一、昨日反馈

1. 数据类型转换
   1. 转换成字符串
      1. 使用 变量.toString(); //不能转换undefined和null
      2. 使用String(变量); //能转换所有的数据类型
      3. 其他类型的变量和字符串相加（连）操作的时候，会把其他类型的变量通过String自动转换成字符串，然后在完成连接操作
   2. 转换成数值型
      1. parseInt(变量); //检查字符0处的字符是否是数字，如果是继续向后检查，直到发现一个非数字为止，然后将前面的数字部分转换成整型；如果发现0处的字符不是数字，则转换成NaN。
      2. parseFloat(变量); //转换成浮点型，规则和parseInt一样，只不过parseFloat允许有一个小数点。
      3. 强制转换Number(变量); // false🡪0; true🡪1; undefined🡪NaN; null🡪0;
   3. 转换成浮点型
      1. 下面的值转换成浮点型之后为false：""、0、 0.0、 undefined、 null、NaN。
      2. 除了上面列举的值，其他值转换成布尔都是true，包括"0", [], {};
2. 流程控制
   1. 顺序结构
   2. 分支结构
   3. 循环结构：可以给循环起一个名字，比如（ a:for(…)…）for循环中，前面的a就表示它的名字，用于break等指定循环使用
3. 函数基础
   1. 定义函数
      1. 常规的方式：function 函数名(){} //可以先调用，后声明
      2. 表达式方式：var 函数名 = function(){}; //只能先定义，后使用
      3. 立即调用模式： ( function(){} ) (); //红色的小括号必须的，表示里面声明的函数是一个整体；绿色的小括号表示调用函数。
4. 作用域
   1. 分类：全局作用域和局部（函数）作用域
   2. 作用域规则
      1. 函数可以使用函数之外定义的变量
      2. 函数内部优先使用内部的变量，如果内部没有，才去函数外部查找。（注意变量提升）
      3. 函数内部没有用var声明的变量，也是全局变量。
5. 变量提升和函数预加载
   1. 变量提升
      1. 如果在声明变量之前，就使用了一个变量。这种情况不会报错，它实际执行的过程是把声明变量的过程提升到使用之前，注意的是只是把声明变量的过程提升到使用之前，并没有把变量的赋值过程提升。
   2. 函数预加载
      1. 函数声明必须是常规方式（ function 函数名(){} ）
      2. 函数的调用和声明必须在同一个script标签中

二、今日目标

1、理解什么是DOM，它的作用是什么

2、能够通过多种方式查询页面中的元素

3、能够对页面中的元素进行增删改克隆操作

4、能够设置或获取元素的css样式

5、能够通过dom节点为元素绑定事件

6、能够获取兼容各个浏览器的事件对象

7、能够通过事件对象获取当前的keyCode

三、DOM介绍

3.1、什么是DOM

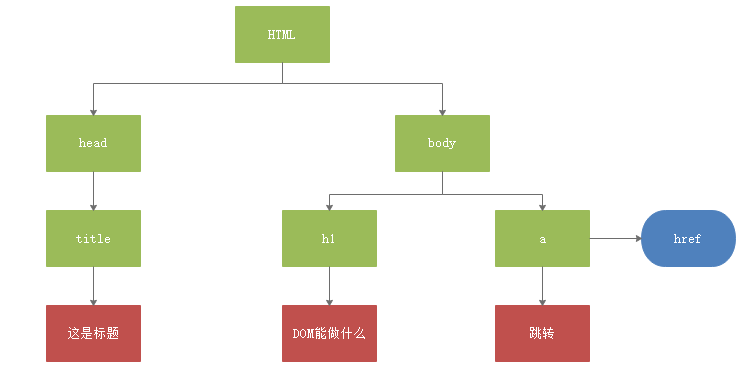
DOM全称是Document Object Model（文档对象模型），它是JS将HTML按文档结构和内容层次抽象出的模型，使得JavaScript有了访问HTML的能力，能够实现对HTML中内容的操作。DOM存在广泛，PHP以及其他语言也有各自的DOM模型。

