jQuery - YUI3 Rosetta Stone

Rev 19 March 2010 carlos.bueno.org/jq-yui.pdf

jQuery 1.4.2 | YUI 3.0.0

Getting Started

```
$.foo.bar()
```

```
YUI().use('node', 'module2', 'module3',
  function(Y) {
    Y.foo.bar()
  }
);
```

The **`jQuery`** and **`$`** objects are globals and the jQuery library itself is statically loaded, so they are available immediately.

YUI3 is a little different. It is sandboxed and by default dynamically loaded. The **`Y`** object is local to the function you pass as the last argument to **`YUI().use()`**. Usually you will put all code that uses YUI3 inside one of these functions. This function executes **after** all the referenced modules are loaded and accounted for. This makes for a cleaner Javascript namespace at the cost of some boilerplate. The return value of **`YUI().use()`** is also a Y object, which you can assign to a global variable [eg **`Y = YUI().use(...);`**] and debug with it in a Javascript console.

jQuery 1.4.2 | YUI 3.0.0

Common Idioms

```
$('div.foo:first')
```

```
Y.one('div.foo')
```

jQuery and YUI3 use similar selector syntax, but jQuery has added extensions, mainly convenience pseudo-classes, to the Sizzle CSS3-compliant selector engine. YUI3 comes with three different selector engines; see the section on Selectors.

```
var foo = $('div.foo:first');
foo.some_method();
```

```
var foo = Y.one('div.foo');
if (foo) {
    foo.some_method();
}
```

Return the first element which matches the selector. **`:first`** is a jQuery extension.

If no elements match, **`Y.one()`** returns **`null`** and you should check for it. jQuery selector methods always return a list object with 0 or more elements.

```
$('div.foo')
```

```
Y.all('div.foo')
```

Select all div elements with a class of foo.

```
var foo = $('div.foo');
if (foo) {
    // do something
}
```

```
var foo = Y.all('div.foo');
if (foo.size() > 0) {
    // do something
}
```

If no elements match the selector, **`Y.all()`** returns an empty **`NodeList`**, not the empty list **`[]`**. A **`NodeList`** object is truthy even if it contains no elements, unlike the empty list. so use the **`.size()`** property to check for emptiness.

`.find('p.foo:first')`
`.find('p.foo')`
	`.one('p.foo')`
	`.all('p.foo')`
		Finds **`P`** elements with class **`foo`** that are children of the given node.

`$('<div/>')`	`Y.Node.create('<div/>')`	Create a new DOM element. Does not add it to the document tree.

`.html('foo')`
`.text('foo')`
`.val('foo')`
	`.setContent('foo')`
	`.set('text', 'foo')`
	`.set('value', 'foo')`
		`.set()` is a generic method in YUI for modifying element attributes.
		
		`.setContent(html)` is a convenience wrapper around **`.set('innerHTML', html)`**

`.html()`
`.text()`
`.val()`
	`.get('innerHTML')`
	`.get('text')`
	`.get('value')`
		jQuery tends to overload getters and setters in the same method.

`.attr('foo')`
`.attr('foo', 'bar')`
	`.get('foo')`
	`.set('foo', 'bar')`
		Generic attribute getters and setters.

`.click(fn)`
`.focus(fn)`
`.blur(fn)`
`.mouseout(fn)`
`.mouseover(fn)`
	`.on('click', fn)`
	`.on('focus', fn)`
	`.on('blur', fn)`
	`.on('mouseout', fn)`
	`.on('mouseover', fn)`
		`.on()` is not repeat not chainable by default!

`parent.append('<div/>')`	`parent.append('<div/>')`	Creates a new div element and makes it a child of **`parent`**.

`parent = $('<div/>');`
`$('<p>foo<p>')`
`  .click(fn)`
`  .appendTo(parent);`
	`parent = Y.Node.create('<div/>');`
	`child = Y.Node.create('<p>foo</p>');`
	`child.on('click', fn);`
	`parent.appendChild(child);`
		YUI3 builds element trees outside-in. jQuery can do both outside-in and inside-out (see next entry). YUI3 may add support for **`.appendTo()`** in the future.

`child.appendTo(parent)`	`parent.append(child)`
	`parent.appendChild(child)`
		jQuery's **`.appendTo()`** returns the child element. YUI3's **`.appendChild()`** returns the child element but **`.append()`** returns the parent.
		
