



Mobile



Rails 3

Web 2.0

First, what's
"web 2.0"

Ajax

It's tightly
bound to the
tech of Ajax

March 1999

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
28	1	2	3	4	5	6
			10	11	12	13
			17	18	19	20
			24	25	26	27
			31	1	2	3

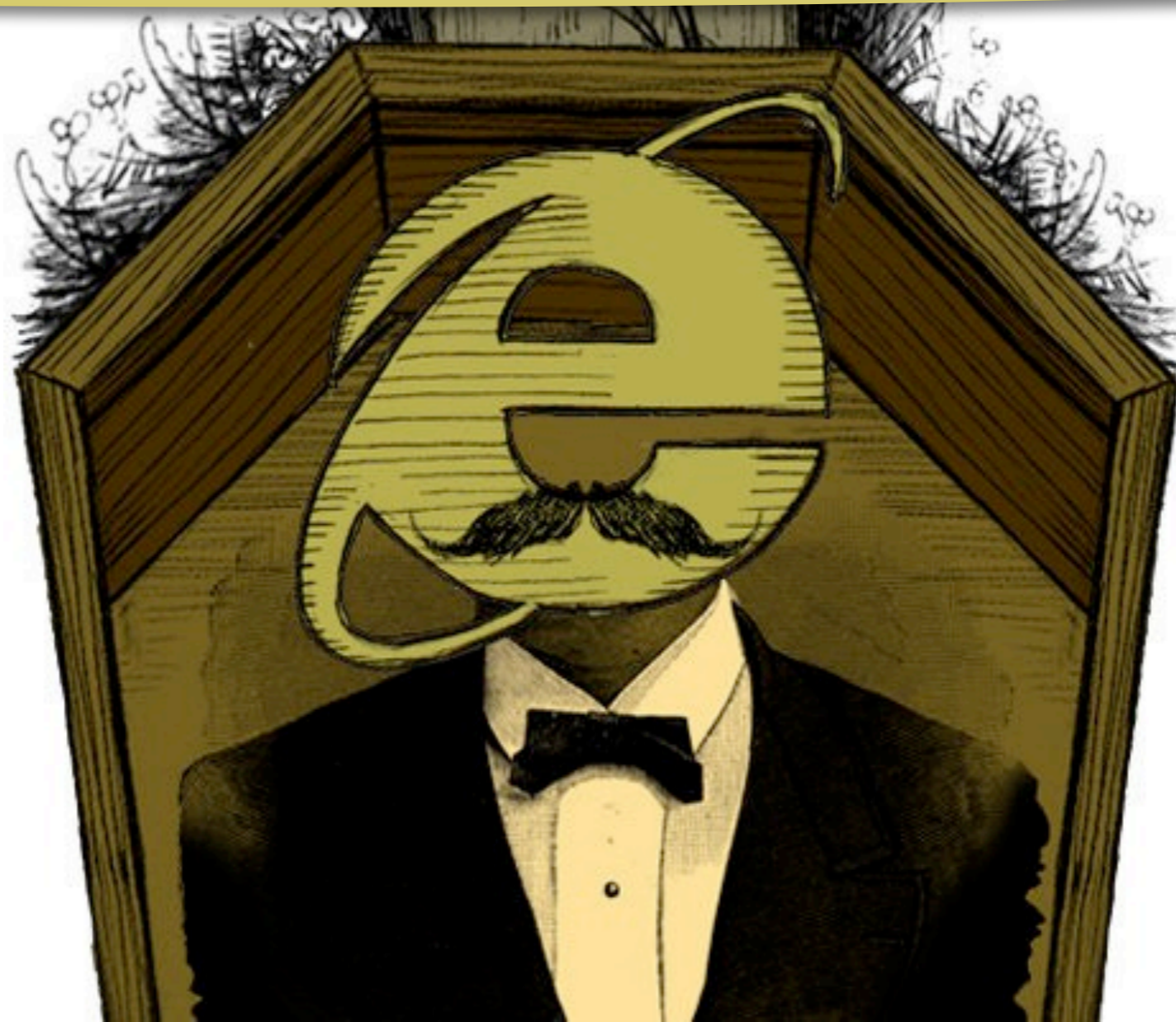
**Ajax was
released in
1999, in IE5**

March 1999

Microsoft[®]

