AlarmBuddy Style Guide

Team Members:
Joshua Ringling: ring7885@stthomas.edu
Hannah Balut: balu2673@stthomas.edu
Alex Notch: notc6377@stthomas.edu
Mitzi Bustamante Chicaiza: bust6340@stthomas.edu

Background:

This style guide is to be used as a reference for all work being done related to the AlarmBuddy project. The attached guides were heavily inspired by style guidelines provided by Google under the CC-By 3.0 License, which encourages the sharing of these and the original source documents. The original style guides provided by Google can be found at https://google.github.io/styleguide/

Documentation Guidelines:

- Any services requiring accounts being used should have detailed instructions on how to create and login to an account.
- Any software or tools used in production such as IDEs should have detailed instructions on downloading and installing the software such that settings are the same used while making the project.

General Rules:

- Variables start with lower case letters and use context appropriate names. Use camel-case for variables that use multiple words.

```
int licenseNumber = 45301;
```

- Indent using tab to properly show scope. Examples: if statements, for loops, while loops, and functions.
- Use curly braces on all if-else statements, loops, and functions even when they are empty or only contain one line.
- Curly braces follow the Kernighan and Ritchie style (or “Egyptian brackets”) for code blocks:
    - No line break before the opening brace.
    - Line break after the opening brace.
    - Line break before the closing brace.
    - Line break after the closing brace *only if* that brace terminates the current body of a method, constructor, or named class. For example, there is no line break after a brace if an else is being used.

```
if (x == 3) {
    for (int i = 0; i < 20; i++) {
        x = x + 1;
    }
} else {
    try {
        something();
    } catch (ProblemException e) {
        recover();
    }
}
```

- In mathematical operations, have spaces between all operators and operands *except* for parenthesis and use parenthesis to show clear mathematical steps.

```
int x = 2 * (1 + 1) ^ 2;
```

- Comment the use of functions as well as any special exceptions that may happen.
- Avoid making comments longer than 120 characters long. Try to wrap excessively long sections onto the next line.

```
//Returns an array of the same length as the input array
//but the output array has each data value square-rooted.
//If the input array has negative numbers in it, then an
//error is thrown.
```

- Try to initialize variables when declaring them. However, this is not always possible.

```
int x = 0;
String school = "St. Thomas";
```

HTML/CSS Style Guide

Joshua Ringling: ring7885@stthomas.edu

Background:

This document defines the formatting and style rules for React. It applies to raw, working files that use React coding.

General Rules:

Basics:

Only include one React component per file.

Always use JSX syntax. While using JSX syntax you are expected to follow the style guide instructions for HTML.

Class vs. React.createClass:

If you have any references to internal state, extending React.Component is a lot cleaner.

```
// Not recommended
const Listing = React.createClass({
    // ...
    render() {
      return <div>{this.state.hello}</div>;
    }
});

// Recommended
class Listing extends React.Component {
    // ...
    render() {
      return <div>{this.state.hello}</div>;
    }
}
```

Mixins:

Do not use mixins. Just don't use them.

Mixins introduce implicit dependencies and can cause clashing naming schemes.

Naming:

Extensions: Always use the .jsx extension for React components.

Filename: Use PascalCase for filenames.

References: Use PascalCase for React components and camelCase for their instances.

PascalCase is just like camelCase, but the first word is capitalized.

Declaration:

Do not use *displayName* for naming components.

Quotes:

Always use double quotes (“) for JSX attributes.

Use single quotes (‘) for all other JavaScript uses.

Spacing:

Always include a single space in your self-closing tag.

```
// Not recommended
<Foo/>

// Not recommended
<Foo                    />

// Not recommended
<Foo
/>

// Recommended
<Foo />
```

Props:

Always use camelCase for prop names and PascalCase if the prop value is a React component.

Always include an alt prop for <img> tags.

Do not use words like “image”, “photo”, or “picture” in <img> or alt props.

Do not use accessKey.

Parentheses:

If JSX tags span more than one line, wrap them in parentheses.

```
// Not recommended
render() {
    return <MyComponent variant="long body" foo="bar">
             <MyChild />
           </MyComponent>;
}

// Recommended
render() {
    return (
      <MyComponent variant="long body" foo="bar">
        <MyChild />
      </MyComponent>
    );
}
```

Tags:

Always self-close tags that have no body.

If your component has multiline properties, close its tag on a new line.

```
// Not recommended
<Foo
  bar="bar"
  baz="baz" />

// Recommended
<Foo
  bar="bar"
  baz="baz"
/>
```

Methods:

Arrow functions are allowed as long as they don't massively hinder performance like causing multiple pointless rerenders.

Bind event handlers for the render method in the constructor.

isMounted:

Do not use isMounted.

isMounted is not available when using ES6 classes and is soon to be officially deprecated.