DARK MODE WITH REACT

Using data-* attributes to coordinate dark mode and phone responsive changes

1

CREATE A STATE VARIABLE

Create a state variable that responds to a toggle button

Home.jsx

```jsx
const [darkMode, setDarkMode] = useState(false);


return (...
    <button onClick{() => setDarkMode(!darkMode)}>
        Toggle Dark Mode
    </button>
```

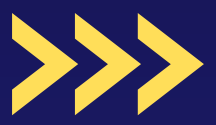
2

USE DATA-* ATTRIBUTE

Create a page wrapper with a data-* attribute whose value toggles based on the state variable (you can even wrap all components in the App file to minimize code repetition).

Home.jsx

```
return (...
<div
    className=“page-container”
    data-darkMode={darkMode ? “true” : “false”}
>
    <div className=“hero-section”> .... </div>
    <div className=“about-section”> .... </div>
    <div className=“project-section”> .... </div>
    ....
</div>
```



3

CREATE CSS STYLINGS

Use descendant combinator syntax to style descendant elements

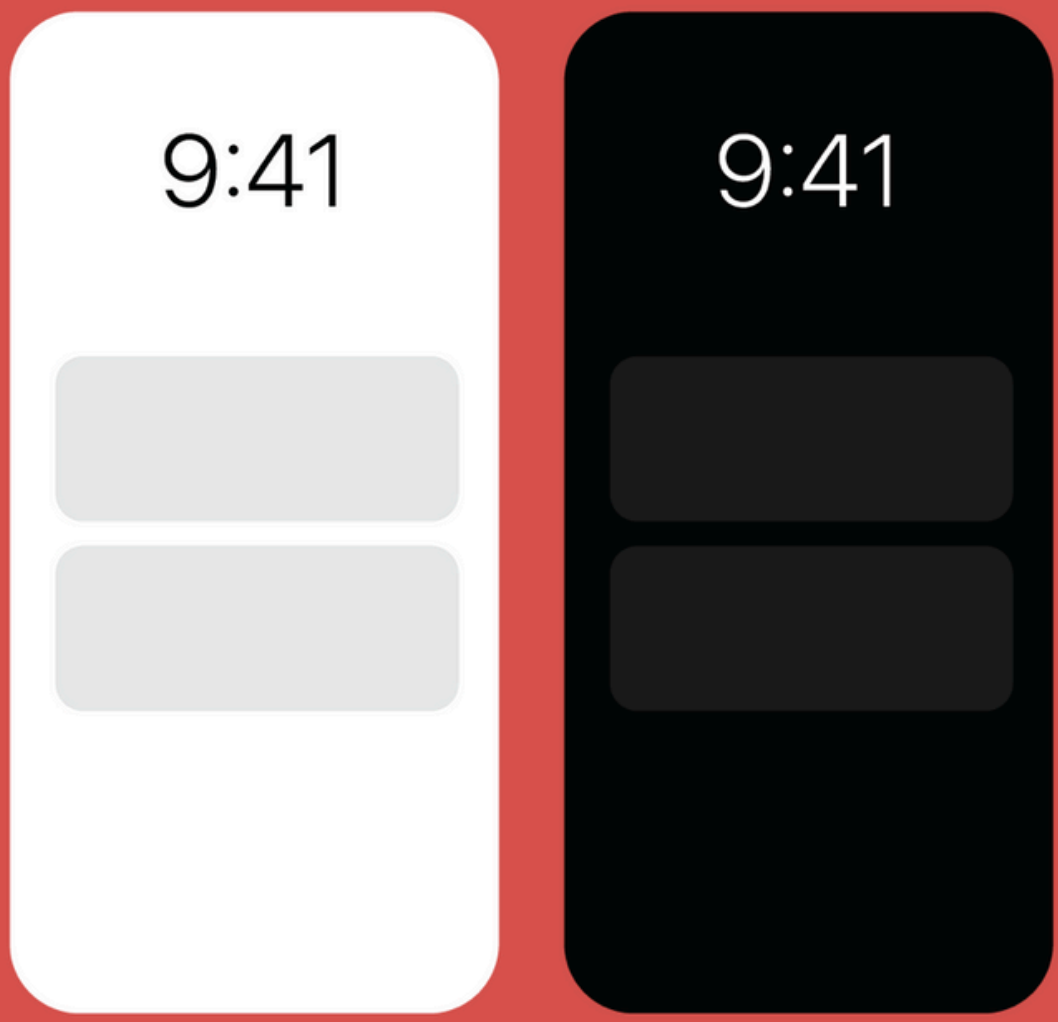
home.css

```
.page-container[data-darkMode=“true”] {
    background-color: #202024;
}
[data-darkMode=“true”] p {
    color: white;
}
[data-darkMode=“true”] button.btn-primary {
    background-color: #57FFCD;
}
```

4

ADMIRE YOUR WORK

[data-darkMode="false"] [data-darkMode="true"]



FOR PHONE RESPONSIVENESS

The “true” and “false” state values for the state variable and data-* attriubute only allow for binary stylings. For responisve changes, using numerical values i.e. 360, 540, 720, 900 etc. to represent different screen widths is more flexible. This might be useful if there is another state variable detecting mobile device use. Otherwise media queries might be the way to go!

```
const handleResize = () => {
    if (mobileAgent) {
        if (document.body.clientWidth < 540) {
            setMobileMode(540);
        } else if (document.body.clientWidth < 720) {
            setMobileMode(720);
        }
    } else {
        setMobileMode(null);
    }
```

Thanks for reading!