jqMobi Javascript Frameworks Cheat Sheet

jqMobi is a Javascript framework targeted at HTML5 browsers

It is comprised of three parts: jqMobi (a blazingly fast query selector library that supports W3C queries), jqUi (a UI/UX library for mobile applications in a webkit browser), and jqPlugins (plugins for mobile applications in a webkit browser).

jqMobi 1.02

Application Preparation

Download the jqMobi Frameworks
https://github.com/appmobi/jQ.Mobi

To use jqMobi in an application, Include the following script tag in your html file:

```
<script type="text/javascript" charset="utf-8" src="jq.mobi.min.js"></script>
```

To use/include all jqPlugins in an application, include the following script tag in your html file:

```
<script type="text/javascript" charset="utf-8" src="jq.web.min.js"></script>
```

You can also include individual plugins if desired, for example:

```
<script type="text/javascript" charset="utf-8" src="plugins/jq.scroller.js"></script>
```

To include jqUi in your application, add the script tag and link tag:

```
<script type="text/javascript" charset="utf-8" src="ui/jq.ui.min.js"></script>
<link rel="stylesheet" type="text/css" href="jq.ui.css" />
```

jqMobi Components

Name & Description	Contents
jqMobi	jq.mobi.min.js - a blazingly fast query selector tool that is optimized for HTML5 browsers
jqPlugins	jq.web.min.js - plugins for mobile applications in a webkit browser that can be used for AppMobi and general mobile web development. Consists of: jq.alphatable, jq.carousel, jq.css3animate, jq.drawer, jq.passwordBox, jq.swipeListener, jq.scroller, jq.selectBox, and jq.template
jqUi	jq.ui.min.js – UI/UX framework for building jqMobi mobile apps targeted at the Webkit browser comprised of css3animate, passwordBox, scroller, and selectBox components of jqPlugins, plus: • Fixed header bar, Auto scrolling content panels, and an optional navbar to segment your application

jqMobi - Query selector library

Usage

To use jqMobi include thefollowing script tag in your html file:

```
<script src="jq.mobi.min.js"></script>
```

This creates two objects you can work with. It will NOT override a pre-existing $ object.

```
$("#main")
jq("#main")
```

Query Selector

W3C spec'ed queries are supported. The following can be done:

```
$("input[type='text']")
```

The following is NOT supported by the browsers:

```
$("input:text")
```

In some functions, an additional selector can be used. This can be a string, array, or a jqMobi object. We currently do not support functions.

Syntax:

Basic call

```
$("#id").hide()
```

A Dom element, selector, list of nodes, or HTML string can be specified.

```
$("span").bind("click",function(){console.log("clicked");}); // -> find all span elements and attach a click event
```

An HTML string can be passed and it will create the object for you.

```
var myDiv=$("<div id='foo'>") //Creates a div object and returns it
```

jqMobi Ajax calls

```
.get(url,callback) //Makes an Ajax request to the URL and executes the callback funtion with the result
.post(url,data,callback,dataType) //Makes an Ajax POST request to the URL with the data and executes the callback with the result. An optional dataType can be passed in, as some webservices require the header
.getJSON(url,data,callback) //Makes an ajax request with the data and executes callback function passing in a JSON object from the Ajax response into the callback function.
```

If more access is needed, the following can be used:

```
.ajax {
   type:'POST', //defaults to GET
   url:'/api/getinfo', //defaults to window.location
   contentType:'application/json', //defaults to application/x-www-form-urlencoded
   headers:{},
   dataType:'application/json', //defaults to text/html
   data:{username:foo}, //Can be a Key/Value pair string or object. If it's an object, $.serialize is called to turn it into a Key/Value pair string
   success:function(data){}, //function to call on successful Ajax request
   error:function(data){}, //function to call when an error exists in the Ajax request
}
```

If the url contains the pattern =? in it, a jsonP request will be made. These can ONLY be GET requests