By Microsoft, in
the heyday of the
browser wars

As an aside: We all love to hate IE6... and I indulge in that hate often enough



But IE6 was actually a pretty good browser for 2001

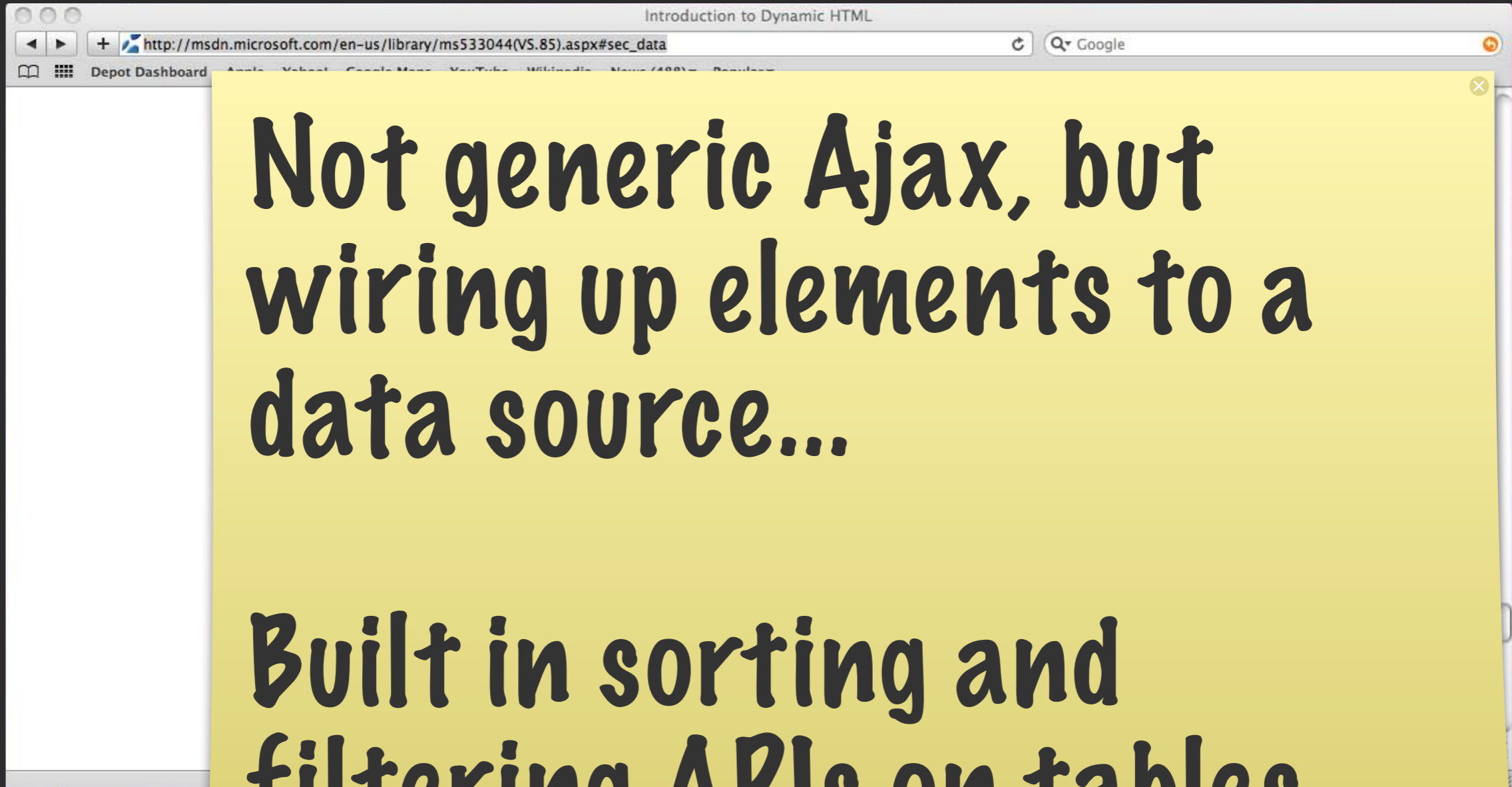
Friday					Saturday				
				3					4
				10					11
				17					18
				24					25
				31					1
26	27	28	29	30	31	1	2	3	4

August 2001

IE was actually pretty far ahead of its time



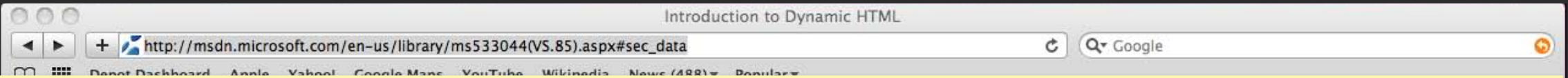
Data binding is a **DHTML feature** that lets you easily bind individual elements in your document to data from another source, such as a database or comma-delimited text file. When the document is loaded, the **data is automatically retrieved** from the source and formatted and **displayed within the element**.



