Another look at my Gulp build system

Every so ofter I like to look at my build process and see where I can make improvements or changes that are necessary because the code has changed.

For this iteration the changes are:

- Finally switched to the current recommended method of defining tasks.
- Moved from Imagemin to gulp-libsquoosh
- Moved to markdown-it as the Markdown parser
- The syntax for gulp-exec changed so I moved the PDF generation code to the new syntax
- Another major and unexpected change is that node-sass is deprecated in favor of gulp-sass and the current dart-sass version
    - As a result the `sass` task has been modified to run with the new gulp-sass in synchronous mode
    - **dart-sass is now the normative SASS version and the first one that will receive new features, updates and fixes**
- Dependencies
    - Prince XML to generate PDFs from HTML
        - Because it's a commericla product I cannot share it. You can download it for evaluation purposes from the website
    - libsquoosh is used internally by gulp-libssquoosh to optimize images
    - markdown-it is used to parse Markdown files

We will cover each of the top level bullets in more detail below

Change to how we define tasks

The first change I made was how we define tasks. Gulp 3.x used something like this to define tasks:

```
gulp.task('markdown', () = {
  const config = {
    options: {
      preset: 'commonmark',
```

```
      'commonmark'      xhtmlOut: true,
      linkify: true,
      typographer: true,
    },
  };
  return gulp
    .src('src/md-content/*.md')
    .pipe(markdownPlugin(config))
    .pipe(gulp.dest('src/html-content/'));
});
'src/md-content/*.md'
```

In Gulp 4.x defining tasks like we did in the past is still supported but no longer recommended. The preferred way to define tasks is to create and export a function for each task. The same task now looks like this:

```
function markdown() {
  const config = {
    options: {
      preset: 'commonmark',
      html: true,
      xhtmlOut: true,
      linkify: true,
      typographer: true,
    },
  };
  return gulp
    .src('src/md-content/*.md')
    .pipe(markdownPlugin(config))
    .pipe(gulp.dest('src/'src/md-content/*.md'
exports.markdown = markdown;
```

One of the unexpected advantages of this method is that we can now create private utility functions that are used when composting tasks with `gulp.parallel` and `gulp.series` and only export those tasks we want to make available to the user.

Right now I expose all task in the `gulpfile` so I can run every stage of the

different workflows from Gulp, but that's not necessarily a best practice so we may change the tasks that are exported in the future.

Switch to `gulp-libsquoosh`

Imagemin is a great tool for optimizing images but, unfortunately, the Gulp plugin leaves a lot to be desired and the codec plugins need to be compiled for every platform you use them on.

LibSquoosh is the Javascript CLI for Squoosh, an image compression application. gulp-libsquoosh is a Gulp plugin for libSquoosh.

It has the advantages that supported compression formats are implemented as WASM libraries so they'll work everywhere without compilation. If you're doing image manipulation work, i might be possible to use them on projects outside of gulp-libsquoosh, but I haven't tried it yet.

The codecs are also bundled with libsquoosh so there's a single installation to make the full set of codices work.

Switch to `markdown-it`

I moved the Markdown task to use markdown-it as the Markdown parser.

```
function markdown() {
  const config = {
    options: {
      preset: 'commonmark',
      html: true,
      xhtmlOut: true,
      linkify: true,
      typographer: true,
    },
  };
  return gulp
    .src('src/md-content/*.md')
    .pipe(markdownPlugin(config))
    .pipe(gulp.dest('src/'src/md-content/*.md'
```

```
exports.markdown = markdown;
```

In theory, working with Markdown-it plugins should be easier but it's not working and rather than hold of the rest of the update I will remove the plugin code and add it once I've figured out how to make plugins work.

Gulp-exec changes

As of version 5 of gulp-exec, the plugin no longer use lodash templates. Upgrade from:

```
.pipe(exec('prince --verbose \
 --input=html \
 --javascript \
 --style ./src/css/article-style.css \
 <%= file.path %>', options))
```

to the new syntax using ES6 template literals (broken into multiple lines for readability, should all be in one line):

```
.pipe($.exec((file) => `prince --verbose
--input=html
--javascript
--style ./src/css/article-style.css
${file.path}`, options))
```

This was a hard one to troubleshoot since the plugin doesn't tell you what caused the error or what the specific error was. I happened to find the solution when looking at the issue tracker in the repository and the solution buried at the bottom of the README file.

Changes in gulp-sass

dart-sass is now the normative SASS version and the first one that will receive new features, updates and fixes. LibSASS and RubySASS have been deprecated and, as far as I know, work on them continues on a best-effort basis for security

patches only.

As a result the `sass` task has been modified to run with the new gulp-sass in synchronous mode that uses the sass running dart-sass as the SASS compiler.

Sincce I'm evaluating whether a solution based on PostCSS plugins would be better than SASS going forward, I just wanted to get SASS working so the current solution contrinues to work.

Code and docs

You can look at the JSDocs for the plugin at https://caraya.github.io/gulp-starter-2021/ and the build file itself on the Github repository. They will both continue to be updated as the project evolves until it's time to release a new version.

Comments and suggestions are always welcome on Twitter (@elrond25) or via issues in the repository.