FEWD WEEK 4 • CLASS 7:

Animation

https://slides.com/jennifermeade/fewd-4-7/live

OBJECTIVES

- Apply CSS transitions to smoothly transition changes
- Understand and use CSS Transform properties
- Use keyframes to create sophisticated animation effects
- Understand why some values are animatable while others aren't

TRANSITIONS

TRANSITIONS

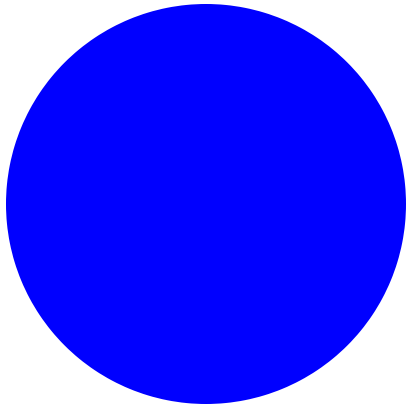
The CSS transition property smoothly transitions from one value to another over time.

TRIGGERING EVENTS

Transitions require a triggering event. Often we use javascript to listen for events, but we can also use a pseudo class such as `:hover` to trigger a change.

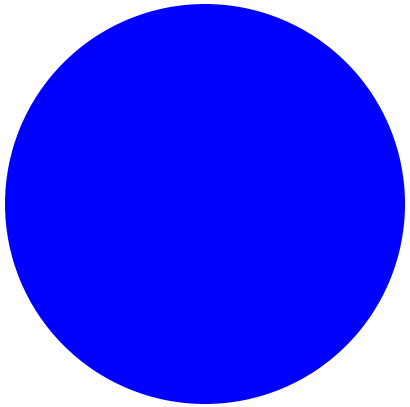
TRANSITION

```
div {
  background: blue;
}

div:hover {
  background: red;
}
```



NO TRANSITION

```
div {
  background: blue;
  transition: background 2s;
}

div:hover {
  background: red;
}
```



TRANSITION

BASIC TRANSITION SYNTAX

```
div {
  width: 100px;
  height: 100px;
  background: turquoise;
  transition: width 2s;
/* Property to transition and time it takes in seconds */
}

div:hover {
  width: 300px;
}
```

It's important to understand that only animatable properties will transition.

TRANSITION PROPERTIES

The transition property is shorthand for:

- transition-duration
- transition-timing-function (http://easings.net/)
- transition-delay

MULTIPLE TRANSITIONS

For multiple transitions, separate each with a comma.

```
div {
  width: 100px;
  height: 100px;
  background: turquoise;
  transition: width 2s, background 1s;
}
div:hover {
  width: 300px;
  background: purple;
}
```

TRANSITIONS WITH ALL

```
div {
    width: 100px;
    height: 100px;
    background: turquoise;
    transition: transition all 2s;
}

div:hover {
    width: 300px;
    background: yellowgreen;
}
```

You can use the keyword **all** to transition all of the animatable changes transitions, but beware!

ANIMATABLE PROPERTIES

Animatable properties have numerical values.

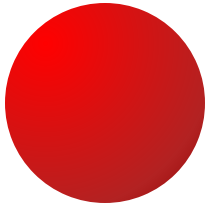
background
background-color
background-position
background-size
border
border-bottom
border-bottom-color
border-bottom-left-radius
border-bottom-right-radius
border-bottom-width
border-color
border-left
border-left-color
border-left-width
border-right
border-right-color
border-right-width
border-spacing
border-top
border-top-color
border-top-left-radius
border-top-right-radius
border-top-width
bottom
box-shadow
clip
color
column-count
column-gap
column-rule
column-rule-color
column-rule-width
column-width
columns
filter
flex
flex-basis
flex-grow
flex-shrink
font
font-size
font-size-adjust
font-stretch
font-weight
height
left
letter-spacing
line-height
margin
margin-bottom
margin-left
margin-right
margin-top
max-height
max-width
min-height
min-width
object-position
opacity
order
outline
outline-color
outline-offset
outline-width
padding
padding-bottom
padding-left
padding-right
padding-top
perspective
perspective-origin
right
text-decoration-color
text-indent
text-shadow
top
transform
transform-origin
vertical-align
visibility
width
word-spacing
z-index

TRANSITIONS

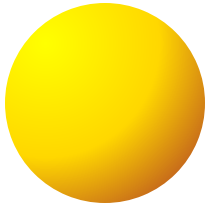
CSS TRANSFORMS

CSS TRANSFORMS

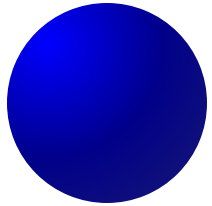
CSS Transforms can **rotate**, **scale**, **skew** and **move** elements in 2d or 3d without affecting the document flow.



