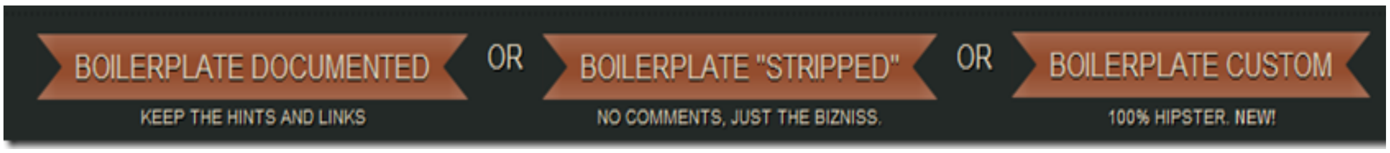
Getting Started Using HTML5 Boilerplate

Whether we like it or not, HTML5 is all the rage now days. With the recent news on “code name” Windows 8’s upcoming support for HTML5 and JavaScript that hype has intensified even more. I’m personally in favor of what HTML5 brings to the table although I do worry about browser compatibility issues that will naturally crop up. Compatibility issues are something that Web developers have been dealing with since the days of Netscape 4 (layers) and IE4 (divs) though so it’s really nothing new; it’s just intensified with all of the new functionality that the various HTML5 specs define. Fortunately, there are several options available that can help reduce cross-browser issues.

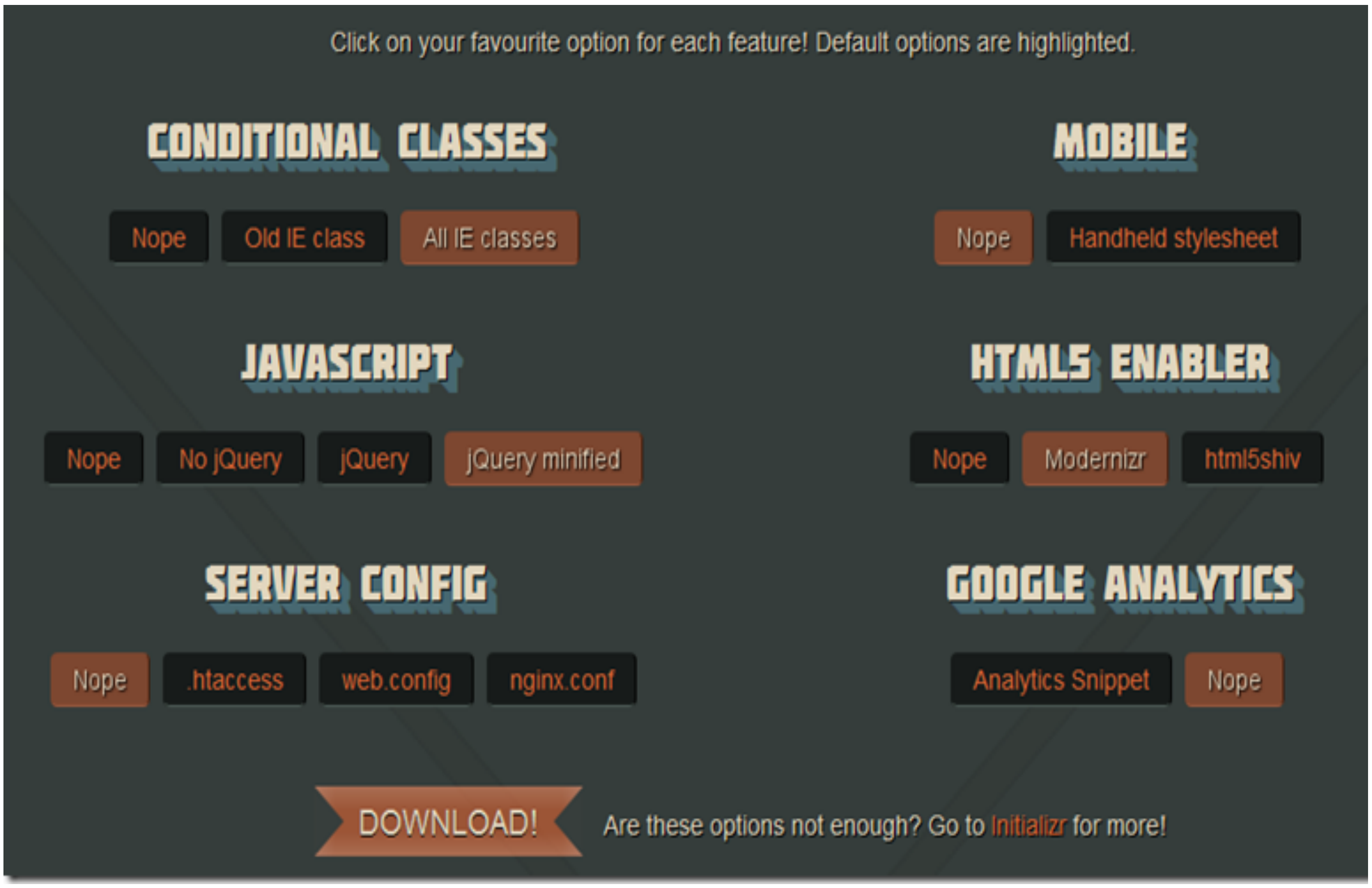
Getting started building HTML5 compatible sites can be challenging especially for people who haven’t been involved in Web development projects for awhile. Fortunately, there are a few sites available that greatly simplify the process of getting started building HTML5 websites and offer some excellent “boilerplate” code that can help both new and experienced Web developers. One of the best ones out there is called HTML5 Boilerplate (http://html5boilerplate.com) and others exist such as http://initializr.com (which is based on HTML5 Boilerplate). The creators of the site (including JavaScript guru Paul Irish from the jQuery and Chrome teams) define the functionality offered by HTML5 Boilerplate in the following way:

“HTML5 Boilerplate is the professional badass's base HTML/CSS/JS template for a fast, robust and future-proof site. After more than three years in iterative development, you get the best of the best practices baked in: cross-browser normalization, performance optimizations, even optional features like cross-domain Ajax and Flash. A starter apache .htaccess config file hooks you the eff up with caching rules and preps your site to serve HTML5 video, use @font-face, and get your gzip zipple on. Boilerplate is not a framework, nor does it prescribe any philosophy of development, it's just got some tricks to get your project off the ground quickly and right-footed.”

Features offered in HTML5 Boilerplate include cross-browser compatibility (they deal with IE6, IE7 and IE8 in a clever way), inclusion of caching and compression rules, utility classes such as .no-js and .clearfix, .png support in IE6, Modernizr support, Google analytics support, mobile browser optimizations, IE specific classes for maximum cross-browser control, JavaScript profiling and testing support, CDN hosted jQuery with a local fallback script, plus quite a bit more. When you visit the site you’re presented with 3 options including **Boilerplate Documented**, **Boilerplate Stripped** and **Boilerplate Custom**:



If you select **Boilerplate Documented** you’ll get a .zip file with the skeleton code needed to get started building an HTML5 site including documentation, built-in jQuery CDN support, CSS, caching, and more. **Boilerplate Stripped** contains the same HTML/CSS/JS but removes comments while **Boilerplate Custom** allows you to build a custom site skeleton and control what gets added or removed. An example of the custom options that can be selected are shown next:



Click on your favourite option for each feature! Default options are highlighted.

CONDITIONAL CLASSES

Nope Old IE class [✓ All IE classes]

MOBILE

[✓ Nope] Handheld stylesheet

JAVASCRIPT

Nope No jQuery jQuery [✓ jQuery minified]

