第一章JavaSript入门

1、js>行为（动画） 对象 函数 变量

2、面向对象（java），面向过程，基于对象（js）：基于window对象，依附于html

3、浏览器架构：

用户界面

数据存储

浏览器引擎

浏览器内核

网络 js解析器 UI后端

4、ecmascript

Es5就是js

5、Javascript具有以下几个基本特点：

是一种脚本编写语言

基于对象的语言

简单性

安全性

动态性

跨平台性

6、内嵌方式

<script type= “text/javascript”>

Js代码

</script>

7、外联方式

首先将js代码保存成独立的js文件

<script type=”text/javascript” src=”js文件.js”></script>

8、单行注释 多行注释

// /\*\*/

9、console.log控制台打印

第二章JavaScript变量及数据类型

1、变量是存取数据的内存空间

2、变量命名规则：以字母或下划线开头可以包含字母数字下划线，不能包含特殊字符（空格@&等，）

3、变量的创建/声明/定义方法：

方法一：先创建后使用

Var x;//创建变量

X=10//使用变量

方法二：直接使用由系统自动创建

Y=9；

4、创建变量同时初始化：

Var z=8；

同时创建多个变量之间用逗号隔开

5、骆驼Camel命名法：第一个单词第一个字母是小写的，接下来的每个单词第一个字母都以大写字符开头

myHappyDate

帕斯卡Pascal命名法：第一个单词第一个字母是大写的，接下来每个单词第一个字母都以大写字符开头

MyHappyDate

匈牙利命名法：在以pascal标记法命名的变量前附加一个小写字母（或小写序列），说明该变量的类型

属性+类型+对象描述

-var iTestValue=0;

-var sSecondvalue=”hi

6、常用数据类型：（基本数据类型）

数值型：3 2.32 NaN（类型数值型，叫非数值）

数值型可以做所有算数运算

布尔型（boolean）：true false

布尔型可以做逻辑运算

空类型：undefined（变量未赋值，变量为空）/null（对象为空）

变量为赋值

字符串型（string）：凡是用单引号‘’或双引号“”包裹的内容都是字符串

字符串型只能做字符串连接运

初学，一般容易书写错误，或标点符号不是英文半角

复合数据类型：object（对象），array（数组）

7、类型测试：typeof（第一种方法）

typeof NaN //number

typeof 3 //number

typeof “abc” //string

typeof undefined //undefined

typeof null //object

typeof true //Boolean

instanceof(第二种方法 ，类型名称（必须首字母大写)返回值为false true)

8、字符串相当于文本，

两边都是字符串“+”作为的是字符串连接符

两边都是数值，“+”作为加法运算

一边是字符串一边是数值，会把数值强制转化成字符串，进行字符串链接

变量交换：

x=10,y=20

console.log(“交换之前的x=”+x+”,y=”+y);

var z; //z=undefined

z=x //z=10

x=y //x=20

y=z //y=10

console.log(“交换之后的x=”+x+”，y=”+y);

第三章JavaScript运算符和表达式

1、表达式：由运算符连接操作数组成的式子，不管运算符多长，最终都是一个值。

2、算数运算符：加+，减-，乘\*，除/，取模%，负数-，自增++（+1），自减--（-1）。

++i(加加在前，先加1，再进行其他运算)

i++(加加在后，先进行其他运算，再加1)

计算总结果时，++在前就赋值加1，++在后忽略，不用管

赋值的结果就是赋值加上++，不管在前在后。

3、等于==、严格全等===、不等于！=、严格不等于！==、大于>、小于<、大于等于>=、小于等于<=

4、一个=是赋值、二个=是只比较值不比较数据类型、三个=即比较值也比较数据类型

5、逻辑与&&（两个真才是真）、逻辑或||（一个真就是真）、逻辑非（真就是假，假就是真）！

6、按位与&（全为1则为1）按位|（有1为1）、按位异或^（不同为1）

7、条件运算符：表达式？值1：值2；（条件成立执行问号后面的，条件不成立执行冒号后面的）

alert（） 弹出框 console.log（）控制台打印 document.Write（）页面中打印

8、运算符：+=、-=、\*=, var a=2 console.log(a+=2); a=4

第四章 javascript结构

1、顺序结构，分支结构，循环结构；

2、顺序结构：所有语句全部从上往下，逐条执行（没有结构就是顺序结构）

开始或结束， 判断条件， 执行语句， 程序执行的方向

3、单分支：要么做，要么不做

If(条件){

语句

语句

}

4、双分支：要么做这个，否则做那个

If(条件){

语句

语句

}else{

语句

}

5、多分枝结构：

If(条件){

语句

语句

}else if(条件){

语句

} else if(条件){

语句

}else（条件之外的写在下面）{

语句

}

6、多选一：值相等，跳转到指定位置执行

Switch(表达式/变量){

Case值一：

语句；

Break；（终止之后的代码执行）

Case 值二:

语句；

Break；

Default:（符合表达式或变量，但是case没有声明的东西写在default下面）

语句；

}

上面的条件是变量里面的条件也是变量，上面的是表达式，里面也应当是表达式

7、for循环

For(循环变量初始化；条件；增量{

语句；

语句；

……；

}

8、while循环：直到型循环，知道条件为false退出循环

While(条件){

语句；

语句；

……

增量放在下面

}

9、do-while循环：类似于while循环

Do{

语句；

语句；

……；

增量放在下面。

}条件

10、Do-while与while的区别：

（1、）当条件不成立的情况下，do-while会执行一次，while不执行；

（2、）条件成立的情况下，while比do-while多循环一次；

（3、）do-while比whlie更容易死循环

11、Continue与break区别：

Continue终止本次循环，继续下一次循环

Break终止本次循环，但是会执行下次循环

//打印100个7的倍数

var i=1;//需要打印的数

var j=0;//设置的打印次数

while(i>0){

    if(i%7==0){

        document.write(i+"<br>");

        j++;

    }

    if(j==100){

        break;

    }

    i++;

}

//标记法

第五章javascript函数

1、函数是完成指定功能的程序段，可以反复调用，减少代码冗余。

2、函数定义/声明/创建

Function 函数名（形参，形参，……）{

语句；

语句；

……；

[return 表达式/值;]

}

调用函数

Onclick=“函数名

3、 全局变量：函数外创建（作用整个）

局部变量：函数内创建（只作用于函数内部）

4、无参函数：函数没有参数，不需要参数，一般只完成固定作用

5、有参函数

（1）、单参函数：只有一个函数

（2）、多参函数：有多个函数

（3）、递归函数：函数本身自己调用自己

（4）、匿名函数：没有函数名称的函数（把整个函数括起来，调用（））只能在js里调用，不能放在html里（function（）{}）()

（5）、构造函数：用来创建对象

函数优先加载函数内部局部变量 函数里面有var，函数上面的只就是undefined,下面的就是var的值

（6）、内部私有函数：位于函数内部的函数

（7）、带返回值的函数：函数定义中有return的语句（return是把值赋给test（））

（8）、回调（diao）函数：被当作实参传递的函数。当一个函数的返回值传递给另一个函数的时候使用回调函数

（9）、能重写自己的函数：由于一个函数可以返回另一个函数，因此我们可以用新的函数来覆盖旧的函数

系统（内置）函数:

URL编码函数：encodeURl()

URL解码函数：decodeURI()

数据类型转换函数-转换为整数：parseint（）；转换失败返回NaN

数据类型转换函数——转换为实数：parseFloat（）；转换失败返回NaN

判断是否是非数字：isNaN（）；

--isNaN parseFloat综合实例1

执行字符串表达式或执行js语句字符串：eval（）

第八章javascript内置对象

1、什么是对象：收音机、学生、手机

什么是面向对象：使用对象的时候只关注对象提供的功能，而不关注其内部实现的细节

用对象

写对象

面向对象编程是一种通用的思想，并非只有编程才用，任何事情都能使用

2、对象的组成：属性---变量（静态特性）、方法---函数（动态行为）

3、对象一共有八个：String(字符串对象)、Array（数组）、Math(数学)、Date（时间）、Number（数字）、Function（函数） RegExp（正则表达式） Error（异常）

Get和post的区别：

1. get明文提交、post密文提交
2. get有字符个数限制、post没有个数限制
3. get会提交一次、post会提交两次

###### String(字符串对象)：

--直接赋值

--new String();构造函数创建

创建字符串对象的两种方法：

var str=”Hello”;

var strNew=new String(“Hello”);

###### String对象属性：

--Length属性

--portotype属性（原型属性）

X=”abc”

String.prototype.fn=function(){

Alert(“测试”);

}

x.fn()

prototype属性:为对象增加新方法；覆盖旧方法实现新功能

console.log(str.length)字符串数量。

###### String对象方法：

--charAt(参数为数值)返回字符串中的第n个函数

Var str=”hello”

Console.log(str.charAt(2)); //返回值为”l”,从0开始，h是0，e是1，l是2

--charCodeAt()返回字符串中的第n个字符的Unicode编码a:97;A:65

--concat()连接字符串

--fromCharCode()从字符编码创建一个字符

--indexOf()检索字符串

Var str=”hello”

Console.log(str.indexOf(“lo”)); //有的话返回索引值,如果没有的话返回的值是-1

--lastIndexOf()从后向前检索一个字符串

--localeCompare()用本地特定的顺序来比较两个字符串是否相等

多的话就是-1，少的话就是1

--match()找到一个或多个正则表达式的匹配

--search()检索与正则表达式相匹配的字符串

--slice()截取一个字符串

Var str=”hello”

Console.log(str.slice(1,3)); //he 范围0-2,不包括3，如果里面就一个1，一直到结束

--split()将字符串分割成字符串数组:

Var str=”a&b&c&d”

Console.log(str)

Var arr=str.split(“&”);

Console.log(arr);

--substr()截取字符串:

Var str=”helloworle“

Console.log(str.substr(0,3)); //hel,如果一个值就一直到结束，负数为从后往前截取，截取多少个

Slice（从开始，为止）

Substr(开始，个数)

