Step 1 - Getting Started

Basic workspace - text editor (Text Wrangler) and Adobe Photoshop (CS6).
Next, we're going to create the file structure our site will use. On your MSU WEB SPACE create a folder called:

leaflet

INSIDE that leaflet folder, create a folder called images.

- `images/` – The directory where we will place all of the images for our site.
- `index.html` – The main page for your site.
- `javascripts/` – The directory where your JavaScript files will be stored.
- `stylesheets/` – The directory where your stylesheets will be stored.

Step 2 - Basic Layout

Start by setting up the initial structure for your HTML and CSS. This will include the main HTML containers and some essential CSS properties.

```
<!DOCTYPE html>
 <html>
<head>
    <title>Leaflet | An iPhone app</title>
    <meta charset="utf-8" />
    <meta name="description" content="An iPhone ap." />
    <link rel="shortcut icon" href="/favicon.ico" />
    <link rel="stylesheet" href="stylesheets/site.css" />
</head>
<body>

    <div id="app_info" class="container">

        Main App Info

    </div>

    <div id="app_details" class="container">

        App Features / Screenshots

    </div>

    <div id="footer" class="container">

        Footer Content

    </div>

</body>
</html>
```

The site will consist of three vertical rows of content, `#app_info`,`#app_details`, and `#footer`, each being a direct descendant of the `<body>` tag. Also, each of these three s have a class called `container` assigned to it which will impart some basic styling to the containers including giving them a fixed width and aligning them properly.

Next, create a FOLDER inside of the LEAFLET folder and call it sytlesheets.
Open Text Wrangler and copy the CSS below. When you're done, upload it to your MSU web space and place it INSIDE the folder called STYLESHEETS.

site.css

CSS:

```
01  * { margin: 0; padding: 0; }
02  /*-------------------------------------------
03  Global Styles
04  -------------------------------------------*/
05  body {
06      color: #fff;
07      font-family: "Helvetica Neue", Helvetica, sans-serif;
08      font-size: 14px;
09  }
10  div.container {
11      width: 960px;
12      margin: 0 auto;
13  }
```

REFERENCE:

You are defining styles for the `<body>` tag including the default font styles for our site, in addition to defining the corresponding CSS for the `.container` class. You'll also notice an area with the text /* — Global Styles — */. This is a CSS comment, which we'll be using more of throughout the tutorial to help automatically organize our stylesheet as it grows.

PHOTOSHOP SLICED IMAGES

Place ALL the sliced images your created for last weeks homework and UPLOAD it to the folder: **leaflet/images**

That means your going to go to your MSU web space, open the folder called LEAFLET and open the folder inside of that called IMAGES. That's where you upload the sliced images.

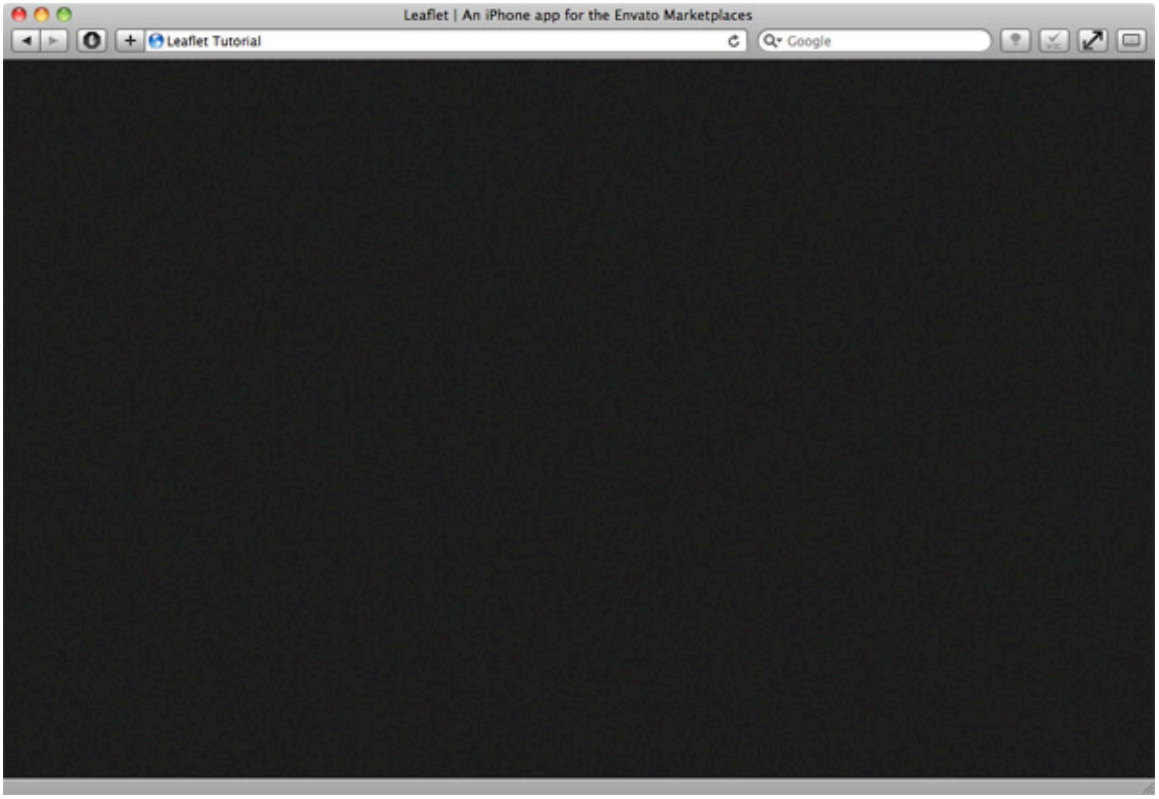
Background Images

To continue with our markup, we start by adding in the texture and spotlight images. This step requires CSS only, since the initial HTML we created in Step 2 already has the structure we need for this step. Let's modify your stylesheet to reflect the changes:

```
01
02
03  * { margin: 0; padding: 0; }
04  /*-------------------------------------------
05  Global Styles
06  -------------------------------------------*/
07  body {
08      background: url('../images/texture.jpg') repeat top left;
09      color: #fff;
10      font-family: "Helvetica Neue", Helvetica, sans-serif;
11      font-size: 14px;
12  }
13  div.container {
14      width: 960px;
15      margin: 0 auto;
16  }
17  /*-------------------------------------------
18  App Info
19  -------------------------------------------*/
20  div#app_info {
21      background: url('../images/spotlight.jpg') no-repeat top center;
22      overflow: auto;
23  }
24
25
```

