HOW-TO:

Optimize your website speed

Briefly guide on how to optimize your website speed.

Artmov

R.002

CONTENTS

1. Stylesheets

Don’t use CSS expressions.

CSS expressions are a great way to make CSS properties more dynamic for Internet Explorer, but they have poor rendering performance.

These are basically JavaScript expressions embedded into CSS properties.

Here’s how a CSS expression looks like:

```
#oDiv {
          left:expression(document.body.clientWidth/
             2-oDiv.offsetWidth/2);
}
```

Expressions applies to IE 5.5 - 7. From IE8 they are no longer supported.

The IE browser reevaluates each expression whenever an event is triggered, such as:

- mouse movement;
- window resize;
- etc.

It's highly recommended that you don't use at all CSS expressions. Even if it applies to IE users only because of the multiple performance issues it causes.

If you do need to use them, try to replicate that with JavaScript and activate the new rules only when needed.

Don’t use CSS @import.

You can embed CSS into your pages either by using the *link* tag or the *@import* rule.

```
<link rel=”stylesheet” type=”text/css”
        href=”master.css” />
```

```
@import url(‘master.css’);
```

The *@import* rule must be added at the start of your stylesheet or the *style* tag to work.

Here’s how:

```
@import url(‘master.css’);

body {
        background:#FFF;
}
```

Internet Explorer doesn't load any stylesheets from the *@import* rule in a parallel download as other browser do, but sequentially.

This means that your page needs more time to render on IE since it takes each file at a time to download and display.

Look for any *@import* rules in your stylesheets and remove them.

```
@import url('master.css');
```

Make sure that you only use the *link* tag to include your stylesheets; this way everything gets downloaded in parallel.

Do you use Google Webfonts?
Make sure that you load the fonts locally from your own stylesheets or from Google's servers, but with the *link* tag method, not the *@import*.

Combine external CSS.

Try to combine any stylesheets you have into one single *.css* file rather than using multiple stylesheet files.

Especially if the stylesheets have a low file size.

It's a good practice to use multiple stylesheets when your site is in development, but in the production make sure you don't use more than three stylesheets per page.

Two stylesheets is a good choice; three being the maximium recommended.

Use efficient CSS.

You can optimize and keep a much lower file size of your stylesheets if you use an efficient method for writing your CSS selectors.

One way to do it is to remove any redundant information you may have in your CSS files.

Here's an example of that:

```
form#login {
        background:#FFF;
}
div#list {
        border:1px solid red;
}
```

In the above example, you don't need to add "*form*" tag along with the "*login*" *ID* since *ID* is already a unique selector.

The same approach applies to any element you'd have with a *ID*. (e.g. *div, ul, p*)

Change the selectors of your stylesheets:

```
#login {
	background:#FFF;
}
#list {
	border:1px solid red;
}
```

Also:

IE7, IE8 may cause performance issues if you use the *:hover* selector on non-link elements, such as *div*, *p* etc. (anything except *a*)

```
p:hover {
	border:1px solid red;
}
```

Remove unused CSS.

Removing CSS selectors and properties that aren't necessary reduces your file size and it gives the stylesheet a faster rendering time.

Lets say you have a *master.css* file, for example, that uses 90% of its code for the inside pages and only 10% for your home page.

It may be a good start, to boost your home page, to have a *homepage.css* file that uses the 10% of code, while the 90% is moved in the *master.css*, which is loaded whenever you have an inside page.

The downside of this is that, if you have a high ratio of users that are going from home to any inside page, you can't cache *master.css* from the home page without loading it.

Find the best approach depending how your users navigate through your site.

Minify CSS.

Compacting CSS (*and caching it*) can improve the download and execution time of pages.

An approach is to keep a development version of your CSS that is easily readable and go with a minified, compact version whenever you are in the production state.

One method to compact your CSS files is to remove any white spaces.

If your current selector is like this:

```
#login {
        background:#FFF;
}
```

Now, simply remove any space around it:

```
#login {background:#FFF;}
```

Tools that you can use to minify CSS.

