Transitions & Animations

Motion powered by CSS

@theNicoKoenig

Why animations?

- Because it’s fun
- Guide the users attention to what's important
- Give visual feedback (yeah, something happened / is happening)

Animations and the browser

- CSS
- JavaScript
- WebGL
- Canvas

CSS Transitions and Animations

- Superb browser support
- Easy to use
- No library needed

Transitions

Transitions

- Defines a transition between two states of an element
- Can be triggered with JavaScript or user interaction

Different states of an element

- Missing transitions look boring
- Transitions for the win!

Some more examples

- Surprise
- Polaroid
- Platform game

CSS Transition Properties

- `transition-property`
- `transition-duration`
- `transition-timing-function`
- `transition-delay`
- `transition`

Example #1

```
.foo {
    background-color: hotpink;
    transition: background-color .5s;
}

.foo:hover {
    background-color: gray;
}
```

Example #2

```
.bar {
    background-color: hotpink;
    transition: border-radius .5s, background-color .5s;
}

.bar:hover {
    border-radius: 50%;
    background-color: gray;
}
```

Property Shorthand vs Single Properties

```
.shorthand {
    transition: background-color .5s ease-in-out .25s;
}


.longhand {
    transition-property: background-color;
    transition-duration: .5s;
    transition-timing-function: ease-in-out;
    transition-delay: .25s;
}
```

Define multiple transitions

```
.shorthand {
    transition: color .5, opacity .25;
}

.longhand {
    transition-property: color, opacity;
    transition-duration: .5s, .25s;
    /*
    transition-timing-function: ease, ease;
    transition-delay: 0s, 0s;
    */
}
```

transition-property

- Takes properties which values can be changed gradually over time can be animated
- The `all` keyword will animate every change of all animatable properties
- List of animatable properties

transition-duration

- Takes a time value
- Possible units are `s` (for seconds) and `ms` (for milliseconds)
- Default value is `0s`

transition-timing-function

- Defines how intermediate values are calculated
- Takes keyword values or function values
- Keyword values
  - `ease` (default, slow start, then fast, slow end)
  - `ease-in` (slow start)
  - `ease-out` (slow end)
  - `ease-in-out` (slow start and end)
  - `linear` (constant speed)
- Function values
  - `cubic-bezier(n,n,n,n)` (fancy stuff)
  - `steps(n)` (flipbook style with n steps)
- Example

transition-delay

- Takes a time value
- Possible units are `s` (for seconds) and `ms` (for milliseconds)
- Default value is `0s`
- Positive values will delay the transition
- Negative values will start the transition immediately and subtract the transition-delay from the transition-duration

Keyframe Animations

Animations

- Animations are defined with keyframes (at least two steps)
- Each keyframe defines the style for an element
- The browser will interpolate between these keyframes

Animation examples

- Leafs
- Circle clone
- Space invader

CSS Animation Properties

- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-timing-function`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-fill-mode`
- `animation-play-state`
- `animation`

Define keyframes (from and to)

```
@keyframes grow-and-shrink {
    from {
        transform: scale(1);
    }
    to {
        transform: scale(2);
    }
}
```

Define keyframes (percentage)

```
@keyframes grow-and-shrink {
    0% {
        transform: scale(1);
    }
    100% {
        transform: scale(2);
    }
}
```

Define keyframes (from, to and percentage)

```
@keyframes grow-and-shrink {
    from {
        transform: scale(1);
    }
    50% {
        transform: scale(.5);
    }
    to {
        transform: scale(2);
    }
}
```

Add animation to element

```
.foo {
 /* animation: grow-and-shrink 1s infinite; */
    animation-name: grow-and-shrink;
    animation-duration: 1s;
    animation-iteration-count: infinite;
}

@keyframes grow-and-shrink {
    /* stuff */
}
```

animation-iteration-count

- Defines how often the animation will be played
- Takes number value or `infinite` keyword
- Default value is `1`

animation-direction

- Defines it an animation plays forwards, backwards or alternating back and forth
- Keyword values are
  - `normal` (default value, from start to end)
  - `reverse` (from end to start)
  - `alternate` (from start to end and end to start)
  - `alternate-reverse` (from end to start and start to end)
- Example

animation-fill-mode

- Defines how the animation applies styles before and after its execution
- Keyword values are
  - `none` (default value, no styles will be applied)
  - `forwards` (keeps styles from last keyframe)
  - `backwards` (applies styles from first keyframe before animation starts)
  - `both` (keeps styles from first and last keyframe are applied)
- Example

animation-fill-mode caveats

- Behavior of `forwards`, `backwards` and `both` depends on `animation-direction` and `animation-iteration-count`
- See MDN docs for deep dive

animation-play-state

- Defines whether an animation is running or paused
- Keyword values are
  - `running`
  - `paused`

Caveats

Pitfalls

- A lot of properties can be animated, but only a few should be animated
  - This is because of how the browser does layouting and rendern
  - Article on Google Developers about browser rendering
  - See Martin do a talk on this topic in an unicorn onsie