

Ten ways to speed up the download time of your web pages

Why is download speed so important?

Do you like to wait for pages to download? Neither do your site users. Read on...

1. Lay out your pages with CSS, not tables

CSS downloads faster than tables because:

- Browsers read through tables twice before displaying their contents, once to work out their structure and once to determine their content
- Tables appear on the screen all in one go - no part of the table will appear until the *entire* table is downloaded and rendered
- Tables encourage the use of spacer images to aid with positioning
- CSS generally requires less code than cumbersome tables
- All code to do with the layout can be placed in an external CSS document, which will be called up just once and then cached (stored) on the user's computer; table layout, stored in each HTML document, must be loaded up each time a new page downloads
- With CSS you can control the order items download on to the screen - make the content appear before slow-loading images and your site users will definitely appreciate it

To learn more about CSS and the amazing things it can do for your website, check out the excellent tutorials both here at DMXzone or at places like HTML Dog.

2. Don't use images to display text

It's our old friend CSS to the rescue again. There's no need to use images to display text as so much can be accomplished through CSS. Have a look at this code:

```
a:link.example, a:visited.example, a:active.example
{
color:#fff;
background:#f90;
font-size:1.2em;
font-weight:bold;
text-decoration:none;
padding:0.2em;
border:4px #00f outset
}

a:hover.example
}
color:#fff;
background:#fa1;
font-size:1.2em;
font-weight:bold;
text-decoration:none;
padding:0.2em;
```

```
border:4px #00f inset
}
```

This will give you a really simple button that appears to be pushed down when you mouseover it - See it in action [here](#) if you like.

3. Call up decorative images through CSS

It's possible to present images as part of the background, called up through CSS. If you've got an image that's 200px by 100px you can use the following HTML code:

```
<div class="pretty-image"></div>
```

And this CSS:

```
.pretty-image
{
background: url(filename.gif);
width: 200px;
height: 100px
}
```

This may at first seem a little pointless but **this technique could really improve the download time** of your pages. Browsers basically download background images after everything else. By using this technique, your text will load instantaneously and your site users can freely roam about the page while your 50kb fancy image downloads.

This technique disables the **ALT** attribute though so if you really want to have one then replace the above HTML code with this:

```
<image src="spacer.gif" class="pretty-image" alt="description" />
```

Spacer.gif is a 1px x 1px transparent image. Now you have a 50 byte transparent image *and* the main content loading first, and your great big decorative image, complete with **ALT** text, loading second. Perfect.

Please note that this technique is only good for decorative images and not informational ones. Any user who has CSS disabled will not be able to see your CSS-embedded images (or their alternative text).

4. Use contextual selectors

This is inefficient:

```
<p class="text">This is a sentence</p>
<p class="text">This is another sentence</p>
<p class="text">This is yet another sentence</p>
<p class="text">This is one more sentence</p>

.text
{
color: #03c;
font-size: 2em
}
```

Instead of assigning a value to each individual paragraph, we can nest them within a `<div>` tag and assign a value to this tag:

```
<div class="text">
<p>This is a sentence</p>
<p>This is another sentence</p>
<p>This is yet another sentence</p>
<p>This is one more sentence</p>
</div>

.text p
{
color: #03c;
font-size:2em
}
```

This second CSS example basically says that every paragraph within `class="text"` should be assigned a colour value of #03c and a font size of 2em.

At first glance this doesn't look too important, but if you can apply this properly throughout your document you can easily knock off 20% of the file size.

You may have noticed the colour values are shorter than normal. #03c is a shortened version of #0033cc - you can assign this abbreviated value to any colour value like this.

5. Use shorthand CSS properties

Font

Use:

```
font: 1em/1.5em bold italic serif
```

...instead of:

```
font-size: 1em;
line-height: 1.5em;
font-weight: bold;
font-style: italic;
font-family: serif
```

Border

Use:

```
border: 1px black solid
```

...instead of

```
border-width: 1px;
border-color: black;
border-style: solid
```

Background

Use:

```
background: #fff url(image.gif) no-repeat top left
```

...instead of:

```
background-color: #fff;  
background-image: url(image.gif);  
background-repeat: no-repeat;  
background-position: top left;
```

Margin, padding, border

Use:

```
margin: 2px 1px 3px 4px (top, right, bottom, left)
```

...instead of:

```
margin-top: 2px;  
margin-right: 1px;  
margin-bottom: 3px;  
margin-right: 4px
```

Use:

```
margin: 5em 1em 3em (top, left and right, bottom)
```

...instead of:

```
margin-top: 5em;  
margin-bottom: 1em;  
margin-right: 1em;  
margin-right: 4em
```

Use:

```
margin: 5% 1% (top and bottom, left and right)
```

...instead of:

```
margin-top: 5%;  
margin-bottom: 5%;  
margin-right: 1%;  
margin-right: 1%
```

These rules can be applied to **margin**, **border** and **padding**.

6. Minimise white space, line returns and comment tags

Every single letter or space in your HTML code takes up one byte. It doesn't sound like much but **it all adds up**. We've found that by working through your page source and eliminating unnecessary white space and comments, you can shave off up to, or even over (if your HTML is *really* inefficient) 10% of its file size.

7. Use relative call-ups

Try to avoid absolute call ups as they take up more space. An example of an absolute call up is: ``.

Much better would be ``. But what if some files are in different folders to other ones? Use these shorthand properties:

- `` - Calls up <http://www.URL.com>
- `` - Calls up <http://www.URL.com/filename.html>
- `` - Calls up <http://www.URL.com/directory/filename.html>
- `` - Calls up index.html within that directory
- `` - Calls up index.html one directory above
- `` - Calls up filename.html one directory above
- `` - Calls up index.html two directories above

8. Remove unnecessary META tags and META content

Most **META** tags are pretty much unnecessary and don't achieve very much. If you're interested, you can see a [list of META tags](#) that are available. The most important tags for search engine optimisation are the keywords and description tags, although due to mass abuse they've lost a lot of importance in recent times.

When using these **META** tags try to keep the content for each **under 200 characters** - anything more increases the size of your pages. Lengthy **META** tags are not good for search engines anyway because they dilute your keywords.

9. Put CSS and JavaScript into external documents

To place CSS in an external document use:

```
<link type="text/css" rel="stylesheet" href="filename.css" />
```

To place JavaScript in an external document use:

```
<script language="JavaScript" src="filename.js" type="text/javascript"></script>
```

Any external file is called up just once and then cached (stored) on the user's computer. Instead of repeating JavaScript or CSS over and over again in HTML files, just write it out once in an external document.

And don't forget, there's **no limit** to the number of these external documents that you can use! For example, instead of making one huge CSS document, have one main one and some others that are specific to certain areas of your site.

10. Use / at the end of directory links

Don't do this: ``

Do this instead: ``

Why? If there's no slash at the end of the URL the server doesn't know if the link is pointing to a file or to a directory. By including the slash the server instantly knows that the URL is pointing to a directory and doesn't need to spend any time trying to work it out.

This article was written by Trenton Moss. He knows an awful lot about [accessibility](#) and the [Disability Discrimination Act](#).

Copyright © 2004 Trenton Moss, webcredible All Rights Reserved