Html and CSS →

HTML--

Certainly! HTML (Hypertext Markup Language) is the standard language for creating and designing web pages. Here are some basic HTML tags and their properties, explained in a simple way:

1. **`<!DOCTYPE html>`**: This declaration defines the document type and version of HTML. It should be the first line in your HTML document.

2. **`<html>`**: This tag wraps the entire HTML document. It serves as the root element.

3. **`<head>`**: Inside the `<html>` tag, the `<head>` tag contains meta-information about the HTML document, such as the title, character set, and links to external stylesheets.

   - **`<title>`**: Specifies the title of the HTML document, which appears in the browser's title bar or tab.

   ```html
   <title>Your Page Title</title>
   ```

4. **`<body>`**: This tag contains the content of the HTML document, such as text, images, links, etc.

   - **Text Tags:**
     - **`<h1> to <h6>`**: Heading tags, where `<h1>` is the largest and `<h6>` is the smallest.

     ```html
     <h1>This is a Heading</h1>
     ```

     - **`<p>`**: Paragraph tag for text.

     ```html
     <p>This is a paragraph.</p>
     ```

     - **`<br>`**: Line break tag to create a line break within text.

     ```html
     <p>This is a line of text.<br>This is a new line.</p>
     ```

   - **Link Tag:**
     - **`<a>`**: Anchor tag for creating hyperlinks.

     ```html
     <a href="https://www.example.com">Visit Example.com</a>
     ```

   - **Image Tag:**
     - **`<img>`**: Image tag for displaying images.

     ```html
     ```

```
<img src="image.jpg" alt="Description of the image">
```

- **List Tags:**
  - **`<ul>`**: Unordered list.

    ```html
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>
    ```

  - **`<ol>`**: Ordered list.

    ```html
    <ol>
        <li>First</li>
        <li>Second</li>
    </ol>
    ```

  - **`<li>`**: List item.

- **Form Tags:**
  - **`<form>`**: Form container.

    ```html
    <form action="/submit" method="post">
        <!-- Form fields go here -->
    </form>
    ```

  - **`<input>`**: Input field.

    ```html
    <input type="text" placeholder="Enter your name">
    ```

5. **Attributes**: These are additional information provided within the opening tags to modify or define the element's behavior.

   - **`class` and `id`**: Used to define the styling or for scripting purposes.

   ```html
   <p class="important-text" id="first-paragraph">This is important text.</p>
   ```

   - **`style`**: Allows you to add inline CSS styles to an element.

   ```html
   <p style="color: red; font-size: 16px;">Styled text.</p>
   ```

   - **`src` and `alt`**: Used with the `<img>` tag for specifying the image source and providing alternative text.

   ```html
   ```

<img src="image.jpg" alt="Description of the image">
```

- **`href`**: Used with the `<a>` tag to specify the destination of the hyperlink.

```html
<a href="https://www.example.com">Visit Example.com</a>
```

Certainly! Let's explore some more HTML tags and properties:

6. **Tables:**
   - **`<table>`**: Defines a table.

     ```html
     <table border="1">
       <tr>
         <td>Row 1, Cell 1</td>
         <td>Row 1, Cell 2</td>
       </tr>
       <tr>
         <td>Row 2, Cell 1</td>
         <td>Row 2, Cell 2</td>
       </tr>
     </table>
     ```

   - **`<tr>`**: Defines a table row.
   - **`<td>`**: Defines a table cell (data cell).

7. **Forms:**
   - **`<input>`**: Input fields for various types of user input.

     ```html
     <input type="text" placeholder="Username">
     <input type="password" placeholder="Password">
     ```

   - **`<textarea>`**: Defines a multiline text input.

     ```html
     <textarea rows="4" cols="50">Enter text here...</textarea>
     ```

   - **`<select>` and `<option>`**: Dropdown list.

     ```html
     <select>
       <option value="option1">Option 1</option>
       <option value="option2">Option 2</option>
     </select>

```

8. **Semantic Tags:**
   - **`<header>`**: Represents the header of a document or section.

     ```html
     <header>
         <h1>Website Title</h1>
     </header>
     ```

   - **`<nav>`**: Represents a navigation menu.

     ```html
     <nav>
         <ul>
             <li><a href="#">Home</a></li>
             <li><a href="#">About</a></li>
             <li><a href="#">Contact</a></li>
         </ul>
     </nav>
     ```

   - **`<article>`**: Represents a self-contained piece of content.

     ```html
     <article>
         <h2>Article Title</h2>
         Content goes here.
     </article>
     ```

   - **`<footer>`**: Represents the footer of a document or section.

     ```html
     <footer>
         © 2023 Your Website
     </footer>
     ```

9. **Media:**
   - **`<audio>`**: Embeds audio content.

     ```html
     <audio controls>
         <source src="audio.mp3" type="audio/mp3">
         Your browser does not support the audio tag.
     </audio>
     ```

   - **`<video>`**: Embeds video content.

     ```html
     <video width="320" height="240" controls>
         <source src="video.mp4" type="video/mp4">
         Your browser does not support the video tag.
     </video>

```

10. **Meta Tags:**
    - **`<meta>`**: Provides metadata about the HTML document, such as character set and viewport settings.

      ```html
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      ```

Certainly! Let's delve into a few more HTML elements and attributes:

11. **`<div>` and `<span>`:**
    - **`<div>`**: Defines a division or a section in an HTML document. Often used for layout purposes.

      ```html

          This is inside a division.

      ```

    - **`<span>`**: Inline container often used for applying styles to a small piece of text.

      ```html
      This is blue text.
      ```

12. **`<strong>` and `<em>`:**
    - **`<strong>`**: Represents strong importance or seriousness. Typically renders as bold text.

      ```html
      Important: Pay attention to this!
      ```

    - **`<em>`**: Represents emphasized text. Typically renders as italic text.

      ```html
      Emphasized: This text is emphasized.
      ```

13. **`<abbr>` and `<blockquote>`:**
    - **`<abbr>`**: Represents an abbreviation or acronym, often with a title attribute for the full form.

      ```html
      <abbr title="World Health Organization">WHO</abbr> is important.
      ```

    - **`<blockquote>`**: Represents a block of text that is a quotation from another source.

      ```html
      <blockquote>
          This is a quoted text from another source.
      </blockquote>
      ```
      ```

14. **`<iframe>`:**
    - Embeds an inline frame, typically used to embed external content such as maps or videos.

      ```html
      <iframe src="https://www.youtube.com/embed/your-video-code" width="560" height="315" frameborder="0" allowfullscreen></iframe>
      ```

15. **`<hr>`:**
    - Represents a horizontal line, often used to separate content.

      ```html
      <p>This is some content above the line.</p>
      <hr>
      <p>This is some content below the line.</p>
      ```

16. **`<sup>` and `<sub>`:**
    - **`<sup>`**: Represents superscript text.

      ```html
      <p>This is 10<sup>th</sup> edition.</p>
      ```

    - **`<sub>`**: Represents subscript text.

      ```html
      <p>H<sub>2</sub>O</p>
      ```

17. **`<address>`:**
    - Represents contact information for the author or owner of a document/article.

      ```html
      <address>
          Written by John Doe.<br>
          Visit us at: 123 Main Street, City.
      </address>
      ```

18. **`<progress>`:**
    - Represents the completion progress of a task.

      ```html
      <progress value="70" max="100">70%</progress>
      ```

Certainly! Let's cover a few more HTML elements:

19. **`<code>` and `<pre>`:**
    - **`<code>`**: Represents a single line of code.

      ```html
      <p>Use the <code>print()</code> function to display output.</p>
      ```

- **`<pre>`**: Represents preformatted text, maintaining both spaces and line breaks.

  ```html
  <pre>
  function example() {
      // Some code here
  }
  </pre>
  ```

20. **`<figure>` and `<figcaption>`:**
    - **`<figure>`**: Represents any content that is referenced from the main content, typically with an associated caption.

  ```html
  <figure>
      <img src="image.jpg" alt="Description">
      <figcaption>Caption for the image</figcaption>
  </figure>
  ```

21. **`<mark>`:**
    - Represents text that is marked or highlighted for reference or notation.

  ```html
  <p>Search for <mark>important keyword</mark> in the document.</p>
  ```

22. **`<time>`:**
    - Represents a specific period in time or a range of time.

  ```html
  <p>Meeting at <time datetime="2023-01-01T12:00">12:00 PM on January 1, 2023</time></p>
  ```

23. **`<dfn>`:**
    - Represents the defining instance of a term.

  ```html
  <p><dfn>HTML</dfn> is the standard markup language for creating web pages.</p>
  ```

24. **`<cite>`:**
    - Represents the title of a creative work, such as a book or movie.

  ```html
  <p><cite>The Great Gatsby</cite> by F. Scott Fitzgerald</p>
  ```

25. **`<bdo>`:**
    - Represents the direction of text. It is used to override the current text direction.

  ```html
  <p><bdo dir="rtl">This text is right-to-left</bdo></p>
  ```

26. **`<details>` and `<summary>`:**
    - **`<details>`**: Represents a disclosure widget from which the user can obtain additional information or controls.

    ```html
    <details>
        <summary>Click to view more details</summary>
        <p>Additional information goes here.</p>
    </details>
    ```

27. **`<fieldset>` and `<legend>`:**
    - **`<fieldset>`**: Groups related elements within a form and can include a border to distinguish the grouping.

    ```html
    <fieldset>
        <legend>Contact Information</legend>
        <!-- Form elements go here -->
    </fieldset>
    ```

    - **`<legend>`**: Represents a caption for the content of its parent `<fieldset>`.

28. **`<datalist>` and `<option>`:**
    - **`<datalist>`**: Contains a set of `<option>` elements that represent the permissible or suggested options available to users.

    ```html
    <input list="browsers" name="browser" id="browser">
    <datalist id="browsers">
        <option value="Chrome">
        <option value="Firefox">
        <option value="Safari">
        <option value="Edge">
        <option value="Opera">
    </datalist>
    ```

These additional HTML elements offer more flexibility and options for creating diverse and structured web content. Feel free to explore and experiment with these elements based on your specific needs.

Css---

Certainly! Cascading Style Sheets (CSS) is a styling language used to control the presentation and layout of HTML documents. Here are some basic CSS tags and properties explained with easy-to-understand descriptions:

1. Selector:

Selectors are used to target HTML elements and apply styles to them.

```css
/* Example Selector */
p {
  color: blue;
}
```

2. Property and Value:

CSS consists of property-value pairs. The property defines the aspect of the element you want to style, and the value specifies how you want to style it.

```css
/* Example Property and Value */
p {
  color: blue; /* Property: color, Value: blue */
}
```

3. Color:

Specifies the text color of an element.

```css
/* Example Color */
p {
  color: red;
}
```

4. Background-color:

Specifies the background color of an element.

```css
/* Example Background-color */
body {
  background-color: lightgray;
}
```

```

### 5. Font-size:

Specifies the size of the text.

```css
/* Example Font-size */
h1 {
  font-size: 24px;
}
```

6. Font-family:

Specifies the font of the text.

```css
/* Example Font-family */
body {
  font-family: "Arial", sans-serif;
}
```

7. Margin:

Specifies the space outside an element.

```css
/* Example Margin */
div {
  margin: 10px;
}
```

8. Padding:

Specifies the space inside an element.

```css
/* Example Padding */
p {
  padding: 5px;
}
```

9. Border:

Specifies the border of an element.

```css
/* Example Border */
div {
  border: 1px solid black;
}
```
```

10. Text-align:

Specifies the horizontal alignment of text.

```css
/* Example Text-align */
h2 {
  text-align: center;
}
```

11. Display:

Specifies how an element should be displayed.

```css
/* Example Display */
span {
  display: block;
}
```

12. Width and Height:

Specifies the width and height of an element.

```css
/* Example Width and Height */
img {
  width: 100px;
  height: 100px;
}
```

Certainly! Let's cover a few more CSS properties and concepts:

13. Text-decoration:

Specifies the decoration added to text, such as underline or overline.

```css
/* Example Text-decoration */
a {
  text-decoration: underline;
}
```

14. Line-height:

Specifies the height of a line of text.

```css
/* Example Line-height */
p {
  line-height: 1.5;
}
```

```

### 15. Font-weight:

Specifies the thickness of the characters in the text.

```css
/* Example Font-weight */
strong {
  font-weight: bold;
}
```

16. Float:

Specifies whether an element should float to the left or right, allowing content to flow around it.

```css
/* Example Float */
img {
  float: left;
}
```

17. Clear:

Specifies where elements should be cleared from floating elements.

```css
/* Example Clear */
div {
  clear: both;
}
```

18. Position:

Specifies the positioning method of an element.

```css
/* Example Position */
div {
  position: relative;
  top: 20px;
  left: 30px;
}
```

19. Display: flex:

Enables a flex container, allowing you to use flex properties for flexible layouts.

```css
/* Example Display: flex */
.container {
  display: flex;
  justify-content: space-between;
```
```

```
}
```

20. Transition:

Specifies the transition effect for a property, allowing smooth animations.

