Ruby on Rails from the other side of the tracks

Tom Armitage
LRUG, August 8th
aka
“working with your design team”

Tom Armitage
LRUG, August 8th
Who here makes stuff on the web? In Rails, maybe?
Who would say they were roughly between “good” and “expert” at either Ruby or Rails?
You may be used to the following screens. But. *This is not the web:*
redirect_to request.referer

else
  flash[:comment] = "I'm sorry, but you can't make a complaint right now."
  redirect_to request.referer
end

def search
  @articles = Article.find_by_contents(params[:searchquery])
  s = SearchRequest.new
  s.query_string = params[:searchquery]
  s.ip = request.remote_ip
  s.save

  # this is a fudge to get around acts_as_ferret sucking.
  # there are better instructions in article.rb on a good solution
  @articles.each_with_index do |a,i|
    if a.published.nil?
      @articles.delete_at(i)
    end
  end
end

@archive_header = "Search results for \"#{params[:searchquery]}\", ordered by relevance"
@query = params[:searchquery]
@archives = get_archives
render :action => 'list'
nor is this:
DROP TABLE IF EXISTS entries;
CREATE TABLE entries (  
  id int(11) NOT NULL auto_increment,  
  image varchar(200) default NULL,  
  file varchar(200) NOT NULL,  
  PRIMARY KEY (id)  
) TYPE=MyISAM;

DROP TABLE IF EXISTS movies;
CREATE TABLE movies (  
  id int(11) NOT NULL auto_increment,  
  movie varchar(200) default NULL,  
  PRIMARY KEY (id)  
) TYPE=MyISAM;
nor is this:
Object.extend = function(destination, source) {
    for (var property in source) {
        destination[property] = source[property];
    }
    return destination;
}

Object.inspect = function(object) {
    try {
        if (object === undefined) return 'undefined';
        if (object === null) return 'null';
        return object.inspect ? object.inspect() : object.toString();
    } catch (e) {
        if (e instanceof RangeError) return '...';
        throw e;
    }
}

Function.prototype.bind = function() {
    var __method = this, args = $A(arguments), object = args.shift();
    return function() {
        return __method.apply(object, args.concat($A(arguments)));
    }
}

Function.prototype.bindAsEventListener = function(object) {
    var __method = this;
(thank god)
the Web is
<legend>Log in</legend>
<label>
  Nickname
</label>
<input type="text" name="unickname" size="20" value=""/>
<label>
  Password
</label>
<input type="hidden" name="returnto" value="/slashdot.org/">
<input type="hidden" name="op" value="userlogin">
<input type="password" name="upasswd" size="20">
<label class="checkbox">
  <input type="checkbox" name="login_temp" value="yes">
  Public Terminal
</label>
<input type="submit" name="userlogin" value="Log in" class="button">
</form>

[ <b>
  <a href="/slashdot.org/login.pl?op=newuserform">
    Create a new account
  </a>
</b> ]

</div>
</div>
</div>
<div id="links">
  <div class="block" id="links-sections">
    <div class="title" id="links-sections-title">
      <h4>
        Sections
      </h4>
    </div>
  </div>
</div>
CSS
Who here would say they had expert-level XHTML?
Why the hell don’t you?
It’s OK, we have people to do this for us:
Designers!
They will save us with their rounded corners and stock photos!
More to the point, some of them *might* be good at that XHTML lark!
Sometimes dedicated people (not “designers”) write markup - so also talk to:
Client-side developers

Markup monkeys
Anyway...
What to do with front-enders

Don’t assume you know better
Don’t outsource
Get them on board
Get them templating
Why?

Close the loop
Give them ownership
Let them do their job
Avoid mistakes
Mistakes, you say?
A list of items.

<ul class='someclass'>
  <li>An item</li>
  <li>Another item</li>
  <li>The third item</li>
</ul>
The developer immediate approach.
This is valid XHTML 1.0 strict, but it may also lead to positional/aesthetic issues.

(It’s also bobbins, semantically.)
<ul class='someclass'>
    for item in @items
      <li><%= item.name %></li>
    end
</ul>

Let's improve this...
if @items.size > 0
<ul class='someclass'>
  for item in @items
    <li><%= item.name%></li>
  end
</ul>
end

That’s better.
if @items.size > 0
  <ul class='someclass'>
    for item in @items
      <li><%= item.name%></li>
    end
  </ul>
else
  <p>You have no items</p>
end

(Best).
How?

Get them into source control

If you explain it well enough, everyone loves version control

Collaborate on working wireframes

Answer their questions

Ask *them* questions

Intervene (eg with helpers)
Some notes
Javascript & AJAX
AJAX is cool!
Javascript is coming back into fashion.
(Who here would say they had expert level Javascript?)
(Work on it - it’s going to come in handy)
Libraries make Javascript much less of a PITA.
Libraries are heavy
Library weigh-in:

prototype.js - 56kb

effects.js - 34kb

controls.js - 29kb

dragdrop.js - 30kb
The problems with Prototype

Scaffolding gives you bad habits:

That’s **146kb** on your page load

And it loads serially

*Use what you need*

*You don’t even need Prototype for basic JavaScript*
Helpers and accessibility
Rails’ HTML helpers are pretty great
Rails’ HTML helper are:

Accessible!

Valid!

Powerful!
Rails’ Javascript helpers, on the other hand...
They work...
...but not like they should.
eg
<a href="" onclick="...">foo</a>
<a href="" class="toggle-user"">foo</a>
Seriously, though:

Javascript has thorny accessibility issues.

AJAX can be really inaccessible:

Screenreaders

Not just screenreaders

Well-written Javascript goes a long way to make things easier
“Hijax”

Write without Javascript

Then progressively add it, focusing on ids and classnames to act as hooks

Best of both worlds

Yes, this doesn’t work for some apps - but Web 2.0 doesn’t need to mean “inaccessible” all the time.
What’s Rails doing about this?
I asked DHH...
“Fuck off”
For everyone reading these slides who wasn’t at the talk: DHH didn’t say this. It’s a joke.
however...
Luke Redpath and Dan Webb rule!
Accessible Javascript Plugin:
http://tinyurl.com/znzmc
It’s awesome
Accessible Javascript Plugin

Minimal changes to your code
No inline reference to Javascript!
Dynamically generated .js
Dynamically generated event handling
...and more

seriously impressive.
Testing
Everybody loves test-driven development, right?
Testing XHTML

Easy: W3C validator

Valid code is easier to debug

if it breaks, it’ll break in a consistent manner

no point writing invalid XHTML

Want to automate that?
def assert_valid_markup(markup=@response.body)
    require 'net/http'
    response = Net::HTTP.start('validator.w3.org') do |
        query = 'fragment=' + CGI.escape(markup) + '&output=xml'
        w3c.post2('/check', query)
    end
    assert_equal 'Valid', response['x-w3c-validator-status']
end
No excuse for developers breaking front-end code any more!
Going further

Test components of your page with something like Hpricot

Counting elements: boring

Checking `<title>` is what it should be: useful

Selenium, Watir

Beyond my scope, but certainly also useful
To summarise
XHTML/CSS/JS are core components of your app, like it or not.
Designers and client-side developers know their stuff, so use them!
Take accessibility seriously
Take validation seriously
Treat your front-end folks, and their code, as first-class citizens. The web is, after all, only XHTML.
Thanks!

Recommended reading:

Designing With Web Standards - Jeffrey Zeldman

Web Standards Solutions - Dan Cederholm

CSS Mastery - Andy Budd

DOM Scripting - Jeremy Keith

The Rhino (O’Reilly js book)