Learn HTML5 in 5 Minutes!

By Jennifer Marsman

There's no question, HTML5 is a hot topic for developers. If you need a crash course to quickly understand the fundamentals of HTML5's functionality, you're in the right place.

I'll cover the new semantic markup, canvas for drawing and animation, audio and video support, and how to use HTML5 with older browsers. Might be a bit more than five minutes, but I promise I'll keep it quick. Stick with me…it'll be worth it!

Semantic Markup and Page Layout

There's a great story about a university who, when building their campus, didn't create any walking paths. They just planted grass and waited.

A year later, the grass was all worn out where people walked most frequently. So that's where the university paved the actual sidewalks.

It makes perfect sense! The sidewalks were exactly where people actually walked.

The HTML5 new semantic elements were based on that exact same logic (see the W3C design guidance to "Pave the Cowpaths").

Semantic elements describe their meaning or purpose clearly to the browser and to the developer. Contrast that with (for example) the <div> tag. The <div> tag defines a division or a section in an HTML document, but it doesn't tell us anything about its content or convey any clear meaning.

```
<div>
```

Developers commonly use IDs and/or class names with these <div> tags. This conveys more meaning to the developers, but unfortunately, it doesn't help browsers derive the purpose of that markup.

```
<div id="header">
```

In HTML5, there are new semantically rich elements that can convey the purpose of the element to both developers and browsers.

```
<header>
```

The W3C mined billions of existing webpages to discover the IDs and class names that developers were already using. Once they threw out div1, div2, etc., they came up with a list of rich descriptive elements that were already being used, and made those the standards.

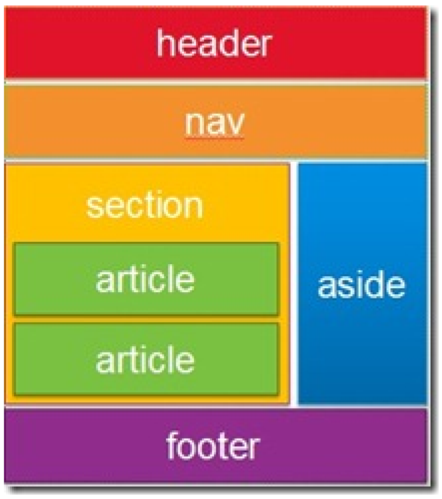
Here are a few of the new semantic elements in HTML5:

- article

- aside
- figcaption
- figure
- footer
- header
- hgroup
- mark
- nav
- section
- time

Because of the semantic richness, you can probably guess what most of these elements do.

But just in case, here's a visualisation:



Headers and **footers** are self-explanatory and **nav** creates a navigation or menu bar. You can use **sections** and **articles** to group your content. Finally, the **aside** element can be used for secondary content, for example, as a sidebar of related links.

