The Problem with ../ ../ ../ imports

- you would need to update all of your imports in your project and add one extra ../ to all of your imports.
- Pretty hard to refactor
- It becomes worse as you get further deeper into it.
- You need to change the entire codebase if you need to extract the code to be used externally as an NPM module

Benefits of Absolute Import

1. You can move your existing code to other components with imports without any changes.
2. You can directly copy/paste code from a component and use it anywhere else without making any changes to the imports.
3. Using the import path, you can quickly identify where the component is actually placed.
4. Cleaner Code.
5. Easier to write.

Difference

```
import React from "react";
import Button from "../../Button/Button";
import { LINKS, STRINGS } from "../../../utils/constants";
import styles from "./App.module.css";

function App() {
  return (
    <div className={styles.App}>
      <Button>{STRINGS.HELLO}</Button>

      <a href={LINKS.HELP}>Learn more</a>
    </div>
  );
}

export default App;
```

../ ../ ../ import

```
import React from "react";
import { LINKS, STRINGS } from "utils/constants";
import Button from "components/Button/Button";
import styles from "./App.module.css";

function App() {
  return (
    <div className={styles.App}>
      <Button>{STRINGS.HELLO}</Button>

      <a href={LINKS.HELP}>Learn more</a>
    </div>
  );
}

export default App;
```

Absolute import

Usage in TS & JS

```
// Using TypeScript
// If you need to set up absolute imports in your
TypeScript application add/update your tsconfig.json file
in the root directory of the project. Then you need to
update the compiler option baseUrl in the file.

{
  "compilerOptions": {
    "baseUrl": "src"
  },
  "include": ["src"]
}
```

```
// Using JavaScript
// Create the jsconfig.json file in the root directory of
the project. Then you need to update the following
snippet.

{
  "compilerOptions": {
    "baseUrl": "src"
  },
  "include": ["src"]
}
```