Not generic Ajax, but wiring up elements to a data source...

Built in sorting and filtering APIs on tables

Because data render quickly and provide immediate interactivity. Once downloaded, the data can be **sorted and filtered** without requiring additional trips to the server



Introduction to Dynamic HTML
http://msdn.microsoft.com/en-us/library/ms533044(VS.85).aspx#sec_data
Google

And not IE6... but IE4...

Just goes to show that we lost a lot in the “nonstandard equals doesn’t exist” early part of this century

This feature requires Microsoft **Internet Explorer 4.0** or later

Ajax

So anyhow Ajax

Client

**IE4+ Made
Clients a Lot
Smarter**

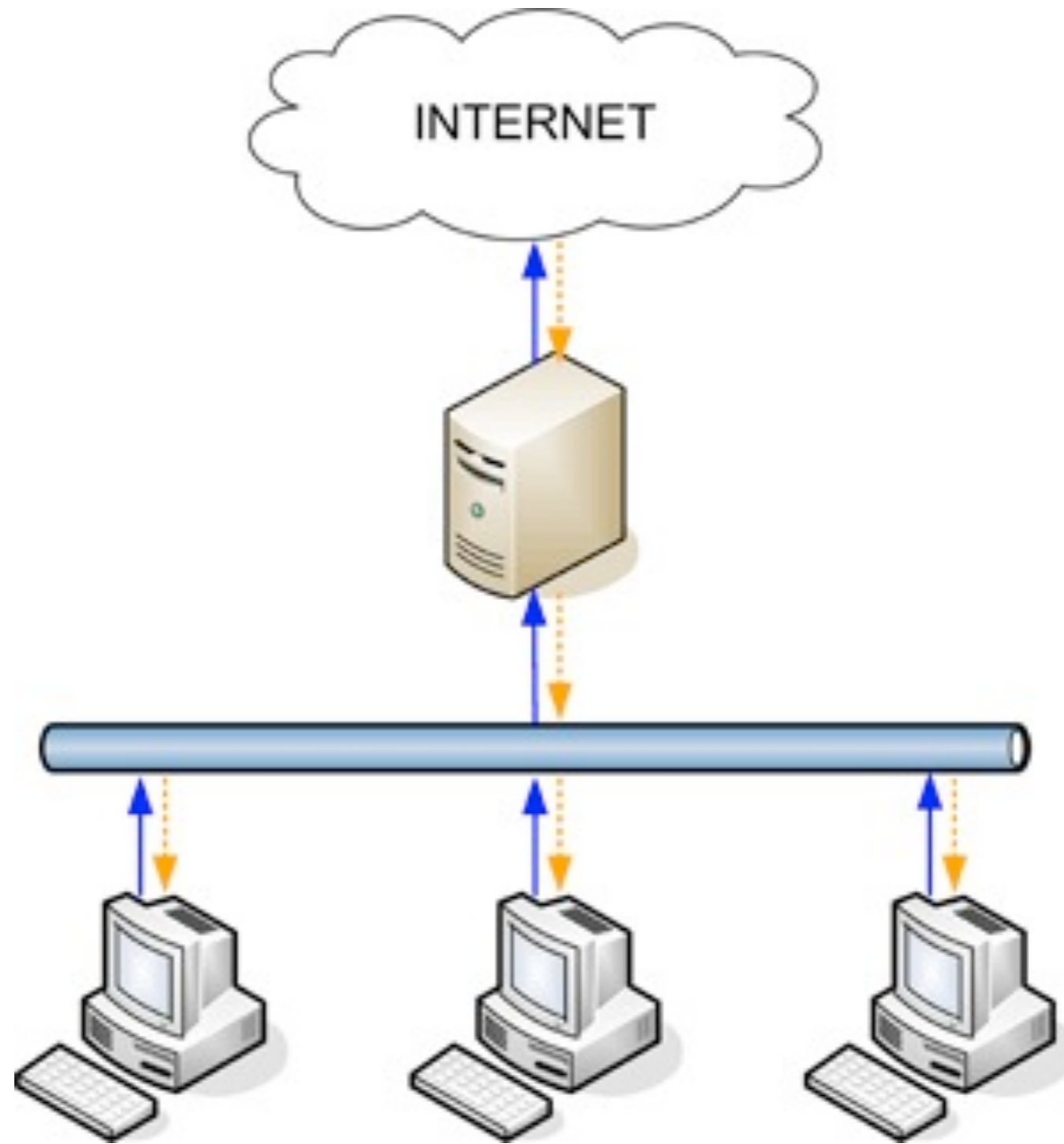
“Web 2.0”

