# Bootstrap 3 Less Workflow Tutorial

Posted September 26, 2013

Bootstrap 3 is still new, and you may not be totally familiar with how it uses Less, and how you can use it too. This is a tutorial on a Less workflow that can get you up and running and using Less right away. It can be confusing to set up a workflow your first time, but once you do it, you'll never want to go back. And the best part is that you can use the workflow with or without Bootstrap 3. But for this post, I am going to talk about how you can use the power of Less with Bootstrap and make your life a whole lot easier.

Nowhere in this post will you see "less is more."

(Read the Part 1 about a Bootstrap Grid Introduction here Bootstrap 3 Grid Introduction Tutorial (http://www.helloerik.com/bootstrap-3-grid-introduction) and The Subtle Magic Behind Why the Bootstrap 3 Grid Works (http://www.helloerik.com/the-subtle-magic-behind-why-the-bootstrap-3-grid-works).)

## Introduction to Less

Less is one of the popular pre-compiled CSS languages. There are a few other very popular languages that function essentially in the same way, but Bootstrap has picked Less as their flavor. Some people like it, some like to use other things (https://github.com/thomas-mcdonald/bootstrap-sass), but if you're getting introduced to Less, I'd recommend just sticking with it until you get the hang of CSS preprocessors.

To fully understand Less, go read the documentation at LessCSS.org (http://www.lesscss.org). There are endless ways to use it, more than I could ever explain. Less can be quite powerful once you've mastered all of the dynamic and repeatable capabilities. If you are not familiar with it at all, you might want to go skim it's site to at least get the basic idea.

Less gets compiled from the its own syntax, into CSS. They are pretty much the same, Less just adds more options and ways to actually "author" it, but if you can write CSS, you can write Less. To compile it, there are command line and apps that compile it for you. I use an app called

Codekit (http://incident57.com/codekit/) (Mac). There is also Less.app (http://incident57.com/less/) which is free (Mac), and SimpLESS (http://wearekiss.com/simpless) (Windows and Mac) which is also free.

If you have Node.js (http://nodejs.org/) installed, there's a npm package for it you can install with `npm install -g less`. Any of these methods will work fine. The command line version is especially useful if you're using a build script or makefile. There are various ways to compress and minify the output CSS as well, since with the workflow I want to show you, you'll be combining the Bootstrap styles in with your custom styles.

None of this is specific to Bootstrap. You can start using Less in any project that uses CSS. I haven't written raw CSS in almost 2 years. If you do want to dive in to the Bootstrap Less world, this is how you get started.

## Without Less

A lot of people just use the bootstrap.min.css file and add their own stylessheet to it. This works when you're trying to do something very quick (or don't have a workflow set up), or when your modifications are minimal. The drawback of this is that you are totally reliant on Bootstrap's HTML classes to make any modifications to your HTML, i.e. if you want to hide something at a certain media size, you have to use `.hidden-lg`, or if you were to set up a responsive section with different widths, your HTML would need to look like `<div class="col-sm-4 col-md-5 col-lg-6"></div>.` If you are just using the bootstrap.min.css, you are stuck with just using the predefined styles that Bootstrap includes, and they all have to go in your HTML markup and can't be hidden away in a stylesheet.

Going without Less also makes it impossible to really change the attributes of a style responsively like this:

```
h1 {
  font-size: 22px;
  margin: bottom 10px;
  @media min-width(@screen-tablet) {
    font-size: 40px;
    margin-bottom: 20px;
    margin-left: 20px;
  }
}
```

By compiling the Less, you're able to do a lot more without a whole lot of hassle. Setting up a workflow is very easy, and the new styles you write in Less are probably not as much work as writing regular CSS.

I should mention an additional alternative is using Bootstrap's customizer. This allows you to change or remove things as you see fit, but it still leaves you with a static CSS file that you can't really leverage for serious customization. If you just don't care and want a CSS file and nothing else, the customizer is a great option.

## How Bootstrap uses Less

Bootstrap makes full use of Less. Almost all of the mobile first/responsive properties of the grid are built using Less variables and math functions. For example, the padding between grid columns is calculated like this:

```
.make-row(@gutter: @grid-gutter-width) {
  margin-left:  (@gutter / -2);
  margin-right: (@gutter / -2);
  .clearfix();
}
```

Or this

```
// Generate the small columns
.make-sm-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  min-height: 1px;
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);
  @media (min-width: @screen-sm) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}
```

You might be asking "Why do I care how a mixin is made?" Well, see how both those mixins are using using a variable called `@gutter` ? That is calculated in the variables folder:

```
@grid-gutter-width: 30px;
```

Starting to see the picture? If you changed that 30px value, it would be calculated all over the place and boom – you have a new gutter width size. The gutter size is used all over the place, but sourced from that singular variable.

Variables are used for almost everything:

- `@font-size-base: 14px`
- `@grid-columns: 12;`
- `@container-desktop: ((940px + @grid-gutter-width));`

Everything is based on some sort of variable, all stored in one place. This lets Bootstrap be built using a set of data that is highly customizable. If you did want to dive into the Bootstrap source you could modify it heavily and still output a usable, working custom version. Or you could overwrite the variables you need in a different file. That's not what I want to talk about today, but it is possible.

