CSS3 TransitionsThank God We Have A Specification!

coding.smashingmagazine.com/2013/04/26/css3-transitions-thank-god-specification/

Rodney Rehm

This article is packed with a number of quirks and issues you should be aware of when working with CSS3 transitions. Please note that I'm not showing any workarounds or giving advice on how to circumvent the issues discussed. Alex MacCaw has already written a very insightful and thorough article on "All You Need to Know About CSS Transitions."

Whereas Alex wrote about achieving particular effects, I'm going to talk about the technical background, especially the JavaScript-facing side. Pitfalls — **this article is all about pitfalls**. Here is the table of contents if you would like to skip to a specific section:

Separation of concerns is nothing new — we've been using template engines for years to accomplish exactly that, separating our HTML from whatever scripting language we were using. **A website has three major concerns:** structure (HTML), layout and style (CSS), and behavior (JavaScript). CSS crossed the line and became behavioral quite a while ago, but that's a whole different discussion.

A couple of weeks ago, I was tasked with developing a JavaScript module that would allow for the use of CSS transitions in a way that the JavaScript side would know nothing about the transitions taking place. **The actual problem is the asynchronousity of transitions.** After writing a bunch of tests, I gave up on the task. It cannot be done with a reasonable amount of code and initialization time. My test results are what this article is all about.

This is fine for properties such as font-size, which take only one argument and are reliably converted to pixel values. However, it doesn't cover how browsers should handle shorthand properties, such as margin — some browsers return nothing, others something semi-useful. Then there are properties with different but equivalent values to consider, such as font-weight's bold and 700. WebKit also has a bug that extracts the computed value of properties from pseudo-elements.

The quirks described here were identified in January 2013 using Firefox 18 (Gecko), Opera 12.12 (Presto), Internet Explorer 10 (Trident), Safari 6.0.2 (WebKit), Chrome 23 (WebKit), as well as Gecko's and WebKit's nightly build channels.

Without further ado, **let's dive into the specifications and implementations**, a world riddled with misconceptions. Please note that in order to be concise, I've omitted vendor prefixes from the examples.

> *"Not knowing is difficult to handle. It's easier to assume."*
>
> *– Dr. Axel Rauschmayer*

… But assumptions are often wrong. I discovered the information in this article by creating a CSS3 Transitions Test Suite.

1. Specifying A Transition

Besides the shorthand transition property, the CSS3 transition specification defines the following four CSS properties for specifying an animated change of state:

- transition-property,
- transition-duration,
- transition-delay,
- transition-timing-function.

CSS Properties to Transition

The transition-property property defines the **property (or properties) to animate**. The default is all, meaning that all properties a browser can transition will be animated on change (if there's a transition-duration greater than 0s). The property accepts one value or a list of comma-separated values (like all other transition-* properties).

The specification states that a browser should accept and preserve any property it doesn't recognize. So, the following example would still run a transition on padding lasting 2 seconds:

```
transition-property: foobar, padding;
transition-duration: 1s, 2s;
```

Contrary to the specification, WebKit parses the above to transition-property: all. Firefox and Opera parse it to transition-property: all, padding.

Duration of a Transition

The transition-duration property defines the **amount of time** a transition should take to get from the initial state to the target state. It accepts a <time> value in seconds or milliseconds (for example, 2.3s and 2300ms both specify 2.3 seconds).

While the specification makes it clear that values must be a positive number, Opera also accepts -5s — at least for getComputedStyle(). Opera and Internet Explorer (IE) do not accept values lower than 10ms, although the specification mentions no such limitation. In all fairness, you wouldn't notice a transition lasting 9 milliseconds anyways. WebKit (except for the current WebKit nightly) has a bug in its getComputedStyle() implementation, returning values such as 0.009999999776482582s instead of 0.01s. At least all browsers agree on returning second-based values.

Delay of a Transition

The transition-delay property defines the **time to wait** before executing a transition, also using <time> values. The delay may be a negative value, which will start the transition immediately and make it appear as though the transition had started at the given offset in time — essentially starting with a jump.

As with transition-duration, IE and Opera don't accept values between -10ms and 10ms. WebKit's floating point issues appear here, too.

