JQUERY DAY01:

* JQUERY
  * javascript 类库(JS 库)
    * 简单来说,Javascript 类库就是一个 JS 文件
  * 定义 - 预定义了很多的对象(方法和属性)和函数
    * 目的 - 为了简化 Javascript 的开发
    * JS 库举例
      * jQuery - 目前全球最火的
      * Prototype - 目前全球第二火的
      * Dojo
    * 问题
      * 没有任何办法将所有 JS 库学习掌握
        * 以某个 JS 库为案例,学习 JS 库的特点
      * JS 库都是基于 Javascript 封装的
  * jQuery
    * 基本内容
      * jQuery - 就是一个 JS 文件
      * 06 年创建 jQuery
      * 版本
        * 1.11.3 版本 - 1.xx 版本
      * 2.xx 版本 - 不再支持 IE8 及之前浏览器
      * 四个方面
        * jQuery - 针对 PC 端 WEB
      * jQuery UI - 针对 UI 布局效果
      * jQuery Mobile - 针对移动端 WEB
      * QUnit - 专门用于测试 Javascript
    * 使用 jQuery 的步骤
      * 在 HTML 页面引入 jQuery 文件
      * 使用 jQuery 的选择器定位元素
      * 使用 jQuery 的 API 完成需求
    * jQuery 对象
      * 工厂函数 - $()、jQuery()
        * 作用 - 接收 jQuery 的选择器内容
      * 约定 - 在 jQuery 变量前使用"$"符号
        * 注意 - 建议,不使用"$"可以的
    * DOM 对象与 jQuery 对象
      * 概念
    * DOM 对象 - 通过 DOM 获取的 HTML 页面元素
    * jQuery 对象 - 封装 DOM 对象后所产生的对象
      * jQuery 对象的底层还是 DOM 对象
      * 转换
        * jQuery 对象转换为 DOM 对象?可以
    * jQuery 对象是数组对象 - jQuery 对象[索引值]
      * jQuery 提供 get(index)方法
        * index - 索引值
    * DOM 对象转换为 jQuery 对象?可以
      * 使用 jQuery 的工厂函数 - $(DOM 对象)
    * DOM 对象与 jQuery 对象的属性或方法可以不可以相互调用?不可以
    * jQuery 的特点
      * 具有相对完善的处理机制 - 并不报错
      * 链式操作 - 链
      * 隐式迭代 - 显式迭代
  * jQuery 选择器
    * 基本内容
      * jQuery 选择器的用法类似于 CSS 中的选择器
        * jQuery 的作者个人非常喜欢 CSS 的选择器
      * jQuery 选择器的特点
        * 数量众多 - 几十个选择器 - 记不住
    * 混合(复合)使用时,难度大 - 没有标准答案
    * 九类选择器
    * 选择器的用法
      * 基本选择器
        * #id
    * .className
    * element
    * *
    * selector1,selector2 - 并集
    * 层级选择器
      * sel1 空格 sel2 - 祖先元素与后代元素
    * sel1 > sel2 - 父元素与子元素
    * sel1 + sel2 - 指定元素的下一个元素
    * sel1 ~ sel2 - 指定元素后面所有的兄弟元素
    * 过滤选择器
      * 用法 - 在选择器前增加":"
    * 基本过滤选择器
      * :first - 匹配第一个
        * first()方法
      * :last - 匹配最后一个
        * last()方法
      * :even - 匹配索引值为偶数
      * :odd - 匹配索引值为奇数
      * :eq(index) - 匹配索引值为 index 的元素
        * eq(index)方法
      * :gt(index) - 匹配索引值大于 index 的元素
      * :lt(index) - 匹配索引值小于 index 的元素
      * :not(selector) - 与 selecor 选择器相反的元素
      * :header - 匹配 h1~h6 标题元素
        * 在实际开发中很少使用
      * :animated - 匹配 jQuery 的动画效果
        * 只能匹配 jQuery 实现的动画效果
    * 子元素过滤选择器 - 在其前面增加空格
      * :nth-child(index) - 匹配第 index 个子元素
      * :first-child - 匹配第一个子元素
      * :last-child - 匹配最后一个子元素
      * :only-child - 匹配唯一一个子元素
    * 内容过滤选择器
* :contains(文本) - 匹配包含指定文本内容的元素
* :empty - 匹配没有子元素也没有文本元素的空元素

* :parent - 匹配包含子元素或文本元素的元素
* :has(selector) - 匹配包含 selector 的元素 的父元素
* 可见性过滤选择器
* :visible - 匹配可见的所有元素
* :hidden - 匹配不可见的所有元素
* 属性过滤选择器
* [attrName] - 匹配包含属性 attrName 的元素
* [attrName=value] - 匹配包含属性 attrName 等于 value 的元素
* [attrName!=value] - 匹配包含属性 attrName 不等于 value 的元素(包含没有 attrName 属性的元素)
* [attrName^=value] - 匹配包含属性 attrName 以 value 开始的元素
* [attrName$=value]
* [attrName*=value]
* [attrName1][attrName2] - 交集
* 表单对象属性过滤选择器
* :enabled - 可用输入框
* :disabled - 不可用输入框
* :checked - 被选中
* :selected - 被选中
* 表单选择器
* 扩展内容
* Javascript 面向对象
* Javascript 是基于原型的面向对象
* Java 是基于类(Class)的面向对象
* 全局函数
* function fn(){} - 全局函数 fn()
* window.fn = function(){} - window 对象的方法
* 自调函数

```
(function(){
    所有内容定义在这里
})();
```

* 定义的全局变量及全局函数,存储在浏览器的内存中
* 博客程序 - Hexo(使用 NodeJS)
* 生成静态页面
* MarkDown 编辑器 - 编写文章
* 完成某个功能的步骤
* 明确需求 - 要完成的是什么
* 分析思路 - 要怎么去完成
* 代码实现 - 使用代码功能实现
* 代码优化 - 代码量减少|性能提高|...
* 案例 - 使用 jQuery 完成用户注册表单验证
* 用户名
* 获取焦点时 - 4-20 位字符，支持英文、数字及'-'、'_'组合
* 失去焦点时 - 验证逻辑
* 用户名不能为空
* 长度在 4-20 之间
* 内容只能英文、数字及'-'、'_'
* 密码
* 获取焦点时 - 6-20 位字符，可使用字母、数字的组合
* 失去焦点时
* 密码不能为空
* 长度在 6-20 之间
* 内容只能字母、数字
* 确认密码
* 获取焦点时
* 失去焦点时
* 密码不能为空
* 长度在 6-20 之间
* 内容只能字母、数字
* 两次密码是否一致
* Email
* 获取焦点时 - 完成验证后，可以使用该邮箱登录和找回密码
* 失去焦点时
* Email 不能为空
* 格式是否正确

```
/^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/
```

* 点击"注册"按钮,保证上述所有内容验证成功
* 样式说明
* 获取焦点提示信息 - focus
* 验证失败提示信息 - error
* 隐藏提示信息 - hide

<title>05_DOM 对象与 jQuery 对象之间的转换</title>

```
<input type="text" value="请输入你的用户名" id="username">
<script>
    // jQuery 对象如何转换为 DOM 对象
    var $username = $("#username");
    // 1. jQuery 对象是数组对象 - jQuery 对象[索引值]
    var user1 = $username[0];
    console.log(user1.value);// 测试
    // 2. jQuery 对象提供 get(index)方法
    var user2 = $username.get(0);
    console.log(user2.value);// 测试
    // DOM 对象如何转换为 jQuery 对象
var username = document.getElementById("username");
    var $user = $(username);// $(DOM 对象)
    console.log($user.val());// 测试
</script>
```

