day29-JQ

今日内容介绍

- JQ语法知识
  - JQ快速入门(引入)
  - JQ和JS对象互相转换(必须掌握)
    - js--->jq
    - jq--->js
  - JQ事件的使用(掌握常用事件)
  - JQ动画(了解)
  - JQ选择器(掌握基本选择器,属性选择器,....)
  - JQ操作样式,操作属性,操作Dom(掌握)
  - JQ遍历(掌握)
- JQ案例
  - 定时弹出广告
  - 隔行换色
  - 复选框全选
  - 省市联动
  - 表格换色
  - 电子时钟

第一章-JQ知识点

1.1 JQ介绍-了解

jQuery的概述

jQuery是一个优秀的javascript库(类似Java里面的jar包)，兼容css3和各大浏览器，提供了dom、events、animate、ajax等简易的操作。 并且jquery的插件非常丰富，大多数功能都有相应的插件解决方案。jquery的宗旨是 write less, do more。

说白了: JQ就是js库, 封装了JS常见的操作,我们使用JS起来更加的简单 (特别是dom这块)

jQuery的作用

jQuery最主要的作用是简化js的Dom树的操作

jQuery框架的下载

jQuery的官方下载地址：http://www.jquery.com

jQuery的版本

- 1.x：兼容IE678，使用最为广泛的，官方只做BUG维护，功能不再新增。因此一般项目来说，使用1.x版本就可以了，最终版本：1.12.4 (2016年5月20日)

- 2.x：不兼容IE678，很少有人使用，官方只做BUG维护，功能不再新增。如果不考虑兼容低版本的浏览器可以使用2.x，最终版本：2.2.4 (2016年5月20日)
- 3.x：不兼容IE678，很多老的jQuery插件不支持这个版本。目前该版本是官方主要更新维护的版本.
- 注:开发版本与生产版本，命名为jQuery-x.x.x.js为开发版本，命名为jQuery-x.x.x.min.js为生产版本，开发版本源码格式良好，有代码缩进和代码注释，方便开发人员查看源码，但体积稍大。而生产版本没有代码缩进和注释，且去掉了换行和空行，不方便发人员查看源码，但体积很小

1.2 JQ快速入门

1.需求

等页面加载完成之后 弹出 hello...

2.步骤

1. 拷贝jq库到项目
2. 把jq引入页面
3. 根据jq语法, 弹出hello Jquery...

3.实现

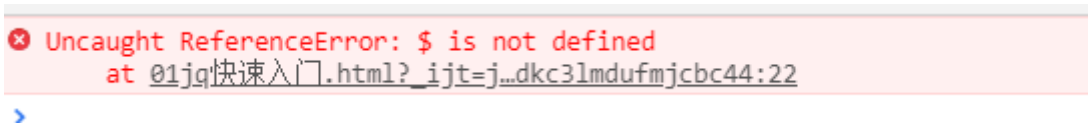
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>1_JQ快速入门</title>
    <script src="js/jquery-3.3.1.min.js"></script>
</head>
<body>

</body>
<script>
    // 页面加载完成就弹出hello...
    jQuery(document).ready(function () {
        alert("hello1...");
    });

    // 省略格式
    $(document).ready(function () {
        alert("hello2...");
    });

    // 省略格式---->掌握
    $(function () {
        alert("hello3...");
    });

</script>
</html>
```

4.小结

4.1步骤

1. 拷贝jq到项目
2. 把jq引入页面
3. jq语法 $(...)

4.2注意事项



90%以上的几率是JQ库没有导入或者JQ库路径写错了

1.3 JQ和JS对象转换【重点】

对象说明

JS对象: document.getElemxxx() 获得的都是JS对象 **大部分都是属性 eg:innerHTML....**

JQ对象: $(选择器\js对象) **大部分都是方法**

jQuery本质上虽然也是js，但如果使用jQuery的属性和方法那么必须保证对象是jQuery对象而不是js方式获得的DOM对象。使用js方式获取的对象是js的DOM对象，使用jQuery方式获取的对象是jQuery对象。两者的转换关系如下

转换语法

js的DOM对象转换成jQuery对象，==语法：**$(js对象)**==

jQuery对象转换成js对象，==语法：**jquery对象[索引]**== 或 **jquery对象.get(索引)**; 一般索引写0

```
<head>
    <meta charset="UTF-8">
    <title>02_JQ和JS对象转换</title>
    <script src="js/jquery-3.3.1.min.js"></script>
</head>
<body>

<span id="spanId">span...</span><br/>
<input type="button" value="往span标签插入内容(jquery对象)" onclick="fun1()">
<input type="button" value="往span标签插入内容(js对象)" onclick="fun2()">

</body>
<script>

    function fun1() {
        // 1.获取span标签对象--->js对象
        var jsSpan = document.getElementById("spanId");

        // 2.js对象转换为jquery对象: $(js对象)
        var jqSpan = $(jsSpan);

        // 3.使用jquery对象往span标签插入内容
        jqSpan.html("js--->jq");

    }


    function fun2() {
```

```
        // 1.获取span标签对象--->jq对象
        var jqSpan = $("#spanId");

