重构实验

- 实验名称： 重构实验
- 实验目的
    1. 理解重构在软件开发中的作用
    2. 熟悉常见的代码环味道和重构方法
- 实验内容和要求
    1. 阅读：Martin Fowler 《重构-改善既有代码的设计》
    2. 掌握你认为最常见的 6 种代码坏味道及其重构方法
    3. 从你过去写过的代码或 Github 等开源代码库上寻找这 8 种坏味道，并对代码进行重构；**反对拷贝别人重构例子。**

代码坏味道

1. **神秘命名（Mysterious Name）**

   函数、模块、变量和类命名不够直观明了，需要重命名以求其能更清晰的表示自己的功能和用法。

2. **重复代码（Duplicated Code）**

   如果在一个以上的地点看到相同的代码结构，阅读这些重复的代码时就必须加倍仔细，如果要修改重复代码，必须找出所有的副本来修改。因此采用提炼函数提炼出重复的代码，然后让

这两个地点都调用被提炼出来的那一段代码。

3. **全局数据（Global Data）**

   全局数据从代码库的任何地方都可以修改，且没有机制可以探测出到底哪段代码做出了修改，当造成 bug 时难以定位。因此最好对此封装处理，用一个函数包装起来，将这个函数（及其封装的数据）搬移到一个类或模块中，只允许模块内的代码使用它，从而尽量控制其作用域。

4. **依恋情结（Feature Envy）**

   一个函数跟另一个模块中的函数或者数据交流格外频繁，远胜于在自己所处模块内部的交流。因此要么直接搬移函数，要么独立提炼出“依恋”的那部分再进行搬移。

5. **数据泥团（Data Clumps）**

   两个类中相同的字段、许多函数签名中相同的参数。这些总是绑在一起出现的数据真应该拥有属于它们自己的对象。应找出这些数据以字段形式出现的地方，运用提炼类将它们提炼到一个独立对象中。然后将注意力转移到函数签名上，运用引入参数对象或保持对象完整为它瘦身。

6. **临时字段（Temporary Field）**

   某个类内部某个字段仅为某种特定情况而设。这样的代码让人不易理解，因为通常认为对象在所有时候都需要它的所有字段。在字段未被使用的情况下猜测当初设置它的目的会造成阅读困难。对此最好提炼并搬移函数。

7. **冗赘元素（Lazy Element）**

   可能有这样一个函数，它的名字就跟实现代码看起来一模一样；也可能有这样一个类，根本就是一个简单的函数。通常只需要使用内联函数或是内联类。如果这个类处于一个继承体系中，可以使用折叠继承体系

8. **过多注释（Excessive Comments）**

   一段代码有着长长的注释，这些注释之所以存在乃是因为代码很糟糕。注释可以带我们找到本章先前提到的各种坏味道。找到坏味道后，我们首先应该以各种重构手法把坏味道去除。完成之后我们常常会发现：注释已经变得多余了，因为代码已经清楚地说明了一切。

重构案例

1. 神秘命名

翻了一下大一最初学 C 语言留下的文件，当时在学习数组、函数等知识时就为图省事只用一个字母表示，如 a[ ]表示一个学生名单的数组,f( )表示加法运算，可以看出很早就留下了命名不规范的隐患。现在看来至少应当改为 students[ ]，add_numbers( )。

2. 重复代码

用 js 实现勾选衣物计算碳排放的项目，其中部分代码如下：

```
topsSelect.addEventListener('change',function(){
   var selectedCategory = this.id;
```

```
    var selectedClothes=clothesData[selectedCategory][this.value];
    topWeight=selectedClothes.weight;
    topCarbonFactor=selectedClothes.carbonFactor;
    topTimes=selectedClothes.washingTimes;
})
  bottomsSelect.addEventListener('change',function(){
    var selectedCategory = this.id;
    var selectedClothes=clothesData[selectedCategory][this.value];
    bottomWeight=selectedClothes.weight;
    bottomCarbonFactor=selectedClothes.carbonFactor;
    bottomTimes=selectedClothes.washingTimes;

})
  outerwearsSelect.addEventListener('change',function(){
    var selectedCategory = this.id;
    var selectedClothes=clothesData[selectedCategory][this.value];
    outerwearWeight=selectedClothes.weight;
    outerwearCarbonFactor=selectedClothes.carbonFactor;
    outerwearTimes=selectedClothes.washingTimes;
  })
  dressesSelect.addEventListener('change',function(){
    var selectedCategory = this.id;
    var selectedClothes=clothesData[selectedCategory][this.value];
    dressWeight=selectedClothes.weight;
    dressCarbonFactor=selectedClothes.carbonFactor;
    dressTimes=selectedClothes.washingTimes;
})
```

