SVG Logo for index.html file

Below is the SVG code used to create the My Company logo. Read each line and fill in the blanks using the hints provided. This exercise will help you understand how SVG works with text and how Tailwind CSS can style SVGs.

```
<!-- _____ -->
```

Hint: Add a comment describing the purpose of this code

```
<svg class="_____ _____ _____ _____"
```

Hint: Tailwind classes for size and color

```
  xmlns="_____"
```

Hint: Declares this is SVG code

```
  viewBox="0 0 ___ ___"
```

Hint: Internal width and height of the SVG space

```
  aria-label="_____">
```

Hint: Used for screen readers to describe this element

```
 <text x="__" y="__"
```

Hint: Position where the text begins

```
    font-family="_____"
```

Hint: Preferred font with a backup option

```
    font-weight="___" font-size="__">
```

Hint: Font thickness and size

```
 _____
```

Hint: This is the actual logo text

</text>

</svg>

Keyframes Explanation for Cat SVG

@keyframes buttonClick

- • Used to create a press effect on the button when clicked.
- • 0%: Start at normal size.
- • 50%: Scale down slightly to 90%.
- • 100%: Return to original size.

@keyframes moveEyes

- • Animates the cat's eyes to simulate looking around.
- • 20%: Shift left and up.
- • 40%: Return to center.
- • 65%: Shift right and up.
- • 100%: Return to horizontal center, slightly lower.

@keyframes leftWhiskers

- • Makes the left whiskers swing slightly.
- • 20%: Rotate clockwise 5 degrees.
- • 60%: Rotate counterclockwise 5 degrees.

@keyframes rightWhiskers

- • Makes the right whiskers swing in a different rhythm.
- • 5%: Slight counterclockwise tilt.
- • 90%: Rotate clockwise 7 degrees.

@keyframes move

- • Adds a full-body tilt animation to the cat.
- • 20%: Slight tilt forward.
- • 40%: More pronounced tilt forward.
- • 60%: Tilt backward.
- • 85%: Stronger backward tilt.

Keyframes Explanation for Cat

@keyframes buttonClick

- • Used to create a press effect on the button when clicked.
- • 0%: Start at normal size.
- • 50%: Scale down slightly to 90%.
- • 100%: Return to original size.

@keyframes moveEyes

- • Animates the cat's eyes to simulate looking around.
- • 20%: Shift left and up.
- • 40%: Return to center.
- • 65%: Shift right and up.
- • 100%: Return to horizontal center, slightly lower.

@keyframes leftWhiskers

- • Makes the left whiskers swing slightly.
- • 20%: Rotate clockwise 5 degrees.
- • 60%: Rotate counterclockwise 5 degrees.

@keyframes rightWhiskers

- • Makes the right whiskers swing in a different rhythm.
- • 5%: Slight counterclockwise tilt.
- • 90%: Rotate clockwise 7 degrees.

@keyframes move

- • Adds a full-body tilt animation to the cat.
- • 20%: Slight tilt forward.
- • 40%: More pronounced tilt forward.
- • 60%: Tilt backward.
- • 85%: Stronger backward tilt.

HTML + Tailwind CSS: Button Layout

This annotation explains what each class and element does in a beginner-friendly way.

<body class="bg-gradient-to-t from-blue-200 to-white text-gray-800 font-serif px-4 py-6">

> 💬 *This applies a top-to-bottom gradient from light blue to white as the background, sets text color to gray, uses a serif font, and adds padding of 1rem (16px) on all sides.*

<h1 class="text-4xl font-bold text-center text-blue-900 mb-6">SVG Essentials</h1>

> 💬 *Creates a large, bold, centered heading in dark blue with a margin below it.*

> 💬 *This is a container for the buttons. It uses Flexbox to stack items vertically on small screens and horizontally with wrapping on larger ones. The buttons are centered and spaced evenly with gaps and a bottom margin.*

<button data-add="blue-eyes" class="inline-block uppercase bg-blue-200 hover:bg-blue-300 px-4 py-2 rounded w-64 text-center">Blue eyes</button>

> 💬 *This button adds the 'blue-eyes' class when clicked. It has a light blue background that darkens on hover, white text, padding, rounded corners, fixed width (16rem), and centered text.*

<button data-remove="blue-eyes" class="inline-block uppercase bg-green-200 hover:bg-green-300 px-4 py-2 rounded w-64 text-center">Green eyes</button>

> 💬 *This button removes the 'blue-eyes' class. Similar styling as above, but with a green background and hover effect.*

<button data-add="sad" class="inline-block uppercase bg-gray-300 hover:bg-gray-400 px-4 py-2 rounded w-64 text-center">Sad</button>

Adds the 'sad' class. Styled with gray background and hover transition.

<button data-remove="sad" class="inline-block uppercase bg-yellow-200 hover:bg-yellow-300 px-4 py-2 rounded w-64 text-center">Happy</button>

Removes the 'sad' class. Uses yellow for background colors.

