ELLIPSIS LOADER

import React, { useState, useEffect } from "react";

import { StaticElementComponent } from "../components/StaticElementComponent";

import googleWritingSvg from "../images/google-writing-svg.svg";

import styled, { keyframes } from "styled-components";

export const StaticElementContainer = ({ children }) => {

const [isImageLoaded, setIsImageLoaded] = useState(false);

useEffect(() => {

const image = new Image();

image.src = googleWritingSvg;

image.onload = () => {

setIsImageLoaded(true);

};

}, []);

const LoaderContainer = styled.div`

display: flex;

justify-content: center;

align-items: center;

height: 65vh;

`;

const ellipsisAnimation = keyframes`

0% {

transform: scale(0);

}

25% {

transform: scale(1);

}

50% {

transform: scale(0);

}

100% {

transform: scale(0);

}

`;

const Ellipsis = styled.div`

display: flex;

justify-content: center;

align-items: center;

position: relative;

width: 64px;

height: 64px;

div {

position: absolute;

top: 27px;

width: 11px;

height: 11px;

border-radius: 50%;

background-color: rgb(219, 219, 219);

animation-timing-function: cubic-bezier(0, 1, 1, 0);

}

div:nth-child(1) {

left: 6px;

animation: ${ellipsisAnimation} 0.6s infinite;

}

div:nth-child(2) {

left: 6px;

animation: ${ellipsisAnimation} 0.6s infinite;

animation-delay: -0.2s;

}

div:nth-child(3) {

left: 26px;

animation: ${ellipsisAnimation} 0.6s infinite;

animation-delay: -0.4s;

}

div:nth-child(4) {

left: 45px;

animation: ${ellipsisAnimation} 0.6s infinite;

animation-delay: -0.6s;

}

`;

const LoadingSpinner = () => {

return (

<LoaderContainer>

<Ellipsis>

<div></div>

<div></div>

<div></div>

<div></div>

</Ellipsis>

</LoaderContainer>

);

};

if (!isImageLoaded) {

// Render loading state or placeholder if the image is not loaded yet

return <div><LoadingSpinner /></div>;

}

// Render the StaticElementComponent with the children once the image is loaded

return (

<StaticElementComponent>

{children}

</StaticElementComponent>

);

};

---------------------------------------------------  
  
SNAKE LOADER

import React, { useState, useEffect } from "react";

import { StaticElementComponent } from "../components/StaticElementComponent";

import googleWritingSvg from "../images/google-writing-svg.svg";

import styled, { keyframes } from "styled-components";

export const StaticElementContainer = ({ children }) => {

const [isImageLoaded, setIsImageLoaded] = useState(false);

useEffect(() => {

const image = new Image();

image.src = googleWritingSvg;

image.onload = () => {

setIsImageLoaded(true);

};

}, []);

const LoaderContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 65vh;

position: relative;

`;

const snakeAnimation = keyframes`

0%, 100% {

transform: scaleX(0);

transform-origin: right;

animation-timing-function: ease-in-out;

animation-delay: 0s;

}

50% {

transform: scaleX(1);

transform-origin: left;

animation-timing-function: ease-in-out;

animation-delay: 0s;

}

60% {

transform: scaleX(1);

transform-origin: left;

animation-timing-function: ease-out;

animation-delay: 0.2s;

}

70% {

transform: scaleX(1);

transform-origin: left;

animation-timing-function: ease-in;

animation-delay: 0.2s;

}

80% {

transform: scaleX(1);

transform-origin: left;

animation-timing-function: ease-out;

animation-delay: 0.2s;

}

90% {

transform: scaleX(1);

transform-origin: left;

animation-timing-function: ease-in-out;

animation-delay: 0.2s;

}

`;

const SnakeLoader = styled.div`

position: absolute;

top: 0;

left: 0;

height: 4px; /\* Adjust the height of the loader bar \*/

width: 100%;

background-color: #2196F3; /\* Blue color \*/

animation: ${snakeAnimation} 3s linear infinite;

`;

const LoadingSpinner = () => {

return (

<LoaderContainer>

<SnakeLoader />

</LoaderContainer>

);

};

if (!isImageLoaded) {

// Render loading state or placeholder if the image is not loaded yet

return <div><LoadingSpinner /></div>;

}

// Render the StaticElementComponent with the children once the image is loaded

return (

<StaticElementComponent>

{children}

</StaticElementComponent>

);

};