Bootstrap also has a huge list of mixins that you can use in your own Less. These mixins allow you to quickly add common attributes to any styles you make. And with the way that Less lets you use parameters, you don't have to do anything but specify how you want things to appear. For instance, one of the more common needs is a gradient. Bootstrap has a mixin already made, so all you need to do in your style is add this inside of the style you want it to mix into:

```
.horizontal-three-colors(@red; @yellow, 50%; @white);
```

Just one simple line where you chose 3 colors and a fade point, and that outputs everything you need in the CSS from this mixin:

```
.my-style {
  background-image: -webkit-gradient(left, linear, 0 0, 0 100%, from(#ff0000), color-stop(50
  background-image: -webkit-linear-gradient(left, #ff0000, #ffff00 50%, #ffffff);
  background-image: -moz-linear-gradient(left, #ff0000, #ffff00 50%, #ffffff);
  background-image: linear-gradient(to right, #ff0000, #ffff00 50%, #ffffff);
  background-repeat: no-repeat;
  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffff0000', endColorstr=
}
```

No one wants to write that by hand. Using Less variables and a mixin, it's easy. This isn't even specific to Bootstrap, it just  happens to include it. There are a lot of other examples, check Bootstrap's mixins.less file to see them all.

## Key Advantages

Using Less, especially with Bootstrap 3, gives you a few advantages that really make it a awesome choice:

- Changing variables in one place can have wide-reaching effects, for super easy global customization
- Augment, shift, and change existing styles and variables, or roll your own
- Easily override Bootstrap variables/mixins/classes with your own
- Have a clean, customized singular CSS file
- You can easily comment out any part of Bootstrap you don't need or want, right down to the single line
- Componentization. You can break your Less up into smaller files for very easy editing and separation, great for applications where you're using version control like GitHub

DRY: Don't repeat yourself. Using a preprocessor like Less lets you do things once, and reuse them anywhere. Once you have your variables and mixins built up, or use Bootstrap's defaults, you'll be able to just style things out without having to write new styles each time.

Let me show you how I do things. It's always a little different, but when I am starting a project, be it a website, prototype, UI idea, or just something undetermined, I have developed a system that works for me. Your experience with it may vary, and if you have other ways of doing things, be sure to post them in the comments below so we can all learn from each other!

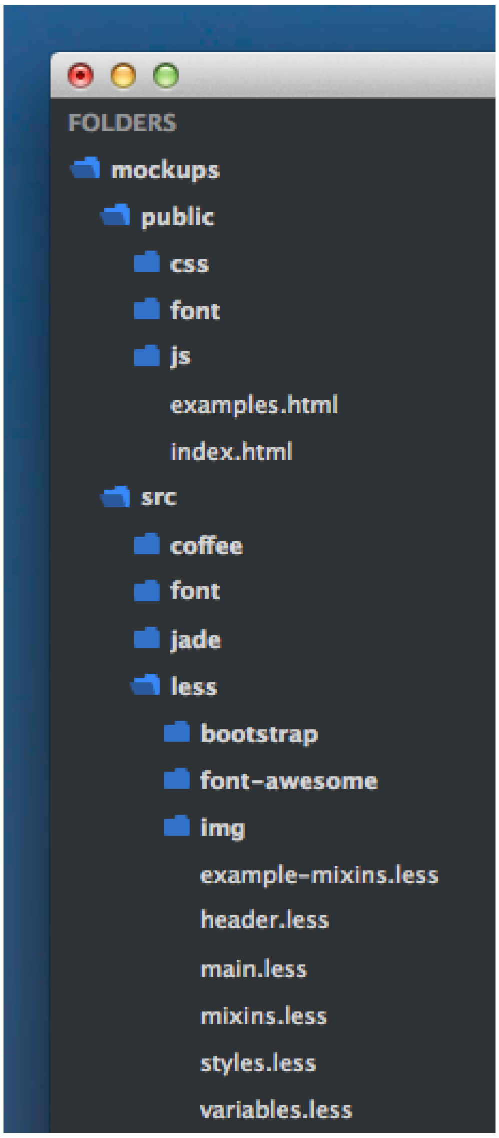
## My Example Workflow

I've been using this same method for almost 2 years now. I like to clone the repository from GitHub, and then copy the files from the repository on my local computer to my project folder. This gives me a clean, decoupled set of the Less files that are now orphaned from the repository. I prefer this so I can keep the Bootstrap repository updated on my machine, without having to check it for changes that might break what I have. When I'm ready, I can test any new changes locally without having to interrupt my project with an unpredicted Bootstrap Less change.

There are lots of ways to achieve the same results. Some people like to have the Github sources linked right in their project and to just pull it down. I prefer to add the files to my projects manually. Do whatever works for you.

I typically have some sort of "src" folder that holds my Less (and jade and coffeescript). This all compiles to a separate "public" directory where my actual project is output. You don't have to use any HTML templating language of course, you could still use this method for just Less.

I create a folder called “less” and in that, I make another folder called “bootstrap”. That is where all the Bootstrap Less files go. I leave the Bootstrap folder alone, this makes it really easy to update when a new version comes out. Technically, you could make this a Github form repo and just keep it up to date there, but I like to do it manually. This is how the project looks now:



```
FOLDERS
mockups
  public
    css
    font
    js
    examples.html
    index.html
  src
    coffee
    font
    jade
    less
      bootstrap
      font-awesome
      img
      example-mixins.less
      header.less
      main.less
      mixins.less
      styles.less
      variables.less
```

You can see in my src folder I have a Less folder that has the Bootstrap folder inside it (and font-awesome, which comes as a Less source as well). These will all get pulled down into a master CSS file and exported to the public/css folder above.