<title>06_实现读取超链接内文字</title>

```
<a>这是一个超链接</a>
<script>
    // DOM
    // 1. 获取<a>元素
    var a = document.getElementsByTagName("a")[0];
    // 2. 通过 innerHTML 属性或 textContent 属性
    console.log(a.innerHTML);
```

```
        // jQuery
        console.log($("a").text());
    </script>
```

<title>07_jQuery 具有完善的处理机制</title>

```
<input type="text" value="请输入你的用户名" id="username">
        <script>
            // DOM - 获取错误,直接报错
        //var user = document.getElementById("username");
        /*问题 - 结果如何?
         * * 报错 - 17 行:Cannot read property 'value' of null
         * 解决
         * * 如果 id 属性值传递错误,错误提示内容
         * * 如果 id 属性值传递正确,打印 value 属性值*/
        /*if(user){//正确
                console.log(user.value);
        }else{//错误
                console.log("id 不存在.");
        }*/
        // jQuery
        var $user = $("#username");
        // 结果如何?并不报错
        console.log($user.val());
        </script>
```

```
    <script>
        /* jQuery 绑定事件
         * * 使用工厂函数定位元素
         * * 调用封装的事件方法
         *     click(function(event){})*/
            $("#btn1").click(function(){
                /** 改变指定元素的背景颜色
                 * css(attrName,attrValue)
                 * * attrName - CSS 的属性名称
                 * * attrValue - CSS 的属性值*/
                $("#one").css("background","pink");
        });
        $("#btn2").click(function(){
                // 类似于批处理
                $(".mini").css("background","pink");
        });
        $("#btn3").click(function(){
                $("div").css("background","pink");
        });
        $("#btn4").click(function(){
                // 所有元素 - HTML 页面包含的所有内容
                $("*").css("background","pink");
        });
        $("#btn5").click(function(){
        // 多个选择器并列使用,中间使用","分隔 - 并集
                $("span,#two").css("background","pink");
        });
```

```
    </script>
```

<title>11_表格隔行变色</title>

```
    <script>
    /** 需求 - 表格中索引值为奇数的行,背景颜色为蓝色
         * 分析
         * * <table>表格中<tr>元素表示行
         * * 在指定<table>元素内,所有<tr>元素
         *     * table 与 tr 属于祖先与后代关系
         *     * 不包含<thead>元素中的<tr>元素 */
    //$("#data tr:not(:first):odd").css("background","blue");
        $("#data>tbody>tr:odd").css("background","blue");
    </script>
```

<title>12_内容过滤选择器</title>

```
    <script>
        $("#btn1").click(function(){
                $("div:contains(di)").css("background","pink");
        });
        $("#btn2").click(function(){
        // 空元素 - 既不包含子元素也不包含文本元素
                $("div:empty").css("background","pink");
        });
        $("#btn3").click(function(){
                // 含有 class 为 mini 元素的 div 父元素
                $("div:has(.mini)").css("background","pink");
        });
        $("#btn4").click(function(){
                // 不为空的元素
                $("div:parent").css("background","pink");
        });
    </script>
```

<title>15_实现多页签切换效果</title>

```
    <script>
            // 1. 获取所有的 li 元素,并且绑定 click 事件
        $("#tab>li").click(function(){
        // 2. 事件处理函数
                // a. 获取绑定事件的目标元素
                // 问题 - this 指代目标元素(DOM 对象)
                // b. 得到目标元素的 value 属性值(1,2,3)
                //var index = this.value;
                // c. 获取对应的 id 的 div 元素
                //var $div = $("#content"+index);
              // e. 将其他 2 个 div 的 style="z-index: 1;"属性删除 removeAttr(name)
                $("div[style]").removeAttr("style");
            // d. 将 style="z-index: 1;"属性设置给获取的 div 元素 attr(name,value)
        $("#content"+this.value).attr("style","z-index: 1;");
        });
```

```
        // 获取可用的 input 元素,修改 value 属性值
        $("input:enabled").val("值改变啦...");
        // 获取不可用的 Input 元素,修改 value 属性值
        $("input:disabled").val("值又改变啦...");
        // 获取多选框被选中的个数
        alert($("input[name=newsletter]:checked").length);
        // 获取下拉列表中被选中的文本内容
        alert($("select:eq(0)>option:selected").text());
    </script>
```

<title>18_多选框选中的个数</title>

```
        // 1. 获取 button 按钮,绑定 click 事件
        $("#btn").click(function(){
        // 2. 处理函数中,输出多选框被选中的个数
            alert($("input[name=check]:checked").length);
        });
```

<title>20_动态列表效果</title>

```
<script>
        /* 需求
        * * 页面加载完毕后,列表中 富士-爱国者 隐藏
        * * 点击"显示全部品牌"按钮
        *    * 将隐藏列表显示
        *    * 将按钮的文本内容修改为"精简显示品牌"
        * * 点击"精简显示品牌"按钮
        *    * 将显示列表隐藏
        *    * 将按钮的文本内容修改为"显示全部品牌" */
        // 1. 获取 富士-爱国者 li 元素
        var $li = $("ul>li:gt(5):not(:last)");
        // 2. 隐藏 - hide()
        $li.hide()
        // 3. 为按钮绑定 click 事件
        $("a>span").click(function(){
            /** is(expr)方法
             * * 判断当前元素是否匹配 expr 的内容
             * * 返回值为 Boolean 值
             *    * true - 匹配*/
            if($li.is(":hidden")){
                // 将隐藏列表显示
                $li.show();
            // 将按钮的文本内容修改为"精简显示品牌"
                $("a>span").text("精简显示品牌");
            }else{
                // 将显示列表隐藏
                $li.hide();
            // 将按钮的文本内容修改为"显示全部品牌"
                $("a>span").text("显示全部品牌");
            }
            // 阻止<a>元素的默认行为(页面跳转)
            return false;
        });
    </script>
```

JQUERY DAY02:

* 原生 DOM
  * document 对象
    * 属性
      * documentElement 属性 - 指向<html>
    * 方法
      * getElementById
      * getElementsByName
      * getElementsByTagName
      * querySelector
      * querySelectorAll
  * element 对象
    * 元素树
      * parentElement - 父元素
      * children - 所有子元素
    * 操作属性
      * getAttribute()
      * setAttribute()
      * removeAttribute()
  * node 对象
    * nodeName、nodeValue、nodeType
    * 节点树
      * parentNode
      * childNodes
    * 替换节点
    * 删除节点
    * 插入节点
    * 复制节点
* 操作 DOM - 解析 HTML 元素
  * 基本操作
    * html() - 作用类似于 innerHTML 属性
      * 获取 - html()
      * 设置 - html(html 代码)
    * text() - 作用类似于 textContent 属性
      * 获取 - text()
      * 设置 - text(文本内容)
    * val() - 作用类似于 value 属性
      * 获取 - val()
      * 设置 - val(value)
    * attr() - 作用类似于 getAttribute 和 setAttribute
      * 获取 - attr(name)
      * 设置 - attr(name,value)
      * 删除 - removeAttr(name)
  * 样式操作
    * attr("class",className)
      * 设置样式 - 覆盖原有所有的样式
    * addClass() - 追加样式
      * 基于原有的样式,增加一个样式
    * removeClass() - 移除样式