Plugins

jqMobi is built with the extendability to add plugins. To create a plugin, extend the $.fn object by passing a reference of the main jqMobi object

```
(function($){
  $.fn['foo']=function(){
     alert("bar");
  }
})(jq);
```

jqMobi API functions

```
.map(elements,callback) //Executes callback function on each element
.each(elements,callback) //Iterate through elements and execute callback
.extend(target,{params}) //Extends an object with additional arguments
.isArray(data) //Returns true/false if data is an array
.isFunction(data) //Returns true/false if data is a function
.isObject(param) //Returns true/false if param is an object
.ready(callback) // Callback executed when DOMContentLoaded happens
.find(selector) // Find all chidlren that match the given selector
.html(['new html']) // Get/Set the elements .innerHTML
.text(['new text']) // Get/Set the elements .innerTEXT
.css('property',['value']) //Get/Set the elements css property to value
.empty() //Sets the elements .innerHTML to an empty string
.hide() //Sets the elements display css attribute to "none"
.show() //Sets the elements display css attribute to "block"
.toggle() //Toggles the elements display css attribute
.val(["value"]) //Get/Set the elements value property
.attr("attribute",["value"]) //Get/Set the elements attribute
.removeAttr("attribute") //Removes the attribute from the elements
.remove() //Remove an element from the Dom
.addClass("className") //Adds the css clas name to the selected elements
.removeClass("className") //Removes a css class from the selected elements
.hasClass("className",[element]) //Checks to see if an element has a class
.append(element,[insert]) //Appends an element to the selected elements
.prepend(element) //Prepends an element to the selected elements
.insertBefore(target) //Inserts a collection before the target (adjacent)
.insertAfter(target) //Inserts a collection after the target (adjacent)
.get([index]) //Get raw DOM element based on index. () returns first element
.offset() //Calculates the first elements offset on the screen
.parent(selector) //Returns the parent nodes based off selector
.children(selector) //Returns the children of the elements
.siblings(selector) //Returns the siblings of the elemnts
.closest(selector,[context]) //Returns the closest element based off selector
.filter(selector) //Filters the elements based off selector
.not(selector) //Return all matches that do NOT match the selector
.data(key,[value])//Gets/Sets a data-* attribute for the param
.end() //Rolls back the jqMobi elements when filters were applied
.clone() //Clones the nodes in the collection
.size() //Returns the number of elements in a collection
.serialize(grouping) //Serializes a form into a query string
.jsonP(options) //Execute a jsonP call, allowing cross domain scripting
.bind("event",function(){}) //Binds a function to the event listener
.unbind("event",[callback]) //Unbinds a function to the event listener
.one("event",callback) //Bind event to each element - only executes once
.delegate(selector,"event",callback) //Delegate an event bassed off selector
.undelegate(selector,"event",[callback]) //Unbind an event registered through delegate
.on("event",selector,callback) //Similar to .delegate()
.off("event",selector,[callback]) //Removes event listeners for .on()
.trigger("event",data) //Trigger an event and pass in optional data
.proxy(callback,context) //Creates a proxy function so the 'this' context can be changed in the function
```

jqMobi Helper calls

```
.param() //Serialize a JSON object into KVP for a querystring
.parseJSON(string) //Backwards compatability JSON parsing call. Uses the browsers native JSON parser
.parseXML(string) //Parses a string and returns a XML document version
.uuid //Utility function to create a pseudo GUID
.Event(type,props) //Creates a custom event to be used internally
```

jqMobi OS detectors

```
$.os.webkit      //True if webkit found in the user agent
$.os.android     //True if anroid useragent
$.os.ipad        //True if iPad useragent
$.os.iphone      //Tru if iPhone user agent
$.os.webos       //True if WebOS detected
$.os.touchpad    //True if WebOS and Touchpad user agent
$.os.ios         //True if iPad or iPhone
$.os.blackberry //True if Blackberry PlayBook or OS >=6
```

jqUi

UI/UX library for mobile applications in a webkit browser

Usage

There are four special registered div blocks for a layout based off id's. The ids are header (top header), content (content area), navbar (bottom navbar), and jqUi (app container).

