一旦你沉淀下去,也许你不用为前进而努力了,但是你却永远见不得阳光了.你沉淀了下去,上面的泥沙就会不断地把你压住,最后你会暗无天日.

我不该是个懦夫，我要比懦夫强大十倍，甚至几十倍。

who am I . I am champion.

ajaxSend( ):函数用于设置当AJAX请求即将被发送时执行的回调函数。

onclick=”calc（this）”;

this指代当前触发的元素对象本身。指的就是按钮对象。

找儿子的时候不能用childNodes[ ]，必须要用getElementsByTagName（“span”）[ 0 ];

往span里面放值：span.innerHTML = “5”;

value是文本节点用的。

$(“span”).text(“要放的内容”);

HTML中的节点分为：1.元素节点2.属性节点3.文本节点

<ul width=”80px ”>

<li>苹果</li>

<li>香蕉</li>

</ul>

ul是元素节点。width是属性节点。“香蕉”是文本节点。

<select id=”sel”>

<option value=”1 ”>选项1</option>

<option value=”2 ”>选项2</option>

</select>

var sel = document.getElementById(“sel”)；

sel.innerHTML;//得到：<option value=”1 ”>选项1</option> <option value=”2 ”>选项2</option>

sel.value = 1;

Selector

window.onload = function(){

unsel=

sel = [ ];

}

用jQuery移除一个元素的的属性：

$(“#code”).removeAttr(“hiden”);

获得一个文本框中的内容：

$(“input”).val( );

得到一个元素里面的文本内容：

$(“span”).text();

得到元素里面的所有内容：

$(“span”).html( );

给一个元素里面放内容：

$(“span”).html(prod\_name);

DOM对象-->Jquery对象

实现方式： $(DOM对象)

function f( ){

var obj = document.getElementById(“d1”);

var $obj = $(obj);

$obj.html(‘hello JQuery’);

}

jQuery对象-->DOM对象

实现方式：

var $obj = $(“#d1”);

//jQuery对象-->DOM

var obj = $(obj).get(0);

obj.innerHTML = ‘hello JQuery’;

HTML是基础，web是一个超文本文件的集合。通常是以.html后缀的文件。

文档类型的声明：

HTML5的格式：<!DOCTYPE html>如果声明出错，会导致JavaScript出错。

XHTML1.0文档严格定义类型，此类型定义的文档只可以使用HTML中定义的标签与元素，不能包含不被W3C推荐的标签。

元素，即标记。

一般页面只要没有async属性和defer属性，浏览器都会按照<script>元素在页面中的先后位置顺序对script文件来进行解析。

<html>

<head></head>

<body>

<!--这里放置内容-->

</body>

</html>

一般在<head>里面放置script文件，意味着必须等到全部的JavaScript代码都被下载，解析和执行完毕后，才能开始呈现页面的内容。所以会使得浏览器在呈现页面在时，出现一段空白期。

现在web程序都会把JavaScript引用放到<body>元素中页面内容的后面

defer该属性的用途是表明脚本在执行时不会影响页面的构造。也就是说，脚本会被延迟到整个页面都解析完毕后再运行。

在不知道一个值是否为null时想要把该值转为字符串用:

var value = null;

alert(String(value));

属性用来修饰元素，属性的声明必须位于开始标记里面。一个元素的属性可能不止一个，多个属性之间用空格隔开。多个属性之间不区分先后顺序。但是定义属性不能重复。

<p align=”center ”></p>

<meta>元数据标记，网页的最基本的信息，非常重要的信息。

<meta name=”keywords ”content=”友谊 , 网站制作 , 前端代码”>

content里面的内容都是写给搜索引擎看的。  
<meta name=”content-type ”content=”text/html;charset=utf-8 ”/>该网页用的是什么样的语言。编码集为utf-8型的。

<meta http-equiv=”refresh ”content=”3 ”>每隔3秒自动刷新一次。

<meta http-equiv=”refresh ”content=”3;url=http://www.souhu.com ”>

<p></p>一般用来写正文的，是段落标记，独占一行。

语义化：标题用标题标签，正文用正文标签。

<div></div>分区标记。

块元素<h1></h1>、p、div、独占一行，会与上下内容换行。

行内元素

span不换行

sub:下标

sup:上标

<s></s>删除线。

<hr/>水平线。

&lt;左尖括号。&gt;右尖括号。

<pre></pre>里面有什么东西都原样输出。

bgcolor:背景颜色；

background：背景图；

路径：URL统一资源定位器。用来定位资源。

绝对路径：指的是文件下最高级目录下开始的完整的路径。无论当前路径是什么，使用绝对路径总能找到链接的文件。即完整的URL组成。相当于写信的完整地址。

http：//www.w3.org/TR/CSS2/syndate.html

http:协议名

[www.w3.org:主机名](http://www.w3.org:主机名)

/TR/CSS2/:目录路径

syndata.html:要链接的文件名

相对路径：从文件当前位置出发，去找要的文件。

在设置超链接的时候如果指定target=”\_blank ”这时点击超链接会出现一个新的窗口。

<a href=”javascript : ; ”>关于我们</a>这时的点击超链接时不会自动置顶。

表格的常用属性：

合并单元格：border-collapse：collapse；

去掉单元格之间的间距：cellspacing

单元格的填充（相当于内边距）：cellpadding

align是水平方向上的对其。

valign是垂直方向上的对其。

valign可以等于top/middle/bottom。

对于行内级元素，如<img>，可以设置垂直对齐方式，定义行内元素的基线相对于该元素所在行的基线的垂直对齐。

vertical-align:baseline/top/middle/center。只对行内元素有效。

colspan:行霸占

rowspan:列霸占

表格可以划分为3个部分：表头、表主体和表尾。

在分了thead和tfoot、tbody以后，thead和tfoot的高度会变小，rowspan（跨行）不可以跨组。

有序列表：

<ol type=”a ”></ol>

type表示的开头的序列的样式。type还可以等于A,

start表示可以从几开始。start里面必须填数字。

<ul type=”square ”></ul>ul的type属性可以表示响应的形状。默认的形状是黑色的实心圆。

结构标记：

<header></header>页面头部内容

<section></section>页面的主题部分

<footer></footer>



以后会很流行的页面布局：结构标记。

结构标记：通常是一些

header：通常是一些介绍信息、导航信息、站点信息。

nav：元素来定义页面的导航链接部分。

section：标签用于定义文档的节。用来为页面上的内容分块，比如定义章节、页眉、页脚或者文档中其他部分。

article：定义网页中一个帖子、文章或者用户评论。

aside：边栏，用于放广告之类的内容。

表单就是用来获取用户信息的。用于显示、收集、并提交信息到服务器。表单有两个部分。 第一部分是前端可见的。第二部分是提交后的表单处理。

<form></form>

属性有method可以定义表单是get还是post形式提交表单。

action：定义表单被提交时发生的动作，通常包含服务方脚本的URL：

action=“/login.do”

当按钮、文本框加上disabled的时候，就不能再点击了

required：必须填写，不填写的时候，会出现提示信息。（不兼容低版本IE）

disabled：禁用，不能写入任何的文本。

readonly：只读，不能写。

单选框必须有name

<input type=”radio ”name=”sex ” />男

name：size菜单行数。

multiple：允许多行。仅适用于多行菜单selected加在option上，定义菜单

$(“#selectList option : selected ”)//选中下拉框中选中的那一行

用js可以直接提交表单，不用写submit。

所有的表单元素提交都得有name。

enctype定义特殊形式的表单。和提交附件有关，表单里的编码方式。

<fieldset></fieldset>控件分组。给表单元素分组。

<fieldset>

<legend>用户登录</legend>

</fieldset>

<iframe></iframe>在网页里面显示其他的网页内容

frameborder属性有0和1的属性。表示该iframe是否有边框。

iframe的属性还有name=”window ”

<a href=”http://www.baidu.com ”target=”window ”></a>

这时候跳转就能在iframe里面显示出超链接网页来。

scrolling=“yes”或者“no”表示的是知否添加滚动条。

细节<details></details>只有Chrome（谷歌）浏览器支持

摘要<summary></summary>

<meter>度量衡，可以显示一个电池的电量。

<time></time>时间元素，常用来定义公里时间或者日期。并不能为网页带来网页显示效果上的变化。常用于对网页添加与时间相关的附加信息。给程序看的。

向要提示出来的信息都得加title。

<mark></mark>把文字用荧光笔画过。

CSS（层叠样式表）的优点：

1.控制页面更加的多样性。

2.属性更加统一，更好记。

实现了将内容与表现分离，提高了代码的可重用性和可维护性。

W3C建议使用CSS样式取代HTML属性，实现内容和表现的分离。

外联样式：<link rel=“stylesheet”style=”text/css ”href=”.css”>

rel是relationship的简写，意思是关系。stylesheet意思是当前文档和网页的关系是样式表。

<style>样式里面只能写CSS样式。

内联方式：定义在单个的HTML元素中写style。样式和结构没有分离。

内部样式表：样式定义在HTML页的头元素中。适合于做个简单的页面。

外部样式：用HTML页面引用的方法，样式表的内容只存在CSS文件里面。更适合大型网页。更方便样式的修改和维护。

继承性：大多数CSS的样式规则可以被继承。

样式优先级。

浏览器缺省设置。

就近优先原则，离你的所要修饰的样式越近的，就听哪个。

内联样式的优先级更高。

内部样式和外部样式的优先级一样高。还是哪个样式离的近就听哪个的。

手动的调整优先级，给属性后面直接加 !important;

在body里面定义文字的大小作用不到标题里面。

而用 \* { }通用选择器。通用选择器，写在 \* 里的样式会给页面中所有的元素都用一遍。

会降低CSS的效率，所以尽量少用。

类选择器可以被多次调用。

同一个属性里面只能写一个class，但是可以连着写class=“ blue smal l”；

限定条件：当p元素调用了blue这个class，样式才生效。

h1.blue {

background-color : yellow ;

}

限定条件，当 h1元素调用了blue这个class，样式才生效。

id选择器的优先级大于class类选择器。

后代选择器，写法：父元素 子元素。

特点：只要是后代，都符合条件，无论子代或者子代的子代。

div >span : 只选中div里面的子代。

伪类选择器：

伪类选择器用于向某些选择器添加特殊的效果。

使用冒号（:）作为结合符，结合符左边是其他选择器，右边是伪类。

CSS伪类选择器可以分为：

链接伪类选择器：

link（未访问过的），适用于尚未访问的链接。

visited（已经访问过的），用于访问过的链接。

动态伪类，用于呈现用户操作。

hover，用于鼠标悬停在某个元素的时候。

active，用于元素被激活的时候。

focus，用于元素获取焦点时。

低版本的IE浏览器不支持除了a:hover以外的任何hover。

选择器优先级的算法：



id选择器的效率比较高。在前面再加元素选择器会影响效率。

尺寸单位：%：子元素占父元素的百分比。

px：像素（计算机上的一个点），一个文字的大小大约占14px；

em:1em相当于当前字体尺寸，2em等于当前字体尺寸的两倍。

R:red G:green B:blue

十六进制的RGB颜色值。#f2f2f2;

#ffffff；白色，所用的光混合在一起。

#000000；黑色。

#rgb的简写：#aa bb cc

width:80%;

min-width:300px;

max-width:600px;

textarea{

min-width:200px;

min-height:150px;

max-width:200px;

max-height:150px;

}

要给想要设置的东西制定范围区间才能保持元素在控制范围内。

因为最大和最小的相同，所以这个就是不变的文本域。

溢出：当在一个块中写111111，不会换行。

overflow：

visible : 溢出的内容可见。

hidden：溢出的内容被隐藏。

scroll：无论任何情况均显示滚动条。

auto：浏览器自动判断，

哪些元素可以设置宽和高：

块级元素都可以设置宽和高：

div，p，h1，ul，ol，li，dl，dt

行内元素设置宽和高不生效。比如：a/span/b/i等。

solid ( 实线 )、dashed（虚线）、dotted（点状线）

border-radius：边框的边角。

边框阴影：

box-shadow：向方框添加一个或者多个阴影。

#d1{

box-shadow : 5px 5px 5px 5px #999;第3个5px为让阴影

模糊。 h-shadow v-shadow blur spread；

val

color : red ;

}

默认框的阴影的颜色和字体的颜色相同。

设置边框的图片：border-image:url( .png) 26 stretch(拉伸)/repeat(平铺)/round(铺满);

outline(轮廓)：一般用来绘制元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

简写方式：outline：color style width;

outline：red solid 1px ;

outline-width：轮廓的宽度。

CSS中一切皆为框。

框模型：box。

margin能决定元素上下左右的间距。

当两个块元素都设置了margin-top/bottom，这时两个块相遇，此时哪个元素大的就听哪个的。

body自带margin。还有p，div，h1，h2，h3...，pre等都是自带margin。

加padding后会将元素增大。

padding：上 右 下 左

行内元素是不可以设置宽高的，但是可以用padding撑开元素的大小。

给行内元素设置margin是只有margin-left和margin-right生效。上下是不生效的。

background-color：背景色，

background-position:设置的是背景图在div中的位置。

CSS sprites：CSS精灵，把图片用position拼在一块。

background:color url( 图像URL ) repeat attachment position

渐变：两种或者多种颜色之间的平滑过渡。

可以指多个中间颜色值。

transparent：透明色。

background-image：背景图路径。

background-repeat：背景图重复有：repeat/no-repeat/repeat-x/repeat-y。

background-position：背景图位置（偏移）值为：

x-position y-position

如果x是正值，背景图向右偏移。

如果x是负值，背景图向左偏移。

如果y是正值，背景图向右偏移。

如果y是正值，背景图向左偏移。

background：url(images/reg\_phone.jpg) no-repeat 0px 40px;

渐变：用background-image定义

background-image：linear-gradient( 180deg , yellow , red );颜色的渐变。

线性渐变：linear-gradient

径向渐变：

文字类的：

控制文字，指定字体：font-family : value1 , value2 ;

文字大小：font-size：value ;

文字加粗：font-weight: normal/bold/value;

文字样式：font-style：normal/italic;

大小写字母显示：font-variant：normal/small-caps

Times和宋体是比较古典一种的字体。微软雅黑是一种比较官方的。英文为Arial。

可以用font-weight:normal去掉h1,等自带加粗的字体样式。

text-align:left/right/center文字的居左，居右和居中，让内容的位置变化，比如文字、图片等。

line-height：每行所占的高度。当为单行文字，当行高等于height时，内容垂直居中。如果是多行文字，增加line-height可以增加行距。

text-decoration：underline/none ,文字可以设置带下划线的，也可以不带下划线。

文本的溢出属性：white-space：nowrap, wrap:换行，nowrap：不换行。因为文字遇到边框后会自动换行。加上white-space后，文字不会换行。

控制英文字与字之间的间距：word-spacing。

控制文字与文字之间的间距：letter-spacing

设置表格中文字垂直居中情况用：vertical-align:top/middle/bottom;

border-collapse：单元格之间的间距，合并：collapse;实现的边之间的重叠。

separate,分开。让table的双线边框合并成单线。

caption可以设置单元格的标题。

table的高和宽可以自适应。

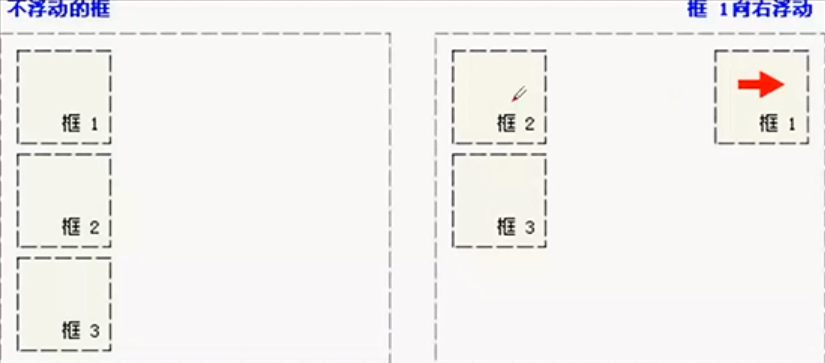
可以定义宽高，table-layout : fixed,让table的宽固定。这时，代码的效率更高。

即可让内容为fix-layout,又让效率提高。

定位概念：定位元素框相对于其正常位置应该出现的位置，或者相对于父元素。另外一个元素甚至浏览器窗口本身的位置。

普通定位：页面只块级元素从上到下接着一个排列，

浮动定位：浮动只能向左或者向右。



浮动的元素不会互相重叠，浮动的元素不会上下浮动。

浮动的和不浮动的会重叠。框会重叠，但是内容不会重叠。

float：left/right/none

当浮动以后会出现bug,错误浮动bug，可以用清除浮动来解决。

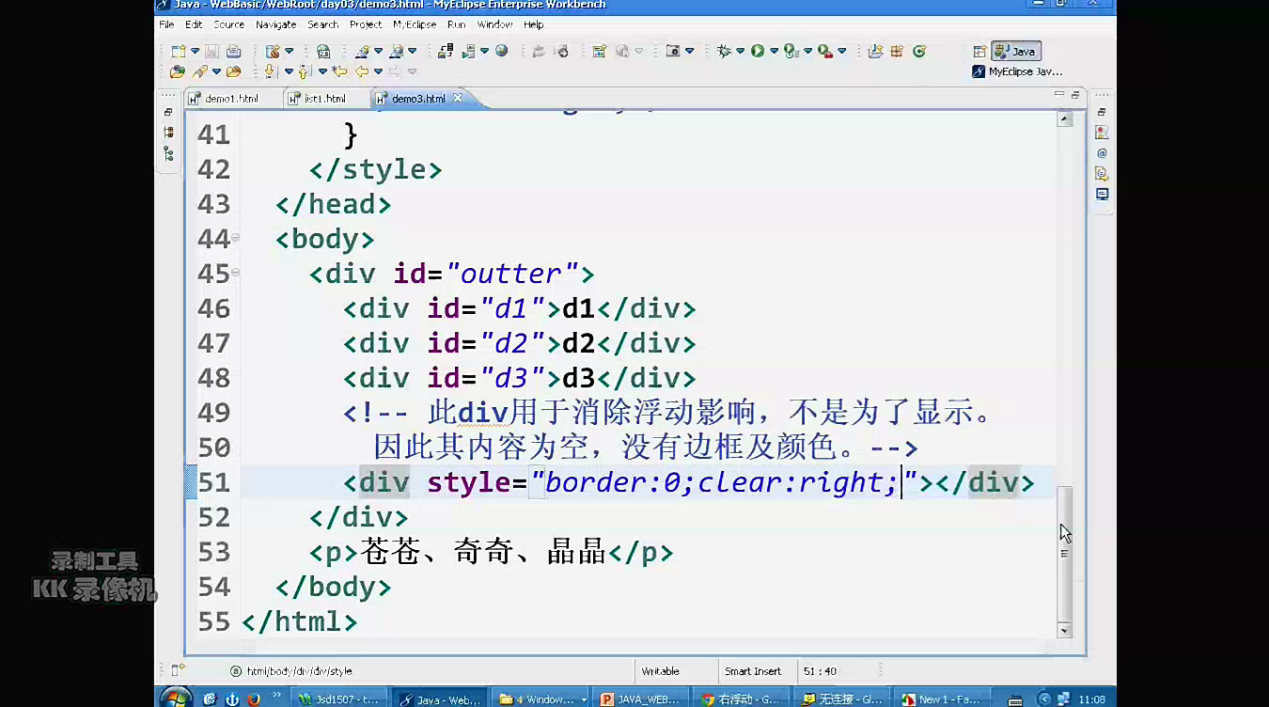
p{

clear : both;//可解决由于浮动产生的bug。

}

当都往左浮动的时候，外面的div就没有了高度，所以是不行的，要自适应高度，就要给父元素加overflow：hidden。

消除浮动影响的方法：



使用ul的规则，ul里只能直接嵌套子ul或者是li。但是嵌套一个span、div是不行的。想要使用span或者div就在<i></i>里面写。

ul.tabs{

margin:0;

padding:0;

}

list-style:none/去掉列表的项目符号。

text-decotration:none;/可以去掉超链接的下划线

.tabs li{

float : left;

font:14px “微软雅黑”；

}

行内元素加上浮动后，就变成块元素。

当.line span{

float : left;/\*此时该行内元素变成块元素\*/

}

当margin-top为负的时候，就是让该元素块往上移。

ul/form/p/h1都会自带内外边距。

display会有inline/block/inline-block;

inline-block：有宽和高的行内元素。即具有行元素的属性又具有块元素的属性。

当display：inline-block，如果加margin-top该行内的元素时，该行的元素都会下来。

inline-block还算是行内元素。

在低版本IE下，行内元素后面加line-block，行内元素照样显示在一行。

如果是块元素后面加inline-block,块元素在同一行显示，

当display：none的时候，就显示不出来该块。让生成的元素根本没有框，该框及其所有内容就不会再显示，不占用文档中的空间。

一般不用不会把div变成行内元素，因为在低版本IE下会有兼容问题。

显示效果：visibility，

可取值：visible：默认值，元素可见。

hidden：元素不可见，但是依然占据空间。

collapse：在表格元素时，可删除一行或者一列，且不影响表格的布局。

display:none时，该元素就会不存在。元素彻底消失，不在占位置了。

visibility：hidden的时候，该元素只是隐藏了，而不是不存在。

opacity属性：用于设置元素不透明级别。

如果设置了父元素为的opacity:0.5为半透明的状态，它的子元素也会继承它，这样它的子元素也会变成透明状的。利用布局原理，可以让一个框为透明的，让另外一框为不透明的。

opacity：value。当opacity：0.0（完全透明），opacity：1.0（完全不透明）。

光标：cursor（鼠标）；

vertical-align属性：设置单元格内容的垂直对齐方式。对于行内块级元素，如 <img>,可以设置垂直对齐方式，行内元素都可以设置该属性。

cursor：default；

pointer；指上去是一只小手

crosshair；指上去是一把剪刀

text；指上去是一个Ⅰ型的符号。

wait;指上去是一个等待的符号。

help;指上去是一个箭头加一个问号的形式。

n-resize：指上去是各个方向的双向箭头。（n 上 e 右 s 下 w 左）

无序列表：ul，

ul{

list-style-type：circle/disk/square/none;

list-style-image:url(images/star.jpg);

list-style-position：inside/outside;