* 不传递参数 - 删除所有样式
* 需要传参数 - 删除指定样式
* 传递多个样式名称,中间用空格隔开
* toggleClass(className) - 切换样式
* 在没有样式与指定样式之间切换
* hasClass(className) - 判断样式
* 判断是否包含某个指定样式
* 注意 - 并不是判断是否含有样式
* css()方法
* 获取 - css(attrName)
* 获取指定 CSS 的属性名的值
* 设置
* css(attrName,attrValue)
* css(options)
options - {key:value}
* 遍历节点
* 父元素 - parent()
* 不传递参数 - 获取指定元素的父元素
* 传递参数 - 获取指定元素的指定(符合 expr)父元素
* 注意
* parent(expr) - 父元素
* parents(expr) - 祖先元素(从父级元素到根元素<html>)
* 子元素
* 所有子元素 - children()
* 指定子元素 - children(expr)
* 兄弟元素
* 上一个兄弟元素 - prev()
* 下一个兄弟元素 - next()
* 所有兄弟元素 - siblings()
* 选择器 ele1~ele2 - 获取 ele1 后面所有 ele2 的兄弟元素
* find(expr) - 在指定元素中的后代元素查找指定元素
* 创建节点
* 元素节点 - $(HTML 代码)
* 文本节点 - text()
* 属性节点 - attr()
* 通用写法 - $(HTML 代码)
* 插入节点
* 内部插入 - 插入在指定元素内(子元素)
* append() * prepend()
* appendTo() * prependTo()
* 外部插入 - 插入在指定元素外(兄弟元素)
* before() * after()
* insertBefore() * insertAfter()
* 删除节点
* remove() - 删除自身节点及后代节点
* empty() - 删除后代节点,保留自身节点(清空)
* 替换节点
* replaceWith - 前面的元素是被替换元素
* replaceAll - 就是颠倒了的 replaceWith
* 复制节点
* DOM - cloneNode(boolean)
* boolean 参数 - 表示是否复制后代节点
* 默认为 false,不复制
* jQuery - clone(boolean)
* boolean 参数 - 表示是否复制事件
* 面试题:
以下哪些说法是正确的?BD
A cloneNode()方法的参数表示是否复制子节点
B cloneNode()方法的参数表示是否复制后代节点
C cloneNode()方法的参数表示是否复制事件
D jQuery 的 clone()方法的参数表示是否复制事件
以下哪个是错误的? C
A var a = {} // 对象
B var b = [] // 数组
C var c = ()
D var d = // // 正则表达式
alert("xxx\nyyy"),要求换行?
* 事件
* ready() - 作用类似于 window.onload
* 写法
* $(document).ready(function(){});
* $().ready(function(){});
* $(function(){});
* ready 与 onload 的区别
* ready
* 有简写方式
* 在 HTML 页面中编写多个
* 必须等待 HTML 页面所有 DOM 元素都加载完毕后再执行
* 执行的速度快
* onload
* 无简写方式
* 在 HTML 页面中只能存在一个
* 必须等待 HTML 页面所有内容都加载完毕后再执行
* 执行的速度慢
* JS 库冲突 - "$"符号的使用
* 先引入其他 JS 库,后引入 jQuery
* jQuery 选择不再使用"$"符号
* 在 ready()内部定义形参"$"
* 在 ready()内部"$" - jQuery
* 在 ready()外部"$" - 其他 JS 库
* 使用自调函数接收 jQuery
* 在自调函数内部"$" - jQuery
* 在自调函数外部"$" - 其他 JS 库
* jQuery.noConflict();
* 先 jQuery,再引入其他 JS 库
* jQuery.noConflict();
* 事件绑定
* bind(type,data,callback)
* type - 设置绑定的事件名称

```
            * data - 作为 event.data 属性值传递给事件对象的额外数据对象
            * callback - 事件的处理函数
        * bind()支持的事件名称
            blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error
        * 单事件绑定 - type:指定一个事件名称
    * 多事件绑定 - type:指定多个事件名称,中间使用空格
    * 解绑事件 - unbind(type,callback)
        * 解绑 bind()绑定的事件
        * 解绑例如 click()绑定的事件
    * 事件冒泡
    * 三个阶段
        * 事件捕获
        * 事件触发
        * 事件冒泡
    * 取消冒泡
        * event 事件对象的 stopPropagation()
        * addEventListener(,,bool)
    * 事件对象 event
    * target - 绑定事件的目标元素
    * pageX/clientX/offsetX/x - 鼠标坐标值 x
    * pageY/clientY/offsetY/y - 鼠标坐标值 y
    * 阻止默认行为
        * preventDefault();
        * return false;
    * 事件模拟 - trigger()
* 扩展内容
    * this 用法
        * this 用于指代 DOM 对象(具有上下文环境)
        * 在全局域使用 this 指代 window 对象
        * Javascript 的构造器中
        * 注意
            * this 的用法是 Javascript 中一大难点(最难)
            * 实际开发中,一般错误都是 this
        * 建议
            * 会就用,不会就不用
    * 开发建议
        * 误区
            * 一次性将代码编写到最好
        * 广告
            * 没有最好,只有更好
        * 建议
            * 以完成需求目标为目的
            * 利用技术能力优化代码
    * 代码调试
        * 基本调试
            * console.log()
            * 浏览器的 F12 功能列表介绍
            * Elements - 快速页面元素定位
        * Network - 抓取客户端与服务器端之间的交互数据
            * Sources - 专门用于 Javascript 代码调试
            * Timeline - 用于页面性能优化(加载时间)
            * Resource - 查看浏览器的本地存储
            * Console - 控制台
                * error() - 错误日志
    * warn() - 警告日志
    * info() - 信息日志
    * log() - 日志
```

<title>05_实现开关门效果</title>

```
<script>
        /** 点击 id 为 d2 元素时
        * * 如果 id 为 d1 元素显示的话,隐藏 - hide()
        * * 如果 id 为 d1 元素隐藏的话,显示 - show()
        *   * is(":hidden") - 返回 true,表示隐藏*/
        // 1. 获取 id 为 d2 元素,绑定 click 事件
        $("#d2").click(function(){
        // 2. 事件的处理函数 - 判断是否隐藏
            if($("#d1").is(":hidden")){
            // 3. 如果 id 为 d1 元素隐藏的话,显示
                $("#d1").show();
                $("#d2").text("<<");
            }else{
            // 4. 如果 id 为 d1 元素显示的话,隐藏
                $("#d1").hide();
                $("#d2").text(">>");
            }
        });
    </script>
```

<title>06_样式操作</title>

```
<script>
        $("#b1").click(function(){
            /** attr("class",className)
            * * 设置样式 - 会覆盖原有的所有样式*/
            $("#mover").attr("class","one");
        });
        $("#b2").click(function(){
/** 先利用 attr()设置 one 样式,再利用 addClass()设置 mini 样式
        * 问题 - 当前只有 mini 样式,还是也包含 one 样式?
            *   * A 只有 mini 样式
            *   * B 包含 one 和 mini 样式(答案)
            * addClass() - 追加样式
            * * 表示在原有基础上,累加一个样式*/
            $("#mover").addClass("mini");
        });
        $("#b3").click(function(){
            /** removeClass() - 删除样式
            * * 不传递参数 - 删除所有样式
```

```
            * * 需要传参数 - 删除指定样式
            *    * 指定多个样式,之间使用空格隔开*/
            $("#mover").removeClass("one mini");
        });
        $("#b4").click(function(){
            /** toggleClass(className)
             * * 表示在没有样式与指定样式之间切换 */
            $("#mover").toggleClass("one");
        });
        $("#b5").click(function(){
// hasClass(className) - 判断是否包含某个指定样式
            console.log($("#mover").hasClass("one"));
        });
        $("#b6").click(function(){
            $("#mover").css({
                "background" : "green",
                "width" : 400
            });
        });
    </script>
```

<title>07_实现伸缩二级菜单</title>

```
<script>
        // 1. 为 span 元素绑定 click 事件
        $("span").click(function(){
        // 2. 事件的处理函数 - 判断当前 span 元素的样式
            // 定位触发 click 事件的 span
            if($(this).hasClass("open")){
// 3. 当前 class 为 open,改为 closed,ul 的 class 改为 hide
                $(this).attr("class","closed");
                $(this).next().attr("class","hide");
            }else{
// 4. 当前 class 为 closed,改为 open,ul 的 class 改为 show
                $(this).attr("class","open");
                $(this).next().attr("class","show");
            }
        });
    </script>
```

