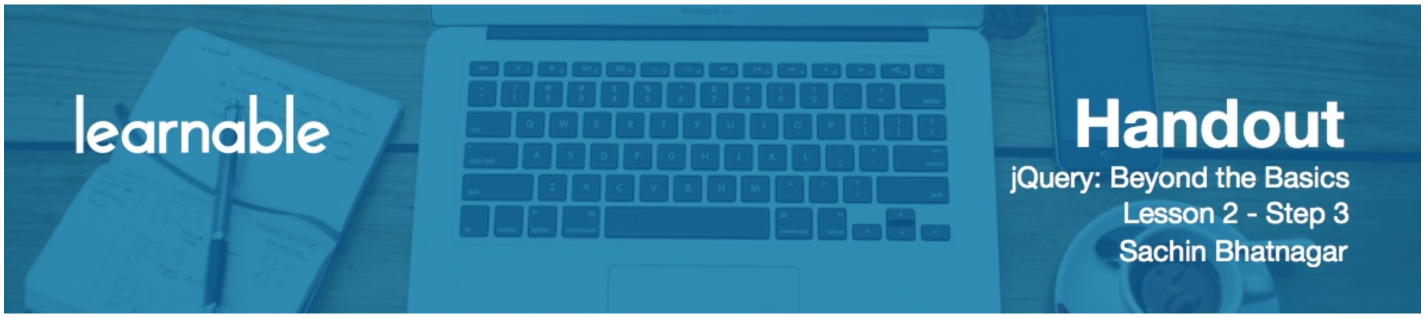

Planning a Slideshow with fade in and out

At the previous step, we explored jQuery's fade methods that let you play with an element's visibility by animating its opacity. Now we are going create a simple photo slideshow that will loop through a given set of images by fading them in and out. The actual HTML layout of the slideshow structure mainly contains an unordered list, where each list item contains an image tag with a given image.

To build the slideshow, we require two kinds of functions. The `preloader` method first loads and caches the images before the slideshow starts playing. If we do not use a `preloader` then images may or may not load up on time to show up in the slideshow, leading to empty slides. The `preloader` method is already included so you don't have to build one yourself. The `slideshow` method itself runs the actual slideshow and uses the `fadeIn` and `fadeOut` methods.

Creating a preloader for slideshow

Open the *Slideshow_Begin.htm* file from the accompanying zip archive to begin.

First up, we have a couple of variables. The `slides` variable accesses the list items within the slideshow unordered list since that is what we were playing with all throughout this example. Instead of calling the selector again and again we have cached it in a variable.

Next, we're using a variable called `slideCount`, which is set to zero. This would be used to iterate through the list items. The `totalSlides` variable is set to obtain the number of list items in the unordered list by accessing the `length` property of the `slides` variable. And finally, we have an empty array in the variable `slideCache` which will be used to cache all the images in by the `preloader` function.

The `preloader` is an **immediately invoked function expression (IIFE)**. IFFEs run automatically when the page loads up and the jQuery `ready` method fires. What we are doing here in the `preloader` function is checking to see if our `slideCount` variable is less than the number of slides or list items. If yes, we are instantiating a new JavaScript `Image` object and assigning it to the `slideCache` at a specific index. Once instantiated, we then set the source attribute of this `Image` object to the image file as fetched from the image tag within our list item. The first time that the sequence runs, the `slideCount` variable is set to zero hence by using the `eq` method on the slide selector with the `slideCount` variable we are fetching the first list item. We are getting hold of its image tags source attribute value.

Next the `onload` function executes when the image is loaded and cached in the previous step. This is where we simply increment the `slideCount` variable by 1. Then we reload the `preloader` function again. Since the slide count value has incremented, the `if` statement again checks to see if the value is less than the total number of slides, and the process repeats. Once all images are cached this way, we are resetting the `slideCount` variable to zero, since it will be reused in the `SlideShow` function, and we're calling the `SlideShow` function immediately.

Creating the Slideshow

One small thing to note here is that typically JavaScript functions should begin with a lower-case character but here the `SlideShow` function has been named so only to make it obvious and readable.

```
function SlideShow() {
    slides.eq(slideCount).fadeIn(1000).delay(2000).fadeOut(1000, function() {
        slideCount < totalSlides - 1 ? slideCount++ : slideCount=0;
        SlideShow();
    });
}
```

We have essentially created a looping function which keeps calling itself again and again. The ternary operator increments the `slideCount` variable such that we access successive list items to fade in and out from the display. When all slides have been shown, the `slideCount` variable is reset to zero, thus restarting the slideshow from the first image.

Go ahead and try this out now.