Shimmer Loader

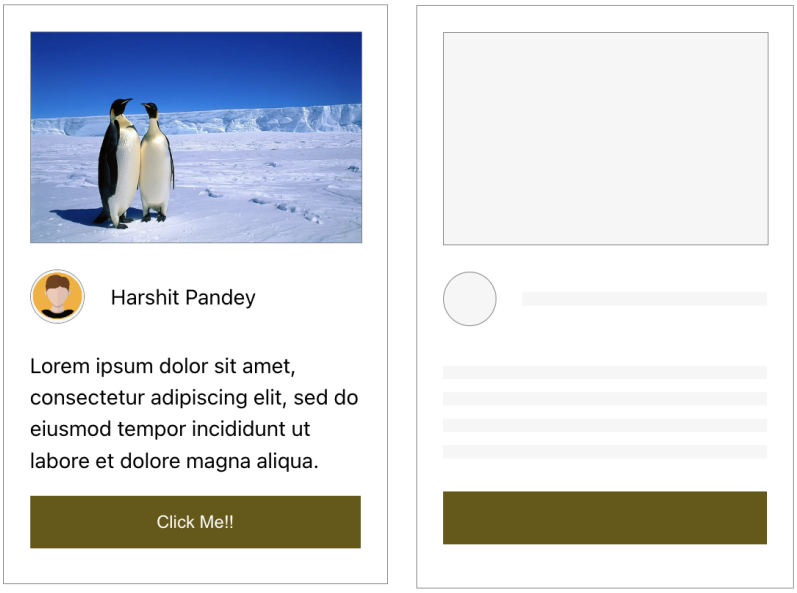
Requirements

1. Component should be re-useable in different screen sizes.
   - Desktop and Mobile screens
2. Component should be configurable color, animation speed, delay.
3. Component should have a toggle between loading state and Actual component.
4. Component should have support for different kind of loaders. Text loader(Can be multi line), Image loader(Block).
5. Component should allow us to pass HTML interface(skeleton).

Assumptions

1. Final output is predictable
   Suppose in the loader we show an image and text, but actual content contains only text. In such cases, loader looks very out of place.
2. Different types of loaders
   Text Loader(can be multi-line) and Image Loader(Block loader)
3. No progressive loading inside component. All items inside the component loads at once.
   This can be achieved in future.
4. No user interactions on loader. Component doesn't expect callbacks for events like "click".
   This can be achieved in future.

Architecture



<ShimmerLoader> expects a type of loader or it expects a skeleton of the interface.

```
<ShimmerLoader>
   {/* if loading == false*/}
   {props.children}
   {/* if skeleton */}
   {skeleton}
   {/* else */}
   <ShimmerBlock></ShimmerBlock> {/* type is "block"  */}
   <ShimmerText></ShimmerText> {/* type is "text"  */}
 </ShimmerLoader>
```

Components -

<ShimmerLoader> - Main component responsible for loading experience. Accepts the props to configure the component.

<ShimmerBlock> - Block level loader with width = 100% and height = 100%

<ShimmerText> - Text loader with width = 100% (configurable) and height = 20px (default), can be one line or multi-line.

Skeleton - HTML Interface(React component) which resembles the component.

APIs

loading	Boolean (default = true) Thought: loading can be a promise.
type	String (default = “text”) “text” | “block”
loaderStyles	Object with camelCased css properties To override default shimmer styles.
className	Inject class names to shimmer
speed	Animation speed
width	
height	
colors	*Not implemented as of now*