HTML5 ENABLER

Nope [✓ Modernizr] html5shiv

SERVER CONFIG

[✓ Nope] .htaccess web.config nginx.conf

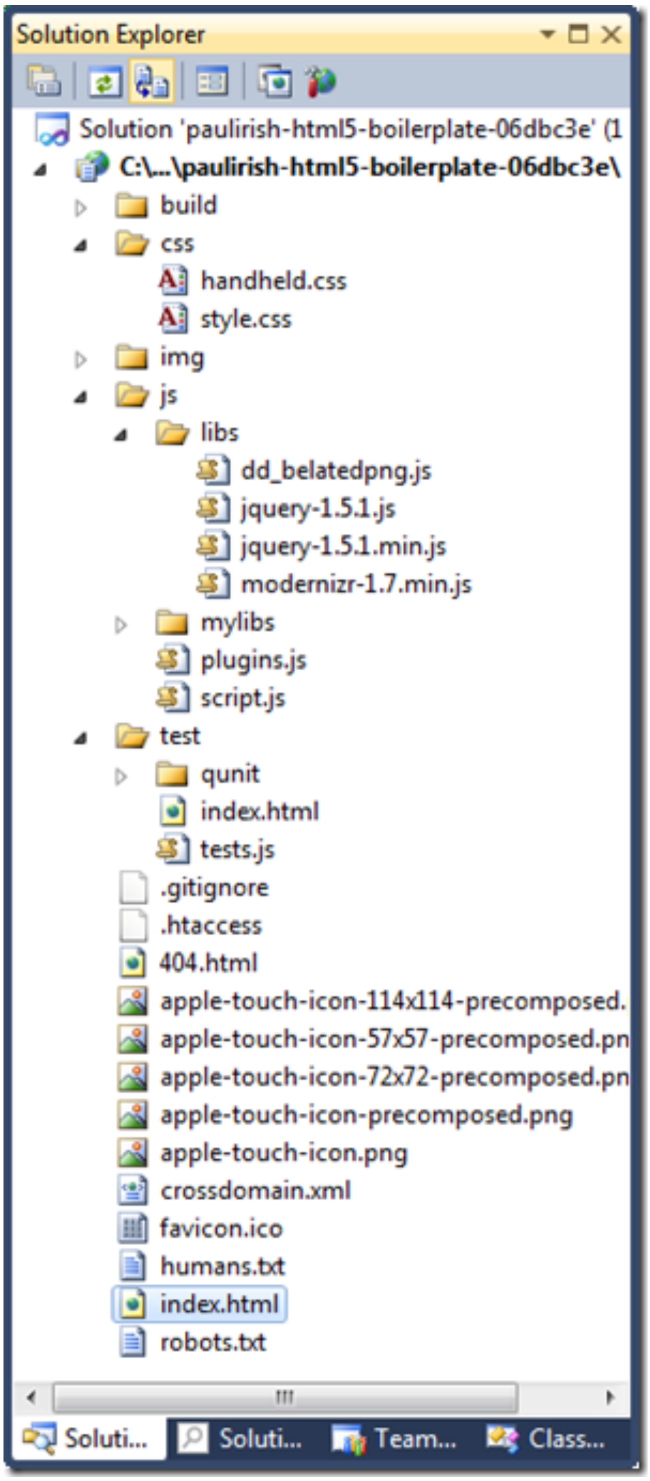
GOOGLE ANALYTICS

Analytics Snippet [✓ Nope]

DOWNLOAD! Are these options not enough? Go to Initializr for more!

What's in HTML5 Boilerplate?

If you select the documented or stripped options you'll get a .zip file that contains an HTML5 skeleton to help jumpstart your Web development. Here's what the site structure looks like. Looking through the folders you'll see that they contain CSS, JavaScript and HTML files as well as favicon.ico, crossdomain.xml for Flash and Silverlight, a 404.html page, robots.txt for search engines and QUnit testing functionality.



CSS Support

Two CSS files are included out of the box including the main CSS file used by the site named **style.css** as well as a starter script for hand held devices named **handheld.css**. The style.css file includes several different tips, tricks and best practices that can be used to handle rendering HTML across multiple browsers (including IE6). It adds in some clever features such as hiding content from a page and ensuring the content doesn't take up any space while still making it available to screen readers. Here's an example of a CSS class named **visuallyhidden** that performs the screen reader trick:

```
/* Hide only visually, but have it available for screenreaders: by Jon Neal.
  www.webaim.org/techniques/css/invisiblecontent/  &  j.mp/visuallyhidden */
.visuallyhidden { border: 0; clip: rect(0 0 0 0); height: 1px; margin: -1px; overflow: hidden; padding: 0; po
/* Extends the .visuallyhidden class to allow the element to be focusable when navigated to via the keyboard:
.visuallyhidden.focusable:active,
.visuallyhidden.focusable:focus { clip: auto; height: auto; margin: 0; overflow: visible; position: static; w
```

