////////////////////////////////////////////////////////////////////////////////////////////

~内部引入与外部引入

内部的话就是在<head>标签内写<script type = "text/javascript"></script>。外部引入我比较喜欢，有封装的感觉。

<script type = "text/javascript" src = "" charset = "gbk" />src里面写资源位置。但还是改成utf-8会比较好，省掉很多麻烦。

////////////////////////////////////////////////////////////////////////////////////////////

~弱类型语言

var a;

a = 1; //输出一下后

a = "string"; //可以这么玩！

基本类型有，数值类型，字符串类型，布尔型。null

////////////////////////////////////////////////////////////////////////////////////////////

~比较运算符

因为是弱类型语言，所以比较起来会随意一点。 == 运算符是不在乎类型的

（x = 10）x == 10 true； x == ‘10’true；

但 === 就比较严格，必须类型相同。也只有弱类型语言才需要三个=。那么既然有 ===

也有 ！== 含义和 === 相反理解方式相同。

////////////////////////////////////////////////////////////////////////////////////////////

~函数

function fun（param）{

}

弱类型语言，大家都是 var 哪还有什么好争得，所以形参里不要特意声明（int。。。）返回值也不用列出类型，而且没有就是没有，不需要void。简约到骨子里了，在下去只怕连程序的影子都没了

下面这种写法会立即执行大括号内的代码

(**function** () { /\* code \*/ } ());

!**function** () { /\* code \*/ } ();

~**function** () { /\* code \*/ } ();

-**function** () { /\* code \*/ } ();

+**function** () { /\* code \*/ } ();

我目前所写过的函数，都是为了事件处理，onclick，onload，onblur，etc. JavaScript的函数也有引用（类似于C++的函数指针吧），因此如果要给按钮之类的事件更换函数，可以这么写。

myButton.onclick = myFunc;

然而，我最开始是这么想的。

myButton.setAttribute(“onclick”, “myFunc”);

结果是，该节点压根就没有setAttribute这个方法。而且，无论是<button>还是<input>，都没有。这不是元素节点啊，但，他奶奶的，好奇怪啊。

////////////////////////////////////////////////////////////////////////////////////////////

~对象

基于原型的对象，Java的对象是基于类的。至于什么叫基于原型，我个人是这么想的。JavaScript只有Object这一个类，那些看上去不同的对象都是先创建Object对象，然后在动态的添加属性或方法。而Java是有类定义的，类与类之间是不相同的。

var p = new Object();

它可以动态的添加属性与方法

p.name = "jack";

p.func = speak; //speak为函数

有添加就有删除

delete p.name;

delete p.func;

也可以这么删

p.name = undefined; // undefined是全局变量

p.func = undefined;

// 这个还真是非常动态，Java是不行的。怎么说吧，这个看了，很没安全感。而且也不方便管理

除了Object，还有几个常用的的内置对象

* Number
* String
* Date
* Array
* Math

但，就目前而言，对我来讲，真正用得到的，也就String，Date。以后再补吧

////////////////////////////////////////////////////////////////////////////////////////////

~常用函数

全局函数

escape() // 对字符串进行编码

unescape() // 解码，不过这两个函数貌似不用了

* encodeURI() // 用过，编码

eval() // 计算字符串，并将它作为脚本运行

isNan() // 是否数字

isFinite() // 是否无穷大

parseFloat() // 将字符串变为浮点数

parseInt() // 将字符串转为整数

alert() //提示框

confirm() //带选择按钮的提示框，按钮带值

prompt() //带输入框的提示框（有提示信息）

其实这些东西都是window对象的函数，window，BOM（browser object model）的核心。真要细分，document也是window的子对象。另外，上面写的那些函数都是有参数的，全局函数以及window对象的函数。alert(), confirm()以字符创作参数。prompt()有两个字符串，一个是提示信息，还有一个是输入框默认信息。当然可以不写。

Window对象有很多函数的，不一一列举了（不过以后还是有可能的），下面在看两个

setTimeout(callback, delay)，参数两个，一个回调函数，一个延迟毫秒数。

// 真他妈脑抽了，刚才居然在想callback是不是要写在setTimeout里面，唉。

setInterval(callback, delay)，同理，但上面的只调用一次，这个会调用到天荒地老。

window.open(url, windowName, windowFeatures, optionalArg4)

