活动 5.2：使用 Knockout 创建 ShoppingCart 网页

问题陈述

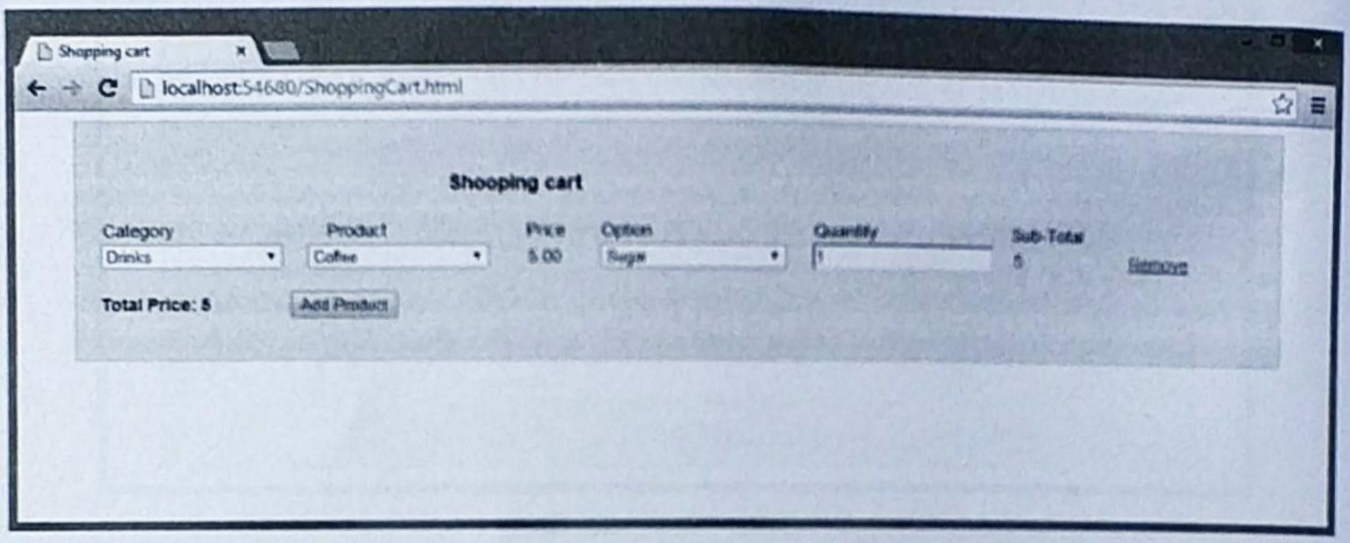
Cosmo 是 San Diego 一家快餐零售店。Cosmo 的顾客走到零售店订购并带走快餐。为了扩展业务和客户，Cosmo 的管理层决定创建 Web 应用程序以使顾客可以在线下订单。

为了实现此目标，管理层与软件开发公司 EarnestIT 分享了其需求。EarnestIT 的项目主管 James 和他的项目团队一起收集了 Cosmo 的需求。您是 EarnestIT 项目团队的一员，被分派了创建 Shopping Cart 网页的任务。

此页面应该基于以下准则：

- 顾客可以选择所需的食品以下订单。
- 顾客可以在购物车中动态添加另一食品。
- 顾客可从购物车删除所选食品。

应该显示 ShoppingCart 网页，如下图所示。



ShoppingCart 网页

先决条件：要执行此活动，需要使用 **ShoppingCart.zip** 文件。

解决方案

要创建 ShoppingCart 网页，需要执行以下任务：

1. 创建 ShoppingCart.html 文件。
2. 创建 StyleSheet.css 文件。
3. 创建 ScriptFile.js 文件。
4. 查看 ShoppingCart 页面。

任务 1：创建 ShoppingCart.html 文件

要创建 **ShoppingCart** 页面，需要执行以下步骤：

1. 在 Microsoft WebMatrix 中打开 **ShoppingCart** 网站。
2. 创建新文件 **ShoppingCart.html**。
3. 将 **ShoppingCart.html** 文件的现有代码段替换为以下代码段：

```
<!DOCTYPE html>
<html lang="en">
    <head>
     <meta charset="utf-8" />
     <title>Shopping cart</title>
 <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></scri
pt>
<script src="http://ajax.aspnetcdn.com/ajax/knockout/knockout-2.2.1.js"
type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="CSS/StyleSheet.css"></link>
<script src="js/ScriptFile.js"> </script>
</head>
```