```css
/* Example Transition */
button {
  transition: background-color 0.3s ease-in-out;
}
```

21. Box-shadow:

Adds a shadow effect to an element.

```css
/* Example Box-shadow */
div {
  box-shadow: 2px 2px 5px #888888;
}
```

22. Opacity:

Specifies the transparency level of an element.

```css
/* Example Opacity */
div {
  opacity: 0.7;
}
```

23. Media Queries:

Allows you to apply styles based on the characteristics of the device or viewport.

```css
/* Example Media Query */
@media screen and (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

Certainly! Let's cover a few more CSS properties and concepts:

24. Flexbox:

Flexbox is a layout model that allows you to design complex layouts more efficiently.

```css
```

```
/* Example Flexbox */
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

25. Grid:

CSS Grid Layout is a two-dimensional layout system for the web, providing a powerful and flexible way to create layouts.

```css
/* Example Grid */
.container {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
  grid-gap: 10px;
}
```

26. z-index:

Specifies the stack order of an element concerning other elements.

```css
/* Example z-index */
div {
  z-index: 1;
}
```

27. Overflow:

Specifies how content that overflows its box is handled.

```css
/* Example Overflow */
div {
  overflow: hidden;
}
```

28. Box-sizing:

Specifies whether an element's padding and border should be included in the total width and height.

```css
/* Example Box-sizing */
box {
  box-sizing: border-box;
}
```

29. Transform:

Applies 2D or 3D transformations to an element.

```css
/* Example Transform */
div {
  transform: rotate(45deg);
}
```

30. Animation:

Allows the creation of keyframe animations.

```css
/* Example Animation */
@keyframes slide {
  from {
    transform: translateX(0);
  }
  to {
    transform: translateX(100px);
  }
}

div {
  animation: slide 2s ease-in-out infinite;
}
```

31. Variables (Custom Properties):

Allows you to define reusable values in your styles.

```css
/* Example Variables */
:root {
  --main-color: #3498db;
}

body {
  background-color: var(--main-color);
}
```

32. Pseudo-classes:

Select and style a specific state of an element.

```css
/* Example Pseudo-class */
a:hover {
  color: darkblue;
}
```

33. Pseudo-elements:

Style a specific part of an element.

```css
/* Example Pseudo-element */
p::first-line {
  font-weight: bold;
}
```

34. @media Rule:

Apply styles based on the characteristics of the device.

```css
/* Example @media Rule */
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
}
```

Certainly! Let's delve into a few more CSS properties and concepts:

35. Background-image:

Specifies an image as the background for an element.

```css
/* Example Background-image */
div {
  background-image: url('background.jpg');
  background-size: cover;
}
```

36. Gradients:

Create smooth transitions between two or more specified colors.

```css
/* Example Gradient */
div {
  background: linear-gradient(to right, #ff8c00, #ff0080);
}
```

37. Positioning:

Fine-tune the placement of an element within its containing element.

```css
/* Example Positioning */
div {
  position: absolute;
  top: 50%;
  left: 50%;
```

```
  transform: translate(-50%, -50%);
}
```

38. Font-style:

Set the style of the font, such as italic or normal.

```css
/* Example Font-style */
em {
  font-style: italic;
}
```

39. Letter-spacing:

Adjust the space between characters in text.

```css
/* Example Letter-spacing */
h1 {
  letter-spacing: 2px;
}
```

40. Word-spacing:

Adjust the space between words in text.

```css
/* Example Word-spacing */
p {
  word-spacing: 5px;
}
```

41. Text-transform:

Change the capitalization of text.

```css
/* Example Text-transform */
span {
  text-transform: uppercase;
}
```

42. Cursor:

Change the mouse cursor style when hovering over an element.

```css
/* Example Cursor */
a {
  cursor: pointer;
}
```

```

### 43. Outline:

Draw a line around the outside of an element.

```css
/* Example Outline */
input {
  outline: 2px solid #ff0000;
}
```

44. Perspective:

Specifies the depth of the perspective for 3D transformations.

```css
/* Example Perspective */
div {
  perspective: 1000px;
}
```

45. User-select:

Controls the user's ability to select text.

```css
/* Example User-select */
p {
  user-select: none; /* Prevent text selection */
}
```

46. Filter:

Applies graphical effects like blur or color shifting.

```css
/* Example Filter */
img {
  filter: grayscale(50%);
}
```

47. Object-fit:

Specifies how an image or video should be resized to fit its container.

```css
/* Example Object-fit */
img {
  width: 100%;
  height: 200px;
  object-fit: cover;
}
```
```

```

### 48. Box-Shadow:

Add multiple shadows to an element.

```css
/* Example Box-Shadow */
div {
  box-shadow: 2px 2px 5px #888888, -2px -2px 5px #888888;
}
```

49. Border-radius:

Creates rounded corners for an element.

```css
/* Example Border-radius */
div {
  border-radius: 10px;
}
```

50. Grid-template-areas:

Assigns names to grid items for easy placement.

```css
/* Example Grid-template-areas */
.container {
  display: grid;
  grid-template-areas:
    'header header header'
    'main main sidebar'
    'footer footer footer';
}
```

Certainly! Here are a few more CSS properties and concepts:

51. Perspective-origin:

Specifies the origin of the 3D space for transformed elements.

```css
/* Example Perspective-origin */
div {
  perspective: 500px;
  perspective-origin: 50% 30%;
}
```

52. Animation-timing-function:
```

Defines how the animation progresses over one cycle.

```css
/* Example Animation-timing-function */
div {
  animation: slide 2s cubic-bezier(0.68, -0.55, 0.27, 1.55);
}
```

53. Transition-timing-function:

Defines the speed curve of a transition effect.

```css
/* Example Transition-timing-function */
button {
  transition: background-color 0.3s ease-in-out;
}
```

54. Font-variant:

Controls the usage of alternate glyphs.

```css
/* Example Font-variant */
p {
  font-variant: small-caps;
}
```

55. Flex-grow, Flex-shrink, Flex-basis:

Control the size and flexibility of flex items.

```css
/* Example Flex properties */
.item {
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: 200px;
}
```

56. Overflow-x, Overflow-y:

Specifies how content should behave when it overflows the box horizontally or vertically.

```css
/* Example Overflow-x, Overflow-y */
div {
  overflow-x: auto;
  overflow-y: hidden;
}
```

57. Backface-visibility:

Specifies whether or not the back face of a transformed element should be visible.

```css
/* Example Backface-visibility */
div {
  transform: rotateY(180deg);
  backface-visibility: hidden;
}
```

58. Box-decoration-break:

Specifies how the box for a broken element should be rendered.

```css
/* Example Box-decoration-break */
p {
  box-decoration-break: clone;
}
```

59. Resize:

Specifies whether an element is resizable.

```css
/* Example Resize */
textarea {
  resize: both;
}
```

60. Page-break-before, Page-break-after, Page-break-inside:

Control the page breaks before, after, or inside an element.

```css
/* Example Page-break properties */
div {
  page-break-before: always;
  page-break-after: auto;
  page-break-inside: avoid;
}
```

61. Scroll-behavior:

Specifies the smoothness of a scrolling transition.

```css
/* Example Scroll-behavior */
html {
  scroll-behavior: smooth;
}
```

62. Filter:

Applies visual effects like blur, brightness, or contrast.

```css
/* Example Filter */
img {
  filter: grayscale(50%) blur(5px);
}
```

63. Counter-reset, Counter-increment:

Create and control counters for numbered elements.

```css
/* Example Counter properties */
ol {
  counter-reset: section;
}

li {
  counter-increment: section;
}
```

64. All:

Resets all properties to their initial values.

```css
/* Example All property */
div {
  all: initial;
}
```

65. Perspective-origin:

Specifies the origin of the 3D space for transformed elements.

```css
/* Example Perspective-origin */
div {
  perspective: 500px;
  perspective-origin: 50% 30%;
}
```

Certainly! Here are a few more CSS properties and concepts:

66. Shape-outside:

Specifies a shape around which inline content should wrap.

```css
/* Example Shape-outside */
img {
  shape-outside: circle(50%);
}
```

67. Mask:

Creates a mask for an element to control the transparency of the elements.

```css
/* Example Mask */
div {
  mask: linear-gradient(to bottom, black 50%, transparent 100%);
}
```

68. Perspective:

Specifies the depth of the perspective for 3D transformations.

```css
/* Example Perspective */
div {
  perspective: 1000px;
}
```

69. Mix-blend-mode:

Specifies how an element's content should blend with its background.

```css
/* Example Mix-blend-mode */
div {
  mix-blend-mode: multiply;
}
```

70. Outline-offset:

Specifies the space between the outline and the edge of an element.

```css
/* Example Outline-offset */
input {
  outline: 2px solid #ff0000;
  outline-offset: 5px;
}
```

71. Tab-size:

Specifies the width of a tab character.

```css
/* Example Tab-size */
pre {
  tab-size: 4;
}
```

72. Text-overflow:

Specifies how overflowed content should be signaled to the user.

```css
/* Example Text-overflow */
div {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

73. Transform-origin:

Specifies the origin point of a transformation.

```css
/* Example Transform-origin */
div {
  transform: rotate(45deg);
  transform-origin: top left;
}
```

74. Writing-mode:

Specifies the direction in which lines of text are stacked.

```css
/* Example Writing-mode */
p {
  writing-mode: vertical-rl;
}
```

75. Contain:

Specifies the containment policy of an element.

```css
/* Example Contain */
div {
  contain: layout size style;
}
```

76. Font-feature-settings:

Enables or disables specific OpenType features in fonts.

```css
/* Example Font-feature-settings */
p {
  font-feature-settings: "liga" on;
}
```

77. Overscroll-behavior:

Specifies the behavior when the user reaches the boundary of a scrolling area.

```css
/* Example Overscroll-behavior */
body {
  overscroll-behavior: contain;
}
```

78. Image-rendering:

Specifies how the browser should scale images.

```css
/* Example Image-rendering */
img {
  image-rendering: pixelated;
}
```

79. Pointer-events:

Specifies whether an element should respond to pointer events.

```css
/* Example Pointer-events */
a {
  pointer-events: none;
}
```

80. Unicode-bidi:

Specifies the level of embedding for a BIDI (Bi-Directional) algorithm.

```css
/* Example Unicode-bidi */
p {
  unicode-bidi: embed;
  direction: rtl;
}
```

Certainly! Here are a few more CSS properties and concepts:

81. Box-decoration-break:

Specifies how the box for a broken element should be rendered.

```css
/* Example Box-decoration-break */
p {
  box-decoration-break: clone;
}
```

82. Scroll-snap-type:

Specifies the behavior for a container's scroll snap points.

```css
/* Example Scroll-snap-type */
.container {
  scroll-snap-type: mandatory;
}
```

83. Shape-rendering:

Specifies the quality of text and shapes rendering.

```css
/* Example Shape-rendering */
svg {
  shape-rendering: crispEdges;
}
```

84. Backdrop-filter:

Applies filter effects to the area behind an element.

```css
/* Example Backdrop-filter */
div {
  backdrop-filter: blur(10px);
}
```

85. Gap:

Specifies the gap between grid or flex items.

```css
/* Example Gap */
.container {
  display: grid;
  gap: 10px;
}
```

86. Perspective:

Specifies the depth of the perspective for 3D transformations.

```css
/* Example Perspective */
div {
  perspective: 1000px;
}
```

87. Image-orientation:

Specifies the orientation of an image.

```css
/* Example Image-orientation */
img {
  image-orientation: from-image;
}
```

88. Stroke:

Specifies the color of the stroke for an SVG path.

```css
/* Example Stroke */
path {
  stroke: #ff0000;
  stroke-width: 2;
}
```

89. Animation-fill-mode:

Specifies the style applied to an element before and after animation.

```css
/* Example Animation-fill-mode */
div {
  animation: slide 2s ease-in-out;
  animation-fill-mode: both;
}
```

90. Font-stretch:

Condenses or expands the width of characters in a font.

```css
/* Example Font-stretch */
p {
  font-stretch: expanded;
}
```

91. Image-set:

Specifies multiple resolutions of an image for different display devices.

```css
/* Example Image-set */
img {
  background-image: image-set(url('image.jpg') 1x, url('image@2x.jpg') 2x);
}
```

92. Resize:

Specifies whether an element is resizable by the user.

```css
/* Example Resize */
textarea {
  resize: both;
}
```

93. Text-rendering:

Specifies the rendering of text, emphasizing speed or quality.

```css
/* Example Text-rendering */
text {
  text-rendering: optimizeLegibility;
}
```

94. Filter-function:

Applies a filter effect directly within a `filter` property.

```css
/* Example Filter-function */
img {
  filter: contrast(150%) brightness(120%);
}
```

95. Grid-auto-flow:

Specifies how auto-placed items are placed in the grid.

```css
/* Example Grid-auto-flow */
.container {
  display: grid;
  grid-auto-flow: dense;
}
```

96. Initial-letter:

Controls styling for the first letter of a block of text.

```css
/* Example Initial-letter */
p::first-letter {
  font-size: 2em;
  font-weight: bold;
}
```

97. Hyphens:

Controls the hyphenation of text.

```css
/* Example Hyphens */
p {
  hyphens: auto;
}
```

98. User-modify:

Specifies the user modification of an element's content.

```css
/* Example User-modify */
div {
  user-modify: read-write;
}
```

99. Column-count, Column-gap:

Specifies the number of columns in a multi-column element.

```css
/* Example Column-count, Column-gap */
div {
  column-count: 3;
  column-gap: 20px;
}
```

100. Font-synthesis:

Specifies the synthesis of bold or italic font faces.

