

# PhoneGap & jQuery Mobile

## Lesson 3

# Thomas Mak

`makzan@42games.net`

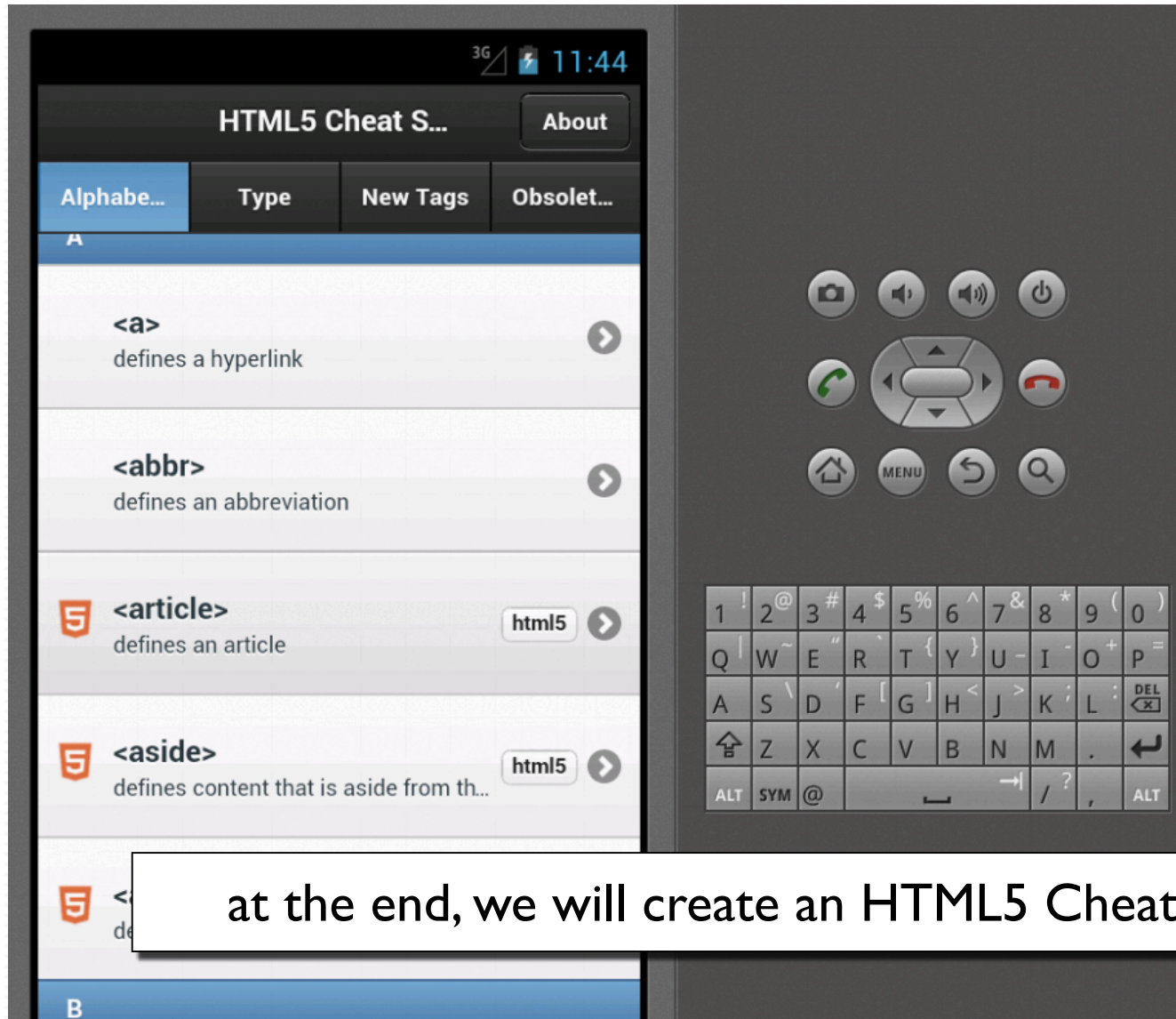
# Source Codes

<https://github.com/makzan/PhoneGap-Course-Examples>

# jQuery Mobile

- List View
- Creating an HTML5 cheat sheet App

# HTML5 Cheat Sheet App



# Preparing jQuery Mobile

- First, we need to prepare several jQuery Mobile files
  - Latest jQuery.js file (`jquery-1.7.2.min.js`)
  - jQuery Mobile js file (`jquery.mobile-1.1.0.js`)
  - jQuery Mobile CSS file (`jquery.mobile-1.1.0.css`)
  - jQuery Mobile images files (`images/`)
- These files are available on <http://code.jquery.com>

# Preparing jQuery Mobile

- Let's prepare two more empty files.
  - **Stylesheet** (`app.css`)
  - **JavaScript** (`app.js`)
- Leave it empty now and we will add things into these files.

# Basic List View

In this section, we will create a basic list view with HTML tags.



# Basic List View

HTML5 Cheat Sheet		About
a		
abbr		
article		
aside		
audio		
b		
base		
bdo		
blockquote		
br		
body		
button		
canvas		
caption		
center		
cite		

the result after this section.

# Basic List View

```
<!DOCTYPE html>
<html lang='en'>
  <head>
    <meta charset='utf-8'>
    <meta content='width=device-width, minimum-scale=1, maximum-scale=1'
name='viewport'>
    <title>HTML5 Cheat Sheet</title>

    <link href='jquery.mobile-1.1.0.css' rel='stylesheet'>
    <link href='app.css' rel='stylesheet'>

    <script src='jquery-1.7.2.min.js'></script>
    <script src='jquery.mobile-1.1.0.js'></script>
    <script src='app.js'></script>
  </head>
  <body>

  </body>
</html>
```

the index.html starting point.

# Basic List View

```
<body>
  <div data-role='page' id='news'>
    ...
  </div>
  <div data-role='page' id='about'>
    ...
  </div>
</body>
```

two pages in the body in index.html

# Basic List View

```
<div data-role='page' id='main'>
  <div data-position='fixed' data-role='header'>
    <h1>HTML5 Cheat Sheet</h1>
    <a class='ui-btn-right' data-role='button' href='#about'>About</a>
  </div>
  <section data-role='content'>
    <ul data-role='listview' id='tags'>
      <li>a</li>
      <li>abbr</li>
      <li>article</li>
      <li>aside</li>
      <li>audio</li>
      <li>b</li>
      <li>base</li>
    </ul>
  </section>
</div>
```

the first page with a list of HTML tags.

# Basic List View

```
<div data-role='page' id='main'>
  <div data-position='fixed' data-role='header'>
    <h1>HTML5 Cheat Sheet</h1>
    <a class='ui-btn-right' data-role='button' href='#about'>About</a>
  </div>
  <section data-role='content'>
    <ul data-role='listview' id='tags'>
      <li>a</li>
      <li>abbr</li>
      <li>article</li>
      <li>aside</li>
      <li>audio</li>
      <li>b</li>
      <li>base</li>
    </ul>
  </section>
</div>
```

**<ul> tag with data-role='listview'.**

# Basic List View

```
<div data-role='page' id='main'>
  <div data-position='fixed' data-role='header'>
    <h1>HTML5 Cheat Sheet</h1>
    <a class='ui-btn-right' data-role='button' href='#about'>About</a>
  </div>
  <section data-role='content'>
    <ul data-role='listview' id='tags'>
      <li>a</li>
      <li>abbr</li>
      <li>article</li>
      <li>aside</li>
      <li>audio</li>
      <li>b</li>
      <li>base</li>
    </ul>
  </section>
</div>
```

normal <li> tag for list item.

# Basic List View

```
<div data-role='page' id='about'>
  <div data-role='header'>
    <h2>About</h2>
    <a data-icon='arrow-l' data-rel='back'>HTML5 Cheat Sheet</a>
  </div>
  <div data-role='content'>
    <p>
      This is a cheatsheet app to quickly reference specific tags and its
      explanation. Every row links to the detail explanation from Mozilla
      Developer Network.
    </p>
    <p>
      email :
      <a href='mailto:twentyfive@mz-lab.com'>html5-cheatsheet@mz-lab.com</
a>
    </p>
  </div>
</div>
```

the second page with app information.

# Running jQuery Mobile in Mobile with PhoneGap

In this section, we will try to put our app up and running in the Android (virtual) device.