4. 添加以下代码段以创建报头：

```
 <body>
        <div class="liveExample">
            <div id="hh1">
                <h2>Shooping cart</h2>
            </div>
 <span class="space1">Product</span> <span class="space2">Category</span>
 <span class="space3">Price</span><span class="space4">Option</span>
 <span class="space5">Quantity</span><span class="space6">Sub-Total</span>
<div>
```

5. 添加以下代码段以将 `Category()` 方法与 `<select>` 元素绑定：

```
<!-- ko foreach:lines -->
<!-- ko foreach:$parent.menus -->
<select style="width:150px" data-bind='options:,categories,
optionsText:"name", value:$parent.category' ></select>
```

6. 添加以下代码段以将 `Product()` 方法与 `<select>` 元素绑定：

```
<!-- /ko -->
<!-- ko with: category -->
   
<select style="width:150px" data-bind='options: products, optionsText:"name",
value:$parent.product'></select>
```

7. 添加以下代码段以将 price 变量与 元素绑定：

```
<!-- /ko -->
<!-- ko with: product -->
 &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp;<span data-bind="text: price"></span>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
```

8. 添加以下代码段以将 Option() 方法与 <select> 元素绑定：

```
<select style="width:155px;" data-bind='options: options,
optionsText:"name"'></select> &amp;nbsp;&amp;nbsp;&amp;nbsp;
```

9. 添加以下代码段以将 quantity 变量与 <input> 元素绑定：

```
<input data-bind="value:$parent.quantity, valueUpdate:'keyup'" />
<!-- /ko -->
```

10. 添加以下代码段以将 subtotal 方法与 元素绑定：

```
&amp;nbsp;&amp;nbsp;&amp;nbsp;<span style="width:155px;" data-bind="text:
subtotal"></span>&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp;
&amp;nbsp;&amp;nbsp;&amp;nbsp;
```

11. 添加以下代码段以将 removeLine 方法与 <a> 元素绑定：

```
&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp;<a href="#" data-
bind="click:$parent.removeLine">Remove</a><br />
<!-- /ko -->
```

12. 添加以下代码段以将 grandTotal 方法与 元素绑定：

```
</div><p>
<b>Total Price:<span data-bind="text: grandTotal">
</span></b>&amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp; &amp;nbsp;&amp;nbsp;&amp;nbsp;
&amp;nbsp;&amp;nbsp;&amp;nbsp;
```

13. 添加以下代码段以将 addLine 方法与 <button> 元素绑定：

```
<button data-bind="click: addLine">Add Product</button>
   </p>
   </div>
   </body>
</html>
```

14. 保存 **ShoppingCart.html** 文件。

任务 2：创建 StyleSheet.css 文件

要创建 **StyleSheet.css** 文件，需要执行以下步骤：

1. 在 **css** 文件夹中创建文件 **StyleSheet.css**。

2. 将现有代码段替换为以下代码段：

```
body { font-family: arial; font-size:14px; }
```

3. 添加以下代码段以将 CSS 样式应用到选择器 ID 为 liveExample 的 元素：

```
.liveExample { padding:1em; background-color:#EEEEDD; border:1px solid #CCC;
max-width:972px; margin:0 auto; padding:20px }
```

4. 添加以下代码段以将 CSS 样式应用到 <input> 元素：

```
.liveExample input { font-family:Arial; }
```

5. 添加以下代码段以将 CSS 样式应用到选择器 ID 为 hh1 的 元素：

```
#hh1{ width:400px; margin:20px auto}
```

6. 添加以下代码段以将 CSS 样式应用到各种类选择器以设置 padding 和 margin 属性：

```
.space1{ padding:0 1px;}
.space2{ margin:0 122px;}
.space3{ margin-left:-10px;}
.space4{ margin-left:30px;}
.space5{ margin-left:135px;}
.space6{ margin-left:115px;}
```

7. 保存 **StyleSheet.css** 文件。

任务 3：创建 ScriptFile.js 文件

要创建 **ScriptFile.js** 文件，需要执行以下步骤：

1. 在 **js** 文件夹中创建文件 **ScriptFile.js**。
2. 添加以下代码段以防止 jQuery 代码在文档就绪前运行：

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

3. 添加以下代码段以创建 Option() 函数，该函数用于将 quantity 变量创建为 observable：

```
function Option(name, quantity) {
var self = this;
self.name = name;
self.quantity = ko.observable(quantity);
}
```

