4. Placeholders in HTML5.

HTML5 has introduced many features to the browser; some HTML-based, some in the form of JavaScript APIs, but all of them useful.  One of my favorites if the introduction of the placeholder attribute to INPUT elements.  The placeholder attribute shows text in a field until the field is focused upon, then hides the text.  You've seen this technique a billion times with JavaScript, but native HTML5 support is even better!

**The HTML**

<input type="text" name="first\_name" placeholder="Your first name..." class="placeholder">

You'll notice that all you need to do is add a placeholder attribute with the generic text of your choice.  Nice that you don't need JavaScript to create this effect, huh?

**Checking for Placeholder Support**

Since placeholder is a new capability, it's important to check for support in the user's browser:

function hasPlaceholderSupport() {

var input = document.createElement('input');

return ('placeholder' in input);

}

If the user's browser doesn't support the placeholder feature, you will need to employ a fallback with MooTools, Dojo, or the JavaScript toolkit of your choice:

/\* mootools ftw! \*/

var firstNameBox = $('first\_name'),

message = firstNameBox.get('placeholder');

firstNameBox.addEvents({

focus: function() {

if (firstNameBox.value == message) { searchBox.value = ''; }

},

blur: function() {

if(firstNameBox.value == '') { searchBox.value = message; }

}

});

6. Jquery Method (attach() and detach())

The .append() method inserts the specified content as the last child of each element in the jQuery collection (To insert it as the first child, use [.prepend()](http://api.jquery.com/prepend/)).

The .append() and [.appendTo()](http://api.jquery.com/appendTo/) methods perform the same task. The major difference is in the syntax-specifically, in the placement of the content and target. With .append(), the selector expression preceding the method is the container into which the content is inserted. With .appendTo(), on the other hand, the content precedes the method, either as a selector expression or as markup created on the fly, and it is inserted into the target container.

Consider the following HTML:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <h2>Greetings</h2>  <div class="container">  <div class="inner">Hello</div>  <div class="inner">Goodbye</div>  </div> |

You can create content and insert it into several elements at once:

|  |  |
| --- | --- |
| 1 | $( ".inner" ).append( "<p>Test</p>" ); |

Each inner <div> element gets this new content:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <h2>Greetings</h2>  <div class="container">  <div class="inner">  Hello  <p>Test</p>  </div>  <div class="inner">  Goodbye  <p>Test</p>  </div>  </div> |

You can also select an element on the page and insert it into another:

|  |  |
| --- | --- |
| 1 | $( ".container" ).append( $( "h2" ) ); |

If an element selected this way is inserted into a single location elsewhere in the DOM, it will be moved into the target (not cloned):

|  |  |
| --- | --- |
| 1  2  3  4  5 | <div class="container">  <div class="inner">Hello</div>  <div class="inner">Goodbye</div>  <h2>Greetings</h2>  </div> |

**Important**: If there is more than one target element, however, cloned copies of the inserted element will be created for each target except for the last one.

#### Additional Arguments

Similar to other content-adding methods such as [.prepend()](http://api.jquery.com/prepend/) and [.before()](http://api.jquery.com/before/), .append() also supports passing in multiple arguments as input. Supported input includes DOM elements, jQuery objects, HTML strings, and arrays of DOM elements.

For example, the following will insert two new <div>s and an existing <div> as the last three child nodes of the body:

|  |  |
| --- | --- |
| 1  2  3  4 | var $newdiv1 = $( "<div id='object1'/>" ),  newdiv2 = document.createElement( "div" ),  existingdiv1 = document.getElementById( "foo" );    $( "body" ).append( $newdiv1, [ newdiv2, existingdiv1 ] ); |

Since .append() can accept any number of additional arguments, the same result can be achieved by passing in the three <div>s as three separate arguments, like so: $('body').append( $newdiv1, newdiv2, existingdiv1 ). The type and number of arguments will largely depend on how you collect the elements in your code.

The .detach() method is the same as [.remove()](https://api.jquery.com/remove/), except that .detach() keeps all jQuery data associated with the removed elements. This method is useful when removed elements are to be reinserted into the DOM at a later time.

**Example:**

Detach all paragraphs from the DOM

|  |  |
| --- | --- |
|  | **<!doctype html>**  <html lang="en">  <head>  <meta charset="utf-8">  <title>detach demo</title>  <style>  p {  background: yellow;  margin: 6px 0;  }  p**.off** {  background: black;  }  </style>  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>  </head>  <body>    <p>Hello</p>  how are  <p>you?</p>  <button>Attach/detach paragraphs</button>    <script>  $( "p" ).click(**function**() {  $( **this** ).toggleClass( "off" );  });  **var** p;  $( "button" ).click(**function**() {  **if** ( p ) {  p.appendTo( "body" );  p = null;  } **else** {  p = $( "p" ).detach();  }  });  </script>    </body>  </html> |