# day10-JavaScript高级

* 今日内容
* DOM-案例1\_电灯开关
* BOM-(Browser Object Model)

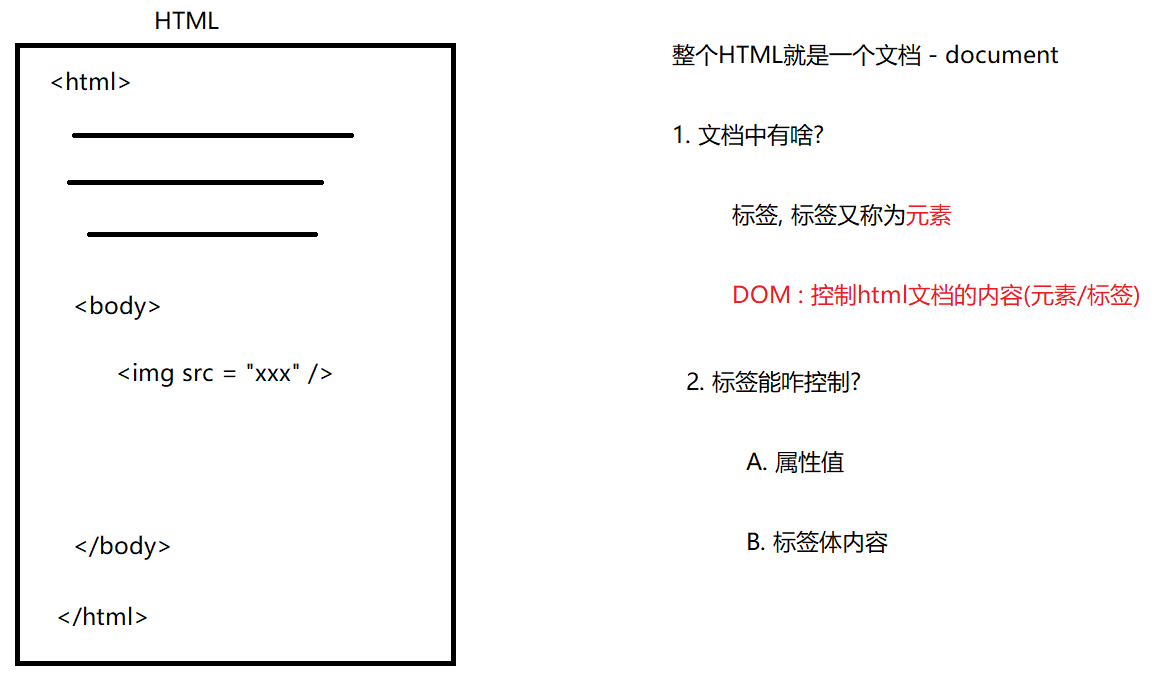
1. BOM概述
2. Window对象
3. Location对象
4. History对象

* BOM-案例3\_自动跳转首页
* DOM-案例4\_动态表格\_添加-删除
* HTMLDOM
* 事件
* DOM-案例5\_表格全选
* DOM-案例6\_表单验证

## 1. DOM-案例1-电灯开关

### 1.1 DOM简单学习

* 概述 : 文档对象模型（Document Object Model，简称DOM）
* 功能 ：控制html文档的内容



* 操作步骤 :

1. 获取页面标签(元素)对象：Element

通过getElementById();



2. 操作Element对象

通过API查找对应的Element对象有哪些属性可以修改.

A. 修改属性值：

|  |
| --- |
|  |

B. 修改标签体内容

|  |
| --- |
|  |

### 1.2 事件简单学习

* 功能： 某些组件被执行了某些操作后，触发某些代码的执行。
* 如何绑定事件  
    1. 直接在html标签上，指定事件的属性(操作)，属性值就是js代码  
         事件：onclick--- 单击事件  
    
    2. 通过js获取元素对象，指定事件属性，设置一个函数

|  |
| --- |
|  |

### 1.3 案例1\_电灯开关

* 分析 :