}

相对定位：元素在加相对定位后，没有任何区别，元素 本身所占的空间仍然保留，只用position：relative； 相对于原来的位置发生偏移。还要加top、等才能发生偏移。相当于那个元素上写了position：relative的元素进行偏移。（适合于比较小范围的移动的时候用）

绝对定位：将元素的内容从流中完全的移除，不占任何空间，并使用偏移来固定该元素的位置，相对于最近的已定位。position：absolute；（想要移动的那个元素）这种定位是在于他的父元素的位置。但是父元素必须定位，position：relative；（绝对定位于这个元素）如果没有加他，绝对定位的位置会相对于body开始移动。（适合于比较复杂的移动，动静比较大的时候用）

固定定位：position：fixed；

z-index（设定Z轴的值）:10 ,Z轴越大，越靠近屏幕。z-index要生效，要求参与其中的元素必须都定位。

z-index : 3

用Jquery时，先引入JQuery文件：

<script type=”text/javascript ”src = “JQuery-1.1.js”></script>

<script type=”text/javascript ”>

//当页面包括图像已经完全呈现的时，会发生

//ready( )事件

$(document).ready(function(){

});

</script>

ready( )函数不应与<body onload = “”>一起使用

val( )方法返回或者设置被选元素的value属性。

和后台相关的标签必须有id,和样式有关的有class。

定位：定义元素相对于正常位置应该出现的位置，或者相对于父元素，另外一个元素甚至浏览器窗口本身的位置。

让一行里面摆多个块，可以使用浮动定位。浮动的元素是有限制的，浮动一直在父元素的里面浮动。不可能出父元素。

加浮动和不加浮动的框，框可以重叠，但是内容不会重叠。

浮动与overflow：hidden;把该元素加给父元素，这样

#parent{

/\*给该父元素不定高

让它的高自适应，只定宽\*/

width : 300px;

overflow: hidden;//这时它的高度会自适应

}

伪类选择器：

1. 链接伪类：

a :link{ }没有访问过的链接

javascript : ;//这是不能变成访问过链接的

a :visited{ }未访问过的链接。

什么是JavaScript，如何编写、调试js

前三大语言：

1. HTML：专门编写网页内容的语言
2. CSS：专门美化网页内容样式的语言//前面这两个语言只能看，不能用

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 交互：①用户操作（包含输入，包含点击）②程序处理或相应操作 ③返回结果

javaScript：专门编写交互程序的语言。运行于js解释器（浏览器已经有1.排版引擎，解析HTML/CSS的 2.js解释器--解析JavaScript用的）/引擎中。解释执行（逐行读取js用的）。

由于IE和Navigator浏览器各有各的JavaScript语言，W3C做了DOM标准，专门操作网页的标准。

DOM（文档对象）标准是所有浏览器都兼容的标准。

DOM视图、DOM事件，定义了事件和事件的处理接口DOM样式CSS。DOM遍历和范围，定义了遍历和操作文档树的接口。

BOM：专门操作浏览器窗口的变成接口。

BOM》弹出浏览器窗口的功能，移动、缩放和关闭浏览器窗口的功能、提供浏览器详细信息的navigator对象。提供浏览器所加载页面的详细信息的location对象、提供用户显示器分辨率详 细信息的screen对象、对cookie的支持等对象。

js = ECMAScript标准（核心语法） + DOM标准 + BOM

2004年Netscape倒闭了，后来出现了Mozilla（一个非盈利的提供源码的组织）。后来出现了FireFox。

JavaScript的作用：

1.客户端数据计算。简单的计算

2.表单输入验证。例如正则表达式的验证。

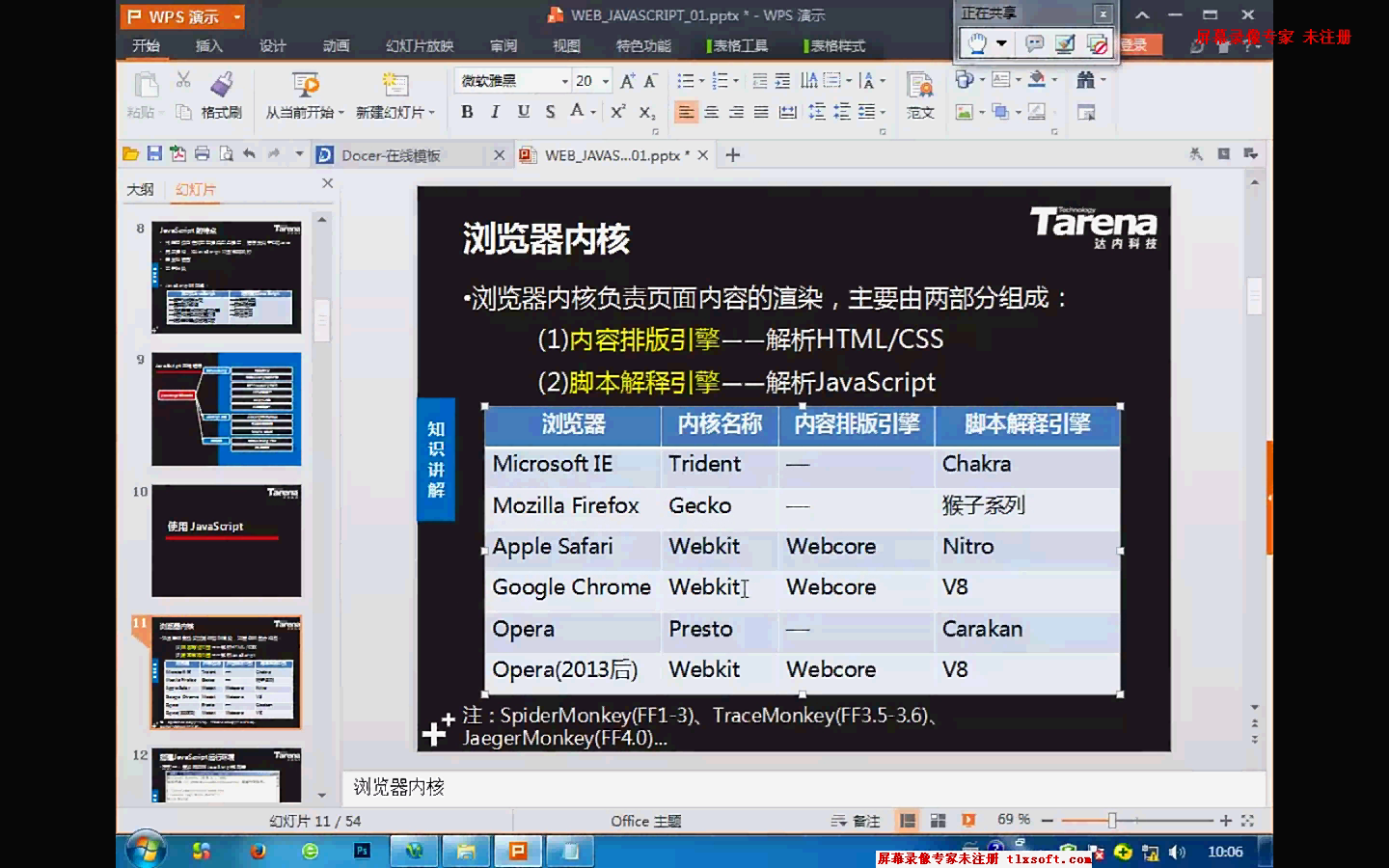
3.浏览器事件的触发操作并处理。一般都是事件触发。

4.网页显示的效果。可以对网页上的任何元素进行想要的操作。

任何使用js:

如何编写，以及如何运行。

Micorosoft IE 内核名称：Trident（用来判断是否为IE11的）



js程序也可以用node.js去执行。

如何编写：4种方式：1.最简单的，直接在浏览器控制台中，编写js脚本。Console：专门编写调试js程序的窗口。必须要会的一个工具。console.log( );//直接输出。 console特指控制台窗口。.：的 log：该功能接收一个字符串参数，在控制台输出该字符。

字符串：一串字符的集合。字符串必须用引号包裹，单双引号都行。

程序中的关键字区分大小写。

字符串中的字符不区分大小写。

2个按键：1.多行程序是：shift+Enter

2.上下箭头。

在网页中编写js脚本程序。：

1. 在<script>中元素。document：表示当前网页，该文档对象。document.write();只能往body里面写。直接编写在<script>中的js程序，随着网页加载过程解释执行。
2. 单独用js文件：用<script src=”url ”></script>

解释执行就是一行一行的执行。

脚本程序：不需要预编译，边解释边执行的程序，没有编译的过程。

3.元素的事件处理程序中编写--onclick=””//里面可以写js程序。写在事件里面的程序必须在事件发生的时候才能执行。window：当前窗口。alert（）：弹出警告框。

变量：内存中保存一个数据的存储空间，再起个名字。

程序中所有的数据都要保存在变量中，再进行处理。

1. 如何声明：var 变量名；

2.赋值：把一个数据存入到变量中。如何赋值：变量名 = 值；

如何取值：放到盒子的数据最终是为了取出来用。取出数据进行运算。

逗号的优先级小于等号的优先级。

为一个没有声明过的变量赋值，不会出错，会自动创建同名变量，再保存数据。

尝试从一个未声明过的变量中取值，会出错 。

一切默认的值为undefined;

在程序正式执行前，会将所有var声明的变量提前到开始位置。

声明提前：在程序正式执行之前，都会将所有var声明的变量提前到开始位置，集中创建。赋值留在原地。

1. 常量：一旦声明并初始化后，值不可改变的量。

只要保存一旦创建，不可擅自改变的量时。

比如 π；如何声明常量：const 常量名 = 值；

常量名要全大写，HTML5中只支持弧度，不支持角度。

数据类型：

内存中存储不同数据的存储格式。

js中的数据类型分为：1.原始类型有五种：值保存在变量本地的数据类型。

5种：Number类型专门保存数字的类型。不加引号，所占8个字节。

String 专门保存字符串的类型。

Boolean专门保存真/假的类型。

undefined 只有一个值undefined

null 表示不指向任何地址

2.引用类型：值不保存在变量本地的数据类型。

n.toFixed(2)该方法为保留2位的数的四舍五入，平时计算全部按8位进行计算，在最后交钱的时候变成2位。

toString(2):按二进制进行解析；charCodeAt(0);//解析为Unicode编号

字符串内容一旦创建，不可改变。要想改变只能创建新字符串，替换旧字符串。

变量是在内存中临时存储的。数据库就是在硬盘上存储数据的文件。

客户端的数据在内存中存储的。

1. 数据类型转换

js是弱类型语言。不用声明变量存储的类型。在赋值时动态决定变量的数据类型。同一个变量，先后可保存不同类型的数据。运算时，js会根据需要，动态转换数据的类型。1）隐式转换：无需干预，自动完成的类型转换。在算数计算中，一切类型都隐式转为number类型再进行计算。

“2”-->2 true/false --> 1/0

特例：在加运算中，只要有一方是字符串，两数据都转为字符串，+运算转为字符串拼接。在其他运算的时候，直接转为其他运算。

NaN:说明运算中包含了无法转为数字的值。

typeof( x )：专门判断x的数据类型，返回数据类型名。

typeof( NaN ):number

NaN和任何数据做计算，永远返回NaN。

1. 运算符合表达式，无论表达式的结果如何复杂，他最后的结果都是一个值。

2）强制转换：程序员通过调用专门的函数，执行转换。

1.任意类型转为String类型

var str = x.toString( );//js中，一切数据都有toString方法。

var str = String(x); -->隐式转换，何时使用，程序在需要时自动调用。

2.任意类型

var num = Number( x );//要求里面必须是纯数字的。-->隐式转换

var num = parseInt(str);//从开始位置逐个2读取每个字符，直到碰到第一个不是数字的字符时，停止读取。并且自动忽略开头碰到的空格。

比如：var width= “12px”；//给它加2px直接用parseInt（）来执行。

var add = parseInt（width）+parseInt（“2px”）;

var width = add +”px ”;

var money = ”$400 ”;//parseInt（money），变为NaN。

parseInt（）遇到小数点也会退出。

var n = “23.23”; var d = parseFloat( str );

比如：parseFloat（“34.56”）；

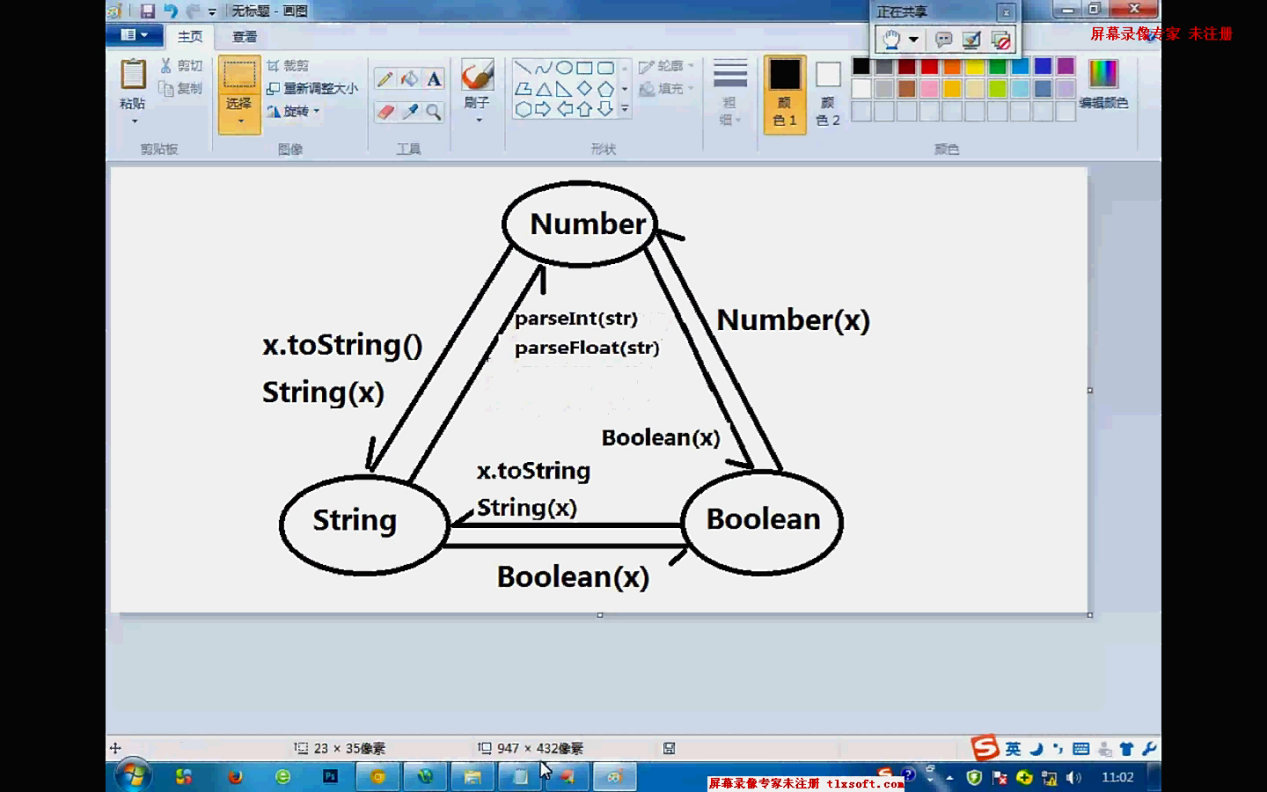
String --- Number时， 用parseInt( );

不是字符串的转为Number型的（不知道的情况下用）Number（x）;

1. 任意类型转boolean：1种方法：var bool = Boolean( x );

只有5个值被转为false：“”、NaN、 undefined、 null、 0

其余都为true。



var input = window.prompt(“输入提示”)；

凡是从页面上获得的一切数据都是字符串类型的。

window对象是浏览器的默认对象，是可以省略的。

程序：让计算机按照人的想法去执行任务。

1. 关系运算：将两个值做比较：> 、< 、>= 、<=、==、！=，在比较的时候把字符串都隐式转换为了number类型的进行比较。

只能返回两个值之一：如果成立，返回true，否则返回false

他们之间存在着隐式转换：将所有类型都转为number类型再进行比较。

但是有三种特殊情况：

1.两个字符串做比较：比较他们的Unicode值。

2.NaN和不是能转为数字的比较为true；NaN和谁比较都不同包括自己。所以只能用isNaN(num);在判断的时候一定用IsNaN( )来判断。

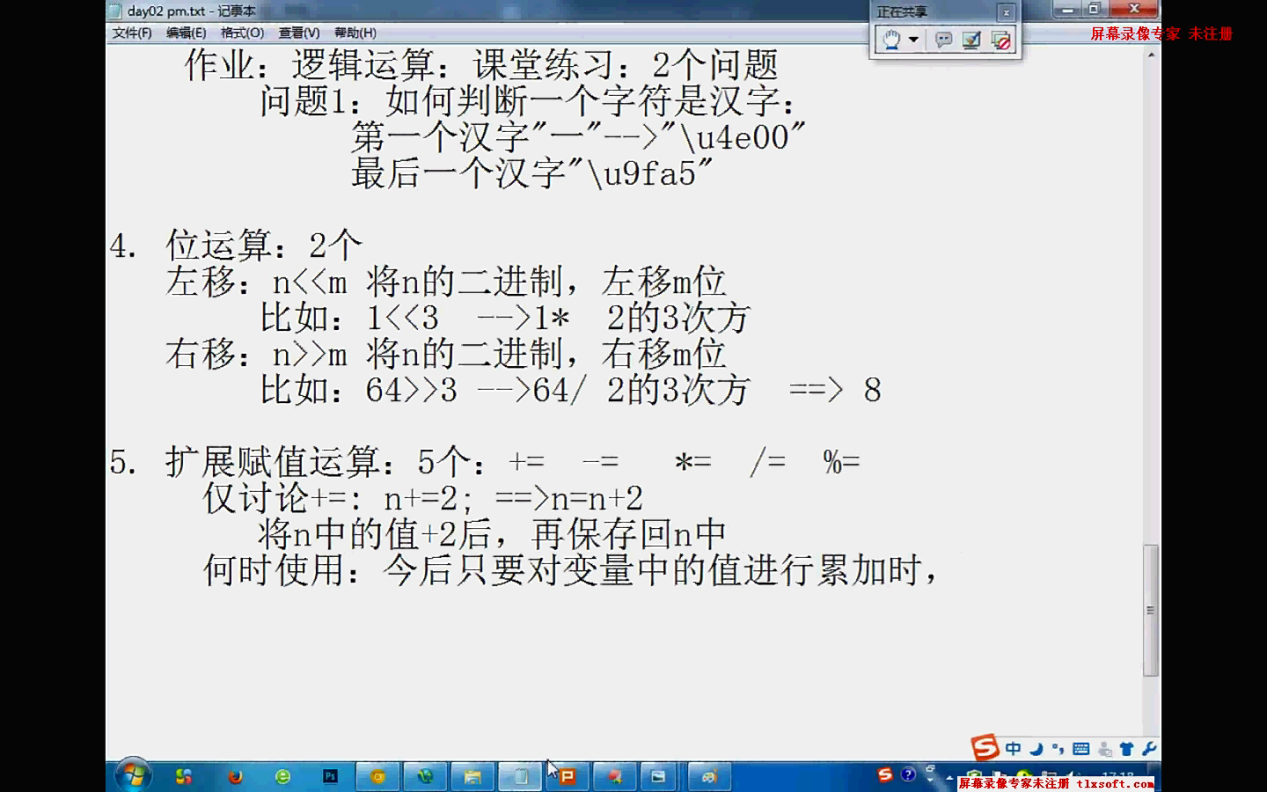
3.undefined和null进行比较

undefined == null //true，它们本质上是一样的。

===（全等）：要求数据类型相同，再比较值是否相等。（不希望隐式转换的时候用）

何时使用：只要不希望关系运算中自动隐式转换时，不带自动隐式转换。

位运算：



函数：函数就是封装一个专门任务的步骤清单的代码段，然后给它起一个函数名。

何时使用：只要定义一个专门的任务，都要分装一个函数。

1.使用函数：声明 定义 调用

一切的声明都用function来声明。

function 函数名（[参数变量列表]）{

函数体：包含任务的步骤

return 返回值

}

如何调用： var返回值 = 函数名（[参数值列表]）;//调用时才能执行

反复调用，就会反复执行相同操作。

强调：一般值的的个数和顺序，应与声明时的参数列表一致。

返回值：方法调用后，返回的执行结果。——1个数据。

return单独使用，表示退出函数执行的意思。

函数仅负责返回值，不负责保存返回值。

变量作用域：一个变量的可用范围

js中有两种作用域：1.全局作用域；一个变量可以在程序的任何位置被访问。

2.函数作用域：一个变量仅在函数调用时，内部被访问。

js中两种变量：

全局变量：定义在全局作用域中的变量。

2种。1.直接在任何函数外声明的变量。

全局对象都属于window——全局对象。

2.无论在任何位置，为从未声明过的变量赋值时。自动在全局创建同名全局变量。所有无主的都是全局变量，都是window的变量。

window.m = 100 ;

console.log(m);