<title>08_遍历节点</title>

```
<script>
        // 1. 获取橘子的父元素
        console.log($("li:eq(1)").parent().attr("title"));
        // 2. 获取 ul 下所有子元素的个数
        console.log($("ul").children().length);
        console.log($("ul").children(":eq(1)").attr("title"));
        // 3. 获取橘子的上一个和下一个兄弟
        console.log($("li:eq(1)").prev().attr("title"));
        console.log($("li:eq(1)").next().attr("title"));
        // 4. 获取橘子的所有兄弟元素的个数
        console.log($("li:eq(1)").siblings().length);
        // 5. 利用 find()获取 ul 所有 li 元素的个数
        console.log($("ul").find("li").length);
    </script>
```

<title>09_实现购物车动态操作</title>

```
<script>
        // 1. 为 button 绑定 click 事件
        $("button").click(function(event){
            // 判断当前是-还是+
            var elem = event.target;
            if($(elem).text() == "+"){
                // 修改 span 元素的文本内容
                var num = parseInt($(elem).prev().text());
                $(elem).prev().text(num+1);
            }else{
                var num = parseInt($(elem).next().text());
                $(elem).next().text(num-1);
                // num=0
                if(num == 0){
                    $(elem).next().text("0");
                }
            }
            // 修改小计的值
            // 数量
var number = parseInt($(elem).parent().children("span").text());
            // 单价
var price  =
        parseInt($(elem).parent().prev().children("span").text());
            // 修改小计
    $(elem).parent().next().children("span").text(number*price);
            // 合计
    var $spans = $("#data>tbody>tr :last-child>span");
            // 遍历
            var total = 0;
            for(var i=0;i<$spans.length;i++){
                var span = $spans[i];
                var num = parseInt($(span).text());
                total += num;
            }
            $("#data>tfoot>tr>td:last>span").text(total);
        });
    </script>
```

<title>10_创建 DOM 节点</title>

```
<script>
 // 创建<li title='香蕉'>香蕉</li>元素,添加到 ul 元素
        // 1. 创建<li title='香蕉'>香蕉</li>元素
            /* a. 创建元素节点<li></li>
            var $li = $("<li></li>");
            // b. 设置文本
            $li.text("香蕉");
            // d. 设置属性
            $li.attr("title","香蕉");
```

```
            //var $li = $("<li title='香蕉'>香蕉</li>");
            // 2. 添加到 ul 元素
            //$("ul").append($li);
            $("ul").append($("<li title='香蕉'>香蕉</li>"));
        </script>
    <title>11_内部插入节点</title>
    <script>
            // 操作天津节点和魔兽节点
    // append 后面的节点插入在 append 前面的节点的后面
            //$("#ms").append($("#tj"));
     // prepend 后面的节点插入在 prepend 前面的节点的前面
            $("#ms").prepend($("#tj"));
// appendTo 前面的节点插入在 appendTo 后面的节点的后面
            //$("#ms").appendTo($("#tj"));
            // prependTo
            //$("#ms").prependTo($("#tj"));
        </script>
    <title>13_左右选项移动操作</title>
    <script>
            $("#add").click(function(){
            $("#first>option:selected").appendTo($("#second"));
            });
            $("#add_all").click(function(){
                $("#first>option").appendTo($("#second"));
            });
            // select 元素的事件
            $("#first").dblclick(function(){
            $("#first>option:selected").appendTo($("#second"));
            });
        </script>
    <title>14_删除节点</title>
     <script>
            // remove()删除苹果节点
            $("ul>li:first").remove();
            // empty()删除橘子节点
            $("ul>li:first").empty();
        </script>
    <title>15_动态操作表格</title>
    <script>
            // 添加
            // 1. 为 id 为 add 按钮绑定 click 事件
            $("#add").click(function(){
            // 2. 事件处理函数
// a. 创建<tr><td>Allen</td><td>$3700</td>
<td>2011-12-05</td><td><a href="#">删除</a></td></tr>
var $tr = $("<tr><td>Allen</td><td>$3700</td>
<td>2011-12-05</td><td><a href='#'>删除</a></td></tr>");
                // b. 添加到 tbody 元素
                $("#data>tbody").append($tr);
                // c. 为新增 a 元素绑定 click 事件
                $("a").click(moveEle);
            });
            // 删除
            // 1. 为 a 元素绑定 click 事件
            $("a").click(moveEle);
            function moveEle(){
            // 2. 事件处理函数
                // a. 定位 a 元素所在的 tr 元素
                var $tr = $(this).parent().parent();
                // b. 调用 remove()进行删除
                $tr.remove();
            }
     </script>
    <title>16_替换 DOM 节点</title>
    <script>
            // 点击苹果节点时,将苹果节点替换为小米
            $("ul>li:first").click(function(){
                // replaceWith 前面的元素是被替换元素
            //$(this).replaceWith($("<li title='小米'>小米</li>"));
                $("<li title='小米'>小米</li>").replaceAll($(this));
            });
        </script>
    <title>17_复制 DOM 节点</title>
    <script>
            // 当点击苹果节点时,复制所有 li 元素,添加到 ul 上
            /* DOM
            var ping = document.getElementsByTagName("li")[0];
            ping.onclick = function(){
                var copy = ping.cloneNode(true);
     var ul = document.getElementsByTagName("ul")[0];
                ul.appendChild(copy);
            }*/
            // jQuery
            $("ul>li:first").click(function(){
                var copy = $(this).clone(true);
                $("ul").append(copy);
            });
        </script>
    <title>18_图片提示</title>
    <script type="text/javascript">
        var title;
     // 1. 为包含<img>元素的<a>元素绑定鼠标悬停事件
     $("a[class=tooltip]").mouseover(function(event){
            title = $(this).attr("title");
            $(this).removeAttr("title");// 删除 title 属性
            // 获取对应大图片的路径
            var src = $(this).attr("href");
            // 创建 div 包含 img 元素
var $div = $("<div id=
'tooltip'><img src='"+src+"'><br>"+title+"</div>");
```

```
        // 添加到 body 元素下
        $(this).append($div);
        // event.pageX|Y - 将创建的 div 重新定位
        $("#tooltip").css({
                "top" : event.pageY+20,
                "left" : event.pageX+10
        }).show();
 }).mouseout(function(){
        $(this).attr("title",title);
        $("#tooltip").remove();
 }).mousemove(function(event){
        $("#tooltip").css({
                "top" : event.pageY+20,
                "left" : event.pageX+10
        });
 });
</script>
 <title>26_加载时间对比</title>
      var startTime = new Date().getTime();
 function test1(){
        var endTime1 = new Date().getTime();
        console.log("ready : "+(endTime1 - startTime)+"毫秒")
 }
 function test2(){
        var endTime2 = new Date().getTime();
 console.log("onload : "+(endTime2 - startTime)+"毫秒")
 }
 $(document).ready(function(){
        test1();
 });
 window.onload = function(){
        test2();
 };
</script>
<body>
<img src="img/demo.jpg"
                style="width:200px;height:200px;"/>
</body>
  <title>27_单事件绑定</title>
<script>
            /*  鼠标点击 h5 元素时,显示或隐藏 div
        $("h5").click(function(){
                var div = $(this).next();
                if(div.is(":hidden")){
                        div.show();
                }else{
                        div.hide();
                }
        });
        */
        /** bind(type,callback)绑定事件
         * * type - 设置当前绑定的事件名称
         * * 类似于 DOM 中的 addEventListener()方法 */
        $("h5").bind("click",function(){
                var div = $(this).next();
                if(div.is(":hidden")){
                        div.show();
                }else{
                        div.hide();
                }
        });
      </script>
 <title>28_多事件绑定</title>
<script>
            /*  鼠标悬停事件 1)显示 2)隐藏
        $("h5").mouseover(function(){
                $(this).next().show();
        }).mouseout(function(){
                $(this).next().hide();
        });
        // bind(type) - type 指定多个事件名称,中间使用空格
        $("h5").bind("mouseover mouseout",function(){
                var div = $(this).next();
                if(div.is(":hidden")){
                        div.show();
                }else{
                        div.hide();
                }
        });
      </script>
</body>
 <title>31_阻止默认行为</title>
<script>
            // 表单提交时,触发 submit 事件
        $("form").submit(function(event){
                // 阻止页面默认行为
                //event.preventDefault();
                return false;
        });
      </script>
 <title>32_模拟事件</title>
<script>
            // 点击 button 按钮,向 div 元素中写入文本内容
        $("#btn").click(function(event,a,b){
                $("#test").text("button 被我点击了..."+a+" "+b);
        }).trigger("click",["a","b"]);
        // trigger()模拟用户点击 button
      </script>
```

