FEWD Week 4 • Class 7: Animation

CODE ALONG

Review Time!

Quick Review

- Name one thing I need to do to make my web page responsive.
- What is mobile first?
- What's the basic syntax for a media query?
- Where do media queries go in our CSS file? Why?

What We’ll Cover

- CSS Transforms
- CSS Transitions
- Animatable Properties
- CSS Keyframes Animation

CSS Transforms

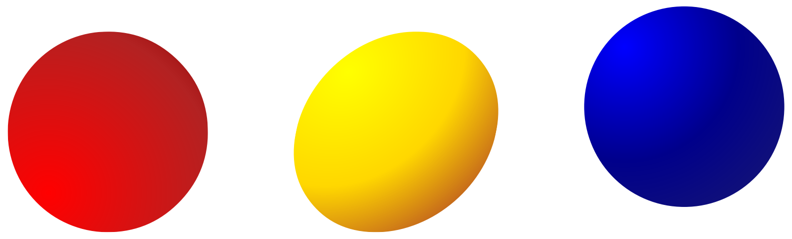
Objectives

- Understand CSS Transform properties
- Use multiple transform properties
- Apply transforms using :active, :hover and :checked pseudo classes

CSS Transforms

CSS Transforms allow us to change the shape and position of elements in 2d or 3d without affecting the document flow. With transform you can:

- translate
- rotate
- skew
- scale



CSS Transform Syntax

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

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

CSS Transform 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 */
```

same as

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

Translate Values

Translate accepts all units of measure.

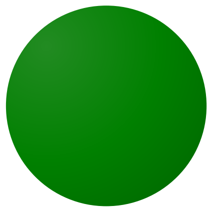
- px
- %
- em/rem
- vh/vw/vmin/vmax

```
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);
}
```

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



CSS Transform 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 */
```

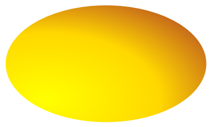
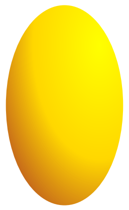
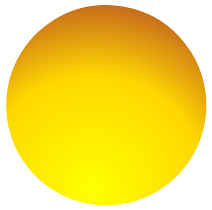
Rotate Values

Rotate accepts deg, turn, rad, grad.

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

```
div {
 transform:
  rotateY(400grad);
}
```

```
div {
 transform:
  rotate(6.2831853rad);
}
```

CSS Transform Skew

Skews elements along the x and y axes:

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

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

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

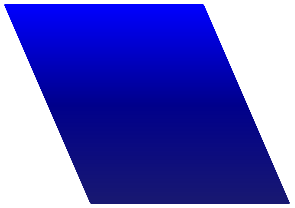
Skew Values

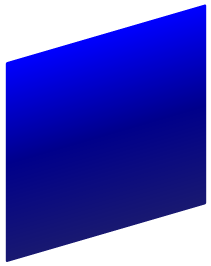
Skew accepts rad, grad, deg or turn.

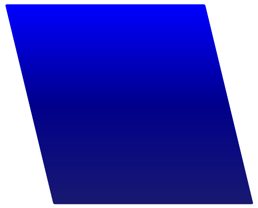
```
div {
 transform:
  skewX(.1turn);
}
```

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

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







CSS Transform 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.

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

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

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

Multiple Transforms

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

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

</> CODE ALONG

Transforms

Transitions

Objectives

- Apply CSS transitions to smoothly transition changes
- Understand which values are animatable
- Use the :hover, :active, :checked pseudo classes to trigger changes

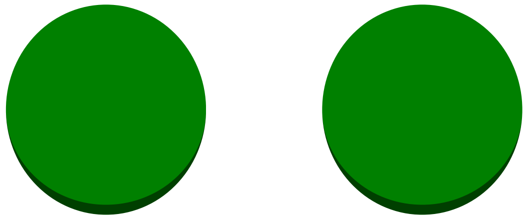
Transitions

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 pseudo classes like :hover.

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



Transition Syntax

```
div {
  width: 100px;
  height: 100px;
  background: turquoise;
  transition: width 2s; /* Property to transition
                           How long the transition takes in seconds */
}
```

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

►**HEADS UP:** Not all properties are animatable.

Transition Properties

Transition is shorthand for the transitions properties

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

Multiple Transitions

For multiple transitions, separate them with a comma

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

🚩**HEADS UP:** You can also use the keyword **all** to transition all of the animatable changes transitions, but beware!

Transitions with Transform

Transitions work especially well with transformations

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

div:hover {
    width: 300px;
    height: 300px;
    background: purple;
    transform: rotate(180deg);
}
```

CODE ALONG

Transitions

Try It Yourself

Keyframe Animations

Keyframes Animations

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 give you more control

```css
@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 */
}
```

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

Animation Properties

The animation property is shorthand for:

- animation-name
- animation-duration
- animation-iteration-count (number or inifinite)
- animation-direction (alternate, reverse, alternate-reverse)
- animation-fill-mode (forwards, backwards, both)
- animation-timing-function

CODE ALONG

Fun with Keyframes

Go Build Awesome Things!