Royal Enfield Design

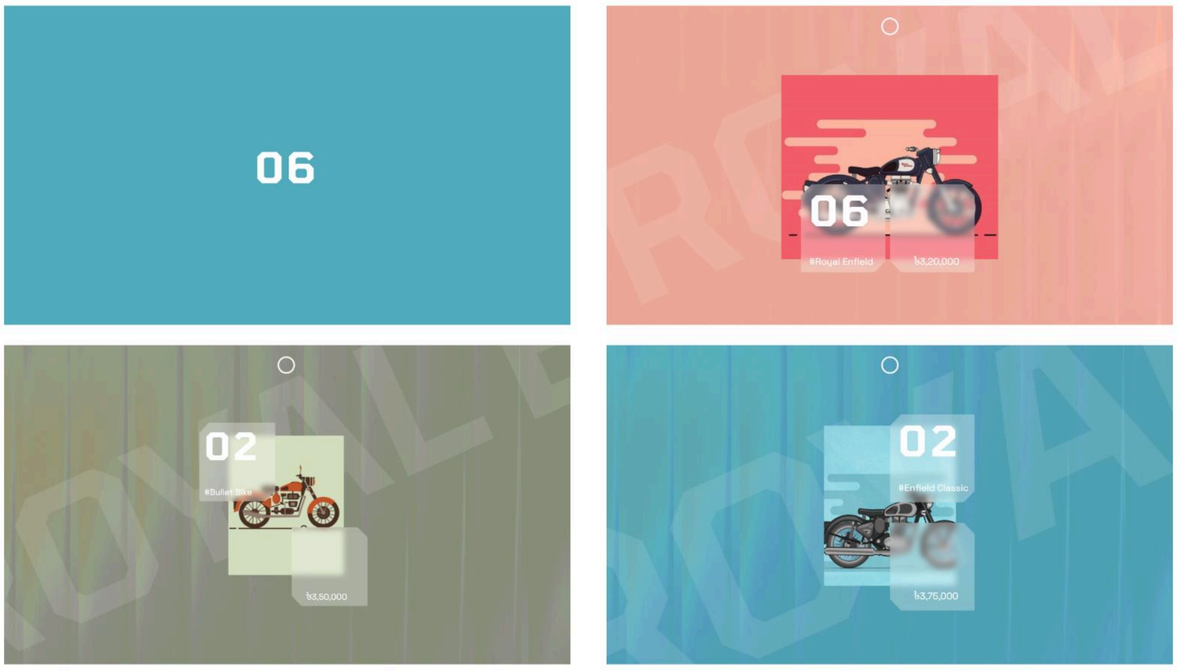


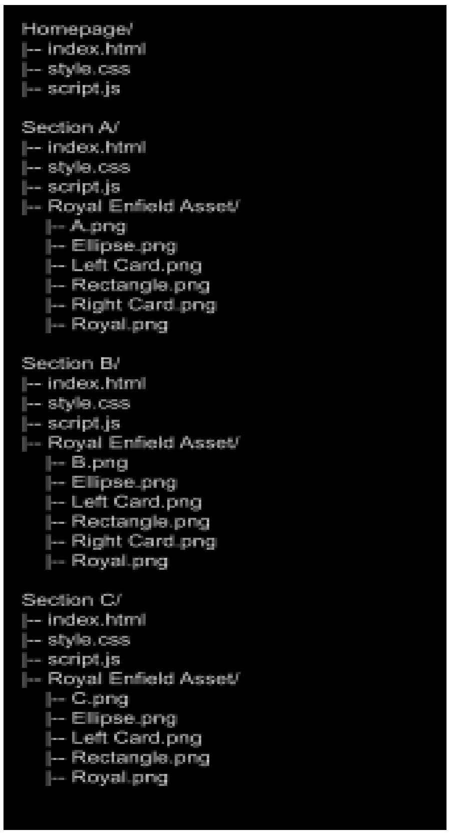
Figure (01): Final Output

1. Objective

The goal of the project is to design an interactive card layout with:

1. Hover effects for scaling and animation.
2. Dynamic reactivity using JavaScript for counters.
3. A 12-grid layout system for responsive design.
4. Alignment with modern design principles and adherence to Figma specifications.

2. File Structure



```
Homepage/
|-- index.html
|-- style.css
|-- script.js

Section A/
|-- index.html
|-- style.css
|-- script.js
|-- Royal Enfield Asset/
    |-- A.png
    |-- Ellipse.png
    |-- Left Card.png
    |-- Rectangle.png
    |-- Right Card.png
    |-- Royal.png

Section B/
|-- index.html
|-- style.css
|-- script.js
|-- Royal Enfield Asset/
    |-- B.png
    |-- Ellipse.png
    |-- Left Card.png
    |-- Rectangle.png
    |-- Right Card.png
    |-- Royal.png

Section C/
|-- index.html
|-- style.css
|-- script.js
|-- Royal Enfield Asset/
    |-- C.png
    |-- Ellipse.png
    |-- Left Card.png
    |-- Rectangle.png
    |-- Royal.png
```

Homepage: In this section, I will discuss the structure and functionality of the given homepage code. The source code includes three key components: HTML, CSS, and JavaScript. Each plays a specific role in creating the layout, styling, and interactivity of the homepage.

Fig:(02) Homepage

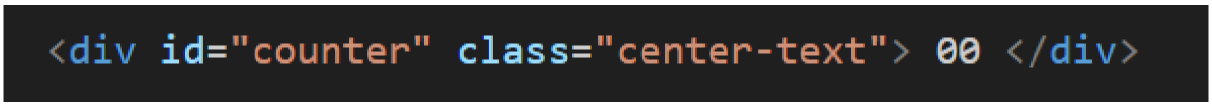
Source Code Breakdown

1. HTML

The HTML structure provides the layout for the webpage, including:

- Document Metadata
  - The <!DOCTYPE html> declaration specifies the HTML5 standard.
  - The <html> tag with lang="en" declares the document's language.
  - The <head> section includes metadata, fonts, stylesheets, and the page title.
- Content
  - The <body> tag contains the content:
    - A <div> with a class of main for the main container.
    - An inner <div> with an id of counter to display the counter value, initialized to 00.

Key Code Snippets



```
<div id="counter" class="center-text"> 00 </div>
```

This displays the counter, styled with CSS for centering and visual appeal.

External Links

- Google Fonts are linked to style the text.
- An external stylesheet (style.css) and script (script.js) are linked.