Here is a simple example of some code that uses these elements.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Title</title>
    <link href="css/html5reset.css" rel="stylesheet" />
    <link href="css/style.css" rel="stylesheet" />
</head>
<body>
    <header>
        <hgroup>
            <h1>Header in h1</h1>
            <h2>Subheader in h2</h2>
        </hgroup>
    </header>
    <nav>
```

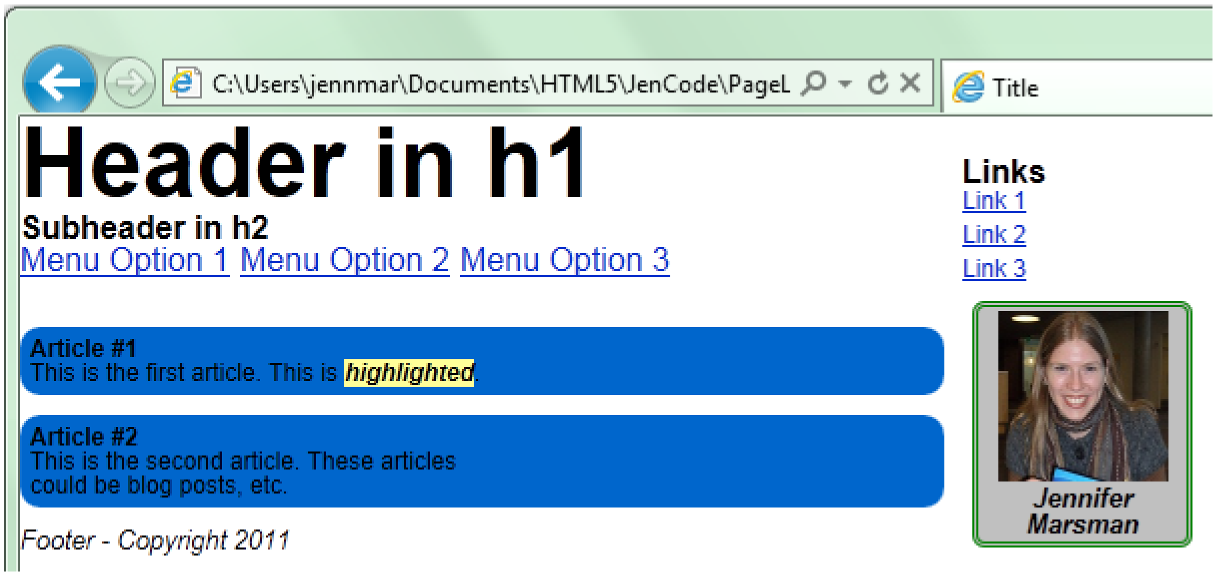
```
        <ul>
            <li><a href="#">Menu Option 1</a></li>
            <li><a href="#">Menu Option 2</a></li>
            <li><a href="#">Menu Option 3</a></li>
        </ul>
    </nav>
    <section>
        <article>
            <header>
                <h1>Article #1</h1>
            </header>
            <section>
                This is the first article.  This is <mark>highlighted</mark>.
            </section>
        </article>
        <article>
            <header>
                <h1>Article #2</h1>
            </header>
            <section>
                This is the second article.  These articles could be blog
posts, etc.
            </section>
        </article>
    </section>
    <aside>
        <section>
            <h1>Links</h1>
            <ul>
                <li><a href="#">Link 1</a></li>
                <li><a href="#">Link 2</a></li>
                <li><a href="#">Link 3</a></li>
            </ul>
        </section>
        <figure>
            <img width="85" height="85"
                src="http://www.windowsdevbootcamp.com/Images/JennMar.jpg"
                alt="Jennifer Marsman" />
            <figcaption>Jennifer Marsman</figcaption>
        </figure>
    </aside>
    <footer>Footer - Copyright 2011</footer>
</body>
</html>
```

I should call out a few other new elements in this code…

Did you notice the **hgroup** element, which grouped together my h1 and h2 headers?

The **mark** element allowed me to highlight or mark some important text. Finally, the **figure** and **figcaption** elements specify a figure in my content (like an image, diagram, photo, code snippet, etc.) and let me associate a caption with that figure, respectively.

Here's what that webpage would look like when combined with some CSS. (NOTE: I borrowed this CSS from my talented teammate Brandon Satrom's TechEd talk, but the less-than-beautiful end effect was all me.) 😊



C:\Users\jennmar\Documents\HTML5\JenCode\PageL

Title

Header in h1

Subheader in h2

Menu Option 1 Menu Option 2 Menu Option 3

Links

Link 1

Link 2

Link 3

Article #1

This is the first article. This is *highlighted*.

Article #2

This is the second article. These articles could be blog posts, etc.

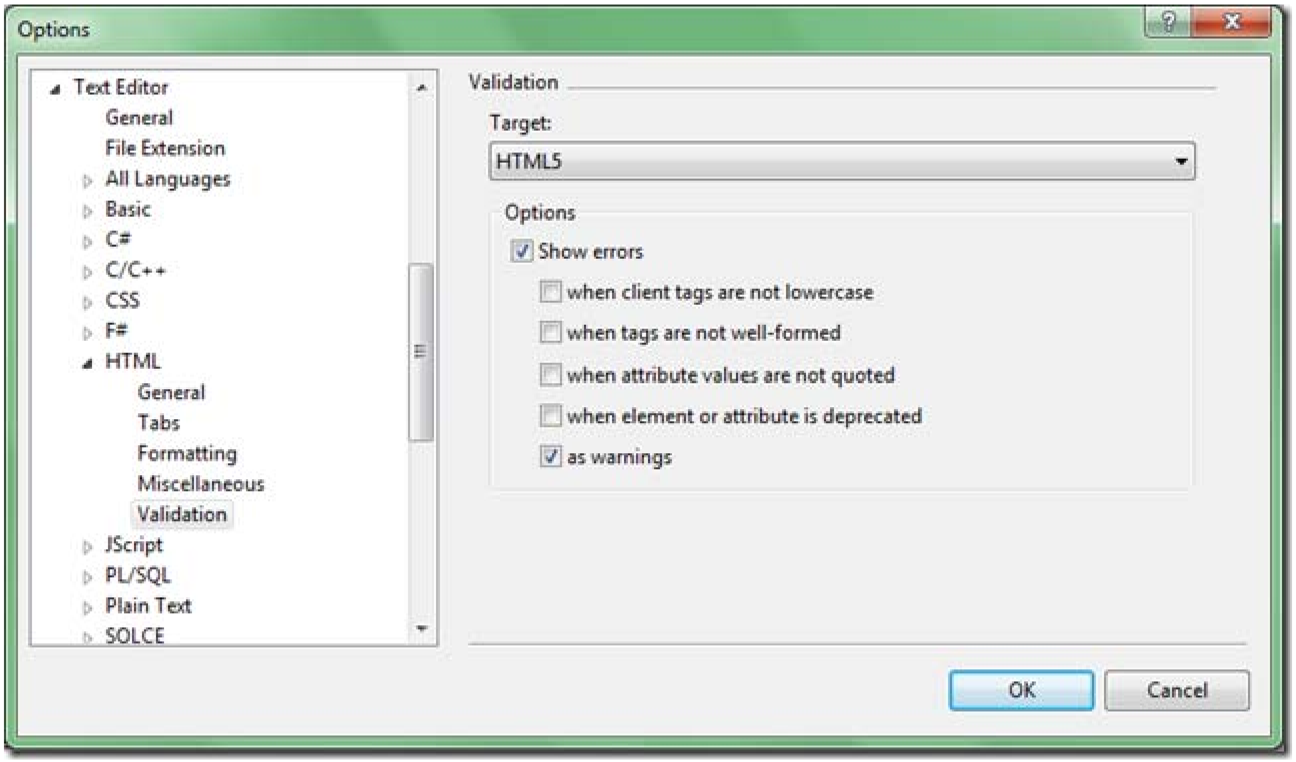
Footer - Copyright 2011

Jennifer Marsman

Now, imagine this in the hands of someone actually good at CSS (which I am not). The result is pretty powerful. The descriptiveness of the HTML makes it super easy to work with.

Finally, if you're trying to follow along in Visual Studio but you're seeing squiggly lines everywhere that VS doesn't understand the HTML5 elements, **make sure you have Visual Studio 2010 SP1 installed.**

Then, in the Visual Studio menu, go to Tools, Options. In the left-hand navigation pane, expand Text Editor, HTML, and then click Validation. From the Target dropdown menu, select HTML5. This will give you HTML5 IntelliSense support. Whew!



Validation

Target: HTML5

Options

[✓] Show errors

[ ] when client tags are not lowercase

[ ] when tags are not well-formed

[ ] when attribute values are not quoted

[ ] when element or attribute is deprecated

[✓] as warnings

To dive deeper into semantic elements, check out:

- How to Enable HTMl5 Standards Support
- Semantics section of the W3C HTML5 specification
- *Dive into HTML5* chapter on semantics entitled "What Does It All Mean?"

Drawing in HTML5 using the Canvas element

Another new element in HTML5 is the <canvas> tag. It's just what it sounds like——a blank surface for drawing. You need to use JavaScript to manipulate and draw on the canvas.

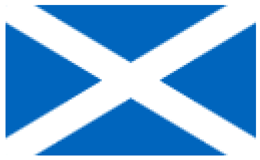
You may want to give your canvas element an **id** attribute so you can programmatically access it from your JavaScript code (or if you're using jQuery and it's the only canvas on the page, you could access it using $('canvas') without needing to name it).

You can also (optionally) specify a **height** and a **width** for the canvas. Between the <canvas> and </canvas>, you can specify some text to display in browsers that don't support the canvas element.

Here is a simple example of using the canvas to draw. (I'm attempting to draw the flag of Scotland. Please forgive any inaccuracies.)

