PhoneGap & jQuery Mobile

Lesson 2

Thomas Mak

makzan@42games.net

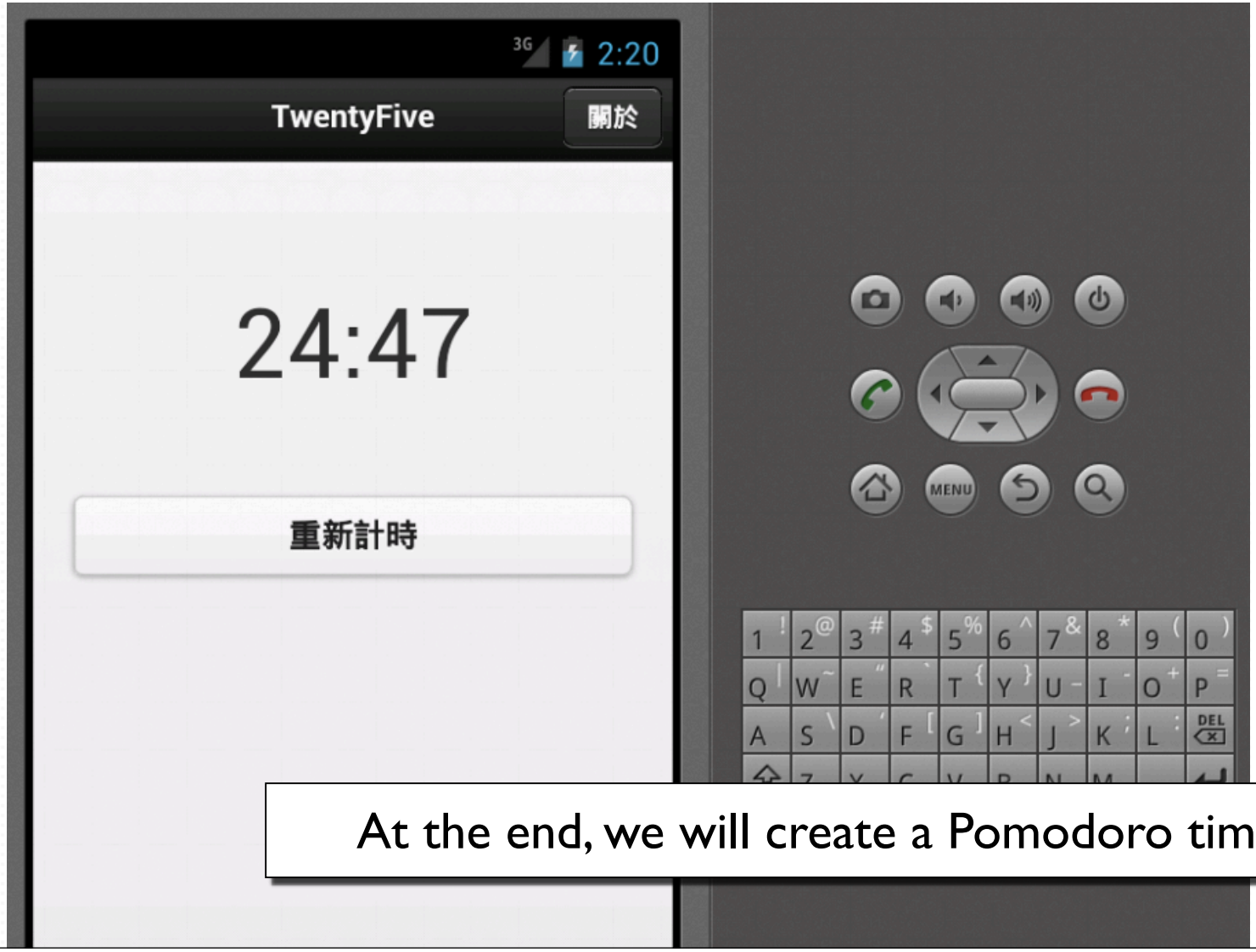
Source Codes

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

jQuery Mobile

- Pages
- Page Header
- Page Content
- Page Transition

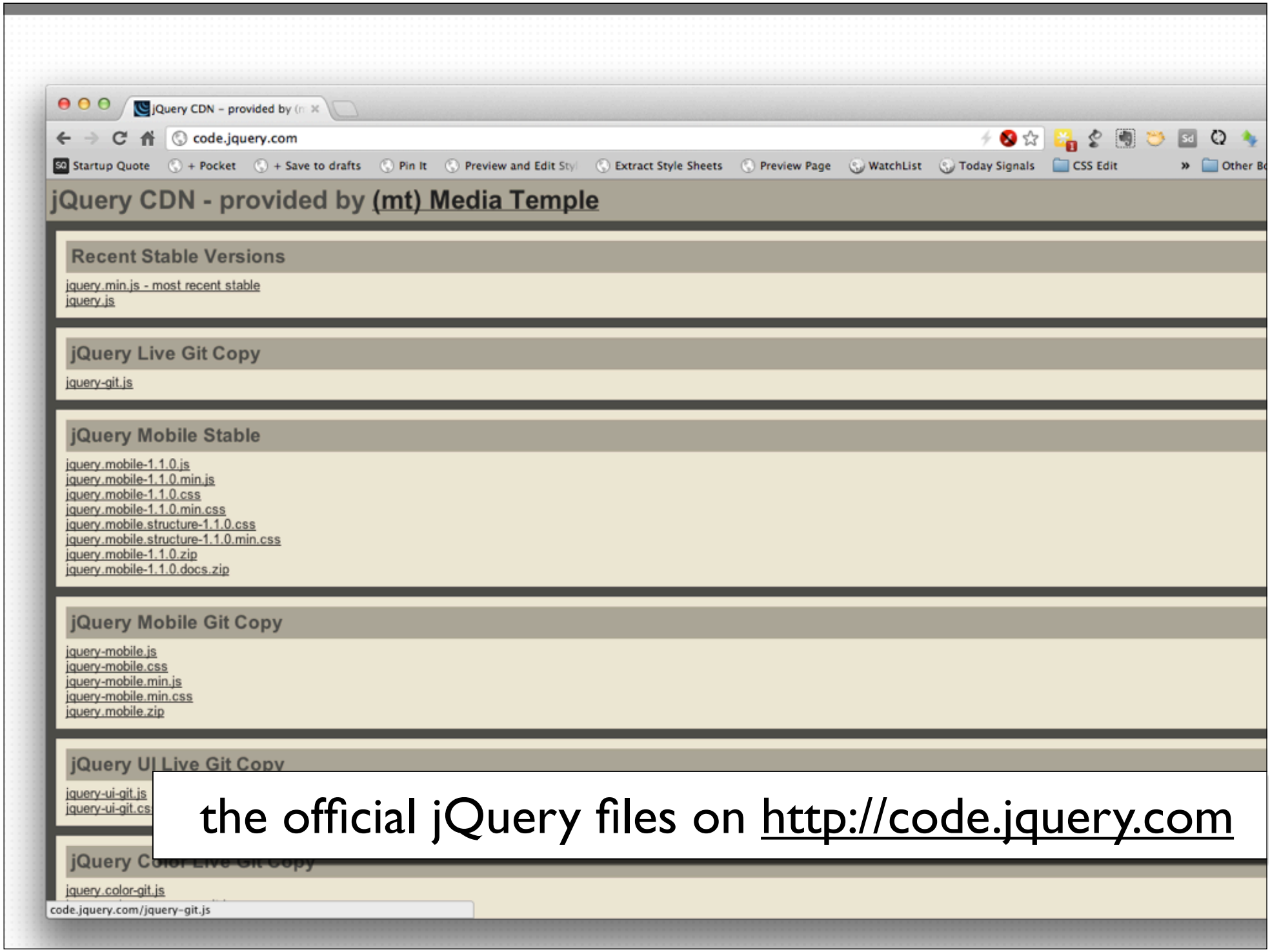
Our first jQuery Mobile page



At the end, we will create a Pomodoro timer app.

Our first jQuery Mobile page

- 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


jQuery CDN – provided by
code.jquery.com
Startup Quote
+ Pocket
+ Save to drafts
Pin It
Preview and Edit
Extract Style Sheets
Preview Page
WatchList
Today Signals
CSS Edit
jQuery CDN - provided by (mt) Media Temple
Recent Stable Versions
jquery.min.js - most recent stable
jquery.js
jQuery Live Git Copy
jquery-git.js
jQuery Mobile Stable
jquery.mobile-1.1.0.js
jquery.mobile-1.1.0.min.js
jquery.mobile-1.1.0.css
jquery.mobile-1.1.0.min.css
jquery.mobile.structure-1.1.0.css
jquery.mobile.structure-1.1.0.min.css
jquery.mobile-1.1.0.zip
jquery.mobile-1.1.0.docs.zip
jQuery Mobile Git Copy
jquery-mobile.js
jquery-mobile.css
jquery-mobile.min.js
jquery-mobile.min.css
jquery.mobile.zip
jQuery UI Live Git Copy
jquery-ui-git.js
jquery-ui-git.cs
jQuery Color Live Git Copy
jquery.color-git.js
code.jquery.com/jquery-git.js
the official jQuery files on http://code.jquery.com

Our first jQuery Mobile page

- 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.

Our first jQuery Mobile page

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

Our first jQuery Mobile page

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

viewport meta tags allows us to define the content scaling and device width behavior in mobile.

Our first jQuery Mobile page

```
<body>
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
  </div>
</div>
</body>
```

the body in `index.html`

Our first jQuery Mobile page

```
<body>
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
  </div>
</div>
</body>
```

jQuery depends on the HTML5 data attributes to assign behaviors on different DOM elements.

