项目代码风格要求

V1. 0. 20130605

作者：张子阳

191811496@qq.com

JimmyZhang.cnblogs.com

1. C# 代码风格要求

1.1 注释

类型、属性、事件、方法、方法参数，根据需要添加注释。

如果类型、属性、事件、方法、方法参数的名称已经是自解释了，则不需要加注释；否则必须添加注释。

当添加注释时，添加方式如下图所示：

```
namespace ConsoleApp {
    /// <summary>
    /// 产品售光时被调用的委托
    /// </summary>
    public delegate void SalesOutEventHandler();

    /// <summary>
    /// 产品类，描述产品的基本信息
    /// </summary>
    public class Product {

        /// <summary>
        /// 定义产品被售光时的处理逻辑
        /// </summary>
        public event SalesOutEventHandler OnSalesOut;

        /// <summary>
        /// 根据产品Id查找产品
        /// </summary>
        /// <param name="id">产品的Id</param>
        /// <returns>符合此Id的产品实例，当不存在该产品时，返回null</returns>
        public Product GetProductById(int id) {
            return new Product();
        }

        /// <summary>
        /// 产品类型，描述产品种类，参考《需求说明》
        /// </summary>
        public enum ProductType {

        }
    }
}
```

1.2 类型（类、结构、委托、接口）、字段、属性、方法、事件的命名

优先考虑英文，如果英文没有合适的单词描述，可以使用拼音，使用中文是不符合要求的。。

唯一可以使用中文的地方是枚举的枚举项，枚举项实际已经不属于本节标题的范畴了。这里只是放到一起说明，如下图所示：

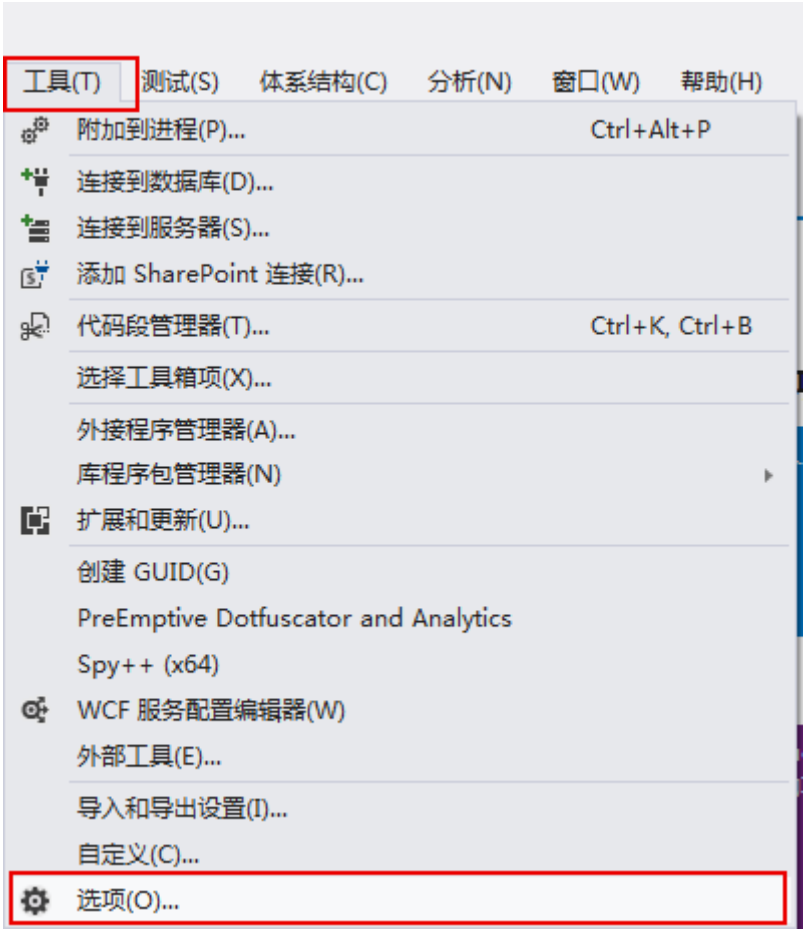
```
public enum ProductType {
    未定义 = 0,
    服装 = 1,
    数码 = 2
}
```