## Importing Less

Inside of the Less folder, I make a file called styles.less. This is what will be compiled to styles.css. You can name your Less file whatever you want, I’ve just always called it styles.less.

This styles file is important as it will contain only imports to other Less files. It doesn't have any styles itself. The way I do it, I import all of my granular Less files through this file. It looks something like this:

```
@import "bootstrap/bootstrap.less";
@import "font-awesome/font-awesome.less";

@import "variables.less";
@import "mixins.less";
@import "nav.less";
@import "header.less";
@import "footer.less";
```

It almost always goes the Bootstrap folder first, then font-awesome, then my own variables and mixins. I want to make sure these global pieces are at the top, since this is the order the CSS will be built.

Now that styles.less is made, the new granular files can be added at any time. I usually break it up quite a bit, into logical "things" that all are related organizationally on the page/site/UI.

## My Variables

When importing my own files, variables are first, so they can be used on anything below. Even though this file is named the same as the Bootstrap variables.less file, it won't overwrite anything unless you specifically name the variable the same. You can override existing Bootstrap variables with your own, or just start creating what you need. Most of the time, whatever I'm building has a basic set of colors, so I define those first, something like

```
@primary: hsl(202, 100%, 50%);
```

I always use HSL, so that I can adjust the colors without having to look them up or find a hex. Once you have a primary (and/or secondary) color, you can make a mini palette with some less:

```
// Single primary color that all mixins use. Str8 ballin.
@primary:        hsl(202, 100%, 50%);

// Primary Variants -
@primaryLight:   hsl(hue(@primary), 100%, 70%);
@primaryDark:    hsl(hue(@primary), 60%, 40%);
@primaryFaded:   hsl(hue(@primary), 60%, 65%);

// 180degree Variants
@variant:         spin((@primary), 180);
@variantLight:    hsl(hue(@variant), 100%, 70%);
@variantDark:     hsl(hue(@variant), 60%, 40%);
@variantFaded:    hsl(hue(@variant), 60%, 65%);
```

```
// 90degree Variants
@corner:         spin((@primary), 70);
@cornerLight:    hsl(hue(@corner), 100%, 70%);
@cornerDark:     hsl(hue(@corner), 60%, 40%);
@cornerFaded:    hsl(hue(@corner), 60%, 65%);
```

And there you go – I have a palette with 11 colors, all based off the first @primary variable. This isn't a lesson in Less though, it's about a Bootstrap 3 Less workflow, so I'll try to stay focused…

## My Mixins

Mixins come next, since they use the variable. You can mixin in anything from Bootstrap or your own new variables into a new mixin. It can get a little crazy, but most of the time you can just keep it simple. Having your own mixins is great for things you have to do a lot and want to shorten it up. One that I find myself using often is a mixin for coloring text shades of gray. Instead of using HSL or maybe lighten() or darken(), I've made this mixin. All I would do here is add a number in the parenthesis for how "gray" I want it, zero being black, 100 being white:

```
.gray(67);
```

Which is defined like this:

```
.gray(@percent) {
  color: lighten(@black, percentage(@percent / 100));
}
```

It's a bit quicker than typing color: hsl(0,0,50%). Not a huge deal, but for a designer like me, sometimes a shade of gray can be 1% off. This makes it easy for me. Here are some mixins I've made or adapted that I find very useful:

- Fixed screen layered background (https://gist.github.com/erikflowers/6393836)
- Super simple background-cover mixin (https://gist.github.com/erikflowers/5230313)
- Easy retina-display background image mixin. (https://gist.github.com/erikflowers/4522361) Super awesome.
- CSS Text Select Highlight (https://gist.github.com/erikflowers/6710354)
- Dead simple text shadows – 1px black or white (https://gist.github.com/erikflowers/6710372)
- Make a UL have no style, either inline or block (https://gist.github.com/erikflowers/6710391)

Building your own mixins and variables is a lot of fun, and is essential for making a very custom designed website. Anytime I think of something handy, I add it to my mixins file and insert that into each new project.

## Building Up My Styles

Once I've imported the Bootstrap files, font-awesome, and my own variables and mixins, I start with a main.less file this holds styles that I consider to be the most global. Things like styles for html, body, h1-h6, strong, em, essentially anything that is an element I just want changed everywhere. This file also holds things that I have no other good place to put yet, sort of a catchall. Usually by the end of a project this file is pretty small.

I usually end up with something like this for my styles.less file – for instance, this is the actual helloerik.com styles.less file:

```
@import "bootstrap/bootstrap.less";
@import "font-awesome/font-awesome.less";

@import "variables.less";
@import "mixins.less";
@import "main.less";

@import "nav.less";
@import "header.less";
@import "footer.less";
@import "home.less";
@import "posts.less";
@import "all-posts.less";
@import "loop-nav.less";
```

Sometimes, if there are global elements that are reused in lots of different parts of the site, I'll add a globals.less for custom buttons and other UI or layout elements that aren't a part of anything else, but also aren't main elements. Typically it ends up being a place for things that "widgets" around a site, or discrete elements like a custom tweet button. Do what feels best for you and your project.

As I move down the list, things get more specific. There is no technical need to break down the files into granular files, but I find it so much easier to keep things sorted out and scoped correctly. Since I use Github for most projects, this also lets the commits be on a more specific basis as the Less file usually corresponds with some equivalent code file of HTML or whatever is building the output HTML. As you can see above, I start with the nav.less, then header.less, and eventually end up at something pretty specific, loop-nav.less. But that had enough of its own styling that I wanted it in its own file, to keep it from cluttering up other areas.