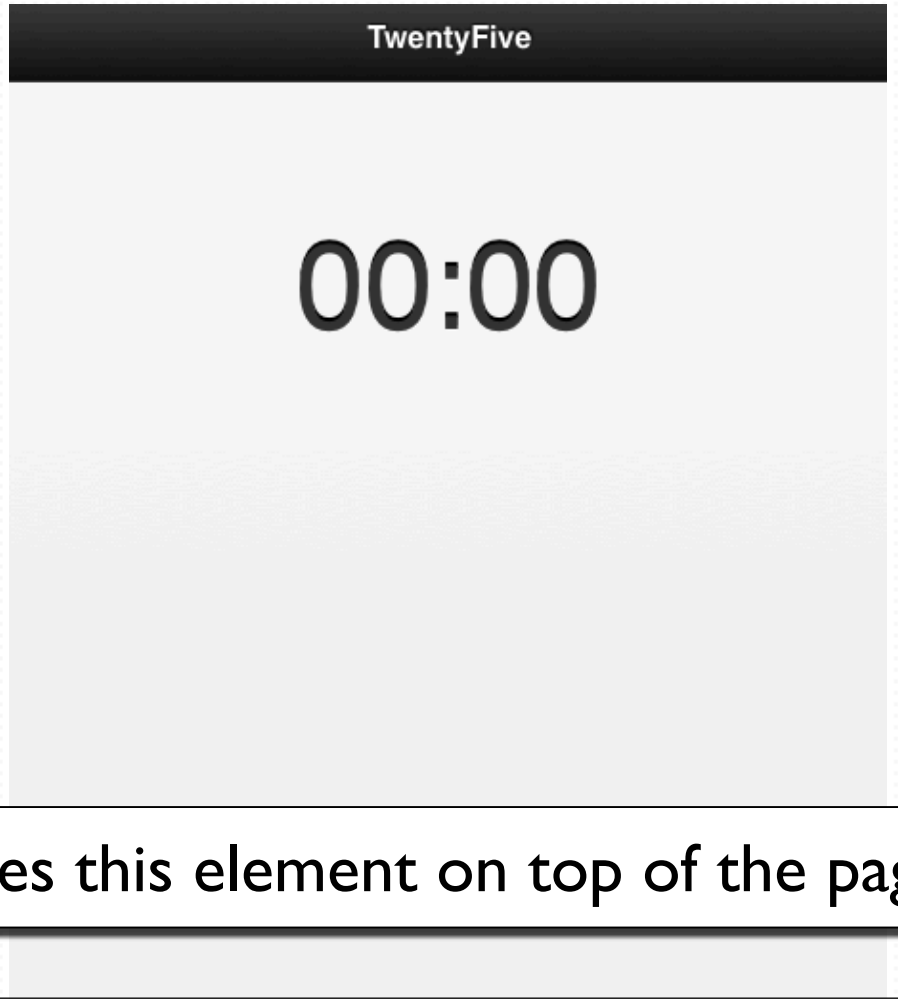
Our first jQuery Mobile page

```
<body>
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
  </div>
</div>
</body>
```

data-role='page' makes this DOM element a full page. jQuery Mobile displays a page in full screen and links different pages as different views in app.

Our first jQuery Mobile page

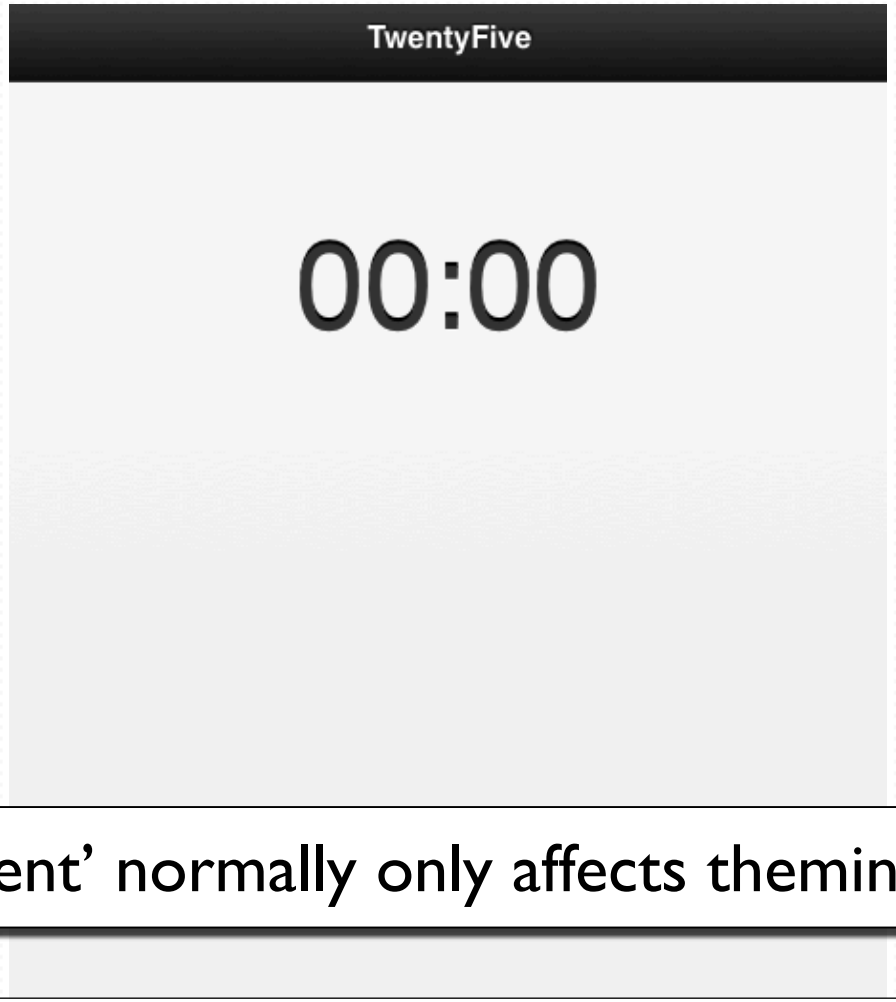
```
<body>
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
  </div>
</div>
</body>
```



data-role='header' makes this element on top of the page.

Our first jQuery Mobile page

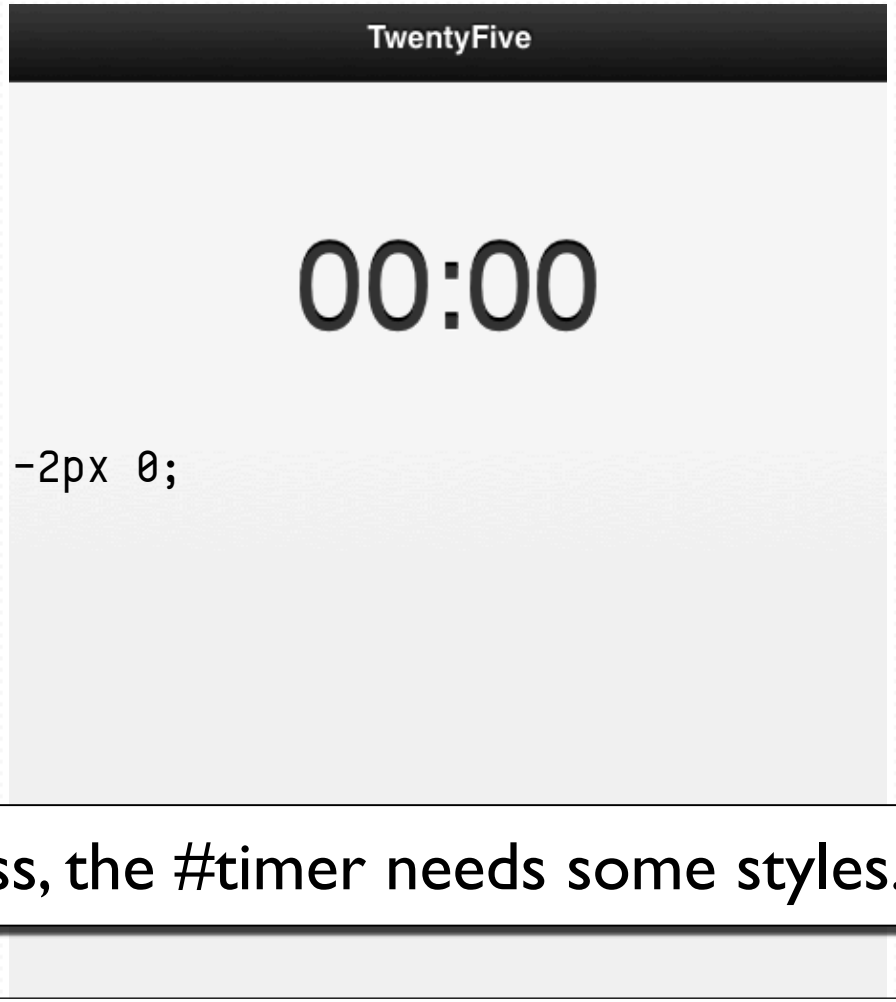
```
<body>
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
  </div>
</div>
</body>
```



data-role='content' normally only affects theming.

Our first jQuery Mobile page

```
#timer {
  text-align: center;
  font-size: 4em;
  text-shadow: #fff 0 1px 0, #000 0 -2px 0;
}
```



app.css, the #timer needs some styles.