1.3 不使用缩写

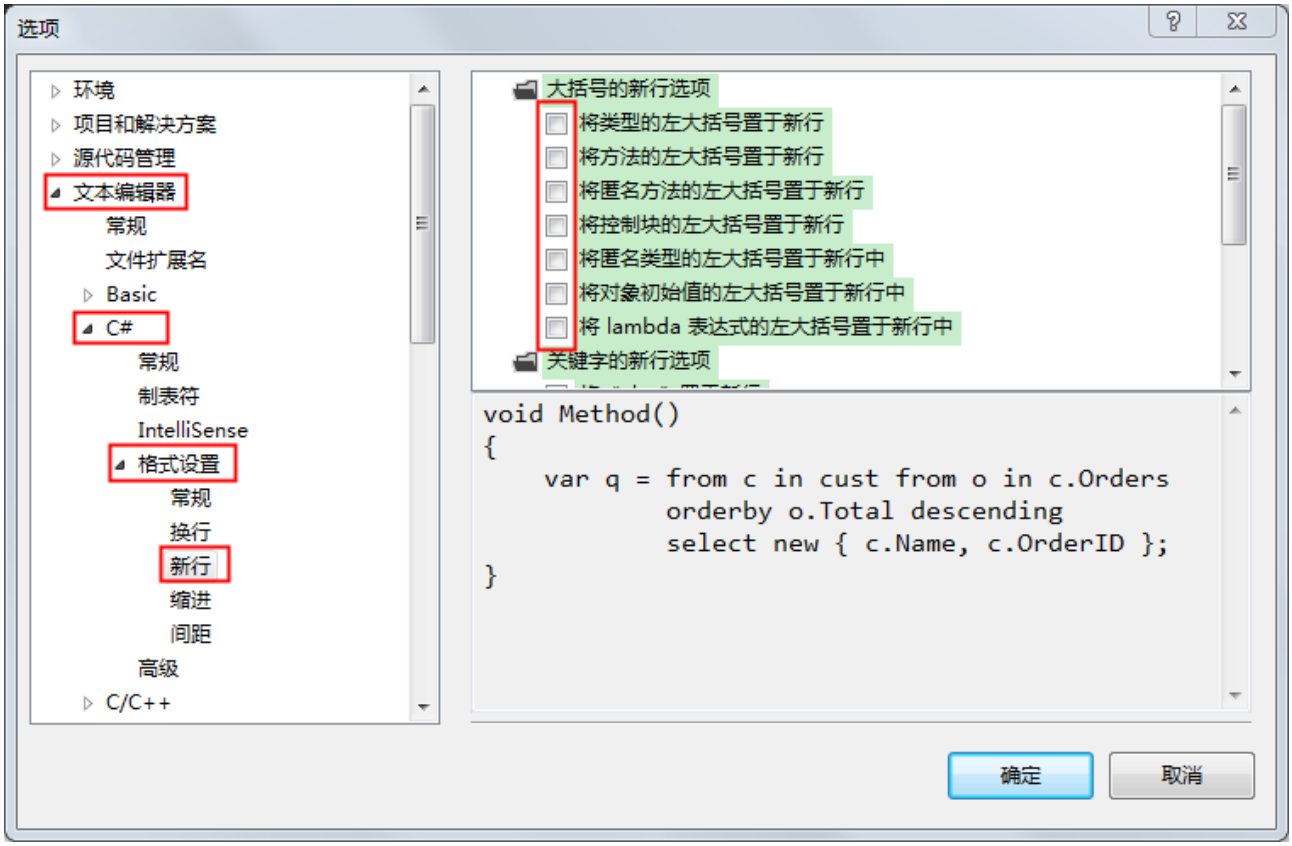
所有类型、方法、参数、变量的命名不得使用缩写，包括大家熟知的缩写，例如 msg。

