1. **Normalize.css preserves useful defaults rather than "unstyling" everything.** For example, elements like sup or sub "just work" after including normalize.css (and are actually made more robust) whereas they are visually indistinguishable from normal text after including reset.css. So, normalize.css does not impose a visual starting point (homogeny) upon you. This may not be to everyone's taste. The best thing to do is experiment with both and see which gels with your preferences.
2. **Normalize.css corrects some common bugs that are out of scope for reset.css.** It has a wider scope than reset.css, and also provides bug fixes for common problems like: display settings for HTML5 elements, the lack of font inheritance by form elements, correcting font-size rendering for pre, SVG overflow in IE9, and the button styling bug in iOS.
3. **Normalize.css doesn't clutter your dev tools.** A common irritation when using reset.css is the large inheritance chain that is displayed in browser CSS debugging tools. This is not such an issue with normalize.css because of the targeted stylings.
4. **Normalize.css is more modular.** The project is broken down into relatively independent sections, making it easy for you to potentially remove sections (like the form normalizations) if you know they will never be needed by your website.
5. **Normalize.css has better documentation.** The normalize.css code is documented inline as well as more comprehensively in the [GitHub Wiki](https://github.com/necolas/normalize.css/wiki). This means you can find out what each line of code is doing, why it was included, what the differences are between browsers, and more easily run your own tests. The project aims to help educate people on how browsers render elements by default, and make it easier for them to be involved in submitting improvements.

**CSS: reset or normalize?**

Building for the web can be like building on quicksand. Browsers have tended to do mostly the same thing, but have occasional, maddeningly unpredictable differences. For example, browsers all come with “user agent stylesheets” — a default set of CSS styles, so that a heading looks like a heading etc., even before you style the page[1](https://the-pastry-box-project.net/oli-studholme/2013-june-3#fn1). Of course, every browser engine uses a slightly different set of defaults.

One example of this was [default list styles](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Consistent_list_indentation), where Internet Explorer and Opera initially[2](https://the-pastry-box-project.net/oli-studholme/2013-june-3#fn2) indented lists with margin-left: 30pt; in their default browser stylesheets, while Firefox and KHTML went withpadding-left: 40px;. If you wanted to change the default indent, specifying ul {padding-left: 0;} would lead to very different results across browsers.

**CSS resets**[**3**](https://the-pastry-box-project.net/oli-studholme/2013-june-3#fn3)

To get a little stability, some developers reset all margins and padding using the universal selector:

\* {margin: 0; padding: 0;}

With this at the start of your stylesheet, when you specified a list indent you got what you expected. However, using \* meant the default margin and padding were nuked for *all* elements, which became painful as soon as you added a <form> element.

[Tantek Çelik](http://tantek.com/log/2004/09.html#d06t2354) and [Eric Meyer](http://meyerweb.com/eric/thoughts/2004/09/15/emreallyem-undoing-htmlcss/) started discussing a more targeted way to address user agent style differences in 2004, with the [YUI CSS Reset](http://nate.koechley.com/blog/2006/05/09/second-yui-release/)appearing in 2006, and the [Meyer Reset](http://meyerweb.com/eric/thoughts/2007/04/12/reset-styles/) in 2007.

The point of a reset is to zero out as much as possible … [and] to serve as a starting point for your own baseline styles — [Eric Meyer](http://meyerweb.com/eric/thoughts/2011/01/26/reset-v2-0/#comment-542624)

Here’s the first rule of [Eric’s current (v2.0) CSS Reset](http://meyerweb.com/eric/tools/css/reset/):

html, body, div, span, applet, object, iframe,

h1, h2, h3, h4, h5, h6, p, blockquote, pre,

a, abbr, acronym, address, big, cite, code,

del, dfn, em, img, ins, kbd, q, s, samp,

small, strike, strong, sub, sup, tt, var,

b, u, i, center,

dl, dt, dd, ol, ul, li,

fieldset, form, label, legend,

table, caption, tbody, tfoot, thead, tr, th, td,

article, aside, canvas, details, embed,

figure, figcaption, footer, header, hgroup,

menu, nav, output, ruby, section, summary,

time, mark, audio, video {

margin: 0;

padding: 0;

border: 0;

font-size: 100%;

font: inherit;

vertical-align: baseline;

}

…

This resets several properties on many (but not all) elements back to the equivalent of plain text. Because only appropriate elements are reset, this avoids some of the problems of \* {margin: 0; padding: 0;}. We can then define styles for these reset “unstyled” properties, safe in the knowledge we’re building on a stable, cross-browser base. This “unstyled” styling also acts as a reminder to consciously set appropriate styles for these elements.