From here, it's super simple. I just write my Less like you would CSS, add imported files when I feel it's necessary, and build up the projects variables file with things that are reused all over. Each time I hit "save", CodeKit compiles all the Less to a minified CSS file, and refreshes my browser, so I can see the changes instantly. It's an addictive workflow, instant feedback is awesome (It also does this with my Jade and CoffeeScript files, so I never have to actually press refresh, it just does it). Once I'm at this point, styling my own stuff is easy, and modifying Bootstraps stuff is easy. It's all easy. This keeps Bootstrap's files all self contained and keeps all my files organized and separate.

## Responsive Less

If you're new to Bootstrap 3, you'll notice that there is no responsive specific files anywhere. That's because Bootstrap 3 is built mobile-first with the responsive properties built right in.

This is another huge reason to use less. In my post about the Bootstrap 3 Grid Introduction Tutorial (http://www.helloerik.com/bootstrap-3-grid-introduction), I talk about how you can build the grid using just Less mixins and media queries. If you're just using the CSS files alone, you can't do any of this. Here's an example of how you can use your own Less to build styles in a much more power, flexible, and responsive way:

```
.main-content {
  .make-row();
  .sidebar {
    padding: 20px;
    @media(min-width: @screen-desktop) {
      padding: 30px;
    }
    h2 {
      font-size: 20px;
      @media(min-width: @screen-desktop) {
        font-size: 30px
      }
    }
    .make-sm-column(2);
    .make-md-column(3);
  }
  .content-area {
    line-height: 1.2em;
    @media(min-width: @screen-desktop) {
      line-height: 1.6em;
    }
    h1 {
      font-size: 30px;
      @media(min-width: @screen-desktop) {
        font-size: 40px;
      }
    }
    .make-sm-column(10);
    .make-md-column(9);
  }
}
```

You have complete control over what CSS is output. When you're building a site with a custom design and using Bootstrap 3 with the intent of going mobile first, you almost can't do it without using Less. Go read the article (http://www.helloerik.com/bootstrap-3-grid-introduction), I explain it in much more detail there, but this should give you an idea of the type of control you have when using Less and the Bootstrap included mixins and such.

## A Workflow Summary

With all that said and done, here's the workflow in summary:

1. Get Bootstrap (http://getbootstrap.com/)
2. Choose a Less compile method, an app or the command line
3. Make a "source" folder to hold your less. Call it whatever you want.
4. Put a folder called "bootstrap" in there, and put the Bootstrap Less files all in there
5. Inside your "source" folder, make a styles.less that imports bootstrap/bootstrap.less
6. In that same styles.less folder, import the rest of your desired Less files (main.less, header.less, footer.less, etc etc)
7. Write your Less in those granular files – not in the styles.less
8. Compile styles.less to wherever you want to refer to your styles.css. Minify/compress if you want.
9. That's it!

If you set this up once, you'll never want to go back to doing it with just CSS. You don't even have to use Bootstrap to do this, just follow the same steps but don't include the Bootstrap parts. This will get you using Bootstrap Less and your own Less so quick and easy, you'll find yourself saying "Wow, Less really is an increased value over the value I had previously."

Got questions, ideas, your own workflows to share, or places I screwed up? Post them in the Disqus thread below.

Like this article? Read the Part 1 about a Bootstrap Grid Introductions here Bootstrap 3 Grid Introduction Tutorial (http://www.helloerik.com/bootstrap-3-grid-introduction) and The Subtle Magic Behind Why the Bootstrap 3 Grid Works (http://www.helloerik.com/the-subtle-magic-behind-why-the-bootstrap-3-grid-works).

**Tweet (https://twitter.com/intent/tweet?url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial&text=Bootstrap 3 Less Workflow Tutorial via @Erik_UX&hashtags=UX,servicedesign)**

**Share (https://www.facebook.com/sharer/sharer.php?u=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial)**

**Share (https://plus.google.com/share?url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial)**

in Share (http://www.linkedin.com/shareArticle?mini=true&url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial&title=Bootstrap 3 Less Workflow Tutorial&source=<p>Working with Bootstrap 3 Less can be intimidating. This tutorial and workflow example will help you make the most of this powerful styling language and get the most out of Bootstrap.</p> )

**Posted by Erik Flowers (http://www.helloerik.com/author/admin) on September 26, 2013 (Thursday, September 26th, 2013, 7:10 pm)**

I love all things experience design. I work as a Principal Service Experience Designer at Intuit in Mountain View, CA.

**Follow @Erik_UX**

**119 Comments** Hello Erik - User Experience Blog and Portfolio 1 Login

Recommend 6 Share Sort by Best

Join the discussion…

**moeTi** • 2 years ago

I like how you named one of the file 'home.less' :-D

I always downloaded the CSS files of bootstrap because I had no idea what to do with LESS, but now I wanted to finally learn about. Good post, thanks!