Timing Functions

The transition-timing-function property defines the **mathematical function used to calculate a property's value** at time t. There are three basic types: cubic-bezier(x1, y1, x2, y2), step(<number>, start|end), and keywords that map to predefined cubic bezier curves. Most likely, you already know the keywords linear, ease, ease-in, ease-out and ease-in-out. The math behind cubic beziers gets ridiculously unimportant when using Lea Verou's charming little Cubic Bezier Editor. While cubic bezier curves make smooth transitions, the step() functions don't. They instead jump to the next value (i.e. the next step) at a regular interval. This allows for frame-by-frame animations; see "Pure CSS3 Typing Animation With steps()" for an example.

The computed value of linear is usually represented as cubic-bezier(0, 0, 1, 1) — except for WebKit, which actually returns linear. But not to worry: WebKit will still return cubic-bezier(0.25, 0.1, 0.25, 1) instead of ease. The current WebKit nightly returns the keyword for all defined keywords, though. Looking on the bright side, in a couple of months WebKit won't be inconsistent with itself — only with the rest of the browser world.

The specification stipulates that the x values must be between 0 and 1, while the y values may exceed that range. Contrary to the specification, WebKit allows x to exceed the bounds, at least computationally. At the time of writing, the Android browser (version 4.0) mixes up ranges for x and y, essentially disallowing "bounce" effects.

2. When A Transition Is Complete

I already mentioned that CSS transitions run asynchronously. The specification provides the TransitionEnd event to allow JavaScript to synchronize with the end of a transition. Sadly, the specification isn't very specific about this event. In fact, it simply states that an event is to be fired for every property that has undergone a transition. If you

need a single word to describe the situation, “nightmare” isn’t far off.

While the specification says that shorthand properties (such as padding) should run transitions for all properties that it covers (padding-top, padding-right, etc.), it doesn’t say which property should be named in a TransitionEnd event. While Gecko, Trident and Presto agree on triggering events for the longhand sub-properties (such as padding-top), even if a transition was defined for a shorthand property (such as padding), **WebKit would take the opportunity to screw things up.** WebKit would trigger an event for padding if (and only if) you specified transition-property: padding, but transition-property: all would trigger the event for padding-left et al. For some reason, iPhone 6.0.1’s Safari browser *might* also triggers events for font-size and line-height when padding is being transitioned. Confused yet?

```
.example {
 padding: 1px;
 transition-property: padding;
 transition-duration: 1s;
}
.example:hover {
 padding: 10px;
}
```

The above CSS will trigger different TransitionEnd events across browsers:

Gecko, Trident, Presto
    padding-top, padding-right, padding-bottom, padding-left
WebKit
    padding

```
.example {
 padding: 1px;
 transition-property: all, padding;
 transition-duration: 1s;
}
.example:hover {
 padding: 10px;
}
```

The CSS above will trigger different TransitionEnd events across browsers:

Gecko, Trident, Presto, WebKit
    padding-top, padding-right, padding-bottom, padding-left
Safari 6.0.1 on iPhone (not iPad, mind you!)
    padding-top, padding-right, padding-bottom, padding-left, font-size, line-height

I said that you could specify a negative transition-delay to “jumpstart” your transition. But what happens for transition-duration: 1s; transition-delay: -1s;? Gecko and WebKit immediately jump to the target value and trigger an event. Trident and Presto won’t trigger any events.

That floating point issue that WebKit experiences in getComputedStyle() is also present in TransitionEnd.elapsedTime — consistently in all browsers. Math.round(event.elapsedTime * 1000) / 1000 will “fix” that for you.

WebKit and IE have implemented an unspecified extension to background-position that causes them to trigger TransitionEnd events for background-position-x and background-position-y, instead of background-position.

So, even if you *knew* that a transition was taking place, you wouldn’t be able to rely on the TransitionEnd.propertyName that you’re given. While you *could* write loads of JavaScript to equalize the behavior, you wouldn’t be able to do this in a future-proof way without doing proper feature detection for every single property. And this could include properties you might not even know are animatable.

3. Transitionable Properties

The specification lists a number of CSS properties that a browser is supposed to support animated transition for.