**Problems with CSS resets**

CSS resets have been a lifesaver, but especially with the rise of frameworks they are now often used as-is. For example, Eric assumed people would build on the reset styles he proposed, overriding them as appropriate, and version 1 of the Meyer Reset included this rule for a time:

/\* remember to define focus styles! \*/

:focus {

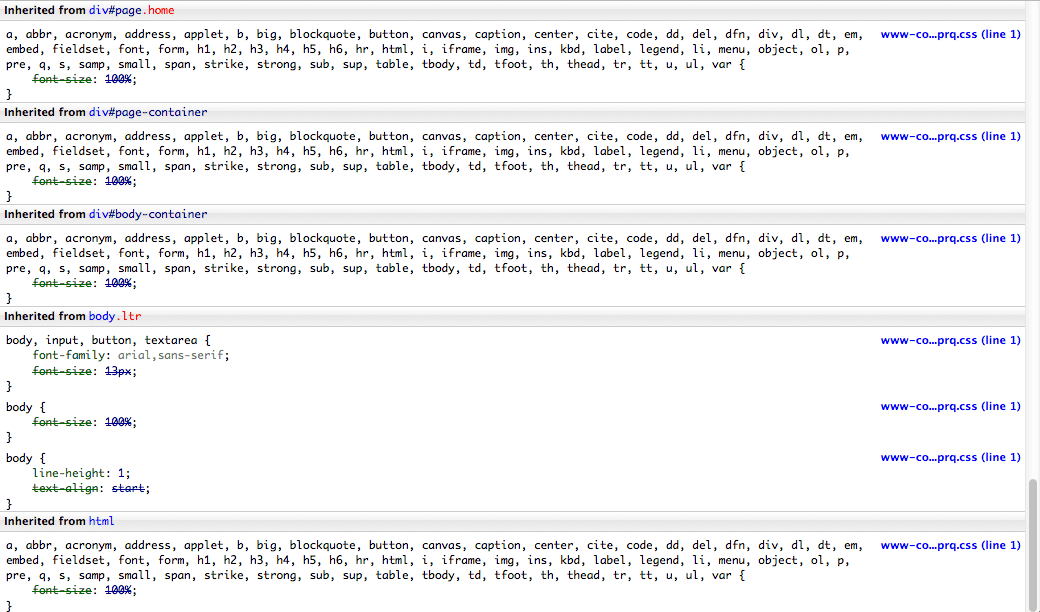
outline: 0;

}

Sadly, not everyone did define focus styles, and Eric has removed this from v2.

Using a reset can also start to feel a little perverse. Resetting browser default styles does force us to deliberate on how each element should appear, helping ensure we use elements for their semantics and not their default styling. But for elements like i and em that’s almost always the browser default style. Other default browser styles, like the text sizes for headings which used to be ridiculously large, have changed to become passable defaults. There’s also the problem of someone wanting to use a reset HTML element after handoff, still with only the “unstyled” reset styles specified.

For me the main issue with resets is inheritance, leading to spam in your browser dev tools. When you’re trying to track down a CSS problem on a deeply nested element in someone else’s code, this does not help:

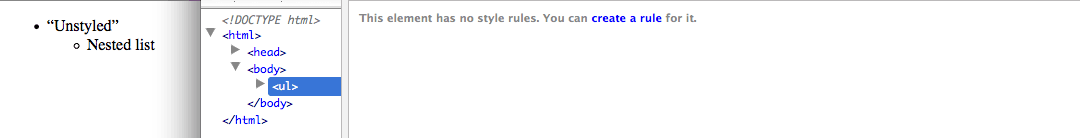
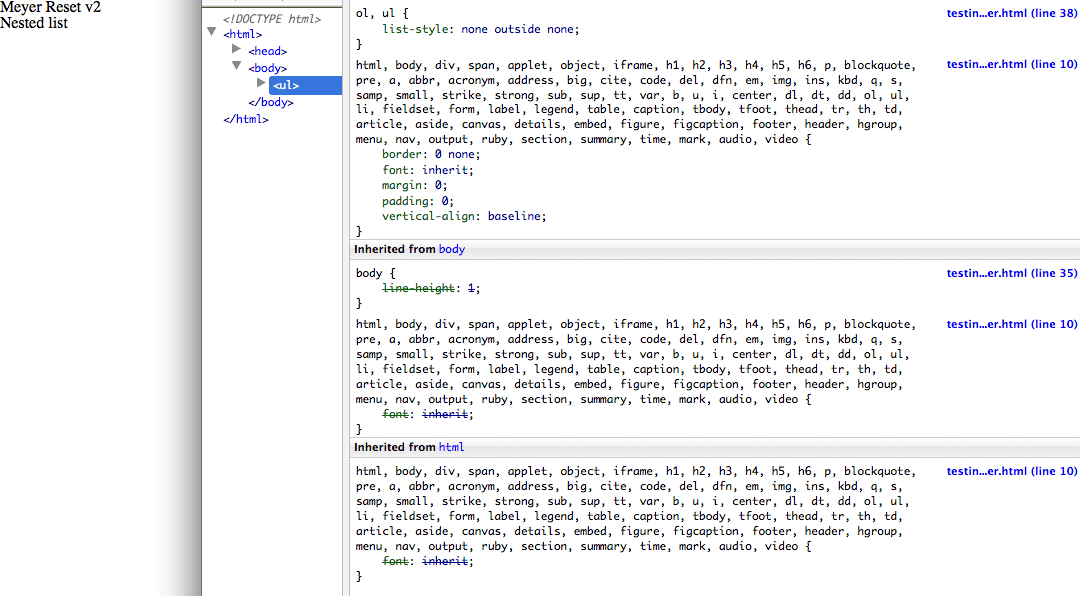
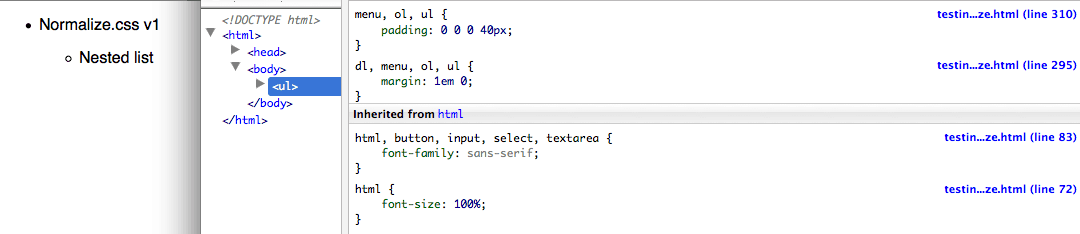
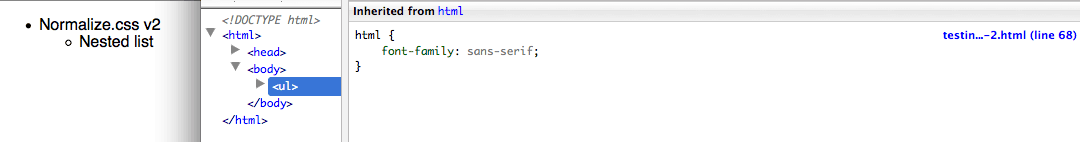
CSS reset rules repeating due to inheritance

**Normalize.css**

Nicolas Gallagher and Jonathan Neal have taken a different approach with [Normalize.css](http://necolas.github.io/normalize.css/), “a small CSS file that provides better cross-browser consistency in the default styling of HTML elements”. As with CSS resets it gives us a reliable cross-browser starting point — the main reason for using a reset in the first place — but the two approaches are philosophically different.

