JavaScript 高级程序设计第三版

第一章 JavaScript 简介

1.JavaScript诞生于1995年，Netscape公司

2. JavaScript的组成

ECMAScript

DOM

BOM

JavaScript

ECMA欧洲计算机协会

ECMAScript就是对实现该标准规定的各个方面内的语言描述

BOM浏览器对象模型，与浏览器进行交互

DOM 文档对象模型，把整个页面映射为一个多层节点结构，与网页内容交互

第二章 在HTML中使用JavaScript

1.<script type=”text/javascript” >js内容</script>,type属性可选

<script src=”xxx”></script>,在HTML中必须要有结束标签</script>，在XHTML中可以省略

<a onclick=”javascript:alert(‘hello world’)”>click</a>

2.js代码从上到下依次解释，不要包括</script>，需要转义 eg. alert(“<\/script>”)

3.外部js文件一般带.js后缀，但不是必须的

4.现代一般吧js代码放在<body>之后，避免加载时间过长，页面空白

5.<script>属性：

defer:脚本在执行不会影响页面构造，脚本会延迟到整个页面加载之后再加载，只适用于外部js文件

async:告诉浏览器立即下载文件 ，并不保证有async属性的异步脚本执行先后顺序，只适用于外部js文件

6.MIME描述消息内容，文件类型的因特网标准

7.<!DOCTYPE html>表示是html5类型

8.<noscript>用于不支持js浏览器的替代js的内容

第三章 基本概念

1.ECMAScript区分大小写

2.标识符：第一个字母必须是字母，\_或者$,其他可以是字母，\_，$或者数字

3.js注释使用单行 // ,多行 /\*注释内容\*/ 进行注释

4.严格模式: “use strict”; //是一个编译指示， 告诉js引擎切换到严格模式

5.var 声明会变成该作用域的局部声明，没有var的会变成全局声明

6.var a=2,b=false;不能定义eval,arguments的变量，否则会出错。

7.js有5种基本数据类型：Undefined,Null,Boolean,Number,String.还有一种复杂数据类型Object,Object是一种由无序的key/value组成。

8.typeof是操作符,typeof( a ),括号不是必须的，判断类型

9.null被当成一个空的对象的引用，会返回object

10.udefined类型,eg.声明变量但未初始化

11.Null类型，只有一个值null，表示空对象指针，typeof null 会返回object，eg.var car=null;

12.Boolean类型，true、false区分大小写

13.Number类型

1)整数，浮点数都是，保存浮点类型的内存空间是整数类型的2倍，注意0.1

2)NaN与任何值都不相等，isNaN(x)可以判断x是不是数值，返回布尔型，isNaN(“10”)会自动转化成10，true会自动转化成1

3)parseInt(String,16)🡺以16进制转化，若parseInt(String)则默认十进制1234blue🡺1234

4)paseFloat(String),转化为浮点型

5)Infinity无穷数

14.常用字符字面量：\n换行 \t 制表 \b空格 \r回车

15.ECMAScript字符串是不可变得，一旦创建，它的值就不会改变，要改变某个变量保存的字符串，首先要销毁原来的字符串，然后再用新值填充该变量,几乎每个值都有toSttring()方法。toString(2)，以二进制形式转换

16.Object类型，对象就是一组数据和功能的集合，所有对象的基础类型

1)Constructor,保存当前对象的函数，即构造函数

2)valueOf(),返回对象的字符串、数值或布尔型表示

17.操作符： ++,--,~,^,&,|,<<,>>,>>>,&&,||,!=,==,>,<

18.语句：

1)if else;do while;while;for

2)for (pros in window ) { document.write(window[pros]);}//对象的属性是没有顺序的

3)label语句，eg. start: for(..){..;break start;},label主要是用于continute或者break

4)with(object){属性名的前面不用带对象名},不能在严格模式下使用

eg. with(document){write(“<br />”);}

5)switch case在比较时使用的是全等操作符，不会转换类型（===）

19.function functionName(){….}

1)函数在执行完return语句后停止，并立即退出，位于return后面的代码不会执行

2)return 后面可以不带任何返回值，函数在停止执行后将返回undefined值

3)arguments,可以利用arguments对象访问参数数组，arguments[0]代表第一个参数，具有length属性，与参数值会同步，但是内存空间是独立的。

4)严格模式下，arguments不允许重写，赋值会变得无效，即参数还是undefined.

