

Performance & the Toolkit

Overview

1. Course Updates
2. CSS Spriting
3. Performance Tips
4. Tools & Plugins



Course Updates

Quiz 3 grades are not up. 😞

Tonight or tomorrow morning. Probably tonight. Sorry!

First project checkpoint is on **Nov 13.**

...expect email soon!

Questions about anything so far?

Today's lecture is all about practical quick front-end engineering tips.

- Reducing the page load time
- Reducing your development time



Today's lecture is all about practical quick front-end engineering tips.

- **Reducing the page load time**
- Reducing your development time



Only 10-20% of the user's time is usually spent downloading the actual HTML page.

Where's the rest of the time go?

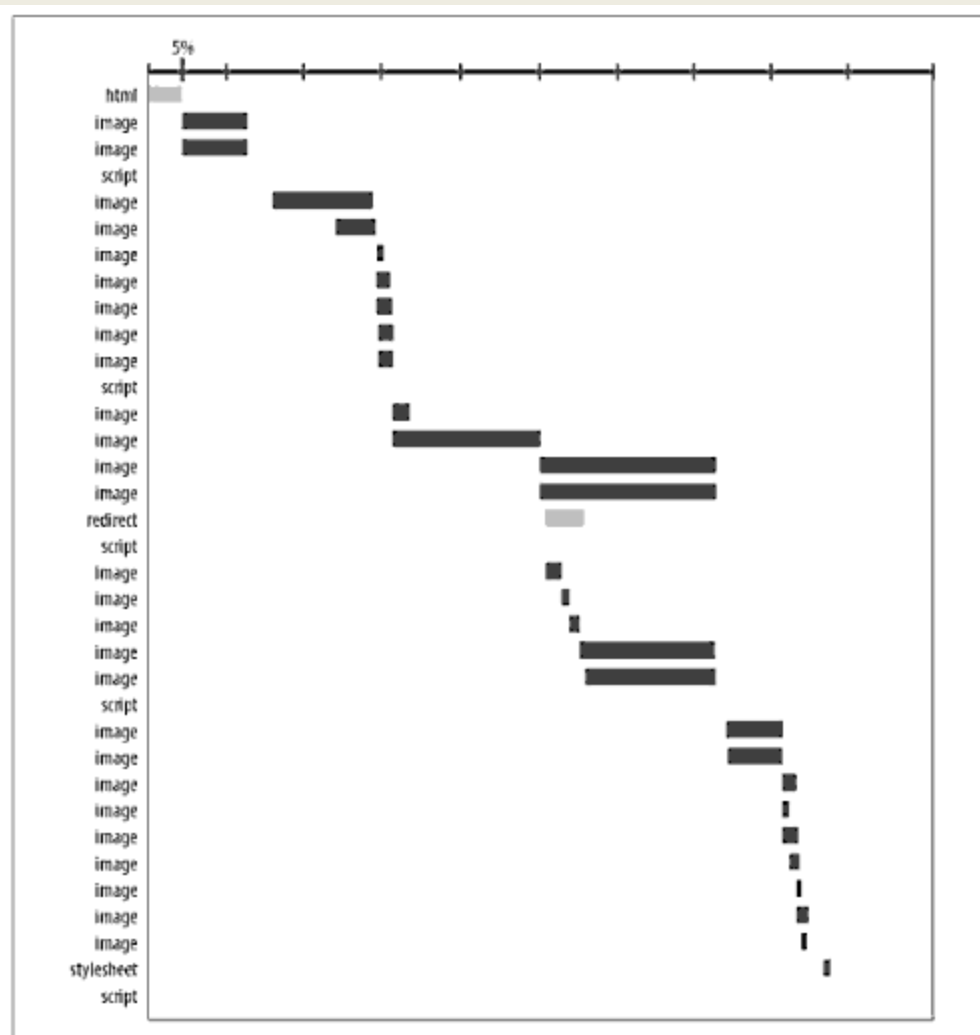


Only 10-20% of the user's time is usually spent downloading the actual HTML page.

Where's the rest of the time go?

Making requests for and downloading everything else: images, Javascript files, CSS files, etc.





Yahoo's home page loading for the first time in IE



So most of our energy should be spent on minimizing the amount of time we spend on additional files and images that our page relies on.

We'll cover a lot of small ideas, but one big one helps a lot and is much more complex than the others:

“Spriting” images using CSS.