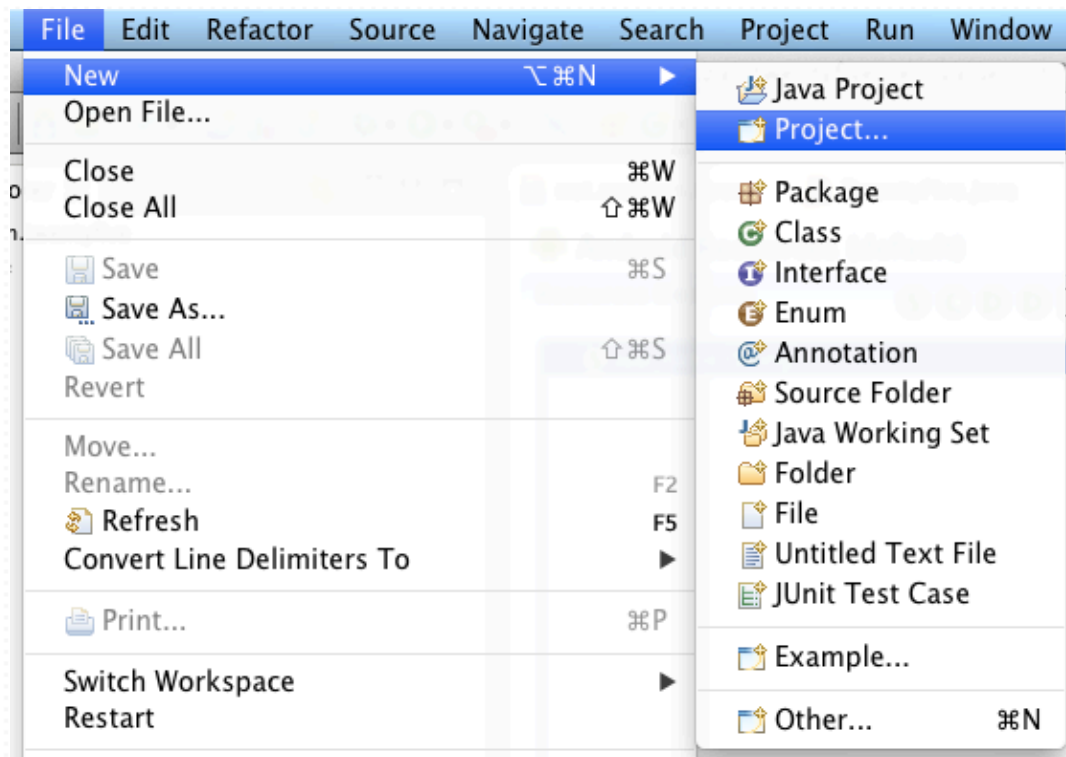


# Running jQuery Mobile in Mobile with PhoneGap

Basically, we will:

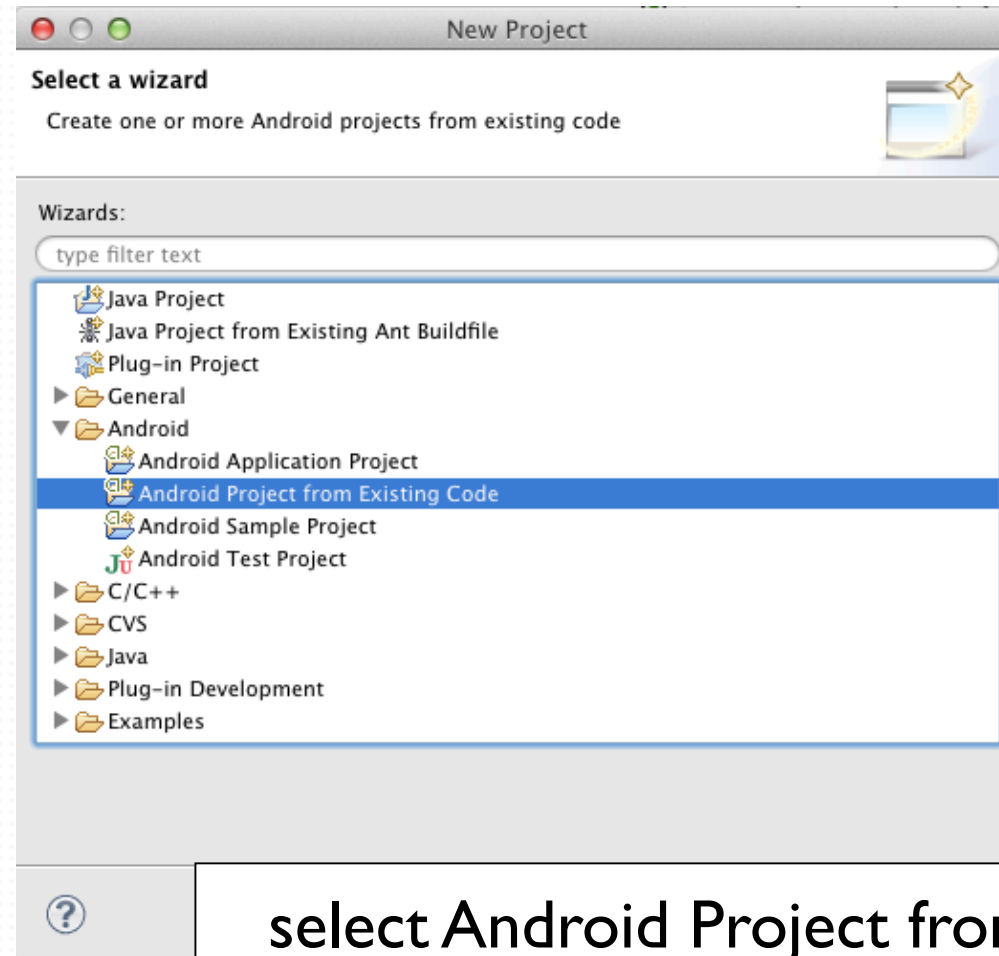
- Clone the PhoneGap Android example proj.
- Rename all occurrences from 'example' to our new app name.
- Place the files inside assets/www/ folder.
- Put it on Android device and run it.

