javascript笔记

**1. 定义：它叫什么？**

Javascript（脚本语言）

知识扩充：

JavaScript 可以出现在 HTML 的任意地方。使用标记<script>…</script>，你可以在 HTML 文档的任意地方插入 JavaScript，甚至在<HTML>之前插入也不成问题。不过如果要在声明框架的网页（框架网页）中插入，就一定要在<frameset>之前插入，否则不会运行。

**2. 它用来干什么？**

最初用来进行数据的验证，出现scriptease（1992年）（是一个脚本，）

1. 数据的验证

djfkds 提交——> 判断——> 返回信息

1. 将动态的内容写入到网页中（比如：轮播图）
2. 对事件进行响应（ondblclick[双击] ,onclick【单击】）、
3. 可以读写html内容，

<script>

document.write(“<div></div>”);

</script>

以上只是一个例子，可以这么做

e) 可以监测浏览器（）

f) 创建cookie(储存在用户本地终端上的数据)(保存登录记录)

cookie可以设置它的生存周期

g)模拟动画

3它是什么，它的特点

它是 基于对象 和 事件驱动 的 解释性（c语言 它是编译，可执行文件，） 松散型 语言

基于对象：面向对象开发的语言

事件驱动：鼠标事件（7个），键盘事件（3个）

解释性：它是不需要编译，它是直接在浏览器中**边解释边执行**（浏览器是JavaScript的宿主环境）

宿主环境就是程序的运行环境

松散型：数据 你能接触到的所有东西（数字，字，……）

C语言：(结构严谨) 每个语句必须有分号

int a = 1;

char c = ‘a’;

javascript（结构松散）可以不加分号，但是养成加封号的习惯

var a = 10

var c = ‘a’

4.基础语法部分不存在兼容性

5.**Javacsript的组成部分（三大部分）**

ECMAscript语法

数据类型，运算符，流程控制，函数，面向对象

BOM(浏览器对象模型)

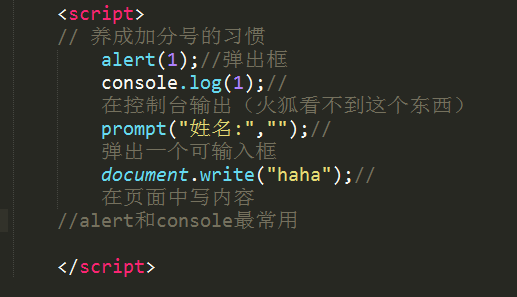
DOM(文档对象模型)

6.js难点：各个浏览器的兼容性，语法的松散，自由性

7.**js基础语法**

1.**javascript输出工具**

<script>



// 养成加分号的习惯

alert(1);//弹出框

console.log(1);//在控制台输出（火狐看不到这个东西）

prompt("姓名:","");//弹出一个可输入框

document.write("haha");//在页面中写内容

//alert和console最常用

</script>

**8.javaScript放置引用**

1.外部方式

<script src = “1.js”></script>

2.嵌入样式

<script><script>

3.超链接和重定向，href，form（action）,

必须加JavaScript的冒号

<a href="javascript:alert(‘我是超链接’);">哈哈</a>

<!-- 重定向 -->

<form action="javascript:alert('我是表单')" method = "get">

<input type="text">

<input type="submit">

</form>

4.在元素内写点击事件，后面跟代码

eg：<div style = "width:200px;height:200px;background:red;" onclick = "alert(1)"></div>

**示例代码**

<body>

<a href="javascript:alert(‘我是超链接’);">哈哈</a>

<!-- 重定向 -->

<form action="javascript:alert('我是表单')" method = "get">

<input type="text">

<input type="submit">

</form>

<div style = "width:200px;height:200px;background:red;" onclick = "alert(1)"></div>

<!-- onclick是鼠标点击事件 -->

</body>

****

**9.script标签不能嵌套script标签，script标签引入外部js文件时，内部不要写代码**

10.**JavaScript注释：**

单行注释 //

多行注释 /\* \*/

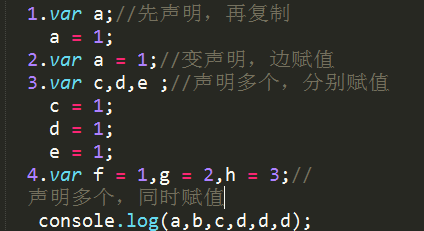
**11.javascript变量：**

undefined未定义

用var定义了一个变量，但是没有给他赋值，就是undefined

var a;等价于a=undefined;

12.声明变量的方式



13学会使用控制台，找错误

**14写了一个变量，直接给他赋值，没有用关键字声明，则它是全局变量**

**15. a = 1;**

**a = 2;**

**2会把1覆盖，最终a = 2;**

**16变量的命名规范：**

1.区分大小写

2.变量名必须以字母，下划线，$开头

Eg:var $$;

var \_zm;

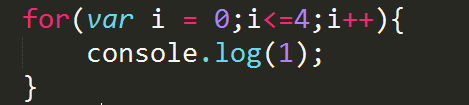
var z\_m556;

3.不能用关键字（var，new,do,else等）或者保留字（abstract,enum,int,short等）命名

保留字：指在高级语言中已经定义过的字，使用者不能再将这些字作为变量名或者过程名使用。一般出现这种情况可能由于考虑扩展性。

4. javascript自己的命名习惯

驼峰命名法：getElementById 第一个单词的首字母小写，其他单词的首字母大写（常用在对象）

首字母大写：Object 只有首字母大写，其他字母都是小写（常用在对象中）

5.命名一定要有意义

6.每句代码后边加上分号

17.声明变量注意的问题：

1覆盖已有变量（仅限于变量的覆盖）

18.数据类型

——1.初始类型（基类型）

number （只要是数值就是number型的） eg: 1 2 3 1.2

在javascript中没有字符类型，一个字符或者一串字符都是字符串

|  |  |  |
| --- | --- | --- |
| 类型 | 可取的值 | **typeof值** |
| string(字符串) | ‘c’,’12345’,’shihaibo’ | string |
| number(数值类型) | 1 2 3 1.2 | number |
| boolean(布尔值) | true，false | boolean |
| undefined | undefined | undefined |
| null | null | **object** |

——2.引用类型

object 对象（值得研究）

function 函数

Array 数组

json(节省)对象的另一种方式

19.内存的结构 栈 堆 代码块区 静态区

代码块区

堆区

栈区

静态区

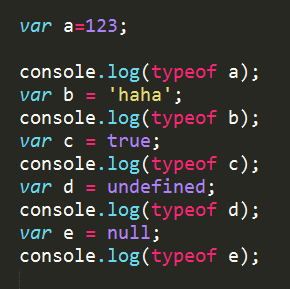
简单的在栈中，访问速度快，容量有限

复杂的在堆中，访问速度慢，容量无限

数字，布尔，undefined在栈中，string ,null,object,函数，数组都是在堆中的

20.typeof 操作符 是用来**检测数据类型**的

只要是加引号的都是字符串类型（在sublime中变为黄色，这个不是固定的，因为sublime的颜色可以调）



21.string 类型（存在于堆中）

—— 用单双引号来说明，他所包围的值都是字符串

—— 引号必须成对出现，不能交叉使用，可以嵌套使用（eg:’””’）

转义字符：

——\n 换行

——\t 制表符

——\b 空格

——\r 回车

——\’ ‘

22. 最大值Number.MAX\_VALUE

 JavaScript 中可表示的最大的数。它的近似值为 1.7976931348623157 x 10308。

最小值Number.MIN\_VALUE

 JavaScript 中可表示的最小的数（接近 0 ，但不是负数）。它的近似值为 5 x 10-324。

在script中输入

C:\Users\Administrator\Desktop\2016-08-15_154843.png

在控制台输出：

C:\Users\Administrator\Desktop\2016-08-15_154908.png

23.null是一个空对象，相当于一个占位符

24.undefined 声明了，但是没有赋值(声明了变量，但没有给他赋值)

25.**JavaScript运算符**

**——算数运算符**

+ - \* / % ++ --

eg1:加法运算符

var a = 3;

var b = 2;

var c = a + b;

console.log(a,b,c);

var a = '3';

var b = '2';

var c = a + b;//当有字符串时，加号表示连接

console.log(a,b,c);

var a = '3';

var b = 2;

var c = a + b;//当有字符串时，加号表示连接

console.log(a,b,c);

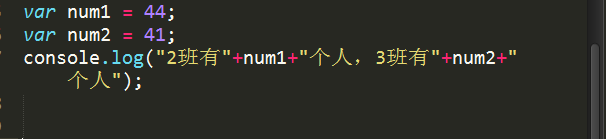
var a = '3';

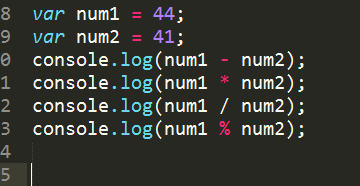
var b = 2;

var c = 'a' + 'b';

console.log(a,b,c);

eg2:用加号连接字符串

eg3.- / \* %



a++ ++a

自增运算符

a++先使用，后加1，++a先加1，后使用

Eg4. ++ --

var a = 1;

a++; //a = a+1;

++a; //a = a+1

console.log(a);

26.number（注意八进制，十六进制和十进制的区别）

<script>

console.log(123);

console.log(0123);

// 八进制的123

// ==1 \* 8的二次 + 2 \* 8的一次 +3 \*8的零次

//64+16+3=83

console.log(0x123);

// 十六进制的123

// 291==1 \* 16的二次 + 2 \* 16的一次 + 3\*16的零次

</script>

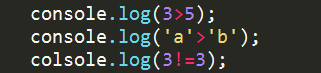
27.关系运算符（得到的结果是true/false）

C:\Users\Administrator\Desktop\2016-08-15_165424.png

= 赋值运算符，不是比较运算符

== （不加数值类型的判断） ===恒等于（连数值类型一起判断）

eg：



两对判断： 1. !=—— == 2. === —— !==

注意：

1.都是数值，正常比较

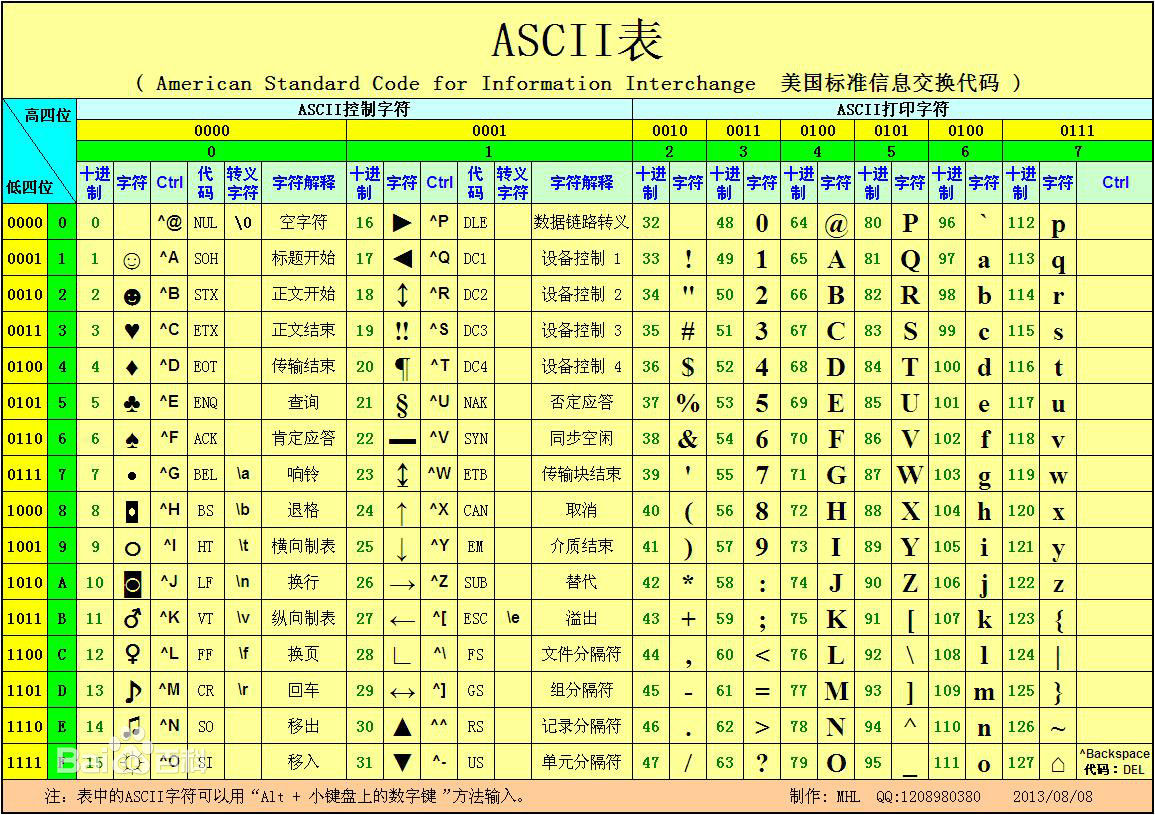
2.如果两个都是字符串进行比较，从他们的第一个开始先转换为ASCII码值，开始依次比较，直到比较出大小

3.**当一个字符串，另一个是数值时，把字符串尝试转换为数值类型，然后进行比较，如果不能转换为数值类型，返回假**

4.**undefined==null为真**

5.如果一个数值和布尔值进行比较，会把布尔值转换为数值再进行比较。true为1，false为0

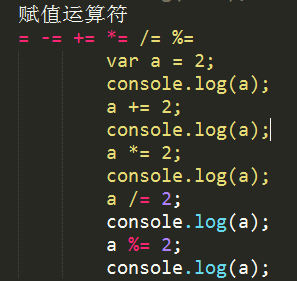
28.ACSII表



29.赋值运算符：（6个）

var a ;

a = 2;



30.逻辑运算符(**注意：得到的结果不是布尔值**)

与&& 或|| 非!

例如：alert(4>3); //弹出true

alert(4&&3);//弹出3

alert(4||3);//弹出4

|  |  |  |  |
| --- | --- | --- | --- |
|  | 左 | 右 | 值 |
| **&&** | true | ture | ture |
| true | false | false |
| false | trure | false |
| false | false | false |
| **||** | true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

**!** 你真我假 你假我真

**31.短路原则**

&& 如果两边左边是假的，则右面短路

|| 如果两边左边是真的，则右边短路

eg:

var a = 2;

console.log((a<0)&&(a++<2));

console.log(a);//结果为2，短路原则，

eg:

var a = 1;

console.log((a>0)&&(a++<2));

console.log(a);

32.**一元运算符（**只操作一个数据

**）**

**typeof , +（正） , -（负） , delete（删除对象）, new（新建对象） , ++, --**

33. **逻辑运算符注意：**

underfined false

null false

boolean 就是本身的值

number 除了0之外都是真的

string 除了空字符串以外都是真的

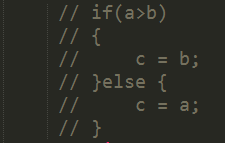
对象 真的

8-16

1. 三元运算符

C:\Users\Administrator\Desktop\2016-08-16_085406.png

等价于

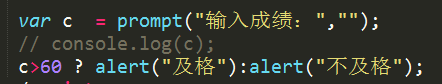


2. 输入框的值会在控制台输出

var c = prompt("输入成绩：","");

console.log(c);

eg:输入成绩，大于等于60的，弹出及格，小于60的弹出不及格



3.特殊运算符

1.特殊运算符：（）

第一：函数调用，函数声明

第二：优先运算

2.特殊运算符：,

用于声明多个变量

4.运算符总结：

1算术运算符：

+ - \* / % ++ --

+ 如果有字符串，则作为连接符

a++ 先使用在加1

++a 先加1再使用

2关系运算符

< > <= >= == != === !==

==是不加数据类型进行比较

===加上数据类型进行比较

3逻辑运算符

&&与 ||或 非!

短路问题

如果&&左边是假的，则右边的不执行

如果||左边的是真的，则右边的不执行

4一元运算符

+ - typeof new del !

