5.5 Animation

Animations and keyframes

Developers often use JavaScript and JavaScript libraries like jQuery to produce animations. However, CSS can also be used to produce animations. A ***CSS animation*** transforms an element's styles over a set time period, producing an animation. CSS animations have three advantages over JavaScript animations:

1. CSS animations do not require any JavaScript code.
2. CSS animations often put less load on the computer and can use techniques to produce smoother animations when the computer's resources are limited.
3. CSS animations allow the browser to more efficiently control animations and stop animations from running in browser tabs that are not visible.

A CSS animation's behavior is defined with the ***@keyframes*** rule, which contains a keyframe list. A ***keyframe list*** has a name and contains the keyframes or the properties and values that will be animated. A keyframe list contains two keyframe selectors:

- ***from*** - The animation starting state that lists the CSS properties and values that apply when the animation begins
- ***to*** - The animation ending state that lists the CSS properties and values that the "from" values become by the time the animation ends

Percentages may be used to specify keyframes at various points during the animation. Ex: `0%` is equivalent to `from` and `100%` is equivalent to `to`. The value `50%` indicates the animation state at the halfway point.

To create an animation, two CSS properties must be defined:

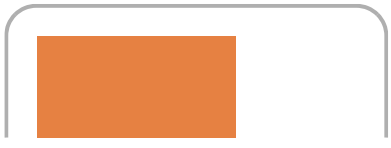
- ***animation-name*** - Names the keyframe list associated with the animation
- ***animation-duration*** - Length of the animation in seconds (s) or milliseconds (ms)

An animation begins immediately when the browser renders the webpage unless an ***animation-delay*** is used to delay the start of the animation.

PARTICIPATION ACTIVITY 5.5.1: Animating the background color.

Start 2x speed

```
div {
  width: 100px;
  height: 100px;
  background-color: orange;
  animation-name: changeColors;
```



```
  animation-duration: 1s;
  animation-delay: 2s;
}

@keyframes changeColors {
  from { background-color: blue;  }
  to   { background-color: green; }
}
```

```
<div></div>
```

Captions

1. "animation-name" names the keyframe list associated with the animation.
2. "animation-duration" specifies the animation will last 1 second.
3. "animation-delay" tells the browser to wait 2 seconds before starting the animation.
4. After the 2 second delay, the animation begins with "from", so the background is initially blue.
5. During the 1 second duration, "to" indicates the background color becomes green.
6. The background becomes orange again after the animation completes.

Feedback?

PARTICIPATION ACTIVITY 5.5.2: Keyframes and animation.

1) A valid keyframe list must include the `from` and `to` keyframe selectors.

- ○ True
- ◉ False

Correct

A valid keyframe list must include a starting and ending state. However, `from` and `to` keyframe selectors can be replaced with percentages. Ex:

```
@keyframes changeColors {
  0%   { background-color: blue;  }
  100% { background-color: green; }
}
```

2) A property that is listed in the `from` keyframe selector but not the `to` keyframe selector will still animate.

- ○ True
- ◉ False

Correct

Only properties that are listed in both the `from` and `to` keyframe selectors are animated.

3) An animation without an `animation-delay` property begins immediately.

- ◉ True
- ○ False

Correct

The `animation-delay` property delays the animation's start.

4) If `animation-duration` is assigned the value 0s, the animation occurs very quickly.

- ○ True
- ◉ False

Correct

No animation occurs unless a positive length value is assigned.

Feedback?

PARTICIPATION ACTIVITY 5.5.3: Percentages for keyframes.

The webpage below shows a smiley face that moves to the right while changing colors to blue, then moves back to the left while changing colors back to red. Add two keyframes to the animation:

1. 25% through the animation, make the smiley face appear 200 pixels to the right and 100 pixels below the smiley face's starting location. The smiley face should also become yellow.
2. 75% through the animation, make the smiley face appear against the left side of the webpage and 100 pixels below the smiley face's starting location. The smiley face should also become green.

Rendering the webpage should animate the smiley face down and to the right, up, down and to the left, and up again while changing colors.

