

6

Effect Methods

The jQuery library provides several techniques for adding animation to a web page. These include simple, standard animations that are frequently used and the ability to craft sophisticated custom **effects**. In this chapter, we'll closely examine each of the effect methods, revealing all of the mechanisms jQuery has for providing visual feedback to the user.



Some of the examples in this chapter use the `$.print()` function to print results to the page. This is a simple plug-in, which will be discussed in Chapter 10, *Plug-in API*.

Pre-packaged effects

These methods allow us to quickly apply commonly-used effects with a minimum of configuration.

.show()

Display the matched elements.

```
.show([duration][, callback])
```

Parameters

- `duration` (optional): A string or number determining how long the animation will run
- `callback` (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

With no parameters, the `.show()` method is the simplest way to display an element.

```
$('.target').show();
```

The matched elements will be revealed immediately with no animation. This is roughly equivalent to calling `.css('display', 'block')`, except that the `display` property is restored to whatever it was initially. If an element has a `display` value of `inline`, then is hidden and shown, it will once again be displayed `inline`.

When a duration is provided, `.show()` becomes an animation method.

The `.show()` method animates the width, height, and opacity of the matched elements simultaneously.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

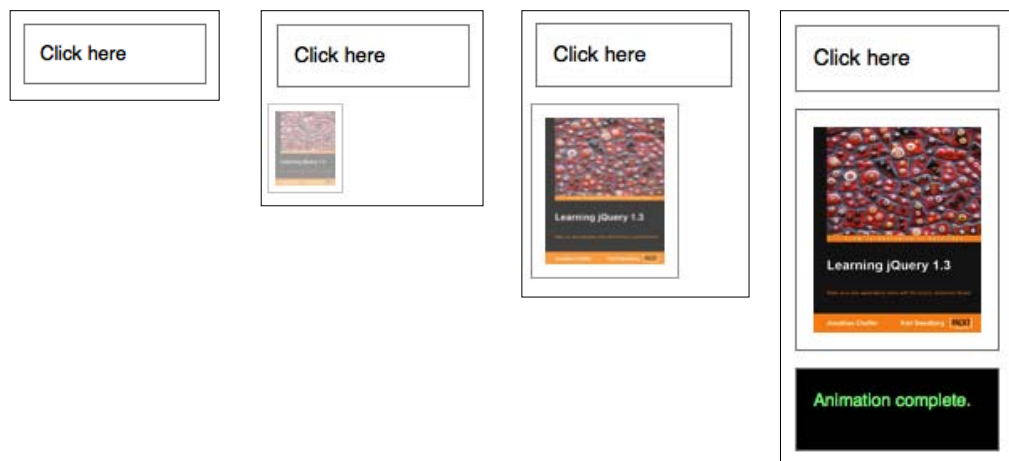
```
<div id="clickme">
  Click here
</div>

```

With the element initially hidden, we can show it slowly.

```
$('#clickme').click(function() {
  $('#book').show('slow', function() {
    $.print('Animation complete.');
```

```
  });
});
```



.hide()

Hide the matched elements.

```
.hide([duration][, callback])
```

Parameters

- **duration** (optional): A string or number determining how long the animation will run
- **callback** (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

With no parameters, the `.hide()` method is the simplest way to hide an element.

```
$('.target').hide();
```

The matched elements will be hidden immediately, with no animation. This is roughly equivalent to calling `.css('display', 'none')`, except that the value of the `display` property is saved in jQuery's data cache so that `display` can later be restored to its initial value. If an element has a `display` value of `inline`, and then is hidden and shown, it will once again be displayed `inline`.

When a duration is provided, `.hide()` becomes an animation method. The `.hide()` method animates the width, height, and opacity of the matched elements simultaneously. When these properties reach 0, the `display` style property is set to `none` to ensure that the element no longer affects the layout of the page.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

```
<div id="clickme">
  Click here
</div>

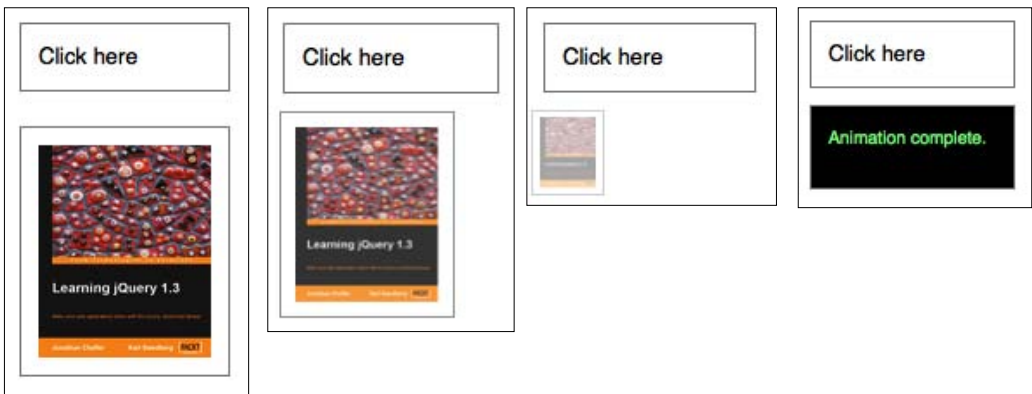
```