# Running jQuery Mobile in Mobile with PhoneGap

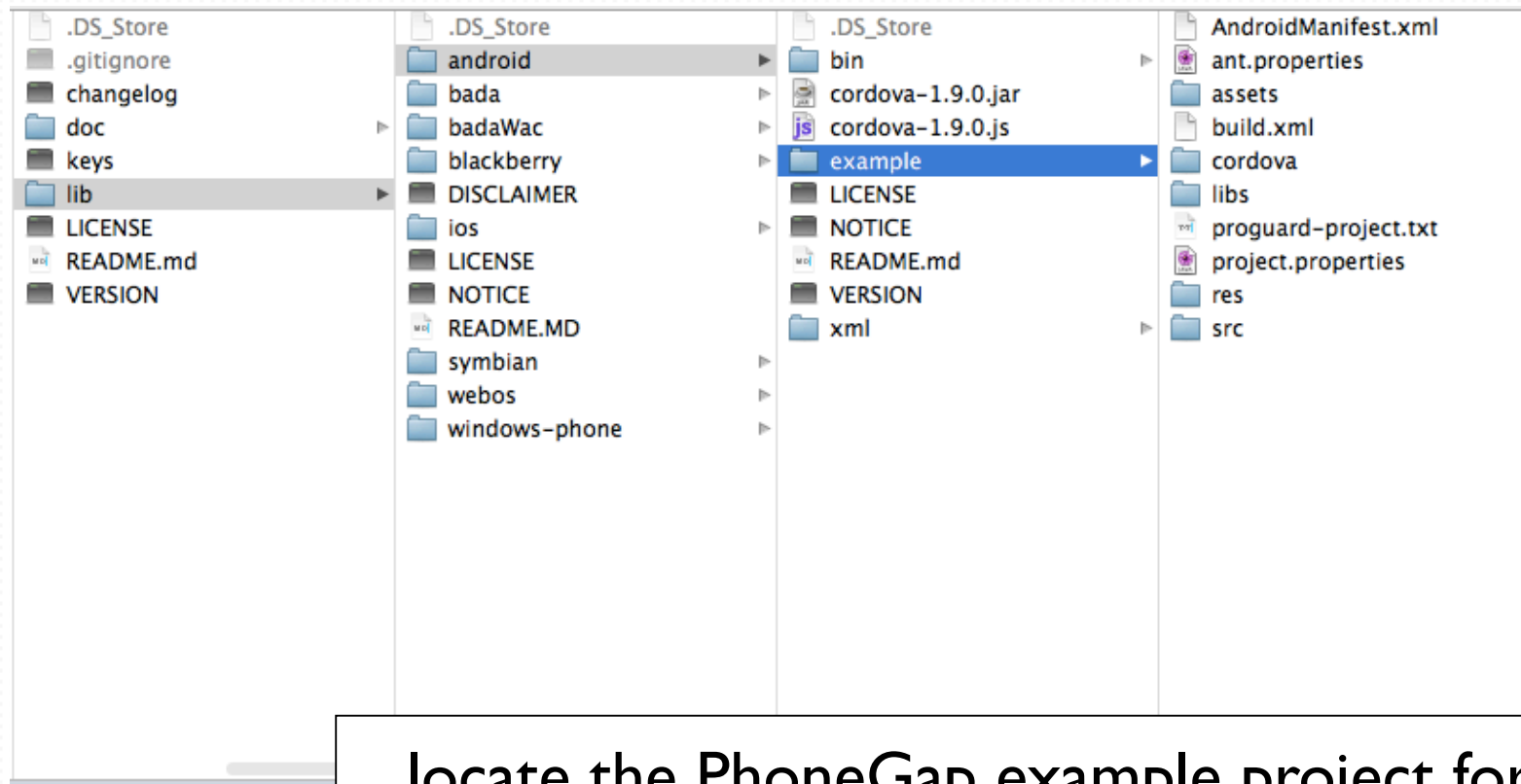


create a new project in Eclipse.

# Running jQuery Mobile in Mobile with PhoneGap

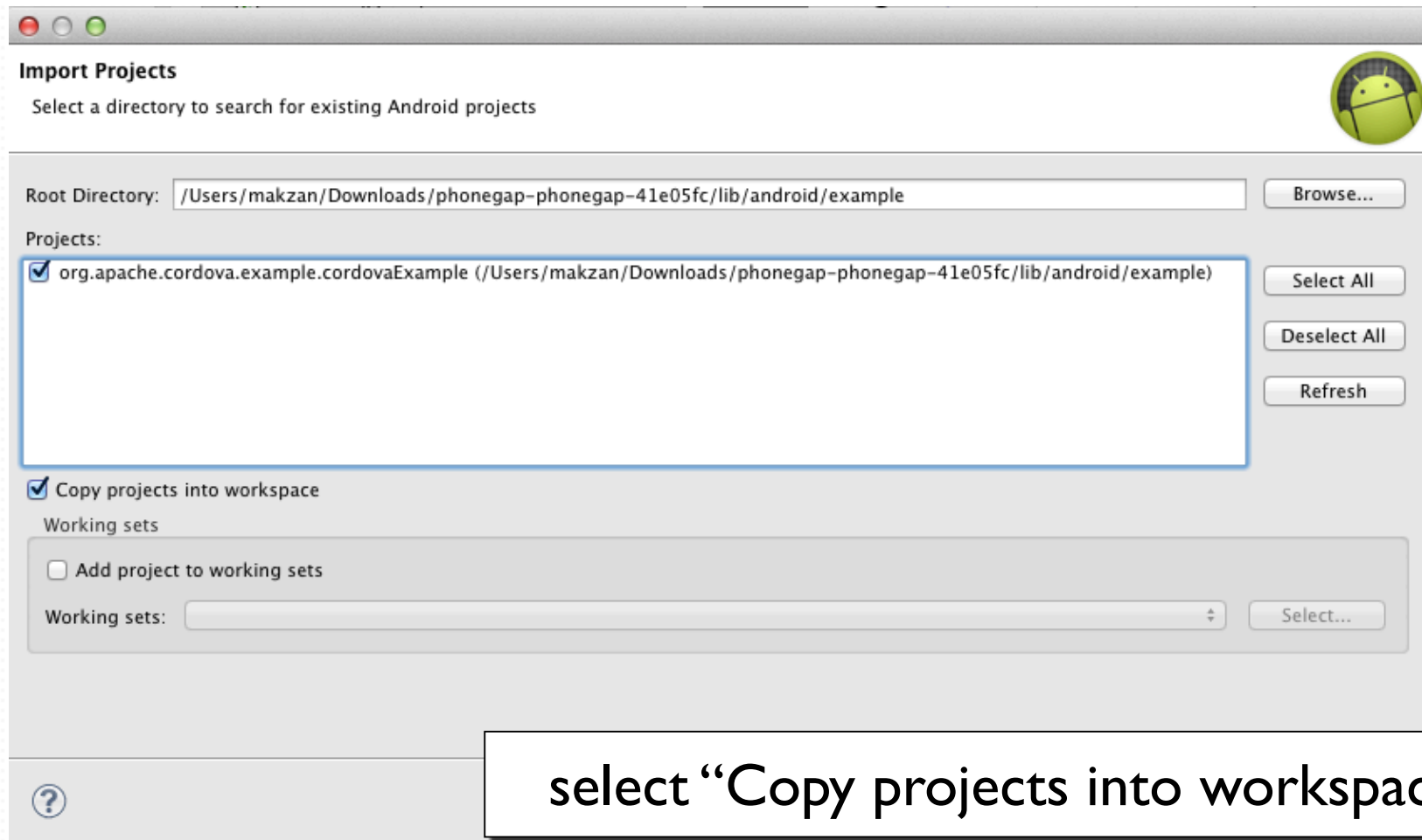


# Running jQuery Mobile in Mobile with PhoneGap

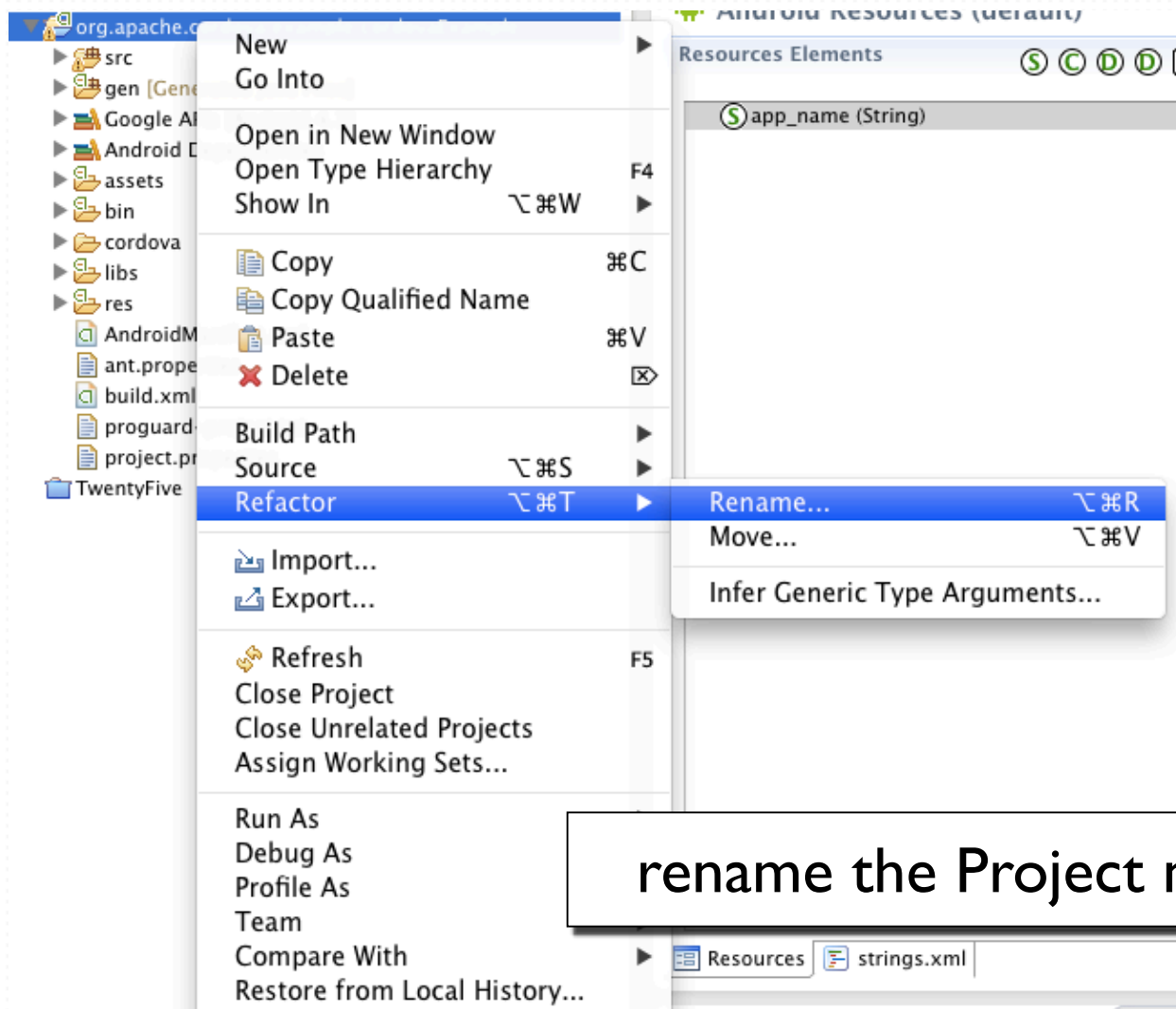


locate the PhoneGap example project for Android  
It's in <phonegap folder>/lib/android/example

