第三十一到第三十三天：我是精明的小卖家（一）

普通

Varsha (/mentor/detail/id/2) | 零基础学院 (/college/detail/id/5)

开始学习

人数　有 287 人在学习该课程，有 194 人已经完成该课程

时间　平均用时 3.5 天

关键词　JavaScript

课程概述

作业提交截止时间：09-01

第三十一到第三十三天：我是精明的小卖家（一）

课程目标：

这次的任务将会分解为好几个子任务，是我们以后在工作中会经常遇到的项目类型--MIS系统，我们今天将会以MIS系统中的一个简单的页面示例，来让大家尝试一下稍微复杂的页面功能，我们如何进行拆解。

课程描述

最终需求描述：

- 这是一个数据报表定制页面，有许多筛选表单，有一个表格，以及几个图表，需要我们在不使用任何框架的情况下，完成这个页面的开发
- 页面初始化的时候，显示默认的数据及图表
- 根据用户的表单选择，可以进行数据和图表的切换，但这些切换动作都是在当前页面中完成
- 用户可以在当前页面上做一些数据修改，我们先设定这些修改仅对自己有效，将数据存在浏览器本地
- 当用户选择了某种数据视图（选择了某些选项所展示的数据及图表）后，可以复制当前URL给其他人，其他人也能看见同样的视图，而不是回到初始化效果
- 我们会提供一份数据用例：链接：https://pan.baidu.com/s/1ckhyN9_vyKnWgB9icJmBpg (https://pan.baidu.com/s/1ckhyN9_vyKnWgB9icJmBpg) 密码: tbwy

任务拆解

和上一个任务画流程图不一样的是，这个任务不仅要画流程图，还需要画一个整体的模块关系图，整个任务可以大致分为几个模块：路由模块，表单模块，数据处理模块，表格模块，图表模块。

- 路由模块负责指挥页面的其它部分，该以什么样的状态来呈现页面
- 表单模块负责承接用户的交互及告诉后面的模块如何组装数据，展示图表
- 数据处理模块，则根据用户的输入，对数据进行组合，提供给页面展现
- 表格模块负责用表格的形式展现数据
- 图表模块负责用图表的方式展现数据

从简单的开始

需求

- 表单：我们先从最简单的表单开始，我们的数据维度有：月、地区、商品种类，表单选项的任务就是做这几个维度的筛选或者组合的设置。我们先只做一个，用来做示例，比如我们选择用地区，请用一个Select或者单选，让用户可以选择地区。选项应该包括华东、华南、华北三个地区。
- 数据处理：接下来，我们根据用户选择的地区表单，从完整数据中，把对应选择地区的数据取出来。
- 表格：最后，将上一步取出来数据渲染成表格，表格有一个表头，用于显示数据标题：商品、地区、1月、2月……12月，然后共有14列：商品、地区、以及12个月的销售情况

设计

上面需求还是一个比较简单的逻辑，表单的变更事件触发表格的更新，而表格更新依赖的数据来源于表单的选择。

伪代码类似：

HTML：

```
<select id="region-select">
    <option>....</option>
</select>
或者
<input type="radio">
……

.....

<div id="table-wrapper">
</div>
```

JS:
这是一种实现思路，渲染表格的方法接受数据参数，但不关注数据怎么来的

```
region-select的change事件 = function() {
    渲染新的表格(根据select选项获取数据)
}

function 根据select选项获取数据() {
    dosomething
    返回数据
}

function 渲染新的表格(data) {
    输出表头：商品、地区、1月、2月、…… 12月
    遍历数据 {
        输出每一行的表格HTML内容
    }
    把生成的HTML内容赋给table-wrapper
}
```

这是另外一种实现思路，表单变化时通知表格进行渲染，但不关注他用什么数据渲染

```
region-select的change事件 = function() {
    渲染新的表格()
}

function 根据select选项获取数据() {
    遍历数据 {
        向要返回的数据list中添加符合表单所选项的数据
    }
    返回数据
}

function 渲染新的表格() {
    根据表单选项获取数据
    渲染表格
}
```

两种思路在这个例子中，还不是那么有明显区别，随便选择一种。

稍微复杂一些

需求

- 我们现在加入第二个表单，商品种类，依然是select或者radio，自选。
- 两个表单项都存在，做并集的选择，比如选了华北，和手机，表示要看华北地区手机的销售情况
- 两个表单项的选择互相不干扰，即改变其中一个时候，不会导致另外一个的选项的变化

设计

很明显，这里要调整的是“根据select选项获取数据()”这个方法

function 根据select选项获取数据() {
遍历数据 {
向要返回的数据list中添加符合两个表单项所选的数据
}
返回数据
}

再复杂一些

需求

- 我们发现，有时候我们不止要看某个地区的数据，我们可能会想同时看好几个地区的数据，或者我们想看某个地区所有商品的销售情况，所以我们需要把地区及商品从单选改成多选
- 同时，为了方便多选，我们提供了一个功能叫做全部选择，分别给地区和商品各增加一个全选的CheckBox，全选有如下状态和逻辑：
  - 点击全选时，如果单个选项中只要有一个不是被选上的状态，则进行全选操作
  - 点击全选时，如果单个选项中所有选项都已经是被选上的状态，则无反应
  - 点击最后一个未被选中的单个选项后，全选CheckBox也要置为被勾选状态
  - 如果当前是全选状态，取消任何一个子选项，则全选CheckBox也要置为未勾选状态
  - 不允许一个都不勾选，所以当用户想取消唯一一个被勾选的子选项时，无交互反应，不允许取消勾选