With the element initially shown, we can hide it slowly.

```
$('#clickme').click(function() {
  $('#book').hide('slow', function() {
    $.print('Animation complete.');
```

```
  });
```

```
});
```



.toggle()

Display or hide the matched elements.

```
.toggle([duration] [, callback])  
.toggle(showOrHide)
```

Parameters (first version)

- `duration` (optional): A string or number determining how long the animation will run
- `callback` (optional): A function to call once the animation is complete

Parameters (second version)

- `showOrHide`: A Boolean indicating whether to show or hide the elements

Return value

The jQuery object, for chaining purposes.

Description

With no parameters, the `.toggle()` method simply toggles the visibility of elements:

```
$('.target').toggle();
```

The matched elements will be revealed or hidden immediately with no animation.

If the element is initially displayed, it will be hidden; if hidden, it will be shown.

The `display` property is saved and restored as needed. If an element has a `display` value of `inline`, then is hidden and shown, it will once again be displayed `inline`.

When a duration is provided, `.toggle()` becomes an animation method. The `.toggle()` method animates the width, height, and opacity of the matched elements simultaneously. When these properties reach 0 after a hiding animation, the `display` style property is set to `none` to ensure that the element no longer affects the layout of the page.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

```
<div id="clickme">
  Click here
</div>

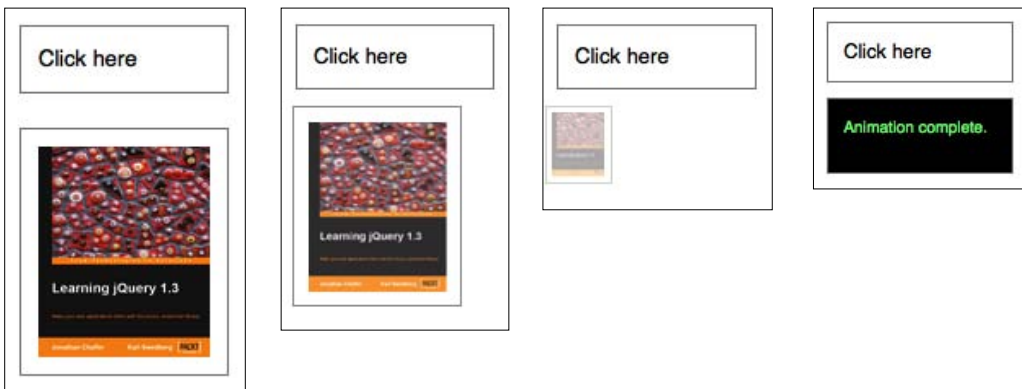
```

We will cause `.toggle()` to be called when another element is clicked.

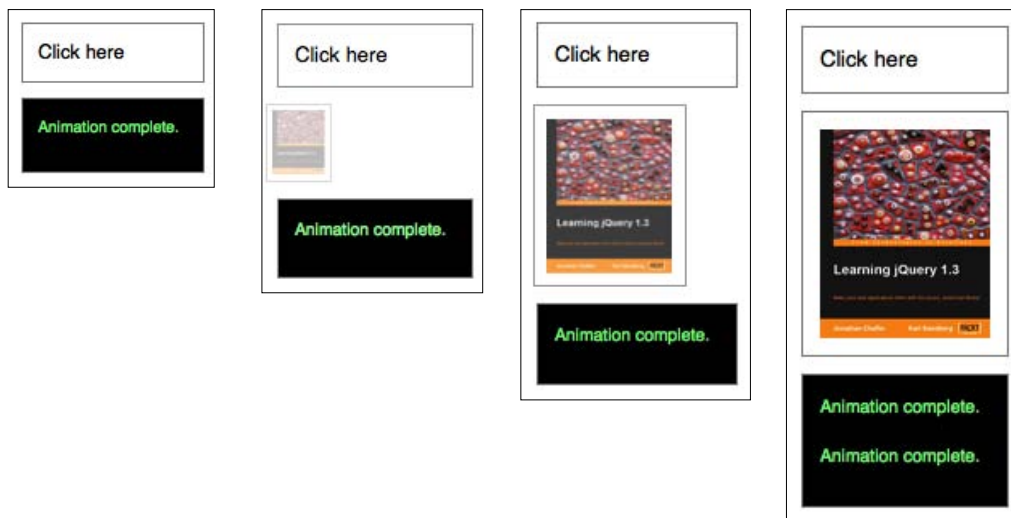
```
$('#clickme').click(function() {
  $('#book').toggle('slow', function() {
    $.print('Animation complete.');
```

```
  });
});
```

With the element initially shown, we can hide it slowly with the first click:



A second click will show the element once again:



The second version of the method accepts a Boolean parameter. If this parameter is `true`, then the matched elements are shown; if `false`, the elements are hidden. In essence, the following statement

```
$('#foo').toggle(showOrHide);
```

is equivalent to:

```
if (showOrHide) {
    $('#foo').show();
}
else {
    $('#foo').hide();
}
```



There is also an event method named `.toggle()`. For details on this method, see Chapter 5, *Event Methods*.

