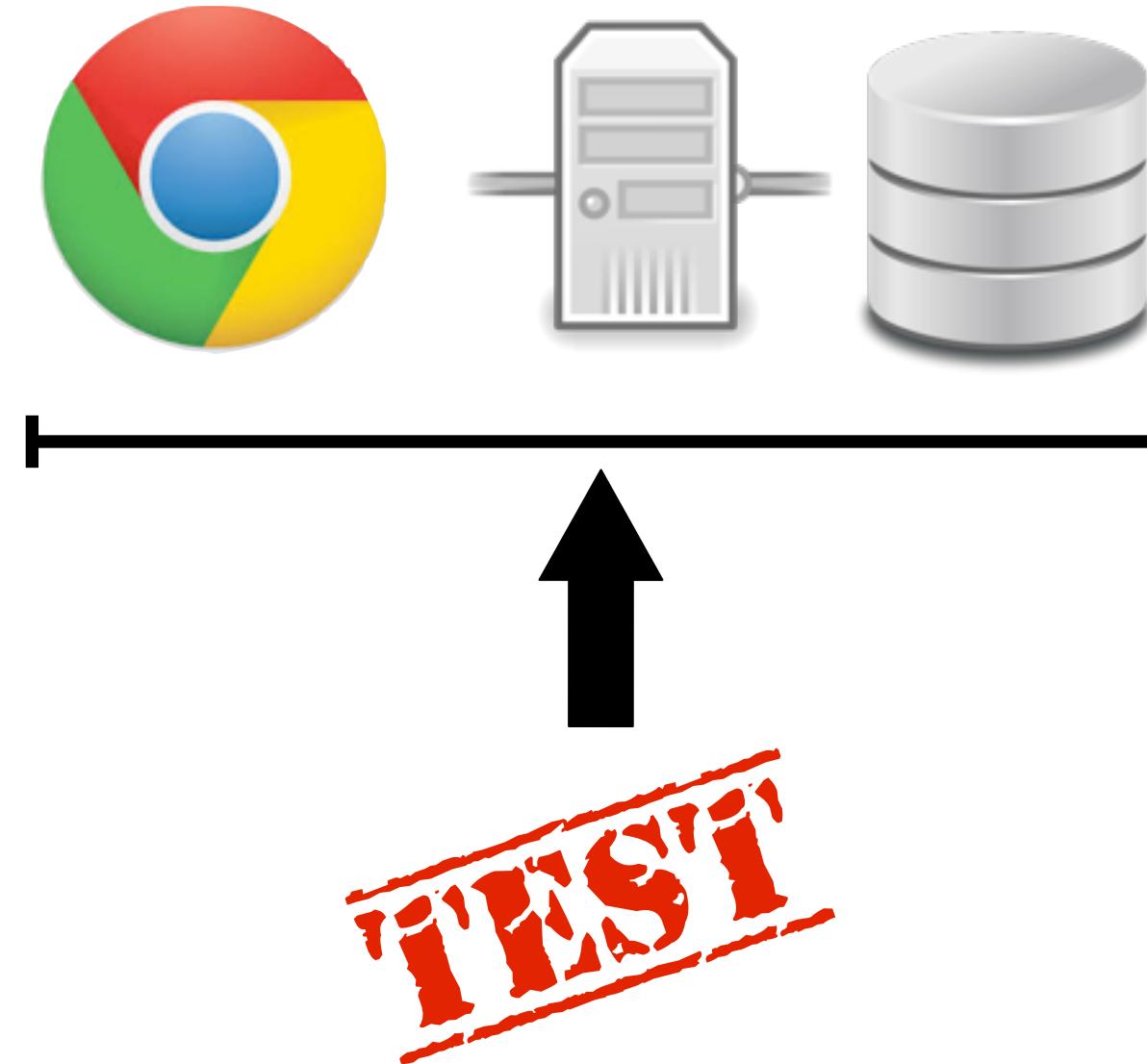


- Test technique
- Rapide (1s à 500 ms)
- Le code testé est partiellement groupé avec le reste de l'application

Exemple test intégration

```
@Test
public void i_can_retrieve_auctions_bided_by_an_investor() {
    Investor investor = investors.add(buildInvestor());
    Auction auction = auctions.add(buildAuction());
    auctions.addBidToAuction(auction.getId(), buildBid());
    assertThat(auctions.fromInvestors(investor.get_id())).isNotEmpty();
}
```

Tests d'acceptances



- Test fonctionnel
- lent +500ms
- Toute l'application est testé de bout en bout.