```
<!DOCTYPE HTML>
<html>
<body>
    <canvas id="myCanvas">Your browser does not support the canvas
tag.</canvas>
    <script type="text/javascript">

        var canvas = document.getElementById('myCanvas');
        var ctx = canvas.getContext('2d');

        // Draw blue rectangle
        ctx.fillStyle = '#0065BD';
        ctx.fillRect(0, 0, 125, 75);

        // Draw white X
        ctx.beginPath();
        ctx.lineWidth = "15";
        ctx.strokeStyle = "white";
        ctx.moveTo(0, 0);
        ctx.lineTo(125, 75);
        ctx.moveTo(125, 0);
        ctx.lineTo(0, 75);
        ctx.stroke();

    </script>
</body>
</html>
```

Here's what the code produces:



Now let's walk through the code.

First, I create the actual canvas and give it an ID of "myCanvas". If this code were viewed in a browser that doesn't support the HTML5 canvas element, it would display "Your browser does not support the canvas tag" instead of drawing the flag.

Next, I have a script. Remember, the canvas tag is only a container for graphics; you must use JavaScript to actually draw and render graphics on it. First, I grab a reference to the canvas using the "myCanvas" ID, and then get the canvas's context which provides methods/properties for drawing and manipulating graphics on the canvas. I specified "2d" to use a 2-dimensional context to draw on the page.

Then, I draw the blue rectangle. I use **fillStyle** to specify the blue color. I use **fillRect** to draw the rectangle, specifying the size and position. Calling fillRect(0, 0, 125, 75) means: starting at position (0, 0)——the upper left-hand corner——draw a rectangle with width=125 pixels and height=75 pixels.

Finally, I draw the white X on the flag. I first call **beginPath** to start the process of painting a path. I specify a **lineWidth** of 15 pixels (using the guess-and-check method of trying different values until it looked correct) and a **strokeStyle** of "white" to make the path's color white. Then I trace out the path using **moveTo** and **lineTo**. These methods position a "cursor" for you to draw; the difference is that moveTo moves the cursor without drawing a line and lineTo draws a line while moving. I start by moving to position (0, 0)——the upper left-hand corner——and then drawing a line to (125, 75)——the lower right-hand corner. Then I move to position (125, 0)——the upper right-hand corner——and draw a line to (0, 75)——the lower left-hand corner. Finally, the **stroke** method actually renders these strokes.

Quick Comparison of Canvas vs. SVG

Scalable Vector Graphics (SVG) is an earlier standard for drawing in a browser window. With the release of HTML5's canvas, many people are wondering how they compare.

In my eyes, the biggest difference is that canvas uses immediate mode rendering and SVG uses retained mode rendering. This means that canvas directly causes rendering of the graphics to the display. In my code above, once the flag is drawn, it is forgotten by the system and no state is retained. Making a change would require a complete redraw. In contrast, SVG retains a complete model of the objects to be rendered. To make a change, you could simply change (for example) the position of the rectangle, and the browser would determine how to re-render it. This is less work for the developer, but also more heavyweight to maintain a model.

The ability to style SVG via CSS in addition to JavaScript is worth calling out as another consideration. A canvas may be manipulated through JavaScript only.

Here is a high-level overview of other differences:

	Canvas	**SVG**
Abstraction	Pixel-based (dynamic bitmap)	Shape-based
Elements	Single HTML element	Multiple graphical elements which become part of the Document Object

		Model (DOM)
Driver	Modified through Script only	Modified through Script and CSS
Event Model	User Interaction is granular (x,y)	User Interaction is abstracted (rect, path)
Performance	Performance is better with smaller surface and/or larger number of objects	Performance is better with smaller number of objects and/or larger surface

For a more detailed comparison, I want to point you to some sessions (from which I pulled this fabulous table, with permission):

- Patrick Dengler's "Thoughts on When to use SVG and Canvas"
- Jatinder Mann's "Deep Dive Into HTML5 Canvas"
- John Bristowe's "An Introduction to HTML5 Canvas"

Audio and Video Support

One of the big features that is new in HTML5 is the ability to support playing audio and videos. Prior to HTML5, you needed a plug-in like Silverlight or Flash for this functionality. In HTML5, you can embed audio and video using the new <audio> and <video> tags.

From a coding perspective, the audio and video elements are very simple to use. (I'll give you a more in-depth look at their attributes below.) The audio and video elements are also supported in all major browsers (the latest versions of Internet Explorer, Firefox, Chrome, Opera, and Safari). However, the tricky part is that you need codecs to play audio and video, and different browsers support different codecs. (For a wonderful in-depth explanation of video containers and codecs, read http://diveintohtml5.org/video.html.)

Fortunately, this isn't a show-stopper. **The support for audio and video was implemented in a brilliant way,** where there is support to try several different file formats (the browser will try each and then drop down to the next one if it can't play it).

As a best practice, you should provide multiple sources of audio and video to accommodate different browsers. You can also fallback to Silverlight or Flash. Finally, any text between the opening and closing tags (such as <audio> and </audio>) will be displayed in browsers that do not support the audio or video element.

For example:

```
<audio controls="controls">
    <source src="laughter.mp3" type="audio/mp3" />
    <source src="laughter.ogg" type="audio/ogg" />
    Your browser does not support the audio element.
</audio>
```

With this code, the browser will first try to play the laughter.mp3 file. If it does not have the right codecs to play it, it will next try to play the laughter.ogg file. If the audio element is not recognized at all by the browser, it will display the text "Your browser does not support the audio element" where the audio control should be.

One caveat to audio and video: there is no built-in digital rights management (DRM) support; you have to implement this yourself as the developer. See this link from the W3C which explains their position. (If you have a need for DRM content, also check out the Silverlight DRM documentation, which might be an easier solution.)

Now let's dive into each of these new elements.

Audio

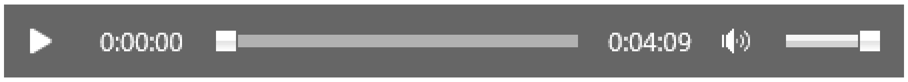
First, let's look at <audio> in more detail.

