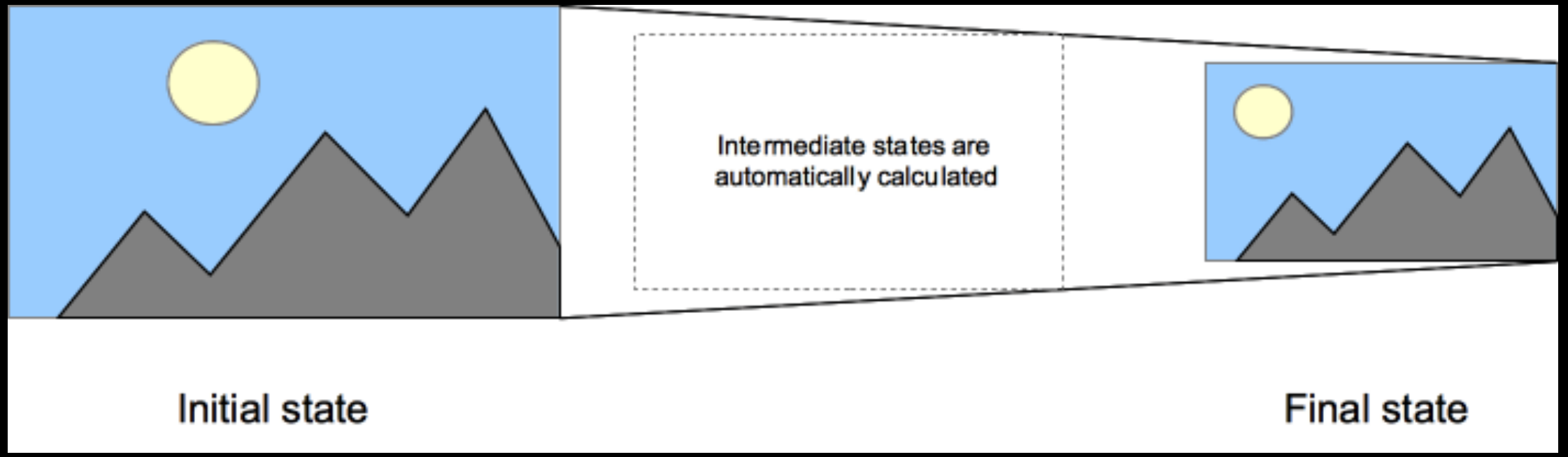
Transitions

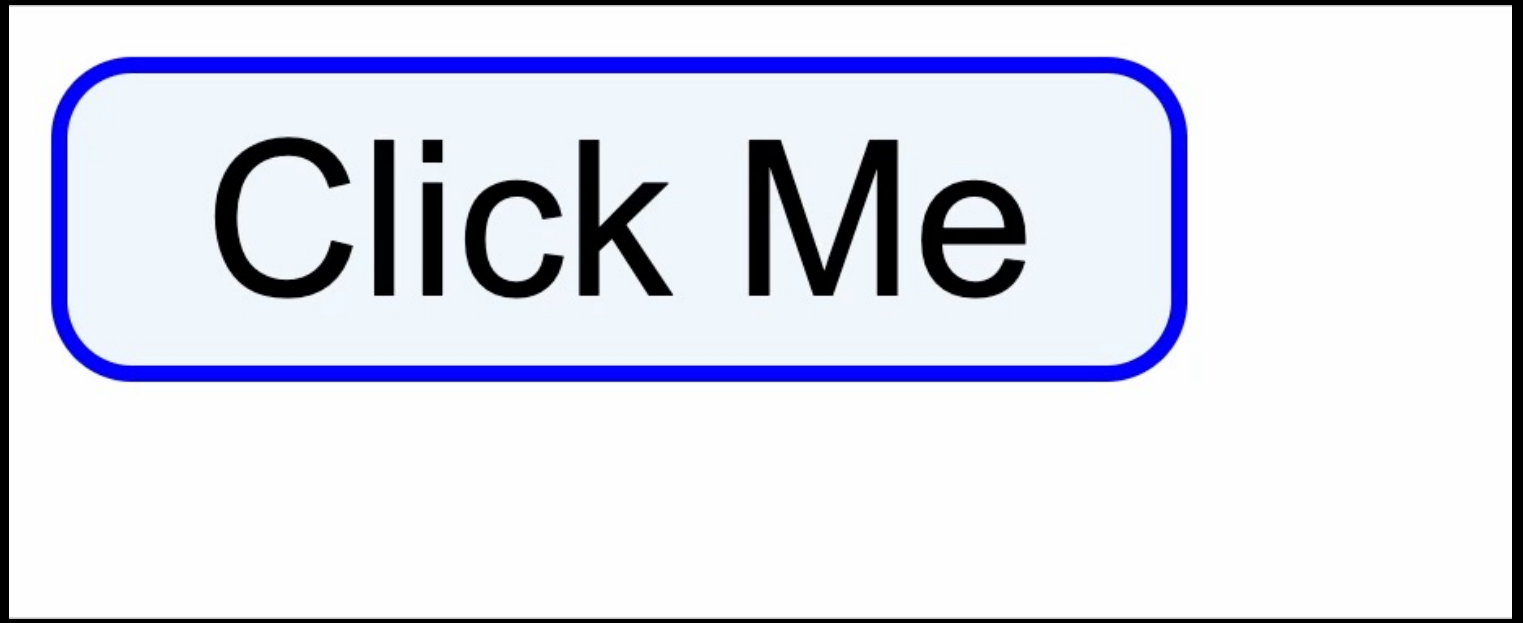


CSS transition is a property that enables smooth animation between changes in CSS property values

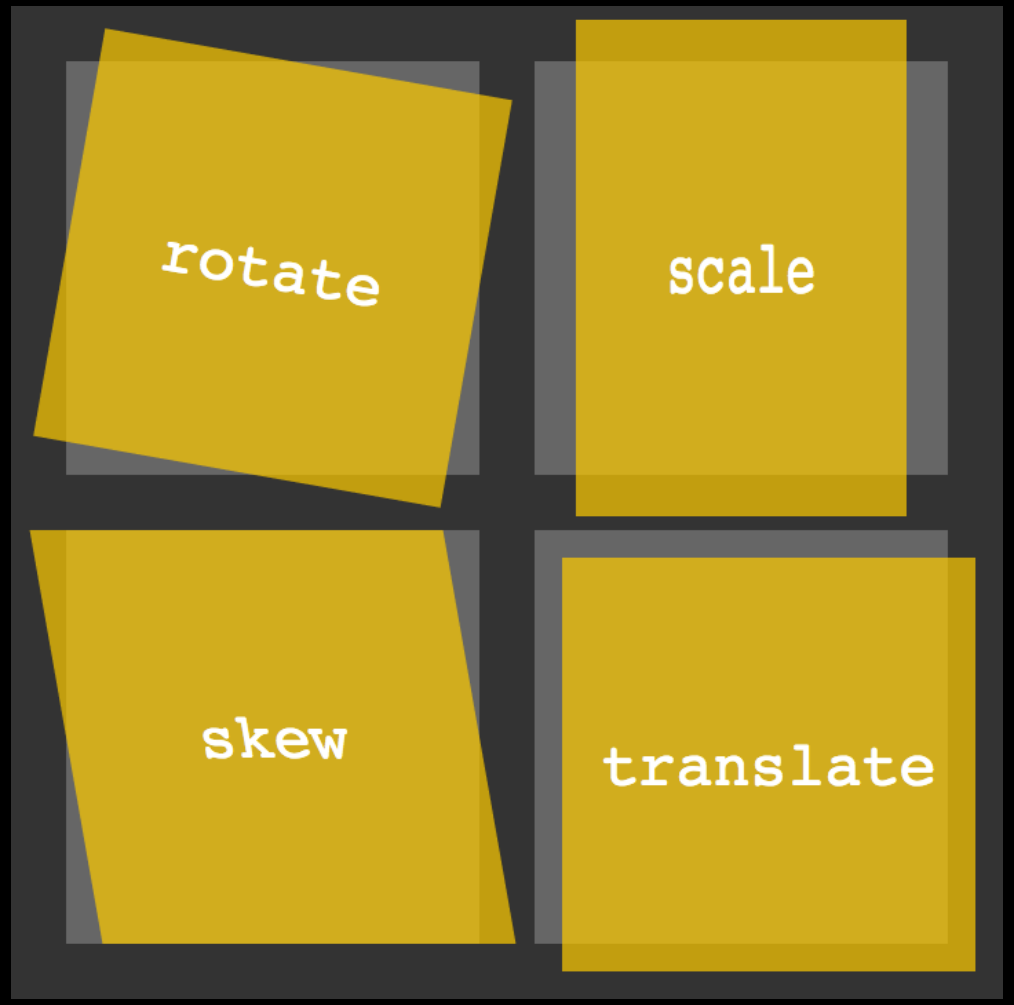
- **transition-property**: Defines which CSS properties will transition.
- **transition-duration**: Sets how long the transition lasts.
- **transition-timing-function**: Controls the speed curve of the transition.
- **transition-delay**: Specifies a delay before the transition starts.