.slideDown()

Display the matched elements with a sliding motion.

```
.slideDown([duration][, callback])
```

Parameters

- **duration** (optional): A string or number determining how long the animation will run
- **callback** (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.slideDown()` method animates the height of the matched elements. This causes lower parts of the page to slide down, making way for the revealed items.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. If any other string is supplied, or if the duration parameter is omitted, the default duration of 400 milliseconds is used.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

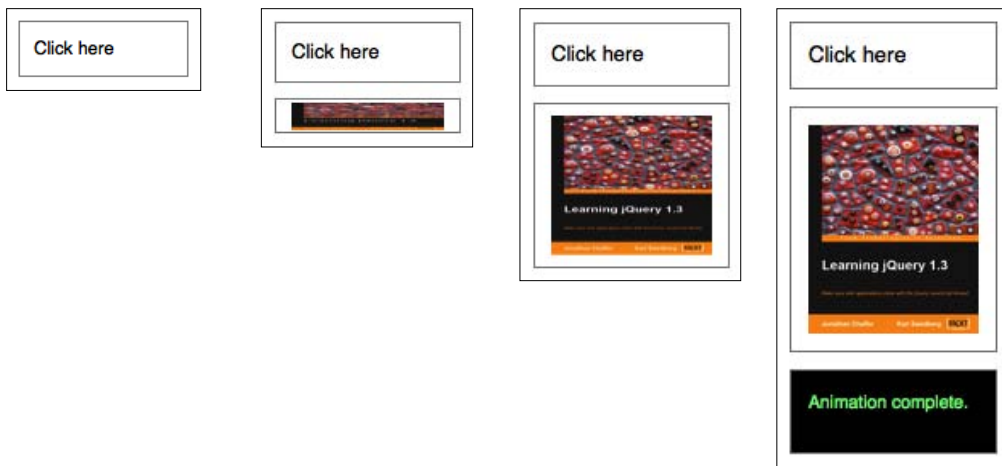
```
<div id="clickme">
  Click here
</div>

```

With the element initially hidden, we can show it slowly.

```
$('#clickme').click(function() {
  $('#book').slideDown('slow', function() {
    $.print('Animation complete.');
```

```
  });
});
```

.slideUp()

Hide the matched elements with a sliding motion.

```
.slideUp([duration] [, callback])
```

Parameters

- `duration` (optional): A string or number determining how long the animation will run
- `callback` (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.slideUp()` method animates the height of the matched elements. This causes lower parts of the page to slide up, appearing to conceal the items. Once the height reaches 0, the `display` style property is set to `none` to ensure that the element no longer affects the layout of the page.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. If any other string is supplied, or if the `duration` parameter is omitted, the default duration of 400 milliseconds is used.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

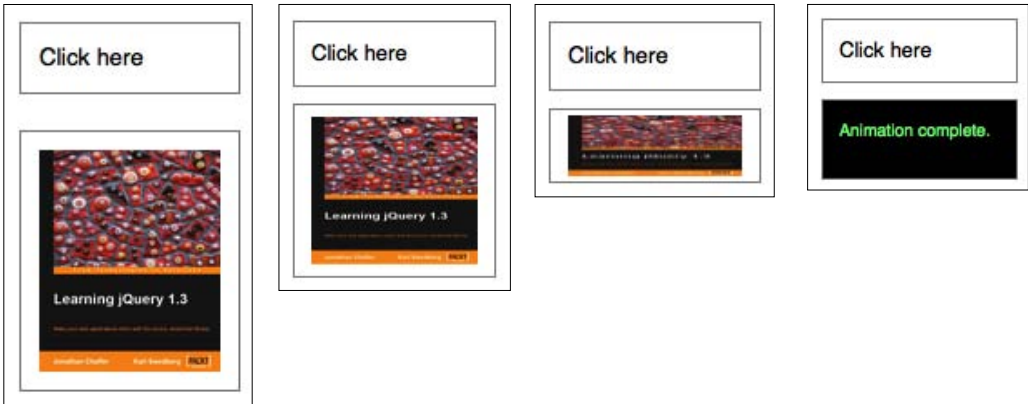
```
<div id="clickme">
  Click here
</div>

```

With the element initially shown, we can hide it slowly.

```
$('#clickme').click(function() {
  $('#book').slideUp('slow', function() {
    $.print('Animation complete.');
```

```
  });
});
```



.slideToggle()

Display or hide the matched elements with a sliding motion.

```
.slideToggle([duration] [, callback])
```