        // 2.jq对象转换为js对象: jquery对象[0]   或者  jquery对象.get(0)
        var jsSpan = jqSpan[0];

        // 3.使用js对象往span标签插入内容
        jsSpan.innerHTML = "jq--->js";
    }
</script>
```

1.4 JQ中事件的使用【重点】

基本事件的使用【重点】

- 事件在jq里面都封装成了方法. 事件名就是去掉了JS里面on. 语法:

```
jq对象.事件方法名(function(){});
```

- 点击事件

```
<body>
<input type="button" value="jq点击事件" id="inputId">
</body>
<script>

    <!--事件绑定:  jq对象.事件方法名(匿名函数)-->
    $("#inputId").click(function () {
        alert("jq点击事件....")
    });
</script>
```

- 获得焦点和失去焦点

```
<body>
<input type="text" name="username" id="inputId">
</body>
<script>
    // 获得焦点
    $("#inputId").focus(function () {
        console.log("获得焦点...");
    });

    // 失去焦点
    $("#inputId").blur(function () {
        // this: js对象  谁失去焦点,this就表示谁的js对象
        console.log("失去焦点...获得文本输入框输入的内容js:"+this.value);

        // this:js对象 ---->转换为jq对象 $(this)
        console.log("失去焦点...获得文本输入框输入的内容jq:"+$(this).val());
    });

</script>
```

- 内容改变事件

```
<body>
<select id="starSelectId">
    <option value="Jordan">乔丹</option>
    <option value="James">詹姆斯</option>
    <option value="Kobe">科比</option>
    <option value="Iverson">艾弗森</option>
</select>
</body>
<script>
    // 为下拉框绑定一个内容改变事件
    $("#starSelectId").change(function () {
        // 获得下拉框改变后的值
        console.log("获得下拉框改变后的值js:"+this.value);
        console.log("获得下拉框改变后的值jq:"+$(this).val());
    });

</script>
```

- 鼠标相关的事件

```
<body>
 <div  id="divId" style="border: 1px solid red">1111</div>
</body>
<script>
    // 为div绑定一个鼠标移入事件
    $("#divId").mouseover(function () {
        console.log("鼠标移入...")
    });

    // 为div绑定一个鼠标移出事件
    $("#divId").mouseout(function () {
        console.log("鼠标移出...")
    });

</script>
```

- 键盘相关事件

```
<body>
<input id="inputId" type="text"/>
</body>
<script>
    // 为文本输入框绑定一个键盘按下事件 keydown
    $("#inputId").keydown(function () {
        console.log("键盘按下...")
    });

    // 为文本输入框绑定一个键盘弹起事件 keyup
    $("#inputId").keyup(function () {
        console.log("键盘弹起...")
    });
</script>
```

jQuery的事件绑定与解绑

- 事件的绑定

```
jq对象.on(事件名,function(){} );
其中：事件名称是jQuery的事件方法的方法名称，不带小括号,例如：click、mouseover、mouseout、focus、blur等
```

- 事件的解绑

```
jQuery元素对象.off(事件名称);
其中：参数事件名称如果省略不写，可以解绑该jQuery对象上的所有事件
```

- 实例代码

```
<body>
<input type="button" value="jq绑定点击事件" id="btn01">
<input type="button" value="jq解绑点击事件" id="btn02">
</body>
<script>

    <!--事件绑定:  jq对象.事件方法名(匿名函数)-->
   /* $("#btn01").click(function () {
        alert("jq点击事件1....")
    });*/

    <!--事件绑定:  jq对象.on(事件方法名,匿名函数)    注意:事件方法名不带括号-->
    $("#btn01").on("click",function () {
        alert("jq点击事件2....")
    })

    /*
        jQuery元素对象.off(事件名称);
        其中：参数事件名称如果省略不写，可以解绑该jQuery对象上的所有事件
    */
    // 需求: 点击第二个按钮,解绑第一个按钮的点击事件
    $("#btn02").on("click",function () {
        // 解绑第一个按钮的点击事件
        // $("#btn01").off();// 解绑btn01按钮上的所有事件
        $("#btn01").off("click");// 解绑btn01按钮上的点击事件
    });

</script>
```

事件切换

- 普通写法

```
<body>
<div  id="divId" style="border: 1px solid red">1111</div>
</body>
<script>
    // 普通写法
    // 为div绑定一个鼠标移入事件
    $("#divId").mouseover(function () {
```

```
        console.log("鼠标移入...")
    });

    // 为div绑定一个鼠标移出事件
    $("#divId").mouseout(function () {
        console.log("鼠标移出...")
    });

</script>
```

- 链式写法: 调用了事件, 返回是当前的标签对象

```
<body>
<div  id="divId" style="border: 1px solid red">1111</div>
</body>
<script>
    // 链式写法
    // 为div绑定一个鼠标移入事件
    // 为div绑定一个鼠标移出事件
    // 类似java中链式编程: new StringBuilder("java").append().append()...;
    $("#divId").mouseover(function () {
        console.log("鼠标移入...")
    }).mouseout(function () {
        console.log("鼠标移出...")
    });