ROTATE



SKEW



TRANSLATE

CSS TRANSFORM SYNTAX

```
selector {
  transform: property-value(value);
}

div {
  transform: rotate(45deg);
}
```

TRANSLATE

Move elements along the x, y and z axes:

- translateX(*x*)
- translateY(*y*)
- translateZ(*z*)
- translate(*x*, *y*)
- translate3d(*x*, *y*, *z*)

```
div {
  transform: translate(20px, 5px);
}
/* moves 20px right, 5px down */
```

is the same as:

```
div {
 transform:
  translateX(20px) translateY(5px);
}
```

TRANSLATE VALUES

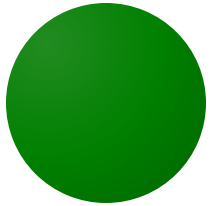
Translate accepts all units of measure.

- px
- %
- em/rem
- vh/vw/vmin/vmax

```
div {
  transform: translate(-2rem, 100%);
}
/*

Result:
> left by 2 rem
> down by 100% the height of
  the element (not its parent)

*/
```

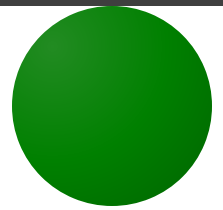
PERSPECTIVE ENABLES 3D

```
.parent {
  perspective: none;
}
.ball {
  transform:
    translate3d(300%, 20%, -100px);
}
```



NO PERSPECTIVE

```
.parent {
  perspective: 50px;
}
.ball {
  transform:
    translate3d(300%, 20%, -100px);
}
```



PERSPECTIVE

ROTATE

Rotates elements along the x, y and z axes:

- rotateX(*x*)
- rotateY(*y*)
- rotateZ(*z*)
- rotate(*z*)

```
div {
  transform: rotateY(1turn);
}
/* flips horizontally */
```

```
div {
 transform: rotateX(360deg);
}
/* flips vertically */
```

ROTATE VALUES

Rotate accepts deg, turn, rad, grad.

- degrees (360)
- turns (1)
- radians (2π)
- gradians (400)

```
div {
 transform: rotateX(1turn);
}

div {
 transform: rotateY(400grad);
}

div {
 transform: rotate(6.2831853rad);
}

div {
 transform: rotate(360deg);
}
```

SKEW

Skews elements along the x and y axes:

- skewX(*x*)
- skewY(*y*)
- skew(*x*, *y*)

```
div {
  transform: skewY(.1turn);
}
```

```
div {
 transform: skewX(6deg);
}
```

SKEW VALUES

Skew accepts deg, turn, rad, grad.

- degrees (360)
- turns (1)
- radians (2π)
- gradians (400)

```
div {
 transform: skewX(.1turn);
}

div {
 transform: skewY(-25deg);
}

div {
 transform: skew(100grad, 10grad);
}
```

SCALE

Scales elements along the x, y and z axes:

- scaleX(*x*)
- scaleY(*y*)
- scaleZ(*y*)
- scale(*x*, *y*)

```
div {
  transform: scaleY(2);
}
/* double height */
```

```
div {
 transform: scale(2, .5);
}
/* double width, ½ height */
```

SCALE VALUES

Scale accepts integers and float values (numbers with or without a decimal).

```
div {
 transform: scale(.5);
}

div {
 transform: scaleZ(2);
/* only in 3d */
}

div {
 transform: scaleX(2.5);
}
```

MULTIPLE TRANSFORMS

```
div {
  transform: translate(20px, 30px) rotate(45deg) scale(1);
}
```

For multiple transform, you need them to be in the same declaration (or cascade causes the last one to override the others).

TRANSFORM ORIGIN

- transform-origin: *x-axis y-axis z-axis*;
- The x-axis and y-axis can accept:
  - a keyword (left, center, top, right, bottom)
  - a length (e.g., 50em or 100px, etc.)
  - a percentage
- The z-axis can only accept a length

The default value is 50% 50% (the element's center point).

Values can be passed as one, two or all three values.

TRANSFORMS

KEYFRAME ANIMATIONS

KEYFRAME ANIMATION

Keyframes differ from transitions. They give us finer control and don't *require* a triggering event. You create your keyframes and then add them to elements with the animation property

KEYFRAMES SYNTAX

```
@keyframes colorchange { /* Give it any name you want*/
    0% {background-color: red;} /* Set start values */
    50% {background-color: yellow;}  /* Any number of interim steps */
    100% {background-color: turquoise;} /* Set end values */
}


div {
    width: 100px;
    height: 100px;
    animation-name: colorchange;
    animation-duration: 4s;
    animation-timing-function: steps(3, start);
}
```

KEYFRAMES ALT SYNTAX

```
@keyframes colorchange {
    from {
        background-color: red;
        opacity: 0;
    }
    to {
        background-color: turquoise;
        opacity: 1;
    }
}
```

You can even skip the **from** and just have it animate from the values set on the element it is applied to.

ANIMATION PROPERTIES

The animation property is shorthand for:

- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-fill-mode
- animation-timing-function

ANIMATION-DELAY

The animation-delay specifies the amount of time to wait before running the animation. It accepts time in seconds (specified with an s) or milliseconds (specified with ms). A **negative value** causes the animation to begin immediately, but partway through its cycle. The delay is only applied before the first cycle begins.

ANIMATION-ITERATION-COUNT

The animation-iteration-count accepts:

- A number representing the number of times the animation should run
- The infinite value, which causes the animation to loop continuously.