		YUI3's **`.append()`** can take either a Node, a bare DOM element, or a string to be converted to HTML.

`.addClass('foo')`
`.removeClass('foo')`
`.toggleClass('foo')`
`.hasClass('foo')`
	`.addClass('foo')`
	`.removeClass('foo')`
	`.toggleClass('foo')`
	`.hasClass('foo')`
		CSS class name manipulation.

`.removeClass('foo').addClass('bar')`	`.replaceClass('foo', 'bar')`	Replace a node's CSS class 'foo' with 'bar'.
`.empty()`	`.get('children').remove(true);`	jQuery's **`.empty()`** also deregisters any events associated with the

		elements being destroyed. The **`true`** argument passed to **`.remove()`** enables the same behavior in YUI3.
`.siblings()`	`.get('parentNode').get('children')`	Note that the YUI3 code is not equivalent: it will contain all child elements including the caller. YUI3 may add support for **`.siblings()`** in a later release.

`.show()`
`.hide()`
	`.setStyle('display', null)`
	`.setStyle('display', 'none')`
		YUI3 does not provide convenience wrappers for show/hide with animations and effects.

jQuery 1.4.2	YUI 3.0.0	Selectors
`$('*')`	`Y.all('*')`	Select all nodes. Note that the default selector engine for YUI3 is CSS 2.1. For all examples in this section, use the **`selector-css3`** module for YUI.
`$(':animated')`		Psuedoclass to select all elements currently being animated. No YUI3 equivalent.
`$(':button')`	`Y.all('input[type=button], button')`	Extension. In both jQuery and YUI3 you can run multiple selectors separated by commas.
`$(':checkbox')`	`Y.all('input[type=checkbox]')`	Extension.
`$(':checked')`	`Y.all(':checked')`	CSS3
`$('parent > child')`	`Y.all('parent > child')`	Immediate child selector (child must be one level below parent)
`$('parent child')`	`Y.all('parent child')`	Descendent selector (child can be at any level below parent)
`$('div.class')`	`Y.all('div.class')`	Class selector
`$(":contains('foo')")`	`Y.all(':contains(foo)')`	Extension to select all elements whose text matches 'foo'. jQuery can take quotes or not. YUI3 requires no quotes. The text matching is plain string comparison, not glob or regexp. Be careful with this one as it will return all matching ancestors, eg **`[html, body, div]`**.

`$(':disabled')`
`$(':enabled')`
	`Y.all(':disabled')`
	`Y.all(':enabled')`
		CSS3. **`'input[disabled]'`** and **`'input:not([disabled])'`** also work in both libraries.

`$(':empty')`	`Y.all(':empty')`	CSS3. Selects all elements that have no child nodes (excluding text nodes).
`$(':parent')`		Extension. Inverse of `:empty`.
`$('div:eq(n)')`	`Y.all('div').item(n)`	Extension. Selects *nth* element. YUI's `item()` will return `null` if there is no `nth` element. jQuery's selector will return the empty list `[]` on a match failure.

`$('div:even')`
`$('div:odd')`
	`Y.all('div').even()`
	`Y.all('div').odd()`
		Extension. Selects all even or odd elements. Note that elements are 0-indexed and the 0th element is considered even. See also YUI3's `NodeList.modulus(n, offset)`.

`$(':file')`	`Y.all('input[type=file]')`	Extension. Find input elements whose type=file.
`$('div:first-child')`	`Y.all('div:first-child')`	CSS3. Selects the first child element of divs.
`$('div:first)`	`Y.one('div')`	The `.one()` method returns `null` if there is no match, and a single Node object if there is.

`$('div:gt(n)');`
`$('div:lt(n)');`
	`Y.all(Y.all('div')._nodes.slice(n + 1));`
	`Y.all(Y.all('div')._nodes.slice(0,n));`
		Extension. `:gt` (greater than) selects all elements from index `n+1` onwards. `:lt` (less than) selects all nodes from 0 up to `n-1`. Note that in the YUI3 example we have to access the private `_nodes` array and perform a `slice()`. `NodeList.slice()` and friends may be added in an upcoming point release. The double call to `Y.all()` is explained in Arrays vs NodeList.

`$('div:has(p)')`		Extension. Selects elements which contain at least one element that matches the specified selector. In this example, all `div` tags which have a `p` tag descendent will be selected.
`$(':header')`	`Y.all('h1,h2,h3,h4,h5,h6,h7')`	Extension. Selects all heading elements. Rarely used.