22 • Reply • Share ›

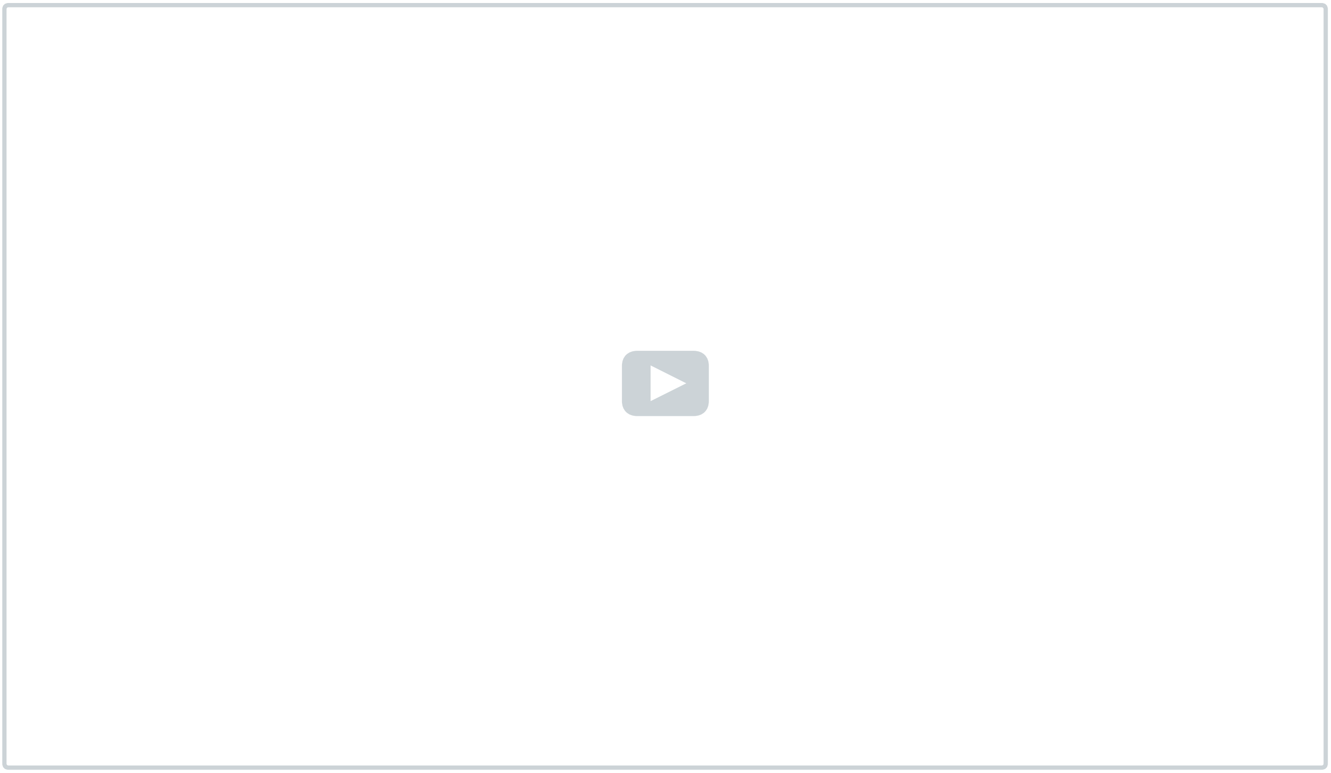
**Hef** • 2 years ago

Thank you!! Great article. Number one Bootstrap 3 usage manual.

6 ^ | v • Reply • Share ›

**cweekly** • a year ago

I urge readers to use Grunt (and the grunt-recess module) as this is how Bootstrap itself is developed. Best to use the same lessc compiler as the original project. And once you set up Grunt, it's easy to add live-reload and other game-changing improvements to your workflow. Check out Paul Irish



(this talk is a little old, but remains a great starting point for learning about modern FE dev workflow automation).

3 ^ | v • Reply • Share ›

**Kyle Farris** • 2 years ago

This has been extremely informative. Thank you for sharing this, it's been a big help getting me started on the Less and Bootstrap bandwagon.

3 ^ | v • Reply • Share ›

**Antonio Brandao** • 2 years ago

Hello, really great post. But what abut the file "theme.less" in the bootstrap less folder? You don't use it? You don't mention it in this article. I thought of importing it to the "styles.less" folder and use it as a base to customise my theme design. Does this make sense?

3 ^ | v • Reply • Share ›

**Erik Flowers** Owner ➦ Antonio Brandao • 2 years ago

It does. I start just with the raw bootstrap styles, building upon those and not the theme.less. There's nothing wrong with starting that way though if you don't want to build up those common styles on your own.

It's all about the level of custom design you want to use. I like to pare it down to just the minimal bootstrap styles, and v3 is perfect for that.

3 ^ | ˅ • Reply • Share ›

**RonWade** • a year ago

Thanks, this article is really helpful. I was wondering how to set up a workflow that would leave Bootsrap's files untouched, and allow me to overwrite specific variables. You answered all my concerns.

2 ^ | ˅ • Reply • Share ›

**TerminatorFiles** • 6 months ago

First and foremost, thanks to Erik for this awesome post about his LESS workflow. I've read through all the comments, to see what else I can learn from you and other commentators... or add my own.

Some users wonder if its needed to comment out some of the unused Bootstrap LESS files, in case some parts are not used. I would too consider this good practice, especially when related to mobile and possible additional costs for your mobile users.

However... please try to see things in perspective: you can try your utter best to get the final styles.css from a minified version of (lets say) 200KB, to an optimized 160KB... saving you 40KB. Put next to the CSS there are lots of other files, like JavaScript files, images etc. If you add or remove a single image, the app or site can de/increase with lots more that the 40KB you saved trying to over-optimize.

