Material Design: What's New

I first looked at Material Design when it was first released along with the Polymer Team's paper elements (the reference implementation of Material Design). Back then it wasn't themable so most applications and sites that used Material Design looked the same.

I was surprised to see a lot of sessions about Material Design at Google I/O this year. It picked my attention; I wanted to see what, if anything, changed.

What is Material Design?

Material Design is many things, among them:

- A design system
- A set of design guidelines
- A set of components to use in our own projects
- A customizer to make the components and design process better fit brands other than Google

The idea is to provide a common framework to talk and exectute design projects.

Getting everything ready

The new Material Design (that I call Material Design 2.0) uses NPM to host the components. So we need to run the following command to generate a `package.json` NPM configuration file.

```
npm init -y
```

You can later edit the file to change the information. I generate the file automatically to make life easier (what can I say, I'm lazy).

SCSS

It uses SCSS to style the document so we need to make sure that Node-SASS is available.

```
npm i -D node-sass
```

Javascript and Typescript

MDC works with ES2015 and Webpack on the Javascript side so we need to install Babel, ESLint and Webpack related packages.

Eventually I want to move to Typescript to work with this project to unify all the scripts I write into one language. That's why I add both the ES2015 and Typescript presets; so I can work in either language to start and then migrate to Typescript without having to change my Webpack configuration, all the work has been done up front.

I also use ESLint to lint my ES2016 code. The nice thing is that we can use ESLint for both ES2015 an Typescript. Just to be on the safe side I also install.

In the process of implementing this project I found out that you have to install an additional package if you're working with Typescript. `@types/npm` provides type definitions for NPM making a series of errors disappear when linting.

```
# Load Babel Core and the ES2015 and Typescript presets
# Save them as dev dependencies
npm i -D @babel/core @babel/preset-es2015 @babel/preset-typescript
# Install @types/npm to make sure NPM and Typescript
# play well together
# Save it as dev dependency
npm i -D @types/npm
# Load Webpack and loaders
# Save them as dev dependencies
npm i -D webpack webpack-dev-server babel-loader \
css-loader extract-loader file-loader sass-loader
# Load ESLint and Google's eslint config
# Save them as dev dependencies
```

```
npm i -D eslint eslint-config-google
# (OPTIONAL) Load Typescript and TSLint
# Save it as dev dependency
# npm i -D typescript tslint
```

The other part of configuring the application is to create scripts that will run from the command line. This is different than configuring a task runner to run specific tasks (something we may end up doing anyways). Some of the tasks we'll run from NPM include:

- Initializing the project's Typescript configuration
- Start Webpack's development server
- Lint Javascript files
- Lint Typescript files

We may add others as we walk through the process.

```
{
  "name": "mdc-demo""mdc-demo""version": "1.0.0",
  "description"t implementation of an application using Express for the ba
   and material design components for the UI",
  "main": "ind": "s",
  "scripts":"scripts""scripts": {
    "init".json || (tsc --init -t ESNext -m ESNext)",
    "start": "webpack-d": ""start"-progress --open",
    "lint:js": "eslint ./sr": ""lint:js",
    "lint:ts": "tslint ./src/ts": ""lint:ts",

  "keywords": [],
  "author": "Ca": ""keywords": [],
  "author".com>",
  "license": "MIT",
  "dependencies":": ""license": "MIT",
  "dependencies"s/npm": "^2.0.29",
    "b": "-core": "^6.22.1",
 ": "babel-loader": "^": ""babel-loader""babel-core"15": "^6.9.0",
  ": ""babel-loader": "^7.0.0",
    "babel-preset-es2015" "eslint-config-google": "^0.9.1",
```

```
    "extra": ""eslint-config-google""eslint"ader": "^1.1.11",": ""eslint-
    "extract-loader"   "webpack": "^": ""file-loader"ack-dev-server": "^2
  ": ": "^4.9.0",
    "sass-loader": "^6.0.4",
    "webpack": "^3.0.0",
    "webpack-dev-server": "^2.4.3"
  }
}
```

ESLint Configuration

To make sure ESLint work as intended run `eslint --init` to create a blank configuration file. Give the following answers when prompted

- How would you like to configure ESLint? **Use a popular style guide**
- Which style guide do you want to follow? **Google**
- What format do you want your config file to be in? **JavaScript**

The `--init` option will create an

```
module.exports = {
  "extends": "google",

  "parserOptions": {
    "ecmaVersion": 6,
    "sourceType": "script",
  },
  "env": {
    "browser": true,
    "node": true,
    "es6": true
  }

};
```

Webpack Configuration

Since we're using Webpack we need a Webpack configuration file to tell Webpack how to work with our content. I've set up entry points for each of the types of content I expect to work with: SCSS, and Javascript.

I'll delay working with Typescript until I can figure out issues with Webpack Dev Server.

```
// const path = require('path');

module.exports = [
  {
    entry: './src/sass/app.scss',
    outpu'./src/sass/app.scss'// This is necessary for webpack to compile
      // But we never use style-bundle.jsename: './src/css/app.js',
    },
    module: {
      rules: [{
        test: /\.scss$/,
        use: [
          {
            loader: 'file-loader',
            options: {
              name: './dist/css/bundle./\.scss$/'./src/css/app.js'},
          },
          {loader: 'extract-loader'},
          {loader: 'css-loader'},
          {
            loader: 'sass-loader',
            options: {
              includePaths: ['./node_modules'],
            },
          },
        ],
      }],
    },
  },
```

```
  {
    entry: './src/js/app.js',
    output: {
      filename: './dist/js/bundle.js'    module: {
      loaders: [{
        test: /\.js$/,
        loader: 'babel-lo/\.js$/'babel-loader',
        query: {
          presets: ['es2015'],
        },
      }],
    },
  }
];
```

