Setting CSS3 properties using JavaScript

Setting CSS3 properties using JavaScript

Date created: March 6th, 2011

Setting CSS properties using JavaScript is nothing new, and for the most part follows a very predictable path:

```
document.getElementById("adiv").style.height="100px"

document.body.style.backgroundColor="yellow"
```

You'd first access the "`style`" object of an element, then follow it up with the desired CSS property name to set it. If the CSS property name is hyphenated, such as "`background-color`", drop the hyphen and capitalize the character following it when referring to it in JavaScript. Simple enough right? With setting CSS3 properties, technically nothing changes; the problem is knowing which property to set! You see, as browsers raced to support CSS3 properties before they're finalized, the use of vendor prefixes was what the browsers turned to offer an "interim" solution while the details of the official properties are hashed out . All major browsers have their own vendor prefix, which are:

Major CSS vendor prefixes/ extensions

Prefix	Description
-ms-	Microsoft CSS prefixes, such as `-ms-filter`, `-ms-behavior`, and `-ms-zoom`.
-moz-	Mozilla CSS prefixes, such as `-moz-box-shadow`, `-moz-border-radius`, and `-moz-transform`.
-webkit-	Webkit CSS prefixes (Safari, Chome etc), such as `-webkit-box-shadow`, `-webkit-border-radius`, and `-webkit-transform`.
-o-	Opera CSS prefixes, such as `-o-transition`, `-o-text-overflow`, `-o-transform`.
-khtml-	Konqueror CSS prefixes,such as `-khtml-user-select` and `-khtml-border-radius`

For example, to add a CSS3 box shadow (`box-shadow` property) to an element at the moment, you need to define 3 separate CSS properties to cover all your bases (IE9+, FF3.5+, Safari 4+/Chrome and Opera 8.5+):

```
box-shadow: 7px 7px 8px #818181;

-webkit-box-shadow: 7px 7px 8px #818181;

-moz-box-shadow: 7px 7px 8px #818181;
```

Setting CSS vendor properties in CSS is mostly just an exercise in patience, but when it comes to JavaScript,

you can add logistics to it as well. While the normal rules of setting CSS using JavaScript still apply for setting vendor specific CSS properties, how do we know which property to set? For example, in FF3.5, `-moz-box-shadow` is the supported property, while in IE9, it's the official `box-shadow` property that it recognizes instead. Sure we can just set all of the possible properties, but the question is, would you still respect yourself as a coder at the end of the day if you did that? A more efficient, elegant way is to probe the browser to see which CSS property it supports, then set that property only.

Setting CSS3 properties in JavaScript

So how exactly can we go about setting a CSS3 property when different browsers may support a different variation of the property? The key lies in the fact that if a browser supports a particular CSS property, it will return a string value (empty string if not set yet) when you request it from an element on the page. If it doesn't, the value ***undefined*** is returned instead. With that in mind, we can perform a one-time check before setting a CSS3 property which variant of the property it supports, and always turn to that one in our code. With that said, lets construct a function that accepts an array of CSS properties and returns the one the browser supports as a string:

```
function getsupportedprop(proparray){

    var root=document.documentElement

    for (var i=0; i<proparray.length; i++){


        if (typeof root.style[proparray[i]]=="string"){

var boxshadowprop=getsupportedprop(['boxShadow', 'MozBoxShadow', 'WebkitBoxShadow'])

document.getElementById("mydiv").style[boxshadowprop]="5px 5px 1px #818181"
```

The below makes use of the `getsupportedprop()` function to see which CSS `box-shadow` property your browser supports- "`boxShadow`", "`MozBoxShadow`", or "`WebKitBoxShadow`" when the button is clicked on:

```
function alertboxshadow(){

    alert(getsupportedprop(['boxShadow', 'MozBoxShadow',
'WebkitBoxShadow']))
```

If you're using FF3+, you should get "`MozBoxShadow`", in IE9+ and Opera9+, "`boxShadow`", and in Safari4+ and Chrome, "`WebKitBoxShadow`". In other browsers, the value ***undefined*** should be returned.

Armed with the above function, we can now go about setting CSS3 properties in JavaScript more easily, by passing into the function a list of possible variants of a particular CSS3 property, and letting the function figure out which one the browser supports and should be adopted. To drive this point home, lets set up a demo where 3 different CSS3 properties can be dynamically added or removed to a link button with relative ease, thanks to the aforementioned function:

```
var shadowprop=getsupportedprop(['boxShadow', 'MozBoxShadow', 'WebkitBoxShadow'])
```

```
var roundborderprop=getsupportedprop(['borderRadius', 'MozBorderRadius',
'WebkitBorderRadius'])

var csstransform=getsupportedprop(['transform', 'MozTransform', 'WebkitTransform',
'msTransform', 'OTransform'])

function changecssproperty(target, prop, value, action){

    if (typeof prop!="undefined")


        target.style[prop]=(action=="remove")? "" : value

<a id="cfbutton" href="http://www.codingforums.com">Coding Forums</a>

var z=document.getElementById("cfbutton")

<a href="javascript:changecssproperty(z, shadowprop, '6px 6px 8px
rgba(0,0,0,.5)')">Add Shadow</a>

<a href="javascript:changecssproperty(z, shadowprop, '', 'remove')">Remove Shadow</a>

<a href="javascript:changecssproperty(z, roundborderprop, '15px')">Add Round
Border</a>

<a href="javascript:changecssproperty(z, roundborderprop, '', 'remove')">Remove Round
Border</a>

<a href="javascript:changecssproperty(z, csstransform, 'rotate(25deg)')">Add
Transform Rotate</a>

<a href="javascript:changecssproperty(z, csstransform, '', 'remove')">Remove
Transform</a>
```

Demo:

Coding Forums

- Add Shadow | Remove Shadow
- Add Round Border | Remove Round Border
- Add Transform Rotate | Remove Transform

As you can see, we first use the `getsupportedprop()` function to get and cache each variant of CSS3's `box-shadow`, `border-radius`, and `transform` properties the browser supports. Then, using these properties, we go about manipulating them dynamically in JavaScript. Manipulating CSS3 values in JavaScript is now a streamlined process!