I used to do this and work for hours to strip my code to the bare minimum, only to find out that some of the other designers working on the same project, included a few extra images, which entailed to the same (or even bigger) amount of KB, which I was trying to save in the first place.

Don't over optimize ;).

That said... thanks again to Erik for the awesome post. Our new fansite is going to be build a lot faster thanks to his LESS blog post(s)... and other WinLess, LiveReload tools and methods!

1 ^ | ˅ • Reply • Share ›

**Jessy Cormier** → TerminatorFiles • 6 months ago

I would tend to agree with you for both parts (don't over optimize). I'm trying to figure out a way to write the code more cleanly then just doing fast overrides though.

If we don't touch the bootstrap files and only write overwrites then we tend to see the output CSS look something like this:

.test { stuff }

.test { more stuff}
.test {extra stuff more processing because we can sigh*** }

where as if we customize each bootstrap file we could avoid this problem. (but obviously gain another problem, which would be upgrade-ability...)

Any thoughts on this?

• Reply • Share ›

**Erik Flowers** Owner Jessy Cormier • 6 months ago

I tend to want to make the upgradeability and toolchain flow as simple as I can, which means I don't do anything but import Bootstrap as is, and have commented out the pieces I don't need. I never feel like the output CSS is too complex for browsers to parse fast, and I also uglify it down to illegibility, so I shave off whatever I can since no one needs to see the CSS.

I realize there's selector crawl overhead, but really, I don't work at a scale that ever really effects anyone.

2 • Reply • Share ›

**Jessy Cormier** Erik Flowers • 6 months ago

Thanks for your feedback **@Erik Flowers**

• Reply • Share ›

**salma** • a year ago

Erik, how can I use parts of bootstrap, for example I already have an HTML5 theme that I would like to modify. I would like to use bootstrap Wells and Buttons within the theme. I compiled bootstrap-style.less with netbeans using @import bootstrap.less

I added the link to bootstrap-style.css below the theme style.css link but still I am not able to get the bootstrap buttons to work. Here is my HTML code:

```
<noscript>
<link rel="stylesheet" href="css/skel.css"/>
<link rel="stylesheet" href="css/style.css"/>
<link rel="stylesheet" href="css/style-wide.css"/>
<link rel="stylesheet" href="css/style-normal.css"/>

</noscript>
<link rel="stylesheet" href="css/bootstrap-style.css"/>
<link rel="stylesheet" href="css/custom.css"/>

<div class="row-fluid">
<div class="span3">
<div class="well well-lg"><h1 class="">Patient Information</h1>
```

<img class="media-object" data-src="holder.js/200x200" alt="500x500" src="data:image/svg+xml;base64..." style="width: 200px; height: 200px; margin:0px auto">
<button type="button" class="bootstrap-style btn btn-warning">Click Here</button>
</div>
</div>

<div>

1 • Reply • Share ›

**Robbie Ferguson** • a year ago

Thanks for this **@Erik Flowers**... I'm afraid I've fallen behind the times a little on this stuff... I have been needing to get my head around this for some time but being actively developing sites day in and out, I have had trouble finding the time to just stop what I'm doing and learn what's current. This post really helped me get my head around the basics of Less and how it relates to Bootstrap. I'm sure it's a good starting point as I endeavor to upgrade my skillset through trial-and-error, and I can finally stop using CSS overrides on my Bootstrap deployments. :)

1 • Reply • Share ›

**Erik Flowers** Owner → Robbie Ferguson • a year ago

yeah, CSS overrides through CSS are dead. It's all Less. If you're on a mac go buy CodeKit and use that to compile your less ;)

1 • Reply • Share ›

**Robbie Ferguson** → Erik Flowers • a year ago

I'm on Linux. I wrote a quick shell script to take care of the compiling & minifying for me and automatically put the resulting CSS files onto the CDN and change the db entry to force users' browsers to re-cache. Works for me :)

• Reply • Share ›

**Alexandru Giuseppe Ispas** • a year ago

What about the default "js", "css", "fonts" folders which are included in Bootstrap. How should I arrange those? I understand that for the less files I need to create a bootstrap folder inside my less folder and paste there all the default less files from bootstrap but where should I put the js files, fonts which are included also in Bootstrap folder? How do you suggest to arrange this assets for a good workflow ?

1 • Reply • Share ›

**Erik Flowers** Owner → Alexandru Giuseppe Ispas • a year ago

I explain it in the "My Example Workflow" section above. You should have your Less stored in a pre-compiled folder outside of your actual website. The JS and other files go in the active folder, but your Less (and whatever else you precompile, jade, coffeescript) are in a different working folder.

Your less is then spit out into your CSS folder, and you'd just leave the bootstrap javascript alone. In that section above you can see an image of how my folders are set up. Your public folder can still be set up with your root HTML in the root folder, then a css/ fonts/ and js/ folder there too. The Less is output into the css/ folder as 1 css file.

• Reply • Share ›

**Alexandru Giuseppe Ispas** → Erik Flowers • a year ago

So, you're saying that I should create inside the "public folder" css/ / fonts/ js/ and put there all the bootstrap default files alongside my own custom js,css files ?

• Reply • Share ›

**Erik Flowers** Owner → Alexandru Giuseppe Ispas • a year ago

You do not need the bootstrap.css file as everything it has is already contained i your own styles.css custom file that you compiled. The Less you have already pulls in all the bootstrap less, and your own less.

1 • Reply • Share ›

**Dave Diomede** • a month ago

How do you reference the sources in your html doc?

