CPSC1520 – JavaScript 1 Exercise: Working with Timers

Timers

There are often times when you may need something to happen on the page without a trigger from the user. Examples include: timed slideshows, rotating site or product reviews, simple animations (e.g. fades and slides), etc.). In order to trigger functions automatically, we can take advantage of several functions in the browser's global scope: **setTimeout** and **setInterval**.

To demonstrate how these functions can be used to automate actions on the page, we will revisit the image carousel from the previous exercise:

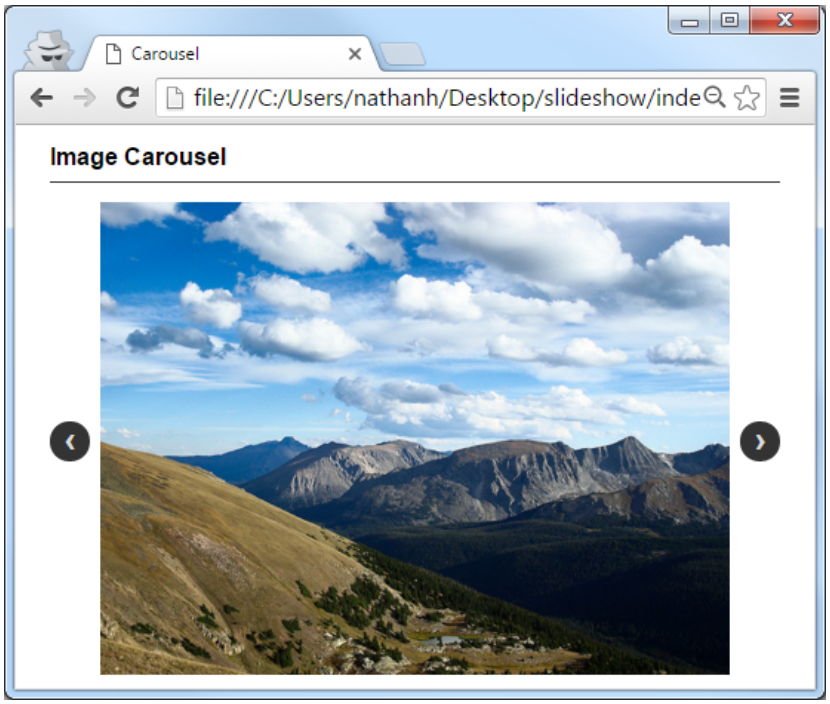

Figure 1. Image carousel from previous exercise

We will implement a timer for the carousel that will progress onto the next image in the slideshow every 3 seconds.

Operation of setTimeout and setInterval

Each of the two functions mentioned here have identical APIs and only differ in how they operate, with setTimeout only calling a function a single time and setInterval continuing to call the function after the specified amount of time. Both functions are called with the following three parameters:

1. Function to call
2. Time to wait before/between calling the function (specified in milliseconds)
3. A comma separated list of arguments to the called function

Typically you will see only the first two parameters used in these calls to due to the fact that IE (all versions) do not implement the list of arguments.

Once called, each function returns an id, which can then be used to identify the specific context that was created. This is useful as each function has a paired clear function (e.g. clearTimeout) that can be used to cancel the context that was created *before* the next function call. Examples of creating a timeout context and then cancelling follow.

```
var to = setTimeout(someFunction,3000);
```

Example 1. Using setTimeout to set a onetime call to a function for 3 seconds in the future

```
clearTimeout(to);
```

Example 2. Using the paired function clearTimeout to cancel the previously created timeout context

Please read more about these timer functions (and their paired clear functions) here1.

Automating the Slideshow

In order for the slideshow to automatically move onto the next image a timer must be initialized. Unlike the example 1 above, we will need to set an interval so that the site continues to progress to a new slide every 3 seconds, and not just once after 3 seconds.

Firsts, a function must be made to do the actual updating of the displayed image. We can create this function so that it can be used in all places where we currently update the current slide:

```
function updateSlide(index) {
    // display the new current image
    document.querySelector('.carousel>img').src = 'images/' + images[index];
    // update the active selector bullet
    document.querySelector('.image-tracker .active').classList.remove('active');
    document.querySelectorAll('[data-idx]')[index].classList.add('active');
}
```

Example 3. A function to update the current slide image

Now this function can be used in all places where we are currently updating the displayed image:

```
// replace lines 35-40 with the following call
updateSlide(currentImg);
```

Example 4. Update the existing code to use the updateSlide function

Now all that's left is to setup the interval to call the updateSlide function every 3 seconds. Because we cannot rely on the setInterval's third parameter (i.e. the argument list), we will have to use an anonymous function in the call to setInterval that can update the current image and make the call:

```
// add a variable to track the context of the interval
var slideshowInterval;

...

// now call the setInterval function to begin the slideshow
slideshowInterval = setInterval(function () {
    currentImg += 1;
    if (currentImg == images.length) {
        currentImg = 0;
    }
    updateSlide(currentImg)
}, 3000);
```

Example 5. Making the final call to setInterval

That's it! Reload the page and your slideshow should move along every 3 seconds.

References

1. https://developer.mozilla.org/en/docs/Web/API/NodeList#Why_is_NodeList_not_an_Array