



PERCONA  
Performance Consulting Experts

---

# High Performance Web UI Design

EdUIConf 2009  
September 22, 2009

Baron Schwartz  
Percona, Inc

# Introduction

- Degree in Computer Science from UVA
- Lead author of High Performance MySQL, 2nd Ed.
- Creator of Maatkit
- Director of Consulting at Percona
  - We help build faster, more scalable applications
  - We're known for MySQL ([mysqlperformanceblog.com](http://mysqlperformanceblog.com))
  - We offer *full-stack* consulting, including front-end performance

# Agenda

---

- What is performance?
- How to think about Web UI performance
- Causes of slow performance, solutions to them
- Resources

# Performance Fundamentals

- All that matters is time & tasks
  - Concurrency, throughput, etc all reduce to time & tasks
- Tasks are things the user does
  - User = browser
  - User = person
- Time is spent doing two things
  - Executing
  - Waiting to execute

# What Is Performance?

---

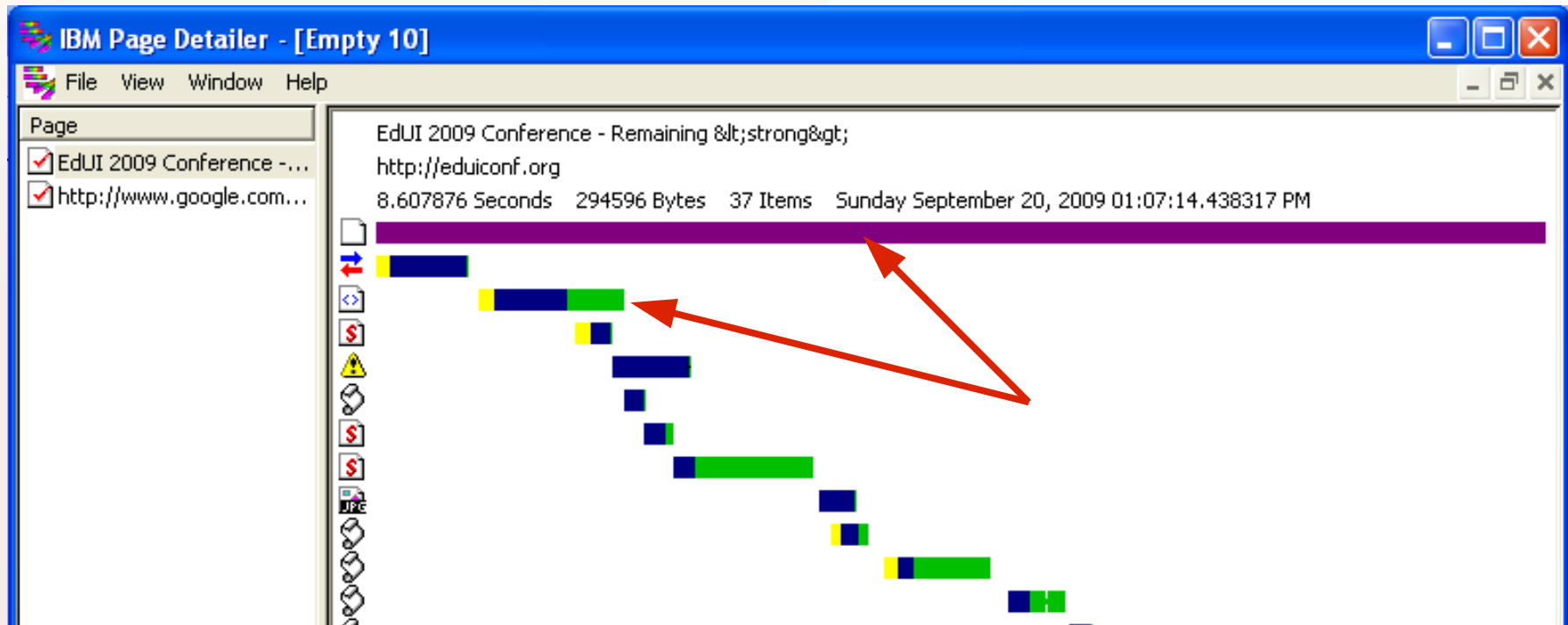
- Client-side (front-end) performance is easy
  - With minimal effort, you can get big wins
  - Simple rules, easy for anyone to apply
- Web UI Performance is more complex
  - There's browser, content, and functionality performance
  - We have control over content and functionality

# Performance Optimization

- You cannot optimize what you cannot measure
- Measure First
  - Identify the tasks
  - Identify (measure) where the time goes
- Then Optimize
  - Eliminate tasks
  - Speed up tasks
  - Enable parallelism

# The Truth About Download Time

- The HTML document takes only 10-20% of the total download time



# Common Problems

---

- Too many or too slow HTTP requests
- Delayed rendering
- Delayed functionality
- AJAX overload or Javascript overload

# A Step-By-Step Approach

---

- Get the content to the browser as quickly as possible
- Once there,
  - Make the content render in the browser as quickly as possible
  - Make the functionality available as soon as possible
  - Fix slow-performing Javascript thereafter
- Let's look at some problems and how to solve them.