Transitions

```css
.btn {
    height: 20px;
    width: 70px;
    border: 1px solid blue;
    border-radius: 5px;
    background-color: aliceblue;
    transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out;
    transition-delay: 1s;
    /*transition: all 1s ease-in-out 1s;*/
}
.btn:hover {
    height: 25px;
    width: 80px;
    border: 1px solid red;
}
.btn:active {
    height: 25px;
    width: 80px;
    border: 1px solid red;
    background-color: indianred;
}
```



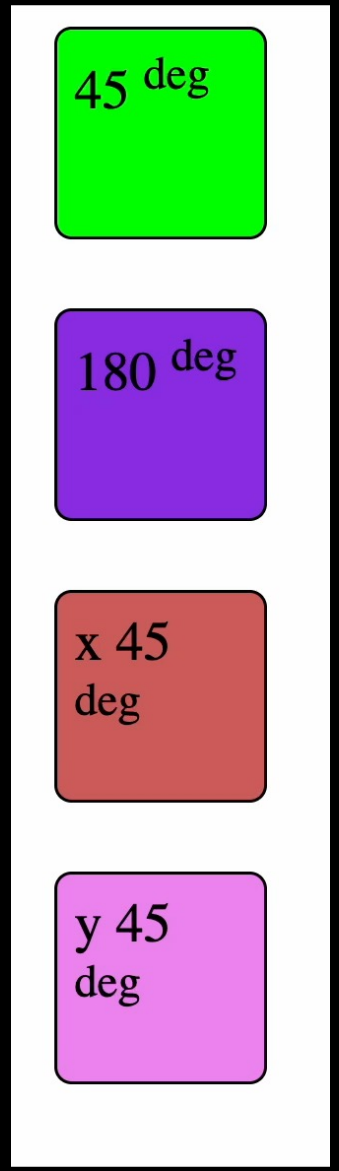
CSS Transform



- Allows modification of an element's shape and position.
- Can perform operations like rotate, scale, and translate.
- Does not affect the layout of surrounding elements.
- Used to create visual effects like 3D space transformations.
- Implemented with functions like rotate(), scale(), and translate().

CSS Transform (Rotate)

