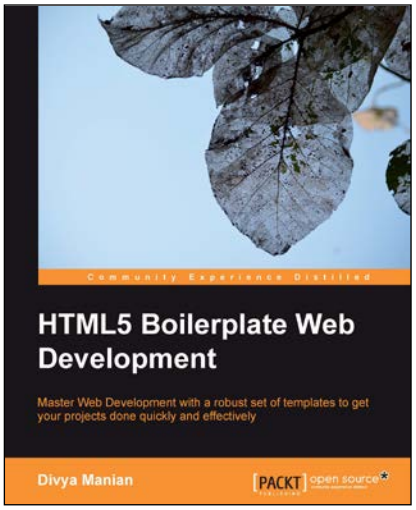
[PACKT] open source
community experience distilled
PUBLISHING

HTML5 Boilerplate Web Development

Divya Manian



Chapter No. 6
"Making Your Site Better"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.6 "Making Your Site Better"

A synopsis of the book’s content

Information on where to buy this book

About the Author

Divya Manian is the co-creator of the HTML5 Boilerplate framework. She has worked on projects to benefit the web development community such as HTML5 Please, Move the Web Forward, and HTML5 Readiness. She is also a member of the W3C. Previously, she used to be an embedded C++ programmer.

I would like to thank Nicolas Gallagher, the lead developer and maintainer of HTML5 Boilerplate for all the work in keeping the project up-to-date, and Paul Irish, co-creator of HTML5 Boilerplate for the initial effort and collaboration in bringing this framework alive.

HTML5 Boilerplate Web Development

Getting Started with HTML5 Boilerplate will enable you to master setting up new projects with minimal effort and deploy them to production in the most effective manner with the least time spent while also ensuring robust performance. It takes you through a step-by-step process of creating a website and teaches you to take full advantage of the defaults provided within HTML5 Boilerplate, be it styles, mark up, or code, so that you can accomplish your goals with as few cross-browser issues as possible.

What This Book Covers

Chapter 1, Before We Begin, covers all you need to get set up for your projects to use HTML5 Boilerplate without much effort. We also broadly look at the files that are included as part of this project and how they help you.

Chapter 2, Starting Your Project, covers how to get started with HTML5 Boilerplate with an example, single page website. In this chapter, we look at the basic essentials of configuring the default setup that works for your project.

Chapter 3, Creating Your Site, covers how to customize the styles and the markup of your website along with some tips on how to take advantage of HTML5 Boilerplate's default style options.

Chapter 4, Adding Interactivity and Completing Your Site, will help you discover how to do feature-detection, add some interactivity with JavaScript, and finalize your website implementation.

Chapter 5, Customizing the Server, looks at how you can ensure that your website gets loaded as quickly as possible by using HTML5 Boilerplate's custom configurations for the web servers that host your site.

Chapter 6, Making Your Site Better, looks at the optional features that can also be used to provide a better experience for the users of your site, which would fit well with HTML5 Boilerplate.

Chapter 7, Automate Deployment With the Build Script, helps you to make your sites ready to be deployed live by looking at the Build Script that provides tools to minify CSS, JS, HTML, and images.

Appendix, You Are an Expert, Now What? covers some basics of unit testing and provides additional research information on some of the decisions that were arrived at for the features that HTML5 Boilerplate provides.

6
Making Your Site Better

The nature of website design and development is such that not all optimizations and recommendations apply in all scenarios. In this chapter, we will look at the various optimization tools available and which situations they are best suited for, to make an HTML5 Boilerplate site load and render faster.

Finding the best experience for Internet Explorer

Internet Explorer versions 8 and below have had very haphazard support for standards and consistent rendering. Depending on the number of users who visit your site using Internet Explorer, you may or may not want to spend the effort optimizing for Internet Explorer.