1.获取图片对象

2.绑定单击事件

3.每次点击切换图片

\* 规则：

\* 如果灯是开的 on,切换图片为 off

\* 如果灯是关的 off,切换图片为 on

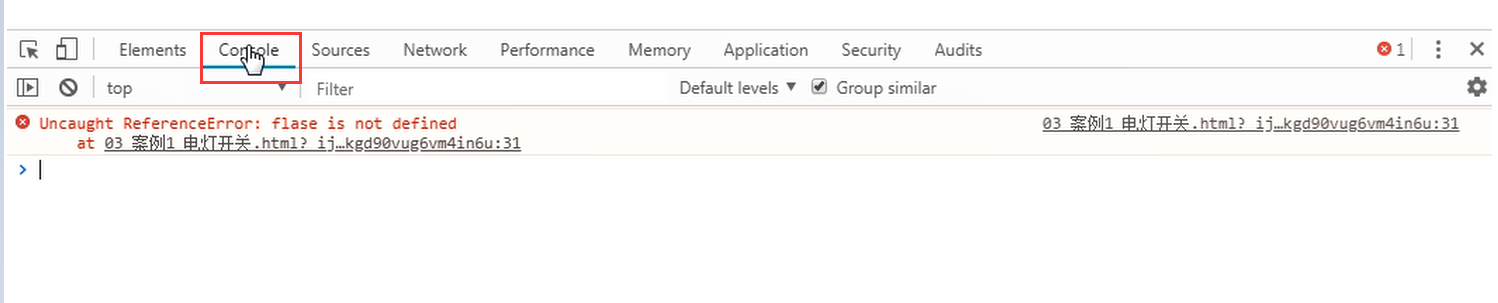
\* 使用标记flag来完成

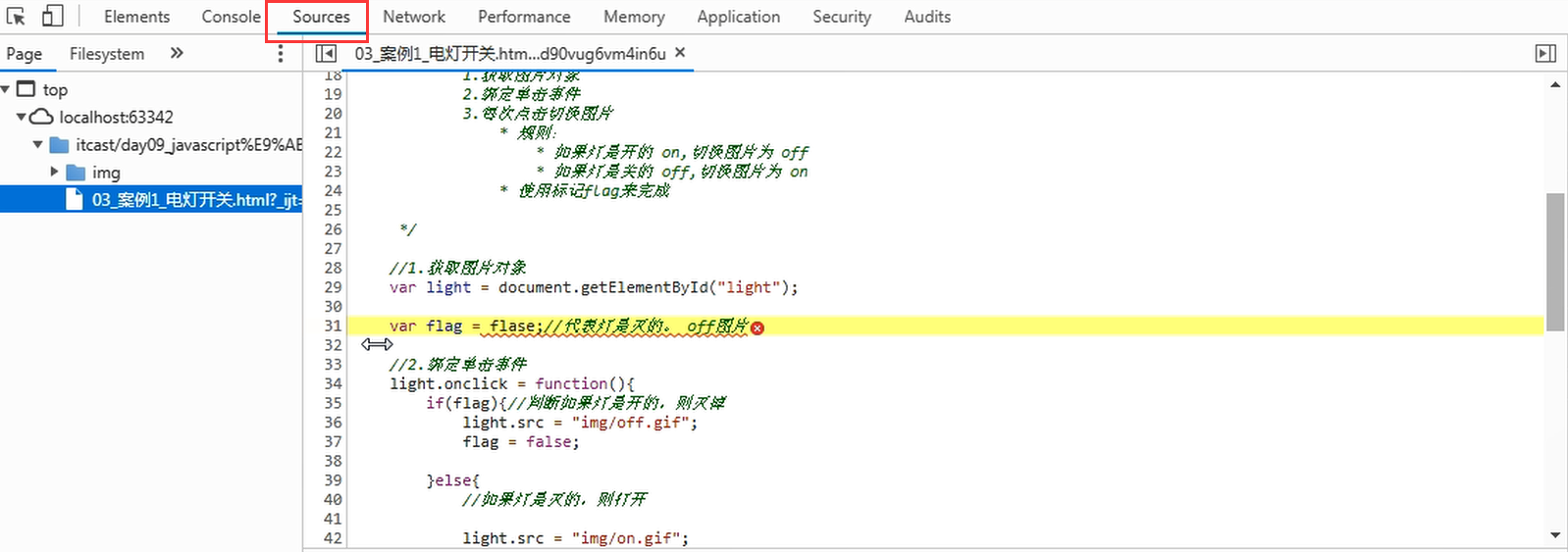
* 代码实现 :

|  |
| --- |
|  |

* 错误查找 :