HTML CSS

```
1 <div id="smiley">&#9786;</div>
2
```

Render webpage Reset code

Your webpage

▼ View solution

Explain

```
--- START FILE: CSS ---


#smiley {
   color: red;
   font-size: 50pt;
   position: absolute;
   left: 0px;
   top: 0px;
   animation-name: move;
```

```
    animation-duration: 3s;
}

@keyframes move {
    0%   { color:red; left:0px; top:0px; }
    25%   { color:yellow; left:200px; top:100px; }
    50%  { color:blue; left:200px; top:0px; }
    75%   { color:green; left:0px; top:100px; }
    100% { color:red; left:0px; top:0px; }
}


--- END FILE: CSS ---
```

Feedback?

Timing, iteration count, and direction

In the smiley face animation above, each transition from one keyframe to the next began with a slow start, then fast, then a slow end. The ***animation-timing-function*** property controls an animation's speed between keyframes. Several timing functions are available:

- `ease` - Slow start, then fast, then slow end (default)
- `linear` - Same speed throughout
- `ease-in` - Slow start
- `ease-out` - Slow end
- `ease-in-out` - Slow start and end
- `cubic-bezier(n1,n2,n3,n4)` - Specify numbers that control speed based on a Bezier curve

Other animation properties include:

- ***animation-iteration-count*** - Indicates the number of times the animation will run. The value `infinite` runs the animation repeatedly without stopping. Ex: `animation-iteration-count: 3` runs the animation three times.
- ***animation-direction*** - Indicates animation direction
  - `normal` - Normal direction (default)
  - `reverse` - Reverse direction
  - `alternate` - Alternate between normal and reverse
  - `alternate-reverse` - Alternate between reverse and normal
- ***animation*** - Shorthand property indicating the animation name, duration, timing function, delay, iteration count, and direction. Ex: `animation: move 3s linear 2s infinite normal`.

PARTICIPATION ACTIVITY 5.5.4: Keyframes and timing functions.

1) Which keyframe selector is equivalent to `to`?

- 0%
- 100%
- 50%

Correct

100% and `to` indicate the ending animation state.

2) Which keyframe selector specifies the animation state when the animation is three quarters finished?

- 0%
- 50%
- 75%

Correct

75% is three quarters through the animation.

3) Which timing function makes the animation progress at the same speed the entire time?

- linear
- ease
- cubic-bezier

Correct

"linear" is the same speed throughout the animation.

4) How many times will the animation below run?

```
animation: move 4s ease 1s 2 reverse;
```

- once
- twice
- infinite

Correct

The "2" indicates the iteration count.

5) What color is `#thing`'s font right as the animation completes?

```
#thing {
  animation:
changeColors 4s
ease 1s 2
reverse;
}
@keyframes
changeColors {
  0%   {
color:red; }
  50%  {
color:blue; }
  100% {
color:green; }
}
```

- (●) red
- ( ) blue
- ( ) green

Correct

The animation direction is `reverse`, so the 100% keyframe is first, and the 0% keyframe is last. The 0% keyframe makes the font color red.

Feedback?

Transitions

A ***CSS transition*** animates an element's transition from one state to another when an element's CSS property changes value. Ex: A transition may animate an element getting wider when the element's width is increased. Transitions are commonly used with the `:hover` pseudo-class to trigger an animation when the user mouses over an element.

Transitions differ from CSS animations in two ways:

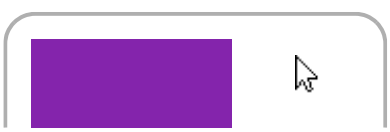
1. Transitions execute when an element's property values are changed, unlike CSS animations that execute at a particular time.
2. Transitions provide less control over the animation than CSS animations.

The ***transition*** property defines a transition by specifying one or more CSS properties and each property's transition duration.

PARTICIPATION ACTIVITY 5.5.5: Transitioning the width and height when hovering.