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:

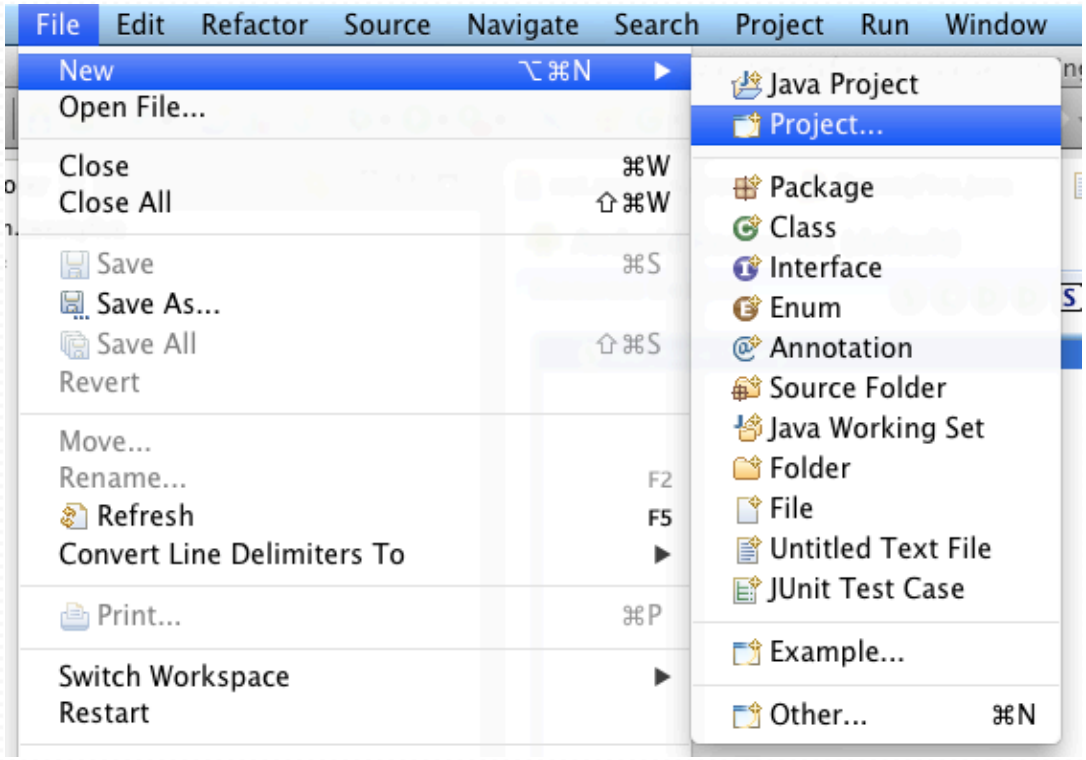
- Clone the PhoneGap Android example proj.

- Rename all occurrences from ‘example’ to our new app name.

- Place the files inside assets/www/ folder.

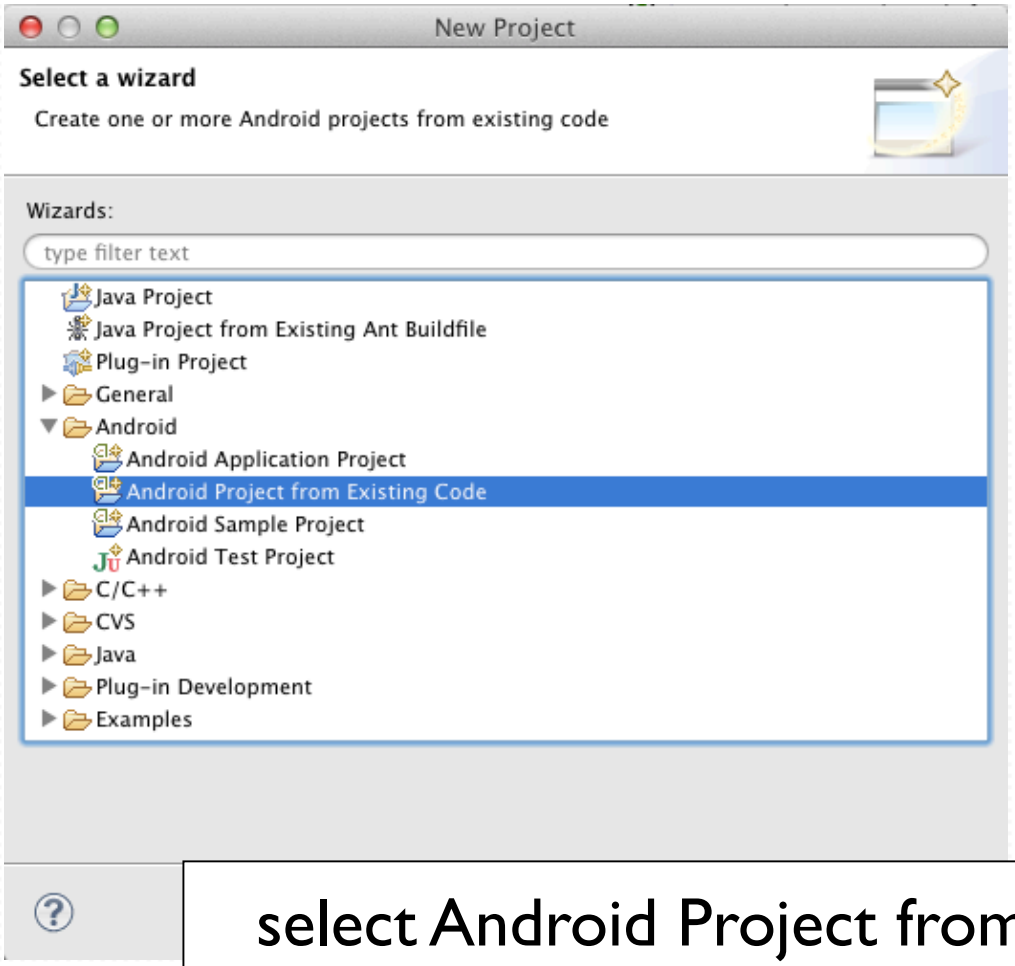
- Put it on Android device and run it.

Running jQuery Mobile in Mobile with PhoneGap



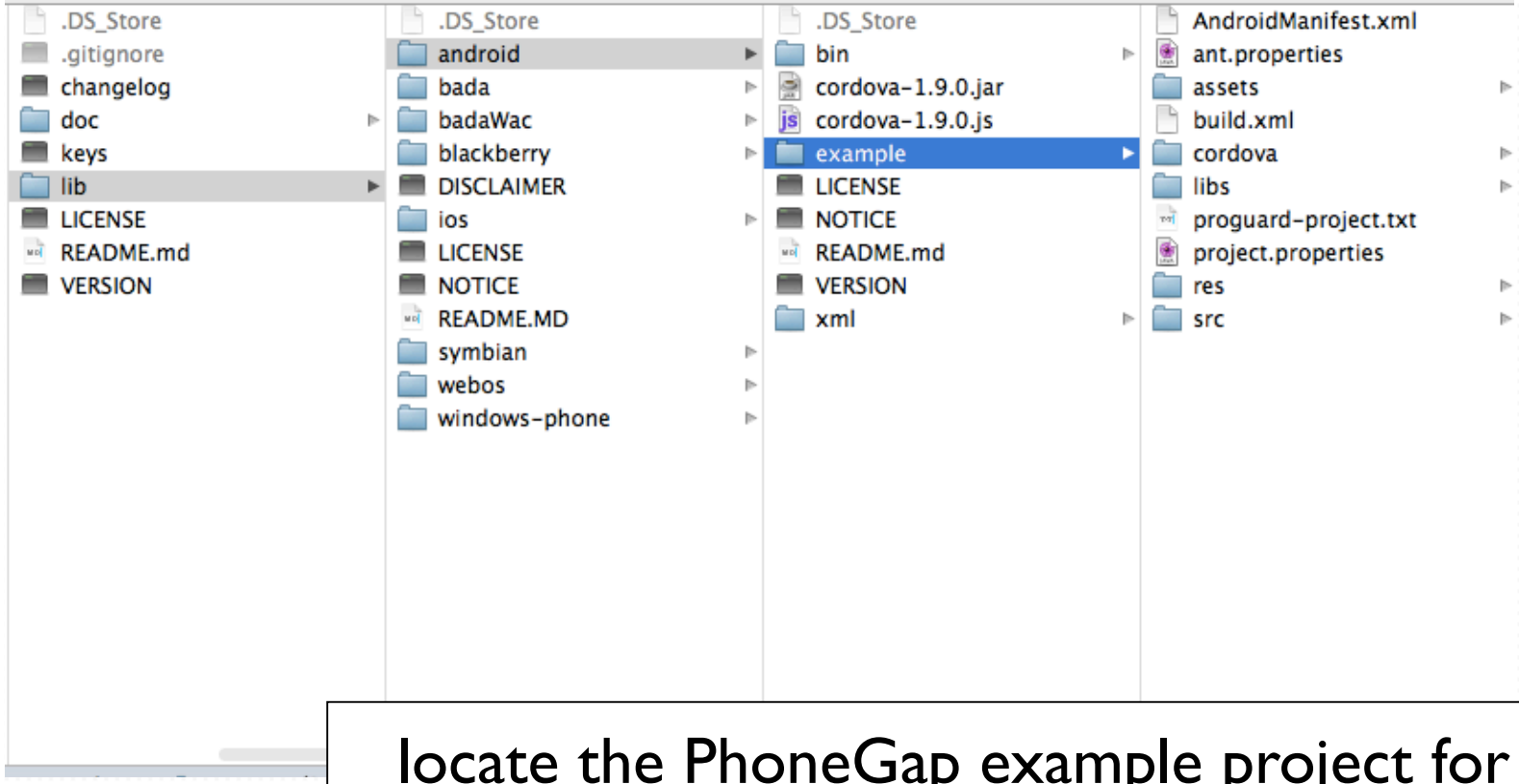
create a new project in Eclipse.