JQUERY DAY03:

* 动画效果
  * 回顾内容
    * CSS3 中的动画效果
    * 浏览器运行 HTML 页面(HTML|CSS|JAVASCRIPT)时
      * 解析 HTML 和 CSS
      * 解析 JAVASCRIPT
  * 预定义动画
    * 显示与隐藏
      * show() - 显示
        * 无参 - 无动画效果
* 有参 - 有动画效果
  * show(speed,callback)
    * speed - 设置动画执行的速度
      * 预定义速度 - slow|normal|fast
      * 设置时间,单位为毫秒
    * callback - 回调函数,在动画执行完毕后的回调函数
      * hide() - 隐藏
        * 无参 - 无动画效果
* 有参 - 有动画效果
  * hide(speed,callback)
    * speed - 设置动画执行的速度
      * 预定义速度 - slow|normal|fast
      * 设置时间,单位为毫秒
    * callback - 回调函数,在动画执行完毕后的回调函数
    * 滑动式动画
      * slideUp(speed,callback) - 向上滑动
        * 当不传递任何参数时
  * 代码并没有报错,底层具有判断机制
  * 效果正确并且具有动画效果(底层预定义速度)
* slideUp(speed,callback)
  * speed - 设置动画执行的时长
  * callback - 在动画执行完毕后的回调函数
    * slideDown(speed,callback) - 向下滑动
      * 当不传递任何参数时
  * 代码并没有报错,底层具有判断机制
  * 效果正确并且具有动画效果(底层预定义速度)
* slideDown(speed,callback)
  * speed - 设置动画执行的时长
  * callback - 在动画执行完毕后的回调函数
    * 淡入淡出 - 用法与上述一致
      * fadeIn() - 淡入
      * fadeOut() - 淡出
    * 动画切换
      * toggle(speed,callback) - 替换 show()+hide()
        * speed - 设置动画执行的时长,单位为毫秒
* callback - 动画执行完毕后的回调函数
* slideToggle(speed,callback) - 替换 slideUp()+slideDown()
  * 自定义动画
    * animate(param,duration,easing,callback)
      * param - 设置自定义动画的参数(CSS 的属性)
* duration - 可选项,设置动画执行的时长(单位为毫秒)
      * easing - 可选项,要使用的擦除效果的名称
      * callback - 可选项,动画执行完毕后的回调函数
    * animate(params,options)
      * params - 设置自定义动画的参数(CSS 的属性)
      * options - 选项
        * duration - 设置动画执行的时长(单位为毫秒)
* easing - 要使用的擦除效果的名称
* complete - 动画执行完毕后的回调函数
* queue - boolean 值,默认值为 true
  * 注意
    * 不能使用 CSS 中所有的背景颜色
  * 并发和排队效果
    * 并发效果 - 设置多个动画同时执行
      * 第一种 - animate(param,duration,easing,callback)

```
animate({
  name : value,
  name : value
},time);
```

      * 第二种 - animate(params,options)

```
animate({
  name : value
},{
  duration : time
}).animate({
  name : value
},{
  duration : time,
  queue : false
});
```

    * 排队效果 - 设置多个动画,按照先后顺序依次执行
      * 实现方式 - 多个 animate()的连续调用
* 类数组操作
  * 基本内容
  * 数组 - Array,用于存储多个内容(一维数组|二维数组)

```
var arr1 = new Array();
for(var i=0;i<arr1.length;i++){
    arr1[i] = new Array();
}
```

