Testing Javascript

TDD Javascript

Respect your Javascript

Corey Haines

@coreyhaines

Journeyman Developer

That's Me!

www.coreyhaines.com

Why don't we test Javascript?

- History of scripting language
- UI-focused
- Hard
- No good tools

But Javascript Is Here!

Javascript Environments

Out of Browser

- Browser
- Node.js
- Env.js
- More and more!

TDD

- Not tool-specific, concept-specific
- Focused, micro-level tests
- Design technique!

Pain in testing means
your design has problems

TDD Tools

- Jasmine
- QUnit
- JSUnit
- JSpec
- Should.js
- Lots More!

BDD

- Not tool-specific, concept-specific
- Feature-level Full Stack tests driving Example-level Isolation tests
- Outside-in

BDD Tools!

- Jasmine
- QUnit
- JSUnit
- JSpec
- Should.js
- Lots More!

Would you write this code in your view?

```
%h1=Total Pay
- users = User.where(:invited_at => yesterday)
- sorted_users = users.sort_by(&:email)
- amount = 0
- users.each do |user|
  - pay = user.salary
  - amount = amount + pay
%h2=amount
```

Why not?

Separation of Concerns

So why do we mix presentation-specific code with business logic in our javascript?

```
var users, pay, display;

users = $('#users tr');
pay = 0;
users.each(function(item){
  pay += item.find('td.pay').val().to_i();
};
display = $("#total_pay");
display.text(pay);
```

Design Javascript?

4 Rules of Simple Design

Tests Pass

Of course

No Duplication (DRY)