2. CSS

The CSS file (style.css) styles the webpage with the following features:

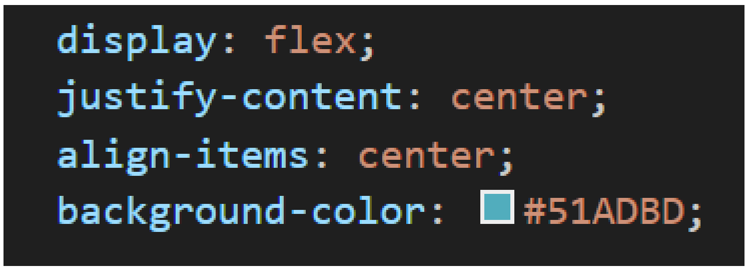
- Global Settings
  - All elements have margin and padding set to 0, and box-sizing: border-box ensures consistent sizing.

Key Classes

- .main
  - A flex container that takes up the full viewport width (100vw) and height (100vh).
  - It centers the content using justify-content: center and align-items: center.
  - The background color is a calming blue shade (#51ADBD).
- .center-text

- Styles the counter text with:
    - A font size of 4.5rem.
    - The "Tomorrow" font-family (linked from Google Fonts).
    - Bold font weight (600).
    - A letter-spacing of 0.3125rem.
    - White color for contrast against the blue background.

Example:



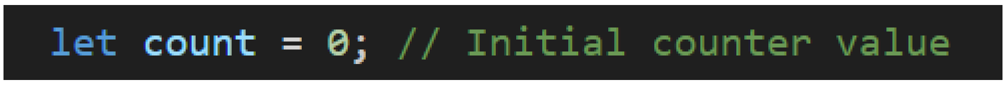
```css
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #51ADBD;
```

3. JavaScript

The JavaScript file (script.js) handles the functionality of the counter.

Step-by-Step Explanation

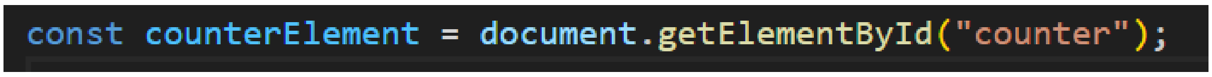
1. Initialize Counter Value



```javascript
let count = 0; // Initial counter value
```

The variable count is initialized to 0 and represents the current counter value.

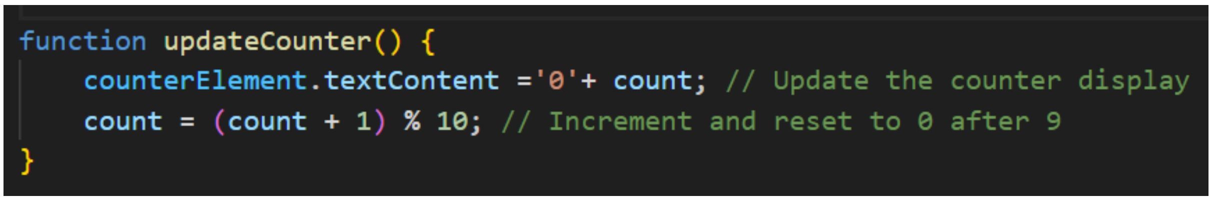
2. Access Counter Element



```javascript
const counterElement = document.getElementById("counter");
```

The counterElement variable stores a reference to the HTML <div> with id="counter".

3. Update Counter Function

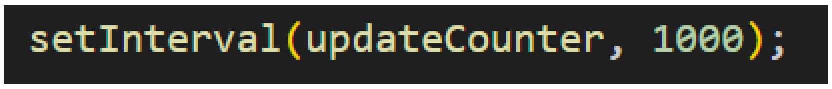


```javascript
function updateCounter() {
    counterElement.textContent ='0'+ count; // Update the counter display
    count = (count + 1) % 10; // Increment and reset to 0 after 9
}
```

Display Logic: The textContent of counterElement is updated with the current value of count, prefixed by 0.

Increment Logic: The value of count is incremented by 1. Using the modulo operator (% 10), it resets to 0 after reaching 9.

4. Set Interval



```
setInterval(updateCounter, 1000);
```

The setInterval function calls updateCounter every 1000 milliseconds (1 second), ensuring the counter updates in real time.

How the Code Works Together

1. The HTML provides the structure of the webpage, with a container (main) and a counter display (counter).
2. The CSS ensures the layout is visually centered, styled, and consistent across devices.
3. The JavaScript updates the counter every second, displaying it in the specified format and ensuring a loop from 00 to 09.

Output

- The webpage initially shows 00 at the center.
- Every second, the counter increments by 1.
- After 09, it resets to 00.

SECTION A, B, And C

Now, I will discuss Sections A, B, and C, as the source code in all three sections functions in the same way. Here, I will analyze the provided source code and elaborate on it, highlighting some key points.

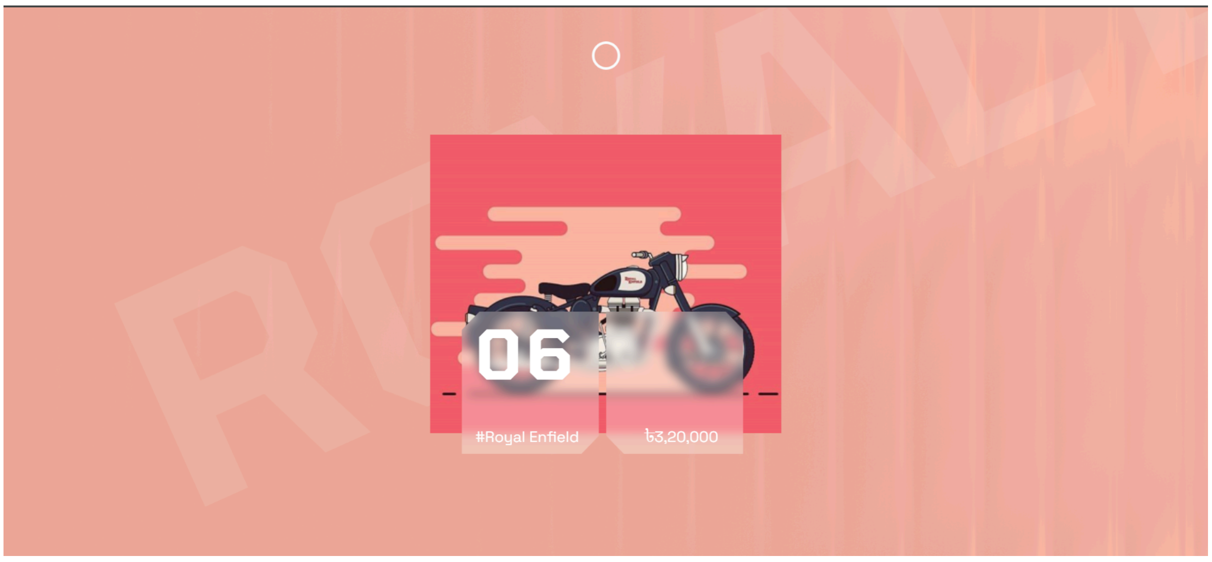


Fig: 03 (Section A)

