Headless browsers

No head
No pain

Who ?

- Nicolas Chambrier

  Twitter: **@naholyr**
  Blog: **http://naholyr.fr**
  Mail: **naholyr@gmail.com**

- Consultant chez Clever Age
  Une SSII qu'elle est bien :)

- Bientôt développeur chez Dijiwan avec @DavidBruant et @oncletom pour faire du Node.JS (entre autres)

What ? Why ?

Quoi ?

- Un "vrai" navigateur, mais sans fenêtre
- Exécute le JS, gère les intéractions...

Pourquoi ?

- Automatiser certaines tâches:
  - Copie d'écran ?
  - Tests fonctionnels
  - Autres (robots)

Présentation

Les forces en présence

Zombie.js

- Module Node.JS
- Navigateur "headless" "full-stack"
- Moteur WebKit ? NON ! → implémentation maison !
- API d'intéraction simplifiée

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

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

  // Fill email, password and submit form
  browser.fill("email", "zombie@underworld.dead")
  .fill("password", "eat-the-living")
  .pressButton("Sign Me Up!", function () {

    // Form submitted, new page loaded.
    assert.ok(browser.success);
    assert.equal(browser.text("title"), "Welcome To Brains Depot");

  });

});
```

PhantomJS

- Application standalone
- Navigateur "headless" "full-stack"
- Moteur WebKit + JavaScriptCore
- Note: Effectue réellement le rendu

```
var page = new WebPage(),
    url = 'http://lite.yelp.com/search?find_desc=pizza&find_loc=94040&find_submit=Search';

page.open(url, function (status) {
    if (status !== 'success') {
        console.log('Unable to access network');
    } else {
        var results = page.evaluate(function() {
            var list = document.querySelectorAll('span.address'), pizza = [], i;
            for (i = 0; i < list.length; i++) {
                pizza.push(list[i].innerText);
            }
            return pizza;
        });
        console.log(results.join('\n'));
    }
    phantom.exit();
});
```

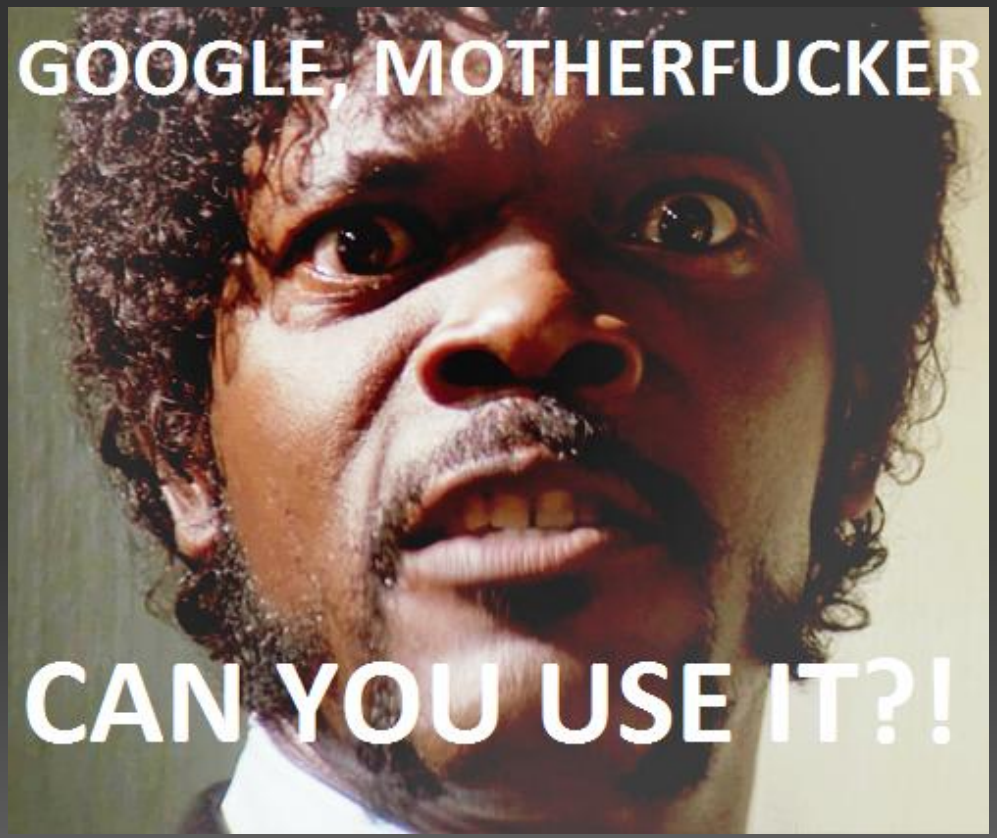
CasperJS

- Surcouche de PhantomJS:
  - API simplifiée
  - Ajout d'une API de test