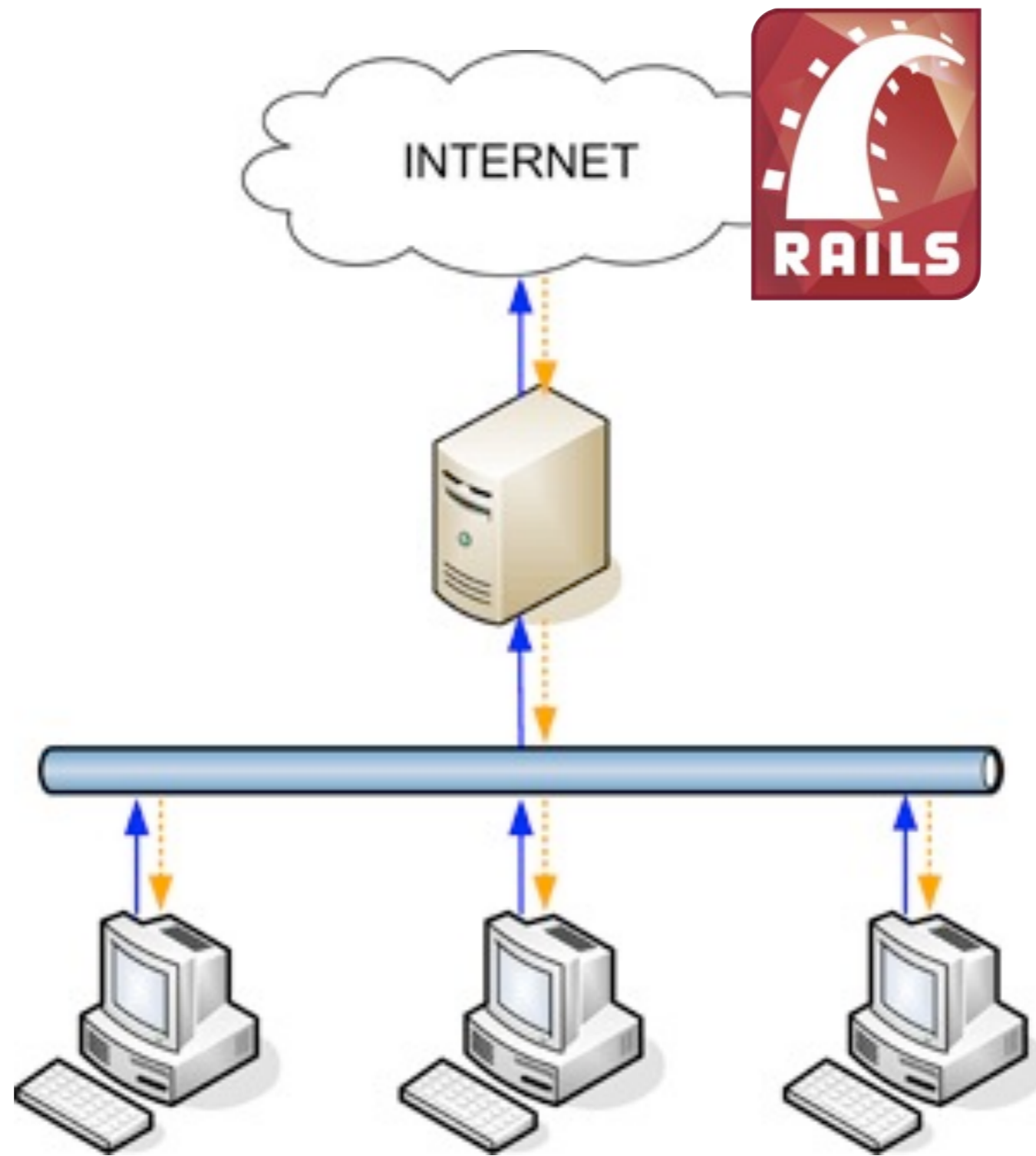
Made Use of

the Tech

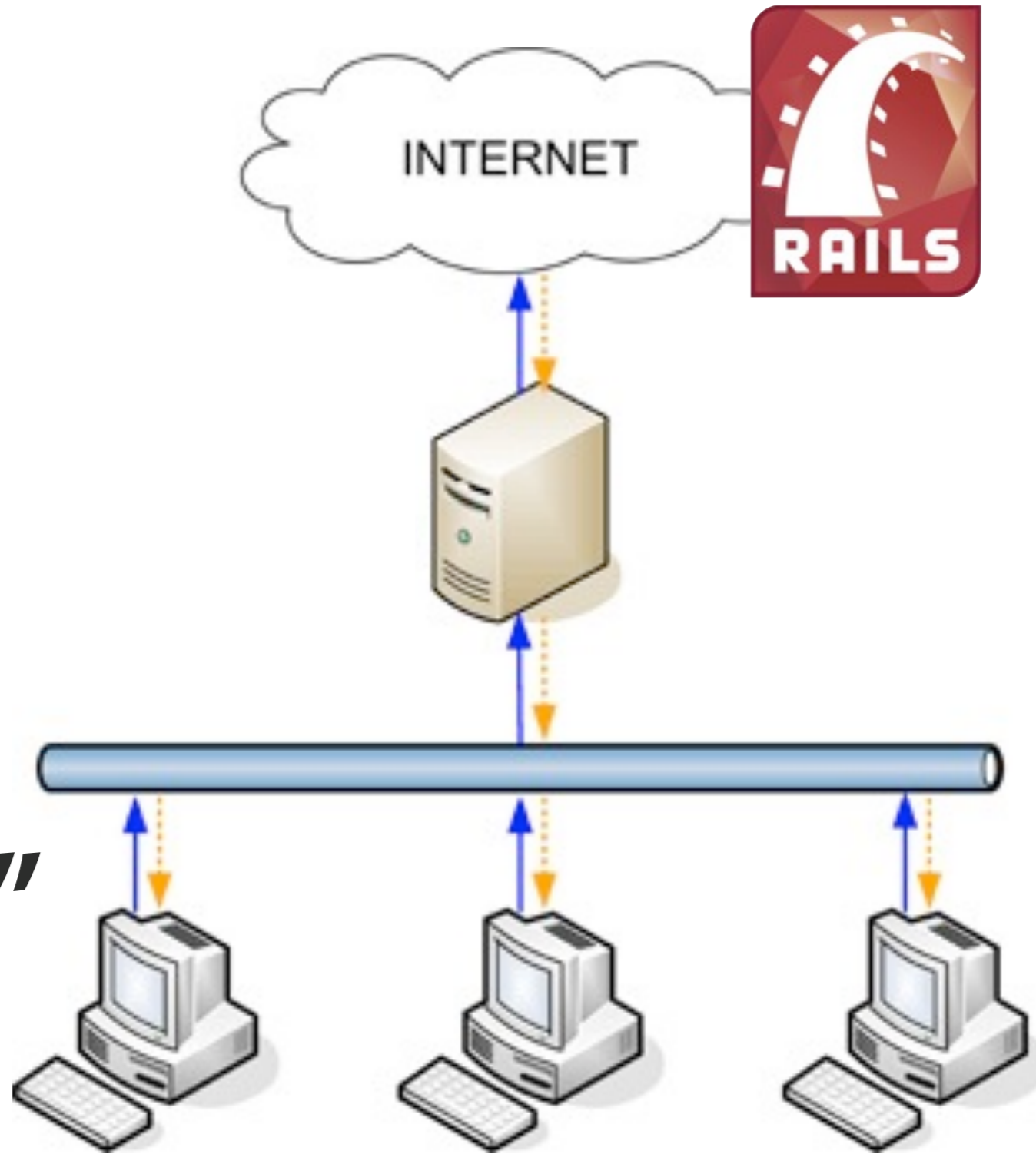
A Decade
Later

Evolution of Client-Side Capabilities





“client”



Not New

None of this
tech came out
last week

Newer than
Ajax in 2005

Not Enough Explanation

Especially about
how to use the
tech together

IE6 Factor



IE6 Factor



People

Feel

Burned

**“I’ll look at
HTML5 when I
can use it...”**

“...In 2019”

WHERE'S MY #@%&!*\$

JETPACK?!