局部变量：定义在函数作用域中的变量。

2种：1.在函数内部声明的变量。

2.参数变量天生就是局部变量。

为每个函数对象都分配一个地址：比如buy（）0x9091

函数其实是引用类型的对象。函数名其实是指向函数对象的变量。

在调用时，会创建局部变量，就不会用全局的。如果局部没有，才会用全局的。

调用结束后，释放所用局部变量。

有局部变量，就不会有全局变量。

分支结构：

1.函数中的声明提前：在程序执行前或者函数被调用前，将var声明的变量和function声明的函数提前到当前作用域的顶部集中创建。

function fun1( ){

console.log(1);

}

fun1( );//输出为2

function fun1( ){

console.log(2);

}

强调：仅声明提前，赋值留在原地。

var a = 100;

function fun(){

console.log(a);//该变量为全局变量a=100;

a=90;//这里的a没有声明变量，所以也是全局变量。

console.log(a);//输出90；

}

fun( );

console.log(a);//90

只有函数对象才能指向对象名（）

foo（）；//2

var foo = 100;

console.log( foo( ) )//报错，因为foo里面的内存已经不在指向地址了，而是指向100；

2.按值传递：在js中无论变量间赋值或使用变量传递参数时，都是将变量中的值，复制一个副本给对方。即形参不改变实参。

全局函数：ECMAScript中预定义了一组全局函数才是全局变量，可供用户直接使用。

执行环境及作用域：执行环境定义了变量或函数有权访问的其他数据。

全局执行环境是最外层的一个执行环境。全局执行环境被认为是window对象 。因此所有全局变量都是作为window对象的属性和方法创建的。某个执行环境中的所有代码执行完毕后，该环境被释放。全局执行环境直到浏览器被关闭时，全局变量才能被销毁。

当代码在一个环境中执行时，会创建变量对象的一个作用域链。作用域链的用途是保证对执行环境有访问的所有变量和函数的有序访问。

在 if(){}和for循环中没有块级作用域的概念。和Java恰恰相反。例如：

for(var i=0;i<10;i++){

doSome(i);

}

alert(i);//10

window.parseInt(str)/parseFloat(str)

window.isNaN(num)；

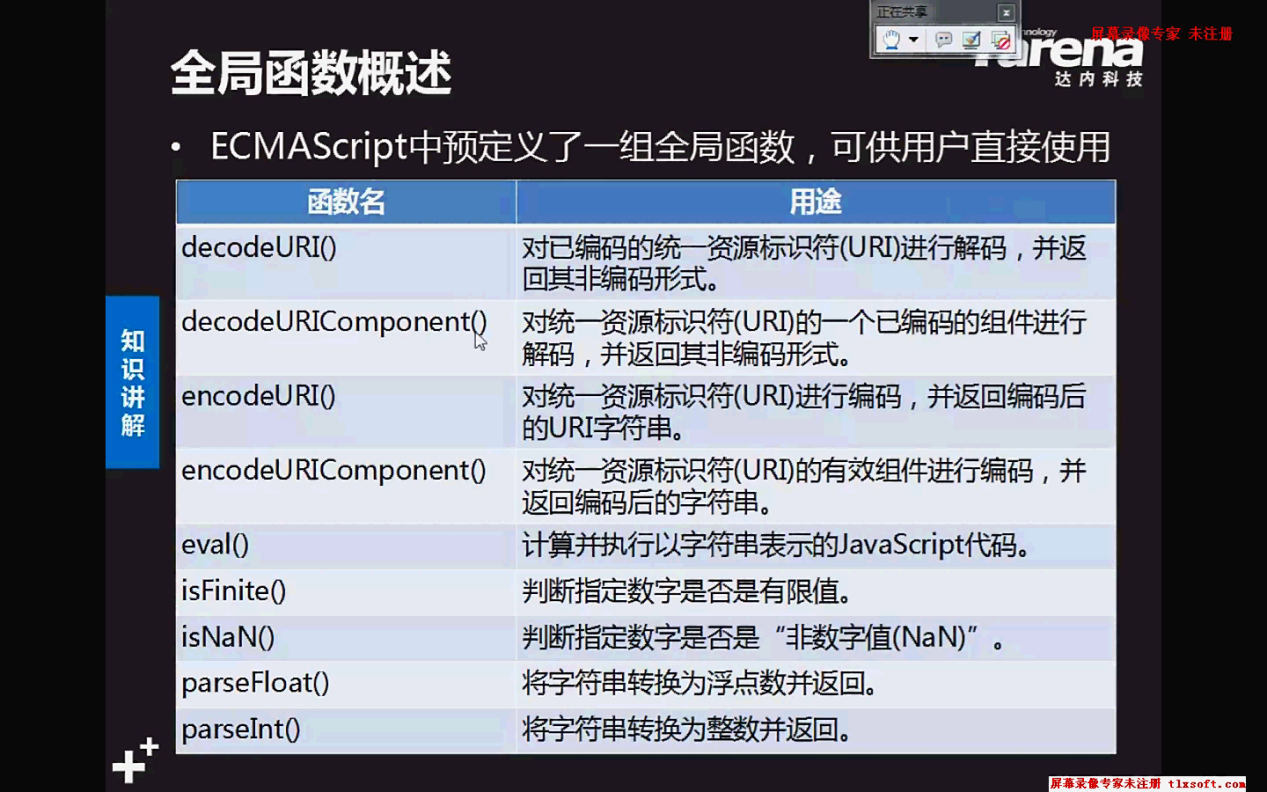
eval（）；//计算里面字符串的函数。

isFinite（）；//判断一个是否为有限数。

BOM里面的函数：

alert（）；

prompt（）；



编码：将url中的非法字符，改为合法字符表示。

解码：将url中的非法字符编码后的内容，恢复成原文。

/s?word = %E5%BC%AO 张 在utf-8中占三个字节，Unicode格式编码的一种，url中不予许出现多字节字符。解决：将url中的多字节变成单字节。

url中不允许参数值中出现保留字符。比如：/解决非汉字的Unicode编码用encodeURIComponent( );

input = encodeURIComponent（）；//将代码转成Unicode格式的编码。

eval（）：计算字符串格式的表达式的值，或者执行字符串格式的js语句。类似一个小的引擎。可以执行js语句。

isFinite（4/0）；判断除数书否为0；

在需求中找名词。该名词就要别处理。

数组：一组连续的变量组成的集合，统一的一个名字。

它可以批量管理多个数据。

如何使用： 创建 赋值 取值

创建：4种：

1.var 变量名 = [ ] ; -->创建一个空数组对象。

2.var 变量名 = [ 值1，值2,. . .]-->创建数组的同时，为数组指定初始元素。（数组中每个值）

var week = [‘一’,’二 ’,’三’,’四 ’,’五 ’];

数组是引用类型的对象。变量中保存了数组对象的地址值。

3.var arr = new Array( );-->创建一个空数组对象。

Array：js中的数组类型。

4.var arr = new Array(n);//创建n个元素的数组。

new Array(值1，值2);

找到数组：使用变量等效于直接使用变量引用的数组对象。

1. 赋值：

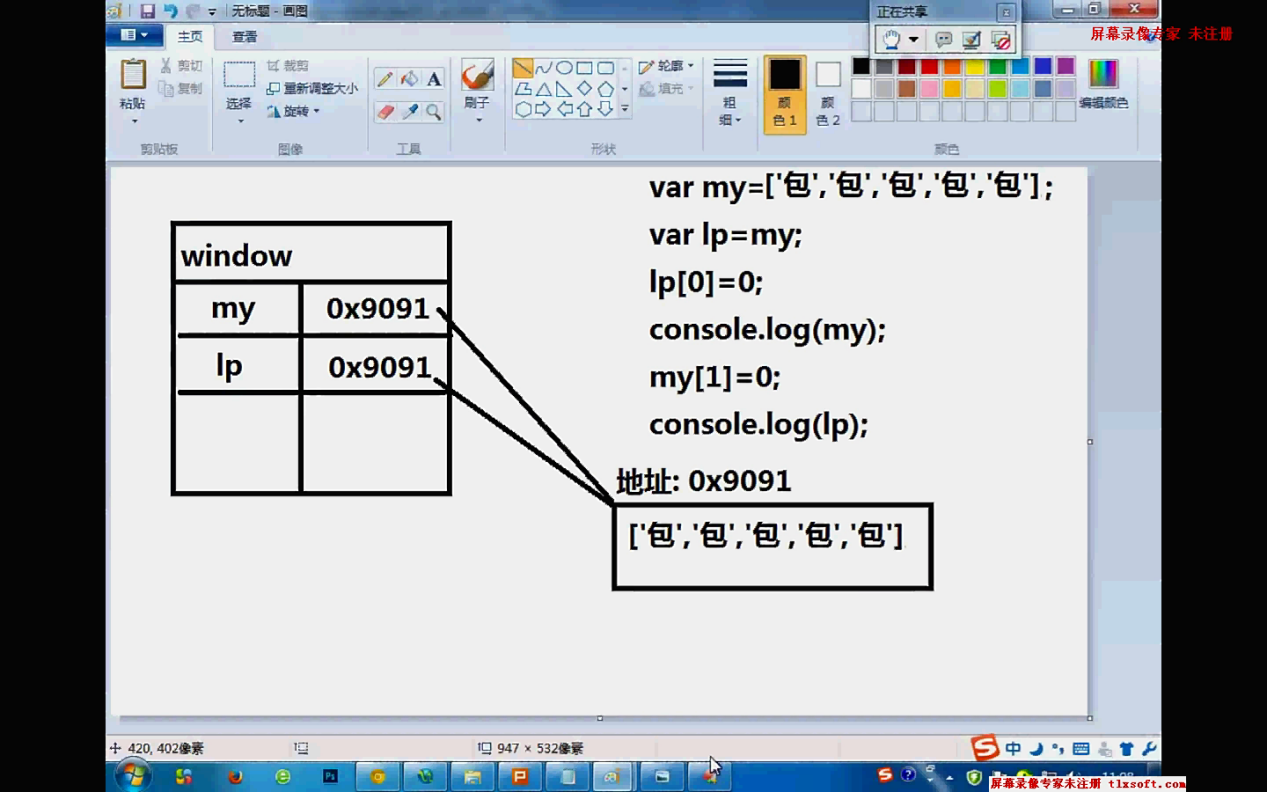
他两所指向的是同一个对象，所以当一个对象里面的内容发生改变时，另外一个指向它的对象里面的内容也发送改变。

数值变量[ i ] = 新值；

3.取值：数组变量[ i ]

undefined:所有没有值的变量的默认值，自动赋值。

null:主动释放一个变量引用的对象。



释放：my = null;//释放了引用的对象。

lp=null;//必须把两个都释放掉，该对象才会被回收。

当使用完一个较大的对象时，主动释放对象。

垃圾回收器：专门释放对象内存的一个程序，在底层，后台，伴随当前程序同时运行，会定时的回收，当没有对象引用时，该对象被回收。

\*js数组下标越界不会出错。想为一个不存在的位置赋值，不会出错，会自动创建指定下标的新元素。从不存在的位置取值，也是不会出错的。

将输入语句和循环条件同时被循环；

while（input=prompt(“输入员工姓名”)！=“exit”）{

emps[i++] = input;

}

console.log(emps);

数组是对象，封装了一组数据，并提供了对数据的操作方法。

3.关联数组：可以自己定义下标名称的数组。（类似于Map）

[ “范冰冰”，91, 65, 95]；

sname shx 语文 eng

如何创建关联数组：var fbb = [ ];

fbb[ “sname”] =”范冰冰”;

fbb[“shx”]=91;

fbb[“语文”]=65；

访问关联数组中的元素：fbb[“sname”];

关联数组中的.length属性失效。下标是不能重复的。里面的key不允许重复。

优势：利用Hash算法，精确定位某个下标的位置，不用遍历。查找速度非常快。

遍历：

for( var key in fbb){

console.log(key+“： ”+fbb[ key ] );

}

数组API：

var str = arr.toString( );-->自动，逗号分隔开

var str = arr.join(“”)//

字符串拼接时效率提高的方法：将要拼接的字符串先放到数组中：

var arr = [“a”];

arr[arr.length] =”b ”;

arr[arr.length] =”c ”;//从而实现字符串的拼接。

给数组拼接的时候，不会新建数组，只会扩容，不会造成内存的浪费。

arr.join(“连接符”);//再把字符串中的逗号用“”代替。

固定套路：1.将字符拼接为单词。

在遇到要拼接字符串的时候，先把字符串装到数组中，然后进行拼接，最后用arr.join(“”);变成字符串。

var str = arr.toString( )-->自动拼接，逗号隔开，把数组变成字符串。

2.将单词拼接为句子。

1. 拼接和截取：原对象保持不变，返回新对象。

var newArr = arr.concat(新值1，另外一个数组，新值2);

var subArr = arr.slice(starti [, endi+1]);//字符串的截取。第二个参数可省略。

如果省略，从start一直取到结尾。两个参数都可以取负值。

var ar = arr.slice(14);//截取

console.log(ar);//得到一个新的数组。

取一个数组的后4位的时候。

var a = [“1”,”2 ”,”3 ”];

a.splice(1,1);//从下标为1的地方开始删除，并且删1项。

a.splice(0,0,”www”);//给下标为0的地方插入一项www。

a.splice(1,1”sd”,”gg”);//第一个参数，起始的位置，第二个参数（删除的项数），第三个参数（插入任意数量的项）

var arr = arr.splice(2,2，“x”，“x”);//替换,从下标为2开始替换2个数为x。

arr.reverse( );//原样颠倒位置。

数组排序：arr.sort( );//本来就是按字符串排的。

function compareNum(a,b){

return a-b;

}

想要进行数字比较，就要arr.sort(compareNum);

\*\*\*问题：默认将所有元素转为字符串进行升序排列。

因此不能正常进行排序，所以得加上比较器函数。专门比较任意两数大小的函数。

有两个特点：2个参数，分别表示要比较的任意两值。无论比较逻辑是什么，都要返回一个数字。

函数对象作为参数。将函数作为对象传递。

函数名后加（）表示立即执行，如果不加（），将函数作为对象传递时，仅使用函数名就行。

js中函数本质是一个对象。

栈和队列：js中没有真正的栈和队列，一切都是用数组对象模拟的。

栈：1.末尾出入栈（先入后出）用的比较多

arr.push();//入栈

arr.pop();//出栈,出来的时最后进入的的

1. 开头出入栈：每次入栈新元素时，已入栈元素的位置都会向后顺移。

入栈： arr.unshift( );

出栈： arr.shift( );

队列：只允许从末尾进入数组，必须从开头出数组。

结尾入队列：arr.push( );

开头出队列：var first = arr.shift( );

1. 二维数组：数组中的元素，又引用了另一个子数组对象。

只要保存横行竖列的二维数据。

保存上下级关系的数据时。

var arr = [ ];

arr[0] = [“石家庄”,”保定 ”,”唐山 ”];

arr[1]=[“德州”,”济南”,”烟台 ”];

2.创建数组时，同时初始化子数组。

var data = [

[0,2,2,4],

[2,6,9]

}

强调：二维数组中，行下标不能越界。

外层循环遍历行，内层循环遍历列。

1. String：字符串，多个字符组成的一个集合（数组）

凡是数组中，不修改原来对象的API，字符串都能用。

比如：length属性，字符个数都可以用[ i ]进行访问每个字符。只读的，不能修改

arr.slice( );//截取，indexof

凡是数组对象中的直接修改对象的API，字符串都不能用。

比如：push,sort,splice

内置对象：已经定义好的11个对象。

ES标准中，有浏览器已经实现的对象。

3个原始类型：String，Number，Boolean

Array，Date，Math，RegExp，

Error

Function（函数的），Object（对象的）

Global（全局，在浏览器中为window）。

包装类型：专门封装原始类型数据，提供对原始类型数据的操作方法。

console.log(n.toString(2));//将数据转为二进制的格式

console.log(new Number(n).toString( 2 ));

转义字符：当字符串的内容和程序的特殊符号冲突时，或字符串中包含功能字符时。“ \ ”。

str.toUpperCase( );//全部转大写

str.toLowerCase( );//全部转小写

str.charAt(i);//获取下标为i 的字符。

str.charCodeAt(i);-->返回字符的Unicode号。

str.indexOf(“who”,index+1);

var subStr = str.slice(starti[,endi+1]);

1.pid.slice(6,-5+1);

2.var subStr = str.substring(starti[,endi+1]);//不支持负数参数

3.var subStr = str.substr( starti , 取几个);

从i开始，取5个字符-->unicode

substr的start可以接受负数，表示从字符串尾部开始，这点和slice相同，但是substr的长度小于1时，返回空字符串。

一般直到位置用slice（支持负数，substring不支持负数），如果知道长度用substr。

var char = String.fromCharCode(unicode);

正则表达式：

var arr = str.split(“分隔符”);

arr中以数组形式保存每段子字符串。

今后只要分段处理字符串时，都要先分隔，再遍历。

固定套路：将字符串切割为字符数组：Hello

var chars = str.split(“ ”);

concat(str1,str2,str3);//用于拼接两个或多个字符。

var s1 =‘AA’;

var s2 = s1.concat(‘BB’,’CC’,55);

console.log(s2);//AABBCC55

手机号的正则表达式：(\+86)?\s\*1[34578]\d{9}

var arr = reg.exec(str);//返回一个数组，要交g,其中：

arr[0]保存了当前关键字的内容

arr.index中保存了当前关键字的位置

reg.lastIndex：保存下次开始查找的位置 index+kword.length

匹配中间的某一段用String对象的方法比如：replace，search，match，判断整体是否满足正则表达式时用test（）。

“字符串”.replace/search/match(regExp);

regExp.test(“字符串”);

JS中没有继承的概念，但是它有原型的概念，每个对象都有一个property属性，该属性用来指定该对象的原型对象，该对象的来源，类似于继承。它的父对象叫它的原型对象。

封装：JS中有不严格的封装，对象中分装的属性和方法是可以任意被访问、修改的。

函数只有调用时才执行，反复调用，会反复执行相同的操作。

在调用方法前会先按值传递，先复制一份值。在局部的时候会使用传过来的。

函数仅负责返回值，但是不负责保存返回值。如何获得返回值：var 变量 = 函数名（）；

变量作用域：一个变量的可用范围，1.全局作用域：一个变量可以在程序的任意位置被访问。

1. 函数作用域：一个变量仅能在函数调用时，在内部被访问。

局部变量和参数变量天生都是局部变量。

ECMAScript中预定义了一组全局函数，可供用户直接使用。

在JavaScript中没有块级作用域的概念，出了函数，就是全局。

do{

var input = prompt(“请输入”);

switch（input）{

........

