CSS

Cascading Style Sheets

19

Webkits

CSS

Cascading Style Sheets

Mr. K's Technology

CSS

Cascading Style Sheets

Mr. K's Technology

Information for this Unit Was obtained from

Mr. K's Technology

Welcome to HTML.net
Free tutorials on HTML and CSS
http://www.html.net/

Safari Developer Library

tizag.com
unlock your potential

http://www.tizag.com

ABOUT WEB ELEMENTS

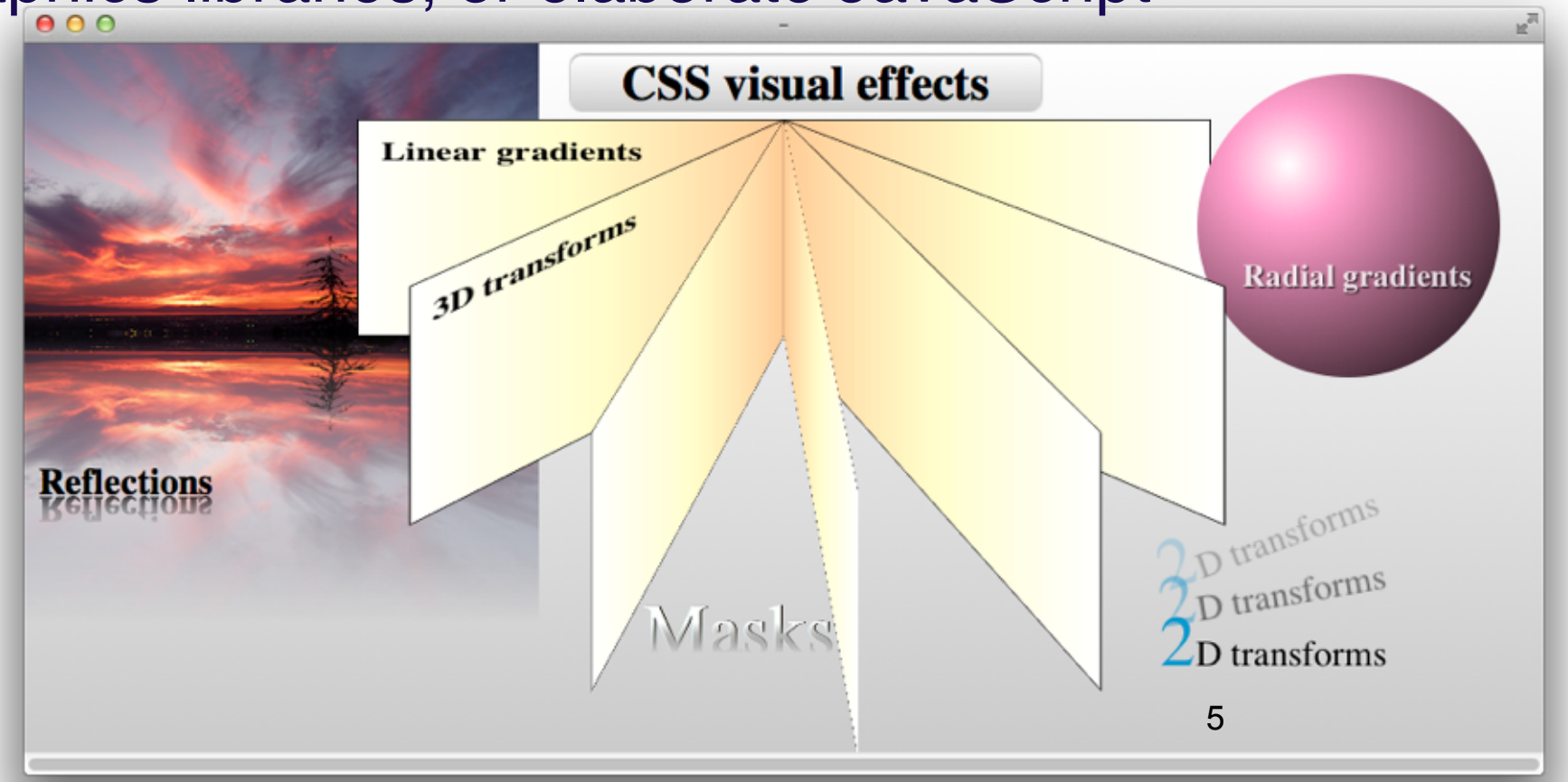
CSS

http://www.w3schools.com/css/

Welcome to the website for the WebKit Open Source Project!

WebKit is an open source web browser engine. WebKit is also the name of the Mac OS X system framework version of the engine that's used by Safari, Dashboard, Mail, and many other OS X applications. WebKit's HTML and JavaScript code began as a branch of the KHTML and KJS libraries from KDE.

Use CSS to create stunning visual effects—masks, gradients, reflections, lighting effects, animations, transitions, 3D rotations, and more. Apply any or all of these effects interactively, triggered by mouse events or touch events; make HTML elements visibly respond to the user—without requiring plug-ins, graphics libraries, or elaborate JavaScript programs.