# Running jQuery Mobile in Mobile with PhoneGap

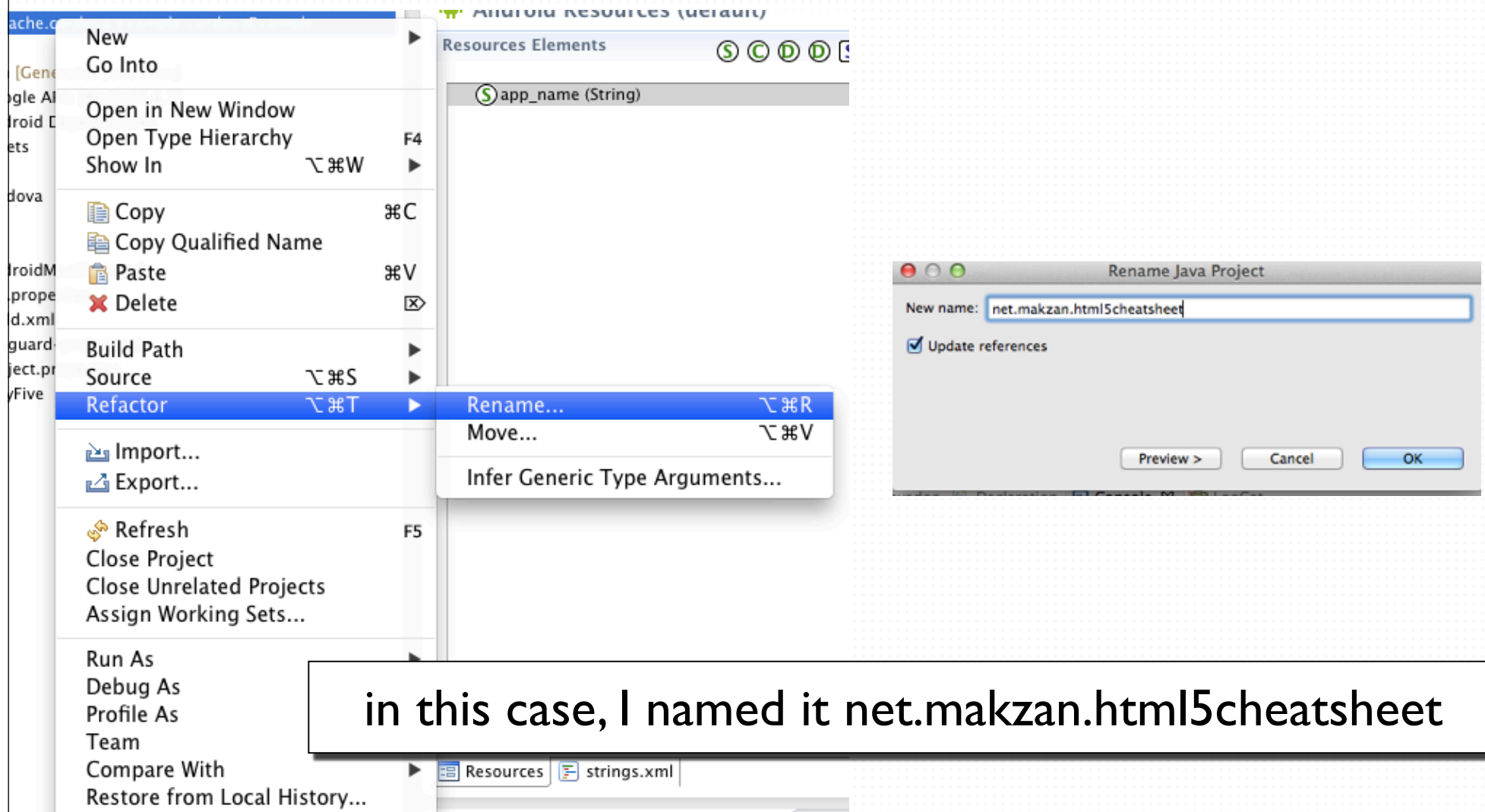


# Running jQuery Mobile in Mobile with PhoneGap

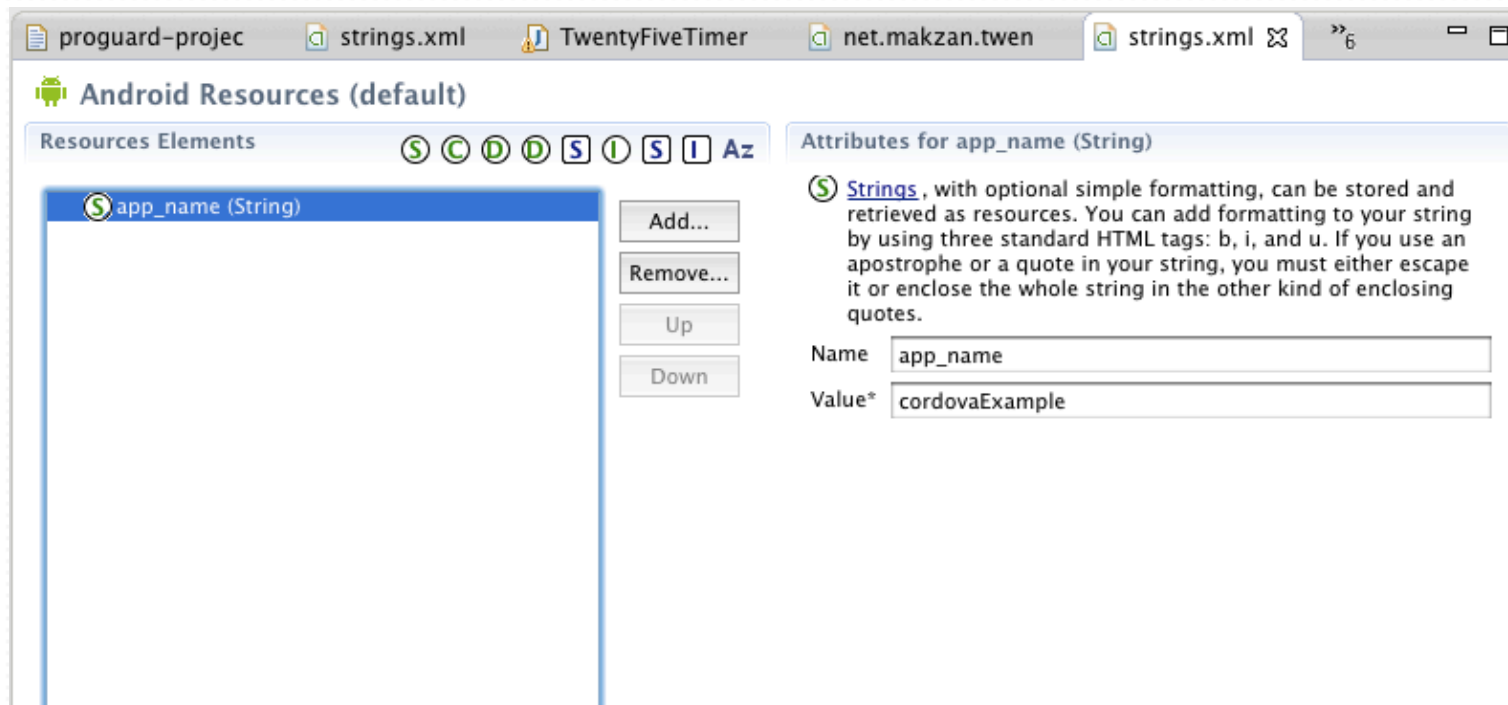


rename the Project name to the new name.