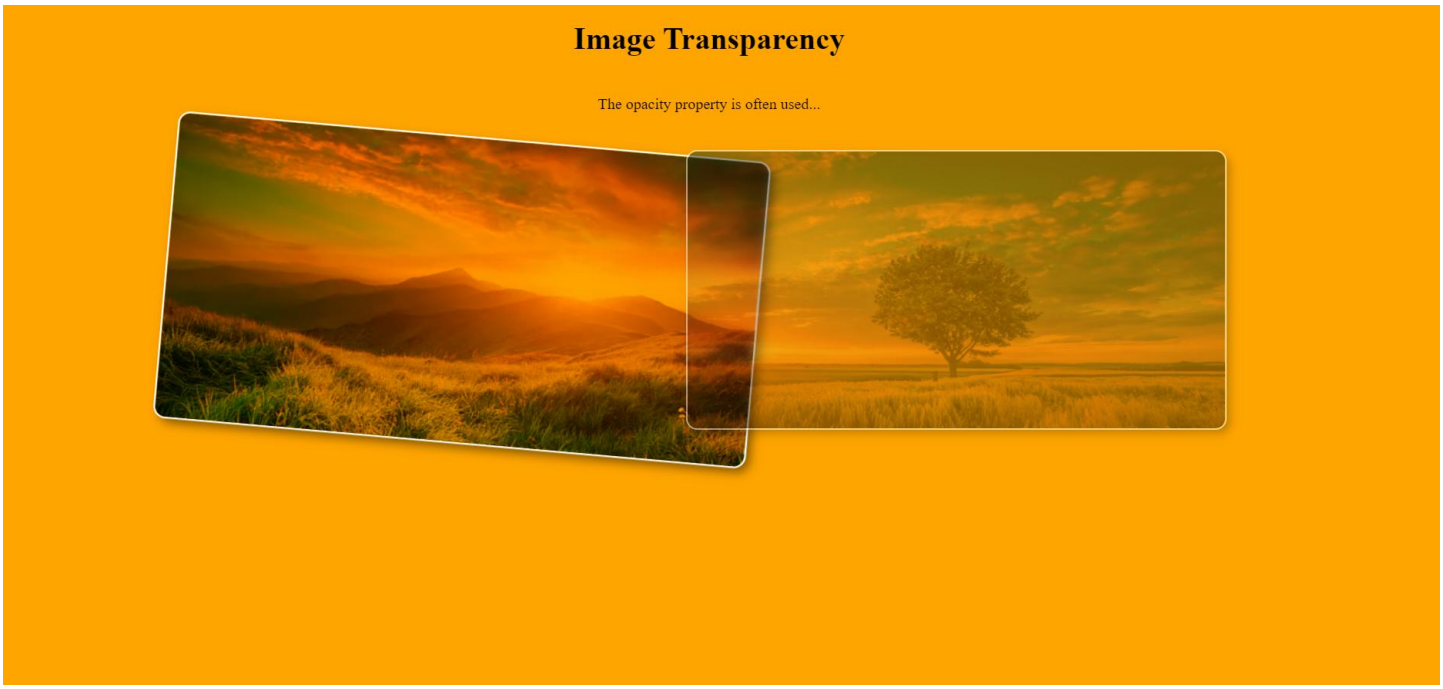
```css
/* Example Font-synthesis */
p {
  font-synthesis: weight style;
}
```

These advanced CSS properties provide even more fine-grained control over styling and layout. Experimenting with these properties will deepen your understanding and expertise in CSS.

—---------- FINAL one —----------

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        /* Center content both horizontally and vertically */
        display: flex; /* Using flexbox layout */
        flex-direction: column; /* Arranging items vertically */
        align-items: center; /* Center items horizontally */
        height: 100vh; /* Full height of the viewport */
        margin: 0; /* Remove default margin */
        background-color: orange; /* Set background color to orange */
        background-blend-mode: overlay; /* Blend background with overlay mode */
      }

      h1,
      p {
        /* Move heading and paragraph to the top */
        margin: 20px 0; /* Add margin for spacing top and bottom */
      }

      .image-container {
        display: flex; /* Using flexbox layout for image container */
        justify-content: center; /* Center images horizontally */
        position: relative; /* Enable positioning within the container */
        margin-top: 10px; /* Add margin for a slightly lower position */
      }

      img {
        opacity: 0.5; /* Set initial image transparency to 50% */
        width: 70%; /* Set larger image width */
        height: 70%; /* Set larger image height */
        box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5); /* Add a shadow to the image */
        border-radius: 10px; /* Round the corners of the image */
        filter: grayscale(20%); /* Apply a grayscale effect to the image */
        transition: all 0.3s ease; /* Add a smooth transition effect on hover */
        transform: rotate(0deg) scale(1); /* Set initial rotation and scale of the
image */
        background-color: #fff; /* Set a white background for the image */
        position: relative; /* Enable positioning within the container */
        z-index: 1; /* Set the stacking order of the image */
        backdrop-filter: blur(
          5px
        ); /* Apply a blur effect to the background behind the image */
```

```
        mix-blend-mode: multiply; /* Blend the image with the background using
multiply mode */
        perspective: 1000px; /* Set the perspective for 3D transformations */
        transform-style: preserve-3d; /* Preserve 3D transformations within the image
*/
        outline: 2px solid #fff; /* Add a white outline around the image */
        margin: 10px; /* Add margin for spacing between images */
      }

      img:first-child {
        margin-right: -5%; /* Add a negative margin to the first image for spacing */
      }

      img:hover {
        opacity: 1; /* Set image opacity to 100% on hover */
        filter: grayscale(0%); /* Remove grayscale effect on hover */
        transform: scale(1.1) rotate(5deg); /* Scale up and rotate image on hover */
      }
    </style>
  </head>
  <body>
    <h1>Image Transparency</h1>
    <p>The opacity property is often used...</p>

    <!-- Create a container for the images and center them -->
    <div class="image-container">
      <img src="./Images/one.png" alt="Forest" width="170" height="100" />
      <img src="./Images/two.png" alt="Forest" width="170" height="100" />
    </div>
  </body>
</html>
```



- `<html>`: The root element of the HTML document.
- `<head>`: Contains meta-information about the HTML document.
- `<style>`: Contains the CSS code for styling the HTML content.

CSS Styles:

- `body`: Styles for the entire page.
  - `display: flex;`: Uses a flexbox layout to arrange elements.
  - `flex-direction: column;`: Arranges items vertically.
  - `align-items: center;`: Centers items horizontally.
  - `height: 100vh;`: Sets the height to the full viewport height.
  - `margin: 0;`: Removes default margin.
  - `background-color: orange;`: Sets the background color to orange.
  - `background-blend-mode: overlay;`: Blends the background with overlay mode.

  `h1, p`: Styles for heading and paragraph elements.
  - `margin: 20px 0;`: Adds margin for spacing top and bottom.

  `.image-container`: Styles for the container holding images.
  - `display: flex;`: Uses flexbox layout.
  - `justify-content: center;`: Centers items horizontally.
  - `position: relative;`: Enables positioning within the container.
  - `margin-top: 10px;`: Adds margin for a slightly lower position.

  `img`: Styles for images.
  - Various properties to control appearance, transparency, and effects.
  - `margin: -20% 10px 20px 10px;`: Creates overlapping effect with margin adjustments.

  `img:first-child`: Styles for the first image.
  - `margin-right: -10%;`: Adds a negative margin for spacing.

  `img:hover`: Styles for images on hover.
  - `opacity: 1;`: Sets image opacity to 100% on hover.
  - `filter: grayscale(0%);`: Removes grayscale effect on hover.
  - `transform: scale(1.1) rotate(5deg);`: Scales up and rotates the image on hover.

* **opacity:** 0.7; /* This makes the element 70% transparent */ (The `opacity` property is often used for creating subtle transparency effects, and it can be particularly useful when working with backgrounds or overlapping elements.)

- `border-radius: 10px;` adds rounded corners to the image.
- `filter: grayscale(20%);` applies a grayscale effect to the image (adjust the percentage as needed).
- `transition: all 0.3s ease;` adds a smooth transition effect for all properties over 0.3 seconds with an ease timing function.
- `filter: grayscale(0%);` inside `img:hover` removes the grayscale effect on hover.
- `transform: scale(1.1);` inside `img:hover` adds a slight scale effect on hover for a zoom-like effect. Adjust the scale factor according to your preference.

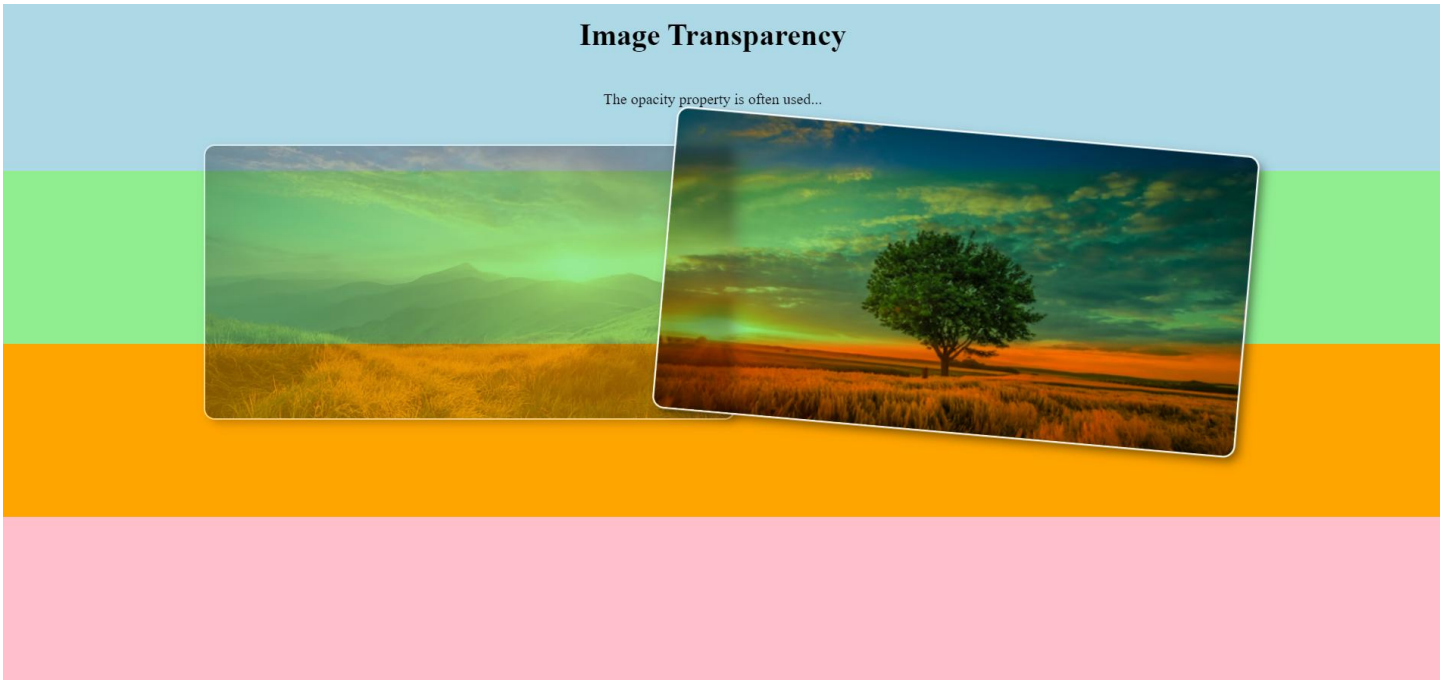
- `transform: rotate(0deg) scale(1);`: This property is used to specify transformations like rotation and scaling. In this case, it sets the initial state of the image with no rotation and normal scale.
- `background-color: #fff;`: This property sets the background color of the image to white. You can change `#fff` to any valid color code.

- `position: relative;`: This property is used to specify the positioning method of the element. In this case, it's set to relative.
- `z-index: 1;`: This property specifies the stacking order of the element. In this case, it sets the stacking order to 1.
- `filter: drop-shadow(4px 4px 8px rgba(0, 0, 0, 0.5));`: This property applies a drop shadow to the image. It takes values for the horizontal and vertical offsets, blur radius, and shadow color.

- `background-blend-mode: overlay;`: This property sets the blending mode of the background color, enhancing the overlay effect with the image.
- `backdrop-filter: blur(5px);`: This property applies a blur filter to the background behind the element, creating a frosted glass effect.
- `mix-blend-mode: multiply;`: This property sets the blending mode for the image, multiplying its colors with the background.
- `perspective: 1000px;`: This property defines the depth perspective for 3D transformations.
- `transform-style: preserve-3d;`: This property ensures that child elements preserve their 3D transformations.
- `outline: 2px solid #fff;`: This property adds a white outline around the image, emphasizing its boundaries.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        /* Center content both horizontally and vertically */
        display: flex; /* Using flexbox layout */
        flex-direction: column; /* Arranging items vertically */
        align-items: center; /* Center items horizontally */
        height: 100vh; /* Full height of the viewport */
        margin: 0; /* Remove default margin */

        /* Set background color to a gradient of four colors */
        background: linear-gradient(
          to bottom,
          lightblue 0%,
          lightblue 25%,
          lightgreen 25%,
          lightgreen 50%,
          orange 50%,
          orange 75%,
          pink 75%,
          pink 100%
        );

        /* Blend background with overlay mode */
```