</script>
```

1.5 JQ动画【了解】

基本效果

- 方法

方法名称	解释
show([speed],[easing],[fn]])	显示元素方法
hide([speed,[easing],[fn]])	隐藏元素方法
toggle([speed],[easing],[fn])	切换元素方法，显示的使之隐藏，隐藏的使之显示

- 参数

参数名称	解释
speed	三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)
easing	用来指定切换效果，默认是"swing"，可用参数"linear"
fn	在动画完成时执行的函数，每个元素执行一次

- 实例

```
<body>
<img src="../img/b.jpg" height="407" width="310" id="imgId" hidden/><br/>
<input type="button" value="show()显示" id="btn01">
<input type="button" value="hide()隐藏" id="btn02">
<input type="button" value="toggle()切换" id="btn03">
</body>
<script>
    $("#btn01").click(function () {
        // 让img显示
        $("#imgId").show(1000,function () {
            alert("显示完毕...")
        });
    });

    $("#btn02").click(function () {
        // 让img隐藏
        $("#imgId").hide(3000);
    });

    $("#btn03").click(function () {
        // 切换显示和隐藏
        $("#imgId").toggle(3000);
    });

</script>
```

滑动效果

- 方法

方法名称	解释
slideDown([speed,[easing],[fn]])	向下滑动方法
slideUp([speed,[easing],[fn]])	向上滑动方法
slideToggle([speed],[easing],[fn])	切换元素方法，显示的使之隐藏，隐藏的使之显示

- 参数

参数名称	解释
speed	三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)
easing	用来指定切换效果，默认是"swing"，可用参数"linear"
fn	在动画完成时执行的函数，每个元素执行一次

```
<body>
<img src="../img/b.jpg" height="407" width="310" id="imgId" hidden/><br/>
<input type="button" value="slideDown()向下滑动" id="btn01">
<input type="button" value="slideUp()向上滑动" id="btn02">
<input type="button" value="slidetoggle()切换" id="btn03">
</body>
<script>
    $("#btn01").click(function () {
        // 让img向下滑动
        $("#imgId").slideDown(2000);
    });

    $("#btn02").click(function () {
        // 让img向上滑动
        $("#imgId").slideUp(3000);
    });

    $("#btn03").click(function () {
        // 切换向下滑动和向上滑动
        $("#imgId").slideToggle(1000);
    });

</script>
```

淡入淡出效果

- 方法

方法名称	解释
fadeIn([speed,[easing],[fn]])	淡入显示方法
fadeOut([speed,[easing],[fn]])	淡出隐藏方法
fadeToggle([speed],[easing],[fn])	切换元素方法，显示的使之隐藏，隐藏的使之显示

- 参数

参数名称	解释
speed	三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)
easing	用来指定切换效果，默认是"swing"，可用参数"linear"
fn	在动画完成时执行的函数，每个元素执行一次

```
<body>
<img src="../img/b.jpg" height="407" width="310" id="imgId" hidden/><br/>
<input type="button" value="fadeIn()淡入" id="btn01">
<input type="button" value="fadeOut()淡出" id="btn02">
<input type="button" value="fadeToggle()切换" id="btn03">
</body>
<script>
    $("#btn01").click(function () {
        // 让img淡入
        $("#imgId").fadeIn(5000);
    });

    $("#btn02").click(function () {
        // 让img淡出
        $("#imgId").fadeOut(2000);
    });

    $("#btn03").click(function () {
        // 切换淡入和淡出
        $("#imgId").fadeToggle(1000);
    });
</script>
```

1.6 JQ选择器

基本选择器【重点】

- 语法

选择器名称	语法	解释
标签选择器（元素选择器）	$("html标签名")	获得所有匹配标签名称的于元素
id选择器	$("#id的属性值")	获得与指定id属性值匹配的元素
类选择器	$(".class的属性值")	获得与指定的class属性值匹配的元素

```
<body>
<!--获得完成之后，改变对应标签的背景色为红色-->
<input type="button" value="选择id为one的" id="btn01"/>
<input type="button" value="选择标签为div的" id="btn02"/>
<input type="button" value="选择类名是mini的" id="btn03"/>
<hr/>
```

```
<div id="one">
    <div class="mini" title="one">
        111
    </div>
</div>

<div id="two">
    <div class="mini" title="one">
        222
    </div>
    <div class="mini" title="two">
        333
    </div>
</div>

<div id="three">
    <div class="mini">
        444
    </div>
    <div class="mini">
        555
    </div>
    <div class="mini">
        666
    </div>
</div>

<span id="four"></span>



</body>
<script>
    // 基本选择器: 标签选择器,id选择器,类选择器
    $("#btn01").click(function () {
        // 把id为one的div背景色改成红色
        $("#one").css("background","red");
    });

    // 把标签名为div的背景色改成红色
    $("#btn02").click(function () {
        $("div").css("background","red");
    });

    // 把class属性值为mini的div的背景色改成红色
    $("#btn03").click(function () {
        $(".mini").css("background","red");
    });
</script>
```

层级选择器

- 语法

选择器名称	语法	解释
==后代选择器==	$("A B ")	选择A元素内部的所有B元素
子选择器	$("A > B")	选择A元素内部的所有B子元素
兄弟选择器	$("A + B")	获得A元素同级的下一个B元素
兄弟选择器	$("A ~ B")	获得A元素同级的所有B元素