There are advantages to using CSS instead of graphic images to create visual effects:

- Because they are resolution-independent, CSS effects scale up smoothly when zoomed.
- Text formatted with CSS is searchable; images are not.
- CSS is compact and compresses well compared with graphic images.
- CSS is just text; it can be quickly modified using a text editor or the output of a script.

Safari supports CSS visual effects on Mac OS X and iOS.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

Here's what we are planning to demonstrate

Using Gradients (17),

Using a Gradient as a Mask (18),

Using Reflections (18),

Using CSS Filters (18),

Animating CSS Transitions (19),

Animating With Keyframes (19),

Using 2D and 3D Transforms, and

Adding Interactive Control to Visual Effects.

Animating CSS Transitions

You can create animations entirely in CSS, with no need for plug-ins, graphics libraries, or elaborate JavaScript programs. Normally, when the value of a CSS property changes, the affected elements are re-rendered immediately using the new property value. If you set CSS transition properties, however, any changes in the values of the specified CSS properties are automatically rendered as animations. This kind of automatic animation is called a **transition**.

You trigger a transition simply by changing any of the specified CSS values. CSS property values can be changed by a pseudo-class, such as :hover, or by using JavaScript to change an element's class name or to change its individual CSS properties explicitly. The animation flows smoothly from the original state to the changed state using a transition timing function over a specified duration.

For more complex animations that can use arbitrary intermediate states and trigger conditions, see "Animating With Keyframes" later in this lesson.

Transitions are especially powerful if you combine them with 2D and 3D transforms. For example, Figure 5-1 (next slide) shows the results of applying an animated transition to an element as it rotates in 3D.

Note: Not all CSS properties are animatable. In general, any CSS property that accepts values that are numbers, lengths, percentages, or colors is animatable. Some CSS properties that accept discrete values can be animated as well, but display a jump between values rather than a smooth transition when changed, such as the visibility property.

Setting Transition Properties

To create an animated transition, first specify which CSS properties should be animated, using the -webkit-transition-property property, a CSS property that takes other CSS properties as arguments. Set the duration of the animation using the -webkit-transition-duration property.

For example, Listing 5-1 creates a div element that fades out slowly over 2 seconds when clicked.

Listing 5-1 Setting transition properties

```
<div style=" -webkit-transition-property: opacity; -webkit-transition-duration: 2s;" onclick="this.style.opacity=0">
<p>Click me and I fade away. . .</p>
</div>
```

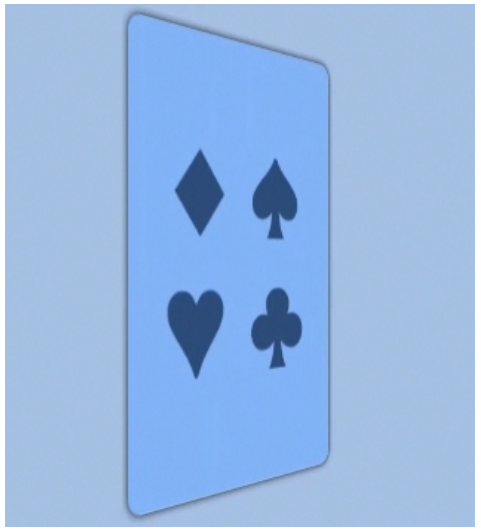


Figure 5-1 Card Flip example

There are two special transition properties: all and none:

- **-webkit-transition-property: all;**
- **-webkit-transition-property: none;**

Setting the transition property to all causes all changes in CSS properties to be animated for that element. Setting the transition property to none cancels transition animation effects for that element.

To set up an animation for multiple properties, pass multiple comma-separated parameters to -webkit-transition-property and -webkit-transition-duration. The order of the parameters determines which transition the settings apply to. For example, Listing 5-2 defines a two-second -background-color transition and a four-second opacity transition.

Listing 5-2 Creating multiple transitions at once

```
div.zoom-fade {
 -webkit-transition-property: background-color, opacity;
 -webkit-transition-duration: 2s, 4s;
}
```

Webkits

All **webkit** elements can be used within an internal style sheet

Using Timing Functions

Timing functions allow a transition to change speed over its duration. Set a timing function using the -webkit-transition-timing-function property. Choose one of the prebuilt timing functions—ease, ease-in, ease-out, or ease-in-out—or specify cubic-bezier and pass in control parameters to create your own timing function. For example, the following snippet defines a 1-second transition when opacity changes, using the ease-in timing function, which starts slowly and then speeds up:

```
style="-webkit-transition-property: opacity; -webkit-transition-duration: 1s; -webkit-timing-function: ease-in;"
```

All **webkit** elements can be used inline within the HTML style tag

Using the cubic-bezier timing function, you can, for example, define a timing function that starts out slowly, speeds up, and slows down at the end. The timing function is specified using a cubic Bezier curve, which is defined by four control points, as Figure 5-2 illustrates. The first and last control points are always set to (0,0) and (1,1), so you specify the two in-between control points. The points are specified using x,y coordinates, with x expressed as a fraction of the overall duration and y expressed as a fraction of the overall change.