在上述代码段中，name 参数的值传递到本地变量 name。此外，quantity 变量声明为 observable。

4. 添加以下代码段以创建 Product() 函数，该函数获取包含选项的可观察数组的一个产品：

```
function Product(name, price, quantity, options) {
var self = this;
self.name = ko.observable(name);
self.price = price;
self.quantity = ko.observable(quantity);
self.options = ko.observableArray(ko.utils.arrayMap(options, function(option)
{
return new Option(option.name, option.quantity);
}));
self.addOption = function(name, quantity) {
self.options.push(new Option(name, quantity));
};
}
```

5. 添加以下代码段以创建 Category() 函数，该函数获取包含产品的可观察数组的一个类别：

```
function Category(name, products) {
var self = this;
self.name = name;
 self.products = ko.observableArray(ko.utils.arrayMap(products,
function(product) {
return new Product(product.name, product.price, product.quantity,
product.options);
}));
}
```

6. 添加以下代码段以创建 Menu() 函数，该函数包含类别的 observableArray：

```
function Menu(categories) {
var self = this;
self.categories = ko.observableArray(ko.utils.arrayMap(categories,
function(category) {
return new Category(category.name, category.products);
}));
self.addCategory = function(name) {
self.categories.push(new Category(name));
};
}
```

7. 添加以下代码段以创建 CartLine() 函数，它显示购物车中的商品和每一行的小计：

```
function CartLine() {
                    var self = this;
                    self.category = ko.observable();
                    self.product = ko.observable();
                    self.quantity = ko.observable(1);
                    self.subtotal = ko.computed(function() {
return self.product() ? self.product().price * parseInt("0" + self.quantity(),
10) :0;
                    });
                    self.category.subscribe(function() {
                        self.product(undefined);
                    });
```

```
}
```

8. 添加以下代码段以创建 ViewModel() 方法，它显示此屏幕的整体 ViewModel 以及初始状态：

```
function ViewModel(menus) {
var self = this;
self.lines = ko.observableArray([new CartLine()]);
self.menus = ko.utils.arrayMap(menus, function(menu) {
return new Menu(menu.categories);
});
self.addLine = function() {
self.lines.push(new CartLine())
};
self.removeLine = function(line) {
self.lines.remove(line)
};
self.grandTotal = ko.computed(function() {
var total = 0;
$.each(self.lines(), function() {
total += this.subtotal()
})
return total;
});
}
ko.applyBindings(new ViewModel(data));

});
```

9. 添加以下代码段以创建一个数组，它包含类别、产品、价格和可用选项的相关信息：

```
var data = [{
       "categories":[
           {
           "name":"Drinks",
           "products":[{
                   "name":"Coffee",
               "price":"5.00",
               "quantity":"0",
               "options":[{
                   "name":"Sugar",
                   "quantity":"0"
               },
               {
                   "name":"Cream",
                   "quantity":"0"
               }
           ]},
       {
           "name":"Tea",
           "price":"3.00",
           "quantity":"0",
           "options":[{
               "name":"Sugar",
               "quantity":"0"
```

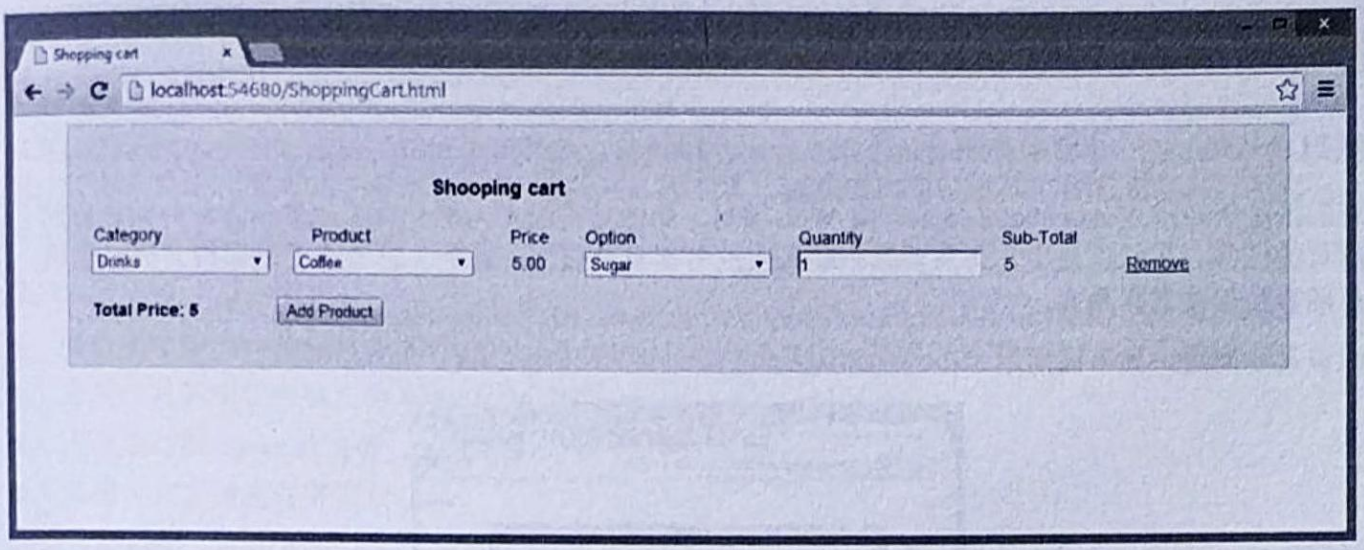
```
            },
            {
                "name":"Milk",
                "quantity":"0"
            }
            ]}
        ]},
    {
"name":"Food",
"products":[
    {
    "name":"Donut",
                "price":"6.00",
                "quantity":"0",
                "options":[
                    {
                    "name":"Chocolate",
                    "quantity":"0"},
                {
                    "name":"Dutchie",
                    "quantity":"0"}
                ]},
            {
                "name":"Muffin",
                "price":"7.00",
                "quantity":"0",
                "options":[
                    {
                    "name":"Bran",
                    "quantity":"0"},
                {
                    "name":"Carrot",
                    "quantity":"0"}
                ]}
            ]}
        ]}];
```