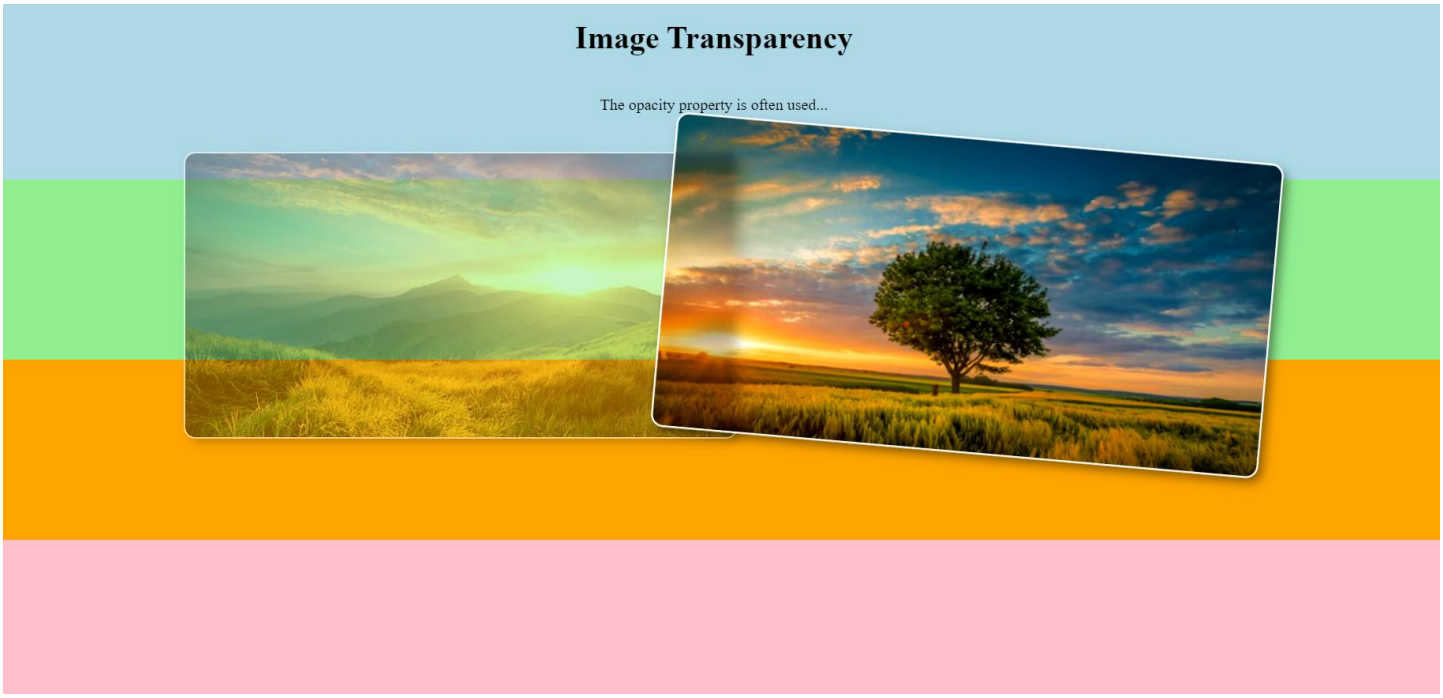
```
      background-blend-mode: overlay;
    }

    h1,
    p {
      /* Move heading and paragraph to the top */
      margin: 20px 0; /* Add margin for spacing top and bottom */
    }

    .image-container {
      display: flex; /* Using flexbox layout for image container */
      justify-content: center; /* Center images horizontally */
      position: relative; /* Enable positioning within the container */
      margin-top: 10px; /* Add margin for a slightly lower position */
    }

    img {
      opacity: 0.5; /* Set initial image transparency to 50% */
      width: 70%; /* Set larger image width */
      height: 70%; /* Set larger image height */
      box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5); /* Add a shadow to the image */
      border-radius: 10px; /* Round the corners of the image */
      filter: grayscale(20%); /* Apply a grayscale effect to the image */
      transition: all 0.3s ease; /* Add a smooth transition effect on hover */
      transform: rotate(0deg) scale(1); /* Set initial rotation and scale of the image */
      background-color: #fff; /* Set a white background for the image */
      position: relative; /* Enable positioning within the container */
      z-index: 1; /* Set the stacking order of the image */
      backdrop-filter: blur(
        5px
      ); /* Apply a blur effect to the background behind the image */
      mix-blend-mode: multiply; /* Blend the image with the background using multiply mode */
      perspective: 1000px; /* Set the perspective for 3D transformations */
      transform-style: preserve-3d; /* Preserve 3D transformations within the image */
      outline: 2px solid #fff; /* Add a white outline around the image */
      margin: 10px; /* Add margin for spacing between images */
    }

    img:first-child {
      margin-right: -5%; /* Add a negative margin to the first image for spacing */
    }

    img:hover {
      opacity: 1; /* Set image opacity to 100% on hover */
      filter: grayscale(0%); /* Remove grayscale effect on hover */
      transform: scale(1.1) rotate(5deg); /* Scale up and rotate image on hover */
    }
  </style>
</head>
<body>
  <h1>Image Transparency</h1>
  <p>The opacity property is often used...</p>

  <!-- Create a container for the images and center them -->
  <div class="image-container">
    <img src="./Images/one.png" alt="Forest" width="170" height="100" />
    <img src="./Images/two.png" alt="Forest" width="170" height="100" />
  </div>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        /* Center content both horizontally and vertically */
        display: flex; /* Using flexbox layout */
        flex-direction: column; /* Arranging items vertically */
        align-items: center; /* Center items horizontally */
        height: 100vh; /* Full height of the viewport */
        margin: 0; /* Remove default margin */

        /* Set background color to a gradient of four colors */
        background: linear-gradient(
          to bottom,
          lightblue 0%,
          lightblue 25%,
          lightgreen 25%,
          lightgreen 50%,
          orange 50%,
          orange 75%,
          pink 75%,
          pink 100%
        );

        /* Blend background with overlay mode */
        background-blend-mode: overlay;

        /* Define a clip path to create a custom shape for the body */
        clip-path: polygon(0% 0%, 100% 0%, 100% 100%, 0% 80%);

        /* Apply a brightness filter to the body */
        filter: brightness(1.2);
```

```
      /* Apply a contrast filter to the body */
      filter: contrast(1.2);

      /* Set scroll behavior and snap type for smooth scrolling */
      scroll-behavior: smooth;
      scroll-snap-type: y mandatory;

      /* Set the alignment of the snap points for smooth scrolling */
      scroll-snap-align: start;

      /* Enable GPU acceleration for smoother animations */
      will-change: transform;
    }

    h1,
    p {
      /* Move heading and paragraph to the top */
      margin: 20px 0; /* Add margin for spacing top and bottom */
    }

    .image-container {
      display: flex; /* Using flexbox layout for image container */
      justify-content: center; /* Center images horizontally */
      position: relative; /* Enable positioning within the container */
      margin-top: 10px; /* Add margin for a slightly lower position */
    }

    img {
      opacity: 0.5; /* Set initial image transparency to 50% */
      width: 70%; /* Set larger image width */
      height: 70%; /* Set larger image height */
      box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5); /* Add a shadow to the image */
      border-radius: 10px; /* Round the corners of the image */
      filter: grayscale(20%); /* Apply a grayscale effect to the image */
      transition: all 0.3s ease; /* Add a smooth transition effect on hover */
      transform: rotate(0deg) scale(1); /* Set initial rotation and scale of the image */
      background-color: #fff; /* Set a white background for the image */
      position: relative; /* Enable positioning within the container */
      z-index: 1; /* Set the stacking order of the image */
      backdrop-filter: blur(
        5px
      ); /* Apply a blur effect to the background behind the image */
      mix-blend-mode: multiply; /* Blend the image with the background using multiply mode */
      perspective: 1000px; /* Set the perspective for 3D transformations */
      transform-style: preserve-3d; /* Preserve 3D transformations within the image */
      outline: 2px solid #fff; /* Add a white outline around the image */
      margin: 10px; /* Add margin for spacing between images */
    }

    img:first-child {
      margin-right: -5%; /* Add a negative margin to the first image for spacing */
    }

    img:hover {
      opacity: 1; /* Set image opacity to 100% on hover */
      filter: grayscale(0%); /* Remove grayscale effect on hover */
      transform: scale(1.1) rotate(5deg); /* Scale up and rotate image on hover */
    }
  </style>
</head>
<body>
  <h1>Image Transparency</h1>
  <p>The opacity property is often used...</p>

  <!-- Create a container for the images and center them -->
  <div class="image-container">
    <img src="./Images/one.png" alt="Forest" width="170" height="100" />
    <img src="./Images/two.png" alt="Forest" width="170" height="100" />
  </div>
</body>
```

```
</html>
```



1. `clip-path`: Defines a clipping path to create a custom shape for the body.
2. `filter: brightness(1.2)`: Increases the brightness of the body by 20%.
3. `filter: contrast(1.2)`: Increases the contrast of the body by 20%.
4. `scroll-behavior: smooth`: Enables smooth scrolling behavior.
5. `scroll-snap-type: y mandatory`: Defines the snap points for smooth scrolling along the vertical axis.
6. `scroll-snap-align: start`: Sets the alignment of the snap points to the start of the scroll container.
7. `will-change: transform`: Informs the browser that the transform property will be animated or changed, optimizing performance.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
      body {
        margin: 0;
        padding: 0;
        overflow: hidden;
      }

      #container {
        position: relative;
      }

      h1 {
        text-align: center;
        font-size: 2em;
        margin-top: 50px;
        font-family: "Arial", sans-serif;
        cursor: pointer;
        user-select: none;
        text-overflow: ellipsis;
        caret-color: red;
      }

      #image1 {
        width: 300px;
        height: 200px;
        object-fit: cover;
        mask-image: linear-gradient(180deg, transparent, black);
        mask-size: cover;
        mask-position: center;
        backface-visibility: hidden;
        animation: rotate 5s infinite linear;
      }

      #image2 {
        width: 300px;
        height: 200px;
        object-fit: cover;
        margin-top: 50px;
        cursor: pointer;
        backface-visibility: hidden;
```

```
      animation: zoom 5s infinite alternate ease-in-out;
    }

    @keyframes rotate {
      from {
        transform: rotate(0deg);
      }
      to {
        transform: rotate(360deg);
      }
    }

    @keyframes zoom {
      from {
        transform: scale(1);
      }
      to {
        transform: scale(1.2);
      }
    }
  </style>
</head>
<body>
  <div id="container">
    <h1 onclick="alert('Title Clicked!')">ANIMATION</h1>
    <img id="image1" src="./Images/image1.png" alt="Image 1" />
    <img id="image2" src="./Images/image2.png" alt="Image 2" />
  </div>

