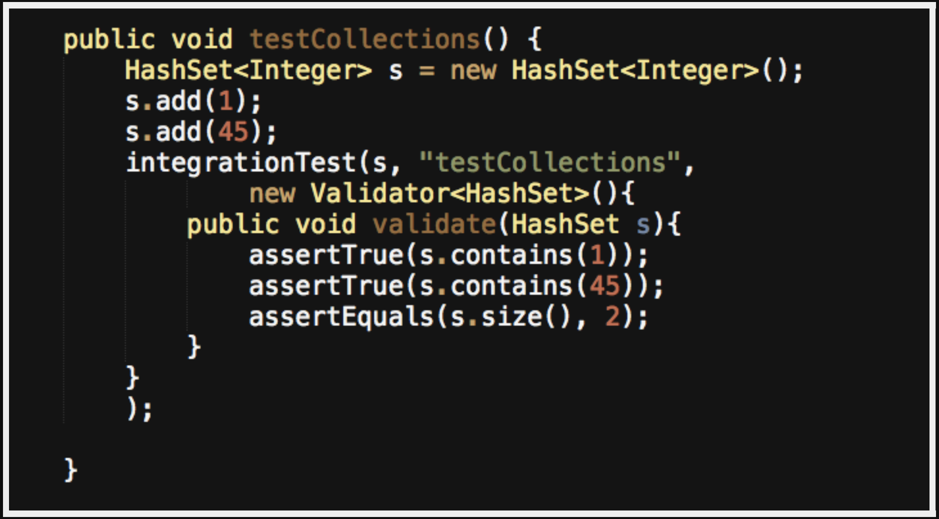
# Test-Driven Development that Feels Great

by Toby Ho **tobyho.com** **smalljs.org** **@airportyh**

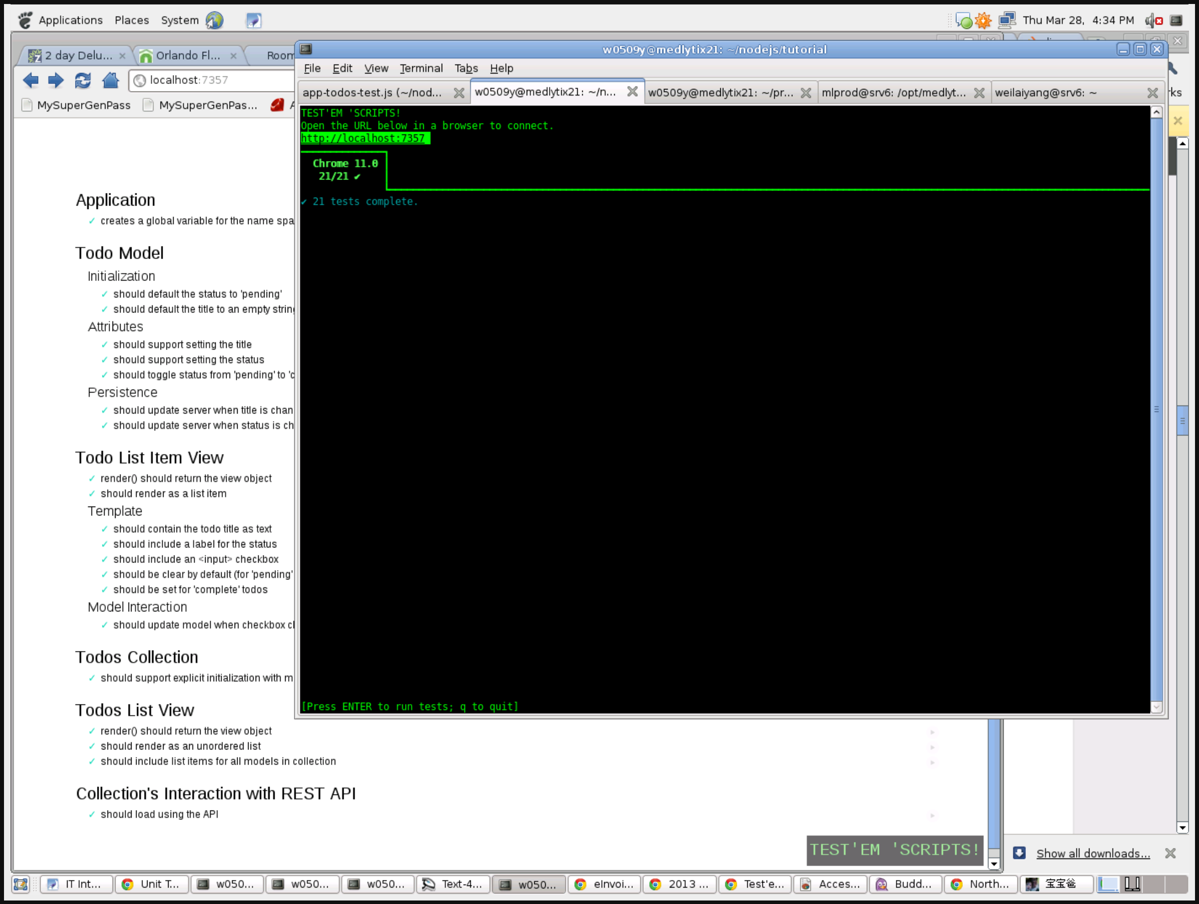
Toby Ho
**tobyho.com**
**smalljs.org**
**@airportyh**