This list contains properties of CSS2.1. Any of the newer properties will be marked as animatable in their respective specifications — as order of the Flexible Box Layout shows.

The property's value type is an important factor. The property margin-top accepts <length> and <percentage> values, but according to the list of transitionable CSS properties, only <length> is to be animated. But that didn't keep browser vendors from implementing transitions for <percentage> values anyway. The word-spacing property is a different story, though. The specification includes <percentage> values, but at the time of writing, no browser is able to animate that.

Ignoring the (inherently unreliable) TransitionEnd events, a property is transitioned from value A to value B if its getComputedStyle() value is different from A and B at a given time during the transition. Because there is no such thing as a "CSS property value changed" event, **you're left with polling the DOM**. setTimeout()'s resolution is not good enough to do this for fast transitions (a duration of less than a few hundred milliseconds). requestAnimationFrame() is your friend for this. The browser will call you before it repaints to screen, allowing you to grab a couple of intermediate values during transitions. Except for Opera, all engines have this feature already.

Instead of bloating this article with a full compatibility table, I've sent my results to Oli Studholme (@boblet), who has updated his list of "CSS Animatable Properties" accordingly.

4. Priority Of Transition Properties

The specification on the transition-property property states that we're allowed to define a property multiple times:

> *"If a property is specified multiple times in the value of 'transition-property' (either on its own, via a shorthand that contains it, or via the 'all' value), then the transition that starts uses the duration, delay, and timing function at the index corresponding to the last item in the value of 'transition-property' that calls for animating that property."*

So, we can make padding transition for 1 second, while making padding-left take 2 seconds; or define a default transition style using transition-property: all and overwrite that for particular properties.

In Firefox and IE, this works fine. Opera mixes up the priority order, though. Instead of simply using the *last* applicable property in the list, it treats padding-left as more specific than padding and all.

The real problem is WebKit. It's somehow managed to execute a transition multiple times if a property is specified multiple times. To really freak out WebKit, try running a transition for transition-property: padding, padding-left with the very small transition-duration: 0.1s (warning: this is not a good idea for epileptics). WebKit will *render* the transition at least twice. But the real beauty is the TransitionEnd events, of which you could receive up to *hundreds* for a single transition.

5. Transitioning From And To auto

The CSS property value auto translates to "Dear browser, please calculate some reasonable value for this." Paragraphs (<p>) and any block-level elements will be as wide as their parent if they have width: auto. There are times when you'll change from width: auto to a specific width — and want to transition that change. The specification neither enforces nor denies the use of auto values for transitionable properties.

Firefox, IE and Opera cannot transition from or to auto values. IE makes a little exception for z-index, but that's it. WebKit, on the other hand, is capable of transitioning from and to pretty much any CSS property that accepts the auto value. WebKit doesn't like clip too much; for that property, it will only trigger a TransitionEnd event, without generating or showing any intermediate values or states during the transition.

For the other properties, such as width and height, WebKit's behavior is not quite what you'd expect. If width: auto translated to a calculated width of 300px and you transitioned that to 100px, then your transition would not shrink from 300 to 100 pixels. Instead, it would grow from 0 to 100 pixels.

For a full compatibility table, have a look at "CSS Animatable Properties."

6. Implicit Transitions

An “implicit transition” happens when a change to one property causes another property to be transitioned — or if you change a property on a parent element and cause a child to transition either the inherited property or a dependent property. Confused? Consider `font-size: 18px; padding: 2em;` — the padding is calculated as 2 × font-size, because that's what em does, giving us 36 pixels.

There are various **relative value types**: `<percentage>`, `<length>`, `em`, `rem`, `vh`, `vw`, etc. Using a relative value, such as `padding: 2em`, makes the browser recalculate the property's `getComputedValue()` every time its depending value (such as `font-size`) changes. That in turn triggers a transition for `padding` because the computed style has changed. This transition is considered to be “implicit” because the `padding` property was not modified explicitly.