# Too Many HTTP Requests

- Causes
  - Too many image files, CSS files, Javascript files
  - Redirects
- Solutions
  - Combine images, stylesheets, Javascript files
  - Let the browser cache
    - Add an Expires header
    - Don't let file versioning defeat caching
    - Use cookies, but cautiously
  - Avoid redirects

# Example of YSlow

http://www.eduiconf.org/

Inspect

Console HTML CSS Script DOM Net **YSlow**

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

**Grade D** Overall performance score 65 Ruleset applied: YSlow(V2) URL: http://www.eduiconf.org/

**ALL (22)** FILTER BY: [CONTENT \(6\)](#) | [COOKIE \(2\)](#) | [CSS \(6\)](#) | [IMAGES \(2\)](#) | [JAVASCRIPT \(4\)](#) | [SERVER \(5\)](#)

E <b>Make fewer HTTP requests</b>
F
<a href="#">Use a Content Delivery Network (CDN)</a>
F <a href="#">Add Expires headers</a>
F <a href="#">Compress components with gzip</a>
A <a href="#">Put CSS at top</a>
D <a href="#">Put JavaScript at bottom</a>
A <a href="#">Avoid CSS expressions</a>
N/A <a href="#">Make JavaScript and CSS external</a>

**Grade E on Make fewer HTTP requests**

This page has 9 external Javascript scripts. Try combining them into one.  
This page has 4 external stylesheets. Try combining them into one.  
This page has 11 external background images. Try combining them with CSS sprites.

Decreasing the number of components on a page reduces the number of HTTP requests to render the page, resulting in faster page loads. Some ways to reduce the number of components include: combine files, combine multiple scripts into one script, combine m CSS files into one style sheet, and use CSS Sprites and image maps.

[»Read More](#)

Copyright © 2009 Yahoo! Inc. All rights reserved.

# Slow HTTP Requests

- Causes
  - Bloated page components
  - Inlined styles, Javascript files
  - Old-style code: <FONT>, nested tables
- Solutions
  - Use minimal, standards-compliant, semantic code
  - Compress and minify components
  - Use a Content Delivery Network
  - Use Keep-Alive
  - Don't use too many hostnames

# Serialized HTTP Requests

---

- Causes
  - Browsers don't download in parallel with scripts
  - Demo
    - <http://stevesouders.com/hpws/move-scripts.php>
- Solutions
  - Put Javascript `<script>` elements at the bottom



# Delayed Functionality

- Causes
  - Scripts at the top
  - Ad-serving, link-previewing, click-tracking
  - Using document.onload
  - Using Javascript instead of CSS
- Solutions
  - Move scripts to the bottom
  - Load non-essential items after the page is done loading
  - Use document.ondomready
  - Toggle Adblock on/off and look for a difference
  - Use CSS for menus ([alistapart.com](http://alistapart.com), [cssplay.co.uk](http://cssplay.co.uk))

# AJAX and Javascript Overload

---

- **Causes**
  - Inefficient programming algorithms
  - Doing unnecessary work
- **Solutions**
  - Profile your Javascript
  - Cache results of computations and re-use them
    - Including, but not limited to, AJAX calls