```
<audio controls="controls">
    <source src="laughter.mp3" type="audio/mp3" />
    <source src="laughter.ogg" type="audio/ogg" />
    Your browser does not support the audio element.
</audio>
```

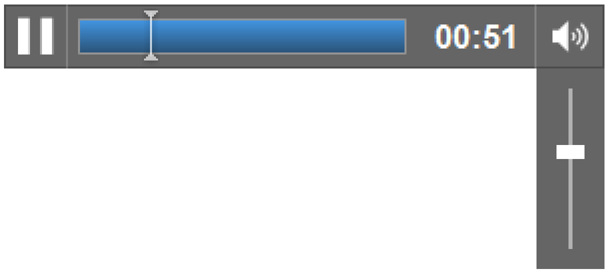
We already discussed the fallback effect of trying each source until it hopefully finds one that can be played.

Note that there is a **controls** attribute. This will display audio playback controls including a play/pause button, the time, a mute button, and volume controls. In most situations, it's good to display audio controls to the user; **I hate visiting a website with sound and being unable to stop it, mute it, or turn it down.** Don't you?

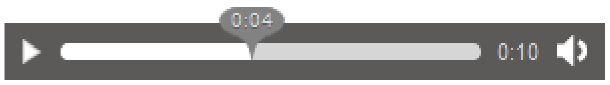
Here's what the audio controls look like in Internet Explorer:



The controls look different in different browsers. Here are what they look like in Chrome (with a song playing). The drop-down volume pops down when you hover over the sound icon on the far right.



Here are the controls in Firefox (with a song paused). Like Chrome, it also has a pop-up volume control (not shown) when you hover over the sound icon on the far right.



Other fun attributes on the audio element include:

Attribute	Possible Values	Description
autoplay	autoplay	Starts the audio playing as soon as it's ready
controls	controls	Displays audio playback controls on the page
loop	loop	Causes audio to repeat and play again every time it finishes

preload	auto, metadata, none	Determines whether to load the audio when the page is loaded. The value *auto* will load the audio, *metadata* will load only metadata associated with the audio file, and *none* will not preload audio. (This attribute will be ignored if autoplay is specified.)
src	(some URL)	Specifies the URL of the audio file to play

So this code sample would not only display audio playback controls, but also start the audio playing immediately and repeat it over and over in a loop.

```
<audio controls="controls" autoplay="autoplay" loop="loop">
    <source src="laughter.mp3" type="audio/mp3" />
    <source src="laughter.ogg" type="audio/ogg" />
    Your browser does not support the audio element.
</audio>
```

If you'd like to play around with the <audio> element yourself in your browser, there is a great "Tryit Editor" on http://w3schools.com that allows you to edit some sample code and see what happens. Or try the How to add an HTML5 audio player to your site article.

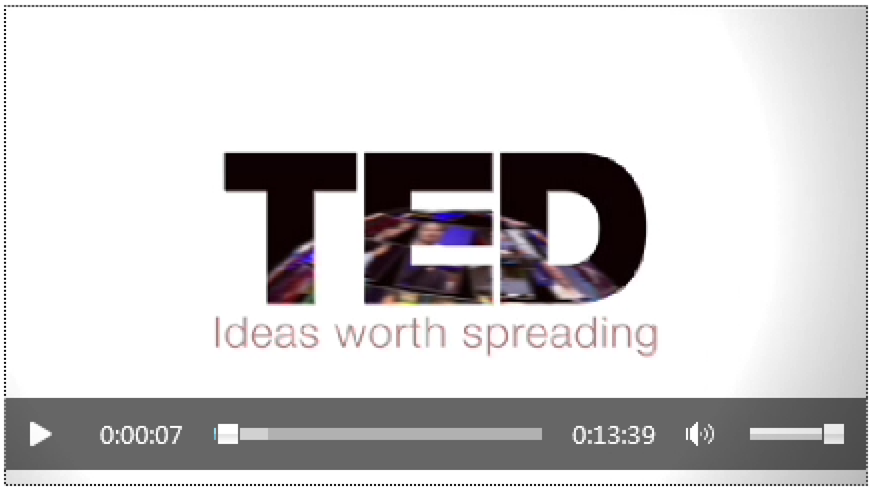
Video

Now, let's examine the <video> element.