    * 类数组对象 - Object,存储方式类似于数组的结构
      * Arguments 对象 - 接收函数的参数
      * jQuery 对象 - 底层就是 DOM 对象
  * 类数组对象
    * 属性 - length 属性(数组的长度|元素的个数)
    * 方法
      * get(index)方法 - 根据 index 返回对应的 DOM 对象
      * eq(index)方法 - 根据 index 返回对应的内容
      * index(obj)方法 - 根据 obj 返回对应的索引值

```
    * 隐式迭代 - 显式迭代
      * 隐式迭代 - 1)要遍历谁 2)得到遍历后的结果
        * $().each() - jQuery 对象方法
* $.each()    - jQuery 全局函数
* 显式迭代 - 1)要遍历谁 2)如何遍历的 3)遍历的结果
        * while
* do..while
* for
* forin - for 的加强用法
* jQuery UI
  * 基本内容
    * jQuery UI - 必须基于 jQuery 使用
    * 使用 jQuery UI 的步骤(JS 文件的引入顺序不能改变)
      * 引入 jQuery 文件
      * 引入 jQuery UI 的 JS 文件
      * 引入 jQuery UI 的 CSS 文件
  * Effect(效果)
    * animate()方法
      * 扩展了 jQuery 中的 animate()方法
      * 允许操作 CSS 中所有有关背景的属性
      * 注意 - 背景颜色指定具体属性名称
        * backgroundColor
* borderBottomColor
* ...
    * effect()方法
      * 执行 jQuery UI 默认提供的动画效果的方法
    * hide()和 show()
  * Interactions(交互组件)
    * draggable() - 鼠标拖动指定元素
      * 选项
        *
      * 事件
* start - 鼠标开始拖动当前元素时触发(mouseover)
* drag  - 鼠标拖动当前元素,一直触发(mousemove)
* stop  - 鼠标结束拖动当前元素时触发(mouseout)
    * droppable() - 鼠标拖放指定元素
      * 选项
        * accept - 设置当前元素允许拖放的元素
      * 事件
        * drop - 拖放事件,必须事件
    * resizeable() - 鼠标缩放指定元素
    * sortable() - 鼠标排序指定元素
  * Widget(部件)
    * accordion() - 手风琴效果
    * autocomplete() - 自动提示
      * source - 设置提示内容的备选项
    * datepicker() - 日期控件
    * dialog() - 对话框
      * 选项
        * modal - 设置为 true,设置为模式对话框
      * 对话框弹出,页面其他元素变为不可操作
        * autoOpen - 设置是否自动打开对话框
      * false - 自动关闭
        * 事件
          * open - 打开对话框
  * close - 关闭对话框
    * tabs() - 多页签效果
* 扩展内容
  * WEB 前端开发遇到一些问题
    * 看到的效果,和实现的逻辑不一定是一致的
    * 代码逻辑、语法正确,最终的效果不一定不正确
  * 形参与实参
    * 形参 - 定义函数指定的参数
      function fn(a,b,c,d){}
    * 实参 - 调用函数指定的参数
      fn(1,2,3,4)
* 作业 - 用户登录注册 dailog()
<title>04_导航动态显示效果</title>
<script>
    /** 使用鼠标悬停事件效果
  * * mouseover 和 mouseout
  * * hover(over,out) - 模拟一个鼠标悬停事件
  *    * over - 用于模拟 mouseover 事件(处理函数)
  *    * out - 用于模拟 mouseout 事件(处理函数)*/
$("#navigation>ul>li").hover(function(){
      // 显示子菜单 - slideDown
      $(this).children("ul").slideDown("fast");
},function(){
      // 隐藏子菜单 - slideUp
      $(this).children("ul").slideUp("fast");
});
</script>
<title>06_简单自定义动画一</title>
<script>
    // 当鼠标点击 div 时,div 向右移动 200px
$("#panel").click(function(){
      /** animate(param)
        * * param - 格式是{key:value}
        * * duration - 设置时长*/
      $("#panel").animate({
            left : 200
      },3000).animate({
            top : 200
      },3000).fadeOut(3000);
});
</script>
---
<script>
    $("#panel").click(function(){
      /** animate(params,options)
        * * params - 格式为 {key:value}
```

```
        * * options - 格式为 {key:value} */
        $(this).animate({
            left : 200
        },{
            duration : 3000
        }).animate({
            top : 200
        },{
            duration : 3000
        }).fadeOut(3000);
    });
</script>
<title>08_广告动态轮播效果</title>
<script>
        // 定义 4 张为一组
        var rows = 4;
        // 初始化默认为第一组
        var page = 1;// page = 4
        // 为两个 button 按钮绑定 click 事件
        $(".prev").click(function(){
            // 图片向左移动
            var $parent = $(".v_content");
            var $list = $parent.children(".v_content_list");
            // 获取执行动画的元素
            var $ul = $list.children("ul");
            // 获取所有图片的个数
            var len = $list.find("li").length;
            // 获取移动的距离
            var width = $parent.width();//592px
            // 将所有图片分几组
            var max = Math.ceil(len/rows);
            // 判断 page 的最大值
            if(page == max){
                page = 0;
                $list.animate({
                    left : -(width*page)
                },"slow");
                page++;
            }else{
                $list.animate({// 向左移动
                    left : -(width*page)
                },"slow");
                page++;
            }
        $(".highlight_tip").children("span").eq(page-1).addClass("cur
rent").siblings().removeClass();
        });
        $(".next").click(function(){
            // 图片向右移动 - 作业
        });
<script>
        /* 并发效果一
        $("#panel").click(function(){
            $(this).animate({
                width : 300,
                height : 300
            },3000);
        });*/
        // 并发效果二
        $("#panel").click(function(){
            $(this).animate({
                width : 300
            },{
                duration : 3000
            }).animate({
                height : 300
            },{
                duration : 3000,
                queue : false
            });
        });
    </script>
<title>10_排队效果</title>
<script>
        $("#panel").click(function(){
            $(this).animate({
                left : 200
            },3000).animate({
                top : 200
            },3000).animate({
                width : 300
            },3000).animate({
                height : 300
            },3000).fadeOut(3000);
        });
<title>11_动态放大或缩小相册中的图片</title>
// 1. 如何将不同大小的图片,正确显示在相同大小的 div 中
        // 遍历 4 张图片
        $("img").each(function(index,domEle){
            // 得到每一张图片
            var $img = $(domEle);
            // 得到最外层的 div
            var $parent = $img.parent();
            // 添加中间层 div
            var $div = $("<div class='p_Img'></div>");
            $div.append($img);
            $parent.append($div);
            // 获取图片的宽度和高度
            var imgWidth = $img.width();
            var imgHeight = $img.height();
```

```
            var minWidth = $(".p_Img").width();
            var minHeight = $(".p_Img").height();
            // 2. 点击每张图片时,放大最大,缩小为初始化
            $img.click(function(){
                var $div = $(this).parent();
        if($div.width() == 90 && $div.height() == 90){
                    $(this).parent().animate({
                        width : imgWidth,
                        height : imgHeight
                    },3000);
                    $(this).parent().parent().animate({
                        width : imgWidth,
                        height : imgHeight
                    },3000);
                }else{
                    $(this).parent().animate({
                        width : minWidth,
                        height : minHeight
                    },3000);
                    $(this).parent().parent().animate({
                        width : minWidth,
                        height : minHeight+8
                    },3000);
                }
            });
        });
```

<title>13_扩展 animate 方法</title>

```
        $("#button").click(function(){
            $("#effect").animate({
                width : 480,
                height : 270,
                backgroundColor : "green"
            });
        });
```

<title>16_鼠标移动元素的选项</title>

```
/* * draggable(options)
  * * options - 格式为 {key:value}
  * * 选项
  *     * axis - 设置当前元素只能向 x 轴或 y 轴拖动
  *     * containment - 设置当前元素在指定区域中拖动
  *         * 'parent' - 指当前元素的父元素
  *         * 'document'或'window' - 指当前页面
  *         * 数组
  *             * 格式 - [x1,y1,x2,y2]
  *             * 绘制矩形,左上角(x1,y1),右下角(x2,y2)
  *         * 选择器
  *     * cursor - 设置拖动当前元素时,鼠标的样式
  *         * 参考 CSS 中的 cursor 属性的值
  *     * revert - 当拖动结束时,当前元素是否回到初始位置
  *         * 默认值为 false,如果设置为 true,回到初始位置.
  *     * snap - 设置拖动当前元素是否吸附到指定元素
  *         * Boolean - 设置为 true,吸附到所有其他元素
  *         * 选择器 - 设置当前元素吸附到指定元素
  *     * snapMode - 设置吸附的方式
  *         * inner - 吸附到指定元素的内部
  *         * outer - 吸附到指定元素的外部
  *         * 注意 - 该选项必须配合 snap 选项一起使用
  *     * handle 和 helper - 实现复制当前元素,拖动的是复制后的元素
  *     * grid - 设置当前元素每次拖动的距离
  *         * Array - [x,y]*/
    $("#draggable1").draggable({
        axis : "y"
    });
    $("#draggable2").draggable({
        axis : "x"
    })
    $("#draggable3").draggable({
        containment : "#containment-wrapper"
    })
    $("#draggable4").draggable({
        cursor : "pointer"
    });
    $("#draggable5").draggable({
        revert : true
    });
    $("#draggable6").draggable({
        snap : "#containment-wrapper"
    });
    $("#draggable7").draggable({
        snap : "#containment-wrapper",
        snapMode : "both"
    })
    $("#draggable8").draggable({
        handle : "p",
        helper : "clone"
    })
    $("#draggable9").draggable({
        grid : [50,50]
    })
```

<title>18_鼠标拖放元素</title>

```
    $("#draggable,#draggable-nonvalid").draggable({
        revert : true
    });
    $("#droppable").droppable({
        // accept 选项 - 设置当前元素允许接收拖放的元素
        accept : "#draggable",
        // drop 事件 - 拖放事件(必要事件)
        drop : function(event,ui){
            $(this).children("p").text("已经拖放.");
```

```
                $(this).append(ui.draggable);
            }
        });
```

<title>19_购物车案例</title>

```
        $( "#catalog" ).accordion();
        $( "#catalog li" ).draggable({
                appendTo: "body",
                helper: "clone"
            });
            $( "#cart ol" ).droppable({
                activeClass: "ui-state-default",
                hoverClass: "ui-state-hover",
                accept: ":not(.ui-sortable-helper)",
                drop: function( event, ui ) {
                    $( this ).find( ".placeholder" ).remove();
$( "<li></li>" ).text( ui.draggable.text() ).appendTo( this );
                }
            });
```

<title>28_操作对话框</title>

```
        // 设置样式为对话框
$("#dialog").dialog({
            // false - 自动关闭
            autoOpen : false,
            buttons : {
                "确认" : function(){
                    $("#dialog").dialog("close");
                },
                "取消" : function(){
                    $("#dialog").dialog("close");
                }
            }
});
$("#opener").button().click(function(){
            // 打开对话框
            $("#dialog").dialog("open");
});
```

