CSS Animation

CSE/ISE 102: Introduction to Web Design and Programming
Stony Brook University

Overview of Topics

- CSS Transitions
- CSS Transforms
- CSS Animation

- Reference: Robbins, Chapter 17

Transitions

Transitions

- Transitions allow style changes to fade smoothly from one to another
  - “tweening” fills in the frames between abrupt changes between states

- This CSS feature is still in flux, so browser prefixes (-webkit-, -moz-, -o-, -ms-) are needed

Transition Basics

- We need to know five things:
  - Which CSS property to change
  - How long the transition should take (the duration)
  - How the transition accelerates (its timing function)
  - Whether there should be a pause (delay) before starting
  - What will trigger the transition (e.g., a state change)

Relevant CSS Properties

- `transition-property`
- `transition-duration`
  - measured in seconds (s) or milliseconds (ms)
- `transition-delay`
  - also measured in seconds or milliseconds
- `transition-timing-function`

Timing Functions

- `ease`
  - starts slowly, accelerates quickly, then slows down at the end (default)
- `ease-in`
  - starts slowly, then speeds up
- `ease-out`
  - starts fast, then slows down
- `ease-in-out`
  - similar to `ease` with less acceleration in the middle

Timing Functions (II)

- `linear`
  - consistent speed
- `steps(#, start|end)`
  - divides change into steps, specifies whether change occurs at beginning or end of each step
- `step-start`
  - changes in one step at the start of the duration time
- `step-end`
  - changes in one step at the end of the duration

```
a.smooth
{
    background-color: mediumblue;
    transition-property: background-color;
    transition-duration: 0.3s;
}

a.smooth:hover
{
    background-color: red;
}
```

Transition details

Transition trigger

`transition` Shorthand

- Use the `transition` property to combine all of these properties into a single declaration:

  transition: *property duration timing-function delay;*

- If you provide two time values, the duration must be listed first
- e.g.,

  transition: background-color 0.3s ease-in-out 0.2s;

Multiple Transitions

- Use commas to separate multiple values for each property:

```
transition-property: background-color, color,
letter-spacing;
```

- Values are matched up according to their relative positions in the list
  - This wraps around if one list has fewer values than the others
- This can be done for the shorthand property as well:

```
transition: background-color 0.3s ease-out,
            color 2s ease-in,
            letter-spacing 0.3s ease-out;
```

Universal Transitions

- Use “all” for `transition-property` to apply the same transition effects regardless of which property might change

```
transition: all 0.2s ease-in-out;
```

Transforms

CSS3 Transforms

- We can rotate, relocate, resize, and skew HTML elements in 2D and 3D space
  - We'll focus on 2D transformations for the moment
- Transforms work on all major browsers with the standard browser prefixes

Transformation Types

- Rotation
- Translation
- Scaling
- Skewing
- Matrix transformations (we won't cover these)

The `transform` Property

- Format:

  `transform:` *function(s);*

- When an element transforms, its element box keeps its original position (like with relative positioning)

Rotation

- The `rotate()` function takes an angle specified in positive or negative degrees

```
transform: rotate(-10deg);
```

- By default, the object rotates around its center
  - We can change this…

`transform-origin`

- Use the `transform-origin` property to set the origin point for a transformation
- Options: percentage | length | left | center | right | top | bottom
  - Use two keywords/values for `transform-origin`
- Examples (all three are equivalent):
  - `transform-origin: center top;`
  - `transform-origin: 50% 0%;`
  - `transform-origin: 150px 0;`

Translation

- Translation transforms the position of an element
- Comes in three forms:
  - `translateX(`*length*`)` — horizontal axis
  - `translateY(`*length*`)` — vertical axis
  - `translate(`*translateX*, *translateY*`)` — both axes
    - if you only supply one argument to `translate()`, it will only translate along the X axis

Scaling

- Scaling changes the size of an element
- Three forms (argument is a unitless number):
  - `scaleX()` — horizontal scaling
  - `scaleY()` — vertical scaling
  - `scale(`*x, y*`)` — scales in both directions
    - A single argument will be applied to both directions
- Example: `transform: scale(2, .5);`

Skewing

- Skewing changes the angle of the horizontal or vertical axis (or both) by a specified number of degrees
  - Single arguments to `skew()` are treated like `translate()`
- Three forms:
  - `skewX()` — modifies horizontal axis
  - `skewY()` — modifies vertical axis
  - `skew(`*x*, *y*`)` — modifies both axes
- Example: `transform: skew(15deg, 30deg);`

Applying Multiple Transforms

- You can apply multiple transforms to an element by supplying multiple functions to the `transform` property:

```
transform: rotate(-5deg) scale(1.5)
translate(50px,30px);
```

- Order matters: translate, rotate gives a different result than rotate, translate
- If you want to add a new transform to a different state (i.e., a hover state), you need to repeat all of the original transforms in addition to the new one
  - otherwise, the previous (general) transforms are lost

Smooth Transforms

- We can combine transitions and transforms
- Just use `transform` as the transition property:

```
img { transition: transform 0.3s
linear; }
```

3D Transforms

- CSS3 also includes a way to add space and perspective to a page through 3D transformations
- We won't go into excessive detail on this, but we'll briefly describe one basic example
  - See the Robbins book for links to more detail…

Perspective

- The `perspective` property tells the browser that an element's children should behave as though they are in a 3D space
- The value is a positive integer that specifies a distance from the element's origin on the Z axis
  - good values range from 300–1500
- e.g., `perspective: 600;`

More Perspective Properties

- `perspective-origin: left` | `center` | `right` | `top` | `bottom` | *length* | *percentage*
  - specifies the position of your eyes relative to the transformed elements
- `backface-visibility: visible` | `hidden`
  - determines whether the reverse side of the element is visible when it spins around

3D Transform Functions

- `translate3d()`
- `translateZ()`
- `scale3d()`
- `scaleZ()`
- `rotate3d()`
- `rotateX()`
- `rotateY()`
- `rotateZ()`

Animation

Keyframe Animation

- Keyframe animation lets you specify multiple states along the path of a transition or animation
  - each keyframe is a “point along the way” that specifies the beginning or end of a segment of animation
  - CSS transitions are two-keyframe animations
- Animation is a complex topic; we will only discuss a simple example here
  - See the book for links to good tutorials

The Basic Process

- Start by defining your keyframes (animation segments) with a `@keyframes` rule
- Then add animation properties to the elements that will be animated in each keyframe
- Let’s see an example…

Basic Keyframe Syntax

```
@keyframes animation-name
{
    keyframe { property: value; }
    keyframe { property: value; }
}
```

Establishing The Keyframes

```
@keyframes colors
{
    0% { background-color: red; }
   20% { background-color: orange; }
   40% { background-color: yellow; }
   60% { background-color: green; }
   80% { background-color: blue; }
  100% { background-color: purple; }
}
```

Adding Animation Properties

- These are very similar to the transition properties we've already seen:
  - which animation to use, how long it should take, how it should accelerate, whether to pause before it starts
- Other properties:
  - how many times it should repeat (`animation-iteration-count`)
  - whether it plays forward, in reverse, or alternate directions (`animation-direction`)
  - whether it should be running or paused (`animation-play-state`)
- A shorthand `animation` property is also available

```
#magic
{
  animation-name: colors;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

or:

```
#magic
{
  animation: colors 5s linear infinite
alternate;
}
```

Next Time

- More CSS Techniques
  - Robbins, Chapter 18
- Introduction to JavaScript
  - Robbins, Chapters 19–20