```
<video width="320" height="240" controls="controls">
    <source src="movie.ogg" type="video/ogg" />
    <source src="movie.mp4" type="video/mp4" />
    <source src="movie.webm" type="video/webm" />
    Your browser does not support the video tag.
</video>
```

As we discussed above, the video element has support for multiple sources, which it will try in order and then fall down to the next option.

Like audio, video has a **controls** attribute. Here is a screenshot of the video controls in Internet Explorer:



Other fun attributes of the video element include:

Attribute	Possible Values	Description
audio	muted	Sets the default state of the audio (currently, "muted" is the only option)
autoplay	autoplay	Starts the video playing as soon as it's ready
controls	controls	Displays video playback controls on the page
height	(value in pixels)	Sets the height of the video player
loop	loop	Causes video to repeat and play again every time it finishes
poster	(some URL)	Specifies the URL of an image to represent the video when no video data is available
preload	auto, metadata, none	Determines whether to load the video when the page is loaded. The value *auto* will load the video, *metadata* will load only metadata associated with the video file, and *none* will not preload video. (This attribute will be ignored if autoplay is specified.)
src	(some URL)	Specifies the URL of the video file to play
width	(value in pixels)	Sets the width of the video player

Again, to play around with the <video> element yourself, use the "Tryit Editor" from http://w3schools.com that allows you to edit some sample code and see what happens.

To learn more about video and audio, check out:

- 5 Things you need to know to start using <audio> and <video> today
- How to add an HTML5 audio player to your site
- W3C Schools HTML5 video

Develop with HTML5 while retaining support for older browsers

We've discussed a lot of cool new functionality in HTML5, including the new semantic elements, the canvas tag for drawing, and the audio and video support.

You may think that this stuff is really cool, but you can't possibly adopt HTML5 when many of your users don't have HTML5-compatible browsers yet. Not to mention that the browsers that DO support HTML5 support different pieces of it; not all of the new HTML5 functionality is supported in all browsers and various features may be implemented differently.

But there is a way to use the new features without breaking your site for users with older browsers. You can use polyfills.

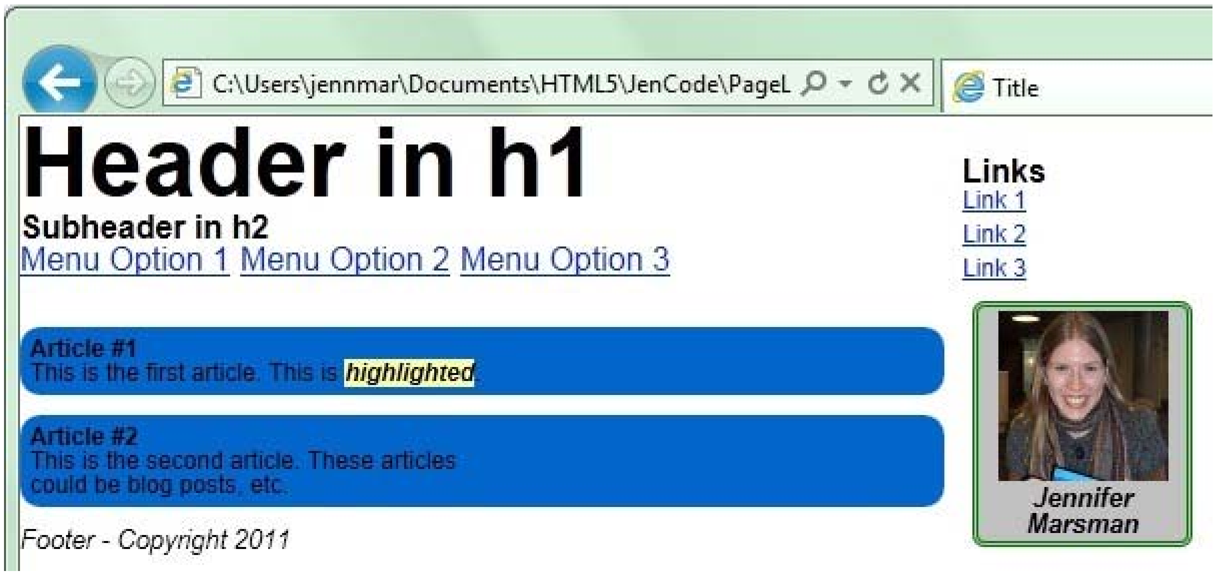
According to Paul Irish, a polyfill is "a shim that mimics a future API, providing fallback functionality to older browsers." A polyfill fills in the gaps in older browsers that don't support the HTML5 functionality in your site. Learning to use polyfills will let you use HTML5 today without leaving behind users of older browsers.

One way to get polyfill support is the JavaScript library Modernizr (but there are many polyfills available). Modernizr adds feature detection capability so you can check specifically for whether a browser supports (for example) the canvas element and provide a backup option if it doesn't.

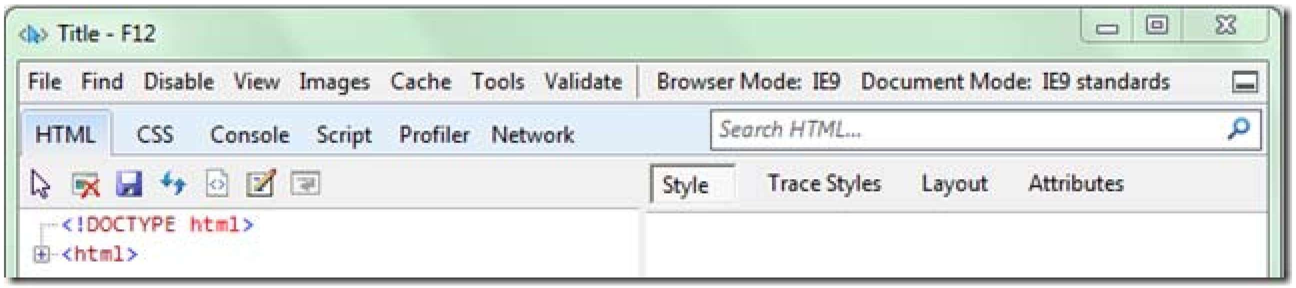
Let's walk through an example. Remember the code sample that I used when introducing semantic elements and page layout? Here it is again:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Title</title>
    <link href="css/html5reset.css" rel="stylesheet" />
    <link href="css/style.css" rel="stylesheet" />
</head>
<body>
    <header>
        <hgroup>
            <h1>Header in h1</h1>
            <h2>Subheader in h2</h2>
        </hgroup>
    </header>
    <nav>
        <ul>
            <li><a href="#">Menu Option 1</a></li>
            <li><a href="#">Menu Option 2</a></li>
            <li><a href="#">Menu Option 3</a></li>
        </ul>
    </nav>
    <section>
        <article>
            <header>
                <h1>Article #1</h1>
            </header>
            <section>
                This is the first article.  This is <mark>highlighted</mark>.
            </section>
        </article>
        <article>
            <header>
                <h1>Article #2</h1>
            </header>
            <section>
                This is the second article.  These articles could be blog
posts, etc.
            </section>
        </article>
    </section>
    <aside>
        <section>
            <h1>Links</h1>
            <ul>
                <li><a href="#">Link 1</a></li>
                <li><a href="#">Link 2</a></li>
                <li><a href="#">Link 3</a></li>
            </ul>
        </section>
        <figure>
            <img width="85" height="85"
                src="http://www.windowsdevbootcamp.com/Images/JennMar.jpg"
                alt="Jennifer Marsman" />
            <figcaption>Jennifer Marsman</figcaption>
        </figure>
    </aside>
    <footer>Footer - Copyright 2011</footer>
</body>
</html>
```