Parameters

- `duration` (optional): A string or number determining how long the animation will run
- `callback` (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.slideToggle()` method animates the height of the matched elements. This causes lower parts of the page to slide up or down, appearing to reveal or conceal the items. If the element is initially displayed, it will be hidden; if hidden, it will be shown. The `display` property is saved and restored as needed. If an element has a `display` value of `inline`, then is hidden and shown, it will once again be displayed `inline`. When the height reaches 0 after a hiding animation, the `display` style property is set to `none` to ensure that the element no longer affects the layout of the page.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

```
<div id="clickme">
  Click here
</div>

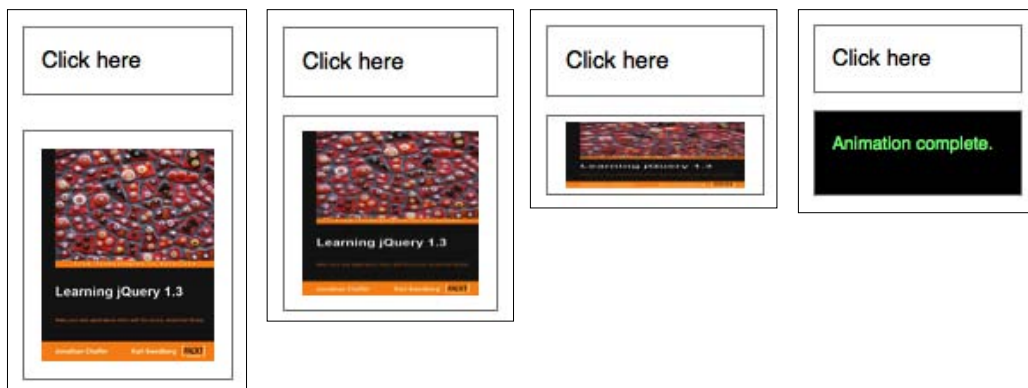
```

We will cause `.slideToggle()` to be called when another element is clicked.

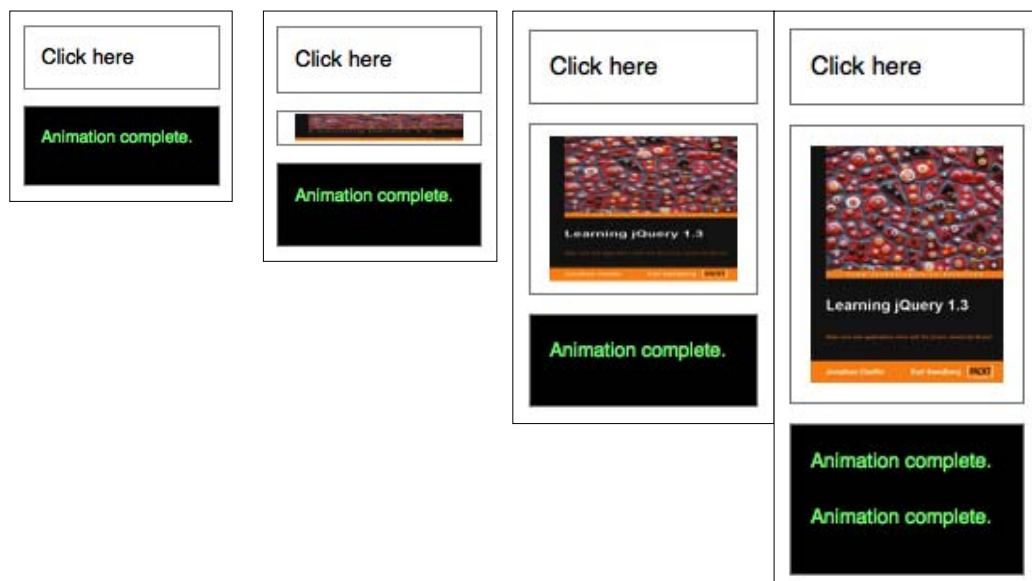
```
$('#clickme').click(function() {
  $('#book').slideToggle('slow', function() {
    $.print('Animation complete.');
```

```
  });
});
```

With the element initially shown, we can hide it slowly with the first click:



A second click will show the element once again:



.fadeIn()

Display the matched elements by fading them to opaque.

```
.fadeIn([duration][, callback])
```

Parameters

- **duration** (optional): A string or number determining how long the animation will run
- **callback** (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.fadeIn()` method animates the opacity of the matched elements.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. If any other string is supplied, or if the `duration` parameter is omitted, the default duration of 400 milliseconds is used.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