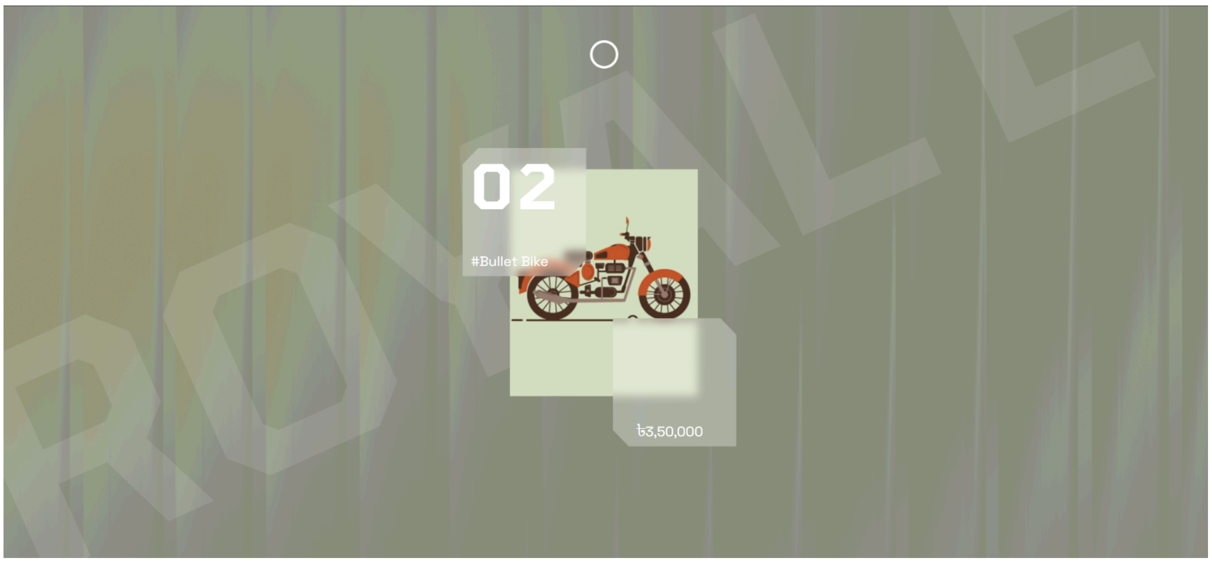


Fig: 04 (Section B)