Typescript Configuration

I've also configured Typescript for future work. It appears that Typescript doesn't work with the Webpack Dev Server. Until I can figure out what the issue is I can't use Typescript directly in the project but I can transpile the code manually and then run it through the Webpack pipeline. Nevertheless, here's the configuration I'm using.

```
{
  "compilerOptions": {
    // Specify ECMAScript target version: 'ES3'
    // (default), 'ES5', 'ES2015', 'ES2016',
    // 'ES2017','ES2018' or 'ESNEXT'.
    "target": "es2015",
    //  Specify module code generation
    "es2015""module": "es2015",
    // Allow javascript files to be compiled.
    "allowJs": true,
    // Report errors in .js files.
    "checkJs"
    // Specify JSX code generation
```

```
    "jsx": "": ""jsx": "react",
    // Generates corresponding '.map' file.
    "sourceMap"// Redirect output structure to the directory.
    "outDir": ".": ""outDir": "./dist/",
    //Enable all strict type-checking options.
    "strict": true,
    // Allow default imports from modules with no
    // default export. This does not affect code
    // emit, just typechecking.
    "allowSyntheticDefaultImports": true,
    // Enables emit interoperability between CommonJS
    // and ES Modules. Implies 'allowSyntheticDefaultImports'
    "esModuleInterop": true
  }
}
```

Editorconfig

To save myself the hassle I use the default editorconfig configuration to make sure that I don't have to worry about tabs/spaces, line ending and other issues that come when working in different operating systems and with programmers with different habits.

```
root = true

[*]
indent_style = space
indent_size = 2
end_of_line = lf
charset = utf-8
trim_trailing_whitespace = true
insert_final_newline = true

[*.md]
trim_trailing_whitespace = false
```

So after all this configuration script and files, we're finally ready to look at how to

create Material Design coomponents.

How does it work?

Material Design uses SASS for styling and ES2016 (Typescript for us) alongside HTML markup that tells the browser what to render and how to style it.

Building elements

For this example we'll build a card element, like something I would use to display content like short tidbits of information, photos or similar content.

The process is fairly straightforward. We first install the module using NPM:

```
npm install @material/card
```

Note that all Material Design Components are scoped under the `@material` owner. This way you know you're installing the correct component when scoping it like I did in the example.

Once it's installed we must do two things:

1. Write Markup
2. Write the SASS styles for the component
3. If we need to, write JavaScript to add behaviors to the component

The Markup: HTML

The card markup can be as simple as a plain container with nothing else inside but text:

```
<div class="mdc-card">
  <h1>Title</h1>

  <p>The content goes here</p>
</div>
```

To fairly complex cards with multimedia text and navigation items at the bottom.

```
<div class="mdc-card">
  <div class="mdc-card__media mdc-card__media--square">
    <div class="mdc-card__media-content">Title</div>
  </div>

  <p>The content goes here</p>

  <div class="mdc-card__actions">
    <div class="mdc-card__action-buttons">
      <button class="mdc-button mdc-card__action mdc-card__action--button"
      <button class="mdc-button mdc-card__action mdc-card__action--button"
    </div>
    <div class="mdc-card__action-icons">
      <i class="material-icons mdc-card__action mdc-card__action--icon" ta
      <i class="material-icons mdc-card__action mdc-card__action--icon" ta
    </div>
  </div>
</div>
```

When creating a card there are many CSS classes that will help in structuring the card and, later, give you the names to use when adding styles.

Styling the component: SASS/SCSS

When I first saw how many classes you had available I got scared. There seems to be enough to drive yourself crazy if you haven't read the docs and look at what you're options are. There are also some gotchas that I would not have figure out if I hadn't gone back to re-read the docs again.

All the classes in the table (taken from mdc-card README on the Github Repo) are provided by the component. Anything else is up to you to create.

CSS Class	Description
`mdc-card`	Mandatory, for the card element
`mdc-card--outlined`	Removes the shadow and displays a hairline outline instead
`mdc-`	The main tappable area of the card. Typically contains most (or

CSS Class	Description
`card__primary-action`	all) card content *except* `mdc-card__actions`. Only applicable to cards that have a primary action that the main surface should trigger.
`mdc-card__media`	Media area that displays a custom `background-image` with `background-size: cover`
`mdc-card__media--square`	Automatically scales the media area's height to equal its width
`mdc-card__media--16-9`	Automatically scales the media area's height according to its width, maintaining a 16:9 aspect ratio
`mdc-card__media-content`	An absolutely-positioned box the same size as the media area, for displaying a title or icon on top of the `background-image`
`mdc-card__actions`	Row containing action buttons and/or icons
`mdc-card__actions--full-bleed`	Removes the action area's padding and causes its only child (an `mdc-card__action` element) to consume 100% of the action area's width
`mdc-card__action-buttons`	A group of action buttons, displayed on the left side of the card (in LTR), adjacent to `mdc-card__action-icons`
`mdc-card__action-icons`	A group of supplemental action icons, displayed on the right side of the card (in LTR), adjacent to `__action-buttons`
`mdc-card__action`	An individual action button or icon
`mdc-card__action--button`	An action button with text
`mdc-card__action--icon`	An action icon with no text

Now that we've got the markup ready to go, we can look at the SCSS/SASS we need to write to make it look the way we want it to.

Build a site with MDC

Firebase for Static Site Hosting

Links and Resources

- Material Design
- Material Component Github Repo
- Material Components Catalog