}

}while（input ！=0）

console.log(input);//出了函数就是全局，这里没有函数，所以该变量就是全局的。分支或者循环中声明的变量，出了结构照样使用。

1. 模式匹配：按照规则查找或替换字符串中的子内容。

查找：search只能从开始位置找第一个匹配的关键字。

var i = str.search（/kword/i）;//要在str中找 kword;忽略大小写

何时使用：仅判断有没有关键字时。返回的是匹配字符的下标位置。

indexOf( )不支持模式匹配，当找不到是返回-1；

1. 获得所有关键字的内容

var arr = str.match(/kword/ig);ig为忽略大小写的全局的

把所有匹配的内容获得一个数组。只能返回内容，无法返回关键字的位置。arr中保存了所有关键字的内容。

只有在（/kword/i）不加g的时候，会得到一个index，表示的是第一个满足reg的字符串的位置。

arr.length表示匹配的关键字的长度。

如果没有找到返回null。都得处理null和undefined。

match方法只能获得第一个以后关键字的位置。所以，它无法获得关键字的下标位置。

即能返回内容又能返回位置的函数exec( )。

替换：var newStr = str.replace(模式\\ig,”替换内容”);//得到一个新内容

既能返回内容，又能返回位置：

只要把内容变成\ \ig,检查正则表达式的。

\d==>[0-9]

\w==>[a-zA-Z\_]

\s==>[空字符]：代表空字符，比如空格\n\t

\D,\W,\S-->除了\d,\w,\s以外的、[^0-9]除了数字以外的。

量词：{ 6,8 }//6-8位。

？==》0次或者1次。

\* ==>{ 0, }

+ ==>至少一次。

(\+86 | 0086 )?\s\*1[34578]\d{9};//手机号的正则

^ 匹配字符串的开头 、/^a/

$ 匹配字符串的结尾 /a$/

String中支持正则的API：split、search、match、replace

正则表达式对象：

var reg =/正则表达式/ig; 如果正则在运行时不会改变，写死的

var reg = new RegExp( “正则表达式”，“ig”);//创建正则表达式对象，

var str1 =”\\d{6}”; str1

var str2 = “\\d{15}”都要改双斜线。

var bool = reg.test(str); /^[0-9]/

在写正则表达式的时候，都要在表达式前加^,后加$

var reg = /^\d{6}$/;//必须完整匹配

1.不全由小写字母或者数字组成，可能包含大写或者符号。

（？！[a-z 0-9]\*$）

？开始预判 ！不全有 \*可能没有 $结束符

1. 不全由大写字母或者小写字母组成，可能包含数字或符号。

(？！[a-zA-Z] \* $)

匹配一个6位密码，只能是字母，数字的组合

至少包含一位大写字母，至少包含一个数字。按顺序来匹配，如果前面不满足就不看后面。屏蔽了符号。(?![A-Z0-9]\*$)//必须包含一个小写

（?![a-z0-9]\*$）(?![a-zA-Z]\*$)[a-zA-Z0-9]{6};

reg.exec(str)-->即获得内容，又获得位置。

回去一段HTML文本中所有超连接的href属性的值。得到了所有url地址。然后再记下url。查找用户输入中的敏感词。

<a\s+(.\*)href\s\*=\s\*[‘ ‘’][^’”]\*[‘ ”]：获取页面的所有href

当if(!arr.indexof)为真时，说明arr有indexof方法

[^’”]：除了单双引号以外的所有字符。

正则：贪婪模式与懒惰模式。

默认为贪婪模式：默认总是匹配最长的符合条件的字符串。直接从最后往前找。（.\*）或者（.+）多少个都行，引起的贪婪模式。

懒惰模式：仅匹配最短的符合条件的字符串。贪婪转为懒惰 （）

程序中，本次查找后：RegExp.$n:取本次匹配结果中第n个（）匹配的子内容。

定义函数去掉字符串前空字符，去掉字符串后空字符以及同时去掉字符串前后空字符。

当要匹配哪个就用$n直接匹配到那个。

var birth=19931015;

var reg = /(\d{4})(\d{2})(\d{2})/;

birth = birth.replace(reg，“$1年$2月$3日”);

var reg = /<li>\s\*</li>/;

var arr = str.split(reg);

1. Math:专门执行数学计算的对象。封装了数学计算中常用的常量。

1）取整：3种方法：上取整 下取整 四舍五入

ceil( ) floor( ) round( )

Math.round(0.6) = Math.floor(0.6+0.5)=1;

toFixed属于Number对象的四舍五入。按任意小数位数

round Math对象 只能取整

var n = 23.56;

n.fixed( 1 )=”23.6 ”;//用于显示的时候很方便。

1. 乘方/开方：

乘方：Math.pow(n,m)：计算n的m次方

开方：Math.sqrt(n)：计算n的平方根。

1. 获得最大值/最下值：Math.max(12.3 , 36 , 69);

固定套路：Math.max.apply(Math , arr);-->

相当于Math.max(arr[0] ,. . .)//变相实现获取数组中最大值

直接得到一个数组中的最大数：var arr = [5,6,9,23];

Math.max.apply(Math，arr);

parseInt(Math.random( )\*101+60);//60-100之间。

1. Date：封装了一个时间点，提供了对时间和日期的操作API。

如何创建一个Date对象：4中

1.var now = new Date( );//创建一个新日期对象的同时，保存了当前时间点的毫秒。自己的客户端时间。

自定义时间对象：

2.var time1= new Date( “xxxx/xx/xx [xx:xx:xx]”);

3.var time2= new Date(年，月-1，日[,时,分，秒]);

日期之间可以相减。

time1 - time2：两日期可以直接相减，可以省略time1.getTime( )，t2.getTime( );

date.getTime( )：获得日期对象中的毫秒数。

每个分量都有一对get/set方法

getDate（）获得星期，没有setDate（）

1. 返回值：月中的日从1开始到31结束。

Date计算： 2种

1. 两日期对象可直接相减，结果毫秒差。
2. 对任意分量做加减：先get出来，计算，再set回去。

复制日期对象的办法：date.setxxx(date.getxxx()+-n);

var date1 = new Date( );

date1.setXXX

enddate.toLocaleDateString( );//抛开时间部分只获得日期部分，比如得到：2015/10/15

1. 异常处理

function对象里面有一个闭包。

异常处理：错误就是导致程序无法继续执行的异常状态。

js中一旦发生错误就会自动创建一个Error类型的对象。

js中有几种错误类型（6种）

SyntaxError：语法错误

ReferenceError：引用错误，找不到变量或对象。

TypeError：类型错误，错误的使用了对象中的方法时。比如：console.log();

RangeError：范围错误，参数超范围。

EvalError：调用eval 函数

错误处理：在程序发生错误时，保证程序不退出或者正常退出。

如何处理：try{}catch{}块

try{

正常代码段中，可能出错的代码块

}catch（err）{//err表示错误，catch块仅在发生错误时才执行

//一旦发生错误err中就会自动存入Error对象

1.记录/显示错误信息

2.继续向调用者抛出异常

}[finally{

无论对错一定都会执行的

释放资源

}]

何时需要定义错误处理？有可能出错，都要包在tryCatch中。

一旦发生错误，会自动创建一个对象。

tryCatch用法：解决浏览器兼容性问题。

try{

new XMLHttpRequest（）；

document.write(“”)

}catch（err）{

}

XMLHttpRequest这个类型是jQuery中的内容 。IE8不支持。

抛出自定义异常：throw new Error(“自定义错误消息”);

比如：如果程序员甲定义了一个方法，被程序员乙调用，如果乙错误的使用甲的方法，甲的方法中就要抛出自定义错误消息。

乙在调用时，就要将甲的方法包在tryCatch块中。

Function对象是所有对象的。js中一切都是对象。函数也是对象，函数名其实是引用函数定义对象的变量。

1. arguments对象,是一个类数组对象，[i]访问每个元素，.length是元素的个数

重载：程序中可定义多个函数名相同，不同参数列表的函数。当两个函数名相同时，程序根据传入的参数个数，自动判断选择哪个函数执行。

js中不支持重载，但可用arguments对象模拟重载效果。

arguments对象：函数对象内，自动创建的专门接收所有参数值的类数组对象。

但是可以借助

函数对象的本质：1.

1. Function对象的本质
2. 匿名函数
3. 闭包

任何数+undefined = NaN

创建函数对象的三种方式：

1. 声明方式：function 函数名（参数）{

函数体；

return 返回值

}//函数名和函数定义都被提前。在调用前后定义都行；

1. 函数直接量： var 函数名=function(){

函数体；

return 返回值；

}//在声明时不会被提前，

函数定义留在本地。必须定义在调用前。

1. 使用new创建函数类型对象：以动态的方式创建函数类型对象

var 函数名 = new Function(“a”,”b ”,”c ”，“函数体”);//在括

var compare = new Function(“a”,“ b ”, ”return a-b“”);

号里面必须字符串，

在js中不分单双引号。

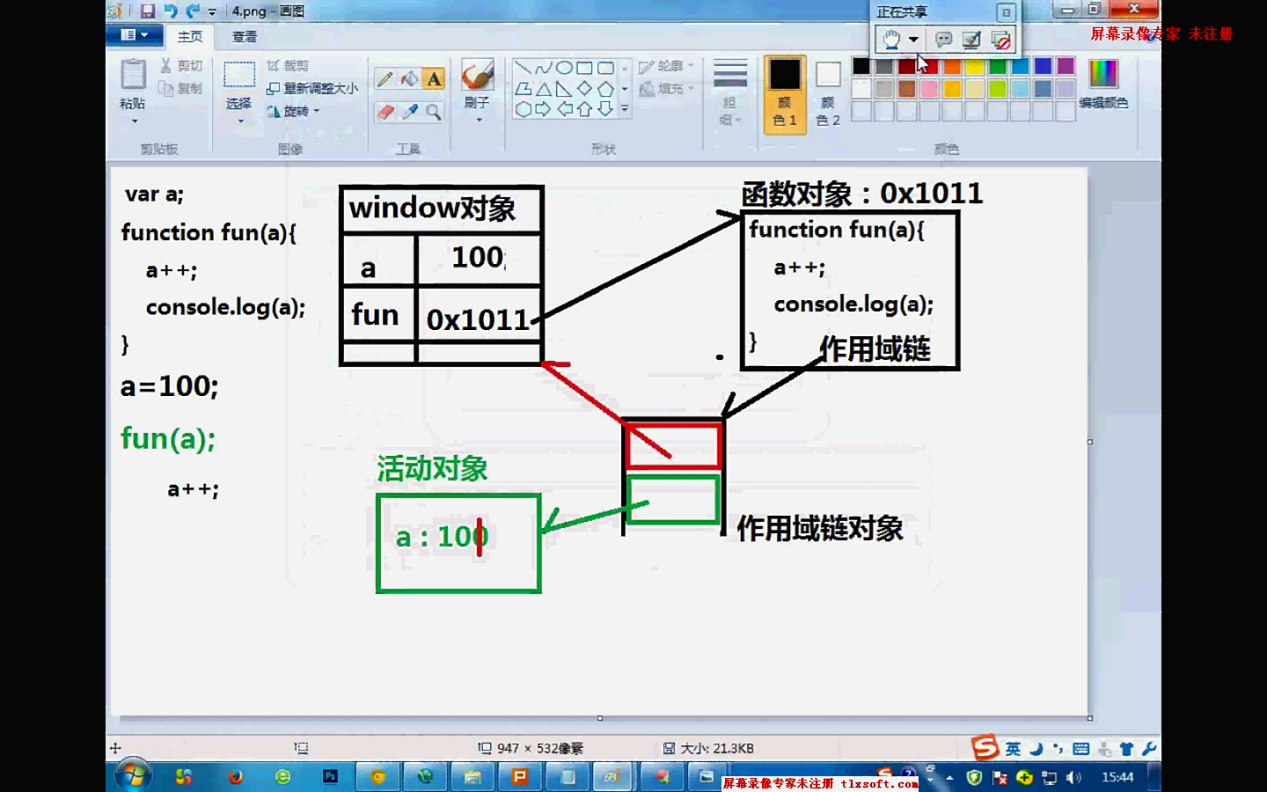
内存中的函数对象：

在创建函数对象时： 同时创建2个对象：

1.函数对象，即函数的定义

2.作用域链对象：保存了对象可用的变量位置，

默认第一项指向window，所以在任何位置都能访问到该函数



匿名函数：定义时，不指定函数名的函数。

在函数的调用时，又会创建一个新的对象。在作用域链对象中追加指向活动对象（专门保存局部对象的，等于参数）的引用。

在调用后：默认仅释放活动对象。

作用域链中活动对象的引用出栈，活动对象因无人引用而释放。

何时使用：2大用途：

1. 匿名函数会自调，不用任何对象调用就能执行：定义完，立刻执行。执行完就释放。

执行完立刻释放。但是想要第二次调用时，找不到了。  
比如：把资源加载一次时，用匿名函数。何时使用：确定函数只执行一次时。

如何自调：（function（参数）{

函数体；

}）（参数值）；

\*\*\*自调，定义在哪，就在哪执行不提前。

1. 匿名函数回调：先将函数作为对象传递给另一函数，然后由另一个函数自主决定在需要时调用。让别的函数去调用。

//在sort（）之前没有比较器对象，在调用完后，就释放该对象。

节省了内存空间。

var arr = [1,23,63,17];

arr.sort(function(a,b){return a-b;});

在调用给定的函数叫做回调函数。只要将一个函数对象传递给其他方法做参数调用时，用回调函数。这样在执行调用者前不会创建该对象，在执行完后，更没有改对象，这样会很省内存空间。

1. 闭包的三大特点：

问题：

局部变量：无法共享，不能长久保存。

全局变量的缺陷：容易全局污染，随处可以修改。

既可以共享，长久保存，又不会全局污染。只能用闭包。

该函数能使用函数外定义的变量。

function getCounter（）{

var n = 1;

var counter = function( ){return n++};//局部函数，要想局部函数的数据被外界得到，只有return该局部函数。

retrun counter;

}

解决3步：1.定义外部函数，封装被保存的局部变量。

2.定义内层函数，执行对外层函数局部变量的操作

3.外层函数返回内层函数的对象。并且外层函数被调用，结果被保存在一个全局变量中。

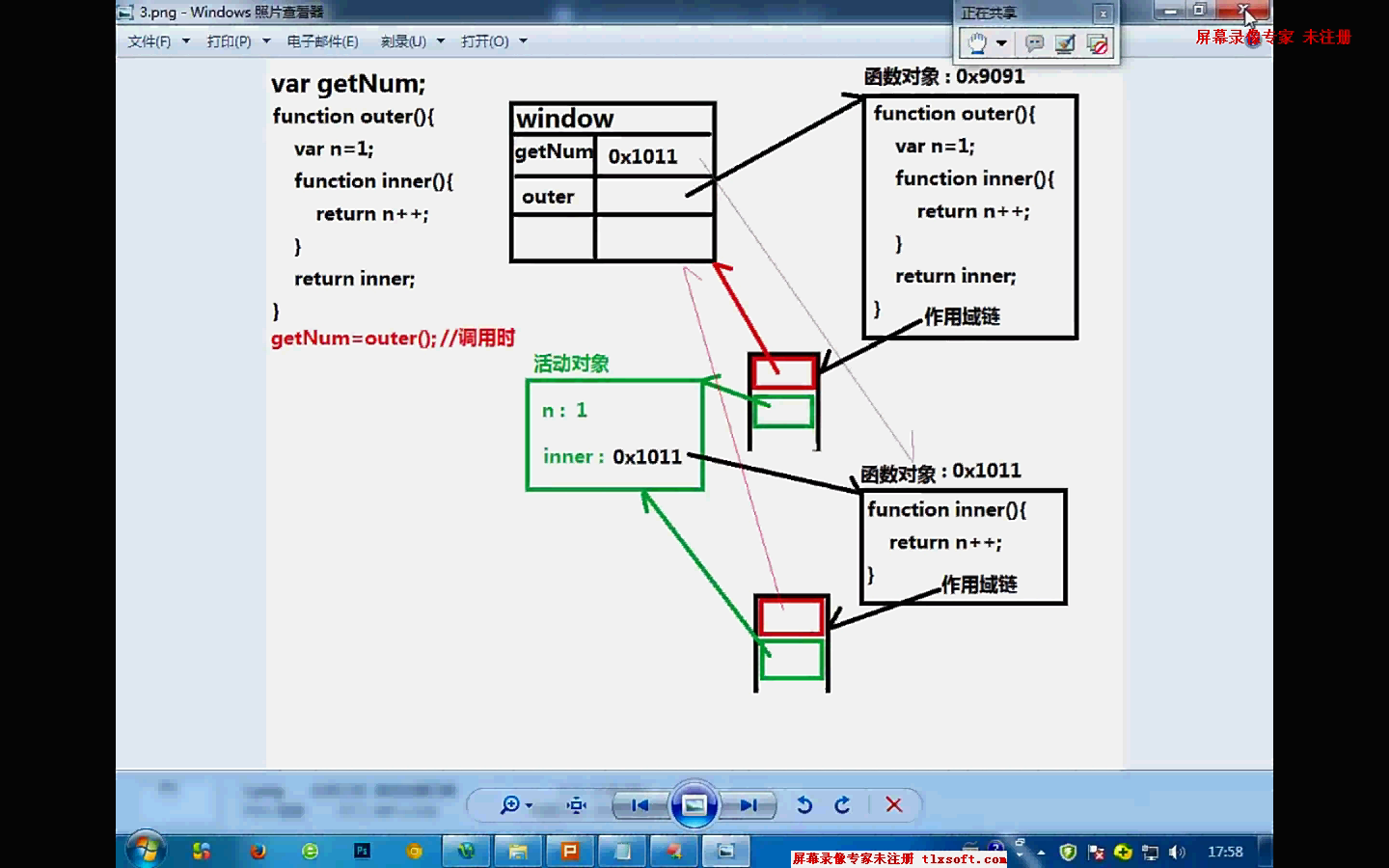
var myCounter = getCounter();

何时使用闭包：即反复使用局部变量，又避免全局污染。

外层函数保护变量，内存函数操作变量，外层函数返回内部函数对象。最后调用外层函数被全局对象接住。

\*\*\*何时使用：反复使用局部变量，又要避免全局污染就要用闭包。

在调用一次函数就创建一个新的对象。只用在调用时，定义的函数对象才会被执行。



作用域链对象里面类似一个栈并指向全局的位置。

1. \*\*\*外层函数被调用几次，就有几个受保护的局部变量的副本。
2. \*\*\*来自一个闭包的函数被调用几次，受保护的局部变量就变化几次。

只要见到[ ]就是数组。

只要见到{ }就是对象。

1. 面向对象：在程序中都是一个对象来描述现实中一个具体的东西。

属性：描述一个东西特点的变量；

功能：东西可以执行的操作。

什么是对象：封装多个数据的存储空间。

自定义对象 ：封装现实中一个东西的属性和功能的存储空间。

1. 创建对象的3中方式：

1）var obj = {‘属性名1’：值1，

‘属性名2’：值2,

‘功能名1’：function( ){ . . . }};

把散的功能和属性放在一起就叫做封装。

js中一切都是对象，所有对象的底层都是hash数组。js用hash数组解决了一切存储问题。

var lilei={

“sname”:”Leilei ”,

“age”:12,

“intrself”:function(){

alert(“I am lei”);

}};

相当于 var lilei={

sname:”Lielei”,

age:12,

intrSelf:function(){

alert(“I am Li Lei”);

}

}

对象在 js中就是一个数组，里面封装了对象的属性和方法。

里面所有的东西都是lilei.name在window对象中就会变成lilei[“name”];

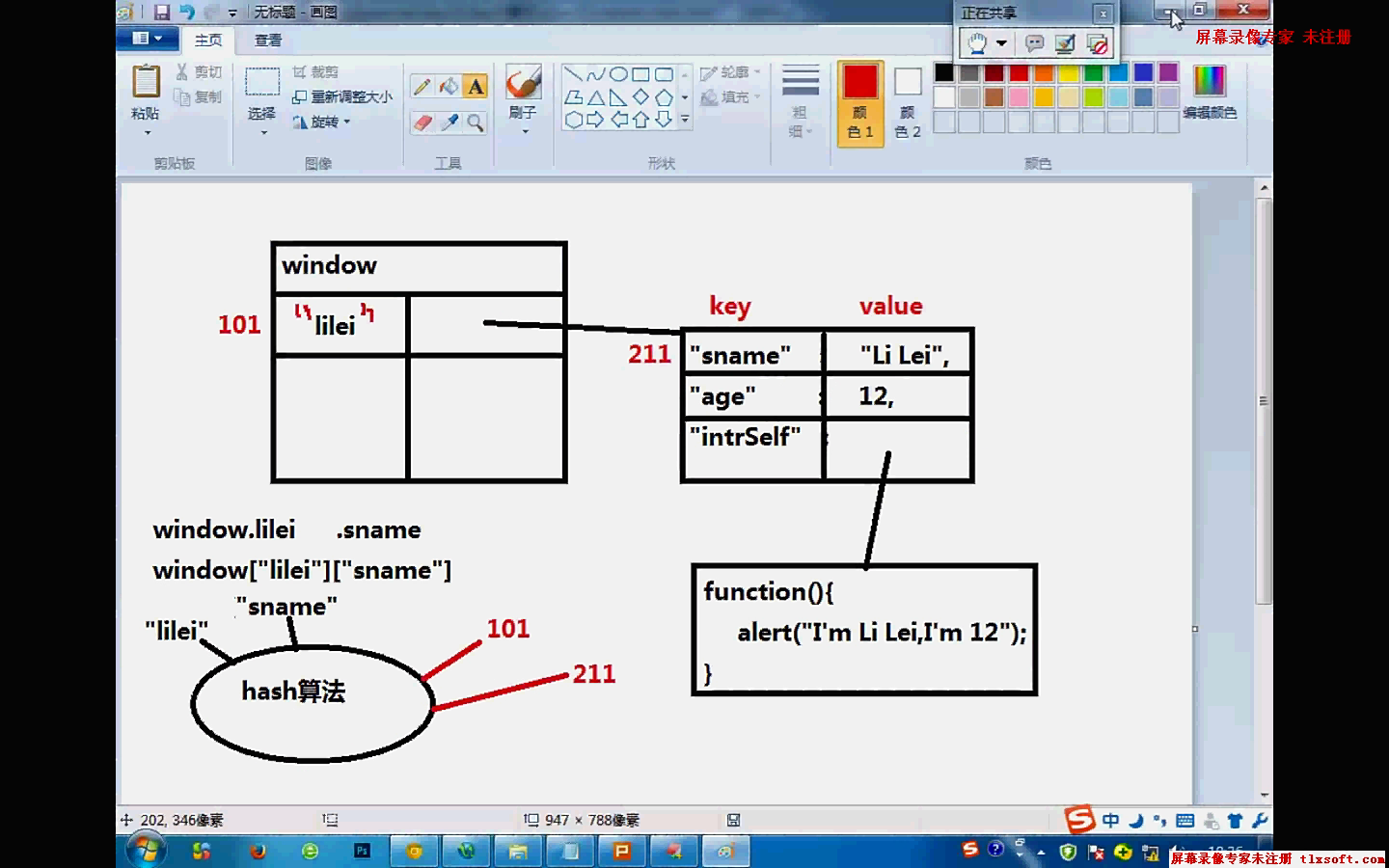
for(var key in lilei){

console.log(key:lilei[key]);//把key当做变量用了，不能用.key，相当于lilei[i]

}

自动省略属性名和方法名的引号。

属性：如何访问属性：obj.属性名 obj[“属性名”]



访问对象中不存在的属性，等效与访问数组中不存在的下标：不会出错，返回undefined。

2）利用构造函数，使用构造函数反复创建相同结构的对象。

构造函数：描述一类对象结构的特殊函数。

定义构造函数

function 构造函数名(属性参数1，. . . ){

this.属性名 = 属性参数1；//在当前正在创建的对象中

}

1. 利用构造函数创建对象

var obj = new 构造函数名（）；

new

1. 定义构造函数：

function 构造函数名（属性参数， . . .）{

this.属性名 = 属性参数；

this.方法名 = function( ){

this.属性名

}

}

var obj = new 构造函数名（属性值1）

new：1.创建一个空对象：new obj = { }；

继承：js中一切继承都是用原型对象实现的。

原型对象：每个函数对象都有一个原型对象。Student.prototype原型对象。构造函数的原型对象负责保存所有子对象共享的。

任何对象都有一个\_proto\_属性指向构造函数的。

构造函数名.prototype

Array.prototype

判断一个函数里面有没有该函数方法String.prototype.trim，当里面有该方法时，返回true。

1. 判断自有属性：obj.hasOwnProperty(“属性名”);//自己有的
2. 判断共有属性：“属性名”in obj

判断是共有的时用！obj.hasOwnProperty(“属性名”)&&“属性名”in obj;