- First, we add the `background` property to the already present `body` tag telling our image to continually repeat in both directions starting from the top left of the page.
- Then, we've added the spotlight by attaching it as a background image to the`#app_info` defined as `div#app_info`. We've also defined an `overflow: auto` property which will ensure the div stretches to fit our content (this is an alternative to using the "clear" CSS property).

Now we can preview our page and it should look like this:



You may be confused as to why you aren't seeing the spotlight image. This is because you haven't added any content into `#app_info` (where the spotlight is assigned) nor have we set a height property for it, thus our browser automatically thinks the height is 0px. But don't worry, as soon as wyou start adding content the spotlight will appear.

Step 5 - iPhones Graphic

Now let's add our iPhones. For this, we'll need to modify our `#app_info` structure by adding a for the graphic.

```
<div id="app_info" class="container">

    <div class="phones"></div>

</div>
```

- Here, we've added `<div class="phone">` which will be left completely empty. Its height, width and background image will be set via CSS.

Next, add the following CSS into your stylesheet:

```
div#app_info div.phones {
    background: url('../images/phones.png') no-repeat top;
    float: left;
    height: 625px;
```

```
4         width: 421px;
5     }
6
```

1. The `background` property sets the iPhone graphic as the background of the .
2. The `float` property tells our to position itself leftmost of whatever parent container it is in.
3. Lastly, the `height` and `width` properties are set to match our graphic's dimensions exactly.

Here's what our page should look like so far:

Step 6 - Logo & Description

To add in our logo and app description we're going to modify `#app_info's` structure again.

```
01<div id="app_info" class="container">
```

```
    <div class="phones"></div>

    <div class="info">

        <h1>Leaflet</h1>

        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam vel velit velit, i
mollis nunc imperdiet lectus luctus sit amet rhoncus purus laoreet. Praesent laoreet tristiq

    </div>

</div>
```

- The first element added is the `<div class="info">` container. This entire container will be positioned to the right of the iPhone's graphic.
- The `h1` tag will be the container for our logo where its text will be hidden and it will be assigned with our logo image as its background using CSS (This has some SEO benefits).
- The last element is a simple `p` tag with some placeholder text.

The CSS for this section is below:

```
div#app_info div.info {
    float: right;
    width: 500px;
    margin-top: 100px;
}
    div#app_info h1 {
        width: 315px;
        height: 80px;
        text-indent: -999999px;
        background: url('../images/logo.png') no-repeat;
    }
    div#app_info .info p {
        font-size: 15px;
        line-height: 26px;
        text-shadow: 2px 1px 8px #000;
        margin: 38px 0px 23px 0px;
    }
```

Here is a breakdown of the above code:

- The first block, `div#app_info div.info` contains a `float` property (to align our info container to the right), a `width` property and a `margin-top` property to give us some spacing on top.
- The next block is for our `h1` tag. The first properties, `height and width`, are set to the dimensions of our logo image. The next property, `text-indent`, is going to hide the text

within the h1 tag by indenting it out of the view of our browser window. Then, we use the `background` property to set the logo image as the h1′s background.

- The last element defined is our `p` tag. For the first two properties, `font-size` and`line-height`, we're going to get these values directly from the PSD. Next is `text-shadow`, which is a CSS3 property that will give our text a bit of dimension. Lastly, we're going to position the text using some margins (for top and bottom).

Here's what our page should look like so far:

Step 7 - App Store Button

To complete the `#app_info` container, we're going to add in the app store Button. Below you'll find the completed `#app_info` section along with the added button code:

```
01<div id="app_info" class="container">
02
```

```
03     <div class="phones"></div>
04
05     <div class="info">
06
07         <h1>Leaflet</h1>
08
09         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam vel velit velit, in
mollis nunc imperdiet lectus luctus sit amet rhoncus purus laoreet. Praesent laoreet tristiqu
10
11         <div class="app_store">
12             <a href="http://itunes.apple.com/us/app/leaflet/id401517510?mt=8">Buy
13         </div>
14
15     </div>
16
</div>
17
```

First, the button elements are contained inside a with the class of `app_store`.

Next, we add an `<a>` tag with the link to buy our app.