Exemple test d'acceptance

Gherkin : BDD Style

Scenario: subtract two numbers

Given I have entered 10 in the calculator

And I have entered 7 in the calculator

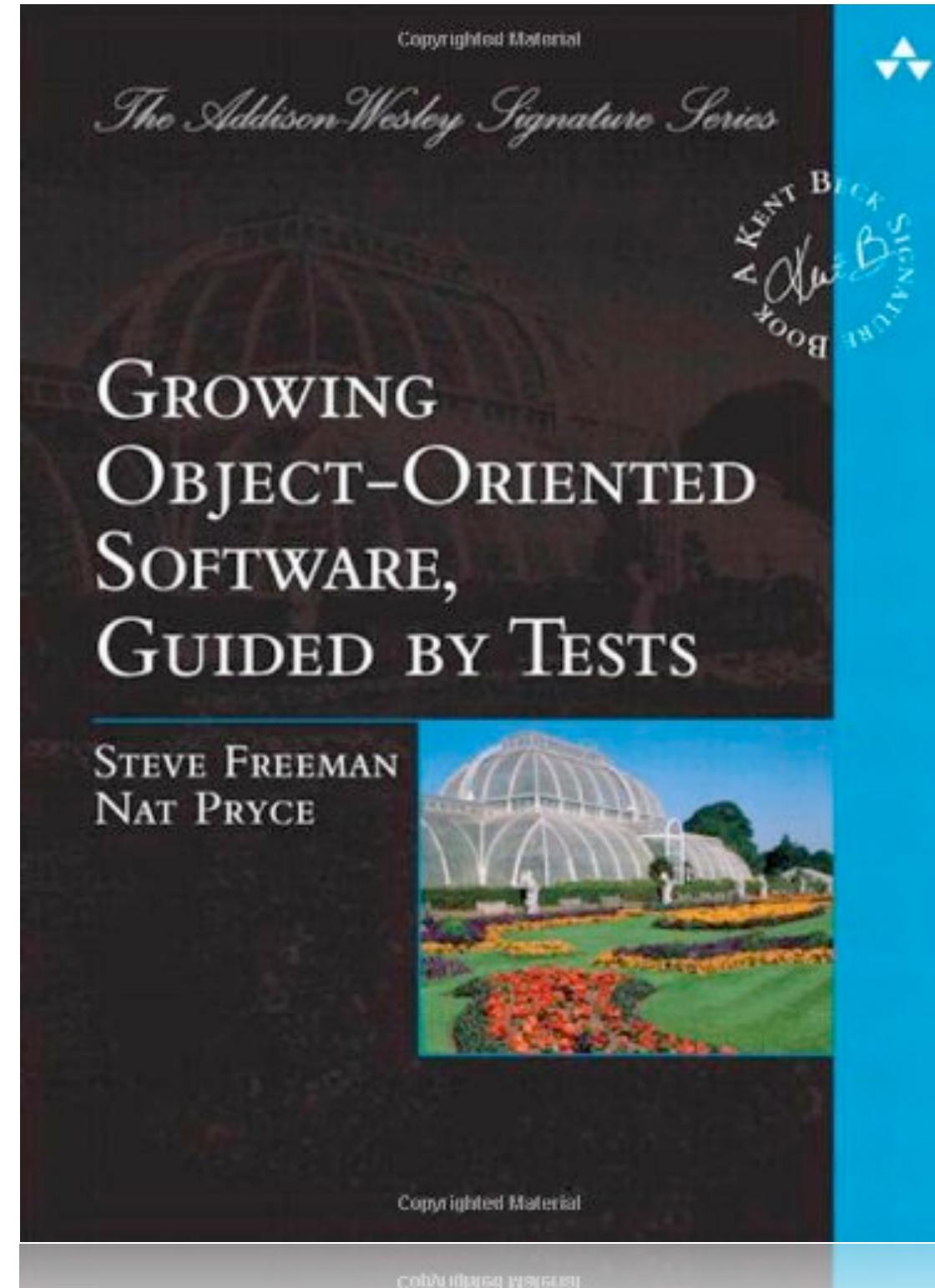
When I press subtract

Then the result should be 3 on the screen

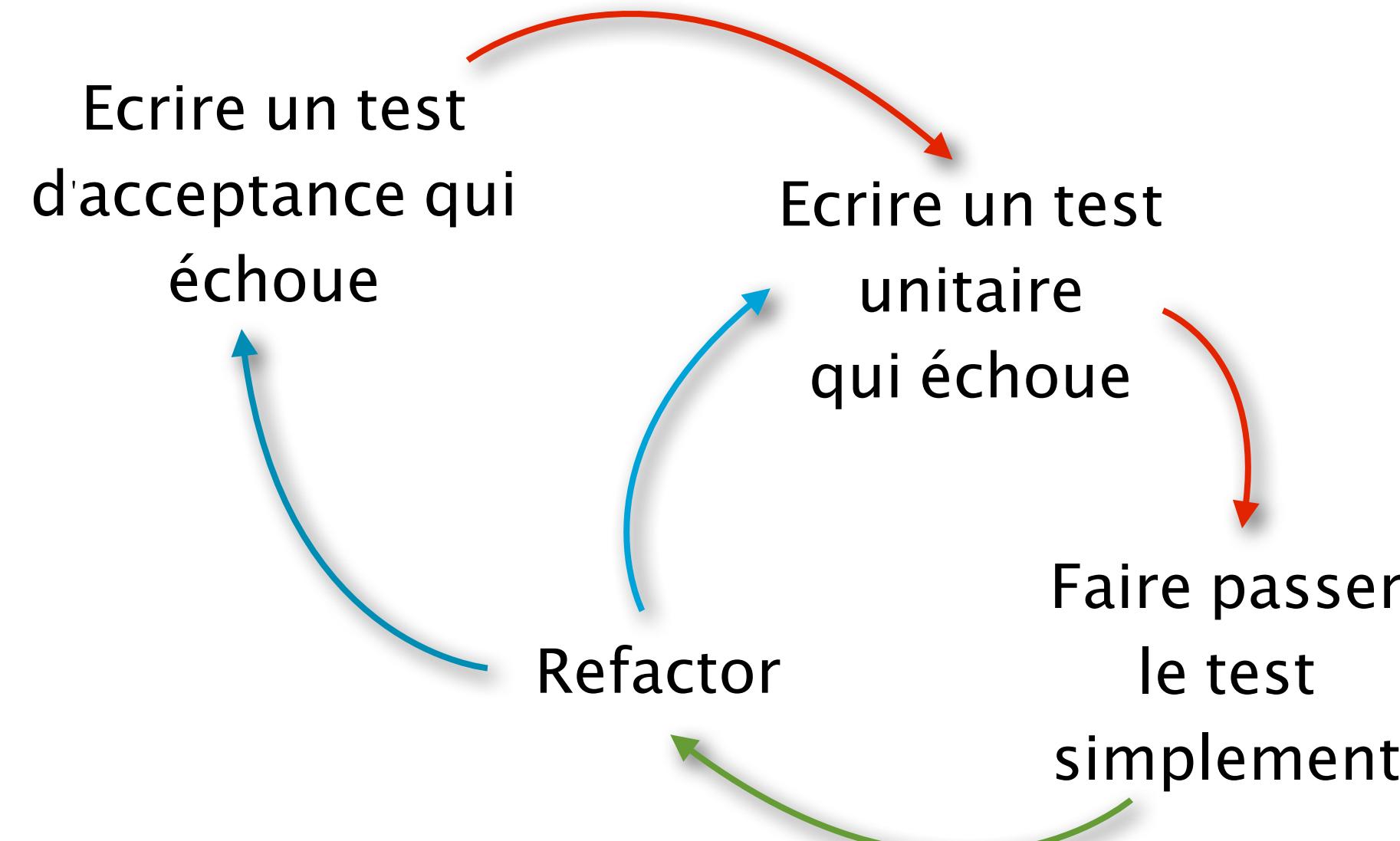
Exemple test d'acceptance

```
it 'should find my company account from siren', (done) ->
  browser = new Browser()
  browser.visit home, ->
    browser.clickLink 'Inscrivez-vous', ->
      browser.choose 'une entreprise'
      browser.pressButton 'Envoyer', ->
        browser.fill 'sirenCode', '394149496'
        browser.clickLink 'Rechercher', ->
          expect(browser.evaluate('$(`input[name="name"]`).val()')).to.equal 'XEBIA'
          done()
```

GooS



Technique outside-in



Browser Headless ?

Un navigateur "sans tête"

Un navigateur sans interface graphique

Exécute depuis la ligne de commande

Ultra rapide

Automatisable !

On voit pas toujours ce qu'il se passe...

Browser Headless

- PhantomJS (qt)
- HtmlUnit (java)
- Zombie.js (node.js)
- ...



PhantomJS

**Full Web Stack
No Browser Required**

<http://phantomjs.org/>
<http://github.com/ariya/phantomjs>



**DEVOXXX
FRANCE**

27 au 29 mars 2013

PhantomJS

- Navigateur headless basé sur QtWebKit (JavascriptCore)
 - vrai rendering (pas d'émulation)
 - api full javascript
 - pur headless (pas besoin de X11)
- Crée par Ariya Hidayat (@ariyahidayat) en 2010
- Multi-plateformes (Linux, MacOSX, Window)
- (Très) rapide
- Utilisé par beaucoup de gros acteurs (Bootstrap, Grunt, Zepto...)

Exemple de code

```
var page = require('webpage').create();

page.open('http://localhost:8080', function (status) {

    var title = page.evaluate(function() {
        return document.title;
    });
    console.log("page title = " + title);

    var titles = page.evaluate(function() {
        var productTitleElements = document.querySelectorAll(".produit h2");
        var productTitles = [];
        for(var i=0; i<productTitleElements.length; i++) {
            productTitles.push(productTitleElements[i].textContent);
        }
        return productTitles;
    });
    console.log("article titles = ", JSON.stringify(titles));

    phantom.exit();
});
```

Contexte d'évaluation

- Double contexte d'exécution client/serveur
- Communication délicate (uniquement des types natifs)
- Erreurs vite arrivées

```
// BAD
var title = null;
page.evaluate(function() {
    title = document.title ;
});
console.log(title) // prints null.

// GOOD
var title =
page.evaluate(function() {
    return document.title;
});
console.log(title) // prints the
title.
```

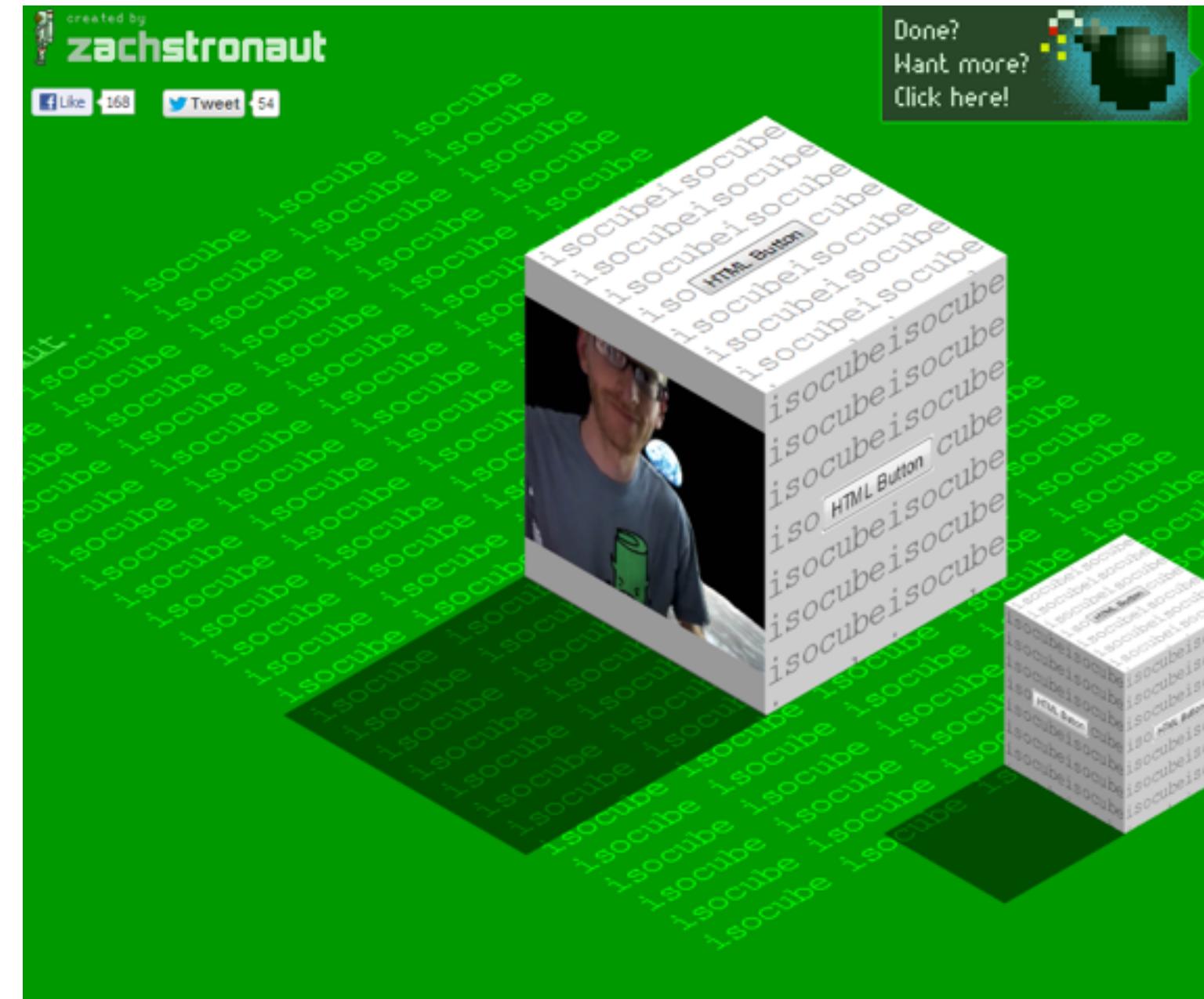
Features

- Interpréteur coffeescript embarqué
- Gestion du viewPort
- Permet de prendre des captures d'écran
- Accès à la console de debug de chrome
- Moteur WebKit : Respect de beaucoup de standards :
 - Canvas, LocalStorage, Css Animations, WebWorkers...