在原型关系中包含且子对象自己没有。

delete只能删除自己有的属性而不能删除共有的属性。

原型链：由各级对象的\_proto\_逐级继承形成的关系。

function是所有函数的祖宗，object是所有对象的祖宗。

var arr = [ ];

平时获得一个对象的父对象用：Object.getPrototypeOf( arr ); == Array.prototype

使一个对象的父对象为另外一个对象：

Object.setPrototypeOf(Plane.prototype,Flyer.prototype);//Plane继承了Flyer。

jQuery的函数执行是等到页面加载完毕时。比如：

$(function(){

alert(“页面加载完毕开始执行”);

})();

三种创建全局变量的方式：

var n = 1; window.n = 1; window[“n”] = 1

delete window.n不能被删除

扩展共有属性：通过构造函数.prototype添加trim方法。

判断自有属性或者共有属性：

isPrototypeOf()方法用于判定一个prototype对象是否存在于另外一个对象的原型链中。

父对象.isPrototypeOf(子对象);

判断对象是不是数组：

用Array.isArray([]);//返回true则是数组。

if(!Array.isArray){

Array.isArray = function(obj){

return Array.prototype.isPrototypOf(obj);

}

}

如何判断某个对象是否包含指定成员。3种

1. obj.hasOwnProperty(“成员名”)；//成员名包含属性和方法，如果找到返回true。没有找到返回false。

强行给不存在的属性赋值，会自动创建同名属性。

方法：如何在方法中，访问当前对象自己。

要用this关键字：在运行时，指定正在调用方法的对象。

（.前的对象）

this的本质是window下唯一的一个指针，指向当前正在调用方法的对象。

obj.hasOwnProperty(“成员名”)如果找到，返回true，否则返回false。

如果不包含，arr.indexOf返回undefined-->false，如果包含，返回值为function-->true

if(arr.indexOf){//判断arr里面是否有indexOf成员

}

判断方法是否可以省略与undefined的全等比较，如果确定属性值一定不是null，0，“”，NaN也可省略。

如果在方法内，访问当前对象自己的属性必须加this.属性名字，不用再你的作用域链中找也不用再活动对象中找了（因为直接访问sname时，在作用域链和活动对象中是找不到的，原型链中一个指向window另一个指向活动对象），直接在当前对象中找。

\*\*\*this和定义在哪无关，仅和调用时使用的当前对象有关。

\*\*\*如果无主的调用或者赋值，默认this都是window。

[window.]a = 100;

this始终指的是正在用的对象。

/\*经典的 this案例\*/

写一个匿名方法，不用别任何对象调用，就直接能执行，这叫函数的自调。因此整个过程也没用创建任何对象。

//this-->window

var a = 2;//this-->window

function fun(){

console.log(this.a);

}//this-->window

var o = {a:3,fun:fun};//this-->o

var p = {a:4};//this-->p

o.fun();//得到：3 this-->o

用到了自己的就用this.属性名。

var obj= {“属性名”：值，

“方法名”：function( ){ this.属性名}};

o.fun();

p.fun = o.fun();

//p.fun = 0x1011

p = {a:4 , fun : 0x1011 };

返回o.fun中的值，赋给p.fun,

(0x1011)();//匿名函数

call( )：在调用方法的一瞬间更换调用方法的对象。

var fun = function(){};

Object.prototype.toString.call(fun);//在调用查看最原始的toString方法的时候，换为fun的toString方法。

return Object.prototype.toString().call(obj) == “[Object Array]”;

sessionStorage方法：sessionStorage方法针对一个session进行数据存储。由cookie完成，但是cookie不适合大量数据的存储，因为它们由每个服务器的请求来传递，这使得cookie速度很慢而且效率不高。

Object是一种类型。

原型：保存所有子对象共有属性和方法的对象。

所有元素的祖宗是Object的原型对象。Object.\_proto\_

原型链：所有父子级对象间由\_proto\_形成的多级引用关系。

当要在任何地方都能调用到$时，用：

window.$ = function(selector){

return document.querySelectorAll(selector);

}

DOM可以对网页里面的任何元素做想要的操作。增删改查元素。

DHTML（Dynamic HTML）指所用能够实现动态网页设计的统称。

DHTML = HTML + CSS + JavaScript

XHTML：严格的HTML。

XML：可扩展的标记语言，可以自定义标签。

<演员>

<姓名>范范</姓名>

<年龄>32</年龄>

</演员>

专门用于存储/传输自描述的结构化数据。做传输的时候被json（JavaScript对象表示方法）替代了。

{“姓名”：“范范”，“年龄”：32}

var str = ‘{“name”:”libin ”, ”age ”: 25}’;

var obj = JSON.parse(str);//将字符串转为json格式。

JSON.stringify(obj);//将json转为字符串形式。

alert(obj.name);

event对象 ：事件发生时自动创建封装事件的对象。

事件是指：用户在操作鼠标或键盘时，对页面的状态的改变。通过以下的时间属性来发生事件。

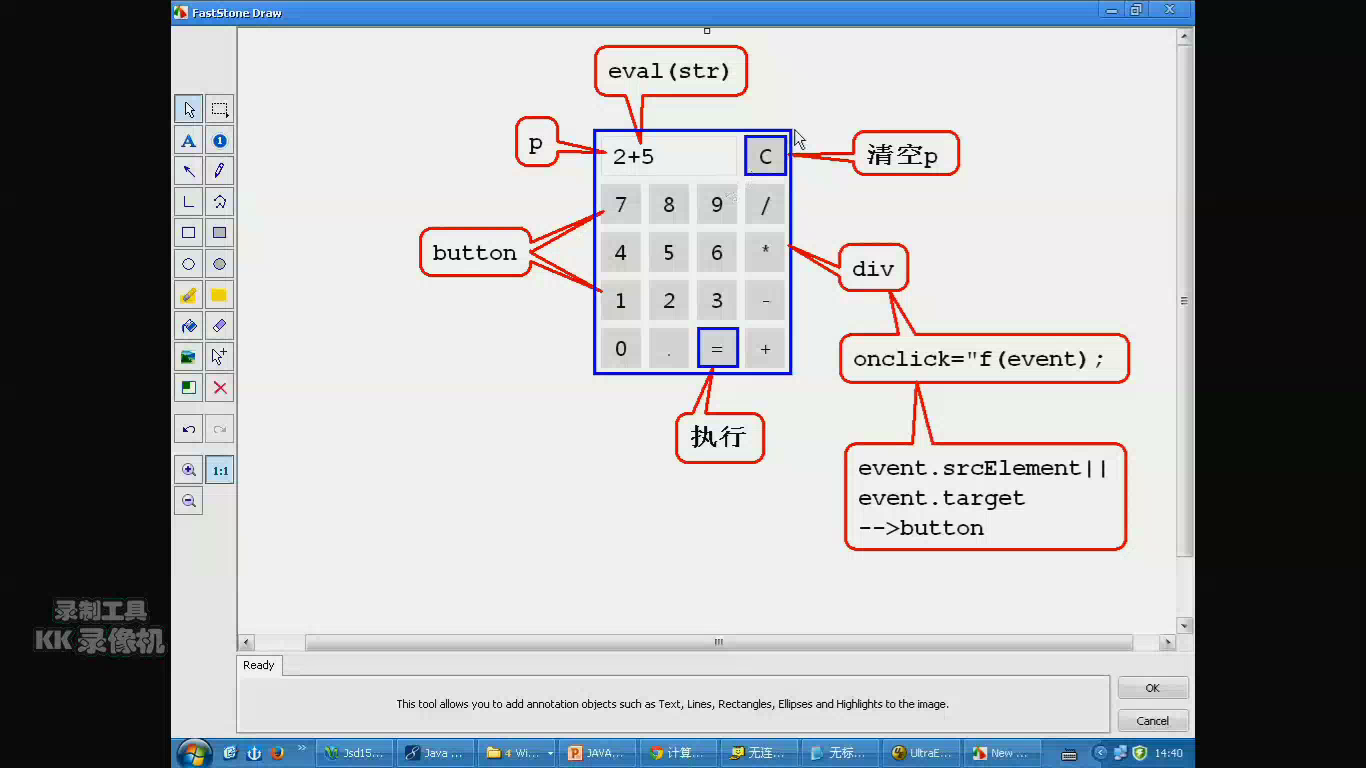
-鼠标事件：1.onclick（点击事件）2.ondblclick(双击事件)3.onmouseDown(鼠标按下事件)4.onmouseup(鼠标弹起事件) 5.onmouseover（鼠标悬停事件）6.onmouseout（鼠标悬停移开事件）

-键盘事件：1.onkeydown(键盘按下事件)2.onkeyup(键盘弹起事件)；

-状态改变事件1.onload(页面加载事件) 2.onchange(点击后页面发生变化) 3.onblur(光标切换时间) 4.onsubmit（表单拦截事件）

取消事件：onXXX = “return false;”

任何事件触发后将会产生一个event对象。它是一个浏览器对象。



只用在计算器整个div上注册一个事件，传入event，获取事件源，就能判断出是哪个元素的事件。

navigator是对服务端的设置。可以启动cookie，关闭cookie。关闭插件。

history：前进和后退两个按钮。

location：地址栏。

screen：显示器对象。拿到分辨率，宽高对象等。

BOM：浏览器对象模型（API），专门操作浏览器窗口的API。

DOM：文档对象模型，专门操作网页内容的API。可以对网页中任意对象做任何修改。DOM是一个标准。

核心DOM：操作所有结构化文档（XML，HTML）的通用API。

HTMLDOM：针对HTML文档的简化API。网页中一切都是对象。同一网页中的所有对象，在内存中父子相连，形成一棵DOM树。

DOM树：文档中的每个元素，属性，文字，注释，都被看做一个节点对象--node。

文档节点（document node）是个类型也是个对象。

每个标签是一个元素节点（element node）；是所有元素节点的祖宗。

所有文本节点（text node） 12 \* 8 0 120 - 96 =

每个属性节点（attribute node）

注释节点（comment node）

Node类型是所有节点对象的父类型。Node.prototype是所有节点的父对象。

Node类型定义了3个公共属性。

nodeType：节点类型的数值常量。nodeType==1，元素节点Element\_node。nodeType==3，拿到的是文字节点，专门用于判断获得节点类型，不是1的就不要。

nodeName：节点名称，如果是元素节点，返回标签名。判断是获得的标签名、

文本节点，返回#text经常用来判断该节点是不是你想要的。

nodeName返回的都是全大写的标签名。

nodeValue：节点的值，只对文本的内容有效。

因为一个网页里面只用一个body标签，所以在获得body的时候只用：

document.body.nodeType

childNodes：获得当前节点对象下的所有子节点类数组对象。

只有元素才有id属性，才有innerHTML属性。

得到属性节点：getAttributeNode（“href”）；

href属性对象-->Attr

元素的parentNode为元素。向上的

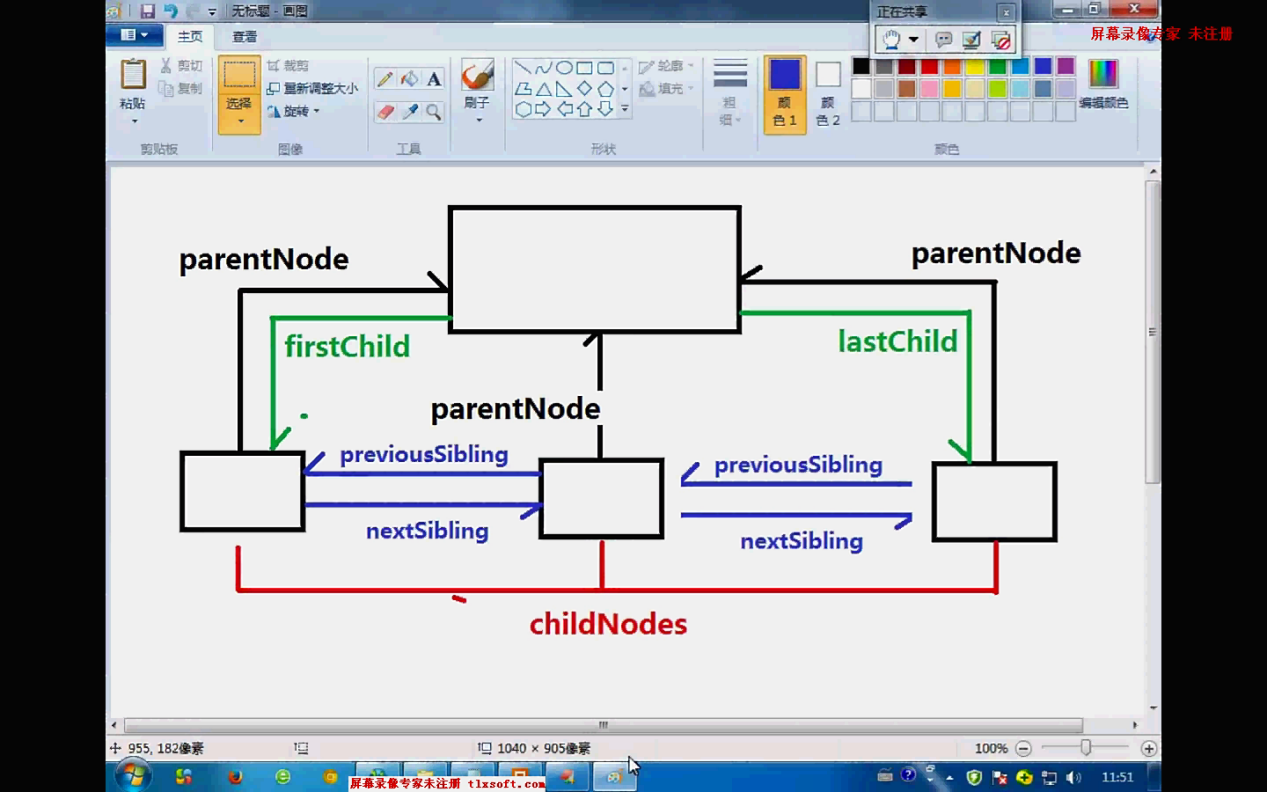
元素向下childNodes[ ]

DOM树6种关系：6个属性：

父子：parentNode childNodes firstChild lastChild

兄弟：previousSibling（前一个兄弟）

nextSibling(下一个兄弟)



找到table对象的第一个tbody对象。

table.tBodies：返回table对象中包含的所有对象。

var tbody = table.tBodies[ 0 ];

//遍历tbody下的所有子节点，如果当前子节点是

单行为无色，双行为有色：先遍历tbody下的所有子节点，tbody.childNodes[i]

if（tbody.childNodes[i].nodeName ==”TR ”）{

//如果为r%2 == 1;

if(r%2==1){

//修改当前节点的className属性为blue

tbody.childNodes[ i ].className =”blue ”;

}

r++;

}

动态结合，自己不保存任何实际数据，每使用一次都重新查找一遍。

for(var i = 0;i<parent.childNodes.length;i++){

}等于

for(var i=0,len=parent.childNodes.length;i<len;i++){

}//这样写效率高

遍历:从指定父元素开始，按照深度优先的原则遍历所有各级子节点。

1.定义一个函数，查找任意

html是网页的根节点。不是根元素。

//输出指定元素及其直接子元素

function getChildren（parent）{

console.log(/\*如果是parent是元素节点，就返回nodeName，否则就返回nodeValue\*/);

parent.nodeType==1?parent.nodeName

parent.nodeValue

//遍历所有parent的直接子节点

//当页面加载后开始执行

window.onload=function(){

getChildren（）；

}

}

event（事件）：是一个特殊的对象，指页面的元素状态改变，用户在操作鼠标或键盘时触发的动作。

5种查找的方法：

1.var elem = document.getElementById(“id值”);

2.var elems = parent.getElementsByTagName(“子标签名”);//返回的是一个对象数组。

3.var elems = document.getElementsByName(“name属性”);//是根据控件名称获取这个控件对象，因为控件名称可以相同，返回的是一个对象数组。 得到的是name属性所附的元素

4.var elems = parent.getElementsByClassName(“class属性”);

5.Selector API：JQuery的核心

elems：包含了所有属性和方法的完整对象的集合。

var elem = parent.querySelector(“任意选择器”);

var elems = parent.querySelectorAll(“任意选择器”);

window.$ = function(selector){

return document.querySelector(selector);

}

看HTML DOM的 API主要看：

1. 看继承关系。谁在谁的范围下面。
2. 看属性的位置，在父对象还是爷爷对象。

核心DOM类型，可对XML、HTML文档中的所有元素节点执行增加、删除、修改，查询操作。

HTML DOM类型，专门针对HTML文档中的HTML元素。简化操作HTML元素对象的标准。主要用来修改属性。

要先一级一级的往上找哪个方法在。

innerHTML是批量修改元素内容：

1. innerHTML：获得/设置元素开始标签和结束标签之间的HTML原文。

1）删除父元素下所有子元素：parent.innerHTML=“”;

2）批量替换父元素下所有子元素：parent.innerHTML=”<li>电影</li>”

2.textContext（IE8不认）/innerText:获得开始标签和结束标签之间的纯文本内容，不包含标签。只希望去掉标签，仅获得文本时。

元素的节点 attributes属性返回节点的属性集合。

获得元素的属性：

var value = element.getAttribute(“属性名”);

只要获得任意属性的值。

修改属性：

element.setAttribute(“属性名”，value)；//IE8不支持

只能用element.attributes[‘属性名’]=value；

var a = document.getElementById（“bdlink”）;

a.removeAttribute(‘class’);

判断元素是否包含某种属性：

element.hasAttributes(‘属性名’);//返回true或false

Property（属性）和Attribute（特性）都叫属性，

Property：对象在内存中存储的属性。

Attribute：元素对象在开始标签中定义的HTML属性和自定义属性。能够在开始标签里面的属性。

访问HTML标准属性时。二者完全一致。

当访问的是自定义属性时，不能用property直接访问。因为property的属性时存储在内存中的。

比如：<a href=”http://tmooc.cn”>

1. href-->a.getAttribute(“href”);特性-->核心DOM，都能访问到。

如果访问自定义属性，二者不通用，Attribute可以访问到，property不能访问到。

<li data-age=”29 ”>Eric</li>

li.data-age是不能访问到的。用li.getAttribute(“data-age”);是可以访问到的。

window.onload=function(){}

将元素对象挂载到指定父元素下：

追加：parent.appendChlid(newElem);

只有向已经在页面中的父元素追加新节点，才导致渲染，比如：

var a = document.creatElement(“a”);

div.appendChild(a);

<div>

<a href=”http://tmooc.cn”>go to tmooc</a>

</div>

创建、删除节点：

插入新元素：parent.insertBefor(newElem,oldElem);

删除节点：parent.removeChild(oldElem);

oldElem.parentNode.remove(oldElem);//自己把自己给删了

替换节点：parentNode.replaceChild(newElem,oldElem);

select对象：selectedIndex属性：记录的是当前选中项的下标。

select.selectedIndex

Table对象：

Form对象：

如何找到一个form对象

var form = document.forms[i/name];

如何找到form中的一个数据采集元素：

var elem = form.elements[i/name];

事件：onsubmit：在正式提交表单前自动触发。

Form对象：表示网页中的一个Form元素。

如何找到：document.form[序号 | name属性]

elem.focus();//获得焦点

elem.blur();//失去焦点

form.onsubmit=function(){

只要任意一个验证未通过，

就取消事件

1.获得event对象e;

var e = window.event || arguments[0];

if(e.preventDefault){

e.preventDefault()；//核心DOM里面的方法

}else{

e.returnValue=false；//IE8

}

}

表单提交：直接点击submit按钮

如果当前form中任意元素获得焦点，可按回车自动提交，只要自动表单提交前，都会先触发onsubmit，里面可以做验证。给form绑定onsubmit。

BOM:

window对象：

1.定时器：天生做动画用的。

2.常用BOM对象：Navigator

1)BOM：专门操作浏览器窗口的对象，窗口是相框，网页是相片。

window对象：1.充当全局对象。2.当前打开的窗口，BOM的常用对象。

window对象常用属性：一般只能读取，不能设置



defaultStatues：设置会返回窗口状态中的默认文本。

innerHeight：返回窗口的文档显示区的高度。

innerWidth：返回窗口的文档显示区的宽度。

outerHeigth：窗口的高

outerWidth：窗口的宽

pageXOffset：设置页面相对于窗口显示区的左上角X位置。

pageYOffset：文档左上角到文档显示区左上角的距离。（可以利用该属性，判断该页面停留在哪个楼层）

窗口打开和关闭的方法：

1）打开新链接：4种效果：

1.在当前窗口打开新链接，可后退（默认的情况下，就可以退回）

分别用html：<a href=”http://tmooc.cn”/\*target=”\_self”\*/></a>

\_self：指的是当前窗口本身，和window意思一致

和js方法：

2.在当前窗口打开新链接，禁止后退（比如付完款后，就不能后退了）

用js做：location表示的是当前窗口的URL对象，只能在当前窗口打开，不能在新窗口打开。

location.replace(“http://tmooc.cn”);

replace做了两件事：1.打开链接。2.把原来的URL 替换掉了。

1. 在新窗口打开新链接，可同时打开多个

<a href=”http://tmooc.cn”target=”\_blank”>在新窗口中打开，可打开多个</a>

1. 在新窗口打开新链接，只能打开一个

target：目标窗口的名称

\_self：自动获得当前窗口名称

\_blank：创建一个新窗口，随机生成一个不重复的名字

窗口名：内存中同一个窗口名只能打开一个，后打开的，会替换先打开的。

<a href=”http://tmooc.cn ”target=”tmooc”></a>

window.name=”自定义窗口名 ”

js中的写法：

function fun4(){

open(“http://tmooc.cn”,”tmooc ”);