`$('div:hidden')`	`var hidden = [];`
	`Y.all('div').each(function(node) {`
	`    if ((node.get('offsetWidth') === 0 &&`
	`        node.get('offsetHeight') === 0) ||`
	`        node.get('display') === 'none') {`
	`        hidden.push(node);`
	`    }`
	`});`
	`hidden = Y.all(hidden);`
		Extension. This is a weird one. In jQuery > 1.3.2 `:hidden` selects all elements (or descendents of elements) which take up no visual space. Elements with `display:none` or whose `offsetWidth/offsetHeight` equal 0 are considered hidden. Elements with `visibility:hidden` are not considered hidden.
		
		The YUI3 equivalent would essentially be a port of the jQuery code that implements `:hidden`. This might be a good candidate for a patch to YUI3.

`$('#id')`	`Y.all('#id')`	CSS3. Identity selector.
`$('input:image')`	`Y.all('input[type=image]')`	Extension. Selects all inputs of type image.
`$(':input')`	`Y.all('input,textarea,select,button')`	Extension. Selects all user-editable form elements.
`$(':last-child')`	`Y.all(':last-child')`	CSS3.

`$('div:last')`	`var lst = Y.all('div');`
	`if (lst) {`
	`    var last = lst.item(lst.size()-1);`
	`}`
		The YUI equivalent is cumbersome, but I'm not sure if **`:last`** is popular enough to warrant a patch.

`$('input[type=checkbox][checked]')`	`Y.all('input[type=checkbox][checked]')`	CSS3, multiple attribute selector
`$(':not(div)')`	`Y.all(':not(div)')`	CSS3. Negation selector.
`$(':password')`	`Y.all('input[type=password]')`	Extension.
`$(':radio')`	`Y.all('input[type=radio]')`	Extension.
`$(':reset')`	`Y.all('input[type=reset]')`	Extension.
`$(':selected')`	`Y.all('option[selected]')`	Extension.
`$(':submit')`	`Y.all('input[type=submit]')`	Extension.
`$(':text')`	`Y.all('input[type=text]')`	Extension. Does not select **`textarea`** elements.

jQuery 1.4.2	YUI 3.0.0	Effects

`$('#foo').animate(`
`  {`
`    width:   100,`
`    height:  100,`
`    opacity: 0.5`
`  },`
`  {`
`    duration: 600,`
`    easing:   'swing'`
`  }`
`);`
	`var a = new Y.Anim(`
	`  {`
	`    node: '#foo',`
	`    to: {`
	`        width:   100,`
	`        height:  100,`
	`        opacity: 0.5`
	`    },`
	`    duration: 0.6,`
	`    easing:   Y.Easing.bounceOut`
	`  }`
		The basic syntax and capabilities of both animation libraries are very similar. jQuery has convenience methods for effects like **`.fadeIn()`**, **`.slideUp()`**, etc. jQuery core has two easing functions: 'linear' and 'swing', but jQuery UI comes with many more effects as plugins.
		
		YUI3 has several easing algorithms built-in, and offers powerful tools such as animations over Besizer curves. Make sure to load the **`'anim'`** module in your call to **`YUI().use()`**.

```
);
a.run();
```

```
$('#.foo').fadeOut();

// or

$('#.foo').hide(600);
```

```
var a = new Y.Anim(
  {
    node: '#foo',
    to: {opacity:  0.0},
    duration: 0.2,
    easing:   Y.Easing.easeOut
  }
);
a.on('end', function(ev) {
    ev.target._node
      .setStyle('display', 'none');
});
a.run();
```

.fadeOut() fades the opacity to 0, then sets **display:none** on the element. **fadeIn()** is naturally the inverse. Note that jQuery effects tend to default to 200 or 600ms while YUI defaults to 1,000ms. YUI durations are in fractions of seconds; jQuery durations are set in milliseconds.

Annoyingly, YUI Anim objects have events you can attach functions to, but you have to poke the private **_node** property to retrieve the element being animated.