Start ☐ 2x speed

```
div {
  width: 100px;
  height: 100px;
  background-color: purple;
```



```
  transition: width 1s, height 1s;
}

div:hover {
  width: 120px;
  height: 120px;
}
```

```
<div></div>
```

Captions

1. When the width property or height properties are changed, the transition to the new values will be animated over 1 second.
2. The width and height properties are increased to 120px when the mouse hovers over the .
3. So, when the mouse cursor hovers over the , the width and height transitions from 100px to 120px, animated over 1 second.
4. When the cursor no longer hovers over the , the transition from 120px to 100px is animated over 1 second.

Feedback?

The ***transition-timing-function*** property controls the speed of the transition. Several timing functions are available, and all complete in the same amount of time:

- `ease` - Slow start, then fast, then slow end (default)
- `linear` - Same speed throughout
- `ease-in` - Slow start
- `ease-out` - Slow end
- `ease-in-out` - Slow start and end
- `cubic-bezier(n1,n2,n3,n4)` - Specify numbers that control speed based on a Bezier curve

The ***transition-delay*** property delays the transition's start.

PARTICIPATION ACTIVITY 5.5.6: Transitions.

1) A transition can animate one or more CSS properties.
   - (•) True
   - ( ) False

Correct

Each property to be animated and time value is separated by a comma.

2) The `transition` property below makes the width take 3 seconds longer than the height to complete the transition.

```
transition: width 3s, height 1s;
```

- ○ True
- ◉ False

Correct

The width transitions in 3 seconds and height in 1 second. So the width transition takes 3 - 1 = 2 seconds longer to complete.

3) The `#div1` transition takes longer to complete than the `#div2` transition.

```
div {
  width: 100px;
  height: 100px;
  transition: width 1s;
}

#div1 {
transition-timing-function: ease-in; }
#div2 {
transition-timing-function: linear; }
```

- ○ True
- ◉ False

Correct

All timing functions take an equal amount of time to complete a transition. The linear function uses a constant speed for the entire transition, unlike the other functions that animate more slowly or quickly at times.

4) According to the CSS below, a div element would not decrease in size until 500 milliseconds after the cursor hovered over the div.

```
div {
  width: 100px;
  height: 100px;
  transition:
width 1s;
  transition-
delay: 500ms;
}

div:hover {
  width: 80px;
}
```

- (●) True
- ( ) False

Correct

`transition-delay` delays the transition for the specified number of seconds or milliseconds.

5) The CSS below causes a paragraph to disappear when the mouse hovers over the paragraph.

```
p {
  transition:
opacity 500ms;
}

p:hover {
  opacity: 0;
}
```

- (●) True
- ( ) False

Correct

The opacity transitions to 0 in half a second (500 milliseconds) when the mouse hovers over the paragraph, making the paragraph disappear. The paragraph reappears when the mouse is moved off the paragraph.

Feedback?

Transformations

The ***transform*** property applies a 2D or 3D transformation to an element. A ***transformation*** is a graphical operation that alters the position, shape, or orientation of an object. The `transform` property is assigned a transformation function. A selected number of 2D transformation functions are summarized in the table below.

Transformations are used in animations and transitions to create engaging webpages.

Table 5.5.1: Selected 2D transformation functions.

Function	Description	Example
translate(x, y)	Moves an element on the x-axis `x` distance and along the y-axis `y` distance	`/* Moves right 10px and up 20px */` `translate(10px, -20px)`
scale(x, y)	Increases (values > 1) or decreases (values < 1) the width and height by the `x` and `y` multiplier	`/* Halves the width, doubles the height */` `scale(0.5, 2)`
rotate(angle)	Rotates clockwise by `angle`	`/* Rotates clockwise 45 degrees */` `rotate(45deg)`

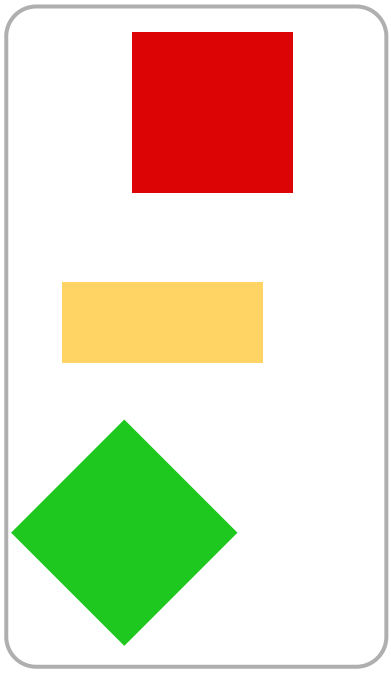
Feedback?

