jQuery - YUI3 Rosetta Stone

If you see a bug or want to add to this list, you can check out the project on GitHub or leave a comment. Thanks!

- **Common Idioms**
  Some parts of libraries are more popular than others. This first section is a mixed bag of popular idioms in YUI3 and jQuery.
- **Selectors**
  jQuery uses the Sizzle CSS selector library, which is CSS3-compliant but also has a great many extra pseudoclasses and extensions, some of which (eg `:first`) are used very often while others (eg `:header`) are obscure.

  YUI3 can operate with three selector engines: browser native (the default), CSS2, and CSS3. This section documents mostly the differences between jQuery and the CSS3 specification.
- **Animations & Effects**
  The animation libraries of both libraries have substantial overlap, though jQuery makes it a bit easier to combine show() and hide() operations with animated effects.
- **Arrays vs NodeList**
  The fundamental unit of jQuery is a Javascript Array containing 0 or more DOM elements. These Array objects have extra `.on()`, `.click()`, `.map()`, etc methods attached to them in addition to the built-in list operations like `.slice()` and `.concat()`.

  The fundamental units in YUI are Node objects, which wrap DOM elements, and NodeLists which are collections of Nodes. NodeLists are not Arrays and are not natively iterable.
- **Ajax & Related Diseases**

Common Idioms

jQuery 1.4.2	YUI 3.0.0	

`$.foo.bar()`	`YUI().use('node', function(Y) {`
	`  Y.foo.bar()`
	`});`
		The **`jQuery`** and **`$`** objects are globals and the jQuery library itself is statically loaded, so they are available immediately. YUI3 is sandboxed and by default is dynamically loaded. The **`Y`** object is only available inside the anonymous function sandbox, and the function only executes after all scripts are present and accounted for. This makes for a cleaner environment (eg, you can mix multiple versions of YUI in the same page) at the cost of some counterintuitive onload behavior. Also see Selectors below.

`$('div.foo:first')`	`Y.one('div.foo')`	jQuery and YUI3 use the similar selector engine syntax, but jQuery has added extensions to the Sizzle CSS3-compliant selector engine. YUI3 comes with three different selector engines; see the section on Selectors. **`Y.one()`** returns single items. If no elements match the selector, it returns **`null`**, not the empty list **`[]`**.
`$('div.foo')`	`Y.all('div.foo')`	If no elements match the selector, **`Y.all()`** returns an empty **`NodeList`**, not the empty list **`[]`**. A **`NodeList`** is "truthy" 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')`	
`$('<div/>')`	`Y.Node.create('<div/>')`	

`.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')`	

`.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/>')`	

`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')`
	`.addClass('foo')`
	`.removeClass('foo')`
	`.toggleClass('foo')`	

`.removeClass('foo').addClass('bar')`	`.replaceClass('foo', '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.

Selectors

jQuery 1.4.2	YUI 3.0.0	
`$('*')`	`YUI().use('node', 'selector-css3', function(Y) { Y.all('*') } );`	Select all nodes. Note the **`selector-css3`** module for YUI. For the rest of the examples in this section, please assume this module.
`$(':animated')`		Psuedoclass to select all elements currently being animated. No YUI3 equivalent.
`$(':button')`	`Y.all('input[type=button], button')`	In both jQuery and YUI3 you can run multiple selectors separated by commas.
`$(':checkbox')`	`Y.all('input[type=checkbox]')`	
`$(':checked')`	`Y.all(':checked')`	CSS3
`$('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)`**.
`$(':file')`	`Y.all('input[type=file]')`	
`$('div:first-child')`	`Y.all('div:first-child')`	CSS3. Selects the first child element.
`$('div:first)`	`Y.one('div')`	The **`.one()`** method returns **`null`** if there is no match, and a single Node object if there is.
`$('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

`$('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.

Effects

jQuery 1.4.2

```
$('#foo').animate(
  {
    width:   100,
    height:  100,
    opacity: 0.5
  },
  {
    duration: 600,
    easing:   'swing'
  }
);
```

YUI 3.0.0

```
var a = new Y.Anim(
  {
    node: '#foo',
    to: {
        width:   100,
        height:  100,
        opacity: 0.5
    },
    duration: 0.6,
    easing:   Y.Easing.bounceOut
  }
);
a.run();
```

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 some complex tools such as animations over Besizer curves. Make sure to load the **`"anim"`** module inside **`YUI().use()`**.

jQuery 1.4.2

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

// or

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

YUI 3.0.0

```
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 1,000. 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	
`$('.foo').list_method(args)`	`Y.all(Y.all('.foo')._nodes.list_method(args))`	Any Array operation that you can perform on a jQuery list can be translated to YUI in this form. YUI NodeList objects are not native Arrays, but the private **`_nodes`** property is. However, calling list operations like **`.concat()`** on **`._nodes`** results in an array of DOM elements, not a NodeList. To generate a new NodeList for the new array, you have wrap it in a call to **`Y.all()`**. All of this wrapping and unwrapping suggests a patch to YUI.
`$('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)`
	`)`
		`.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.

`$('.foo').filter(`
`  function(idx) {`
`  return this.property === 'value';`
`  }`
`);`
	`var filtered = [];`
	`Y.all('.foo').each(`
	`  function(node) {`
	`  if (node.get(property) === 'value') {`
	`  filtered.push(node._node);`
	`  }`
	`  }`
	`);`
	`filtered = Y.all(filtered);`
		Given a list of elements, return only those whose *property* matches *value*.

`$('.foo').map(`
`  function(idx, el) {`
`  some_function($(el));`
`  }`
`);`
	`var mapped = [];`
	`Y.all('.foo').each(`
	`  function(node) {`
	`  mapped.push(`
	`  some_function(node)._node`
	`  );`
	`  }`
	`);`
	`mapped = Y.all(mapped);`
		jQuery's **`.map()`** returns a list of the return values of calls to the given function.

Ajax & Related Diseases

jQuery 1.4.2	YUI 3.0.0	

`$.ajax({`
`  url:  url,`
`  data:  data,`
`  success:  successFn`
`});`
	`YUI().use('io', function(Y) {`
	`  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.

`$('#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.