2005


```
public void testCollections() {
    HashSet<Integer> s = new HashSet<Integer>();
    s.add(1);
    s.add(45);
    integrationTest(s, "testCollections",
            new Validator<HashSet>(){
        public void validate(HashSet s){
            assertTrue(s.contains(1));
            assertTrue(s.contains(45));
            assertEquals(s.size(), 2);
        }
    }
    );

}
```

# Got 'Scripts? Test'em



```
TEST'EM 'SCRIPTS!
Open the URL below in a browser to connect.
http://localhost:7357

  Chrome 11.0
   21/21 ✔

✔ 21 tests complete.

[Press ENTER to run tests; q to quit]
```

Application

- ✓ creates a global variable for the name spa

Todo Model

Initialization

- ✓ should default the status to 'pending'
- ✓ should default the title to an empty strin

Attributes

- ✓ should support setting the title
- ✓ should support setting the status
- ✓ should toggle status from 'pending' to 'c

Persistence

- ✓ should update server when title is chan
- ✓ should update server when status is ch

Todo List Item View

- ✓ render() should return the view object
- ✓ should render as a list item

Template

- ✓ should contain the todo title as text
- ✓ should include a label for the status
- ✓ should include an <input> checkbox
- ✓ should be clear by default (for 'pending'
- ✓ should be set for 'complete' todos

Model Interaction

- ✓ should update model when checkbox cl

Todos Collection

- ✓ should support explicit initialization with m

Todos List View

- ✓ render() should return the view object
- ✓ should render as an unordered list
- ✓ should include list items for all models in collection

Collection's Interaction with REST API

- ✓ should load using the API

TEST'EM 'SCRIPTS!
Applications Places System
Thu Mar 28, 4:34 PM
w0509y@medlytix21: ~/nodejs/tutorial
File Edit View Terminal Tabs Help
app-todos-test.js (~/nod...
w0509y@medlytix21: ~/n...
w0509y@medlytix21: ~/pr...
mlprod@srv6: /opt/medlyt...
weilaiyang@srv6: ~
localhost:7357
Show all downloads...

## Our Agenda

- TDD Quick Start
- Why TDD?
- Getting Faster and Better
- Main Event: Man vs Machine!

