点击浏览器前进后退按钮禁止跳转

页面2不能前进后退

页面1：

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
</**head**>  
<**body**>  
页面1  
<**a href="跳转2.html"**>kjiuhe</**a**>  
<**script language="JavaScript"**>  
 **javascript**:**window**.**history**.forward(1);  
</**script**>  
</**body**>  
</**html**>

页面2：

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
</**head**>  
<**body** >  
<**a href="跳转3.html"**>页面2</**a**>  
</**body**>  
</**html**>

页面3：

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
</**head**>  
<**body**>  
页面3  
<**script language="JavaScript"**>  
 **javascript**:**window**.**history**.back()//也可以改成GO（-1）  
</**script**>  
</**body**>  
</**html**>

# 9实现一个表格，点击一下成绩，顺序会从升序变成降序，或者从降序变成升序

<http://www.cnblogs.com/xiao-t/archive/2012/12/27/2836248.html>

var x=document.getElementsByTagName("input");

<input name="myInput" type="text" size="20" /><br />

<input name="myInput" type="text" size="20" /><br />

<input name="myInput" type="text" size="20" /><br />

onload 事件会在页面或图像加载完成后立即发生。**window**.onload

document.getElementById("myList").document.getElementById("myList").appendChild(newListItem);

appendChild(newListItem);中的ITEM如果是已有的元素则会移动他！

//实现代码

<!DOCTYPE **HTML**>  
<**html**>  
<**head**>  
 <**meta charset="utf-8"**>  
 <**title**>table排序</**title**>  
</**head**>  
  
<**body**>  
<**table id="tableTest" width="400" border="1"**>  
  
 <**tbody**>  
 <**tr**>  
 <**td**>2</**td**>  
 <**td**>bbb</**td**>  
 <**td**>**&nbsp;**</**td**>  
 <**td**>**&nbsp;**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>5</**td**>  
 <**td**>eee</**td**>  
 <**td**>**&nbsp;**</**td**>  
 <**td**>**&nbsp;**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>3</**td**>  
 <**td**>ccc</**td**>  
 <**td**>**&nbsp;**</**td**>  
 <**td**>**&nbsp;**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>4</**td**>  
 <**td**>ddd</**td**>  
 <**td**>**&nbsp;**</**td**>  
 <**td**>**&nbsp;**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>1</**td**>  
 <**td**>aaa</**td**>  
 <**td**>**&nbsp;**</**td**>  
 <**td**>**&nbsp;**</**td**>  
 </**tr**>  
 </**tbody**>  
</**table**>  
<**input type="button" id="sort" value="表格排序"** />  
<**script**>  
 **window**.onload = **function** () {  
 **var** oTable = **document**.getElementById(**'tableTest'**);  
 **var** oTbody = oTable.**tBodies**[0];*//第一个TBODY元素* **var** oBtn = **document**.getElementById(**'sort'**);  
 **var** arr = [];*//用来存放每一个tr 每一行* **var** isAsc = **true**;*//用来判断升序，还是降序* oBtn.**onclick** = **function** () {  
 **for** (**var** i = 0; i < oTbody.**rows**.**length**; i++ ) {  
 arr[i] = oTbody.**rows**[i];*//这里是把每一个tr存放到一个数组，而不是排序的依据（这里是cells[0].innerHTML） 获得每一个行元素* }  
  
 *//数组根据cells[0].innerHTML来排序* arr.sort(**function** (td1, td2){  
 **if**(isAsc) {  
 **return** parseInt(td1.**cells**[0].**innerHTML**) - parseInt(td2.**cells**[0].**innerHTML**);  
 } **else** {  
 **return** parseInt(td2.**cells**[0].**innerHTML**) - parseInt(td1.**cells**[0].**innerHTML**);  
 }  
  
 });  
 *//把排序后的tr 重新插入tbody* **for**(**var** j =0; j < arr.**length**; j++ ) {  
 oTbody.appendChild(arr[j]);  
 }  
 *//判断升序，降序* isAsc = !isAsc;  
 }  
 }  
</**script**>  
</**body**>  
</**html**>

