

# REFACTORIZING HTML: MAKING THE GOODS BETTER

When you visit a website, the first thing you notice is the design - the interplay of colors and well-crafted look creating a whole that's greater than its parts. Most people don't spend much time considering the underpinnings of their site - they simply look at it in their browser of choice and make sure that everything displays correctly. When a search engine visits your site, the first thing it sees is the markup and if the markup isn't doing its part to paint a clear picture of your site then you're missing out on some easy wins for search engine optimization.

The act of converting HTML from an older, murkier implementation to a fresh, clean one is known as refactoring. The practice of refactoring has its roots in the software development industry where it's used to describe the "process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure."<sup>[1]</sup> When we refactor HTML we're applying the same principle, improving the internals of the page without affecting how the page displays.

## WHY REFACTOR?

- ✓ **Improve search engine optimization**
- ✓ **Decrease maintenance costs** by creating simpler code that's easier for developers to understand
- ✓ **Decrease payload size** - users enjoy a faster experience as there's less data to download
- ✓ **Improve your cross-browser display** by building to a standard HTML doctype
- ✓ **Prepare your web presence** for expansion into international markets with specific layout (right-to-left) or linguistic (multi-byte character) challenges

## HOW DO YOU DO IT?

Once you've decided that refactoring delivers benefits for your enterprise, you need a plan. A good place to start is to audit the existing site content and how it is implemented in the design. Identify standard pieces of content and assign semantic HTML <sup>[2]</sup> elements to them:

- » **main title:** h1
- » **subtitle:** h2
- » **section header:** h3
- » **paragraph:** p
- » **lists:** ul or ol

Some developers will advise that tables should not be used in semantic HTML. This is absolutely not true. Tables are still the best element available for displaying tabular data - however, they should not be used for defining page layout because they don't provide clear indicators of how content is related.

Once you've conducted an inventory and are ready to update your markup you need to select an HTML doctype. The doctype describes the rules that you will follow when building the page and tells browsers how to interpret the markup that you're providing. Choosing a modern doctype and validating against it provides a strong foundation for consistent cross-browser display.

## MORE THAN JUST MARKUP

The HTML markup is only part of the story of your web site. Markup provides a framework to hang a design on, but CSS is what is really used to implement the design. Some tips to getting the most from CSS development to accompany your HTML refactoring include:

- » **Develop a glossary** of common ids and classes with a description of what each of them is for - this helps ensure that developers are building pages in a consistent manner.
- » **Create a library** of standard snippets for common chunks of markup and CSS code - this will let you identify standard site components (tabbed content, quotes, etc.) and implement them in the same way across all pages.
- » **Consider a reset stylesheet** to level the playing field between browsers - examples include <http://developer.yahoo.com/yui/reset/> and <http://meyerweb.com/eric/tools/css/reset/>

If markup is the framework and CSS provides the design then JavaScript is what makes it move. JavaScript provides a dynamic element to your site to either enable additional features or improve the user experience. Some tips to get the most from your JavaScript functionality include:

- » **Adopt a JavaScript library** - libraries have figured out many of the hard bits and extrapolated out the browser-specific issues of JavaScript so you don't have to.
- » **Leverage plugins for standard JavaScript behavior** (i.e., tabbed interfaces or sortable tables) - many JavaScript libraries have a plugin system and using the same solution as others gives you the benefit of a solid implementation without needing to devote hours to build it yourself.

## HELP FROM THE BROWSER

Refactoring a website can be a significant undertaking. Fortunately there are a few tools available to help you on your way. The Firefox web browser enables several extensions to make your task easier:

- » **Firebug**<sup>[3]</sup> makes it easy to view the DOM and see exactly how your CSS styles are being applied.
- » **HTML Validator**<sup>[4]</sup> can run an automated validation check on every page you visit and provide a simple icon to identify if the page is valid or not.
- » **Web Developer**<sup>[5]</sup> tools package a few different tools together, including an easy way to run the W3C HTML Validator<sup>[6]</sup> tool on your markup.

Next time you're considering budget allocations for improving your website, give refactoring some serious consideration. It has many benefits and usually provides an untapped resource for major wins without major investment.

## FOOTNOTES/REFERENCES

1. <http://c2.com/cgi/wiki?WhatIsRefactoring>
2. [http://en.wikipedia.org/wiki/HTML#Semantic\\_HTML](http://en.wikipedia.org/wiki/HTML#Semantic_HTML)
3. <https://addons.mozilla.org/en-US/firefox/addon/1843>
4. <http://users.skynet.be/mgueury/mozilla/>
5. <https://addons.mozilla.org/en-US/firefox/addon/60>
6. <http://validator.w3.org/>

Find out how AtreNet can help you with your next HTML refactoring project.  
Contact: Tushar Atre PH: 831.464.0120 | CEL: 831.419.9050 | [tushar@atre.net](mailto:tushar@atre.net)