1. 判断超出的内容显示省略号—从前往后截取—正数
2. 判断文件上传类型是否正确—从后往前截取—负数

--substring()返回字符串的一个子串

不能用负数

--toLowerCase()将字符串转化成小写

--toUpperCase()将字符串转换成大写

--valueOf()返回字符串

其他类型的数据转换为字符串

--toString()返回字符串

###### Array数组对象：

创建对象方法：

--var arr(对象名)=new Array(); //只知道当前对象是数组对象，没有长度，没有值

--var arr(对象名)=new Array(大小)； //知道长度，但是不知道值

--var arr(对象名)=new Array(值1，值2，值3，值4)；//长度和值都有

--var arr(对象名)=[值1，值2，值3，值4]; //建议使用，直接赋值

遍历数组：

第一种方法：for(var i=0;i<a.length;i++){

Console.log(a[i]);

}

第二种方法：for(I in arr){

Console.log(arr[i]);

}

###### Array对象属性：

--length属性

--prototype属性（原型）

注：prototype属性

--为对象添加新方法

--覆盖旧方法实现新功能

###### Array对象的方法：

--join()将数组元素按指定分隔符连接起来，返回一个字符：（前）

Var arr=[“a”,”b”,”c”,”d”];

Var str=arr.join(“”);

Console.log(str);

--pop()删除并返回数组的最后一个元素（出栈）（前）

--push()给数组添加元素（入栈）

--concat()连接数组

Var arr1=[1,2,3,4]

Var arr2=[2,3,4,5]

Var arr3=arr1.concat(arr2);

--reverse()颠倒数组中元素的顺序

--shift()将元素移出数组（后）

--unshift()在数组头部插入一个元素（后）

--slice()返回数组的一部分

--sort()对数组元素进行排序

--splice()插入、删除或替换数组的元素

//splice(下标，删除或替换的个数，~~~~替换或插入的值)

--tostring()将任意转换成一个字符串

--IndexOf()

###### Math对象属性：

--E属性—返回算术常量e，即自然对数的底数（约等于2.718）

--LN10属性—返回10的自然对数（约等于2.302）

--LN2属性—返回2的自然对数（约等于0.693）

--LOG2E返回2为底的e的对数（1.414）

--LOG10E返回以10为底的e的对数（约等于0.434）

--PI属性—返回圆周率（约等于3.14159）

--SQRT1\_2—返回2的平方根的倒数0（0约等于0.707）

--SQRT2—返回2的平方根（约等于1.414）

###### Math对象的方法：

--random()返回0~1之间的随机数（举例）

Console.log(Math.random());

--floor(x)对数进行下舍入

--ceil()对数进行向上取舍

--found()四舍五入

###### Date对象—日期对象：

--创建日期对象

--new Date();构造函数创建

###### Date对象方法：

--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（0~6）

--getMonth()从Date对象返回月份（0~11）+1

--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）

--getSecondes()返回Date对象的秒数（0~59）

--getMilliseconds()返回Date对象的毫秒（0~999）

--getTime()获取毫秒数

--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）

--setSecondes()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

--toString()把Date对象转换为字符串

--toTimeString()把Date对象的时间部分转换为字符串

--toDateString()把Date对象的日期部分转换为字符串

--toUTCString()根据世界时，把Date对象转换为字符串

--toLocaleString()根据本地时间格式，把Date对象转换为字符串

--toLocaleTimeString()根据本地时间格式，把Date对象转换为字符串

--toLocaleDateString()根据本地时间格式，把Date对象的日期部分转换为字符串

--UTC()根据世界时返回1970年1月1日到指定日期的毫秒数

###### Number对象属性：

--MAX\_VALUE可表示的最大的数

--MIN\_VALUE克表示的最小的数

--NaN非数字值

--NEGATIVE\_INFINITY负无穷大，溢出时返回该值

--POSITIVE\_INFINITY正无穷大，溢出时返回该值

--prototype向对象添加属性和方法

###### Number对象方法：

--toString把数字转换为字符串，使用指定的基数。

var num=new Number(2.6327)

var str=num.toString()

console.log(num);

console.log(str); 控制台打印出来的蓝色是数值，黑色是字符串

--toFixed把数字转换为字符串，结果的小数点后有指定位数的数字

var num=new Number(2.6327)

Console.log(num.toFixed(3)); 结果是2.633，四舍五入

###### Function对象：

--JavaScript的函数属于Function对象

--创建函数的方式

--function(参数,参数……){

语句；

……；

}

--var对象名=new Function([参数1,[参数2,[参数n]]],[函数体])