5三元运算符

? :

6特殊运算符

1.括号（）用于函数调用，函数声明

2.逗号 ，用于声明多个变量

7.赋值运算符

= += -= \*= /= %=

5.流程控制

——流程：程序代码的执行顺序

三大流程控制

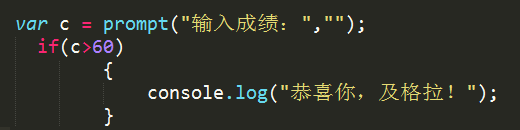
1.顺序结构

从上到下，一行行解释执行

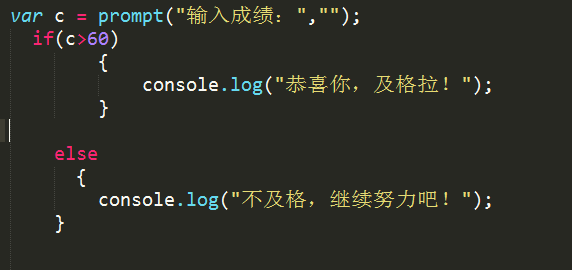
2.选择（分支/条件）结构

有选择的执行某一行语句

A 单路分支 只有if



B 双路分支 既有if又有else，肯定有一条语句执行

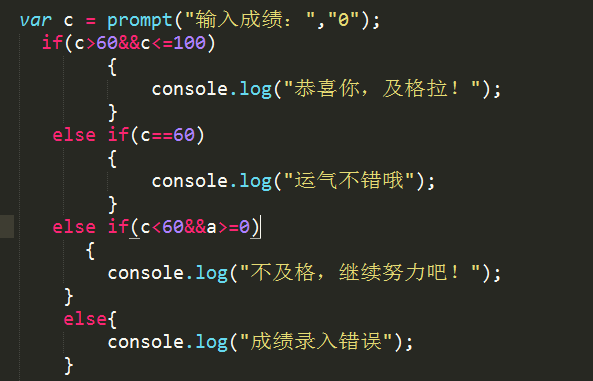


C 多路分支



什么都不输入，默认为0

修改之后的，比较完整的



写代码的时候还要尽可能的考虑可能发生的情况，并对他进行处理

D if的嵌套



Eg:



E switch语句

var a =1;

switch(a){

case 1:表达式1;

break;//跳出switch

case 2:表达式2;

break;

case 3:表达式3;

break;

case 4:表达式4;

break;

default:表达式;//全部不满足上面的条件，执行default

}

例子：输入两个数字，进行算术运算

prompt返回数据类型是字符串。

parseFloat() 函数可解析一个字符串，并返回一个浮点数。

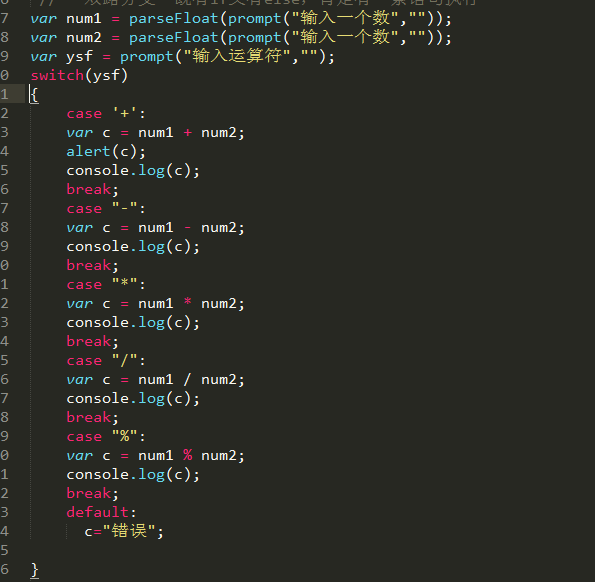
该函数指定字符串中的首个字符是否是数字。如果是，则对字符串进行解析，直到到达数字的末端为止，然后以数字返回该数字，而不是作为字符串。

如果参数字符串的第一个字符不能被解析成为数字，则 parseFloat 返回 NaN。

parseInt(); 将字符串转为整型数字，如果字符串中的首个字符是数字，则对字符串进行解析，直到到达数字的末端为止；如果字符串的第一个字符不能被解析为数字，返回NaN

Number(); 也是如此

示例代码：



**判断某种范围的时候用if语句，判断单个值的时候用switch语句**

3.循环结构（重复的执行同一个代码块）

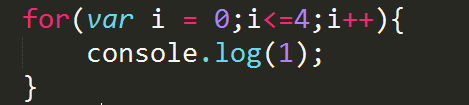
——for循环

语法：

for(初始值;终止条件;步进值)

{

}



**break与continue：**

break跳出整个循环，continue跳出当前的循环，执行下一次循环

例1

for(var i = 0; i<=5;i++)

{

if(i==3)

{

continue;//跳出本次循环，执行结果为：0,1,2,4,5

}

console.log(i);

}

例2：

for(var i = 0; i<=5;i++)

{

if(i==3)

{

break;//跳出整个循环，执行结果为0,1,2

}

console.log(i);

}

——for循环的嵌套

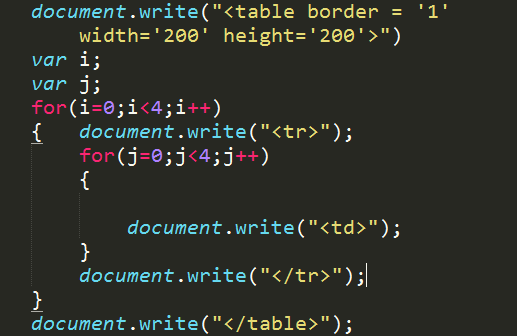
For()

{

For()

}

例如：



示例代码：

document.write("<table border = '1' width='200' height='200'>")

var i;

var j;

for(i=0;i<4;i++)

{ document.write("<tr>");

for(j=0;j<4;j++)

{

document.write("<td>");

}

document.write("</tr>");

}

document.write("</table>");

document.write("<pre>789\t89087</pre>")

**重点：循环，函数，对象**

eg:1-100之间的奇数

——while循环（引用范围比for循环更广）

语法：

while(终止条件)

{

}

while的终止条件可以判断（for与while的区别）

eg:求0~5的和

var i=1;

var sum = 0;

while(i<=5)

{

sum=sum+i;

i++;

}

console.log(sum);

——do…while循环（至少执行一次）

语法：

do{

}while()

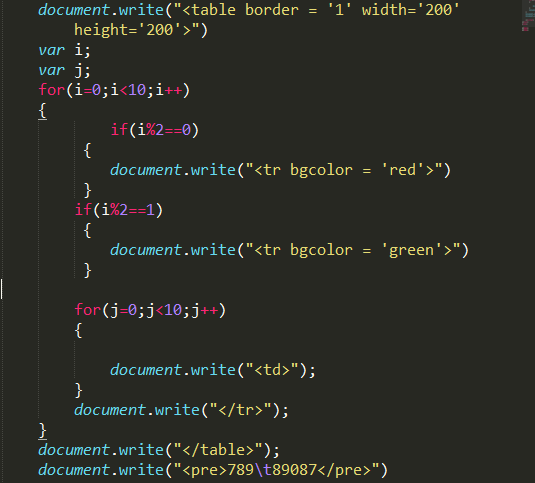
**while与do…while的区别：**

while先判断后执行，do…while先执行，后判断

6.**for与while的区别：**

当知道执行次数的时候一般用for  
当条件循环时一般用while，知道循环多少次，只知道范围

7.例子：用js输入表格，并且变色



示例代码：

document.write("<table border = '1' width='200' height='200'>")

var i;

var j;

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

{

if(i%2==0)

{

document.write("<tr bgcolor = 'red'>")

}

else//if(i%2==1)

{

document.write("<tr bgcolor = 'green'>")

}

for(j=0;j<10;j++)

{

document.write("<td>");

}

document.write("</tr>");

}

document.write("</table>");

8.例子： 操场是有一百多个人，三排余1人，四排余2人，五排与3人，问：总共多少人

var i;

for(i=101;i<200;i++)

{

// console.log(i);

if(i%3==1&&i%4==2&&i%5==3)

{

console.log(i);

}

}

9.把<script></script>中的内容显示在网页中，使用document.write(“”);

10.

// 九九乘法表

for(var i = 1; i < 10; i++)

{

for(var j = 1; j <= i;j++ )

{

document.write(j+"\*"+i+"="+j\*i);

document.write('&nbsp');

}

document.write("<br>");

}

11密码输入的问题

第一次输入xxxxxx，第二次输入xxxxxx，第三次输入xxxxxx，登录成功，否则登录次数超过3次，密码锁定

<script>

var i = 1;

do{

var password = prompt("请第"+i+"次输入密码","");

if(password == "123456")

{

alert("登录成功");

break;

}

if(i==3)

{

alert("登录次数超过3次，锁定");

}

i++;

}while(i<4)

</script>

tr不是table的直接子元素，而是tbody的直接子元素

**8-17**

下午：

1.知识总结

body 文档层

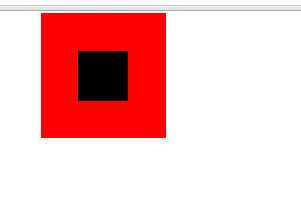
块 四层：背景，文档，浮动，定位

图片上面的东西用定位做

文字有文字属性

盒子 外间距 ，内间距，边框，content

**2.绝对居中的布局方式**



示例代码：



3.函数（注意函数的作用域）：能实现某些功能的代码块，可以重复的调用（将功能做一个包装）

一、函数的特点：

使程序更加的简洁

逻辑更加条理

调用更加方便

维护更加容易

二、**函数的创建**

—— 1.通过function关键字来创建（**会被优先解析到内容中**

function函数名()

{

函数体;

}

—— 2.字面量（直接量）的形式来创建（**没有函数名**）

把整个函数保存到变量中,调用方式和之前一样(注意：覆盖的问题)

var aa = function(){

console.log(1)

}

// aa = 2;aa被覆盖掉了

aa();

—— 3.通过new的方式（对象的方式）来创建（明白函数也是一种对象类型）——不常用

语法：var aa = new Function([参数1],[参数2],[参数3],…”函数体”);\

Eg:

var a = 1;

var fun = new Function(a,alert(a));

**注意：函数覆盖问题**

function aa(){

console.log("直接使用function");

}

var aa = function(){

console.log("使用字面量声明");

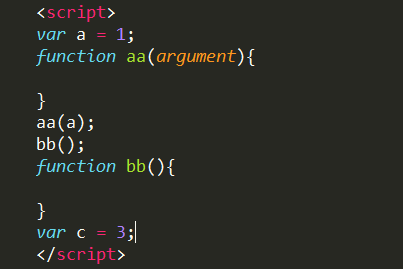
}

aa();//弹出使用字面量声明

在遇到script标签的时候，解析器会把所有的函数（不包含字面量形式的函数）解析到内存中，然后在一步步执行的过程中遇到字面量声明的函数时，才会解析它，所以会覆盖掉之前的函数

字面量声明，不优先解析字面量，而把它保存到变量中，所以会覆盖上面的

说明：



遇到以上的代码，解析器会首先把函数function aa(){},function bb(){},解析出来，然后把var a;var c;解析出来，然后再往下执行

函数例子：



示例代码：

biaoge(4,4,'yellow','blue');

function biaoge(m,n,color1,color2){

document.write("<table border = '3' width='200' height='200'>");

for(var i = 1;i<=m;i++)

{ if(i%2==1){

document.write("<tr bgcolor = "+color1+">")

}

else{

document.write("<tr bgcolor = "+color2+">")

}

for(var j= 1;j<=n;j++)

{

document.write("<td></td>");

}

document.write("</tr>");

}

}

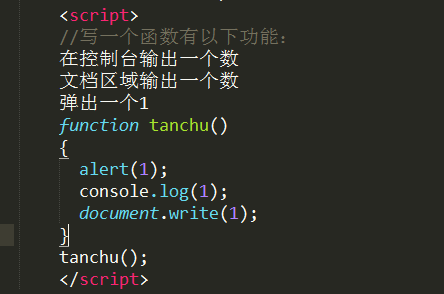
document.write("</table>")

**二、函数调用方法：**

——1.用（）来调用（用于具有函数名的调用）

函数名（）;

3.Eg:



**8-18**

一、**图片在div中居中的方法**

情况一：在div中添加img

在父元素中添加position:relative;

在子元素中添加

position:absolute;

left:50%;//将图片的左边移动到div的中间

margin-left:-674px;//然后在往左移动图片本身宽度的一半

情况二：在div中设置背景图片（做中国移动banner右边的六个小图标）

background:url( );

background-repeat: no-repeat;

background-position: center;

二、复习:

1.变量：把数据保存起来的一个容器

这个容器中的数据随时可以更换

2.数据类型：

初始类型

数值类型 number

字符串类型 string

null object

undefined underfined

布尔 boolean

引用类型

对象 object

字符串，函数，数组本质上是对象

注意：1.单引号双引号的用法

2.自增自减的用法

3.&& ||的短路问题（还有给变量赋值的问题var a = 1&&2;var a = 0&&2;）

3.运算符：

算数运算符

关系运算符

逻辑运算符

**赋值运算符（如果是数字，+=得到的是数值，*如果是字符串，+=得到的是字符串*）**

一元运算符

三元运算符(双路分支)

特殊运算符

4.流程控制

顺序，选择（分支），循环

选择：

单路分支

if(){

}

双路

If(){

}else{

}

多路

if(){

}else if{

}else{

}

Switch(){

case 1:表达式;break;

case 2:表达式;break;

case 3:表达式;break;

default:出错处理;

}

循环 break;（跳出整个循环）

continue;(跳出本次循环，执行下一次循环)

For（）

嵌套for

while(){

}

do{

}while()

5.**函数**

**声明方式三种：**

——function a(){

}

——var a = function{ 先解析了var a ;没有把函数体解析出来

}

——var a = new Function()

三、**函数的调用方法**：

——函数名（）；

用字面量声明的函数为**匿名函数**

——自调用【有两种方式】（用于匿名函数，匿名函数还可以用引用变量来调用）

第一种自调用

**(function(){**

**})()**

**第二种自调用：**

**!function(){**

**}()**

**——**事件触发

**<script>**

**function aa(){**

**alert("我想上厕所");**

**}**

**</script>**

**<style>**

**div{**

**width:100px;**

**height:100px;**

**background:red;**

**}**

**</style>**

**</head>**

**<body>**

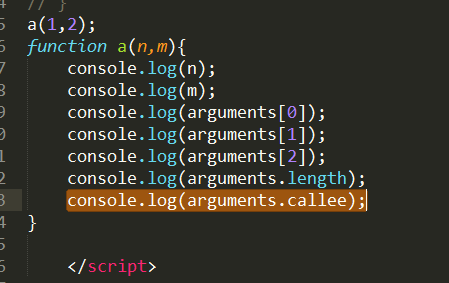
**<div onclick = "aa()"></div>**

**</body>**

四、**函数的参数**（最多25个）

1.实参与形参一一对应

2.arguments对象是比较特别的一个对象，实际上是当前函数的一个内置属性，arguments非常类似array，但实际上又不是一个array实例。保存函数参数的一些数据，



arguments.length; 返回参数的个数，参数的长度

3. console.log(**arguments.callee**);在控制台输出整个函数本身

五、**模拟函数的重载** （javascript不会实现函数重载，会模拟函数重载）

当函数名一样，传入的参数个数及参数类型不一样的时候，执行不同的函数体。

aa(1); 弹出一个数值

aa(1,2); 弹出两个数值

aa("haha"); 弹出两个字符串

模拟函数的例子：

判断参数的个数 arguments.length

判断参数的类型 typeof



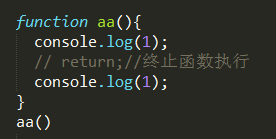
六、**函数返回值**(一个函数只能有一个返回值)

1.关键字return

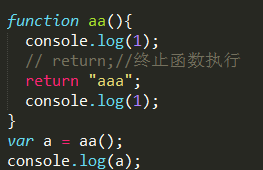
返回：两层含义：返回，终止函数执行，返回return后面的数据

返回值可以是任意一个类型

2.如果没有返回值，会默认为undefined



3.示例代码：



function aa(){

console.log(1);