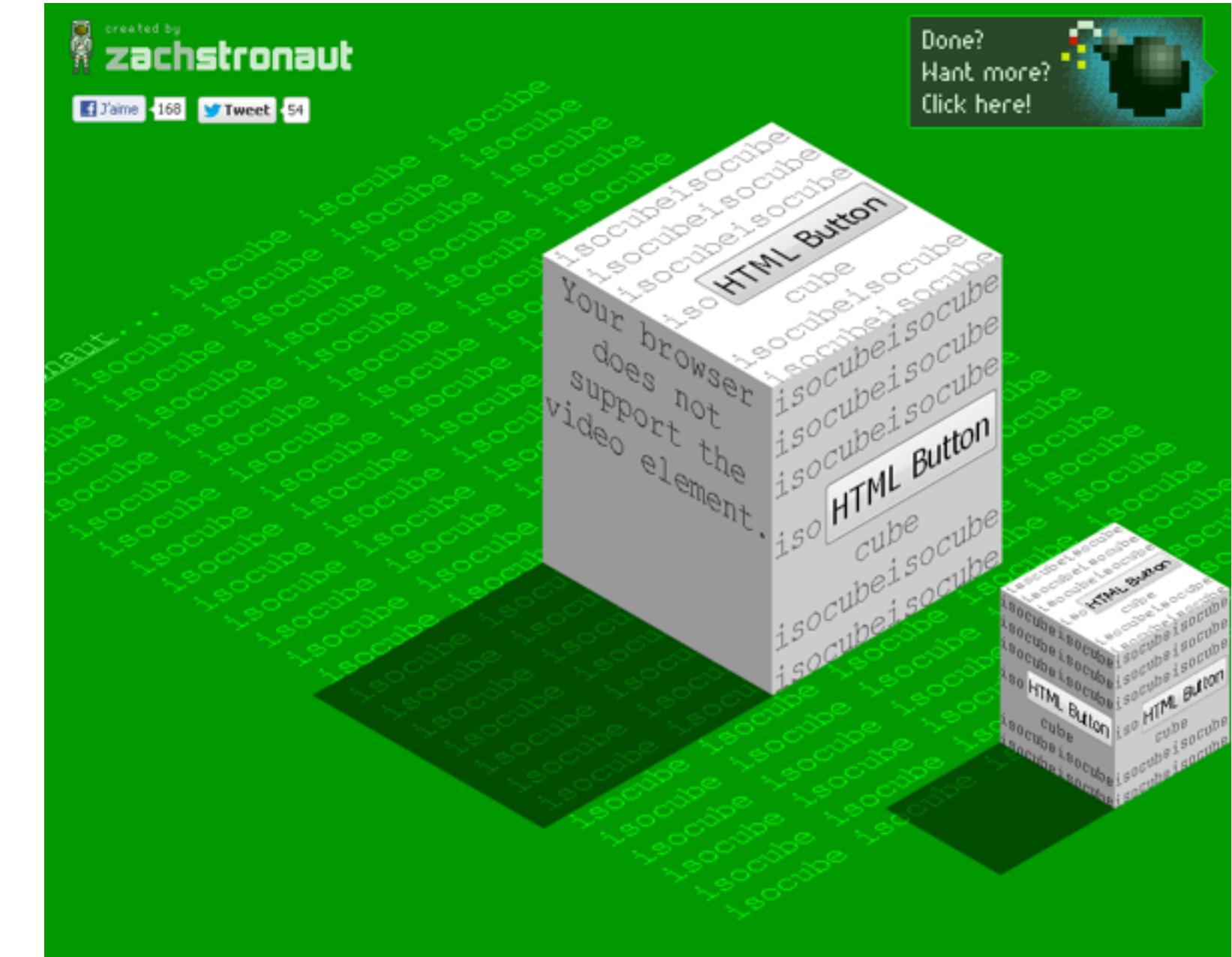
PhantomJS

Rendu très fiable, même sur les derniers standards :

Chrome



Phantom



Les limites du rendering

- Flash
 - Enlevé depuis la version 1.5 (empêchait d'être pur headless)
- CSS 3D
- Video et Audio
- WebGL
- Geolocation
- Les Fonts... (rarement gênant pour le testing)

(Et oui, tout le reste fonctionne)

Écosystème

- GhostWriter (implémente la spec WebDriver)
- Tests Runners
 - Poltergeist (capibara)
 - mocha-phantom-js
 - ...
- Tests Frameworks
 - Lotte
 - WebSpecter
 - CasperJs
 - ...
- Outils de screenshot (capturejs, node-webshot, ...)

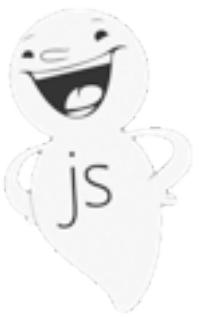
En quelques mots...

Outil très puissant : vrai rendering, fiable, très rapide.

Mais...

- API beaucoup trop brute
- Utilisation du double contexte déroutante (au début)
- Gestion asynchrone délicate (cascade de callbacks)
- Pas un framework de test en soit

Heureusement, il y a...



CasperJS

**navigation scripting & testing utility
for PhantomJS**

<http://casperjs.org>

<http://github.com/nlk0/casperjs>



CasperJS

- Basé sur PhantomJS
- Rajoute une API plus haut niveau :
 - Fluent API
 - Meilleure gestion de l'asynchronisme
 - Méthodes pour interagir avec la page
 - Création de tests fonctionnels
 - Beaucoup de “sugar methods”

Gestion d'étapes

Ouvrir plusieurs urls consécutives

Avec phantom

```
var page = require("webpage").create();

page.open(url1, function(status) {
    if(status=="fail") { phantom.exit(); }

    page.open(url2, function(status) {
        if(status=="fail") { phantom.exit(); }
        page.open(url3, function(status) {
            if(status=="fail") { phantom.exit(); }

            // DO SOMETHING

        });
    });
});
```

Avec casper

```
var casper = require("casper").create();

casper.on("load.finished", function(status) {
    if(status=="fail") { this.exit(); }
});

casper.start(url1);
casper.thenOpen(url2);
casper.thenOpen(url3, function() {

    // DO SOMETHING

});
```

- Plus lisible
- Moins de répétitions
- Évite l'effet pyramide

Plus facile

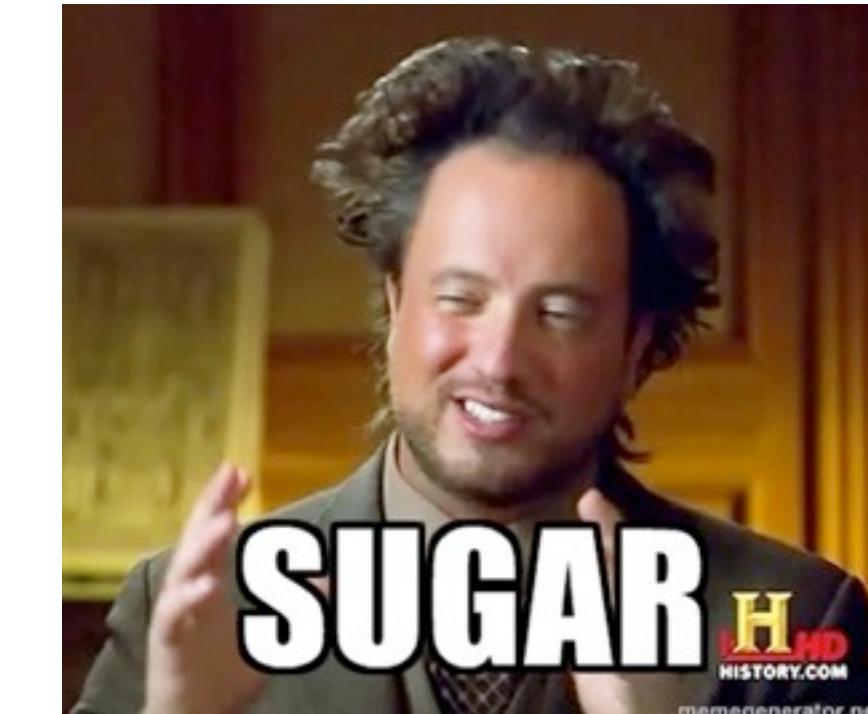
Effectuer une capture d'une partie de l'écran...

Avec phantom

```
var page = require('webpage').create();
page.open("http://localhost:8080", function(st) {
    if(st != "success") {
        phantom.exit();
    }
    var headerClipRect = page.evaluate(function() {
        var clip =
document.querySelector('#header').getBoundingClientRect();
        return {
            top: clip.top, left: clip.left,
            width: clip.width, height: clip.height
        };
    });
    var oldClipRect = page.clipRect;
    page.clipRect = headerClipRect;
    page.render("header.png");
    page.clipRect = oldClipRect;
    phantom.exit();
});
```

Avec casper

```
var casper = require("casper").create()
casper.start('http://localhost:8080', function() {
    this.captureSelector('header.png', '#header');
});
casper.run();
```



More sugar ?

Surcouche de l'api de phantom très riche :

- manipulation de la page (click, fill, mouseEvent...)
- synchronisation (waitForSelector, waitForResource, ..)
- lecture du DOM (getElementAttribute, getFormValues, ...)
- debug (debugHTML, dump ...)

Tester API

```
casper.test.comment('home tests');

casper.start(homeUrl, function() {
    this.test.assertTitle(homeTitle, "home title should match");

    this.test.assertEval(function() {
        return $(".product").length == 7;
    }, "home should display 7 products");

    this.click(".product a").click();
});

casper.then(function() {
    this.test.assertUrlMatch(/^http:\/\/localhost:8080\/product/, "should be on a product page");
});

casper.run(function() {
    this.test.done(3);
});
```

Lancement des tests

```
$ casperjs test --includes=foo.js,bar.js \
    --pre=pre-test.js \
    --post=post-test.js \
    --direct \
    --log-level=debug \
    --fail-fast \
    --xunit=xunit.xml
test1.js test2.js /path/to/another/test/dir ## path to test and test folders to launch
```

- setup et teardown passés en CLI et globaux à TOUS les tests... :/
- permet d'exporter les résultats au format xUnit (pour le CI)

Version 1.1 (TBA)

Une syntaxe de test plus proche du format xUnit

```
casper.test.begin('My suite name', 2, {
    setUp: function(test) {
        // some setup stuff
    },

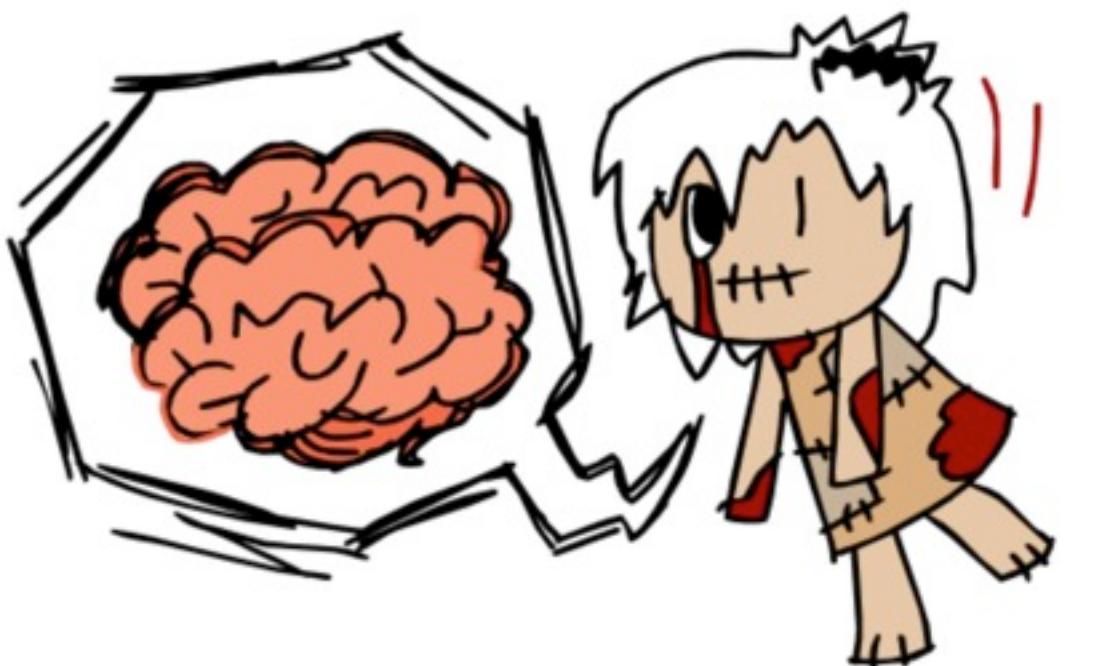
    tearDown: function(test) {
        // some tearDown stuff
    },

    test: function(test) {
        casper.open(url, function() {
            test.assertTitle("my title", "title should match")
        });
        casper.then(function() {
            // some more test
        });
        casper.then(function() {
            test.done(); // finishing the test.
        })
    }
});
```

CasperJS

- la puissance de phantom, l'API casper en plus
- exécution très rapide
- Sortie au format xUnit (pour intégrer dans un CI)
- l'API tester de la 1.1 semble très prometteuse
- API de test 1.0 pas encore très mature (setup/teardown limités...)
- launcher python ou ruby (difficile sous window...)

