**正课:**

**1. 插件:**

**官方插件: jQuery UI**

**\*\*\*封装自定义插件:**

**第三方插件:**

**2. jQuery的Ajax封装**

**\*\*\*\*\*跨域请求**

**1. 插件:**

**官方插件: jQuery UI**

**.autocomplete();**

**HTML: <input ... />**

**JS: $(文本框).autocomplete({**

**source:客户端数组|"远程.php路径"**

**})**

**PHP: 2种:**

**1. 返回只有一个键的关联数组, 只要键名为label，客户端autocomplete可自动识别**

**2. 返回包含多个键的关联数组: 2步:**

**1. 再次调用:**

**.autocomplete("instance").\_renderItem=function($ul,item){**

**//$ul: 自动获得当前下拉列表的ul元素**

**//item: 获得结果结合中，当前正在加载的元素对象**

**//操作: 3件事:**

**//1. 创建一个li>div:**

**//从item中获取键的值，拼接到div中**

**//2. 将新的li追加到$ul中**

**//3. 返回新的li的对象**

**}**

**2. 在autocomplete中定义select处理函数:**

**在单击每个列表项时自动触发:**

**select:function(e,obj){**

**//obj.item是当前li对应的原集合中的对象**

**//常用操作: 2个**

**//1. 将item中的主要内容，显示在文本框中**

**//2. 用当前选中项作为参数，跳转到新的url**

**return false;//必须**

**}**

**2. 日期选择: datepicker**

**HTML: <input />**

**CSS:**

**JS: $(文本框).datepicker()**

**3. 对话款: 表单提交**

**HTML:**

**<div id="xxx" title="标题"**

**<form>**

**...**

**//不用写提交和重置按钮**

**</form>**

**</div>**

**css:**

**JS: $(div).dialog(); //仅将div及其内容，变为对话框的样式**

**总结:**

**问题: jquery ui采用侵入的方式，自动加载样式和行为**

**优: 极大的减少了开发人员的代码量**

**缺: 侵入的class和行为都是写死的，不便于维护**

**解决: bootstrap vs jquery ui**

**bootstrap不采用侵入的方式，隐式添加任何代码**

**而是仅提供样式类库**

**由开发人员，自主的选择应用何种class**

**自定义插件:**

**何时: 只要在项目中发现频繁重用的功能，都要封装为自定义插件**

**如何提取:**

**前提: 必须使用HTML,css,js实现了插件的完整样式和功能**

**Step1: 将当前功能的css，提取到一个专门的css文件中**

**强调: css中尽量少的使用id,元素选择器**

**尽量一切都用class去实现！**

**Step2: 将当前功能的js行为，提取到一个专门的js文件中**

**在插件的js文件中查找自定义扩展属性的元素**

**强调: 将来插件都是通过查找自定义扩展属性来为元素添加行为的**

**今后，只要出发事件的元素，都要标记自定义扩展属性**

**如何使用:**

**1. 按插件要求，编写HTML内容结构**

**2. 引入插件的css, 在HTML中对应元素上，手动添加class**

**第三方插件:**

**1. jquery validate**

**2. jQuery的Ajax封装:**

**$.ajax({**

**url:"xxx.php",**

**type:"get|post",**

**data:"参数数据"|{变量:值,...}| $(form).serialize(),**

**//jquery-1.11.3.js**

**//success:data=>{...}**

**})//jquery-3.2.1.js**

**.then(data=>{...})**

**补充: jquery表单操作:**

**$(form).serialize()**