Mobile-first styles for IE

Media Queries are CSS features that allow you to apply different sets of rules depending on the value of a particular media feature. For example, if the browser has minimum width of `500` pixels, you could make all your `h1` elements turn `red`, as shown in the following code:

```
@media only screen and (min-width: 500px) {
h1 { color: red; }
}
```

However, IE6, IE7, and IE8 do not understand media queries that are typically used to adjust widths according to different screen widths. As such, they will never render the optimized styles you create for browsers with screen widths that match a certain media query break point (`min-width: 500px` in the previous snippet). In our Sun and Sand Music Festival website, we have style rules within three different media queries, as shown in the following code snippet:

```
@media only screen and (max-width: 300px){ /*CSS rules */ }

@media only screen and (max-width: 750px) { /*CSS rules */ }

@media only screen and (max-width: 1150px) { /*CSS rules */ }
```

This means IE6, IE7, and IE8 will render the styles as though these queries did not exist! If you specify rules for device widths that are smaller at the end, it is likely that those will be overriding the rules for device widths that are larger, leading to less than optimal designs on Internet Explorer 8 and below.

Ideally, in this situation, you simply want IE to render all the styles and have the user scroll, if necessary, so that the style rules for the largest widths always apply. To do this, we can create a separate `ie.css` file that will render all the rules within `main.css` except these rules will no longer be contained in media queries.

Doing this manually is hard work, and almost impossible to maintain. However, Nicolas Gallagher writes about an elegant solution he invented, which uses Sass to import separate stylesheets for each media query breakpoint and compile them into two separate stylesheets; one without media queries (`ie.css`) and the other with media queries (`main.css`); we will look at this next.

ie.scss

The code snippet for `ie.scss` is as follows:

```
@import "300-up";
@import "750-up";
@import "1150-up" /* Make sure largest is last */
```

main.scss

The code snippet for `main.scss` is as follows:

```
@import "base";
@media (min-width:300px) {
    @import "300-up"; }
```

```
@media (min-width:750px) {
    @import "750-up"; }
@media (min-width:1150px) {
    @import "1150-up"; }
```

Do note that you need each file titled `300-up.scss`, `750-up.scss`, and `1150-up.scss` within the same parent folder as `main.scss` and `ie.scss`.

In the `head` tag of `index.html` page, you can now write the following code:

```
<!--[if (gt IE 8) | (IEMobile)]><!-->
<link rel="stylesheet" href="/css/style.css">
<!--<![endif]-->

<!--[if (lt IE 9) & (!IEMobile)]>
<linkrel="stylesheet" href="/css/ie.css">
<![endif]-->
```

Jake Archibald also has a far more easy-to-write solution using Sass at `jakearchibald.github.com/sass-ie/`. It takes advantage of newer features of Sass 3.2, and has slightly different composition for `main.scss` and `ie.scss`. It requires advanced knowledge of Sass, which is beyond the scope of this book.

Printing with jQuery in IE6 and IE7

IE6 and IE7 do not support the `:after` pseudo selector that all other browsers support. This means our print stylesheet that provides a feature for all links to be printed alongside the linked text will not work in IE6 and IE7. You can simply use jQuery code to overcome this.

Bill Beckelman has written a post about this on his blog at `beckelman.net/2009/02/16/use-jquery-to-show-a-links-address-after-its-text-when-printing-in-ie6-and-ie7/`. IE supports its own proprietary `onbeforeprint` and `onafterprint` events that can be used to our advantage. Based on Bill Beckelman's work, we can write our own simple jQuery code to print link URLs in IE6 and IE7.