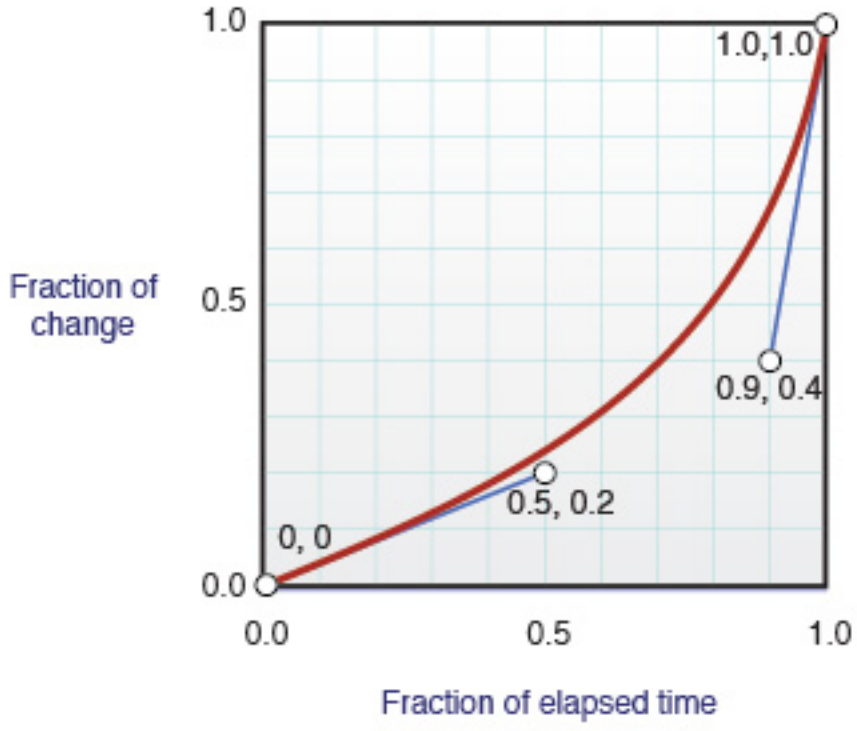


Figure 5-2 Cubic Bezier timing function

For example, Listing 5-3 creates a 2-second animation when the opacity property changes, using a custom Bezier path.

Listing 5-3 Defining a custom timing function

```
div.zoom-fade {
    -webkit-transition-property: opacity;
    -webkit-transition-duration: 2s;
    -webkit-transition-timing-function: cubic-bezier(0.5, 0.2, 0.9, 0.4);
}
```

In the example just given, the custom timing function starts very slowly, completing only 20% of the change after 50% of the duration, and 40% of the change after 90% of the duration. The animation then finishes swiftly, completing the remaining 60% of the change in the remaining 10% of the duration.

Delaying the Start

By default, a transition animation begins as soon as one of the specified CSS properties changes. To specify a delay between the time a transition property is changed and the time the animation begins, use the -webkit-transition-delay property. For example, the following snippet defines an animation that beings 100ms after a property changes:

```
.delay-fade {

    -webkit-transition-property: opacity;

    -webkit-transition-duration: 1s;

    -webkit-transition-delay: 100ms;

}
```

All **webkit** elements can be used from an external style sheet

Setting Several Transition Properties At Once

You can set an animation's transition properties, duration, timing function, and delay using a single shorthand property: -webkit-transition. For example, the following snippet uses the :HOVER pseudostyle to cause img elements to fade in and out when hovered over, using a one-second animation that begins 100 ms after the hover begins or ends, and using the ease-out timing function:

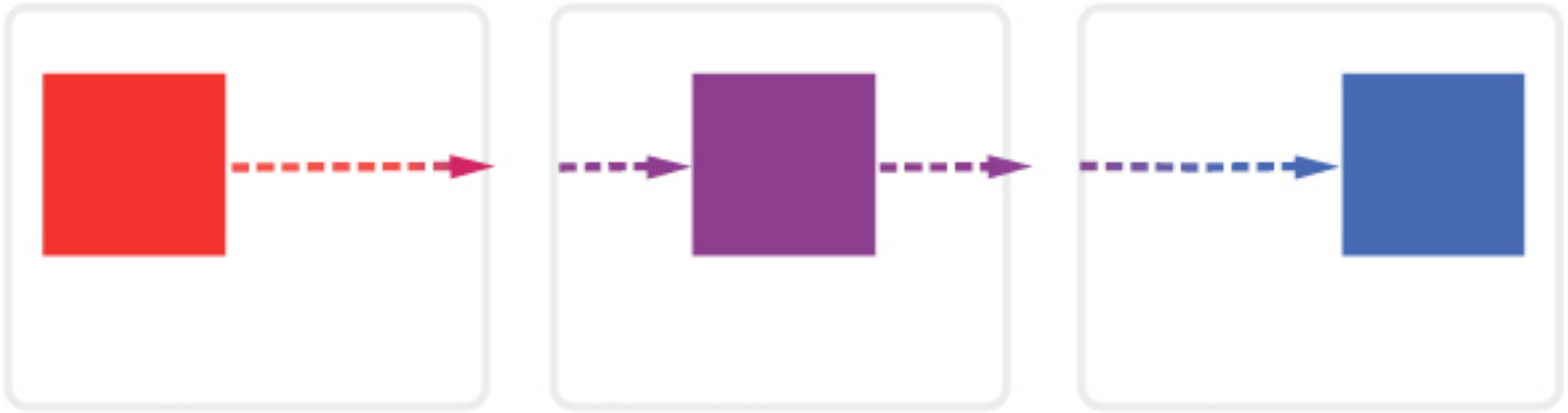
```
<style>

    img {

      -webkit-transition: opacity 1s ease-out 100ms;

         opacity:1; }

    img:hover { opacity:0; }

</style>
```

Handling Intermediate States and Events

Safari Developer Library

When applying a transition to an element's property, the change animates smoothly from the old value to the new value and the property values are recomputed over time. Consequently, getting the value of a property during a transition may return an intermediate value that is the current animated value, not the old or new value.