1.4 代码使用半展开

第一步，打开 Visual Studio，进入“工具”，“选项. . .”，如下图所示：



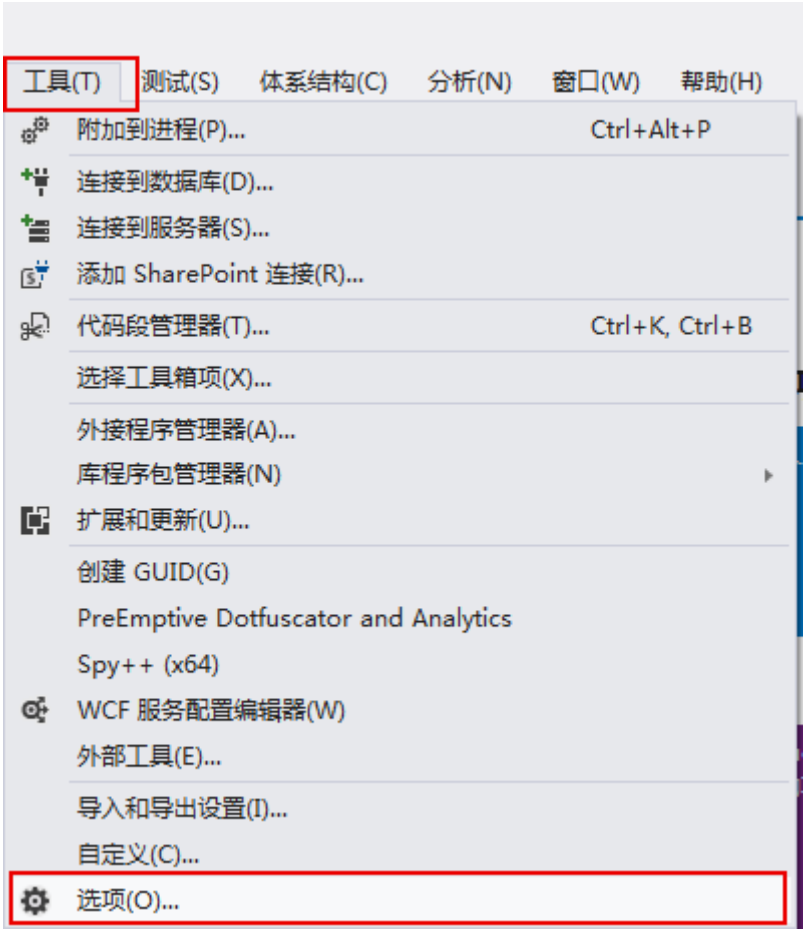
第二步，进入“文本编辑器”，“C#”，“格式设置”，“新行”，取消掉右侧所有复选框中的对号，如下图所示：