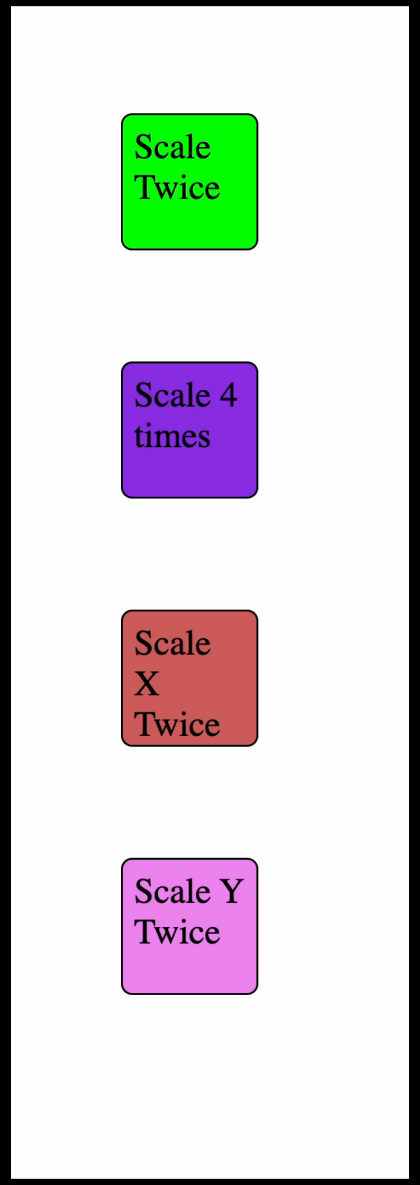
```css
.box {
    height: 50px;
    width: 50px;
    padding: 5px;
    margin: 20px;
    border: 1px solid black;
    border-radius: 5px;
    transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out;
}
#box1 { background-color: lime;}
#box1:hover { transform: rotate(45deg); }
#box2 { background-color: blueviolet;}
#box2:hover { transform: rotate(180deg); }
#box3 { background-color: indianred;}
#box3:hover { transform: rotatex(45deg); }
#box4 { background-color: violet;}
#box4:hover { transform: rotatey(45deg); }
```



- Rotates an element around a fixed point.
- Defined using the rotate() function within the transform property.
- Default rotation point is the element's center.

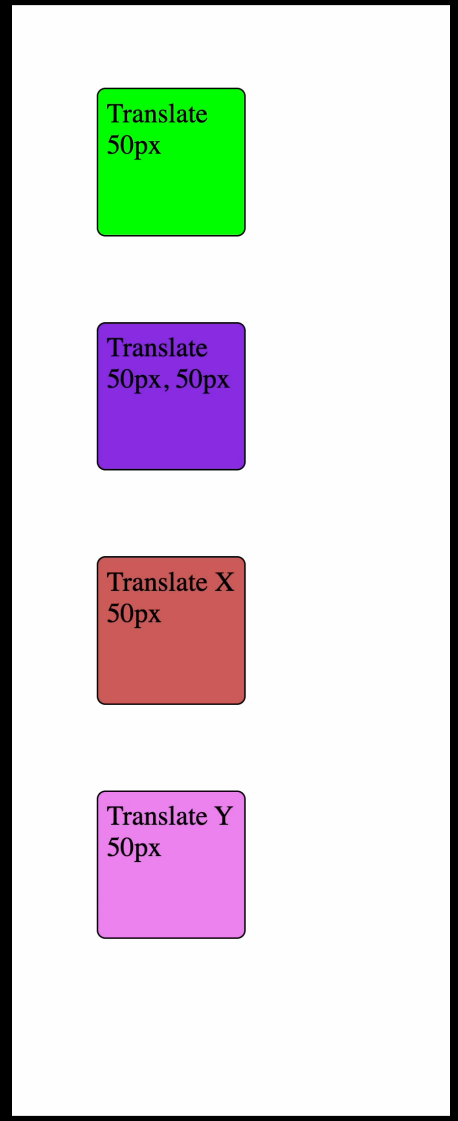
CSS Transform (Scale)

```
.box {
    height: 50px;
    width: 50px;
    padding: 5px;
    margin: 50px;
    border: 1px solid black;
    border-radius: 5px;
    transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out;
}
#box1 { background-color: lime;}
#box1:hover { transform: scale(2); }
#box2 { background-color: blueviolet;}
#box2:hover { transform: scale(4); }
#box3 { background-color: indianred;}
#box3:hover { transform: scalex(2); }
#box4 { background-color: violet;}
#box4:hover { transform: scaley(3); }
```