Zombie.js



**Insanely fast, full-stack,
headless browser testing**



**DEVOXXX
FRANCE**

27 au 29 mars 2013

Zombie.js

- Browser HeadLess
 - Tourne sur node.js
 - Repose sur des bibliothèques d'émulation
-
- v1.4.1 (2.0 en gestation)
 - MIT Licence

Emulation ?

Zombie.js n'est pas un "vrai" navigateur, il se sert de bibliothèque d'émulation pour simuler un navigateur

- JSDom de Elijah Insuas (Dom Emulation)
- HTML5 de Aria Stewarts (HTML5 Parsing)
- Sizzle.js de John Resig (Css Selectors)
- ...

Zombie.js

Un navigateur au rabais ?

- Balisage HTML5
- Champ de formulaire HTML5 (search, url etc...)
- Sélecteurs CSS3
- Cookie & Web Storage
- XMLHttpRequest
- setTimeout/setInterval
- pushState, popState & hashchange
- event, alert, confirm & prompt
- WebSockets & Server-Sent events

Zombie.js

Api Fluide :

```
var Browser = require("zombie");

browser = new Browser();
browser.visit("http://localhost:3000/", function () {

    browser.fill("email", "jeanlaurent@devoxx.fr");
    browser.fill("password", "youDontWantToKnow");

    browser.pressButton("Login", function() {
        assert.ok(browser.success);
        assert.equal(browser.text("title"), "Welcome To Brains Depot");
    });
});

});
```

Zombie.js

Appel chainé :

```
browser
  .fill("email", "jeanlaurent@devoxx.fr")
  .fill("password", "youDontWantToKnow")
  .pressButton("Login", function() {
    assert.ok(browser.success);
});
```

Zombie.js

Support des promesses :

```
browser = new Browser();
browser.visit(home)
  .then(function() {
    browser.clickLink("Panier");
  }).then(function() {
    browser.success.should.equal(true);
    browser.text("h2.font").index0f("panier est vide").should.equal(1);
 });
```

Zombie.js

Programmation avec les sélecteurs CSS

```
browser.queryAll("#content > .produit").length.should.equal(7);
```

```
browser.text("#nb-article").should.equal('1 article');
```

Zombie.js

Quand il ne reste plus aucun espoir...

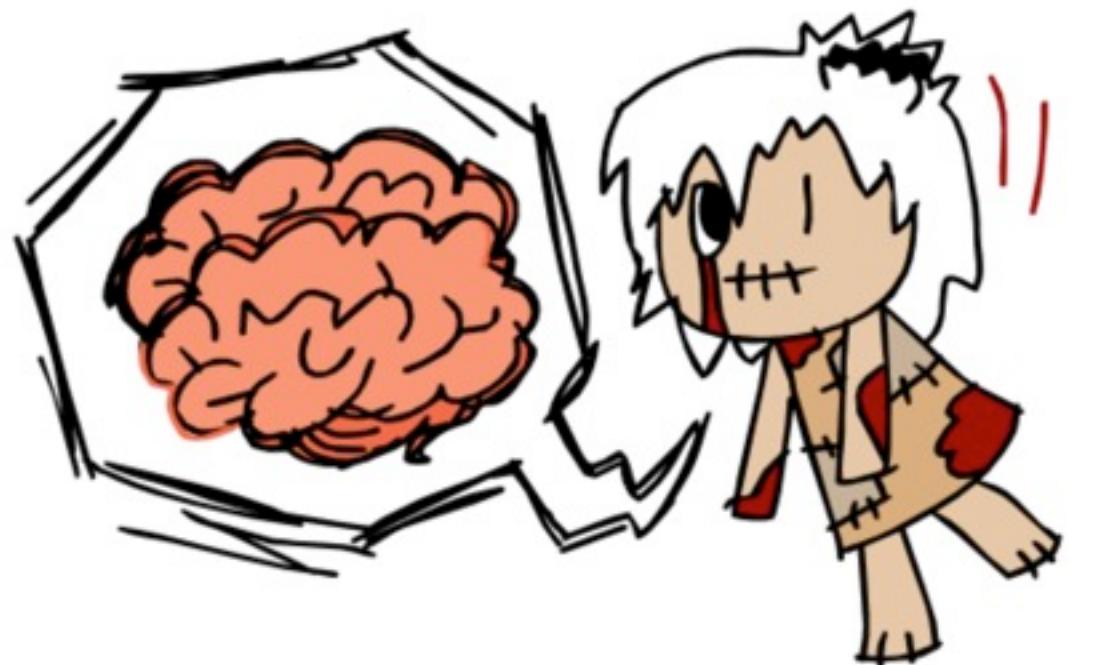
```
expect(browser.evaluate('$(`input[name="name"]`).val()')).to.equal('FooBar');
```



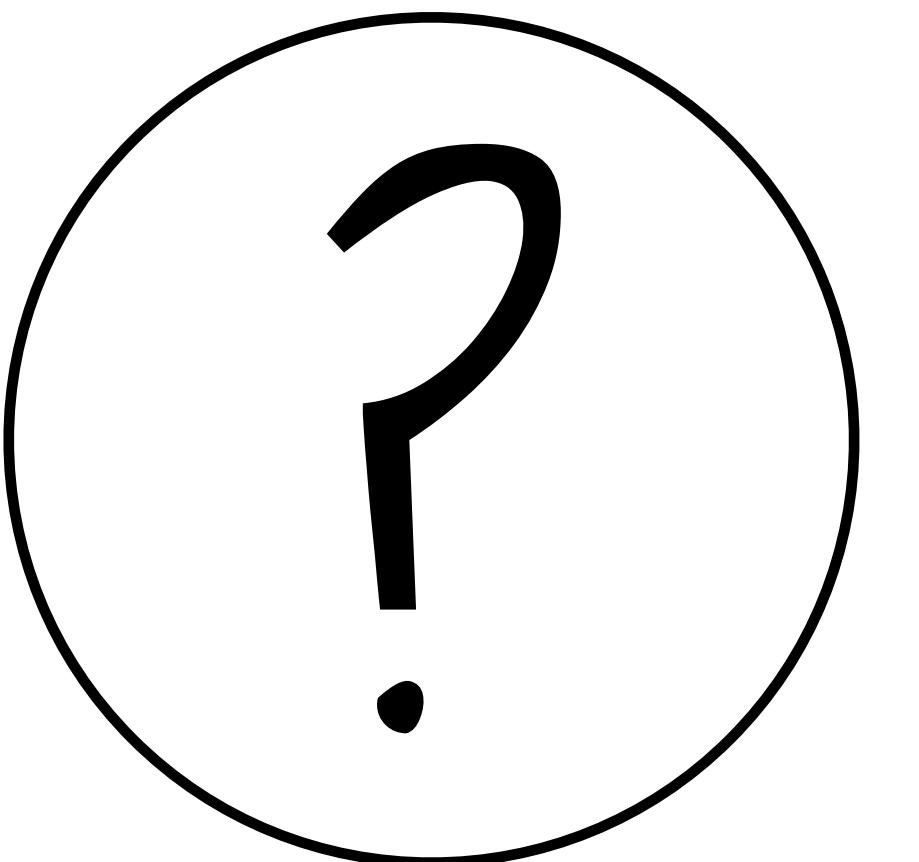
Zombie.js

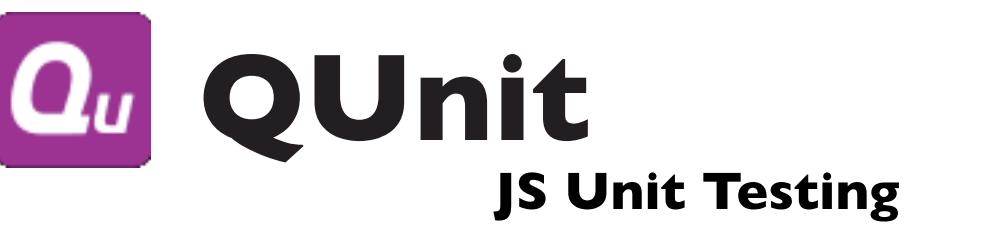
- Ultra rapide
- Fluent API
- Code de tests très lisible
- Emule un navigateur
- Tourne mal sous windows
- Développement en berne
- Erreur parfois cryptique
- Intégration dans un "build" java, non trivial

Demo Time : Zombie.js



27 au 29 mars 2013





<http://qunitjs.com/>
<http://github.com/jquery/qunit>



QUnit

- Framework de tests unitaires très simple et léger
- Développé à l'origine par John Resig (jQuery)
 - Devenu public en 2008
- Tests lancés dans le navigateur
- Format xUnit
- Extensible (plugins)

QUnit

Très simple d'utilisation :

tests.js

```
module("Hello.World");

test("universe should agree", function() {
    ok(42==42, "probably true");
});

module("JS.Troll");

test("equality should be trolling material", function() {
    equal('\n\r\t', 0, "a whitespace string is equal to
zero");
});

test("transitivity should be trolling material", function() {
    equal(0, "", "zero is equal to empty string");
    equal(0, "0", "zero is equal to the strong '0'");
    notEqual("", "0", "the empty string is not equal to the
'0' string");
});
```

qunit.html

```
<html>
<head>
    <title>Hello QUnit</title>
    <link rel="stylesheet" href="/qunit.css">
</head>
<body>
    <div id="qunit"></div>
    <div id="qunit-fixture"></div>
    <script src="/qunit.js"></script>
    <script src="/tests.js"></script>
</body>
</html>
```

Résultat

Hello QUnit

Hide passed tests Check for Globals No try-catch
Module: < All Modules > ▾

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/25.0.1364.172 Safari/537.22

Tests completed in 40 milliseconds.
5 assertions of 5 passed, 0 failed.

