ANIMATIONS

Animation in CSS brings life to web pages, enhancing user experience by making interactions more engaging and visually appealing. By incorporating animations, you can create smooth transitions, transform elements, and define complex sequences of movements. CSS animations are highly efficient, running directly in the browser without the need for external libraries or plugins.

There are three primary properties in CSS that are used to achieve animations: transition, transform, and animation. Each serves a unique purpose and offers different levels of control over how elements change their appearance or position.

1. Transition

Definition:

The transition property allows you to define smooth changes in property values over a specified duration. This is particularly useful for hover effects and other state changes.

Use Case:

Transitions are commonly used to improve the aesthetics and usability of a website by making state changes feel more natural. For instance:

- Buttons and Links: Changing background colors, borders, or shadows on hover.
- Menus: Smoothly showing or hiding dropdowns.
- Images and Cards: Enlarging or adding effects like shadows when hovered over.

Key Properties:

- transition-property: Specifies the CSS property that will be affected by the transition.

- transition-duration: Defines how long the transition takes to complete.
- transition-timing-function: Determines the speed curve of the transition.
- transition-delay: Sets a delay before the transition starts.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
          <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title>Transition Example</title>
    <style>
        .button {
            background-color: blue;
            color: white;
            padding: 10px 20px;
            border: none;
            cursor: pointer;
            transition-property: background-color, transform;
            transition-duration: 0.5s, 0.3s;
            transition-timing-function: ease, ease-in-out;
            transition-delay: 0s, 0.1s;
        }

        .button:hover {
            background-color: green;
            transform: scale(1.1);
        }
    </style>
</head>
<body>
    <button class="button">Hover me</button>
</body>
</html>
```

Explanation**:**

- Initial State: The button has a blue background, and when not hovered, it remains unchanged.
- transition-property: Specifies that background-color and transform will be transitioned.
- transition-duration: background-color changes over 0.5s, and transform changes over 0.3s.
- transition-timing-function: background-color uses ease (smooth start and end), and transform uses ease-in-out (smooth start, acceleration, and smooth end).
- transition-delay: background-color starts immediately, while transform starts after a 0.1s delay.
- Hover State: When the button hovers, the background colour changes to green and the button scales up by 10%.

This example demonstrates how transitions can create a more engaging and interactive experience for users by smoothly animating changes in appearance and behaviour.

2. Transform

Definition:

The transform property in CSS allows you to modify the appearance and layout of elements by applying various transformations such as translation, rotation, scaling, and skewing.

Use Case:

You can use the transform property along with transformation functions to manipulate elements in 2D or 3D space. Some common transformation functions include translate(), rotate(), scale(), skew(), and their 3D counterparts.

Key Properties:

- translate: Moves the element from its current position along the X and Y axes.
- rotate: Rotates the element clockwise or counterclockwise around a fixed point.
- scale: Scales the element in size along the X and Y axes.
- skew: Skews the element along the X and Y axes.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title>Transform Property Example</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background-color: blue;
            transition: transform 0.5s ease-in-out;
        }

        .box:hover {
             transform: rotate(45deg) scale(1.2) skew(20deg, 10deg)
translate(50px, 50px);
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

Explanation

- Initial State: The blue box appears with its default size and shape.

- Hover State: When hovered, the box rotates 45 degrees clockwise, scales up by 20%, skews by 20 degrees along the X-axis and 10 degrees along the Y-axis, and translates 50 pixels both horizontally and vertically. The transition effect is smooth due to the specified easing function (ease-in-out).

This example demonstrates the use of the transform property to apply multiple transformations to an element, resulting in various visual effects.

2D Transformations:

Definition:

2D transformations are transformations that occur in the x and y dimensions, meaning they are limited to the plane of the screen.

Use Case:

2D transformations are commonly used for creating animations, visual effects, and layout adjustments within the 2D plane.

Key Properties:

- rotate(): Rotates an element around a fixed point.
- scale(): Increases or decreases the size of an element.
- skew(): Skews or shears an element along the x-axis and/or y-axis.
- translate(): Moves an element horizontally and/or vertically.

Example:

```
.element {
      transform:   translate(50px,   20px)   rotate(30deg)
scale(1.2);
}
```

Explanation

In this example, the transform property applies a horizontal translation of 50 pixels, a vertical translation of 20 pixels, a rotation of 30 degrees, and a scale of 1.2 to the element.

3D Transformations:

Definition:

3D transformations are transformations that occur in the x, y, and z dimensions, allowing elements to be transformed in 3D space.

Use Case:

3D transformations are used for creating more advanced visual effects, such as realistic animations, 3D objects, and interactive experiences.

Key Properties:

- rotate3d(): Rotates an element around a custom axis in 3D space.
- scale3d(): Scales an element in 3D space.
- translate3d(): Moves an element in 3D space.

Example:

```
.element {
    transform: rotateX(45deg) rotateY(30deg) scale3d(1.2, 1.2, 1.2);
}
```

Explanation

In this example, the `transform` property applies a rotation of 45 degrees around the x-axis, a rotation of 30 degrees around the y-axis, and a scale of 1.2 in all three dimensions (x, y, and z) to the element.

3. Animation

Definition:

The animation property is a shorthand property that allows you to apply animations to an element. It combines several animation sub-properties into a single property for easier and more concise animation configuration.

Use Case:

The animation property is used to define the animation sequence for an element. It can include the animation name, duration, timing function, delay, iteration count, direction, and other animation-related properties.

Key Properties:

- animation-name: Specifies the name of the keyframe animation.
- animation-duration: Sets the duration of the animation cycle.
- animation-timing-function: Defines the timing function used for the animation.
- animation-delay: Specifies the delay before the animation starts.
- animation-iteration-count: Determines the number of times the animation should repeat.
- animation-direction: Sets the direction of the animation (normal, reverse, alternate, or alternate-reverse).
- animation-fill-mode: Specifies how the animation applies styles before and after its execution.
- animation-play-state: Allows you to pause or resume an animation.

Example:

```
.element {
    animation: myAnimation 3s ease-in-out 1s infinite
alternate;
```

```
}

@keyframes myAnimation {
  0% {
    transform: translateX(0);
  }
  50% {
    transform: translateX(100px);
  }
  100% {
    transform: translateX(0);
  }
}
```

In this example:

- The animation property applies an animation named myAnimation to the .element class.
- The animation has a duration of 3s (3 seconds).
- The timing function used is ease-in-out, which creates a smooth easing effect at the beginning and end of the animation.
- The animation has a delay of 1s (1 second) before it starts.
- The animation repeats infinitely (infinite).
- The animation alternates between the normal and reverse directions (alternate).

The @keyframes rule defines the keyframes of the animation, which specify the styles to be applied at different points during the animation sequence. In this case, the animation moves the element horizontally by translating it from 0 to 100px and back to 0.

Breakdown of the animation property:

```
animation: myAnimation 3s ease-in-out 1s infinite alternate;
             ^          ^     ^        ^     ^       ^
          name      duration timing  delay  count  direction
```

You can also define each sub-property separately:

```
.element {
  animation-name: myAnimation;
  animation-duration: 3s;
  animation-timing-function: ease-in-out;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

The animation property provides a powerful and flexible way to create animations in CSS, allowing you to control various aspects of the animation sequence, such as timing, repetition, and direction.

4. Animate.css

Definition:

Animate.css is a library of CSS classes that animate HTML elements when they are triggered through different events, such as page load, scrolling, or user interaction.

Use Case:

To use Animate.css in your project, you need to include the Animate.css file in your HTML document, typically in the `<head>` section. You can either download the library from the official website or include it via a CDN (Content Delivery Network) link.

Example:

```
<head>
  <!-- Include Animate.css via CDN -->
                          <link            rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css" />
</head>
```

Once included, you can apply the desired animation classes to your HTML elements. Animate.css provides various categories of animations, such as attention seekers, bouncing entrances, fading entrances, flippers, and more.

```
<div class="animate__animated animate__bounceInLeft">This element will bounce in from the left.</div>
```

In this example, the animate__animated class is a base class required for all animations, and animate__bounceInLeft is the specific animation class that makes the element bounce in from the left.