3.2、节点与DOM模型

DOM模型呈现树状结构，因此也叫“树模型”，树中的内容（标签、属性、文本）称为“节点”，节点在dom中就是对象。包含有元素节点（标签）、属性节点、文本节点。dom模型中，每个节点，都是一个JavaScript对象。

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

* **整个文档是一个文档节点，用document对象表示，最大的节点**
* 每个 HTML 元素是元素节点，比如html、head、body、a、h1
* HTML 元素内的文本是文本节点
* 每个 HTML 属性是属性节点，比如href
* 注释是注释节点



3.3、节点之间的关系

节点之间的关系犹如人类家族中的族谱。节点之间的关系包括：

* 父子关系（父子、后代、祖先级）
* 兄弟（姐妹）关系

单词：

* parent -- 父母
* child -- 孩子
* children -- 孩子（复数），表示所有孩子
* siblings -- 兄弟姐妹、同胞

四、元素(标签)节点查询操作

4.1、根据document对象查询

**document是JS内置的一个对象，表示整个HTML文档**。根据document查询，意思就是在整个文档范围内查询。

单词：

* get -- 获取、得到
* Element -- 元素
* Elements -- 很多元素
* By… -- 根据…./通过….
* Class -- 类
* Tag -- 标签
* query -- 查询
* Selector -- 选择器

练习：

* getElementById() -- 根据元素的id值获取一个元素类型的节点 返回值是一个dom对象
* getElementsByTagName() -- 通过标签名获取很多个元素类型的节点 返回一个包含很多dom对象的数组

具体查询方法见下表：

|  |  |  |
| --- | --- | --- |
| **方法名** | **返回值** | **支持** |
| getElementById(元素id) | 一个dom对象 | 所有浏览器 |
| getElementsByTagName(标签名) | 包含dom对象的数组 | 所有浏览器 |
| getElementsByClassName(类名) | 包含dom对象的数组 | IE8+ |
| getElementsByName(元素的name属性) | 包含dom对象的数组 | 所有浏览器 |
| querySelector(css选择器) | 一个dom对象 | IE7+ |
| querySelectorAll(css选择器) | 包含dom对象的数组 | IE7+ |

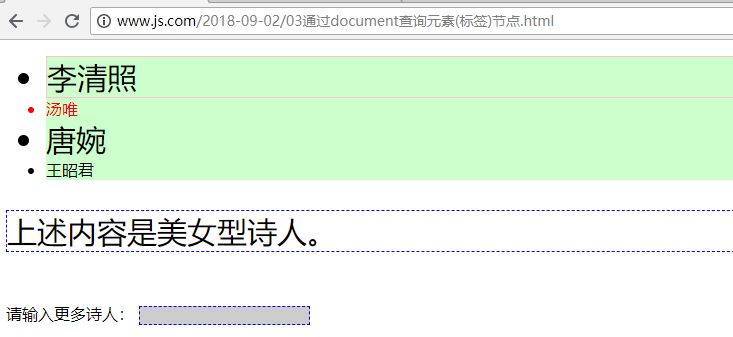
演示的HTML：

<**ul**>  
 <**li class="a"**>李清照</**li**>  
 <**li id="tangwei"**>汤唯</**li**>  
 <**li class="a"**>唐婉</**li**>  
 <**li**>王昭君</**li**>  
</**ul**>  
<**p class="a"**>上述内容是美女型诗人。</**p**>  
<**br**>  
请输入更多诗人： <**input type="text" name="shiren"** />

查询的JS代码：

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* script标签一定要放到ul/p/input标签下面 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//------------------------根据id查询***var *tangwei*** = **document**.getElementById(**'tangwei'**);  
*//console.log(tangwei);  
//tangwei.style.样式名称 = 值; 给标签添加样式****tangwei***.**style**.**color** = **'red'**;  
  
*//-----------------------根据标签名查询***var *lis*** = **document**.getElementsByTagName(**'li'**); *// 返回数组，数组中包含满足条件的4个li  
//console.log(lis);***for**(**var *i***=0; ***i***<***lis***.**length**; ***i***++){  
 ***lis***[***i***].**style**.**backgroundColor** = **'#ccffcc'**;  
}  
  