// return;//终止函数执行

return "aaa";

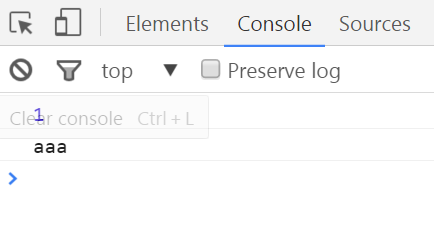
console.log(1);

}

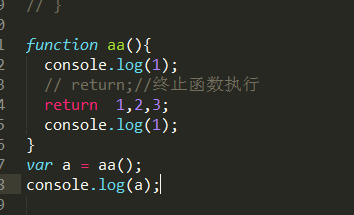
var a = aa();

console.log(a);

输出结果：



4.一个函数只能有一个返回值



输出结果：

1

3（3会把1和2覆盖）

5.一个语句是否结束看的是分号

6一个函数中可以有多条return语句

Eg: function diff(iNum1, iNum2) {

if (iNum1 > iNum2) {

return iNum1 - iNum2;

} else {

return iNum2 - iNum1;

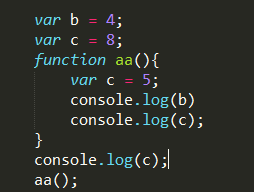
}

}

七、**变量的作用域**

1.**函数内部可以读到函数外部的值，但是在函数外部读不到函数内部的值**

例:1：



输出结果：

8

4

5

例2：**作用域链**的例子

var a = 4;

function aa(){

var a = 5;

function bb(){

var a = 6;

console.log(a);

}

bb()//调用函数bb

}

aa()

输出结果：6

八、**环境问题**

宿主环境（浏览器）

执行环境：（决定了变量和函数的访问权限）

1.全局环境：整个页面

2.函数：一个函数就是一个环境

3.作用域：一段代码的作用范围

4.**作用域链**：一个函数嵌套另一个函数时，他们的作用域形成了一条链子，就是作用域链（一层一层往上）

Eg:

var a = 4;

function aa(){

var a = 5;

function bb(){

var a = 6;

console.log(a);

}

bb()//调用函数bb

}

aa()

5**.eval**  解析执行js代码

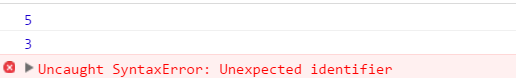
eval(“alert(1)”);

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

例子：

C:\Users\Administrator\Desktop\2016-09-17_070551.png

输出结果：



九、**回调函数**：将一个函数作为参数，传入到另一个函数中使用

例：

<script>

function aa(fun){ //fun是用来接收bb的

fun(); //fun();==>bb();执行bb

}

function bb(){

console.log(1);

}

aa(bb); //将bb函数作为参数传入aa函数中

</script>

例：加法，减法，乘法，除法

function add(i,j){

return i+j;

}

function sub(i,j){

return i-j;

}

function mul(i,j){

return i\*j;

}

function div(i,j){

return i/j;

}

function jisuan(n,m,fun){

var result = fun(n,m);

console.log(result);

}

var num1 = 8;

var num2 = 4;

jisuan(num1,num2,add);

jisuan(num1,num2,sub);

jisuan(num1,num2,mul);

jisuan(num1,num2,div);

十、**递归函数**：自己调用自己（递归函数必须有一个初始条件）面试必考题

例：求5的阶乘

第一种解法：

function jiecheng(n)

{

if(n==1||n==0){

return 1;

}

else{

return n\*jiecheng(n-1);

}

}

var result = jiecheng(5);

console.log(result);

第二种解法：

function jiecheng(a)

{

if(a==1||a==0)

{

return 1;

}

else{

return a\*arguments.callee(a-1);

}

}

alert(jiecheng(5))

十一、**闭包**（函数可以使用函数之外定义的变量）

岳英俊老师讲解的闭包：

闭包

// 需要把局部变量保存下来的时候用到闭包

// 闭包的形成，1.一个函数内部嵌套另一个函数，2.当里层函数引用外部函数变量的时候，3.并且当我们调用里层函数的时候，闭包形成

2016-9-4 理解的闭包：

在js中，函数内部可以访问函数外部的变量（全局变量），在函数的外部不能访问函数内部的变量（局部变量）。但有时需要在函数外部访问函数内部的变量（局部变量），所以需要在函数的内部再定义一个函数，此时内部的函数就可以访问外部函数的变量了

**函数嵌套时**内部函数调用外部函数的局部变量

Javascript语言的特殊之处，就在于函数内部可以直接读取全局变量。

另一方面，在函数外部自然无法读取函数内的局部变量。

闭包就是能够读取其他函数内部变量的函数。

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

十二**、内置顶层函数**

1.——confirm

var a = confirm("你确定要继续吗？");

if(a){

alert("请继续");

}else{

alert("真遗憾");

}

2.——顶层：作用域是全局的，无论在那个地方都可以使用

**九个内置顶层函数：**

**1.escape**

**2.unescape**

例子：

var str = "原晓晓";

document.write(escape(str)); //编码

document.write("<br>");

document.write(unescape("%u539F%u6653%u6653")); //反编码

3**.Number();**将任何类型的数据转换为数值类型,如果不能转为数值，则返回NaN

console.log(Number(false));

console.log(Number(true));

console.log(Number(" 123 "));

console.log(Number("undefined"));//输出NaN

console.log(Number(0123));//把八进制转为十进制输出来

console.log(Number("45o")); //返回NaN

console.log(Number("o 45")); //返回NaN

注意：Number();只能转换标准的数值类型，与parseInt（）不同

4.**String();**把任何类型的数据转换为字符串

console.log(String(false));

console.log(String(true));

console.log(String(" 123 "));

console.log(String("undefined"));

5.**Boolean();**把任何类型的数据转换为真假值

console.log(Boolean(false));

console.log(Boolean(true));

console.log(Boolean(" 123 "));

console.log(Boolean("undefined"));

6.**parseFloat(**);把数字转为浮点型

console.log(parseFloat(12));

console.log(parseFloat(true));//返回NaN

console.log(parseFloat(12.5));

console.log(parseFloat("undefined"));

console.log(parseFloat("5"));

console.log(parseFloat("1fdf"));//返回1

console.log(parseFloat("fdf1"));//返回NaN

注意：parseFloat();可以转换不标准的数值

7.**isNaN();**是不是一个数，是返回false，不是返回true

console.log(isNaN(12));

console.log(isNaN(false));

console.log(isNaN(12.5));

console.log(isNaN("undefined"));

8.**eval;** Eval如果传入的字符串可以解析的js代码，则解析执行

如果不能解析，则报错

9.**parseInt();**把一个数字转换为整型

console.log(parseInt(12));

console.log(parseInt(12.5));

注意：

如果parseInt();有两个参数，则第二个表示进制（权），第一个是数值，

然后一位一位的解析，如果从左往右第一位小于权，第二位也小于权，则全部解析转换，如果从左往右第一位小于权，第二位大于权，则只解析第一位，第二位不管，如果从左往右第一位大于权，则返回NaN

例如：

console.log(parseInt(11,2)); 返回11

console.log(parseInt(12,2)); 返回1

console.log(parseInt(21,2)); 返回NaN

**8-19**

十三、**数组**

1.**数组声明的方法**

——第一种：隐式声明

var arr = [1,2,3,4,5];

例如：

var arr1 = ["",,123,"a",function aa(){}];

//数组中可存放任何类型

console.log(arr1);

——第二种：通过对象声明

var arr1= new Array();

例如：4

var arr1 = new Array(1,2,3);

console.log(arr1);

——第三种

var arr = [];

arr[0] = 1;

arr[1] = 2;

2.**数组值的调用（用下标的方式调用）**

console.log(arr[0]);

console.log(arr[1]);

console.log(arr[2]);

console.log(arr[3]);

console.log(arr[4]);

例子：**数组的遍历(**一共有三种办法1.for循环，2.while循环.3for in 循环**)**

var arr = [1,2,3,4,5,6,7,8,9];

function xunhuan(){

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

{

console.log(arr[i]);

}

}

xunhuan();

3**.for in 循环**，**用来遍历数组和对象**

for(var i in arr){

}

eg:

for(var i = 0 in arr)

{

document.write(arr[i]);

document.write("<br/>");

}

4.一维数组

//一维数组

var arr = [1,2,3,4,5];

arr[0] = 1;

arr[1] = 2;

arr[2] = 2;

arr[3] = 4;

arr[4] = 5;

**例：一维数组求最大值**

var arr = [1,8,1,4,5];

var arrmax=arr[0];

for(var i = 1;i<arr.length;i++)

{

if(arrmax<arr[i])

{

arrmax=arr[i];

}

}

console.log(arrmax);

5.二维数组

var arr = [[1,2,2,5,5,6,5,5],

[21,233,4,50,34],

[58,58,598]];

arr1[0] = [1,2];

arr1[0][0] = 1;

arr1[0][1] = 2;

arr1[1] = [21,233,4,50];

arr1[1][0] = 21;

arr1[1][1] = 233;

arr1[1][2] = 4;

arr1[1][3] = 50;

arr1[2] = [58,58,598];

**例：求二维数组中的最大值**

**var arr = [[1,2,2,5,5,6,5,5],**

**[21,233,4,50,34],**

**[58,58,598]];**

**var max=arr[0][0];**

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

**{**

**for(var j = 0;j<arr.length;j++)**

**{**

**if(max<arr[i][j])**

**{**

**max=arr[i][j];**

**}**

**}**

**}**

**alert(max);**

**求二维数组中最长的元素**

var arr = [[1,2,2,5,5,6,5,5],

[21,233,4,50,34],

[58,58,598]];

var max=arr[0];

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

{

if(max.length<arr[i].length)

{

max=arr[i];

}

}

alert(max);

**例：**

**删除数组中的空值**

var arr = [2,,"aa",true,,null,[1,2]];

var newarr = [];

var j = 0;

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

{

if(arr[i]!==undefined){

newarr[j]=arr[i];

j++;

}

}

document.write(newarr);

6.push添加数组元素

var arr = [];

arr.push(1);

arr.push(2);

arr.push(2);

arr.push(1);

alert(arr);7.封装

7.遍历一个既有数组又有数值元素的数组(封装)

var arr1= [1,[5,2,1],4,[6,7],5];

function haha(arr)

{

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

{

if(typeof arr[i]=='number')

{

document.write(arr[i]+"&nbsp;");

}

else

{

for(var j = 0;j<arr[i].length;j++)

{

document.write(arr[i][j]+"&nbsp;");

}

}

}

}

haha(arr1);

8.**数组排序**

var arr = [8,9,6,4,2,8,3];

var t;

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

{

for(var j = i+1;j<arr.length;j++)

{

if(arr[i]>arr[j])

{

t = arr[i];

arr[i] = arr[j];

arr[j] = t;

}

}

}

document.write(arr);

**8-22**

1.dd是dl的直接子元素

2.网页中的文字最小12像素， 正文部分一般为14p像素

3.3D视图的问题

打开火狐浏览器，单击菜单栏——开发者——Tilt，可以旋转放大缩小

**知识点的复习**

一、js的三大部分：

ECMA语法 BOM DOM

二、变量：可以变化的量，存放在内存中的栈区（内存分为四部分：栈区，堆区，代价块区，静态区）

栈的长度是有限的，存放的东西是有范围的，访问速度快

堆区，长度无限，访问速度慢

三、数据类型

——初始类型（简单类型）

数值类型 number “number”

字符串 string “”

布尔类型

undefined 定义了变量，没有赋值

null 只是一个占位符 “object”

检测数据的类型typeof

——引用类型（复杂类型）

object “object”

function “function”

array,json(节省)【对象里面的一种类型】 “object”

四、运算符

例一、

var a = 68;

var b = “34”;

alert(a-b); 68-“34” 隐式转换 Number

例二、

var a = 68;

var b = “34px”; //Number(“34px”)

alert(a-b); 依然是隐式转换 Number，结果为NaN

**五、隐式转换**（两边的类型不一样）

**六、九大内置顶层函数**

escape/unescape

**Boolean(); 为假的情况:**

false

Number 0

String “”

Undefined

null

NaN

Number 适用于任何类型

parseInt 适用于字符串类型（注意两个参数的情况）

两个参数

第一个参数代码数值，第二个参数代表进制

parseFloat； 适用于字符串类型

例子：

var num = “300.89px”;

alert(Number(num)); //NaN,Number只能解析标准的数值

alert(Number(num)); //300

alert(Number(num)); //300.89

1.面试流程：笔试（考的都是一些细节） 面试

**2.关系运算符(注意隐式类型的转换)结果为布尔类型，要么为真，要么为假**

===全等于 ！==全不等于

==只比数值，不比类型

===既比数值又比类型

0 ASCII码值 43，A: 65 ,a :97,

**3.逻辑运算符 用Boolean();进行隐式转换**

**注意：返回值不一定是布尔型**

**例：**

var a = “”;

var b = 56;

var c = 0;

alert(c||b||a); 返回56

三个条件都为假的时候，会把a（最后最后为假）的值返回

三个条件都为真的时候，会把最后为真的值返回

**例：**

var a = “”;

var b = undefined;

var c = 0;

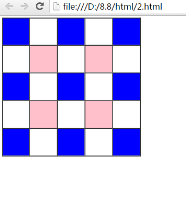
alert(c||a||b); 返回undefined

**七、循环**

1.while和if括号里的表达式都是调用了Boolean();进行隐式转换

下午

**实现五行五列表格交叉换色**



**第一种解法：**

示例代码：

var color = "";

document.write("<table border = '1' width = 800 height = 800>");

for(var i = 1;i<=5;i++)

{

for(var j = 1;j<=5;j++)

{

if(i%2==0 && j%2==0)

{

color = "pink";

}

else if(i%2 == 1 && j%2== 1)

{

color = "blue";

}else{

color = "#fff";

}

document.write("<td bgcolor = "+ color+"></td>")

}

document.write("</tr>");

}

document.write("</table>");

**第二种方式：用字符串连接的方式实现表格的交叉换色**（用+=运算符，注意：tab +=”</table>” 是添加不是加）

var tab = "<table border = 1 width=200 height=200>";

var color = "";

for(var i = 0;i < 5;i++)

{

tab += "<tr>";

for(var j = 0;j < 5;j++)

{

if(i%2 == 0 && j%2 == 0)

{

color = 'red';

}

else if(i%2 == 1 && j%2 == 1)

{

color = "green";

}

else{

color = "#fff";

}

tab += "<td bgcolor = "+color+">1111</td>";

}

tab += "</tr>";

}

tab += "</table>";

document.write(tab);

**八、找错误的方式：**

1.控制台

2.firebug查看元素

添加断点（还未试）

**九、函数定义**(面试问题)

1.基本定义：什么是函数：能够实现某个功能的代码块

用关键字定义

functino aa(){}

2.字面量(直接量)——匿名函数

var aa = function(){}

3.对象的方式定义

var aa = new Function(“函数”);

eg:

var add = new Function("num","return num+20");

var sum = add(122);

console.log(sum);

**十、函数的调用方式（三种）**

1.函数名（）;

2.自调用

第一种自调用——(function(){

})()

第二种自调用——! function(){

}()

3.事件调用（onclick = “alert(1)”）//鼠标点击

onmouseover = “alert(‘移入’)”; //鼠标移入

onmouseout = “alert(‘移出’)” //鼠标移出

**十一、函数参数**

**1.实参的访问方式**

——用形参接收实参

——用arguments对象进行访问

Javascript中每个函数都会有一个arguments对象实例arguments，它引用着函数的实参，可以用数组下标的方式“[]”引用arguments的元素，arguments.length为函数实参的个数，arguments.callee引用函数自身

**访问最后一个参数**arguments[arguments.length-1];

**2.函数返回值**

return 不是每个函数都需要返回值，根据函数的功能来决定需不需要使用返回值

函数的返回值只能有一个

3. function demo(){

var a = 200;

function demo2(){

alert(a); //弹出undefined

var a = 400;

}

demo2();

}

demo();