}

<a href=”javaScript:fun4();”></a>

1. 窗口大下与定位：

window.innerHeigth/Width:文档显示区宽高

outerHeight/Width:整个窗口的宽高

1. screen.heigth/width:桌面的完整分辨率宽高

screen.availHeight/avaiWidth:去掉任务栏后剩下的宽高

以上的宽高只能读取不能设置。用下面的方法去设置

1. 调整一个窗口的大小：window.resizeTo(width,heigth);调整到多大

位置：1.获得窗口位置左上角X坐标：window.screenLeft | | window.screenX

获得窗口位置左上角Y坐标：window.screenTop | | window.screenY;

2.事件发生时，鼠标的坐标：event.screenX | | screenY（相对于整个屏幕而言）

3.将窗口移动到指定坐标：window.moveTo( x , y );

//打开一个窗口，设置窗口的位置，并随机出现在屏幕上

open(“url”,”name ”,”config ”);

var config =” top=?,left=?,heigth=50,width=50,resizabel=yes”;

定时器：让浏览器按指定时间间隔反复执行同一任务setInterval();

周期性定时器：让浏览器按指定时间间隔反复执行同一任务，如果不手动停止，则一直反复执行。

一次性定时器：让浏览器等待一段时间间隔，执行一次任务后自动停止。在一次定时器的结尾，每次都重新启动一个一次性定时器。

尽量使用一次性定时器，代替周期性定时器。

如何使用：周期性和一次性用法完全相同。

1. 动画的每一步要执行的任务（函数对象）

function step(){

每一步要做的事情

//一次性时， 根据条件判断是否有必要继续注册

}

2、将一步的任务放入定时器，反复调用。

timer = setInterval( stp , 间隔毫秒数);//被多个方法同时调用，

//timer = setTimeout(stp,间隔毫秒数 | 等待毫秒);

该变量应该是全局变量。

3、必须用全局变量，临时存储定时器的编号。

clearInterval(timer)

js中没有多线程。都是同步的。用的是轮循插入的方式每隔一秒插入一个cal( )方法模拟的多线程的异步。

1. window的常用对象，常用BOM对象：navigator，history，location，screen。

常用属性和方法：cookieEnabled，返回指明浏览器中是否启用cookie的Boolean值。

platform：返回运行浏览器的操作系统平台。

userAgent：返回客户机发送服务器的user-agent，存的是浏览器的版本信息、浏览器的版本号。

1. 使用cookieEnabled属性检查cookie是否开启。
2. 使用plugins属性加测浏览器插件。

plugins：保存所有插件对象的

1. 使用userAgent属性识别浏览器的名称和版本号的字符串。

history：window对象中保存当前窗体访问过的url历史记录栈，控制前进、后退和刷新。

<a href=”javascript :history.go(-1); ”>后退</a>

<a href=”javascript : history.go(1);”>前进一次</a>

<a href=”javascript: history.go(2);”>前进两次</a>

historty.go(0);//刷新当前页。go(n)：前进，或者后退n次。

location.search：得到地址栏中的请求参数。如何进一步获得参数名和参数值。

location：当前窗口正在打开的url地址对象。

location.replace(“新url”);//在当前窗口打开新链接。

history中旧url被新url替换。

使用location在当前窗口打开新链接：3种：

location.href=”新url ”;

location.assign(“新url”);

事件：浏览器自动触发的或用户手动触发的对象状态的改变；

2.事件：

当用户与web页面进行某些交互时，解释器就会创建相应的event对象以描述事件信息。

常见的事件：

1. 鼠标经过特定的元素。
2. 用户点击页面上的某项内容。
3. 用户按下键盘上的某个按钮
4. 用户滚动窗口或者改变窗口大小
5. 页面元素加载完成或者加载失败。

事件处理函数：当事件触发时，自动执行的函数。

事件处理函数，指用于响应某个事件而调用的函数。

事件处理函数的本质是对象的一个特殊属性。

每一个元素对象都可以触发各种事件。

onmousedown：把某个元素拖着走。

onmouseup：拖完以后把该元素放到某个位置。

onmouseover事件：鼠标进入边界，触发一次。

onmousemove：就是当鼠标放在某个图面的位置时，放大这个区域。

事件处理：

1. 事件定义（绑定事件处理函数）3中

HTML：<标签 onxxx=”函数调用/js语句 ”>相当于

把js语句eval(“fun( )”);this-->window，this是谁掉指的就是谁

js：elem.onxxx = 函数对象

事件名叫做去掉on的click，onclick是属性名。

DOM标准： elem.addEventListener（“事件名”，函数对象，是否在捕获阶段触发 true/false）。可以决定事件处理的时机。//默认在冒泡时触发事件。

在IE8中事件定义：elem.attachEvent（“on事件名”，函数对象）；

都是用于事件的绑定。

1. 事件周期：

从浏览器捕获的事件触发事件后一直到最后到一个事件触发完，经历的过程。

1. 由外向内捕获。：从外层
2. 目标触发：自动执行目标元素上绑定的事件处理函数。
3. 事件冒泡：由内向外，从目标元素的父元素开始，向上执行每层的事件处理函数，到最外层html结束。
4. event对象：什么时候创建，封装了什么。事件发生时创建，封装了事件信息。比如：

srcElement（IE的）/target：事件源对象。

eventPhase：事件处的传播阶段。

screenX , screenY、clientX/offsetX、pageX/screenX：事件发生的X坐标。

clientY/offsetY/pageY/screenY/y：事件发生的y坐标

keycode，charCode：键盘事件中按下的按键。

button：鼠标哪个按键别按下了

cancelBubble：是否取消事件冒泡。

returnValue：是否阻止事件默认行为。

var e = window.event | | arguments[0] ;//不存在兼容性问题

var src = e.srcElement | | e.target ;

src.style.backgroundColor = "yellow";

alert(src.className);

src.style.backgroundColor="";

IE8：window全局的event属性，当事件发生时，也会自动创建event对象，会保存在window的event里面。

取消

取消或利用冒泡：

if(e.stopPropagation){+

e.stopPropagation();

}else{

e.cancleBubble = true;//IE的取消冒泡的方法

}

1. \*取消事件：

if(e.preventDefault){

e.preventDefault();//DOM

}else{

e.returnValue = false;//事件取消

}

何时取消：在表单提交前，如果验证未通过，则取消之后的自动提交。

方法与函数的区别：

方法：定义对象时，对象的方法，出现在对象中。

函数：函数其实也是方法。

JavaScript类库的出现，是为了简化JavaScript的开发。JavaScript类库封装了预定义的对象和实用的函数，可以帮助开发人员来建立有高难度交互的web 2.0特性的客户端页面，并兼容各大浏览器。

特点：兼容各大浏览器。

web1.0 - 以内容为主的网站（门户网站，就像在超市买东西，各买各的）

web2.0 - 以人的关系为主的网站（社交网站，你看的一篇文件可能会影响到别人也看）

web3.0 - 基于移动互联网的社交网站（微信）

web前端 - 全端开发JavaScript

1. 客户端 - PC端和移动端
2. 服务器端 - NodeJS：基于js开发的服务端的技术
3. 数据库端 - MySQL

浏览器（里面有内核，里面有两个引擎用于加载js或者css，来回切换）。

其他浏览器的内核（WebKit -Apple）：Safari，Chrome，Firefox

3.cookie

JQuery：

它的核心内容：

1.事件处理

1. Ajax(Asynchronous Javascript And Xml)：是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。使用JavaScript向服务器提出请求并处理响应而不阻塞用户。核心对象XMLHttpRequest。JavaScript 可在不重载+页面的情况与Web服务器交换数据，即在不需要刷新页面的情况下，就可以产生局部刷新的效果。
2. JQuery是一个优秀的轻量级的JavaScript类库，JQuery可以兼容CSS3，及各种浏览器。使得开发人员更方便处理HTML页面，处理事件，操作DOM、实现动画效果，并且方便地为网站提供Ajax交互。
3. 使得用户的HTML页面保持代码和HTML内容分离。

注意：JQuery 2.x开始不再支持Internet Explorer 6,7,8

JQuery：

概述：jQuery就是一个js文件

分类：

JQuery - web版本

jQuery UI（User Interface）- 集成UI内容

JQuery Mobile - 移动版本（WebApp）

版本1.4.2 - 企业开发使用

1.8.3版本 - 企业使用最新

通常在开发的时候用jQuery1.3.压缩版。

jQuery的选择器：jQuery的根基。

window.$ = HTMLElement.prototype = function(selector){

return ( this==window ? document : this).querySelector( selector );

}

$(selector)-jquery的工厂函数，该函数返回jQuery对象

jQuery的约定 - jQuery对象前增加“ $ ”符号。

DOM对象与jQuery对象。

DOM对象：通过DOM获取的元素，称为DOM对象。

jQuery对象：通过jQuery包装DOM后产生的对象。

jQuery对象的底层还是DOM。

DOM对象与jQuery对象可以互相转换。

jQuery事件将DOM事件封装成对应的。

选择器-是jQuery的根基

1. 基本选择器。
2. 层级选择器
3. 过滤选择器 - 在选择器前，具有 ”: ”符号的
4. 基本过滤器
5. 子元素过滤器
6. 内容过滤器
7. 可见性过滤器
8. 属性过滤器
9. 表单对象属性过滤选择器
10. 表单选择器

$(“input”);

JQuery中DOM操作

基本操作：

\*html() - 原生DOM的innerHTML属性

\*val() - 类似于原生DOM中的value属性。两种用法：1.获取 2.设置

\*text() - text(文本内容)

\*attr() - 获取或者设置指定元素的属性

attr(attrName) - 直接得到该属性，类似于getAttribute（attrName）;

attr(attrName,attrValue) - 类似于setAttribute(‘属性名’，‘属性值’)；

removeAttr(attrName) - 类似于 removeAttribute(‘属性’);

样式操作：

attr(“class”, ””);//可以给某个元素加上样式class

addClass（“”）；追加样式

removeClass（“”）移除样式

removeClass（）移除所有样式

toggleClass（“ ”）切换样式

hasClass(“ ”)是否有某个样式

css(“ ”) 读取css的值

css( “ ”, ””) 设置多个样式

遍历节点：

parent( ) - 获取指定节点的父节点，没有选择器

children( )/children(selector) - 获取指定节点的所有子节点

next( )/next(selector) - 获取指定节点的下一个兄弟节点

prev( ) /prev(seelctor)- 获取上一个兄弟节点

siblings( )/siblings(selector)其他兄弟节点

find（expr）/查找满足选择器的所有后代，必须传参数

创建节点：

元素节点 - createElement - $(HTML) : 工厂函数$(“<li>df</li>”)

文本节点 - createTextNode - text()

属性节点 - setAttribute - attr(“”,””)

在jQuery中只用一句话：$（“#city”）.append($(“<li>dd</li>”));

删除节点：

remove( ) - 删除自身及后代节点

empty( ) - （清空）删除后代节点，但是保留自身节点

插入节点

1. 内部插入

\* append

\* prepend

\* appendTo

1. 外部插入

jQuery的off( )方法：该方法通常用于移除其他方法，并且通过on（）方法添加的事件处理程序。

jQuery的empty( )方法：该方法通常移除被选中的元素，包括所有文本和子节点。

jQuery的live( )方法向尚未创建的元素添加事件处理器。

$(“btn”).click(function(){

$(“p”).empty( );

});

ready（） - 类似于window.onload的作用

1.$(document).ready(function(){ });

2. $().ready(function(){ });

3. $(function(){ });

4. jQuery（function($){ }）

var i = 3;

function init( ){

alert(“外层init”+i);

}

(function($){

var i = 2;

function init(){

alert(“内层init: ”+i);

}

init( );

})(jQuery)//这样写的好处在与写了一个闭包

init();

执行结果：

内层init：2

外层init：3

ready与onload的区别

\*ready有简写方式，在一个HTML页面中允许有多个。等待HTML页面中所有DOM结构（元素节点，文本节点，属性节点）加载完毕后就可以执行。这个的加载速度快。

onload \* 没有简写方式，在一个HTML页面中不能给同一个元素附上该事件。必须等待HTML页面中所有内容全部加载完毕后才能执行。

事件绑定和解绑。

在JavaScript中“”和‘ ’是一样的。

在java中 “”是字符串， ’’表示的为字符。

\*动画效果

1. 显示与隐藏。

show( ) - 显示

hide（） - 隐藏

类数组操作- JQuery是数组对象。

var JQuery = function(selector , context){

return new jQuery.fn.init( selector , context );

}

var init = jQuery.fn.init = function(selector , context ){

return jQuery.makeArray(selector , this );

}

var makeArray = function( arr ,results ){

var ret = [ ];

return ret ;

}

window.jQuery = window.$ = jQuery ;

get( index ) -根据角标返回DOM对象。

eq（index）- 根据角标返回DOM对象。

index - 根据DOM对象返回对应角标。

用for循环时，将jQuery对象转成了DOM。

隐式迭代：在遍历的过程中，只关注入口和出口。

var arr = [ ];

for(var i = 0;i<arr.length ; i++){

var text = arr[i] ;//出口

}

显示迭代 ：- for循环 for( var i = 0; i < length; i++)

在遍历的过程中，从开始到结束控制遍历的所有过程。

jQuery的遍历

1.each（callback）； -- 对象方法，完整写法 $(“img”).each(callback);//对象是$()，是jQuery对象

2.$.each(object , callback); -- 全局函数，object：需要遍历的当前元素或者DOM对象。

$().each(function(index,domEle))方法

index:当前遍历数组或者对象的索引值

domEle - 遍历数组或对象得到的每个元素（DOM对象）

日历插件 - My97DatePicker(国产的插件)

目录说明

lang - 存储国际化(i18n)文件

本地化（i10n）

skin - 皮肤（日历控件长的样子）

如何使用my97日历插件

本地

bootstrap：响应式设计

$() = jQuery( ) ;

this所用的地方

1. 在具有上下文环境中使用this。this指代上下文环境
2. 在JavaScript创建构造器时使用this。this指代new之后的对象
3. 在jQuery中的隐式迭代中使用this。this指代DOM对象。

this的用法是JavaScript整个内容中最难的部分。

jQuery最重要的是支持插件。

插件的作用：为了扩展或者简化jQuery的开发。

特点：数量多，jQuery官方的插件有700多种。

插件库地址：http：//plugins.jquery.com。提供了大量的插件，并给出了每个插件的用户评级、版本及bug等。库中列出了每个插件的ZIP文件下载、演示、示例代码及教程。

学习插件的目的：

1. 掌握jQuery插件的使用方法。
2. 掌握如何学习jQuery插件的使用。
3. 日历插件
4. 表单验证插件

jQuery.extend（）:用一个或多个其他对象来扩展一个对象，返回被扩展的对象。

软件服务器：中间件，为了运行web应用的一种软件。Apache，weblogic。

软件架构：

\*B/S - 浏览器（browser）/服务器端（server）

互联网 - 企业级

互联网 - 网易、腾讯、百度等

好处：软件升级 - 服务器端的升级。

软件服务器：为了向客户提供服务，必须安装能够监听特定端口，并接受客户的端请求的应用程序，即服务器应用程序。

服务器应用程序包括：

Web服务器：提供web页面服务。

DNS服务器：把域名解析为计算机的IP地址。

数据库服务器：提供数据的永久存储和高效查询。

FTP服务器：文件上传和下载服务

SMTP/POP3服务器：邮件收发服务

SSH服务器：远程登录服务。

Samba服务器：文件和打印机共享服务。

访问web应用的时候：

http：//www.XXX.com - 虚拟地址，IP是真实地址。

客户端访问网络上的一地址台服务器，必须：

访问协议 - 主机地址 - 端口号

访问协议：类似两个人通话必须先确定所说的语言。

主机地址：服务器的域名或IP地址

端口号：一台机器上可能提供多种服务，不同服务使用不同的端口。

XAMPP软件

Apache - 软件服务器

当启动失败时，原因-端口号被占用

解决方法：去config里面修改 - conf目录里面 - httpd.conf文件里面，修改listen 8888

在本地搭建web应用程序的过程：

1. \*当要修改访问的IP地址时，需要找到C:\Windows\System32\drivers\etc中的host文件，然后给最后一行加上192.168.43.5 www.jd.com，当前IP + 域名 即可，然后另存为，最后替换host文件。
2. 再找到D:\xampp\htdocs中的所有文件，删除所有文件或者把这些文件放到一个地方，再新建一个名为index.html的文件，名称必须是index.html，编辑里面的内容，再次访问 [www.jd.com即可访问到index.html](http://www.jd.com即可访问到index.html)

关于80端口号：

在开始菜单输入“cmd”运行命令窗口

先输入net -a，然后在窗口中输入netstat -ano，找到占用80端口号对应的

netstat –ano | findstr “:80 ”

## 输入tasklist /fi "PID eq 4":查找PID为4的服务。

使用taskkill /pid 序号 /f，回车即可关闭进程。

PID值。

然后启动任务管理器，在其中找到的“详细信息”，找到对应进程的PID，结束即可。a316694e-f74a-4ed8-afc9-1c5fcd93efbc数据库：

基本概念：数据仓库，用于存储或操作数据内容

两大阵营：

1. 关系型数据库（SQL） - 是目前主流数据库，以表（行列）的形式存储数据。

\*Oracle - 企业级开发98%的市场份额。

Oracle认证：1.初级 - DBA 2.中级 -管理

\*MySQL - 也是Oracle的产品，互联网开发98%市场份额都是这款产品。是瑞典MySQL AB公司开发的。后期被SUN公司收购，SUN被Oracle收购。

特点：免费、开源、持续更新

LAMP组合 - 目前开发互联网网站的主流架构

1. Linux

A - Apache

M - mysql

p - PHP ,都是免费开源的

SQL分为四大类：

2、非关系型数据库（NOSQL） - 是以文档方式存储数据。是以key - value形式存储数据。

NoSQL：是一种运动（抗议关系型数据库），逐步发展之后，出现多种产品。

\*MongoDB - JSON格式

创建数据库是在系统磁盘上划分一块区域用于数据的存储和管理，如果管理员在设置权限时为用户创建数据库，则可以直接使用，否则需要用户自己创建数据库。

SQL语言并不区分大小写。（官方建议大写）

进入mysql：mysql -uroot -p //不能加；分号

密码：15963159

show databases;

DDL(创建数据库)：

\*create database 数据库名称 default character

set utf8 collate utf8\_general\_ci；

设置数据库的的字符集。

修改字符集：alert database jd character set utf8;

删除数据库：

drop database 数据库名称

修改数据库：Alert database 数据库名称 character set utf8

在写SQL语句的时候，必须切换到数据库中。

用法： - USE 数据库名称

执行 use day0107

会出现 Database changed

show tables;//显示有几张表

自动向mysql中导入sql语句：

1.执行set names utf8;//设置连接的两端字符集都为utf8。就可已导入sql，设置了客户端命令行选择的编码，把连接编码也设置成了utf8。

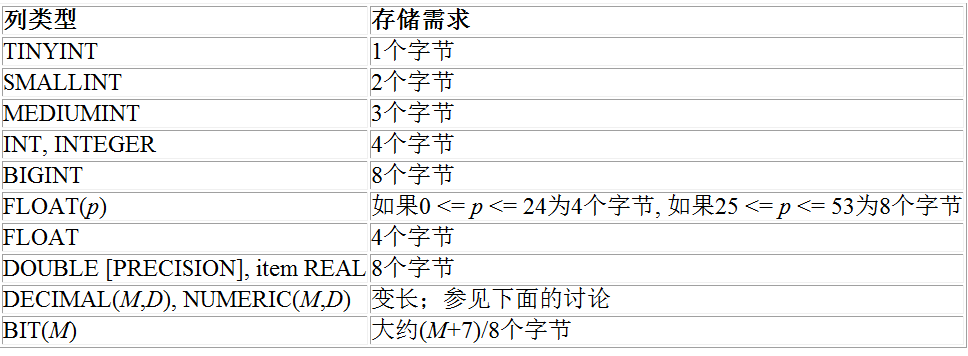
/s : 查看设置的字符集。

2.source 文件位置和文件名字

数据表操作：

mysql中的数据类型：

数值（Number）：int - 整数



日期（Date）日期和时间

DATE - 日期（默认格式：yyyy-MM-dd）

DATETIME - 日期时间（yyyy-MM-dd hh:mm:ss）

TIMESTAMP - 时间戳

字符串类型

CHAR - 长度固定的字符串，定义一个字符串的长度为10,没占用的就用空格补齐。

VARCHAR - 长度可变的字符串

定义一个字符串的长度为10，实际占多少就存多少

创建数据表：

create table li(

字段名 数据类型

);

CREATE TABLE myuser(

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(30),

pwd VARCHAR(20)

);

查看表结构： DESCRIBE myuser ;

DML(增删改)

INSERT INTO myuser(id,name,pwd) VALUES(null , ’jerry’, ’123’);//自增列用null补齐。

update 表名 set 字段名 = 字段值 where 字段名 = 字段值；

也可以 update myuser set id = 09，name = ‘sde’, pwd = ’980 ’；

PHP语言 - 类似于JavaScript的语言。

JavaScript是客户端（HTML）的脚本语言。

PHP是服务器端的脚本语言。

PHP文件是不能右键打开的，是不能用浏览器直接打开。

PHP文件是应该运行在服务器（Apache）上的。JavaScript是运行在浏览器萨上的。

PHP文件允许的内容：

1. 允许写HTML代码。
2. 允许编写PHP代码。

第一个PHP文件：

创建PHP文件，扩展名为“.PHP”