PARTICIPATION ACTIVITY 5.5.7: translate(), scale(), and rotate() transformation functions.

Start 2x speed

```
#example1 {
  background-color: red;
  transform: translate(30px, -10px);
}
#example2 {
  background-color: yellow;
  transform: scale(1.2, 0.5);
}
#example3 {
  background-color: green;
  transform: rotate(45deg);
}
```

```
<div id="example1"></div>
<div id="example2"></div>
<div id="example3"></div>
```



Captions

1. All three squares are displayed at default locations.
2. translate() moves the square 30 pixels to the right and 10 pixels up.

3. scale() multiplies the yellow square's width by 1.2 and the height by 0.5, so the square is 20% wider and 50% shorter.
4. rotate() rotates the green square 45 degrees clockwise.

Feedback?

PARTICIPATION ACTIVITY 5.5.8: Animations, transitions, and transformations practice.

The webpage below displays a welcome message and a large button. When the cursor hovers over the button, the font color turns red, and the button grows larger. When the button is pressed, the background color is darkened.

The button transition is defined in the `.button` class and specifies that "all" properties should be animated in a transition in 100 milliseconds using the "ease-in-out" timing function. The `:hover` pseudo-class is used to scale the button 5% larger and change the font color, and the `:active` pseudo-class is used to change the background color.

Make the following modifications to the CSS so the rendered webpage behaves like the expected webpage:

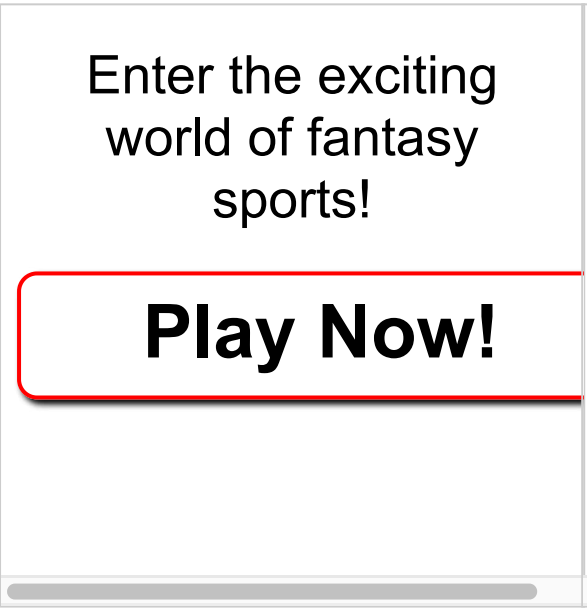
1. Add an animation that runs as soon as the page is rendered that moves the text onto the screen. Use the `translate()` function to move the welcome message to -300px initially, then to 60px half way through the animation, then back to 0px at the end.
2. Add an animation that wiggles the button one second after the page renders. Use the `rotate()` function to initially rotate 0 degrees, then 3 degrees, then -3 degrees, and then 0 degrees. The animation should run twice.
3. Experiment with different timing functions. The expected webpage uses linear timing for the two animations listed above.
4. Finally, add a transformation to the `:active` pseudo-class that uses the `scale()` function to scale the button size down 5%.