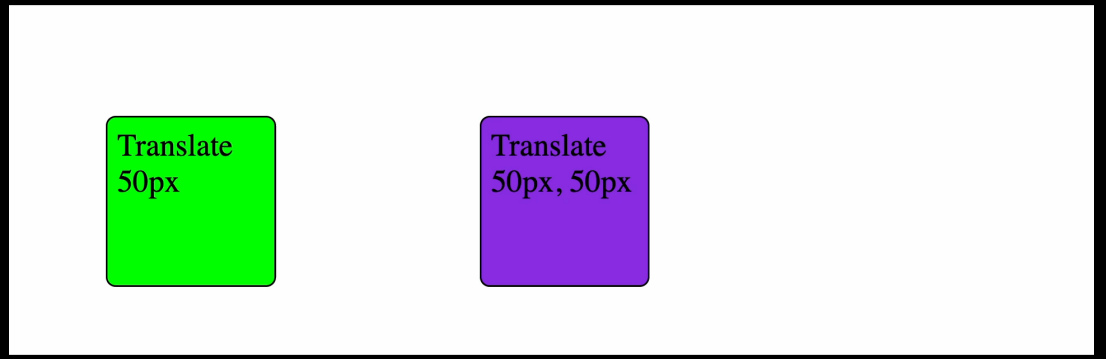
CSS Transform (Translate)

```css
.box {
    height: 75px;
    width: 75px;
    padding: 5px;
    margin: 50px;
    border: 1px solid black;
    border-radius: 5px;
    transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out;
}
#box1 { background-color: lime;}
#box1:hover { transform: translate(50px); }
#box2 { background-color: blueviolet;}
#box2:hover { transform: translate(50px, 50px); }
#box3 { background-color: indianred;}
#box3:hover { transform: translatex(50px); }
#box4 { background-color: violet;}
#box4:hover { transform: translatey(50px); }
```



CSS Transform (Skew)

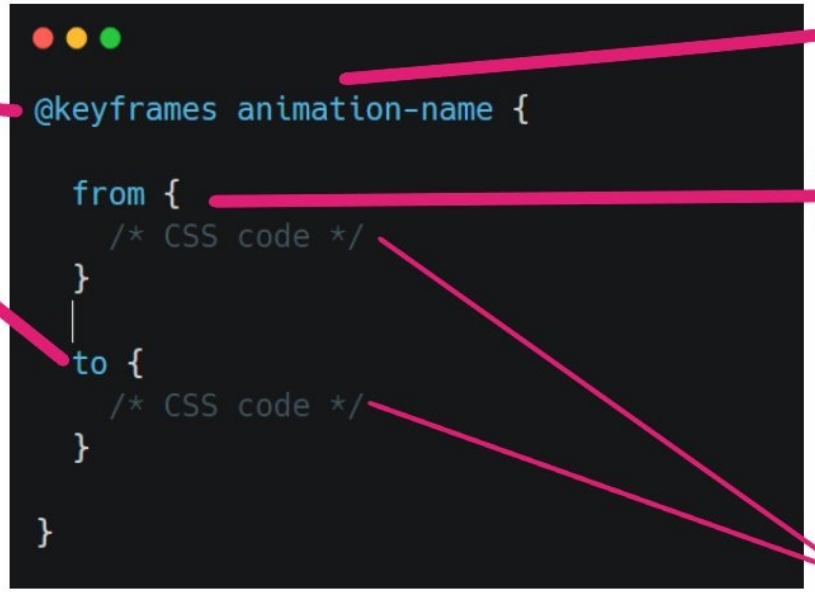
```
.box {
    display: inline-block;
    height: 75px;
    width: 75px;
    padding: 5px;
    margin: 50px;
    border: 1px solid black;
    border-radius: 5px;
    transition-property: all;
    transition-duration: 1s;
    transition-timing-function: ease-in-out;
}
#box1 { background-color: lime;}
#box1:hover { transform: skew(45deg); }
#box2 { background-color: blueviolet;}
#box2:hover { transform: skew(90deg); }
```



Animation

The keyframes at rule rule start with "@keyframes" keyword

animation name which is specified in animation-name property



specify where animation should end.
You can write 100% as well instead of "to"

Specify when the style change will happen.
You can write 0% as well, which is same as "from"

CSS properties.