For example, suppose you define a transition for the left and background-color properties and then set both property values of an element at the same time. The element's old position and color transition to the new position and color over time as illustrated in Figure 5-3. Querying the properties in the middle of the transition returns an intermediate location and color for the element.



To determine when a transition completes, you need to set a JavaScript event listener function for the DOM event that is sent at the end of a transition. The event is an instance of WebKitTransitionEvent, and its type is webkitTransitionEnd.

For example, the snippet in Listing 5-4 displays an alert panel whenever a transition ends. ***We will learn this later.***

Listing 5-4 Detecting transition end events

```
document.body.addEventListener( 'webkitTransitionEnd', function(event) { alert( "Finished transition!" ); }, false );
```


ABOUT WEB ELEMENTS

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

Animating With Keyframes

Safari supports two types of CSS animation: transition animations and keyframe animations. To simply animate changes in CSS properties whenever those properties change, you should probably use an animated transition (see "Animating CSS Transitions").

To create complex animations—such as a bumblebee's flight or a change that starts and stops or repeats indefinitely—or to trigger animations under arbitrary conditions using JavaScript, use keyframe animations.

Keyframe animations include the following features:

- Choose any number of CSS properties to change and define points along a timeline that have specific states.
- Animation between two defined points is automatic but can be guided by specifying a timing function.
- You trigger keyframe animations explicitly.
- Keyframe animations can be set to repeat a finite number of times or to repeat indefinitely, proceeding in the same direction each time, or alternating forward and backward.
- Keyframe animations can be paused and resumed.
- All of the elements in a class can be animated as part of a single animation.

When a keyframe animation completes, an animated element returns to its original CSS style. You can override this behavior, however, and make the final animation state persistent. You can also change an element's style when the animation ends by installing a JavaScript listener function for the webkitAnimationEnd event.

Create stunning effects by combining animation with 2D and 3D transforms. For example, Figure 6-1 shows an animation of text elements in 3D space. See the PosterCircle sample code project for the complete source code for this example.

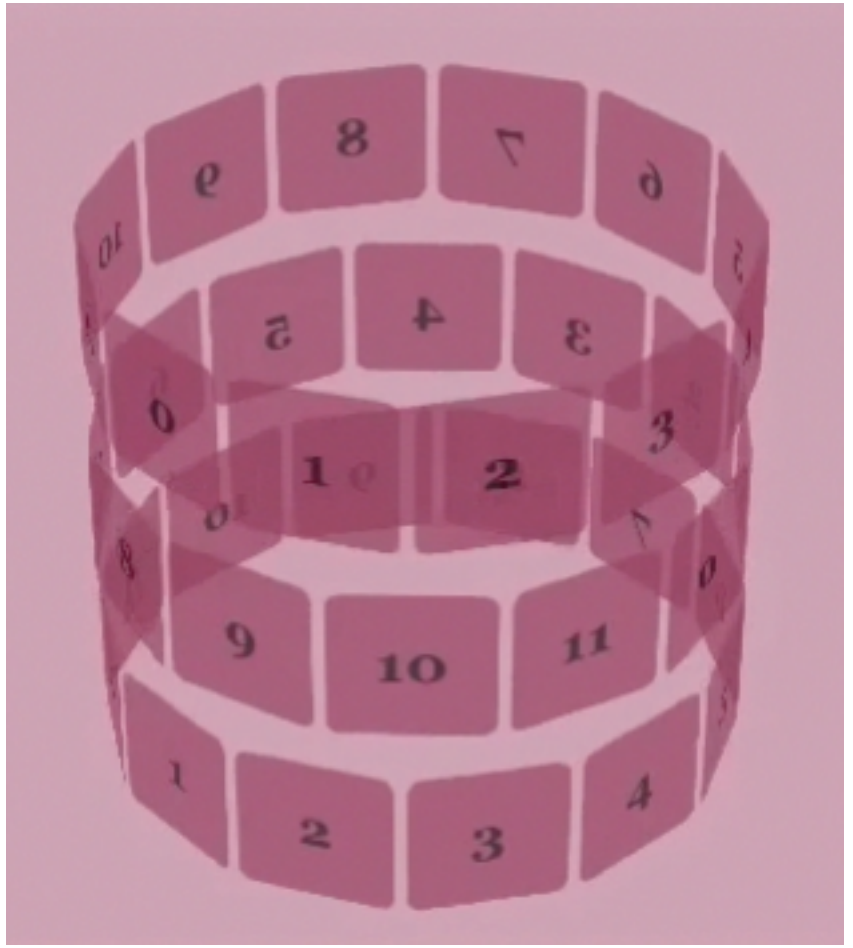


Figure 6-1 Animated text elements

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

Safari Developer Library

Creating a Keyframe Animation

A keyframe animation has—at minimum—a name, a duration, and one or more keyframes. A keyframe is a CSS rule consisting of a set of properties and values associated with a point in time.