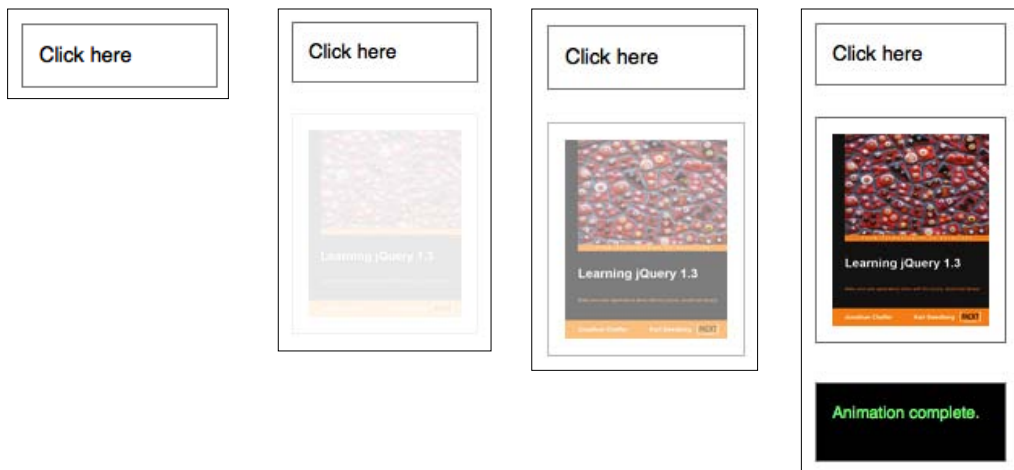
We can animate any element, such as a simple image:

```
<div id="clickme">
  Click here
</div>

```

With the element initially hidden, we can show it slowly.

```
$('#clickme').click(function() {  
    $('#book').fadeIn('slow', function() {  
        $.print('Animation complete.');    });  
});
```



.fadeOut()

Hide the matched elements by fading them to transparent.

```
.fadeOut([duration][, callback])
```

Parameters

- **duration** (optional): A string or number determining how long the animation will run
- **callback** (optional): A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.fadeOut()` method animates the opacity of the matched elements. Once the opacity reaches 0, the `display` style property is set to `none`, so the element no longer affects the layout of the page.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. If any other string is supplied, or if the `duration` parameter is omitted, the default duration of 400 milliseconds is used.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

```
<div id="clickme">
  Click here
</div>

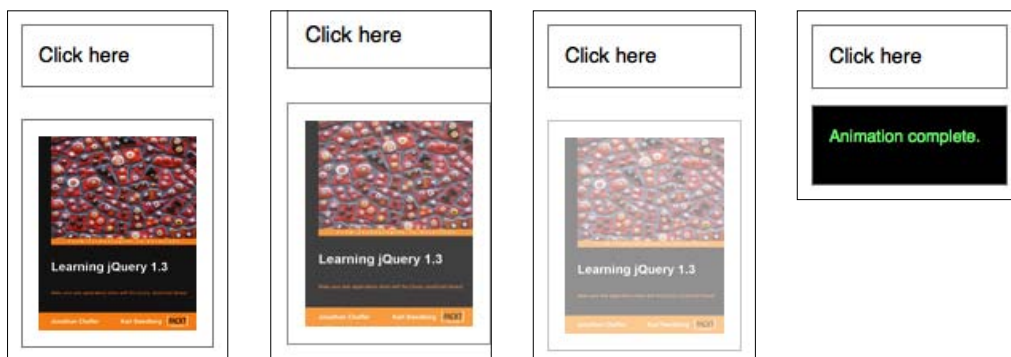
```

With the element initially shown, we can hide it slowly.

```
$('#clickme').click(function() {
  $('#book').fadeOut('slow', function() {
    $.print('Animation complete.');
```

```
});
```

```
});
```



.fadeTo()

Adjust the opacity of the matched elements.

```
.fadeTo(duration, opacity[, callback])
```

Parameters

- **duration**: A string or number determining how long the animation will run
- **opacity**: A number between 0 and 1 denoting the target opacity
- **callback (optional)**: A function to call once the animation is complete

Return value

The jQuery object, for chaining purposes.

Description

The `.fadeTo()` method animates the opacity of the matched elements.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. If any other string is supplied, the default duration of 400 milliseconds is used. Unlike the other effect methods, `.fadeTo()` requires that `duration` be explicitly specified.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

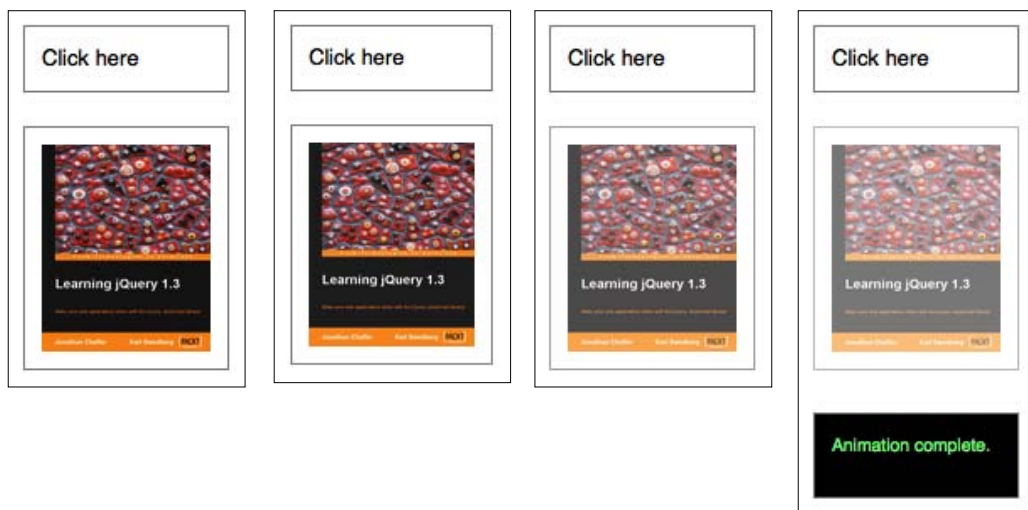
```
<div id="clickme">
  Click here
</div>