This code contains a number of new HTML5 elements that aren't supported in older browsers. Remember, in Internet Explorer 9, it looked like this:

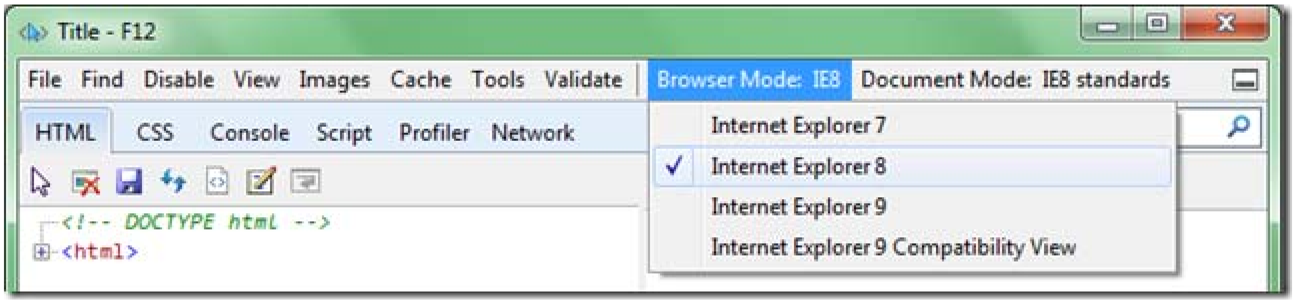

C:\Users\jennmar\Documents\HTML5\JenCode\PageL
Title
Header in h1
Subheader in h2
Menu Option 1 Menu Option 2 Menu Option 3
Links
Link 1
Link 2
Link 3
Article #1
This is the first article. This is highlighted.
Article #2
This is the second article. These articles could be blog posts, etc.
Jennifer Marsman
Footer - Copyright 2011

We can use the Internet Explorer developer tools to see what this would look like in older versions of IE. In Internet Explorer, press F12 to access the developer tools.


Title - F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE9 Document Mode: IE9 standards
HTML CSS Console Script Profiler Network
Search HTML...
Style Trace Styles Layout Attributes

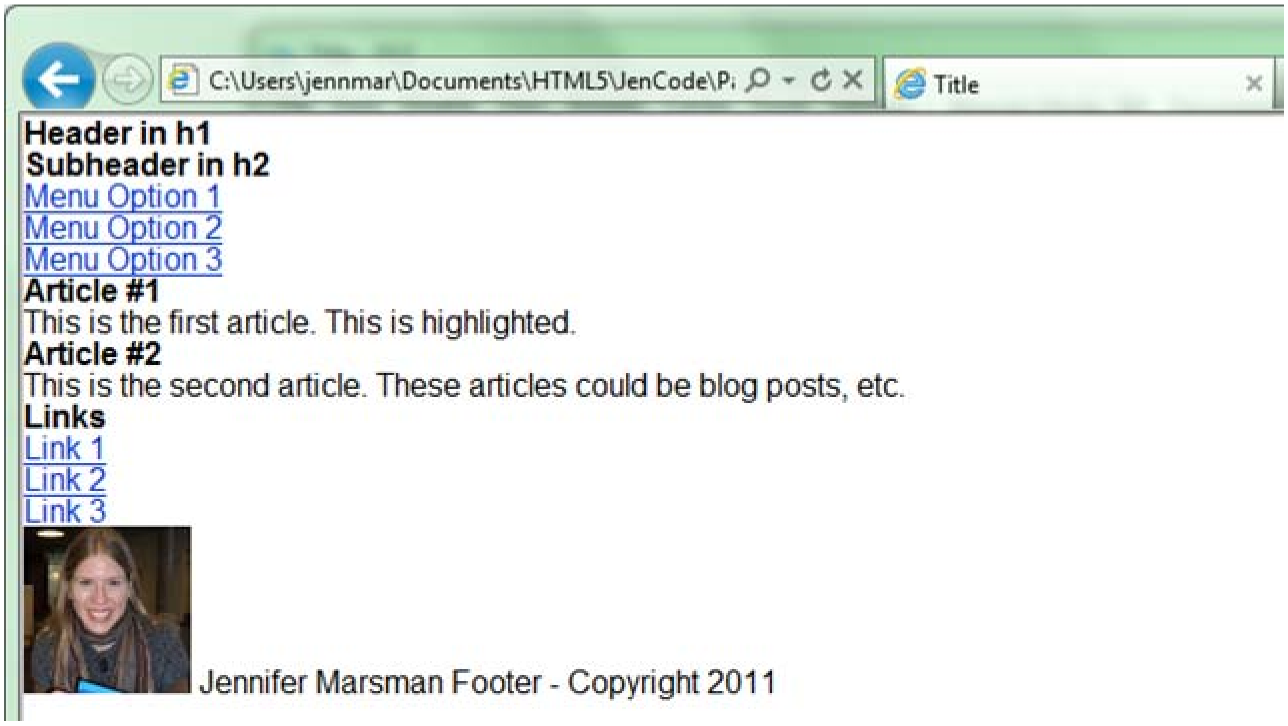
```
<!DOCTYPE html>
<html>
```

Note that the Browser Mode (in the grey menu bar across the top) is currently set to IE9. Click on the Browser Mode, and from the resulting dropdown menu, select "Internet Explorer 8" (which does not have HTML5 support).


