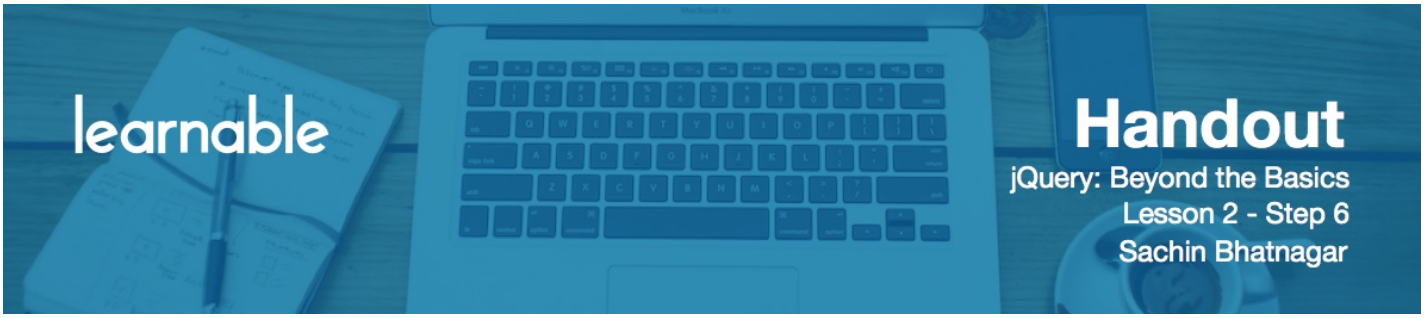

Introduction to staggering animation

We can easily use multiple `animate` methods on multiple elements since they don't interfere with each other and work independently. Imagine there are five elements and we have animated their widths individually using five instances of the `animate` method. All these elements will be animated simultaneously. But what if you wanted to begin animation after the previous one has ended?

This is called **staggering animation**, and while natively jQuery doesn't provide us with a way, we can still easily get this to work.

Working with staggering animation demo

Open the *stagger_Begin.html* file before proceeding.

There is a button which is quite similar to the animated menu that we built earlier. We also have four colored panes that are just empty list items in an unordered list. What we want to do is animate the `height` property of these panes such that each pane begins animation at a little delay from the previous one.

We have two functions for opening and closing the panes, and we're using a simple variable called `menuStatus` to set the state of the menu so that our button essentially toggles them open or closed. So let's see how we can first open the panes out:

```
function openMenu() {
  $('.navMenu > li').each(function(index, element) {
    $(element).animate({
      height:'180px'
    }, 500);
  });
}
```

When you now click on the button, all panes animate simultaneously, because each `animate` method works independent of the rest. What we want to do is delay each `animate` method successively so that they start off at a delay from its previous sibling. This is as simple as inserting the `delay` method between the selector and the `animate` method:

```
function openMenu() {
  $('.navMenu > li').each(function(index, element) {
    $(element).delay(200 * index).animate({
      height:'180px'
    }, 500);
  });
}
```

So quite simply, the first time the `each` method loops through list items, its index value is set to zero, and therefore it effectively eliminates the effect of the `delay` method, since the delay value computes to zero milliseconds. On the next iteration the delay is set to 200 milliseconds, since the index value being multiplied would be 1. On the next iteration the delay value becomes 400 milliseconds since 200 is multiplied by the index value of two and so on.

Next we want to close the menu in the reverse order, where the last pane, the blue one, closes first, followed by the red and so on:

```
function closeMenu() {
  $($('.navMenu > li').get().reverse()).each(function(index, element) {
    $(element).delay(200 * index).animate({
```

```
      height:'5px'
    }, 500);
  });
}
```

The `get` method gives us access to the internal DOM structure of the unordered list. It is followed by the `reverse` method, which will reverse the order of the list items. This entire statement is enclosed within a jQuery select method. This way we get list items that will count back from the last list item to the first.

Lastly we have to set the `menuStatus` variable correctly:

```
function openMenu() {
  $('.navMenu > li').each(function(index, element) {
    $(element).delay(200 * index).animate({
      height:'180px'
    }, 500);
  });
  menuStatus = true;
}
function closeMenu() {
  $($('.navMenu > li').get().reverse()).each(function(index, element) {
    $(element).delay(200 * index).animate({
      height:'5px'
    }, 500);
  });
  menuStatus = false;
}
```

Try this out!

Introduction to easing methods

Easing refers to a function that describes the speed of an animation. When used properly, animations can be made more lively and pleasing. jQuery provides us with two ways of controlling the speed of animation:

- **Linear easing** - an element will be animated at constant speed from start to finish.
- **Swing easing** - animation accelerates from the start and then decelerates towards the end to reach the end state.

These are not the only easing functions that jQuery supports. By using an additional easing plug-in, many more easing functions can be brought in for use.

Working with easing methods

Open the *stagger_easing_Begin.htm* file, which is basically the stagger animation example, that we built a while back.

We'll use Robert Penner's plugin that provides easing equations for the transitional animations. Visit gsgd.co.uk/sandbox/jquery/easing to read more about this plugin.

Include this library from a CDN:

```
<head>
    [...]
    <script src="//cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js"></script>
    [...]
</head>
```

Now incorporate the easing function:

```
function openMenu() {
  $('.navMenu > li').each(function(index, element) {
    $(element).delay(200 * index).animate({
      height:'180px'
    }, 500, 'easeOutBounce');
```

```
  });
  menuStatus = true;
}
function closeMenu() {
  $($('.navMenu > li').get().reverse()).each(function(index, element) {
    $(element).delay(200 * index).animate({
      height:'5px'
    }, 500, 'easeInBounce');
  });
  menuStatus = false;
}
```

Reload the page and note how the panes bounce a bit and the whole animation looks slappy and cool.