// 前三个还是很好理解的，最后一个与历史记录有关，至于这个历史记录是哪个意思，我，诶呀，一头雾水啊。

多查文档！

////////////////////////////////////////////////////////////////////////////////////////////

~function对象

我倒是不奇怪可以把函数作为参数传给另一个函数（C语言的函数指针），但像创建对象那样创建函数，还真是。。。。

var myFunc = new Function（“...”,“....”）;

// 最后一个参数是函数体，前面的都是参数。这样创建的函数是没有名字的，myFunc只能算是引用。

////////////////////////////////////////////////////////////////////////////////////////////

DOM对象

DOM（document object model）文档对象模型

1. .html是一颗树，数据结构的树。<html>是根节点，往下<head>，<body>等为子节点。然后<body>等标签又有<div><h1>这种子节点，子子孙孙无穷尽也。但要注意的是，并不是只有节点才可以成为节点的孩子。

例：<h1>hello world ! <small>come to fuck!</small></h1>

首先很明显的，<small>是<h1>的孩子，但同时”hello world！”也是<h1>的孩子。同理，“come to fuck！”是<small>的孩子

一个节点就是一个对象，对象有类型（element），对象本身又有值（text）和属性（attribute）

// 敲了这么久代码，这一刻我才突然反应过来。唔，HTML里只有文本，没有其他任何东西，但真要说起来，什么程序不是单纯的文本文档呢。这就是所谓的“跑得太快结果连自己是在跑都给忘了”么。他奶奶的！

* document.getElementById()对应一个对象（元素）
* .innerHTML与 .value对应对象的值
* .style与 .className之类的是对象的属性，然后属性也有自己的值

2. 当HTML或定义的CSS被发送给web浏览器时，原网页会被转换成DOM对象，而不是文本。无论代码简单或复杂，集中到一个文件还是分散到多个文件，都是如此。然后浏览器会直接使用该对象模型，也就是DOM树。

修改DOM树（增删查改），结果会直接呈现在用户浏览器上，而不需要重新加载页面。这就大大提高了web编程的灵活性。一旦学会了它就可以，嗯，不对。东西本身还是很好学的，能用到什么地步，能写出多强力的网站还是看个人想象力。

DOM树的基础单位是节点，节点分为element节点、text节点、attribute节点，虽然分了三类，但大部分情况下还是对element节点进行操作。

// 长见识嘿，.value这玩意也算节点

以下为节点属性：

* nodeName 节点名称
* nodeValue 节点的值
* parentNode 返回父节点
* childNodes 返回子节点列表，仅对element节点有用
* firstChild & lastChild // 这四个没什么好讲的
* previousSibling & nextSibling
* attributes 仅用于element节点，返回属性列表

以下为节点方法：

* insertBefore( newChild, referenceNode ) // 将新子节点插到某子节点之前。
  + replaceChild( newChild, oldChild ) // 新的替换旧的
* removeChild( oldChild ) // 删删删！
* appendChild( newChild ) // 新节点会自动添加至列表末端
* hasChildNodes() // true or false
* hasAttributes() // true or false

注意，这些方法都是针对子节点的，看名字就懂。因为，如果只是想获取值或属性的话，节点属性就够用了。

1. 常用节点

document是用的最多的节点之一（我目前只用过这个）。该节点代表整个DOM树，功能强大，比如创建节点。

* createElement( elementName )
* createTextNode( text )
* createAttribute( attributeName )

这三个方法可以创造一个非常完整的节点，类型、内容、属性。但也只是创建，要想发挥作用必须要插入DOM树，这里就要用到上面insertBefore()之类的方法。

元素节点用的也蛮多，（element，document的子节点）。<a>,<body>等等等等。它只有两组专用方法。（只有是什么意思，嫌少？）

* getAttribute（name） 返回名为name的属性值
* removeAttribute（name） 删删删
* setAttribute（name， value） 注意，创建名为name的属性并赋值
* getAttributeNode（name） 返回名为name的属性节点，属性节点
* removeAttributeNode（node） 删除与给定node相匹配的属性节点
* getElementsByTagName(elementName) 返回具有指定名称的元素节点列表。