Title - F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE8 Document Mode: IE8 standards
HTML CSS Console Script Profiler Network
Internet Explorer 7
✓ Internet Explorer 8
Internet Explorer 9
Internet Explorer 9 Compatibility View

```
<!-- DOCTYPE html -->
<html>
```

After I make this change and switch to a non-HTML5-compatible browser, this is what my webpage looks like:



C:\Users\jennmar\Documents\HTML5\JenCode\P

Title

Header in h1
Subheader in h2
Menu Option 1
Menu Option 2
Menu Option 3
Article #1
This is the first article. This is highlighted.
Article #2
This is the second article. These articles could be blog posts, etc.
Links
Link 1
Link 2
Link 3
Jennifer Marsman Footer - Copyright 2011

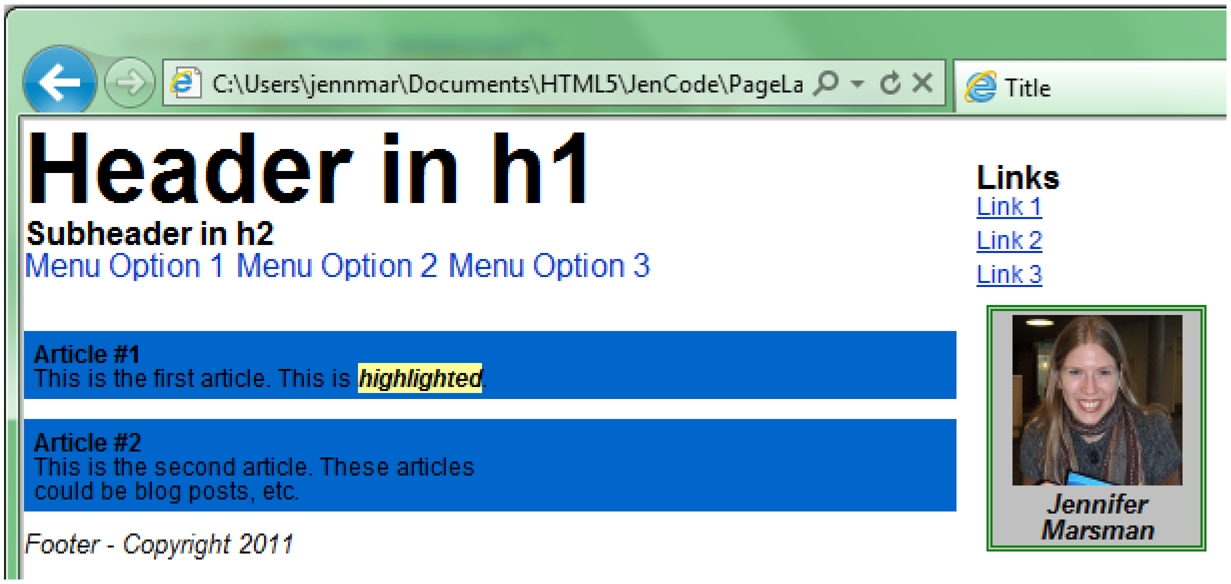
Although this looks like a monster problem to fix, it's not actually that bad. The reason that this doesn't work is that IE8 doesn't recognize the new HTML5 elements that I'm using, so it doesn't add them to the DOM, so you can't style them using CSS.

However, just adding a reference to Modernizr (without making any other code changes!) will brute-force these elements into the DOM. Download it from http://www.modernizr.com/download/ and add a reference in the <head> section like so:

```
<head>
    <meta charset="utf-8" />
    <title>Title</title>
    <link href="css/html5reset.css" rel="stylesheet" />
    <link href="css/style.css" rel="stylesheet" />
    <script src="script/jquery-1.6.2.min.js" type="text/javascript"></script>
    <script src="script/modernizr-2.0.6.js" type="text/javascript"></script>
</head>
```

I added two script references, one to jQuery and one to Modernizr. I don't actually need the jQuery reference at this point, but we will need it for the next script, so I'm adding it now.

Just this simple change now gets my site to this state in Internet Explorer 8:



C:\Users\jennmar\Documents\HTML5\JenCode\PageLa

Title

Header in h1

Subheader in h2

Menu Option 1 Menu Option 2 Menu Option 3

Article #1
This is the first article. This is *highlighted*.

Article #2
This is the second article. These articles could be blog posts, etc.

Footer - Copyright 2011

Links
Link 1
Link 2
Link 3

Jennifer Marsman

It's not perfect, but that is pretty close to the original version that we see in Internet Explorer 9. Modernizr added these new HTML5 elements that IE8 didn't understand into the DOM, and since they were in the DOM, we could style them using CSS.

But Modernizr does more than that! Notice that one of the differences between our IE8 and IE9 versions of the webpage is that the IE9 version has nice rounded corners on the two articles and the figure, and the IE8 version doesn't. We can also use Modernizr to fix this.

```
<script type="text/javascript">
    if (!Modernizr.borderradius) {
        $.getScript("script/jquery.corner.js", function() {
            $("article").corner();
            $("figure").corner();
        });
    }
</script>
```

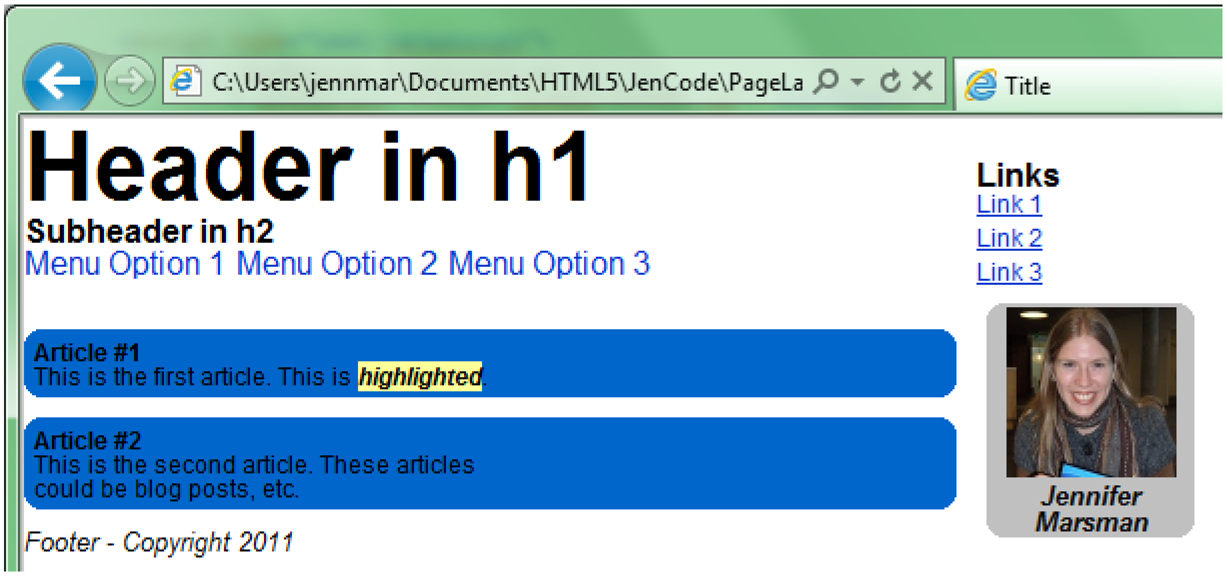
In this script, we're checking the Modernizr object to see if there is support for "borderradius" (a CSS3 feature). If not, I use a jQuery script called jquery.corner.js (which is available for download at http://jquery.malsup.com/corner/ and requires that extra reference to jQuery which I made earlier). Then I simply call the **corner** method from that script on my articles and figures to give them rounded corners.

