Automated Testing in JavaScript

@WebDev Party #1

高見龍

photo by quaziefoto


WEBDEV
PARTY
SHARE AND RESEARCH

You can get this slide on

www.eddie.com.tw/slides

高見龍

a.k.a Eddie or Aquarianboy

- Live and work in Taipei, Taiwan.
- Serving in my own little tiny company.
- Flash / AS3 / Ruby / Rails / Python programming for living.
- A little bit Objective-C for personal interests.
- Technical Education and Consultant.
- PTT Flash BM (since 2007/4).
- Adobe Certificated Flash Developer (Since 2006/7).
- Linux Professional Institute Certification (Since 2005/3).

photo by Eddie

anyone do test?

people don’t test..

In your company, who does these tests?

> does they really know how to do test?

not just launch a browser, and click.. click.. click..

how do you developers test?

Step1:

Write some code!

Step 2:

Ctrl + S = Save!

Step 3:
Alt + Tab

Step 4:
F5

Step 5:
and.. repeat the Step 1

Browser plugins or console!

LiveReload!

Selenium

What’s Automated Testing?

> Automated testing means that tests are run every single time a file is saved.

Test Level 1:
by eyes, hands, and your instinct.

Test Level 2:
ALERT, or Console.log()

Test Level 3:
Test Runner

Test Level 4:
Test Framework

Test Level 5:
Test Framework + Automated Testing Tools

Why test?

Cross-browser issues !!

some browsers just don’t die.

> less bugs == you have more time developing new features.

Why test JavaScript?

cause JS is popular now, and it's getting more and more complicated.

Why test JavaScript automatically?

cause we're Lazzzzzzzzzzzzy!!

What’s Unit Test?

Unit test is a piece of code that tests a piece of production code.

so, unit test might be also Buggy!

> a good unit test should be short and focus on a single behavior of a function/method.

> the heart of a unit test is the assertion.

How can code be testable?

Not every code can be easily tested.

TDD

Twitter-Driven Development

Test-Driven Development

> Test-first development is hard — it's hard because it forward-shifts your confusion.

Test != Debug

xUnit

Test should be fast, and easy to run repeatedly.

test case, test suite, test runner

Spy, Stub, and Mock

FAKE!! something not REALLY!

spy

"Spies are functions that keep track of how and often they were called, and what values were returned."

This is useful in asynchronous and event-driven applications.

Sinon.js

```
it "check if 'strip_tag' method was triggered", ->
  spy = sinon.spy()

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  namecard.bind 'strip_tag', spy

  namecard.trigger 'strip_tag'

  # Expect the spy was called at least once
  expect(spy.called).toBeTruthy()
```

Sinon.js

```
it "check if 'strip_tag' method was triggered", ->
  spy = sinon.spy()

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  namecard.bind 'strip_tag', spy

  namecard.trigger 'strip_tag'

  # Expect the spy was called at least once
  expect(spy.called).toBeTruthy()
```

Sinon.js

```
it "check if 'strip_tag' method was triggered", ->
  spy = sinon.spy()

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  namecard.bind 'strip_tag', spy

  namecard.trigger 'strip_tag'

  # Expect the spy was called at least once
  expect(spy.called).toBeTruthy()
```

Sinon.js

```
it "check if 'strip_tag' method was triggered", ->
  spy = sinon.spy()

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  namecard.bind 'strip_tag', spy

  namecard.trigger 'strip_tag'

  # Expect the spy was called at least once
  expect(spy.called).toBeTruthy()
```

Sinon.js

```
it "check if 'strip_tag' method was triggered", ->
  spy = sinon.spy()

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  namecard.bind 'strip_tag', spy

  namecard.trigger 'strip_tag'

  # Expect the spy was called at least once
  expect(spy.called).toBeTruthy()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  spy = sinon.spy jQuery, 'ajax'

  namecard.save();

  # check Spy was called
  expect(spy).toHaveBeenCalled()

  # Check url property of first argument
  expect(spy.getCall(0).args[0].url).toEqual "/namecard/1"
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  spy = sinon.spy jQuery, 'ajax'

  namecard.save();

  # check Spy was called
  expect(spy).toHaveBeenCalled()

  # Check url property of first argument
  expect(spy.getCall(0).args[0].url).toEqual "/namecard/1"
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  spy = sinon.spy jQuery, 'ajax'

  namecard.save();

  # check Spy was called
  expect(spy).toHaveBeenCalled()

  # Check url property of first argument
  expect(spy.getCall(0).args[0].url).toEqual "/namecard/1"
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  spy = sinon.spy jQuery, 'ajax'

  namecard.save();

  # check Spy was called
  expect(spy).toHaveBeenCalled()

  # Check url property of first argument
  expect(spy.getCall(0).args[0].url).toEqual "/namecard/1"
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->

  namecard = new app.NameCard
    name: 'eddie'
    tel: '0928617687'
    address: 'Taipei, Taiwan'

  spy = sinon.spy jQuery, 'ajax'

  namecard.save();

  # check Spy was called
  expect(spy).toHaveBeenCalled()

  # Check url property of first argument
  expect(spy.getCall(0).args[0].url).toEqual "/namecard/1"
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Sinon.js