实现思路

1用arr存储表格对应的行

用一个布尔变量存储是升序还是降序

2 arr[i] = oTbody.**rows**[i]（指针）

3 在sort函数里面传入表格的行的排名

4 然后一个一个利用arr插入表格的行

javascript中有关table元素的属性、方法：

1. caption：保存着对<caption>元素（如果有）的指针
2. tBodies：是一个<tbody>元素的HTMLCollection
3. tFoot：保存着对<tfoot>元素（如果有）的指针
4. tHead：保存着对<thead>元素（如果有）的指针
5. rows：是一个表格中所有行的HTMLCollection
6. createTHead()：创建<thead>元素，将其放到表格中，并返回引用
7. createTFoot()：创建<tfoot>元素，将其放到表格中，并返回引用
8. createCaption()：创建<caption>元素，将其放到表格中，并返回引用
9. deleteTHead()：删除<thead>元素
10. deleteTFoot()：删除<tfoot>元素
11. deleteCaption()：删除<caption>元素
12. deleteRow(pos)：删除指定位置行
13. insertRow(pos)：向rows集合中指定位置插入一个行

有关tbody的属性、方法：

1. rows：保存着tbody所有行的HTMLCollection
2. deleteRow(pos)：删除指定位置的行
3. insertRow(pos)：向rows的集合中的指定位置插入一行，并返回对新插入行的引用

有关tr的属性、方法：

1. cells：保存着tr所有td（单元格）的HTMLCollection
2. deleteCell(pos)：删除指定位置的单元格
3. insetCell(pos)：向cells的集合中的指定位置插入一个单元格，并返回对该单元格的引用

       使用JS实现一组图片动画效果或者使用HTML5 Canvas渲染一系列图片等案例中，需要图片全部加载完成方可运行动画效果。此时程序中就会涉及多张图片预加载代码。当接二连三的案例中都涉及图片预加载时，就需要考虑把这个功能封装为一个通用方法。

（1）下面为JS实现图片预加载方法loadImages()：

**Js代码  收藏代码**

1. //实现一系列图片的预加载
2. //参数sources:图片信息关联数组
3. //参数callback:回调函数——图片预加载完成后立即执行此函数。
4. **function** loadImages(sources, callback){
5. **var** count = 0,
6. images ={},
7. imgNum = 0;
8. **for**(src **in** sources){
9. imgNum++;
10. }
11. **for**(src **in** sources){
12. images[src] = **new** Image();
13. images[src].onload = **function**(){
14. **if**(++count >= imgNum){
15. callback(images);
16. }
17. }
18. images[src].src = sources[src];
19. }
20. }

（2）在JS中调用预加载函数：

**Js代码  收藏代码**

1. //存储图片链接信息的关联数组
2. **var** sources = {
3. ietoHell : "img/IEtoHell.jpg",
4. fuckIE : "img/fuckIE.jpg"
5. };
7. //调用图片预加载函数，实现回调函数
8. loadImages(sources, **function**(images){
9. //TO-DO something
10. ctx.drawImage(images.ietoHell, 20,20,100,100);
11. ctx.drawImage(images.fuckIE, 140,20,100,100);
12. });

注意事项：

**（1）先绑定image.onload事件，后加载图片**

**Js代码  收藏代码**

1. images[src] = **new** Image();
2. images[src].onload = **function**(){
3. **if**(++count >= imgNum){
4. callback(images);
5. }
6. }
7. images[src].src = sources[src];

