Ajax Intro Lab

In this lab, we're going to make an Ajax call and get some values back in a real-world type situation.[1]

1. Start off by running the web site in Visual Studio. Just go Debug-Start Debugging or hit F5.
2. Hit the "Product" button/tab in the upper right corner. You should see a list of products.

We eventually want to get some more details for the user but we should them only when the user needs them. Our goal in this lab will be to fetch those details when the user hovers over the product name. When he does, we'll retrieve the details and display them in a tooltip-looking div.

3. First, edit the html page in Visual Studio. It is Views/Product/Index.cshtml. Look through it.
4. Let's add the <div> where the detail info will be displayed. Just put this in the page somewhere near to top of the page tag but outside anything else:

```
<div id='productDetails'
  style='visibility:hidden; position:absolute;
  z-index:10; background-color: ffffcc;'></div>
```

   This will make it invisible and dynamically positioned.

5. Add some new JavaScript to your page:

```
NodeList.prototype.forEach = Array.prototype.forEach;
var cells = document.querySelectorAll('[data-product-id]');
cells.forEach(function (td) {
  var id=td.getAttribute('data-product-id');
  td.addEventListener('mouseenter', function () {
    showDetailsDiv(id);
  });
  td.addEventListener('mouseleave', function () {
    hideDetailsDiv();
  });
  td.addEventListener('mousemove', function (e) {
    followMouse(e);
  });
});
```

6. Run and test. It should run fine until you mouseover the product name. Then you should see an error that showDetailsDiv is undefined. No problem, we can fix that.

Write the functions

7. Add three JavaScript functions, one called showDetailsDiv which accepts a productID, one called hideDetailsDiv which receives no parameters, and a third called followMouse which receives an event object.
8. showDetailsDiv should do this for now:

```
var div = document.getElementById('productDetails');
div.innerHTML = "Details for " + productId;
div.style.visibility = "visible";
```

9. Run and test. At this point, your invisible <div> should suddenly show up when your mouse rolls over any product description.
10. In the hideDetailsDiv function, do this:

[1] So real-world in fact that we're using ASP.NET MVC, Microsoft's framework for creating dynamic web pages. This course is not about MVC and you're not expected to know it. So don't worry if you don't understand any of the non-HTML and non-JavaScript parts of this lab. And, hey, you may just learn a little MVC by exposure, so enjoy! (We do offer an ASP.NET MVC course. You should sign up for it).

```
var div = document.getElementById('productDetails');
div.style.visibility = "hidden;";
```
11. Run and test again. Now it should appear when you mouse over any description and disappear when you mouse out.
12. In the followMouse(e) function, do this:
```
var div = document.getElementById('productDetails');
div.style.top = (e.pageY + offsetY) + "px";
div.style.left = (e.pageX + offsetX) + "px";
```
13. You'll probably want to adjust offsetX and offsetY to make it look good. Run and test until you get it looking good.

Getting some real data
14. We've provided a service that supplies product info. Go ahead and point your browser at /Product/GetProductInfo/10. See that stuff? It's JSON data being returned. Try a few different productIDs to see the data change in your browser.

Let's use that data to populate our div.

15. Go into the showDetailsDiv(productID) function.
16. Instantiate an XMLHttpRequest().
17. Write the request using the 'GET' method, point it at '/Product/GetProductInfo/' + productID, and leave it asynchronous. (Hint: use the open() method).
18. Assign a new function called displayDetails to the onreadystatechange event.
19. Finally send it using the send().

Writing the callback
20. Create your new callback function. We're naming it *displayDetails*.
21. Check that readyState is 4 and that status is 200. If either is not true, exit the function.
22. But if they're both true, pull the response out of the responseText property.
23. Grab a reference to the div using getElementById.
24. Set its innerHTML property to some nicely formatted text that shows the category and supplier of the product.
25. Run and test.

26. Bonus!! If you have some time, add some other details to your tooltip to spice it up.