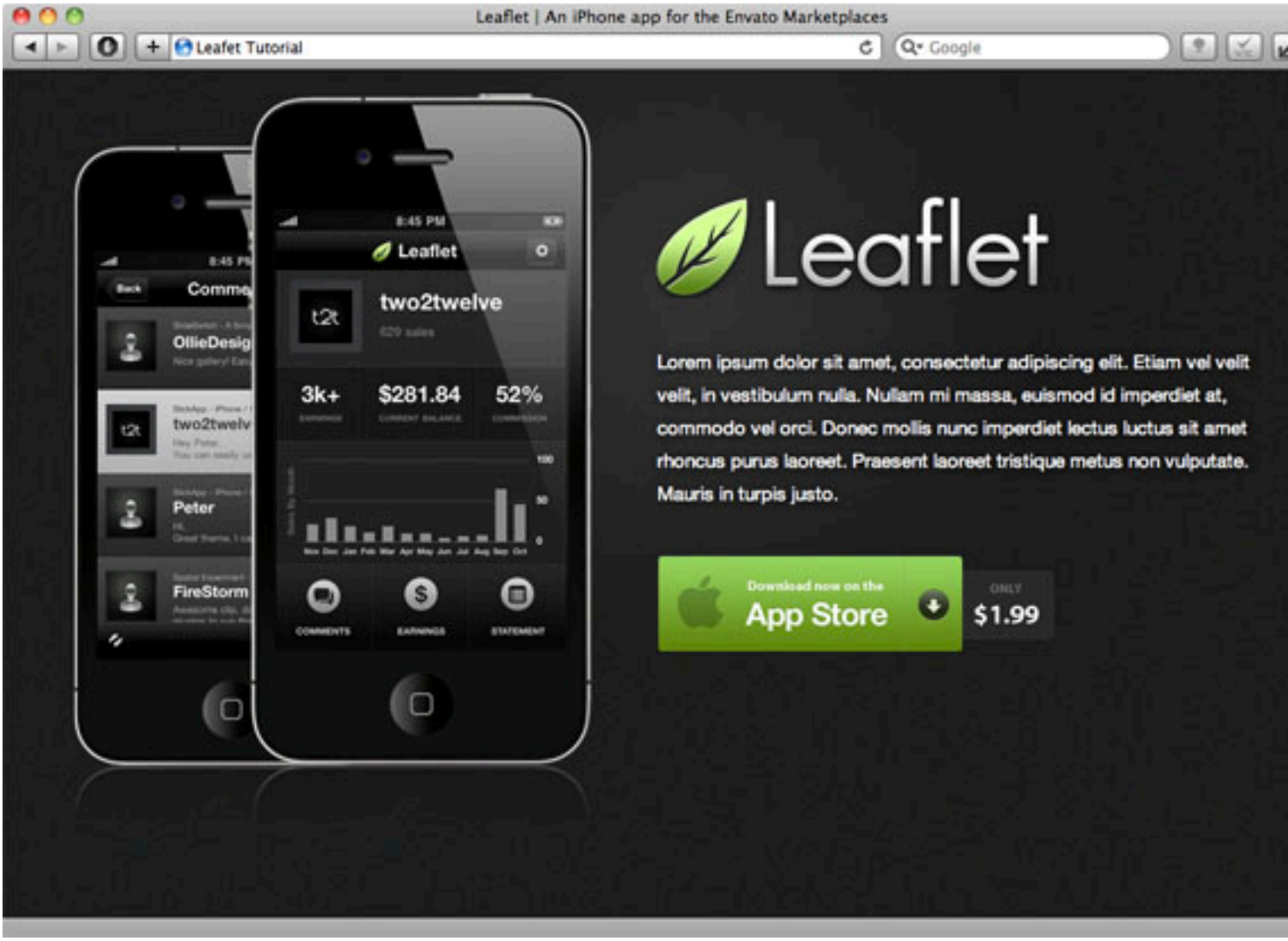
Below is the corresponding CSS:

```
01
02
03   div.app_store {
04       margin: 0px 0px 18px -12px;
05       width: 326px;
06       height: 105px;
07       background: url('../images/app_store.png') no-repeat top;
08   }
09       div.app_store a {
10           width: 100%;
11           height: 100%;
12           display: block;
13           opacity: 0;
14           filter: alpha(opacity=0);
15           text-indent: -999999px;
16           background: url('../images/app_store.png') no-repeat bottom;
       }
       div.app_store a:hover { filter:none; }
```

- **We start by defining some basic styles for the `<div class="app_store">`container. These include some margins (on the bottom and left), height and width (set to the button's dimensions).**
- **Next, for the `<a>` tag, we set the height and width to 100%, essentially making it the same size as the `.app_store` container. Then, we use the `display: block`property, without which the height property would be ignored. Using `opacity` and`filter` we set the opacity to 0% (the filter property is for IE browsers only). We also make use of `text-indent` to indent the text within the <a> tag out of view and lastly, setting the `background` image to the same as value as the `.app_store` container, but this time the positioning is set to `bottom`.**

- Since IE browsers don't support transparent PNG's, we can't apply the JavaScript "glowing" effect that we'll be creating later to IE. As a work around, we're going to create a regular hover effect by removing the `filter` property on `div.app_store a:hover`.

Here's what our page should look like so far:



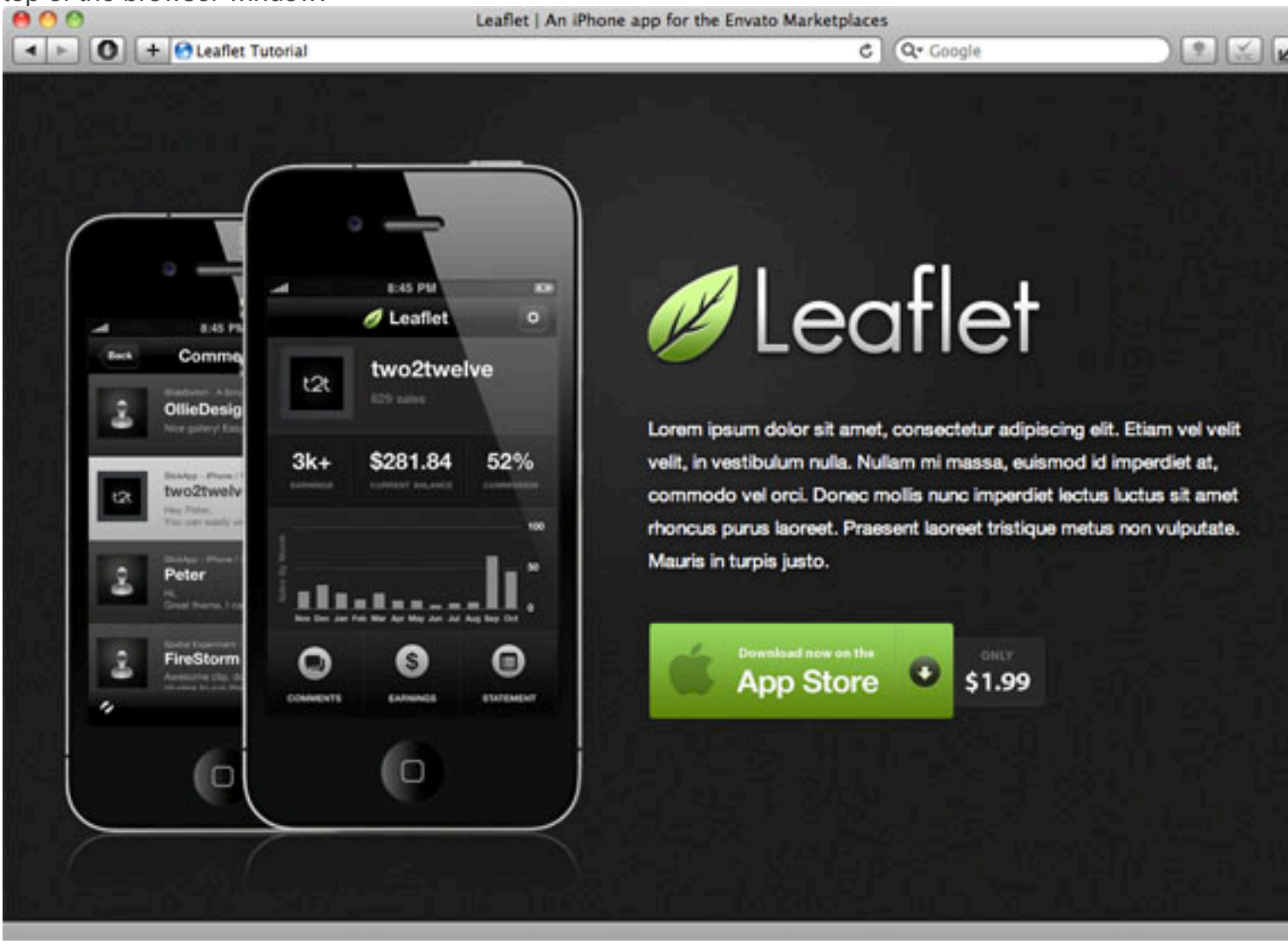
Step 8 - Adding Some Spacing

The last step for the `#app_info` area is to add some top padding. Let's do this by modifying our CSS and adding a `padding-top` property:

```
1  div#app_info {
2      padding-top: 50px;
3      background: url('../images/spotlight.jpg') no-repeat top center;
4      overflow: auto;
   }
```

5

Now we should have some much needed spacing between the start of our site and the top of the browser window:



Step 9 - App Details Setup

We can move on to the `#app_details` area now. We'll need to implement the horizontal separator line as well as other essential properties for the `#app_details`container.

Add this section to your stylesheet:

```
/*-------------------------------------------
App Details
-------------------------------------------*/
div#app_details {
    background: url('../images/h_seperator.png') repeat-x top;
    padding: 40px 0px 30px 0px;
```

```
5         overflow: auto;
6     }
7
8
```

- Firstly, we add in our horizontal separator line as a background image to the`#app_details` and ensure it is repeated horizontally with `repeat-x`.
- We also add some padding for the top and bottom of the line and container contents.
- Lastly, the `overflow: auto` property will ensure the stretches to fit our content.

Here's what our page should look like so far:

Step 10 - Screenshots

For the screenshots area, we're going to arrange our thumbnail images in a grid, add a slick hover state, and later, enhance the hover effect with some Javascript.

Let's start by creating an unordered list for the thumbnails. Below is the HTML which gets inserted into the `#app_details` container.

```
<div id="app_details" class="container">

    <div class="screenshots">
        <ul>
            <li>
                <a href="images/screen_dashboard_full.png" rel="screenshots" title="Leaflet
                    <img src="http://tutsplus.s3.amazonaws.com/tutspremium/web-d
Dashboard" />
                    <span></span>
                </a>
            </li>
            <li class="right">
                <a href="images/screen_comments_full.png" rel="screenshots" title="View You
                    <img src="http://tutsplus.s3.amazonaws.com/tutspremium/web-d
Your Comments" />
                    <span></span>
                </a>
            </li>
            <li class="bottom">
                <a href="images/screen_earnings_full.png" rel="screenshots" title="Check Yo
                    <img src="http://tutsplus.s3.amazonaws.com/tutspremium/web-d
Your Earnings" />
                    <span></span>
                </a>
            </li>
            <li class="bottom">
                <a href="images/screen_statement_full.png" rel="screenshots" title="View Yo
                    <img src="http://tutsplus.s3.amazonaws.com/tutspremium/web-d
Your Statement" />
                    <span></span>
                </a>
            </li>
        </ul>
    </div>

</div>

```

This may seem like a lot of code for a simple list of images, but not to worry, below you'll find a breakdown of each line:

- The first element we've inserted is `<div class="screenshots">`. This is the container for our screenshots and will be assigned several CSS properties for alignment and spacing.
- Next we have an `<ul>` tag which opens the unordered list.
- Then, each individual thumbnail is wrapped in a `<li>` tag. Inside each <li> there is an `<a>` tag that has both a `rel` and `title` property (these will be utilized later with Javascript). Next, is the image itself in a simple `<img>` tag. We also have a blank span tag which will be used to create a "hovered" state for the thumbnails.