```
<body>
<input type="button" value="选择body下所有的div" id="btn01"/>
<input type="button" value="选择body下所有子div" id="btn02"/>
<input type="button" value="选择id为one后的第一个div兄弟标签" id="btn03"/>
<input type="button" value="选择id为one后的所有div兄弟标签" id="btn04"/>
<hr/>
<div id="one">
    <div class="mini" title="one">
        111
    </div>
</div>

<div id="two">
    <div class="mini" title="one">
        222
    </div>
    <div class="mini" title="two">
        333
    </div>
</div>

<div id="three">
    <div class="mini">
        444
    </div>
    <div class="mini">
        555
    </div>
    <div class="mini">
        666
    </div>
</div>

<span id="four"></span>

</body>
<script>
    $("#btn01").click(function () {
        // 选择body下所有的div,背景色改为红色
        $("body div").css("background","red");
    });


    $("#btn02").click(function () {
        // 选择body下所有子div,背景色改为红色
        $("body > div").css("background","red");
    });
```

```
    $("#btn03").click(function () {
        // 选择id为one后的第一个div兄弟标签,背景色改为红色
        $("#one + div").css("background","red");
    });

    $("#btn04").click(function () {
        // 选择id为one后的所有div兄弟标签,背景色改为红色
        $("#one ~ div").css("background","red");
    });
</script>
```

属性选择器

- 语法

选择器名称	语法	解释
属性选择器	$("A[属性名]")	包含指定属性的选择器
属性选择器	$("A[属性名=值]")	包含指定属性等于指定值的选择器

```
<body>
<!--获得完成之后, 改变对应标签的背景色为红色-->
<input type="button" value="选择含有title属性的div" id="btn01"/>
<input type="button" value="选择含有title属性的div,并且属性值为one的div" id="btn02"/>
<hr/>
<div id="one">
    <div class="mini" title="one">
        111
    </div>
</div>

<div id="two">
    <div class="mini" title="one">
        222
    </div>
    <div class="mini" title="two">
        333
    </div>
</div>

<div id="three">
    <div class="mini">
        444
    </div>
    <div class="mini">
        555
    </div>
    <div class="mini">
        666
    </div>
</div>

<span id="four"></span>

</body>
```

```
<script>
    $("#btn01").click(function () {
        // 选择含有title属性的div,背景色设置为红色
        $("div[title]").css("background","red");
    });

    $("#btn02").click(function () {
        // 选择含有title属性的div,并且属性值为one的div,背景色设置为红色
        $("div[title='one']").css("background","red");
    });
</script>
```

基本过滤选择器

- 语法

选择器名称	语法	解释
首元素选择器	:first	获得选择的元素中的第一个元素
尾元素选择器	:last	获得选择的元素中的最后一个元素
非元素选择器	:not(selecter)	不包括指定内容的元素
偶数选择器	:even	偶数，从 0 开始计数
奇数选择器	:odd	奇数，从 0 开始计数
等于索引选择器	:eq(index)	指定索引元素
大于索引选择器	:gt(index)	大于指定索引元素
小于索引选择器	:lt(index)	小于指定索引元素

```
<body>
<!--获得完成之后，改变对应标签的背景色为红色-->
<input type="button" value="选择第一个div" id="btn01"/>
<input type="button" value="选择最后一个div" id="btn02"/>
<input type="button" value="选择偶数div" id="btn03"/>
<input type="button" value="选择奇数div" id="btn04"/>
<input type="button" value="选择第3个div" id="btn05"/>
<hr/>
<div id="one">
    <div class="mini" title="one">
        111
    </div>
</div>

<div id="two">
    <div class="mini" title="one">
        222
    </div>
    <div class="mini" title="two">
        333
    </div>
</div>
```

```
<div id="three">
    <div class="mini">
        444
    </div>
    <div class="mini">
        555
    </div>
    <div class="mini">
        666
    </div>
</div>

<span id="four"></span>

</body>
<script>
    $("#btn01").click(function () {
        // 选择第一个div
        $("div:first").css("background","green");
    });

    $("#btn02").click(function () {
        // 选择最后一个div
        $("div:last").css("background","green");
    });

    $("#btn03").click(function () {
        // 选择偶数div
        $("div:even").css("background","green");
    });

    $("#btn04").click(function () {
        // 选择偶数div
        $("div:odd").css("background","green");
    });

    $("#btn05").click(function () {
        // 选择第三个div,索引为2
        $("div:eq(2)").css("background","green");
    });
</script>
```

表单属性选择器

- 语法

选择器名称	语法	解释
可用元素选择器	:enabled	获得可用元素
不可用元素选择器	:disabled	获得不可用元素
选中选择器	:checked	获得单选/复选框选中的元素
选中选择器	:selected	获得下拉框选中的元素

```
<body>
<!--获得完成之后，打印选中复选框的值-->
<input type="button" value="获得选中复选框的值" id="btn01"/>
<hr/>
<input type="checkbox" value="basketball" checked="checked">篮球<br/>
<input type="checkbox" value="football">足球<br/>
<input type="checkbox" value="badminton" checked="checked" >羽毛球<br/>
<input type="checkbox" value="ping-pong">兵乒球<br/>

</body>
<script>
    $("#btn01").click(function () {

        // 1.获得复选框选择的选项
        // var jqCkBox = $("input[type='checkbox']:checked")

        // 2.获得值--->复选框只选择了一个
        // console.log(jqCkBox.val());

        // ........................................