<title>32_自定义右键菜单</title>

```
        // 禁止鼠标右键功能 - document|window 对象绑定
contextmenu 事件
        $(document).contextmenu(function(event){
            event.preventDefault();
        });
        $(document).mousedown(function(event){
            if(event.button == 2){
                // 鼠标右键
                $("#menu").removeAttr("style")
                                .menu()
                                .position({
                                    my: "left top",
                                    at: "left top",
                                    of : event,
                                    collision: "fit"
                                });
            }else if(event.button == 0){
                $("#menu").attr("style","display:none;");
            }
});
```

<title>33_JQUERYUI 网盘案例</title>

```
            // 登录、注册按钮样式
            $("button:lt(2)").button();
            // 工具栏按钮样式
            $("button:eq(2)").button({
                icons : {
                    primary: "ui-icon ui-icon-arrowthick-1-n"
                }
            }).next().button({
                icons : {
                    primary: "ui-icon ui-icon-folder-collapsed"
                }
            }).next().button({
                icons : {
                    primary: "ui-icon ui-icon-arrowthick-1-s"
                }
            }).next().button({
                icons : {
                    primary: "ui-icon ui-icon-trash"
                }
            }).next().button({
                icons : {
                    primary: "ui-icon ui-icon-extlink"
                }
            }).next().button({
                icons : {
                    primary: "ui-icon ui-icon-pencil"
                }
            });
            // 左侧菜单样式
            var icons = {
                header: "ui-icon-circle-arrow-e",
                activeHeader: "ui-icon-circle-arrow-s"
            };
            $( "#nav" ).accordion({
                icons: icons
            });
            $( "#toggle" ).button().click(function() {
                if ( $( "#nav" ).accordion( "option", "icons" ) ) {
                    "#nav" ).accordion( "option", "icons", null );
                else {
                     v" ).accordion( "option", "icons", icons );
                }
```

```
        });
        $("#progressbar").progressbar({ // 使用进度
            value : 20
        });
        // 拖动文件
    $("#sortable li").draggable({
            revert: "invalid",
            cursor: "move",
            helper: "clone",
            cursorAt: { top: 56, left: 56 }
        });
        // 回收站
        $( "#trash" ).droppable({
            accept: "#sortable > li",
            activeClass: "ui-state-hover",
            drop: function( event, ui ) {
                deleteImage(ui.draggable);
            }
        });
        function deleteImage($item){
            $item.fadeOut(function(){
$item.appendTo($("#trash")).fadeIn(function(){
                    $item.animate({
                        width : 100
                    });
                });
            });
        }
        var availableTags = [ // 自动提示
            "ActionScript",
            "AppleScript",
            "Asp",
            "BASIC",
            "C",
            "C++",
            "Clojure",
            "COBOL",
            "ColdFusion",
            "Erlang",
            "Fortran",
            "Groovy",
            "Haskell",
            "Java",
            "JavaScript",
            "Lisp",
            "Perl",
            "PHP",
            "Python",
            "Ruby",
            "Scala",
            "Scheme"
        ];
        $( "#search-text" ).autocomplete({
            source: availableTags
        });
        // 登录和注册功能
        $("#userLogin").dialog({
            autoOpen : false,
            modal : true,
            width : 350,
            buttons : {
                "登录" : function(){
                    $(this).dialog("close");
                }
            }
        });
        $("button:eq(0)").click(function(){
            $("#userLogin").dialog("open");
        });
        $("#userRegist").dialog({
            autoOpen : false,
            modal : true,
            width : 350,
            buttons : {
                "注册" : function(){
                    $(this).dialog("close");
                }
            }
        });
        $("button:eq(1)").click(function(){
            $("#userRegist").dialog("open");
        });
        // 自定义右键菜单
        $(document).contextmenu(function (event) {
            event.preventDefault();
        });
        $(document).mousedown(function (event) {
            if(event.button == 2){
$( "#menu").removeAttr("style").menu().position({
                    my: "left top",
                    at: "left top",
                    of : event,
                    collision: "fit"
                });
            }else if(event.button == 0){
                $("#menu").attr("style","display:none");
            }
        });
```

JQUERY DAY04:

* 基本内容
  * 组件|插件|组件化开发
  * 优点
    * 开发更灵活
    * 低耦合 - 代码之间的关系越来越小
  * jQuery 插件的作用
    * 扩展 jQuery 的功能
    * 呈现组件化特点
  * jQuery 插件的目的
    * 掌握学习的插件的使用方式
    * 掌握如何学习新的 jQuery 插件
* 日期插件 - layDate 插件
  * 问题
    * 目前更多的日期插件,已经脱离 jQuery
  * layDate 插件
    * 将 laydate 文件夹拷贝到工程目录中
    * laydate 目录中的结构不能被改变的
  * layDate 初步使用
    * 在 input 输入框中增加对应内容
      * 指定 class 的值为 laydate-icon
* 绑定 onclick 事件,指定 laydate()方法
    * 定义 input 输入框
      * 指定 class 的值为 laydate-icon
* 通过 JS 代码绑定 click|focus 事件,指定 laydate()方法
  * laydate()方法的选项
    * elem - 指定日期控件显示的元素
* 表单验证插件
  * 引入文件
    * 引入 validation 插件的 CSS 文件
    * 引入 jQuery 文件
    * 引入 validation 插件的 JS 文件
    * 引入 validation 插件提供的中文文件
  * 核心方法 validate()
    * 作用 - 用于表单验证
  * validate()方法的选项
    * rules - 定义验证规则
    * messages - 自定义错误提示信息
  * 自定义验证方法