浏览器F12 \ fn + F12 -> 打开开发者调试器





## 2. BOM-(Browser Object Model)

### 2.1 BOM\_概述

* 概念：Browser Object Model 浏览器对象模型

\* 将浏览器的各个组成部分封装成对象。

* 组成：

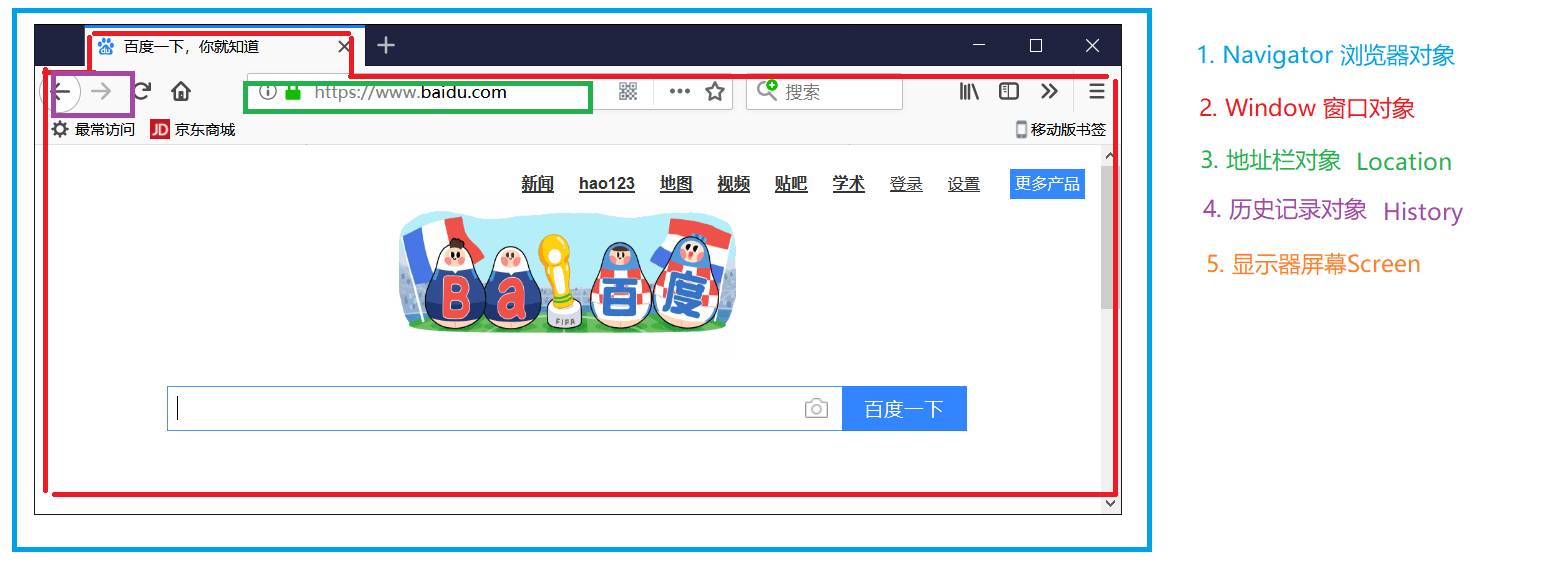
\* Window：窗口对象

\* Navigator：浏览器对象

\* Screen：显示器屏幕对象

\* History：历史记录对象

\* Location：地址栏对象



### 2.2 BOM\_Window\_弹出方法

* 特点

\* Window对象不需要创建可以直接使用 window使用。 window.方法名();

\* window引用可以省略。 方法名();

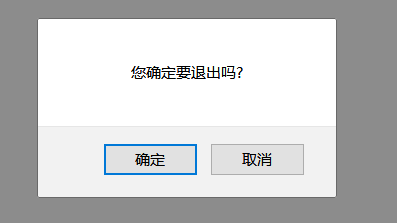
* 与弹出框有关的方法

alert() 显示带有一段消息和一个确认按钮的警告框。

confirm() 显示带有一段消息以及确认按钮和取消按钮的对话框。

\* 如果用户点击确定按钮，则方法返回true

\* 如果用户点击取消按钮，则方法返回false



prompt() 显示可提示用户输入的对话框。

\* 返回值：获取用户输入的值

### 2.3 BOM\_Window\_打开关闭方法

* close() 关闭浏览器窗口。

\* 谁调用我 ，我关谁

* open() 打开一个新的浏览器窗口

\* 返回新的Window对象

|  |
| --- |
|  |

### 2.4 BOM\_Window\_定时器方法

* setTimeout() 在指定的毫秒数后调用函数或计算表达式。

\* 参数：

1. js代码或者方法对象

2. 毫秒值

\* 返回值：唯一标识，用于取消定时器

* clearTimeout() 取消由 setTimeout() 方法设置的 timeout。
* setInterval() 按照指定的周期（以毫秒计）来调用函数或计算表达式。
* clearInterval() 取消由 setInterval() 设置的 timeout。

|  |
| --- |
|  |

### 2.5 案例2\_轮播图

* 分析：

1.在页面上使用img标签展示图片

2.定义一个方法，修改图片对象的src属性

3.定义一个定时器，每隔3秒调用方法一次。

|  |
| --- |
|  |

### 2.6 BOM\_Window\_属性

1. 获取其他BOM对象：

history

location

Navigator

Screen:

|  |
| --- |
|  |

2. 获取DOM对象

document

### 2.7 BOM\_Location

1. 创建(获取)：

1. window.location

2. location

2. 方法：

\* reload() 重新加载当前文档。刷新



3. 属性

\* href 设置或返回完整的 URL。



### 2.8 案例3\_自动跳转首页

* 需求 : 网页中读秒倒计时, 5秒之后, 跳转到首页.
* 分析 :

