May 2019

J Dickinson

F1 Architectural Summary

Single Page Client-Side JavaScript Web Application

The primary function of the SPA is to consume the Ergast API for the data to be displayed on desktop and mobile devices through a modern user interface.

Project setup

```
Vue CLI 3.7.0
```

```
$ vue create vue-f1-app
```

```
src/

   ├──── assets/
   ├──── components/
   ├──── css/
   ├──── views/
   |
   ├──────── App.vue
   └──────── main.js
   └──────── router.js
   └──────── server.js
```

```
main.js
```

This is the main JavaScript entry point of the application. Vue library and the App component are imported from App.vue to create a Vue instance using the assigned DOM element #app

```
server.js
```

I added `server.js` with the intention of externalizing API calls.
A baseURL is defined here and reused within requests made by the HTTP client Axios

Global and Scoped presentation

Interface Theories for Component based Design is an academic article that I used to establish a conceptual framework from which to author a custom naming pattern. The article defines components in two ways:

Component Description
A component is defined in isolation.
This definition must answer the question: what does it do?

Component Interface Description
A component is defined in relation to the environment.
This definition must answer the question: how can it be used?

I applied the same naming convention within the context of CSS to author the custom naming pattern: https://cssreactions.com

The Naming Pattern Continuum

Content Dependant		Content Independant

`.icon-block {`
`  position: relative;`
`}`
	`<div class=”icon-block pad-all-5”>`	`.pad-all-5 {`
		`  padding: 5px;`
		`}`

CSS Reactions is added globally and component specific classes have been authored separately and added locally.

Webpack

Vue CLI 3 automatically integrates Webpack.

I am able to manipulate the configuration from `vue.config.js` in the following way:

```
module.exports = {
  chainWebpack: config => {
    config;
    if (process.env.NODE_ENV === "production") {
      ("./");
    } else {
      // mutate for development...
    }
  }
};
```

In order to preview the production build in the dist folder:

```
// for production...
publicPath: './'

With lint version:
("./");
```

This corrects the file paths.

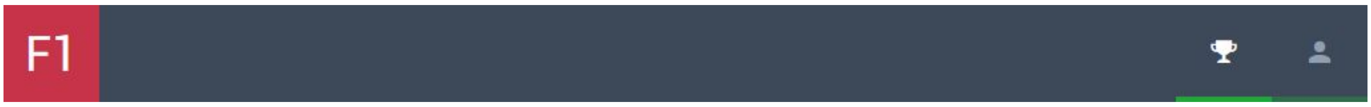
Single page components

Reusable code and the separation of concerns

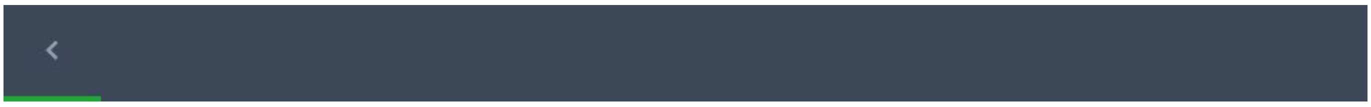
In order to support the development of reusable code, I created the following rules:

1. Views can contain components
2. Components are reusable and independant

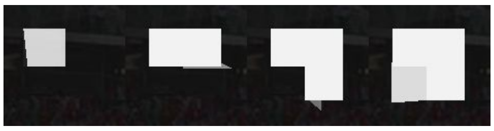
`<Navigation />`



`<NavigateBack />`



`<Spinner />`



Communication between components

In order for components to share data, I setup the following rules:

“while attempting to keep within the context of the first two rules”

3. Two types of components can exist:
    a. Parent components
    b. Child components
4. Parent components can contain child components

Child-to-Parent Communication

`ContentRow` is reusable as a child component to its `ParentComponents:`

```
<View>
  <ParentComponent>
    <ChildComponent/>
  </ParentComponent>
</View>
```

```
<DriverList>
  <AlonsoContentRow>
    <ContentRow />
  </AlonsoContentRow>
</DriverList>
```

Causes of component dependency

I felt the need to add more content to the web application by introducing static content. I ended up mixing the static content with dynamic content. This caused me to create seperate views and single use components, each requiring a slightly different set of data. If I re-developed the SPA, I would only use data from the API in order to improve the level of component independence.

Vulnerabilities

I think it's important to establish a secure and stable development environment from the beginning.

Currently, the latest Vue CLI scaffolding does not seem to be free of vulnerabilities after a default base install. I was able to identify two areas that exposed the application to risk and excluded them from my initial installation:

@vue/cli-plugin-unit-jest: ^3.7.0
Preprocessor SCSS

```
$ npm install -g @vue/cli
$ vue --version
3.7.0
$ vue create vue-f1-app
$ cd vue-f1-app
$ npm install
$ npm audit

found 65 vulnerabilities (64 low, 1 high) in 42992 scanned packages
1 vulnerability requires semver-major dependency updates.
64 vulnerabilities require manual review. See the full report for details.
```

I manually installed each package of the Vue-CLI setup and discovered that @vue/cli-plugin-unit-jest introduces 63 vulnerabilities. In addition to this, adding preprocessor SCSS capability introduced a major vulnerability that I couldn't seem to shake off.

Workarounds

I was able to identify the textContent of an element, however this would differ according to the season selected on the previous view. I highlighted the season winner using CSS as a workaround.

```
class="{Highlighted: tableResult.Driver.givenName.indexOf('Fernando') > -1}"
```