The style.css file also includes the popular clearfix solution that can be used to prevent margins from collapsing on child elements when the children are floated. If you've ever written HTML and CSS where you floated children left or right and then had to add a final child with "clear:both" in the style to make things look right then this fix greatly simplifies the process since you can simply apply the clearfix class:

```
/* The Magnificent Clearfix: Updated to prevent margin-collapsing on child elements.
   j.mp/bestclearfix */
.clearfix:before, .clearfix:after { content: "\0020"; display: block; height: 0; overflow: hidden; }
.clearfix:after { clear: both; }
/* Fix clearfix: blueprintcss.lighthouseapp.com/projects/15318/tickets/5-extra-margin-padding-bottom-of-page
.clearfix { zoom: 1; }
```

Print styles are also included to simplify the process of removing backgrounds, handling links, printing table rows, images, plus more:

```
/**
 * Print styles.
 *
 * Inlined to avoid required HTTP connection: www.phpied.com/delay-loading-your-print-css/
 */
@media print {
  * { background: transparent !important; color: black !important; text-shadow: none !important; filter:none
  -ms-filter: none !important; } /* Black prints faster: sanbeiji.com/archives/953 */
  a, a:visited { color: #444 !important; text-decoration: underline; }
  a[href]:after { content: " (" attr(href) ")"; }
  abbr[title]:after { content: " (" attr(title) ")"; }
  .ir a:after, a[href^="javascript:"]:after, a[href^="#"]:after { content: ""; }  /* Don't show links for ima
  pre, blockquote { border: 1px solid #999; page-break-inside: avoid; }
  thead { display: table-header-group; } /* css-discuss.incutio.com/wiki/Printing_Tables */
  tr, img { page-break-inside: avoid; }
  @page { margin: 0.5cm; }
  p, h2, h3 { orphans: 3; widows: 3; }
  h2, h3{ page-break-after: avoid; }
}
```

Moving into the **js** folder in the project you'll find jQuery, IE6 png support and Modernizer scripts included in the js/libs folder. Custom scripts can go in the scripts.js file and plugin scripts can go in plugins.js. Any custom libraries used in the site can go in the mylibs folder. Keep in mind that the folders are only a recommended folder structure and as the HTML5 Boilerplate site likes to say, all of the code is "delete-key friendly" so feel free to reorganize things to fit your needs.

The index.html File

Opening up the **index.html** file provided by HTML5 Boilerplate you'll find some interesting code right at the top:

```
<!doctype html>
<!-- paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/ -->
<!--[if lt IE 7 ]> <html class="no-js ie6" lang="en"> <![endif]-->
<!--[if IE 7 ]>    <html class="no-js ie7" lang="en"> <![endif]-->
<!--[if IE 8 ]>    <html class="no-js ie8" lang="en"> <![endif]-->
<!--[if (gte IE 9)|!(IE)]><!--> <html class="no-js" lang="en"> <!--<![endif]—>
```