**十二、闭包**：

——**闭包的条件**

1.发生函数嵌套

2.内部函数调用外部函数的变量

3.内部函数被调用

满足上述三个条件，就形成了闭包

**——闭包的作用：**

1.解决在函数外部访问局部变量的问题

2.可以将局部变量保存下来

例子：这个例子就是闭包问题

function aa(){

var num = 30;

return function(){

return num++;

}

}

var cc = aa();

alert(cc()); //输出30

alert(cc()); //输出31

alert(cc()); //输出32

**alert(cc);//弹出内部函数整体**

**十三、全局变量与局部变量**

**全局变量：**

只要是在函数外面定义的就是全局变量

没有用var声明变量的也是全局变量

**局部变量：**

函数内部定义的变量，函数里面的参数是局部变量

**十四、回调函数：将函数作为参数传给另一个函数使用**

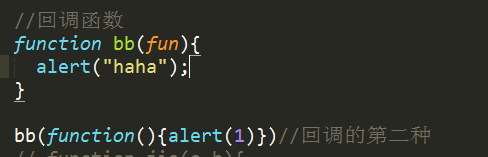
**——回调函数的作用**：在同一时间可以处理很多事情

**——回调函数的两种形式**

——1.函数指针（函数名）

——2.函数定义（将函数的定义作为参数传递进去）

例一：回调的第二种



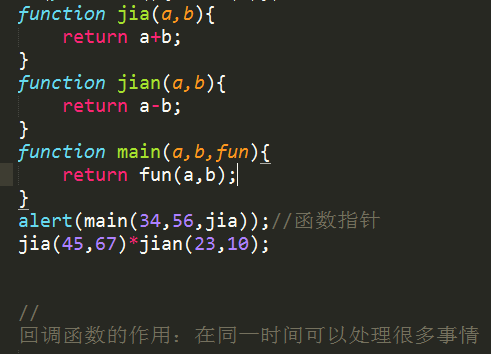
示例代码：

function bb(fun){

}

bb(function(){alert(1)}) //函数定义（将函数的定义整体作为参数）

例二：回调的第二种



示例代码：

function jia(a,b){

return a+b;

}

function jian(a,b){

return a-b;

}

function main(a,b,fun){

return fun(a,b);

}

alert(main(34,56,jia));//函数指针（将函数名作为参数）

jia(45,67)\*jian(23,10);

**8-23**

**一、数组**

（类比变量进行理解）变量：存储数据的容器

1.**概念：**

数组：存储一组或一系列相关数据的容器

**2.数组的定义（两种方式）**

1.var arr = new Array(); 用对象的方式定义

arr[0] = “xxx”;

arr[1] = “yyy”;

2.var arr = [45,25,45,78];

**例子：**

var arr = new Array(5); //5代表数组长度

alert(arr); //弹出…

var arr = new Array("5");//5代表数组元素，数组长度为1

alert(arr);//弹出5

var arr1 = [45,23,45,12];

alert(arr1.length);

**3.访问数组元素（用下标访问）：**

alert(arr[2]);

**4.数组的遍历（循环的3中方式）**

1.for循环

2.for in 除了可以遍历数组还可以遍历对象

3.while循环

**例：遍历数组并求平均值**

var sum = 0;

var average;

var arr = [1,2,3,4,5,6,7,8,9,10];

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

{

sum += arr[i];

average = sum/arr.length;

}

alert(sum);

alert(average);

例： 求一维数组的最大值

var arr = [34,2,56,1,12];

var max = arr[0];

for(var i = 1;i<=arr.length;i++)

{

if(max<arr[i])

{

max = arr[i];

}

}

alert(max);

例：数组排序（冒泡排序）

注意：ctrl+D选中重复的部分

var arr = [1,5,63,4,6,2,7,5];

var arr1 = [4,5,6,88,1,2,5,3];

function arrSort(a,type){

var temp;

type = type || "desc";//默认不传值时，降序排列（如果type传值时，就以传的值为准，如果不传值，就以desc为准）

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

{

for(var j = i+1;j<arr.length;j++)

{

if(type == "desc")

{

if(a[i]<a[j])

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

else if(type == "asc")

{

if(a[i]>a[j])

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

}

return a;

}

arrSort(arr);

document.write(arr);

**5.for循环与for in循环的区别**

1. for循环可以更改初始值和最大值，而for in循环不能更改初始值和最大值

2.用for和for in遍历数组是一样的，i指索引值，arr[i]指数组元素

var arr = [1,4,5,6]

for(var i in arr)

{

alert(arr[i]);//i:下标 arr[i]:元素

}

3.而用for in遍历对象时，i指对象的属性，arr[i]指属性值

**6.浅拷贝（拷贝地址）**

**例：**

var arr1 = [1,2,3,4,5];

var arr2 = [];

arr2 = arr1;//把内存中的地址给了arr2

arr1[1] = 20;

alert(arr1[1]); //输出20

alert(arr2[1]); //输出20

**7. 深拷贝（拷贝数值）**

**例：**

var arr1 = [1,2,3,4,5];

var arr2 = [];

for(var i in arr1)

{

arr2[i] = arr1[i];

}

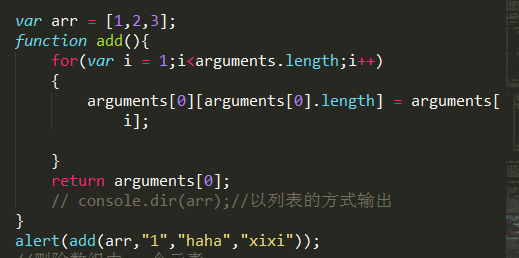
arr1[1] = 20;

alert(arr1[1]); //输出20

alert(arr2[1]); //输出2

8. **写一个函数，实现对任意数组添加任意元素的功能**

arguments在函数一定义就已经产生，主要用来记录函数的参数信息的对象，arguments.length记录参数的个数，也是用下标访问每一个参数，arguments.callee返回整个函数



var arr = [1,2,3];

function add(){

for(var i = 1;i<arguments.length;i++)

{

arguments[0][arguments[0].length] = arguments[i];

}

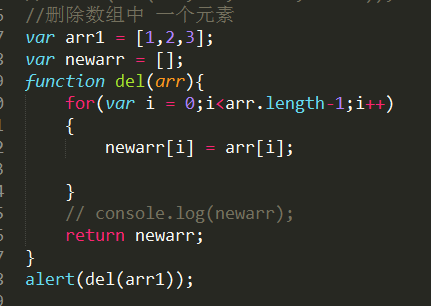
return arguments[0];

// console.dir(arr);//以列表的方式输出

}

alert(add(arr,"1","haha","xixi"));

**9.** **删除数组中 一个元素**

****

**示例代码：**

var arr1 = [1,2,3];

var newarr = [];

function del(arr){

for(var i = 0;i<arr.length-1;i++)

{

newarr[i] = arr[i];

}

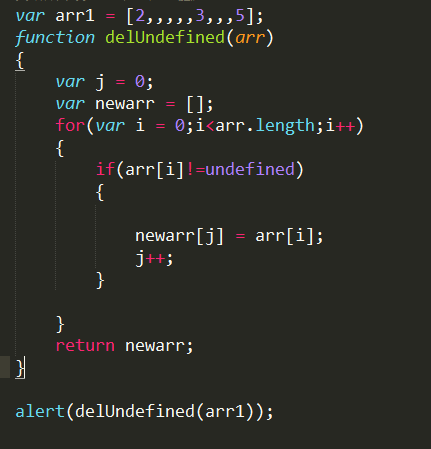
// console.log(newarr);

return newarr;

}

alert(del(arr1));

**10删除数组中的空值**

****

**示例代码：**

var arr1 = [2,,,,,3,,,5];

function delUndefined(arr)

{

var j = 0;

var newarr = [];

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

{

if(arr[i]!=undefined)

{

newarr[j] = arr[i];

j++;

}

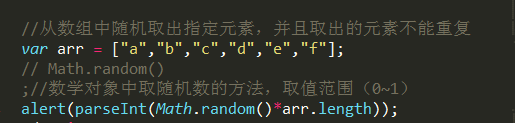
}

return newarr;

}

alert(delUndefined(arr1));

11. Math.random();//数学对象中取随机数的方法，取值范围（0~1）

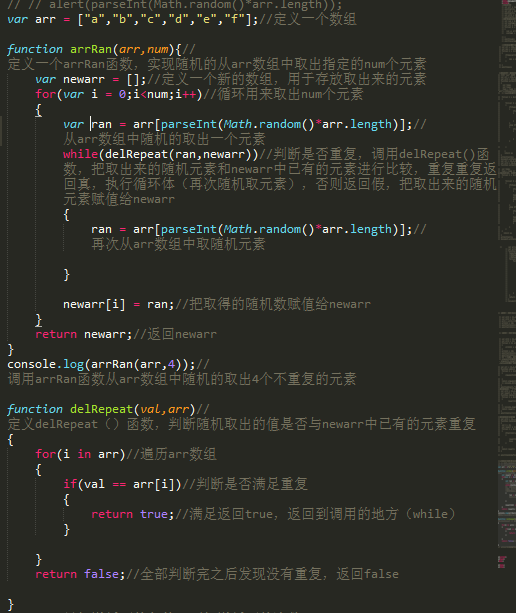


例如：

var arr = ["a","b","c","d","e","f"];

alert(parseInt(Math.random()\*arr.length));

12. 从数组中随机取出指定元素，并且取出的元素不能重复



示例代码：

var arr = ["a","b","c","d","e","f"];//定义一个数组

function arrRan(arr,num){//定义一个arrRan函数，实现随机的从arr数组中取出指定的num个元素

var newarr = [];//定义一个新的数组，用于存放取出来的元素

for(var i = 0;i<num;i++)//循环用来取出num个元素

{

var ran = arr[parseInt(Math.random()\*arr.length)];//从arr数组中随机的取出一个元素

while(delRepeat(ran,newarr))//判断是否重复，调用delRepeat()函数，把取出来的随机元素和newarr中已有的元素进行比较，重复重复返回真，执行循环体（再次随机取元素），否则返回假，把取出来的随机元素赋值给newarr

{

ran = arr[parseInt(Math.random()\*arr.length)];//再次从arr数组中取随机元素

}

newarr[i] = ran;//把取得的随机数赋值给newarr

}

return newarr;//返回newarr

}

console.log(arrRan(arr,4));//调用arrRan函数从arr数组中随机的取出4个不重复的元素

function delRepeat(val,arr)//定义delRepeat（）函数，判断随机取出的值是否与newarr中已有的元素重复

{

for(i in arr)//遍历arr数组

{

if(val == arr[i])//判断是否满足重复

{

return true;//满足返回true，返回到调用的地方（while）

}

}

return false;//全部判断完之后发现没有重复，返回false

}

8.24（**对象**）

一、**对象**

1.javascript语言（脚本语言），是基于对象和事件驱动的解释性松散型语言

2. 对象的概念：

一切皆对象（包括看不到的，eg：事情）

3.属性：事物的特征（数据类型中除了函数以外都可以描述属性）

4.方法：事物的功能或者动作（用函数来描述）

5.java :class 类

类与对象的关系：

先有对象，才有类

类是对象的抽象，对象是类的具体化（实例化）

javascript：没有类 代替类的叫做构造器

eg:工厂——>工厂生产——>手机

类

构造器（手机里的每一个小零件）

6.**对象的创建（三种方式，第二种和第三种常用）**

1.用new关键字：

格式：

var obj = new Object();

2.json(节省)

格式：

var obj = {};

3.构造器（构造函数）

格式：

function person(){

}

var lisi = new person(); //对象是类的具体化

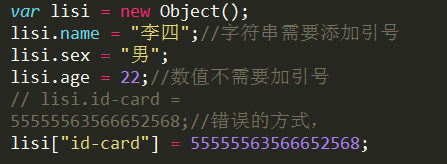
7.添加属性和方法

**添加属性：**

格式：

对象.属性名 = 属性值;

对象[“属性名”] = 属性值



**添加方法：**

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

1.格式:

对象.方法名 = function(){};（一般为匿名函数）

对象[“方法名” ] = function(){};



2.键值对的方式添加属性和方法（注意逗号）



3.构造器添加属性和方法



**8.访问属性和方法**

**属性：**

1.对象.属性名

2.对象[“属性名”]

**方法：**

1.对象.方法名();

2.对象[“方法名”]();

**9.删除对象的某一个属性或方法**

**delete 对象.属性名**

**delete 对象.方法名**

只是删除了属性值

**10删除（销毁）对象**

**对象 = null;**

**11对象的遍历（只有一种方法：for in）**

**for in**

****

**下午：**

**1.for in在遍历对象和数组的区别**

数组：

for(var i in arr)

{

i：数组下标

arr[i]:数组元素

}

对象

for(var i in obj)

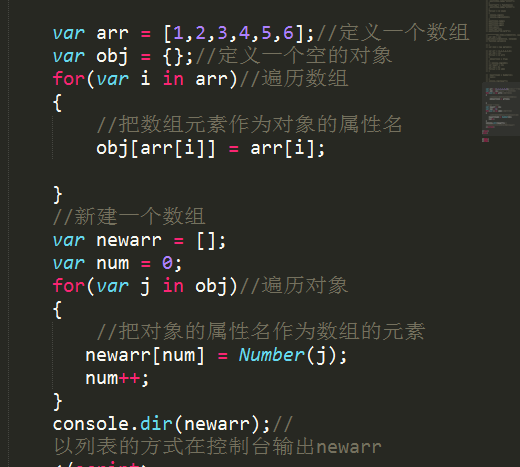
{

i：对象的属性名

arr[i]:对象的属性值

}

例子：



2.对象在内存中的存储

Lisi = null;

把栈区中的地址清空，并且把堆区中的内容全部清空

**3.对象的特性**

**——继承性**：一个对象可以继承另外一个对象的属性

**只有原型可以实现继承**

继承可以解决代码复用，让编程更加靠近人类思维。当多个类存在相同的属性(变量)和方法时，可以从这些类中抽象出父类，在父类中定义这些相同的属性和方法，所有的子类不需要重新定义这些属性和方法，只需要通过继承父类中的属性和方法。

**——封装性**

**作用**：隐蔽性、安全性

**封装的方式（四种）：**

**1) 工厂函数**

格式：

//工厂函数

function phones(){

var phone = new Object();

phone.price = 4500;

phone.call = function (){

alert("可以打电话");

}

return phone;

}

var phone6 = phones();

phone6.call();

**2) 构造函数constructor（造成内存的浪费）**

**存放共有属性**

**3) 原型prototype（解决内存的浪费的问题）**原型存储在代码块区

1.原型既可以实现封装，也可以实现继承

2.构造函数可以添加一个原型,原型里添加方法

3.原型里的方法可以共享给子对象（实例化的对象）

4.原型只能给构造函数添加，一个构造函数只能有一个原型，否则会覆盖

5.一个实例化对象可以有多个原型

**6.原型链：多个原型形成的链，只要是链上的方法，实例化的对象都会继承**

function person (name){

this.name = name;

}

person.prototype = {

study:function(){

alert("Good good study,day day up!");

},

work:function(){

alert(this.name+"完成了作业");

}

}

function student(card,name,banji){

this.card = card;

this.name = name;

this.banji = banji;

}

student.prototype = new person();//person实例化的对象

// {

// study:function(){

// alert("Good good study,day day up!");

// },

// work:function(){

// alert(this.name+"完成了作业");

// }

// }

var xiaoming = new student(2016001,"小明","WUIF1607-2班");//xiaoming是实例化对象

// xiaoming.work = function (){

// alert("没有完成作业");

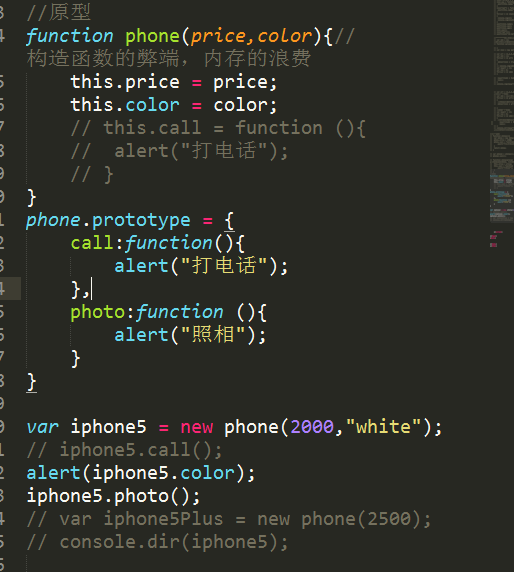
// }//会调用自己的work方法，弹出没有完成作业

xiaoming.work();