HEY 2010: LET'S SEE

But this new tech actually isn't vaporware





















Opera Mini





People Don't Use Cell Phones Made in 2001

The mobile space doesn't have the problem of people who refuse to upgrade forever... so this nice property is probably here to stay

+ 'S

Best Distribution System

No Approval Process

Instant Updates



Opera Mini



**Start
Caring
Now**

Mobile

**Very Different
Constraints**

Inspect

Console HTML CSS Script DDM Net **YSlow** Options

Grade Components Statistics Tools Rulesets YSlow(V2) Edit Printable View Help

Grade your web pages with YSlow


YSlow gives you:

- Grade based on the performance of the page (you can define your own ruleset)
- Summary of the page components
- Chart with statistics
- Tools for analyzing performance, including Smush.it™ and JSLint

Autorun YSlow each time a web page is loaded

Run Test

[»Learn more about YSlow](#)



Copyright © 2009 Yahoo! Inc. All rights reserved.

For desktop websites, the gold standard is YSlow!

**Fast, Reliable
Connections**

HTTP Overhead

Dominates

Small Assets

Hard Disks

HTTP

Caches Persist

“Forever”

**Lowest
Common
Denominator?**

2001

2006?

Powerful Processor

Unlimited
"Battery"

**Prefer New
Download to
Stale Data**

YSlow!

One Large JS
or CSS File

: cache \Rightarrow true



Spriting

```
.fanta {  
  background: sprite-image("bottles/fanta.png");  
}  
  
.seven-up {  
  background: sprite-image("bottles/seven-up.png");  
}  
  
.coke {  
  background: sprite-image("cans/coke.png") no-repeat;  
}
```



Expires Header

```
# # We use cached-busting location names
# # with the far-future expires
# # headers to ensure that if a file does
# # change it can force a new
# # request.
```



ETags

```
def show
  @article = Article.find(params[:id])

  fresh_when(
    :etag => @article,
    :last_modified => @article.created_at.utc,
    :public => true
  )
end
```



25K

Components

**Lots Still
Applies**

**Some Missing
Information**

Mobile Constraints

Poor HTTP Caching

Flaky and Offline Connections

Long
Downloads Can
Get Droppped

**Lowest
Common
Denominator?**

HTML5 Offline

In fact, these specs were written with this use case in mind by the vendors!

Application Cache

Local Storage

**Prefer Stale
Data to a New
Download**

Some People
Pay Per kb

Limited Battery,
Limited CPU

Revisit Orthodoxy

Priority

Download Data

Only Once

The Unofficial Apple Weblog (TUAW)

http://i.tuaw.com/#_a RSS Google

Depot Dashboard Apple Yahoo! Google Maps

TUAW The Unofficial Apple Weblog

- Sixteen new Apple patents, from CoverFlow to iDVD >
- Five apps I love on my iPad, and three that need more work >
- Chitika tracking iPad purchases, estimates 560K so far >
- iTriage provides mobile health advice with style >
- How to make \$150 million in a day >
- Apple rejects iPad app for pinch-to-expand >
- Apple releases new iPhone Ad: "Shopper" >
- Poll: Who's gonna keynote? >
- iPad Wi-Fi issues continue to irritate >
- Dr. Seuss iPhone apps get supersized for the iPad >

Next 10 articles

AOL Search

Go to "http://i.tuaw.com/2010/04/08...ides-mobile-health-advice-with-style/"

Home Page
Read Story
Back Button

Home Page

6k

Read Story

16k

Home Page

6k

Slow Connections

With a slow connection,
this constant downloading
is going to take FOREVER

Offline

When offline, you can't see something you **just** saw because you're in a tunnel

Incremental Rendering

Rendering the page
incrementally actually costs
CPU and battery resources

**Cache Cleared
on a Dime**

16k?!!

```
<script language="JavaScript" type="text/javascript">
var sns_checked = false;
var current_tab = "blogsmith";
function tabTo(tab)
{
    document.getElementById('formerrors').innerHTML = '';

    document.getElementById('cmtuinfo_email').style.display='none';
    document.getElementById('cmtuinfo_blogsmith').style.display='none';
    document.getElementById('cmtuinfo_sns').style.display='none';
    document.getElementById('cmtuinfo_'+tab).style.display='block';

    document.getElementById('cmtutab_email').className='';
    document.getElementById('cmtutab_blogsmith').className='';
    document.getElementById('cmtutab_sns').className='';
    document.getElementById('cmtutab_'+tab).className='currenttab';

    if (!sns_checked && tab == 'sns')
    {
        image1 = new Image();
        image1.src = "http://www.blogsmithmedia.com/framework.weblogsinc.com/media/
loading.gif";
        sns.init('ch1ga1KvP7TotwTC');
        sns_checked = true;
    }

    current_tab = tab;
}
```

```
<div id="replyindicator"></div>
<div>

  <label for="Comments">Your comments:</label>
  <textarea name="Comments" id="Comments" rows="8" style="width:98%"></textarea>
</div>
<div class="cmtchecks">
  <input type="checkbox" checked="checked" id="RememberMeYes" name="RememberMe" />
  <label for="RememberMeYes">Remember me</label>
</div>

<div class="cmtchecks">
  <input type="checkbox" checked="checked" id="EmailMe" name="EmailMe" />
  <label for="EmailMe">E-Mail me when someone replies to this comment</label>
</div>


<div id="cmtbuttons">
  <input type="submit" id="addCommentButton" value="Add Comment" />
</div>
<input type="hidden" name="Form" value="Comments" /><input type="hidden"
name="ButtonSave" value="Save" />

<input type="hidden" id="sourceID" name="SourceID" value="" />
<input type="hidden" id="postID" name="PostID" value="" />
```