        // 1.获得复选框选择的选项
        var jqCkBoxArr = $("input[type='checkbox']:checked")

        // 2.获得值--->复选框选择了多个
        jqCkBoxArr.each(function (i, e) {
            // 注意:遍历出来的e是js对象,表示js标签对象
            console.log(i)
            console.log($(e).val());
        });
    });
</script>
```

1.7 JQ操作样式

API方法	解释
css(name)	获取CSS样式
css(name,value)	设置CSS样式

```
<body>
<div id="divId" style="width: 100px;height: 100px;background-color: red"></div>
<input type="button" value="获得背景色的样式值" id="btn01"/>
<input type="button" value="设置背景色的样式值为green" id="btn02"/>

</body>
<script>
    $("#btn01").click(function () {
        // 获得div背景色的样式值
        console.log($("#divId").css("background"));
    });

    $("#btn02").click(function () {
        // 设置背景色的样式值为green
        $("#divId").css("background","green");
```

```
    });

</script>
```

1.8 JQ操作属性

API方法	解释
attr(name,[value])	获得/设置属性的值
prop(name,[value])	获得/设置属性的值(checked，selected)

- attr与prop的注意问题
  - attr和prop分界点 1.6
  - checked 和 selected 建议 使用prop获取, 其他使用attr获取

```
<body>
<img src="img/b.jpg"><br/>
<input id="btn01" type="button"  value="获取图片的src属性值"><br/>
<input id="btn02" type="button"  value="修改图片的src属性值"><br/>

</body>
<script>
    $("#btn01").click(function () {
        // 获取图片的src属性值
        console.log($("img").attr("src"));
        // console.log($("img").prop("src"));
    });

    $("#btn02").click(function () {
        // 修改图片的src属性值
        $("img").attr("src","img/a.gif");
        // $("img").prop("src","img/a.gif");
    });
</script>
```

1.9 使用JQ操作DOM

jQuery对DOM树中的文本和值进行操作

- API

API方法	解释
val([value])	获得/设置标签里面value属性相应的值
text([value])	获得/设置元素的文本内容
html([value])	获得/设置元素的标签体内容

- 解释

```
val()          获得标签里面value属性的值    value属性的封装
val("...")     给标签value属性设置值

text()            获得标签的内容，如果有标签，忽略标签。
text("...")       设置文本，如果含有标签，把标签当做字符串.不支持标签

html()         获得标签的内容，如果有标签，一并获得。
html("....)    设置html代码，如果有标签，将进行解析。支持标签 封装了innerHTML
```

- val()和val(...)

```
<body>
<input type="text" value="hello..." id="inputId"/><br/>
<input type="submit" value="获得value" id="btn01"/>
<input type="submit" value="设置value" id="btn02"/>
</body>
<script>
    $("#btn01").click(function () {
        // 1.获得文本输入框对象
        // 2.获取文本输入框value的值
        console.log($("#inputId").val());
    });

    $("#btn02").click(function () {
        // 1.获得文本输入框对象
        // 2.获取文本输入框value的值
        $("#inputId").val("hello jquery...");
    });
</script>
```

- text()和text(...)

```
<body>
<p id="pId">段落</p>
<input type="submit" value="获得text" id="btn01"/>
<input type="submit" value="设置text" id="btn02"/>
</body>
<script>
    $("#btn01").click(function () {
        // 获得p标签的文本
        console.log($("#pId").text());
    });

    $("#btn02").click(function () {
        $("#pId").text("段落标签");
    });
</script>
```

- html()和html(...)

```
<body>
<p id="pId">段落</p>
```

```
<input type="submit" value="获得标签体html" id="btn01"/>
<input type="submit" value="设置标签体html" id="btn02"/>
</body>
<script>
    $("#btn01").click(function () {
        // 获得p标签的标签体
        console.log($("#pId").html());
    });

    $("#btn02").click(function () {
        // 设置p标签的标签体
        $("#pId").html("段落标签");
    });
</script>
```

- text()和html()区别

```
<body>
<p id="pId"><font>段落</font></p>
<input type="submit" value="获得p标签的文本(text)" id="btn01"/>
<input type="submit" value="获得p标签的标签体(html)" id="btn02"/>
<br/>
<input type="submit" value="设置p标签的文本(text)" id="btn03"/>
<input type="submit" value="设置p标签的标签体(html)" id="btn04"/>
</body>
<script>

    $("#btn03").click(function () {
        // 设置p标签的文本
        $("#pId").text("<font color='red'>段落标签</font>")// 整个参数就是一个文
本
    });

    $("#btn04").click(function () {
        // 设置p标签的标签体
        $("#pId").html("<font color='red'>段落标签</font>");// 段落标签文本会变
红
    });