1. Hello.World: universe should agree (0, 1, 1)	Rerun	1 ms
2. JS.Troll: equality should be trolling material (0, 1, 1)	Rerun	1 ms
3. JS.Troll: transitivity should be trolling material (0, 3, 3)	Rerun	0 ms

Assertions

Peu nombreuses :

- `ok()`
- `equal()`
- `deepEqual()`
- `strictEqual()`
- `throws()`
- `notEqual / not[...]`

et c'est tout...

```
// ok()
ok(3 > 2); // pass
ok(1==2); // fail

// equals()
equal("same value", "same value"); // pass
equal("some value", "another value"); // fail

// deepEqual()
var a = { someValue : 2 };
var b = { someValue : 2 };
equal(a, b); // fail because a and b have not the same reference.
deepEqual(a, b); // pass because a and b have the same attributes.

// strictEqual()
equal(1, true); // pass because (1==true) is true
strictEqual(1, true); // fail because (1===true) is false"

// throws()
throws(function() {
    throw "thrown";
}, "thrown"); // pass because expected exception is raised.
```

Tests asynchrones

Utilisation de asyncTest et de start

```
asyncTest("stuff and remote work loading", function() {  
  
    var stuff = new Stuff();  
    ok(!stuff.readyToWork);  
  
    $.get("/some/work/url", function(work) {  
  
        stuff.loadWork(work);  
  
        ok(stuff.readyToWork);  
  
        start(); // telling the test to start.  
    });  
  
});
```

Ecosystème

Plugins divers :

- Assertions (canvas, html...)
- Runners (junit)
- Integration (drupal, rails)
- Framework integration (sinon, jsTestDriver)
- Themes

Définition d'une assertion

```
QUnit.extend( QUnit.assert, {
    /**
     * given number should be strictly
     greater than given minimum
     */
    greaterThan : function(number, minimum,
message) {
        QUnit.push(number > minimum, number,
">" + minimum, message);
    }
});

// [...] using the assertion in a test
test("testing my assertion",
function(assert) {
    var five = 5;
    var three = 3;
    assert.greaterThan(five, three, "5
should be greater than 3")
});
```

Sinon.JS

Standalone test spies, stubs and mocks for JavaScript

<http://sinonjs.org>

<http://github.com/cjohansen/Sinon.JS>



**DEVOXXX
FRANCE**

27 au 29 mars 2013

Sinon.JS

- Librairie de test pour javascript
- Crée par Christian Johansen ([@cjno](#))
 - Créateur de juicer, core developer de Buster.js
 - Auteur du livre « Test-Driven Javascript Development »
- Release initiale en juin 2010
 - Version actuelle : 1.6 (18/02/2013)
- Aucunes dépendances, fonctionne sous tous les environnements

Features

- Spies, Stubs, Mocks
- Fake Timers
- Fake Ajax Requests
- Fake Server
- Sandboxing
- Test assertions / Matchers

Spies

- Permet d'espionner des fonctions

- Nombre d'appels
- Paramètres d'appels
- Contexte d'appel (this)
- Exceptions remontés
- Ordre des appels

- Peut filtrer les appels

- Informations sur les appels Individuels

```
var spy = sinon.spy();

spy("hello");
spy("world");

spy.called          // returns true
spy.calledOnce    // returns false
spy.calledTwice   // returns true

spy.withArgs("hello").called // returns true
spy.withArgs("foo").called  // returns false

spy.getCall(0).args // returns ["hello"]

spy.withArgs("hello").calledBefore(spy.withArgs
("foo")) ; // returns true

spy.calledOn(window) ; // true, default context
```

Stubs

Spies etendus de comportements d'execution

- API des spies
- Plus des méthodes de stubbing

```
var stuff = {  
    work : function() { return "ok";}  
};  
  
// replacing object method with a stub  
var stub = sinon.stub(stuff, "work");  
  
stuff.returns("stuffed");  
  
stub(); // returns "stuffed"  
greeter.greet(); // this works too  
  
stuff.restore(); // original method restored  
stuff.work(); // return "ok" again
```

```
var stub = sinon.stub();  
  
stub.returns("stuffed");  
stub.withArgs("hello").returnsArg(0);  
stub.withArgs("throw").throws("Woups");  
  
stub();           // returns "stuffed"  
stub("dolly");  // returns "stuffed"  
stub("hello");  // returns "hello"  
stub("throw");  // throws Woups  
  
// this work too  
stub.withArgs("hello").calledOnce;  
stub.threw("Woups"); // returns true
```

Mocks / Expectations

Mocker un objet convertie toutes ses methodes en expectations

Une expectation

- étend l'api stub et spy
- dispose de méthodes de vérification d'appels

Mock.restore() permet de retrouver le comportement initial

```
test("mock should mock", function() {
    var greeter = {
        greet : function(name) {
            console.log("hello " + name);
        }
    };
    var mock = sinon.mock(greeter);
    // expects greet to be called once with
    'john'
    mock.expects("greet").withArgs("john").once();

    // mock.verify() // would raise
    greeter.greet("john");

    mock.verify(); // will pass
});
```

Fake Timers

Emule un mode synchrone pour les timers

Permet de tester du code asynchrone plus simplement

Sous le capot : remplace les méthodes setTimeout et setInterval

Peut aussi mocker la classe Date, mais attention (moment.js...)

```
var clock = sinon.useFakeTimers();
var spy = sinon.spy();

window.setTimeout(spy, 150);
clock.tick(149);
spy.called; // returns false

clock.tick(1);
spy.called; // returns true

// with setInterval :

var clock = sinon.useFakeTimers();
var spy = sinon.spy();

window.setInterval(spy, 30);
clock.tick(100);

spy.calledThrice; // returns true
```

Fake XHR

Remplace l'object XHR par une implementation synchrone.

Permet de répondre aux requêtes ajax sans serveur et de manière bloquante.

Possibilité de choisir les requêtes que l'on veut mocker (via des filtres)

```
var xhr = sinon.useFakeXMLHttpRequest();
var requests = [];
xhr.onCreate = function (request) {
  requests.push(request);
};

var spy = sinon.spy();
$.get("/").then(spy);

requests[0].respond(200, {}, "some
text");

spy.calledOnce; // returns true
spy.withArgs("some text").calledOnce; // returns true
```

Fake server

API de haut niveau pour manipuler les Fake XHR

Permet de répondre aux requêtes par url et méthode

Réponses forcément 'statiques', pas de logique possible au niveau du serveur



```
// mock XHR with the fake server
var server = sinon.fakeServer.create();

server.respondWith("GET", "/", // method & url
  [200, // response code
   { "Content-Type": "text/plain" }, // headers
   "fake response"] // response content
);

var spy = sinon.spy();
$.get("/").then(spy);

spy.called; // returns false

server.respond(); // respond to received requests

spy.calledOnce; // returns true
spy.withArgs("fake response").calledOnce; // returns true
```

Sandboxing

Permet de faciliter la gestion des features de faking

Les classes de test heritent de sandbox.

```
var sandbox = sinon.sandbox.create();

sandbox.useFakeTimers();
sandbox.useFakeServer();
// all code from there uses fake timers and
server.

// [...]
var spy = sandbox.spy(someObject, "method");

sandbox.restore();
// original behaviour is now restored
// sandbox spies are also removed
```

Adapters

Permet d'utiliser certaines ou toutes les fonctionnalités de sinon dans d'autres frameworks :

- Jasmine
- Chai
- QUnit
- NodeUnit
- ...

Exemple d'utilisation de sinon-qunit

```
// using sinon in qunit test
test("sinon adapter should work", function () {

    var stub = this.stub();
    stub.returns("stuffed");

    var result = stub();

    equal(result, "stuffed", "result value should be stubbed");
    ok(stub.calledOnce, "spy should have been called once");
});
```

En quelques mots...

Terriblement puissant

Aucune logique possible dans les stubs ou avec les fakeServer

Play (very) well with others

Les stubs ne font pas de proxification.

API très complète, fluente et lisible

Les spies et stubs deviennent vite indispensable dans les tests

Attention au faking de Date.