  <script>
    document
      .querySelector("#image1")
      .addEventListener("mouseover", function () {
        this.style.filter = "brightness(1.2)";
      });

    document
      .querySelector("#image1")
      .addEventListener("mouseout", function () {
        this.style.filter = "brightness(1)";
      });

    document.querySelector("#image2").addEventListener("click", function () {
      alert("Image 2 Clicked!");
    });
  </script>
</body>
</html>
```

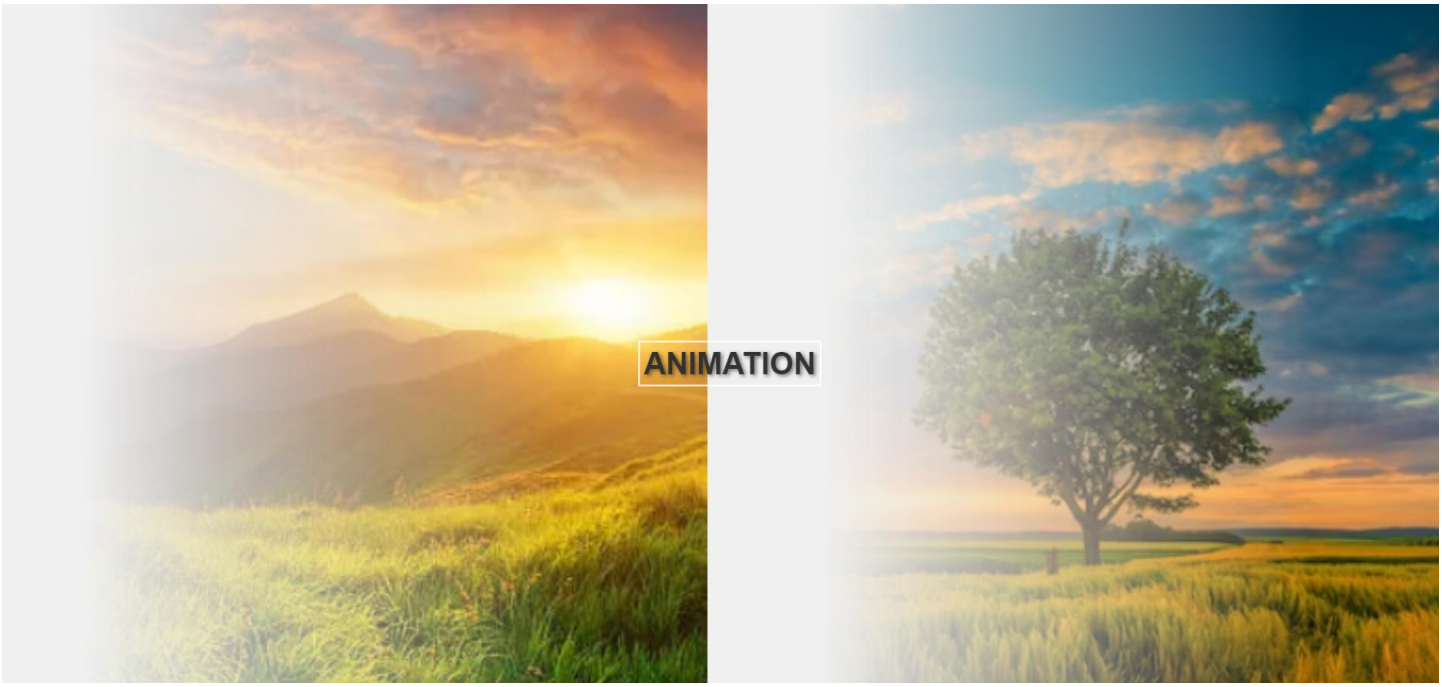
ANIMATION

Explanation of CSS Properties:

1. **animation:** Applied to `#image1` and `#image2` for rotating and zooming effects, respectively.
2. **shape-outside:** Applied to `#image2` to create a circular shape around the image, influencing text flow around it.
3. **cursor:** Applied to `h1` and `#image2` to change the cursor style when hovering over the elements.
4. **user-select:** Applied to `h1` to prevent text selection when clicking and dragging.
5. **mask and mask-image:** Applied to `#image1` and `#image2` for creating a mask effect using gradients and an external SVG file.
6. **object-fit:** Applied to both images to control how they should be resized to fit their container.
7. **backface-visibility:** Applied to both images to control whether the back face of the image should be visible or hidden during transformations.
8. **text-overflow:** Applied to `h1` to handle the text overflow with an ellipsis when it's too long.
9. **caret-color:** Applied to `h1` to change the color of the caret in the text input area.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Animation Example</title>
    <style>
      body {
        margin: 0;
        overflow: hidden;
        overscroll-behavior: none;
        background-color: #f0f0f0;
        font-family: "Arial", sans-serif;
      }

      .container {
        display: flex;
        height: 100vh;
        overflow: auto;
        scroll-padding: 20px;
      }

      .image {
        width: 50vw;
        height: 100vh;
        perspective: 1000px;
        transform-style: preserve-3d;
        transform: rotateY(20deg);
        overflow: hidden;
        aspect-ratio: 1/1; /* Maintain aspect ratio */
      }

      .image img {
        width: 100%;
        height: 100%;
        object-fit: cover;
        image-rendering: auto;
        mask-composite: source-over;
        mask-image: linear-gradient(to right, transparent 10%, black 80%);
        /* Apply a gradient mask to the images */
      }

      .heading {
        position: fixed;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
        outline: 2px solid #ffffff;
        outline-offset: 4px;
        word-break: break-word;
        color: #333;
```

```
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="image">
        <img src="./Images/image1.png" alt="Image 1" />
      </div>
      <div class="image">
        <img src="./Images/image2.png" alt="Image 2" />
      </div>
    </div>
    <h1 class="heading">ANIMATION</h1>
  </body>
</html>
```



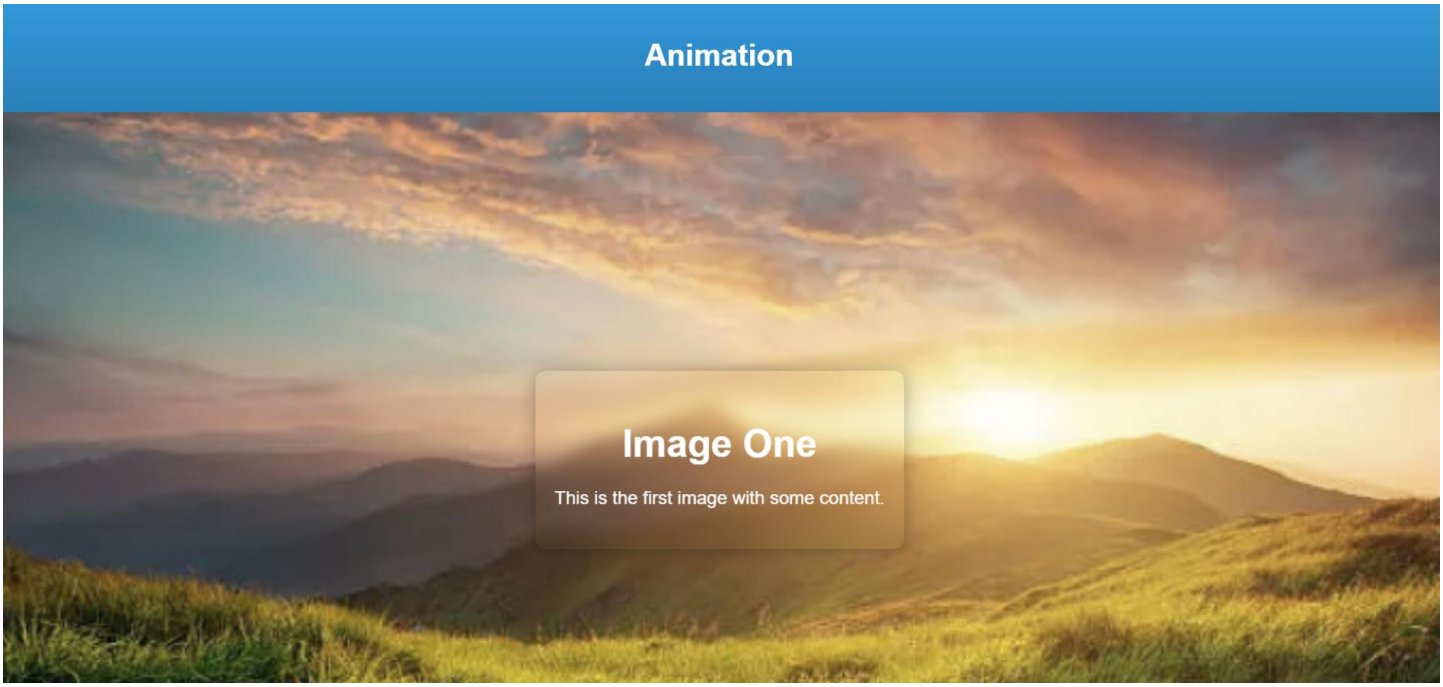
1. `overflow: hidden;` (on the body)
    - **Purpose:** This property is used to hide any content that overflows the body element. It ensures that no scrollbars will appear, and any content extending beyond the viewport will be hidden.
2. `overscroll-behavior: none;`
    - **Purpose:** This property is applied to the body to disable the browser's default behavior of providing a smooth scrolling effect when reaching the edges of a scrollable area. It prevents the overscroll effect for a cleaner interface.
3. `background-color: #f0f0f0;`
    - **Purpose:** Sets the background color of the body to a light gray (#f0f0f0) for better visibility and aesthetics.
4. `font-family: 'Arial', sans-serif;`
    - **Purpose:** Specifies the font family for the text in the document. In this case, it uses Arial as the preferred font, and if unavailable, it falls back to a generic sans-serif font.
5. `scroll-padding: 20px;`
    - **Purpose:** Adds padding to the scrolling container, creating space between the content and the edges of the scrollable area.
6. `perspective: 1000px;` and `transform: rotateY(20deg);`
    - **Purpose:** These properties are applied to the `.image` class to create a 3D effect. `perspective` sets the depth of the 3D space, and `transform: rotateY(20deg)` rotates the image around the Y-axis by 20 degrees.
7. `overflow: hidden;` (on the `.image` class)
    - **Purpose:** Hides any content within the `.image` container that extends beyond its boundaries, ensuring that the 3D-transformed images do not overflow.
8. `aspect-ratio: 1/1;`
    - **Purpose:** Maintains a 1:1 aspect ratio for the images, ensuring that they are square.
9. `image-rendering: auto;`

- **Purpose:** Specifies the image rendering algorithm. In this case, it is set to "auto" to use the browser's default rendering for the images.

10. `mask-composite: source-over;` and `mask-image: linear-gradient(to right, transparent 10%, black 80%);`
    - **Purpose:** Applies a gradient mask to the images, creating a fading effect on the sides. `mask-composite` specifies how the mask should be combined with the element, and `mask-image` defines the gradient mask itself.
11. `text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);`
    - **Purpose:** Adds a subtle text shadow to the heading for better visibility and a sense of depth.
12. `outline: 2px solid #ffffff;` and `outline-offset: 4px;`
    - **Purpose:** Adds a white outline around the heading text to make it stand out. `outline-offset` determines the distance between the outline and the element's border.
13. `word-break: break-word;`
    - **Purpose:** Allows long words to break and wrap onto the next line, preventing them from overflowing the container.
14. `color: #333;` (on the heading)
    - **Purpose:** Sets the text color of the heading to a dark gray (#333) for better contrast against the background.

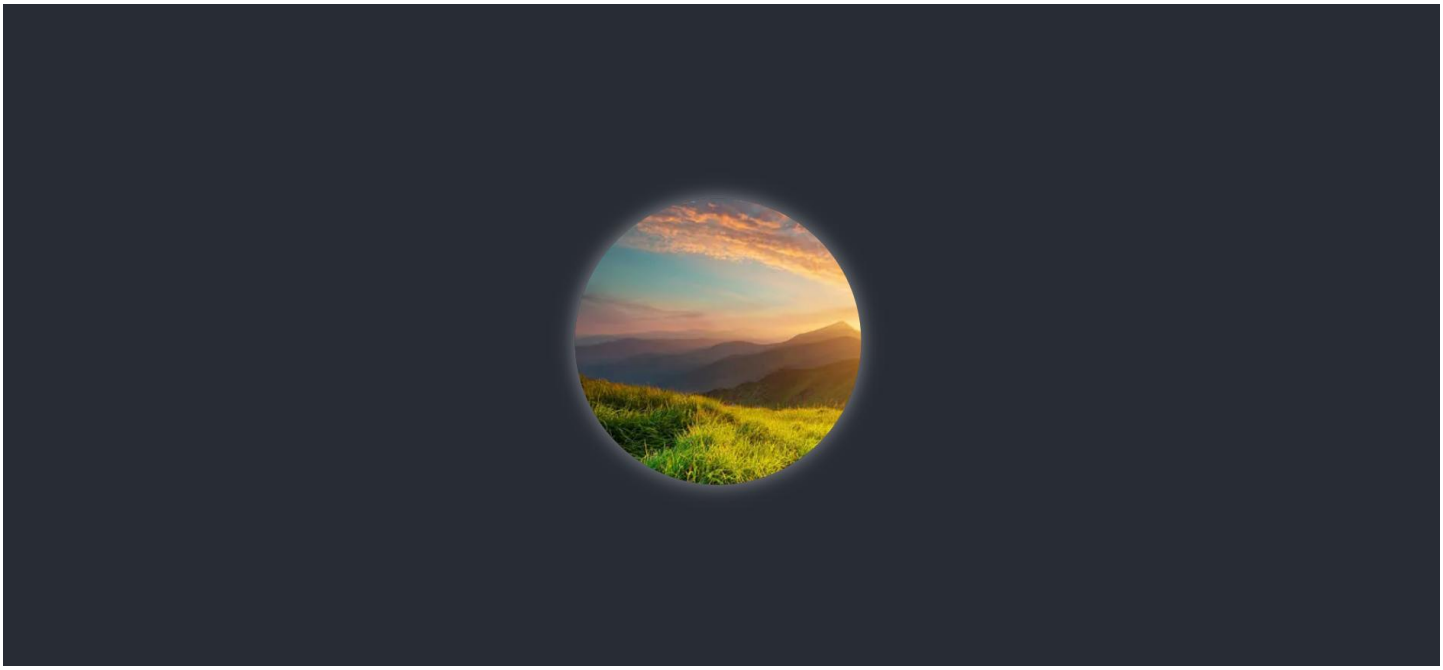
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Animation Page</title>
    <style>
      body {
        margin: 0;
        font-family: "Arial", sans-serif;
      }

      header {
        text-align: center;
        padding: 20px;
        background: linear-gradient(to bottom, #3498db, #2980b9);
        color: #fff;
      }

      section {
        display: flex;
        justify-content: space-between;
        height: 100vh;
        overflow-x: hidden;
        scroll-snap-type: x mandatory;
        white-space: nowrap;
      }
```