1.显示页面效果 <p>

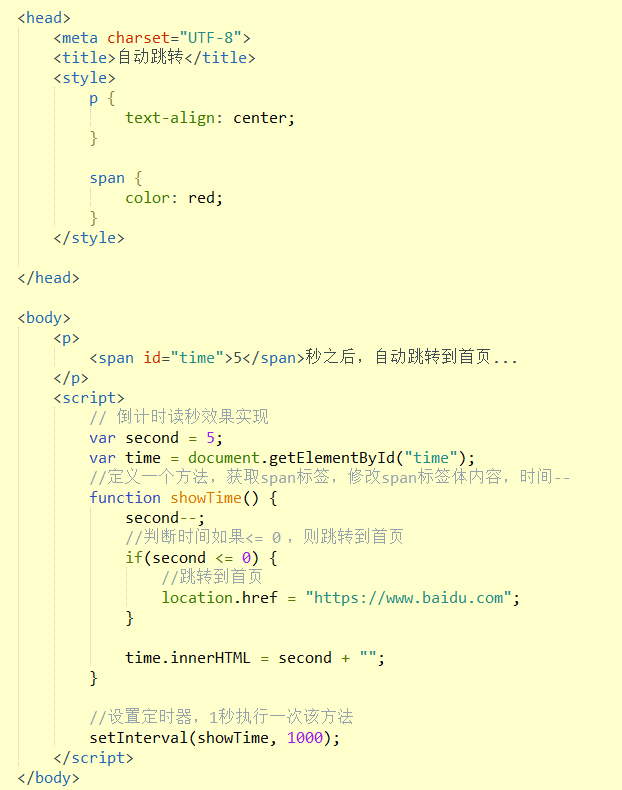
2.倒计时读秒效果实现

2.1 定义一个方法，获取span标签，修改span标签体内容，时间--

2.2 定义一个定时器，1秒执行一次该方法

3.在方法中判断时间如果<= 0 ，则跳转到首页

* 代码实现：



### 2.9 BOM\_History

* 创建(获取)：

1. window.history

2. history

* 方法：

\* back() 加载 history 列表中的前一个 URL。

\* forward() 加载 history 列表中的下一个 URL。

\* go(参数) 加载 history 列表中的某个具体页面。

\* 参数：

\* 正数：前进几个历史记录

\* 负数：后退几个历史记录

* 属性：

\* length 返回当前窗口历史列表中的 URL 数量。

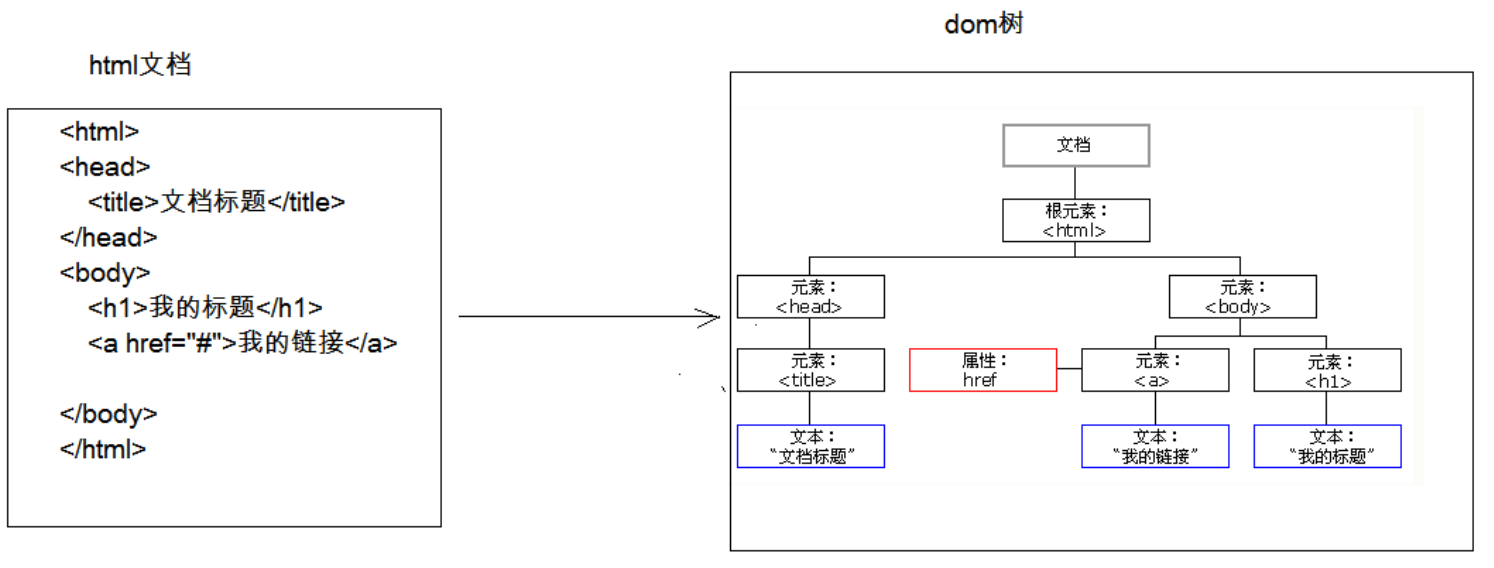
## 3. DOM(Document Object Model)

### 3.1 DOM\_概述

* 概念： Document Object Model 文档对象模型

将标记语言文档的各个组成部分，封装为对象。

可以使用这些对象，对标记语言文档进行CRUD(增删改查)的动态操作



* W3C DOM 标准被分为 3 个不同的部分
  + 核心 DOM - 针对任何结构化文档的标准模型

\* Document：文档对象

\* Element：元素对象

\* Attribute：属性对象

\* Text：文本对象

\* Comment:注释对象

\* Node：节点对象，其他5个的父对象

* XML DOM - 针对 XML 文档的标准模型
* HTML DOM - 针对 HTML 文档的标准模型

### 3.2 DOM\_Document对象\_获取Element方法

* Document对象创建

创建(获取)：在html dom模型中可以使用window对象来获取

小提示: xml中的document需要通过(流)进行加载.