**原因：**如果图片从缓存中加载，速度非常快以至于没来得及绑定事件就加载完毕，自然不会触发绑定事件。

**（2）for...in循环 与 for循环的区别**

        for循环用于迭代数组（array）

        for...in循环用于迭代对象（object, {}）或者关联数组（hash array）

源引：http://taro.iteye.com/blog/1562901

## 将ULXIAMIAN 的LI标签绑定事件并弹出对应标签内的文本值。

## 使用事件委托的方法（可能是子节点的子节点 所以要判断父节点）

## 改进版：

<**div**>  
 <**ul id="test"**>  
 <**li**>Glen</**li**>  
 <**li**>Tane</**li**>  
 <**li**>John</**li**>  
 <**li**>Ralph<**ul**><**li**>test</**li**></**ul**>  
 </**li**>  
 </**ul**>  
</**div**>  
<**script**>  
 **window**.onload = **function**(){  
 **var** obj\_lis = **document**.getElementById(**"test"**) ;  
  
 obj\_lis.**onclick** = **function**(e){  
  
 **var** tar=e.**target**;  
 **var** oar=tar.**parentNode**;  
 **if**(tar.**parentNode**===obj\_lis) {  
 alert(tar.**innerHTML**);  
 }  
 }  
  
 }  
</**script**>

# JS编程题 计算器

<!DOCTYPE **HTML** PUBLIC **"-//W3C//DTD HTML 4.01 Frameset//EN"**>  
<**html**>  
<**head**>  
 <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"**>  
 <**title**>计算器</**title**>  
 <**style type="text/css"**>  
 **\*** {  
 **margin**: 0**px**;  
 **padding**: 0**px**;  
 }  
  
 **td** {  
 **background-color**: **#303133**;  
 **width**: 100**px**;  
 **height**: 80**px**;  
 **color**: **white**;  
 **text-align**: **center**;  
 **font-size**: 20**px**;  
 }  
  
 .**td\_orange** {  
 **background-color**: **#FA6F14**;  
 }  
 </**style**>  
  
 <**script type="text/javascript"**>  
 *//显示用 在onclick="appContent(this)" 传入的是当前被点击的对象* **function** *appContent*(td){  
 *//找到显示字符串等式的td标签* **var** content = **document**.getElementById(**"content"**);  
 *//找到显示结果的td标签* **var** result = **document**.getElementById(**"result"**);  
 *//获取字符串的等式* **var** text = td.**innerText**;  
  
 *//如果是删除键 删除一个字符* **if**(**"del"** == text){  
 **if**(content.**innerText**.**length** > 0){  
 *//删除最后一个字符* content.**innerText** = content.**innerText**.substring(0,content.**innerText**.**length**-1);  
 *//一个新的字符串，该字符串值包含 stringObject 的一个子字符串，其内容是从 start 处到 stop-1 处的所有字符* }  
 *//如果是全部删除* }**else if**(**"c"** == text){  
 content.**innerText** = **""**;  
 *//如果是按了等于号* }**else if**(**"="** == text){  
 **var** resultText = *parse*(content.**innerText**);  
 result.**innerText** = content.**innerText** + **"="** + resultText;  
 content.**innerText** = **""**;  
 *//除了上面三种情况,其他的都是尾加* }**else**{  
 content.**innerText** = content.**innerText** + text;  
 }  
  
 }  
  
 */\*\*  
 \* 解析字符串的等式为一个正确的结果  
 \*/* **function** *parse*(content){  
  
 *//寻找最后一个左括号* **var** index = content.lastIndexOf(**"("**);  
  
 *//如果等式中有左括号* **if**(index > -1){  
 *//寻找右括号,从左括号的位置开始寻找* **var** endIndex = content.indexOf(**")"**,index);  
  
 *//如果等式中有右括号* **if**(endIndex > -1){  
 *//调用自己算出括号中的结果* **var** result = *parse*(content.substring(index + 1,endIndex));  
 *//然后继续调用自己,  
 //其实这里完成的工作就是"2+3+(2+3\*2)"转化成了"2+3+8",也就是用括号中的结果替换括号所在位置  
 //+ 不是真正的+号 而是字符串的连接* **return** *parse*(content.substring(0,index) + (**""** + result) + content.substring(endIndex + 1))  
 }  
 }  
  
 index = content.indexOf(**"+"**);  
  
