jQuery: display elements sequentially

Displaying elements sequentially is a common task often required in jQuery. There are two solutions to this problem: one involves the use of a recursive function and the other one relies on animation queues.

The first solution works as intended, but it's not very efficient:

```
(function() {

   var lis = $('li', '#test');

   $('#run').click(function() {
      var i = 0;

      (function displayImages() {
         lis.eq(i++).fadeIn(200, displayImages);
      })();
   });
})();
```

The self-executing function `displayImages()` uses an incremental internal counter to loop through all the elements contained within a given jQuery set. Being used as the callback function of the `fadeIn()` method, it will be invoked until the internal counter reaches the end of the element's set.

The other solution is more efficient (and recommended) because it makes use of the built-in animation queuing mechanism:

```
$('#run').click(function() {
    var lis = $('li', '#test');
```

```
    var delay = 0;

    lis.each(function() {
        var $li = $(this);
        $li.queue('fade', function(next) {
            $li.delay(delay).fadeIn(500, next);
        });

        $li.dequeue('fade');

        delay += 250;
    });
});
```

The `queue()` method creates an animation queue which progressively moves to the next element every time the loop runs. Our queue is named `fade` and we can reset it thanks to the `dequeue()` method.

We've added a small delay between each queue step by setting a progressive delay timer which increases by 250 every time our loop moves to the next element. This timer is used in the jQuery's `delay()` method.

You can see the demo on this page.