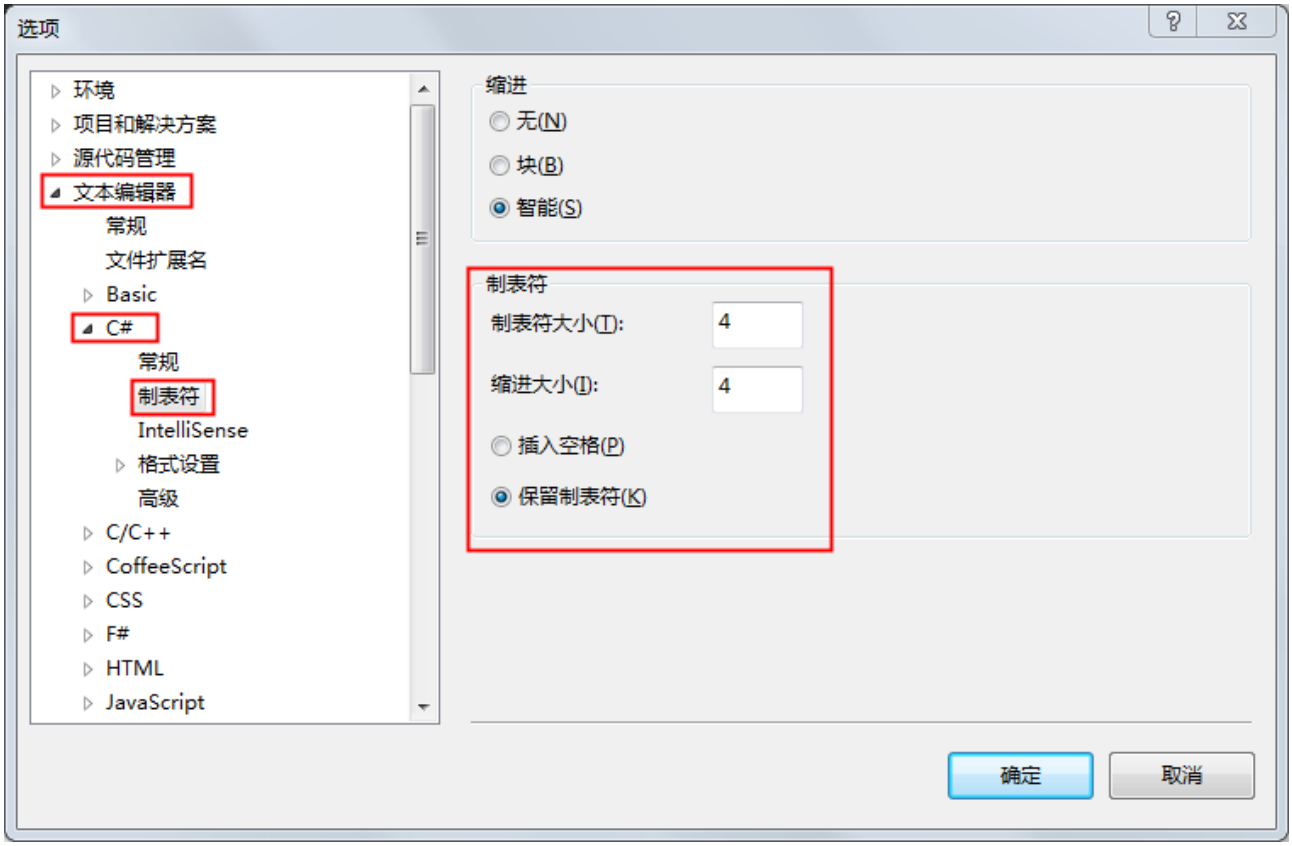
第三步，点击“确定”，完成设置。

1.5 使用 Tab 作为缩进，并设置缩进大小为 4

第一步，打开 Visual Studio，进入“工具”，“选项...”，如下图所示：



第二步，进入“文本编辑器”，“C#”，“制表符”，如下图所示，设置制表符。



第三步，点击“确定”，完成设置。

1.6 一个.cs 源文件至多定义两个类型

如果两个类型的关系是紧密相关的，比如 产品、产品类型，此时 Product 类，和 ProductType 枚举可以定义在同一个 Product.cs 文件中。

但不能在一个.cs 文件中出现两个不相关的类型定义，例如将 Product 类和 Reseller 类（分销商）定义在一个 BasicInfo.cs 文件中。

1.7 类型名称和源文件名称必须一致

当类型命名为 Product 时，其源文件命名只能是 Product.cs。

1.8 所有命名空间、类型名称使用 Pascal 风格（单词首字母大写）

如下图所示，红色标记的为使用 Pascal 风格的类型：

```
namespace ConsoleApp {

    public delegate void SalesOutEventHandler();

    public class Product {

        public event SalesOutEventHandler OnSalesOut;

        public Product GetProductById(int id) {
            return null;
        }

       private enum ProductType { }
    }
}
```

注意 ProductType 是私有类型，不管类型是公有的还是私有的，其命名总是采用 Pascal 风格。

1.9 本地变量、方法参数名称使用 Camel 风格（首字母小写，其后每个单词的首字母大写）

红色标记的为使用 Camel 风格的变量或者方法参数：

```
public class Product {
    public float Price { get; set; }

    public float GetProductPrice(int productId) {
        Product item = this.GetProductById(productId);
        float price = item.Price;

        if (price < 100) {
            return price;
        } else {
            return price * 0.95f;
        }
    }

    public Product GetProductById(int id) {
        return new Product();
    }
}
```

1.10 私有方法、受保护方法，仍使用 Pascal 风格命名

示例代码如下：

```
public class Product {
    public float Price { get; set; }

    private float GetProductPrice(int productId) {
        return 0;
    }

    protected Product GetProductById(int id) {
        return null;
    }
}
```

1.11 如果 if 语句内容只有一行，可以不加花括号，但是必须和 if 语句位于同一行

范例 1.9 中的 if 判断实际上与下面的语句是等效的：

```
if (price >= 100) return price * 0.95f;

return price;
```

1.12 调用类型内部其他成员，需加 this；调用父类成员，需加 base

示例代码如下：

```
public class ProductBase {
    protected Product GetProductById(int id){
        return new Product();
    }
}

public class Product : ProductBase {
    public float Price { get; set; }

    public float GetProductPrice(int id) {

        Product item = base.GetProductById(id);

        return this.CalculatePrice(item);
    }

    private float CalculatePrice(Product item) {
        float price = item.Price;
        if (price >= 100) return price * 0.95f;
        return price;
    }
}
```

1.13 类型内部的私有和受保护字段，使用 Camel 风格命名，但加“_”前缀

代码示例如下：

```
public class Product {

    protected float _id;

    private float _price;
}
```