• Reply • Share ›

**Erik Flowers** Owner → Dave Diomede • a month ago

They are not referenced, they are compiled into a single CSS file. This all assumes that you compile the Less first into a public output file and don't rely on using the less.js to compile at runtime.

• Reply • Share ›

**Dave Diomede** → Erik Flowers • a month ago

Thank you for the reply and the great tutorial.

• Reply • Share ›

**adrianmak** • 4 months ago

I'm confused. If you're using bootstrap mixins to generate grid css , why you still imported a full bootstrap less ? As you will not write bootstrap css class in html directly

• Reply • Share ›

**Erik Flowers** Owner → adrianmak • 4 months ago

I am not using the bootstrap CSS, I include the bootstrap.less file and whatever other ones it imports, and build the final CSS with that. It doesn't duplicate anything. The grid less files are a part of the main Bootstrap import, not as a part of my own variables. I just take bootstrap less as a whole and add on what I want.

• Reply • Share ›

**adrianmak** ➦ Erik Flowers • 4 months ago

Then, the compiled css will include original bootstrap css class, right ?

• Reply • Share ›

**Erik Flowers** Owner ➦ adrianmak • 4 months ago

Yes, the final output includes the bootstrap less. You'll only need 1 file at the end, it will have bootstrap, and your own styles.

• Reply • Share ›

**Ryan Serrano** • 4 months ago

Hello there, excellent post, like you said I've been using min.css for a while now and I start looking around to automatize my workflow, LESS got my attention and I intend to learn this, I feel like when I discover classes, loops and variables on PHP lol, so I believe this is the way to go... BUT I don't quite get it yet, Can you explain to me like a 12 Years Old the work flow? I mean, I think it goes like this... I do all the steps you mention on this great article, then I'm going to obtained a min.ccs file that I'm going to use on my project? and if I don't like the looks I can export another one fixing the css variables to my designer desires and replace the old one? is this great or am i missing something? Thanks again for this post.

• Reply • Share ›

**Dapo Adediran** • 6 months ago

Thanks! This is a life saver!

• Reply • Share ›

**iBcht** • 7 months ago

Good experience you shared there ! For Wordpress front-end developers, I advice to try Roots.io, a very good bootstrap/less/grunt starter theme.

I just noticed one thing : "Inside of the Less folder, I make a file called styles.less. This is what will be compiled to styles.css. You can name your Less file whatever you want, I've just always called it styles.css." -> I guess you were talking about "styles.less" at the end ;)

• Reply • Share ›

**Jonas Donbæk** • 8 months ago

Hi Erik,

Thank you for this great article!

I got some troubles when i want to compile the css. You can see my folder structure here:

see more

^ | ˅ • Reply • Share ›

**Erik Flowers** Owner ➦ Jonas Donbæk • 8 months ago

I am not sure that the variable @navbar-inverse: hsla(267, 100%, 50%, 1); exists. All you are doing there is setting that variable to be a color. But I don't see that variable used anywhere in the Bootstrap Less.

I see these:

@navbar-inverse-color: lighten(@gray-light, 15%);
@navbar-inverse-bg: #222;
@navbar-inverse-border: darken(@navbar-inverse-bg, 10%);

So that is the navbar-inverse class text color, background, and border. So if you wanted to change the navbar-inverse colors, you'd add your own version of those.

But if @navbar-inverse isn't a variable that is used somewhere, it won't do anything.

It looks like the class that uses the colors first is ".navbar-inverse", but that's a class, not a variable.

I would suggest adding a class to your header.less like this

```
.navbar-inverse {
background-color: @navbar-inverse-bg;
border-color: @navbar-inverse-border;

}
```

And change the values there, either as a variable, or change them directly to colors.

^ | ˅ • Reply • Share ›

**Dima Karpov** • 9 months ago

This post helped me so much during this project, again want to say thank you.

^ | ˅ • Reply • Share ›

**Katie Womersley** • 9 months ago

Thanks for sharing this! It was a very clear and useful guide and I'm excited to have a more productive workflow. Thanks!

• Reply • Share ›

**Biju UI Developer: www.bijudes** • 9 months ago

Thanks a lot

• Reply • Share ›

**Dima Karpov** • 9 months ago

Thank you a lot, this is super informative. It's like you open new planet for me.

• Reply • Share ›

**Ian Baldwin** • 10 months ago

Thank you! This article answered everything I needed to know about getting started with LESS in Bootstrap 3.

• Reply • Share ›

**Sawyer** • a year ago

Hello,
so in src/bootstrap you put the the file bootstrap.min.css?
And styles.css import all .less files to the bootstrap.min.css , is that correct?

• Reply • Share ›

> **Mr Kuuk** Sawyer • 10 months ago
>
> I reply to my own post :), i got it up and running now, had to think twice with the workflow.
> Thanks
>
> • Reply • Share ›

**Teo Maragakis** • a year ago

I use a similar workflow. Still, very nice and a great tutorial for Bootstrap beginners!

• Reply • Share ›

**Saptarshi** • a year ago

This is such a fantastic tutorial! Thanks for sharing Erik. As someone who is just starting out with Sass (yes, not Less) this helped me a lot, because the basic principles are the same between Sass and Less.

• Reply • Share ›

**Unai Aizpurua** • a year ago

Hi Erik,

Very interesting post, and great ideas for how establish a great workflow.
I have a little question rounding my head.

