xfive.co

ITCSS: Scalable and Maintainable CSS Architecture - Xfive

More articles from Lubos

9-12 minutes

How do I make my CSS scalable and maintainable? It's a concern for every front-end developer. ITCSS has an answer.

Last year when we started to plan our HEROized redesign and new Xfive.co website, I was looking for a CSS architecture which would allow for easy website development and further maintenance.

CSS Modules were quite young and exotic at that time and I've always considered the Atomic Design chemistry analogy to be a bit artificial. Then I came across Harry Roberts's ITCSS in the June 2015 issue of the net magazine and immediately fell in love with this simple, down to earth CSS approach.

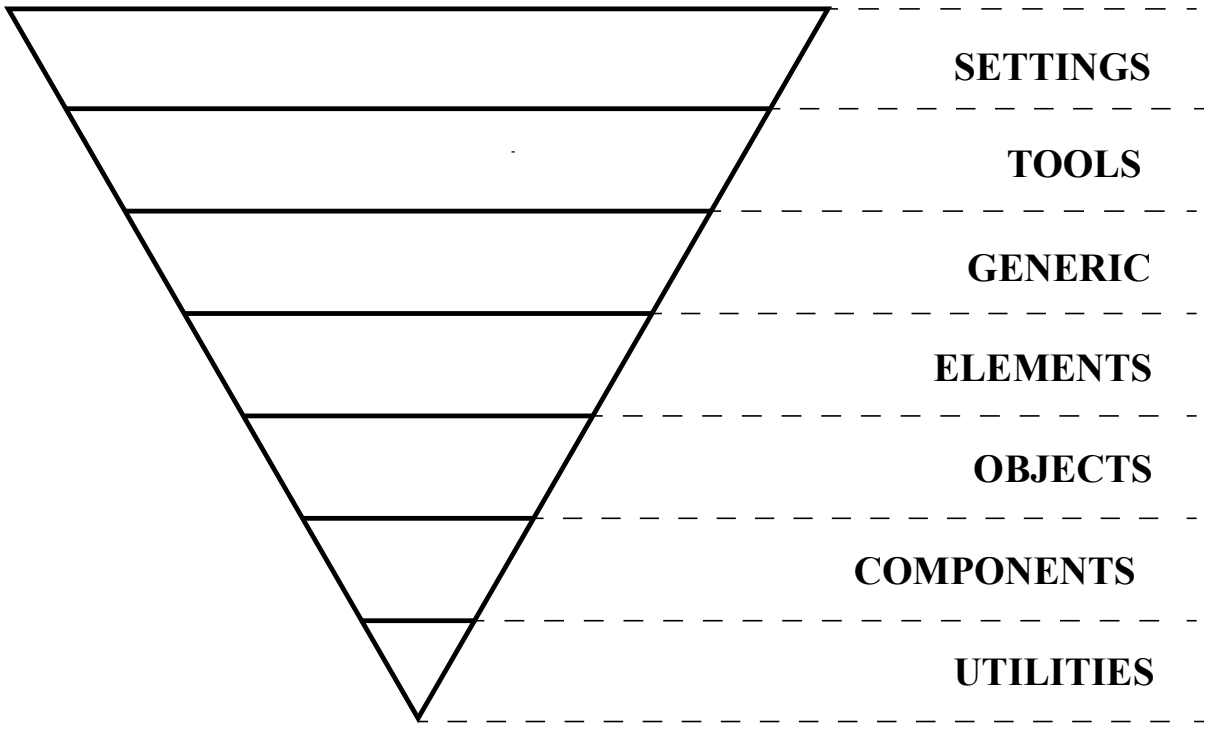
What is ITCSS?

ITCSS stands for *Inverted Triangle CSS* and it helps you to organize your project CSS files in such way that you can better **deal with** (not always easy-to-deal with) CSS specifics like **global namespace, cascade and selectors specificity**.

ITCSS can be used with preprocessors or without them and is compatible with CSS methodologies like BEM, SMACSS or

OOCSS.

One of the key principles of ITCSS is that it separates your CSS codebase to several sections (called *layers*), which take form of the inverted triangle:

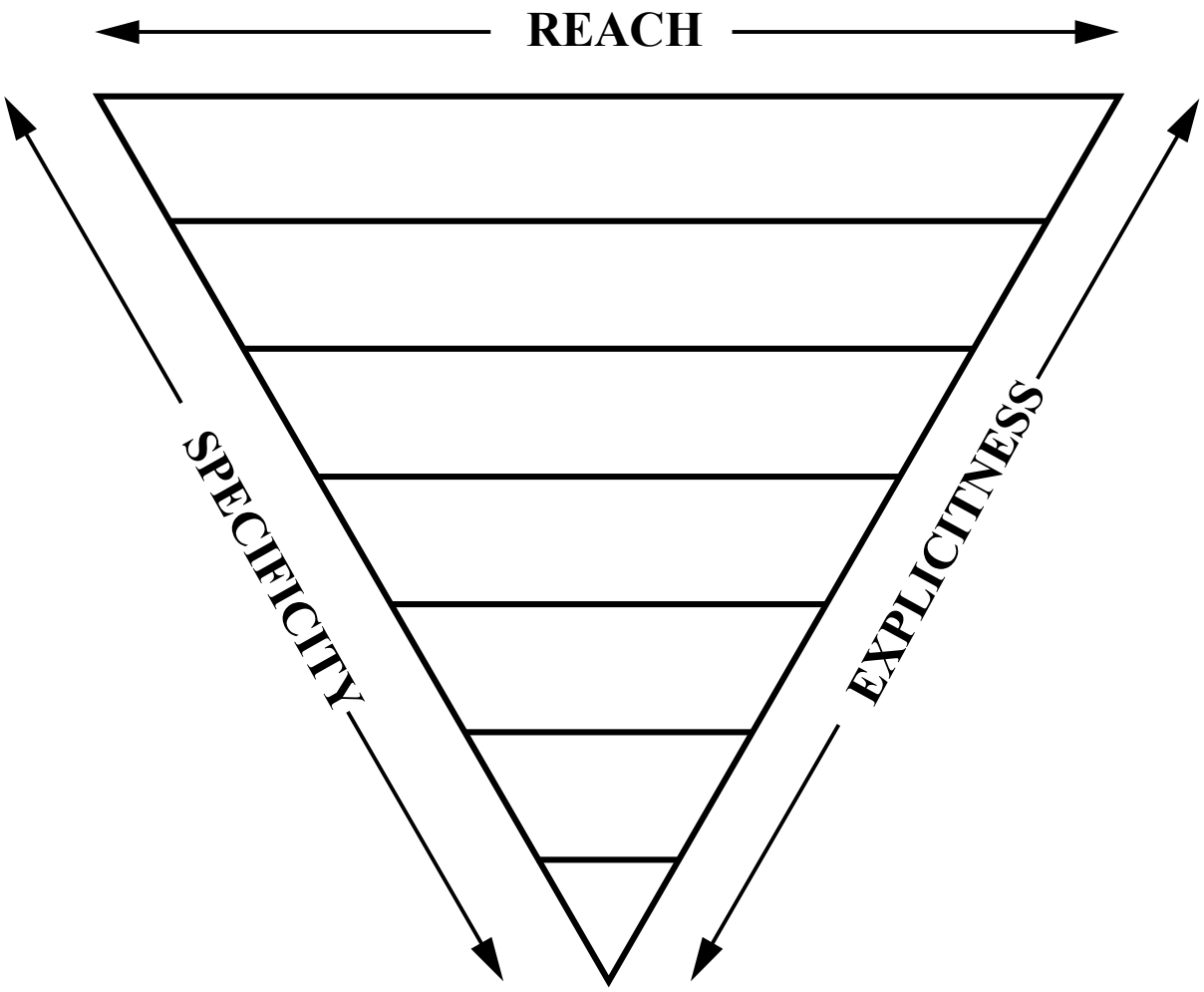


Those layers are as follows:

- **Settings** – used with preprocessors and contain font, colors definitions, etc.
- **Tools** – globally used mixins and functions. It's important not to output any CSS in the first 2 layers.
- **Generic** – reset and/or normalize styles, box-sizing definition, etc. This is the first layer which generates actual CSS.
- **Elements** – styling for bare HTML elements (like H1, A, etc.). These come with default styling from the browser so we can redefine them here.
- **Objects** – class-based selectors which define undecorated design patterns, for example media object known from OOCSS

- **Components** – specific UI components. This is where majority of our work takes place and our UI components are often composed of Objects and Components
- **Utilities** – utilities and helper classes with ability to override anything which goes before in the triangle, eg. hide helper class

The triangle also shows how styles represented by selectors are ordered in the resulting CSS: from generic styles to explicit ones, from low-specificity selectors to more specific ones (but still *not too* specific, IDs are not allowed) and from far reaching to localized ones.



Such CSS organization should help you avoid Specificity Wars and is represented by a healthy specificity graph.

Documentation

Update 10/27/2016: The net magazine has just republished the original article from the print version (see the resources below).

Normally, at this point I would refer you to the ITCSS webpage for further study. However, nothing like open source documentation exists.

ITCSS remains partially proprietary and if you want to fully utilize it, you should study the original introduction in the net magazine. I'm not here to judge author's intentions (I'm thankful to him for sharing his knowledge), but I think this prevents ITCSS wider adoption (which might be the intention, after all).

> The partially proprietary character of ITCSS prevents its wider adoption.

This shouldn't prevent you from start to use it in your projects, though, if you are really interested in doing so. Get that particular issue of the net magazine to learn ITCSS fundamentals, and then study available online resources and examples to help you with its adoption in real-life projects.