Running jQuery Mobile in Mobile with PhoneGap



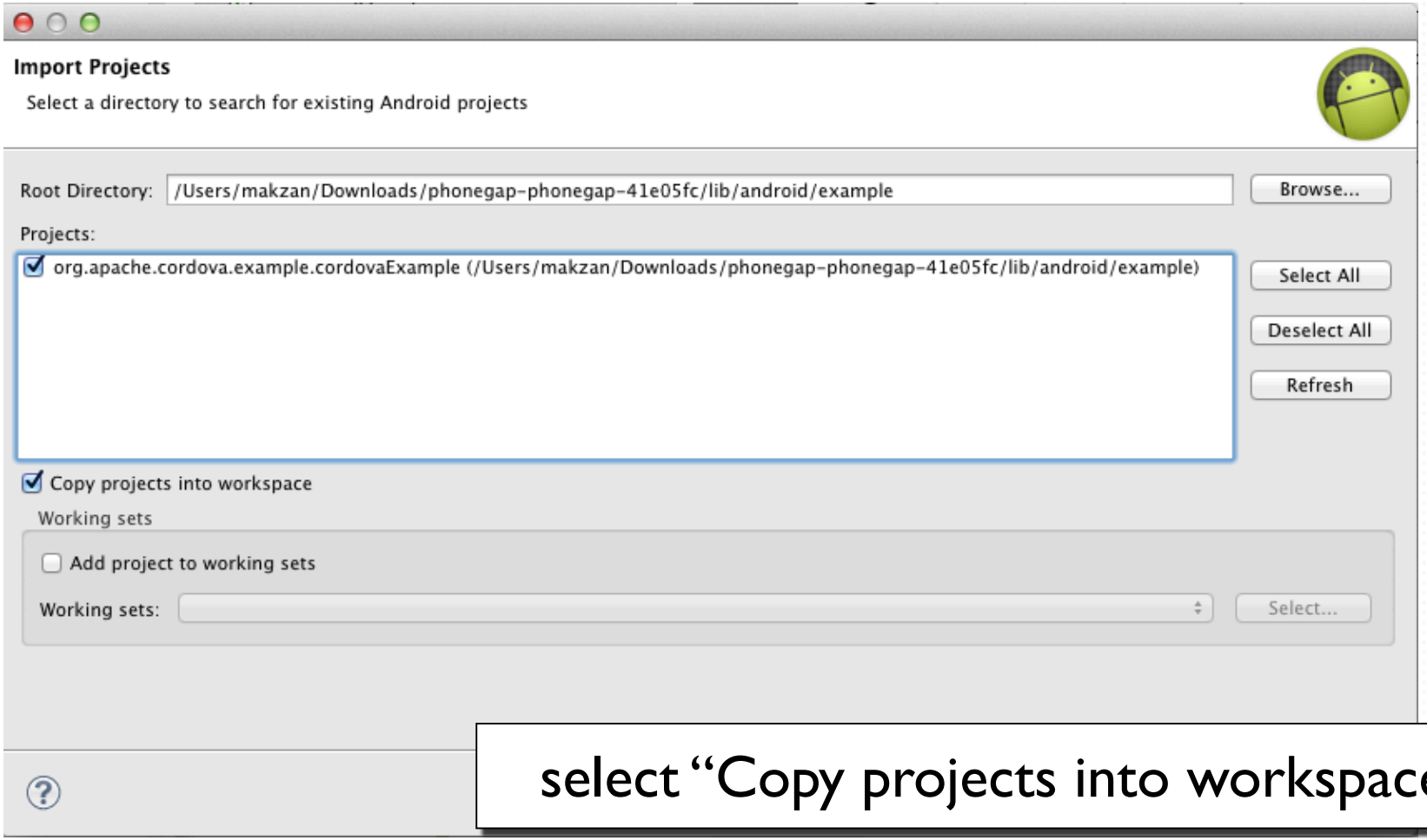
select Android Project from Existing Code.

Running jQuery Mobile in Mobile with PhoneGap



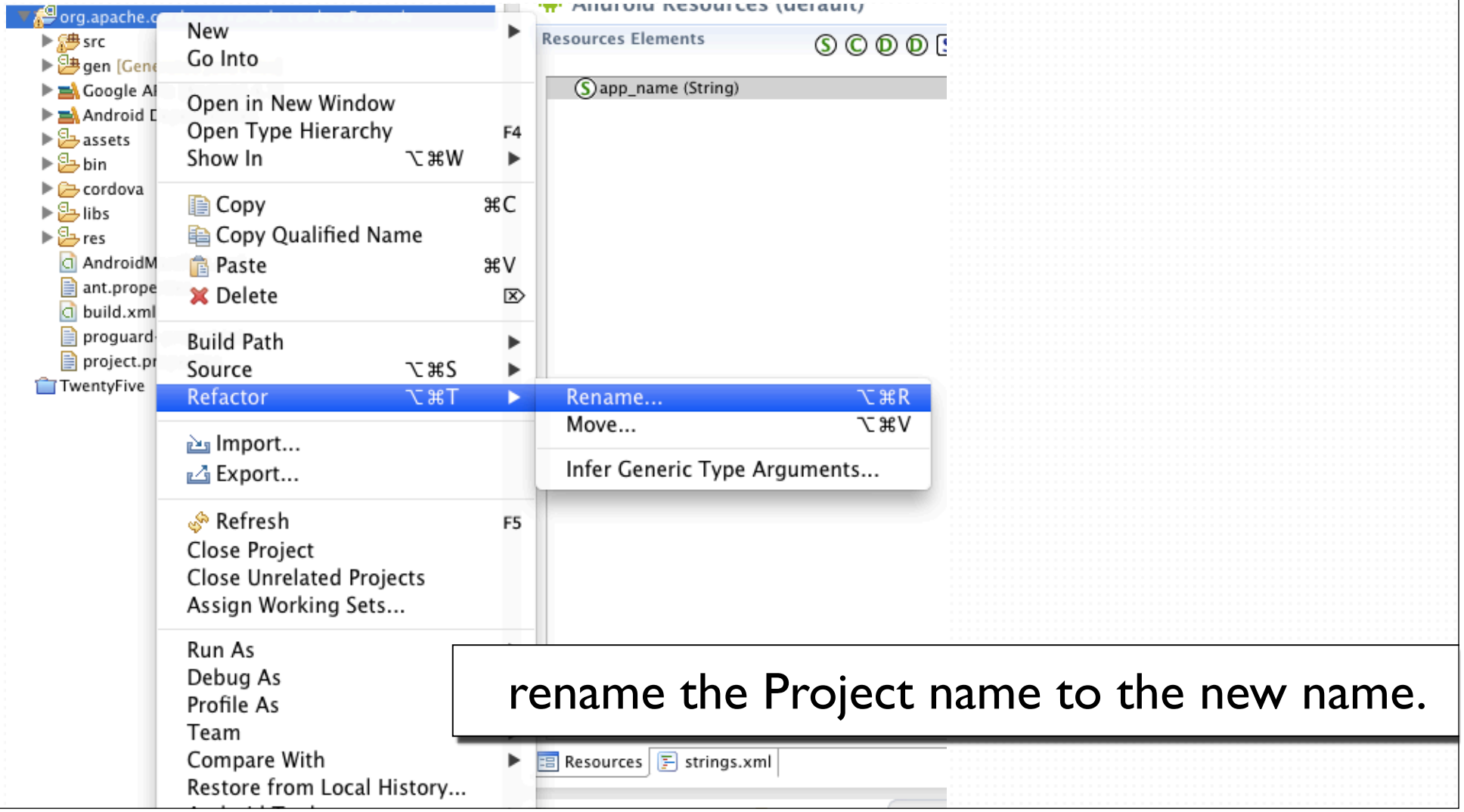
locate the PhoneGap example project for Androi
It's in <phonegap folder>/lib/android/example

Running jQuery Mobile in Mobile with PhoneGap



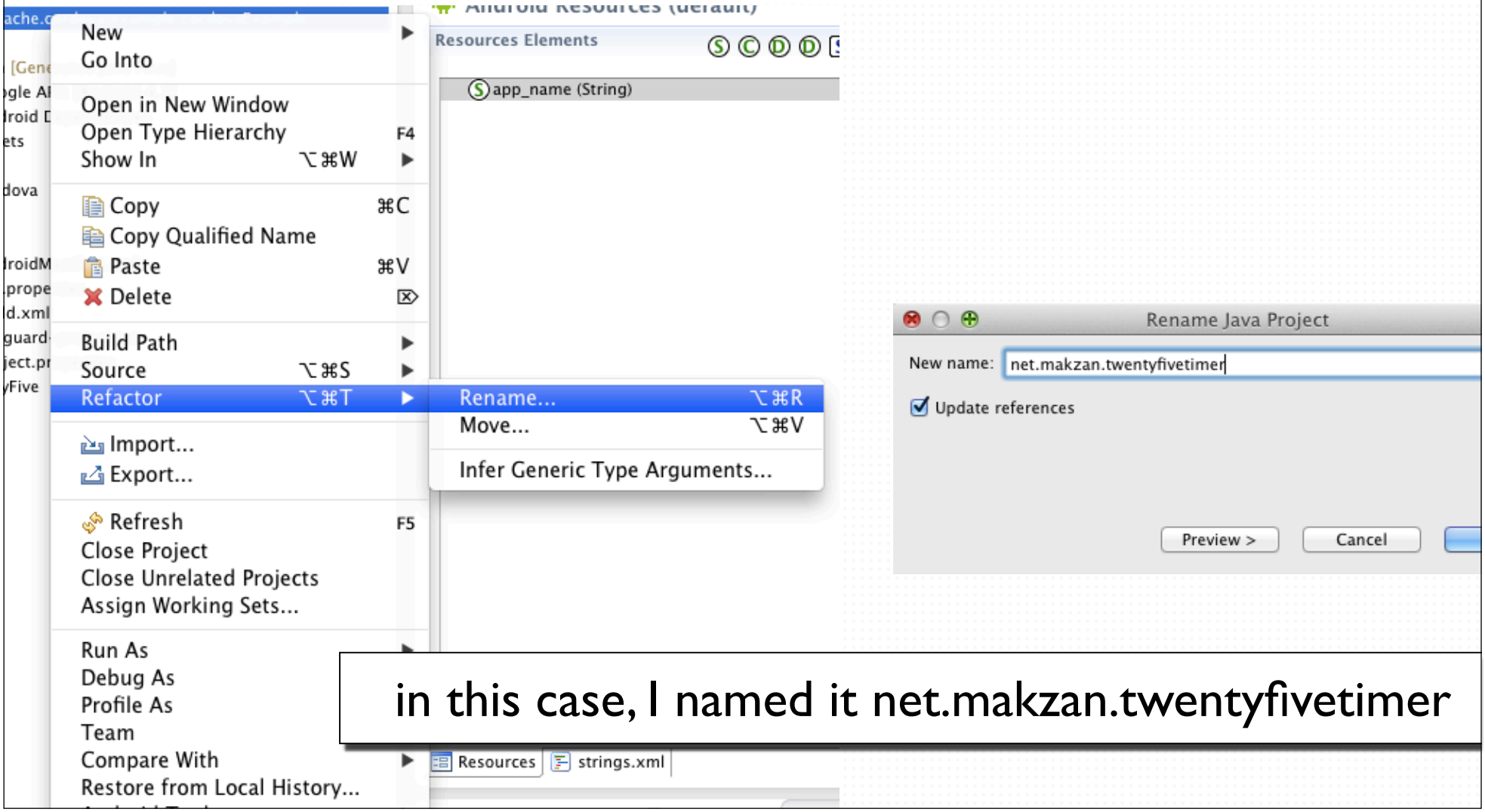
select "Copy projects into workspace"