  
*//---------------------根据元素的类名查询***var *as*** = **document**.getElementsByClassName(**'a'**); *// 返回数组，数组中包含类名为a的元素***for**(**var *i***=0; ***i***<***as***.**length**; ***i***++){  
 ***as***[***i***].**style**.**fontSize** = **'30px'**;  
}  
  
*//---------------------根据元素的name属性查询***var *input*** = **document**.getElementsByName(**'shiren'**); *// 返回数组，数组中包含属性name为shiren的元素****input***[0].**style**.**backgroundColor** = **'#ccc'**;  
  
*//--------------------根据css选择器选择元素  
//document.querySelector('css选择器');***var *liqingzhao*** = **document**.querySelector(**'.a'**); *// 返回一个dom对象，class为a的有很多，也只返回第一个****liqingzhao***.**style**.**border** = **'solid 1px pink'**;  
  
**var *all*** = **document**.querySelectorAll(**'p, input'**); *//返回数组***for**(**var *i*** in ***all***){  
 ***all***[***i***].**style**.**border** = **'dashed 1px blue'**;  
}

最终效果：



4.2、相互关系查询

相互关系查询可以分为下面三种情况：

* 根据父节点查找子节点
* 根据子节点查父节点
* 查询兄弟节点

单词：

* child -- 孩子
* children -- 很多孩子
* Node -- 节点
* nodes -- 很多节点
* first -- 第一个
* last -- 最后一个
* Element -- 元素
* Sibling -- 兄弟，姐妹
* Siblings -- 所有的兄弟，姐妹
* previous -- 上一个
* next -- 下一个

具体属性/方法见下表：

| **方法/属性** | **说明** | **返回** | **支持** |
| --- | --- | --- | --- |
| **父节点.children** | 查询所有子节点，非w3c标准 | 数组 | 所有浏览器 |
| **父节点.childNodes** | 查询所有子节点，包括文本节点 | 数组 | 所有浏览器 |
| 父节点.firstChild | 查询第一个子节点，包括文本节点 | 对象 | 所有浏览器 |
| 父节点.firstElementChild | 查询第一个子节点，不包括文本节点 | 对象 | IE8+ |
| 父节点.lastChild | 查询最后一个子节点，包括文本节点 | 对象 | 所有浏览器 |
| 父节点.lastElementChild | 查询最后一个子节点，不包括文本节点 | 对象 | IE8+ |
| **父节点.getElementBy....()** | 节点继续调用getEle...系列方法查询子节点 | 数组/对象 | 所有浏览器 |
| 子节点.parentNode | 查询父节点 | 对象 | 所有浏览器 |
| 兄弟.previousSibling | 查询上一个兄弟，包括文本节点 | 对象 | 所有浏览器 |
| 兄弟.previousElementSibling | 查询上一个兄弟，不包括文本节点 | 对象 | IE8+ |
| 兄弟.nextSibling | 查询下一个兄弟，包括文本节点 | 对象 | 所有浏览器 |
| 兄弟.nextElementSibling | 查询下一个兄弟，不包括文本节点 | 对象 | IE8+ |

演示的HTML代码：

<**ul id="u"**>  
 <**li**>泽拉斯</**li**>  
 <**li**>蚂蚱</**li**>  
 <**li**>拉克丝</**li**>  
 <**li**>安妮</**li**>  
 <**li**>维克托</**li**>  
</**ul**>

测试的JS代码：

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 根据父节点查找子节点 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//先找到父节点  
//var ul = document.getElementsByTagName('ul')[0];  
//var ul = document.getElementById('u');***var *ul*** = **document**.querySelector(**'#u'**);  
  
*// ------------- 查询所有ul的子节点 -- children  
//var lis = ul.children; // 返回数组。 查询ul的所有子节点  
//console.log(lis);  
  
// ------------- 查询所有ul的子节点 -- childNodes  
//var lis = ul.childNodes; //  
//console.log(lis);  
  
//---------------查询所有ul的子节点 ---  
//var lis = ul.getElementsByTagName('li'); //在ul中查找所有ul后代中的li  
//console.log(lis);  
  
//---------------查询ul的第一个子节点 --- firstChild  
//var li1 = ul.firstChild;  
//console.log(li1); //不好用，会把第一个空白当做第一个子元素  
  