5)ECMAScript没有重载，一个函数名只能对应一个函数，后面定义的会覆盖前面的，不存在函数签名的特性。

第四章 变量、作用域和内存问题

1.基本类型值指的是简单的数据段，而引用类型值指那些可能由多个值构成的对象。

2.Undefined,Null,Bolean,Number,String,5种基本数据类型是按值访问;引用类型的值是按引用访问的。

3.不能给基本类型的数据添加属性。

4.复制变量值：var num1=5;var num2=num1;num1和num2相互独立;复制引用时，这个值得副本实际上是一个指针，而这个指针指向存储堆中的一个对象，改变其中一个变量将会影响另一个变量，eg.var obj1=new Object();var obj2=obj1;obj1.name=”xxx”; alert(obj2.name);

5.对象放在堆中，A　instanceof B (A 是否属于B类)

6.ECMAScript所有函数的参数是按值传递的，即使在函数内部修改了参数的值，但原始的引用仍然保持未变

7.作用域的用途是保证对执行环境有权访问的所有变量和函数的有序访问，活动对象在最开始的时候只包含一个变量，即arguments对象，在全局环境中是不存在的

8.如果局部环境中存在着同名标识符，就不会使用位于父环境中的标识符

9.js具有自动垃圾收集机制，执行环境会管理执行代码执行过程中使用的内存

10.BOM和DOM对象就是使用C++以COM(Component Object Model,组件对象模型)对象的形式实现的.

10.分配给WEB浏览器的可用内存比分配给桌面应用的少，防止运行js而耗尽内存导致系统崩溃

11.一旦数据不再使用，最好设置为null，来释放引用

12.引用类型是一种数据结构，用于将数据和功能结合在一起，对象是引用类型的实例。

13.对象字面量表示法：var person={name:”xxx”,age:29}

第五章 引用类型

1.Object类型，数值属性名会自动转换为字符串

1)var person=new Object();person.age=18; //与var person={}一样

2)var person={age:18} //对象的字面量法

2.Array类型,每一项可以保存任何类型的数据，而且数组大小是自动调整的

1)var colors=new Array(); //var colors =new Array(20);指定大小

2)var colors=new Array(“red”,”blue”);

3)var colors=[“red”,”blue”] //数组的字面量表示法

A.colors.length

B.value instanceof Array ; Array.isArray(value); //确定某个变量是不是数值

C. colors.toString(),colors,colors.toLocaleString();返回数组字符串

colors.join(“||”)返回以||作为分隔符的数组字符串

D.colors.push(“black”,”yellow”) //逐个添加到数组末尾，并返回修改后数组的长度

colors.pop() //从数组末尾移除最后一项，并返回该项值，length减1

colors.shift() //从数组中移除第一项，并返回该值，length 减1

colors.unshift(“red”,”blue”) //在数组前端添加，并返回数组长度，red在blue之前

3.sort(),按照升序的方法；reverse(),按照降序的方法。是按照字符串比较大小的

sort(function(num1,num2){return num1-num2;}) //降序

4.slice(m,n) //若只有一个参数，则返回该参数指定的位置开始到结束的所有项；若有两个参数，返回m到n-1之间的所有项；若有负数，则用数组长度加上它

5.splice(m,n) // 删除数组中m到n-1之间的所有项

splice(m,n,”red”,”blue”) //从位置m，删除n项(从位置为m的开始删除)，开始插入”red”,”blue”

6.indexOf() //从数组的开头开始向后查找lastIndexOf() //从后往前找,没找到的情况返回-1

7.every(function(){…}) //对每一项给定函数，每一项true，则true

some(function(){…}) //对每一项给定函数，一项true，则true

filter(function(){…}) //对每一项给定函数，返回true的数组

forEach(function(){…}) //没有返回值，对每一项执行操作

8.reduce() //从数组的第一项开始，逐个遍历到最后；reduceRight() //从最后一项开始

9.Date类型,从1970年1月1日0时算起

1)var d=new Date();var d=new Date(“2016-01-01”);

2)Date.parse(“日期字符串”) //返回毫秒数，日期字符串格式：yyyy-mm-dd,dd/mm/yyyy,d

3)d.valueOf(),返回毫秒值，与Date.parse(d)一样

4)Date.now() //返回当前的毫秒值，可以进行计算运行程序用了多少时间

5)var now=+Date.now() //利用+把Date对象转换成字符串

6)重要方法：

getTime() //返回日期的毫秒数 , getFullYear() //取得四位数的年份

getMonth(),getDate() //返回天数，getDay() //返回星期几，getHours();getMinutes(),getSeconds()