Running jQuery Mobile in Mobile with PhoneGap



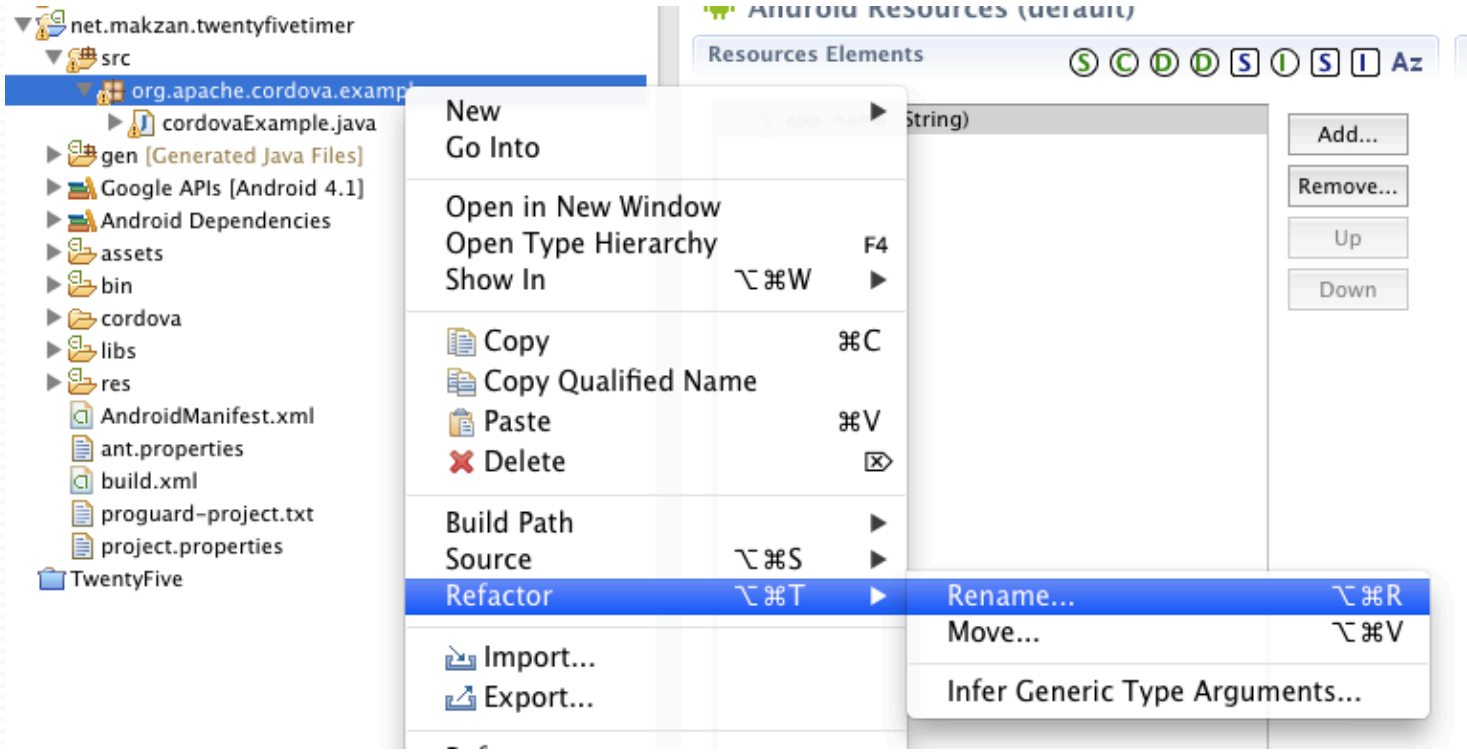
rename the Project name to the new name.

Running jQuery Mobile in Mobile with PhoneGap



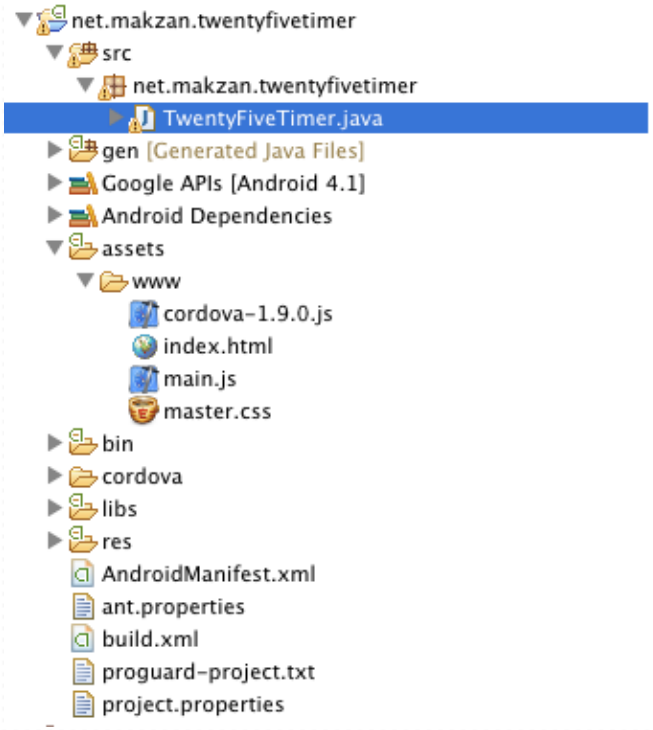
in this case, I named it net.makzan.twentyfivetimer

Running jQuery Mobile in Mobile with PhoneGap



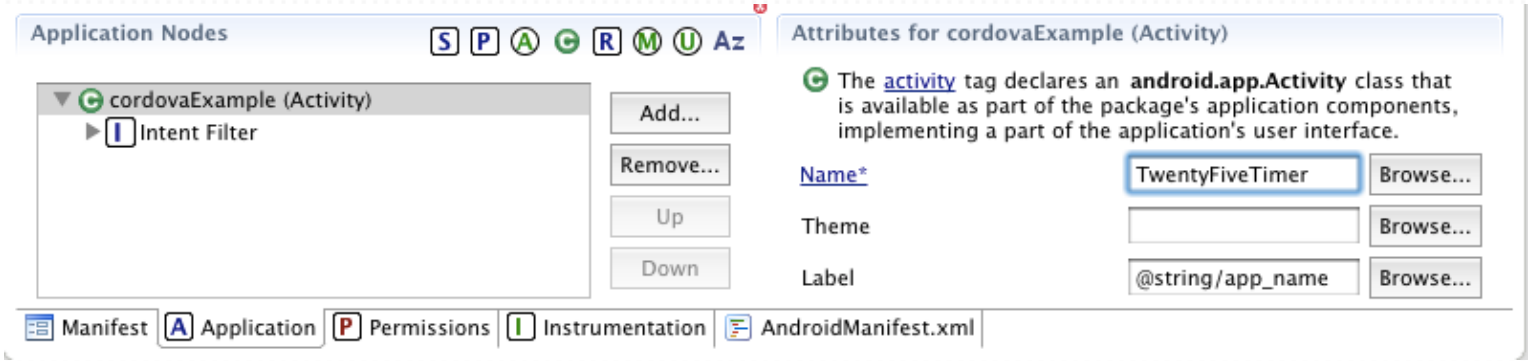
then rename the src package name.