# Running jQuery Mobile in Mobile with PhoneGap



# Running jQuery Mobile in Mobile with PhoneGap

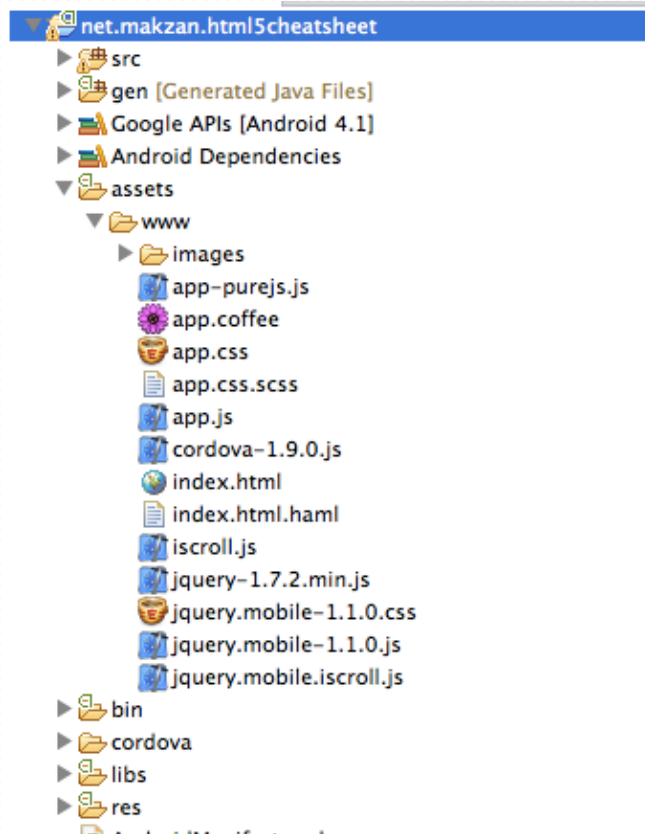


one more place.

open strings.xml and change the app\_name to the name which people will read it as app name.

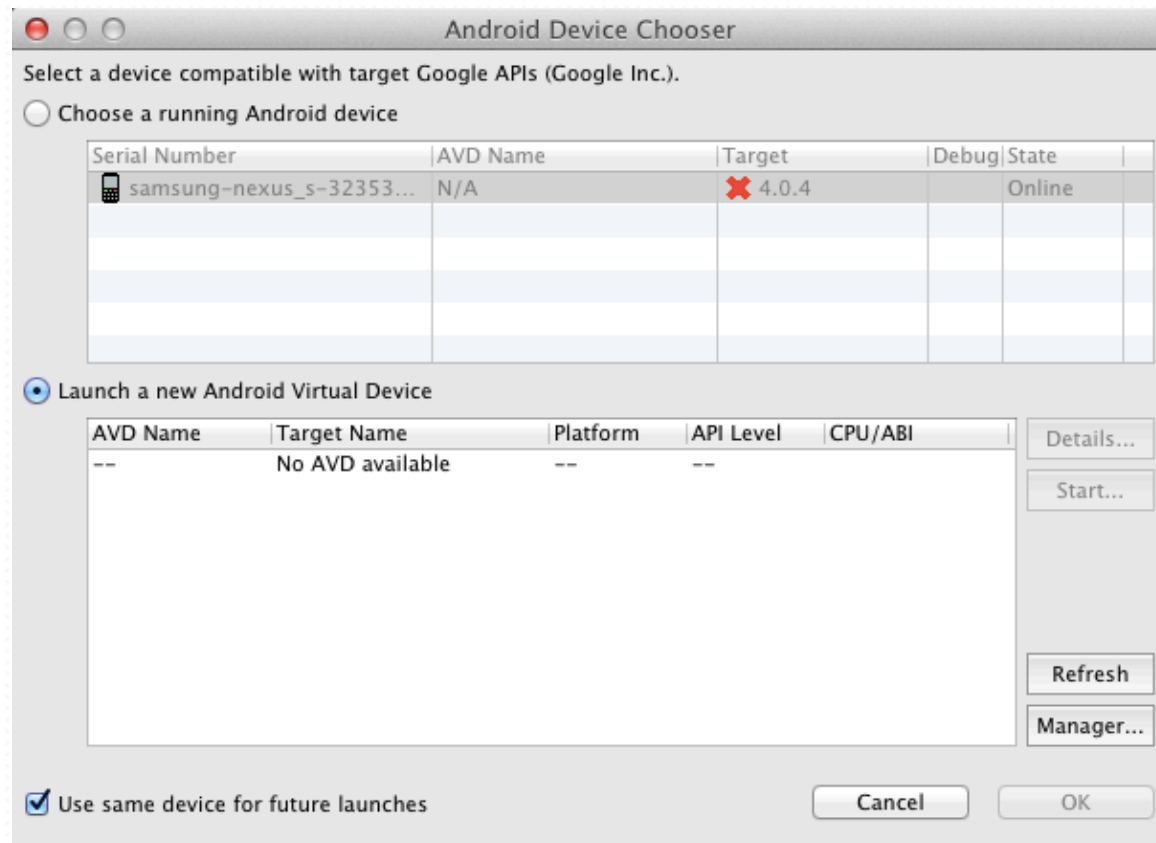


# Running jQuery Mobile in Mobile with PhoneGap



finally, move our project files into assets/www/ folder

# Running jQuery Mobile in Mobile with PhoneGap



done. and run it on Android virtual device.

# Making the list dynamic

In this section, we will use javascript to update the list view.

# Making the list dynamic

HTML5 Cheat Sheet		About
<a>		
<abbr>		
<article>		
<aside>		
<audio>		
<b>		
<base>		
<bdo>		
<blockquote>		
<body>		
<button>		
<canvas>		
<caption>		
<center>		

the result after this section.

# Making the list dynamic

```
// Tag class
var Tag = (function() {
  /*
   name - string, tag name
   explain - string, one sentence description
   html4 - bool, available in html4
   html5 - bool, not obsoleted in html5
  */
  function Tag(name, explain, html4, html5) {
    this.name = name;
    this.explain = explain != null ? explain : '';
    this.html4 = html4 != null ? html4 : 'true';
    this.html5 = html5 != null ? html5 : 'true';
  }

  Tag.prototype.toString = function() {
    return "<" + this.name + ">";
  }

  return Tag;
})(); // end of Tag class
```

**we need a Tag class to store tag data.**

# Making the list dynamic

```
<div data-role='page' id='main'>
  <div data-position='fixed' data-role='header'>
    <h1>HTML5 Cheat Sheet</h1>
    <a class='ui-btn-right' data-role='button' href='#about'>About</a>
  </div>
  <section data-role='content'>
    <ul data-role='listview' id='tags'></ul>
  </section>
</div>
```

in the `<ul>` list view, we delete all list items. we are going to create them in javascript

# Making the list dynamic

```
(function() {  
    // our code goes here.  
}).call(this);
```

in our app.js, first we have an anonymous function to define a local scope.

# Making the list dynamic

```
// Tag class
var Tag = (function() {
  /*
  name - string, tag name
  explain - string, one sentence description
  html4 - bool, available in html4
  html5 - bool, not obsoleted in html5
  */
  function Tag(name, explain, html4, html5) {
    this.name = name;
    this.explain = explain != null ? explain : '';
    this.html4 = html4 != null ? html4 : 'true';
    this.html5 = html5 != null ? html5 : 'true';
  }

  Tag.prototype.toString = function() {
    return "<" + this.name + ">";
  }

  return Tag;
})(); // end of Tag class
```

defining our Tag class.



# Making the list dynamic

```
// Tag class
var Tag = (function() {
  /*
  name - string, tag name
  explain - string, one sentence description
  html4 - bool, available in html4
  html5 - bool, not obsoleted in html5
  */
  function Tag(name, explain, html4, html5) {
    this.name = name;
    this.explain = explain != null ? explain : '';
    this.html4 = html4 != null ? html4 : 'true';
    this.html5 = html5 != null ? html5 : 'true';
  }

  Tag.prototype.toString = function() {
    return "<" + this.name + ">";
  }

  return Tag;
})(); // end of Tag class
```

constructor with four parameters.

