Presentation Layer Testing

Experimental!

Mea Culpa

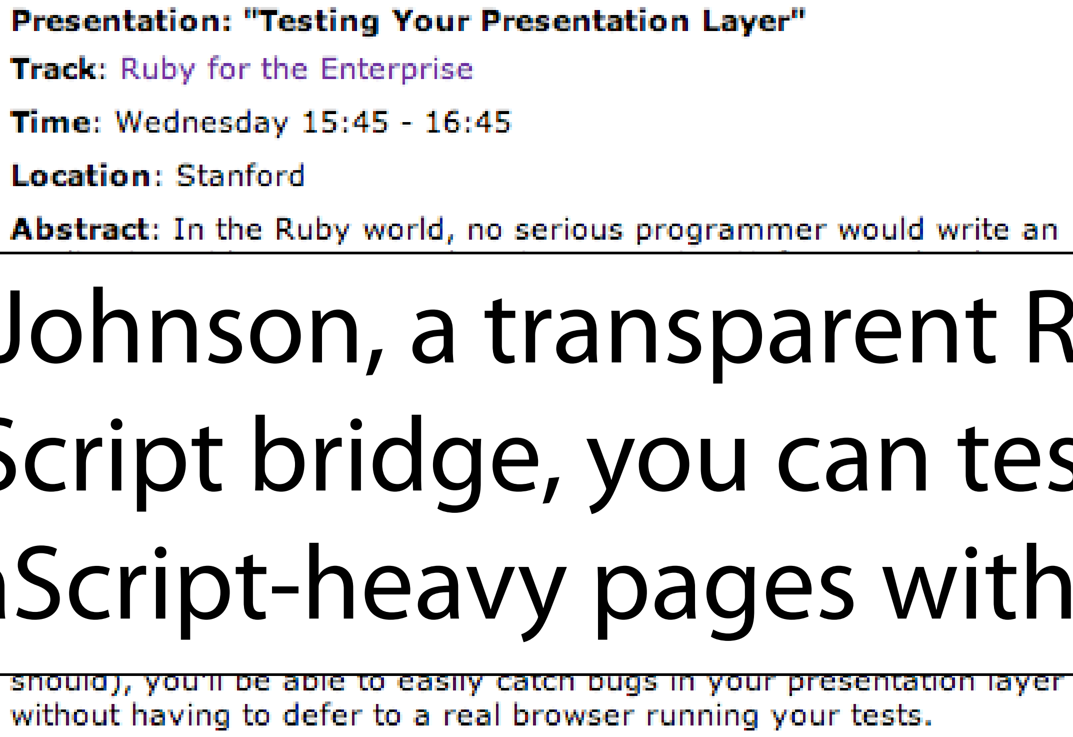
Presentation: "Testing Your Presentation Layer"

Track: Ruby for the Enterprise

Time: Wednesday 15:45 - 16:45

Location: Stanford

Abstract: In the Ruby world, no serious programmer would write an application without a comprehensive test suite. Unfortunately, the realities of integration testing the presentation layer of an Ajax-heavy web application has forced today's programmers to rely on kludges like HTML parsing in Ruby (which can't be used to test JavaScript comprehensively) or Selenium (which requires real-life browsers on every platform armed and ready). Screw that! With Johnson, a transparent Ruby to JavaScript bridge, you can test your JavaScript-heavy pages with ease. Because Johnson is embedded inside of Ruby, you'll be able to run tests directly in JavaScript, and then access JavaScript components in the tests (think Rails.dispatch_to("controller", "action")). I'll show you how to use Screw.Unit for browserless tests of your presentation layer. If you use continuous integration (and you should), you'll be able to easily catch bugs in your presentation layer without having to defer to a real browser running your tests.



Presentation: "Testing Your Presentation Layer"