```

With the element initially shown, we can dim it slowly.

```
$('#clickme').click(function() {
  $('#book').fadeTo('slow', 0.5, function() {
    $.print('Animation complete.');
```

```
  });
});
```

With duration set to 0, this method just changes the opacity CSS property, so `.fadeTo(0, opacity)` is the same as `.css('opacity', opacity)`.

Customized effects

This section describes how to create effects that are not provided out of the box by jQuery.

`.animate()`

Perform a custom animation of a set of CSS properties.

```
.animate(properties[, duration][, easing][, callback])
.animate(properties, options)
```

Parameters (first version)

- **properties:** A map of CSS properties that the animation will move toward
- **duration (optional):** A string or number determining how long the animation will run
- **easing (optional):** A string indicating which easing function to use for the transition
- **callback (optional):** A function to call once the animation is complete

Parameters (second version)

- **properties**: A map of CSS properties that the animation will move toward
- **options**: A map of additional options to pass to the method. Supported keys are:
 - **duration**: A string or number determining how long the animation will run
 - **easing**: A string indicating which easing function to use for the transition
 - **complete**: A function to call once the animation is complete
 - **step**: A function to be called after each step of the animation
 - **queue**: A Boolean indicating whether to place the animation in the effects queue. If `false`, the animation will begin immediately
 - **specialEasing**: A map of one or more of the CSS properties defined by the `properties` argument and their corresponding easing functions

Return value

The jQuery object, for chaining purposes.

Description

The `.animate()` method allows us to create animation effects on any numeric CSS property. The only required parameter is a map of CSS properties. This map is similar to the one that can be sent to the `.css()` method, except that the range of properties is more restrictive.

All animated properties are treated as a number of pixels, unless otherwise specified. The units `em` and `%` can be specified where applicable.

In addition to numeric values, each property can take the strings `'show'`, `'hide'`, and `'toggle'`. These shortcuts allow for custom hiding and showing animations that take into account the display type of the element.

Animated properties can also be **relative**. If a value is supplied with a leading `+=` or `-=` sequence of characters, then the target value is computed by adding or subtracting the given number to or from the current value of the property.

Durations are given in milliseconds; higher values indicate slower animations, not faster ones. The `'fast'` and `'slow'` strings can be supplied to indicate durations of 200 and 600 milliseconds, respectively. Unlike the other effect methods, `.fadeTo()` requires that `duration` be explicitly specified.

If supplied, the callback is fired once the animation is complete. This can be useful for stringing different animations together in sequence. The callback is not sent any arguments, but `this` is set to the DOM element being animated. If multiple elements are animated, it is important to note that the callback is executed once per matched element, not once for the animation as a whole.

We can animate any element, such as a simple image:

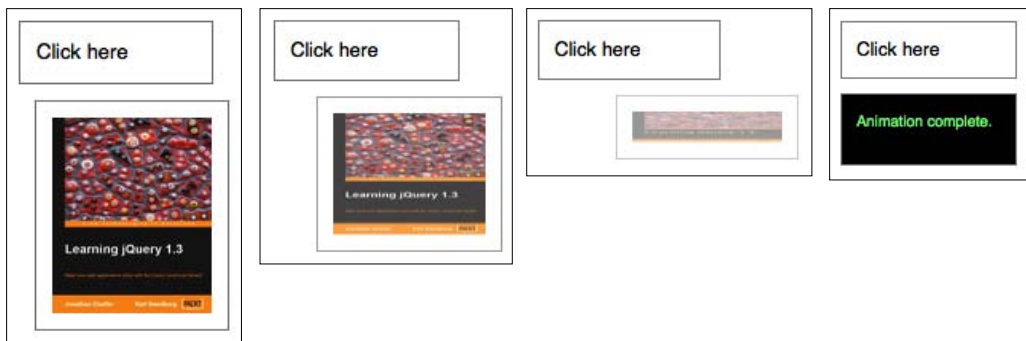
```
<div id="clickme">
  Click here
</div>