```
var links = [];
var casper = require('casper').create();

function getLinks() {
    var links = document.querySelectorAll('h3.r a');
    return Array.prototype.map.call(links, function (e) {
        return e.getAttribute('href');
    });
}

casper.start('http://google.fr/', function () {
    // search for 'casperjs' from google form
    this.fill('form[action="/search"]', {
        q: 'casperjs'
    }, true);
});

casper.then(function () {
    // aggregate results for the 'casperjs' search
    links = this.evaluate(getLinks);
    // now search for 'phantomjs' by filling the form again
    this.fill('form[action="/search"]', {
        q: 'phantomjs'
    }, true);
});

casper.then(function () {
    // aggregate results for the 'phantomjs' search
    links = links.concat(this.evaluate(getLinks));
});

casper.run(function () {
    // echo results in some pretty fashion
    this.echo(links.length + ' links found:');
    this.echo(' - ' + links.join('\n - ')).exit();
});
```

Des alternatives ?

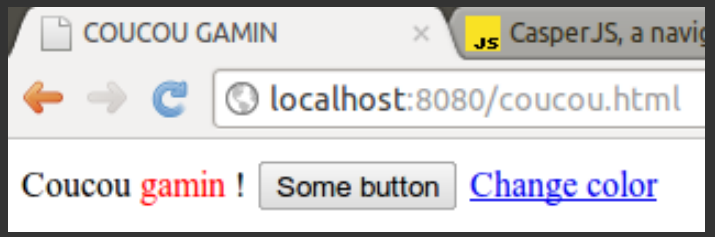
Oui, comme XBrowser (.Net), sans doute plein d'autres...



Tests

Fight!

La page à tester



- Une page moisie avec:
  - Des couleurs qui changent sur un setTimeout
  - Un bouton (qui fait des trucs) pas dans un formulaire
  - Un lien (qui fait d'autres trucs)
  - Des scripts en inline bien dégueulasses

- Des tests côté client avec Jasmine

Zombie.js

Exemples de tests:

```
assert.equal(browser.text('title'), 'COUCOU TOTO');
assert.equal(browser.text('#coucou'), 'toto');
browser.wait(3000, function () {
  assert.equal(browser.evaluate('document.title'), 'COUCOU GAMIN');
  assert.equal(browser.text('#coucou'), 'gamin');
```

Note: pas de rendu de la page!

```
console.log('Color:', browser.evaluate('getComputedStyle(document.getElementById("coucou")).color'));
// null :'(
assert.equal(browser.query('#style').href, 'red.css');
browser.clickLink('Change color', function () {
  assert.equal(browser.query('#style').href, 'blue.css');
});
```

Attention à setTimeout!

```
(new Browser()).visit("http://localhost:8080/coucou.html", function (err, browser, status) {
  assert.equal(browser.text('title'), 'COUCOU TOTO');
  browser.wait(3000, function () {
    assert.equal(browser.evaluate('document.title'), 'COUCOU GAMIN');
  });
});
```

Pourquoi ? Parce qu'il n'y a pas un "vrai navigateur" qui tourne derrière.

PhantomJS

Bordel! Où est assert ???