Il y avait moins de vert que de rouge alors que sinon js est juste génial, donc voilà, je rajoute des trucs

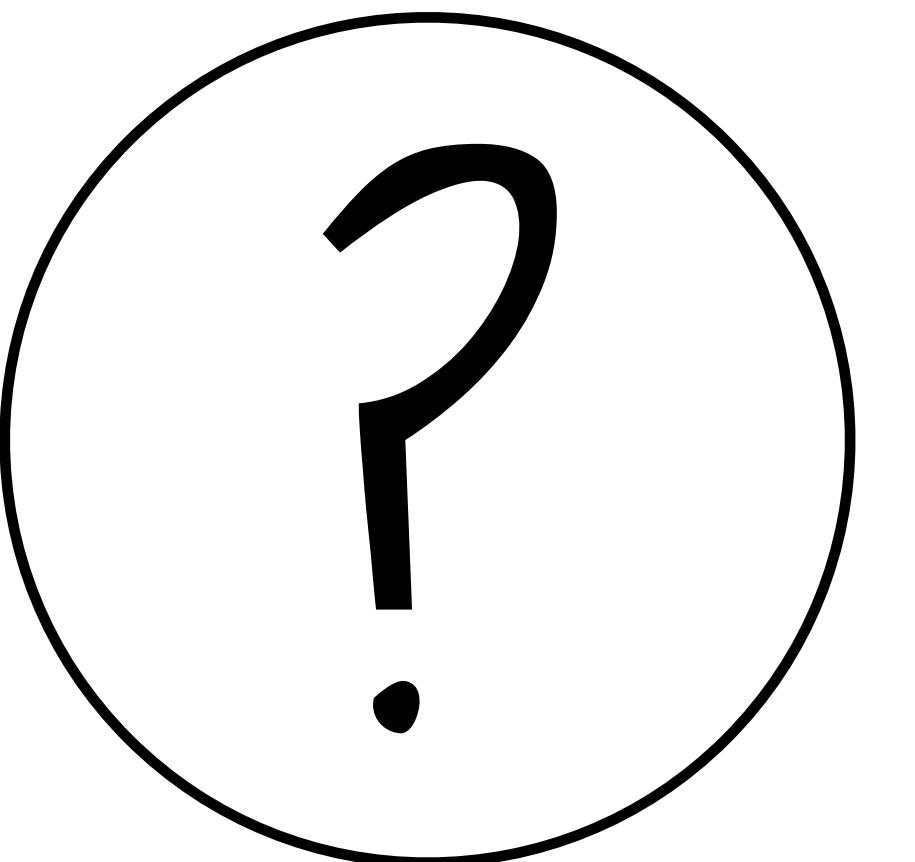
Le faking d'ajax fonctionne moyennement sous IE6/7 (étonnement..)

Sinon.JS

Demo time



27 au 29 mars 2013



Karma Testacular

कर्म

Spectacular Test Runner for JavaScript



DEVOXXX
FRANCE

27 au 29 mars 2013

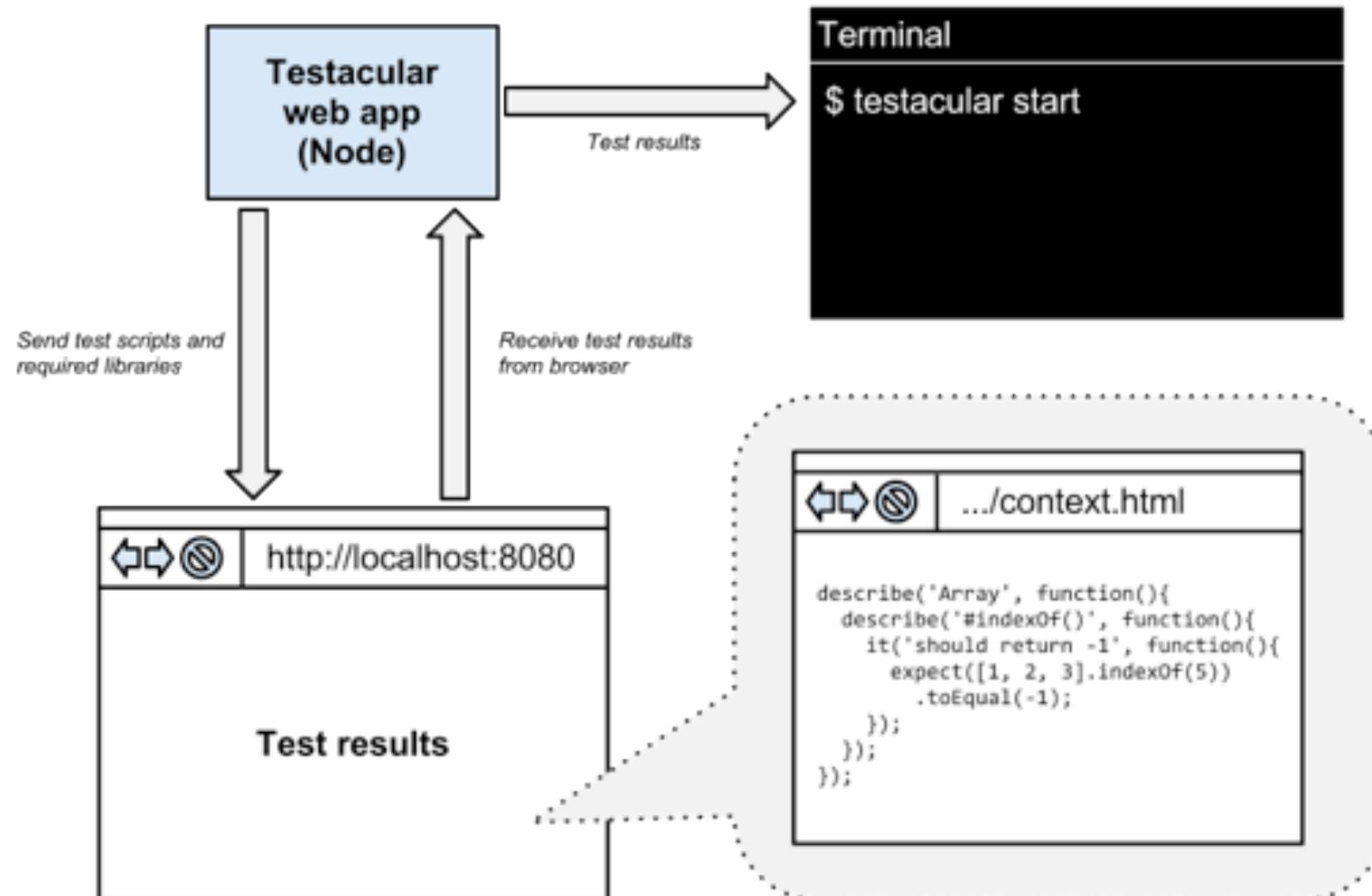
KarmaJs

- Lanceur de test multi-navigateur
 - Tourne sur node.js
 - Permet de faire du test unitaire et du test dit "end to end".
 - Développé par l'équipe d'Angular.js
-
- v0.8
 - MIT Licence



Karma

Architecture



Karma

Est capable de gérer plusieurs navigateurs simultanément notamment

- Chrome
- Firefox
- Opera
- PhantomJS
- ~~Internet Explorer~~



KarmaJS

- Supporte plusieurs systèmes de tests
 - Mocha
 - Jasmine
 - Qunit
 - Angular Scenario

Karma

Fichier de configuration

```
basePath = '';
files = [
  ANGULAR_SCENARIO,
  ANGULAR_SCENARIO_ADAPTER,
  '*.coffee'
];
reporter = 'progress';
port = 8080;
runnerPort = 9100;
colors = true;
logLevel = LOG_DEBUG;
autoWatch = true;
browsers = ['Chrome'];
singleRun = false;
proxies = {
  '/': 'http://localhost:4516/'
}
urlRoot = '/karma/';
```

Karma

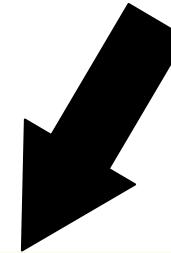
Angular Scénario exemple :

```
it 'should show the title', ->
  browser().navigateTo '/serpodile'
  expect(element('title').text()).toBe 'Serpodile'
```

Karma

Angular Scénario :

- Fonctionne uniquement avec une application Angular.js
- Pas de gestion de callback (Yeepeee !!!)
- Attention tout élément est une Future.



```
expect(element('title').text()).toBe 'Devoxx'
```

Demo Time :

Karma

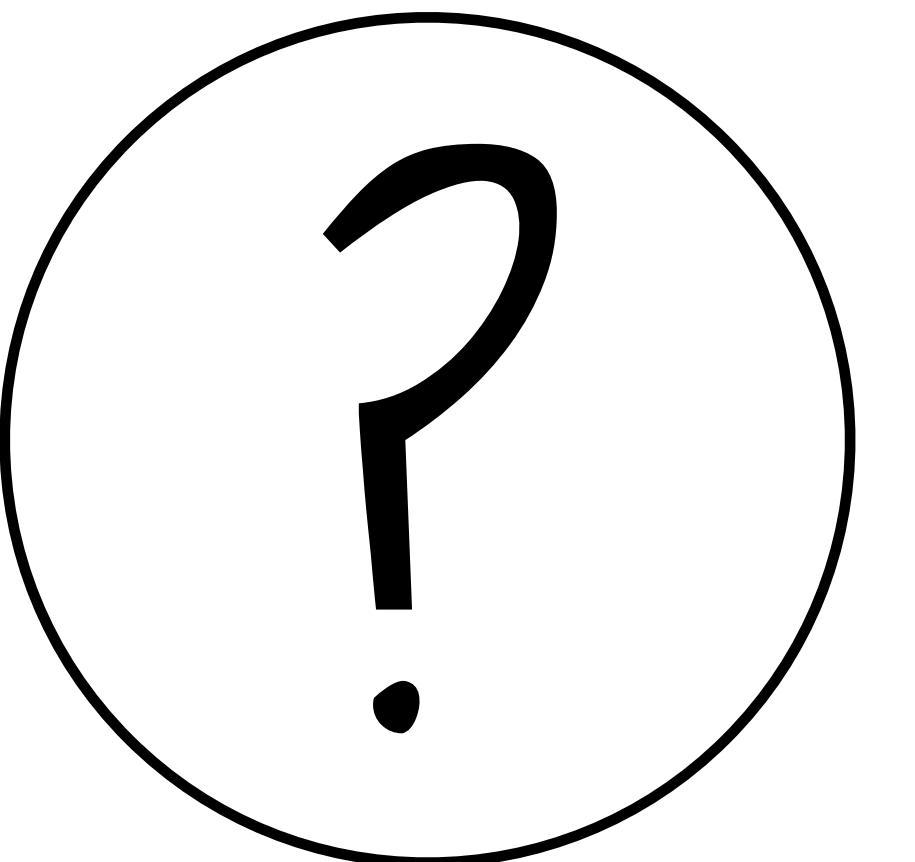
कर्म



27 au 29 mars 2013

Karma

- Multi-Navigateur
 - Angular Scénario
 - Rapide
 - Watch de fichier
-
- End 2 End testing en angular uniquement
 - Manque de documentation
 - Nécessite que les navigateurs soient installés
 - Fichier de configuration verbeux



Chai.js



Chai Assertion Library

<http://chaijs.com/>



27 au 29 mars 2013

chai.js

- Bibliothèque d'assertion de test
 - Permet 3 styles différents d'assertions
 - Tourne sur node.js, ou dans un navigateur
-
- 1.5.0
 - MIT Licence