//---------------查询ul的第一个子节点 --- firstElementChild  
//var li1 = ul.firstElementChild;  
//console.log(li1); //好用，但是IE8+才支持  
  
//---------------查询ul的最后一个子节点 --- lastChild  
//---------------查询ul的最后一个子节点 --- lastElementChild  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 根据子点查找父节点 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//随便找一个子节点  
//var mazha = document.getElementsByTagName('li')[1];  
//var ul2 = mazha.parentNode;  
//console.log(ul2);  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 兄弟关系查询 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//随便找一个子节点***var *mazha*** = **document**.getElementsByTagName(**'li'**)[1];  
*//查询上一个兄弟  
//var zelasi = mazha.previousSibling;  
//console.log(zelasi); //不好使，会认为空白是上一个兄弟  
  
//var zelasi = mazha.previousElementSibling;  
//console.log(zelasi); //好使，但是IE8+支持  
  
//查询下一个兄弟 -- nextSibling  
//查询下一个兄弟 -- nextElementSibling*

效果：打开注释，依次查看即可。

4.3、遗留DOM

早期DOM访问形式，在一些特定元素的获取上比较方便。被保留下来使用，W3C标准化之后 称为“0级DOM”。

|  |  |  |  |
| --- | --- | --- | --- |
| 方法/属性 | 说明 | 返回 | 支持 |
| document.body | 访问body节点 | 返回body节点 | 所有浏览器 |
| document.forms | 访问所有的表单 | 返回数组 | 所有浏览器 |
| document.formName.name | 访问表单项 | 返回单个表单项或数组 | 所有浏览器 |
| document.anchors | 访问所有的锚点 | 返回数组 | 所有浏览器 |
| document.links | 访问所有的链接 | 返回数组 | 所有浏览器 |
| document.images | 访问所有的图片 | 返回数组 | 所有浏览器 |
| document.all | 访问所有的元素 | 返回数组 | 所有浏览器 |

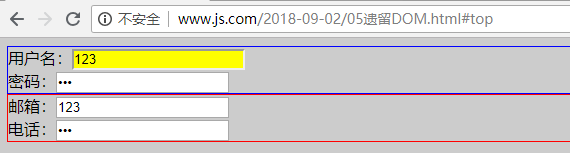
HTML代码：

<**form name="f1"**>  
 用户名：<**input type="text" name="username" value="123"**><**br**>  
 密码：<**input type="password" name="pwd" value="123"**>  
</**form**>  
  
<**form name="f2"**>  
 邮箱：<**input type="text" name="email" value="123"**><**br**>  
 电话：<**input type="password" name="tel" value="123"**>  
</**form**>

JS代码：

*//获取body节点***var *body*** = **document**.**body**;  
***body***.**style**.**backgroundColor** = **'#ccc'**;  
  
*//获取form***document**.**forms**[1].**style**.**border** = **'solid 1px red'**;  
  
*//获取form，通过name值***document**.**f1**.**style**.**border** = **'solid 1px blue'**;  
  
*//找表单中各个项***document**.**f1**.**username**.**style**.**backgroundColor** = **'yellow'**;

效果：



五、元素（标签）节点增删改操作

5.1、创建、增加节点

方法一：父节点.appendChild(子节点);

<**ul**>  
 <**li**>林青霞</**li**>  
 <**li**>张曼玉</**li**>  
 <**li**>刘欢</**li**>  
 <**li**>陈冠希</**li**>  
</**ul**>  
  
  
<**script**>  
 *//创建一个 <li>谢霆锋</li>  
 //创建元素类型的节点* **var *li*** = **document**.createElement(**'li'**);  
 *//创建文本类型的节点* **var *text*** = **document**.createTextNode(**'谢霆锋'**);  
 *//添加节点，将文本节点添加到元素节点中* ***li***.appendChild(***text***); *// 得到<li>谢霆锋</li>  
 //添加节点，将组合好的li，添加到ul中* **document**.querySelector(**'ul'**).appendChild(***li***);  
</**script**>

方法二：父节点.insertBefore(新节点, 参照的节点);