Two Problems

1. Boilerplate

2. Structural



There are also some more technical patents for iChat video encoding and error adjustments on touchscreens, as well as overall patents for the MacBook Air SuperDrive and iDVD. It seems like the USPTO is just cleaning out Apple's old patents -- most of these were filed back in 2007. Now, maybe they can set the legal patent team up [on newer accomplishments](http://tuaw.com/tag/ipad).

```
<p class="posttags" style="clear:both;">
  <strong>Tags:</strong>
  <a href="http://i.tuaw.com/tag/apple/">
    apple</a>,
  <a href="http://i.tuaw.com/tag/error/">
    error</a>,</pre>
```

Ideal

**Download
Boilerplate
HTML Once**

Download

HTML

Templates

Once

Use
Lightweight
Transport for
Data

**Download Data
Prospectively**

**App Should
Work Without a
Connection**

**(with stale
data)**

**App Should
Work While
Making a
Connection**

**(with stale
data)**

**Applications
Should Avoid
Incremental
Rendering**

Arch?

REST

HTML

==

Static Asset

**Updatable, But
Caching
Semantics**

Semantic HTML

Native Protocol

For performance, we're going to want to leverage what the browser knows

HTTP

Rack

=

This means that we can use
all of the HTTP stuff in Rack
for free

Native Representation

JSON

JSON is also very small, so it satisfies many of the other constraints

render :json => @article



```
class PostsController
  respond_to :json

  def index
    @articles = Article.all

    respond_with @articles
  end
end
```



**Treat the
Browser Like an
API Client**

Wait!



Poor HTTP

Caching, Right?

Right!

New Tools

HTML5 Offline

APIs

(§6.6)

Application Cache

Local Storage

(HTML5 Web
Storage Spec)

**Gives Us
Power Back**

**We Need to be
More Explicit**

App Cache

```
<html manifest="app.manifest">
```



CACHE MANIFEST

```
javascripts/jquery.js  
media/logo-tuaw-iphone-v3.png  
media/favicon-v9-1.png
```



Content-Type:
text/cache-manifest



Rack::Offline

```
run Rack::Offline.new {  
  cache "stylesheets/style.css"  
  cache "images/masthead.jpg"  
  cache "javascripts/application.js"  
  cache "javascripts/jquery.js"  
  
  network "/"  
}
```



```
manifest = Rack::Offline.new {  
  cache "stylesheets/style.css"  
  cache "images/masthead.jpg"  
  cache "javascripts/application.js"  
  cache "javascripts/jquery.js"  
  
  network "/"  
}  
  
match "/app.manifest" => manifest
```



Rails::Offline

```
match "/app.manifest" => Rails::Offline
```



You'll Always Get One Stale

The constraint is that the browser can serve up the HTML fast and wait for a connection, so you'll get one stale hit after update

```
$(applicationCache)  
  .bind("updateready", displayReload);
```



Web Storage

KVS

```
localStorage.article = "Hello!"
```



5MB Limit

Ask For More

**Overwriting a
Key Will
Reclaim**

```
delete localStorage["article47"]
```



```
for(prop in localStorage)
```

Safari, not Firefox

Basic Strategy:
Sip, Don't Gulp

Rails

Adapt Techniques for Building APIs

```
class Post < ActiveRecord::Base
  scope :recent, limit(10).
           order("created_at desc")

  # or

  def self.recent(amount)
    limit(amount).order("created_at desc")
  end
end
```



```
class PostsController < ApplicationController
  respond_to :html, :json, :atom

  def index
    @posts = Post.recent
    respond_with @posts
  end
end
```



**ActionDispatch
and Rack are
Very Robust**

In Real Life...

Auth(z)

Rate Limiting

External Sources

Process Updates in Background

Caching

Payments?