```

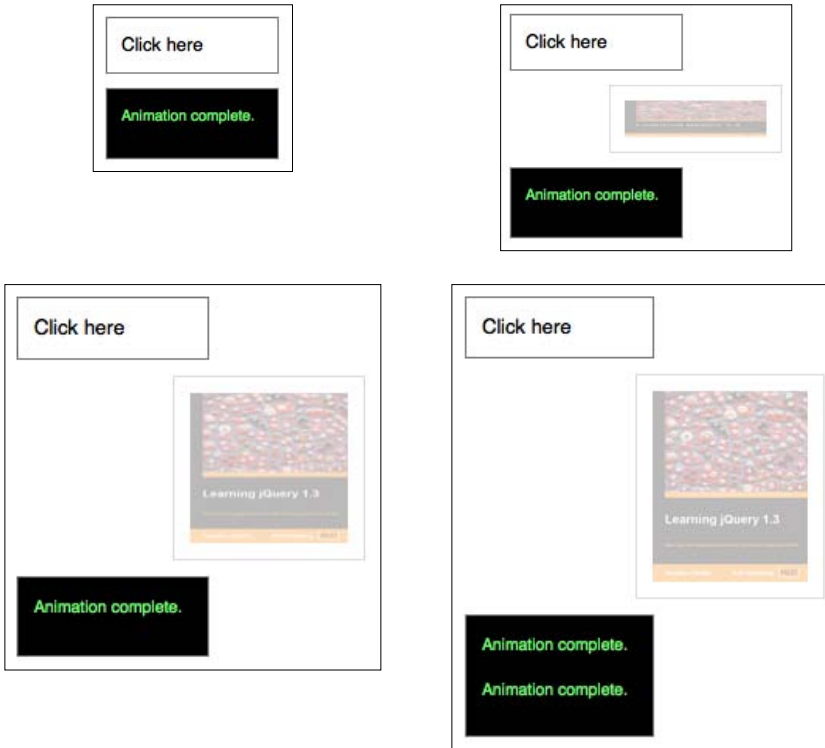
We can animate the opacity, left offset, and height of the image simultaneously.

```
$('#clickme').click(function() {
  $('#book').animate({
    opacity: 0.25,
    left: '+=50',
    height: 'toggle'
  }, 5000, function() {
    $.print('Animation complete.');
```

```
});
});
```




Note that we have specified `toggle` as the target value of the `height` property. As the image was visible before, the animation shrinks the height to 0 to hide it. A second click then reverses this transition:



The `opacity` of the image is already at its target value, so this property is not animated by the second click. As we specified the target value for `left` as a relative value, the image moves even farther to the right during this second animation.

The `position` attribute of the element must not be `static` if we wish to animate the `left` property as we do in the example.

[ The jQuery UI project extends the `.animate()` method by allowing some non-numeric styles, such as colors, to be animated. The project also includes mechanisms for specifying animations through CSS classes rather than individual attributes.]

The remaining parameter of `.animate()` is a string naming an **easing function** to use. An easing function specifies the speed at which the animation progresses at different points within the animation. The only easing implementations in the jQuery library are the default, called `swing`, and one that progresses at a constant pace, called `linear`. More easing functions are available with the use of plug-ins, most notably the jQuery UI suite.

As of jQuery version 1.4, we can set per-property easing functions within a single `.animate()` call. In the first version of `.animate()`, each property can take an array as its value: The first member of the array is the CSS property and the second member is an easing function. If a per-property easing function is not defined for a particular property, it uses the value of the `.animate()` method's optional easing argument. If the easing argument is not defined, the default `swing` function is used.

We can simultaneously animate the width and height with the `swing` easing function and the opacity with the `linear` easing function:

```
$('#clickme').click(function() {
    $('#book').animate({
        width: ['toggle', 'swing'],
        height: ['toggle', 'swing'],
        opacity: 'toggle'
    }, 5000, 'linear', function() {
        $.print('Animation complete.');
```

```
    });
});
```

In the second version of `.animate()`, the options map can include the `specialEasing` property, which is itself a map of CSS properties and their corresponding easing functions. We can simultaneously animate the width using the `linear` easing function and the height using the `easeOutBounce` easing function.

```
$('#clickme').click(function() {
    $('#book').animate({
        width: 'toggle',
        height: 'toggle'
    }, {
        duration: 5000,
        specialEasing: {
            width: 'linear',
            height: 'easeOutBounce'
        },
        complete: function() {
            $.print('Animation complete.');
```

```
    }
});
});
```

As previously noted, a plug-in is required for the `easeOutBounce` function.

.stop()

Stop the currently running animation on the matched elements.

```
.stop([clearQueue] [, jumpToEnd])
```

Parameters

- `clearQueue` (optional): A Boolean indicating whether to remove queued animation as well. Defaults to `false`
- `jumpToEnd` (optional): A Boolean indicating whether to complete the current animation immediately. Defaults to `false`

Return value

The jQuery object, for chaining purposes.

Description

When `.stop()` is called on an element, the currently running animation (if any) is immediately stopped. For instance, if an element is being hidden with `.slideUp()` when `.stop()` is called, the element will now still be displayed, but will be a fraction of its previous height. Callback functions are not called.