可以看到，在每个事件监听器中，都有相似的代码段用来获取所选服装的信息并将其赋值给对应的变量。这种重复代码不仅降低了代码的可维护性，还增加了出错的可能性。为了消除这种重复，可以考虑将重复的部分提取出来，形成一个函数，然后在每个事件监听器中调用这个函数，以减少重复代码的量。

将重复代码提取为一个函数 `updateSelectedClothes(category, value)`，然后在每个事件监听器中调用这个函数。以下是重构后的代码：

```
function updateSelectedClothes(category, value) {
    var selectedCategory = category.id;
    var selectedClothes = clothesData[selectedCategory][value];
    return {
        weight: selectedClothes.weight,
```

```
        carbonFactor: selectedClothes.carbonFactor,
        washingTimes: selectedClothes.washingTimes
    };
}

topsSelect.addEventListener('change', function() {
    var selectedClothes = updateSelectedClothes(this, this.value);
    topWeight = selectedClothes.weight;
    topCarbonFactor = selectedClothes.carbonFactor;
    topTimes = selectedClothes.washingTimes;
});

//后略
```

3. 全局数据

下面是我处理人物性别形象图片时写的部分 js 代码：

```
var currentCharacter = "Animate/female.png";

    $("#gender-btn").click(function() {
        // 切换图片路径
        if (currentCharacter === "Animate/female.png") {
            currentCharacter = "Animate/male.png";
            $("#gender-btn").text("我是女生！");
        } else {
            currentCharacter = "Animate/female.png";
            $("#gender-btn").text("我是男生！");
        }

        // 更新角色图片
        $("#character").fadeOut(200,function(){
            $(this).attr("src", currentCharacter).fadeIn(200);
        });

        // 重置衣服
        resetTBO();
        resetDresses();
    });
```

问题在于，`currentCharacter` 这个全局变量可能遭到其他函数调动使之值变为`"Animate/female.png"`和`"Animate/male.png"`之外

的值，导致函数`$("#gender-btn").click(function()`不能正常运行，因此可以将 **currentCharacter** 变量封装在一个对象中，并将切换逻辑放在对象的方法中。这样可以更好地组织代码，并且可以避免污染全局命名空间。以下是封装后的示例：

```
var characterSwitcher = {
    currentCharacter: "Animate/female.png",

    toggleCharacter: function() {
        if (this.currentCharacter === "Animate/female.png") {
            this.currentCharacter = "Animate/male.png";
            $("#gender-btn").text("我是女生！");
        } else {
            this.currentCharacter = "Animate/female.png";
            $("#gender-btn").text("我是男生！");
        }

        this.updateCharacterImage();
        this.resetClothes();
    },

    updateCharacterImage: function() {
        $("#character").fadeOut(200, function() {
            $(this).attr("src",
characterSwitcher.currentCharacter).fadeIn(200);
        });
    },

    resetClothes: function() {
        resetTBO();
        resetDresses();
    }
};

$("#gender-btn").click(function() {
    characterSwitcher.toggleCharacter();
});
```

通过封装，我们将 **currentCharacter** 变量和切换逻辑都放在了 **characterSwitcher** 对象中，使得代码更加清晰和易于维护。