Every piece of knowledge
has one and only one
representation

Reveals Intent

Good Names

Small

Well, we are in Javascript

4 Rules of Simple Design

4 Rules of Simple Design

Test Pass

4 Rules of Simple Design

Test Pass

No Duplication

4 Rules of Simple Design

Test Pass

No Duplication

Reveals Intent

4 Rules of Simple Design

Test Pass

No Duplication

Reveals Intent

Small

SOLID Principles

Single responsibility

Single responsibility

Open-Closed Principle

Single responsibility

Open-Closed Principle

Liskov Substition Principle

Single responsibility

Open-Closed Principle

Liskov Substition Principle

Interface Segregation

Single responsibility

Open-Closed Principle

Liskov Substition Principle

Interface Segregation

Dependency Inversion

Design guidelines

Stratified Design

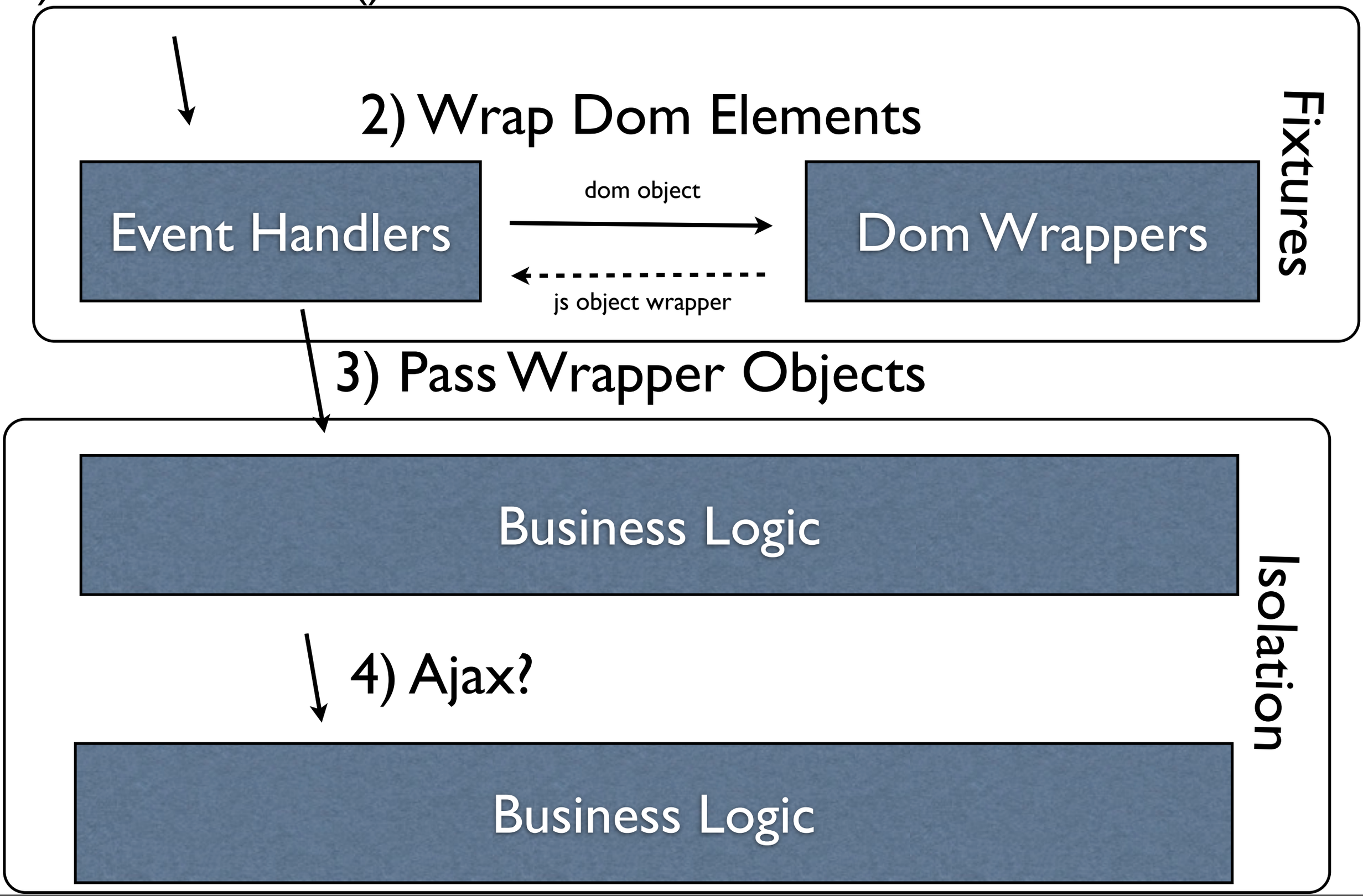
It is all about the abstractions

Find a canonical representation

DRY

HTML element
data-* attributes

Test Strategies



```
var users, pay, display;

users = getUsers($('#users tr'));
pay = calculatePayFor(users);
display = getDisplay('#total_pay');
display.setValue(pay);
```

```
function calculatePay(users) {
  return users.inject(0, function(sum, user) {
    sum += user.salary();
    return sum;
  };
}
```

Get away of the DOM

ASAP!!

Wrap interactions with DOM

Builders = { ... }

Behaviors = { ... }

Builders

Object Creation is cheap

```
ball = Builder.createFoo(domElement);
```

```
Builder.createFooDom(ball);
```

Use same code for js and html rendering

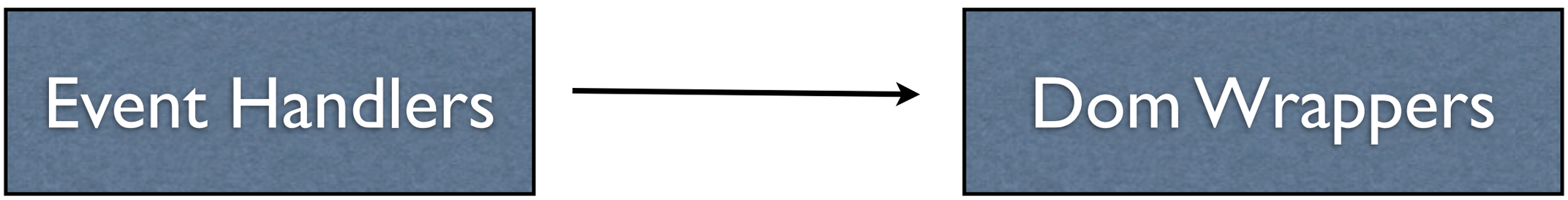
Wrap based on behavior!

Interface Segregation

Updateable
Persistable
Draggable

```
draggableBall = Behaviors.makeDraggable(ball);
```

Guideline:
Only the top level gets to use $() selectors



Spies/Stubs not Mocks

Thanks for your time

Questions?

Corey Haines
@coreyhaines