This way you generate your .css file from .less, great. But what about the .js files or font files that grunt generates when you compile Bootstrap?
I need to compile outside codekit or everything can be done in one time?

Regards, and keep with your great blog!

• Reply • Share ›

**magebay** • a year ago

Great tip, thank you!

• Reply • Share ›

**Vijay** • a year ago

Thanks Erik, After doing all the things you have explained, finally in the HTML Page Should I add the less css and js in this order:
<link rel="stylesheet/less" type="text/css" href="less/styles.less">
<script src="http://cdnjs.cloudflare.com/aj..." type="text/javascript"></script>

or

I can use the compiled styles.css file in the html instead of styles.less, please clarify.

• Reply • Share ›

**Alexandru Giuseppe Ispas** • a year ago

I want to create some custom variables using Bootstrap's official generator or Bootswatchr, like colors, buttons, navs ... and I'm thinking to download first the original less files and then to create a file called "variables.less" outside the Bootstrap folder and paste there the new variables. Do you think this a good approach ?

• Reply • Share ›

**Erik Flowers** Owner → Alexandru Giuseppe Ispas • a year ago

I am not sure what you mean, but if you are adding anything to the Bootstrap Less source, you'll want to follow my guide and have your own styles.less that imports bootstrap.less and then your own variables.less etc. Just keep bootstrap in its own /bootstrap folder.

So if you replace a bootstrap variable in your own files, it will replace the Bootstrap one. Get it?

• Reply • Share ›

**Alexandru Giuseppe Ispas** → Erik Flowers • a year ago

I get it know. What I need is creating a folder called bootstrap with all the original files and then outside of this folder to create my own variables file, mixins and so on which will overwrite the original ones if it's the case.

• Reply • Share ›

**Erik Flowers** Owner Alexandru Giuseppe Ispas • a year ago

Yes. And @import the bootstrap/bootstrap.less and your own files into a single styles.less.

• Reply • Share ›

**Nina** • a year ago

This is a very good tutorial. Now I have a better understanding of the LESS workflow. Other resource links lead me to more information. GOOD JOB!

• Reply • Share ›

**Sarapan Malam** • a year ago

thank you.. very helpful

• Reply • Share ›

**Jeremiah Blanch** • a year ago

Hi Erik, this is a really useful post, thanks. Can I clarify: You put all the original Bootstrap LESS files into the bootstrap folder, but then you don't modify them? So to put in brand colors and sizes etc, you would overwrite the values of the bootstrap variables in your own LESS files? But then how do you have bootstrap build itself with the correct colors (when you simply import the entire root bootstrap.less file)? Or do you import bootstrap's variables.less, then import your own variables.less, and then import the rest of the bootstrap less?

• Reply • Share ›

Load more comments

ALSO ON **HELLO ERIK - USER EXPERIENCE BLOG AND PORTFOLIO** WHAT'S THIS?

**3 Analogies For the Aggregate Nature of Service Design**
2 comments • a year ago

**12 Hours to TajRiba Keynote. Helping bring true UX to Africa.**
1 comment • 2 years ago

**The Subtle Magic Behind Why the Bootstrap 3 Grid Works**
110 comments • 2 years ago

**My Intuit Service Design Journey, Year 1**
1 comment • 23 days ago

← Previous(http://www.helloerik.com/return-to-tajriba-prototyping-look-forward-to-surprises)
(http://www.helloerik.com/the-first-tajriba-bringing-ux-to-africa)Next →

## Follow me on Twitter (https://twitter.com/intent/user?screen_name=Erik_UX)

**Follow @Erik_UX**

HELLOERIK.COM

The UX blog of Erik Flowers.

HelloErik RSS Feed (http://www.helloerik.com/feed/)

RECENT WRITINGS

My Interview with Campfires.io (http://www.helloerik.com/my-interview-with-campfires-io) | My Intuit Service Design Journey, Year 1 (http://www.helloerik.com/my-intuit-service-design-journey-year-1) | What it's like to work at a very specific, single Silicon Valley tech giant (http://www.helloerik.com/what-its-like-to-work-at-a-very-specific-single-silicon-valley-tech-giant) | A Simple Decision to Try and Follow a Lifelong Dream (http://www.helloerik.com/a-simple-decision-to-try-and-follow-a-lifelong-dream) | Service Design, Business Silos, and Hot Air Balloons (http://www.helloerik.com/service-design-business-silos-and-hot-air-balloons) | Don't Let Your Customer Ecosystem be a Dirty Aquarium (http://www.helloerik.com/dont-let-your-customer-ecosystem-be-a-dirty-aquarium)

GET IN TOUCH

It's always a good time to contact me: erik@helloerik.com (mailto:erik@helloerik.com)

Connect on LinkedIn (http://www.linkedin.com/in/erikflowers)

@Erik_UX on Twitter (http://www.twitter.com/Erik_UX)

erikflowers on Github (https://github.com/erikflowers)

SEARCH

Search the site!

SHARE

Tweet (https://twitter.com/intent/tweet?url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial&text=Bootstrap 3 Less Workflow Tutorial via @Erik_UX&hashtags=UX,servicedesign)

Share (https://www.facebook.com/sharer/sharer.php?u=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial)

Share (https://plus.google.com/share?url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial)

Share (http://www.linkedin.com/shareArticle?mini=true&url=http://www.helloerik.com/bootstrap-3-less-workflow-tutorial&title=Bootstrap 3 Less Workflow Tutorial)