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* insertBefore \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//创建一个 <li>渣渣辉</li>***var *newLi*** = **document**.createElement(**'li'**);  
**var *newText*** = **document**.createTextNode(**'渣渣辉'**);  
***newLi***.appendChild(***newText***); *//得到 <li>渣渣辉</li>  
//找到参照的张曼玉所在的li***var *zhangmanyu*** = **document**.getElementsByTagName(**'li'**)[1];  
*//找到li的父节点***var *ul*** = **document**.querySelector(**'ul'**);  
*//父节点.insertBefore(新节点, 参照的节点);****ul***.insertBefore(***newLi***, ***zhangmanyu***);

5.2、克隆节点

**语法：原来的节点.cloneNode([true]);**

<**ul class="a" style="color**:**red**;**"**>  
 <**li**>鲁智深</**li**>  
 <**li**>张飞</**li**>  
 <**li**>猪八戒</**li**>  
 <**li**>李逵</**li**>  
</**ul**>  
  
<**script**>  
 *//克隆一份ul，然后放到页面中  
 //先找到要克隆的ul* **var *oldUl*** = **document**.querySelector(**'ul'**);  
 *//克隆一个新的ul  
 //var newUl = oldUl.cloneNode(); //只克隆ul标签本身，不包含里面的内容* **var *newUl*** = ***oldUl***.cloneNode(**true**); *//克隆ul标签并包含里面的内容  
 //把新的ul放到body中* **document**.**body**.appendChild(***newUl***);  
</**script**>

5.3、替换节点

方法：父节点.replaceChild(新节点, 待替换的节点);

<**ul**>  
 <**li**>林黛玉</**li**>  
 <**li**>贾宝玉</**li**>  
 <**li**>薛宝钗</**li**>  
 <**li**>刘姥姥</**li**>  
</**ul**>  
  
<**script**>  
 *// 父节点.replaceChild(新节点, 待替换的节点);  
  
 //找父节点* **var *ul*** = **document**.getElementsByTagName(**'ul'**)[0];  
  
 *//新节点* **var *newLi*** = **document**.createElement(**'li'**);  
 **var *text*** = **document**.createTextNode(**'王熙凤'**);  
 ***newLi***.appendChild(***text***);  
  
 *//找到待替换的节点* **var *jiabaoyu*** = **document**.getElementsByTagName(**'li'**)[1];  
  
 *//执行替换* ***ul***.replaceChild(***newLi***, ***jiabaoyu***);  
</**script**>

5.4、删除节点

方法：父节点.removeChild(子节点);

HTML代码：

<**ul**>  
 <**li**>林黛玉</**li**>  
 <**li**>贾宝玉</**li**>  
 <**li**>薛宝钗</**li**>  
 <**li**>刘姥姥</**li**>  
</**ul**>

JS代码：

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 删除节点 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//父节点.removeChild(子节点);***var *ul*** = **document**.querySelector(**'ul'**);  
**var *xiaoLiu*** = **document**.getElementsByTagName(**'li'**)[3];  
***ul***.removeChild(***xiaoLiu***);

六、设置/获取元素的css样式

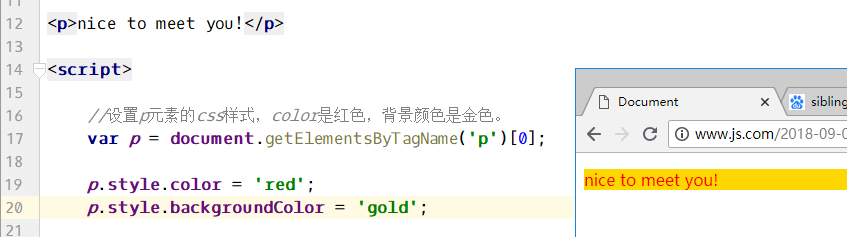
6.1、设置css样式

语法： elementNode**.**style**.**css样式 = 值

css样式的写法：

①、一个单词的直接写即可。比如color height …

②、样式名称带中横线的，去掉中横线，后面单词首字母大写。比如fontSize lineHeight backgroundColor



6.2、获取css样式

使用 "node.style.样式名称" 的方式只能获取行内样式和js已经设置过的样式。

要想获取全部的样式，则必须使用下面的方法：

* 在IE中支持node.currentStyle.样式名称
* 火狐支持getComputedStyle(node).样式

为了兼容各个浏览器，所以需要自己封装一个函数，用于获取完整的css样式：

<**p**>nice to meet you!</**p**>  
  
<**script**>  
  
 *//设置p元素的css样式，color是红色，背景颜色是金色。* **var *p*** = **document**.getElementsByTagName(**'p'**)[0];  
  