    $("#btn01").click(function () {
        // 获得p标签的文本
        console.log($("#pId").text());// 段落
    });

    $("#btn02").click(function () {
        // 设置p标签的标签体
        console.log($("#pId").html());// <font>段落</font>
    });
</script>
```

jQuery创建,插入

- API

API方法	解释
$("A")	创建A元素对象
append(element)	添加成最后一个子元素，两者之间是父子关系
prepend(element)	添加成第一个子元素，两者之间是父子关系
appendTo(element)	添加到父元素的内部最后面
prependTo(element)	添加到父元素的内部最前面
before(element)	添加到当前元素的前面，两者之间是兄弟关系
after(element)	添加到当前元素的后面，两者之间是兄弟关系

```
内部插入:父节点/子字点之间操作
a.append(c);    把c添加到a的内部的最后面
a.prepend(c);   把c添加到a的内部的最前面
a.appendTo(c);  把a添加到c的内部的最后面
a.prependTo(c); 把a添加到c的内部的最前面

外部插入(了解): 兄弟节点之间操作
a.after(c);     把c添加到a的后面
a.before(c);    把c添加到a的前面
```

- append()和prepend()--->父子标签

```
<body>
<p id="pId">
    <font id="fId">八戒</font>
</p>
<input type="button" value="append()" id="btn01">
<input type="button" value="prepend()" id="btn02">
</body>
<script>
    // 需求1: 点击按钮1,添加沙悟净到p标签的内部,在八戒的后面
    $("#btn01").click(function () {
        // 1.获得p标签的jquery对象
        // 2.调用append方法
        $("#pId").append($("<font>沙悟净</font>"));

    });

    // 需求2: 点击按钮2,添加孙悟空到p标签的内部,在八戒的前面
    $("#btn02").click(function () {
        // 1.获得p标签的jquery对象
        // 2.调用prepend方法
        $("#pId").prepend($("<font>孙悟空</font>"));

    });

</script>
```

- appendTo()和prependTo()--->父子标签

```
<body>
<p id="pId">
    <font id="fId">八戒</font>
</p>
<input type="button" value="appendTo()" id="btn01">
<input type="button" value="prependTo()" id="btn02">
</body>
<script>
    // 需求1: 点击按钮1,添加沙悟净到p标签的内部,在八戒的后面
    $("#btn01").click(function () {
        // $("#pId").append($("<font>沙悟净</font>"));// 把沙悟净添加到p标签内部的最后面
        $("<font>沙悟净</font>").appendTo($("#pId"));// 把沙悟净添加到p标签内部的最后面
    });

    // 需求2: 点击按钮2,添加孙悟空到p标签的内部,在八戒的前面
    $("#btn02").click(function () {
        // $("#pId").prepend($("<font>孙悟空</font>"));
        $("<font>孙悟空</font>").prependTo($("#pId"));// 把孙悟空添加到p标签内部的最前面
    });

</script>
```

- after()和before() --->兄弟标签

```
<body>
<p id="pId">

    <font id="fId">八戒</font>

</p>

<input type="button" value="after()" id="btn01">
<input type="button" value="before()" id="btn02">
</body>
<script>
    // 需求1: 点击按钮1,添加沙悟净到八戒的后面
    $("#btn01").click(function () {
        // 1.获得八戒标签的jquery对象
        // 2.调用after方法,添加沙悟净
        $("#fId").after($("<font>沙悟净</font>"));
    });

    // 需求2: 点击按钮2,添加孙悟空到八戒的前面
    $("#btn02").click(function () {
        // 1.获得八戒标签的jquery对象
        // 2.调用before方法,添加孙悟空
        $("#fId").before($("<font>孙悟空</font>"));
    });

</script>
```

jQuery移除节点(对象)

- API

API方法	解释
remove()	删除指定元素(自己移除自己)
empty()	清空指定元素的所有子元素

```
remove()   自己移除自己
empty()    清空当前标签里面的内容
```

```
<body>
<p id="pId">
    <font id="fId">八戒</font>
</p>

<input type="button" value="remove()" id="btn01">
<input type="button" value="empty()" id="btn02">
</body>
<script>
    // 需求1:点击按钮1,移除p标签以及子标签
    $("#btn01").click(function () {
        // 获得p标签的jquery对象,调用remove方法
        $("#pId").remove();
    });

    // 需求2:点击按钮1,移除p标签内部的子标签,p标签还在
    $("#btn02").click(function () {
        // 获得p标签的jquery对象,调用empty方法
        $("#pId").empty();
        // $("#pId").html("");
    });

</script>
```

1.10 JQ遍历

复习JS方式遍历

- 语法

```
for(var i=0;i<元素数组.length;i++){
    元素数组[i];
}
```

- eg

```
//定义了一个数组
var array = [1,2,3,4,"aa"];
//a. 使用原来的方式遍历
for(var i = 0; i < array.length;i++){
  console.log("array["+i+"]="+array[i]);
}
```

jquery对象方法遍历【重点】

- 语法

```
jquery对象.each(function(index,element){});