Running jQuery Mobile in Mobile with PhoneGap



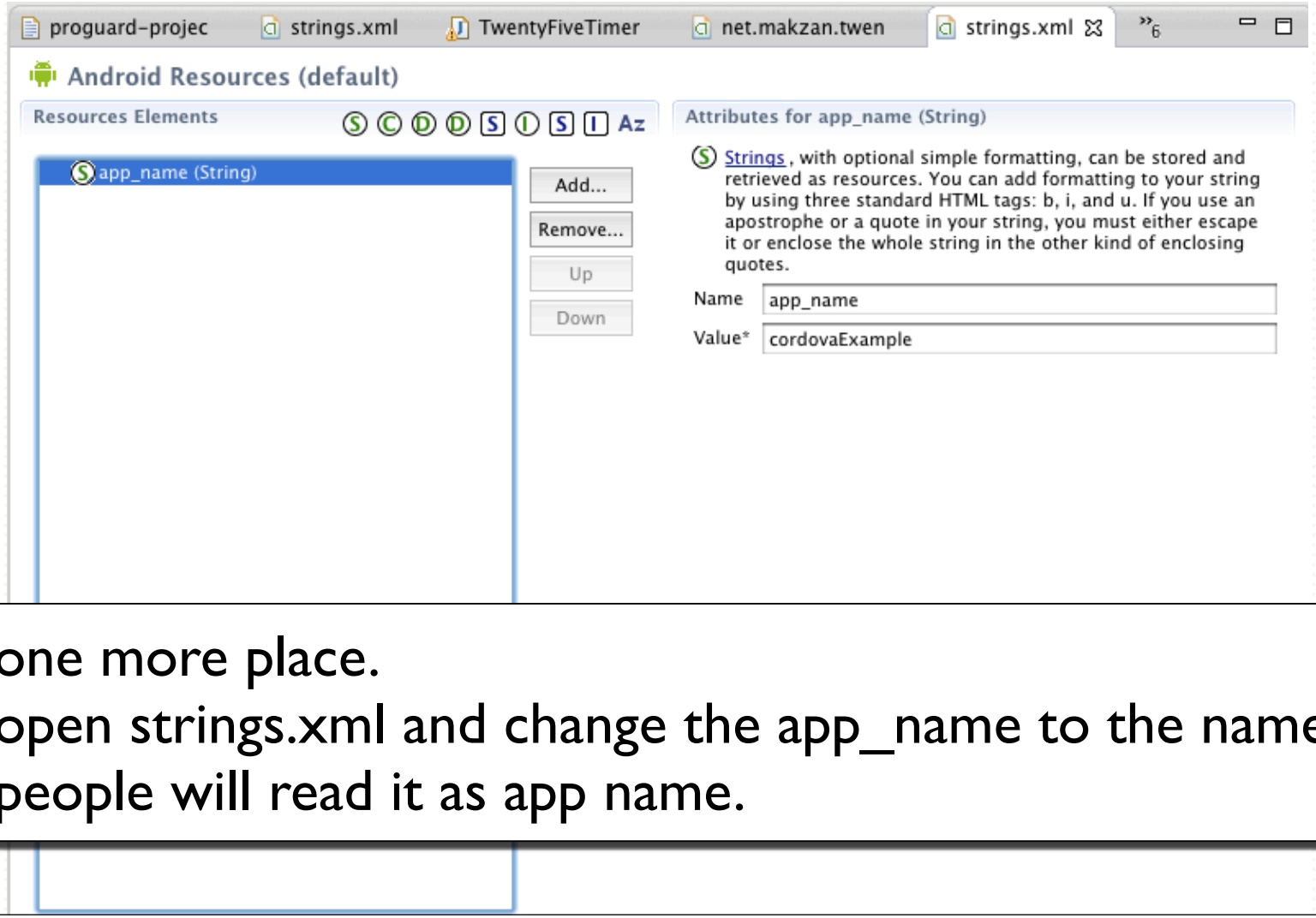
and the Java class name

Running jQuery Mobile in Mobile with PhoneGap



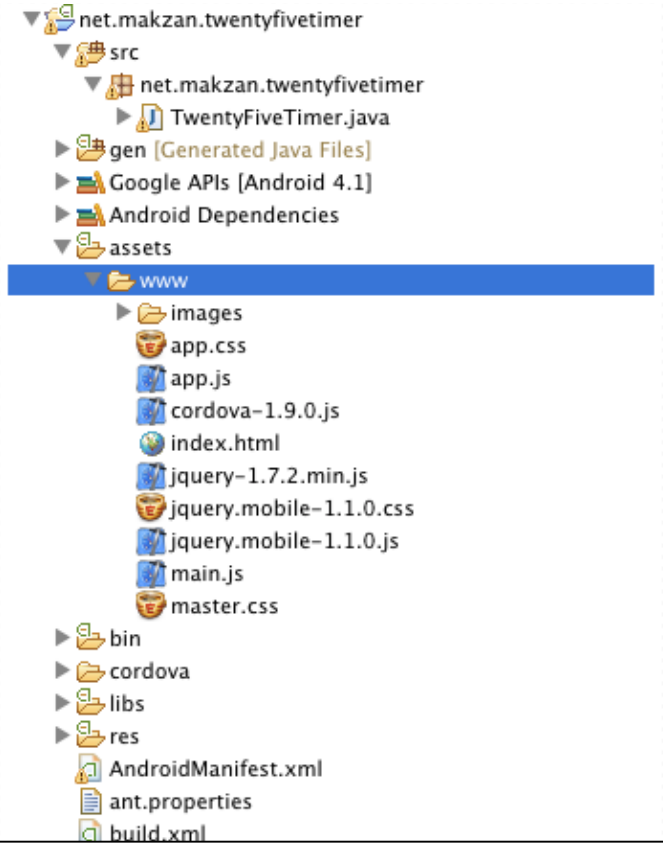
in the Application tap of Android Manifest, we need to change the Application Nodes to match the Java class name.

Running jQuery Mobile in Mobile with PhoneGap



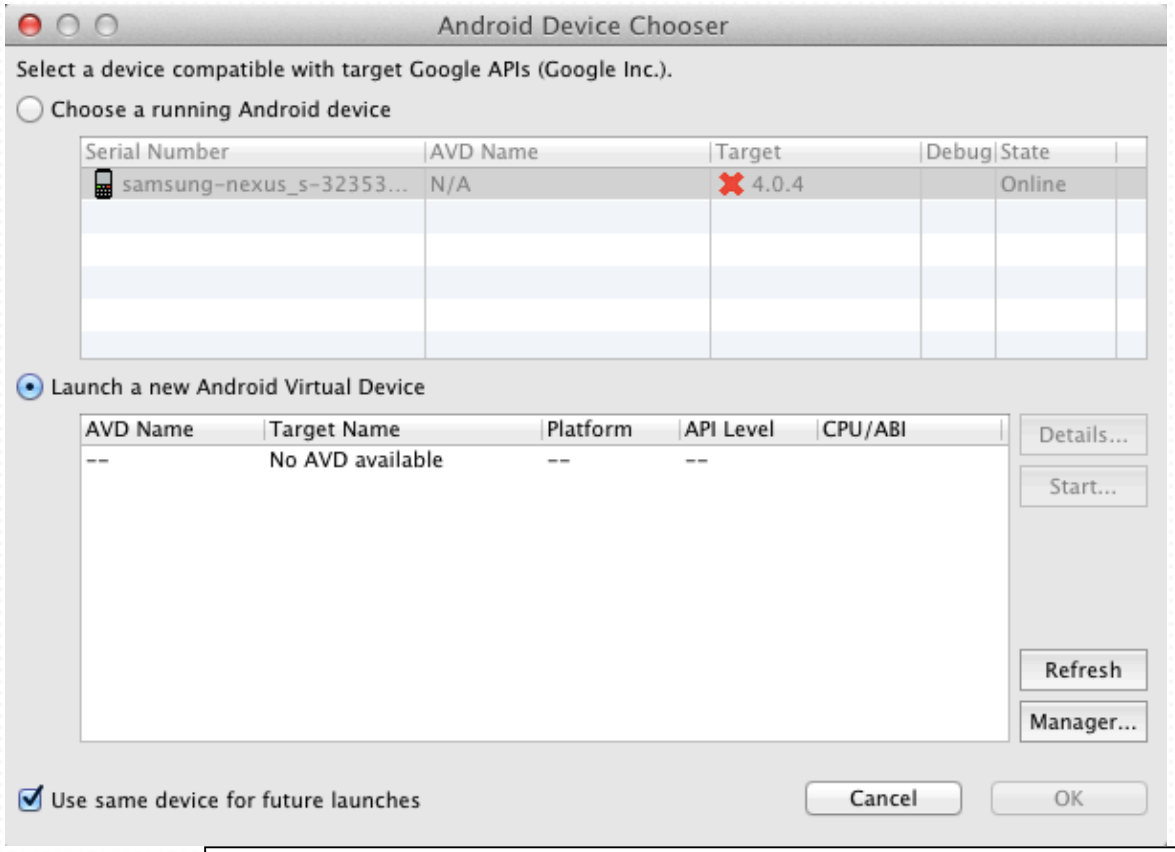
one more place.
open strings.xml and change the app_name to the name which people will read it as app name.

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.

Linking pages

In this section, we will create two pages and links them together.

Linking pages

```
<body>
<div data-role='page' id='main'>

</div>
<div data-role='page' id='contact'>

</div>
</body>
```

we can define more pages and jQuery Mobile will handle which page to show instead of displaying them all together.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

this is our new ‘main’ page.
two anchors are added.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

by default,
the first link inside header is top left button.
the second link is top right button.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

we can override this behavior by assigning `ui-btn-right` to make it appear on top right.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

`data-role='button'` makes it visually a button instead of text link.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

and a button in content appears differently than the one in header.

Linking pages

```
<div data-role='page' id='main'>
  <div data-role='header'>
    <h1>TwentyFive</h1>
    <a class='ui-btn-right' data-role='button' href='#contact'>關於</a>
  </div>
  <div data-role='content'>
    <p id='timer'>00:00</p>
    <a data-role='button' id='reset'>重新計時</a>
  </div>
</div>
```

we link it to the #anchor which is a page role.