```
it "check if ajax method was triggered while saving", ->
server = sinon.fakeServer.create()
spy = sinon.spy()
server.respondWith("GET", "/namecard/1",
      [200, {"Content-Type": "application/json"},
      '{"id":1,"name":"eddie", "tel":"0928617687""}']);

namecard = new NameCard({id: 1})
namecard.bind 'change', spy
namecard.fetch()
server.respond()
# Expect that the spy was called with the new model
expect(spy.called).toBeTruthy()
expect(spy.getCall(0).args[0].attributes).toEqual
  id: 1
  name: "eddie"
  name: "0928617687"

server.restore()
```

Demo

Demo

Browser Console

Demo

QUnit

Demo

Zombie.js with Node.js

ScreenFlow File Edit Actions View Window Help

dl.dropbox.com/u/6931090/Calculator/form.html

賺大錢計算機 首頁 關於

人客，緊來喔!

不要小看這個小小計算機，它是你賺大錢的最好幫手! 想賺海角七億的就快來加入會員吧!

Email:

加入

Demo

Jasmine
in Ruby on Rails 3.1

Demo

Jasmine + Guard + Phantom.js
in Ruby on Rails 3.1

Conclusions

Writing tests is an investment.

not all tests are good!

> If you write bad tests, you might find that you gain none of the benefits, and instead are stuck with a bunch of tests that are time-consuming and hard to maintain.

In test-driven development tests are written as specification before writing production.

> Proper test-driven development ensures that a system will never contain code that is not being executed.

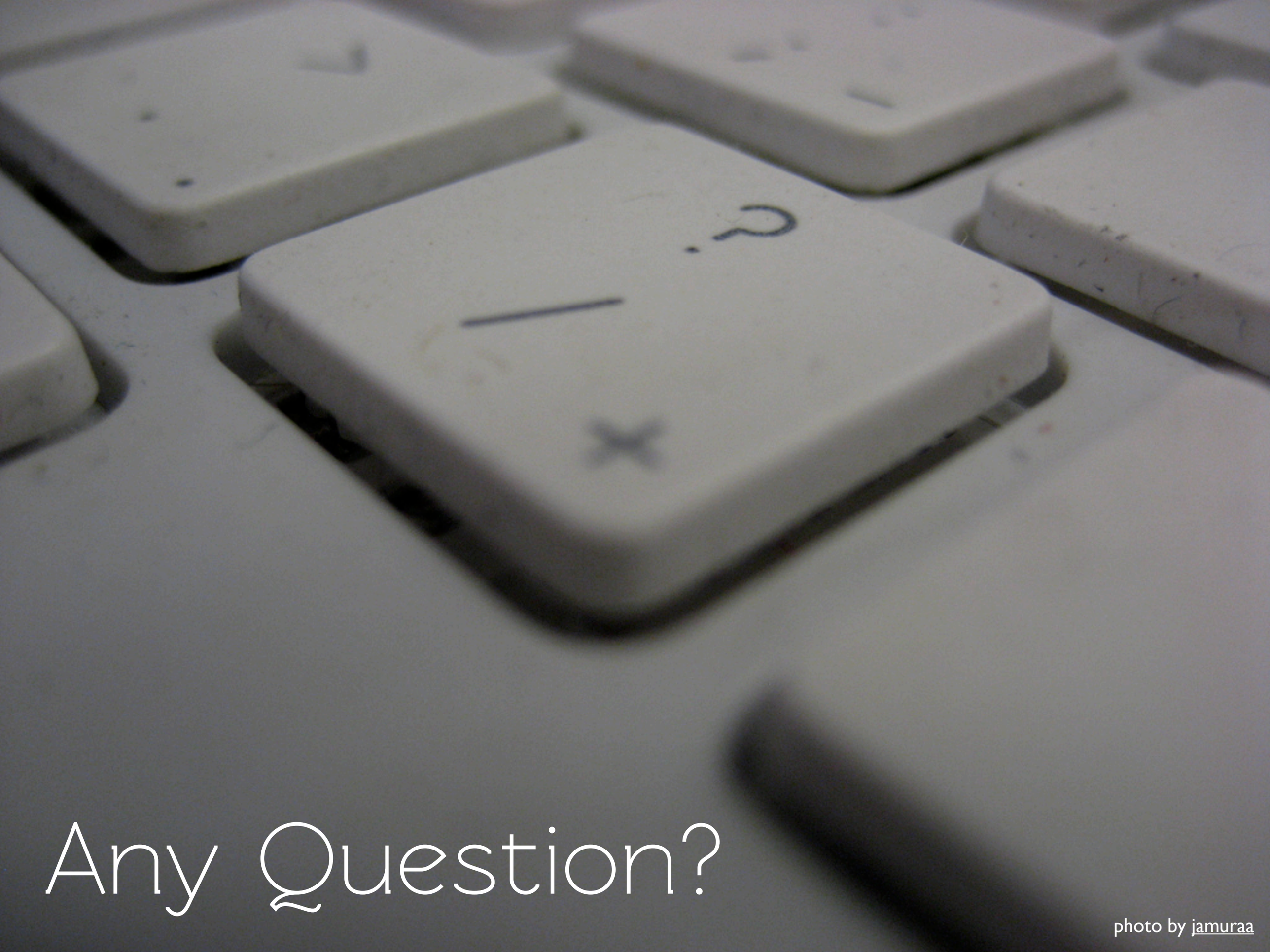
> TDD will not automatically make great designs.

> TDD requires us to think about the results before providing the solution.

Don’t fear hard-coding!

Unit Test = You Need Test!

Just give it a try!



Any Question?

photo by jamuraa

Contacts

高見龍

- Website http://www.eddie.com.tw
- Blog http://blog.eddie.com.tw
- Plurk http://www.plurk.com/aquarianboy
- Facebook http://www.facebook.com/eddiekao
- Google Plus http://www.eddie.com.tw/+
- Twitter https://twitter.com/#!/eddiekao
- Email eddie@digik.com.tw
- Mobile +886-928-617-687

photo by Eddie