Chapter 64: requestAnimationFrame

Parameter	Details
callback	"A parameter specifying a function to call when it's time to update your animation for the next repaint." (https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame)

Section 64.1: Use requestAnimationFrame to fade in element

- **View jsFiddle**: https://jsfiddle.net/HimmatChahal/jb5trg67/
- **Copy + Pasteable code below**:

```
<html>
    <body>
        <h1>This will fade in at 60 frames per second (or as close to possible as your hardware
allows)</h1>

        <script>
            // Fade in over 2000 ms = 2 seconds.
            var FADE_DURATION = 2.0 * 1000;

            // -1 is simply a flag to indicate if we are rendering the very 1st frame
            var startTime=-1.0;

            // Function to render current frame (whatever frame that may be)
            function render(currTime) {
                var head1 = document.getElementsByTagName('h1')[0];

                // How opaque should head1 be?  Its fade started at currTime=0.
                // Over FADE_DURATION ms, opacity goes from 0 to 1
                var opacity = (currTime/FADE_DURATION);
                head1.style.opacity = opacity;
            }

            // Function to
            function eachFrame() {
                // Time that animation has been running (in ms)
                // Uncomment the console.log function to view how quickly
                // the timeRunning updates its value (may affect performance)
                var timeRunning = (new Date()).getTime() - startTime;
                //console.log('var timeRunning = '+timeRunning+'ms');
                if (startTime < 0) {
                   // This branch: executes for the first frame only.
                   // it sets the startTime, then renders at currTime = 0.0
                   startTime = (new Date()).getTime();
                   render(0.0);
               } else if (timeRunning < FADE_DURATION) {
                   // This branch: renders every frame, other than the 1st frame,
                   // with the new timeRunning value.
                   render(timeRunning);
               } else {
                   return;
               }

               // Now we are done rendering one frame.
               // So we make a request to the browser to execute the next
               // animation frame, and the browser optimizes the rest.
               // This happens very rapidly, as you can see in the console.log();
               window.requestAnimationFrame(eachFrame);
           };
```

```
            // start the animation
            window.requestAnimationFrame(eachFrame);
        </script>
    </body>
</html>
```

Section 64.2: Keeping Compatibility

Of course, just like most things in browser JavaScript, you just can't count on the fact that everything will be the same everywhere. In this case, `requestAnimationFrame` might have a prefix on some platforms and are named differently, such as `webkitRequestAnimationFrame`. Fortunately, there's a really easy way to group all the known differences that could exist down to 1 function:

```
window.requestAnimationFrame = (function(){
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        function(callback){
            window.setTimeout(callback, 1000 / 60);
        };
})();
```

Note that the last option (which fills in when no existing support was found) will not return an id to be used in `cancelAnimationFrame`. There is, however an efficient polyfill that was written which fixes this.

Section 64.3: Cancelling an Animation

To cancel a call to `requestAnimationFrame`, you need the id it returned from when it was last called. This is the parameter you use for `cancelAnimationFrame`. The following example starts some hypothetical animation then pauses it after one second.

```
// stores the id returned from each call to requestAnimationFrame
var requestId;

// draw something
function draw(timestamp) {
    // do some animation
    // request next frame
    start();
}

// pauses the animation
function pause() {
    // pass in the id returned from the last call to requestAnimationFrame
    cancelAnimationFrame(requestId);
}

// begin the animation
function start() {
    // store the id returned from requestAnimationFrame
    requestId = requestAnimationFrame(draw);
}

// begin now
start();

// after a second, pause the animation
```

```
setTimeout(pause,1000);
```