To create a keyframe animation, perform the following steps:

1. Create a named set of keyframes in CSS using the @-webkit-keyframes rule. The set must include at least one keyframe.
2. Set the -webkit-animation-duration property to a positive, nonzero value.
3. Set the -webkit-animation-name property to the name of the set of keyframes.

The following example defines a single keyframe in a set named glow. The keyframe specifies a blue background color. The example defines the div:hover pseudo-class as having the animation name glow and the animation duration of 1s. When the user hovers over or touches any div element, it turns blue for one second.

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

Listing 6-1 Simple keyframe animation

```
<html>
<head>
   <title>Blue Glow</title>

<style>

@-webkit-keyframes glow {
   0% { background-color: blue; }
   100% { background-color: blue; }
}

div:hover {
   -webkit-animation-name: glow;
   -webkit-animation-duration: 1s;
}
</style>
</head>

<body>

   <div>      <p>Hover turns me briefly blue.</p>
   </div>

   <div>      <p>Me too.</p>
   </div>

</body>
</html>
```

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

Creating Keyframes

Keyframes are specified in CSS using the @-webkit-keyframes rule. The rule consists of the @-webkit-keyframes keyword, followed by the animation name then by a series of style rules for each keyframe. The style rules are grouped in blocks surrounded by braces, and each is preceded by a relative point in time (typically a percentage of the animation's duration).

Any number of CSS properties can be specified by the style rules.

Listing 6-2 shows a set of keyframes where the animation is named wobble-glow and the keyframes move an element back and forth while changing its background color over time. The left property animates back and forth between 75px and 150px, while the background color cycles between red, white, and blue.

Listing 6-2 Declaring keyframes

```
@-webkit-keyframes wobble-glow {
  0%  { left: 100px; background: red;}
  40% { left: 150px; background: white;}
  60% { left: 75px; background: blue;}
  100% { left: 100px; background: red;}}
}
```

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

The relative point in time is given either as a percentage of the animation's duration or by the keywords from or to . For example, 0% specifies the start of an animation, 50% is halfway through an animation, and 100% is the end of an animation. The from keyword is equivalent to 0% and the to keyword is equivalent to 100%. When the animation executes, it transitions smoothly from one state to the next in increasing order, from 0% to 100%.

One of the CSS properties that you can specify in a keyframe is -webkit-animation-timing-function. This property specifies the rate of change for the animation from the current keyframe to the next keyframe.

Note: *You may specify any CSS properties in the style rules, but you should specify the same set of properties in all of the rules. Exact behavior when the set of properties varies from rule to rule is currently unspecified and subject to change. One exception is the -webkit-animation-timing-function property; if not specified for a particular keyframe, the default timing function is used.*

Setting Animation Properties

You typically set keyframe animation properties by defining a CSS class or pseudo-class, but you can also set the animation properties by using the HTML style attribute in an HTML tag or by setting the style property of an element using JavaScript.

You must set the -webkit-animation-duration and -webkit-animation-name properties in order to see an animation. You can also set the following animation properties:

- -webkit-animation-iteration-count—Sets the number of times to repeat the animation. The default value is 1. Can be set to an integer value or to infinite.
- -webkit-animation-direction—Causes repeating animation to either proceed in the same direction each time or to alternate directions. Can be set to normal (default) or alternate. If set to alternate, the animation goes forward and backward—from 0% to 100% and from 100% to 0%—on alternate iterations. The webkit-animation-iteration-count value must be greater than one1 for this property to have any effect.
- -webkit-animation-play-state—Pauses or resumes an animation. Set this property to paused to pause and or running (default) to continue the animation.
- -webkit-animation-delay—The time to wait between triggering the animation and beginning the animation (default is 0s).
- -webkit-animation-timing-function—The default timing function to use between keyframes. You can override this property on a per-keyframe basis by specifying additional timing functions within keyframes.
- -webkit-animation-fill-mode—Causes the animation to change an element's style before or after the animation runs.By default, an element's style is unchanged by a keyframe animation, both before and after the animation runs.
  If set to forwards, the end state of the animation is persistent.
  If set to backwards, the beginning state of the animation is applied during the delay before starting the animation.
  If set to both, both backwards and forwards are specified.

Animation Timing Functions

Safari Developer Library

ABOUT WEB ELEMENTS

Animation timing functions control an animation's rate of change. A timing function is a mathematical function that provides a smooth curve or path for the rate of change. Specify a timing function using the -webkit-animation-timing-function property. Include the timing function property in a keyframe to specify how to transition from that keyframe to the next. Specify a default timing function to use between all pairs of keyframes by setting the timing function property for an element or a class.

The timing function determines the speed of an animation from the current to the next keyframe. You can specify one of the predefined timing functions:

- ease (default)
- ease-in
- ease-out
- ease-in-out
- linear

You can also specify cubic-bezier as the animation timing function. The cubic Bezier curve is defined by four control points, as illustrated in Figure 6-2. The first control point is always (0,0) and the last control point is (1,1), so you specify the two intermediate control points of the curve.