10. 保存 **ScriptFile.js** 文件。

任务 4：查看 ShoppingCart 页面

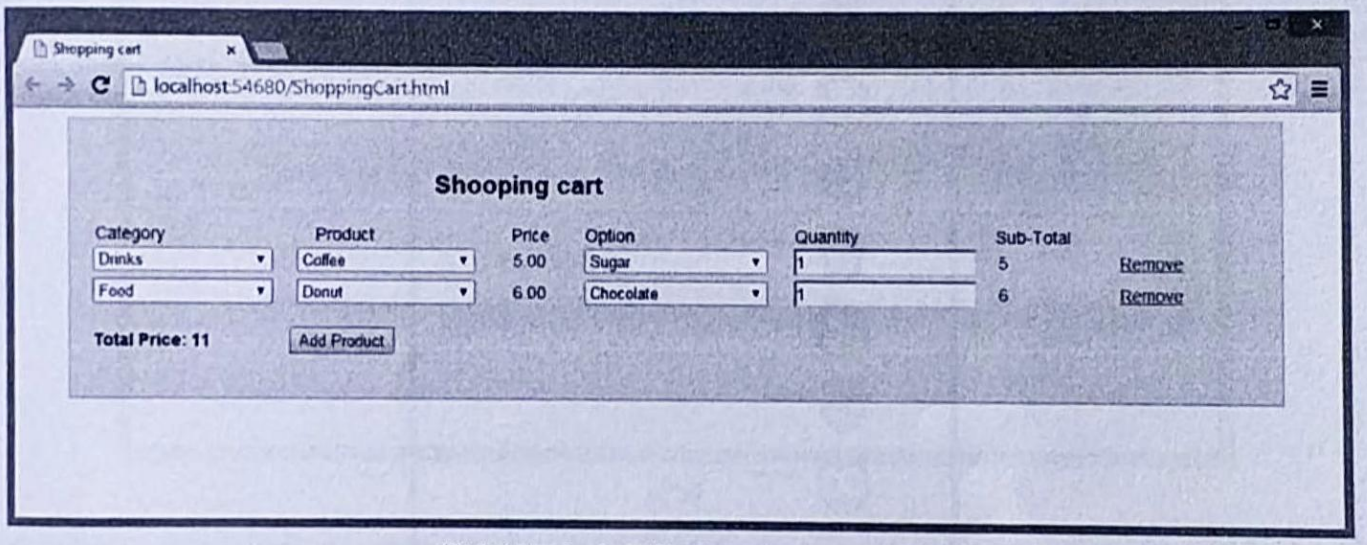
要查看 **ShoppingCart** 页面，需要执行以下步骤：

1. 右击 **ShoppingCart.html** 页面，并选择 **Launch in browser** 选项。应该显示 **ShoppingCart** 网页，如下图所示。



ShoppingCart 网页

2. 单击 **Add Product** 按钮以添加新食品，并从 **Category** 选项中选择 **Food**。
   应该显示 **ShoppingCart** 网页，如下图所示。



带有新添加食品的 ShoppingCart 网页