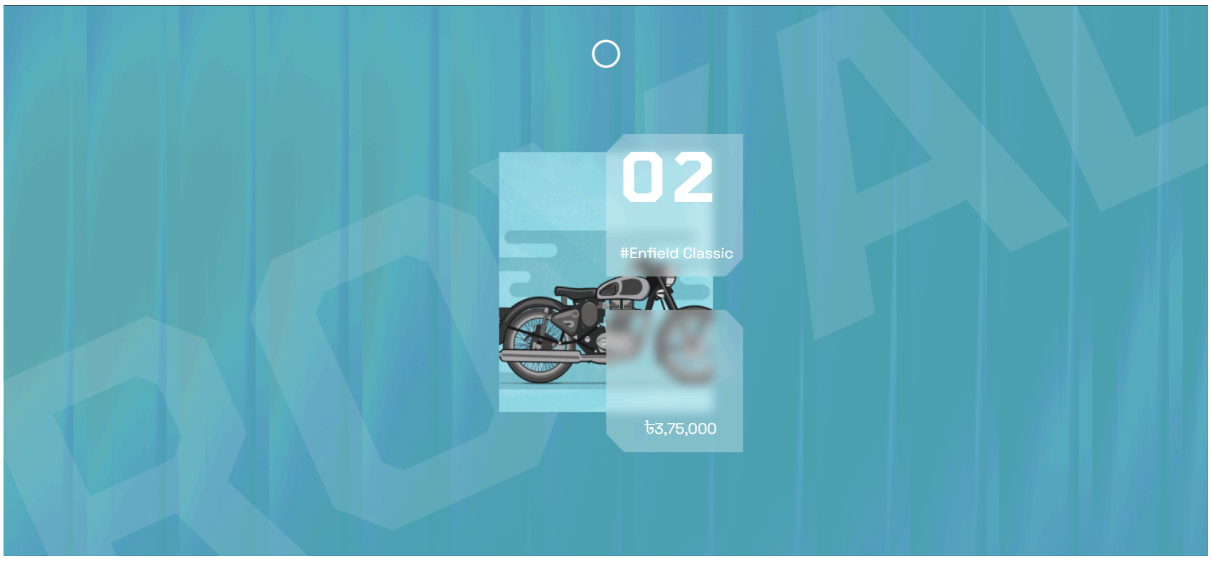


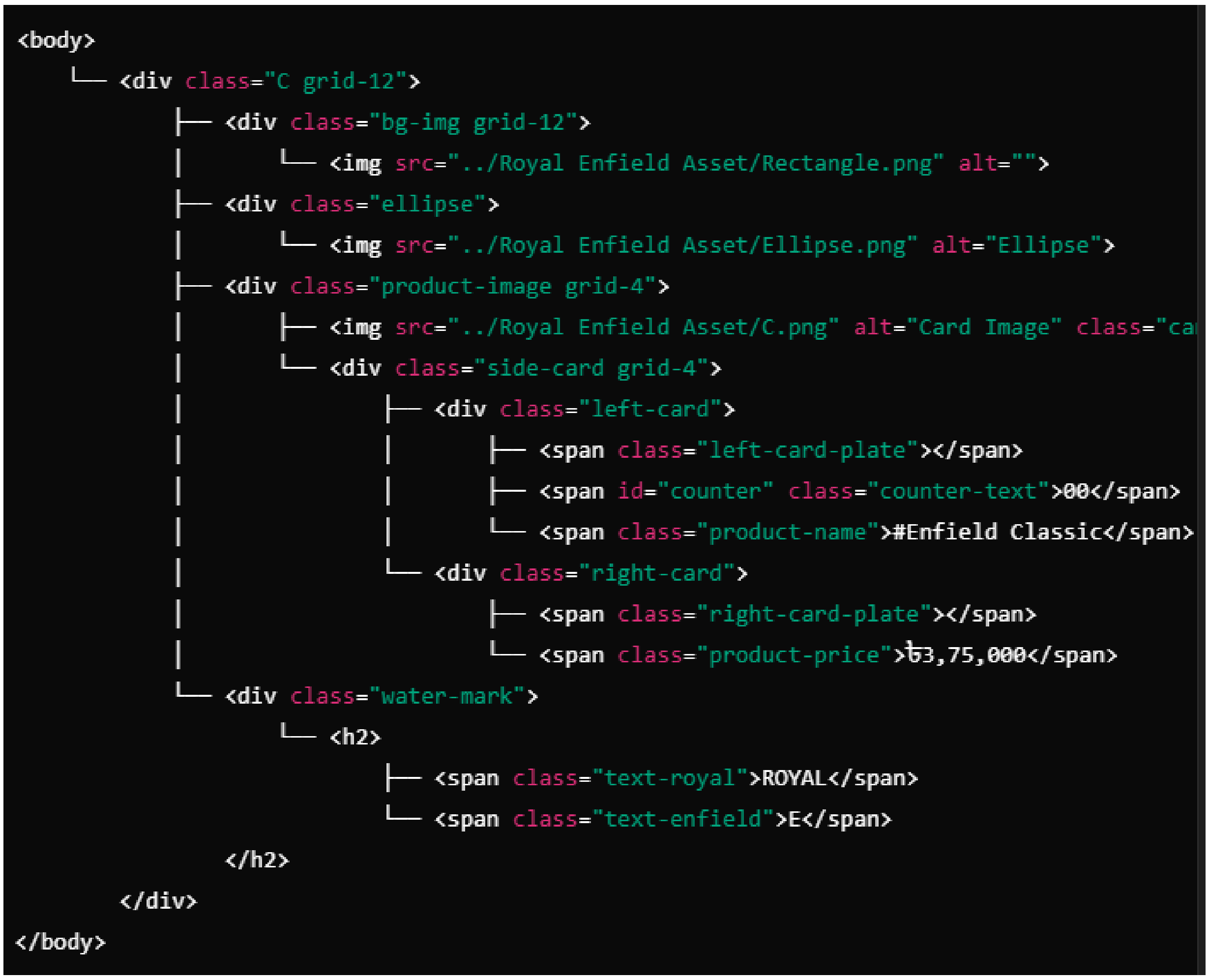
Fig: 05 (Section C)

Overview

This code represents the structure, style, and functionality of a product display page for "Section A." It features:

- A background image for aesthetics.
- An ellipse decoration for added design.
- A product image with accompanying details such as name, price, and an animated counter.
- A watermark displaying the brand name.
- Dynamic animations, hover effects, and responsive behavior.

File Structure



```
<body>
    └── <div class="C grid-12">
            ├── <div class="bg-img grid-12">
            |       └── <img src="../Royal Enfield Asset/Rectangle.png" alt="">
            ├── <div class="ellipse">
            |       └── <img src="../Royal Enfield Asset/Ellipse.png" alt="Ellipse">
            ├── <div class="product-image grid-4">
            |       ├── <img src="../Royal Enfield Asset/C.png" alt="Card Image" class="ca
            |       └── <div class="side-card grid-4">
            |               ├── <div class="left-card">
            |               |       ├── <span class="left-card-plate"></span>
            |               |       ├── <span id="counter" class="counter-text">00</span>
            |               |       └── <span class="product-name">#Enfield Classic</span>
            |               └── <div class="right-card">
            |                       ├── <span class="right-card-plate"></span>
            |                       └── <span class="product-price">৳3,75,000</span>
            └── <div class="water-mark">
                    └── <h2>
                            ├── <span class="text-royal">ROYAL</span>
                            └── <span class="text-enfield">E</span>
                    </h2>
        </div>
</body>
```

1. HTML Explanation

File: index.html

The HTML file includes the structure of the webpage. It uses a flex-based grid layout to position elements effectively.

Key Sections:

- Head Section
  - Includes metadata, Google Fonts, and the external CSS file style.css.
  - Google Fonts:
    - Inter
    - Montserrat
    - Open Sans
    - Poppins
    - Roboto
    - Space Grotesk
    - Tomorrow

- Body Section
  Contains a grid-based layout divided into multiple elements:
  - bg-img: A background image of the product.
  - ellipse: Decorative circular graphic for aesthetic enhancement.
  - product-image:
    - Includes the main product image and two side cards (left and right).
    - The left card displays a counter and product name.
    - The right card shows the product price.
  - water-mark: A large faded "ROYAL" text for branding.

- Scripts
  - script.js: Contains the JavaScript logic to animate a counter.

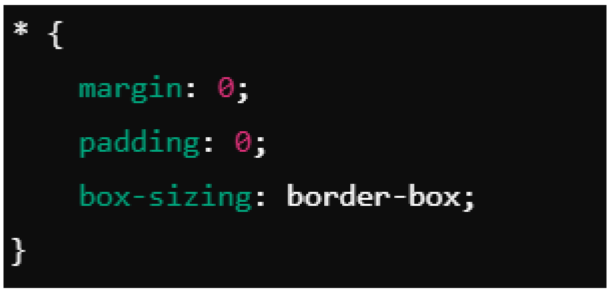
2. CSS Explanation

File: style.css

The CSS file provides responsive styling, animation, and interaction effects.

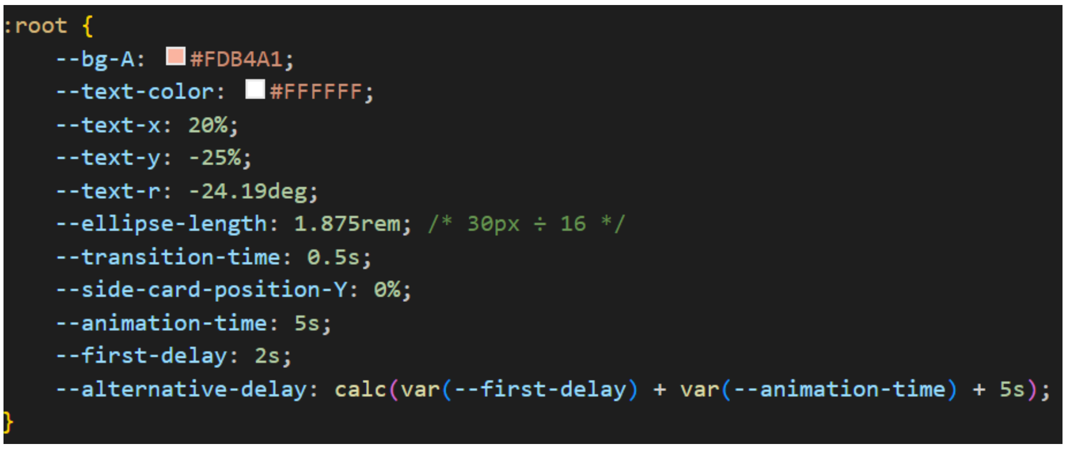
Key Elements:

1. **Global and Default Styling**



```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

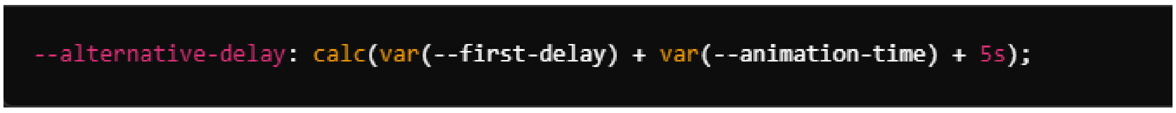
- Resets all margins and paddings to 0 for all elements.
- Sets the box-sizing to border-box, ensuring padding and border are included in an element's width and height calculations.



```css
:root {
    --bg-A: #FDB4A1;
    --text-color: #FFFFFF;
    --text-x: 20%;
    --text-y: -25%;
    --text-r: -24.19deg;
    --ellipse-length: 1.875rem; /* 30px ÷ 16 */
    --transition-time: 0.5s;
    --side-card-position-Y: 0%;
    --animation-time: 5s;
    --first-delay: 2s;
    --alternative-delay: calc(var(--first-delay) + var(--animation-time) + 5s);
}
```

Defines reusable CSS variables (--variable-name) for properties like colors, dimensions, animation durations, and positions.

These variables are centralized for easier updates and maintainability.



```css
--alternative-delay: calc(var(--first-delay) + var(--animation-time) + 5s);
```

This line defines the --alternative-delay variable by calculating a dynamic delay value, which is the sum of three values:

1. **var(--first-delay)**:
   This refers to the --first-delay CSS variable, which is set to 2s (2 seconds). It represents the initial delay before an animation begins.
2. **var(--animation-time)**:
   This is the value of the --animation-time CSS variable, set to 5s (5 seconds). This indicates how long the animation takes to complete.

3. **5s**:
   This is a fixed value added to the calculation, meaning an additional 5 seconds will be added after the animation duration and the first delay.
4. **calc() function**:
   The calc() function is used to perform mathematical operations in CSS. Here, it adds the three values together: 2s + 5s + 5s.

Result:

The result of the calc() function is 12s (2s + 5s + 5s). So, the --alternative-delay variable will be set to 12 seconds. This value can be used to control how long an animation (or transition) should be delayed before it begins.

Purpose in Context:

The --alternative-delay variable is useful for controlling the timing of animations, especially when multiple animations are running on the same page. It can be used to stagger the start times of different animations or to create overlapping animations with different durations.

2. Body Styling

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden;
}
```

- Makes the body a flex container.
- Centers content horizontally and vertically using justify-content and align-items.
- Hides any overflowing content with overflow: hidden.

3. Grid Width Classes

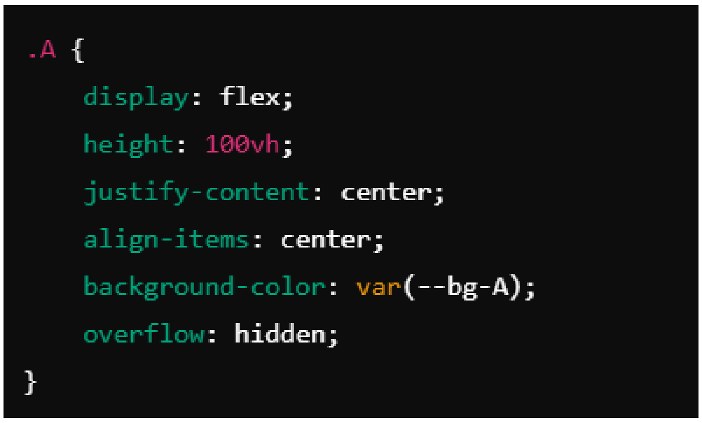
```
.grid-1 { width: 8%; }
.grid-2 { width: 16.66%; }
... (similar patterns for other grids)
.grid-12 { width: 100%; }
```

Provides classes to set fractional widths for grid-based layouts.

Can be used to divide a layout into equal parts.

4. Section A Styling

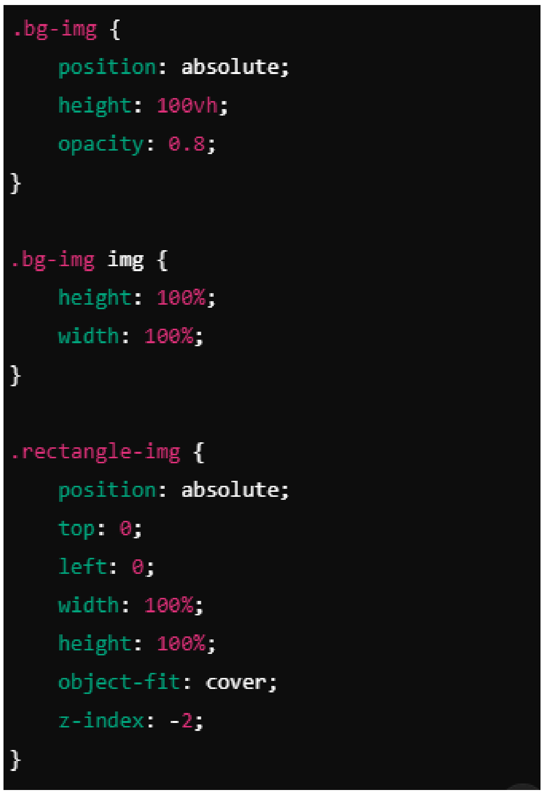
The container for the central content:



```css
.A {
    display: flex;
    height: 100vh;
    justify-content: center;
    align-items: center;
    background-color: var(--bg-A);
    overflow: hidden;
}
```

- Defines a section with the class .A.
- Sets its height to 100% of the viewport (100vh) and centers its content both horizontally and vertically.
- Applies a background color from the --bg-A variable.