Animation Properties

- **animation-name**: Specifies the name of the @keyframes defined animation.
- **animation-duration**: Defines the total time the animation takes to complete one cycle.
- **animation-timing-function**: Controls the pacing of the animation (e.g., linear, ease-in).
- **animation-delay**: Sets a delay before the animation starts, allowing for a pause before initiation.
- **animation-iteration-count**: Indicates the number of times the animation should repeat.
- **animation-direction**: Specifies the direction of the animation, allowing for reverse or alternate cycles.

Animation

```css
.box {
    height: 75px;
    width: 75px;
    border: 1px solid black;
    border-radius: 5px;
    position: absolute;
    left: 10;
    background-color: lime;

    animation-name: ghumakkad;
    animation-duration: 4s;
    animation-timing-function: ease-in-out;
    animation-delay: 0s;
    animation-iteration-count: 4;

    animation-direction: alternate;

    /* animation: ghumakkad 4s ease-in-out 0s 4
    alternate; */
}

@keyframes ghumakkad {
    from {left: 10px}
    to {left: 300px}
}
```

Animation


```
@keyframes ghumakkad {
    0% {left: 10px; top: 0px}
    50% { left: 150px; top: 100px }
    100% {left: 300px; top: 0px}
}
```

Practice Set

Animation, Transition & Transform

- Create a webpage with a progress bar that showcases a smooth loading animation. The progress bar should fill up from 0 to 100% with a smooth transition effect and a slight bounce when it reaches 100%.

Advanced Practice Set

Animation, Transition & Transform

```html
<head>
  <title>Interactive Card</title>
  <style>
    /* Your CSS code here */
  </style>
</head>
<body>
  <div class="card">
    <div class="card-front">
      <h2>Front Side</h2>
    </div>
    <div class="card-back">
      <h2>Back Side</h2>
    </div>
  </div>
</body>
```

1. CSS Transition: Apply a transition to the .card to smoothly transform the card when hovered. Use transform 0.6s ease-in-out.
2. CSS Transform (Rotate): Rotate the .card by 180 degrees when hovered.
3. CSS Transform (Scale): Scale the .card to 1.1 times its original size when hovered.
4. CSS Transform (Translate): Translate the .card 10px to the right and 20px down when hovered.
5. CSS Transform (Skew): Skew the .card by 10 degrees along the X-axis and 5 degrees along the Y-axis when hovered.
6. CSS Animation: Create a keyframe animation named flip that rotates the .card from 0 degrees to 180 degrees.
7. Animation Properties: Apply the flip animation to the .card with a duration of 2 seconds, an ease-in timing function, a delay of 0.5 seconds, 3 iterations, and alternate direction.

Advanced Practice Set (Solution)

Animation, Transition & Transform

```css
/* Base styles for the card */
.card {
  width: 200px;
  height: 300px;
  perspective: 1000px;
}

.card-front, .card-back {
  width: 100%;
  height: 100%;
  position: absolute;
  backface-visibility: hidden;
}

.card-front {
  background-color: #fff;
  color: #000;
}

.card-back {
  background-color: #000;
  color: #fff;
  transform: rotateY(180deg);
}
```

```css
/* CSS Transition */
.card {
  transition: transform 0.6s ease-in-out;
}

/* CSS Transform (Rotate) */
.card:hover {
  transform: rotateY(180deg);
}

/* CSS Transform (Scale) */
.card:hover {
  transform: scale(1.1);
}

/* CSS Transform (Translate) */
.card:hover {
  transform: translate(10px, 20px);
}

/* CSS Transform (Skew) */
.card:hover {
  transform: skew(10deg, 5deg);
}
```

```css
/* CSS Animation */
@keyframes flip {
  from {
    transform: rotateY(0deg);
  }
  to {
    transform: rotateY(180deg);
  }
}

.card {
  animation-name: flip;
  animation-duration: 2s;
  animation-timing-function: ease-in;
  animation-delay: 0.5s;
  animation-iteration-count: 3;
  animation-direction: alternate;
}
```