<button data-add="move-around" class="inline-block uppercase bg-purple-200 hover:bg-purple-300 px-4 py-2 rounded w-64 text-center">Move around</button>

Adds the 'move-around' class to animate the SVG. Has a purple background and hover color.

<button data-remove="move-around" class="inline-block uppercase bg-red-200 hover:bg-red-300 px-4 py-2 rounded w-64 text-center">Sit still</button>

Removes the 'move-around' animation class. Uses red color theme.

SVG Container and Cat Graphic

> 💭 *Creates a container for the SVG image, centers it horizontally, limits its max width, and adds padding.*

<svg class="cat w-full h-auto" viewBox="0 0 640 480" xmlns="http://www.w3.org/2000/svg" aria-label="Cartoon Cat">

> 💭 *Starts the SVG. It fills the full width of its container with auto height. The viewBox defines a 640x480 drawing area, and the aria-label helps screen readers.*

<path d="..." fill="#999999" />

> 💭 *Draws the main shape of the cat using a path. The fill color is gray.*

<g transform="rotate(-10 382.5 241)">

> 💭 *A group that rotates the right eye and its inner shapes slightly to give it a dynamic tilt.*

<ellipse fill="#ffffff" cx="382.5" cy="241" rx="56.5" ry="84.75" />

> 💭 *Draws the outer white part of the eye.*

<ellipse class="eye-color" fill="#1b6811" cx="382.38" cy="286.68" rx="31" ry="35" />

> 💭 *The iris of the eye, colored green. It can change color with CSS.*

<ellipse fill="#ffffff" cx="380.38" cy="303.68" rx="6" ry="6" />

> 💭 *A small white ellipse to represent light reflection in the eye.*

<g> ... </g> (Second Eye)

> 💭 *Another group to create the left eye, symmetrical to the right one.*

<path d="..." fill="#fcbfdb" />

> 💭 *Draws the cat's nose area or mouth region in pink.*

<path d="..." fill="none" />

> 💭 *Outlines a part of the body or fur without filling it with color.*

<path class="mouth" d="..." fill="#000000" />

> 💭 *Shapes the cat's mouth in black. This element can be animated by toggling its class.*

<g class="left-whiskers"> ... </g>

> 💭 *Group of lines representing the cat's left whiskers.*

<g class="right-whiskers"> ... </g>

> 💭 *Group of lines representing the cat's right whiskers.*

Page Content Container (SVG Text Section)

<!-- Page Content -->

> 💬 *This comment helps developers understand that this section contains the main written content of the page.*

> 💬 *This wraps the content. It:*
> *• Limits the width to a maximum of 5xl.*
> *• Centers the content (mx-auto).*
> *• Uses a grid layout.*
> *• Shows 1 column by default and 2 columns on medium+ screens (md:grid-cols-2).*
> *• Adds spacing between grid items (gap-6).*
> *• Uses larger text (text-lg) and more spacing between lines (leading-relaxed) for easier reading.*

 ...

> 💬 *This is a paragraph of text that explains what SVG is and why it's useful for web graphics. It's placed inside the grid layout for clear, readable formatting.*

 ... (second paragraph)

> 💬 *The second paragraph continues the explanation. It focuses on how SVG works with CSS/JavaScript, boosts accessibility and SEO, and is easier to manage in code projects.*

JavaScript for SVG Interactivity

```
const buttons = document.querySelectorAll(".buttons button");
```

💬 Selects all <button> elements inside the element with class 'buttons'. These are the buttons students click to trigger changes on the SVG.

```
const catSVG = document.querySelector("svg");
```

💬 Selects the first <svg> element on the page (the cat graphic). This is what the buttons will update.

```
buttons.forEach((button) => {
```

💬 Loops through each button so we can add event listeners to them individually.

```
  button.addEventListener("click", () => {
```

💬 Adds a 'click' event to each button. When the button is clicked, it runs the function inside.

```
    const addClass = button.getAttribute("data-add");
```

💬 Checks if the button has a 'data-add' attribute. If it does, it stores that value in the variable 'addClass'.

```
    const removeClass = button.getAttribute("data-remove");
```

💬 Checks if the button has a 'data-remove' attribute. If it does, it stores that value in the variable 'removeClass'.

```
    if (addClass) catSVG.classList.add(addClass);
```

💬 If the 'addClass' variable has a value, we add that class to the SVG element (to change how it looks or moves).

```
    if (removeClass) catSVG.classList.remove(removeClass);
```

💭 If the 'removeClass' variable has a value, we remove that class from the SVG element.

```
    button.classList.add("clicked");
```

💭 Temporarily adds a class to animate the button when clicked.

```
    setTimeout(() => {
```

💭 Starts a timer that waits 300 milliseconds before running the next function.

```
      button.classList.remove("clicked");
```

💭 Removes the 'clicked' class to reset the button animation.

```
    }, 300);
```

💭 The timeout is set for 300 milliseconds (same duration as the animation).

```
  });
```

💭 Ends the click event function.

```
});
```

💭 Ends the forEach loop.