# Making the list dynamic

```
// Tag class
var Tag = (function() {
  /*
   name - string, tag name
   explain - string, one sentence description
   html4 - bool, available in html4
   html5 - bool, not obsoleted in html5
  */
  function Tag(name, explain, html4, html5) {
    this.name = name;
    this.explain = explain != null ? explain : '';
    this.html4 = html4 != null ? html4 : 'true';
    this.html5 = html5 != null ? html5 : 'true';
  }

  Tag.prototype.toString = function() {
    return "<"; + this.name + ">";
  }

  return Tag;
})(); // end of Tag class
```

when we convert this Tag to string,  
we make it in format <name>.

# Making the list dynamic

```
var tags = [];  
  
tags.push(new Tag('br', 'inserts a single line break'));  
tags.push(new Tag('a', 'defines a hyperlink'));  
tags.push(new Tag('abbr', 'defines an abbreviation'));  
tags.push(new Tag('article', 'defines an article', false, true));  
tags.push(new Tag('aside', 'defines content that is aside from the  
page', false, true));  
tags.push(new Tag('audio', 'defines audio content', false, true));  
tags.push(new Tag('b', 'defines bold text'));  
tags.push(new Tag('base', 'defines a base URL for all the links in the  
page'));  
tags.push(new Tag('bdo', 'defines direction of the text'));  
...
```

now we have Tag data. we can create  
our list of HTML tag data.

# Making the list dynamic

```
var tags = [];  
  
tags.push(new Tag('br', 'inserts a single line break'));  
tags.push(new Tag('a', 'defines a hyperlink'));  
tags.push(new Tag('abbr', 'defines an abbreviation'));  
tags.push(new Tag('article', 'defines an article', false, true));  
tags.push(new Tag('aside', 'defines content that is aside from the  
page', false, true));  
tags.push(new Tag('audio', 'defines audio content', false, true));  
tags.push(new Tag('b', 'defines bold text'));  
tags.push(new Tag('base', 'defines a base URL for all the links in the  
page'));  
tags.push(new Tag('bdo', 'defines direction of the text'));  
...
```

the 3rd parameter means its available in HTML4.

the 4th parameter means its available in HTML5 and not obsolete.

# Making the list dynamic

```
$(function(){  
  
    //append li to the ul#tags  
    for (var i=0, len=tags.length; i<len; i++) {  
        tag = tags[i];  
        $('#tags').append("<li>" + tag + "</li>");  
    }  
  
    // refresh the tags listview programatically.  
    $('#tags').listview('refresh');  
  
})
```

in the jQuery ready function, we construct the <li> list from our tags data.

# Making the list dynamic

```
$(function(){  
  
    //append li to the ul#tags  
    for (var i=0, len=tags.length; i<len; i++) {  
        tag = tags[i];  
        $('#tags').append("<li>" + tag + "</li>");  
    }  
  
    // refresh the tags listview programatically.  
    $('#tags').listview('refresh');  
  
})
```

we will keep adding more tags in this line.

# Adding subtitle to list item

In this section, we will add detail text in every list item.

# Adding subtitle to list item

HTML5 Cheat Sheet		About
<b>&lt;br&gt;</b>	inserts a single line break	
<b>&lt;a&gt;</b>	defines a hyperlink	
<b>&lt;abbr&gt;</b>	defines an abbreviation	
<b>&lt;article&gt;</b>	defines an article	
<b>&lt;aside&gt;</b>	defines content that is aside from the page	
<b>&lt;audio&gt;</b>	defines audio content	
<b>&lt;b&gt;</b>	defines bold text	
<b>&lt;base&gt;</b>	defines a base URL for all the links in the page	

the result after this section.



# Adding subtitle to list item

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];  
    $('#tags').append("<li><h1>" + tag + "</h1><p> " + tag.explain + "</p></li>");  
}
```

we can define list title with h1-h6.

we can add subtitle paragraph under the title with <p>.

# Filtering new and obsolete tag

In this section, we will add indicator to distinguish new tags and obsolete tags.

# Filtering new and obsolete tag

HTML5 Cheat Sheet		About
	defines a button	
<b>&lt;canvas&gt;</b>	defines a graphics drawing area	html5
<b>&lt;caption&gt;</b>	defines a table caption	
<b>&lt;center&gt;</b>	center align the content	obsolete
<b>&lt;cite&gt;</b>	defines a citation	
<b>&lt;code&gt;</b>	defines computer code text	
<b>&lt;col&gt;</b>	defines attributes for table columns	
<b>&lt;colgroup&gt;</b>	defines groups of table columns	
<b>&lt;command&gt;</b>		

the result after this section.

# Filtering new and obsolete tag

```
.obsolete {  
  color: #888;  
  opacity: 0.5;  
}
```

in the app.css, we define .obsolete class.

# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
  tag = tags[i];  
  
  var obsolete = '';  
  var obsoleteClass='';  
  if (!tag.html5)  
  {  
    obsolete = "<p class='ui-li-count'>obsolete</p>";  
    obsoleteClass = 'obsolete';  
  }  
  
  var newTag = '';  
  if (!tag.html4 && tag.html5)  
    newTag = "<p class='ui-li-count'>html5</p>";  
  
  $('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</li>");  
}
```

we change list creating logic in the app.js

# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];  
  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
  
    var newTag = '';  
    if (!tag.html4 && tag.html5)  
        newTag = "<p class='ui-li-count'>html5</p>";  
  
    $('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</li>");  
}
```

we will apply the obsolete class to obsoleted list item

# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];  
  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
  
    var newTag = '';  
    if (!tag.html4 && tag.html5)  
        newTag = "<p class='ui-li-count'>html5</p>";  
  
    $('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</li>");  
}
```

we will add two tags after the subtitle text.

# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];
```

```
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }
```

```
    var newTag = '';  
    if (!tag.html4 && tag.html5)  
        newTag = "<p class='ui-li-count'>html5</p>";
```

```
    $('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " +  
    }
```

so, we need to prepare the obsolete class, obsolete and newTag string.



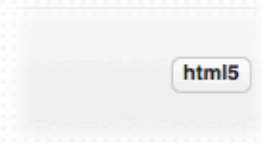
# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];  
  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
}
```

```
var newTag = '';  
if (!tag.html4 && tag.html5)  
    newTag = "<p class='ui-li-count'>html5</p>";
```

```
$('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " +  
    }  
}
```

so, we need to prepare the obsolete class, obsolete and newTag string.



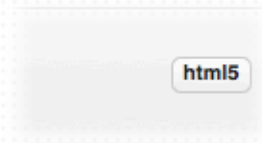
# Filtering new and obsolete tag

```
for (var i=0, len=tags.length; i<len; i++) {  
    tag = tags[i];  
  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
}
```

```
var newTag = '';  
if (!tag.html4 && tag.html5)  
    newTag = "<p class='ui-li-count'>html5</p>";
```

```
$('#tags').append("<li class='" + obsoleteClass + "'><h1>" + tag + "</h1><p> " +  
    }  
}
```

so, we need to prepare the obsolete class, obsolete and newTag string.



# Adding link to list item

In this section, we will add a link to the tag detail explanation in Mozilla Developer Network.

# Adding link to list item

HTML5 Cheat Sheet		About
<b>&lt;br&gt;</b>	inserts a single line break	>
<b>&lt;a&gt;</b>	defines a hyperlink	>
<b>&lt;abbr&gt;</b>	defines an abbreviation	>
<b>&lt;article&gt;</b>	defines an article	html5 >
<b>&lt;aside&gt;</b>	defines content that is aside from the page	html5 >
<b>&lt;audio&gt;</b>	defines audio content	html5 >
<b>&lt;b&gt;</b>	defines bold text	>
<b>&lt;base&gt;</b>		>

the result after this section.

# Adding link to list item

```
$('#tags').append("<li class='" + obsoleteClass + "'><a href='http://  
developer.mozilla.org/en/HTML/Element/' + tag.name + "'><h1>" + tag +  
</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</a></li>");
```

it's simply an anchor link.

please note that we should put all content inside the anchor.

# Adding link to list item

```
$('#tags').append("<li class='" + obsoleteClass + "'><a href='http://  
developer.mozilla.org/en/HTML/Element/' + tag.name + "'><h1>" + tag +  
</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</a></li>");
```

note: it is not valid to put block tag until HTML5 spec.

# Adding thumbnail

In this section, we will add a thumbnail to indicator if the tag is new in HTML5 or not.