 **if**(index > -1){  
 **return** *parse*(content.substring(0,index)) + *parse*(content.substring(index + 1));  
 }  
  
 index = content.lastIndexOf(**"-"**);  
  
 **if**(index > -1){  
 **return** *parse*(content.substring(0,index)) - *parse*(content.substring(index + 1));  
 }  
  
 index = content.lastIndexOf(**"\*"**);  
  
 **if**(index > -1){  
 **return** *parse*(content.substring(0,index)) \* *parse*(content.substring(index + 1));  
 }  
  
 index = content.lastIndexOf(**"/"**);  
  
 **if**(index > -1){  
 **return** *parse*(content.substring(0,index)) / *parse*(content.substring(index + 1));  
 }  
  
 **if**(**""** == content){  
 *// /8(这种情况)* **return** 0;  
 }**else**{  
 *//转化为数字--当剩下最后一个数字的时候* **return** content - 1 + 1;  
 }  
 }  
  
 </**script**>  
  
</**head**>  
<**body**>  
<**table align="center"**>  
 <**tr**>  
 <**td class="td\_orange" colspan="5" id="content"**></**td**>  
 </**tr**>  
 <**tr**>  
 <**td class="td\_orange" colspan="5" id="result"**></**td**>  
 </**tr**>  
 <**tr**>  
 <**td onclick="***appContent*(**this**)**"**>1</**td**>  
 <**td onclick="***appContent*(**this**)**"**>2</**td**>  
 <**td onclick="***appContent*(**this**)**"**>3</**td**>  
 <**td onclick="***appContent*(**this**)**"**>\*</**td**>  
 <**td onclick="***appContent*(**this**)**"**>/</**td**>  
 </**tr**>  
 <**tr**>  
 <**td onclick="***appContent*(**this**)**"**>4</**td**>  
 <**td onclick="***appContent*(**this**)**"**>5</**td**>  
 <**td onclick="***appContent*(**this**)**"**>6</**td**>  
 <**td onclick="***appContent*(**this**)**"**>+</**td**>  
 <**td onclick="***appContent*(**this**)**"**>-</**td**>  
 </**tr**>  
 <**tr**>  
 <**td onclick="***appContent*(**this**)**"**>7</**td**>  
 <**td onclick="***appContent*(**this**)**"**>8</**td**>  
 <**td onclick="***appContent*(**this**)**"**>9</**td**>  
 <**td onclick="***appContent*(**this**)**"**>(</**td**>  
 <**td onclick="***appContent*(**this**)**"**>)</**td**>  
 </**tr**>  
 <**tr**>  
 <**td class="td\_orange" onclick="***appContent*(**this**)**"**>c</**td**>  
 <**td onclick="***appContent*(**this**)**"**>0</**td**>  
 <**td onclick="***appContent*(**this**)**"**>.</**td**>  
 <**td onclick="***appContent*(**this**)**"**>del</**td**>  
 <**td class="td\_orange" onclick="***appContent*(**this**)**"**>=</**td**>  
 </**tr**>  
</**table**>  
</**body**>  
</**html**>

思路：

1两个函数 一个用于显示结果和输入的情况

2一个用于把输入的表达式变成结果

3目前觉得全都用index = content.indexOf(**"\*"**);也是没问题的

//笔试经验

1可以跳出 不要跳出太多次就行 可以先关闭浏览器 看好时间

实现一个MAP（考察的是原型）

<!DOCTYPE **HTML** PUBLIC **"-//W3C//DTD HTML 4.01 Frameset//EN"**>  
<**html**>  
<**head**>  
 <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"**>  
  
  
  