5. Background and Images



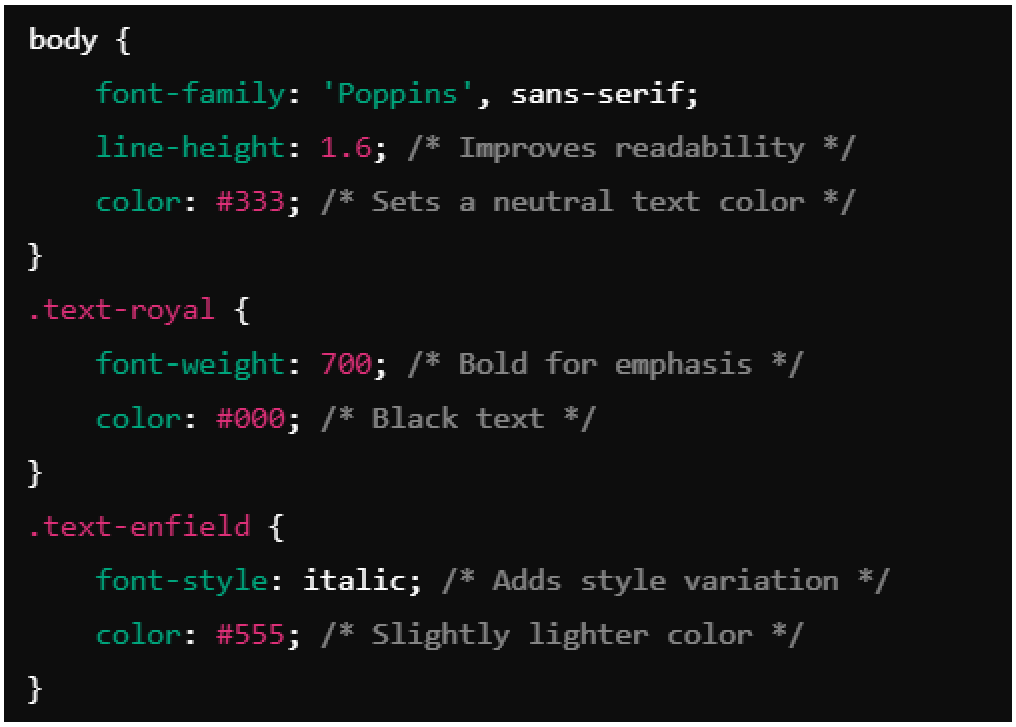
```css
.bg-img {
    position: absolute;
    height: 100vh;
    opacity: 0.8;
}

.bg-img img {
    height: 100%;
    width: 100%;
}

.rectangle-img {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: cover;
    z-index: -2;
}
```

- .bg-img creates a semi-transparent full-page background image.

- .rectangle-img ensures the background image fully covers the section without distortion using object-fit: cover.
- z-index: -2 places the image behind other content.

6. Typography and Fonts

The code includes several font families (e.g., *Poppins*, *Roboto*). CSS styles likely define font usage across different text elements.

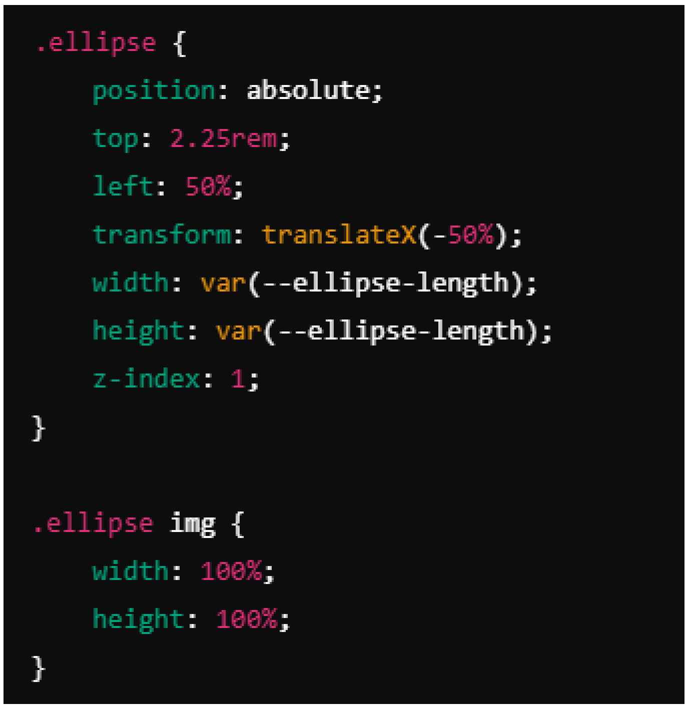


```css
body {
    font-family: 'Poppins', sans-serif;
    line-height: 1.6; /* Improves readability */
    color: #333; /* Sets a neutral text color */
}
.text-royal {
    font-weight: 700; /* Bold for emphasis */
    color: #000; /* Black text */
}
.text-enfield {
    font-style: italic; /* Adds style variation */
    color: #555; /* Slightly lighter color */
}
```

The code includes several font families (e.g., *Poppins*, *Roboto*). CSS styles likely define font usage across different text elements.

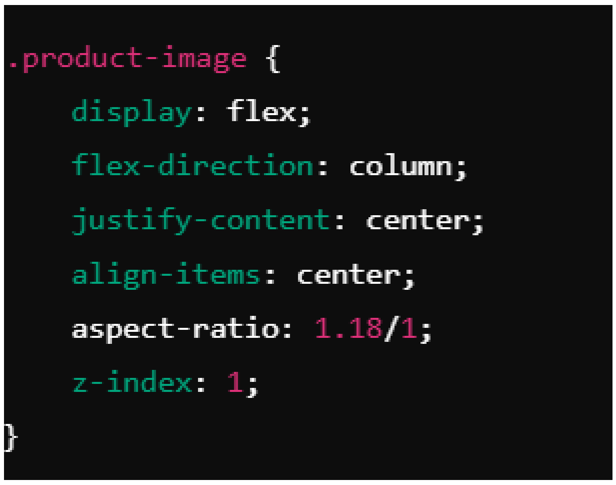
- Purpose: Ensures consistent, aesthetically pleasing typography.