 ***p***.**style**.**color** = **'red'**;  
 ***p***.**style**.**backgroundColor** = **'gold'**;  
  
 */\*\*\*\*\*\*\*\*\*\*\*\* 获取元素的css样式 \*\*\*\*\*\*\*\*\*\*\*\*\*/  
 /\*var c = p.style.color;  
 console.log(c);  
 var f = p.style.fontSize;  
 console.log(f);\*/  
 //获取完整样式的写法（非IE）  
 //var f = getComputedStyle(p).fontSize;  
 //console.log(f);  
  
 //获取完整样式的写法（IE8）  
 //var f = p.currentStyle.fontSize;  
 //console.log(f);  
  
 //兼容各个浏览器的获取完整css样式的写法* **function** *getStyle*(node, styleName){  
 **if**(node.**currentStyle**){  
 *//说明是IE* **return** node.**currentStyle**[styleName];  
 }**else**{  
 **return** getComputedStyle(node)[styleName];  
 }  
 }  
  
 **console**.log(*getStyle*(***p***, **'fontSize'**));  
  
</**script**>

七、事件和事件对象

7.1、什么是事件

浏览网页时，当我们做出点击鼠标、按键盘、移动鼠标等行为时，这些行为会被浏览器内置的JavaScript引擎所捕获，并执行对应的某些操作（函数）。那么你的行为（动作）+ JavaScript引擎捕获 + 执行对应的操作 = 事件。

所以，一个完整的事件应该包括：

* 用户行为；
* 浏览器捕获你的行为；
* 执行对应的操作（函数）

常见行为有：鼠标点击、鼠标的移动、鼠标的移入和移出、键盘控制等等。

事件的作用是：通过事件，我们（浏览网页的人）就可以和浏览器进行一些交互了。

7.2、事件绑定方式

语法：

node.事件名 = function(){

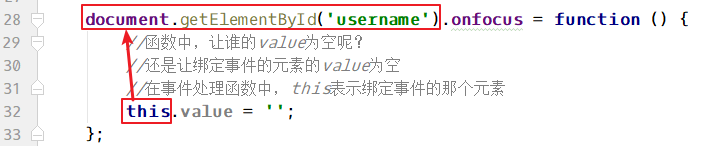
//事件被触发时，执行这个函数。

};  
//例如  
document.getElementById('btn').onclick = function(){  
 console.log('你点击我了');  
}

<**input type="button" value="点我试试" id="btn"** />  
<**input type="text" id="username" value="请输入用户名"** />  
  
<**script**>  
 *//找到要绑定事件的元素节点* **var *btn*** = **document**.getElementById(**'btn'**);  
 *//单击事件* ***btn***.onclick = **function**(){  
 alert(**'你真敢点'**);  
 };  
  
 *//找到username，绑定获取焦点事件* **document**.getElementById(**'username'**).**onfocus** = **function** () {  
 **document**.getElementById(**'username'**).**value** = **''**;  
 };  
</**script**>

在事件处理函数中，this表示绑定事件的那个元素：

所以上面获得焦点的事件可以优化代码为：



**练习**：

页面中有很多个td，点击td的时候，让td的背景颜色发生变化：

HTML代码：

<**table**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**></**td**>  
 <**td**></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**></**td**>  
 <**td**></**td**>  
 <**td**></**td**>  
 </**tr**>  
</**table**>

css代码：

<**style**>  
 **table**,**td**{  
 **border**:**solid** 1**px #ccc**;  
 **border-collapse**: **collapse**; */\*合并边框\*/* }  
 **td**{  
 **width**:150**px**;  
 **height**:50**px**;  
 */\*color:rgb(30,50,10);\*/* }  
 **table**{  
 **margin**:10**px auto**; */\*让表格左右居中对齐\*/* }  
</**style**>