## TDD

### Quick Start!!

## How to do TDD

1. Write a test
2. See it fail
3. Write some code
4. See it pass
5. Refactor as see fit (optional)
6. Start again at 1

## Test Framework: Mocha

## What A Test Looks Like: TDD

```
test("adds 1 and 2", function(){
  assert.equal(add(1, 2), 3);
});
```

## What A Test Looks Like: BDD

```
describe("hello test", function(){
  it('adds 1 and 2', function(){
    expect(add(1, 2)).to.equal(3);
  });
});
```

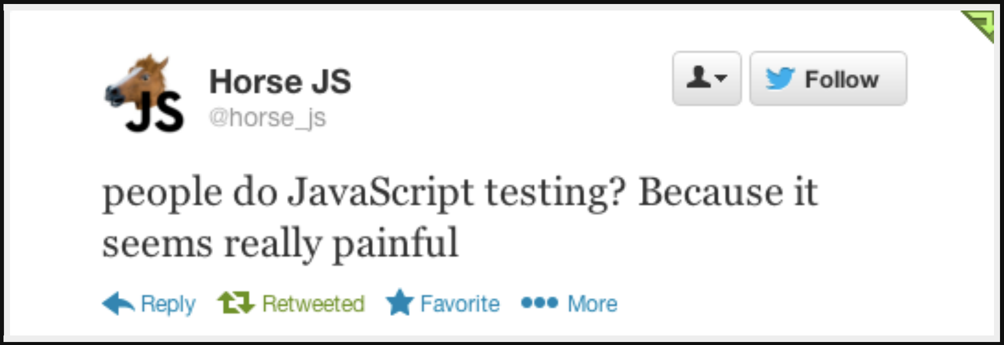
## See it in action

## Why TDD?

## Why TDD?

- Reduced defect rates
- Protection against regression
- Better API designs
- Re-use tests as your specs

I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests.I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests. I will write unit tests.