Linking pages

```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

this is content in the second page DIV.

Linking pages

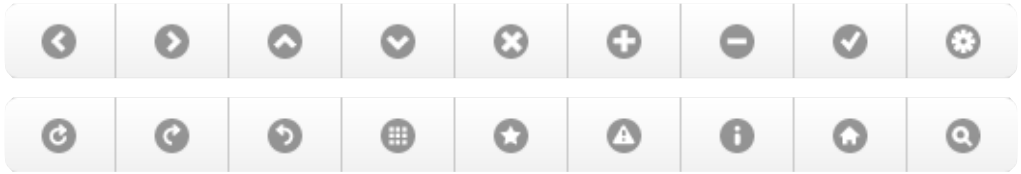
```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

we add a back button on top left of header.

Linking pages

```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

jQuery Mobile comes with some default icons.



Linking pages

```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

`data-rel` defines the link relationship.
'back' means it will links to previous page.

Linking pages

```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

`tel:` pops up the phone app with the number.

Linking pages

```
<div data-role='page' id='contact'>
  <div data-role='header'>
    <h2>關於</h2>
    <a data-icon='arrow-l' data-rel='back'>TwentyFive</a>
  </div>
  <div data-role='content'>
    <p>這是一個 25 分鐘倒計時器。運用了一種稱為 <a href='http://
www.pomodorotechnique.com/'>Pomodoro</a> 的工作時間分配技巧。</p>
    <p>電話：<a href='tel:+85366331234'>+853 6633 1234</a></p>
    <p>電郵：<a href='mailto:twentyfive@mz-lab.com'>twentyfive@mz-
lab.com</a></p>
  </div>
</div>
```

`mailto:` pops up an email composing window.

Linking pages

result of our app.

Pages Transition

In this section, we will take a look at the page transition effects and how they work.

Pages Transition

```
<div data-role='page' id='main'>
    ...

    <a data-role='button' data-transition='fade' href='#contact'>Fade</a>
    <a data-role='button' data-transition='pop' href='#contact'>Pop</a>
    <a data-role='button' data-transition='flip' href='#contact'>Flip</a>
    <a data-role='button' data-transition='turn' href='#contact'>Turn</a>
    <a data-role='button' data-transition='flow' href='#contact'>Flow</a>
    <a data-role='button' data-transition='slide' href='#contact'>Slide</a>
    <a data-role='button' data-transition='slideup' href='#contact'>Slide Up</a>
    <a data-role='button' data-transition='slidedown' href='#contact'>Slide Down</a>
    <a data-role='button' data-transition='slidefade' href='#contact'>Slide Fade</a>
    <a data-role='button' data-transition='none' href='#contact'>None</a>

...
</div>
```

the default transition is fade.
let's try different transition effects.

Pages Transition

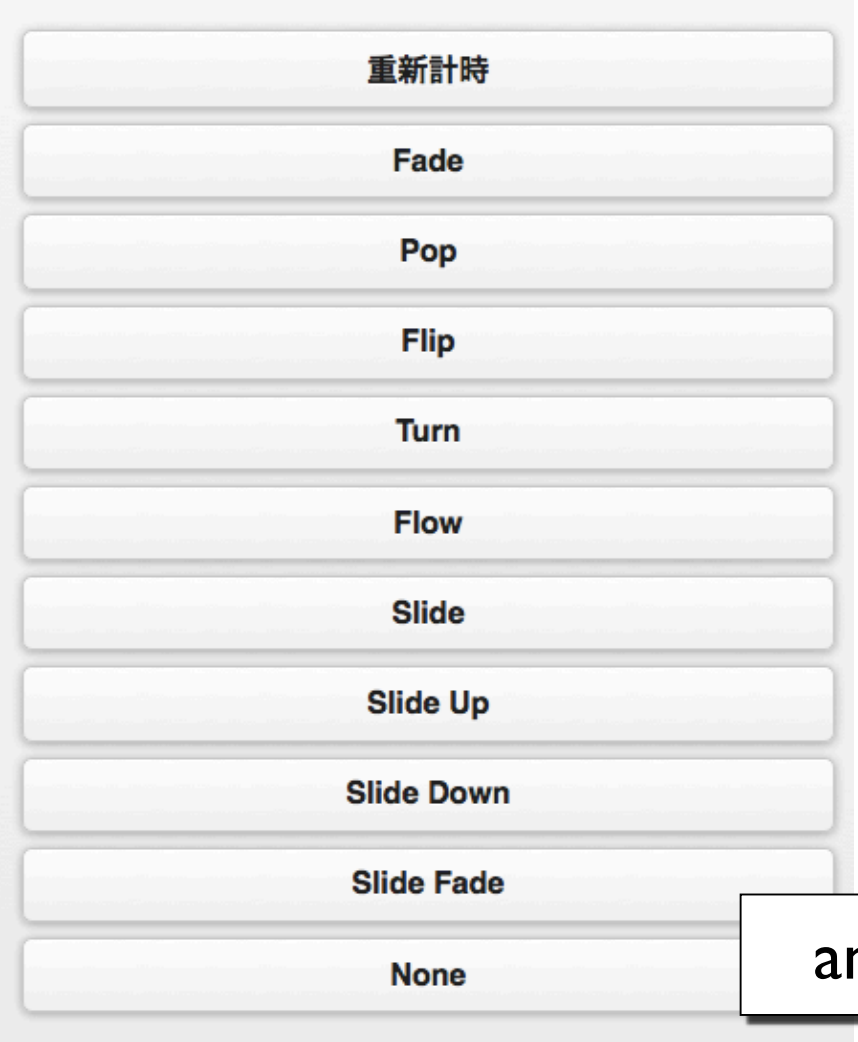
```
<div data-role='page' id='main'>
    ...

    <a data-role='button' data-transition='fade' href='#contact'>Fade</a>
    <a data-role='button' data-transition='pop' href='#contact'>Pop</a>
    <a data-role='button' data-transition='flip' href='#contact'>Flip</a>
    <a data-role='button' data-transition='turn' href='#contact'>Turn</a>
    <a data-role='button' data-transition='flow' href='#contact'>Flow</a>
    <a data-role='button' data-transition='slide' href='#contact'>Slide</a>
    <a data-role='button' data-transition='slideup' href='#contact'>Slide Up</a>
    <a data-role='button' data-transition='slidedown' href='#contact'>Slide Down</a>
    <a data-role='button' data-transition='slidefade' href='#contact'>Slide Fade</a>
    <a data-role='button' data-transition='none' href='#contact'>None</a>

...
</div>
```

we can use data-transition to define the transition effect.
later, we will change the default transition effect in JavaScript.

Pages Transition



and our app looks like this now.

CSS based Transition

```
.slide.out, .slide.in {
  -webkit-animation-timing-function: ease-out;
  -webkit-animation-duration: 350ms;
  -moz-animation-timing-function: ease-out;
  -moz-animation-duration: 350ms;
}
```

first, we need to know the duration and the easing function.

CSS based Transition

```
/* keyframes for slidein from sides */
@-webkit-keyframes slideinfromright {
    from { -webkit-transform: translateX(100%); }
    to { -webkit-transform: translateX(0); }
}
@-webkit-keyframes slideinfromleft {
    from { -webkit-transform: translateX(-100%); }
    to { -webkit-transform: translateX(0); }
}
```

and we'll see animation based on CSS keyframes.

CSS based Transition

```
@-webkit-keyframes flowinfromleft {
    0% { -webkit-transform: translateX(-100%) scale(.7); }
  30%, 40% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(0) scale(1); }
}
@-webkit-keyframes flowouttoleft {
    0% { -webkit-transform: translateX(0) scale(1); }
  60%, 70% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(-100%) scale(.7); }
}
```

and we can define steps between 0 to 100%.

CSS based Transition

```
@-webkit-keyframes flowinfromleft {
    0% { -webkit-transform: translateX(-100%) scale(.7); }
  30%, 40% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(0) scale(1); }
}
@-webkit-keyframes flowouttoleft {
    0% { -webkit-transform: translateX(0) scale(1); }
  60%, 70% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(-100%) scale(.7); }
}
```

actually, it is better to use translate3d to gain GPU rendering in mobile safari.

CSS based Transition

```
@-webkit-keyframes flowinfromleft {
    0% { -webkit-transform: translateX(-100%) scale(.7); }
  30%, 40% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(0) scale(1); }
}
@-webkit-keyframes flowouttoleft {
    0% { -webkit-transform: translate3d(0, 0, 0) scale(1); }
  60%, 70% { -webkit-transform: translateX(0) scale(.7); }
    100% { -webkit-transform: translateX(-100%) scale(.7); }
}
```

actually, it is better to use translate3d to gain GPU rendering in mobile safari.

Adding logic to app.js

In this section, we will add some javascripts.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

    // init the timer value
    $('#timer').html('25:00');

  });
}).call(this); // end of the app module
```

app.js, this is just an initializer.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

  });
}).call(this); // end of the app module
```

this is related specifically to mobile safari.
we can disable the callout, those 'copy, define' menu in iOS.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

    // init the ti
    $('#timer').ht

  });
}).call(this); // end of the app module
```

then we define the default page transition.
slide transition looks more native than others.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

    // init the timer value
    $('#timer

  });
}).call(this); // end of the app module
```

and we can set the button to reacts without delay.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

    // init the timer value
    $('#timer').html('25:00');

  });
}).call(this); // end of the app module
```

and finally we init the timer value.

Adding logic to app.js

```
// The app module
(function() {

  // jQuery ready callback
  $(function() {
    // disable the default mobile safari long tap behavior on elements
    document.documentElement.style.webkitTouchCallout = 'none'

    // default page transition
    $.mobile.defaultPageTransition = 'slide'

    // make the button reacts faster
    $.mobile.buttonMarkup.hoverDelay = 0

    // init the timer value
    $('#timer').h

  });
}).call(this); // end of the app module
```

and the code is put into an anonymous function.

