CHAPTER 6

Creating an Adaptive Layout

In Chapter XXX, I showed you how to create a layout for your app using the Metro support for the single-page content model. In this chapter, I show you how to make that layout adapt to different views and orientations. Views are available on most Windows 8 devices and allow the user to select different ways to interact with an app—including having two Metro apps run side by side. Orientations arise on devices that can be held in different positions or are easily rotated and that are equipped with a sensor to report on their state. You need to think carefully about how you will accommodate different views and orientations in your app to create a first-class Metro experience.

I also show you how to deal with high–pixel density displays. These displays are increasingly common on tablet and phone platforms and present the user with a crisper display than is possible with traditional hardware. For the most part, Windows 8 takes care of pixel density for you, but there is one key exception that requires attention: bitmap images. I explain how Windows 8 approaches pixel density and show you the Metro features that help you present the right resolution bitmap for the hardware being used. Table 6-1 provides the summary for this chapter.

Table 6-1. Chapter Summary

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Listing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapt to Metro views using CSS</td>
<td>Use media rules with the -ms-view-state property.</td>
<td>6-1–6-4</td>
</tr>
<tr>
<td>Adapt to Metro views using JavaScript</td>
<td>Handle the resize event emitted by the DOM window object and read the current view from the Windows.UI.ViewManagement.ApplicationView.value property.</td>
<td>6-5, 6-6</td>
</tr>
<tr>
<td>Adapt to view changes in imported content</td>
<td>Listen for the resize event in the ready handler for the WinJS.UI.Pages.define method and use CSS media rules with the -ms-view-state property.</td>
<td>6-7, 6-8</td>
</tr>
<tr>
<td>Break out of the snapped view</td>
<td>Call the Windows.UI.ViewManagement.ApplicationView.tryUnsnap method</td>
<td>6-9</td>
</tr>
<tr>
<td>Adapt to device orientation using JavaScript</td>
<td>Handle the orientationchanged event emitted by the Windows.Graphics.Display.DisplayProperties object.</td>
<td>6-10, 6-11</td>
</tr>
<tr>
<td>Adapt to device orientation using CSS</td>
<td>Use media rules with the orientation property.</td>
<td>6-12</td>
</tr>
<tr>
<td>Set and override orientation preferences.</td>
<td>Set the initial preferences in the app manifest and override them using the Windows.Graphics.Display.DisplayProperties object.</td>
<td>6-13, 6-14</td>
</tr>
</tbody>
</table>
Creating the Example Project

I have created an example project called AppViews so that I can demonstrate the different features in this chapter. Once again, I have used the Visual Studio Blank App project template. You can see the additions I have made to default.html, which I will use as my HTML master page, in Listing 6-1.

Listing 6-1. The contents of the default.html file in the AppViews Project

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>AppViews</title>
    <!-- WinJS references -->
    <link href="/Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
    <script src="/Microsoft.WinJS.1.0/js/base.js"></script>
    <script src="/Microsoft.WinJS.1.0/js/ui.js"></script>
    <!-- AppViews references -->
    <link href="/css/default.css" rel="stylesheet">
    <link href="/css/views.css" rel="stylesheet">
    <script src="/js/default.js"></script>
  </head>
  <body>
    <div id="gridContainer">
      <div id="topLeft">Top Left</div>
      <div id="topRight">Top Right</div>
      <div id="bottomLeft">Bottom Left</div>
      <div id="bottomRight">Bottom Right</div>
    </div>
  </body>
</html>
```

I used the CSS grid feature to create a master layout that is a 2-by-2 grid. The grid will be contained in the div element whose id is gridContainer, and each of the child elements contains a label to indicate its position. You can see the CSS properties I have used to create the layout in Listing 6-2, which shows the css/default.css file. There is a second CSS file linked to default.html—this file is called views.css. It is currently empty, and I’ll come back to it later in this chapter.