ANIMATION-DIRECTION

The animation-direction accepts:

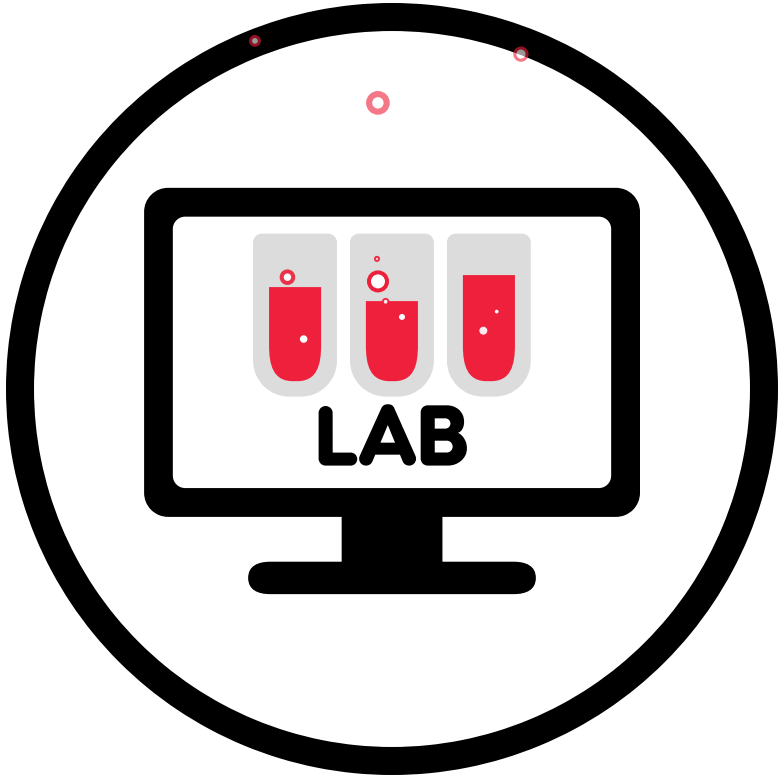
- normal: The default value.
- alternate: Alternates the direction of the animation at the conclusion of each cycle.
- reverse: Runs the animation beginning on the last keyframe running to first.
- alternate-reverse: Starts the first cycle at the last keyframe and then alternates.

ANIMATION-FILL MODE

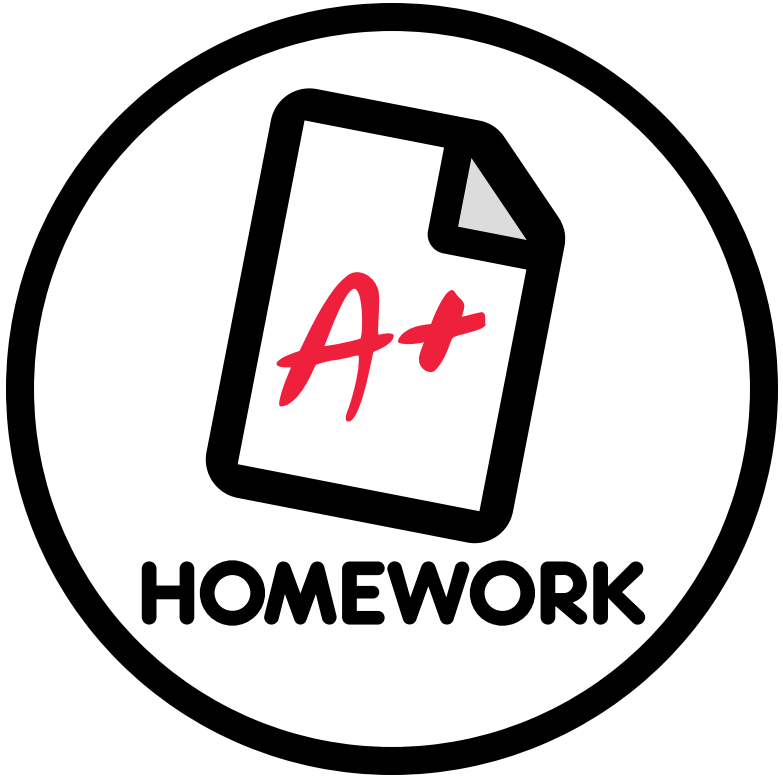
The animation-fill-mode accepts:

- forwards: At the end of the animation, stay on the last keyframe.
- backwards: At the beginning of the animation (and during any delay), stay on the first keyframe.
- both: Wait on the first keyframe until the animation starts and remain on the last when it finishes.

KEYFRAME ANIMATIONS



MATCHMAKER LOGO



CLASS 7 HOMEWORK

https://github.com/jmeade11/FEWD/tree/master/Final%20Project

HOMEWORK FOR NEXT CLASS

- Let's make sure you have a final project proposal ready for Monday.
- In the next class, we'll spend some time on wireframing your proposal, so make sure you've got a clear project in mind.

Here's your chance to get creative. Your final project proposal is due on Monday. You'll need to have a clear objective concept and a set of wireframes (pencil drawing are fine).

EXIT SURVEY

https://goo.gl/EB4XFw

GO BUILD AWESOME THINGS!