Using material design for typography work

Material Design is Google's Design System. It started as a Google-only system that you bought into whole sale with little to noo chance of customizing it.

The new iteration of Material Design does several things that the original did not do well or at all, including framework integrations.

The "new Material" also allows for easier customization and provides SASS/SCSS mixins and functions to do so.

In this post we'll look at the typograaphy options available to Material Design, both standard and customized.

Setting up

There are two ways to setup a Material Design project. We'll talk about them below.

The easy way

The easy way loads the entire library of CSS functions and scripts from a CDN.

We also load Roboto and Material Design Icons from Google Fonts.

```
<link rel="stylesheet" href="https://unpkg.com/material-components-web@la
<!-- Fonts -->
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Robo

<script src="https://unpk<link rel="stylesheet" href="https://fonts.googl
```

This will load all of Material Design, whether we use it or not. We'll have to decide if this is worth the extra download sizes.

This method will not let you customize individual packages so, unless you're OK with the defaults this may not be the best option.

The hard way

The hard way uses WebPack and SASS. The typography module doesn't use Javascript so we'll skip it in this example.

> The first step is to get WebPack and plugins configured. The plugins are:

- webpack-dev-server: Development server
- sass-loader: Loads a Sass file and compiles it to CSS
- node-sass: Provides binding for Node.js to Sass, peer dependency to sass-loader
- css-loader: Resolves CSS @import and url() paths
- extract-loader: Extracts the CSS into a `.css` file
- file-loader: Serves the `.css` file as a public URL
- autoprefixer: Automatically adds prefixes to CSS as needed

The command too install the Node modules is:

```
npm install --save-dev webpack \
webpack-cli \
webpack-dev-server \
css-loader \
sass-loader \
node-sass \
extract-loader \
file-loader \
autoprefixer
```

We then write a `webpack.config.js` to process the SCSS file that we'll create later in the process.

```
const autoprefixer = require('autoprefixer');

module.exports = [{
  entry: './scss/app.scss',
    output: {
    './scss/app.scss'// style-bundle.js is necessary
    // for webpack to compile but never used
```

```
    filename: 'style-bundle.js'le: {
      rules: [{
        test: /\.scss$/,
        use: [
          {
            loader: 'file-loader',
  /\.scss$/'file-loader',
            options: {
              name: 'bundle.css',
            },
          },
          { loader: 'extract-loader' },
          { loader: 'css-loader' },
          { loader: 'sass-loader' },
        ]
      }]
    },
  },
];
```

We are then ready to install the specific modules that we want to work with. In this case, typography.

```
npm install @material/typography
```

Finally, we need to import the component's SCSS using the `@import` statement.

> We can then add any additional styles that we need.

```
@import "@material/typography/mdc-typography";
```

An Example

This example covers the basic layout of site

```
<body class="mdc-typography">
  <h1 class="mdc-typography--headline1">Towards the Splendid City</h1>

  <h2 class="mdc-typography--headline2">Pablo Neruda</h2>

  <h3 class="mdc-typography--headline3">Nobel Lecture, December 13, 1971</

  <p class="mdc-typography--body1">My speech is going to be a long journey
<body>
```

All material design components are built with this typograhy already baked in. For elements that are not part of material design we can use the following classes as I did in the example above.

Material Design Typography Classes

CSS Class	Description
mdc-typography	Sets the font to Roboto
mdc-typography--headline1	Sets font properties as Headline 1
mdc-typography--headline2	Sets font properties as Headline 2
mdc-typography--headline3	Sets font properties as Headline 3
mdc-typography--headline4	Sets font properties as Headline 4
mdc-typography--headline5	Sets font properties as Headline 5
mdc-typography--headline6	Sets font properties as Headline 6
mdc-typography--subtitle1	Sets font properties as Subtitle 1
mdc-typography--subtitle2	Sets font properties as Subtitle 2
mdc-typography--body1	Sets font properties as Body 1
mdc-typography--body2	Sets font properties as Body 2
mdc-typography--caption	Sets font properties as Caption
mdc-typography--button	Sets font properties as Button
mdc-typography--overline	Sets font properties as Overline

Customizing Material Design Typography

Material design typography also offers you SCSS mixins to customize how typography will work in the document.

Mixin	Description
`mdc-typography-base`	Sets the font to Roboto
`mdc-typography($style)`	Applies one of the typography styles, including setting the font to Roboto.
`mdc-typography-overflow-ellipsis`	Truncates overflow text to one line with an ellipsis. Only works for content using `display: block` or `display: inline-block`
`mdc-typography-baseline-top($distance)`	Sets the baseline height of a text element from top.
`mdc-typography-baseline-bottom($distance)`	Sets the distance from text baseline to bottom. This mixin should be combined with `mdc-typography-baseline-top` when setting baseline distance to following text element.

Possible values for `$style`

- headline1
- headline2
- headline3
- headline4
- headline5
- headline6
- subtitle1
- subtitle2
- body1
- body2
- caption
- button
- overline

This gives us one level of custmization but it won't change the defaults, it will only use the defaults for the elements we're working with.

If you're familiar with SASS you can use

All styles can be overridden using Sass global variaables with the format `$mdc-typography-styles-{style}` before the component is imported.

The variable should contain a map that with all the properties we want to override for a particular style.

Assuming that the fonts are available on the system or loaded using `@font-face` we can use code like this to change the global font defaults for the document.

```
$mdc-typography-font-family: unquote("Arial, Helvetica, sans-serif");

// Imports and the rest of our code goes here
```

unquote is part of the SASS standard library.

We can also change the value of multiple elements in the same page. Assuming, again, that Marvin Vision was available on your system we can customize styles for both `headline1` and `headline2`.

We can do the same thing with any style available.

```
$mdc-typography-styles-headline1: (
  font-family: unquote("MarvinVision, Helvetica, sans-serif")
);
$mdc-typography-styles-headline2: (
  font-family: unquote("Arial, Helvetica, sans-serif"),
  font-size: 3.25rem
);

"Arial, Helvetica, sans-serif"// Imports and the rest of our code goes her
```

So far we've concentrated on how to use the built-in styles from Material Design. There is nothing to prevent us from mix and matching material design typography

with other design element or art directing this kind of mixed applications.

A full example of Material Design typography is in this Github repo

Another example is how Una Kravetz created a Material Design theme using variable fonts from Google Fonts

It'll be interesting to see what we can do with Material Design moving forward.

References

- Material Design: Getting Started
- Material Design Components: Typography
- material-starter-kit-variable-fonts