如何编写PHP。以<? php ?>

PHP页面与HTML页面的区别：

HTML页面 - 静态页面

静态资源 - HTML | CSS | JAVASCRIPT

PHP页面 - 动态页面

动态资源 - 根据用户的操作，动态变化。

PHP语法：

常量：const - const 常量名 = 常量值

define（常量名，常量值）；

变量：定义之后值可以改变的量。PHP中的变量可以先后赋值为不同类型的值。 $变量名 = 值；

例如：$userName = ‘Tom’;

$dataArray = [3,5,’Tea’,true,time()];

PHP中的数值在内部也可分为八种类型：

四种标量类型：boolean、integer、float/double、string

两种复合类型：arr：有三种，1.直接定义的数组。2.索引数组。3,。关联数组。object：要先先创建类，然后才再new 对象。

要创建一个新的对象，要写一个类的形式。

两种特殊类型：resource - 资源类型，用于保存外部资源的一种引用，使用场景：在文件上传中，保存上传的文件

NULL

数据类型：四种标量类型

“ ” - 可以识别变量名的

\* 性能相对低（有匹配的过程）

PHP中的连接符用的是”. ” 而不是+

基本与JavaScript的元素保持一致。

PHP预定义函数：

预定义变量：

$\_GET - 接受客户端以请求类型为GET方法发送的数据

$\_POST - 接收客户端以请求类型为POST方法发送的数据内容

$\_REQUEST - $\_GET、$\_POST等

$\_FILE - 专门用于文件上传

$\_COOKIE - 接收客户端保存的Cookie数据

预定义函数：数据库函数

连接数据库：PHP为连接MySQL数据库提供了多套扩展函数库，使用其中任意一套库，都必须保证如下两点：

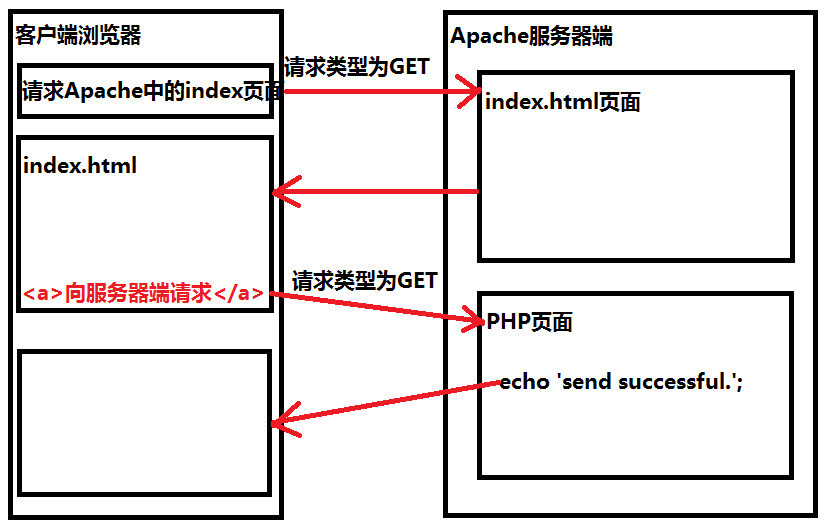
1. 将函数库的定义文件保存到PHP HOME/ext目录下，然后找到php\_mysql.dll和php\_mysqli.dll文件必须存在。

在XAMPP软件的安装目录/php目录中

\*php.ini文件 - PHP的主配置文件

\*在PHP中配置有关mysql数据库的。

1. 在PHP的主配置文件（php.ini）中声明启用该函数库



一连接，二定义SQL语句，三执行$mysqli\_query($conn,$sql);

设置字符集，$mysql->query(set name utf-8);

$conn = new mysqli("localhost","root","15963159","jindong","3306");

$sql = 'select \* from teamei';

$result = mysqli\_query($conn,$sql);//得到一个结果集对象

//mysqli\_query()方法执行的select语句

/\*

该方法返回值为mysqli\_result对象

\*属性

num\_rows - 得到记录条数

field\_count - 得到字段数量

mysqli\_fetch\_array(结果集对象)

HTTP：

URL概念：URL - 统一资源定位符，网址和链接都是URL。

URI：统一资源标识符，可以直接找到该用户的，最具体的。

URL与URI的区别：所有的URL都是URI，但是URI不一定是URL。

完整的URL：

网络协议：IP地址（虚拟地址）：端口号/路径；参数？查询数据#锚点

http://127.7.0.0.1：80/11\_SERVER

<ftp://code.tarena.com:80/11_server$http>/day01.rar；反斜杠后面的叫做路径。

一般实际使用时：

<http://www.baidu.com/news>

<http://127.0.0.1:8080?key=value>

<http://localhost:80#mylink>

路径：

绝对路径：无论位置在哪，目标元素的路径是固定不变的。不能在项目中用绝对路径，因为在发布到生产上时，http后面的ip和端口号会发生改变。

相对路径：根据当前位置的变化，路径也会变化。

查询数据

格式为key = value&key = value

主机/端口号：计算机中对外提供的服务程序可以绑定到任意一空闲端口上，从而实现监听客户端连接请求的任务。



网络协议

网络协议 - 客户端与服务器端之间的协议

主流协议

http协议 - https协议加密的协议。加密协议用的是OpenSSL协议。

socket协议 - HTML5的web socket ws

HTTP协议 - 多用于B/S架构，浏览器/服务器

\* http有一些问题：

1. 每次客户端与服务端交互式，都得先建立连接，交互完毕后，再关闭链接。这叫做短连接。再第二次发送请求时，还的再告诉服务器，它的当前状态。
2. 无状态：服务器端只能记得住当次请求状态，后面就不清楚了。

在socket协议中，就不存在无状态这种情况，socket用于C/S架构中。

HTTP/0.9：1991年制定，有严重缺陷，只支持GET请求，不支持MIME类型，和快被HTTP/1.0代替。

主流版本：HTTP/1.0:1996年定制，支持多种请求方法，支持多媒体对象，得到广泛应用。HTTP 2.0：关注HTTP协议的性能优化以及更强大的服务逻辑远程执行框架。

客户端先向DNS服务器发送请求，返回一个ip。然后再发用HTTP请求。

请求协议：GET请求，和POST请求

响应协议：

请求类型： 请求行与请求方法，请求消息请求服务器对资源进行一些操作。

请求消息的起始行称为”请求行 ”，包含如下三部分：

<method><request-URL><versioin>

请求方法：\*GET ：从服务器获取一份文档。-最常使用的

\*HEAD：只获取文档的响应头部。

\*POST：向服务器发送需要处理的数据

\*PUT：将请求主体中的数据存储在服务器上。

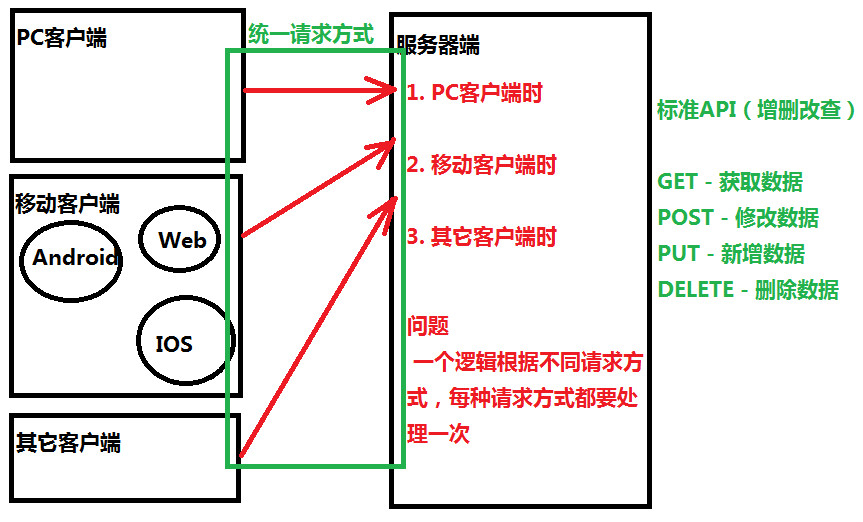
\*TRACE：对可能经过代理服务器传送到服务器上去的消息进行追踪。

\*options：决定可以在服务器上执行那些方法

\*DELTE：从服务器上删除一份文档

标准API：一种标准的请求方式，把不同客户端终端的请求进行统一请求方式。

其他客户端：类似于车票自助系统。



标准API - 标准的请求方式

GET - 获取数据

POST - 修改数据

PUT - 新增数据

DELETE - 删除数据

状态码：1XX - 服务器端的信息

3xx - 表示重定向

\*302 - 表示重定向

\*304 - 表示服务器端资源没有变化，访问缓存

\*问题 - 这次请求没有真正地访问服务端

\*305 - 表示代理

\* 4XX - 客户端错误

\* 404 - 网页找不到了

\*403 - 拒绝访问

\*405 - 表示请求类型不允许

\*5XX - 服务器端错误

\*500 - 服务器端错误

\*502 - 路径错误

\*504 - 请求超时

请求头

请求头的格式：

key：value

MIME类型 - 文件类型

MIME类型是如何定义的

//定义MIME类型

<mime-mapping>

<extension>html</extension>

<mime-type>text/html</mime-type>

</mime-maping>

\*常见的MIME类型

HTML - text/html

htm - text/html

xhtml -

css - text/css

js - application/javascript text/javascript

json - application/json

jpg - image/jpeg

jpeg - image/jpeg

text - text/plain

webm - video/webm

\*Accept - Encoding - 表示服务器端是否接受压缩

\*gzip - 是指一种服务器端的压缩格式，比如jQuery.min的就是gzip。

问题 - 客户端请求的数据内容越大。

对网络带宽的要求越高，流量占用大。用户体验不好。

Accept-Language - 表示服务器接受的语言。

\* zh - CN - 简体中文

zh - TW - 繁体中文

us - 英文

Referer：防盗链。表示当前的网站的来源，获取请求头中的Referer的值，判断是否包含自己网站的地址。

响应头：响应头可用于描述响应本身、服务器、响应主体的特征。

响应头可分为四种：

通用头部：

Connection/Date/Cache-Control/Pragma

请求专用头部：只能出现在请求消息中

Host/Referer/User-Agent/Client-IP/Accept/Cookie/Cache-Control

响应专业头部：只能出现在响应消息中

HTTP/1.1

Date

Content - Type

实体头部：描述消息主体特征

Location/Content-Type/Content-Length/

Content-Encoding/Expires/Last-Modified

请求头：

Cache-Control - 缓存控制

\*max-age = 0; - 设置缓存最大活动周期

\* 0 - 表示没有缓存

\* 设置缓存保存的最大时间为单位为毫秒

\* Content - Length - 请求数据的大小

\* Content - Type - 请求的MIME类型

post请求参数的形式：\* application/x-www-form-urlencoded

Keep-Alive - 设置保持连接的超时和最大存活时间。

\* timeout = 5，max = 99

请求参数：

\* GET请求类型

\*将请求数据 - URL？key=value（浏览器地址栏），请求地址的长度是有限制的。请求数据内容过多，服务器不能接受到完整的参数。如果请求数据中包含中文，要进行转码的。

\*POST请求类型

\*浏览器地址URL，不包含请求类型。

\*安全相对比较高。对请求数据的大小长度没有限制的。

\*如果请求数据中包含中文的话，可以不进行转码。

Unicode码：一般是解决中问题的。%E%B%C%BA%91%E9%

将中文内容

Cookie - 将Cookie自动携带到请求头，我们可以利用浏览器存储的Cookie，请求至服务器端，服务器端可以根据Cookie进行记住账号等操作。

GET与POST请求方式的请求协议的区别

\*GET请求方式

\* 请求行

\*请求类型 - GET

\*请求地址 - URL？请求参数

\* 请求体 - 空

POST请求方式

\*请求行

\*请求类型 - post

\*请求地址 - 只有URL

\* 请求参数

头消息

\* 通用（请求与响应）头消息

\* Connection

\* Content-Length

\* Content-Type

\* Date

\* 特殊响应头

\* 控制是否缓存的信息

\* Cache -Control - 0 表示不缓存 / 一般设置这个即可

\* Expire - 0 表示不缓存，针对IE浏览器的，Expires头部指定缓存的过期确切时间。由于很多服务器的时钟头部同步，所以推荐最好用剩余秒数来代替过期的绝对时间。

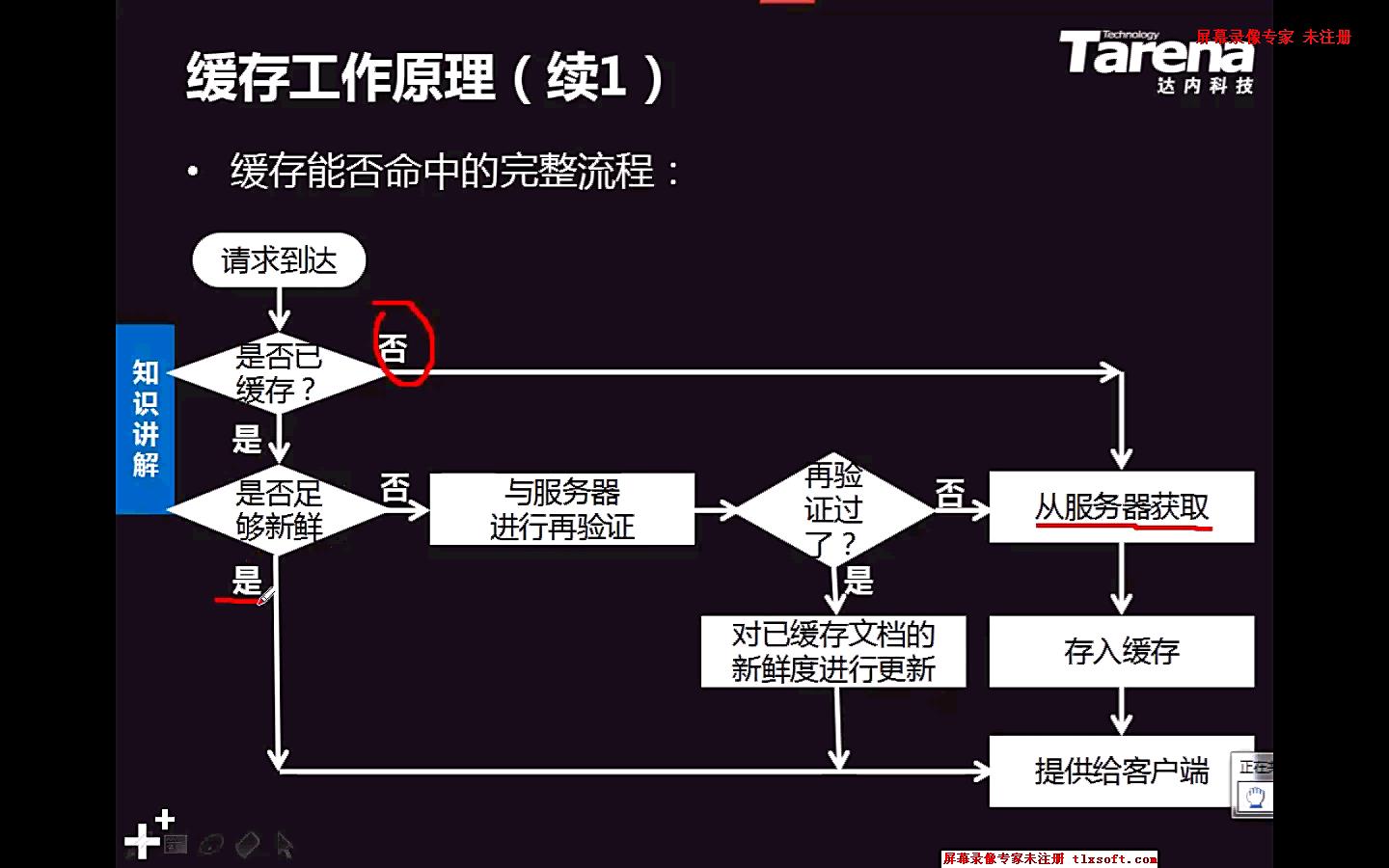
\* 注意 - web浏览器是允许缓存的。

没有缓存的两种条件：1.从来没有访问过。2.有缓存，但是被清空了。

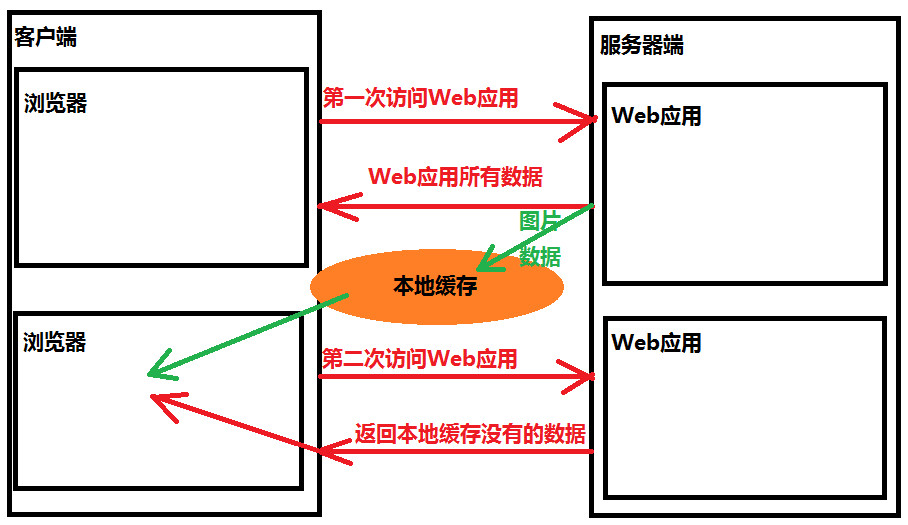
请求到达后 -- 是否要缓存 - 否 - 每一次都从服务器中获取。- 存入缓存 - 提供给客户端。

- 要 - 是否是已缓存内容 - 否 - 从服务器中获取，然后加入缓存中

304（访问缓存） - 是 - 直接提供给客户



取缓存图：



控制HTTP头信息

在HTML页面中 使用 <meta />元元素

PHP页面

\* header( )函数设置头信息

\* response.addHeader(key,value)