其中:(参数名字随便取的)
index:就是元素在集合中的索引
element：就是集合中的每一个元素对象
```

- eg

```
// jquery对象方法遍历:  jq对象.each(function(index,element){})
// index: 数组元素的索引, element:遍历出来的元素,js对象,  这两个变量可以自定义
$(arr).each(function (index, element) {
    console.log("索引为:"+index+",元素为:"+element);
});
```

jquery的全局方法遍历

- 语法

```
$.each(jquery对象,function(index,element){});

其中,
index:就是元素在集合中的索引
element：就是集合中的每一个元素对象
```

- eg

```
// jquery的全局方法遍历: $.each(jq对象,function(index,element){})
    $.each($(arr),function (index, element) {
        console.log("索引为:"+index+",元素为:"+element);
    })
```

jQuery3.0新特性for of语句遍历

- 语法

```
for(变量 of jquery对象){
    变量;
}

其中,
变量:定义变量依次接受jquery数组中的每一个元素
jquery对象：要被遍历的jquery对象
```

- eg

```
 // jQuery3.0新特性for  of语句遍历:  for(变量 of jq对象){}
// jq对象:要遍历的对象    变量:用来存储遍历出来的元素
for (e of $(arr)){
    console.log(e);
}
```

第二章-JQ案例

案例:使用JQuery完成页面定时弹出广告

一,需求分析

- 进入页面3s后弹出广告,3s后广告隐藏

二,思路分析

1. 在index.html的顶部定义一个广告区域, 设置隐藏
2. 创建定时任务,3s后显示广告
3. 创建showAd()
4. 创建hideAd()

三,代码实现

```
<div id="divId" style="width: 100%; height: 200px; display: none">
    <img src="../img/ad.jpg" width="100%" height="100%">
</div>
```

```
<script>
    // 1.设置一次性定时器: 3s后显示广告
    setTimeout("showAd()", 3000);

    // 2.创建showAd函数:
```

```
    function showAd() {
        // 2.1 显示图片,显示完图片后,隐藏图片
        $("#divId").show(5000,function () {
            // 隐藏图片
            // hideAd();
            // 设置一次性定时器: 5s后隐藏广告
            setTimeout("hideAd()",5000);
        });
    }

    // 3.创建隐藏图片函数:
    function hideAd() {
        $("#divId").hide(3000);
    }
</script>
```

四,小结

1. 创建定时任务, 3s展示广告

```
setTimeout("showAd()",3000);
```

2. 创建showAd()

```
function showAd(){
    //a.展示广告
    //b.setTimeout("hideAd()",3000);
}
```

3. 创建hideAd()

```
function hideAd(){
    //a.隐藏广告

}
```

案例:使用JQuery完成表格的隔行换色

一,需求分析

	商品名称	商品价格	商品数量	操作
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改

二,思路分析

1. 使用筛选选择器, 匹配出奇数(odd)行, 设置背景色
2. 使用筛选选择器, 匹配出偶数(even)行, 设置背景色

三,代码实现

```
<script>
    // 偶数行,背景色设置为蓝色
    $("tr:even").css("background","blue");

    // 奇数行,背景色设置为粉色
    $("tr:odd").css("background","pink");

</script>
```

四,小结

1. 筛选选择器
2. 操作样式 css(css属性,css属性值)

案例:使用JQuery完成复选框的全选效果

一,需求分析

	商品名称	商品价格	商品数量	操作
☑	Mac	18000	10	删除 修改
☑	Mac	18000	10	删除 修改
☑	Mac	18000	10	删除 修改
☑	Mac	18000	10	删除 修改

二.思路分析

1. 准备页面
2. 给最上面的复选框设置点击事件,创建函数响应这个事件
3. 在匿名函数里面

```
//a.获得最上面复选框的选中状态
//b.获得下面5个复选框, 设置checked选中状态和最上面复选框的状态一致
```

三,代码实现

- js代码

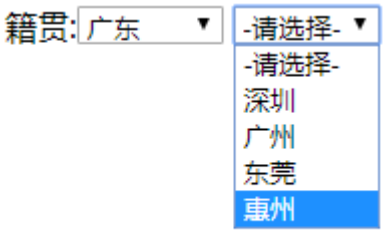
```
<script>
    function selectAll(obj) {
        // 1.获得复选框的checked值
        console.log(obj.checked);
        // 2.获得下面的所有复选框,复选框checked值设置为与全选复选框的值一样的
        $(".itemSelect").attr("checked",obj.checked);
    }
</script>
```

四,小结

1. 触发点: 最上面的复选框的点击事件
2. 把下面的复选框状态设置成和最上面的一样

案例:使用JQuery完成省市联动效果

一,需求分析



二,思路分析

1. 创建页面, 初始化数据
2. 给省份的select设置内容改变事件

```
pSelect.change(function(){
    // 1.获得省份下拉框value属性的值
    // 2.根据value属性的值获得对应的城市(数组)
    // 3.获得城市下拉框对象
    // 4.清空城市下拉框中的选项
    // 5.循环遍历所有的城市
    // 6.在循环中把遍历出来的城市添加到城市下拉框
});
```

三,代码实现

- js代码

