

wru

unit tests have never been that easy

Andrea Giammarchi [@WebReflection](#)

Senior SW Engineer [@NOKIA](#)

Why Do You Unit Test ?

- to be sure what I “thought” is what I get
- to avoid regressions per each new behavior
- to better analyze a specific problem
- to provide more robust code
- to convince others my code works
- to cover every single line I wrote

Why Do You Unit Test ?

- to be sure what I “thought” is what I get
- to avoid regressions per each new behavior
- to better analyze a specific problem
- to provide more robust code
- to convince others my code works
- to cover every single line I wrote
- ... just because I feel better knowing whatever crap I wrote passes whatever crap I am testing ...

Why Don't You Unit Test ?

- my deadlines are too tight: I have no time
- I don't want to spend hours for the setup
- I don't know where to start
- I don't think tests are useful


Why Don't You Unit Test ?

- my deadlines are too tight: I have no time
- I don't want to spend hours for the setup
- I don't know where to start
- I don't think tests are useful

- my code is perfect because I am too skilled

Why Don't You Unit Test ?

- my deadlines are too tight: I have no time
- I don't want to spend hours for the setup
- I don't know where to start
- I don't think tests are useful

- my code is perfect because I am too skilled
- I don't give a  about my own code

Are You Sure Your Code Works ?

```
function ofCourseIam_itsSimpleCode(object) {  
    alert(object);  
}
```

How can anything go possibly wrong ?

Are You Sure Your Code Works ?

```
function ofCourseIam_itsSimpleCode(object) {  
    alert(object);  
}
```

How can anything go possibly wrong ?

```
ofCourseIam_itsSimpleCode(Object.create(null));  
// Could not convert JavaScript argument arg 0
```

```
ofCourseIam_itsSimpleCode(Proxy.create({}));  
// Could not convert JavaScript argument arg 0
```

try { } catch (o_O) { } Is Not The Answer

```
function ofCourseIam_itsSimpleCode(object) {  
  try {  
    alert(object);  
  } catch (o_O) {  
    // oh well ... how do I show now  
    // the object that caused problems  
    alert("hey, something unexpected happened!");  
  }  
}
```

try { } catch (o_O) { } Is Not The Answer

Unless strictly necessary for API misbehavior reason, a code surrounded in every place by try catches has never been better, faster, or safer, than any other code tested against as many input as possible.

That “try” should never fail against your logic or who knows what kind of disaster could happen and any empty catch, as any uncaught exception, could open doors for any sort of bug.

try catch is good for unit tests though so that we can improve our code quality and be sure that once everything is “green” our code will be more robust.

OK, Got It, Let Me Try This “wru”

- `git clone git://github.com/WebReflection/wru.git`
- `open wru/test/test.html`

OK, Got It, Let Me Try This “wru”

- `git clone git://github.com/WebReflection/wru.git`
- `open wru/test/test.html`

There Are Errors: 1

test that should pass (1, 0, 0)

async test (1, 0, 0)

test that should fail (1, 1, 0)

test that should throw an error (1, 0, 1)

IT WORKED!

can you believe it ?

Boilerplate Included

The handy part of wru is that you don't need to setup anything ... have a look into the build folder:

For Web:

<https://github.com/WebReflection/wru/blob/master/build/template.html>

For Server:

<https://github.com/WebReflection/wru/blob/master/build/template.js>

What Is wru Good For ?

- gists !!! The fact is a gist does not mean you should not test it. Let anybody else see that it works and improve or change it with a counter proof and without any problem about testing environment
- quick prototyping, so that once you have certain logic to test in place, you can basically copy and paste them if you need/want a different testing environment
- whatever test you want ... the fact is tiny does not mean it cannot be all you truly need

You Really Test What You Need

Many frameworks out there are surely more powerful, probably more semantic, but most of the time these frameworks execute your code in a completely fake, changed, or even worse buggy, environment.

If you want to be sure that *What You Green Is What You Get*, `wru` is the less obtrusive testing framework ever: a single variable you should avoid and nothing else.

This name is **wru**

You Really Test What You Need

Many frameworks out there are surely more powerful, probably more semantic, but most of the time these frameworks execute your code in a completely fake, changed, or even worse buggy, environment.

If you want to be sure that *What You Green Is What You Get*, wru is the less obtrusive testing framework ever: a single variable you should avoid and nothing else.

This name is **wru**



wru

give it a try

<https://github.com/WebReflection/wru>

And Thank You For Listening