- There are also a few classes assigned to the last three 's which will be assigned some CSS to fix the spacing and padding issues.

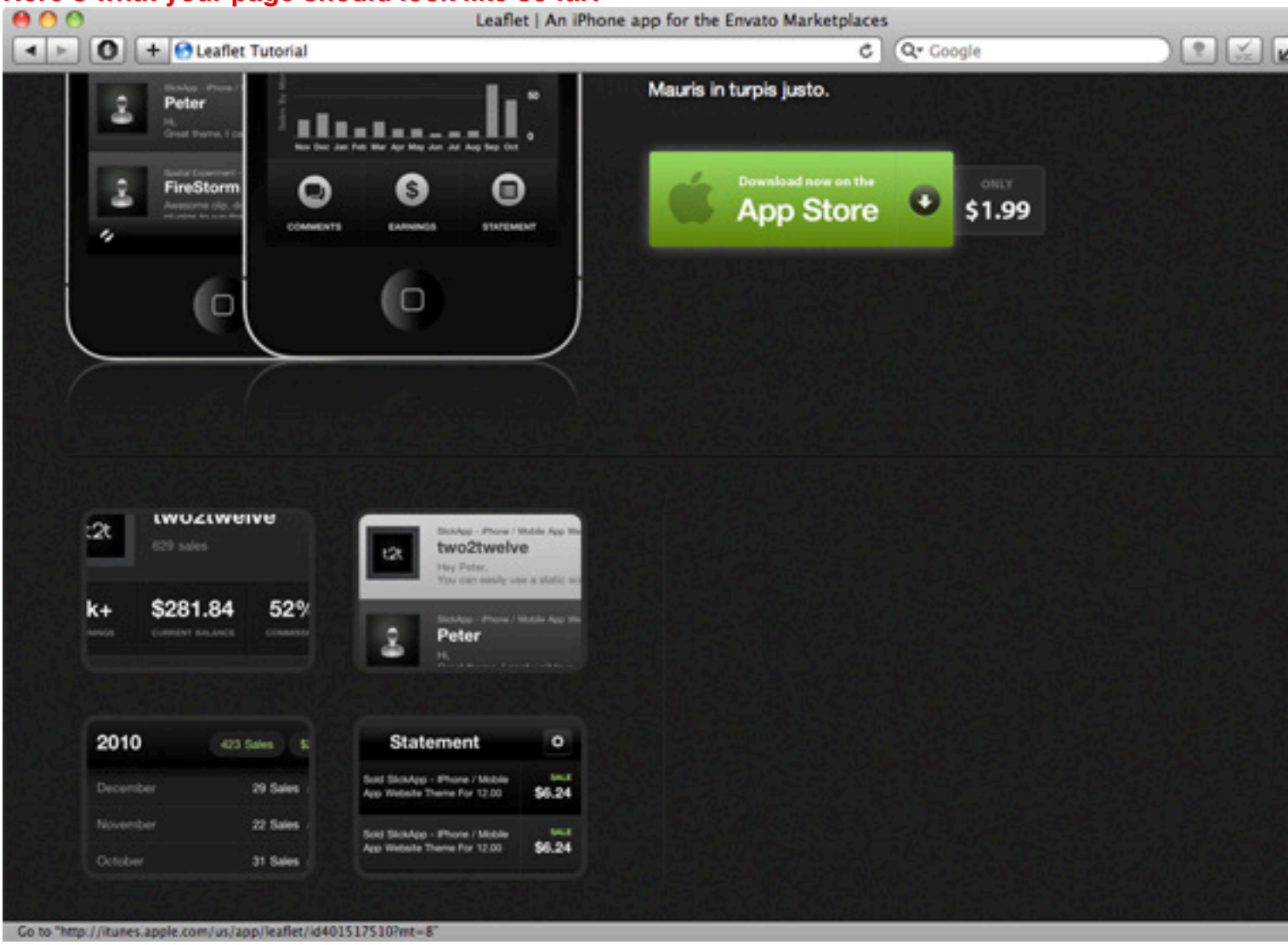
Below is the CSS for this area, along with a line-by-line breakdown:

```
01
02
03
04  div#app_details div.screenshots {
05      width: 460px;
06      float: left;
07      padding-left: 12px;
08      border-right: 1px solid #292929;
09  }
10      div#app_details div.screenshots ul{ list-style: none; }
11          div#app_details div.screenshots ul li {
12              float: left;
13              margin: 0px 30px 30px 0px;
14              position: relative;
15          }
16          div#app_details div.screenshots ul li.right { margin-right: 0; }
17          div#app_details div.screenshots ul li.bottom { margin-bottom: 0; }
18          div#app_details div.screenshots ul li span {
19              width: 176px;
20              height: 120px;
21              background: #000 url('../images/magnify.png') no-repeat center center;
22              position: absolute;
23              top: 5px;
24              left: 4px;
25              display: none;
26              -moz-border-radius: 7px;
                -webkit-border-radius: 7px;
                border-radius: 7px;
            }
```

- The first declaration, targeting `div#app_details div.screenshots`, defines a width, some padding, adds a border to the right side and floating the entire element to the left.
- Next, we have `div#app_details div.screenshots ul`. By using the `list-style: none` property, we're simply saying we don't want to use the browsers default bullets for our list.
- Then, each `<li>` in our list is set to float left (which will automatically create a grid for us). Then we space them out using margin on the right and bottom. Also, added is a`position: relative` property, which is needed for our hover state (more on that below).
- The `.right` and `.bottom` classes are here to overwrite the margin that were applied by the CSS targeting `div#app_details div.screenshots ul li`. By adding these classes, we won't encounter extra margins at the end or to the right of our list.
- Lastly, we create the style for the `<span>` tag (which will act as the hover state). This includes a `height, width` and `background` property (the magnify icon used can be found here). In

addition, we are positioning this span element over our image using the `position`, `top` and `left` properties, then finally hiding the element using the display property. If this seems a bit confusing now, it will definitely make more sense once we add the corresponding Javascript later on in the tutorial.