4. 依恋情结

在进行碳足迹计算时，我经常调用计算函数中的结果，如下：

```
  function submit ()
  {
      topStandardWeight = topWeight*1e-6;

      bottomStandardWeight = bottomWeight*1e-6;

      outerwearStandardWeight = outerwearWeight*1e-6;

      dressStandardWeight = dressWeight*1e-6;

      distance=distanceData[departureCity][destinationCity];

      standardWeight=topStandardWeight+bottomStandardWeight+outerwearSta
ndardWeight+dressStandardWeight;

      Times=topTimes+bottomTimes+outerwearTimes+dressTimes;

      electricEmissions=0.53*0.086*Times/7;

      waterEmissions=1e-3*(0.28*110*Times/7);

      usageEmissions=electricEmissions+waterEmissions;

      productionEmissions = 1e3*(topStandardWeight *
topCarbonFactor+outerwearStandardWeight*outerwearCarbonFactor+bottomSta
ndardWeight*bottomCarbonFactor+dressStandardWeight*dressCarbonFactor);

      transitEmissions = 1e3*(distance * standardWeight *
transitCarbonFactor);
```

```
      disposalEmissions = disposalCarbonFactor * standardWeight;

        totalEmissions =
        productionEmissions +
        transitEmissions +
        usageEmissions +
        disposalEmissions;

}
```

这样容易产生重复的运算，因此可以将运算结果从内部传出，来避免反复访问计算函数 `submit()`

实现代码如下：

```
 window.location.href = "./result.html?" +
    "emissions=" + totalEmissions +
    "&productionEmissions=" + productionEmissions +
    "&transitEmissions=" + transitEmissions +
    "&usageEmissions=" + usageEmissions +
    "&disposalEmissions=" + disposalEmissions;
```