A. window.document

B. document

* 获取Element对象的方法

1. getElementById() ： 根据id属性值获取元素对象。id属性值一般唯一

2. getElementsByTagName()：根据元素名称获取元素对象们。返回值是一个数组

3. getElementsByClassName():根据Class属性值获取元素对象们。返回值是一个数组

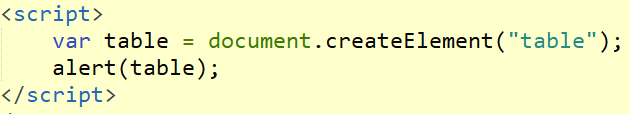
4. getElementsByName(): 根据name属性值获取元素对象们。返回值是一个数组

* 示例代码 :



### 3.3 DOM\_Document对象\_创建DOM对象

* createAttribute(name) 创建一个属性节点
* createComment() 创建注释节点。
* createElement() 创建元素节点。
* createTextNode() 创建文本节点。



### 3.4 DOM\_Element对象

1. 获取/创建：通过document来获取和创建

2. 方法：

1. removeAttribute()：删除属性

2. setAttribute()：设置属性



### 3.5 DOM\_Node对象

* 特点(理解)：所有dom对象都可以被认为是一个节点
* 方法：

\* CRUD dom树：

\* appendChild()：向节点的子节点列表的结尾添加新的子节点。

\* removeChild()：删除（并返回）当前节点的指定子节点。

\* replaceChild()：用新节点替换一个子节点。

* 练习材料:

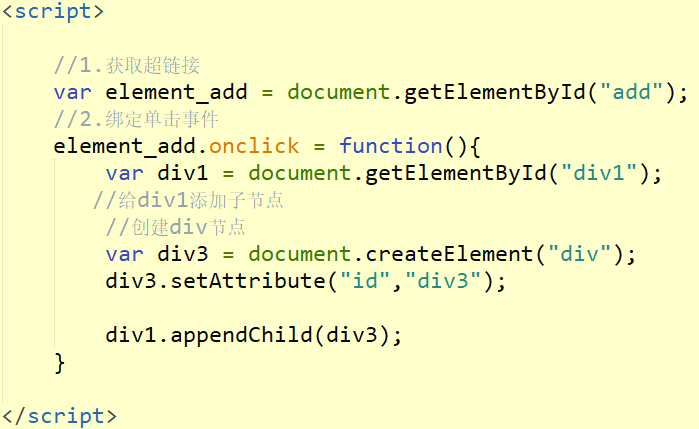
1. 样式

|  |
| --- |
| <style>  div {  border: 1px solid red;  }  #div1 {  width: 200px;  height: 200px;  }  #div2 {  width: 100px;  height: 100px;  }  #div3 {  width: 100px;  height: 100px;  }  </style> |

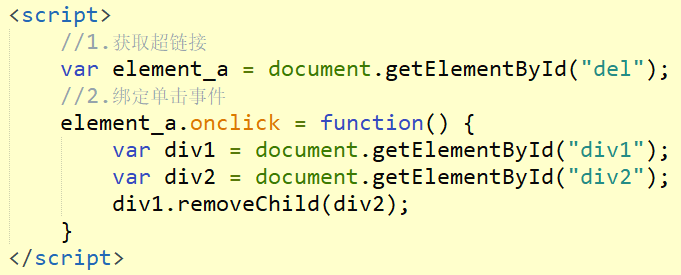
2. div

|  |
| --- |
| <div id="div1">  div1  <div id="div2">div2</div>  </div>  <a href="javascript:void(0);" id="add">添加子节点</a> |

* appendChild



* removeChild



* replaceChild()



* 注意 :

href="javascript:void(0);"

作用 : 拦截a标签跳转的逻辑, 执行自己添加的监听事件.

在做页面时，如果想做一个链接点击后不做任何事情，或者响应点击而完成其他事情.

可以设置其属性 href = "#"，但是这样会有一个问题.

就是当页面有滚动条时，点击后会返回到页面顶端，用户体验不好。

### 3.6 案例4\_动态表格\_添加

* 分析 :

1. 给添加按钮绑定单击事件

2. 获取文本框的内容

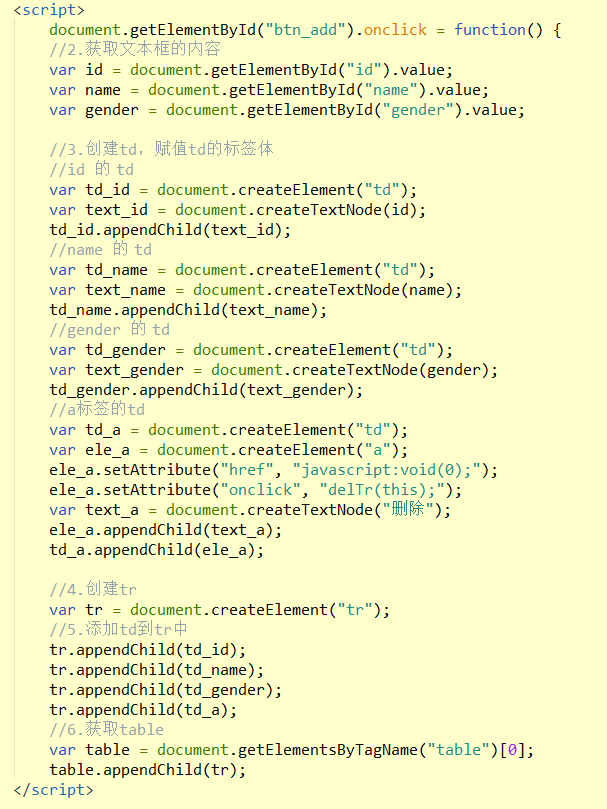
3. 创建td，设置td的文本为文本框的内容。

4. 创建tr

5. 将td添加到tr中

6. 获取table，将tr添加到table中

* 示例代码 :