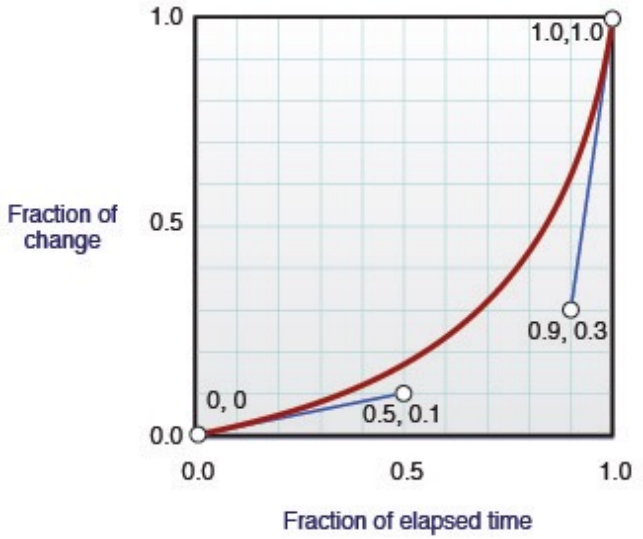


Figure 6-2 Animation timing function control points

The control points are specified as x,y coordinates, where x is a fraction of the duration between keyframes and y is a fraction of the change in properties.

For example, the following line of code sets the second point to (0.5, 0.1) and the third point to (0.9, 0.3) using the cubic-bezier function:

```
-webkit-animation-timing-function: cubic-bezier(0.5, 0.1, 0.9, 0.3);
```

This example causes the animation to begin very slowly, completing only 10% of the change after 50% of the animation and 30% of the change after 90% of the animation. Since the function always ends at the point (1,1), the rest of the animation goes very quickly.

Starting Animations

Safari Developer Library

Once a set of keyframes is created, an animation is triggered when the following conditions are met:

- An element's animation-name property is set to the name of the group of keyframes.
- The element's animation-duration property has been set to a positive nonzero value.
- The element's animation-play-state property is not set to paused.

Consequently, there are several ways to trigger an animation. Here are a few of the most common methods:

- Setting the class name at runtime—Set up the animation as part of an unassigned CSS class, then use JavaScript to assign that class to an object by setting the object's className property at runtime.
- Setting the animation name at runtime—Set up the animation using an element's CSS class definition without specifying the animation name, then trigger the animation by setting the element's style.webkitAnimationName property in JavaScript or by setting -webkit-animation-name property through a CSS pseudoclass.
- Setting the animation play state at runtime—set up the animation using an element's CSS class definition while specifying the animation state as paused, then trigger the animation by setting the element's style.webkitPlayState property to running in JavaScript or setting -webkit-animation-play-state property to running through a CSS pseudoclass.
- Starting out running—for an animation that runs when the page loads, set up the animation as part of a CSS class that includes at least one HTML element.

Listing 6-3 is an example that shows how to apply an animation to an element when the user clicks it.

ABOUT WEB ELEMENTS

Listing 6-3 Starting an animation

```
<html>
<head>
   <title>animated div</title>
<style>
// slide right, then diagonally down and right
@-webkit-keyframes slide {
   0% {top: 0%; left: 0%;}
   50% {top: 0%; left 50%;}
   100% {top: 100%; left: 90%;}
}
// animate over 2s, repeat animation 4 times, back and forth, with linear timing
.div-slide {
   position: relative;
   -webkit-animation-name: slide;
   -webkit-animation-duration: 2s;
   -webkit-animation-timing-function: linear;
   -webkit-animation-iteration-count: 4;
   -webkit-animation-direction: alternate;
}
</style>
</head>

<body>
  <div onclick="this.className='div-slide';">
  <P>Click to slide right and down, then back.</p>
  </div>
</body>
</html>
```

Controlling Animation Using JavaScript

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

Control a CSS animation using JavaScript either by changing an element's className property or by changing an element's animation style properties individually. One thing to remember is that the JavaScript name for a property is not always the same as the CSS name. Compound names in CSS typically contain dashes, but in JavaScript the dash is the subtraction operator, so JavaScript names omit the dash. Compound names in JavaScript usually capitalize the first character of each word after the first word; so if a CSS name were -extension-foo-bar, the JavaScript name would usually be extensionFooBar.

As an example, to set the CSS property -webkit-animation-play-state from JavaScript, set the element's style.webkitAnimationPlayState property.

Listing 6-4 creates three buttons and a round div element with a radial gradient background. The first button animates the div element by changing its class name. The second and third buttons pause and continue the animation by setting the div element's style.webkitAnimationPlayState property.