Sub-properties:

Animate.css doesn't have sub-properties per se, but it provides many different animation classes that you can use directly on your HTML elements. Some examples of animation classes include:

- animate__bounce
- animate__fadeIn
- animate__flipInX
- animate__rotateIn
- animate__zoomIn
- animate__slideInUp

  and many more

By using Animate.css, you can easily add sophisticated animations to your web pages without writing complex CSS keyframe animations from scratch. The library is well-documented and provides a wide variety of animations that you can mix and match to create engaging user experiences.

5. transform-style

The transform-style property in CSS defines how nested elements are rendered in 3D space. It controls whether the children of an element

are positioned in the same 3D plane as the parent element or rendered flat, like in a 2D space.

The transform-style property accepts two values:

1. **flat (default)**: Child elements are rendered flat, like in a 2D space.
2. **preserve-3d**: Child elements are positioned in their own 3D plane, preserving their 3D transformation.

Example:

```
.parent {
  transform-style: preserve-3d;
  perspective: 500px; /* Required for 3D effects */
}

.child {
  transform: rotateY(45deg);
}
```

In this example, the .child element will be rotated in 3D space because the .parent element has transform-style: preserve-3d set. The perspective property is also applied to create a 3D effect.

6. Perspective:

The perspective property in CSS sets the distance between the z-plane and the user, creating a 3D effect. It is used in conjunction with 3D transformations like rotateX, rotateY, and scale3d.

The perspective property can be set on the transformed element itself or on its parent element. The lower the value, the more exaggerated the 3D effect will be.

Example:

```
.element {
  perspective: 500px;
```

```
  transform: rotateY(45deg);
}
```

In this example, the .element will have a 3D rotation effect due to the perspective and transform properties applied.

7. visibility:

The visibility property in CSS controls the visibility of an element. It can be used to hide or show elements without affecting the layout.

The visibility property accepts three values:

1. **visible** (default): The element is visible.
2. **hidden**: The element is invisible, but it still takes up space in the layout.
3. **collapse**: This value is specific to table elements and causes the table row or column to be hidden and not take up any space.

Example:

```
.visible {
  visibility: visible;
}

.hidden {
  visibility: hidden;
}
```

In this example, elements with the class .visible will be visible, while elements with the class .hidden will be invisible but still occupy space in the layout.

It's important to note that the visibility property differs from the display property in CSS. display: none removes the element completely from the layout, while visibility: hidden hides the element but preserves its space in the layout.

Conclusion

Animations in CSS have become an essential part of modern web design, providing a visually appealing and engaging experience for users. CSS offers several powerful properties that enable developers to create smooth and dynamic animations, enhancing the interactivity and overall appeal of web pages.

The first property we discussed is the transition, which allows for a smooth transition between two states of an element. By specifying the properties to be transitioned, the duration, and the timing function, developers can create subtle animations that respond to user interactions, such as hovering over a button or expanding a menu.

Next, we explored the transform property, which enables 2D and 3D transformations of elements. With Transform, developers can rotate, scale, skew, and translate elements, opening up a world of possibilities for creating captivating animations. When combined with transitions, these transformations can create dynamic and immersive experiences.

Finally, we delved into the animation property, which provides a more complex and powerful way to create animations. This property allows developers to define keyframes, specifying the styles to be applied at different points during the animation sequence. Animations can be customised with various properties, such as duration, timing functions, iteration counts, and direction, enabling a wide range of creative possibilities.

By leveraging these three properties – transition, transform, and animation – developers can bring their web designs to life, making them more engaging, interactive, and visually appealing. Whether it's a subtle hover effect, a captivating page transition, or a complex animation sequence, CSS provides the tools necessary to create a seamless and enjoyable user experience.

It's important to note that while animations can enhance the overall user experience, they should be used judiciously and with consideration for performance and accessibility. Overusing animations or using them

inappropriately can negatively impact the user experience, especially on lower-end devices or for users with disabilities.

In conclusion, CSS animations have become an integral part of modern web development, enabling developers to create dynamic and engaging user experiences. By mastering the transition, transform, and animation properties, developers can unlock a world of creative possibilities and bring their web designs to life in a visually stunning and captivating way.

References

1. https://developer.mozilla.org/en-US/docs/Web/CSS/transition
2. https://developer.mozilla.org/en-US/docs/Web/CSS/transform
3. https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/translate3d
4. https://developer.mozilla.org/en-US/docs/Web/CSS/animation
5. https://animate.style/