First, we check if `window.onbeforeprint` exists, because this would indicate this code is being executed on one of the IE browsers. We also want to verify if this browser supports generated content or not, as we only need to use this code when it is not supported. The following code snippet checks for the presence of the `window.onbeforeprint` event:

```
if (Modernizr.generatedcontent == false &&window.onbeforeprint !==
undefined) {
```

Then, we set functions to execute when either `onbeforeprint` or `onafterprint` occurs, as shown in the following code:

```
window.onbeforeprint = printLinkURLs;
window.onafterprint = hideLinkURLs;
```

Then, we write the following functions:

```
functionprintLinkURLs() {
$("a[href]").each(function() {
$this = $(this);
$this.data("originalText", $this.text());
$this.append(" (" + $this.attr("href") + ")");
});
}

functionhideLinkURLs() {
  $("a[href]").each (function() {
     $(this).text($(this).data("originalText"));
  });
}
```

Styling disabled form elements in Internet Explorer

Internet Explorer up to version 9 has no way to indicate a form field is disabled other than the color of the text used in that field. Sometimes, a field simply has an icon rather than text (or it could be an empty input textbox), in which case it is almost impossible to discern which buttons are disabled and which are not.

For Internet Explorer 7 and above, by just adding the following rules in `main.css`, you can get the disabled fields to display significantly differently from the enabled ones:

```
.lt-ie9 input[type='text'][disabled],
.lt-ie9 textarea[disabled] {
```

```
background-color: #EBEBE4;
}
```

If you have to support Internet Explorer 6, then make sure you add a class called `disabled` on the form elements that have the `disabled` attribute set and alter the previous rule to the following:

```
.lt-ie9 input.disabled,
.lt-ie9 textarea.disabled {
background-color: #EBEBE4;
}
```

Suppressing IE6 image toolbar

In IE6, all images, have a toolbar visible when hovered over. You can disable them by adding the following code within the `head` tag in the `index.html` file:

```
<metahttp-equiv="imagetoolbar" content="false">
```

Writing CSS3 easier with tools

CSS3 is at the bleeding edge. Some properties require what is known as a vendor prefix. For example, the 3D transforms property `perspective` is implemented as follows, in different browsers:

```
-webkit-perspective //Safari, Chrome
-ms-perspective // Internet Explorer
perspective // Firefox
```

Only a short while ago, Firefox implemented this property as `-moz-perspective`, but have since dropped support for the `-moz-` prefix.

As you will come to realize, it is really hard to keep track of which browser requires a prefix and which browser does not, and it is not quite feasible to keep all the sites that we create updated on a regular basis every time a browser adds or drops support for a prefix.

To make this easier, we could use abstractions without these prefixes such that a tool that has an updated index of which property requires which prefix could convert them into the required final CSS.

This is exactly what Sass (`sass-lang.com`) or Less (`lesscss.org`) provide. Sass is a language that comes with a compiler that converts code written in Sass to CSS, in Ruby. Less is a similar language, but written in JavaScript.

In both cases, the languages are extensions of the syntax used in CSS, which means you can copy your existing CSS files into Sass or Less files and have them compile into pure CSS files without any errors.

The extra features that these languages provide are the ability to use mixins, variables, functions, and more.

For Sass, **Compass** is an additional framework that provides a ready library of CSS3 mixins found at `compass-style.org/reference/compass/css3`. Less has many options; the most popular and frequently updated can be found within Twitter Bootstrap and are available at `twitter.github.com/bootstrap/less.html#mixins`. The following sections show you how to create a rule that uses CSS transforms in Sass and Less.

Sass

The code snippet for Sass is as follows:

```
.btn-arrow {
  @include transform(scale(2));
}
```

Less

The code snippet for Less is as follows:

```
.btn-arrow {
.scale(2);
}
```

Output CSS

The output CSS would be as follows:

```
.btn-arrow {
-webkit-transform: scale(2);
     -moz-transform: scale(2);
      -ms-transform: scale(2);
       -o-transform: scale(2);
transform: scale(2);
}
```

Converting HTML5 Boilerplate CSS to Sass or Less

You could typically just rename the `main.css` file to `main.scss` or `main.less` and start using that as your base Sass or Less file. To compile these files to the corresponding Less or Sass files, you can either use GUI-based browser refreshing software that compiles these files automatically like **LiveReload** (`livereload.com/`) or **Codekit** (`incident57.com/codekit`).