If more than one animation method is called on the same element, the later animations are placed in the **effects queue** for the element. These animations will not begin until the first one completes. When `.stop()` is called, the next animation in the queue begins immediately. If the `clearQueue` parameter is provided with a value of `true`, then the rest of the animations in the queue are removed and never run.

If the `jumpToEnd` property is provided with a value of `true`, the current animation stops, but the element is immediately given its target values for each CSS property. In our `.slideUp()` example, the element would be immediately hidden. The callback function is then immediately called, if provided.

The usefulness of the `.stop()` method is evident when we need to animate an element on `mouseenter` and `mouseleave`.

```
<div id="hoverme">  
  Hover me  
    
</div>
```

We can create a nice fade effect without the common problem of multiple queued animations by adding `.stop(true, true)` to the chain.

```
$('#hoverme-stop-2').hover(function() {
    $(this).find('img').stop(true, true).fadeOut();
}, function() {
    $(this).find('img').stop(true, true).fadeIn();
});
```



Animations may be stopped globally by setting the `$.fx.off` property to `true`. When this is done, all animation methods will immediately set elements to their final state when called, rather than displaying an effect.

.delay()

Set a timer to delay execution of subsequent items in the queue.

```
.delay(duration, [queueName])
```

Parameters

- **duration**: An integer indicating the number of milliseconds to delay execution of the next item in the queue
- **queueName (optional)**: A string containing the name of the queue. Defaults to `fx`, the standard effects queue

Return value

The jQuery object, for chaining purposes.

Description

Added to jQuery in version 1.4, the `.delay()` method allows us to delay the execution of functions that follow it in the queue. It can be used with the standard effects queue or with a custom queue.

Using the standard effects queue, we can, for example, set an 800-millisecond delay between the `.slideUp()` and `.fadeIn()` of the `foo` element:

```
$('#foo').slideUp(300).delay(800).fadeIn(400);
```

When this statement is executed, the element slides up for 300 milliseconds and then pauses for 800 milliseconds before fading in for 400 milliseconds.

.queue()

Manipulate the queue of functions to be executed on the matched elements.

```
.queue ( [queueName] )  
.queue ( [queueName] , newQueue )  
.queue ( [queueName] , callback )
```

Parameters (first version)

- `queueName` (optional): A string containing the name of the queue. Defaults to `fx`, the standard effects queue

Parameters (second version)

- `queueName` (optional): A string containing the name of the queue. Defaults to `fx`, the standard effects queue
- `newQueue`: An array of functions to replace the current queue contents

Parameters (third version)

- `queueName` (optional): A string containing the name of the queue. Defaults to `fx`, the standard effects queue
- `callback`: The new function to add to the queue

Return value (first version)

An array of the functions currently in the first element's queue.

Return value (second and third versions)

The jQuery object, for chaining purposes.

Description

Every element can have one or many queues of functions attached to it by jQuery. In most applications, only one queue (called `fx`) is used. Queues allow a sequence of actions to be called on an element asynchronously, without halting program execution. The typical example of this is calling multiple animation methods on an element. For example:

```
$ ('#foo').slideUp().fadeIn();
```

When this statement is executed, the element begins its sliding animation immediately, but the fading transition is placed on the `fx` queue to be called only once the sliding transition is complete.

The `.queue()` method allows us to directly manipulate this queue of functions. The first and second versions of the function allow us to retrieve the entire array of functions or replace it with a new array, respectively.

The third version allows us to place a new function at the end of the queue. This feature is similar to providing a callback function with an animation method, but does not require the callback to be given at the time the animation is performed.

```
$('#foo').slideUp();
$('#foo').queue(function() {
    $.print('Animation complete.');
```

```
    $(this).dequeue();
});
```

This is equivalent to:

```
$('#foo').slideUp(function() {
    $.print('Animation complete.');
```

```
});
```

Note that when adding a function with `.queue()`, we should ensure that `.dequeue()` is eventually called so that the next function in line executes.

.dequeue()

Execute the next function on the queue for the matched elements.

```
.dequeue ( [queueName] )
```

Parameters

- `queueName` (optional): A string containing the name of the queue. Defaults to `fx`, the standard effects queue

Return value

The jQuery object, for chaining purposes.

Description

When `.dequeue()` is called, the next function on the queue is removed from the queue and then executed. This function should, in turn (directly or indirectly), cause `.dequeue()` to be called so that the sequence can continue.

.clearQueue()

Remove from the queue all items that have not yet been run.

```
.clearQueue ( [queueName] )
```

Parameter

- `queueName` (optional): A string containing the name of the queue. Defaults to `fx`, the standard effects queue

Return value

The jQuery object, for chaining purposes.

Description

When the `.clearQueue()` method is called, all functions on the queue that have not been executed are removed from the queue. When used without an argument, `.clearQueue()` removes the remaining functions from `fx`, the standard effects queue. In this way it is similar to `.stop(true)`. However, while the `.stop()` method is meant to be used only with animations, `.clearQueue()` can also be used to remove any function that has been added to a generic jQuery queue with the `.queue()` method.