### 3.7 案例4\_动态表格\_删除

* 分析 :

1.确定点击的是哪一个超链接

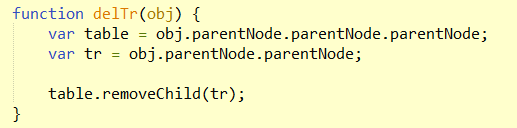
<a href="javascript:void(0);" onclick="delTr(this);" >删除</a>

this : 代表当前点击的超链接控件.

2.怎么删除？

removeChild():通过父节点删除子节点

* 示例代码 :

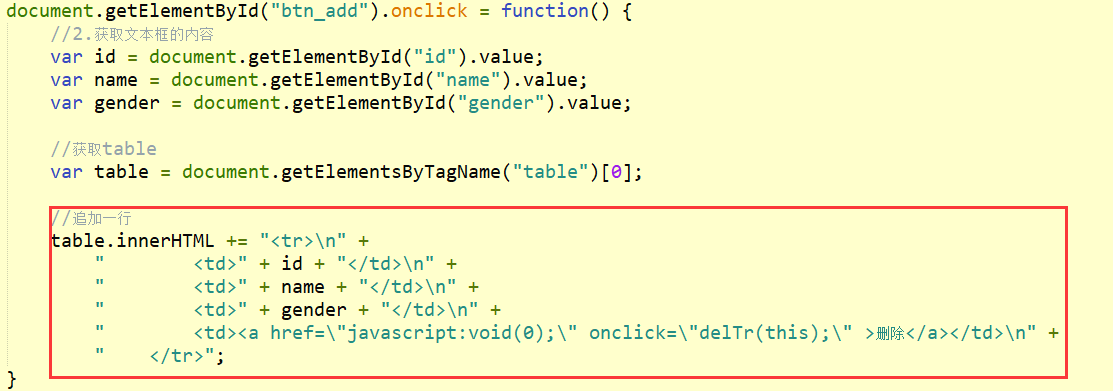


### 3.8 HTMLDOM\_innerHTML

* HTML DOM的基本操作

标签体的设置和获取：innerHTML

* 示例代码 : (替换表格案例的繁琐编写)

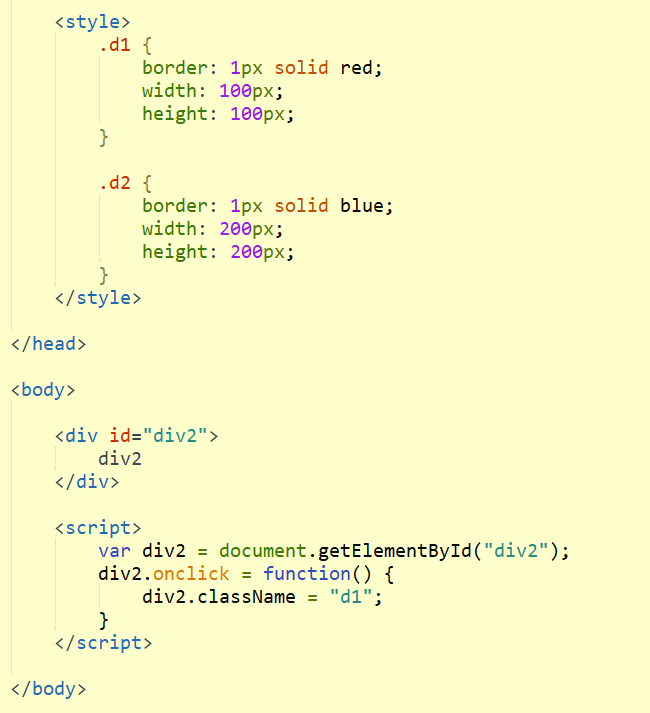


### 3.9 HTMLDOM\_样式控制

* 使用元素的style属性来设置



* 提前定义好类选择器的样式，通过元素的className属性来设置其class属性值。

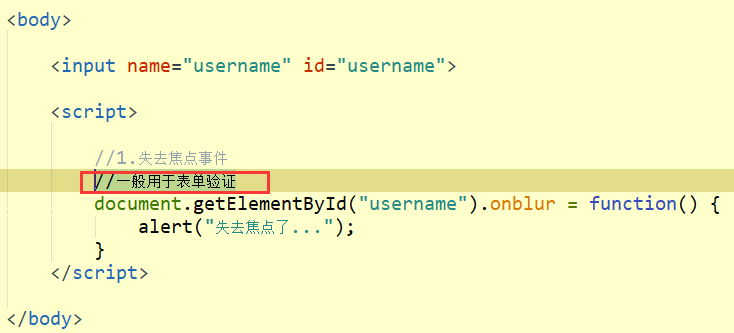


### 3.10 事件\_概述

* 概念：某些组件被执行了某些操作后，触发某些代码的执行。
* 事件：某些操作。如： 单击，双击，键盘按下了，鼠标移动了
* 事件源：组件。如： 按钮 文本输入框...
* 监听器：代码。
* 注册监听：将事件，事件源，监听器结合在一起。 当事件源上发生了某个事件，则触发执行某个监听器代码。
* 常见的事件：  
          1. 点击事件：  
              1. onclick：单击事件  
              2. ondblclick：双击事件  
          2. 焦点事件  
              1. onblur：失去焦点  
              2. onfocus:元素获得焦点。  
    