###### Function对象的属性：

--length参数个数

--arguments属性—代表正在运行的函数，调用函数时的实参

准确来说argunments是一个对象，是实参对象，并且这个对象是以数组格式进行存储的

###### Function对象的方法：

--arguments.callee()方法—该方法调用了当前函数

###### RegExp对象

RegExp（Regular Expression）对象概述

创建正则表达式对象的方式：

--var reg=/表达式/对象的属性；

例如：

--var reg=/内容/

--var reg=new RegExp(‘表达式’[,’对象的属性’])；

元字符-----是用于构建正则表达式的符号

. 匹配任何字符，换行符除外

\d 匹配任何数字字符

\W 匹配任何字符数字（字母或数字和下划线）字符。

\s 匹配空格（空格包括空白字符、tab、换行符、return/enter）

^ 字符串需以模式起始。

$ 字符串需以模式结束。

| 让模式指定一连串可供选择子模式。

限定符-----控制子模式出现于正则表达式的次数

\*限定符前的子模式必须出现0或多次

+限定字符前的子模式必须出现1或多次

？限定字符前的子模式必须出现0或1次

{n}限定字符前的子模式必须出现恰好n次

{n,m}现行符前的子模式至少出现n次，最多出现m次

{n,}至少出现n次

{,m}最多出现m次

字符类[CharacterClass]以方括号围起

等价写法 字符类

\d [0-9] 数字字符

\D [^0-9] 非数字字符

\s [\t\n\x0B\f\r] 空白符

\S [^\t\n\x0B\f\r] 非空白符

\w [a-zA-Z\_0-9] 单词字符（字母、数字、下划线）

\W [^a-zA-Z\_0-9] 非单词字符

转义字符：

字符 含义

- \t 水平制表符

- \r 回车符

- \n 换行符

- \f 换页符

- \cX 与X对应的控制字符（ctrl+X）

- \v 垂直制表符

- \0 空字符

范围类：

如果表示数字[0123456789]

可以使用范围类[0-9]

所有字母[a-zA-Z]

g:global,全文搜索，默认搜索到第一个结果直接停止

i:ingoreCase，忽略大小写，默认大小写敏感

m:multiplelines,多行搜索

test(字符串)方法，仅仅判断字符串是否能够匹配正则表达式匹配返回true，不匹配返回false

var reg=/[u4E00-\u9FA5\u9FA5\uf900-/ufa2d]/

var str=prompt(“请输入文本:””帅哥”)；

if（reg.test(str)）{

alert(“输入文本包含中文”)；

}else{

alert(”输入文本没有中文”)；

}

//检测是否有中文，中文占两个字符。

Function test(){

Var str = document.getElementById(“tb\_input”).value;

Var str\_temp = str.replace(/[\u4e00-\u9fa5]/g,’aa’)

If(str\_temp.length>12){

Alert(“输入错误”)

}else{

Alert(“输入正确”)

}

}

match(要搜索的字符串)方法，可在字符串内搜索指定子字符串，并返回到一个数组中

--实例1（此实例与正则表达式无关）

Match（reg）方法，匹配一个或多个g正则表达式的字符串，返回到数组中

--实例2（参数是正则表达式）

Replace(要搜索的字符串，替换成什么字符串)方法，用于在字符串中用一些字符替换另一些字符

--实例1（此实例与正则表达式无关）

Replace（正则表达式，替换成什么字符串）方法，或替换一个与正则表达式匹配的子串

--实例2（参数是正则表达式）

Search(字符串)方法用于搜索字符串中指定的子字符串，返回第一次匹配的起始位置，没有匹配的返回-1.

--实例1（此实例与正则表达式无关）

Search（正则表达式）或检索与正则表达式相匹配的子字符串，返回第一次匹配的起始位置，没有匹配的返回-1

--实例2（参数是正则表达式）

Split（字符串分隔符）用法，用于搜索制定分隔符，将字符串按照指定分隔符分割，并返回字符串数组

--实例1（此实例与正则表达式无关）

Split（正则表达式）方法，用于搜索匹配正则表达式，将字符串按照指定匹配正则表达式进行分割，并返回字符串数组

--实例2（参数是正则表达式）

###### Error对象：

--每当程序运行发生错误、抛出异常、用户调用Error构造函数，就会创建Error对象，Error对象存储错误相关信息

###### Error创建对象的方式：

--调用Error()构造函数

--调用Error(message)构造函数

--系统抛出异常

###### Error对象的属性：

--name:错误类型（不同浏览器有差异，IE678有的错误类型不同）

--number:错误号（仅支持IE）（32位高16位机器代码，低16位错误代码）

--message:错误信息（不同浏览器有差异）

###### Try-catch异常处理优点：

防止用户看不到友好的系统错误

由于Error在不同浏览器，报错信息有差异，可以使用try-catch进行统一报错信息

可以处理人为编写程序造成异常，增强程序的健壮性

--实例1（统一不同浏览器报错）

--实例2（增强程序的健壮性）

--实例3（try-catch-finally）

String:超出字符显示省略号，判断文件上传类型

Array：最大值，最小值,去重，排序，购物车

Math:获取随机数，随即图片

Date:获取当前时间，倒计时

regExp表单验证，错误清空,一个汉字两个字符

# 第九章JavaScriptBOM对象

DOM（文档对象模型Document Objec Model：当前载入页面所拥有的对象（代表当前文档））

BOM（浏览器对象模型 Browser Object Model：页面以外的所有对象（代表浏览器窗口和桌面平魔））

Body里的就是DOM body以外的BOM

Location

Links

Images

Forms

Anchors

Screen

Navigator

Location

History

Frames

Document

Window

###### Window对象的属性

Innerheight 浏览器窗口实际高度（浏览器的视口，不包括工具栏和滚动条）

Innerwidth 浏览器窗口实际宽度

Length 设置或返回窗口中的框架数量

###### Window对象方法

--alert(“弹出框：提示信息”)

--confirm(“弹出框：提示信息，有确定，有取消”) 返回值有两个：true和flase

--prompt(“提示信息：默认值,带输入框，有确定，有取消”) 确定返回值是输入框里面的取消的话返回值是null

--window.open(“URL”,”窗口名称”,”属性=值,属性=值,属性=值”);