The setInterval Utility

In this section, we will create a utility module for setInterval.

The setInterval Utility

```
// the ticker module
(function() {
  // declare a global variable 'tf' (short form of twenty five)
  var tf = this.tf = {}

  // a ticker is used to create global timeInterval tickers.
  tf.ticker = {};

  // prepare a list for listeners which are interested to the global timer
  tf.ticker.tickListeners = [];

  // a method to add the listener to the list
  tf.ticker.addListener = function (listener) {}

  // a method to remove listener from the list
  tf.ticker.removeListener = function (target) {}

  tf.ticker.tick = function() {}

  // finally start the ticker, every second.
  tf.globalInterval = setInterval(tf.ticker.tick, 1000);

}).call(this);
```

the skeleton without function contents.

The setInterval Utility

```
// the ticker module
(function() {
  // declare a global variable 'tf' (short form of twenty five)
  var tf = this.tf = {}

  // a ticker is used to create global timeInterval tickers.
  tf.ticker = {};

  // prepare a list for listeners which are interested to the global timer
  tf.ticker.tickListeners = [];

  // a method to add the listener to the list
  tf.ticker.addListener = function (listener) {}

  // a method to remove listener from the list
  tf.ticker.removeListener = function (target) {}

  tf.ticker.tick = function() {}

  // finally start t
  tf.globalInterval

}).call(this);
```

the only variable we expose to global scope, hopefully.

The setInterval Utility

```
// the ticker module
(function() {
  // declare a global variable 'tf' (short form of twenty five)
  var tf = this.tf = {}

  // a ticker is used to create global timeInterval tickers.
  tf.ticker = {};

  // prepare a list for listeners which are interested to the global timer
  tf.ticker.tickListeners = [];

  // a method to add the listener to the list
  tf.ticker.addListener = function (listener) {}

  // a method to remove listener from the list
  tf.ticker.removeListener = function (target) {}

  tf.ticker.tick = function() {}

  // finally start the t
  tf.globalInterval = se

}).call(this);
```

an empty object that we will put utility logic inside.

The setInterval Utility

```
// the ticker module
(function() {
  // declare a global variable 'tf' (short form of twenty five)
  var tf = this.tf = {}

  // a ticker is used to create global timeInterval tickers.
  tf.ticker = {};

  // prepare a list for listeners which are interested to the global timer
  tf.ticker.tickListeners = [];

  // a method to add the listener to the list
  tf.ticker.addListener = function (listener) {}

  // a method to remove listener from the list
  tf.ticker.removeListener = function (target) {}

  tf.ticker.tick = function() {}

  // finally start the
  tf.globalInterval =

}).call(this);
```

the list of target listeners which needs global interval.

The setInterval Utility

```
// the ticker module
(function() {
  // declare a global variable 'tf' (short form of twenty five)
  var tf = this.tf = {}

  // a ticker is used to create global timeInterval tickers.
  tf.ticker = {};

  // prepare a list for listeners which are interested to the global timer
  tf.ticker.tickListeners = [];

  // a method to add the listener to the list
  tf.ticker.addListener = function (listener) {}

  // a method to remove listener from the list
  tf.ticker.removeListener = function (target) {}

  tf.ticker.tick = function() {}

  // finally start the
  tf.globalInterval = s

}).call(this);
```

the tick function that will be executed periodically.

The setInterval Utility

```
// a method to add the listener to the list
tf.ticker.addListener = function (listener) {
  tf.ticker.tickListeners.push(listener);
}

// a method to remove listener from the list
tf.ticker.removeListener = function (target) {
  // loop to find the target in the array
  for (var i=0, len= tf.ticker.tickListeners.length; i < len; i++)
  {
    if (tf.ticker.tickListeners[i] == target) {
      tf.ticker.tickListeners.splice(i, 1); // remove that found target from the array
    }
  }
}
```

the addListener and removeListener method.
just normal array manipulation being wrapped.

The setInterval Utility

```
tf.ticker.tick = function() {
  for (var i=0, len= tf.ticker.tickListeners.length; i < len; i++)
  {
    if (tf.ticker.tickListeners[i].tick != undefined)
    {
      tf.ticker.tickListeners[i].tick(); // call the tick function on each listener.
    } else {
      console.log("TickListener instance should expose a method named 'tick'");
    }
  }
}
```

the tick method.
it calls ‘tick()’ to every registered listeners.

Core Count Down Logic

Finally, we prepared all the things and write the core count down logic.

We will name this logic module **TwentyFiveTimer**

Core Count Down Logic

```
(function() {

  var TwentyFiveTimer = (function() {
    function TwentyFiveTimer(element) {}

    TwentyFiveTimer.prototype.reset = function() {}

    // the tick function, core logic of the app
    TwentyFiveTimer.prototype.tick = function() {}

    TwentyFiveTimer.prototype.start = function() {}

    TwentyFiveTimer.prototype.restart = function () {}

    // finish the class implementation, at last we return it to the outter scope.
    return TwentyFiveTimer;

  })();

  // and set it to our global scope to let other module to use it.
  tf.TwentyFiveTimer = TwentyFiveTimer;

}).call(this);
```

skeleton of the TwentyFiveTimer class.

Core Count Down Logic

```
(function() {

  var TwentyFiveTimer = (function() {
    function TwentyFiveTimer(element) {}

    TwentyFiveTimer.prototype.reset = function() {}

    // the tick function, core logic of the app
    TwentyFiveTimer.prototype.tick = function() {}

    TwentyFiveTimer.prototype.start = function() {}

    TwentyFiveTimer.prototype.restart = function () {}

    // finish the class implementation, at last we return it to the outter scope.
    return TwentyFiveTimer;

  })();

  // and set it to our global scope to let other module to use it.
  tf.TwentyFiveTimer = TwentyFiveTimer;

}).call(this)
```

basically it is a class implementation and expose it to the global **tf** variable.

Core Count Down Logic

```
function TwentyFiveTimer(element) {
  this.counter = 0; // the counting variable

  this.element = element; // the element to show timer result

  this.reset();
}

TwentyFiveTimer.prototype.reset = function() {
  tf.ticker.removeListener(this); // remove self instance from the ticker list.

  this.counter = 60 * 25; // 25 minutes;

  $(this.element).html("25:00");
}
```

constructor logic and the reset method.

Core Count Down Logic

```
function TwentyFiveTimer(element) {
  this.counter = 0; // the counting variable

  this.element = element; // the element to show timer result

  this.reset();
}

TwentyFiveTimer.prototype.reset = function() {
  tf.ticker.removeListener(this); // remove self instance from the ticker list.

  this.counter = 60 * 25; // 25 minutes;

  $(this.element).html("25:00");
}
```

constructor takes one parameter which is the DOM element to display the timer result.

Core Count Down Logic

```
TwentyFiveTimer.prototype.start = function() {
  tf.ticker.addListener(this); // add self instance to the ticker list, so we will get tick
invoked.
}

TwentyFiveTimer.prototype.restart = function () {
  this.reset();
  this.start();
}
```

starting and starting over.

Core Count Down Logic

```
// the tick function, core logic of the app
TwentyFiveTimer.prototype.tick = function() {
  this.counter--; // the real counting down.

  if (this.counter < 0) this.counter = 0; // dont forget to set the boundary.

  // the minute
  var minute = Math.floor( this.counter / 60 );
  if (minute < 10) minute = '0' + minute;

  // the second
  var second = this.counter % 60;
  if (second < 10) second = '0' + second;

  // display it, finally.
  $(this.element).html(minute + ":" + second);
}
```

the real counting logic.

Core Count Down Logic

```
$(function() {
  ...

  $('#timer').html('25:00');

  var timer = new tf.TwentyFiveTimer($('#timer'));
  timer.start();

  // handle the reset button
  $('#reset').click( function(e){
    timer.restart();
  });

});
```

at last, we start the timer.
and listen to the restart trigger.

Count Down Finished

Some last minute tweaks to pop up a dialog when count down finished.

Count Down Finished

```
<div data-role='dialog' id='finish'>
    <div data-role='content'>
        到時間了!!
        <a data-role='button' href='#main'>OK</a>
    </div>
</div>
```

back to our HTML, we add one dialog role with a message and a button.

Count Down Finished

```
TwentyFiveTimer.prototype.tick = function() {
  this.counter--; // the real counting down.

  if (this.counter < 0)
  {
    this.counter = 0; // dont forget to set the boundary.

    // call the dialog when the counter reaches 0
    $.mobile.changePage("#finish");

    this.stop();
  }

  ...
}
```

we can use $.mobile.changePage to programmatically change to another page. and dialog is treated as a page.

Count Down Finished

```
TwentyFiveTimer.prototype.tick = function() {
  this.counter--; // the real counting down.

  if (this.counter < 0)
  {
    this.counter = 0; // dont forget to set the boundary.

    // call the dialog when the counter reaches 0
    $.mobile.changePage("#finish");

    this.stop();
  }

  ...
}

TwentyFiveTimer.prototype.stop = function() {
  tf.ticker.removeListener(this);
}
```

and one more method to stop the counter.

Bonus?

```
.ui-btn-corner-all {
  border-radius: 5px;
}
```

I don't like the round button style from the default jQuery Mobile. a 5px border radius is enough to me.

More Bonus?

```
if (windoω.Touch) {

  $('a').bind('touchstart', function(e) {
    e.preventDefault();
  });

  $('a').bind('touchend', function(e) {
    e.preventDefault();
    return $(this).trigger('click');
  });

}
```

anchor link in iOS has some delays.
we can fix it with custom touchstart and touchend event.