          3. 加载事件：  
              1. onload：一张页面或一幅图像完成加载。  
    
          4. 鼠标事件：  
              1. onmousedown    鼠标按钮被按下。  
              2. onmouseup    鼠标按键被松开。  
              3. onmousemove    鼠标被移动。  
              4. onmouseover    鼠标移到某元素之上。  
              5. onmouseout    鼠标从某元素移开。  
                
                
          5. 键盘事件：  
              1. onkeydown    某个键盘按键被按下。      
              2. onkeyup        某个键盘按键被松开。  
              3. onkeypress    某个键盘按键被按下并松开。  
    
          6. 选择和改变  
              1. onchange    域的内容被改变。  
              2. onselect    文本被选中。  
    
          7. 表单事件：  
              1. onsubmit    确认按钮被点击。  
              2. onreset    重置按钮被点击。

### 3.11 常见事件演示

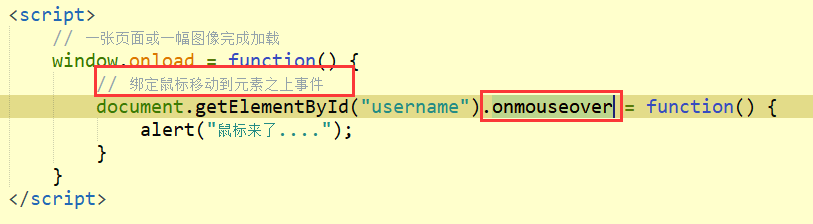
* onblur 失去焦点



* onload 加载事件



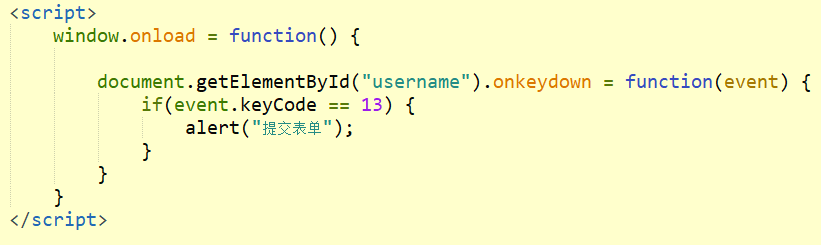
* onmousemove 鼠标被移动



* onmousedown 鼠标按钮被按下



* onkeydown 某个键盘按键被按下



* onchange    域的内容被改变

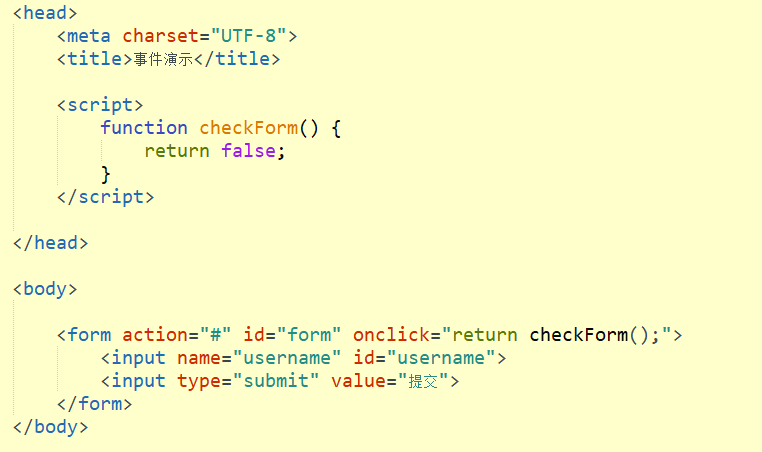


* onsubmit    确认按钮被点击

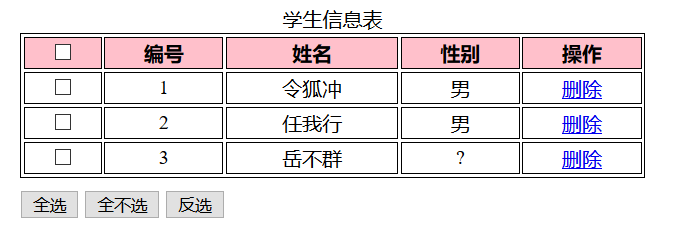
写法1



写法2



### 3.12 案例5\_表格全选



* 需求 :