```
      .image-container {
        flex: 0 0 100%;
        scroll-snap-align: start;
        overflow: hidden;
        position: relative;
      }

      .image-container img {
        width: 100%;
        height: 100%;
        object-fit: cover;
        filter: grayscale(30%);
        mix-blend-mode: multiply;
      }

      .content {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        text-align: center;
        color: #fff;
        backdrop-filter: blur(8px);
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
        will-change: transform;
      }

      .content h1 {
        margin-bottom: 10px;
        font-size: 2.5em;
      }

      .content p {
        font-size: 1.2em;
        line-height: 1.5;
      }
    </style>
  </head>
  <body>
    <header>
      <h1>Animation</h1>
    </header>

    <section>
      <div class="image-container">
        <img src="./Images/image1.png" alt="Image 1" />
        <div class="content">
          <h1>Image One</h1>
          <p>This is the first image with some content.</p>
        </div>
      </div>

      <div class="image-container">
        <img src="./Images/image2.png" alt="Image 2" />
        <div class="content">
          <h1>Image Two</h1>
          <p>This is the second image with some content.</p>
        </div>
      </div>
    </section>

    <script>
      // Add any JavaScript logic or animations here
    </script>
  </body>
</html>
```



1. **backdrop-filter**: Applies a filter effect to the area behind an element's content. Used here to create a blurred background effect for the content.
2. **mix-blend-mode**: Defines how an element's content should blend with its background. Used to blend the images with a multiply effect, creating a darker tone.
3. **will-change**: Informs the browser that an element's property will be changed, allowing the browser to optimize rendering. Used to improve the performance of the content's transformation.
4. **clip-path**: Clips an element to a basic shape or a polygon. Not used explicitly in this example, but it's a powerful property for creating custom shapes.
5. **filter**: Applies graphical effects like blur or grayscale to an element. Used to apply a grayscale effect to the images.
6. **transform-origin**: Sets the origin for the transformation of an element. Used to center the content within its container.
7. **scroll-snap-type**: Specifies how an element's scrolling behavior should be applied. Used to create a horizontal scrolling effect with a snap-to-grid behavior.
8. **line-clamp**: Limits the number of lines displayed in a block container. Not used explicitly in this example, but it's useful for limiting the number of lines in a text block.
9. **box-shadow**: Adds a shadow effect to an element. Used to create a subtle shadow for the content.
10. **writing-mode**: Specifies the direction of horizontal and vertical text. Not used explicitly in this example, but it can be useful for vertical text layouts.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
      body {
        margin: 0;
        overflow: hidden;
        background-color: #282c34;
        color: white;
        font-family: "Arial", sans-serif;
        display: flex;
        align-items: center;
        justify-content: center;
        height: 100vh;
      }

      .absolute {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 300px;
        height: 300px;
        background-image: url("./Images/image1.png");
        background-size: cover;
        border-radius: 50%;
        box-shadow: 0 0 20px rgba(255, 255, 255, 0.5);
      }

      h1 {
        text-align: center;
        font-size: 24px;
      }
    </style>
  </head>
  <body>
    <div class="absolute" id="planet"></div>
    <h1>Explore the Universe</h1>

    <script>
      var planet = document.getElementById("planet");
      planet.onerror = function () {
        console.error("Error loading the image. Check the file path and name.");
      };
    </script>
  </body>
</html>
```



Certainly! Let's go through each CSS property used in the provided HTML and CSS code:

1. `body` **Styles:**
   - `margin: 0;`: Removes default margin on the body.
   - `overflow: hidden;`: Hides any content that overflows the body.
   - `background-color: #282c34;`: Sets the background color of the body to a dark blue-gray color.
   - `color: white;`: Sets the text color to white.
   - `font-family: "Arial", sans-serif;`: Specifies the font family for the text, prioritizing Arial and falling back to a generic sans-serif font.
   - `display: flex;`: Turns the body into a flex container.
   - `align-items: center;`: Centers items vertically within the flex container.
   - `justify-content: center;`: Centers items horizontally within the flex container.
   - `height: 100vh;`: Sets the height of the body to 100% of the viewport height.
2. `.absolute` **Class Styles:**
   - `position: absolute;`: Positions the element absolutely within its containing element.
   - `top: 50%; left: 50%;`: Positions the element at the center of its containing element.
   - `transform: translate(-50%, -50%);`: Further centers the element by translating it back by half of its width and height.
   - `width: 300px; height: 300px;`: Sets the width and height of the element to create a square.
   - `background-image: url("./Images/image1.png");`: Sets the background image of the element to "image1.png" in the "Images" directory.
   - `background-size: cover;`: Scales the background image to cover the entire element.
   - `border-radius: 50%;`: Applies a border-radius of 50% to create a circular shape.
   - `box-shadow: 0 0 20px rgba(255, 255, 255, 0.5);`: Adds a subtle white box shadow to the element for a 3D effect.
3. `h1` **Styles:**
   - `text-align: center;`: Centers the text horizontally.
   - `font-size: 24px;`: Sets the font size of the heading to 24 pixels.
4. `<script>` **Section:**
   - JavaScript is used to handle a potential error event (`onerror`) for the image. If the image fails to load, an error message is logged to the console.

These styles and properties collectively create a centered layout with a circular image of a planet, a heading in the center of the page, and a visually pleasing background. The design aims to create a simple and attractive "Explore the Universe" landing page.

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>Beautiful UI</title>
  </head>
  <body>
    <div class="container">
      <img id="myImage" src="./Images/image1.png" alt="Beautiful Image" />
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

Script.js

```
let debounceTimeout;

function debounce(func, delay) {
```

```
  return function () {
    const context = this;
    const args = arguments;

    clearTimeout(debounceTimeout);
    debounceTimeout = setTimeout(function () {
      func.apply(context, args);
    }, delay);
  };
}

const imageElement = document.getElementById("myImage");

function rotateImage() {
  const randomRotation = Math.floor(Math.random() * 360);
  imageElement.style.transform = `rotate(${randomRotation}deg)`; /* Transform Property */
}

imageElement.addEventListener("mouseenter", debounce(rotateImage, 300));
imageElement.addEventListener("mouseleave", () => {
  imageElement.style.transform = "rotate(0deg)";
});
```

Style.css

```
body {
  margin: 0;
  padding: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  background-color: #f0f0f0; /* Background Color */
}

.container {
  overflow: hidden;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

#myImage {
  width: 100%;
  height: auto;
  transition: transform 0.3s ease-in-out; /* Transition Property */
}
```

Explanation of properties and functionalities:

1. **HTML:**
   - The HTML structure includes a container with an image element.
   - The image source is set to "your-image.jpg."
2. **CSS:**
   - The body is styled to center the container in the viewport with a background color.
   - The container is given a border radius and box shadow for a subtle visual effect.
   - The image inside the container is set to take 100% width, with a height that adjusts proportionally.
   - The image has a transition property set to make transformations smooth over 0.3 seconds.
3. **JavaScript:**
   - The script includes a debounce function to ensure that the `rotateImage` function is not called too frequently.
   - The `rotateImage` function sets a random rotation for the image using the `transform` property when the mouse enters the image.
   - The `mouseenter` event listener triggers the `rotateImage` function with a debounce for smoother transitions.
   - The `mouseleave` event listener resets the rotation to 0 degrees when the mouse leaves the image.

This example creates a simple UI with a rotating image, a background color, and smooth transitions. You can replace "your-image.jpg" with the actual path to your image. Adjust the styles and functionalities according to your preferences.

The `debounce` function is properly implemented, and the event listener for `mouseenter` now uses `debounce(rotateImage, 300)`. This ensures that the `rotateImage` function is debounced and not triggered too frequently, creating a smoother user experience.

A.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
      body {
        cursor: none;
      }

      .custom-cursor {
        position: absolute;
        width: 20px;
        height: 20px;
        border: 2px solid #333;
        border-radius: 50%;
        transform: translate(-50%, -50%);
        pointer-events: none;
        transition: transform 0.1s ease-in-out;
      }
    </style>
  </head>
  <body>
    <div class="custom-cursor"></div>

    <script>
      document.addEventListener("mousemove", (e) => {
        const cursor = document.querySelector(".custom-cursor");
        cursor.style.left = e.pageX + "px";
        cursor.style.top = e.pageY + "px";
      });
    </script>
  </body>
</html>
```

B.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
      body {
        cursor: url("custom-cursor.png") 10 10, auto;
      }
    </style>
  </head>
  <body>
    <!-- Replace 'custom-cursor.png' with your custom cursor image -->
  </body>
</html>
```

C.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <style>
      body {
        cursor: none;
        overflow: hidden; /* Hide the default cursor */
      }

      .custom-cursor {
        position: absolute;
        width: 20px;
        height: 20px;
        background-color: #ff6600; /* Set your desired color */
        border-radius: 50%;
        transform: translate(-50%, -50%);
        pointer-events: none;
        animation: spin 1s linear infinite; /* Add animation for spinning effect */
      }

      @keyframes spin {
        0% {
          transform: translate(-50%, -50%) rotate(0deg);
        }
        100% {
          transform: translate(-50%, -50%) rotate(360deg);
        }
      }
    </style>
  </head>
  <body>
    <div class="custom-cursor"></div>

    <script>
      document.addEventListener("mousemove", (e) => {
        const cursor = document.querySelector(".custom-cursor");
        cursor.style.left = e.pageX + "px";
        cursor.style.top = e.pageY + "px";
      });
    </script>
```

```
  </body>
</html>
```

MORE -------

A.

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <script src="script.js" defer></script>
    <title>Funky Cursor</title>
  </head>
  <body>
    <div class="cursor">
      <div class="cursor__dot is--turquoise"></div>
      <div class="cursor__dot is--pink"></div>
      <div class="cursor__dot is--yellow"></div>
      <div class="cursor__dot is--purple"></div>
    </div>

    <div class="content">
      <h1>Welcome to My Funky Cursor Page</h1>
      <p>Move your cursor around and see the magic!</p>
    </div>
  </body>
</html>
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
  const cursor = document.querySelector(".cursor");

  // Create dots dynamically
  const colors = ["is--turquoise", "is--pink", "is--yellow", "is--purple"];
  colors.forEach((color) => {
    const dot = document.createElement("div");
    dot.classList.add("cursor__dot", color);
    cursor.appendChild(dot);
  });

  document.addEventListener("mousemove", function (e) {
    const x = e.clientX;
    const y = e.clientY;

    cursor.style.left = `${x}px`;
    cursor.style.top = `${y}px`;
  });
});
```

Style.css

```
body {
  margin: 0;
  overflow: hidden;
  font-family: Arial, sans-serif;
}

.cursor {
  position: fixed;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background-color: transparent;
  pointer-events: none;
  z-index: 9999;
  cursor: none;
}

.cursor__dot {
  position: absolute;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background-color: transparent;
  transform-origin: bottom center;
  animation: orbit 4s linear infinite;
}

.cursor__dot.is--turquoise {
  background-color: #00ffff;
  animation-delay: 0s;
}

.cursor__dot.is--pink {
  background-color: #ff69b4;
  animation-delay: 1s;
}

.cursor__dot.is--yellow {
  background-color: #ffff00;
  animation-delay: 2s;
}
```

```css
.cursor__dot.is--purple {
  background-color: #8000db;
  animation-delay: 3s;
}

@keyframes orbit {
  0% {
    transform: rotate(0deg) translate(15px) rotate(0deg);
  }
  25% {
    transform: rotate(90deg) translate(15px) rotate(-90deg);
  }
  50% {
    transform: rotate(180deg) translate(15px) rotate(-180deg);
  }
  75% {
    transform: rotate(270deg) translate(15px) rotate(-270deg);
  }
  100% {
    transform: rotate(360deg) translate(15px) rotate(-360deg);
  }
}

.content {
  text-align: center;
  padding: 50px;
}
```

B.