```
var assert = {
  "errors": 0,
  "tests": 0,
  "equal": function assert_equal (value, expected) {
    assert.tests++;
    if (value != expected) {
      console.error('[!] ' + value + ' ≠ ' + expected);
      assert.errors++;
    }
  },
  "ok": function assert_ok (value) {
    assert.tests++;
    if (!value) {
      console.log('[!] ' + value + ' is falsey');
      assert.errors++;
    }
  }
};
```

ATTENTION: exception = blocage silencieux!

Pourquoi ? On n'appelle jamais phantom.exit(), par contre pourquoi rien n'est affiché ? Aucune idée :(

PhantomJS

Utilisation de setTimeout OK (on termine avec phantom.exit):

```
assert.equal(page.evaluate(function () { return document.querySelector('title').innerHTML; }), 'COUCOU TOTO');
assert.equal(page.evaluate(function () { return document.querySelector('#coucou').innerHTML; }), 'toto');
setTimeout(function () {
    assert.equal(page.evaluate(function () { return document.title; }), 'COUCOU GAMIN');
    assert.equal(page.evaluate(function () { return document.querySelector('#coucou').innerHTML; }), 'gamin');
```

Lui fait le rendu "réel":

```
// Good boys compute styles
var color = page.evaluate(function () { return getComputedStyle(document.getElementById("coucou")).color; });
console.log('Color: ' + color);
```

Note: attention au scope!

Le callback dans evaluate()
s'exécute dans un autre contexte

```
var x = 0;
page.evaluate(function () {
    x = 1;
});
console.log(x); // 0
```

Attention à "document" dans le contexte global: ce n'est pas votre ami !

CasperJS

Phantom, en mieux :)

```
// Timers
casper.then(function () {
  console.log('Testing timers…');
  this.test.assertTitle('COUCOU TOTO');
  this.test.assertEval(function () { return __utils__.findOne('#coucou').innerHTML == 'toto'; }, '
  this.wait(3000, function () {
    this.test.assertEvalEquals(function () { return document.title; }, 'COUCOU GAMIN', 'title = "C
    this.test.assertEvalEquals(function () { return document.querySelector('#coucou').innerHTML; }
  });
});

casper.run(function() {
  this.test.renderResults(true); // Ah! 4 tests \o/
});
```

```
PASS undefined
PASS #coucou = "toto"
PASS title = "COUCOU GAMIN"
PASS #coucou = "gamin"
PASS 4 tests executed, 4 passed, 0 failed.
```

Attention à setTimeout! Utiliser Casper#wait()

Cause: il s'arrête seul après Casper#run().

Conclusion

Kill 'em all !

Zombie.js

Pour:

- Léger et rapide
- Module Node.JS
  = tout l'écosystème Node.JS
  = n'importe quel lib de test Node (et il y en a baucoup!)
- Contexte d'exécution plus simple à prendre en main

Contre:

- Ne calcule pas les styles CSS
- Globalement moins fiable car pas de vrai "browser" qui tourne derrière.

PhantomJS

Pour:

- Plus complet & plus fiable que Zombie.js
- Calcule réellement le rendu de la page

Contre:

- Difficile à installer (on n'est pas des manchots non plus)
- Peut être plus complexe à maîtriser
- Aucune API de test :(

CasperJS

Pour:

- Tous les avantages de PhantomJS!
- API de test plus complète
- API vachement trop plus mieux de manière générale
- Maintenu par @n1k0 (mon idole)

Contre:

- Encore plus difficile à installer: dépendance à Python + PhantomJS

Mon avis

- **Zombie pour des tests simples.**

  S'il suffit, c'est le meilleur choix pour ses faibles dépendances.

- **Casper pour des besoins plus avancés (= besoin du rendu).**

  Phantom seul est vraiment vite désagréable à utiliser.

Goodbye

Questions ?