sitepoint premium

Handout

Principals of CSS Architecture
Lesson 4 - Step 2
Zsolt Nagy

In this video, we're going to talk about establishing hierarchical styles. First, we're going to look at the different elements a page consists of. And then we'll figure out what additional styles go to the top of the hierarchy. We're going to have a look at two different CSS architectures that are hierarchical in nature, ITCSS and SMACSS.

When we have a look at the website, the top level entity that we can identify is often times a page. A page may consist of a header, a main area, a footer, and optionally some side bars. Any of these elements except for the main area may be missing.

Top-level entity is still a page. As we dig a bit deeper, we can see some layout views inside the page. A layout may be a header, a main area, left sidebar, right sidebar, footer, and so on. I Personally have some experience programming in the framework called Marionette.js. It is a JavaScript framework which also has some layout views in JavaScript that represent a DOM segment.

Typically layouts used contain all the elements you are going to see in the next slides. A layout is still a relatively high level element that's responsible for gathering certain content that implement business use cases. Depending on the way how you establish a hierarchy, an optional element inside the hierarchy may be a module.

A module is a collection of components that solve a specific task. It makes sense to separate components into modules in case your application is relatively big and you have relatively large forms, where different sections gather different items. A module can group some of these items that belong together semantically.

According to some points of view, a module is nothing else but a component. Alternatively, a module can also be a layout anchoring to other points of view. Components are self-contained in nature. They obey the single responsibility principle and they solve a specific task.

I have to mention here that some recent articles published in 2017 advocate that we don't need Sass, we don't need block element modifier. We don't need any of these elements that we learned in this course so far. Because of the reason that we can simply include CSS using webpack for example with the ES6 module syntax.

And for this reason, given that the CSS is inside a module and it's self contained, there is not going to be any name clashes whatsoever. This is also an interesting approach, however, I have to warn you that in order for this to happen the front end development team and the styling team have to cooperate together.

On some level, if you write CSS and markup you also have to understand some aspects of application logic. Otherwise you will have a hard time locating and also integration testing your elements.

You may or may not choose to call a higher of component and module. A module is basically a group of components where we based a grouping on a business use case. It is up to you whether you use it or not.

A layout is a specific area on your page which may contain modules, and/or components. Layouts are page sections and typically determine where they're placed inside the page with respect to other layouts. Some examples for layouts are main area, header, footer, sidebars. Another very interesting hierarchy is encouraged by Marionette.js where we have layouts and we can define regions inside layouts. Each region may contain a composite or a simple component. Our top level entity is a page. It consists of everything you can see on a website at a given time, a collection off layouts that contain higher level components and lower level components and potentially modules that encapsulates components.

In case of a web application often times a header static and the main area changes based on what menu item we select. The footer is also typically static. Let's now examine the top four levels of the ITCSS hierarchy. Remember ITCSS is an inverted triangle so the top four levels are the bottom of the screen, starting from Elements and finish with Trumps.

Elements contain basic tag styles. They override the reset or normalizer that we use in the generic layer and give consistent styling to markup elements.

Remember in ITCSS, the higher the specifity, the higher the priority, the CSS tool is taken into consideration. For this reason, the elements layer overrides the generic layer. This is how elements build on top of generic. The objects layer typically contains generally classes that do not depend on the business context.

This means that for instance we can include layouting styles here which determines the generic positioning rules of our elements. We don't know yet what kind of of components we are going to build but we set up the basics, the positioning, the layouting and so on. Our UI components, how surprising, go to the components layer.

Components are already very specific. They build on top of all the other layers that have been introduced so far. And components are the building blocks that we use on our page by placing them in the markup. Component styles are self-contained and most of the time we don't need any additional CSS to synchronize the component styles.

Now imagine all the layout and placing CSS rules are in the objects layer. And all the components styles are in the components layer. Are top layer, Trumps contains overrides. Overrides are generally not welcome in CSS and often times you can see that whenever you create an override, you might be able to modify something inside components or objects or somewhere to generalize, to extract your specific override.

However, sometimes you might need quick fixes and sometimes there's just no other way but to create an override for a very specific case. Given that an architecture should be flexible enough to support this specific cases, there is a place for it and this is the very top level of the ITCSS hierarchy.

Don't overuse it, however, if you need it very much for instance for a hot fix that has to be deployed within 10 minutes, go for it. This was the ITCSS hierarchy. We are going to put it into practice in the next section. SMACSS or SMACSS is also hierarchical.

We can find five different levels of hierarchy in the SMACSS book base layout module state and theme. It's comparable to the ITCSS architecture because we have based on layout for pages layouts. We have modules which could be modules and components, states or rather component states, and what we didn't have before is themes.

Themes are the very end, they could be also supported by ITCSS, but they were not named in the ITCSS hierarchy. Base rules in SMACSS are generic rules in ITCSS. They are responsible for styling generic tag names, overriding the applied reset or normalizer. Layout rules correspond to the elements layer of ITCSS.

Module rules in SMACSS correspond to the components layer in ITCSS. Modules are reusable components. What comes after modules is less defined. This is because SMACSS defines states and themes. In my opinion, component states belong to the component styles therefore, they're supposed to be in the components layer of ITCSS.