OR, you can do this a slightly different way. Modernizr has an optional (not included) conditional resource loader called Modernizr.load(), based on Yepnope.js. This allows you to load only the polyfilling scripts that your users need, and it loads scripts asynchronously and in parallel which can sometimes offer a performance boost. To get Modernizr.load, you have to include it in a custom build of Modernizr which you have to create at http://www.modernizr.com/download/; it is not included in the Development version. With Modernizr.load, we can write a script like this:

```
<script type="text/javascript">
    Modernizr.load({
        test: Modernizr.borderradius,
        nope: 'script/jquery.corner.js',
        callback: function () {
            $('article').corner();
            $('figure').corner();
        }
    });
</script>
```

In short, this implements the same functionality as our previous script. Modernizr.load first tests the Boolean property "Modernizr.borderradius" to see if it is supported. Then, **nope** defines the resources to load if **test** is false. Since IE8 doesn't support the CSS3 property "borderradius", it will load the jquery.corner.js script. Finally, the **callback** specifies a function to run whenever the script is done loading, so we will call the "corner" method on my articles and figures as we did before. There is a brief tutorial on Modernizr.load at http://www.modernizr.com/docs/#load if you want to dive deeper.

Now, by using either of those scripts, our Internet Explorer 8 version (which doesn't support HTML5) looks like this:



Header in h1

Subheader in h2

Menu Option 1 Menu Option 2 Menu Option 3

Links

Link 1

Link 2

Link 3

Article #1

This is the first article. This is *highlighted*

Article #2

This is the second article. These articles could be blog posts, etc.

Jennifer Marsman

Footer - Copyright 2011

Therefore, using polyfills and tools like Modernizr allow you to utilize new HTML5 functionality and still provide a good experience on older browsers as well. For more information, check out http://www.diveintohtml5.org/detect.html which describes in detail how Modernizr detects HTML5 features.

Summary

In this introduction to HTML5, we covered semantic markup, canvas, audio and video, and using HTML5 while retaining support for older browsers. But also note that there's a lot we didn't cover: microdata, storage, CSS3, etc. Here are a few resources to continue learning about HTML5:

IE Test Drive – even if you don't use Internet Explorer, this is an awesome site. It contains a ton of demos: Speed Demos, HTML5 Demos, Graphics Demos, and Browser Demos. Try them in your favorite browser! This site also has links to other resources, too.

Beauty of the Web – showcases the best sites on the web that take advantage of HTML5 hardware acceleration and pinning features with Internet Explorer 9

BuildMyPinnedSite – all the code, ideas, and samples you need to use pinning and Windows integration

HTML5 Labs – This site is where Microsoft prototypes early and unstable specifications from web standards bodies such as W3C. You can play with prototypes like IndexedDB, WebSockets, FileAPI, and Media Capture API.

Videos

Brandon Satrom's "Application Development with HTML5" talk at TechEd 2011 – this is a fabulous hour-long talk that nails what you need to know to do HTML5 development

HTML5 talks from MIX 2011 – a wealth of HTML5 sessions

Tools

Many development tools support HTML5 already. Try these:

- Visual Studio 2010 SP1 – SP1 adds basic HTML5 and CSS3 IntelliSense and validation. For more information, see http://blogs.msdn.com/b/webdevtools/archive/2011/01/27/html5-amp-css3-in-visual-studio-2010-sp1.aspx.
- Web Standards Update for Microsoft Visual Studio 2010 SP1 – this is a Visual Studio extension that adds updated HTML5 and CSS3 IntelliSense and validation to Visual Studio 2010 SP1, based on the current W3C specification.
- WebMatrix – HTML5 is supported by default out of the box (adding a new HTML page uses the HTML5 default doctype and template code)
- ASP.NET MVC 3 tools update
    - The New Project dialog includes a checkbox-enabled HTML5 version of project templates.
    - These templates leverage Modernizr 1.7 to provide compatibility support for HTML5 and CSS3 in down-level browsers.
- Expression Web 4 SP1
    - HTML5 IntelliSense and support in the Code Editor and the Design View
    - Richer CSS3 style editing and IntelliSense
    - SuperPreview Page Interaction Mode and Online Service (remote browsers include Chrome, IE8, IE9, and Safari 4 and 5 on Windows and Mac)

Besides development tools, don't forget:

- Windows Phone "Mango" contains Internet Explorer 9, which supports HTML5 sites.
- Internet Explorer 10 Platform Previews has support for many new CSS3 and HTML5 features; the full list is at http://msdn.microsoft.com/en-us/ie/gg192966.aspx.

HTML5 is here. Go forth and develop amazing websites!

Jennifer Marsman is a Developer Evangelist in Microsoft's Developer and Platform Evangelism group, where she educates developers on Microsoft's new technologies. Prior to becoming a Developer Evangelist, Jennifer was a software developer in Microsoft's Natural Interactive Services division. In this role, she received two patents for her work in search and data mining algorithms. Jennifer has also held positions with Ford Motor Company, National Instruments, and Soar Technology.