Track: Ruby for the Enterprise

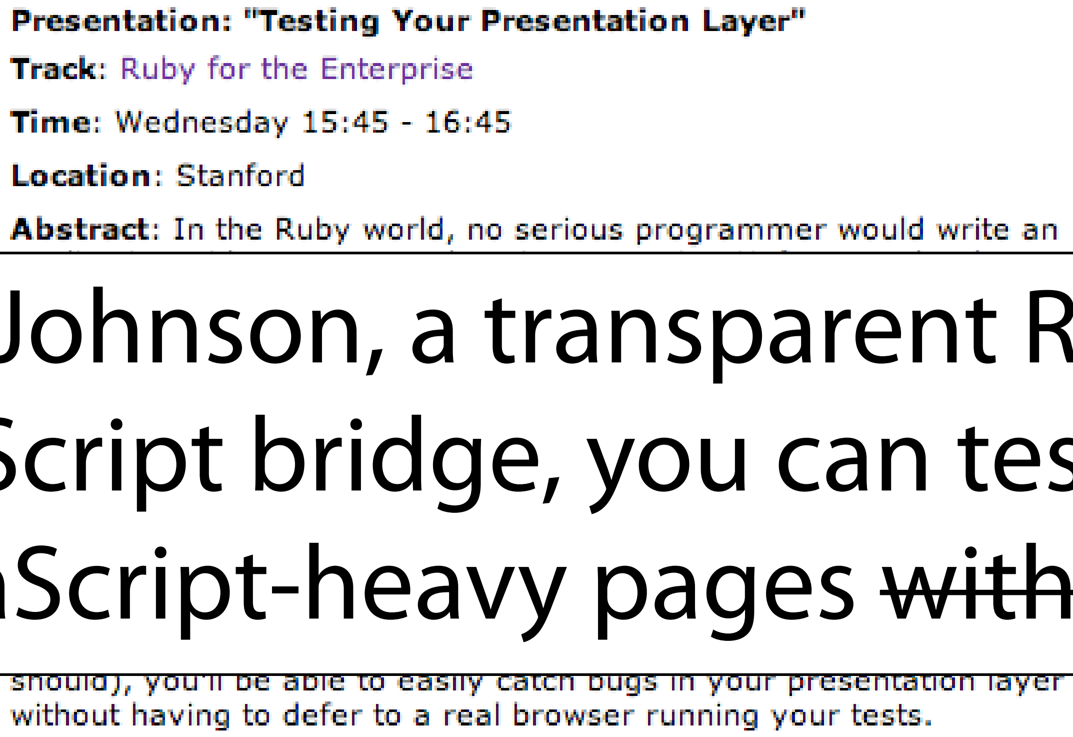
Time: Wednesday 15:45 - 16:45

Location: Stanford

Abstract: In the Ruby world, no serious programmer would write an

With Johnson, a transparent Ruby to JavaScript bridge, you can test your JavaScript-heavy pages with ease

should), you'll be able to easily catch bugs in your presentation layer without having to defer to a real browser running your tests.


Presentation: "Testing Your Presentation Layer"

Track: Ruby for the Enterprise

Time: Wednesday 15:45 - 16:45

Location: Stanford

Abstract: In the Ruby world, no serious programmer would write an

should), you'll be able to easily catch bugs in your presentation layer
without having to defer to a real browser running your tests.

With Johnson, a transparent Ruby to JavaScript bridge, you can test your JavaScript-heavy pages ~~with ease~~

Johnson

JavaScript

Ruby

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```ruby
class MyNumber
  def initialize(num)
    @num = num
  end

  def succ
    @num + 1
  end
end

MyNumber.new(12).succ #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype.succ =
  function() {
    return this.num + 1
  }

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype = {
  succ: function() {
    return this.num + 1
  }
}

new MyNumber(12).succ() #=> 13
```

```
MyNumber = function(num) {
  this.num = num;
}

MyNumber.prototype = {
  succ: function() {
    return this.num + 1
  }
}

new MyNumber(12).succ() #=> 13
```

```ruby
class BiggerNumber < MyNumber
  def initialize(num)
    @num = num + 1
  end
end

BiggerNumber.new(12).succ #=> 14
```

```ruby
class BiggerNumber < MyNumber
  def initialize(num)
    @num = num + 1
  end
end

BiggerNumber.new(12).succ #=> 14
```

```
BiggerNumber = function(num) {
  this.num = num + 1;
}

BiggerNumber.prototype =
  new MyNumber

new BiggerNumber(12).succ #=> 14
```

```
BiggerNumber = function(num) {
  this.num = num + 1;
}

BiggerNumber.prototype =
  new MyNumber

new BiggerNumber(12).succ #=> 14
```

Why?

Johnson exposes Ruby

(weirdly)

```
var x = new Ruby.Array();
x.push(12);
x.flatten();
x.last() == 12 //=> true
```

```
var x = new Ruby.Array();
x.push(12);
x.flatten();
x.last() == 12 //=> true
```

```
var x = new Ruby.Array();
x.push(12);
x.flatten();
x.last() == 12 //=> true
```

```
var x = new Ruby.Array();
x.push(12);
x.flatten();
x.last() == 12 //=> true
```

```
var x = new Ruby.Array();
x.push(12);
x.flatten();
x.last() == 12 //=> true
```

```
var arr =
  new Ruby.Array([1,2,3,4])

arr.collect(function(x) {
  return x + 1
})

//=> [2,3,4,5]
```

```
var arr =
  new Ruby.Array([1,2,3,4])

arr.collect(function(x) {
  return x + 1
})

//=> [2,3,4,5]
```

```
var arr =
  new Ruby.Array([1,2,3,4])

arr.collect(          {
  return x +
})

//=> [2,3,4,5]
```

proc {|x| x + 1 }

Demo

How?

Johnson

env.js

jspec

Webrat

Glue ;)

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

```
Johnson.require("johnson/jspec");
Johnson.require("johnson/browser");
Johnson.require("johnson/browser/jquery");

var alert = function(str) { Ruby.puts(str); };
var merb = {
  visit: function(url) {
    var resp = spec.visit(url);
    document = new DOMDocument(resp.body.to_s());
    $("script[src]").each(function() {
      var src = $(this).attr("src");
      if(src[0] == "/" && !src.match(/jquery.js/)) {
        var resp = Ruby.File.read(Ruby.Dir.pwd() +
          "/public/" + $(this).attr("src"));
        eval(resp);
      }
    });
  }
}
```

Speaking of Webrat

Demo