1.14 不能出现公有字段

如果需要公有字段，使用属性进行包装。

1.15 类型成员的排列顺序

类型成员的排列顺序自上而下依次为：

字段：私有字段、受保护字段

属性：私有属性、受保护属性、公有属性

事件：私有事件、受保护事件、公有事件

构造函数：参数数量最多的构造函数，参数数量中等的构造函数，参数数量最少的构造函数

方法：重载方法的排列顺序与构造函数相同，从参数数量最多往下至参数最少。

```
public class Product {
    private int _field1;
    protected int _field2;

    private int _property1 { get; set; }
    protected int _property2 { get; set; }
    public int Property3 { get; set; }

    private SalesOutEventHandler _event1;
    protected SalesOutEventHandler _event2;
    public SalesOutEventHandler Event3;

    public Product(int param1, int param2) { }
    public Product(int param1) { }
    public Product() { }

    public Product GetProduct(int id, string area) { return null; }
    public Product GetProduct(int id) { return null; }
    public Product GetProduct() { return null; }
}
```

1.16 委托和事件的命名

委托以 EventHandler 作为后缀命名，例如 SalesOutEventHandler。

事件以其对应的委托类型，去掉 EventHandler 后缀，并加上 On 前缀构成。

例如，对于 SalesOutEventHandler 委托类型的事件，其事件名称为：OnSalesOut。

示例代码如下：

```
public delegate void SalesOutEventHandler();

public class Product {
    public SalesOutEventHandler OnSalesOut;
}
```

1.17 返回 bool 类型的方法、属性的命名

如果方法返回的类型为 bool 类型，则其前缀为 Is、Can 或者 Try，例如：

```
public class Product {

    public bool IsSalesOut {
        get { return true; }
    }

    public bool TrySale(int id) {
        return true;
    }

    public bool CanSale(int id) {
        return true;
    }
}
```

1.18 常见集合类型后缀命名

凡符合下表所列的集合类型，应添加相应的后缀。

说明	后缀	示例
数组	Array	int[] productArray
列表	List	List<Product> productList
DataTable/HashTable	Table	HashTable productTable
字典	Dictionary	Dictionay<string,string> productDictionary
EF 中的 DbSet /DataSet	Set	DbSet<Product> productSet

1.19 常见后缀命名

凡符合下表所列的局部变量、方法参数、字段、属性，均需添加相应的后缀。

说明	后缀	示例	示例说明
费用相关	Cost	ShipCost	运输费
价格相关	Price	ProductUnitPrice	产品单价
消息相关	Message（弃用 Note）	SuccessMessage	成功消息
日期相关	Date（弃用 Time）	OrderDate	下单日期
计数、数量相关	Count（弃用 Time）	LoginCount	登录次数

链接地址相关	Url	BlogUrl	博客链接
图片相关	Image	SignImage	签名图片
金额相关	Amount	PrepaidAmount	预付款
点数、积分相关	Point	MemberPoint	会员积分
记录、日志相关	Record（弃用 Log）	ErrorRecord	错误记录
配置相关	Config	DataBaseConfig	数据库配置
状态相关	Status	OrderStatus	订单状态
模式、方式相关	Mode	OpenMode	打开方式
种类相关	Category / Type 二选一	UserCategory	用户种类
工厂类相关	Factory	ConnectionFactory	连接工厂
启用相关	Enabled	ExportEnabled	开启导出
流相关	Stream	UploadStream	上传流
读取器相关	Reader	ExcelReader	Excel 读取器
写入器相关	Writer	ExcelWriter	Excel 写入器
适配器相关	Adapter	IntroOPAdapter	IntroOP 适配器
提供器相关	Provider	MemebershipProvider	会员信息提供器
包装器相关	Wrapper	ProductWrapper	Product 包装器
连接相关	Connection	ExcelConnection	Excel 连接

1.20 常见类型命名

凡存在下表中的类型，需采用下表指定的名称命名。

类型	命名	类型	命名
客户	Customer	分销商	Reseller
零售商	Retailer	经销商/批发商	Dealer
用户	UserInfo（User 为数据库关键字）	订单	OrderInfo（Order 为数据库关键字）
供应商	Supplier	管理员	Admin
密码	Password	会员	Member
评论	Remark（弃用 Comment）	文章	Article
新闻	News	发票	Invoice
导入	Import	导出	Export

公司、企业	Company（弃用 Enterprise）	产品	Product
省份	Province	城市	City
区县	District	地址	Address
角色	Role（弃用 Group）	权限	Authority（弃用 Permission）
仓库	Warehouse	工厂	Plant
登录	Login（弃用 SignIn）	登出	LogOut（弃用 SignOut）
创建	Create（弃用 Add）	编辑	Edit
更新	Update	删除	Remove（弃用 Delete）
照片	Photo	图片	Image

