Week 6

# CSS3 ANIMATIONS & More API

# CSS3 FOR ANIMATION

# VENDOR PREFIXES

I know they are annoying, but sometimes you have to use them.

Use Can I Use to check for feature/prefix compatibility.

# TRANSITION

(https://developer.mozilla.org/en-US/docs/CSS/transition)

Syntax:

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

Default Values (you can also use these individually):

```
transition-delay: 0s
transition-duration: 0s
transition-property: all
transition-timing-function: ease
```

Example:

```
transition: opacity .3s ease; /* Omitting delay */
```

# TRANSFORM

(https://developer.mozilla.org/en-US/docs/CSS/transform)

Syntax:

```
transform: transform-function ...
```

Commonly used transform functions:

```
translate(), rotate(), scale(), skew()
translate3d(), rotate3d(), ... /* more in the link */
```

Example:

```
transform: rotate(20deg) scale(1.5);
```

# ANIMATION

(https://developer.mozilla.org/en-US/docs/CSS/animation)

Syntax:

```
animation: name duration timing-function delay ...
```

Default Values:

```
animation-name: none
animation-delay: 0s
animation-timing-function: ease
animation-duration: 0s
animation-iteration-count: 1
animation-direction: normal
animation-fill-mode: none
```

# ANIMATION

(https://developer.mozilla.org/en-US/docs/CSS/animation)

Example:

```
@keyframes example {
    from {
        margin-left:-20%;
    }
    to {
        margin-left:100%;
    }
}

.animated {
    animation: example 4s linear 0s infinite alternate;
}
```

# TIMING FUNCTIONS

(https://developer.mozilla.org/en-US/docs/CSS/transition-timing-function)

Built-in Functions:

ease
ease-in
ease-out
ease-in-out
linear
step-start
step-end
steps()

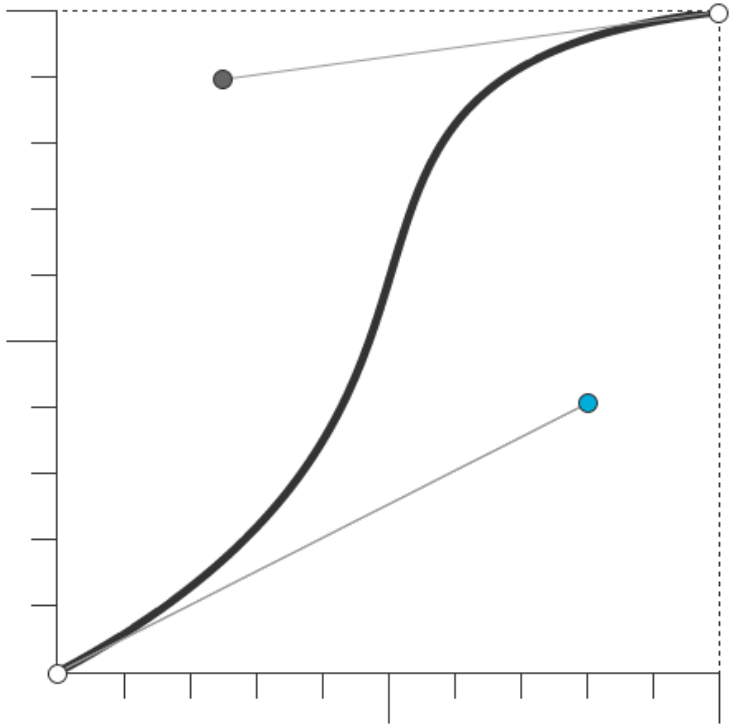
# TIMING FUNCTIONS

This might be more useful:

```
cubic-bezier(p1x, p1y, p2x, p2y)
```

Remember the bezier curves? It could be a headache to figure it out, so here's a nice tool:

http://www.roblaplaca.com/examples/bezierBuilder/



# PUTTING IT TOGETHER

Let's combine `transition` and `transform`:

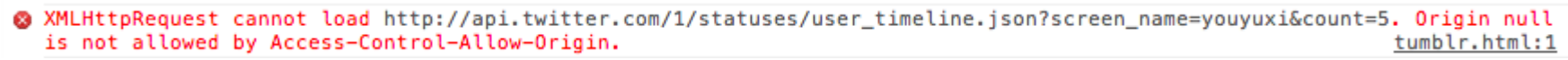
```
.object {
    transition: transform .3s cubic-bezier(.85, .09, .25, .90);
}
.object.rotated {
    transform: rotate(90deg);
}
.object.moved {
    transform: translate(50px, 50px);
}
.object.scaled {
    transform: scale(1.5);
}
```

# THINGS TO KNOW

- CSS3 transforms do not affect the page layout flow.
- They support sub-pixel positioning, an advantage over properties like `top, left`
- When using 3D transforms, it will be powered by the GPU when possible, resulting in smoother animation.

# MORE ABOUT APIs

# CROSS DOMAIN AJAX ISSUE




XMLHttpRequest has something called "same origin policy", which basically boils down to a restriction of ajax requests to URLs with a different port, domain or protocol. This restriction is in place to prevent cross-site scripting (XSS) attacks.

There are two ways to get around it:

- make the request on the server side
- use JSONP

# JSONP

JSONP = JSON with padding

Instead of returning this:

```
{ "data": 99 } // this is JSON
```

The API now returns this:

```
callback({ "data": 99 }); // this is actually javascript
```

# JSONP

Instead of directly making a request for a JSON file, we now create a script tag, and set its `src` attribute to the API's URL.

The result is the returned JavaScript will be executed!

```
callback({ "data": 99 });
```

Now if we have a function actually named `callback`:

```
function callback (response) {
    // you got the data!
    console.log(response.data) // 99
}
```

# JSONP with jQuery

```
var url = 'http://api.example.com/posts?callback=?'
$.getJSON(url, function (data) {
    // you got the data!
})
```

The `callback=?` lets both the API and jQuery know that we're trying to get data using JSONP. And jQuery handles the hassle for you. Nice.

# JSONP with jQuery

```
var url = 'http://api.example.com/posts?callback=?'
$.getJSON(url, function (data) {
    // you got the data!
})
```

The `callback=?` lets both the API and jQuery know that we're trying to get data using JSONP. And jQuery handles the hassle for you. Nice.

**NOTE:**

This is only a convention. Always follow the documentation of API you are working with.