HTML CSS

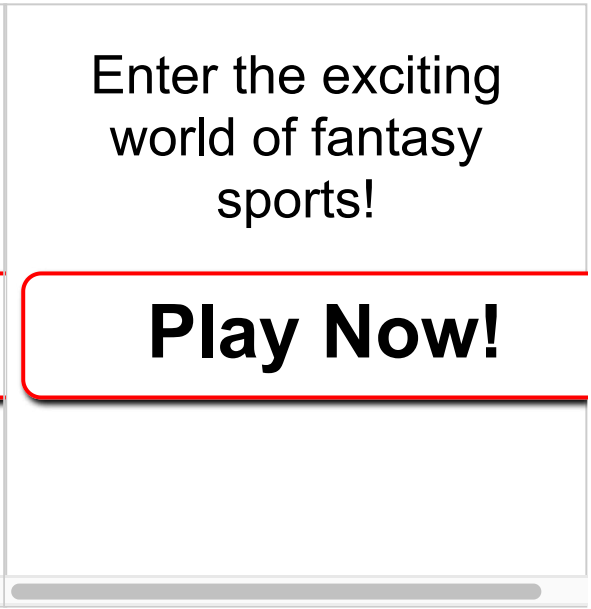
```
1 <div id="welcome">Enter the exciting world of fantasy sports!</di
2 <div class="button">Play Now!</div>
3
```

Render webpage Reset code

Your webpage



Expected webpage



▼ View solution

Explain

```
--- START FILE: HTML ---

<div id="welcome">Enter the exciting world of fantasy
sports!</div>
<div class="button">Play Now!</div>


--- END FILE: HTML ---

--- START FILE: CSS ---
```

```
#welcome {
   margin: 20px;
   font: 20pt arial;
   text-align: center;
   animation: spin 1s linear;
}

@keyframes spin {
   0%   { transform: translate(-300px); }
   50%  { transform: translate(60px); }
   100% { transform: translate(0px) }
}

.button {
   margin: auto;
   text-align: center;
   vertical-align: middle;
   width: 300px;
   height: 60px;
   line-height: 60px;
   font: bold 28pt/150% arial;
   border: solid 2px red;
   box-shadow: 2px 2px 3px;
   border-radius: 10px;
   background: radial-gradient(white, #bbb);
   transition: all 100ms ease-in-out;
   animation: wiggle 200ms linear 1s 2 reverse;
}

@keyframes wiggle {
   0%   { transform: rotate(0); }
   25%  { transform: rotate(3deg); }
   75%  { transform: rotate(-3deg); }
   100% { transform: rotate(0); }
}

.button:hover {
   transform: scale(1.05) ;
   color: red;
   cursor: pointer;
}

.button:active {
   transform: scale(0.95);
   background: radial-gradient(#ddd, #999);
```

```
}


--- END FILE: CSS ---
```

Feedback?

PARTICIPATION ACTIVITY 5.5.9: Transformations in transitions and animations.

1) `translate()`, `scale()`, and `rotate()` are ____ functions.

- transition
- transformation
- translation

Correct

Translating, scaling, and rotating are transformations.

2) Which function moves an element 20 pixels to the left and 5 pixels down?

- `translate(-20, 5)`
- `translate(20px, -5px)`
- `translate(-20px, 5px)`

Correct

-20px is 20 pixels left, and 5px is 5 pixels down.

3) What does the element using the keyframes below look like at the end of the animation?

```
@keyframes
example {
   from {
transform:
scale(0.5, 0.5);
}
   to   {
transform:
rotate(45deg)
scale(1.5, 1.5);
}
}
```

- ○ Scaled smaller
- ○ Rotated 45 degrees only
- ◉ Rotated 45 degrees and scaled larger

Correct

The "to" keyframe selector shows the two transformations, rotate and scale, that form the animation ending state.

4) What does the <p> look like when the mouse clicks the element?

```
p {
   transition:
all 1s;
}
p:hover {
   transform:
translate(10px,
0);
}
p:active {
   transform:
scale(2, 2);
}
```

- ◉ Scaled larger
- ○ Translated 10 pixels to the right
- ○ No change

Correct

The :active pseudo-class scales the <p> larger when clicked.

Feedback?

Exploring further:

- CSS3 Animations from W3Schools
- CSS3 Transitions from W3Schools
- CSS3 2D Transforms from W3Schools
- CSS3 3D Transforms from W3Schools

How was this section?  **Provide section feedback**