Here's what your page should look like so far:



Step 11 - Feature List

Next order of business is to add the list of features of our app using a simple unordered list and some custom bullets.

Below is the HTML for this section which will reside inside the `#app_details` directly below the `.screenshots` container that we've completed in Step 10.

```
01  <div class="features">
02
```

```
        <h2>Leaflet Features</h2>
        <ul>
            <li>Get push notifications for sales and comments</li>
            <li>Slick dashboard displaying your account info</li>
            <li>Check and respond to your comments</li>
            <li>View your monthly and yearly earnings breakdown</li>
            <li>View your statement list and individual entries</li>
        </ul>

        <h2>Requirements</h2>
        <ul class="requirements">
            <li>Leaflet requires iOS 3.0 or higher on an iPhone.</li>
        </ul>

    </div>
```

- We're starting out with the container `<div class="features">` that will be assigned alignment and padding properties via CSS.
- Next is a simple `h2` tag that will act as the title for our list.
- Below that, we start our first list with a `<ul>` tag and define each row with `<li>` tags and our text.
- Just like above, we add another `h2` tag and create another list with some more text, this time a list of our app's requirements.

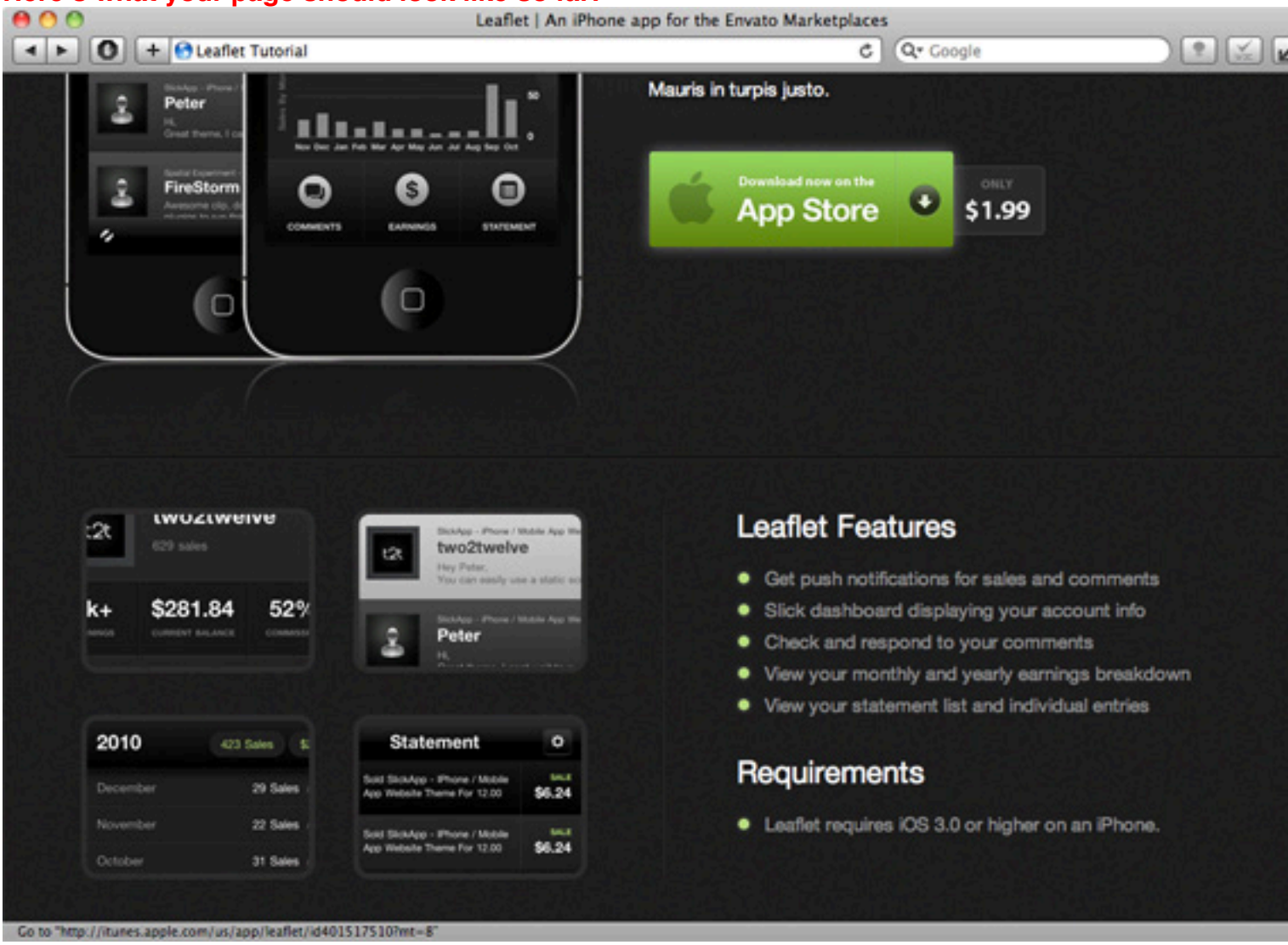
The CSS for the above section is below:

```
div#app_details div.features {
    width: 430px;
    float: right;
    padding-left: 55px;
}
    div#app_details div.features h2 {
        font-size: 24px;
        color: #fff;
        font-weight: normal;
        margin-bottom: 18px;
    }
    div#app_details div.features ul{ list-style: none; margin-bottom: 30px; }
        div#app_details div.features ul li {
            color: #8f8f8f;
            font-size: 15px;
            margin: 0px 0px 8px -3px;
            padding-left: 25px;
            background: url('../images/bullet.png') no-repeat left;
        }
```

- For `div#app_details div.features` we've defined a width, padding and floating the entire element to the right (aligned parallel to the screenshots).

