PROJECT: BUILDING A WEBSITE WITH jQTOUCH

The core to jQuery Mobile is jQuery, an open source framework that makes JavaScript just that little bit easier to work with. For mobile frameworks, though, jQuery Mobile was not the first kid on the block—that was jQTouch. Figure P3.1 illustrates the application you will build in this project.

You will see how you can leverage jQTouch to build a fully functional web application. The focus of the project is creating a tool that allows you to see where a fueling station is located.

What You Will Need

To get started with this project you will need to download the files from www.visualizetheweb.com. The files for this project are contained in a ZIP file that you can extract on your desktop.

All you need to build the project are the following project files:

- Index.html
- map.html
- Images folder (with one image)
- jQTouch documents (containing two JavaScripts and one CSS document)

Setting Up Your jQTouch Project

Let’s get down to business and step through the main document for this project. Most of your work will be completed in the main index.html web page.

The app is an HTML5 document and starts as a standard HTML5 web page:

```html
<html>
<head>
<title>Gas2GO!</title>
```
The following three META elements are specific to Apple’s iPhone. Unlike jQuery Mobile, jQTouch specifically targets first iPhone, then Android, and finally BlackBerry. It does not attempt to run on all devices.

```html
<meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;" />
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent" />
```

The following points to the core jQTouch Cascading Style Sheet document and then a specific theme.

```html
<link type="text/css" rel="stylesheet" media="screen" href="jqtouch/jqtouch.css">
<link type="text/css" rel="stylesheet" media="screen" href="themes/jqt/theme.css">
```

The following two JavaScript files are the core jQuery library and the jQTouch library:

```html
<script type="text/javascript" src="jqtouch/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="jqtouch/jqtouch.js"></script>
```

As with any web page, you can add custom JavaScript. Here, the following script extends the functionality of the core jQTouch framework. These classes allow you to set a default icon, startup screen logo, and status bar color:

```javascript
var jQT = new $.jQTouch({
    icon: 'logo.png',
    addGlossToIcon: false,
    startupScreen: 'jqt_startup.png',
    statusBar: 'black',
});
```

It is useful to have some of the images used frequently in the application preloaded. The following function allows you to do just this:

```javascript
preloadImages: [
    '../themes/jqt/img/back_button.png',
    '../themes/jqt/img/back_button_clicked.png',
    '../themes/jqt/img/button_clicked.png',
    '../themes/jqt/img/grayButton.png',
    '../themes/jqt/img/whiteButton.png',
    '../themes/jqt/img/loading.gif',
];
```
The following is a page animation callback event:

```javascript
$('#pageevents').
  bind('pageAnimationStart', function(e, info){
    $(this).find('.info').append('Started animating ' +
    info.direction + '&hellip; :');
  }).
  bind('pageAnimationEnd', function(e, info){
    $(this).find('.info').append(' finished animating ' +
    info.direction + '&lt;br/&gt;&lt;br/&gt;');
  });

All page animations end with an Ajax callback event as shown:

```javascript
$('#callback').bind('pageAnimationEnd', function(e, info){
  if (!$(this).data('loaded')) {
    $(this).append($('<div>Loading</div>').load('ajax.html .info', function() {
      $(this).parent().data('loaded', true);
    }));
  }
});
// Orientation callback event
$('body').bind('turn', function(e, data){
  $('#orient').html('Orientation: ' + data.
  orientation);
});
</script>
```

Additional Cascading Style Classes extend the core properties of the main jQTouch CSS document. The following extends functionality of the main "header" section:

```html
<style type="text/css">
  div.wrapper { width: 940px; margin: 0 auto; padding:
    0 30px 36px; position: relative; }
  div#header { background: #f5f5f5; height: 72px; border-bottom: 1px solid #eee; margin: 0; }
  div#header h4 { float: left; position: absolute; top: 24px; left: 145px; border-left: 1px solid #ddd; padding-left: 14px; }
  div#header h4 small { font-size: 14px; font-weight: normal; }
  div#header h4 a, div#header h4 a:visited { font-weight: normal; }
  div.page-header { padding: 0 0 8px; margin: 18px 0; border-bottom: 1px solid #ddd; }
  div.page-header h1 { padding: 0; margin: 0; font-size: 24px; line-height: 27px; letter-spacing: 0; }
</style>
```
The “awesome” class is extended here:

```html
.awesome, .awesome:visited {background: #222 url(/images/alert-overlay.png) repeat-x; display: inline-block; padding: 5px 10px 6px; color: #fff; text-decoration: none; -moz-border-radius: 5px; -webkit-border-radius: 5px; -webkit-box-shadow: 0 1px 3px rgba(0,0,0,0.5); background-clip: content-box; text-shadow: 0 -1px 1px rgba(0,0,0,0.25); border-bottom: 1px solid rgba(0,0,0,0.25); position: relative; cursor: pointer; width: 100%;}
.awesome:hover { background-color: #111; color: #fff; }
.awesome:active { top: 1px; }
.small.awesome, .small.awesome:visited { font-size: 11px; }
.medium.awesome, .medium.awesome:visited { font-size: 13px; font-weight: bold; line-height: 1; text-shadow: 0 1px 3px rgba(0,0,0,0.25); }
.large.awesome, .large.awesome:visited { font-size: 14px; padding: 8px 14px 9px; }
.green.awosome, .green.awosome:visited { background-color: #91bd09; }
.green.awesome:hover { background-color: #749a02; }
.blue.awesome, .blue.awesome:visited { background-color: #2daebf; }
.blue.awesome:hover { background-color: #007d9a; }
.red.awesome, .red.awesome:visited { background-color: #e33100; }
.red.awesome:hover { background-color: #872300; }
.magenta.awesome, .magenta.awesome:visited { background-color: #a9014b; }
.magenta.awesome:hover { background-color: #630030; }
.orange.awesome, .orange.awesome:visited { background-color: #ff5c00; }
.orange.awesome:hover { background-color: #d45500; }
.yellow.awesome, .yellow.awesome:visited { background-color: #ffb515; }
.yellow.awesome:hover { background-color: #fc9200; }
</style>
</head>

The next step is to add the first screen that will show on the page. All the screens are managed using DIV elements. As with jQuery Mobile, all the screens are loaded and managed with one web page, and are shown as “screens” on the iPhone.

The following block of HTML is the main home screen that loads by default when the application runs on a website. There are two main sections to the page: the toolbar and main content:

```html
<div id="home">
  <div class="toolbar">
    <h1>Gas2GO</h1>
  </div>
</div>
```
The structure of this screen can be seen in other screens in the application as shown:

```html
<div id="nearest1">
  <div class="toolbar">
    <h1>Station #1288</h1>
    <a class="button back" href="#">Back</a>
  </div>
  <ul class="rounded">
    <p>&nbsp;</p>
    <span class="headlineRed">Station is Closed</span>
    <p>&nbsp;</p>
    <button class="large yellow awesome" style="margin-left:auto;margin-right:auto;">
      <h1><a href="tel:5555555555">Call Station #1288</a></h1>
    </button>
    <p>&nbsp;</p>
    <p><span style="color: #FFF;">
      Time to station: 8 mins<br>
      Amount of Fuel At Station: 15 liters<br>
    </span></p>
  </ul>
</div>
```

You can step through most of the code in the web page and you will see that the app follows the same structure in managing content in a screen.

One screen that deserves additional attention is the settings screen. The settings screen is different for a simple reason: it contains a form.

The setup for the page is very similar to a normal screen, as follows:

```html
<div id="settings">
  <div class="toolbar">
    <h1>Settings</h1>
    <a class="button back" href="#">Back</a>
  </div>
</div>
```
Things get slightly different when you reach the form itself. For the most part, the form is a standard HTML form, as shown in Figure P3.2. What is different is the use of HTML5 Form attributes to control how data is viewed and entered in each FORM element. You can see that the Placeholder attribute is used extensively.

```html
<form method="post">
  <h1>Credit Card</h1>
  <ul class="rounded">
    <li><input placeholder="Credit Card Number" type="text" name="age" id="age" /></li>
    <li><input placeholder="Name on Card Number" type="text" name="age" id="age" /></li>
    <li><input placeholder="Credit Card Number" type="text" name="age" id="age" /></li>
    <li><input placeholder="Amount" type="text" name="weight" id="weight" /></li>
    <li><input type="submit" class="submit" name="action" value="Save Changes" /></li>
  </ul>
</form>
```

Save the files and preview them on an iPhone. You are presented with a solution that looks and feels very much like a native application, but is running in a web page.

![Figure P3.2 A form built with jQTouch.](image)

**Summary**

On the whole, jQTouch is a solid framework for iPhones. It is easy to edit and control. The framework does lose some traction for websites that require support for a broader range of mobile devices. But it makes for a good base to get you started.