 <**script**>  
 Array.**prototype**.remove = **function**(s) {  
 **for** (**var** i = 0; i < **this**.**length**; i++) {  
 **if** (s == **this**[i])  
 **this**.splice(i, 1);  
 }  
 }  
 **function** *Map*() {  
 */\*\* 存放键的数组(遍历用到) \*/* **this**.**keys** = **new** Array();  
 */\*\* 存放数据 对象方便查找是否存在这个值\*/* **this**.**data** = **new** Object();  
 */\*\*  
 \* 放入一个键值对  
 \** ***@param*** *{String} key  
 \** ***@param*** *{Object} value  
 \*/* **this**.put = **function**(key, value) {  
 **if**(**this**.**data**[key] == **null**){  
 **this**.**keys**.push(key);  
 }  
 **this**.**data**[key] = value;  
 };  
  
 */\*\*  
 \* 获取某键对应的值  
 \** ***@param*** *{String} key  
 \** ***@return*** *{Object} value  
 \*/* **this**.get = **function**(key) {  
 **return this**.**data**[key];  
 };  
  
 */\*\*  
 \* 删除一个键值对  
 \** ***@param*** *{String} key  
 \*/* **this**.remove = **function**(key) {  
 **this**.**keys**.remove(key);  
 **this**.**data**[key] = **null**;  
 };  
  
 */\*\*  
 \* 遍历Map,执行处理函数  
 \*  
 \** ***@param*** *{Function} 回调函数 function(key,value,index){..}  
 \*/* **this**.each = **function**(fn){  
 **if**(typeof fn != **'function'**){  
 **return**;  
 }  
 **var** len = **this**.**keys**.**length**;  
 **for**(**var** i=0;i<len;i++){  
 **var** k = **this**.**keys**[i];  
 *//传key val index进去执行* fn(k,**this**.**data**[k],i);  
 }  
 };  
  
 */\*\*  
 \* 获取键值数组(类似Java的entrySet())  
 \** ***@return*** *键值对象{key,value}的数组  
 \*/* **this**.entrys = **function**() {  
 **var** len = **this**.**keys**.**length**;  
 *//key其实是用来获取长度的数组（主要作用）* **var** entrys = **new** Array(len);  
 **for** (**var** i = 0; i < len; i++) {  
 entrys[i] = {  
 **key** : **this**.**keys**[i],  
 *//对象的另一种取值方法* **value** : **this**.**data**[i]  
 };  
 }  
 **return** entrys;  
 };  
  
 */\*\*  
 \* 判断Map是否为空  
 \*/* **this**.isEmpty = **function**() {  
 **return this**.**keys**.**length** == 0;  
 };  
  
 */\*\*  
 \* 获取键值对数量  
 \*/* **this**.size = **function**(){  
 **return this**.**keys**.**length**;  
 };  
  
 */\*\*  
 \* 重写toString 把一组KEY VAL转化成字符串  
 \*/* **this**.toString = **function**(){  
 **var** s = **"{"**;  
 **for**(**var** i=0;i<**this**.**keys**.**length**;i++,s+=**','**){  
 **var** k = **this**.**keys**[i];  
 s += k+**"="**+**this**.**data**[k];  
 }  
 s+=**"}"**;  
 **return** s;  
 };  
 }  
  
  
 **function** *testMap*(){  
 **var** m = **new** *Map*();  
 m.put(**'key1'**,**'Comtop'**);  
 m.put(**'key2'**,**'南方电网'**);  
 m.put(**'key3'**,**'景新花园'**);  
 alert(**"init:"**+m);  
  
 m.put(**'key1'**,**'康拓普'**);  
 alert(**"set key1:"**+m);  
  
 m.remove(**"key2"**);  
 alert(**"remove key2: "**+m);  
  
 **var** s =**""**;  
 m.each(**function**(key,value,index){  
 s += index+**":"**+ key+**"="**+value+**"\n"**;  
 });  
 alert(s);  
 }  
 *testMap*();  
 </**script**>  
  