--print()打印当前页面内容

--close()关闭当前浏览器

--resizeBy()将窗口调整宽度和高度（仅IE支持）调整了多少

--resizeTo()将窗口大小调整到种子定的高度和宽度（仅IE支持）调整到多少

--scrollBy()按照指定的像素值来滚动内容 滚动了多少像素

--scrollTo()把内容滚动到指定的坐标 滚动到多少像素

--setTimeout()在指定的毫秒数后调用函数或计算表达式 延迟器

seTimeout（函数名function,ms）js单线程，所有东西加载完成后在加载延时

--clearTimeout()取消由setTimeout()方法设置的timeout

--setInterval()按照指定的周期（以毫秒计数）来调用函数或计算表达式 不能创建在函数里

--clearInterval()取消由setInterval设置的timeout

###### navigator对象的属性：

--appCodeName返回浏览器的代码名

--appName返回浏览器名称

--appVersion返回浏览器的平台和版本信息

--browserLanguage返回当前浏览器的语言

--cookieEnabled返回指明浏览器中是否启用cookie的布尔值

--platform返回运行浏览器的操作系统平台

--userAgent 返回由客户机发送服务器的user-agent头部的值 返回浏览器当前信息

###### Navigator对象的方法

--javaEnabled() 规定浏览器是否开启java

###### Screen对象

--Screen对象—对象包含有关客户端显示屏幕的信息

###### screen对象的属性

--availHeight返回显示屏幕的高度（除Windows任务栏之外）

--availWidth返回显示屏幕的宽度（除Windows任务栏之外）

--height返回屏幕分辨率的高度

--width返回屏幕分辨率的宽度

--Innerheight 浏览器窗口实际高度（浏览器的视口，不包括工具栏和滚动条）

--Innerwidth 浏览器窗口实际宽度

###### History对象属性

--length返回浏览器历史列表中的URL数量

###### History对象的方法

--back()加载history列表中的前一个URL

--forward()加载history列表中的下一个URL

--go()加载history列表中的某一个具体页面 后满的括号里正值是前进，负值是后退

###### Location对象

Location对象—对象包含有关当前URL的信息（javasript里边管理地址栏的内置对象）

###### Location对象的属性

--hash[打开控制台演示]设置或返回从井号(#)开始的URL（锚） 哈希值

--host设置或返回主机名和当前URL的端口号

--hostname设置或返回当前URL的主机名

--href设置或返回完整的URL—通常用来动态跳转

--pathname设置或返回当前URL的路径部分—URL

--port设置或返回当前URL的端口号

--protocol设置或返回当前URL的协议

--search设置或返回从问号(?)开始的URL(查询部分/)

###### Location对象的方法

--assign()加载新的文档 可以找到以前的文档，可以返回

--reload()重新加载当前文档（无参方法）

--replace()用新的文档替换当前文档 找不到一起拿的文档，返回不了了

###### Frames对象

Frames对象—对象包含所有的框架

# 第十章JavaScriptDOM对象

## DOM概述：

DOM文档对象模型Document Objec Model:当前载入页面所拥有的对象（代表当前文档）

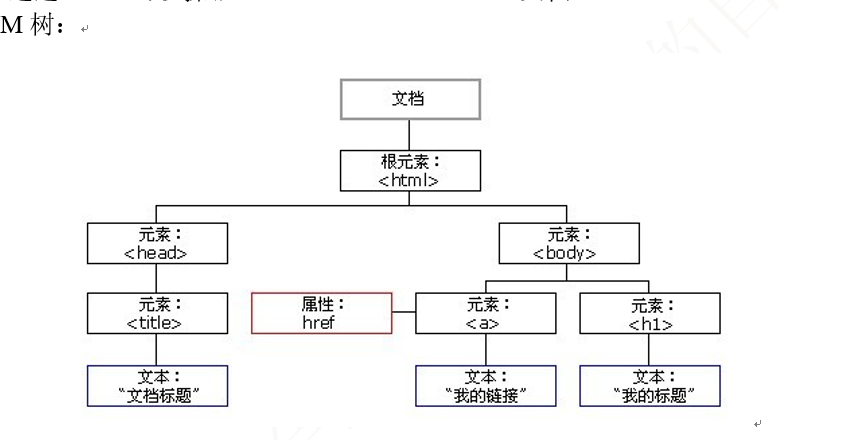
## DOM特点：

--通过DOM可以对整个HTML文档进行新建、添加、更新、删除等操作

--DOM模型符合WEB标准，兼容性好。

--通过DOM可以操纵HTML、XHTML、XML文档。

## DOM树：



HTML文档中的每一个元素都是一个节点，DOM中规定如下：

1. 整个文档就是一个节点
2. 每个HTML标签是一个元素节点
3. 包含在HTML元素中的文本是文本节点
4. 每一个HTML属性是一个属性节点
5. 注释属于注释节点

Document对象获得文档元素：

1. 通过ID获取文档节点：
   1. Document.getElementById(id);

Document.getElementById(“d2”).classname(“aa”)

1. 通过name属性获取文档节点集合（数组）
   1. Document.getElementsByName(name);

Var d1=document.getElementsByName(“d1”);

For(var i=0;i<d1.length;i++){

D1[i].className=”aa”;

}

1. 通过标签

document.getElementsByTagName(tagname);

var d=document.getElementsByTagName(“div”);