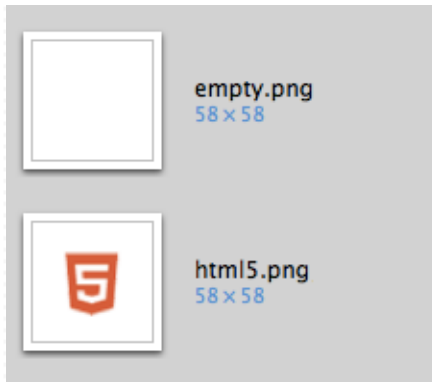
# Adding thumbnail

HTML5 Cheat Sheet		About
 	inserts a single line break	>
<a>	defines a hyperlink	>
<abbr>	defines an abbreviation	>
 <article>	defines an article	html5 >
 <aside>	defines content that is aside from the page	html5 >
 <audio>	defines audio content	>
<b>	defines bold text	>

the result after this section.



# Adding thumbnail



let's prepare two image files to indicate if it is new tag.

# Adding thumbnail

```
var newTag = '';  
var iconFile = 'empty.png';  
if (!tag.html4 && tag.html5)  
{  
    newTag = "<p class='ui-li-count'>html5</p>";  
    iconFile = 'html5.png';  
}
```

then we check if it is empty.png or html5.png

# Adding thumbnail

```
$('#tags').append("<li class='" + obsoleteClass + "'><a href='http://  
developer.mozilla.org/en/HTML/Element/' + tag.name + "'><img  
src='images/' + iconFile + "' class='ui-li-icon'><h1>" + tag + "</h1><p>  
" + tag.explain + "</p>" + obsolete + newTag + "</a></li>");
```

still the list item construction.  
we can add an image tag as first child to make it thumbnail.

# Adding thumbnail




```
.ui-listview .ui-li-icon {  
  left: 0;  
}
```

the thumbnail may have position offset on left.  
we need to reset it to 0 in app.css.

# Dividing the list

In this section, we will divide the tags by alphabet.

# Dividing the list

HTML5 Cheat Sheet		About
A		
<code>&lt;a&gt;</code>	defines a hyperlink	>
<code>&lt;abbr&gt;</code>	defines an abbreviation	>
 <code>&lt;article&gt;</code>	defines an article	html5 >
 <code>&lt;aside&gt;</code>	defines content that is aside from the page	html5 >
 <code>&lt;audio&gt;</code>	defines audio content	html5 >
B		
<code>&lt;b&gt;</code>	defines bold text	
<code>&lt;base&gt;</code>		

the result after this section.

# Dividing the list

```
// before adding dividers, make sure we have sort the tags
tags.sort(function(a, b) {
  if (a.name < b.name)
    return -1;
  if (a.name > b.name)
    return 1;
  return 0;
});
```

first, we sort the tags array to ensure they are sorted alphabetically.

# Dividing the list

```
var currentDivider = '';
```

initialize the current divider with empty string.



# Dividing the list

```
//append li to the ul#tags
for (var i=0, len=tags.length; i<len; i++) {
    tag = tags[i];

    // check if we need to add divider
    var firstCharacter = tag.name.substr(0, 1);

    if (firstCharacter != currentDivider)
    {
        $('#tags').append("<li data-role='list-divider'>" +
firstCharacter.toUpperCase() + "</li>");
        currentDivider = firstCharacter;
    }

    ...
}
```

we check the different of first character.  
if it is different from the previous one, we create a divider.

# Dividing the list

```
//append li to the ul#tags
for (var i=0, len=tags.length; i<len; i++) {
    tag = tags[i];

    // check if we need to add divider
    var firstCharacter = tag.name.substr(0, 1);

    if (firstCharacter != currentDivider)
    {
        $('#tags').append("<li data-role='list-divider'" +
firstCharacter.toUpperCase() + "</li>");
        currentDivider = firstCharacter;
    }

    ...
}
```

divider is a list item with data-role='list-divider'

# Dividing the list

```
//append li to the ul#tags
for (var i=0, len=tags.length; i<len; i++) {
    tag = tags[i];

    // check if we need to add divider
    var firstCharacter = tag.name.substr(0, 1);

    if (firstCharacter != currentDivider)
    {
        $('#tags').append("<li data-role='list-divider'>" +
firstCharacter.toUpperCase() + "</li>");
        currentDivider = firstCharacter;
    }

    ...
}
```

we use substr(0, 1) to get the first character of tag name.

# Nested List

In this section, we will create nested list to display the tag list into different sub-lists.

# Nested List

HTML5 Cheat Sheet	About	< Back	New Tags
All Tags	>		<b>&lt;article&gt;</b> defines an article html5 >
New Tags	>		<b>&lt;aside&gt;</b> defines content that is aside from the page html5 >
Obsolete Tags	>		<b>&lt;audio&gt;</b> defines audio content html5 >
			<b>&lt;canvas&gt;</b> defines a graphics drawing area html5 >
			<b>&lt;command&gt;</b> defines a command button html5 >
			<b>&lt;datalist&gt;</b> defines a dropdown list html5 >
			<b>&lt;em&gt;</b> define the result after this section.

# Nested List

```
<ul data=role='listview'>
  <li> Title
    <ul>
      <li>something</li>
      <li>something</li>
      <li>something</li>
    </ul>
  </li>
  <li> Title
    <ul>
      <li>something</li>
    </ul>
  </li>
  <li> Title
    <ul>
      <li>something</li>
    </ul>
  </li>
</ul>
```

nested list is simply, well, nested <ul><li>

# Nested List

HTML5 Cheat Sheet	About	< Back	New Tags
All Tags	>		<b>&lt;article&gt;</b> defines an articlehtml5 >
New Tags	>		
Obsolete Tags	>		<b>&lt;aside&gt;</b> defines content that is aside from the pagehtml5 >
			<b>&lt;audio&gt;</b> defines audio contenthtml5 >
			<b>&lt;canvas&gt;</b> defines a graphics drawing areahtml5 >
			<b>&lt;command&gt;</b> defines a command buttonhtml5 >
			<b>&lt;datalist&gt;</b> defines a dropdown listhtml5 >
			<b>&lt;math&gt;</b> defines an external interactive content or pluginhtml5 >

first level <li> automatically link to second level list.

# Nested List

```
// construct three ul sub-list, all, new tag, obsolete
$('#tags').append("<li>All Tags <ul id='all-tags'></ul></li>");
$('#tags').append("<li>New Tags <ul id='new-tags'></ul></li>");
$('#tags').append("<li>Obsolete Tags <ul id='obsolete-tags'></ul></li>");
```

before the for-loop, we create three list item for

- all tags
- new tags
- obsolete tags



# Nested List

```
// obsolete
var obsolete = '';
var obsoleteClass='';
if (!tag.html5)
{
    obsolete = "<p class='ui-li-count'>obsolete</p>";
    obsoleteClass = 'obsolete';

    $('#obsolete-tags').append("<li><a href='http://developer.mozilla.org/en/HTML/Element/' + tag.name + '>' + tag.name + obsolete + '</a></li>");
}
```

we append a list item in `#obsolete-tags` if it is obsoleted.

# Nested List

```
// new tag
var newTag = '';
var iconFile = 'empty.png';
if (!tag.html4 && tag.html5)
{
    newTag = "<p class='ui-li-count'>html5</p>";
    iconFile = 'html5.png';

    $('#new-tags').append("<li><a href='http://developer.mozilla.org/en/HTML/Element/' + tag.name + '><h1>" + tag + "</h1><p> " + tag.explain +
"</p>" + newTag + "</a></li>");
}
```

same here for new tag.

# Nested List

```
$('#all-tags').append("<li class='" + obsoleteC.....
```

for all tags, we use the same logic, but we need to change the element target to *#all-tags*.

# Nested List

```
$('#all-tags').append("<li data-role='list-divider'>"......
```

same here for the list divider, we append them to `#all-tags`.

# Nested List

```
<script src='jquery-1.7.2.min.js'></script>
<script>
$(document).bind("mobileinit", function(){
    $.mobile.page.prototype.options.addBackBtn = true;
    $.mobile.listview.prototype.options.headerTheme = "a";
});
</script>
<script src='jquery.mobile-1.1.0.js'></script>
```

we need some initial configuration for the jQuery Mobile here.  
we can listen to the mobileinit to configure it.