Array vs NodeList

jQuery 1.4.2 | YUI 3.0.0

```
$('.foo').array_method(args)
```

```
Y.all(Y.all('.foo')._nodes.array_method(args))
```

Any Array operation that you can perform on a jQuery list can be translated to YUI in this form. YUI NodeList objects are not native Arrays, but the private **_nodes** property is. However, calling list operations like **.concat()** on **._nodes** results in an array of DOM elements, not a NodeList. To generate a new NodeList for the new array, you have wrap it in a call to **Y.all()**. All of this wrapping and unwrapping suggests a patch to YUI.

```
$('div').slice(x, y)
```

```
Y.all(Y.all('div')._nodes.slice(x, y))
```

Return the *xth* to the *yth* div elements.

```
$('div').concat($('p'))
```

```
Y.all(
    Y.all('div')._nodes.concat(
        Y.all('p')._nodes
    )
)
```

NodeList.concat() and friends are coming to a point release of YUI.

```
var foo = $('.foo');
for (var i=0; i<foo.length; i++) {
    // per-element code here.
}
```

```
Y.all('.foo').each(
  function(node, idx, lst) {
    // per-node code here.
  }
);
```

YUI's **.each()** is like the **for** loop. It returns the original NodeList to help with chaining.

```
$('.foo').filter('.bar')
```

```
Y.all('.foo').filter('.bar')
```

The **.filter()** method in both libraries both take CSS selectors as filter criteria. jQuery's **.filter()** can also take a function.

```
var fn = function(idx) {
    return this.property === 'value';
};
$('.foo').filter(fn);
```

```
var filtered = [];
Y.all('.foo').each(
  function(node) {
    if (node.get('property') === 'value') {
      filtered.push(node._node);
    }
  }
);
filtered = Y.all(filtered);
```

Classic functional programming filter function. Given a list of elements, run the function on each and return a list of those which evaluated true. **`NodeList.filter(fn)`** is coming to a future point release of YUI3.

```
$('.foo').map(
  function(idx, el) {
    some_function(el);
  }
);
```

```
var mapped = [];
Y.all('.foo').each(
  function(node) {
    mapped.push(
        some_function(node)
    );
  }
);
mapped = Y.all(mapped);
```

jQuery's **`.map()`** returns a list of the return values of calls to the given function. **`NodeList.map(fn)`** is coming to a future point release of YUI3.

jQuery 1.4.2	YUI 3.0.0	Ajax

```
$.ajax({
  url:      url,
  data:     data,
  success:  successFn
});
```

```
Y.io(url, {
    data: data,
    on:   {success: successFn}
});
```

YUI.io has extra options for failure mode callbacks, headers, cross-frame i/o, etc. jQuery.ajax() has some interesting options for async, context, and filtering. Make sure to load the YUI 'io' module.

```
Y.io(url, {
    data: data,
    on:   {success: successFn},
    xdr:  {use: 'flash'}
});
```

Cross-domain requests via a Flash helper. No jQuery equivalent.

```
$('#message').load('/ajax/test.html');
```

```
var fn = function(txnid, o) {
    Y.one('#message').setContent(
        o.responseText
    );
}
Y.io('/ajax/test.html', {
    on: { success: fn }
});
```

Load the content of a given URL and replace the contents of **`#message`** with it.

jQuery 1.4.2	YUI 3.0.0	CSS

`.addClass('foo')`
`.removeClass('foo')`
`.toggleClass('foo')`
`.hasClass('foo')`
	`.addClass('foo')`
	`.removeClass('foo')`
	`.toggleClass('foo')`
	`.hasClass('foo')`
		CSS class name manipulation.

`.removeClass('foo').addClass('bar')`	`.replaceClass('foo', 'bar')`	Replace node's CSS class 'foo' with 'bar'.
`.css('display', 'block')`	`.setStyle('display', 'block')`	Set a single CSS property

`.css({`
`    height:  100,`
`    width:   100,`
`    display: 'block'`
`})`
	`.setStyles({`
	`    height:  100,`
	`    width:   100,`
	`    display: 'block'`
	`})`
		Set multiple CSS properties with a dictionary.

`.css('display')`	`.getStyle('display')`	Get the current value for a CSS property.

`.height()`
`.width()`
	???	Computed height / width. Excludes padding and borders.

`.innerHeight()`
`.innerWidth()`
	???	Includes padding but not border

`.outerHeight()`
`.outerWidth()`
	`.get('offsetHeight')`
	`.get('offsetWidth')`
		Includes padding and border

`.position()`
`// {left: 123, top: 456}`
	`.getXY()`
	`// [123, 456]`
		Get the computed x,y coordinates. The coordinates are relative to the nearest ancestor element that is relatively or absolutely positioned.