CSS Spriting

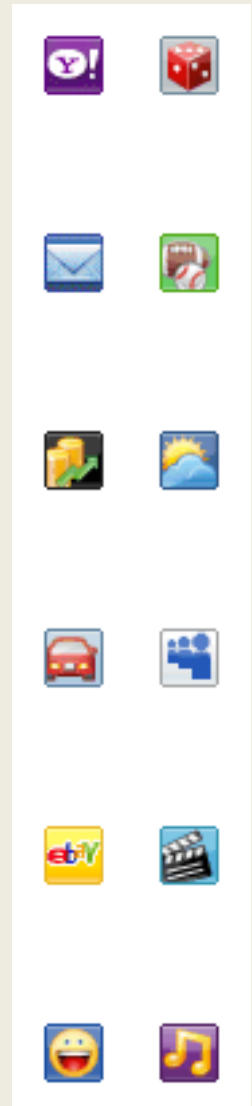
Idea: *reduce the number of requests (separate downloads) that the browser has to make by putting many images that we use in CSS into one big file.*

A huge speed gain because:

- Reduces the number of HTTP requests.
- Allows parallelism in browsers that limit simultaneous downloads.



Sprites in the Wild...



How-to

So we have one big image.

Can separate out individual images using the background-position property.

This can be a little tricky sometimes for tiled images, as we'll see.



Hands-on (sort of)

The best way to learn how to sprite is to do it.

So let's go off the slides and do it.

<http://tonypoor.com/98|30/examples/sprite/icons.htm>

(you can also check out <http://css-tricks.com/css-sprites/> for a tutorial)



Spriting is often used for mouseover effects, like when you roll your mouse over a button and it changes its appearance.



This has an additional benefit: not only do we reduce the number of HTTP requests, but we also make sure that *every state* of the button is loaded at the start.

So browser doesn't have to load the "hover" state after user first moves his mouse over (this would lead to a flicker or delay).



Let's see this in action, too!

<http://tonypoor.com/98|30/examples/sprite/>



Other Performance Tips

Yahoo! engineers developed a very practical list of 34 [performance guidelines](#). Some key ideas:

- Fewer HTTP requests
- Content delivery networks (CDN)
- Stylesheets in the <HEAD>
- Javascript at the bottom
- Minify Javascript/CSS
- Scaling in HTML
- Use PNGs instead of GIFs

Fewer HTTP Requests

Spriting is a great idea. How else can we reduce the number of requests we make?

Fewer HTTP Requests

Combine all of our CSS into one .css file, and all of our Javascript into one .js file.

This is sometimes a good idea. When wouldn't it be?

Fewer HTTP Requests

Combine all of our CSS into one .css file, and all of our Javascript into one .js file.

This is sometimes a good idea. When wouldn't it be?

Probably not the best idea when you don't *need* all the different files on every page of your site.

Content Delivery Networks

CDN = a network of servers spread across many locations... idea is to pick the fastest or closest server to the user and serve content from that

Usually you want to go through a dedicated CDN provider, like **Akamai**.

But this can be kind of expensive and definitely not practical for a small site. So why do I bring it up?

The Google jQuery CDN

This is handy for those of you who use jQuery (*which is quite a few of you, I believe*).

Google hosts the jQuery source for you on its CDN!

```
<script type="text/javascript" src="jquery.js"></script>
```



```
<script type="text/javascript"  
    src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js">  
</script>
```



The Google jQuery CDN

Since everybody uses this, not only do you get the standard CDN benefits, but chances are the user might already have your jQuery cached!

So might not even have to load it at all.



CSS and Javascript Placement

Two quick things:

1. Put stylesheets in the HEAD.
2. Put non-vital Javascript at the bottom of the BODY.



CSS and Javascript Placement

I. Put stylesheets in the **HEAD**.

You're supposed to do this anyway.

Some browsers (IE) block rendering until they get the CSS so that they don't have to re-render later.

So your page isn't loading faster, but it *looks like it because it renders progressively*.

CSS and Javascript Placement

2. Put Javascript at the bottom of the **BODY**.

Unlike CSS (which blocks progressive rendering for anything before it), Javascript blocks progressive rendering for anything *after it*.

Browsers will wait until Javascript is done executing before they continue rendering.