```
<script>
    var citys = [
        ["深圳", "广州", "东莞", "惠州"],
        ["武汉", "黄冈", "黄石", "鄂州", "荆州"],
        ["济南", "青岛", "烟台", "淄博", "聊城"]
    ];
```

```
    // 为省份下拉框绑定内容改变事件
    $("#pId").change(function () {
        // 1.获得省份下拉框value属性的值
        var value = this.value;  // $(this).val()


        // 2.获得城市下拉框对象
        var jqCity = $("#cId");

        // 3.清空城市下拉框中的选项
        // jqCity.empty();
        jqCity.html("<option>-请选择-</option>");


        // 判断
        if (value > -1){
            // 4.根据value属性的值获得对应的城市(数组)
            var cityArr = citys[value];

            // 5.循环遍历所有的城市
            $(cityArr).each(function (index, element) {
                // 6.在循环中把遍历出来的城市添加到城市下拉框
                jqCity.append("<option>"+element+"</option>");

            });
        }

    });

</script>
```

四.小结

1. 遍历 jq对象.each(function(i,ele){})
2. 操作dom a.append(c)
3. 思路
    - 给省份的select设置内容改变事件
    - 获得省份的值
    - 根据省份的值 获得对应的城市的数据
    - 遍历城市的数据, 拼接成option
    - 添加到右边的select里面

扩展案例_表格换色

1.需求

	商品名称	商品价格	商品数量	操作
	Mac	18000	10	删除修改
	Mac	18000	10	删除修改
	Mac	18000	10	删除修改
	Mac	18000	10	删除修改
	Mac	18000	10	删除修改

- 鼠标进入tr的时候, 当前tr的背景色改成red
- 鼠标离开tr的时候, 当前tr的背景色还原

2.技术

- 选择器
- 事件
- css()

3.思路

```
<script>
    // 偶数行,背景色设置为蓝色
    $("tr:even").css("background","blue");

    // 奇数行,背景色设置为粉色
    $("tr:odd").css("background","pink");

    // - 鼠标进入tr的时候, 当前tr的背景色改成red
    $("tr").mouseover(function () {
        $(this).css("background","red");
    });
    // - 鼠标离开tr的时候, 当前tr的背景色还原
    $("tr").mouseout(function () {
        $("tr:even").css("background","blue");
        $("tr:odd").css("background","pink");
    });


</script>
```

扩展案例_电子时钟

1.需求

2019/12/17 16:06:10

2,思路

```
<span>                                                    <span>

//1.创建定时任务
setInterval("getTime()",1000);
//2.创建getTime()
function getTime(){
  //获得时间
  //向span插入
}
```

3.实现

```
<body>
<span id="spanId"></span>

</body>
<script>
    // 1.设置一个循环定时器,每隔1s调用getTime()
    setInterval("getTime()", 1000);

    // 2.实现getTime():
    function getTime() {
        // 2.1 获得当前时间对象,转换为字符串
        var date = new Date().toLocaleString();
        console.log(date);

        // 2.2 获得span标签对象,把时间字符串写入span标签中
        $("#spanId").html(date);
    }
</script>
```

总结

```
必须练习:
1.jq和js对象转换---->1.3
2.jq事件的使用----->1.4
3.jq基本选择器,层级选择器,过滤选择器(奇数,偶数,第几个),表单属性选择器---->1.6
4.jq案例---->6个案例

- 能够引入jQuery
    <script src="js/jquery-3.3.1.min.js"></script>

- 能够掌握JQ和JS对象的转换
    JS-->JQ:  $(js对象)
    JQ-->JS:  jq对象[0] 或者 jq对象.get(0)

- 能够掌握JQ中事件的使用
```

绑定事件:
方式一:
```
jq对象.事件名(function(){

});
```
方式二:
```
jq对象.on(事件名,function(){

});
```
解绑事件:
```
jq对象.off(事件名);
```

- 能够掌握JQ中常用的选择器
  基本选择器
  层级选择器
  属性选择器
  过滤选择器
  表单属性选择器

- 能够使用JQ操作样式
  css(name) 获取CSS样式
  css(name,value) 设置CSS样式

- 能够使用JQ操作属性
  attr(name,[value]) 获得/设置属性的值
  prop(name,[value]) 获得/设置属性的值(checked，selected)

- 能够使用JQ操作DOM
  val([value]) 获得/设置标签里面value属性相应的值
  text([value]) 获得/设置元素的文本内容
  html([value]) 获得/设置元素的标签体内容

  $("html标签代码") 创建元素对象
  append(element) 添加成最后一个子元素，两者之间是父子关系
  prepend(element) 添加成第一个子元素，两者之间是父子关系
  appendTo(element) 添加到父元素的内部最后面
  prependTo(element) 添加到父元素的内部最前面
  before(element) 添加到当前元素的前面，两者之间是兄弟关系
  after(element) 添加到当前元素的后面，两者之间是兄弟关系

- 能够掌握JQ遍历
  方式一:
  ```
  jq对象.each(function(index,element){

  });
  ```
  方式二:
  ```
  $.each(jq对象,function(index,element){

  });
  ```
  方式三:
  ```
  for(变量 of jq对象){
      变量;
  }
  ```