Most browsers run these implicit transitions. The exception is IE 10, which runs them only for the `line-height` property. WebKit runs implicit transitions for all applicable properties except `vertical-align`. Besides font-relative property values, there are width-relative property values (usually `<percentage>`), viewport-relative property values (such as `vh` and `vw`), default initial values (such as `column-gap: 1em` in Opera), and then currentColor. All of these might — or might not — trigger implicit transitions.

In Firefox, these implicit transitions get particularly interesting when both the depending and the dependent properties are transitioned but their `transition-duration` or `transition-delay` do not match. While WebKit and Opera produce transitions that make sense visually, Firefox garbles things a bit. In IE, this is a non-issue because it doesn't do implicit transitions.

Don't forget about inheritance within the cascade. A `font-size` on a DOM element will be inherited by its children, as long as it's not overwritten, potentially causing implicit transitions.

7. Transitions And Pseudo-Elements

Firefox and IE 10 will transition properties on pseudo-elements. Opera, Chrome and Safari will not. WebKit added support in January 2013 — which you can already check out in WebKit nightly and Chrome Canary.

Transitions of generated content bring their own set of funky issues. TransitionEnd events aren't fired at all. At some point in the future, they're supposed to be triggered on the owner element and provide their pseudo-element through TransitionEnd.pseudoElement. But even the “Transition Events” section of the “CSS Transitions” editor's draft doesn't specify that properly yet.

There was a time when we would change the value of the `content` property so that IE 8 would re-render that element in certain circumstances (like when entering a `:hover` state). It turns out that this fix for old IE interferes with this ability for all other browsers. So, when trying to transition a property on a pseudo-element, make sure the `content` is not changed.

IE 10 will not run a transition for a pseudo-element's `:hover` state if the owner element doesn't have a `:hover` state as well:

```
.some-selector:before {
 content: "hello";
 color: red;
 transition: all 1s linear 0s;
}
.some-selector:hover:before {
 color: green;
}
/* This next rule is necessary for IE 10 to transition :before on hover */
.some-selector:hover {}
```

The weird thing about this issue isn't that you need a (possibly empty) `:hover` state on the owner element. It's that if you don't have one, IE 10 will interpret the `:hover` as `:active` (i.e. active when you mousedown on the element). The even weirder part is that the `:active` state persists even after `mouseup` and is removed only by another click on the document.

8. Background Tabs

At the time of writing, IE 10 is the only browser that responds to a tab being in the background or foreground. While it

will finish a running transition if the tab is pushed to the background, it won't start any new transitions there. IE 10 will wait until the tab is pulled into the foreground before starting any new transitions. Fortunately, IE 10 already supports the Page Visibility API, allowing developers to respond to this behavior.

We can expect similar things to happen with other browsers as they continue putting background tabs to sleep.

9. "Invisible" Elements

So, are transitions executed for DOM elements that are not attached to the DOM? Nope, not a single browser does that — why should they? Well, then, what about hidden elements? Most browsers have figured out that there's no need to run a transition on an invisible (i.e. not painted) element. Opera thinks differently about this — it'll run a transition regardless of whether it is painted or not.

10. Transitioning Before The DOM Is Ready?

The `DOMContentLoaded` event is triggered when the document leaves parsing mode. If you're into jQuery, we're talking about `jQuery.ready()` right now. Transitions can be run *before* this event happens.

11. Rendering Quirks

The issues I've described up to this point were found by testing against the specification. The tests were run automatically. But as it turns out, quite a few more problems are visible to the eye. The following quirks have been found by various other developers and could affect your meddling with transitions just as much.

At this time, transitioning a background from gradient to gradient is not possible. Transitioning from gradient to solid color is possible — with a big caveat. If a gradient is in play, then the color transition will happen from white to the target color, appearing to quickly flash white at the beginning of the transition. This can be observed in all current browsers.

Firefox seems to be using a different algorithm for rendering (or smoothing) images as they're being animated (see an example). Apparently, Gecko sacrifies quality for performance during animation. Note that this occurs if a low enough `transform: scale()` is in play.

Firefox won't properly animate from `a:visited` to `a:hover` or vice versa. Instead, it will jump from `a:visited` to `a:link` and then transition to `a:hover`, as you can see in this example. This is mentioned somewhat in "Privacy and the :visited Selector" on the Mozilla Developer Network. While IE 10 agrees with Chrome, Safari and Opera on the proper transition, it also runs the transition from `a:link` to `a:visited` on page load.