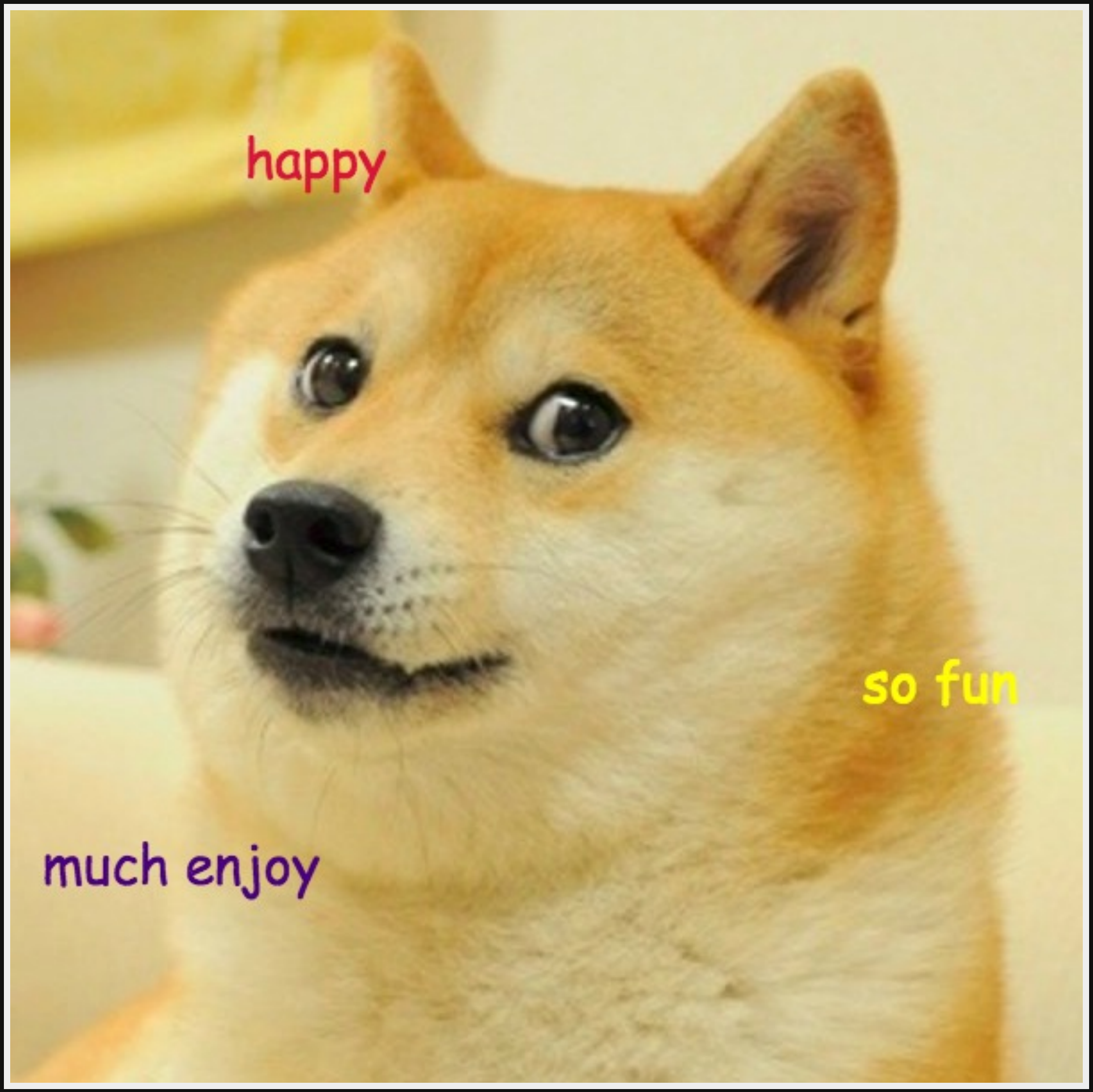
Horse JS
@horse_js

people do JavaScript testing? Because it seems really painful

*“Sounds like having fun at the dentists office.”*

*“Pragmatism: do the right thing, if it's easy.”*

## Lazy Programmer

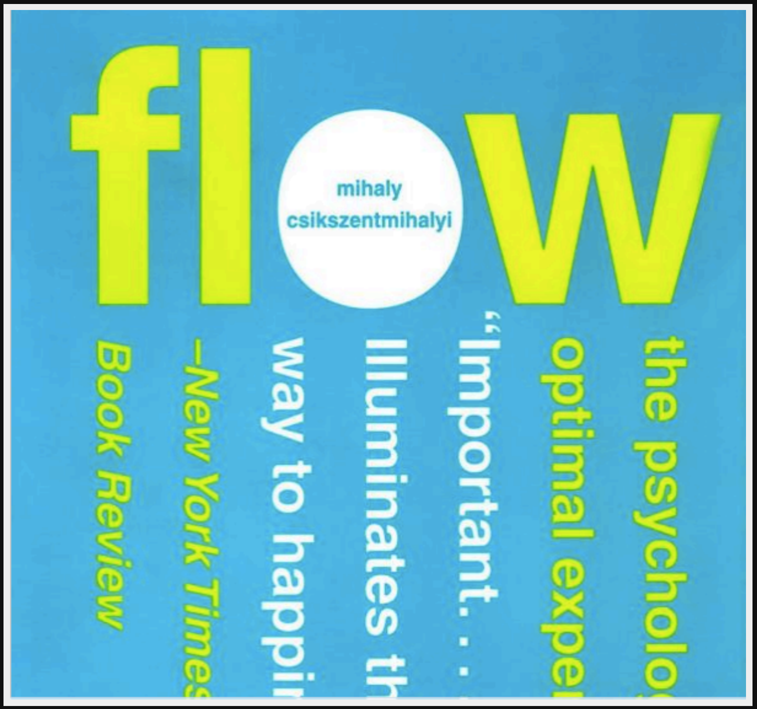

happy
so fun
much enjoy

## What is the Opposite?

## Pain

## Feedback

## Good Feedback



## Good Feedback

- Fast
- Frequent
- Reliable

## Getting Good Feedback

## Good Feedback 1:
## Workflow

## Workflow Tips

1. Don't use slow software
2. Don't run software you don't need
3. Autorun your tests!
4. Don't switch windows, tile'em
5. Typing fast helps! Practice.