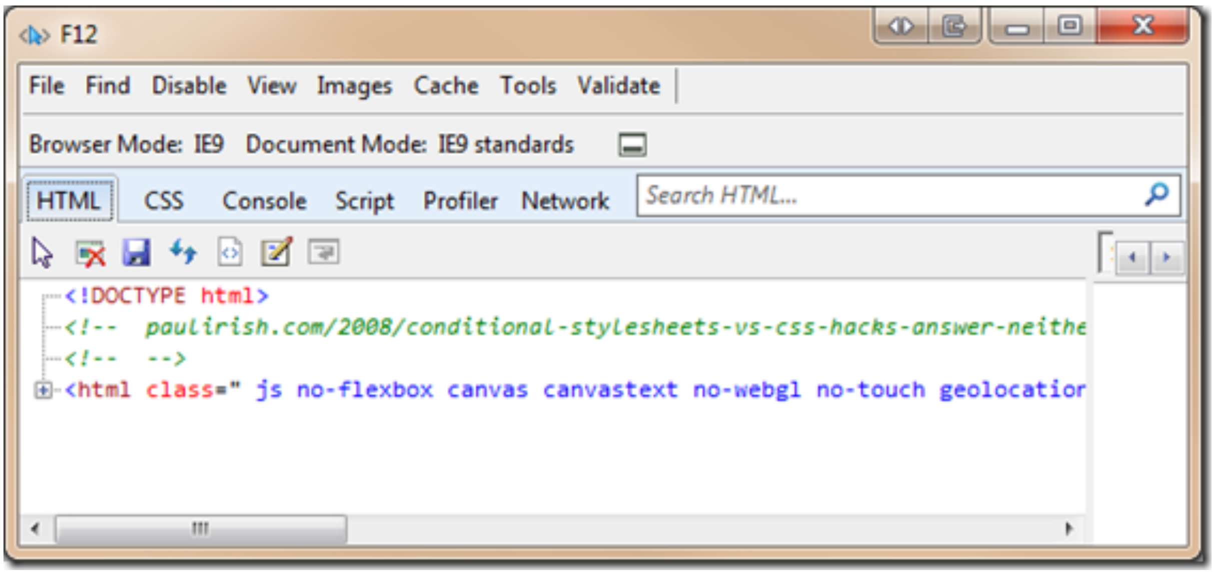
First, you'll note that the standard HTML5 doctype declaration is defined. I can't tell you how happy it makes me to have this simple definition since the different XHTML doctype options were hard to remember. Immediately under the doctype you'll notice the inclusion of several conditional statements to check for IE6, IE7, IE8 and IE9 along with other modern browsers. By default this code adds a **no-js** CSS class name and the appropriate IE class on the root <html> element if Internet Explorer is hitting the site. If another browser such as Chrome or Firefox is hitting the page then only the no-js class is added to the <html> element. The no-js class is used to handle browsers that don't support JavaScript. For example, if you wanted to hide all <h1> elements in the page when a browser doesn't support JavaScript then you could add the following definition into style.css:

```
.no-js h1 { display: none; } /* Hide h1 tags if Javascript is disabled */
```

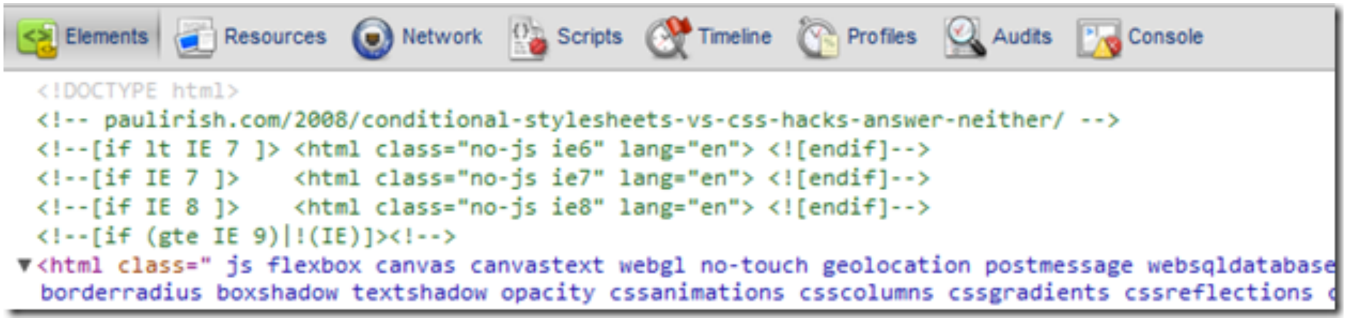
At this point you may wonder how the **no-js** class gets removed from the <html> element when a browser does support JavaScript. After all, if the browser supports JavaScript it wouldn't make much sense to have a no-js class defined. To handle this, the HTML5 Boilerplate code adds a reference to the Modernizr script within the head section of index.html.

```
  <script src="js/libs/modernizr-1.7.min.js"></script>
```

By default, Modernizr will locate the <html> element and change **no-js** to **js** if JavaScript is supported by the browser. An example of how Modernizr modifies the <html> element is shown using the IE9 developer tools (hit F12 when viewing a site in IE9 to get to the developer tools) and the Chrome developer tools (Ctrl + Shift + I) respectively:


F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE9 Document Mode: IE9 standards
HTML CSS Console Script Profiler Network
Search HTML...

```
<!DOCTYPE html>
<!--  paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neithe
<!--  -->
<html class=" js no-flexbox canvas canvastext no-webgl no-touch geolocatior
```