//原型只能给构造函数添加，一个构造函数只能有一个原型，否自会覆盖

// 一个实例化对象可以有多个原型

例子：



function phone(price,color){//构造函数的弊端，内存的浪费

this.price = price;

this.color = color;

// this.call = function (){

// alert("打电话");

// }

}

phone.prototype = {

call:function(){

alert("打电话");

},

photo:function (){

alert("照相");

}

}

var iphone5 = new phone(2000,"white");

// iphone5.call();

alert(iphone5.color);

iphone5.photo();

// var iphone5Plus = new phone(2500);

// console.dir(iphone5);

原型存储在代码块区

4) 混合型（最常用）：构造函数和原型混合起来使用

构造函数：只写共有的属性

原型：只写共有的方法

——多态性

多态是指一个引用(类型)在不同情况下的多种状态。也可以理解成：多态是指通过指向父类的引用，来调用在不同子类中实现的方法。

**this的指向（例子见this.html中）**

**1.普通函数中，this指window对象**

var name = "全局变量";

// alert(window.name); =>alert(name);

function aa(){//普通函数

alert(this.name);//弹出全局变量，this指的是window对象，window是全局对象，所有的全局变量都是给window添加的

}

aa();

**2.构造函数中，this指实例化对象**

**3.在方法中，this指调用对象的方法（哪个对象调用了这个方法，this**就指哪个对象）

var obj = {"value":"obj"};

function demo(){

alert(this.value);

}

obj.bb = demo;

obj.bb();

4.在事件中，this指事件被触发的对象

<style>

.box{

width:200px;

height:200px;

border:1px solid black;

}

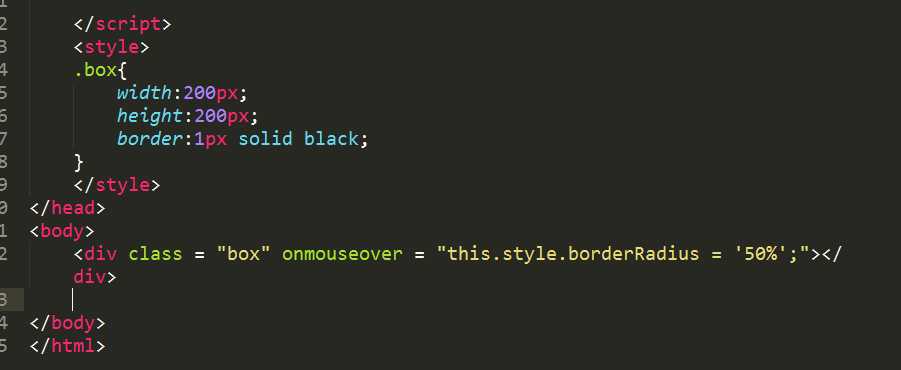
</style>

</head>

<body>

<div class = "box" onmouseover = "this.style.**borderRadius** = '50%';"></div>

</body>



注意：通过脚本语言写css样式时：border-radius：50%；等价于

borderRadius = ‘50%’;

background-iamge等价于backgroundImage

例子：用js获取html元素



5.call()和apply() 用来改变指针方向（this指向）

在使用call函数的同时，也执行了函数

1) call(对象,参数1,参数2,…)

call函数如果对象省略了（没有传的时候），this指window，

var obj1 = {

aa:"aa";

fun:function(){

alert(this.aa);

}

};

var obj2 = {aa:"bb"};

obj1.fun.call(obj2);

2)apply(对象,[参数1,参数2,…]);

**关于this的小练习**

<body>

<input type = "text" id = "myText" value = "文本框">

</body>

<script>

function Obj(){

this.value = "对象";//this指的是window对象

}

var value = "全局变量";

function Fun1(){

alert(this.value);

}

Fun1(); //输出全局变量

Fun1.call(window);//输出全局变量

Fun1.call(document.getElementById('myText'));//输出文本框

Fun1.call(new Obj());//输出对象

8-25

1.Object（基类）是默认存在的，所有的对象都是通过Object派生出来的，

2.原型不能直接给实例化对象添加

**一.js所有对象的分类**

1.内置对象

全局对象goble **Math**对象

2.本地对象：需要实例化的对象（new出来的）

函数function 数组Array 对象object

number var num = new Number(35); 等价于var num = 35;

**string**

Boolean

Date()

正则 RegExp();

3.宿主对象(BOM ,DOM都是宿主对象)

必须在浏览器的环境下才能实现，宿指的就是浏览器

**二Math对象**

1. 属性：

constructor：返回它的构造函数

alert(Math.constructor);

Math.PI π

例子：alert(Math.PI)

2.方法：

1)取绝对值

Math.abs(num);

2) 取近似整数

四舍五入Math.round(num);

向上取整Math.ceil(num);

向下取整Math.floor(num);

3)取最大值

Math.max(num1,num2,num3,…);

4)取最小值

Math.min(num1,num2,num3,…);

5)取随机数(不能传参数) 默认范围0~1

Math.random();

**例子// 取值-8~8之间的随机数（细看）**

alert(Math.floor(Math.random()\*16-8));

**例子//随机取87个人中的一个**

alert(Math.ceil(Math.random()\*87));

6)三角函数的方法

正弦Math.sin(弧度)； 对边比斜边（-1~1之间的小数）

弧度：圆心角所对应的弧

30度角的弧度 = 2\*Math.PI/360\*30

余弦：Math.cos(弧度);

正切：Math.tan(弧度);

7)几次幂

Math.pow(x,y) x:底数 y:幂（指数）

alert(Math.pow(2,10));

8)平方根

Math.sqrt(num);

9)补充：数值对象的方法：

指定小数四舍五入后,小数保留两位，默认四舍五入

numberObj.toFixed(2); numberObj是自己定义的

例子：alert(Math.PI.toFixed(3));

**三、string对象**

1.属性

1)length 字符串中字符出现的个数

2)constructor 返回它的构造函数

2.方法

**1)获取类型**

1>myString.charAt(num); 返回指定位置（num）的字符

num：从0开始

var str = "haha";

// alert(str.length);

// alert(str.constructor);

alert(str.charAt(2));

2>mystring.charCodeAt(下标/数字);返回指定位置字符的ASCII值

3>String.fromCharCode(编码值1, 编码值2,…);接收一个或多个ASCII值，然后返回它们对应的字符

**2)查找类型**

**1>mystr.indexOf(字符);**

返回指定字符在字符串中**第一次出现的位置**，如果字符不存在，返回值为-1

**2>mystr.lastIndexOf(字符);**

返回指定字符在字符串中最后一次出现的位置，如果字符不存在，返回值为-1

**3>mystr.match(字符);**

如果字符串中有这个字符，返回结果为这个字符，如果没有，则返回null

**4>mystr.replace(被替换的字符，新字符);**

返回结果为新字串，不会影响原字符串

例子：把所有的葡萄替换成grape

var str2 = "吃葡萄吐葡萄皮，不吃葡萄不吐葡萄皮";

while(str2.match("葡萄")){

str2=str2.replace("葡萄","grape");

console.log("ok");

}

console.log(str2);

**3)截取类型**

**1>mystr.slice(start,end); 从start开始截取，到end之前结束，结果**为字符串，不会影响原来的字符串，

如果只传一个开始参数，此时截取从开始到后面的所有

**支持负数，表示倒数**

mystr.slice(-3,-1);从最后一个取，倒数第三个开始

**2>mystr.substring(start,end);**

**不支持负数**，其他的与上一个相同

**3>mystr.substr(start,length);**

start:开始位置length截取指定长度

从指定位置开始，截取指定长度的字符 ，支持负数

**4）转换类型**

**1>mystr.split(“分隔符”,length); length:指的是数组的长度**

**将字符串按指定分隔符转换为数组**

**eg:**

var str = "吃葡萄吐葡萄皮，不吃葡萄不吐葡萄皮";

// alert(str.slice(-2));

// alert(str.substring(0,2));

// alert(str.substr(-8,5));

var arr = str.split("，",2);//按照逗号（中文的逗号）进行分割

var arr = str.split("葡萄",3);//按照葡萄进行分割

console.log(arr);

**2>mystr.toLowerCase();将英文大写字母转换为小写**

var str = "DFDFDSFSsdfdff";

alert(str.toLowerCase(str));

**3>mystr.toUpperCase();将英文小写字母转换为大写**

var str = "fkdjfkdf";

alert(str.toUpperCase(str));

例：查找字符串"优逸客官网地址:http://www.sxuek.com,优逸客课程体系:http://www.sxuek.com/uisj/,优逸客学生作品:http://wwww.sxuek.com/sxzp/"

将所有"sxuek"出现的位置，打印出来

//第一种解法

while(str.match("sxuek"))

{

alert(str.indexOf("sxuek"));

str = str.replace("sxuek","12345");

}

//第二种解法

var n = 0;

while(str.match("sxuek"))

{

var num =str.indexOf("sxuek");//19

num += n;

n = n+5;

console.log(num);//19

str=str.slice(0,str.indexOf("sxuek"))+str.slice(str.indexOf("sxuek")+5);

}

**四、数组对象**

1.属性

length 数组长度，元素个数

constructor

2.方法

**添加删除类**

**1) myarr.push(23,34,45);从后面添加元素**

返回值为数组的长度，会影响原数组

var arr = [1,2,3,4,5];

// alert(arr.constructor);

alert(arr.push(18,25,15));//弹出数组长度8

**2)myarr.pop();只能从数组的后面删除一个，不传参数**

返回值为被删除的元素，会影响原数组

var arr = [1,2,3,4,5];

alert(arr.pop());//弹出5

**3)myarr.unshift(元素1,元素2,….); 从数组前面添加**

返回值为数组的长度，会影响原数组

var arr = [1,2,3,4,5];

alert(arr.unshift(2,3,4));//弹出数组长度8

**4)myarr.shift(); 从数组前面删除一个元素，不能传参数**

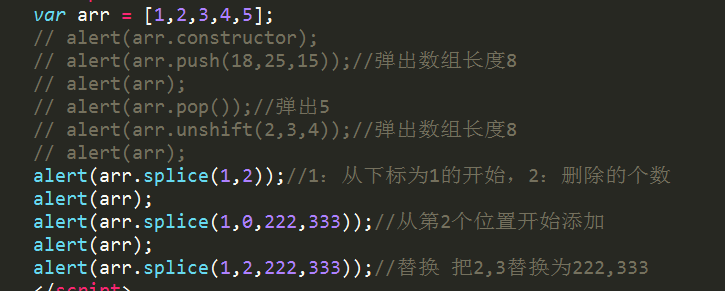
返回值为删除的元素，会影响原数组

**5)myarr.splice(index,数量,添加的元素1,添加的元素2,….); 万能添加删除函数**

index:指定位置，要操作的位置（下标）

数量：为0 表示添加，返回值为空白

否则表示删除的个数 返回值为删除的元素



**转换：**

数组转换为字符串

myarr.join(“分隔符”);分隔符可以省略，默认以逗号转换

返回值为新字符串，参数不传时，分隔符默认为逗号

**截取**

myarr.slice(start,end);与字符串截取用法相同

**数组的连接**

myarr.concat(arr1,arr2,….); 不会影响原数组

**数组的排序**

myarr.sort([callback]); callback回调函数，可以省略

callback省略时，按照ASCII值排序

升序时，传个匿名函数function(a,b){

return a-b;

}

var arr = [1,5,6,8,2,3];

alert(arr.sort(function(a,b){

return a-b;

}));

降序时，传个匿名函数function(a,b){

return b-a;

}

var arr = [1,5,6,8,2,3];

alert(arr.sort(function(a,b){

c

}));

//例题，删除数组中重复的元素

var arr = [23,6,12,4,12,6,67,6,8];

// var newarr = [];

function delpre(arr)//定义一个删除函数

{

var newarr = [];//定义一个数组

for(var i in arr)//遍历数组

{

if(!checkRep(arr[i],newarr))//判断如果不重复

{

newarr.push(arr[i]);//不重复吧元素添加到新的数组中

}

}

return newarr;

}

console.log(delpre(arr));

function checkRep(val,arr)

{

for(var i in arr)

{

if(val == arr[i])

{

return true;

}

}

return false;

}

8-26

**1.终止循环的三种方式**

1) break ：终止距离它最近的循环，

2) continue：结束本次循环，执行下，一次循环

3) 标签语句：可以跳出多层循环

格式：

out://在需要跳出的那层循环层外部添加标签out:

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

{

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

{

break out;//此处的break:out;与标签对应

}

}

**2.用开关法删除数组中相同的元素(这是常用的编程方式)**

var arr = [23,6,12,4,16,67,6,8];

function delRepeat(arr)

{

var newarr = [];

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

{

var flag = 1;

for(var j = i+1;j<arr.length;j++)

{

if(arr[i]==arr[j])

{

flag = 0;

break;

}

}

if(flag == 1)

{

newarr.push(arr[i]);

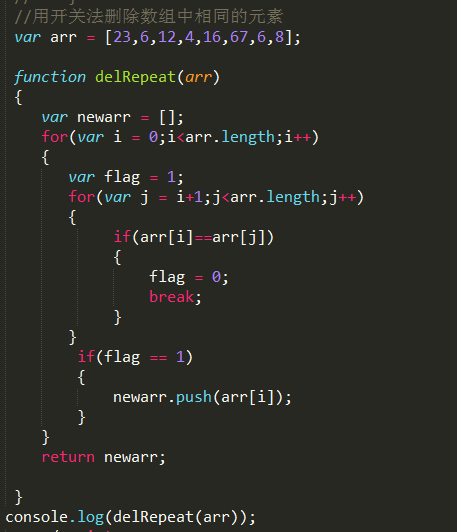
}

}

return newarr;

}

console.log(delRepeat(arr));



一、BOM(Brower Object Model) （浏览器对象模型）

1. window对象

下面是window的子对象：

history（历史对象）\location(地址对象)\document(文档对象)\navigator(浏览器信息对象)…….