To add divs to the content, simply set the class to "panel"

```
<div id="my_id" title="My Title" class="panel">
   <!-- content goes here -->
</div>
```

To add a new div dynamically, call the function addContentDiv(id,content);

```
<script>$.ui.addContentDiv("newdiv","This is some new html");</script>
```

To open links in a new window, set the target property

```
<a href=http://www.appmobi.com target="_blank">AppMobi</a>
```

Select from six transitions by setting the data-transition property (default is slide)

```
<a href="#login" data-transition="slide">Login</a>  //Slide left/right
<a href="#login" data-transition="up">Login</a>  //Slide up/down
<a href="#login" data-transition="down">Login</a>  //Slide down/up
<a href="#login" data-transition="flip">Login</a>  //Flip the page
<a href="#login" data-transition="fade">Login</a>  //Fade in/out
<a href="#login" data-transition="pop">Login</a>  //Pop in/out
```

To navigate to a page transition via javascript, call the function loadContent(_id_,clearHistory,goBackInHistory,transition)

```
<script>$.ui.loadContent("my_id",false,false,"pop");</script>
```

To update content, call the function updateContentDiv(id,content);

```
<script>$.ui.updateContentDiv("login","New Login HTML");</script>
```

To prevent a div from scrolling, set the propert "scrolling" to "no" on the div

```
<div class="panel" scrolling="no"></div>
```

To add custom headers or footers via the <header> or <footer> tags, reference them on the panel with the data-header of data-footer attribute, respectively.

jqUi API functions ($.ui)

```
.loadDefaultHash //Boolean to load/not load panel from hash when app started
                  (default is True)
.blockUI(opacity) //Throw up a mask and block the UI
.unblockUI() //Removes the UI mask
.removeFooterMenu() //Removes the bottom nav bar from app
.showNavMenu //Boolean to show bottom nav bar
.autoLaunch //Boolean to auto launch jqUi
.isAjaxApp //Boolean that when true treats every request as if the anchor had
            data-refresh-ajax=true and data-persist-ajax=true
.showLoading //Boolean to show/not show loading spinner on ajax requests
.launch() //Launch jqUi. If autoLaunch is true, gets called on DOMContentLoaded
.showBackButton //Boolean to show the back button
.resetScrollers //Boolean to reset the scroller position when navigating panels
.ready(function) //Function to fire when jqUi is ready and completed launch
.setBackButtonStyle(class) //Override the back button class name
.goBack() //Initiate a back transition
.clearHistory() //Clear the history queue
.updateBadge(target,value,[position]) //Update a badge on the selected target
.removeBadge(target) //Remove a badge from the selected target
.toggleNavMenu([force]) //Toggle the bottom nav menu
.toggleHeaderMenu([force]) //Toggle the top header menu
.toggleSideMenu([force]) //Toggle the side menu
.updateNavbarElements(elements) //Update the elements in the navbar
.updateSideMenu(elements) //Update the elements in the side menu
.setTitle(value) //Set the title of the current panel
.setBackButtonText(value) //Override the text for the back button
.showMask(text) //Show the loading mask
.hideMask() //Hide the loading mask
.showModal() //Load a content panel in a modal window
.hideModal() //Hide the modal window and remove the content
.updateContentDiv(id,content) //Update the HTML in a content panel
.addContentDiv(id,content,title) //Dynamically create a new panel
.loadContent(target,newTab,goBack,transition) //Initiate a transition or load
                                                via ajax
.scrollToTop(id) //Scroll a panel to the top
.slideTransition(prevPanel,currPanel,goBack) //Initiate a sliding transition
.finishTransition(oldDiv) //Called at end of each transition to hide the old
                           div and reset the doingTransition variable
```

jqPlugins Access ($.ui)

```
.actionSheet(opts) //Shorthand call to jq.actionsheet plugin. Auto wired to
                    jQUi div.
.popup(opts) //Wrapper to jq.popup plugin. If a text string is passed in, acts
              Like an alert box and just gives a message.
```

*** See jq.ui.css for additional button colors and ways to change the theme

jq.scroller

create a fixed height/width div and scroll veritical and/or horizontal

Usage

Create an outer container div that has the height and width of the area to be visible

```
<div id="my_div_container" style="width:100%;height:300px">
   <!-- div from below goes here -->
</div>
```

Create a div with the content inside that is to be scrollable.

```
<div id="my_div" >
   <!-- content goes here -->
</div>
```