Elements Resources Network Scripts Timeline Profiles Audits Console

```
<!DOCTYPE html>
<!-- paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/ -->
<!--[if lt IE 7 ]> <html class="no-js ie6" lang="en"> <![endif]-->
<!--[if IE 7 ]>    <html class="no-js ie7" lang="en"> <![endif]-->
<!--[if IE 8 ]>    <html class="no-js ie8" lang="en"> <![endif]-->
<!--[if (gte IE 9)|!(IE)]><!-->
▼<html class=" js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase
 borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflections c
```

HTML5 Boilerplate's index.html page includes several other interesting features including the following meta tags:

```
<meta charset="utf-8">

<!-- Always force latest IE rendering engine (even in intranet) & Chrome Frame
    Remove this if you use the .htaccess -->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
```

The first meta tag sets the character set to utf-8 and is a short-cut version of what was used previously:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
```

At first glance setting the character set may seem trivial and unimportant but it can actually stop some script attacks that rely on utf-7 so it's recommended that you define it in your pages. The second meta tag is used to force Internet Explorer to use its most modern rendering engine. This comes into play with IE8 and IE9 since they have compatibility modes that a user can select in the address bar and ensures that Internet Explorer compatibility mode rendering isn't used (even if the user selected it). Although this second meta tag doesn't pass the latest W3C validation test it's good to have to prevent IE8/IE9 browsers from reverting back to IE7 mode due to the user enabling compatibility mode.

Moving down further in the <head> element within index.html you'll see that Google's CDN is used to load jQuery:

```
  <script src="//ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.js"></script>
```

This provides several benefits related to caching and location. A user visiting a site that references the Google CDN (Content Delivery Network) URL for jQuery will have the script cached in their browser. If they then visit your site that references the same CDN script file the cached version will be used resulting in faster page load times. Also, Google's CDN servers (note that Microsoft also provides a CDN for jQuery: http://www.asp.net/ajaxlibrary/cdn.ashx#jQuery_Releases_on_the_CDN_0) are located throughout the world so when the jQuery script gets loaded for the first time a server close to the user's location will normally be hit.

Having a reference to the CDN version of jQuery is nice but it's always good to have a fallback in case the CDN is inaccessible for whatever reason. HTML5 Boilerplate uses a trick I like to use as well to embed a local copy of jQuery if the CDN version can't be loaded. It does this by checking for the existence of the window.jQuery object. If the object isn't found then the CDN script wasn't loaded successfully and a local version of the script is loaded instead:

```
<script>window.jQuery || document.write("<script src='js/libs/jquery-1.5.1.min.js'>\x3C/script>")</script>
```

Another key item added into index.html is a reference to the Google analytics script for tracking page hits and other details. If you have a Google analytics account you can update your site's ID in the provided script:

```
  <script>
    var _gaq=[["_setAccount","UA-XXXXX-X"],["_trackPageview"]];
    (function(d,t){var g=d.createElement(t),s=d.getElementsByTagName(t)[0];g.async=1;
    g.src=("https:"==location.protocol?"//ssl":"//www")+".google-analytics.com/ga.js";
    s.parentNode.insertBefore(g,s)}(document,"script"));
  </script>
```

Testing Features

In addition to the HTML, CSS and JavaScript features added by HTML5 Boilerplate, support for QUnit is also included out of the box. QUnit is a JavaScript test library used to test the Query project code but can also be used to test JavaScript code that you write. Within the project created by HTML5 Boilerplate you'll find a folder named test that contains the qunit.js and associated CSS file as well as a sample test HTML page and test script. The test script provides a few basic examples of using QUnit to perform assertions and test logging functionality available in the plugins.js file included with the project:

```
module("example tests");
test("HTML5 Boilerplate is sweet",function(){
  expect(1);
  equals("boilerplate".replace("boilerplate","sweet"),"sweet","Yes. HTML5 Boilerplate is, in fact, sweet");
})