2.window对象（是BOM的核心对象）

二、window对象

1.属性：

1）位置类型：获取浏览器的位置

IE：

window.screenLeft 可以获得浏览器距屏幕左边的边距

window.screenTop 可以获得浏览器距屏幕上方的边距

FF:w3c (现代浏览器):

window.screenX 可以获得浏览器距屏幕左边的边距

window.screenY 可以获得浏览器距屏幕上方的边距

2）获取窗口尺寸（任何浏览器都可以使用,无兼容）（比较常用）

宽:document.documentElement.clientWidth;

高:document.documentElement.clientHeight;

3）获得浏览器的分辨率(px)

X: window.screen.width;

Y: window.screen.height;

4）设置浏览器状态栏

window.status=”值”;

2.方法

**1）窗体控制**

**A 对窗体的移动（只有IE8及IE8以下的版本可以使用）**

1. window.moveBy(x,y); 相对于当前的位置移动（x:x轴偏移的数值）

2. window.moveTo(x,y); 移动到指定的位置

**B 窗体的尺寸（只有IE8及IE8以下的版本可以使用）**

1. window.resizeBy(width,height); 在当前的窗口基础上增大

2. window.resizeTo(width,height); 把窗口调整为指定的宽度和高度

**3）控制滚动条的位置（不管用）**

window.scrollBy(x,y); 可把内容滚动指定的像素数。

**注意：** 要使此方法工作 window 滚动条的可见属性必须设置为true！

x: 必需。把文档向右滚动的像素数。

Y: 必需。把文档向下滚动的像素数

window.scrollTo();

**4）时间函数**

**1. setInterval(callback,时间); 毫秒数**

重复执行回调函数，时间一到执行一次

例子：

var i = 10;

setInterval(function(){//只有这个函数setInterval的返回值是进程号

alert(i);

i--;

},1000)

**清除时间函数：clearInterval(进程号);**

**2. setTimeout(callback,时间);**

**当时间一到，只执行一次回调函数**

clearTimeout(进程号)

例子：注意：区别于上一个

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用setTimeout实现倒计时

var i = 10;

function daojishi(){

if(i == -1)

{

clearTimeout(t)

}

else{

console.log(i);

t = setTimeout(daojishi,1000);

i--;

}

}

var t = setTimeout(daojishi,1000);

下午：

一、计时器的思路

1.获取时、分、秒；document.getElementById

2.定义变量时分秒，改变时分秒，innerHTML

限定条件，秒重置时分曾1

**计时器示例代码：**

body部分：

<div id = "box">

<div id = "hour">00:</div>

<div id = "fen">00′</div>

<div id = "miao">00″</div>

</div>：

style部分：

#box{

width:200px;

height:80px;

border:1px solid black;

margin:0 auto;

border-radius:8px;

font-size: 30px;

line-height: 80px;

text-align: center;

/\*投影： x轴的偏移 y轴的偏移 模糊程度 颜色\*/

box-shadow:0px 0px 10px rgba(45,45,45,0.8);

}

#box div{

width:33.33%;

height:80px;

float:left;

text-align: center;

}

script部分：

var h = 0;

var m = 0;

var s = 0;

var hour = document.getElementById("hour");

var fen = document.getElementById("fen");

var miao = document.getElementById("miao");

var t = setInterval(function(){

s++;

if(s<10)

{

miao.innerHTML = s+'″';

}else if(s>=10)

{

s = -1;

m++;

if(m<10)

{

fen.innerHTML = m+'′';

}

else if(m>=10)

{

m = -1

h++;

hour.innerHTML = h;

}

}

},800)

2、alert(); 警告框

3、prompt(); 输入框

4、confirm(“提示信息”); 带确定和取消的弹出框

确定返回true，否则返回false

5、open(url,name,feafurse,replace)通过脚本打开新的窗口（replace）

open() 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。

url:新窗口的地址（路径），是一个可选的字符串，如果省略了这个参数，或者他的值是空字符串，那么新窗口就不会显示任何文档

name:新窗口的名字

feafurse:设置新窗口的位置，尺寸，参数之间用逗号分隔，属性与值之间用等号

eg:width =400,height=400

height = 100 窗口高度

width = 100 窗口宽度

top = 0 窗口距离屏幕上方的像素值

left = 0窗口距离屏幕左侧的像素值

toolbar = no 是否显示工具栏，yes为显示

menubar ,scrollbars 表示菜单栏和滚动栏

resizeable = no 是否允许改变窗口的大小，yes为允许

location = no 是否显示地址栏，yes为允许

status = no 是否显示状态栏内的信息（通常是文件已经打开），yes为允许

replace：是一个可选的布尔值，规定了装载到窗口的url是在窗口的浏览历史中创建一个新条目，还是替换浏览历史中的当期那条目，支持下面的值：

true – URL 替换浏览历史中的当前条目

false– URL 在浏览历史中创建新的条目

6.close();在火狐和谷歌中只能关闭通过open打开的窗口

8-29

1. 1秒 = 1000毫秒

2. 控制滚动条的例子（8-2901.html）



示例代码：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">

<title>控制滚动条</title>

</head>

<body id = "box">

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

<h1>fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会fdhfsdhfjsdfjiodjfjsdjfoidsfjdfhdf发动机号罚丢话费的话费和第三方ID而恢复ID会</h1><br>

</body>

<script>

var box = document.getElementById("box");

var flag = 1;//开启

var t = setInterval(function (){

scrollBy(0,120);

},1000)

box.onclick = function(){

if(flag == 1)

{

clearInterval(t);

flag = 2;

}

else if(flag == 2)

{

t = setInterval(function (){

scrollBy(0,120);

},1000)

flag = 1;

}

}

</script>

</html>

**3.window对象中最常用的属性方法**

document.documentElement.clientWidth; 获取浏览器的宽度

document.documentElement.clientHeight; 获取浏览器的高度

setInterval(callback); 时间函数

**四、history（历史对象） 记录用户访问过的地址信息**

**1.属性：**

history.length 记录用户访问过的地址信息的**长度**或者**个数**

**2.方法：**

1)前进(不会影响历史记录)【模拟浏览器的前进后退按钮】

history.forward(); 不需要任何参数，只是一个功能

2)后退

history.back();不需要任何参数，只是一个功能

3) history.go(); 参数可以是-1/1/0

-1：后退 1：前进 0：刷新

历史记录为双数的时候才能使用

例子：使用history.forward();前进history.back();后退

history.go(-1/1);后退/前进

例子见D:\8.8\html\history

**五、location（地址对象）记录地址信息**

http://www.baidu.com 协议+域名

协议后面直接跟域名

1.地址分析

http://www.baidu.com:8080/aa/bb/index.html#one?name =zhangsan&password = 123245

http 协议 （protocol）

www.baidu.com 域名（hostname）

8080 端口号（port）

aa/bb/index.html 路径（pathname）

#one 锚链接（又叫hash）

?name =zhangsan&password = 123245 查询字符串（search）

整体叫做href

用location.href访问

fied是访问本地文件的协议

哈希地址：带#的

查询字符串（form表单提交的信息）

域名和端口号可以一起获取：host

2.属性：

1) href

**location.href 获取当前文件的地址信息**（用document.write输出）

**设置文件的地址：**location.href = “http://www.baidu.com”; //指定新的地址

例子：

//获取当前文件的地址信息

document.write(location.href);

//设置文件的地址

location.href = "http://www.baidu.com";

本地的文件没有域名

**2) 方法**

1. location.assign(url);重新加载一个窗口，会形成历史记录

2. location.replace(url); 替换当前的窗口，不会形成历史记录

3. location.reload(); 刷新（和history.go(0);一样）

意思是从服务器端重新载入页面

location.false(true)意思是从缓存中重新载入。

四种方法通过脚本打开新窗口：

window.open

location.assign(url);

location.href(url);

location.replace();

下午：

一、DOM（document object model）文档对象模型

html

head body

title link div

p,a,ul

1.document 对象是DOM的核心，每一个动效都需要用它

可以操作内容，操作样式，操作属性

二、属性

1.document.URL 可以获取当前文档的路径

2.document.title 可以获取或设置当期文档的标题（title标签中的内容）

3.document.bgColor 设置文档的背景颜色

4.document.fgColor 设置文档的前景色（文字）



三、方法

1.document.write(); 重写页面方法 （识别html标签）

window.onload = function(){//等待文档加载完成触发的事件

document.bgColor = "pink";

document.fgColor = "white";

}

注意：.document.write();的位置,写在onload中就可以实现页面重写

**2. 文档加载顺序**

1) 文档的主体 eg:html head body

2) 布局的结构

3) 图片

4) 样式，Javascript代码

5) 执行onload事件内部的代码

3.**document..getElementById(“id名”); 只能获取一个元素**

通过id来获取html中的内容，并将其转换为一个document对象

4.**document.getElementsByTagName(“标签名”); 可以获取多个元素**

通过标签名获取html中的内容

获取到的结果为一个集合，如果集合中有多个元素，操作时需要先遍历，如果集合中只有一个元素，则不需要遍历

**5.document.getElementsByClassName(“类名”); 有兼容问题**

IE8及以下的浏览器不支持

返回结果也是一个集合，操作时需遍历，用法与标签名获取一样

例子：



**解决兼容问题：**

**1.新建一个function.js文件**

**判断浏览器，如果是现代浏览器，直接用document.getElementsByClassName(classname);获取，如果是IE，需要先把所有的标签获取到，**

**例子见classname.html和function.js**

**8-30笔记**

**一、获取元素**

1.document.getElementById(); //获取到的是一个元素

2.document.getElementsByClassName(); //获取到的是一个集合

3.document.getElementsByTagName(); //获取到的是一个集合

4.document.getElementsByName(); 获取表单元素，通过name来获取，结果为一个集合，这个方法只适用于表单（在IE中），在现代浏览器中也可以用于div或者其他

**二、元素的获取**

1) document.all 获取所有的元素

eg:alert(document.all.length);

2) document.forms 获取表单元素 结果为一个集合

3) document.images 获取所有的图片元素

4) documents.links 获取所有的链接标签

**三、操作内容**

1. 对象.innetHTML (内容包括html标签和文字)

设置内容 ，获取内容

Eg:复制的例子

2.对象.innerText (内容只有文字)

设置内容 ，获取内容

注意innerHTML和innerText的区别：

用innerHTML获取到的内容包括html标签和文字，而innerText获取到的内容只有文字，在复制的例子中用前者获取，复制的时候会把样式复制过去，而用后者获取，复制的时候只复制了文字

**四、操作属性（两种方式）**

1. **对象.属性** 获取属性的值，设置属性值

Eg:alert(link.herf); 获取的是绝对路径

link.herf = “”;

link.title = “”; 添加属性

2获取属性Attribute

对象.getAttribute(“属性名”); 获取指定属性的值

通过这个方法获取的值，是属性后面跟的字符串

设置属性：

对象.setAttribute(“属性名”,”属性值”);

**可以操作不标准的属性（自己定义的属性）**

**五、操作样式**

**1. 行内样式**

对象.style.属性名 可获取（只能获取行内样式的属性），可设置

弊端：重复写style

**行内样式的批量修改：**

对象.style.cssText = “width:300px;height:300px;”;

**2. 批量修改**

1) id（例子还没搞好）

先把样式保存起来，然后添加id名

2) className

用法用id相同

**3.获取属性的值 有兼容问题**

FF: getComputedStyle(obj,null).属性名;

IE: 对象.currentStyle.属性;

解决兼容问题，写一个兼容函数

下午：

一、选项卡的动效（例子见选项卡文件夹）

快捷键：ul.conbox>li.con\*3>img[src = “img/&.jpg”]{内容}+h1{&&&}

加号表示同级

需要把选项卡和图片一一对应起来

二、后代选择器和交叉选择器的区别（2016-8-30）

目前的理解是：

后代选择器选择的是自己的子元素，

而交叉选择器选择的是自己本身

三、水平移动div

**8-31笔记**

**一、例子：四个方向的三角形**

做三角形的步骤

三角形是由div变换而成的，不需要给div设置宽高，直接给四个方向添加边框，eg:做角朝上的高度为10像素的三角形，设置上边框为0，下边框为10像素；左边框为5像素，有边框为5像素

例子见 D:\8.8\html\8-31文件夹中的 sanjiaoxing.html

**二、五角星动画**

例子见 D:\8.8\831五角星

**下午：**

**三、实现交互的两种方式：**

input 和点击事件

鼠标，键盘，触屏（相对于移动端手机）是交互工具

**四、事件驱动**

1.事件源：事情发生在谁的身上，谁就是事件

2.事件 onclick/onmouseover /onmouseout

3.事件处理程序： 匿名函数，事情如何实现

**五、鼠标事件**

1)单击事件 onclick

2)双击事件 ondblclick

3) 移入事件： onmouseover

4) 移出：onmouseout

5) 鼠标按下事件 onmousedown

6) 鼠标抬起事件 onmouseup

7) 鼠标移动事件 onmousemove

**六、事件添加（绑定）的格式**

1）事件源(对象).事件(onclick) = function(){

做什么事情

}

**七、键盘事件**

1)按下事件 onkeydown

2) 抬起事件 onkeyup

3) 长按事件 onkeypress

**八、window对象**

1) 等待文件加载完成 onload

2) 监测浏览器窗口大小是否发生变化事件 onresize

**九、表单对象**

1）获得焦点事件（把光标给了它） onfocus

2) 失去焦点事件 onblur

**十、例子**

子选择器

box>div

用正则的方式判断是否是标签名

/表示正则的开始 /表示正则的结束

$(function(){

==>window.onload = function(){}

})

**十一、例子 animate.js eg:4.html**

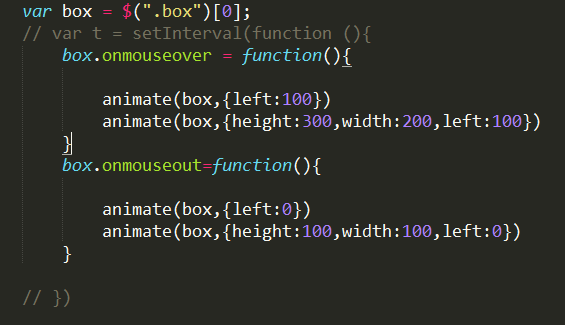
**十二、轮播图动效**

9-1

1.图片在div中居中的方式：text-align:center;

2.animate的用法

Eg:水平位移动画



3.banner图的两种轮播

4.边框的动画

9-2

**一 .DOM模型**

树模型（文档树） 标签用矩形 属性用椭圆

html

head body

meta title link script ul。 Li P

属性（文档的标题 ）

每一部分都称为节点

**二. 节点定义**

是指文档树的标签、属性、文字、注释

**三. 节点分类：**

元素节点：标签

属性节点 ：属性

**文本节点 ：纯文字，空格**

注释节点：注释

文档节点：指document，整个网页

**四. 节点中的获取方法**

1.父节点：对象.parentNode

2.上一个兄弟节点 对象.previousSibling （有兼容问题）

（现代浏览器中）空格也是兄弟元素

IE8中不识别文本节点中的空格

3.下一个兄弟节点： 对象.nextSibling（有兼容问题）

也有兼容问题

4.所有子节点： 对象.childNodes（有兼容问题）getChild()

5.获取第一个子节点: 对象.firstChild（有兼容问题）

6.获取最后一个子节点 : 对象.lastChild（有兼容问题）

**五、节点对象的属性**

1.节点的类型 对象.nodeType

2.节点的名字 对象.nodeName

3.节点的值 对象.nodeValue

**4.获得节点信息的属性**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **节点类型** | **节点名字** | **节点值** |
|  | nodeType（数值） | nodeName | nodeValue |
| **元素节点** | 1 | 标签名（大写） | null |
| **属性节点** | 2 | 属性名 | 属性值 |
| **文本节点** | 3 | #text | 文本 |
| **注释节点** | 8 | #comment | 注释的文字 |
| **文档节点** | 9 | #document | Null |

六、获取所有子节点的兼容函数(在function.js函数中)

**七、节点的方法**

作用：.动态的创建删除更改复制元素，给元素添加属性样式

（一）创建节点：

1.创建元素节点：

**document.createElement(“元素标签名”);**

**eg：var a = document.createElement(“a”); //创建元素**

**document.body.appendChild(a);//追加元素**

2.创建属性节点：

**Eg: var href = document.createAttribute(“href”); //创建属性名**

**href.nodeValue = “http://www.baidu.com”; //给属性添加属性值**

**a.setAttributeNode(href);//给a添加属性href**

3.创建文本节点

**document.createTextNode(“文本”);//创建**

**eg: var text = document.createTextNode(“打开百度”);**

**a.appendChild(text);**

**三种添加属性的方法：**

1. 节点的方法

2. 对象.属性名 = “属性值”；

3. 对象.setAttribute

4.追加元素到页面当中

1) 父对象.appendChild(要追加的元素);插入到最后

2) 父对象.insertBefore(要插入的对象,之前的对象)插入到某个对象之前

5.删除节点

父对象.removeChild(删除的对象);

如果确定要删除节点，最好也清空内存 对象 = null

6.替换节点

父对象.replaceChild(新的对象,旧的对象);

7.克隆节点：先复制需要克隆的元素

Eg: var newthree = three.cloneNode();//

true：复制子节点 false（默认值）：不复制子节点

document.body.appendChild(newthree);