如此可以重定向到结果页面，并将计算的排放量作为 URL 参数传递。

5. 数据泥团

这是经过了封装的衣物碳系数数据：

```
const clothesData = {
  "tops": {
              "0":{"name":"无","material": "无", "weight":
0,"carbonFactor":0,"washingTimes":0},
              "1": {"name":"polo 衫","material": "亚麻", "weight":
190,"carbonFactor":14.35,"washingTimes":48},
              "2": {"name":"polo 衫","material": "纯棉", "weight":
230,"carbonFactor":4.76,"washingTimes":48 },
              "3": {"name":"T 恤","material": "纯棉", "weight":
210,"carbonFactor":4.76,"washingTimes":48 },
              "4": {"name":"T 恤","material": "丝光棉", "weight":
230,"carbonFactor":14.35,"washingTimes":48},
```

```
            "5": {"name":"衬衫","material": "纯棉", "weight":
125,"carbonFactor":4.76,"washingTimes": 24},
            "6": {"name":"衬衫","material": "涤棉", "weight":
140,"carbonFactor":5.47,"washingTimes":24 },
            "7": {"name":"衬衫","material": "化纤", "weight":
205,"carbonFactor":8,"washingTimes":24},
            "8": {"name":"衬衫","material": "亚麻", "weight":
125,"carbonFactor":21.63,"washingTimes":24},
            "9": {"name":"卫衣","material": "纯棉", "weight":
340,"carbonFactor":4.76,"washingTimes":36},
            "10": {"name":"卫衣","material": "丝质", "weight":
280,"carbonFactor":0.82,"washingTimes":36},
            "11": {"name":"卫衣","material": "涤棉", "weight":
300,"carbonFactor":5.47,"washingTimes":36},
            "12": {"name":"卫衣","material": "毛料", "weight":
340,"carbonFactor":13.81,"washingTimes":36},
            "13": {"name":"毛衣","material": "毛纺织", "weight":
350,"carbonFactor":13.81,"washingTimes":36}
        },
  "bottoms":{
            "0":{"name":"无","material": "无", "weight":
0,"carbonFactor":0,"washingTimes":0},
            "1": {"name":"西裤","material": "棉质", "weight":
600,"carbonFactor":0.45,"washingTimes":18},
            "2": {"name":"西裤","material": "羊毛", "weight":
320,"carbonFactor":13.81,"washingTimes":18},
            "3": {"name":"打底裤","material": "棉质", "weight":
120,"carbonFactor":0.45,"washingTimes":36},
            "4": {"name":"打底裤","material": "羊毛", "weight":
120,"carbonFactor":13.81,"washingTimes":36},
            "5": {"name":"打底裤","material": "涤纶", "weight":
120,"carbonFactor":25.7,"washingTimes":36},
            "6": {"name":"牛仔长裤","material": "牛仔", "weight":
750,"carbonFactor":16.42,"washingTimes":48},
            "7": {"name":"运动长裤","material": "纯棉", "weight":
280,"carbonFactor":4.76,"washingTimes":18},
            "8": {"name":"运动长裤","material": "尼龙", "weight":
275,"carbonFactor":8,"washingTimes":18},
            "9": {"name":"运动长裤","material": "涤纶", "weight":
275,"carbonFactor":25.7,"washingTimes":18},
            "10": {"name":"短裤","material": "纯棉", "weight":
350,"carbonFactor":4.76,"washingTimes":24},
            "11": {"name":"短裤","material": "涤纶", "weight":
290,"carbonFactor":25.7,"washingTimes":24},
```

```
            "12": {"name":"短裤","material": "牛仔", "weight":
450,"carbonFactor":16.42,"washingTimes":24},
            "13": {"name":"短裙","material": "棉质", "weight":
250,"carbonFactor":0.45,"washingTimes":24},
            "14": {"name":"短裙","material": "牛仔布",
"weight":625,"carbonFactor":16.42,"washingTimes":24},
            "15": {"name":"短裙","material": "丝绸", "weight":
350,"carbonFactor":0.82,"washingTimes":24},
            "16": {"name":"短裙","material": "麻布", "weight":
200,"carbonFactor":21.63,"washingTimes":24},
            "17": {"name":"半身裙","material": "纯棉", "weight":
325,"carbonFactor":4.76,"washingTimes":24},
            "18": {"name":"半身裙","material": "丝绸", "weight":
350,"carbonFactor":0.82,"washingTimes":24},
            "19": {"name":"半身裙","material": "羊毛", "weight":
250,"carbonFactor":13.81,"washingTimes":24},
            "20": {"name":"半身裙","material": "涤纶", "weight":
200,"carbonFactor":25.7,"washingTimes":24},
            "21": {"name":"吊带裙","material": "丝绸", "weight":
350,"carbonFactor":0.82,"washingTimes":24},
            "22": {"name":"吊带裙","material": "棉质", "weight":
350,"carbonFactor":0.45,"washingTimes":24}
          },
  "outerwears": {
                "0":{"name":"无","material": "无", "weight":
0,"carbonFactor":0,"washingTimes":0},
                "1": {"name":"夹克","material": "纯棉", "weight":
190,"carbonFactor":8.17,"washingTimes":18},
                "2": {"name":"夹克","material": "涤棉", "weight":
190,"carbonFactor":5.47,"washingTimes":18},
                "3": {"name":"夹克","material": "毛料", "weight":
325,"carbonFactor":13.81,"washingTimes":18},
                "4": {"name":"夹克","material": "皮革", "weight":
150,"carbonFactor":73,"washingTimes":18},
                "5": {"name":"夹克","material": "牛仔", "weight":
375,"carbonFactor":8,"washingTimes":18},
                "6": {"name":"西装外套","material": "羊毛", "weight":
290,"carbonFactor":13.81,"washingTimes":18},
                "7": {"name":"西装外套","material": "亚麻", "weight":
275, "carbonFactor":21.63,"washingTimes":18},
                "8": {"name":"西装外套","material": "纯棉", "weight":
245,"carbonFactor":4.76,"washingTimes":18},
                "9": {"name":"西装外套","material": "涤棉", "weight":
245,"carbonFactor":5.47,"washingTimes":18},
```

```
                "10": {"name":"标准羽绒服","material": "羽绒", "weight":
1150,"carbonFactor":3.34,"washingTimes":18},
                "11": {"name":"轻薄羽绒服","material": "羽绒", "weight":
700,"carbonFactor":3.34,"washingTimes":18},
                "12": {"name":"运动外套","material": "涤纶", "weight":
400,"carbonFactor":25.7,"washingTimes":18},
                "13": {"name":"运动外套","material": "尼龙", "weight":
225,"carbonFactor":8,"washingTimes":18},
                "14": {"name":"棉服","material": "白色棉服", "weight":
1000,"carbonFactor":30.93,"washingTimes":18},
                "15": {"name":"棉服","material": "彩色棉服", "weight":
1000,"carbonFactor":28.63,"washingTimes":18},
                "16": {"name":"风衣","material": "化纤", "weight":
1350,"carbonFactor":8,"washingTimes":18},
                "17": {"name":"卫衣外套","material": "纯棉", "weight":
340,"carbonFactor":4.76,"washingTimes":18},
                "18": {"name":"卫衣外套","material": "羊毛", "weight":
350,"carbonFactor":13.81,"washingTimes":18}
                },
  "dresses":{
              "0":{"name":"无","material": "无", "weight":
0,"carbonFactor":0,"washingTimes":0},
              "1": {"name":"长裙","material": "丝绸", "weight":
500,"carbonFactor": 0.82,"washingTimes":24},
              "2": {"name":"长裙","material": "棉质", "weight":
600,"carbonFactor":0.45,"washingTimes":24},
              "3": {"name":"长裙","material": "涤纶", "weight":
160,"carbonFactor":25.7,"washingTimes":24}
            }
}
```