- The title `h2` title tag is defined next.
- Then we add a `list-style: none` property and some margins to the `<ul>`.
- Lastly we define a few properties for each `<li>`. These include the color of the text, font size, and most importantly, setting our bullet image as the background. We're also going to use padding and margin to get the positioning of things just right.

Here's what your page should look like so far:



Step 12 - Footer Links

```
<div id="footer" class="container">
    <ul>
        <li class="twitter">
            <a href="http://twitter.com/leafletapp">Follow <span>@leafletapp</span>
        </li>
        <li class="help">
            <a href="http://getsatisfaction.com/leafletapp">For help & support head
```

```
page.</a>
        </li>
        <li class="press">
            <a href="http://media.leafletapp.com/leaflet_press_kit.zip">Download
        </li>
        <li class="copyright">
            <a href="http://trilabmedia.com">Copyright &copy; 2010 TriLab Media, LLC.<
        </li>
    </ul>
</div>
```

- First, our footer links are wrapped inside a `<ul>` tag.
- Next, each individual link is wrapped inside an `<li>` tag with a class corresponding to the link's icon. We will be using CSS to assign the correct icon to its link.
- Lastly, we define an `<a>` tag with the link's URL and relevant text. We've also wrapped certain words in `<span>` tags which we will later use CSS to add some color to.

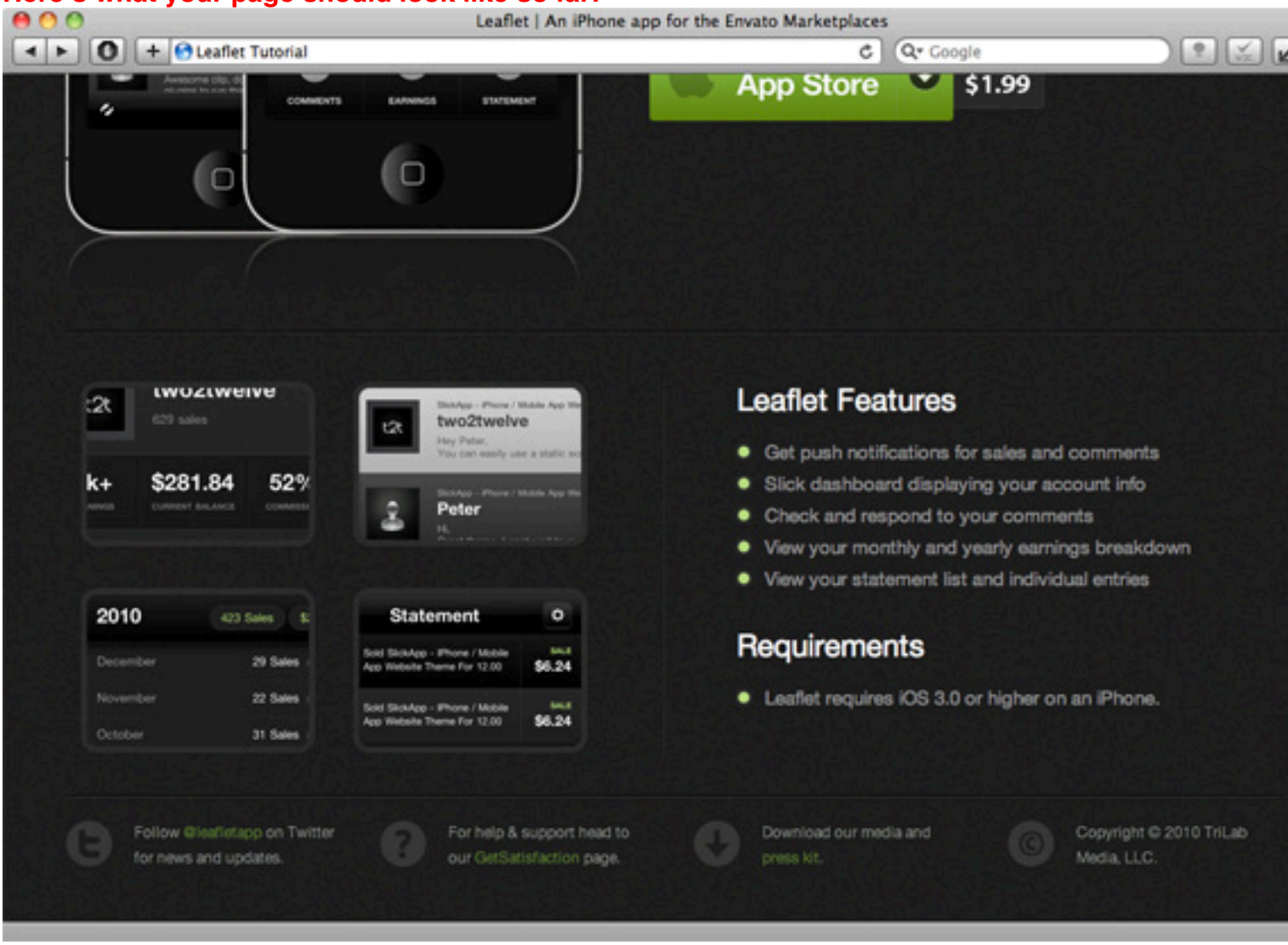
Below is the CSS for the footer section:

```
/*-------------------------------------------
Footer
-------------------------------------------*/
div#footer {
    padding: 20px 0px 80px 0px;
    background: url('../images/h_seperator.png') repeat-x top;
}
    div#footer ul { list-style: none; }
        div#footer ul li {
            float: left;
            width: 215px;
            margin-right: 33px;
        }
        div#footer ul li.copyright { margin-right: 0; }
            div#footer ul li a{
                font-size: 12px;
                color: #ccc;
                text-decoration: none;
                line-height: 20px;
                padding-left: 54px;
                height: 37px;
                display: block;
                background: url('../images/icons.png') no-repeat top left;
                opacity: 0.5;
                filter: alpha(opacity=50);
            }
            div#footer ul li a span { color: #83b546; }
            div#footer ul li a:hover{ opacity: 1; filter: alpha(opacity=100); }
                div#footer ul li.twitter a { background-position: 0px 0px; }
                div#footer ul li.help a{ background-position: 0px -37px; }
                div#footer ul li.press a{ background-position: 0px -74px; }
                div#footer ul li.copyright a{ background-position: 0px -111px; }
```

