**Utility Methods**

jQuery offers several utility methods in the $ namespace. These methods are helpful for accomplishing routine programming tasks. Below are examples of a few of the utility methods; for a complete reference on jQuery utility methods, visit <http://api.jquery.com/category/utilities/>.

$.trim

Removes leading and trailing whitespace.

$.trim(' lots of extra whitespace ');

// returns 'lots of extra whitespace'

$.each

Iterates over arrays and objects.

$.each([ 'foo', 'bar', 'baz' ], function(idx, val) {

console.log('element ' + idx + 'is ' + val);

});

$.each({ foo : 'bar', baz : 'bim' }, function(k, v) {

console.log(k + ' : ' + v);

});

**Note**

There is also a method $.fn.each, which is used for iterating over a selection of elements.

$.inArray

Returns a value's index in an array, or -1 if the value is not in the array.

var myArray = [ 1, 2, 3, 5 ];

if ($.inArray(4, myArray) !== -1) {

console.log('found it!');

}

$.extend

Changes the properties of the first object using the properties of subsequent objects.

var firstObject = { foo : 'bar', a : 'b' };

var secondObject = { foo : 'baz' };

var newObject = $.extend(firstObject, secondObject);

console.log(firstObject.foo); // 'baz'

console.log(newObject.foo); // 'baz'

If you don't want to change any of the objects you pass to $.extend, pass an empty object as the first argument.

var firstObject = { foo : 'bar', a : 'b' };

var secondObject = { foo : 'baz' };

var newObject = $.extend({}, firstObject, secondObject);

console.log(firstObject.foo); // 'bar'

console.log(newObject.foo); // 'baz'

$.proxy

Returns a function that will always run in the provided scope — that is, sets the meaning of this inside the passed function to the second argument.

var myFunction = function() { console.log(this); };

var myObject = { foo : 'bar' };

myFunction(); // logs window object

var myProxyFunction = $.proxy(myFunction, myObject);

myProxyFunction(); // logs myObject object

If you have an object with methods, you can pass the object and the name of a method to return a function that will always run in the scope of the object.

var myObject = {

myFn : function() {

console.log(this);

}

};

$('#foo').click(myObject.myFn); // logs DOM element #foo

$('#foo').click($.proxy(myObject, 'myFn')); // logs myObject