1.21 常见字段、属性命名

字段、属性种类比较繁杂，因此仅列出最常用的几项。

类型	名称	类型	名称
Id（int 型）	Id（“d”小写，弃用 ID）	GuidId（Guid 型）	Id
Name	名称	Title	标题
Remark	备注、描述（弃用 Memo、Description）	Category	种类（弃用 Class、Type）
Linkman	联系人		

2. XHTML 代码风格要求

2.1 如果 XHTML 标记有层次，那么代码也要有层次

下面是书写符合要求的例子：

```
<table>
    <tr>
        <td> </td>
        <td> </td>
    </tr>
    <tr>
        <td> </td>
        <td> </td>
    </tr>
</table>
```

下面是书写 **不符合要求** 的例子：

```
<table>
    <tr>
        <td> </td><td> </td>
    </tr>
    <tr>
    <td> </td>
    <td> </td>
</tr>
</table>
```

2.2 所有标记必须闭合

示例代码如下：

```
<br />
<img src="..." alt="" />
```

2.3 如果标记中间代码超过 20 行，则应在标记末尾加注标识

标注方式如下：

```
<div id="container">
    <!-- 省略20行代码 -->
</div><!-- container -->
<div class="footer">
    <!-- 省略20行代码 -->
</div><!-- footer -->
```

3. CSS 代码风格要求

3.1 代码使用半展开

设置方法参考 1.4 节。

3.2 使用 Tab 作为缩进，并设置缩进大小为 4

设置方法参考 1.5 节。

3.3 代码注释

注释主要说明该样式应用于页面的哪个部分，而非说明样式的应用效果，代码注释风格如下所示：

```
/* 通用设置
-----------------------------------*/
* {
    padding:0;
    margin:0;
}

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

3.4 代码编写

每一个样式设置必须独占一行，不能位于同一行，下面是符合要求的写法：

```
#container {
    font-size:14px;
    border:1px solid red;
}
```

下面是 **不符合要求** 的写法：

```
#container {
    font-size:14px; border:1px solid red;
}
```

3.5 嵌入式样式的比例不超过样式表代码总量的 10%

嵌入式样式为直接写在 HTML 标记内部的样式，如下图所示：

```
<body style="background:#fff;">
```

3.6 内联式样式的比例不超过样式表代码总量的 30%

内联式样式为写在<head></head>中的样式，如下图所示：

```
<head>
<title>CSS</title>

<style type="text/css">
#container {
    font-size:14px;
    border:1px solid red;
}
</style>

</head>
```

内联式样式，**不能** 写在<body></body>之间。

3.7 外联式样式表的比例不少于样式表代码总量的 60%

外联式样式表为写在.css 文件中的样式，通过 link 引入到 XHTML 页面中，如下图所示：

```
<head>
<title>CSS</title>
<link href="StyleSheet.css" rel="stylesheet" />

</head>
```

4. JavaScript 代码风格要求

4.1 代码使用半展开

设置方法参考 1.4 节。

4.2 使用 Tab 作为缩进，并设置缩进大小为 4

设置方法参考 1.5 节。

4.3 代码注释

代码注释需要说明“函数功能”、“入口参数”、“返回值”，注释范例如下：

```
// 提交表单
// formId: 表单Id
// 提交成功返回true，否则返回false
function postForm(formId) {
    return false;
}
```

其中第一行说明函数功能；第二行说明入口参数；最后一行说明返回值

4.4 不得出现内嵌式代码

内嵌式代码是指写在 XHTML 标记中的 JavaScript 代码，下面的写法是 不符合要求 的：

```
<input type="button" onclick="alert('hello')" value="Click" />
```

4.5 内联式代码占 JavaScript 的总量不得超过 40%

内联式代码是指写在<head />或者<body />之间的代码：

```
<head>
<title>CSS</title>

<script type="text/javascript">
function postForm(formId) {
    return false;
}
</script>

</head>
```

4.6 外联式代码占 JavaScript 的总量至少为 60%

外联式代码指写在单独的.js 文件中，然后通过 script 标记连接到 XHTML 页面中的代码。

```
<head>
    <title>JavaScript</title>
    <script src="JavaScript.js" type="text/javascript"></script>
</head>
```

感谢阅读！