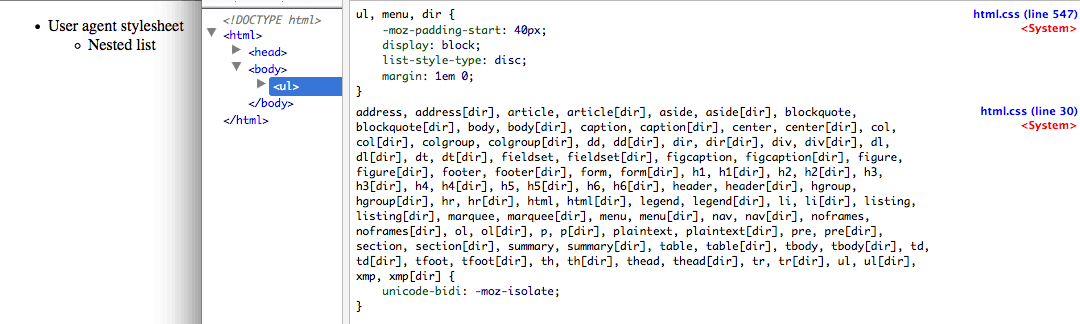
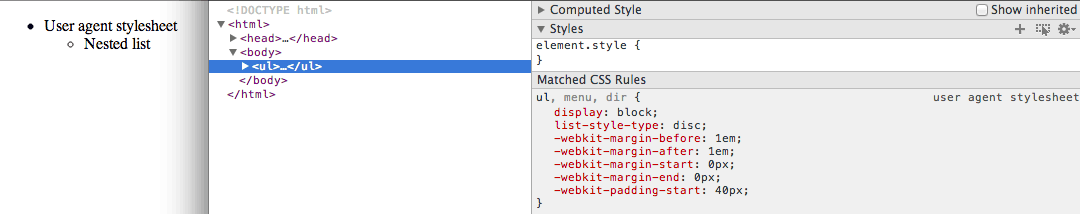
CSS resets override user agent styles to return many elements back to an “unstyled” state, a level foundation we can safely build upon. However, we then need to define styles for most elements before we can build with them. Normalize.css instead addresses only the inconsistencies between user agent default styles, choosing the most appropriate default where there are differences. We get a safe cross-browser foundation here too, but one that includes normalized user agent styles as basic building materials ready for use. It’s basically a kind of an idealized, cross-browser version of CSS 2.1’s [Default style sheet](http://www.w3.org/TR/CSS2/sample.html). For both ways we then need to add our own overriding styles to build the view, but because the browser defaults remain with Normalize.css, in general fewer styles need to be added.

Because the changes in Normalize.css are a lot more targeted, there isn’t an inheritance cascade of overwritten rules in your browser’s developer tools. Here’s a simple <ul>: “unstyled”, with the Meyer Reset, and with Normalize.css versions 1 and 2:

An “unstyled” unordered list elementApplying the Meyer ResetApplying Normalize.css v1Applying Normalize.css v2

You can clearly see the difference in philosophy, with the Meyer Reset example appearing as two lines of plain text with no margins, padding or bullets, while the Normalize.css examples are similar to the default styling. The difference in the styles applied to this <ul> are also easy to see.

However, these are not all the styles being applied to the <ul>. For comparison, here’s the same “unstyled” screenshot, but with the user agent styles visible, in Firefox 21 and Opera Next 15:

Mozilla user agent stylesOpera user agent styles

*This* is the CSS that we’re resetting or normalizing.

Normalize.css version 2 supports modern browsers plus IE 8, whereas version 1 also contains additional support for legacy browsers like IE 6 and 7. These older browsers need more normalization, and this can have minor disadvantages, for example the added vertical margins for the nested list in the Normalize.css v1 screenshot above. This split into two versions is useful if you no longer need to provide old browsers with Grade A support, and also if you want to learn about old browser quirks.

Normalize.css also helps correct some browser bugs, including “display settings for HTML5 elements, correcting font-size for preformatted text, SVG overflow in IE9, and many form-related bugs across browsers and operating systems”. For example, the following CSS fixes WebKit issues with HTML5’s new <input type="search"> element:

/\*\*

\* 1. Address `appearance` set to `searchfield` in Safari 5 and Chrome.

\* 2. Address `box-sizing` set to `border-box` in Safari 5 and Chrome

\* (include `-moz` to future-proof).

\*/

input[type="search"] {

-webkit-appearance: textfield; /\* 1 \*/

-moz-box-sizing: content-box;

-webkit-box-sizing: content-box; /\* 2 \*/

box-sizing: content-box;

}

Without it, WebKit’s default use of -webkit-appearance: searchfield;for type="search" prevents the styling of font, padding, border, and background properties on OS X and iOS, and gives buggy behavior for the border property on Windows.

An added bonus is Normalize.css is heavily commented and [well-documented](https://github.com/necolas/normalize.css/wiki/), helping you learn why each rule is there. This does make it noticeably longer than CSS resets, but when minified and Gzipped even the larger Normalize.css v1 is only 1KB.