CSS and Javascript Placement

2. Put Javascript at the bottom of the **BODY**.

Sometimes you really can't do this...

when JS is vital and needs to be one of the first things the page loads

...but most of the time you can

ie, some jQuery code that just adds fancy behavior to your menus (*unless* user needs to use that menu instantly)

Minify your JS/CSS

Minification removes unnecessary characters from your Javascript and CSS code.



Minify your JS/CSS

Makes it ugly and unreadable, so don't use minification when you're developing. Keep an unminified version for development purposes.

[YUI Compressor](#) is a great tool for this.

Scaling Images

Make the image the exact size that you need it, *before* you use it in your HTML.

If you're just showing a tiny little thumbnail, there's no need to make the user load the entire 2.5 megabyte high res photo!

The Magic of PNGs

.png images are awesome.

They **support alpha transparency** (semi-transparent images) rather than binary transparency (either fully visible or fully opaque), but IE has some historical problems with that little feature.

PNGs actually tend to be way smaller than GIFs, and all browsers support the *normal* non-alpha transparent PNG-8 version just fine. So give it a chance!

With that out of the way...

Who here has used Firebug?

Today's lecture is all about practical quick front-end engineering tips.

- Reducing the page load time
- **Reducing your development time**

Some Other Useful Tools

Development Ideas & Code

- CSS Resets
- IE7 Javascript

Programs & Tools

- YSlow (Firefox Plugin)
- Firebug (Firefox Plugin)
- W3C HTML Validation
- IETester
- JSLint

Some Other Useful Tools

Development Ideas & Code

- CSS Resets
- IE7 Javascript

Programs & Tools

- YSlow (Firefox Plugin)
- Firebug (Firefox Plugin)
- W3C HTML Validation
- IETester
- JSLint

CSS Reset: what is it?

Remember, all browsers have default stylesheets that they apply.

This is how a **** tag is bold by default and how there is **margin between your <P> tags** without you explicitly defining any.

The problem is browsers have *different defaults*.

CSS Reset: what is it?

The idea is to *get rid of all defaults* by using some CSS that “resets” everything back to factory conditions.

Pro: You don't need to worry about differences between browser defaults, so you have a good starting point for all your CSS.

CSS Reset: what is it?

You can find premade CSS reset code out there:

- [Eric Meyer's reset](#)
- [Yahoo!'s YUI reset](#)

Let's take a look.

Some of them are really big, but a lot of times all you *need* to do is reset the padding and margin to 0.

*** { padding: 0; margin: 0; }**

The IE Problem

Quick poll.

What's wrong with IE5 and IE6?

The IE Problem

My answer:

Basically everything.

IE7 really isn't *that bad*, and IE8 is actually fairly good.

The IE Problem

Some possible solutions:

- Break your neck with hacks getting IE6 to display your site 100% correctly.
- Don't bother with IE6 support at all.
- Go somewhere in between.

The IE Sorta-Solution

There's a small Javascript file floating out there that can actually fix a lot of problems with IE6 by giving it most of the CSS features of IE7.

It's called (you guessed it) **IE7**... the script.

<http://dean.edwards.name/IE7/>

The IE Sorta-Solution

Just include this:

```
<!--[if lt IE 7]>  
<script src="http://ie7-js.googlecode.com/svn/version/2.0(beta3)/IE7.js"  
      type="text/javascript">  
</script>  
<![endif]-->
```

Part of a **conditional comment**; all browsers besides those below IE7 will ignore everything between them.

IE7.js Features

- Some support for alpha-transparent PNGs (doesn't work with *tiled* background PNGs)
- Support for a ton of selectors (like the **parent > child** selector).
- Support for **margin: auto**.
- Support for **position: fixed**.

Other Stuff

Other really useful tools include **CSS frameworks** (YUI Grids, YAML, ...) and **Javascript libraries** (jQuery, Prototype, ...).

Won't really talk much about these.

Most CSS frameworks focus on easy grid creation, with some other handy prepackaged features.

Some Other Useful Tools

Development Ideas & Code

- CSS Resets
- IE7 Javascript

Programs & Tools

- YSlow (Firefox Plugin)
- Firebug (Firefox Plugin)
- W3C HTML Validation
- IETester
- JSLint

Some Other Useful Tools

Sent a **how-to** on these out in an email.

For now, let's do a demo of Firebug, since it's by far the **most important tool for a web developer** right now.

If you're already familiar with Firebug, that's cool – this is the last part of today's lecture, so you can call it quits.