7. Ellipses and Product Images



```css
.ellipse {
    position: absolute;
    top: 2.25rem;
    left: 50%;
    transform: translateX(-50%);
    width: var(--ellipse-length);
    height: var(--ellipse-length);
    z-index: 1;
}

.ellipse img {
    width: 100%;
    height: 100%;
}
```

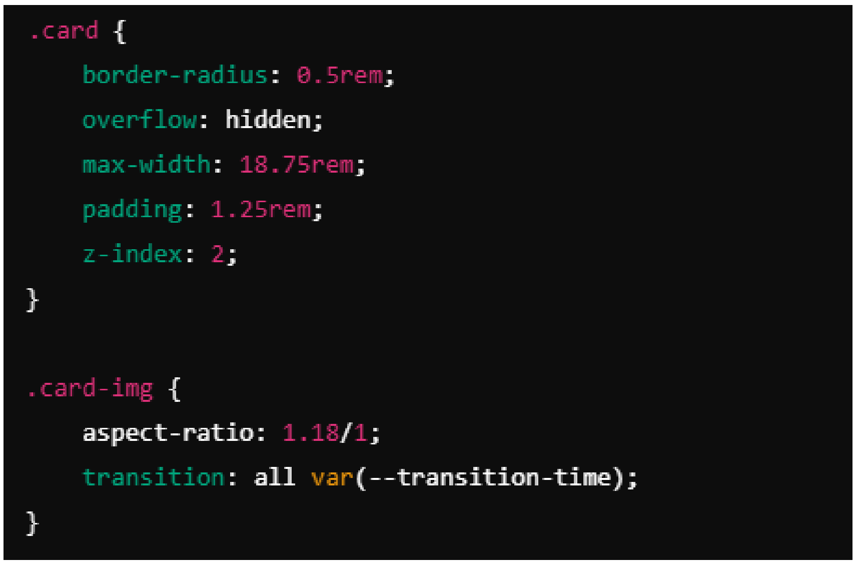
- .ellipse positions a small circle at the top-center using relative units for better responsiveness.
- The dimensions and position are controlled by the --ellipse-length variable.



```css
.product-image {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    aspect-ratio: 1.18/1;
    z-index: 1;
}
```

- Ensures product images maintain a specific aspect ratio for consistent appearance.

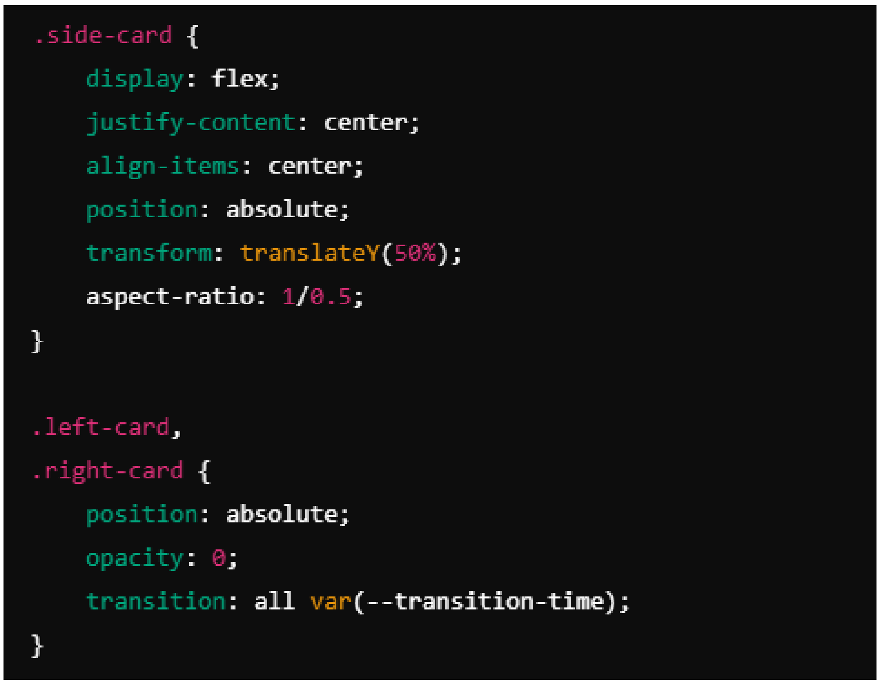
8. Card Styles



```css
.card {
    border-radius: 0.5rem;
    overflow: hidden;
    max-width: 18.75rem;
    padding: 1.25rem;
    z-index: 2;
}

.card-img {
    aspect-ratio: 1.18/1;
    transition: all var(--transition-time);
}
```

- .card styles product cards with rounded corners (border-radius) and padding for spacing.
- .card-img defines a responsive aspect ratio and a smooth transition effect when hovered.

9. Side Cards



```css
.side-card {
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    transform: translateY(50%);
    aspect-ratio: 1/0.5;
}

.left-card,
.right-card {
    position: absolute;
    opacity: 0;
    transition: all var(--transition-time);
}
```

- Defines side cards that are hidden (opacity: 0) by default.
- Positions them off-screen initially, transitioning them into view when hovered.

10. Animations

```css
@keyframes royalAnimation {
    0% { transform: translateX(0%); }
    25% { transform: translateX(-7%); }
    50% { transform: translateX(-14%); }
    75% { transform: translateX(-21%); }
    100% { transform: translateX(-25%); }
}

@keyframes alternativeAnimation {
    0% { transform: translateX(-25%); }
    25% { transform: translateX(-18.75%); }
    50% { transform: translateX(-12.5%); }
    75% { transform: translateX(-6.25%); }
    100% { transform: translateX(0%); }
}
```

- Creates two animations to move elements horizontally in a wave-like manner.
- Smoothly transitions elements between defined keyframes for visual effects.

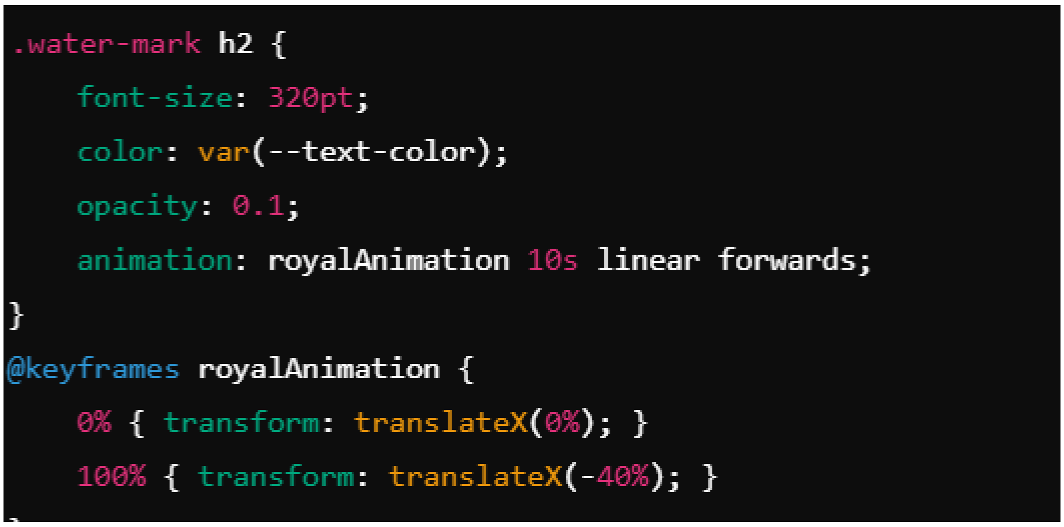
11. Hover Effects

```css
.product-image:hover .card-img {
    transform: scale(1.5);
    cursor: pointer;
}

.product-image:hover .left-card {
    transform: translateX(-55%) translateY(var(--side-card-position-Y));
    opacity: 1;
}

.product-image:hover .right-card {
    transform: translateX(50%) translateY(var(--side-card-position-Y));
    opacity: 1;
}
```

Enlarges product images when hovered and reveals side cards by adjusting transform and opacity.