  
</**head**>  
<**body**>  
  
</**body**>  
</**html**>

# TODOLIST

//上面是输入的部分

//下面的UL负责显示

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
</**head**>  
<**body**>  
<**style**>  
 **\*** {  
 **margin**: 0;  
 **padding**: 0;  
 }  
 **#todos** {  
 **overflow**: **hidden**;  
 **width**: 300**px**;  
 **margin**: 0 **auto**;  
 **border**: 1**px solid #CCCCCC**;  
 **box-shadow**: **#2f3038** 1**px** 2**px** 2**px**;  
 }  
 **input#content** {  
 **width**: 100%;  
 **outline**: 0;  
 **border**: **none**;  
 **padding-left**: 30**px**;  
 **padding-top**: 10**px**;  
 **padding-bottom**: 10**px**;  
 }  
 **#tips** {  
 **border-top**: 1**px solid #CCCCCC**;  
 **list-style**: **none**;  
 }  
  
 **#tips** > **li** {  
 **border**: 1**px solid #CCCCCC**;  
 **padding-left**: 30**px**;  
 **padding-top**: 10**px**;  
 **padding-bottom**: 10**px**;  
 **position**: **relative**;  
 }  
  
 .**wipe** {  
 **position**: **absolute**;  
 **top**: 10**px**;  
 **right**: 20**px**;  
 }  
  
 **#title** {  
 **text-align**: **center**;  
 }  
</**style**>  
<**p id="title"**>todos</**p**>  
<**div id="todos"**>  
 <**label**>  
 <**input type="text" id="content"**>  
 </**label**>  
 <**ul id="tips"**></**ul**>  
</**div**>  
<**script**>  
 *//一般都会在自身里面存一个和dom元素结构对应的数组* ;(**function** () {  
 **function** *Todos*(id, otip, oInput) {  
 **this**.**ele** = **document**.querySelector(id);  
 **this**.**oInput** = **document**.querySelector(oInput);  
 **this**.**oTip** = **document**.querySelector(otip);  
 *// use for search 用来查找需要添加的行的* **this**.**map** = [];*//本质是一个数组* **this**.init();  
 }  
 *Todos*.**prototype** = {  
 contructor: *Todos*,  
 init: **function** () {  
 **this**.closeHandler();  
 **this**.inputHandler();  
 },  
 *//事件代理 对参数中的函数传入事件的目标元素* \_delegate: **function** (e, tagName, fn) {  
 **var** event = e || **window**.**event**,  
 target = event.**target** || event.**srcElement**;  
 **if** (target.**tagName**.toLowerCase() == tagName) {  
 fn.call(**null**, target);  
 }  
 },  
 *//删除某个添加的行* closeHandler: **function** () {  
 **var** self = **this**;  
 self.**oTip**.onclick = **function** (e) {  
 self.\_delegate(e, **'span'**, **function** (target) {  
  
 *//应该先从map里面删除在 删除DOM 不然不能获得LI的位置* **var** index = [].indexOf.call(self.**oTip**.**children**, target.parentNode);*//这一句不是很理解* self.**map**.splice(index, 1);*//删除该元素* self.**oTip**.removeChild(target.parentNode);  
 })  
 }  
 },  
 *//输入时的匹配字显示* inputHandler: **function** () {  
 **var** self = **this**;  
 self.**ele**.onkeyup = **function** (e) {  
 **var** nonce = self.**oInput**.**value**;  
 **var** reg = **new** RegExp(nonce, **'g'**);  
 **var** oLi = self.**ele**.getElementsByTagName(**'li'**);  
 self.**map**.forEach(**function** (ele, ind) {  
 oLi[ind].**style**.**display** = **'block'**;  
 **var** last = **"<span class='wipe'>X</span>"**;  
 **if** (reg.test(ele)) {  
 oLi[ind].**innerHTML** = ele.replace(reg, **"<span style='color: red;'>"** + nonce + **"</span>"** )+ last;  
 } **else** {  
 oLi[ind].**innerHTML**= oLi[ind].**innerText**.slice(0,-1) +last;  
 oLi[ind].**style**.**display** = **'block'**;  
 oLi[ind].**style**.**color** =**"black"** ;  
 }  
 });  
  
 self.enterHandler(e);  
 }  
 },  
 *//添加新行并且清空输入框里面的内容* enterHandler: **function** (e) {  
 **if** (e.**keyCode** == 13) {*//回车键* **var** nonce = **this**.**oInput**.**value**;  
 **this**.**map**.push(nonce);*//map存储的是每行的内容* **var** newNode = **document**.createElement(**'li'**);  
 newNode.**innerHTML** = nonce + **"<span class='wipe'>X</span>"**;  
 **this**.**oTip**.appendChild(newNode);  
  