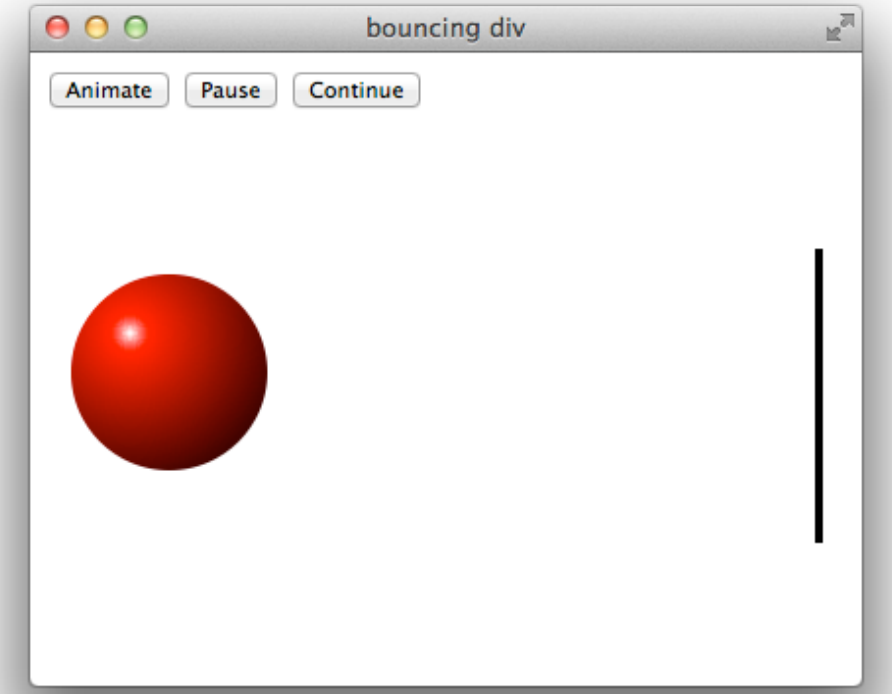


Figure 6-3 illustrates the result.

Listing 6-4 Pausing and continuing an animation

We are introducing these concepts now so when we get to JavaScript it will be easier to understand


```
<html>
<head>
   <title>bouncing div</title>
<style>
@-webkit-keyframes bounce {
   0% {top: 100px; left: 1px; -webkit-animate-timing-function: ease-in;}
   25% {top: 150px; left: 76px; -webkit-animate-timing-function: ease-out;}
   50% {top: 100px; left: 151px -webkit-animate-timing-function: ease-in;}
   75% {top: 150px; left: 226px -webkit-animate-timing-function: ease-out;}
   100% {top:100px; left: 301px;}
   }
.bounce {
   -webkit-animation-name: bounce;
   -webkit-animation-duration: 2s;
   -webkit-animation-timing-function: linear;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: alternate;
   }
</style>
</head>
<body>
<input type="button" value="Animate" onclick= "document.getElementById('ball').className='bounce';">
<input type="button" value="Pause" onclick= "document.getElementById('ball').style.webkitAnimationPlayState='paused';">
<input type="button" value="Continue" onclick="document.getElementById('ball').style.webkitAnimationPlayState='running';"><p>
<div id="ball" style="position:absolute; top: 100px; left: 10px; height:100px; width:100px; border-radius:50px;
background:-webkit-radial-gradient(30% 30%, white, red 10%, black);;">
</div>
<div style="position:absolute; left: 400px; top: 100px; height: 150px; background: black;"><p> 
</div>
</body>
</html>
```

<div style= ***is one HTML line of code***

Handling Animation Events

Safari Developer Library

Three animation-related events are available through the DOM event system:

- webkitAnimationStart—Sent when the animation begins (after any specified delay).
- webkitAnimationIteration—Sent at the end of each iteration.
- webkitAnimationEnd—Sent after the final iteration.

Note that when the final iteration in a series completes, it triggers both a webkitAnimationIteration and a webkitAnimationEnd event.

Each of these events has three significant properties:

- The event.target property is the object being animated.
- The event.animationName property is the name of the animation that generated the event.
- The event.elapsedTime property is the length of time the animation has been running. Elapsed time does not include any time the animation was in the paused play state, nor does it include any specified delay time.

As an example of event handling, Listing 6-5 creates three colored balls (round div elements) that roll away and disappear when clicked. The balls are given a class name that includes animation but begins with the animation play state set to paused; an onclick handler in each ball sets the play state for that element to running. The example adds an event listener for webkitAnimationEnd events on the body element. When a webkitAnimationEnd event occurs, the listener function removes the animated element by changing the element's style.display property to none.

We are introducing these concepts now so when we get to JavaScript it will be easier to understand

ABOUT WEB ELEMENTS

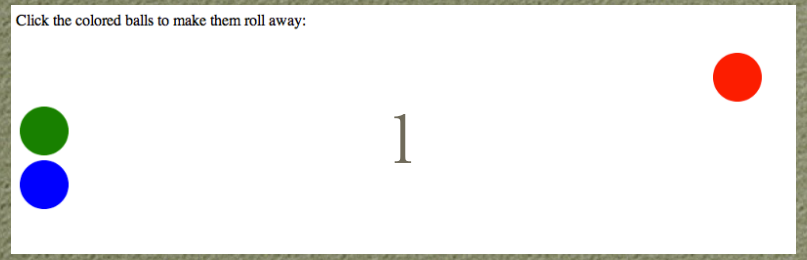
Listing 6-5 Handling animation events

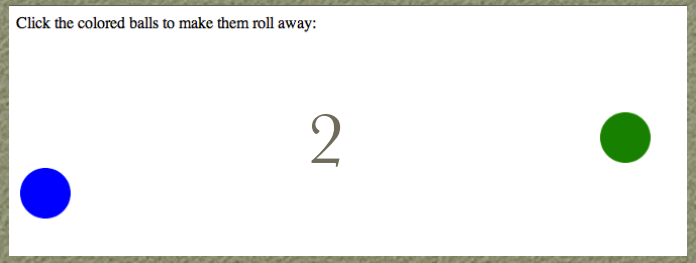
We are introducing these concepts now so when we get to JavaScript it will be easier to understand