Clean CSS

http://www.cleancss.com/

Online YUI Compressor

http://www.refresh-sf.com/yui/

zBugs

http://www.zbugs.com/minify-gzip-compress-css-javascript-online

2. Scripts

Avoid document.write in JavaScript.

This applies whenever you use *document.write* rule to download additional resources.

Here's an example of that.

Lets say you have a *hello.js* file with the following code:

```
document.write('<script src="hi.js"><\/script>');
```

In this case the browser serialize the resources it needs to download for your site:

• download and parse *hello.js*;
• download and parse *hi.js*.

Try to minimize any usage you have on the *document.write* rule to download additional resources.

Use the *script* tag instead and, rather than call *hi.js* with *document.write*, add a new script tag in your HTML.

```
document.write('<script src="hi.js"><\/script>');
```

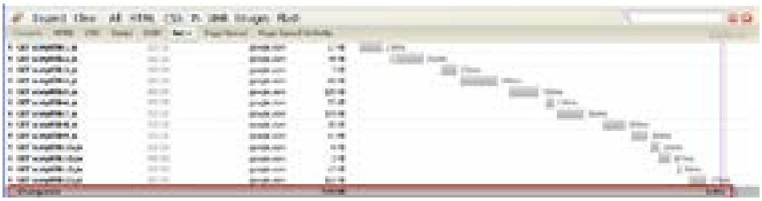
The above code with the *script* tag:

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

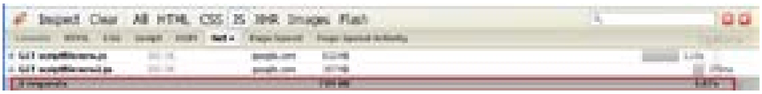
Combine external JavaScript files.

Combine any external JavaScript files into a single file, just as you should try to do with the stylesheets.

Below are some screenshots from the *Google Developers* page that shows exactly how much it takes to download 13 separate scripts: *4.46s*.



Combining them into 2 files, reduces the time to only *1.87s*.



Optimize JavaScript functions.

Try to check if you have multiple JavaScript functions that are called in a page, but aren't required until at a later stage (e.g. *initial loading vs. clicking item*).

If so, it's best to call them when you actually need them.

37signals improved their page load times by profiling the JavaScript code and started to call functions when needed and not on the initial loading stage.

```
$.ready(function() {
  $("article.todolist, section.todolists").sortable();
});
```

This line of code automatically makes their app's to-do lists to be sortable.

It took up to half a second (for heavy pages) to load that.

They've also saw that a user doesn't reorder the to-do list on each page visit.

They've successfully implemented an approach that worked and managed to decrease the time from *555ms* (before) to *93ms* (after).

37signals's article:

http://37signals.com/svn/posts/3137-using-event-capturing-to-improve-basecamp-page-load-times

Minify JavaScript scripts.

Minify your JavaScript files, as you should do with the stylesheets, and get a lower page size.

Here are a few online tools for that:

Online JavaScript Packer

http://jscompress.com/

JavaScript Compressor

http://www.minifyjs.com/javascript-compressor/

Online YUI Compressor

http://www.refresh-sf.com/yui/

3. Images

Combine images with CSS sprites.

Combine your images using the CSS Sprites method to reduce the overall number of round-trips and delays in download.

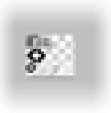
Here's how our first e-book tackled this.

HOW-TO: **Convert PSD to HTML/CSS** Download

The design packed along with this guide, *Monoplate* as it was named, had a series of icons for the home page:

• social icons (*Facebook*, *Twitter*)
• blog icons (*date*, *comments*)
• slider icons (*left*, *right*)

All icons were needed for the home page, so we've used the CSS sprites technique and compact all the above icons in one *ico.png*:



Take in consideration your site's traffic and page usage, when you're implementing the CSS sprites technique.

If you use a *ico.png* file for the sprites, but most of it is required for your forum icons and not the home page or inside pages, there's no point in calling it on the home page or inside pages, especially if these pages are the most visited.

Find a balance between how you group the icons in an image and on what pages is that image required the most.