Call the javascript function

```
var scroller = $("#my_div").scroller();
```

There are additional configuration options that are passed in as an object parameter

```
var options={
   verticalScroll:true,  //vertical scrolling
   horizontalScroll:false,  //horizontal scrolling
   scrollBars:true   //display scrollbars
   vScrollCSS : "scrollBarV", //CSS class for veritcal scrollbar
   hScrollCSS : "scrollBarH", //CSS class for horizontal scrollbar
   refresh:true, //Adds 'Pull to refresh' at the top
   refreshFunction:updateMessage //callback function to execute on pull to
refresh
}
var scroller = $("#my_div").scroller(options);
```

You can also have it scroll to a specific position

```
scroller.scrollTo({x:-100,y:-200});
```

If you want to make persistent scrollbars, override the opacity style for the class in your css

```
.scrollBarV { opacity:.8 !important}
```

jq.carousel

Create vertical or horizontal carousels

Usage

Create a div with content inside that is to be paged between. The height and width of this div must be set, along with overflow:hidden.

```
<div id="my_div" style="width:768px;height:400px;overflow:hidden">
   <div style="float:left;width:766px;height:400;border:1px solid
white;background:yellow;"></div>
   <div style="float:left;width:766px;height:400;border:1px solid
white;background:green;"></div>
</div>
```

Create div to display dots for paging (optional)

```
<div id="carousel_dots" style="text-align: center; margin-left: auto;
margin-right: auto; clear: both;position:relative;top:-40px;z-
index:200"></div>
```

Call the javascript function

```
var carousel = $("#my_div").carousel();
```

There are additional configuration options that are passed in as an object parameter

```
var options={
   vertical:false, // page up/down
   horizontal:true, // page left/right
   pagingDiv:null, // div to hold the dots for paging
   pagingCssName:"carousel_paging", //classname for the paging dots
   pagingCssNameSelected: "carousel_paging_selected" //classname for the
selected page dots
}
var carousel = $("#my_div").carousel(options);
```

jq.css3animate

Helper function for doing CSS3 animations

Usage

Call the function

```
$("#div").css3animate(options);
```

There are additional configuration options that are passed in as an object parameter

```
var options={
   x:20, //x axis move. this can be a number (40), percent (50%), or pixels
(40px) - if it's a number, px is added to it.
   y:20, //y axis move
   opacity:.5, //opacity to change to
   width:"100px", //style.width to change to
   height:"100px", //style.height to change to
   origin:"50% 50%", //offset to start the animation from default is 0 0
   rotateX:"50deg", //rotate along the x-axis
   rotateY:"50deg", //rotate along the y-axis
   skewX:"50deg", //skew along the x-axis
   skewY:"50deg", //skew along the y-axis
   time:"300ms", //time for transition
   timingFunction:"linear", //timing function for animation
   previous:false // move from previous position - not tested with
animation by percentages
   callback:function(){console.log("finished animation")}
}

$("#div").css3animate(options);
```

Chain animations by passing in an animation in the callback function

```
$("#animate").css3Animate({x : 20,y : 30,time : "300ms",callback :
function() {$("#animate").css3Animate({x : 20,y : 30,time :
"500ms",previous : true,callback : function() {reset();}});}});
```

You can create a queue of animations too.

```
var tmp = new $.css3AnimateQueue();
var tmp2 = new $.css3AnimateQueue();
tmp.push({id:"animate",x:20,y:30,time:"300ms"});
tmp.push({id:"animate",x:20,y:30,time:"500ms",previous:true});
tmp.push({id:"animate",x:0,y:0,time:"0ms"});
tmp.push({id:"animate",x:20,y:30,time:"300ms"});
tmp.push({id:"animate",x:20,y:30,time:"500ms",previous:true});
tmp.run();
```

jq.template

Template parsing library, similar to popular scripting languages syntax. The parsing engine is John Resig's micro template engine.