# Nested List

```
<script src='jquery-1.7.2.min.js'></script>
<script>
$(document).bind("mobileinit", function(){
    $.mobile.page.prototype.options.addBackBtn = true;
    $.mobile.listview.prototype.options.headerTheme = "a";
});
</script>
<script src='jquery.mobile-1.1.0.js'></script>
```

we can tell jQuery Mobile to add back button by default.

# Nested List

```
<script src='jquery-1.7.2.min.js'></script>
<script>
$(document).bind("mobileinit", function(){
    $.mobile.page.prototype.options.addBackBtn = true;
    $.mobile.listview.prototype.options.headerTheme = "a";
});
</script>
<script src='jquery.mobile-1.1.0.js'></script>
```




the nested style looks different, we can force it to use the black header by setting the header theme to 'a', which has black header.

# Navigation Bar

In this section, we will replace the nested list and create a navigation bar to show different list.



# Navigation Bar

HTML5 Cheat Sheet				About
Alphabetical	Type	New Tags	Obsolete Tags	
A				
<code>&lt;a&gt;</code>	defines a hyperlink			
<code>&lt;abbr&gt;</code>	defines an abbreviation			
 <code>&lt;article&gt;</code>	defines an article	html5		
 <code>&lt;aside&gt;</code>	defines content that is aside from the page	html5		
 <code>&lt;audio&gt;</code>	defines audio content	html5		
B				
<code>&lt;b&gt;</code>	defines bold text			

the result after this section.

# Navigation Bar

```
<div data-position='fixed' data-role='header'>
  <h1>HTML5 Cheat Sheet</h1>
  <a class='ui-btn-right' data-role='button' href='#about'>About</a>

  <nav data-role="navbar">
    <ul>
      <li><a id='sort-alpha' href="#" class="ui-btn-
active">Alphabetical</a></li>
      <li><a id='sort-type' href="#">Type</a></li>
      <li><a id='sort-new' href="#">New Tags</a></li>
      <li><a id='sort-obsolete' href="#">Obsolete Tags</a></li>
    </ul>
  </nav>
</div>
```

in the header section, we add a `<nav>` tag with `data-role='navbar'`

# Navigation Bar

```
Tag.prototype.toListItem = function() {  
    // obsolete  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
  
    var newTag = '';  
    var iconFile = 'empty.png';  
    if (!tag.html4 && tag.html5)  
    {  
        newTag = "<p class='ui-li-count'>html5</p>";  
        iconFile = 'html5.png';  
    }  
  
    return "<div class='ui-li'><div class='ui-li-count'>HTML/Element</div><div class='ui-li-content'>icon'><h1>Navigation Bar</h1></div></div></li>";  
}
```

**we are going to refactor our code a little bit.**

**first, we put the list string construction inside the Tag class.**

# Navigation Bar

```
Tag.prototype.toListItem = function() {  
    // obsolete  
    var obsolete = '';  
    var obsoleteClass='';  
    if (!tag.html5)  
    {  
        obsolete = "<p class='ui-li-count'>obsolete</p>";  
        obsoleteClass = 'obsolete';  
    }  
  
    var newTag = '';  
    var iconFile = 'empty.png';  
    if (!tag.html4 && tag.html5)  
    {  
        newTag = "<p class='ui-li-count'>html5</p>";  
        iconFile = 'html5.png';  
    }  
  
    return "<li class='" + obsoleteClass + "'><a href='http://developer.mozilla.org/en/  
HTML/Element/' + tag.name + "'><img src='images/' + iconFile + "' class='ui-li-  
icon'><h1>" + tag + "</h1><p> " + tag.explain + "</p>" + obsolete + newTag + "</a></  
li>";  
}
```

# Navigation Bar

```
$(function(){  
  
    $('#sort-alpha').click(function(){  
        refreshListWithAlphaSort();  
    })  
  
    $('#sort-type').click(function(){  
        refreshListWithTypeSort();  
    })  
  
    $('#sort-new').click(function(){  
        refreshListWithNewTags();  
    })  
  
    $('#sort-obsolete').click(function(){  
        refreshListWithObsoleteTags();  
    })  
  
    refreshListWithAlphaSort();  
})
```

in the jQuery ready function, we change to detect the click on the nav bar and construct our list.

# Navigation Bar

```
var refreshListWithNewTags = function () {  
    $('#tags').empty(); // empty the element first.  
  
    // append new tags  
    $('#tags').append("<li data-role='list-divider'> New Tags </li>");  
    for (var i=0, len=tags.length; i<len; i++) {  
        tag = tags[i];  
  
        if (!tag.html14 && tag.html15)  
            $('#tags').append(tag.toListItem());  
    }  
  
    // refresh the tags listview programatically.  
    $('#tags').listview('refresh');  
}
```

and make sure we create the #tags every time.

# Navigation Bar

How you implement the other three functions?

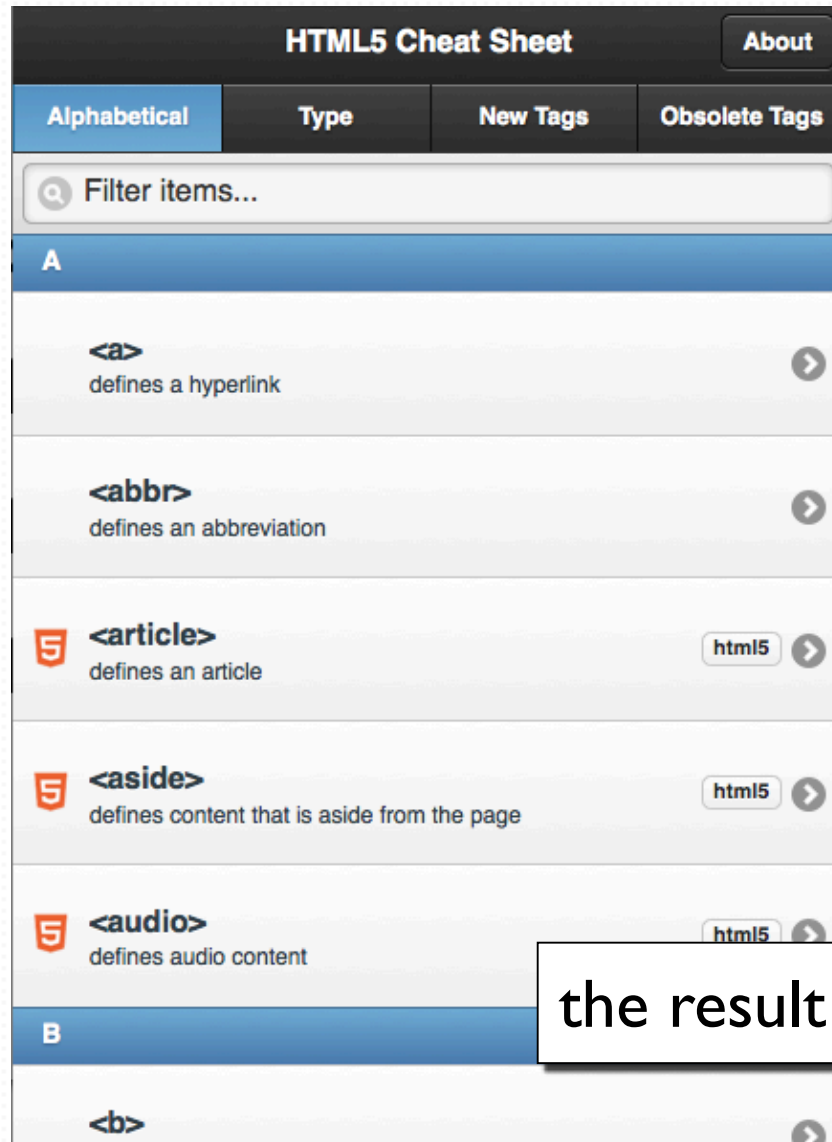
- refreshListWithAlphaSort
- refreshListWithTypeSort
- refreshListWithObsoleteTags

# Bonus: Filtering List

In this section, we will add a search bar on top of the list to let user filter the list item.



# Bonus: Filtering List



# Bonus: Filtering List

```
<section data-role='content'>  
  <ul data-role='listview' id='tags' data-filter='true'></ul>  
</section>
```

when we add data-filter='ture', the filter function is automagically done.

# Bonus: Filtering List

```
$("#tags").listview('option', 'filterCallback', yourFilterFunction)
```

in case you need custom filter function, you can use the above code to define it.