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

d[i].className=”aa”;

}

1. 通过class，只能对第一个进行操作。

var d=document.getElementsByClassName(“d1”);

* 1. for(var i=0;i<d.length;i++){

d[i].className=”aa”

}

5、利用document name名字进行查找（外面不能嵌套其他的元素）

document.myForm.username.value

## 创建节点：

--cleateElement(“标记”)创建一个元素节点

--cleateTextNode()创建一个文本节点

## 添加节点：

--appendChild(node)

--添加节点到当前节点内部的后面(新创建的节点)

--移动节点到当前节点内部的后面(已有节点)

--insertBefore(要添加或移动的节点，参考节点)

父节点.inserBefore（要添加的节点，参考节点）。

--添加节点到当前节点内部的前面(新创建的节点)

--移动节点到当前节点内部的前面(已有节点)

删除节点：

--removeChild(node)删除节点

复制节点：

--cloneNode(true)深拷贝—除了复制节点，还复制所有子节点及文本节点

--cloneNode(false)浅拷贝—只复制节点

替换节点：

--replaceChild(新节点，被替换的节点)

判断节点是否有子节点

--hasChildNodes判断是否有子节点 里面的空格也是节点

是否包含某节点

--contains()判断是否包含某节点

更改为父节点：

--parentNode更改对象为当前节点的父节点 （找父元素）

通过父节点更改为子节点

--firstChild更改为当前对象的第一个子节点(前面不能有空格，否则获得是文本节点)

--lastChild更改为当前对象的最后一个节点子节点(后面不能有空格否则获得的是文本节点)

以集合方式更改为子节点（集合方式）

--childNodes更改为子结点的集合（下标从0开始） 里面包含文本节点的

--children更改为子节点的集合（下标从0开始） 里面不包含文本节点

更改为兄弟节点

--nextSibling更改为下一个兄弟 包含文本元素

--previousibling更改为上一个兄弟 包含文本元素

--nextElementsibling 不包括文本，只包含标签

--previousElementSibling 不包括文本，只包含标签

节点属性

--getAttribute(“属性名”)获得节点属性

--setAttribute(“属性名”，“值”)设置节点属性

--removeAttribute（“属性名”）删除节点属性

样式设置

在获取的是时候如果使用的是style来获取的，那么必须要给元素加上行内样式。

例：

<div id=”d1” style=”left”0px”></div>

Document.getElementById(“d1”).style.left🡪如果上面的div 没写行内样式，undefined

--html属性怎么写，js就怎样写

--.style属性（行内样式，css样式带-去掉，必须符合国际命名）

--.style.cssText属性（行内样式，可以一次添加多个样式）

--className属性（应用类选择器class=类名）

js基于的是window对象

window这个对象是js中的顶级对象

this代表的是当前这个对象

直接打印this，this代表的也是一个对象，所以this代表的是window对象

如果把this放在一个函数中，那么this代表的是window

兼容性问题：

--IE6/7/8支持innerText属性，不支持textContent

--firefox不支持innerText属性，支持textContent

--chrome支持innerText属性和textConent

--兼容写法：对象名.textContent=对象名.innerText=“字符串”

--innerText属性 用文本替换当前节点的子节点及文本节点

--innerHTML属性 替换节点所有子节点及文本节点 替换元素里面的

--outerText属性—firefox不支持

--outerHTML属性—替换当前节点 整个元素一快替换

--InnerHTML跟document.write()的区别：

--InnerHTML:是改变某一个块中的内容，刷新的也是某一块

--Document.write():刷新的是整个页面

# 第11章JavaScript事件处理

## 事件：

窗体、对象、鼠标、键盘动作称为事件。

--例如：鼠标单击事件

载入事件

按下键盘事件

## 事件驱动的过程：

--首先，在这个对象上绑定这个事件

--其次，又在这个对象上，发生了这个事件

--最后，系统（js解析器）自动调用处理函数进行响应

## 事件绑定的方式：

--行内绑定（不建议）：无法实现标记和动作分离

--对象名.事件名=function(){语句；语句；}

--对象名.addEventListener（“事件名”，函数名，捕获过程true/冒泡过程false）（前面不能加on）

说明：IE6/7/8的兼容方式是：对象名.attachEvent（“on事件名”，函数）；

## 事件类型：

## HTML相关类型：

--onload //在页面或图像加载完成后立即发生

--支持该事件的标记<body>,<frame>,<frameset>,<iframe>,<img>,<link>,<script>

--支持该事件的对象：image、layer,window

--onresize //在窗口或框架被调整大小时发生

支持该事件的对象：window

--onscroll //在元素滚动条在滚动时触发

支持该事件的标记：<a>,<address>,<b>,<big>,<blockquote>,<body>,<button>,<cite>,<code>,<dd>,<dfn>,<div>,<dl>,<dt>,<em>,<fieldset>,<form>,<frame>,<h1>-<h6>,<hr>,<i>,<img>,<input>,<kbd>,<label>,<legend>,<li>,<object>,<ol>,<p>,<pre>,<samp>,<select>,<small>,<span>,<strong>,<sub>,<sup>,<table>,<textarea>,<tt>,<ul>,<var>

