The CSS Transitions Cheatsheet

kirupa.com/html5/css_transitions_cheatsheet.htm

by kirupa | 22 August 2015

When it comes to **CSS Transitions**, you don't need a bazillion pages of content or even a book to quickly figure out something. If you are in a rush or just want something you can refer to really quickly, I've created this handy cheatsheet that provides snippets for common CSS Transitions situations. If you find something missing that should be here, post here and I'll get to it shortly :P

Enjoy!

The Topics

Click on any of the topics below to jump directly to the relevant section:

Simple Example

When a user hovers over the #box element, the change in position is animated:

```
#box {
    transform:translate3d(0,-350px,0);



    transition:transform .5s ease-in;
}
#box:hover {
    transform:translate3d(0,0px,0);
    cursor:pointer;
}
```

More Details

The Bare Minimum

To define a transition that works, you don't need to be verbose:

```
transition: .5s;
```

```
transition:all .5s ease-in;
```

Longhand Declaration

The transition property values can be expanded into their individual properties:

```
transition-property:all;

transition-duration: .5s;

transition-timing-function:ease-in;

transition-delay: .1s;
```

There is no "right" or "wrong" preference when choosing between the shorthand version and longhand version. Use whatever you like.

More Details

Vendor Prefixes

A small number of users will be able to view transitions only if you use vendor prefixes:

```
-webkit-transition:all .5s ease-in;

-moz-transition:all .5s ease-in;

-o-transition:all .5s ease-in;

transition:all .5s ease-in;
```

In general, use the -prefix-free library instead of wasting time duplicating code.

Note

Despite what earlier versions of this page said, there is no -ms-transition property. Internet Explorer 10 was the first version to support this property, and it supported it unprefixed right out of the gate. Thanks to Gunnar Bittersman for pointing that out!

More Details

Easing Functions

By default, your transitions will have the **ease-in** ease specified. You can specify other ones easily:

```
transition: color2s ease-in .5s;

transition: color2s ease-out .5s;

transition: color2s ease-in-out .5s;

transition: color2s linear .5s;

transition: color2s ease-in .5s;

transition: color2s step-start .5s;

transition: color2s ease-end .5s;

transition: color2s steps(3, start) .5s;

transition: color2s cubic-bezier(.70, .35, .41, .78) .5s;
```

The **cubic-bezier** ease is awesome, and use a site like cubic-bezier.com to visually define and test your own easing values.

More Details

Specifying Multiple Transitions

In the shorthand world, don't declare multiple transition properties. Instead, declare a single transition property and comma separate the values you'd pass in to them:

```
transition: width .5s ease-in, border-radius1s linear;
```

For the longhand version, you do something similar except each property has its own collection of comma separated values:

```
transition-property: width, border-radius;

transition-duration: .5s,1s;

transition-timing-function:ease-in,linear;
```

More Details

Listening to Multiple Properties

This only works in the longhand version. Just add all the properties you wish to listen for to the transition-property:

```
transition-property: width, border-radius, background-color;

transition-duration: .5s;

transition-timing-function:ease-out;
```

More Details

Delaying the Transition

You can specify how long you want your transition to wait before becoming active:

```
transition: color2s ease-in .5s;
```

More Details

Starting from the Middle

You can specify a negative offset value that specifies a point in time in the middle of the transition to start from:

```
transition:all 2s ease-in -.5s;
```

In this example, the transition will **start** from the .5 second mark and run for the remaining 1.5 seconds.

More Details

Animating Movement Smoothly with Hardware Acceleration

If you are adjusting the movement of an element, use the **translate3d** transform to ensure you get hardware acceleration:

```
.pictureContainer img {
    position:relative;
    top:0px;
    transition:transform .2s ease-in-out;
}
.pictureContainer img:hover {
    transform:translate3d(0px,-150px,0px);
}
```

Unless you can't avoid it, don't animate an element's position via margin, padding, top, left, right, bottom, **translate** transform, or any other CSS property you have for making things move.

More Details

Listening to the transitionend Event

The transitionend event is fired when a transition runs to completion:

```
var blueCircle = document.querySelector("#blueCircle");


blueCircle.addEventListener("transitionend", detectTheEnd,false);
blueCircle.addEventListener("webkitTransitionEnd", detectTheEnd,false);
blueCircle.addEventListener("mozTransitionEnd", detectTheEnd,false);
blueCircle.addEventListener("msTransitionEnd", detectTheEnd,false);
blueCircle.addEventListener("oTransitionEnd", detectTheEnd,false);


function detectTheEnd(e) {


}
```

More Details

Multiple Transitions and the transitionend Event Handler

If the element that fires the transitionend event has multiple transitions defined, you can identify which transition to listen to by checking the value of the event argument's propertyName property:

```
function detectTheEnd(e) {
    if (e.propertyName =="opacity") {

    }else if (e.propertyName =="transform") {

    }
}
```

More Details

Looping a Transition

There is no built-in way to loop a transition. You have to listen to the transitionend event and change the animated property values between an initial value and a final value. Imagine you have some CSS that looks as follows:

```
#circleDiv {
    transition:transform .2s ease-in-out,opacity .2s ease-in-out;
}
    .stateOne {
        opacity:1;
        transform:scale(1,1);
    }
    .stateTwo {
        opacity: .5;
        transform:scale(1.9,1.9);
    }
```

The JavaScript that will help you jump between the initial and final values with each transitionend firing is:

```
var theCircle = document.querySelector("#circleDiv");

function setup() {

    theCircle.addEventListener("mouseover", setInitialClass,false);

    theCircle.addEventListener("transitionend", loopTransition,false);
    theCircle.addEventListener("webkitTransitionEnd", loopTransition,false);
    theCircle.addEventListener("mozTransitionEnd", loopTransition,false);
    theCircle.addEventListener("msTransitionEnd", loopTransition,false);
    theCircle.addEventListener("oTransitionEnd", loopTransition,false);
}
setup();

function setInitialClass(e) {
    theCircle.className ="stateTwo";
}

function loopTransition(e) {
    if (e.propertyName =="opacity") {
        if (theCircle.className =="stateTwo") {
            theCircle.className ="stateOne";
        }else {
            theCircle.className ="stateTwo";
        }
    }
}
```

The loopTransition event handler contains the lines of code that oscillate the CSS property values between what gets represented with stateOne and stateTwo.

More Details

Conclusion

This cheatsheet is a work in progress. If there additional snippets you'd like me to add or changes you'd like me to make, post them in the forums below.

If you have a question about this or any other topic, the easiest thing is to drop by our forums where a bunch of the friendliest people you'll ever run into will be happy to help you out!

THE KIRUPA NEWSLETTER

Get cool tips, tricks, selfies, and more...personally hand-delivered to your inbox!

(View past issues for an idea of what you've been missing out on all this time!)

Kirupa