12. Watermark and Animation

- Uses @keyframes for smooth transitions.
- Example: royalAnimation moves the "ROYAL" watermark across the screen:



```css
.water-mark h2 {
    font-size: 320pt;
    color: var(--text-color);
    opacity: 0.1;
    animation: royalAnimation 10s linear forwards;
}
@keyframes royalAnimation {
    0% { transform: translateX(0%); }
    100% { transform: translateX(-40%); }
```

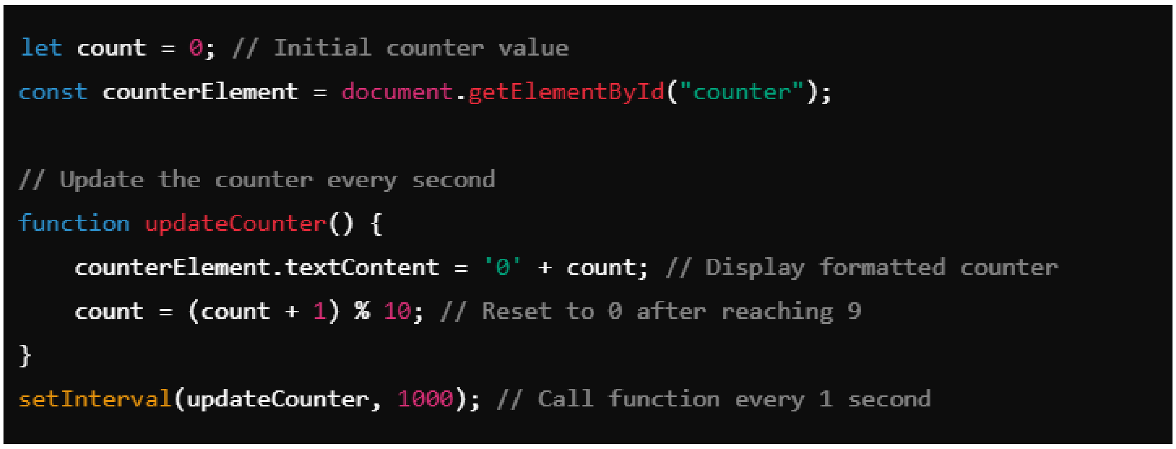
3. JavaScript Explanation

File: script.js

The script handles the counter animation:

Key Features:

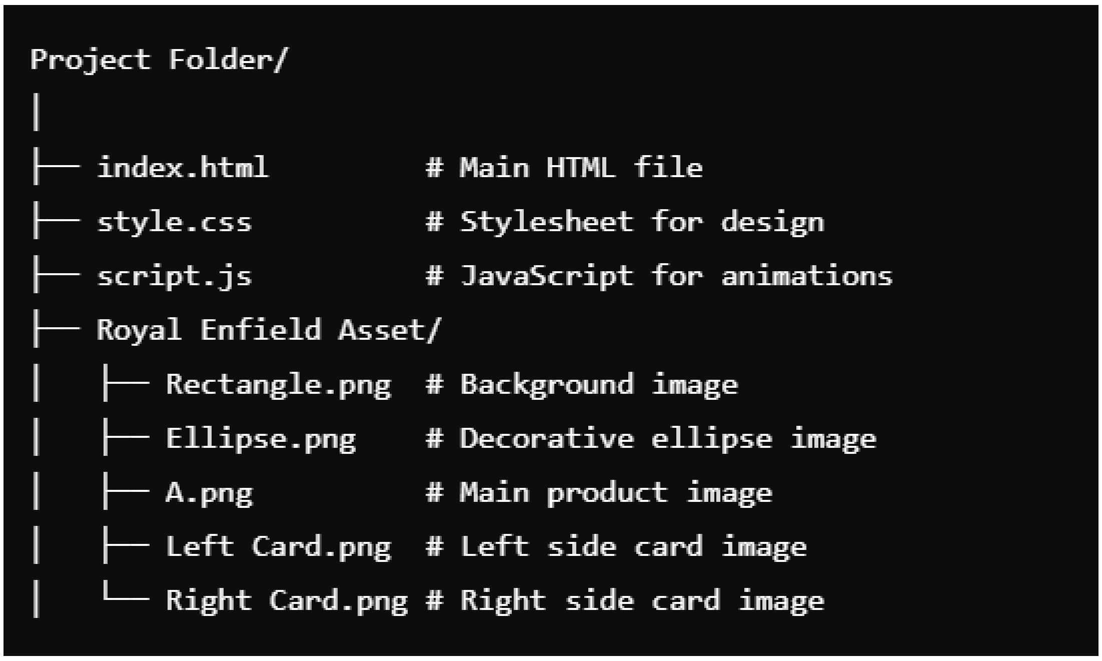
- The counter starts from 00 and increments every second.
- The value resets to 00 after reaching 09.



```javascript
let count = 0; // Initial counter value
const counterElement = document.getElementById("counter");

// Update the counter every second
function updateCounter() {
    counterElement.textContent = '0' + count; // Display formatted counter
    count = (count + 1) % 10; // Reset to 0 after reaching 9
}
setInterval(updateCounter, 1000); // Call function every 1 second
```

4. Complete Workflow

File Structure



```
Project Folder/
|
├── index.html         # Main HTML file
├── style.css          # Stylesheet for design
├── script.js          # JavaScript for animations
├── Royal Enfield Asset/
|   ├── Rectangle.png  # Background image
|   ├── Ellipse.png    # Decorative ellipse image
|   ├── A.png          # Main product image
|   ├── Left Card.png  # Left side card image
|   └── Right Card.png # Right side card image
```

5. Expected Output

Desktop View

- A fullscreen layout with:
  - A background image.
  - The product image is at the center.
  - Left and Right Cards are revealed on hover.
  - A counter that increments every second.
  - A large watermark ("ROYAL") slides diagonally in the background.

Summary

This CSS code is for a visually rich layout, combining animations, hover effects, and responsive design principles. It creates an engaging and interactive user interface, ideal for showcasing products or visual content.