    addMethod(name,function(value,element,params){})
* 瀑布流插件 - Masonry
  * 如何使用 Masonry 插件
    * 先引入 jQuery 文件 * 再引入 Masonry 文件
  * 核心方法 masonry()
    * 默认调用 - 实现瀑布流效果
  * 使用 Masonry 插件的方式
    * HTML 页面
      * div 元素作为瀑布流布局的容器
* 所有显示为瀑布流内容,放置在容器中
    * 调用 Masonry 插件
      * 作为容器的 div 调用 masonry()
  * masonry()的选项
    * 推荐
      * itemSelector - 指定使用哪些元素进行布局
* columnWidth - 设置每列的宽度
    * 布局
      * precentPosition - 设置是否使用百分值
* Boolean 值,如果为 true 表示使用百分值
      * stamp - 标记元素
* 固定在具体位置,布局元素在固定元素下进行布局
* isFitWidth - 设置作为容器的宽度,以适应布局元素的总宽度
  * masonry 的方法 - Masonry 对象
    * appended - 向瀑布流尾部添加元素
    * prepend - 向瀑布流头部添加元素
    * remove - 从瀑布流删除指定的元素
  * masonry 的事件
    * 使用 jQuery 方式进行绑定
      * bind(type,callback)
* on(type,callback)
    * masonry 的事件
      * layoutComplete - 当完成布局时触发
  * 注意
    * 指的是添加元素后,完成布局时的触发
    * 并不是初始化布局完成
  * 使用
    * 配合 appended 和 prepended 方法使用
* removeComplete - 当删除元素时触发
  * 使用 * 配合 remove 方法使用
* 学习 jQuery 插件的方式
* 官方提供 Demo 示例 * 习惯官方提供的 API 帮助文档
  * 使用 jQuery 插件的步骤
    * 注意 - 与 jQuery UI 一致,都是基于 jQuery 实现
    * 步骤
      * 引入 jQuery 文件 * 引入 jQuery 插件文件
  * 开发插件
    * 插件的分类
      * 全局函数插件 - $.each()
        * 和 HTML 页面的元素是无关的
      * 对象方法插件 - $().each()
        * 和 HTML 页面的元素是有关的
      * 选择器插件 - 目前几乎不使用
        * XPath 插件 - 用于解析 XML 文件
    * 插件的机制
      * 全局函数
        * jQuery 官方提供 - jQuery.extend(object)
        * 自己研究 - $.方法名 = function(){}
      * 对象方法
        * jQuery 官方提供 - jQuery.fn.extend(object)
        * 自己研究 - $.fn.方法名 = function(){}

```
    * 开发插件的目的
      * 将来工作中需要自定义 jQuery 插件
      * 为了将来的面试(加分)
  * 扩展内容   * JSON 数据格式
      * Object {
            key : Object,
            key : value
          }
      * Array - [1,2,3,4,5,...]
      * 注意->   * 允许嵌套    * 一般三层
    * 遍历
      for(){
        for(){
          for(){}
        }
      }
    * 中国开发一定遇到的问题=>* 中文乱码
    * WEB 时代
      * web 1.0 - 以内容为主的网站(门户网站)
      * web 2.0 - 以人的关系为主的网站(社交网站)
      * web 3.0 - 基于移动互联网的社交网站(微信)
    * 计算机 3 次革命
      * PC 机的普及 - 联想等   * 互联网的普及 - 新浪等
      * 移动互联网的普及
<title>02_laydate 插件的高级使用</title>
          // input 元素绑定事件 1)click 事件;2)focus 事件
      $("#mydate").click(function(){
            /* * laydate(options)方法
             * * options - 格式是 {key:value}
             *     * istime - 是否开启时间选择
             *     * isclear - 是否开启清空按钮
             *     * istoday - 是否开启今天按钮
             *     * issure - 是否开启确认按钮
             *     * formate - 设置日期格式
             * * 默认值 - yyyy-MM-dd hh:mm:ss(通用格式)
             *       * 格式建议使用默认
             *     * min - 设置最小日期,默认为 1900(1970)
             *     * max - 设置最大日期,默认为 2099
             *     * start - 设置开始日期   */
            laydate({
                istime : false,     isclear : false,
                istoday : false,    issure : false,
                //format : "YYYY 年 MM 月 DD 日",
                min : "1970-01-01 00:00:00",
                max : "2020-12-31 00:00:00",
                start : "2015-12-01 00:00:00"
            });
      });
<title>04_validation 自定义错误</title>
/* * validate(options)方法
     * * messages - 自定义错误提示信息
     *     * value
     * 格式为 {key:value}    * 指定的是元素的 name 属性值
     * messages 会覆盖 validate()方法底层提供的错误信息 */
    $("#commentForm").validate({
          messages : { email : "请输入你的 Email 地址." }
    });
<title>07_validation 插件案例</title>
    /** validation 插件自定义验证规则
     * $.validator.addMethod(name,method,message)方法
     * * name - 设置自定义验证方法的名称
     * * method - 是一个函数,编写具体的验证逻辑
     *     function(value,element,params){}
     *     * value - 验证元素的 value 属性值
     *     * element - 验证的元素(DOM 对象)
     *     * params - 调用自定义验证规则的 value 值
     * * message - 自定义错误提示信息 */
    $.validator.addMethod("checkUser",function(value,element,
params){
          // 使用正则验证 user 必须是英文+数字
          var regExp = /^[a-zA-Z0-9]+$/;
          return regExp.test(value);
    });
    $.validator.addMethod("checkPwd",function(value,element,
params){
          var regExp = /^[a-zA-Z]+$/;
          return regExp.test(value);
    });
    $.validator.addMethod("phoneCN",function(value,element,p
arams){
          var regExp = /^1[0-9]{12}$/;
          return regExp.test(value);
    });
    $.validator.addMethod("regExp",function(value,element,par
ams){
          //var regExp = params;
          return params.test(value);
    });
    /* * 需求 * * 用户名 - 不能为空,6-12 之间英文+数字
     * * 密码 - 不能为空,6-8 之间英文
     * * 确认密码 - 两次密码输入一致
     * * email - 不能为空,格式正确
     * * 手机号 - 不能为空,格式正确
     * validate(options)方法
     * * rules - 设置元素的验证规则
     *     * key - 指定验证元素的 name 属性值
     *     * value - 设置元素的验证规则
     *       * 一种验证规则 - 定义验证规则名称
     *       * 多种验证规则 - {key:value}
     *         * key - 验证规则名称
```

```
*          * value -  规则内容
* * messages -  设置错误提示信息
*     * key -  指定验证元素的 name 属性值
*     * value -  设置错误提示信息
*        *  一种验证规则,直接定义错误提示信息
*        *  多种验证规则,使用{key:value}
*          * key -  验证规则名称
*          * value -  错误提示信息        */
   $("#myform").validate({
     rules : {
           user : {
                 "required" : true,
                 "rangelength" : [6,12],
                 //"checkUser" : true
                 "regExp" : /^[a-zA-Z0-9]+$/
           },pwd : {
                 "required" : true,
                 "rangelength" : [6,8],
                 "checkPwd" : true
           },repwd : {
                 "required" : true,
                 "rangelength" : [6,8],
                 "checkPwd" : true,
                 "equalTo" : "#pwd"
           },mail : {
                 "required" : true,
                 "email" : true
           },phone : {
                 "required" : true,
                 "phoneCN" : true
           },
           /* *  单选框或多选框的验证
            * *  验证规则只需要使用"required"
* *  使用 messages 定义错误信息显示位置出现问题
*     *  在 HTML 元素对应正确的位置上定义<label>
                *          * class="error"
                *          * for=验证元素的 name 属性值
                *          * <label>元素内不能包含任何内容*/
           gender : "required",
           edu : "required"
     },messages : {
           user : {
                 "required" : "请输入你的用户名.",
                 "rangelength" : "用户名长度不正确.",
                 "regExp" : "用户名输入不正确."
           },pwd : {
                 "required" : "请输入你的密码",
                 "rangelength" : "密码的长度不正确",
                 "checkPwd" : "密码输入不正确"
           },repwd : {
                 "required" : "请输入你的密码",
                 "rangelength" : "密码的长度不正确",
                 "checkPwd" : "密码输入不正确",
                 "equalTo" : "两次密码输入不一致"
           },mail : {
                 "required" : "请输入你的 Email",
                 "email" : "输入的 Email 格式不正确"
           },
           phone : {
                 "required" : "请输入你的手机号",
                 "phoneCN" : "手机号输入不正确"
           },gender : "请选择你的性别",
           edu : "请选择你的学历"
     }
});
```

<title>12_masonry 添加元素</title>

```
    //$("#basic").masonry();
/* *  通过对象方式创建瀑布流布局
 * new Masonry(容器 div,选项) */
var masonry = new Masonry("#basic");
//  点击 append 按钮,向布局元素中添加 3 个 div 元素
$(".append-button").click(function(){
     //  创建 3 个元素
     var divs = [createElem(),createElem(),createElem()];
     //console.log(masonry.$element);
     //  不能再使用 jQuery 添加元素
     $.each(divs,function(index,result){
          // masonry.$element - jQuery 对象
          masonry.$element.append(result);
          //  插件对象调用 appended
          masonry.appended(result);
     });
});
$("#basic").on("layoutComplete",function(){
     console.log("已经布局完成....");
});
function createElem(){ //  创建 div 元素的函数
     var $div = $("<div class='item'></div>");
     var rand = Math.random();//0-1
     if(rand < 0.5){ $div.addClass("h2");
     }else if(rand < 0.75){ $div.addClass("h3");
     }else{$div.addClass("h5");     }
     return $div;
}
```

<title>15_瀑布流案例</title>

```
new AnimOnScroll( document.getElementById( 'grid' ), {
     minDuration : 0.4,        maxDuration : 0.7,
     viewportFactor : 0.2
} );
```