设计

我们先不管数据是咋回事，我们先把全选这块逻辑整理一下。首先把radio，select换成CheckBox。

我们有两组CheckBox，从全选的逻辑是一致的，只是文案和具体值不一样，所以在这里，我们要尽可能想办法让两组CheckBox的逻辑能复用，而不要写两遍。

我们先来看看只有一组的情况下，逻辑是如何的：

- 分别给全选的CheckBox和各个单选的CheckBox绑定上点击事件
- 对于全选的CheckBox的点击事件，要做的事情很简单，让所有的CheckBox全部勾选上
- 对于单个的CheckBox，每次点击要做如下判断：
  - 在点击之前它是不是唯一一个被勾选的？如果是的话，阻止这次点击默认事件，或者立马又将其checked状态置为真
  - 点击之后，是不是满足了全选状态，并对应修改全选CheckBox的状态

上面是整个完整一组CheckBox的基本逻辑，大家也可以深入看还有哪些可优化的地方或缺失的逻辑。

接下来我们看如何进行复用代码，先介绍一种思路，抛砖引玉，大家可以自己引申或从其它优秀框架类库中去学习如何封装组件

我们可以HTML部分只留容器，具体的CheckBox由JS生成

```
<div id="region-radio-wrapper"></div>
<div id="product-radio-wrapper"></div>
```

JS，我们暂时不介绍面向对象，设计模式等方式，先用很基础的方式来讲解，后续会有专门的练习，但如果有经验的同学也可以直接运用你的经验，面向对象方面的知识来进行封装

```
function 生成一组CheckBox( CheckBox容器, CheckBox选项的参数对象或者数组 ) {
    生成全选checkbox的html，给一个自定义属性表示为全选checkbox，比如checkbox-type="all"
    遍历参数对象 {
        生成各个子选项checkbox的html，给一个自定义属性表示为子选项
    }
    容器innerHTML = 生成好的html集合

    给容器做一个事件委托 = function() {
        if 是checkbox
            读取自定义属性
            if 全选
                做全选对应的逻辑
            else
                做子选项对应的逻辑
    }
}

// 对象或数组自己根据喜好实现均可
生成一组CheckBox(容器1, [{
    value: 1,
    text: "XXXX"
}, {
    value: 2,
    text: "YYYY"
}]);

生成一组CheckBox(容器2, [{
    value: 1,
    text: "AAAA"
}, {
    value: 2,
    text: "BBBB"
}]);

// 生成一组CheckBox({
//    1: "XXXX",
//    2: "YYYY"
// });
```

这样分别调用两次函数，把容器ID和对应checkbox的数据传入即可。

当然，如果选项不是很多，我们书写自定义属性的成本不大，我们也可以把生成HTML那部分省略，直接把HTML写好在容器中，JS部分只需要做事件处理即可。本例我们推荐把checkbox相关代码还是写在HTML中。这样JS代码会更加简洁，调用生成checkbox逻辑函数时，只需要传入容器ID即可。

这里我们某种程度上解决了代码复用的问题，以后我们会再学习如何将其封装成一个真正意义上的组件。

交互解决了，接下来需要解决数据处理的问题，根据表单选择进行数据多维度的筛选

```
function 获取数据 {
    遍历原始数据 {
        判断是否在商品维度 或者 地区维度的选中范围内 {
            添加到返回数据list中
        }
    }
    返回数据
}
```

多选的表格渲染

需求

现在，我们给表格也提出了更复杂的需求，之前我们仅仅需要单纯的遍历数据，然后一行一行，一格一格输出即可，但当突然出现多选的情况下，我们会期望有更好的阅读体验：

- 当商品选择了一个，地区选择了多个的时候，商品作为第一列，地区作为第二列，并且把商品这一列的单元格做一个合并，只保留一个商品名称
- 当地区选择了一个，商品选择了多个的时候，地区作为第一列，商品作为第二列，并且把地区这一列的单元格做一个合并，只保留一个地区名称
- 当商品和地区都选择了多于一个的情况下，以商品为第一列，地区为第二列，商品列对同样的商品单元格进行合并
- 当商品和地区都只选择一个的情况下，以商品为第一列，地区为第二列

设计思路

选择谁做第几列并不是难事，判断一下两组CheckBox的选择数量即可，稍微有难度的是做单元格合并。

我们如何在通过JS输出表格HTML的时候，在合适的行输出时把rowspan属性加进去，并且在其他行的时候又跳过这行，请大家仔细思考。这个点，我们就不留示例代码，留个大家发挥的空间。

文件拆分

第一个子任务最后一步，我们提出一个新的需求，后续我们代码可能会越来越多，现在表格，表单，数据处理等一堆东西的代码都放在一个文件中，实在是不方便代码维护，所以，请你把你的代码进行拆分，在你的项目根目录下，建立一个js目录，分别创建类似checkbox.js，table.js这样的文件，把对应的代码放入。对了，页面入口主流程的代码，比如一些初始化的工作，放到一个叫做app.js的文件中

如果你有余力，可以开始使用webpack来进行打包，如果没有，或者你发现看了几眼就看不下去了，那么就只在你的html里多添加几个js即可，注意app.js应该放在几个js文件引用的最后一个。

提交

把你的代码放在Github后进行提交

总结

依然把今天的学习用时，收获，问题进行记录

下一个任务预告

明天我们继续该任务的下一个环节

课程优秀学习笔记

暂无优秀学习笔记~