支持该事件的对象：window

## 键盘鼠标相关事件：

--单击事件onclick

--双击事件ondblclick

--卸载事件onunload

--鼠标左键按下onmousedown

--鼠标左键抬起onmouseup

--获取鼠标onmouseover //在鼠标指针移动到元素上时触发

--失去鼠标onmouseout //在鼠标指针移出指定的对象时发生

--键盘按下onkeydown

--键盘抬起onkeyup

--按键盘onkeypress

--鼠标移动Onmousemove 移动发生改变

## 表单相关事件：

--改变事件onchange

--选中事件onselect

--获得焦点事件onfocus

--失去焦点onblur

--重置事件onreset 只能加在form标签中。

--提交事件onsubmit 只能加在form标签中。

## 事件对象：

### 事件对象创建：

--标准方式：系统自动创建

--chrome支持e和window.event

## 事件对象的属性：

1.button 鼠标左键、中键、右键

标准方式（chrome/firefox）：

0 规定鼠标左键

1 规定鼠标中键

2 规定鼠标右键

IE6/7/8拥有不同的参数：

1 规定鼠标左键

4 规定鼠标中键

2 规定鼠标右键

2.shiftKey 键盘shift返回true

3.ctrlKey 键盘ctrl返回ture

4.altKey 键盘alt返回true

5.target 返回触发当前事件的对象

说明：有些情况下，我们封装的addEvent函数，在IE6/7/8中，传递this的时候，this会被更改为window。

解决方法：使用e.target/e.srcElement。

例：

var div1=document.getElementById("div1");

div1.onmousedown=function(x){

var y=x||window.event;

console.log(this); // <div id="div1"></div>

console.log(x.target); // <div id="div1"></div>

}

6.type 返回当前事件的名称

7.screenX 鼠标指针的屏幕水平坐标 相对于屏幕

8.screenY 鼠标指针的屏幕垂直坐标 相对于屏幕

9.clientX 鼠标指针相对于窗口的水平坐标，不包括左右侧边栏和滚动条

10.clientY 鼠标指针相对于窗口的垂直坐标 不包括左右侧边栏和滚动条

IE中写法 X Y

Firefox中写法pageX pageY

11.offsetX 相对于对象区域的x坐标（掌握） 相对于容器的宽高

12.offsetY 相对于对象区域的y坐标（掌握） 相对于容器的宽高

事件对象的方法：

--preventDefault()标准写法，阻止默认动作（事件）

说明：IE兼容写法returnValues属性=false

实例1 实例2

--stopPropagation()标准写法，阻止(事件)冒泡

Event.stopPropagation();

说明：IE兼容写法cancelBubble属性=true

事件流分类：

--冒泡型事件流（由内向外）--常用

--捕获型事件流（由外向内）--很少使用

要想用捕获型事件流就必须用这第三种方法：对象名.addEventListener（“事件名”，函数名，捕获过程true/冒泡过程false）（前面不能加on）

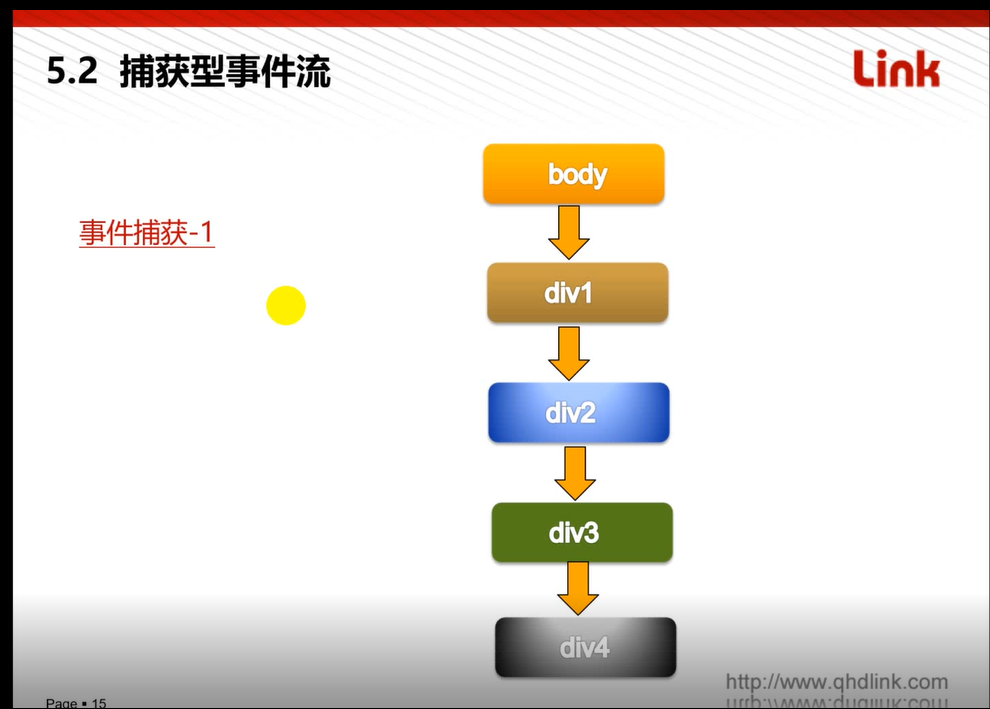
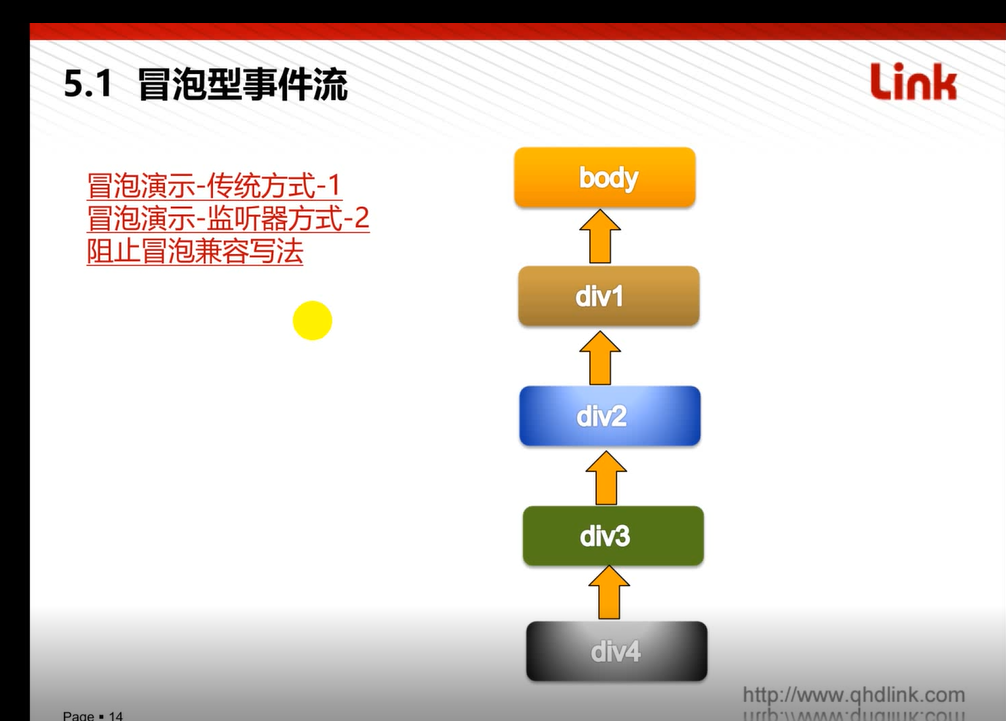
//事件冒泡：