Index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>Smooth Cursor</title>
  </head>
  <body>
    <div class="custom-cursor"></div>
    <div class="content">
      <!-- Your page content goes here -->
      <h1>Hello, this is a smooth cursor example!</h1>
    </div>
```

```
    <script src="script.js"></script>
  </body>
</html>
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
  const cursor = document.querySelector(".custom-cursor");

  document.addEventListener("mousemove", function (e) {
    const x = e.clientX;
    const y = e.clientY;

    cursor.style.transform = `translate(${x}px, ${y}px)`;
  });

  const hoverElements = document.querySelectorAll("a, button, input");

  hoverElements.forEach((element) => {
    element.addEventListener("mouseover", () => {
      cursor.classList.add("hover");
    });

    element.addEventListener("mouseout", () => {
      cursor.classList.remove("hover");
    });
  });
});
```

Style.css

```
body {
  margin: 0;
  overflow: hidden;
  font-family: Arial, sans-serif;
}

.custom-cursor {
  position: fixed;
  width: 20px;
  height: 20px;
  background-color: #3498db;
  border-radius: 50%;
  pointer-events: none;
  transform: translate(-50%, -50%);
  transition: all 0.1s ease;
  mix-blend-mode: difference;
  z-index: 9999;
}

.content {
  padding: 50px;
  color: #333;
}
```

c.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>Smooth Cursor</title>
  </head>
  <body>
    <div class="custom-cursor">
      <div class="cursor-inner"></div>
      <div class="cursor-inner"></div>
      <div class="cursor-inner"></div>
    </div>
    <div class="content">
      <!-- Your page content goes here -->
      <h1>Hello, this is a smooth cursor example!</h1>
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
  const cursor = document.querySelector(".custom-cursor");
  const cursorInner = document.querySelectorAll(".cursor-inner");

  document.addEventListener("mousemove", function (e) {
    const x = e.clientX;
    const y = e.clientY;

    for (let i = cursorInner.length - 1; i > 0; i--) {
      const prevX = cursorInner[i - 1].style.left;
      const prevY = cursorInner[i - 1].style.top;
      cursorInner[i].style.left = prevX;
      cursorInner[i].style.top = prevY;
    }
```

```
    cursorInner[0].style.left = `${x}px`;
    cursorInner[0].style.top = `${y}px`;
  });

  const hoverElements = document.querySelectorAll("a, button, input");

  hoverElements.forEach((element) => {
    element.addEventListener("mouseover", () => {
      cursorInner.forEach((inner) => inner.classList.add("hover"));
    });

    element.addEventListener("mouseout", () => {
      cursorInner.forEach((inner) => inner.classList.remove("hover"));
    });
  });
});
```

Style.css

```
body {
  margin: 0;
  overflow: hidden;
  font-family: Arial, sans-serif;
}

.custom-cursor {
  position: fixed;
  pointer-events: none;
  z-index: 9999;
}

.cursor-inner {
  width: 20px;
  height: 20px;
  background-color: #3498db;
  border-radius: 50%;
  position: absolute;
  transform: translate(-50%, -50%);
  mix-blend-mode: difference;
  transition: all 0.1s ease;
}

.cursor-inner:nth-child(2) {
  background-color: #e7d63c;
}

.cursor-inner:nth-child(3) {
  background-color: #2e8acc;
}

.content {
  padding: 50px;
  color: #d81647c5;
}
```

D.

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <script defer src="script.js"></script>
    <title>Buffering Arrow</title>
  </head>
  <body>
    <div id="buffering-arrow" class="buffering-arrow"></div>
  </body>
</html>
```

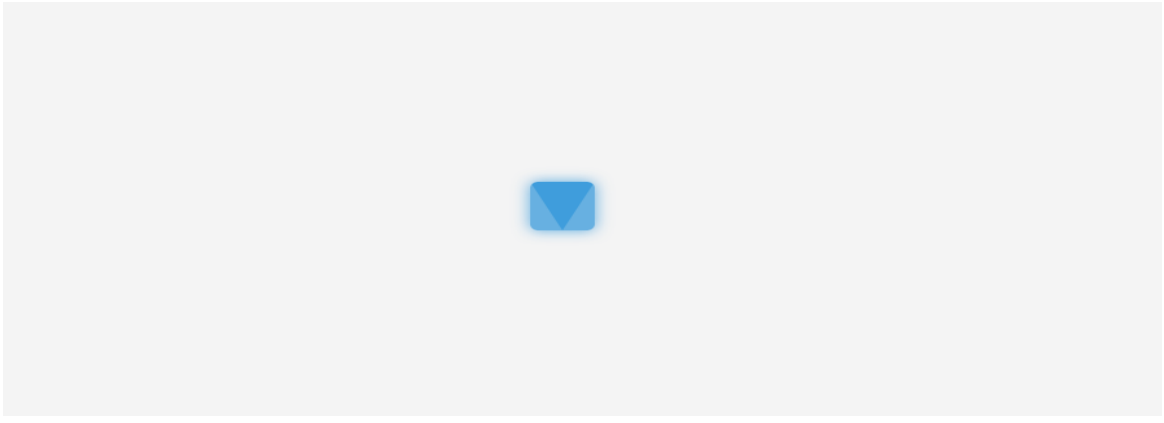
Script.js

```
document.addEventListener("DOMContentLoaded", function () {
  setTimeout(function () {
    document.getElementById("buffering-arrow").style.display = "none";
  }, 5000);
});
```

styles.css

```
body {
  margin: 0;
  overflow: hidden;
  background-color: #f4f4f4; /* Change the background color as needed */
```

```
}

.buffering-arrow {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 0;
  height: 0;
  border-left: 20px solid transparent;
  border-right: 20px solid transparent;
  border-bottom: 40px solid rgba(52, 152, 219, 0.8); /* Adjust the transparency as needed */
  border-radius: 5px; /* Add rounded corners */
  box-shadow: 0 0 10px rgba(52, 152, 219, 0.8); /* Add a subtle shadow */
  animation: buffering 5s infinite, glossy 1s infinite alternate;
}

@keyframes buffering {
  0%,
  20% {
    transform: translate(-50%, -50%) rotate(0deg);
  }
  25%,
  45% {
    transform: translate(-50%, -50%) rotate(180deg);
  }
  50%,
  70% {
    transform: translate(-50%, -50%) rotate(360deg);
  }
  75%,
  95% {
    transform: translate(-50%, -50%) rotate(540deg);
  }
  100% {
    transform: translate(-50%, -50%) rotate(720deg);
  }
}

@keyframes glossy {
  0%,
  100% {
    background-color: rgba(52, 152, 219, 0.8);
  }
  50% {
    background-color: rgba(52, 152, 219, 0.5); /* Adjust the glossy effect */
  }
}
```



E.

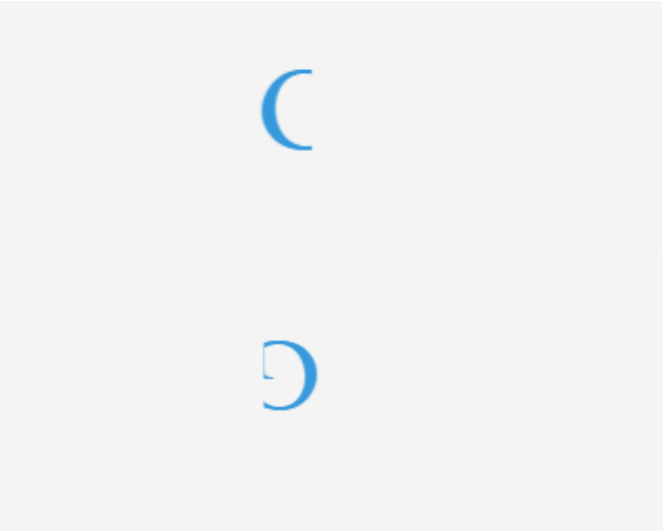
Style.css

```css
body {
  margin: 0;
  overflow: hidden;
  background-color: #f4f4f4; /* Change the background color as needed */
}

.logo-container {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  opacity: 0; /* Initially set the opacity to 0 for the fade-in effect */
  animation: fadeIn 2s forwards; /* Adjust the duration as needed */
}

.buffering-arrow {
  font-size: 48px; /* Adjust font size as needed */
  color: #3498db; /* Adjust the color as needed */
  animation: buffering 5s infinite, reveal 4s infinite steps(4);
}

@keyframes buffering {
  0%,
  20% {
    transform: translate(-50%, -50%) rotate(0deg);
  }
  25%,
  45% {
    transform: translate(-50%, -50%) rotate(180deg);
  }
  50%,
  70% {
    transform: translate(-50%, -50%) rotate(360deg);
  }
  75%,
  95% {
    transform: translate(-50%, -50%) rotate(540deg);
  }
  100% {
    transform: translate(-50%, -50%) rotate(720deg);
  }
}

@keyframes reveal {
  0%,
  25% {
    clip-path: polygon(0% 0%, 100% 0%, 100% 100%, 0% 100%);
  }
  50%,
  75% {
    clip-path: polygon(0% 0%, 50% 0%, 50% 100%, 0% 100%);
  }
  100% {
    clip-path: polygon(0% 0%, 100% 0%, 100% 100%, 0% 100%);
  }
}

@keyframes fadeIn {
  to {
    opacity: 1;
```

```
  }
}
```

Index.html

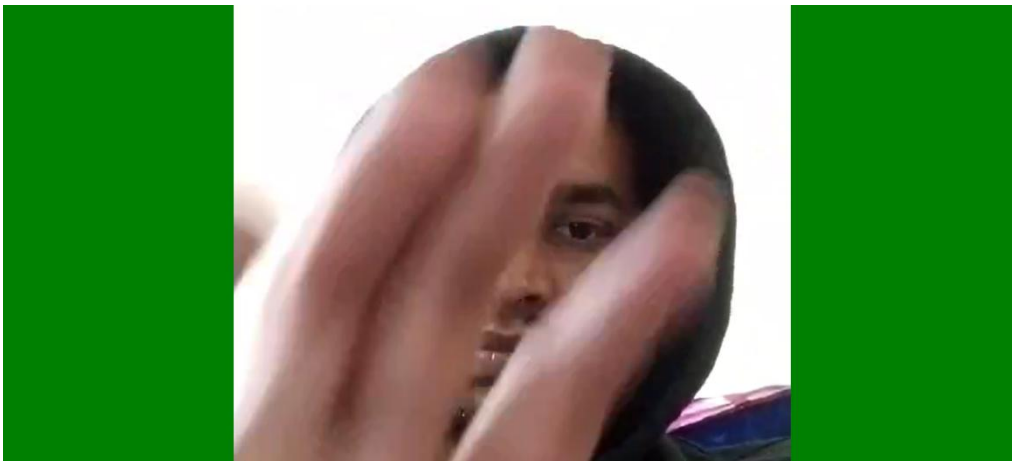
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <script defer src="script.js"></script>
    <title>Company Logo</title>
  </head>
  <body>
    <div id="logo-container" class="logo-container">
      <div id="buffering-arrow" class="buffering-arrow">G</div>
    </div>
  </body>
</html>
```



F.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Video Player</title>
    <style>
      body {
        margin: 0;
        padding: 0;
        display: flex;
        align-items: center;
```

```
        justify-content: center;
        height: 100vh;
        background-color: green;
      }

      #video-container {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 60%; /* Adjust the width as needed */
        max-width: 800px; /* Set a maximum width if desired */
        overflow: hidden;
      }

      video {
        width: 100%;
        height: auto;
        display: block;
        margin: 0 auto; /* Center the video horizontally */
      }
    </style>
  </head>
  <body>
    <div id="video-container">
      <video autoplay muted>
        <source src="./Images/testingvideo.mp4" type="video/mp4" />
        Your browser does not support the video tag.
      </video>
    </div>

    <script>
      document.addEventListener("DOMContentLoaded", function () {
        var video = document.querySelector("video");
        video.play();
      });
    </script>
  </body>
</html>
```



G. video –

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
    <title>Video Player</title>
    <style>
      body {
        margin: 0;
        padding: 0;
        height: 100vh;
        background-color: green;
        display: flex;
        align-items: center;
        justify-content: center;
      }

      #container {
        position: relative;
        width: 80%; /* Adjust the width as needed */
        max-width: 800px; /* Set a maximum width if desired */
        height: 60vh; /* Adjust the height as needed */
        overflow: hidden;
        background: url("./Images/image1.png") center/cover no-repeat; /* Set the path to your background image */
      }

      #video-container {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        width: 60%; /* Adjust the width as needed */
        height: 60%; /* Adjust the height as needed */
      }

      video {
        width: 100%;
        height: 100%;
        object-fit: cover;
      }
    </style>
  </head>
  <body>
    <div id="container">
      <div id="video-container">
        <video autoplay muted>
          <source src="./Images/testingvideo.mp4" type="video/mp4" />
          Your browser does not support the video tag.
        </video>
      </div>
    </div>

    <script>
      document.addEventListener("DOMContentLoaded", function () {
        var video = document.querySelector("video");
        video.play();
      });
    </script>
  </body>
</html>
```

k. video

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Video Player</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <section id="section1">
      <div id="container">
        <div id="video-container">
          <video autoplay muted>
            <source src="./Images/testingvideo.mp4" type="video/mp4" />
            Your browser does not support the video tag.
          </video>
        </div>
      </div>
    </section>