事实上，在经过合理封装之前，我们一直弄不清应该如何系统性的调用数据来进行运算与传值，导致我们的函数编码工作一直不顺利。

6. 临时字段

起初我为了标明当前 Q 版小人是男性还是女性，直接临时赋予了一个布尔变量 flag 进行表示，其 0/1 值决定 female/male，如下：

```
var flag = true;

$("#gender-btn").click(function() {
    // 切换角色性别
    flag = !flag;

    // 更新按钮文本
    if (flag) {
        $("#gender-btn").text("我是男生！");
    } else {
        $("#gender-btn").text("我是女生！");
    }

    // 更新角色图片
    var characterImg = flag ? "Animate/female.png" : "Animate/male.png";
    $("#character").fadeOut(200, function() {
        $(this).attr("src", characterImg).fadeIn(200);
    });
});
```

其实这导致我之后看这段代码总是不舒服，

首先，flag 本身意义不清楚，具有 “神秘命名”的坏味道；

其次，flag 是一个全局变量，产生 bug 难以定位，但单独封装又显得太多余。

最后我决定取缔这个 flag 变量，避免使用多余变量的代码如下：

```
var characterSwitcher = {
    currentCharacter: "Animate/female.png",

    toggleCharacter: function() {
        if (this.currentCharacter === "Animate/female.png") {
            this.currentCharacter = "Animate/male.png";
            $("#gender-btn").text("我是女生！");
        } else {
            this.currentCharacter = "Animate/female.png";
            $("#gender-btn").text("我是男生！");
        }
```

```
        this.updateCharacterImage();
        this.resetClothes();
    },

    updateCharacterImage: function() {
        $("#character").fadeOut(200, function() {
            $(this).attr("src",
characterSwitcher.currentCharacter).fadeIn(200);
        });
    },

    resetClothes: function() {
        resetTBO();
        resetDresses();
    }
};
```

7. 冗赘元素

项目早期，我们的初步设想是每件衣物下放一个按钮，通过按钮实现选择，但后来又用了更便于数据传输运算的下拉选择栏来实现计算/换衣功能，于是这些按钮就成为历史遗留问题了，并没有被后期赋予一些该有的功能。于是在一次讨论后，我们取消了这些没有实际作用的按钮。

8. 过多注释

注释过多往往是代码写的不简洁，比如在第六点提到曾临时用了一个 flag 来表示性别，当时就在后面写了一行注释“0 表示 female，1 表示 male”，现在回想这段注释既多余又业余，哪怕是要保留这个变量，起码也应该将其 rename 为 isFemale，这样就不用注释说明了。

另外，我在 js 文件中写一些功能时，为了不同函数之间的作用能分清，也经常一行一个注释解释这段代码的作用，后来我发现其实可以拆成多个 js 文件，每个文件专攻一个功能，比如 index.js 只负责小人换装系统，就不需要太多注释了。

参考文献：重构: 改善既有代码设计 - 第二版（Martin Fowler）

所用实例：碳排放项目