作业：给银泰/京东添加动画

用节点的知识在浏览器中动态的创建十个div，要求十个div大小不同，颜色不同，位置不同

9-5笔记

一、周末作业 动态的创建指定的div

整体思路：

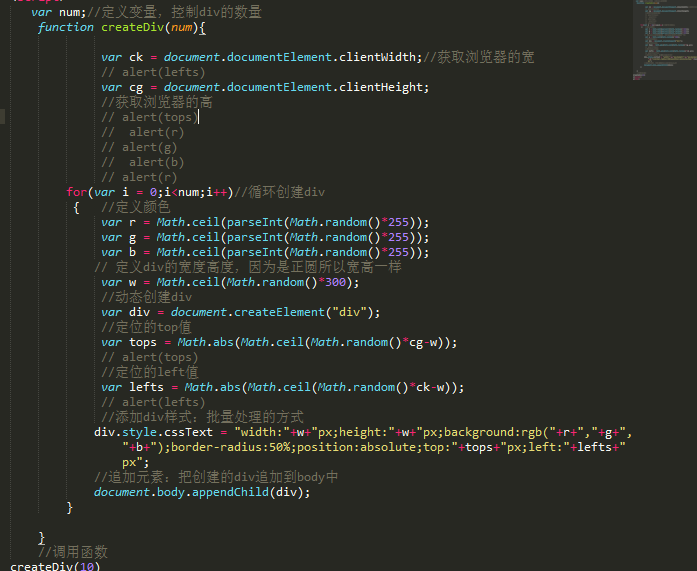
引入js文件

循环创建，追加

在追加之前设置样式，并且定位用cssText

操作样式的两种方式

获取浏览器的宽高



示例代码：

<!-- 2016-9-5动态的创建div位置不同，大小不同，颜色不同 -->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">

<title>Document</title>

<script src = "js/animate1.js"></script>

</head>

<body>

</body>

<script>

var num;//定义变量，控制div的数量

function createDiv(num){

var ck = document.documentElement.clientWidth;//获取浏览器的宽

// alert(lefts)

var cg = document.documentElement.clientHeight;

//获取浏览器的高

// alert(tops)

// alert(r)

// alert(g)

// alert(b)

// alert(r)

for(var i = 0;i<num;i++)//循环创建div

{ //定义颜色

var r = Math.ceil(parseInt(Math.random()\*255));

var g = Math.ceil(parseInt(Math.random()\*255));

var b = Math.ceil(parseInt(Math.random()\*255));

// 定义div的宽度高度，因为是正圆所以宽高一样

var w = Math.ceil(Math.random()\*300);

//动态创建div

var div = document.createElement("div");

//定位的top值

var tops = Math.abs(Math.ceil(Math.random()\*cg-w));

// alert(tops)

//定位的left值

var lefts = Math.abs(Math.ceil(Math.random()\*ck-w));

// alert(lefts)

//添加div样式：批量处理的方式

div.style.cssText = "width:"+w+"px;height:"+w+"px;background:rgb("+r+","+g+","+b+");border-radius:50%;position:absolute;top:"+tops+"px;left:"+lefts+"px";

//追加元素：把创建的div追加到body中

document.body.appendChild(div);

}

}

//调用函数

createDiv(10)

</script>

</html>

二、子选择器imgs>div、

跑马灯思路：

时间函数

获取第一个

改变宽度同时放到后边appendChild

在把宽度变回来

右箭头思路：

获取最后一个元素，

改变宽度

放到前面改变最后一个的宽度

设置左右箭头的位置：

先给左右箭头一起定位

position：absolute;

top:0;

bottom:0;

margin:auto 0;

然后左箭头定位

position:absolute;

left:0

;

再给右箭头定位：

position:absolute;

right:0

**三、操作元素的尺寸**

1.对象.style.width/height（只是内容的宽高） 设置和获取 只能获取行内样式

2.id,className

3.getStyle函数

Getcomputedstyle 错误的，未纠正的

currentStyle()

**4.获取元素的实际宽高（内容的宽+border+padding）**

**对象.offsetWidth**

**对象.offsetHeight**

**5.操作元素的位置**

**1. 对象.offsetLeft** 元素没有定位时，获取的值为元素左边缘距离整个文档的距离 ，元素有定位时，获取的值为元素左边缘距离父容器的距离

文档与浏览器

文档的高度是可变化的

浏览器是指的可视窗口

**2. 对象.offsetTop**

元素没有定位时，获取的值为元素上边缘距离整个文档的距离 ，元素有定位时，获取的值为元素上边缘距离父容器的距离

**作业：漂浮广告，（100\*100）位置不能超出窗口**

**window.onresize = function(){**

**}**

**四、获取具有滚动条元素的位置**

**对象.scrollTop 指元素超出父容器（文档）的距离**

**文档超出可视窗口的距离**

**对象.scrollLeft**

**对象必须有滚动条**

**window.onscroll = function(){** //监测当前滚动条的位置是否发生位置变化，发生变化后触发，可以给任何元素添加

**}**

**window.onscroll = function(){**

**//火狐**

**console.log(document.documentElement.scrollTop)**

**//谷歌/IE8**

**console.log(document.body.scrollTop);**

**}**

**兼容问题**

**window.onscroll = function(){**

**var obj = document.documentElement.scrollTop ? document.documentElement : document.body;**

**console.log(obj.scrollTop);**

**}**

**9-6笔记**

**一、楼层跳转**

**二、事件驱动**

**事件源**

**事件**

**事件处理程序**

**三、事件添加（绑定）**

**1.事件源.on事件 = function(){}**

eg；div.onclick = function(){

alert(1);

}

**2<div onclick = “alert(1)”></div> 在行内样式里边添加**

**3.同一事件源绑定多个相同的事件**

**现代浏览器**

**对象.addEventListener(“mouseover”,事件处理程序,false); 添加事件**

**对象.removeEventListener(“mouseover”,事件处理程序,false); 移除事件**

**注意：不需要写on**

**IE**

**对象.attachEvent(“onmouseover” ,事件处理程序); 添加事件**

**对象.detachEvent(“onmouseover” ,事件处理程序); 移除事件**

**注意:需要加on**

**最后添加的先执行**

**如何处理兼容问题**

**//事件绑定的兼容函数**

**//参数为对象事件事件处理程序**

**// alert(1)**

**function addEvent(obj,event,fun){**

**obj[fun] = function(){**

**fun.call(obj);**

**}**

**if(obj.attachEvent)**

**{**

**obj.attachEvent("on"+event,obj[fun]);**

**//匿名函数没有名字**

**}**

**else{**

**obj.addEventListener(event,obj[fun],false);**

**}**

**}**

**//11移除事件的兼容函数**

**function removeEvent(obj,event,fun){**

**if(obj.attachEvent)**

**{**

**obj.detachEvent("on"+event,obj[fun]);**

**//匿名函数没有名字**

**}**

**else{**

**obj.removeEventListener(event,obj[fun],false);**

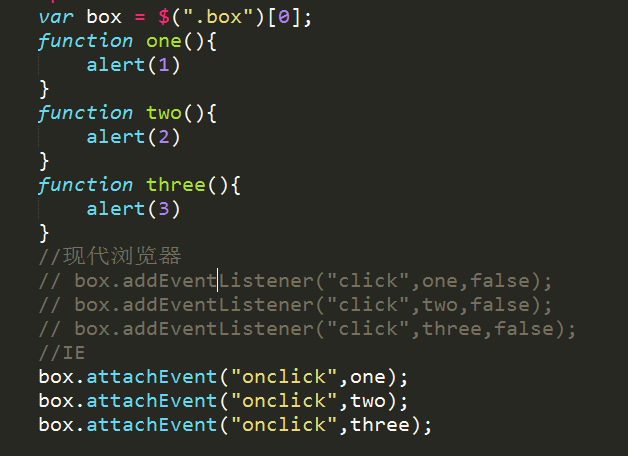
**}**

**}**

什么时候用$

当script标签对写在body的下方时才会使用$函数

例子：



复习：

**this的指向**

**在普通函数中指window对象**

**封装了window函数**

**四、事件对象**

**1.什么是事件对象**

事件对象是用来记录一些事件发生时的相关信息的对象。事件对象只有事件发生时才会产生，并且只能是事件处理函数内部访问，在所有事件处理函数运行结束后，事件对象就被销毁！

**事件源documents对象**

**事件对象：类似arguments对象，事件发生时产生,记录事件发生以后的一些信息（鼠标位置信息和键盘位置信息）**

**js中，用的时候先得获取，否则需要实例化**

**五、事件对象的获取（兼容问题）只能在事件内部获取**

**IE : window.event**

**FF: bigbox.onmouseover = function(e){**

**alert(e);//这个对象就是事件对象**

**}**

**在事件处理程序中传个形参**

**e:参数：表示事件对象**

**六、事件对象里面的信息**

**1.记录事件对象到浏览器的距离**

**e.clientX 事件对象距离浏览器左边的距离**

**e.clientY 事件对象距离浏览器上边的距离**

**2.记录事件对象到屏幕的距离**

**e.screenX 事件对象距离屏幕左边的距离**

**e.screenY 事件对象距离屏幕上边的距离**

**3.记录事件对象到事件源的距离**

**e.offsetX 事件对象距离事件源左边的距离**

**e.offsetY 事件对象距离事件源上边的距离**

**例子：拖拽效果**

**鼠标按下移动抬起事件**

**事件对象阻止浏览器默认行为**

**if(e.preventDefault){**

**e.preventDefault(); //w3c阻止浏览器的默认行为**

**}**

**else{**

**e.returnVlaue = false; //IE阻止浏览器的默认行为**

**}**

**9-7笔记**

**一、对象的三种声明方式**

**1.var obj = new Object();**

**2.var obj = {name:”zhangsan”,sex:”女”}; json格式，来存储数据**

**3.function preson(){ 实现一些功能，具有某些功能的函数**

**this.name = “zhangsan”;**

**this.sex = “男”;//this指实例化对象**

**}**

**person.prototype = {**

**say:function(){**

**},**

**play:function(){},**

**}**

**var liming = new person();实例化对象**

**二、用对象的方式写昨天的例子**

**在两种拖拽里面**

**三、滚轮事件**

**1.滚轮事件（兼容问题太多）**

**兼容问题**

**mouseWheel**

**2.事件对象属性**

**事件对象.wheelDelta 获取滚轮滚动的方向 IE**

**事件对象.detail 获取滚轮滚动的方向FF**

**alert(e.detail); //ff: -3:向上 3：向下**

**alert(e.wheelDelta); //IE 120：向上 -120：向下**

**9-8笔记**

**一、示例：父容器子容器的例子**

移入移出事件 例子见9月8日的文件夹yiruyichu.html

**二、整屏滚动示例**  例子见9月7日的文件夹

**三、事件对象属性**

**事件对象.fromElement 鼠标从哪里来 IE**

**事件对象.toElement 鼠标到哪里去 IE**

**事件对象.relatedTarget FF**

**在mouseover事件中代表IE中的fromElement**

**在mouseout事件中代表IE中的toElement**

**下午：**

**1.阻止浏览器的默认行为（补全案例）9-8文件夹中的Lianxi.html**

**2.键盘事件**

**都触发的顺序 按下，长按，抬起**

**长按是在按下以后，内容发生改变的时候（按下ctrl/shift不会触发）才会触发**

**表单里面的值用text.value 获取**

**按下事件需要响应过程，按下所有的键都会触发**

**抬起事件直接响应**

3.键盘的事件对象

1.获取键盘码（对应的ASCII值） 事件对象.keyCode

键盘码默认以大写的ascii值存储

**特殊的键对应的键盘码**

上：38

下：40

左 ：37

右：39

空格：32

回车：13

Ctrl：17

Alt：18

delete: 46

2.事件对象.altkey

判断当前按下的键是否是alt键，true为是，false为不是

**3.事件对象.ctrlKey**

判断当前按下的键是否是ctrl键，true为是，false为不是

**4.事件对象.shiftKey**

判断当前按下的键是否是shift键，true为是，false为不是

**5.事件对象.type 返回当前被触发的事件名称**

输出事件名称

模拟聊天发送消息 添加单击事件和onkeydown事件

**9-9笔记**

一、完善留言板案例

例子见9-8 liuyanban

**提交按钮与普通按钮区别：**

button-普通按钮，submit-提交按钮；

submit是button的一个特例，也是button的一种  
submit按钮一般出现在网页上需要提交信息到服务器是才使用，  
而button按钮是创建一个按钮，对于实现按钮将会发生什么事这个不一定

一张表单可以有多个普通按钮，且至少有一个提交按钮（需提交表单网页）。

提交按钮一次性提交所有的东西

p标签内部不会自动换行，

提交之后恢复notice中的字符

**二、事件流**

1.定义：

**事件流**：事件的响应流程，当触发页面中的任何一个元素添加的事件时，会按一定的顺序来响应事件，事件的响应过程称为事件流（事件流程）。

eg:son >father >body

事件流适用于每个事件

2. **事件流分类**

1.冒泡型事件流【**默认的，任何一种浏览器都支持**】

单击子容器会一层一层向上触发

从（明确eg:son）确定的事件源到不确定(eg:father,body)的事件源，依次向上响应

缺点：鼠标的移入移出会有bug

2.捕获型事件流 【只有现代浏览器支持，IE8和IE8以下的版本不支持】

从不确定的事件源到确定的事件源依次向下响应

对象.addEventListener(“事件”,事件处理程序,false);

false添加的时间流式冒泡型事件流

true是捕获型的

以上例子见9-9文件夹shijianliu.html

3.**阻止事件流**(事件对象)【适合在冒泡型中使用】

IE：事件对象.cancelBubble = true;

FF：事件对象.stopPropagation();

**区分浏览器的默认行为和事件流**

**4.阻止浏览器的默认行为**

IE： e.returnValue = false;

FF: e.preventDefault();

**5.阻止事件的默认行为 :a submit reset**

return false;

**6.事件委派（事件委托）**

把子容器要绑定的事件委托给他们共同的父容器，执行的时候再交给子容器执行

原理：运用冒泡事件流实现的

e.target 获取目标事件源(鼠标操作谁指谁) FF

e.srcElement获取目标事件源 IE

兼容函数

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

alert(obj);

obj.style.background = “orange”;

示例

原理：运用冒泡事件流实现的

适用于子元素很多，需要添加相同的事件

适用于表格

2016-9-10笔记

一、**放大镜示例** 见9-10放大镜文件夹

**二、表单**

1. **method有两种方式**

get:速度快，提交的信息是有限的，会在地址栏中显示

post:比较安全，速度慢，登录信息用post

**2.get/post区别：**

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。

get是透明的，post是不透明的  
3. 对于get方式，[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvryNbujRsuHuBuWNWuWDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTdn1RLrH6)用Request.QueryString获取变量的值，对于post方式，[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvryNbujRsuHuBuWNWuWDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTdn1RLrH6)用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。   
建议：  
1、get方式的安全性较Post方式要差些，包含机密信息的话，建议用Post数据提交方式；  
2、在做数据查询时，建议用Get方式；而在做数据添加、修改或删除时，建议用Post方式；

2. 常用的表单控件

**3.单行文本框**

**<input type="text" value = "默认值"name = "" maxlength = "15" size = "" disabled = "true">**

maxlength 限定单行文本框字符的最大长度

size设置单行文本框显示的宽度

disabled 属性值true

disabled 属性规定应该禁用 input 元素。被禁用的 input 元素既不可用，也不可点击

disabled 属性无法与 <input type="hidden"> 一起使用。

readOnly = "red"

readonly 属性规定输入字段为只读。

只读字段是不能修改的。不过，用户仍然可以使用 tab 键切换到该字段，还可以选中或拷贝其文本。

readonly 属性可以防止用户对值进行修改，直到满足某些条件为止（比如选中了一个复选框）。然后，需要使用 JavaScript 消除 readonly 值，将输入字段切换到可编辑状态。

readonly 属性可与 <input type="text"> 或 <input type="password"> 配合使用

三。模拟登陆页面 例子见表单文件夹中的denglu.html

四、获取表单和表单控件

1> 获取表单

1.通过直接定位的方式获取

document.getElementsById

document.getElementsByClassName

2. document.forms[下标];

document.forms[“name名称”];

document.forms.name名称;

document.name名称;

2> 获取表单控件

1.直接获取方式

Idname tagname name

2.通过集合来获取

表单对象.elements 获取表单里面所有元素的集合

表单对象. elements[下标]

表单对象. elements.name名称;

表单对象. elements[“name名称”]; 输出结果同上

表单对象. name名称;