API Integration

**All Things Rails
Handles Well**

Plus...

**Traditional
Web App w/
Same Backend**

Client-Side Strategy

First Time

First Time

Download HTML

First Time

Download HTML

Async: Download Application Cache

First Time

Download HTML

Async: Download Application Cache

Display "Loading"

First Time

Download HTML

Async: Download Application Cache

Display "Loading"

Kick off Request for JSON

First Time

Download HTML

Async: Download Application Cache

Display "Loading"

Kick off Request for JSON

Store JSON in localStorage

First Time

Download HTML

Async: Download Application Cache

Display "Loading"

Kick off Request for JSON

Store JSON in localStorage

Populate Template

First Time

Download HTML

Async: Download Application Cache

Display "Loading"

Kick off Request for JSON

Store JSON in localStorage

Populate Template

Remove Loading

Subsequent Times

Subsequent Times

HTML Retrieved from App Cache

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Find Stale Resources in localStorage

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Find Stale Resources in localStorage

Populate Template

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Find Stale Resources in localStorage

Populate Template

Display "Loading"

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Find Stale Resources in localStorage

Populate Template

Display "Loading"

Kick off Request for JSON

Subsequent Times

HTML Retrieved from App Cache

Async: Check for App Cache Updates

Find Stale Resources in localStorage

Populate Template

Display "Loading"

Kick off Request for JSON

Continue as Before


Download Application Cache

```
<html manifest="app.manifest">
```



Download Application Cache

```
manifest = Rack::Offline.new {  
  cache "stylesheets/style.css"  
  cache "images/masthead.jpg"  
  cache "jascripts/application.js"  
  cache "jascripts/jquery.js"  
  
  network "/"  
}  
  
match "/app.manifest" => manifest
```



Display Loading

```
$("#loading").show();
```



Kick Off Request for JSON

```
$.getJSON("/stories.json", updateStories)
```



Store JSON in localStorage

```
localStorage.articles =  
JSON.stringify(json)
```



Populate Template

```
git://github.com/jquery/  
jquery-tmpl.git
```




Populate Template

```
<script type="text/html" id="article">
  {{each(article) articles}}
  <div class="article">
    <h1><a href="{article.url}">
      {article.title}
    </a></h1>
    {article.intro}
    <a href="{article.url}#comments">
      Comments
    </a>
  </div>
  {{/each}}
</script>
```



Populate Template

```
var articles =  
  template.render(json.articles)  
  
// Single DOM insertion  
$("#list").empty().append(articles);
```



```
<script type="text/x-mustache" id="article">
  {{#articles}}
  <div class="article">
    <h1><a href="{{url}}">{{text}}</a></h1>
    {{intro}}
    <a href="{{url}}#comments">Comments</a>
  </div>
  {{/articles}}
</script>
```



Populate Template (Mustache)

```
<script type="text/x-mustache" id="article">
  {{#articles}}
  <div class="article">
    <h1><a href="{{url}}">{{text}}</a></h1>
    {{intro}}
    <a href="{{url}}#comments">Comments</a>
  </div>
  {{/articles}}
</script>
```



Populate Template (Mustache)

```
var template = $("#article").html()

var articles =
  Mustache.to_html(template, json.articles)

// Single DOM insertion
$("#list").empty().append(articles);
```

Remove Loading

```
$("#loading").hide()
```



Second Time

Find Stale Resources in localStorage

```
var articles = localStorage.articles;
```



Populate Template

```
var articles = localStorage.articles;  
  
if(articles) {  
    updateStories(JSON.parse(articles));  
}
```



Display Loading

```
$("#loading").show()
```



**Continue as
Before**

jquery-offline

```
$.retrieveJSON("/stories", function(json) {  
    var html = template.render(json.articles);  
    $("#list").empty().append(html);  
});
```

```
$("#loading").ajaxStart(function() {  
    $(this).show();  
}).ajaxStop(function() {  
    $(this).hide();  
});
```



```
$.retrieveJSON("url",  
  function(json, status, follow) {  
    // json == JS Object  
    // status == "success" || "cached"  
    // follow == { cachedAt: originalTime,  
    //   retrievedAt: timeRetrieved }  
  });
```



**Same Process
for Secondary
Pages**

**Tip: Avoid
Navigation for
Secondary
Pages**

3d Accelerated Transforms

CSS is a Whole
Other Topic

Questions?

June 29 Webinar

Evan Phoenix

Rubinius 1.0

Optional EY Cloud Demo