## Good Feedback 2:
## Fast Tests

## < 1 Second

### Whatever it takes

## Write Lots of Fast Tests, Little to No Slow Tests

## Run Test Subsets

## Run Individual Test Suite

```
suite.only('my test suite', function(){

  ...

})
```

## Run Individual Test Case

```
test.only('my test case', function(){

  ...

})
```

## Divide Project Into Small Sub-modules

## Good Feedback 3.

## Take Small Steps

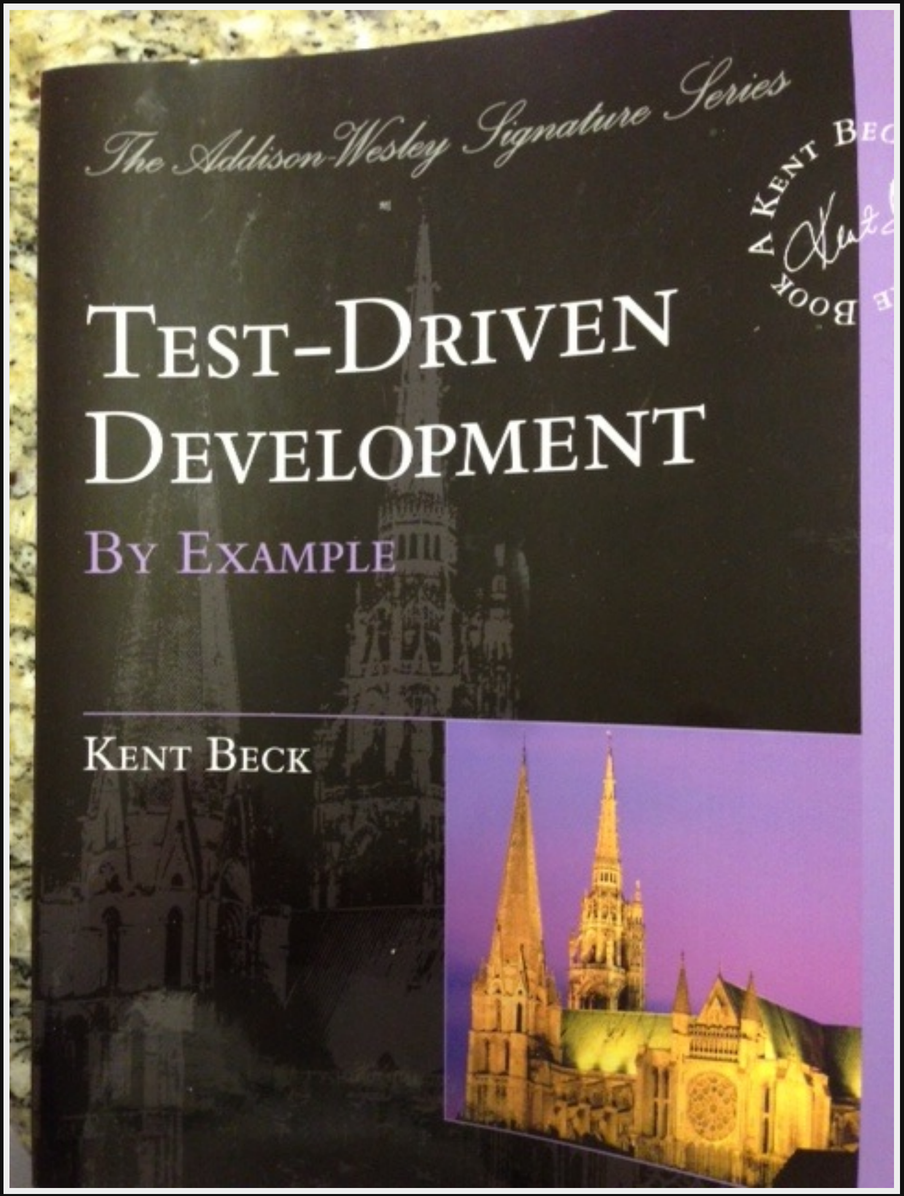
## Smaller Steps -> More Fre-quent Test Runs

## Small Steps Helps Debugging

- Context of what you changed
- Easy undo back to last green point

## Small Steps is a skill

## Test Patterns


*The Addison-Wesley Signature Series*

TEST-DRIVEN DEVELOPMENT

BY EXAMPLE

KENT BECK

## Small Step Test Patterns

1. Child tests
2. Fake it 'til you make it
3. One to many
4. Isolated change
5. Back up and refactor

# Main Event!!!

vs

## Thank You!

### Any Questions?

Toby Ho
**tobyho.com**
**@airportyh**