Usage

Create a template. The easiest way is to create <script> tags with content type of text/html and set an id.

```
<script type='text/html' id='user_info'>
    Name <%=userinfo.name%><br>
    Company <%=userinfo.company%><br>
    Cool <%=userinfo.awesome%>
</script>
```

Process the template by calling a javascript function with the id of the template, and optional data to pass in

```
$("#output").html($.template("my_template"));
```

Pass in an optional object parameter that provides data to be used within the template. Below is a sample that could display user information based on a user object

```
<script type='text/html' id='user_info'>
   Name <%=userinfo.name%><br>
   Company <%=userinfo.company%><br>
   Cool <%=userinfo.awesome%>
</script>

<script type='text/javascript'>
   var user={
       name:"Joe Programmer",
       company:"appMobi",
       awesome:"of course"
   }
$("#template_content").html($.template("user_info",{userinfo:user}));
</script>
```

jq.drawer

A pull widget like the nofication bar on phones

Usage

Create the container div and give it an id

```
<div id="drawer" style="position:relative;top:-
370px;height:320px;background:black;z-index:0">
   This is some content<br><br>
   This is some content<br><br>
   This is some content<br><br>
   This is some content<br><br>
</div>
```

Create the handle class inside the above div that the users will "pull" and assign the class 'drawer_handle'

```
<div id="drawer" style="position:relative;top:-
370px;height:320px;background:black;z-index:0">
   This is some content<br><br>
   This is some content<br><br>
   This is some content<br><br>
   This is some content<br><br>

   <div class="drawer_handle"
style="height:30px;width:100%;background:orange;position:absolute;bottom:0p
x;text-align:center;line-height:30px;font-size:14px;color:black;">PULL
DOWN</div>
</div>
```

Create the drawer with a direction of either up or down

```
var drawer = new jq.drawer("drawer",{direction:"down"});
```

jq.alphatable

A CSS3 Alphabetical/scrolling table. Creates a table that scrolls, but has the alphabetical index for users to jump around with

Usage

To use jq.alphatable, the jq.scroller library MUST be included, and a jq.scroller object for the list (See jq.scroller) must be created.

Create the list, broken up by divs/spans/anchors for each alphabet letter

```
<div id="contacts_A">
<li>Joe Anderson</li>
</div>
<div id="contacts_B">
<li>Joe Bob</li>
</div>
```

Call the javascript function to create the object. The first paramter is the ID for the scroller, the second is the scroller object. The third is optional config parameters.

```
var alphaTable =
$("#contentDIV").alphatable(scroller,{prefix:"contacts_",listCssClass:"cssCl
assName"});
```

There are two optional parameters

```
var options{
   prefix:"contacts_", //prefix for your divs
   listCssClass:"listTable" //CSS class name to style the alphabet list.
You can position it, set the background color, etc.
}
```

jq.selectBox

Replacement for HTML select boxes on Android

Usage

Make sure to wrap current select boxes in a span tag

```
<span><select id="myid"><option>1</option></select></span>
```

On the document.load or appMobi.device.ready listener, an object must be created, then call getOldSelects on the elements (div/spans/document) to select.

```
$.selectBox.getOldSelects("selectTest");
```

jq.passwordBox

Replacement for HTML password boxes on Android

Usasge

Make sure to wrap current password boxes in a span tag

```
<span><input type="password" id="myid"></span>
```

On the document.load or appMobi.device.ready listener, an object must be created, then call getOldPasswords on the elements (div/spans/document) to replace the password boxes in.

```
var pwFixer = new $.passwordBox();
pwFixer.getOldPasswords("selectTest");
```

jq.swipeListener

Detect swipe events on an element

Usage

Create an html elemenet to detect the swipe on. This could also be the whole document.

Call the javascript function to listen for a swipe event

```
var swipe=$(document).swipeListner(); //string or element to listen on
```

There are additional configuration options that are passed in as an object parameter

```
var options={
   vthreshold:50, //vertical pixel threshold
   hthreshold:50, //horizontal pixel threshold
   callback:null //callback function to execute.  It takes one parameter
that is an object of the swipe directions
{up:true,down:false,left:true,right:false}
}

var swipeListener = $("#mydiv").swipeListener(options);
var swipeListener =
$(document).swipeListener({vthreshold:30,hthreshold:50,callBack:function(di
r){console.log(dir)}});
```

View the complete jqMobi Documentation at:
http://www.jqmobi.com/documentation.php

View this Cheat Sheet online at:
http://www.appmobi.com/documentation/index.php?DOC=JQMOBI_CHEAT