Zepto.[js](http://monw3c.blogbus.com/c1517356/)是支持移动WebKit浏览器的JavaScript框架，具有与jQuery兼容的语法。2-5k的库，通过不错的API处理绝大多数的基本工作。以下是Zepto.js的API

Basic call with [CSS](http://monw3c.blogbus.com/tag/css/) selector:

$('p>span').html('yoho').[css](http://monw3c.blogbus.com/tag/css/)({color**:** 'red'});

Instead of a selector, a DOM Element, or a list of nodes can be passed in.

The $ function takes an optional context argument, which can be a DOM Element or a Zepto object:

$('span', $('p')) *// -> find all <span> elements in <p> elements*

$('p').bind('click', **function**(){

$('span', **this**).css({color**:** 'red'}); *// affects "span" children/grandchildren*

});

Context and .find calls are equivalent:

$('span', $('p')) *// same*

$('p').find('span') *// same*

Element functions:

get() *// return array of all elements found*

get(0) *// return first element found*

size() *// the number of elements in collection*

each(callback) *// iterate over collection, calling callback for every element*

index('selector') *// the position of element matching 'selector' in the current collection*

first() *// new collection containing only the first matched element*

last() *// new collection containing only the last matched element*

add() *// merges collections of elements*

eq(n) *// reduce the set of matched elements to the one at the specified index*

find('selector') *// find all children/grandchildren that match the given selector*

closest('selector') *// find the first matching element by going upwards starting from the current element*

parents(['selector']) *// get all ancestors of elements in collection, optionally filtered by a selector*

parent() *// immediate parent node of each element in collection*

children('selector') *// immediate children of each element in collection, optionally filtered by a selector*

siblings('selector') *// elements that share the same immediate parent (siblings) of each element in collection, optionally filtered by a selector*

next() *// next siblings*

prev() *// previous siblings*

filter('selector') *// reduce the current set of elements to match the given selector*

is('selector') *// returns true/false if first element matches the selector*

not('selector') *// remove elements matching 'selector' from the current collection*

not(**function**(index){**return** **true** **/** **false**;}) *// remove elements from current collection if the callback method returns `true`*

remove() *// remove element*

html('new html') *// set the contents of the element(s)*

html(**function**(index, oldhtml){ **return** ...; }) *// set the contents of the element(s) from a method*

html() *// get first element's .innerHTML*

text() *// get first element's .textContent*

text('new text') *// set the text contents of the element(s)*

append(), prepend() *// like html(), but add html (or a DOM Element or a Zepto object) to element contents*

before(), after() *// add html (or a DOM Element or a Zepto object) before/after the element*

appendTo(), prependTo() *// reverse appending/prepending*

show() *// forces elements to be displayed (only works correctly for block elements right now)*

hide() *// removes a elements from layout*

offset() *// get object with top: left: width: height: properties (in px)*

height() *// get first elements height in px, including padding and border (equivalent to jQuery.outerHeight(false))*

width() *// get first elements width in px, including padding and border (equivalent to jQuery.outerWidth(false))*

attr('attribute') *// get element attribute*

attr('attribute', 'value') *// set element attribute*

attr('attribute', **function**(index, oldAttr){ **return** ...; }) *// set the value of 'attribute' from a method, for each element in collection*

removeAttr('attribute') *// removes an attribute*

data('name') *// gets the value of the "data-name" attribute*

data('name', 'value') *// sets the value of the "data-name" attribute*

css('css property', 'value') *// set a CSS property*

css({ property1**:** value1, property2**:** value2 }) *// set multiple CSS properties*

css('css property') *// get this CSS property of the first element, looks at both .style object properties and the computed style*

addClass('classname') *// adds a CSS class name*

addClass(**function**(index, existingClasses){ **return** ...; }) *// adds a CSS class name from a method*

removeClass('classname') *// removes a CSS class name*

removeClass(**function**(index, existingClasses){ **return** ...; }) *// removes a CSS class name from a method*

hasClass('classname') *// returns true of first element has a classname set*

toggleClass('classname'[, **switch**]) *// adds/removes class, or adds/removes it when switch == true/false*

toggleClass(**function**(index, existingClasses){ **return** ...; }) *// adds/removes class from a method*

on(type, [selector,] **function**) *// add event listener to elements*

off(type, [selector,] **function**) *// remove event listener from elements*

bind(type, **function**) *// add an event listener (see below)*

one(type, **function**) *// add an event listener that only fires once*

unbind([type [, **function**]]) *// remove event listeners*

delegate(selector, type, **function**) *// add an event listener w/ event delegation (see below)*

undelegate(selector [, type[, **function**]]) *// remove event listeners w/ event delegation*

live(type, **function**) *// add an event listener that listens to the selector for current and future elements*

die([, type[, **function**]]) *// remove live listener*

trigger(type) *// triggers an event*

submit() *// trigger form submit event*

val() *// returns the value of the form element*

val('value') *// sets the value of the form element*

CSS Animation

animate(transforms, duration, easing, callback)

animate(transforms, { duration**:** milliseconds, easing**:** '...', complete**:** callback })

*// use CSS transform/opacity to do an animation,*

*// optionally supply a callback method to be executed after the animation is complete*

Non-jQuery functions

pluck(property)

*// return property for each element*

*// e.g. pluck('innerHTML') returns an array of all innerHTML properties of all elements found*

Utility functions:

$(document).ready(**function**(){ ... }); *// call function after DOM is ready to use (before load event fires)*

$.isFunction(**function**), $.isObject(object), $.isArray(array); *// returns true if given parameter is a function; an object; or an array, respectively*

$.extend(target, object1 [,objectN]) *// extends (merge) the target object with additional objects. Modifies and returns target*

Event handlers

Adding an event listener:

$('some selector').bind('click', **function**(event){ ... });

Adding an event listener on multiple events:

$('some selector').bind('touchstart touchmove touchend', **function**(event){ ... });

Adding one event listener that uses event delegation to be only active on a range of children/grandchildren (as given with the subselector):

$('some selector').delegate('some subselector', 'touchstart', **function**(event){ alert("I'm touched!") });

Adding a "live" event listener, that fires on all elements that match the selector now and in the future:

$('p.yay').live('click', **function**(){ alert("Clicked a p.yay element!") });

Removing an event listener:

$('some selector').unbind('click', listener);

Removing all event listeners for a particular event:

$('some selector').unbind('click');

Removing all event listeners:

$('some selector').unbind();

Touch events

Zepto has several extensions over the jQuery API to make it easy to react to touch events.

Tapping:

$('some selector').tap(**function**(){ ... });

Double-tapping:

$('some selector').doubleTap(**function**(){ ... });

Swiping (e.g. "delete" button when swiping over a list entry):