1. 3个按钮的点击事件 – 全选\全不选\反选

2. 每一行获得焦点后, 颜色更变

3. 勾选栏事件

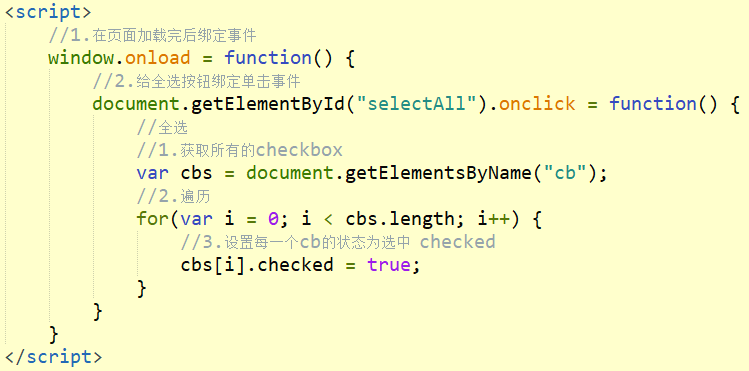
ps : 基本表格搭建, 到资料中复制

* 全选 :

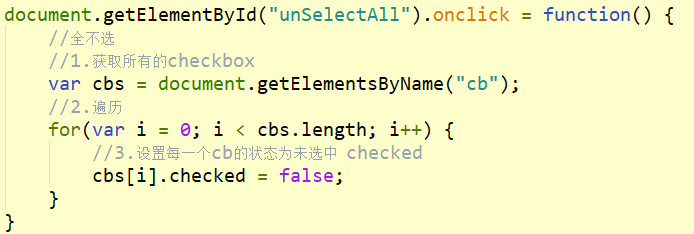
分析 :

\* 获取所有的checkbox

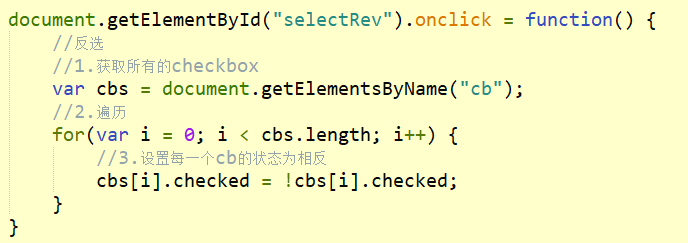
\* 遍历cb，设置每一个cb的状态为选中 checked



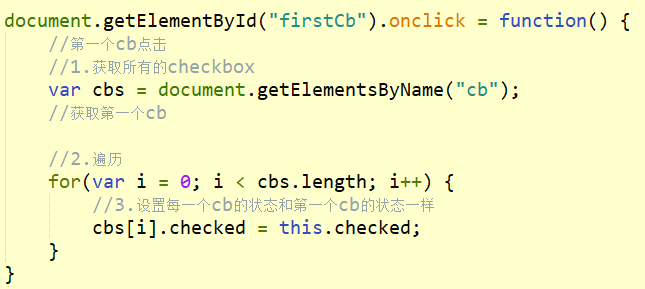
* 全不选 :



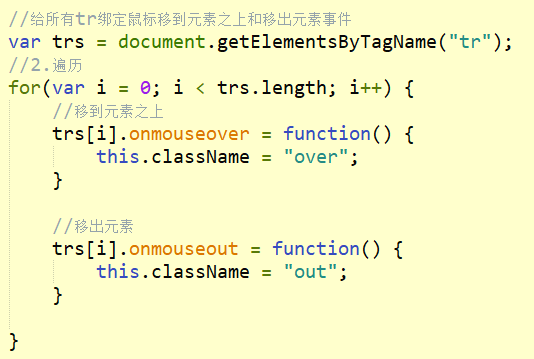
* 反选 :



* 表格checkbox



* tr事件绑定



### 3.13 案例6\_表单验证



* 分析 :

1.给表单绑定onsubmit事件。监听器中判断每一个方法校验的结果。

如果都为true，则监听器方法返回true

如果有一个为false，则监听器方法返回false

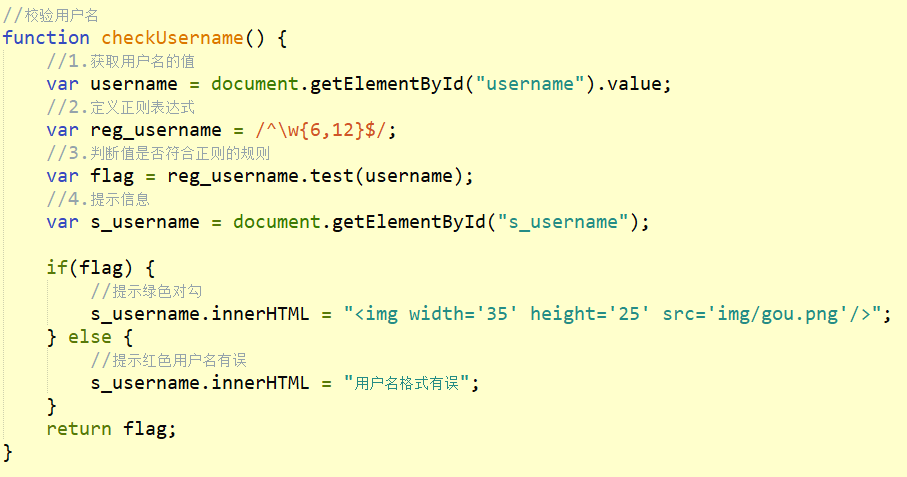
2.定义一些方法分别校验各个表单项。

3.给各个表单项绑定onblur事件。

* 注册按钮



* 用户名表单项校验



* 密码表单项校验