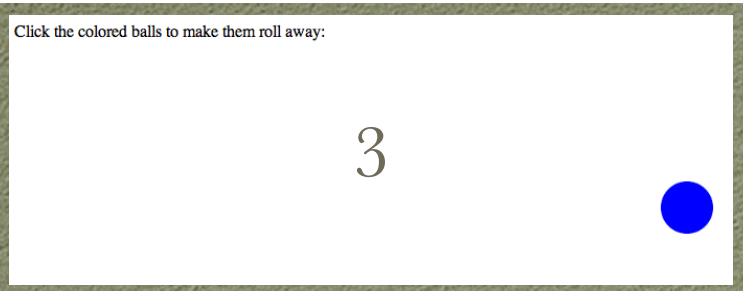

```
<html>
<head>
   <title>rolling divs</title>
<style>
@-webkit-keyframes roll {
   0% {left: 1%; }
   100% {left: 90%;}
   }
.ball {
   position:absolute; left: 3px;
   height:50px; width:50px; border-radius:25px;
   -webkit-animation-name: roll;
   -webkit-animation-duration: 2s;
   -webkit-animation-timing-function: linear;
   -webkit-animation-play-state: paused;
   }
</style>
<script type="text/javascript">
function goAway(event) {
   event.target.style.display="none";
   }
</script>
</head>
<body onload="document.body.addEventListener('webkitAnimationEnd', goAway, false);">
Click the colored balls to make them roll away:
<div class="ball"  style="top:50px; background-color: red;" onclick="this.style.webkitAnimationPlayState = 'running';"></div>
<div class="ball" style="top:105px; background-color: green;" onclick="this.style.webkitAnimationPlayState = 'running';"></div>
<div class="ball" style="top:160px; background-color: blue;" onclick="this.style.webkitAnimationPlayState = 'running';"></div>
</body>
</html>
```

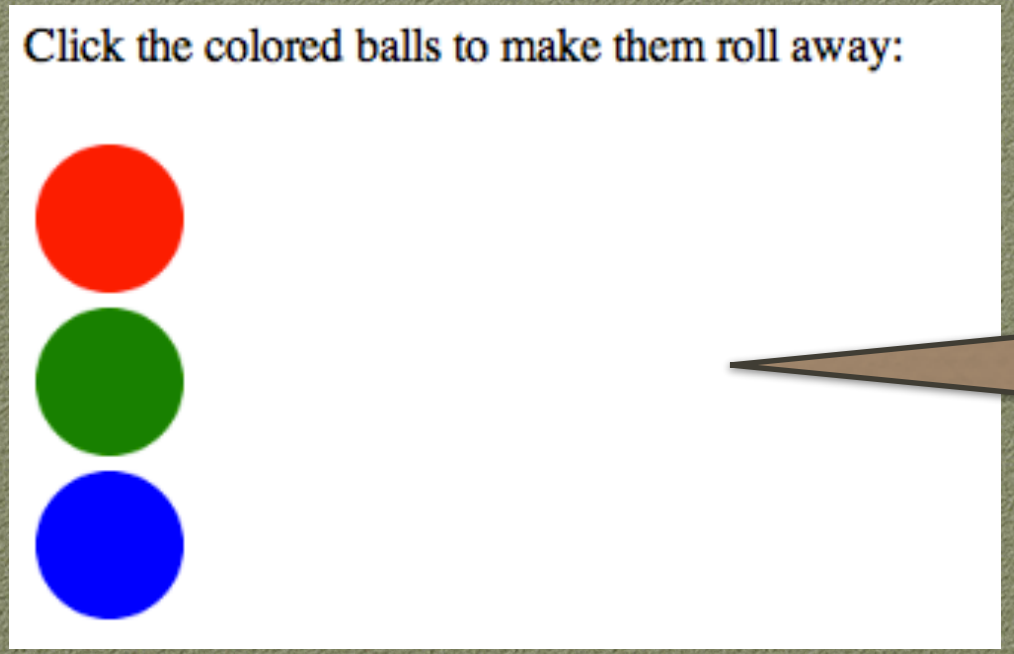
Note: *To see the example in action, copy the listing into a text editor and save it with the .html file extension.*

Safari Developer Library









CSS

Cascading Style Sheets

Mr. K's Technology

Using the information you learned about animating, create the rolling divs page and save with .html

ABOUT WEB ELEMENTS

Webkits

SAVE YOUR WORK

CLOSE ALL OPEN PROGRAMS

ABOUT WEB ELEMENTS

LOG OFF

CSS

Cascading Style Sheets

Mr. K's Technology

Safari Developer Library