This is my first attempt in a while to create a fully customized WordPress theme without using Genesis or other theme frameworks. The idea is to use modern web technologies like CSS Grid, Variable Fonts and others to experiment with what it takes to add these technologies to a WordPress theme using Underscores as the starter theme.

It is also a playground for working with progressive enhancement, responsive design and how to accommodate multiple screen sizes in the same WordPress theme.

Some aspects that I think are important to highlight and also some areas where further work is needed.

Variable Fonts

The theme uses a single variable font, Recursive for everything from monospaced code examples to the casual font face used for the title header and everything in between, replacing 6 font files with a single 523KB WOFF2 one.

Yes, it's over 500KB of fonts, but we've taken care of not interrupting the page loading experience by using several tools and techniques:

- subsetting the font using [Glyphhanger](https://www.filamentgroup.com/lab/glyphhanger
- Using Font Face Observer to add classes based on font loading results
- Use font-display to swap the font in after it has finished loading.

To make everything work in WordPress we need to:

- Enqueue the font or load it a local CSS stylesheet
- Enqueue FontFace Observer
- wrap the inline script that uses Fontface Observer in a PHP function and then use `add_action` to add the script to the footer of all documents. The process is documented in Using Variable Fonts in a WordPress theme

The biggest disadvantage of variable fonts is that they require fairly recent browsers and operating systems to work so they must be treated as an enhancement and alternative fonts must be built into the stacks, preferably fonts that are close in size so that the text will not shift too much when the web font loads.

Another disadvantage is that, as I write this, Recursive is still in beta and there will be several more releases before it is deemed ready for production. I will continue to track the changes and will update the font when needed.

Adapting third-party libraries to use variable fonts

As documented in Modifying Prism.js to use a variable font I've tweaked the Prism.js CSS stylesheet to also use Recursive and its `MONO` axis

Yes, this ties me down to a specific version of Prism but, unless there are major changes in the Prism codebase or they add a new language that I must have, eliminating another potential font download is worth the effort.

Grid

One of the earliest decisions I made was to use CSS Grid for the layout and it works amazingly well.

Theme Structure

I have added the minimum necessary to make the theme work as designed. It usually involves adding styles, adding templates or modifying existing templates.

I've also used the WordPress Unit Test Data to validate that the content works as intended with as many types of content available on WordPress as possible.

To Gutenberg or not to Gutenberg

I've struggled with supporting Gutenberg on my theme or not until I realized that it doesn't matter what I choose, it's the people using the theme that get to choose whether to use it or not.

Header

The header uses the following items:

- Custom logo

- Dynamic title
- Navigation menu

The custom logo and menu are conditional. If the corresponding setting on the appearance admin menu is not selected, then it will not appear in the front end.

Content

Content is pretty much untouched from the original Underscores sources.

There are a few edge cases that I need to address like full-bleed images in a grid when they are not direct children of the grid element but they don't seem to impact the code appearance or the readability of the content itself.

Footer

The footer area uses two widget areas laid out using flexbox rows. Each area is also a flexbox laid using columns.

This may be more work than what's needed but it gives me the flexibility to add multiple widgets in whatever order I choose.

Javascript build system

I've adopted a build system I originally crafted for SASS-based workflows. It works but SASS itself is starting to not support the workflows I've been using for a while.

There are other libraries that I'm evaluating as potential replacements for SASS and they would require minimal effort as far as the build system is concerned.

See PostCSS deep dive

Javascript still runs through Babel but some code still depends on jQuery to work. jQuery is useful but it takes away a lot of the newer functionality available to Javascript.

So one thing I may evaluate going forward is whether the tradeoff between browser support and language features is worth it.

Some things still outstanding

Some things I'm thinking about and considering as I move forward completing the theme. Some of these things are nice to have while others are things I've never attempted before when working on a theme.

Full-bleed figures

Full bleed images, with or without captions, are one of the few things that don't work.

I'm researching what it would take to fix the issue.

jQuery or Javascript

Evaluate if the tradeoff between Javascript features and browser support is worth it, particularly in light that some features already require modern browsers and operating systems.

Plugging things to the customizer

Right now the theme and the customizer have little or no relationship. One of the next tasks is to figure out how to hook the theme into the customizer.

We do this to make sure that users don't need to tweak the HTML/PHP and CSS directly.

AMP compatibility

What, if anything, we need to do to make sure the site is AMP-ready?

Credits

- Underscores Starter Theme
- Building Themes from Scratch Using Underscores from Lynda.com / Linkedin Learning
- humescores from Morten Rand-Hendriksen
- Recursive variable font
- Codepen Work by Jason Pamental, especially FF Meta Variable Font Demo