**3> 表单以及表单控件的事件**

onsubmit 当表单提交的时候触发的事件 事件源只能是form

onreset 当表单重置的时候触发的事件 事件源只能是form

onblur 当失去焦点的时候触发的事件 用于表单控件

onfocus 当获取焦点的时候触发的事件 用于表单控件

**onchange 当内容改变并失去焦点的时候触发的事件**  **用于表单控件**

注意区分：onblur  **onchange**

**表单的属性和方法**

属性：

action 接收请求的URL

elements 表单中的所有控件元素的集合

length 表单控件的个数

enctype 编码类型 仅限于表单

适用于：表单需要用到文件上传，传的是图片的时候需要把默认值改为：

enctype = "multipart/form-data";

**方法：**

**reset();自动重置**

**submit();自动提交**

eg: login.reset();

login.submit();

**表单控件中的属性**

**1.获取和设置值**

**value 属性**

**2.disabled 获取或设置表单控件是否禁用，值为true或false**

**3.type 读取表单控件的类型**

**4.form 所在表单元素对象（和节点中的parentNode类似）**

**5.readOnly 控件只读属性，Boolean不能更改只能复制和读取**

**6.name 字段名**

**表单元素的方法**

**focus() 获得焦点**

**blur() 失去焦点**

**select()选择文本空间中的所有文本内容**

**下午**

**一.select控件**

**length 选项的个数**

**size 一次显示的个数**

**multiple 是否是多选类型**

**options 所有option选项**

**selectedIndex 设置或返回的选中索引值**

**selected 设置默认选中状态**

form是块元素，表单控件才是行内块元素

**2016-9-12**

一、另一种放大镜案例

二、按需加载案例

不需要修改第一屏

注意，布局的效果能用css处理的不用js

src功能

浏览器解析时遇到src会把src里面的图片强制加载进来

data—src

把src变为data-src 然后给img添加src

例子见9-6楼层跳转

下午

**三、正则**\s空格(用来匹配字符串)

**1. 定义：**一个用来描述或者匹配一系列符合某个语法规则的字符串的语言，在很多文本编辑器或者其他工具里，正则表达式通常被用来**检索，替换或者拆分**那些符合某个模式的文本内容，许多程序设计语言都支持利用正则表达式进行字符串操作。

**2.应用场合**

——数据验证，文本替换，内容检索，过滤内容

——可以理解为：执行字符串函数无法完成的特殊的匹配，拆分。替换功能

**3.正则属于本地对象**

**三大对象：**

**内置对象，本地对象，宿主对象**

**4.创建正则表达式**

**（1.）通过构造函数创建var reg = new RegExp(“正则表达式”,”模式修正符”)**

**var str = "sxuek"; //**

**var reg = new RegExp("x")//匹配字符串中的x,构造函数创建，需要些引号**

**alert(reg.test(str));//返回true,匹配成功**

**（2.）用过字面量的方式创建var reg2 = /x/ ;（效率高）**

**var reg2 = /x/ ; //字面量的方式（需要写定界符）**

**例子：**

**var str2 = "javascript";**

**var reg3 = /ava/;**

**console.dir(reg3.exec(str2));**

**输出结果：**

**C:\Users\Administrator\Desktop\2016-09-12_151902.png**

**0：表示匹配结果是一个字符串，该字符串中的下标**

**1表示ava在javascipt中的下标**

**Input表示javascipt整体**

**区别：**

**var reg3 = /ava/;**

**//从左到右匹配，找到后就不找了**

**var reg3 = /ava/g;**

**//每个都找,找到一个就停止，下次从上一次匹配完的位置往后找**

在全局匹配模式下可以对指定要查找的字符串执行多次匹配。每次匹配使用当前正则对象的lastIndex属性的值作为在目标字符串中开始查找的起始位置。lastIndex属性的初始值为0，找到匹配的项后lastIndex的值被重置为匹配内容的下一个字符在字符串中的位置索引，用来标识下次执行匹配时开始查找的位置。如果找不到匹配的项lastIndex的值会被设置为0。当没有设置正则对象的全局匹配标志时lastIndex属性的值始终为0，每次执行匹配仅查找字符串中第一个匹配的项。可以通下面的代码来查看在执行匹配相应的lastIndex 属性的值。

var str = "123#abc";  
var re = /abc/ig;  
console.log(re.test(str)); //输出ture  
console.log(re.lastIndex); //输出7  
console.log(re.test(str)); //输出false  
console.log(re.lastIndex); //输出0  
console.log(re.test(str)); //输出ture  
console.log(re.lastIndex); //输出7  
console.log(re.test(str)); //输出false  
console.log(re.lastIndex); //输出0

**// []或者[ava]匹配a 或者v 或者a**

（3.）通常将正则表达式字符串放在/RegExp/中间 称为定界符

**5. 正则方法：**

**（1）Test函数**

**RegExp.test(str)**

返回一个布尔值，他指出在被查找的字符串中是否存在模式true为能匹配，false为不能匹配

**（2）exec函数：**在字符串中匹配正则，成功返回数组

RegExp.exec(str)

**6.模式修正符**

——**i** 不区分大小写字母的匹配

——m 将字符串视为多行，修饰^与$

——g 全局匹配， 找到所有匹配项

**7.原子**

什么是原子

**原子是正则表达式中最小的元素，包括英文，标点符号等**

**\d** 匹配任意一个数字[0-9]

**\D** 与除了数字以外的任意一个字符匹配[^0-9]

**\w** 与任意一个英文字母数字或者下划线匹配[a-z0-9A-Z]

**\W**  除了字母数字或者下划线外与任何一个字符匹配[^a-z0-9A-Z]

**\s** 与任意一个空白字符匹配 空格，enter,tab制表符

**\S** 除了空白字符外任意一个字符都能匹配[^\n\f\r\t\v]

[\n\f\r\t\v]

\f 换页字符

\n 换行字符

\r 回车字符

\t 制表符

\v 垂直制表符

8/原子表

[ ] 只匹配其中的一个原子

[^] 只匹配“除了”其中字符的任意一个原子

[0-9] 匹配0-9任何一个数字

[a-z] 匹配小写a-z任何一个字母

[A-Z] 匹配大写A-Z任何一个字母

**9.元字符**

在正则表达式中有一些**特殊字符代表特殊意义**叫元字符

**.（点）** 除换行符以外的任何一个字符

**|（杠）** 或的意思，匹配其中一项就代表匹配

**例子：匹配身份证号，旧版是15为数字，新版是18为数字**

/^\d{15}|\d{18}$/

/ / 表示定界符

^ 表示字符串的开始

\d 表示匹配数字

$ 表示字符串的结束

{15} 表示可以匹配15次

| 表示或者

练习

一。检测用户输入内容是否包含“法轮功”，“枪支”，“中石油”，“共产党”等非法内容，如果包含显示内容非法，不包含显示通过

var forms = document.biaodan;

var input = forms.inputs;

input.onmouseout = function(){

var val = input.value;

var reg = /法轮功|枪支|中石油|共产党|毒品/g;

if(reg.test(val))

{

alert("内容非法");

}

}

**二、原子分组**

**匹配多个字符是用()分组，**分组代表一个原子集合或者说一个大原子，并压入堆栈（内存）用于调用，组号是从左到右计数的，**调用时，如果是字面量形式用\1，构造函数方式用\\1，**这种方式我们叫做反向引用

例：

var reg = new RegExp(“(hdw)123\\1”,i)

\\1 表示反向匹配分组的第一部分，i表示不区分大小写

alert(reg.test(“hdw123hdw”))

**反向分组使用场合：未完**

**取消反向分组**

使用形如(?:pattern)的正则就可以避免保存在括号内的匹配结果，反向引用也将会失效

**量词（操作数量）**

可以使用一些元字符，重复表示一些原子或元字符

\* 重复零次或多次

+ 重复一次或多次

？重复零次或一次

{n}重复n次

{n,}重复n次或多次

{n,m}重复n次到m次

一片两片三四片，落尽正则全找见

上面的小标题翻译成正则就是{1},{2},{34},{1}

[]或者（）分组{} 量词

作业：写一个函数，去除所有空格

function tirm(str,type){

if(type =="both")//去除字符串两边的空格

{

str = str.replace(/^\s\*|\s\*&/g,"");

}

else if(type == "left")//去除字符串左边的空格

{

str = str.replace(/^\s\*/g,"");

}

else if(type == "right")//去除字符串右边的空格

{

str = str.replace(/\s\*&/g,"");

}

else if(type == "all")//去除字符串中所有的空格

{

str = str.replace(/\s\*/g,"");

}

return str;

}

var str = " aa bb cc ";

console.log(tirm(str,"all"));

console.log(tirm(str,"left"));

console.log(tirm(str,"right"));

console.log(tirm(str,"both"));

**2016-9-13 正则表达式**

**一、量词的特性**

贪婪（默认的）、吝啬

**二、正则匹配是贪婪的，禁止贪婪用 ?**

——\*? 重复任意次，但尽可能少重复

——+? 重复一次或多次，但尽可能少重复

——?? 重复0次或1次，但尽可能少重复

——{n,m}?重复n次到m次，但尽可能少重复

——{n,}? 重复n次以上，但尽可能少重复

**三、匹配边界**

**1. 字符边界**

^ 匹配字符串的开始

$ 匹配字符串的结束

**.2 单词边界限制**

\b 匹配单词的边界（单词边界）

\B 匹配除单词边界以外的部分（单词内容）

**四、 模式匹配的顺序（从高到低）**

|  |  |  |
| --- | --- | --- |
| **顺序** | **元字符** | **说明** |
| 1 | () | 模式单元 |
| 2 | ? \* + {} | 重复匹配 |
| 3 | ^ $ | 边界限制 |
| 4 | | | 模式选择 |

**五、 字符串中用到的正则的函数**

**1. str.replace(正则或字符串，替换内容) //支持全局g修饰符**

如果模式不是全局，当匹配到一个以后就不会继续匹配，反之则会继续往下匹配

例子：内容的修改

var str1 = "吃葡萄吐葡萄皮，不吃葡萄不吐葡萄皮";

console.log(str1.replace(/葡萄/g,"grape"));

**2. str.search(regexp)**

不支持全局g，返回结果为str中第一个与regexp相匹配的子串的起始位置，找不到返回-1

search() 方法不执行全局匹配，它将忽略标志 g。它同时忽略 regexp 的 lastIndex 属性，并且总是从字符串的开始进行检索，这意味着它总是返回 str 的第一个匹配的位置。

例子：

var str1 = "吃葡萄吐葡萄皮，不吃葡萄不吐葡萄皮";

console.log(str1.search(/葡萄/)); //返回结果为1

**3. str.split方法把字符串转为数组**

**拆分字符串，**split() 方法用于把一个字符串分割成字符串数组。

**4. str.match(regexp)**

 javascript中的match函数是使用正则表达式对字符串进行查找，并将查找的结果作为数组返回,没有找到匹配，返回 null

**match与search 的区别：**

**match()函数只检测RE是不是在string的开始位置匹配，**

**search()会扫描整个string查找匹配；**

JavaScript中match函数方法返回的数组有三个属性：input、index和lastIndex。Input 属性包含整个的被查找字符串。Index 属性包含了在整个被查找字符串中匹配的子字符串的位置。LastIndex 属性包含了最后一次匹配中最后一个字符的下一个位置。如果没有设置全局标志 (g)，数组的0元素包含整个匹配，而第 1 到 n 元素包含了匹配中曾出现过的任一个子匹配。这相当于没有设置全局标志的 exec 方法。如果设置了全局标志，元素0到n中包含所有匹配。

**关键词的过滤**

**例子，把textarea中输入的文字中的非法字符替换为\***

<body>

<textarea name="dhwb" id="" cols="30" rows="10"></textarea>

<div class = "box"></div>

</body>

<script>

var dhwb = document.getElementsByTagName("textarea")[0];

var box = $(".box")[0];

dhwb.onkeyup = function(){ //添加键盘抬起事件

var reg = /中石油|枪支|共产党|毒品|法轮功/g;

var val = this.value;

**box.innerHTML = val.replace(reg,function(arr){**

// replace的第二种方法

var str = "";

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

{

str+="\*"

};

return str;

});

// replace的另外一种用法，在正则中第二个参数为回调函数

}

var str = " aa bb cc ";

console.log(tirm(str,"all")); trim() 函数移除字符串两侧的空白字符或其他预定义字符。

console.log(tirm(str,"left"));

console.log(tirm(str,"right"));

console.log(tirm(str,"both"));

</script>

学习方法

1.准备一个小本，把不会的记录下来，经常翻阅

2.多复习基础知识

3.多练习，看例题，写代码的时候加注释

4.理清思路，看问题会遇到那块知识点

5.善于用firebug审查元素

下午

**1.用正则匹配类名案例**

var class1 = ".Aanner-left";

var class2 = ".box2"

var reg = /^\.[$\_a-zA-Z][$\_a-zA-Z\d\-]\*/;

console.dir(reg.exec(class2));

console.dir(reg.exec(class1));

**2.常用的正则匹配**

**整数或者小数 /^-?\d+\.?\d{0,3}$/**

**只能输入数字 /^[0-9]\*$/**

**验证用户名和密码 /^[a-zA-Z]\w{5,18}$/**

**验证电话号码 /(?:\(\d{3,4}\)|\d{3,4}-?)\d{8}/**

**验证身份证号 /^\d{15}|\d{18}$/**

**验证e-mail地址**

**/^[a-zA-Z0-9]\w+@[a-zA-Z]+\.(com|cc|org|net|cn|com.cn)$/**

**验证URL**

**/^http://([\w-]+\.)+[\w-]+(/[\w-.\?%&=]\*)?$/**

**/^(http[s]?:)?(\/{2})?([a-z0-]+\.)?[a-z0-9]+(\.(com|cn|cc|org|net|com.cn))$/i**

**3. javascript日期对象（获取时间的）**

GMT(Greenwich Mean Time)是格林尼治时间

UTC是协调世界时

JavascriptDATE使用UTC（国际协调时间）

**4. Javascript日期对象（本地对象）**

在javascript中日期也是他的内置对象，所以要对日期进行获取和操作，必须实例化对象

**1.创建日期对象**

var dateobj = new Date();

将会包含本地时间的信息，包括年月日时分秒 星期 时区

可传入的参数格式

1."时.分.秒 月/日/年" "月/日/年 时:分:秒" 字符串

2.年,月,日,时,分,秒 不能加"" 注意：用逗号月份要减1

注意：如果不传参数的话，会得到当前时间的信息

**2.获取日期信息的方法（以下方法不需要传参数）**

getDate() 从Date对象返回一个月中的某一天(1~31)

getDay() 从Date对象返回一周中的某一天(星期几)

getMonth() 从Date对象返回月份（0~11）

getFullYear() 从Date对象以四位数字返回年份

getHours() 返回Date对象的小时（0~23）

getMinutes() 返回Date对象的分钟（0~59）

getSeconds() 返回Date对象的秒数（0~59）

getMilliseconds() 返回Date对象的毫秒（0~999）

getTime() 返回1970年月日至今的毫秒数

getTimezoneOffset() 返回本地时间与格林威治标准时间（GMT）的分钟差

**设置日期的方法**

setDate() 设置Date对象中月的某一天(1~31)

setMonth() 设置Date对象中月份（0~11）

setFullYear() 设置Date对象中的年份（四位数字）

setHours() 设置Date对象中的小时（0~23）

setMinutes() 设置Date对象中的分钟（0~59）

setSeconds() 设置Date对象中的秒数（0~59）

setMilliseconds() 设置Date对象中的毫秒（0~999）

setTime() 以毫秒设置Date对象

setUTCDate() 根据世界时设置Date对象中月份的一天(1~31)

setUTCMonth() 根据世界时设置Date对象中的月份（0-11）

setUTCFullYear() 根据世界时设置Date对象中的年份（四位数字）

setUTCHours() 根据世界时设置Date对象中的小时（0~23）

setUTCMinutes() 根据世界时设置Date对象中的分钟（0~59）

setUTCSeconds() 根据世界时设置Date对象中的秒钟（0~59）

setUTCMilliseconds() 根据世界时设置Date对象中的毫秒（0~999）

**例子：写距离中秋的时间**

**2016-9-14**

下周学习Ajax

一、打字游戏案例

例子见打字游戏9-14号打字游戏文件夹

做打字游戏注意字母的切图，保存为png格式（透明）的图片

做题总结

background-position:10px 0px;// 第一个值代表左右，第二个值代表上下