Resources

I've used ITCSS on 4 projects so far (including Xfive.co) and the following resources helped me to get better understanding of it:

- Manage large CSS projects with ITCSS – ITCSS introduction by Harry Roberts (the original article republished from the print version, missing are shorter columns on the specificity graph and preprocessors)

- Manage large-scale web projects with new CSS architecture ITCSS – ITCSS introduction and interview with Harry Roberts
- Harry Roberts – Managing CSS Projects with ITCSS – a talk given by Harry at DaFED and its accompanying slides
- Manage large CSS projects with ITCSS – a screencast for the net article
- ITCSS Screencast code – code from the above screencast at GitHub
- Another ITCSS project example
- Articles at csswizardry.com and especially the following ones:
- BEMIT: Taking the BEM Naming Convention a Step Further
- More Transparent UI Code with Namespaces
- Immutable CSS
- The Specificity Graph
- inuitcss – OOCSS framework which is based on ITCSS and shows some more advanced concepts and possibilities
- The BEMIT naming convention

You can also check out Chisel, our Yeoman generator for front-end and WordPress projects, which supports ITCSS.

Experience

Here are a few thoughts based on my experience with ITCSS projects:

Less thinking on naming and styles location

ITCSS's prescriptive nature especially when combined with BEMIT naming convention allows you to focus more on solving front-end challenges rather than thinking up names and styles location. This is what Xfive.co main.scss looks like:

```
@import "settings.colors";
@import "settings.global";

@import "tools.mixins";

@import "normalize-scss/normalize.scss";
@import "generic.reset";
@import "generic.box-sizing";
@import "generic.shared";

@import "elements.headings";
@import "elements.hr";
@import "elements.forms";
@import "elements.links";
@import "elements.lists";
@import "elements.page";
@import "elements.quotes";
@import "elements.tables";

@import "objects.animations";
@import "objects.drawer";
@import "objects.list-bare";
@import "objects.media";
@import "objects.layout";
@import "objects.overlays";
```

```
@import "components.404";
@import "components.about";
@import "components.archive";
@import "components.avatars";
@import "components.blog-post";
@import "components.buttons";
@import "components.callout";
@import "components.clients";
@import "components.comments";
@import "components.contact";
@import "components.cta";
@import "components.faq";
@import "components.features";
@import "components.footer";
@import "components.forms";
@import "components.header";
@import "components.headings";
@import "components.hero";
@import "components.jobs";
@import "components.legal-nav";
@import "components.main-cta";
@import "components.main-nav";
@import "components.newsletter";
@import "components.page-title";
@import "components.pagination";
@import "components.post-teaser";
@import "components.process";
@import "components.quote-banner";
@import "components.offices";
```

```
@import "components.sec-nav";
@import "components.services";
@import "components.share-buttons";
@import "components.social-media";
@import "components.team";
@import "components.testimonials";
@import "components.topbar";
@import "components.reasons";
@import "components.wordpress";
@import "components.work-list";
@import "components.work-detail";

@import "vendor.prism";

@import "trumps.clearfix";
@import "trumps.utilities";

@import "healthcheck";
```

Note: We use separate folders for each layer and load newly added stylesheets automatically in Chisel.

Objects reusability for fast development

ITCSS' objects are perfect candidates for building a library of reusable components to allow fast front-end development. UI parts would then be composed of generic objects and project specific components. For example, innuitcss as a generic ITCSS based framework contains a bunch of objects but only one sample component.

Animations

I recommend defining generic, global animations as objects too, eg. *@keyframes o-fade-in* in file *_objects.animations.scss*

Component specific animations should be defined in respective components files, eg. *@keyframes c-hero-scale* in *_components.hero.scss.*

Flexibility

ITCSS is quite flexible in terms of your workflow and tools. One of our developers expressed a concern about how much boilerplate ITCSS comes with. But in fact this is totally up to you - ITCSS doesn't prescribe that you need to have all layers present (only in what order they should be if they are present).

So in a minimal setup you can have just components with default elements styling coming from the browser. Of course, this is not very practical - some settings, reset and/or normalize CSS are used by almost everyone for good reasons.

Critical CSS

ITCSS plays nice with critical CSS due to the inverted triangle key metrics. The component based model allows you to separate above the fold UI into logical components so you can even pick parts of your critical CSS ‘manually’ (more on that in an upcoming article).

File size and styles duplication

If there is any concern with an architecture like ITCSS or basically any component like CSS architecture, it might be the resulting file

size. Components encapsulate styles and allow us to avoid CSS conflicts and overrides but it also means that styles repetition can occur quite often.

ITCSS cannot compete with a functional CSS in this sense. On the other hand, if you find yourself repeating a lot of styling in components, you could consider moving those styles to separate objects.

Conclusion

You cannot go wrong with ITCSS. It's the result of the experience and many years of work by Harry Roberts, one of the most renowned CSS authors out there. If you don't mind digging into the resources a bit, you will be rewarded with a simple but powerful architecture which will allow you to create scalable and maintainable CSS for your small or big projects.

But don't forget to keep an eye on other players like CSS modules, in the meantime.

--

Interested in more ITCSS?

- ITCSS: A Year After - Five insights from the year with Inverted Triangle CSS.
- Check out Chisel, a development framework for creating easy to maintain and fast WordPress websites and front-end templates which supports ITCSS