29
30
31
32

- We start out by defining styles for the `footer` which includes some padding at the top and bottom and setting the background image to the horizontal separator (the same used in the `app_details`).
- The declaration targeting `div#footer ul` tells our `<ul>` element not to use its default style.
- Next, we define styles for each of our `<li>`'s by targeting `div#footer ul li`. These styles include floating each <li> to the left so they are horizontally aligned, setting a fixed width for each, and adding some spacing using margin.
- For each `<a>` tag within the <li> tags, we've set some basic font properties for size, color, line height, etc. Then we've added some padding to the left where the icon will be located. The `height` is set to the height of one of the footer icons in our sprite. Next we're using `display: block`, since without this, our height property would be ignored. Then, we define the `background` image to our icons sprite and positioning it to top left. Lastly, we've set the `opacity` property (the `filter` property is for IE browsers only).
- In the HTML for this section, we've wrapped a few words in a span tag and using`div#footer ul li a span`, we've added a color property to highlight these wrapped words.
- When the `<a>` elements are hovered, `div#footer ul li a:hover` will set the opacity on the `<a>` element to 1.0 (100%).
- The last four lines pertain to setting the correct background position for each icon in the sprite so they correspond correctly with our link list.

Here's what your page should look like so far:



Step 13 - Javascript Enhancements

The last addition to our site will be some Javascript that will enhance a few key areas including a glowing hover effect on the app store button, a hover effect for the screenshot thumbnails, and a "lightbox" effect that appears when a screenshot is clicked.

Let's start by downloading a few Javascript libraries we're going to need, all of which can be downloaded for free using the links below:

- jQuery – jQuery is a popular Javascript framework that we will be utilizing. Once downloaded, add the file named `jquery-1.x.x.js` into your `javascripts/`directory (where 1.x.x is the current version number).

- FancyBox – FancyBox is a jQuery plugin for displaying images and other content in a "lightbox" that floats over a web page. Once downloaded, copy the directory called`fancybox` into the `javascripts/` directory.

Next, we need to append the files we just downloaded into the <head> area of our HTML file:

```
<head>
    <title>Leaflet | An iPhone app for the Envato Marketplaces</title>
    <meta "charset=utf-8" />
    <meta name="description" content="An iPhone app for the envato marketplaces." />
    <link rel="stylesheet" href="stylesheets/site.css" />
    <link rel="stylesheet" href="javascripts/fancybox/jquery.fancybox-1.3.2.css" />
    <script src="javascripts/jquery-1.4.4.js"></script>
    <script src="javascripts/fancybox/jquery.fancybox-1.3.2.pack.js"></script>
    <script src="javascripts/site.js"></script>
</head>
```

- The **first** new line added is the `stylesheet` for the FancyBox plugin.
- **Next**, we've imported the jQuery JavaScript library.
- The FancyBox plugin is added next.
- **Lastly**, we've imported a custom JavaScript file we are going to create below.

Create a file in the `javascripts/` directory (INSIDE your LEAFLET FOLDER create a folder called JAVASCRIPTS and place the code inside it.

`site.js` and in this file paste the following code:

```
$(document).ready(function() {

    if($.support.opacity) {
        $('.app_store').hover(function() {
            $('a', this).stop(true, true).animate({ opacity: 1.0 }, 300);
        }, function() {
            $('a', this).stop(true, true).animate({ opacity: 0 }, 300);
        });
    }

    var screenshots = $('div.screenshots');

    screenshots.find('a').fancybox({
        'speedIn' : 400,
        'speedOut' : 400,
        'overlayShow' : false,
        'padding' : 8,
        'centerOnScroll' : true
    });

    screenshots.find('li').hover(function() {
        $('span', this).stop(true, true).fadeToggle(300);
    });

});
```

```
21
22
23
24
25
```

Let's deconstruct the above code line-by-line:

- The first bit, `$(document).ready`, is a function that encompasses all of our JavaScript code. This function ensures our page is loaded and ready before the code it contains is executed.
- Next up, we create the function for creating a "glow" hover effect on the app store button. We first check to see if the browser supports opacity animation using`if($.support.opacity)`. Then we're selecting the button element with`$('.app_store')`, attaching a `hover()` event and finally executing the `animate`function to fade our glowing button in / out when hovered.
- We use the `var` tag to create a *global variable* for our screenshots container. This will create a JavaScript instance of our screenshots for later use in our code. If we choose not to define a global variable for our screenshots div, we would have to manually select the element ,using `$('div.screenshots')`, each time we need to manipulate it or something within it.
- Initializing the FancyBox plugin is up next. By using `screenshots.find('a')` we are automatically selecting all of the <a> tags in our screenshots area and applying the FancyBox plugin to them. We have also defined some options for the plugin here.
- The last block of code creates the "hover" effect for each screenshot. Every time a screenshot is hovered (using `screenshots.find('li').hover`) we fade in the element that's inside each of the screenshot 's.

Finished!