    <section id="section2">
      <div class="content">
        <h1>Second Section</h1>
        <p>
          This is the second section with additional content. You can scroll
          down to view it.
        </p>
        <!-- Add your additional content here -->
      </div>
    </section>

    <script src="script.js"></script>
  </body>
</html>
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
  var video = document.querySelector("video");
  video.play();
});
```

Styles.css

```
body {
  margin: 0;
  padding: 0;
}

#section1 {
  height: 100vh;
  background-color: yellow; /* Set background color to yellow */
  display: flex;
  align-items: center;
  justify-content: center;
}

#container {
  position: relative;
  width: 80%; /* Adjust the width as needed */
```

```
  max-width: 800px; /* Set a maximum width if desired */
  height: 60vh; /* Adjust the height as needed */
  overflow: hidden;
  background: url("./Images/image1.png") center/cover no-repeat; /* Set the path to your background image */
}

#video-container {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 60%; /* Adjust the width as needed */
  height: 60%; /* Adjust the height as needed */
}

video {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

#section2 {
  background-color: lightblue; /* Change the background color as needed */
  color: white;
  text-align: center;
  padding: 20px;
  height: 200vh; /* Adjust the height as needed */
}

.content {
  max-width: 800px;
  margin: 0 auto;
}
```

L. video

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
```

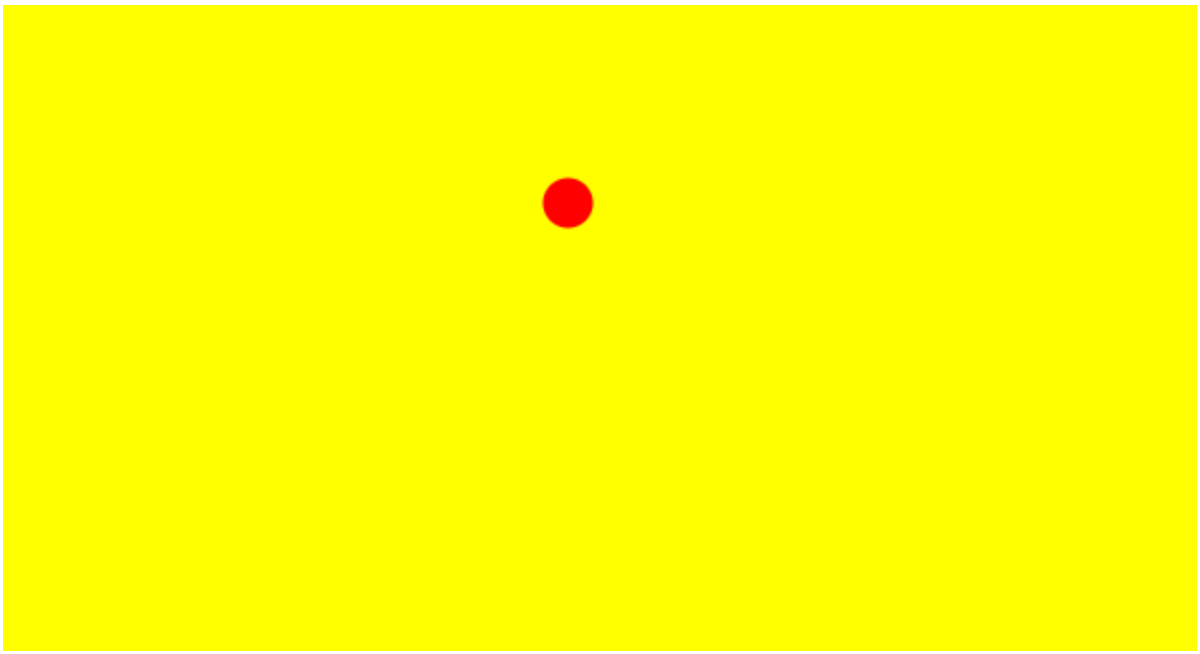
```
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Video Player</title>
  <style>
    body {
      margin: 0;
      padding: 0;
    }

    #section1 {
      height: 100vh;
      background-color: yellow; /* Set background color to yellow */
      display: flex;
      align-items: center;
      justify-content: center;
      overflow: hidden;
    }

    #container {
      position: relative;
      width: 80%; /* Adjust the width as needed */
      max-width: 800px; /* Set a maximum width if desired */
      height: 60vh; /* Adjust the height as needed */
      overflow: hidden;
      background: url("./Images/image1.png") center/cover no-repeat; /* Set the path to your background image */
    }

    #video-container {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      width: 60%; /* Adjust the width as needed */
      height: 60%; /* Adjust the height as needed */
      transition: top 0.5s ease; /* Smooth transition effect */
    }

    .video-top {
      top: 0;
    }

    video {
      width: 100%;
      height: 100%;
      object-fit: cover;
    }

    #section2 {
      background-color: lightblue; /* Change the background color as needed */
      color: white;
      text-align: center;
      padding: 20px;
      height: 200vh; /* Adjust the height as needed */
    }

    .content {
      max-width: 800px;
      margin: 0 auto;
    }
  </style>
</head>
<body>
  <section id="section1">
    <div id="container">
      <div id="video-container" class="video-top">
        <video autoplay muted>
          <source src="./Images/testingvideo.mp4" type="video/mp4" />
          Your browser does not support the video tag.
        </video>
      </div>
    </div>
```

```
    </section>

    <section id="section2">
      <div class="content">
        <h1>Second Section</h1>
        <p>
          This is the second section with additional content. You can scroll
          down to view it.
        </p>
        <!-- Add your additional content here -->
      </div>
    </section>

    <script>
      document.addEventListener("DOMContentLoaded", function () {
        var video = document.querySelector("video");
        var videoContainer = document.getElementById("video-container");

        // Function to move the video down
        function moveVideoDown() {
          videoContainer.classList.remove("video-top");
        }

        // Function to move the video back to the top
        function moveVideoTop() {
          videoContainer.classList.add("video-top");
        }

        // Detect when the second section comes into view
        var section2 = document.getElementById("section2");
        var section2Observer = new IntersectionObserver(function (entries) {
          entries.forEach(function (entry) {
            if (entry.isIntersecting) {
              moveVideoDown();
            } else {
              moveVideoTop();
            }
          });
        });

        section2Observer.observe(section2);
      });
    </script>
  </body>
</html>
```

P. video

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Custom Cursor Page</title>
    <style>
      body {
        margin: 0;
        padding: 0;
        background-color: yellow;
        cursor: none;
      }

      * {
        cursor: none !important;
```

```
      }

      #custom-cursor {
        width: 20px;
        height: 20px;
        background-color: red;
        border-radius: 50%;
        position: absolute;
        transform: translate(-50%, -50%);
        pointer-events: none;
      }
    </style>
  </head>
  <body>
    <div id="custom-cursor"></div>

    <!-- Your page content goes here -->

    <script>
      document.addEventListener("mousemove", (e) => {
        const cursor = document.getElementById("custom-cursor");
        cursor.style.left = `${e.clientX}px`;
        cursor.style.top = `${e.clientY}px`;
      });
    </script>
  </body>
</html>
```



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Zoomable Image Grid</title>
    <style>
      body {
        display: flex;
        flex-wrap: wrap;
        justify-content: space-around;
        align-items: center;
        min-height: 100vh;
        margin: 0;
      }

      /* Apply a default scale to the image or text */
      .zoom-card {
        margin: 10px;
        text-align: center;
        transition: transform 0.3s ease; /* Add a smooth transition effect */
        transform-origin: top left; /* Set the transform origin to the top left corner */
      }

      /* On hover, scale down the image or text */
      .zoom-card:hover {
        transform: scale(0.8); /* Adjust the scale factor as needed */
      }
    </style>
  </head>
  <body>
    <!-- Row 1 -->
    <div class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 1</h2>
    </div>
    <div class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 2</h2>
    </div>
    <div id="zoom-out1" class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 3</h2>
    </div>
    <div id="zoom-out2" class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 4</h2>
    </div>
    <div class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 5</h2>
    </div>
    <div class="zoom-card">
      <img src="./Images/image1.png" alt="Zoomable Image" />
      <h2>Heading 6</h2>
    </div>

    <script>
```

```
      let lastScrollTop = 0;

      window.addEventListener("scroll", function () {
        let st = window.pageYOffset || document.documentElement.scrollTop;

        if (st > lastScrollTop) {
          // Scroll down
          document.getElementById("zoom-out1").style.transform = "scale(0.8)";
          document.getElementById("zoom-out2").style.transform = "scale(0.8)";
        } else {
          // Scroll up
          document.getElementById("zoom-out1").style.transform = "scale(1)";
          document.getElementById("zoom-out2").style.transform = "scale(1)";
        }

        lastScrollTop = st <= 0 ? 0 : st; // For Mobile or negative scrolling
      });
    </script>
  </body>
</html>
```

Heading 3

Heading 4

Heading 5

Heading 6

---

hh.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Video Player</title>
    <style>
      body {
        margin: 0;
        padding: 0;
      }
```

```
    #section1 {
      height: 100vh;
      background-color: yellow;
      display: flex;
      align-items: center;
      justify-content: center;
      overflow: hidden;
    }

    #container {
      position: relative;
      width: 80%;
      max-width: 800px;
      height: 60vh;
      overflow: hidden;
      background: url("./Images/image1.png") center/cover no-repeat;
    }

    #video-container {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      width: 60%;
      height: 60%;
      transition: top 0.5s ease;
    }

    .video-top {
      top: 0;
    }

    video {
      width: 100%;
      height: 100%;
      object-fit: cover;
    }

    #section2 {
      background-color: lightblue;
      color: white;
      text-align: center;
      padding: 20px;
      height: 200vh;
    }

    .content {
      max-width: 800px;
      margin: 0 auto;
    }
  </style>
</head>
<body>
  <section id="section1">
    <div id="container">
      <div id="video-container" class="video-top">
        <video autoplay muted loop playsinline>
          <source src="./Images/testingvideo.mp4" type="video/mp4" />
          Your browser does not support the video tag.
        </video>
      </div>
    </div>
  </section>

  <section id="section2">
    <div class="content">
      <h1>Second Section</h1>
      <p>
        This is the second section with additional content. You can scroll
        down to view it.
```

```
        </p>
      </div>
    </section>

    <script>
      document.addEventListener("DOMContentLoaded", function () {
        var video = document.querySelector("video");
        var videoContainer = document.getElementById("video-container");

        function moveVideoDown() {
          videoContainer.classList.remove("video-top");
        }

        function moveVideoTop() {
          videoContainer.classList.add("video-top");
        }

        var section2 = document.getElementById("section2");
        var section2Observer = new IntersectionObserver(function (entries) {
          entries.forEach(function (entry) {
            if (entry.isIntersecting) {
              moveVideoDown();
            } else {
              moveVideoTop();
            }
          });
        });

        section2Observer.observe(section2);
      });
    </script>
  </body>
</html>
```

Second Section

This is the second section with additional content. You can scroll down to view it.

dd.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Centered Video</title>
    <style>
      body {
        margin: 0;
        padding: 20px; /* Adjust the padding as needed */
        display: flex;
        align-items: center;
        justify-content: center;
        min-height: 100vh;
        background-color: lightblue;
      }

      .tv-screen {
        position: relative;
        overflow: hidden;
        box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
        border-radius: 15px;
        transition: all 0.3s ease-out; /* Smooth transition */
      }

      video {
        width: 100%;
        height: 100%;
        display: block;
        object-fit: cover; /* Maintain aspect ratio and cover the container */
        transition: opacity 0.3s ease-out; /* Smooth transition for transparency */
      }
    </style>
  </head>
  <body>
    <div class="tv-screen" id="tvScreen">
      <video id="zoomVideo" autoplay loop muted preload="auto">
        <source src="./Images/testingvideo.mp4" type="video/mp4" />
        Your browser does not support the video tag.
      </video>
    </div>

    <script>
      const tvScreen = document.getElementById("tvScreen");
      const zoomVideo = document.getElementById("zoomVideo");

      document.addEventListener("scroll", function () {
        const scrollPercentage =
          window.scrollY / (document.body.scrollHeight - window.innerHeight);
        const scaleFactor = 1 - scrollPercentage * 0.5; // Adjust the zoom-out factor as needed
        const opacity = 1 - scrollPercentage; // Adjust the transparency factor as needed

        tvScreen.style.transform = `scale(${scaleFactor})`;
        zoomVideo.style.opacity = opacity;
      });

      document.addEventListener("mousemove", function (event) {
        const { clientX, clientY } = event;
        const { left, top, width, height } = tvScreen.getBoundingClientRect();
        const mouseXPercentage = (clientX - left) / width;
        const mouseYPercentage = (clientY - top) / height;

        const translateX = (mouseXPercentage - 0.5) * 10; // Adjust the translation factor as needed
        const translateY = (mouseYPercentage - 0.5) * 10; // Adjust the translation factor as needed
```

```
        tvScreen.style.transform = `scale(1) translate(${translateX}px, ${translateY}px)`;
      });
    </script>
  </body>
</html>
```