If you are someone familiar with the command line, you can install Less or Sass and run their respective command-line interpreters to compile the files into pure CSS.

If you wish to use a pure Sass or Less file to start with (instead of the contents of the `main.css` file), there are also forks of HTML5 Boilerplate that have the stylesheet converted to Sass. We will see two of them in the following sections.

HTML5 Boilerplate Compass extension

There is a Compass extension that is available for use with Compass at `github.com/sporkd/compass-html5-boilerplate`. Note that it is not as frequently updated as the `main.css` file you find in HTML5 Boilerplate. This is extensively modularized and splits the `main.css` file into multiple Sass files. The CSS comments are also removed from the resulting CSS file.

HTML5 Boilerplate Sass fork

There is a Sass fork of the `main.css` that is frequently updated at `github.com/grayghostvisuals/html5-boilerplate/tree/h5bp-scss` that you can use, if all you want is a base Sass file to start from. This version uses Sass variables but does not split the file into individual files.

Unfortunately, there is no up-to-date Less fork of HTML5 Boilerplate. However, you can rename the `main.css` to `main.less` and then use it as a Less file.

Print considerations

If your web page is likely to be printed, you might want to consider using colors that are printable. Some browsers consider some colors too light to print and force a darker version of the color for printing; `merttol.com/articles/code/too-light-for-print.html` has more details on this interesting quirk.

Appendix, *You Are an Expert, Now What?*, covers the reasoning and rationale behind the print styles in great detail.

Finding and using polyfills

Most of HTML5 and CSS3 features have differing levels of support in different browsers, hence, either use JavaScript code to mimic these features in browsers that do not support them or provide an altering view. Such snippets of code are called **polyfills**.

I help maintain `html5please.com` which is an opinionated list of polyfills for some popular HTML5 and CSS3 features.

Beware of the performance penalty of using a lot of polyfills on a browser that does not support a lot of features.

When you do use your polyfills, make sure you use Modernizr's `load` function like we did for the audio polyfill for Sun and Sand Music Festival website in *Chapter 4, Adding Interactivity and Completing Your Site*. This would prevent unnecessary loading of polyfills on browsers that support the features you want to use.

A comprehensive list of all kinds of polyfills is available on the Modernizr Wiki at `github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills`.

Making your site faster

If your pages use a lot of resources such as images, then perhaps it would be wise to prefetch those resources so your page loads faster. **DNS prefetching** would be a way to do that.

DNS prefetching

DNS prefetching informs the browser of resources on other domain names that are referred to within the page early on during the page load, so it can do the DNS resolution of these domain names.

A browser has to look up a domain name on a **Domain Name Server** (**DNS**) to figure out where it is located on the Internet. Sometimes, it has to go through multiple layers of Domain Name Servers and it could be very slow and is not always consistent. By using DNS prefetching, even before a user clicks a link, or loads a resource, the DNS resolution for that particular domain name is done and the resource can be fetched much faster.

Google states that this saves about 200 milliseconds on a resource hosted on an external domain name.

If you host your assets on a **Content Delivery Network** (**CDN**) like Amazon's S3 or even if you refer to Google's API or Microsoft's API CDN, it would be faster to get these files when they are prefetched.

DNS prefetching is invoked by writing the following code within the `head` tag of a HTML file:

```
<link rel="dns-prefetch" href="//image.cdn.url.example.com">
```

Browsers that understand prefetching would immediately start attempting to resolve the DNS for the link within the `href` attribute. The following is how it would look for Amazon S3:

```
<link rel="dns-prefetch" href="//s3.amazonaws.com">
```

Currently, Firefox 3.5 and higher, Safari 5 and higher, and IE9 and higher, support DNS prefetching.

Making your site more visible on search engines

While the content of your website matters the most, making sure everything else supports better visibility of the content on search engines is important too. The following sections explain some ways you can do this.