// these test things from plugins.js
test("Environment is good",function(){
  expect(3);
  ok( !!window.log, "log function present");

  var history = log.history && log.history.length || 0;
  log("logging from the test suite.")
  equals( log.history.length - history, 1, "log history keeps track" )

  ok( !!window.Modernizr, "Modernizr global is present")
})
```

While a complete discussion of QUnit is outside the scope of this post you can read more about it at http://docs.jquery.com/QUnit.

Conclusion

Whether you're an experienced Web developer or someone looking to get into HTML5 Web development, the code generated by the HTML5 Boilerplate site can help jumpstart your projects. It includes several tips and tricks to handle cross-browser issues that crop up out of the box, implements proven best practices, and ultimately simplifies the process of writing HTML5 compliant web pages. Although the code generated by HTML5 Boilerplate only provides skeleton HTML/CSS/JS code you can visit other sites such as http://initializr.com that build on the skeleton code while supplying functional pages with graphics and placeholders.

Published Sunday, June 05, 2011 4:56 PM by dwahlin
Filed under: ASP.NET, .NET, ASP.NET MVC, jQuery, HTML5, JavaScript, CSS

Comments

# re: Getting Started Using HTML5 Boilerplate

Monday, June 06, 2011 7:30 AM by Jon Howard

fantastic article. Thanks for posting!

# Dew Drop &ndash; June 6, 2011 | Alvin Ashcraft&#039;s Morning Dew

Monday, June 06, 2011 8:10 AM by Dew Drop – June 6, 2011 | Alvin Ashcraft's Morning Dew

Pingback from Dew Drop &ndash; June 6, 2011 | Alvin Ashcraft&#039;s Morning Dew

# re: Getting Started Using HTML5 Boilerplate

Monday, June 06, 2011 3:27 PM by RichardD

I thought it was better to specify the charset in the content-type header, rather than a meta-tag within the page?

Apart from the "missing 4k" bug in IE8, the parser need to restart parsing from the beginning when they encounter the character set declaration.

blogs.msdn.com/.../bugs-in-the-ie8-lookahead-downloader.aspx

# re: Getting Started Using HTML5 Boilerplate

Monday, June 06, 2011 11:51 PM by Micah Smith

So when will we see a NuGet package that has a _Layout.cshtml with boilerplate (and hopefully 960.gs) ready to go?.....

# Interesting .NET Links - June 7 , 2011 | Tech Blog

Monday, June 06, 2011 11:51 PM by Interesting .NET Links - June 7 , 2011 | Tech Blog

Pingback from Interesting .NET Links - June 7 , 2011 | Tech Blog

# re: Getting Started Using HTML5 Boilerplate

Tuesday, June 07, 2011 1:42 AM by Rashmi

What all browsers will support HTML 5 ? I am sure IE6 won't do it.

# re: Getting Started Using HTML5 Boilerplate

Tuesday, June 07, 2011 7:44 AM by Brian O'Connell

I'm wondering why the header on index.html in the template here has a div wrapping the html5 header element. Are we still stuck using redundant elements in html5 or is this the only way to fudge things for cross browser? If I'm stuck with div-itis in html5 then I'll stick with HTML4 for now I think where I can simply use a h element for the header, a ul for a menu and no unnecessary wrapper divs.

# re: Getting Started Using HTML5 Boilerplate

Tuesday, June 07, 2011 7:59 AM by Toni Podmanicki

5!

I'm big fan of HTML5 Boilerplate and it's nice to see some of ASP.NET ninjas write about it. Specially after announcement of HTML5 apps in Win 8.

# re: Getting Started Using HTML5 Boilerplate

Wednesday, June 08, 2011 2:23 AM by Nareen its good article to start exploring HTML 5

# re: Getting Started Using HTML5 Boilerplate

Wednesday, June 08, 2011 5:37 AM by dwahlin

Brian: You can work with divs just like you do now. Html5 simply adds new semantic tags such as header, article, section, footer, etc. They can just as easily be replaced with divs having IDs of course. I'm not sure why the boilerplate code wraps the div there...probably a personal preference of someone...hard to say.

Rashmi: All of the latest releases of browsers support HTML5 features. The challenge is that they don't support them all or don't support them consistently. Check out http://www.HTML5Test.com to get an idea of what is supported and what is not supported with different browsers.

# re: Getting Started Using HTML5 Boilerplate

Wednesday, June 08, 2011 6:04 AM by felix

what about jquery.mobile, is a competitor or will be work together?

# HTML5 Boilerplate | oito ...

Thursday, June 16, 2011 5:51 AM by HTML5 Boilerplate | oito ...

Pingback from HTML5 Boilerplate | oito ...

# HTML5 Boilerplate

Thursday, June 16, 2011 5:51 AM by Tiago Salgado

HTML5 Boilerplate é o modelo profissional HTML/CSS/JS para um site rápido, robusto e preparado para o