Chai.js

```
var expect = chai.expect;  
  
expect(devoxx).to.be.a('string');  
expect(devoxx).to.equal('Awesome');  
expect(devoxx).to.have.length(7);
```

Assert style :

```
var assert = chai.assert;  
  
assert.typeOf(devoxx, 'string');  
assert.equal(devoxx, 'Awesome');  
assert.lengthOf(devoxx, 7)
```

: Expect style

Should style :

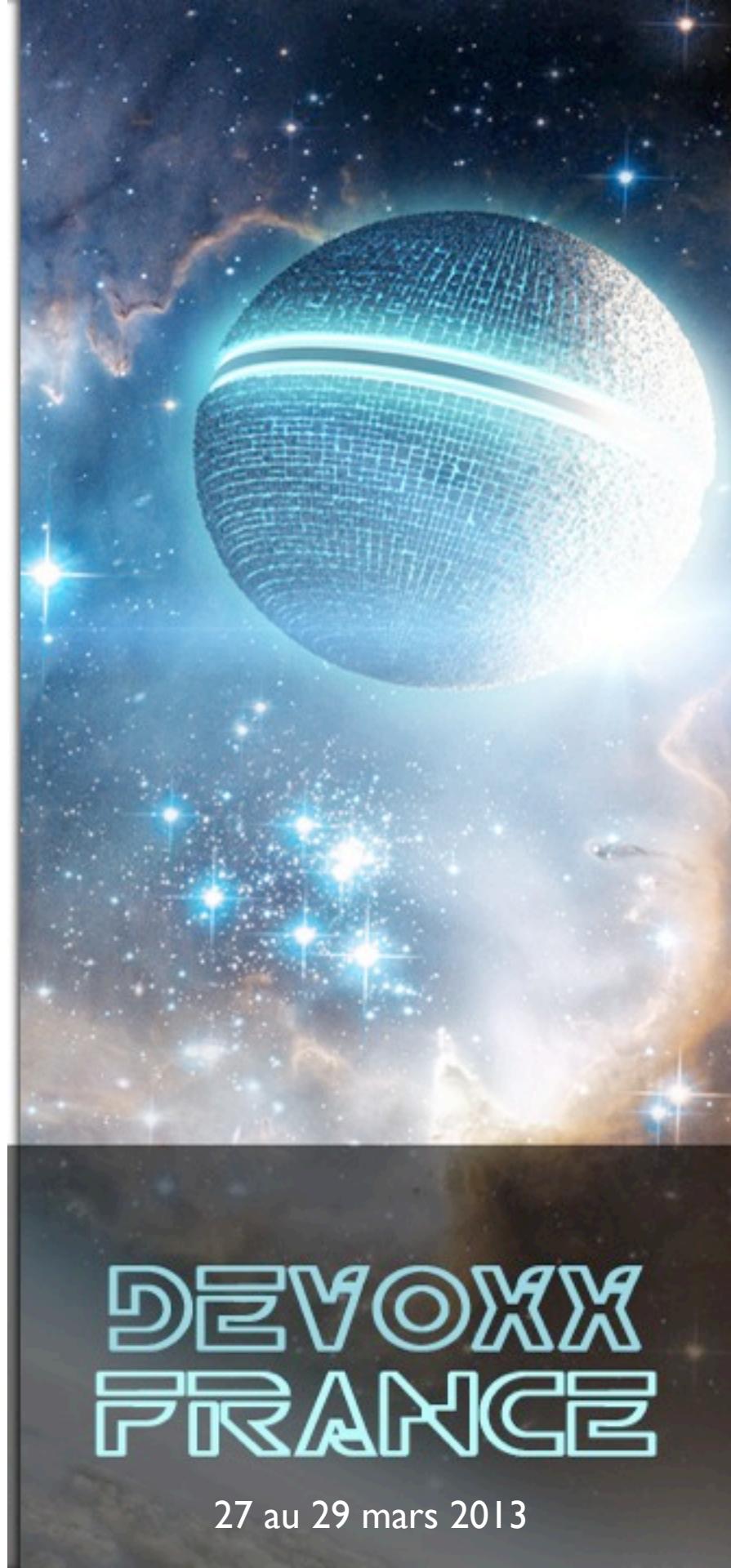
```
chai.should();  
  
devoxx.should.be.a('string');  
devoxx.should.equal('Awesome');  
devoxx.should.have.length(7);  
should.not.exist(undefined);
```

Mocha



simple, flexible, fun

simple, flexible, fun JavaScript test
framework



DEVOXXX
FRANCE

27 au 29 mars 2013

Mocha

- Framework de test
 - Tourne sur Node ou dans un navigateur
 - Permet les tests synchrones ou asynchrones
 - Out of the box s'intègre avec Jenkins, TeamCity, Junit etc..
 - Compatible avec chai.js
-
- 0.4.12
 - MIT Licence

Exemple de test mocha

```
assert = require('assert')

describe 'mocha example', ->
  it 'should uppercase the 1st letter', ->
    assert.equal('Foo', uppcaser('foo'))

  it 'should render uppercase even if its already done', ->
    assert.equal('Foo', uppcaser('Foo'))
```

Demo Time :

Mocha

+ Chai.js



simple, flexible, fun



27 au 29 mars 2013

Mocha + chai.js

- Ultra rapide
- Fluent API
- Code de tests très lisible
- Aucune émulation de navigateurs (scénario)
- Intégration dans un "build" java, non trivial
- `should` assertion intrusive



27 au 29 mars 2013



A browser / node.js javascript testing toolkit

<http://busterjs.com/>

<http://github.com/busterjs>



Buster.js

- Test runner avec quelques fonctionnalités originales
- Encore en version Beta
- Modes de lancement multiples :
 - Module node
 - Capture de navigateur (à la jsTestDriver)
 - Serveur statique intégré
 - Directement dans le navigateur (experimental)
 - Headless avec phantomJS (pas encore implementé)
- Permet de gerer des contextes de test
- Embarque sinon.js

Tests

Deux modes de rédaction pour les tests :

Format xUnit

```
buster.testCase("project.Stuff", {  
  
    setUp: function () {  
        this.stuff = new Stuff();  
    },  
  
    "my stuff should exists" : function ()  
    {  
        assert.defined(this.stuff);  
    },  
  
    "my stuff should be fluffy" :  
    function() {  
        assert(this.stuff.isFluffy());  
    }  
})
```

Format BDD

```
describe("my stuff", function () {  
  
    before(function() {  
        this.stuff = new Stuff();  
    });  
  
    it("should exists", function () {  
        expect(this.stuff).toBeDefined();  
    });  
  
    it("should be fluffy", function()  
    {  
        expect(this.stuff.isFluffy()).toBe(true);  
    });  
})
```

Nested test cases

Unit tests à la sauce BDD...

```
buster.testCase("project.Stuff", {  
  
    setUp: function () {  
        this.stuff = new Stuff();  
    },  
  
    "new stuff" : {  
        "is working" : function() {  
            assert(this.stuff.working());  
        }  
    },  
  
    "broken stuff" : {  
        setUp : function() {  
            this.stuff.broken = true;  
        },  
  
        "is not working" : function() {  
            refute(this.stuff.working());  
        }  
    }  
});
```



Gestion de configs de tests

- descriptif du context de test
- gestion d'extensions
- aucun html requis
- exposition de resources
- proxification de resources

```
config["My tests"] = {  
    env: "browser",           // or "node"  
    extensions: [require("buster-coverage")], // extensions  
    "buster-coverage": {  
        outputDirectory: "coverage_reports" // any plugin config  
    },  
    rootPath: "../",  
    libs : [                  // list of libs to use for tests  
        "lib/jquery.js", "lib/underscore.js"  
    ],  
    sources: [                // list of source files  
        "sources/**/*.js" // Glob patterns supported  
    ],  
    tests: [                  // List of test files.  
        "test/*-test.js"  
    ],  
    resources: [              // list of server resources available to tests.  
        {  
            path: "/todo-items", // proxy resource to remote server  
            backend: "http://localhost:8000/todo/todo-items"  
        },  
        {  
            path: "/user.json", // mocked resource  
            content: JSON.stringify({ id: 1, name: "Christian" }),  
            headers: { "Content-Type": "application/json" }  
        }  
    ]  
};
```

Les plus

refute remplace assert.not[...]

```
assert.IsNotNull(); // n'existe pas  
refute.IsNotNull();
```

Prérequis de test

Désactivation de tests

```
buster.testCase("projet.deferred", {  
    "this test will be executed": function () {  
        assert(true);  
    },  
    "// this test will not launch": function () {  
        assert(false);  
    }  
});
```

```
buster.testCase("My thing", {  
    requiresSupportFor: {  
        "touch events":  
            typeof(document.body.ontouchstart) != "undefined",  
        "XHR": typeof(XMLHttpRequest) != "undefined"  
    },  
    "touch events should trigger an ajax call": function () {  
        // ...  
    }  
});
```

En quelques mots

Encore très jeune (beta 0.7)

Beaucoup de features pas encore là

- Window support
- Headless testing
- Custom “test bed”

Déjà plus de fonctionnalités que beaucoup de “concurrents”