HTTP性能优化

网站性能优化最主要的就是减少HTTP请求及每次响应中内容的长度。可以从连接过程中的下列方面加以考虑：

1. 域名解析：尽可能减少域名解析次数

\*<a href=””></a> <img src=””> 使用HTML5的Canvas|SVG，把图像绘制到页面中。

<script src=””></script>

1. 创建连接：努力减少连接创建次数

尽量减少客户端与服务器端之间的交互

\* 减少请求的次数

\* 使用Ajax的异步交互可以减少请求

\* 使用socket这种协议（实时连接，连接一次后就再不断开了）

1. 发送请求：尽量减少请求次数
2. 等待响应：提高服务器端运行速度

\* 主要从服务端进行优化

\* 将响应数据进行压缩

\* 将缩短等待响应的时间

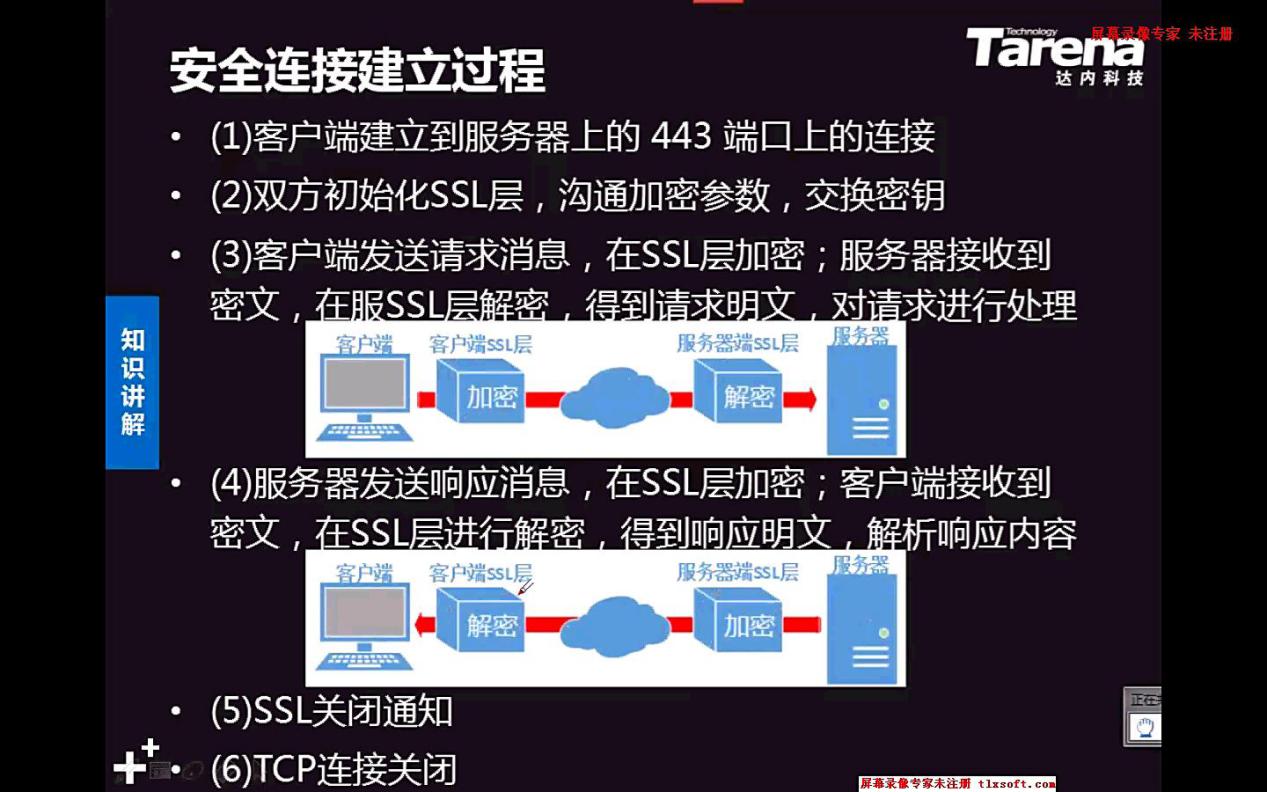
浏览器对HTML | CSS的解析，对JavaScript的解析

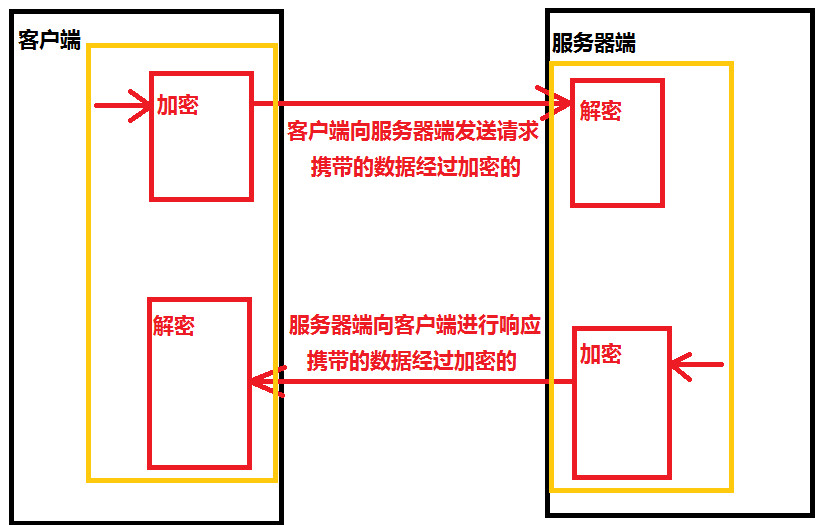
1. 接受响应：尽可能减小响应数据长度

安全的HTTP协议：HTTPS

http协议：默认端口号为80，属于明文传输，可能被窃听和篡改。

https的默认端口号为443，电商网站，银行网站都用。使用OpenSSL的安全协议。





IDE 集成开发工具

Aptana Studio 3

WebStorm

web前端的课程体系

\* HTML + CSS ：HTML一堆标签，Css - Css3动画

\* JavaScript - 最重点

\* ECMAScript 语法 | 面向对象

\* JQuery / HTML5 /Server/Ajax

\*jquery - 使用

\* HTML5 - 发展会越来越好，越来越主流

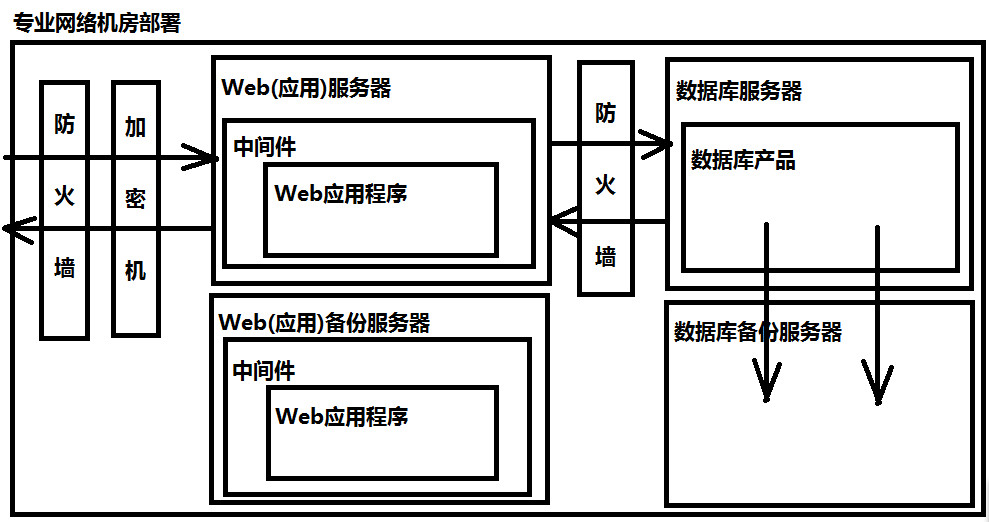
\* Ajax - 绝对重要

\* JS框架

\* 响应式设计

\* MVC模型

\* WebAPP - 移动端很火



AJAX

同步交互：同步交互客户端向服务器端发送请求，直到服务器端进行响应，才能发送下一个。

异步交互：客户端向服务器端发送请求，不用等服务器端响应回来就能接着发请求。

setTimeout就是这样的一个异步调用。

JS 是异步执行的语言，但是人的思维是同步的；

Ajax ： Asynchronous JavaScript and xml：JavaScript和xml的异步

特点：

1.（不严格的定义）客户端与服务器端进行交互，而无需刷新当前页面的技术。

2.实现异步交互的技术

3.ajax实现的是B/S架构的异步交互。

AJAX是一种用来改善用户体验的技术，其实质是，使用XmlHttpRequest对象异步向服务器发请求。服务器返回部分数据，而不是一个完整的页面，以页面无刷新的效果更改页面中的局部内容。

同步交互：执行速度相当比较慢，响应的是完整的HTML页面

异步交互：执行速度相当比较快，响应的是部分数据

AJAX的核心对象：XMLHTTPRequest对象，它是在事件对象中包含的。

function getXhr( ){

var xhr = null;

if（window.XMLHttpRequest）{

xhr = new XMLHttpRequest( );

}else{

xhr = new ActiveXObject(‘Microsoft.XMLHttp’);

}

return xhr;

}

Ajax的核心对象

\* 属性 \* readyState

\* status

\* 方法

\* open（） - 与服务器端建立连接

\* send（） - 向服务端发送请求

\* 事件

\* onreadystatechange事件：监听服务器端的通信状态改变



var xhr = getXhr();//1.创建XMLHttpRequest对象

//2.创建连接

//xhr.open('method',URL,async);async - boolean类型

xhr.open("get","ajaxRequest.php",true);//官方认为使用XMLHttpRequset对象就是为了实现异步交互。

3.客户端向服务端发送请求

\* send(请求参数)方法

\* 请求参数的格式 - key = value

\* post的方法必须多写一个

xhr.setRequestHeader(“Content-Type”,”application/x-www-form-urlencoded”);

xhr.send("user=zhangwuji");

//在get请求的时候，不能发送参数，只能写成xhr.send(NULL);

在post请求的时候，可以写参数

4.客户端接受服务器端的响应

使用onreadystatechange事件

监听服务器的通讯状态

\*readyState属性

\* 得到服务器端当前通信状态

\* 备选项

\* 0 尚未初始化，服务器端未初始化，在客户端上是取不到的

\* 1 正在发送请求

\* 2 请求完成

\* 3 正在响应，客户端正在接受服务器端的数据

\* 4 响应完成

\* statue 状态码 200 OK

\* responseText属性

\* 是HTML格式的文本

\*/

xhr.onreadystatechange = function(){

//alert(xhr.readyState);

if(xhr.readyState == 4){

if(xhr.status == 200){

//接收服务器端的数据·

var data = xhr.responseText;

alert(data);

};

}

}

\* GET与POST的区别

\* GET请求类型

\* send（）方法不起作用，但是不能被省略。

get请求时，要写send(null);请求参数添加URL？key=value

实现ajax的四步

1. 创建XMLHttpRequest对象
2. 创建连接open( )
3. 发送请求send( )
4. 注册监听 - onreadystatechange事件

XML语法：

接受服务端的响应数据时：

\* 使用XMLHTTPRequest核心对象的responseText属性。

该属性只能接受字符串的文本（HTML）格式。

问题：

解析过程比较复杂，拆串，即拆那些返回的字符串用，split

拆串或者拼串。

可以用XML和JSON格式来代替，

XML格式：

\* HTML、XHTML、DHTML和XML的区别

\* HTML就是网页 - 元素定义的大小写都可以

\* XHTML就是严格意义的HTML - 元素定义必须小写

\* DHTML - BOM|DOM，只是一个概念的整合

\* XML：作用1：配置文件 2：数据格式。（用于存储数据）XML允许自定义元素。

声明：<?xml version=”1.0 ”encoding=”UTF-8 ”?>

version - 设置当前XML文件的版本1.0 - 目前唯一版本不会更新了。

encoding - 设置当前XMl文件的编码

定义元素：

\* 根元素

\* 必须是起始标签，只能定义一个

\* 定义元素

元素名可以自定义

分类

\* DOM解析XML：先创建XML的解析器

function parseXML(xmlFile){

//创建

var xmlDoc = null;

if(window.DOMParser){

var parser = new DOMParser();

xmlDoc = parser.parseFromString(xmlFile,"application/xml");

}else{

//返回解析器

var parser = new ActiveXObject("Microsoft.XMLDOM");

parser.async(false);

xmlDoc=parser.loadXML(xmlFile);

}

return xmlDoc;

}

var xmlDoc = parseXML('<province id="01" name="山东省"><city name="青岛市"/><city name="济南市"/><city name="德市"/></province>');

/\*console.log(xmlDoc);

var pro = xmlDoc.getElementsByTagName("province")[0];

console.log(pro);

var city= pro.getElementsByTagName("city")[0].getAttribute("name");;

console.log(city);\*/

/\*

在解析xml中，实际开发中

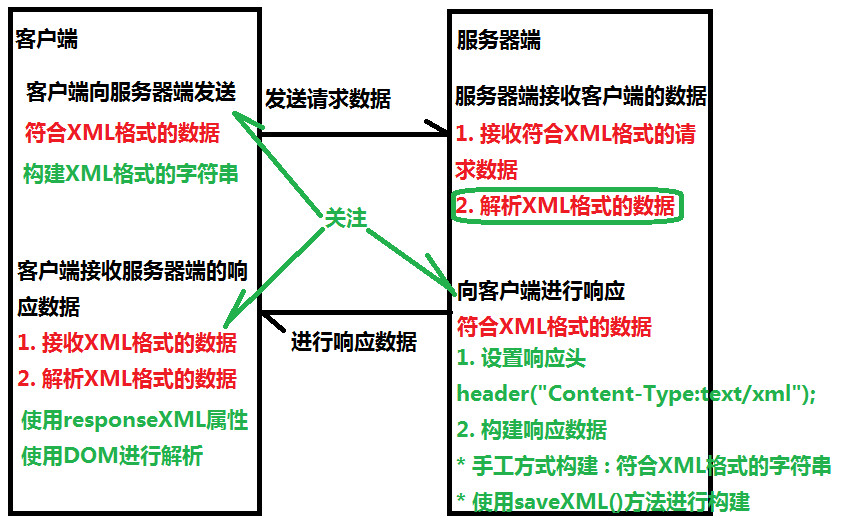
只用getElementsByTagName

\* 请求中的数据格式 - XML

客户端如何构建符合xml格式的数据

服务器端如何接受xml格式的数据

响应的数据格式 - XML



Ajax中的JSON格式

请求格式为JSON

客户端向服务器端发送请求 - JSON

\* 必须构建符合JSON格式的字符串

\* 保证定义字符串时，使用 ‘{“name”：“libin”，“age”：“25”}’//里面使用双引号，外面使用单引号的格式

在发送的时候必须把object形式的json数据通过

JSON.stringify(user);/转为字符串的形式发送

接受：通过JSON.parse(str)把字符串形式的json数据转为JSON格式的数据。

服务器端接受JSON格式的数据

\* PHP接受客户端的数据

响应格式为JSON

服务器端向客户端响应为JSON格式的数据以及客户端接收JSON格式的数据

responseText方法用于那些并非基于XML的格式，HTML和JSON都使用responseText

jQuery中的AJAX：

封装的第一层：类似于原生Ajax的用法

\* $.ajax();--有$的叫做全局函数

$.ajax({

\* url - 请求地址

\* type - 请求类型，默认为GET

\* async - 默认为true

\* contentType - POST方式，默认为application/x-www-form-urlencoded

\* data - 请求数据，格式必须为key:value

\* success - 请求成功后的回调函数

function(data,textStatus){}

\* data - 服务器返回的数据内容

\* textStatus - 表示ajax请求状态

\* error - 请求失败后的回调函数

function(XMLHttpRequest,textStatus,errorThrown){}

\* XMLHttpRequest -

\* textStatus - 表示异常请求状态

\* errorThrown - 错误异常

\* dataType - 设置响应参数类型xml:返回XML文档，可用jQuery处理，html：返回纯文本，script：返回纯文本JavaScript，但是不会自动缓存json：返回JSON数据});预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息来智能判断，比如XML MIME类型就被识别为XML。

当返回的数据不是dataType指定的格式时，响应会出现错误。

封装的第二层 - 基于第一层再次封装：

\* load()；//用法最简单，但是局限性最大 -- 这个叫做对象方法，得用 对象.load()

callback - 可选，当前ajax请求成功后的回调函数

\* 该回调函数的形参（data）就是服务器端响应的

数据内容

data - 可选，发送的参数必须是key/value格式的

注意：服务器端响应的数据自动哦写入到<div>元素中

现象原因 - div元素调用load()方法

load()方法的请求类型

\* 没有请求数据时，请求类型是GET类型

\* 有请求参数时，请求类型为POST类型

\*/

//请求数据格式为key/value，就是Object对象，底层已经将它变为字符串

var user = {"name":"zhangwuji","pwd":"123"};

$("div").load("loadAjax.php",user,function(data){});

$(“#load”).load(url，param，callback);

url：待装入HTML网页网址

data：发送至服务器的的数据。

callback：载入成功时回调函数。

载入远程HTML文件代码并插入至DOM中。在jQuery1.2中，可以指定选择符，来筛选载入的HTML文档，DOM中仅插入筛选出的HTML代码。

$(“#load”).load(“feed.html”);

\* $.get(); //请求类型是GET

$.get(url,data,callback,type);

$.get(“01.php”,{“name”:”libin ”},function(){ },”json ”);

\* $.post(); //请求类型是POST

$.post(url,{“name”:”libin ”},function(){},”json ”);

封装的第三层 - 特殊用法

on( )事件，绑定到选定的jQuery对象中的元素。

events,[selector],[data],callback

$(“p”).on(“click”,{foo : ”bar ”}，myhandler);

function myhandler(evnet){

alert(event.data.foo);

}

\* $.getScript(); - 动态读取脚本（JavaScript代码），需要的时候加载，不需要的时候就不加载。

当浏览器加载HTML页面时，js代码何时被加载，这时当HTML页面被加载时，所有js代码全部被加载。

但是js代码加载过多时，会影响html页面加载的速度。已经被加载的js代码并不是被立即执行。

$("#btn").click(function(){

/\*

\* $.getScript(url,callback)

\* callback - 读取成功后的回调函数

\*/

//动态的加载，只有在执行到这里才会加载该代码

//这种的加载速度快

//getScript也可以读取服务器端的文件

//$.getScript("mycode.js");

//当点击后去动态的加载存储在服务器端的JavaScript代码。

$.getScript("07.php",function(data){

console.log(data);

});

\* $.getJSON(); - 接收json格式的数据

表单的Ajax异步请求

表单的序列化

\* serialize( )方法

\* 返回json字符串

格式 - {key：value}。但是表单元素必须带name属性。

jQuery.form插件有两个核心方法 -- ajaxForm()和ajaxSubmit()，它们集合了从控制表单元素到决定如何管理提交进程的功能。

ajaxForm()：不能提交表单。在document的ready函数中，使用ajaxForm来为ajax提交表单进行准备。

ajaxSubmit():马上由ajax来提交表单，可以在任何情况下进行该项提交。

跨域请求： $.getJSON( );

实现跨域：



完全跨域 - IP不同

\* http：//www.baidu.com

\* http : //www.tedu.cn

跨子域：IP相同，但是端口号不同

\* http：//127.0.0.1:8888

\* http : //127.0.0.1:9999

域名

\* 顶级域名

\* http：//www.baidu.com

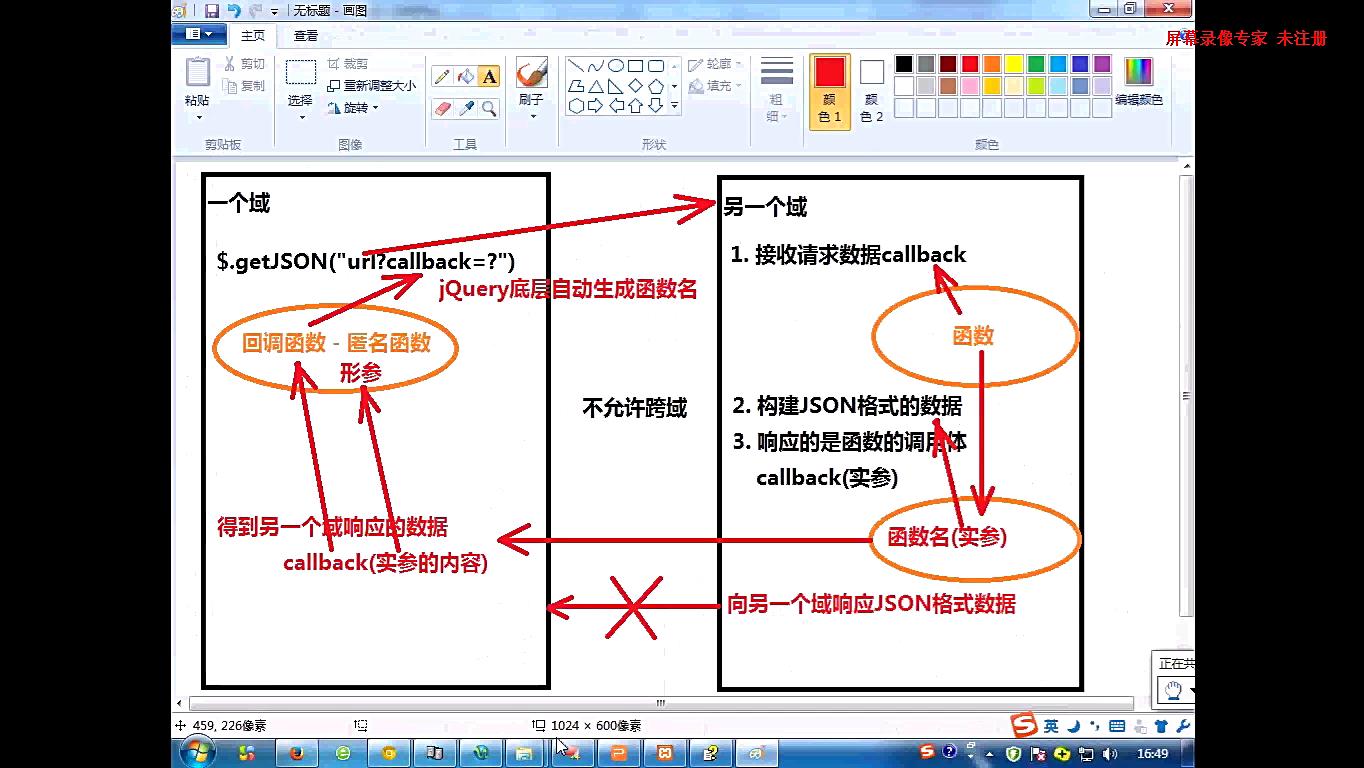
二级域名

\* <http://wenku.baidu.com>

http://www.baidu.com/kongjian

万维网协议默认是不允许跨域请求的

跨域请求的原理图：



Cookie

\* 浏览器的缓存：1.存储在浏览器内存中，一旦关闭浏览器后，数据会被自动销毁。

2.存储在用户电脑硬盘中的，关闭浏览器后，数据不会被销毁。

它是一个缓存文件，会将一些用户数据存储在里面。

很多网站的自动登录就是从浏览器中的cookie获取上次保存的账号和密码来登录的。

web存储中的localStorage就是用来替代cookie的。

Cookie存在的几个问题：

1. 用户数据是以明码来存储的。
2. Cookie中的数据谁都可以读取。
3. A网站生成的cookie文件，归属到百度中，当访问百度时，这是所有的cookie文件会带到请求头中，携带到百度中，这样就会给百度造成很大的压力。一个浏览器最多存200个cookie文件。

如何操作cookie

1. 读取cookie - 将用户名或密码从Cookie中读出来。
2. 写入cookie - 将用户名或密码写入到Cookie文件中。

在用localStorage时，得先判断该浏览器是否支持它

if(!window.localStorage){

alert(“浏览器不支持localStorage”);

}else{

var stortage = window.storage;

var data = {“name”:”li ”,”age ”:”25”};

storage.setItem(“data”,data);//设置的时候

}

取得时候：storage.getItem(storage.getItem(“data”));

在存放sessionStorage时也一样。localStorage比sessionStorage存放数据的时间要长的多。

对于 ;(function（$,undefined）{ })(jQuery)的使用

1.对于function前面的分号，是为了防止多个文件压缩合并时，以为其他文件最后一行语句没加分号，而引起合并后的语法错误。

2.一般js库都采用这种自执行函数来保护内部变量。

3.使用undefined的原因：

（1）因为undefined是window的属性，声明为局部变量之后，在函数中如果再有变量与undefined做比较的话，程序就可以不用搜索undefined到window，可以提高程序的性能。

（2）undefined在版本较旧的浏览器是不被支持的，直接使用会报错，js框架就要考虑到兼容性问题，故增加一个形参undefined

4. $实参：$是jQuery的简写，很多方法和类库也使用$，这里$接收 ，也是为了避免$变量冲突，保证插件可以正常运行。

判断元素是否绑定了事件:

var $events = $(“#foo”).data(“events”);

if( $evnets && $events[“click”]){

//写自己的逻辑

}