Directing search spiders to your site map

Site maps inform search engines of the existence of pages within your site that are otherwise not discoverable; perhaps they are not linked to from other pages on your site, or from external sites.

Some CMSs provide plugins to generate site maps, listed at `code.google.com/p/sitemap-generators/wiki/SitemapGenerators`, or you can write one yourself using the guidelines at `www.sitemaps.org/protocol.html`.

Once you have written your site map, you can let search engine spiders discover it when they crawl your website if you add a link to the sitemap by using the following:

```
<linkrel="sitemap" type="application/xml" title="Sitemap" href="/
sitemap.xml">
```

You can also submit the site map to individual search engines instead of linking to the site map within the HTML page, if you would like to make your page as small as possible.

Implementing X-Robots-Tag headers

You will likely sometimes have a staging server, such as `staging.example.com` for your site `example.com`. If an external site links to the files on the staging server (say you were asking a question about some feature not working on a forum and link to the staging server), it is likely to be indexed by search engines even though the domain name does not figure in the `robots.txt` file or does not hold a `robots.txt` file.

To prevent this, you can add `X-Robots-Tag` HTTP header tags by appending and uncommenting the following code snippet to the `.htaccess` file on the staging server:

```
# ----------------------------------------------------------------------
# Disable URL indexing by crawlers (FOR DEVELOPMENT/STAGE)
# ----------------------------------------------------------------------

# Avoid search engines (Google, Yahoo, etc) indexing website's content
# http://yoast.com/prevent-site-being-indexed/
# http://code.google.com/web/controlcrawlindex/docs/robots_meta_tag.
html
# Matt Cutt (from Google Webmaster Central) on this topic:
# http://www.youtube.com/watch?v=KBdEwpRQRD0

# IMPORTANT: serving this header is recommended only for
# development/stage websites (or for live websites that don't
# want to be indexed). This will avoid the website
# being indexed in SERPs (search engines result pages).
# This is a better approach than using robots.txt
# to disallow the SE robots crawling your website,
# because disallowing the robots doesn't exactly
# mean that your website won't get indexed (read links above).

# <IfModulemod_headers.c>
#   Header set X-Robots-Tag "noindex, nofollow, noarchive"
#   <FilesMatch "\.(doc|pdf|png|jpe?g|gif)$">
#     Header set X-Robots-Tag "noindex, noarchive, nosnippet"
#   </FilesMatch>
# </IfModule>
```

Trailing slash redirects

Search engines consider folder URLs `http://example.com/foo` and `http://example.com/foo/` as two different URLs and as such would consider the content to be duplicates of each other. To prevent this, rewrite the URLs either to change `http://example.com/foo` to `http://example.com/foo/` or `http://example.com/foo/` to `http://example.com/foo`.

The way we do this is to edit the `.htaccess` file for Apache server and add the following rewrite rules (see *Chapter 5*, *Customizing the Apache Server*, for details on how we edit `.htaccess` files).

Option 1: Rewrite example.com/foo to example.com/foo/

The following code snippet helps us to rewrite `example.com/foo` to `example.com/foo/`:

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_URI} !(\.[a-zA-Z0-9]{1,5}|/|#(.*))$
RewriteRule ^(.*)$ $1/ [R=301,L]
```

Option 2: Rewrite example.com/foo/ to example.com/foo

The following code snippet helps us to rewrite `example.com/foo/` to `example.com/foo`:

```
RewriteRule ^(.*)/$ $1 [R=301,L]
```

If you have existing rewrite rules, perform the following steps to make sure you set up your rewrite rules correctly. Not doing so can cause incorrect redirects and 404 errors.

- Keep a backup: Back up the `.htaccess` file you are going to add redirects to, before you start adding them. This way you can quickly go back to the backup file if you are unable to access your site because of an error in the `.htaccess` file.
- Do not append or replace existing rewrite rules: Instead of appending or replacing existing rules from CMSes you are using, merge them within.

- Watch the order of rewrite rules: Make sure you add the slash first and then your existing rules, which might rewrite the end paths.
- Confirm the `RewriteBase` path: If your website is in a subfolder, ensure you have set the right `RewriteBase` path for your rewrite rules. If you have a working `RewriteBase` path, do not remove it.

Finally, consider implementing guidelines from Google's *SEO Starter Guide* at `http://googlewebmastercentral.blogspot.com/2008/11/googles-seo-starter-guide.html`.

Handling users without JavaScript

HTML5 Boilerplate provides a class called `no-js` that gets replaced with a class called `js`, when JavaScript is detected by Modernizr on the `html` tag. Using this class name, you can craft the style for how the website should look when JavaScript is disabled.

In our Sun and Sand Festival website, when JavaScript is not enabled, clicking on the **Day 2** link produces nothing.

You can view how the site works when JavaScript is disabled on various browsers, in the following ways:

- **Firefox**: Go to **Preferences**, click on **Content**, and then uncheck the **EnableJavaScript** checkbox.
- **Chrome**: Download the **Chrome Web Developer** extension and disable JavaScript from within the extension.
- **Safari**: Click the **Disable JavaScript** menu item on the **Develop** menu. You can see the **Develop** menu when you check the **Show Develop** toolbar on the **Advanced** tab in the Safari **Preferences** pane.
- **Internet Explorer**: Click the **Internet Options** in the **Settings** menu, and click **Custom Level** and check the **Disable in the Active scripting** menu.
- **Opera**: Click **Quick Preferences** and unselect the **Enable JavaScript** option.

Let us make sure that the tabs do not render when JavaScript is not available, and make sure the whole listing is displayed at the same time, as shown in the following screenshot:

We can do this by editing `main.css` to take advantage of the `no-js` class. First, we need to remove the tabbed navigation, as shown in the following code:

```
.no-js .t-tab__nav {
display: none;
}
```

Then, we need to make the two lists be positioned statically instead of being absolutely positioned one below the other, as shown in the following code:

```
.no-js .t-tab__body {
position: static;
}
```

We need to make sure the `hidden` class does not get applied in this specific instance for **Day 2**, so we can see all the artists at once, as shown in the following code:

```
.no-js .t-tab__body.hidden {
display: block !important;
visibility: visible;
}
```

Now, when you enable JavaScript again, you will notice that the tabbed navigation appears and everything functions as you expected.

Optimizing your images

Every resource you add to your page is an extra request to the server and extra network trip for the browser before it declares the page complete. Network requests are typically the slowest components of a page load. This is especially obvious on mobile devices when surfing websites on 3G or even lower connections. The smaller your files are, the faster they will reach the browser.

If you can avoid using a large image, you are better off not using them.

8-bit PNGs

If you are considering using a GIF format for your images, always use PNG. PNG formats for images are far less heavy and much smaller. Even further, 8-bit PNGs are significantly smaller in size.

If you are using PNGs, you should use PNG-8 with a full alpha channel that gives you compatibility all the way back to IE6. Ensure you verify the final output to ensure they are not too grainy or pixelated.

Tools for image optimization

There are build tools in HTML5 Boilerplate that will optimize images,which we will look into in the next chapter. There are also standalone tools that are worth looking at, when you want to compress a bunch of images in one go. If you wish to upload your images and optimize them, you can do so at `smushit.com/ysmush.it/`.

ImageAlpha

If you have 24-bit PNG images, you can convert them into 8-bit PNGs that have a full alpha channel with this tool that you can download from `pngmini.com`. It is only for Mac OS X.

GUIs and command-line tools that work on other Operating Systems are outlined at `pngquant.org`.

ImageOptim

If you would like to optimize images of various formats in one go, **ImageOptim** would be your best tool of choice. You can download this from `imageoptim.com`. This is also for Mac OS X only, and takes advantage of several tools to perform these optimizations.

If you would like to use something similar on other systems, you can download the specific tool you need for each image format. The following table lists tools for some of the popular image formats:

Format	Tool
Animated GIF	`Gifsiclewww.lcdf.org/gifsicle/`
JPEG	`Jpegtranjpegclub.org/`
PNG	`Pngcrushpmt.sourceforge.net/pngcrush/`
	`Imageworsenerentropymine.com/imageworsener/`
	`Optipngoptipng.sourceforge.net/`
	`PNGOUT advsys.net/ken/utils.htm`

If you would like to learn more about using these tools for optimization, read more on the slides by Stoyan Stefanov about image optimization for the Web at `www.slideshare.net/stoyan/image-optimization-for-the-web-at-phpworks-presentation`. There are even more clever optimizations that could be done for PNG and JPEG image formats, detailed over on *Smashing Magazine* at `www.smashingmagazine.com/2009/07/15/clever-png-optimization-techniques/` and `http://www.smashingmagazine.com/2009/07/01/clever-jpeg-optimization-techniques/` respectively.

Using image sprites

Network requests take a long time to make for each resource. To make these smaller, you could combine multiple image files into one single image file that needs to be requested once and be cached for a very long time so that the page loads significantly faster. This is especially useful if your page is going to be viewed on devices with very low bandwidth connections to the Internet.

This means, you combine multiple images in one large image and use the CSS background property on all selectors where these images would be used. Let us convert all our artist images into a big sprite and replace the image elements into background images.

The following is our final sprite:

Let us replace our image elements in `index.html`, like the following one:

```
<img width="100" height="100" class="t-media__aside t-image--artist"
src="img/artist-tinariwen.png">
```

With the following:

```
<i class="t-artist__image artist-tinariwen"></i>
```

We do this for each of the artists. Then, in our `style.css`, we add the following code snippet:

```
.t-artist__image {
background: url(../img/artists-image.png) top left no-repeat,
url(../img/bg-artist.png) no-repeat center center;
float: left;
display: block;
}
.artist-asa { background-position:  -0px -0px, 0 0; }
.artist-kidjo { background-position:  -0px -100px, 0 0; }
.artist-kuti { background-position:  -100px -0px, 0 0; }
.artist-sangre { background-position:  -100px -100px, 0 0; }
.artist-tinariwen { background-position:  -200px -0px, 0 0; }
.artist-toure { background-position:  -200px -100px, 0 0; }
```

Nothing has changed in the final page, except we have now reduced the number of network requests to 1 instead of 6 for these images. By optimizing the final sprite, we can make this request even faster.

Generating a sprite would seem like a lot of work, but there are many tools to help with this.

CSS sprites from within Adobe Photoshop

Using the instructions documented at `arnaumarch.com/en/sprites.html`, you can use a script file from Photoshop to select a folder of images and also generate the associated CSS file using images within these files positioned and corrected as a background image.

There are some things to note when using this tool, as explained in the following points:

- Make sure the folder only contains the images you want to add to a sprite
- The resulting CSS file is generated within the folder used to create the sprite
- The generated sprite is opened in Adobe Photoshop and you are required to crop it before you save it out as an image at the location of your choice

CSS sprites with Compass

Compass—the framework on top of Sass—can stitch your images together at compile time and have the images referenced in your Sass file, turned into a sprite in the resulting CSS file.

All you need to do is to make sure you set up a folder within your images folder, such that you have the right names for each of the images, as described in the following list (taken from Compass documentation):

- `images/my-icons/new.png`
- `images/my-icons/edit.png`
- `images/my-icons/save.png`
- `images/my-icons/delete.png`

The name `my-icons` can be any name you prefer. Then in the Sass file, use the following code:

```
@import "my-icons/*.png";
@include all-my-icons-sprites;
```

Use the same name you used instead of `my-icons` in the previous step. Presto! You are done! Compass generates a CSS file that has the following code:

```
.my-icons-sprite,
.my-icons-delete,
.my-icons-edit,
.my-icons-new,
.my-icons-save   { background: url('/images/my-icons-s34fe0604ab.png')
no-repeat; }

.my-icons-delete { background-position: 0 0; }
.my-icons-edit   { background-position: 0 -32px; }
.my-icons-new    { background-position: 0 -64px; }
.my-icons-save   { background-position: 0 -96px; }
```

Now, use the appropriate class name in your markup to add the appropriate image to your element.

SpriteMe

SpriteME, available at `spriteme.org/`, is a bookmarklet that analyses images used on a page and creates sprites out of them. If you have an existing site to convert to using sprites, this would be a great place to start from.

Augmenting Google Analytics

Google Analytics can track several kinds of data and here are some easy, obvious augments you can make to your Analytics data.

Adding more tracking settings

Google Analytics gives you a number of optional settings to track, which you need not use the `.push()` method on; instead you can directly append to the initial array. Instead of the following:

```
var _gaq = _gaq || [];
_gaq.push(['_setAccount', 'UA-XXXXX-X'']);
_gaq.push(['_trackPageview']);
```

You can do the following:

```
var _gaq = [['_setAccount', 'UA-XXXXX-X'],['_trackPageview']];
```

Anonymize IP addresses

In some countries, no personal data may be transferred outside jurisdictions that do not have similarly strict laws (that is, from Germany to outside the EU). Thus, a webmaster using the Google Analytics script may have to ensure that no personal (trackable) data is transferred to the U.S. You can do that with the `_gat.anonymizeIp` option. In use it looks like the following:

```
var _gaq = [['_setAccount', 'UA-XXXXX-X'], ['_gat._anonymizeIp'], ['_
trackPageview']];
```

Tracking jQuery AJAX requests in Google Analytics

Steve Schwartz writes about a simple script you can use in the `plugins.js` that will allow you to track jQuery AJAX requests at `www.alfajango.com/blog/track-jquery-ajax-requests-in-google-analytics`. The following code snippet shows that script:

```
/*
 * Log all jQuery AJAX requests to Google Analytics
 * See: http://www.alfajango.com/blog/track-jquery-ajax-requests
-in-google-analytics/
 */
if (typeof _gaq !== "undefined" && _gaq !== null) {
  $(document).ajaxSend(function(event, xhr, settings){
    _gaq.push(['_trackPageview', settings.url]);
  });
}
```

Tracking JavaScript errors in Google Analytics

If you want to track JavaScript errors on your page using Google Analytics, it is possible to do so with the following script, which you need to add after the Google Analytics variable `_gaq` has been defined in the `index.html` page:

```
(function(window){
var undefined,
link = function (href) {
var a = window.document.createElement('a');
```

```
    a.href = href;
    return a;
        };
    window.onerror = function (message, file, row) {
    var host = link(file).hostname;
        _gaq.push([
          '_trackEvent',
          (host == window.location.hostname || host == undefined || host
    == '' ? '' : 'external ') + 'error',
    message, file + ' LINE: ' + row, undefined, undefined, true
        ]);
      };
    }(window));
```

Summary

In this chapter, we looked at how we can provide a better experience for users of Internet Explorer. We also considered very briefly some tools that can help us write more efficient and robust stylesheets that are easier to maintain in the light of cutting edge developments in CSS. We looked at how to use polyfills and write pages that are faster to load and more secure in general. We looked in detail at how to render the Sun and Sand website when JavaScript is disabled and also stitched the artists' images together into a sprite and saved on several network requests.

In the next chapter, we will look at automating deployment of our site using the build script that HTML5 Boilerplate provides.

Where to buy this book

You can buy HTML5 Boilerplate Web Development from the Packt Publishing website: `http://www.packtpub.com/html5-boilerplate-web-development/book`.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/html5-boilerplate-web-development/book