Pas mal de nouvelles idées

Semble très prometteur

JSCover

Code coverage tool for javascript

<http://tntim96.github.com/JSCover>
<http://github.com/tntim96/JSCover>



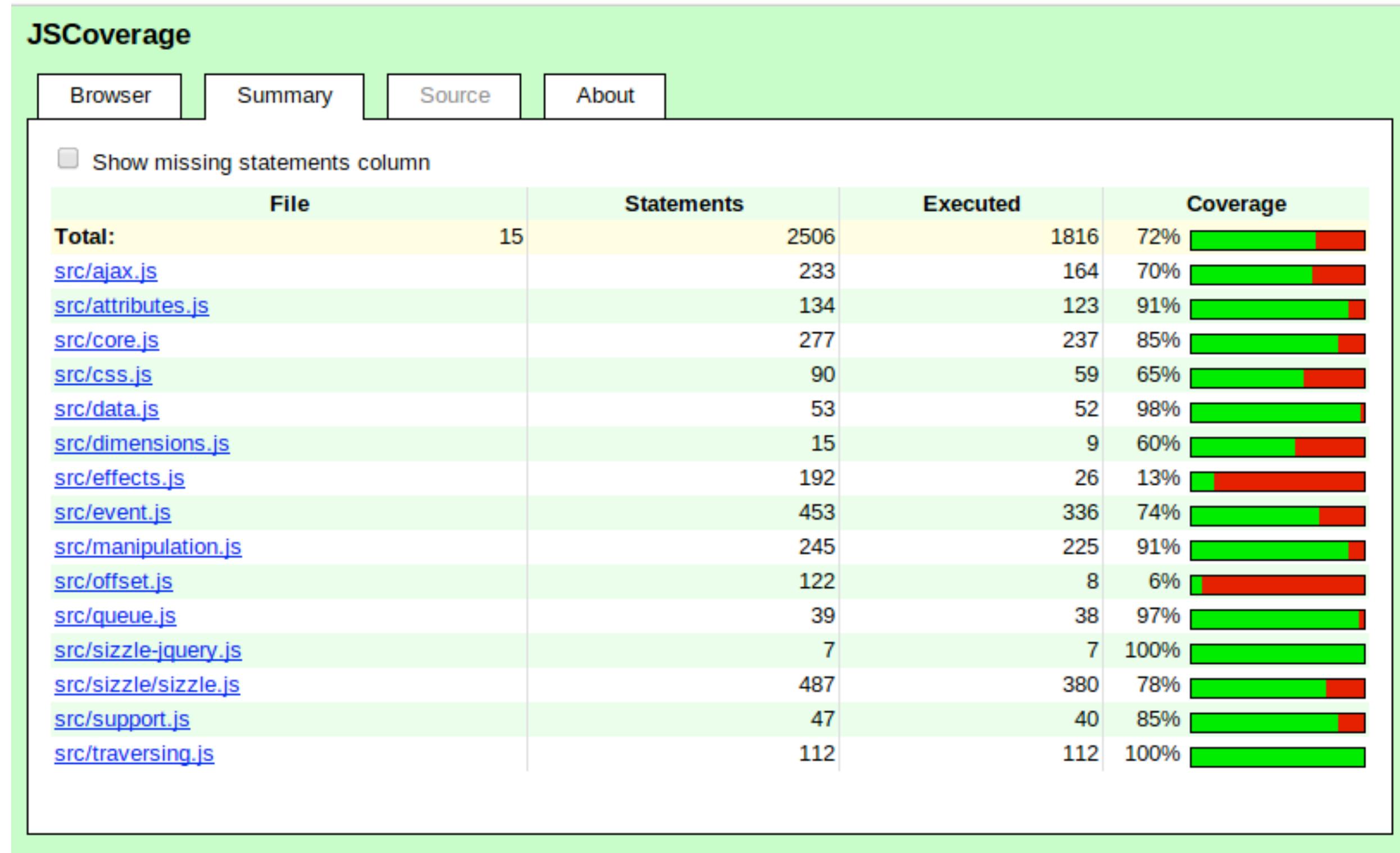
**DEVOXXX
FRANCE**

27 au 29 mars 2013

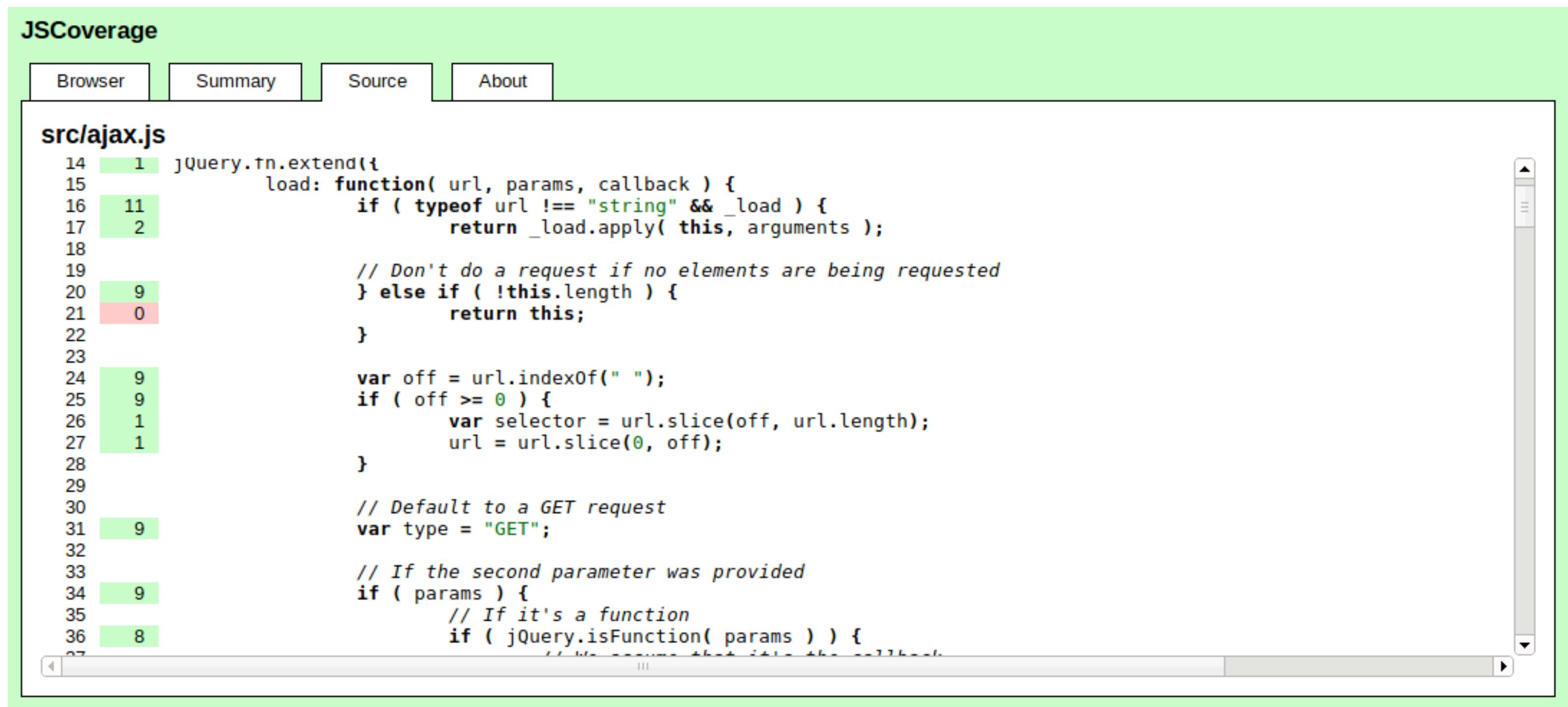
JSCover

- Outil de couverture de code javascript
- Réécriture Java de JsCoverage (c++)
- Deux modes d'instrumentation :
 - build time
 - mode proxy
- Export au format Cobertura (integration jenkins)

Résultat



Vue de détail



JSCover

- Fait ce qu'il est sensé faire
- Développeur actif
- Résultats intégrable dans Jenkins
- Intégration dans le cycle de test délicat dans les deux modes d'instrumentation.
- jar brut, pas de projet maven pour le moment

Plato

Javascript source complexity visualizer

<http://github.com/jsoverson/plato>



Plato

Overview

JavaScript Source Analysis

Summary

Total SLOC [i](#)

6764

Maintainability [i](#)

Average Maintainability [i](#)

70.18



Lines of code [i](#)



Details par fichier

event.js

Maintainability [i](#)

59.76

Difficulty [i](#)

130.60

Estimated # Bugs [i](#)

7.98

SLOC/LSLOC [i](#)

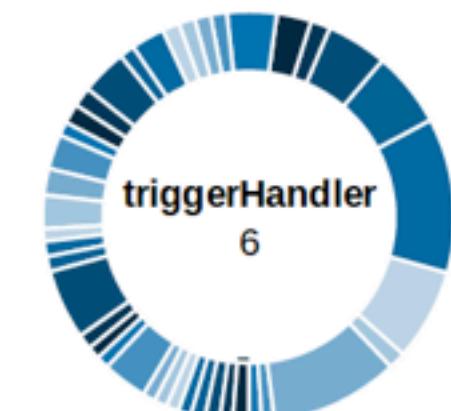
822 / 452

Function weight

By Complexity [i](#)



By SLOC [i](#)



Detail des erreur jsLint

```
429     }
430
431     function filter(e) {
432
433       Complexity : 3
434       Length : 26
435       Difficulty : 8.75
436       Est # bugs : 0.03
437
438       props: "char charCode key keyCode".split(" "),
439       [filter]: function( event, original ) {
440
441         // Add which for key events
442         if ( event.which == null ) {
443
444           Column: 30 "Use '===' to compare with 'null'."
445           event.which = original.charCode != null ? original.charCode : original
446
447           Column: 49 "Use '!==' to compare with 'null'."
448           }
449         }
450       },
451
452       mouseHooks: {
453         props: "button buttons clientX clientY offsetX offsetY pageX pageY screenX screenY".split(" "),
454         [filter]: function( event, original ) {
455           var eventDoc, doc, body,
456             button = original.button;
457
458           // Calculate pageX/Y if missing and clientX/Y available
459           if ( event.pageX == null && original.clientX != null ) {
```

Plato

- Installation facile (npm)
- Utilisation facile
- Dashboard agréable
- Rapports uniquements sous forme de vue html et json (pas d'export possible)
- Et donc pas d'integration dans un CI possible

Wrap Up



DEVOXXX
FRANCE

27 au 29 mars 2013

Les oubliés

- JS Test Driver
- Selenium (fluentlenium)
- HtmlUnit
- Jasmine
- Istanbul

Wrap Up

Pour du tests unitaire :

Mocha + chai.js

Mocha + QUnit + Sinon.js (Mock, Spy, Stubs)

Jasmine ou QUnit (browser needed)

Wrap Up

Pour du tests d'interface :

Zombie.js (Si on favorise l'interface d'écriture des tests)

Casper.js (Si on favorise la fiabilité)

Karma (pour du angular.js)

Wrap Up

A surveiller :

Buster.js

Zombie.js downfall ?



MERCI !



DEVOXXX
FRANCE

27 au 29 mars 2013