Tips to reduce the sprites image size:

- reduce empty spaces in the sprites image
- use a 256 colors pallet when saving

Choose the correct images format.

Find the right image format. It can have a drastic impact on a page total size.

Recommendations:

- use JPEG for photographic pictures;
- use GIF for small and simple graphics;
- use PNG when you need superior quality in transparency and colors.

Correctly set images sizes.

Along with the correct format for images, try to correctly set the images sizes. It's going to be a great boost in page speed.

Particularly if you use many high-res pictures that you later resize from HTML or simply from CSS.

Lets say, for example, that you have a *500x500* image that's saved on the server, but you resize it to *100x100* from HTML or CSS.

Resize and save as *100x100* and not *500x500*.

Setting the correct size gives a better load time for pictures.

Optimize images.

Many bytes can be saved if you compress your images properly, before releasing your site to the public.

PNG and JPEG can be optimized using online tools that compresses their data.

A few online tools that can do that:

PNG Optimizer

http://www.pngoptimizer.com/

PunyPNG

http://punypng.com/

Smush.it

http://www.smushit.com/ysmush.it/

4. HTML

Optimize the order of stylesheets and scripts.

Another great boost in website page speed can be achieved if you correctly set the order of your stylesheets and scripts.

It’s good to have the stylesheets followed by your scripts afterwards.

Some browsers block the downloading of additional scripts or stylesheets until a current script is downloaded and parsed.

An example:

```
<link href=”1.css” rel=”stylesheet” type=”text/css” />
<script type=”text/javascript” src=”1.js”></script>
<link href=”2.css” rel=”stylesheet” type=”text/css” />
```

So the browser downloads the *1.css*, then *1.js* in parallel, but *2.css* won't begin downloading until *1.js* is done (download and parse).

Change the order of your stylesheets:

```
<link href="1.css" rel="stylesheet" type="text/css" />
<link href="2.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="1.js"></script>
```

It's best to always have the stylesheets before any *script* tag.

Move stylesheets at the top.

Quite connected with the previous trick is to move all of your stylesheets at the top of the HTML document, in the *head* section:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>[title]</title>
<link href="1.css" rel="stylesheet" type="text/css" />
<link href="2.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="1.js"></script>
</head>
<body>

</body>
</html>
```

Your stylesheets shouldn't be before the closing *body* tag of HTML:

```
<link href="1.css" rel="stylesheet" type="text/css" />
<link href="2.css" rel="stylesheet" type="text/css" />
</body>
</html>
```

Move scripts at the bottom.

Your scripts should be at the bottom of the document whenever that's possible.

If not, try to find a balance between scripts that needs to be at the top and those can be moved at the bottom.

A good balance is with stylesheets at the top and scripts at the bottom of the document:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>[title]</title>
<link href="1.css" rel="stylesheet" type="text/css" />
<link href="2.css" rel="stylesheet" type="text/css" />
</head>
<body>

<script type="text/javascript" src="1.js"></script>
</body>
</html>
```

Specify image dimensions.

Setting the width and height of your images used in a web page allows the browser to render the page faster.

This eliminates the browser's need to repaint the document since it already knows about the image width and heigh before that image is downloaded.

```
<div id=”image”>
		<img src=”1.png” width=”100” height=”100” />
</div><!-- end #image -->
```

If you specify the size from CSS, make sure you specify it for the image itself and not the parent element.

```
#image img {
		height:100px;
		width:100px;
}
```

Printable **checklists.**

- [ ] Remove CSS expressions
- [ ] Remove CSS @import
- [ ] Combine external CSS
- [ ] Remove unused CSS
- [ ] Make CSS more efficient
- [ ] Minify CSS

- [ ] Avoid document.write in JavaScript
- [ ] Combine external JavaScript files
- [ ] Optimize JavaScript functions
- [ ] Minify JavaScript scripts

- ☐ Combine images with CSS Sprites
- ☐ Choose the correct images format
- ☐ Correctly set images sizes
- ☐ Optimize images

- ☐ The order of stylesheets and scripts
- ☐ Move stylesheets at the top
- ☐ Move scripts at the bottom
- ☐ Specify images dimensions