Transitioning multiple properties is not synchronized in Firefox and Webkit. You can see in this example how making the `border` smaller by the same amount that the `padding` increases (and vice versa) causes the following content to shake a bit. IE 10 and Opera get this right.

Firefox won't animate an element's properties if one of its parent's `position` is changed, as you can see. Webkit, Opera and IE 10 behave correctly.

12. Recommendations For The Specification

Having read the specification from top to bottom and actually tested all of the features, I think a few changes would help:

- Introduce a `TransitionsEnd` (notice the plural), triggered once *all* transitions for an element have completed. It could provide a list of properties that have been animated — but I don't see the use case for knowing *what* has transitioned, as long as I'm informed *when* all animations are done.
- Introduce a `TransitionStart` event, triggered for every property about to be transitioned. Because DOM events don't come cheap and the JavaScript event loop and the rendering thread are not necessarily blocking each other, a single event `TransitionsStart` (there is that plural again) might be the better solution. I don't see why I should be able to `cancel` the event, so this would be a "fire and forget" kind of thing.
- Make it clear what `TransitionEnd` is supposed to be triggered for. That `padding` versus `padding-left` issue in WebKit is rather annoying.
- Clearly specify how "implicit transitions," such as `line-height: 1em` for `transition-property: font-size`, are to be handled.

- Add a `::transitioning` pseudo-class that allows you to define `pointer-events: none` to prevent accidental hover states (among other things). The trick here is to prevent the application of styles that themselves would trigger a new transition or that would alter an already running transition.

In addition to these suggestions, we should be able to accomplish a number of common (simple) things without having to throw a lot of JavaScript at the problem:

- Every once in a while, you'll want to completely mute all transitions — for example, because you're changing the layout and need to calculate dimensions and positions before unleashing your beautiful transitions upon the visitor.
- Sometimes you'll want to remove an object from the DOM and want that to be animated. Right now, you'd add a class, wait for the TransitionEnd event and then remove the element.
- Just as with removing things, you'll want to add a new element and animate its appearance. Right now, you have to insert the element, set some "invisible style," force a repaint and then revert to the new element's actual style.
- Reordering, hiding and showing elements are common for any Web application. Giving that task a little style currently requires us to run utilities such as Isotope. A vanilla CSS solution could shave off some bytes.

13. Use The `delay`, Luke!

Imagine a number of elements packed together tightly. Imagine that the styles of those elements change on hover. Imagine moving your cursor (moderately quickly) over that group. What happens? Exactly: you'll see the styles of those elements flash.

By adding a relatively short delay to your transitions, you can mitigate that effect; 20 milliseconds is undetectable to the human eye, but it's enough for the mouse cursor to pass over small elements. The transitions won't appear to lag because of this, and the visual distraction you might have caused just disappears. Simple trick, I know.

14. Conclusion

- Be very careful when using `transition-property: all`. You *will* get TransitionEnd events for properties that you didn't expect to ever transition.
- Be careful when using shorthand properties, because the number of triggered events varies between browsers.
- Opera and IE don't trigger events when a negative delay cancels out the duration.
- WebKit has real issues with the priority of properties such as `transition-property: margin, margin-left`. Avoid this for now.
- IE doesn't support implicit transitions — for example, triggered for `padding: 2em` when `font-size` changes.
- Firefox and Opera cannot parse `transition-property: all, width`.
- Opera mixes up the priority of properties.
- Transitions on pseudo-elements do not trigger TransitionEnd events.
- IE 10 has a weird `:hover` bug when transitioning pseudo-elements.
- The specification leaves a lot of room for improvement.

Related Content

If you're interested in transitions and animations — and how to use them wisely — have a look at these fantastic resources:

(al)

Rodney Rehm

Rod is a Web Developer based in Lottstetten, Germany. He is mainly developing in PHP and JavaScript. He's quite decent with databases and distributed systems for scalability and redundancy. Created URI.js, jQuery Context Menu, Apple's PList in PHP and works on Smarty - PHP Template Engine.