//多个嵌套的元素同时绑定同一个事件，那么在触发的时候，触发当前最内层的元素，这些事件会由内向外依次进行触发，这种事件称为冒泡型事件，并且默认的事件流是冒泡型事件

//事件委托、事件委派：把当前这个事件委托给了父元素，这样的话看起来就像每一个子元素都有这个事件

e.stopPropagation(); //取消冒泡方法

e.cancelBubble=true;  //IE兼容写法



Return false阻止下面的命令执行 （表单验证里面用）

Function里 有this就是面向对象的，没有就是面向过程的

## 重排、重绘、重构

页面重构：

页面任何变化都可以称为页面重构：完全重构、细节调整

注意：简单的外部组件、测试工具的使用、预留空间（未来的更新）、编程语言、开发环境、版本控制工具（本人使用GIT）、新与稳定技术的选择、编码标准统一、资源合并（自动化工具（本人使用gulp），提高HTTP传输效率）动态编译、资源存储（减少cookies，增加本地存储、数据连接池等的使用）、响应式布局

Document.writeh和innerHTML的区别：

Document.write重排整个页面

Inner HTML可以重绘页面的一部分

浏览器的运行机制：

构建DOM树（parse） 构建渲染树（construct） 布局渲染树（ reflow/layout）绘制渲染树（paint/repaint）

重绘：（repaint或redraw）：当盒子的位置、大小以及其他属性，例如颜色、字体大小等都确定下来之后，浏览器便把这些原色都按照各自的特性绘制一遍，将内容呈现在页面上。重绘是指一个元素外观的改变所所触发的浏览器行为，浏览器会根据元素的新属性重新绘制，使元素呈现新的外观，

触发重绘的条件：改变元素外观的属性。例如color，background-color

重绘条件：重绘发生在元素的可见外观被改变，但并没有影响到布局的时候。比如，修改DOM元素的字体颜色（只有Repaint，因为不需要调整布局）

重绘重排的代价：耗时，导致浏览器卡慢

重排（重构/回流/reflow）：当渲染树中的一部分（或全部）因为元素的规模尺寸，布局，隐藏等改变而需要重新构建，这就称为回流（reflow）。每个页面至少需要一次回流，就是在页面第一次加载的时候。

重绘和重排的关系：在回流的时候，浏览器会是渲染树中受到影响的部分失效，并重新构造这部分渲染树，完成回流后，浏览器会重新绘制受影响的部分到屏幕中，该过程称为重绘

重排必定会引发重绘，但重绘不一定会引发重排。

重排、重绘、重构

页面会引起重构，任意地改变都是页面重构

重排：只要是重新加载了就会重排，比如说隐藏、尺寸、布局改变都会引起重排（也就是回流）

重绘：外观改变必定会引起重绘

优化：

1. 对DOM的操作
2. 让引起多次回流的元素，脱离文档流
3. 对样式进行修改的时候一次修改，尽量使用className