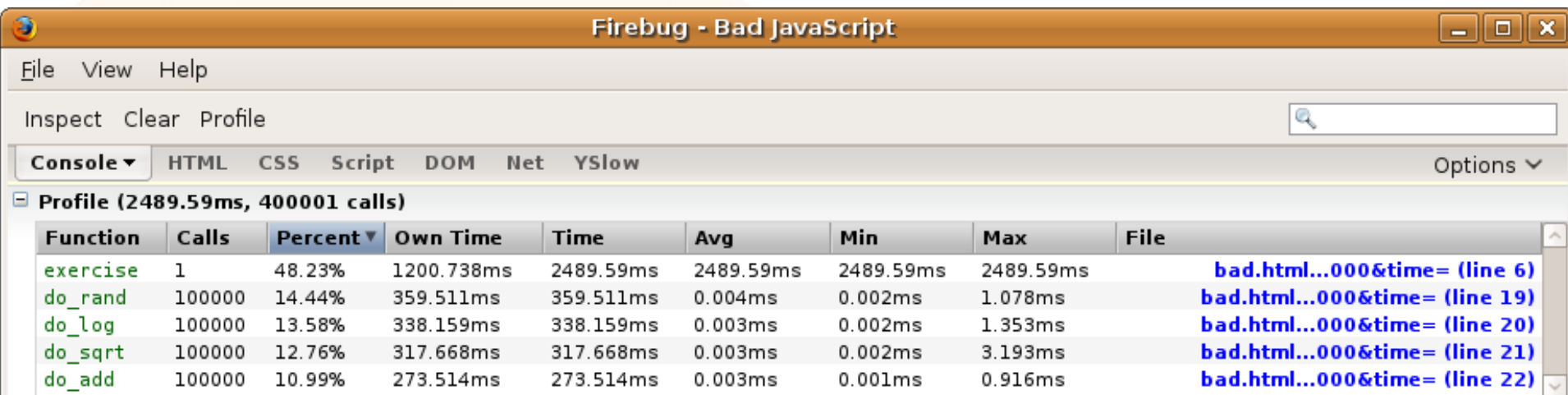
# Profiling Javascript

- Measure, don't guess
- Firebug can profile Javascript at the function level
- Easy to use:
  - Switch to the Console
  - Click Profile
  - Use the interface
  - Click Profile again
- Results are a nice profile table

# Where's the Bottleneck?

```
<html>
<head>
<title>Bad JavaScript</title>
<script language="javascript">
function exercise(form) {
    var iterations = form.iterations.value;
    var start = new Date();
    for ( var i = 0; i < iterations; ++i ) {
        var rand = do_rand();
        do_log(rand);
        do_sqrt(rand);
        do_add(rand);
    }
    var end = new Date();
    form.time.value = end - start;
    return false;
}
function do_rand() { return 1 + Math.random() * 1000; }
function do_log(r) { return Math.log(r); }
function do_sqrt(r) { return Math.sqrt(r); }
function do_add(r) { return r + r; }
</script>
</head>
<body>
<form onSubmit="return exercise(this)">
Iterations: <input type="text" name="iterations" value="1000" /> <br />
Time: <input type="text" name="time" value="" />
<input type="submit" />
</form>
</body>
```

# Profiling Results



The screenshot shows the Firebug interface with the 'Profile' tab selected. The profile shows a total time of 2489.59ms and 400001 calls. The table below lists the functions and their performance metrics.

Function	Calls	Percent	Own Time	Time	Avg	Min	Max	File
<code>exercise</code>	1	48.23%	1200.738ms	2489.59ms	2489.59ms	2489.59ms	2489.59ms	<a href="#">bad.html...000&amp;time= (line 6)</a>
<code>do_rand</code>	100000	14.44%	359.511ms	359.511ms	0.004ms	0.002ms	1.078ms	<a href="#">bad.html...000&amp;time= (line 19)</a>
<code>do_log</code>	100000	13.58%	338.159ms	338.159ms	0.003ms	0.002ms	1.353ms	<a href="#">bad.html...000&amp;time= (line 20)</a>
<code>do_sqrt</code>	100000	12.76%	317.668ms	317.668ms	0.003ms	0.002ms	3.193ms	<a href="#">bad.html...000&amp;time= (line 21)</a>
<code>do_add</code>	100000	10.99%	273.514ms	273.514ms	0.003ms	0.001ms	0.916ms	<a href="#">bad.html...000&amp;time= (line 22)</a>

# Conclusion

- Identify tasks
- Measure first, then optimize
- Make the following as fast as possible
  - Delivering the content
  - Rendering the content and enabling functionality
  - UI performance post-load
- How do you do that?
  - Eliminate tasks
  - Speed up tasks (measure, don't guess!)
  - Do tasks in parallel

# Resources

- Yahoo! developer labs has very good information
  - <http://developer.yahoo.com/performance/>
- Steve Souders' book website has great examples
  - <http://stevesouders.com/hpws/>
- Indispensable tools
  - <http://getfirebug.com/>
  - <http://developer.yahoo.com/yslow/>
  - <http://alphaworks.ibm.com/tech/pagedetailer>
- Expert help
  - <http://www.percona.com/>