JS代码：

<**script**>  
 *//点击页面中的td，点击之后，让被点击的td背景颜色改变成红色  
 //先找到所有的td* **var *tds*** = **document**.getElementsByTagName(**'td'**); *// 返回数组  
 //循环，为每个td都绑定一个单击事件* **for**(**var *i***=0; ***i***<***tds***.**length**; ***i***++){  
 ***tds***[***i***].onclick = **function** () {  
 *//alert(123);  
 //this 表示绑定事件的那个td* **var** r = **Math**.floor(**Math**.random()\*256);  
 **var** g = **Math**.floor(**Math**.random()\*256);  
 **var** b = **Math**.floor(**Math**.random()\*256);  
 **this**.**style**.**backgroundColor** = **'rgb('**+r+**', '**+g+**', '**+b+**')'**;  
 */\*if(this.style.backgroundColor != 'red'){  
 this.style.backgroundColor = 'red';  
 }else{  
 this.style.backgroundColor = 'white';  
 }\*/* };  
 }  
</**script**>

效果：



7.3、常用事件列举

* 页面事件：
  + onload ：当页面载入完毕（页面中的标签和外部资源）后触发
* 焦点事件
  + onfocus ：当获取焦点时触发
  + onblur ：当失去焦点时触发
* 鼠标事件
  + onmouseover ：当鼠标悬浮时触发
  + onmouseout ：当鼠标离开时触发
* 键盘事件
  + onkeypress ：当键盘按下时触发（如果按住某个键不松开，会一直触发press事件）
  + onkeydown ：当键盘按下时触发
  + onkeyup ：当键盘弹起时触发
* 其他事件：
  + onchange ：内容改变时会触发，常用于select>option。
  + onsubmit ：表单提交时触发，**这个事件要给form绑定**而不是给提交按钮绑定
  + onresize ： 页面窗口改变大小时会触发
  + onscroll ：滚动条滚动时触发

7.3、什么是事件对象

事件对象也是一个对象，它提供了一些属性，这些属性描述了当前事件的特点；

不同的事件中，事件对象也有所差异，比如单击事件中，事件对象会提供pageX和pageY属性，表示点击的点距离页面的距离，比如键盘事件中，事件对象会提供keyCode属性，表示按的是什么键。

总之，事件对象中提供了一些属性，这些属性可以很好的描述当前的事件的特点。

7.4、获取事件对象

IE浏览器：window.event;

火狐浏览器：传递给事件处理函数的形参

**下面代码演示获取事件对象的方式**：

<**input type="text"**>  
  
<**script**>  
 **document**.getElementsByTagName(**'input'**)[0].onclick = **function**(a){  
 *//console.log(a); // IE8+ 支持，获取事件对象的方法  
 //console.log(window.event); // IE8浏览器  
 //兼容各个浏览器的获取事件对象的方法  
 /\*var e;  
 if(window.event){  
 e = window.event; //IE8  
 }else{  
 e = a; // IE8+  
 }\*/* **var** e = **window**.**event**||a;  
 };  
</**script**>

7.5、事件对象常用属性

下面列举一些事件对象中的常用属性：

* keyCode：表示键盘上的键对应的数值。
* altKey：表示是否按了alt键，按了结果为true，没按结果为false（组合按键的时候，才会有作用）
* shiftKey：表示是否按了shift键，按了结果为true，没按结果为false（组合按键的时候，才会有作用）
* ctrlKey：表示是否按了ctrl键，按了结果为true，没按结果为false（组合按键的时候，才会有作用）
* pageX: 鼠标距离页面左边的距离
* pageY: 鼠标距离页面上面的距离
* screenX: 鼠标距离屏幕左边的距离
* screenY: 鼠标距离屏幕上面的距离

下面的代码是获取keyCode的代码：

<**script**>  
 *//在页面中任何位置，按键，当键盘弹起的时候，执行函数* **document**.onkeyup = **function**(evt){  
 *//先获取事件对象* **var** e = **window**.**event**||evt; *//兼容各个浏览器的获取事件对象的方式  
 //获取键盘对应的数字* **var** keyCode = e.**keyCode**;  
 *//alert(keyCode);* **if**(keyCode == 13){  
 alert(**'您按了回车键'**);  
 }  
 }  
</**script**>