属性节点和文本节点等用到了再补。

1. 一些常量

意思是，如果以后获取了某节点，但不确定其类型，那么就可以用常量来判断。当然了，要和nodeType一起用才可以判断。

* Node.ELEMENT\_NODE 是表示元素节点类型的常量。
* Node.ATTRIBUTE\_NODE 是表示属性节点类型的常量。
* Node.TEXT\_NODE 是表示文本节点类型的常量。
* Node.DOCUMENT\_NODE 是表示文档节点类型的常量。

DOM刷新了我对网页的认识。不过东西就这么多，怎么用还是看人，想象力是束缚人的关键因素，而想象力的来源又是不断的练习与底蕴。所以不能找借口，要多敲代码。

////////////////////////////////////////////////////////////////////////////////////////////

Window

// Window对象真是强大到不行！

1. event

event是事件对象，当然真正吸引我的还是他的某个属性

var element = window.event.srcElement;

这样就可以返回触发该事件的元素，并用这元素去搞事情！

// 哎呀，卧槽，event对象真是吊啊

event.keyCode == 13，13对应的是回车键

////////////////////////////////////////////////////////////////////////////////////////////

JavaScript数组

这玩意的数组是变长的，真吊诡。。。就看下面第二种创建方式，它指定的只是这个数组的临时长度，不是上限，一旦超过了5，长度会增加。

创建数组

1. var arr = new Array();
2. var arr = new Array(5); // 创建长度为5的数组
3. var arr = new Array([1,2,3]); // 创建长度为3的数组，值分别是......
4. var arr = [1,2,3];

属性

length // 安心，可以直接 . 出来

* join()

join() 方法用于把数组中的所有元素放入一个字符串并返回，中间自动用逗号隔开，不想用逗号可以自己设置，像下面这样：

arr.join(“/”);

arr.join(“\”); // 不知道为什么，我这么设的时候，程序就没法运行了。

arr.join(“”); // 中间不要分隔符

* push()

var newLength = arr.push(1); // 增，并返回长度

* pop()

删除最后一个元素并返回改元素值。

**///////////////////////////////////////////////////////////////////////////////////**

***一些我也不知道要怎么分类的东西***

* <script type=*"text/javascript"*>

</script>

JavaScript的代码都是写在这里面的（.js文件另谈），一旦，该文件内某行代码报错，如：

selectType = ;

那么，<script>标签范围内的其它代码也不会生效，嘶，我还是把那个错误直接贴上来吧。

<script type=*"text/javascript"*>

$(document).ready(**function**(){

selectType = ; 这一行是错的

**if**(selectType > 0 && selectType < 7){

**var** node = $("select[name='selectType']").children()[selectType-1];

$(node).attr("selected","");

}

});

**function** searchUser(){

**var** selectCondition = $("#selectCondition").val();

selectCondition = encodeURI(selectCondition);

$("#selectCondition").val(selectCondition);

$("#search").submit();

}

</script>

就因为那一行错，搞得searchUser() 都挂了。。。。

但在外层重新拿个<script>标签包一下，又OK了。

1. <script type="text/javascript">
2. /\* 鼠标点击特效 \*/
3. var a\_idx = 0;
4. jQuery(document).ready(function($) {
5. $("body").click(function(e) {
6. var a = new Array("富强", "民主", "文明", "和谐", "自由", "平等", "公正" ,"法治", "爱国", "敬业", "诚信", "友善");
7. var $i = $("<span/>").text(a[a\_idx]);
8. a\_idx = (a\_idx + 1) % a.length;
9. var x = e.pageX,
10. y = e.pageY;
11. $i.css({
12. "z-index": 999999999999999999999999999999999999999999999999999999999999999999999,
13. "top": y - 20,
14. "left": x,
15. "position": "absolute",
16. "font-weight": "bold",
17. "color": "#ff6651"
18. });
19. $("body").append($i);
20. $i.animate({
21. "top": y - 180,
22. "opacity": 0
23. },
24. 1500,
25. function() {
26. $i.remove();
27. });
28. });
29. });
30. </script>

Array.prototype.forEach.call(document.querySelectorAll(':checked'), **function** (checkbox) {

checkbox.checked **=** **false**;});