 **this**.**oInput**.**value** = **''**;  
 }  
 }  
 }  
 **new** *Todos*(**'#todos'**, **'#tips'**, **'#content'**);  
 })();  
</**script**>  
</**body**>  
</**html**>

# 证明只有在事件冒泡阶段出发绑定事件

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>任务三十九：UI组件之冻结行列表格</**title**>  
  
</**head**>  
<**body**>  
<**style**>  
 [**class**\*=**'level-'**] {  
 **border**: 1**px solid #593c7f**; }  
  
 [**class**\*=**'level-'**] {  
 **display**: **flex**;  
 **justify-content**: **space-around**;  
 **align-items**: **center**; }  
  
 **body** {  
 **font-family**: **"arial"**; }  
  
 [**class**\*=**'level-'**] {  
 **background**: **#fff**;  
 **padding**: 0 5**px**;  
 **margin**: 0 5**px**; }  
  
 .**level-1** {  
 **height**: 150**px**; }  
  
 .**level-2** {  
 **height**: 120**px**; }  
  
 .**level-3** {  
 **height**: 90**px**; }  
  
 .**level-4** {  
 **height**: 60**px**; }  
  
 .**level-5** {  
 **height**: 30**px**; }  
  
 **#box-container**,  
 **#traverse-container**,  
 **#search-container**,  
 **#insert-delete** {  
 **display**: **flex**;  
 **justify-content**: **center**;  
 **margin-top**: 30**px**; }  
  
 **input** {  
 **margin**: 0 5**px**; }  
  
 */\*# sourceMappingURL=main.css.map \*/*</**style**>  
<**div id="wrapper"**>  
 <**div id="box-container"**>  
 <**div class="level-1" id="top-level"**>Super  
 <**div class="level-2"**>Co  
 <**div class="level-3"**>Apple  
 <**div class="level-4"**>Pear</**div**>  
 <**div class="level-4"**>Pig</**div**>  
 <**div class="level-4"**>Cola</**div**>  
 <**div class="level-4"**>Soccer</**div**>  
 </**div**>  
 <**div class="level-3"**>Phone</**div**>  
 <**div class="level-3"**>  
 <**div class="level-4"**>Book</**div**>  
 <**div class="level-4"**>School</**div**>  
 </**div**>  
 </**div**>  
 <**div class="level-2"**>Note  
 <**div class="level-3"**>Human  
 <**div class="level-4"**>Code</**div**>  
 <**div class="level-4"**>Operate</**div**>  
 <**div class="level-4"**>Man</**div**>  
 </**div**>  
 <**div class="level-3"**>Program  
 <**div class="level-4"**>Element  
 <**div class="level-5"**>Cat</**div**>  
 </**div**>  
 <**div class="level-4"**>Glass</**div**>  
 </**div**>  
 </**div**>  
 <**div class="level-2"**>Fish</**div**>  
 </**div**>  
 </**div**>  
</**div**>  
<**script**>  
 **var *count***=0;  
 **var *levels***=**document**.getElementById(**"box-container"**).getElementsByTagName(**"div"**);  
 **for** (**var *i*** = 0; ***i*** < ***levels***.**length**; ***i***++) {  
 ***levels***[***i***].onclick=**function**(e){  
 **var** self=**this**;  
 **var** ele=**document**.createTextNode(***count***++);  
 *//因为在这个过程中self是改变的* self.appendChild(ele);  
 };  
 }  
</**script**>  
  
</**body**>  
</**html**>