10.RegExp类型

1)var expression=new RegExp(“[bc]at”,”g”); var expression=/[bc]at/g //两者等价

2)g表示全局，i表示忽略大小写，m表示多行模式

3)for(var i=0;i<10;i++){var e=/[bc]at/g ;e.test(“catbatccat”)}//第二次调用是从第一次匹配的末尾开始的,不一定都返回true,会循环查找

for(var i=0;i++;i<10){var e=new RegExp(“[bc]at”,”g”);e.test(“catbatccat”);}一定都返回true

4)expression.source //正则表达式的字符串表示

5)var pattern=/[bc]at/g;var text=“bcatcatbbat”;var matches=pattern.exec(text);

matches.input 表示应用正则表达式的字符串；matches[0] 与整个模式匹配的字符串，其他项是模式组捕获的匹配的字符串，即使标志位为g，每次也只会返回一项，但会从上一次匹配的末尾开始，直至结束，再循环。

11.Function类型,函数是对象，函数名是指针

1)function functionName(){…} || var functionName=function(){…};

2)使用不带括号的函数名，是访问函数指针，而非调用函数

3)在js中没有函数重载，后面的声明会覆盖前面的声明

4)解析器会率先读取函数声明，并使其在执行代码前可用，至于函数表达式，则必须等到解析器执行到它所在的代码行才会被真正的执行。

12.基本包装类型

1)引用类型和基本包装类型的主要区别就是对象的生存期，使用new操作符创建的引用类型的实例在执行流离开当前作用域之前都一直保留在内存中，而自动创建的基本包装类型对象，则只存在一行代码的执行瞬间，然后立即被销毁。

2)var num=10,num.toFixed(2); //num要先自动转换为Number才能使用toFixed(2)方法

3)String类型：var str=new String(“xxx”);String类型的主要放法：

str.charAt(n),返回第n位字符;str1.concat(str2) 字符串拼接 ;substr(m,n)从m开始，n个；

substring(m,n)，从m开始，到n-1;slice(m,n)，m到n-1的所有项；indexOf(“s”),lastIndexOf(“s”)

trim() 删除前面及后缀的所有空格，返回的是字符串的副本，原始的保持不变

toUpperCase(),toLowerCase();

match(),只接受一个参数，要么是正则表达式，要么是RegExp对象,返回一个数组，eg.var pattern=/[bc]at/g;text.match(pattern);

search(),返回第一个匹配项的索引；

replace(“xxx”,”yyy”),用yyy代替xxx,或者repalce(pattern,”yyy”),第一个参数可以是字符串或正则表达式

text.split(“,”,n) 根据，将text分割成n个长度的数组，第二个参数若没有，则最大个数

str1.localeCompare(str2),若str1排在str2之后返回正数1

4)获取url中参数值的办法

var reg = new RegExp("(^|&)"+name+"=([^&]\*)"+"(&|$)");

var r = window.location.search.substr(1).match(reg);

13.单体内置对象

1)有ECMAScript实现提供的，不依赖于宿主环境的对象，这些对象在ECMAScript执行之前就已经存在了，不必显式的实例化对象

2)Global全局对象，isNaN(),parseInt(),parseFloat()

A.encodeURI(uri),对URI进行编码；不会对本身属于特殊字符的符号进行编码

B.encodeURIComponent(uri),会用相对应编码替换多有非字母数字字符

C.decodeURI(uri),decodeURIComponent(uri)

D.eval(),接受js字符串参数，eg.eval(“alert(‘hello’)”);eval(),可以包含在环境中定义的变量

3)window对象，var s=function(){…}(); //声明函数并立即调用

4)Math对象，Math.PI,Math.ceil(a) 比a大的最小整数；Math.floor(a) 比a小的最大整数；Math.random() [0,1)的随机数；Math.max(a,b);Math.round() 四舍五入；Math.sqrt(a) 开方；Math.abs(a) 求a的绝对值；Math.pow(a,b) 求a^b

第六章 面向对象的程序设计

1.任何函数，只要通过new操作符来调用，那它就可以作为构造函数，没有new，那就是普通函数，添加到window

2.对象的constructor属性，最初是用来标识对象类型的

3.ECMAScript中的函数是对象，每定义一个函数也就是实例化了一个对象Function

4.在全局作用域中定义的函数实际上只能被某个对象调用，原型模式prototype属性可解决

1)function Persion(name){Person.prototype.name=name;} //可以让所有的对象实例共享它所包含的属性和方法，存在于实例与构造函数的原型对象之间，而不是实例与构造函数之间

2)Person.isprototypeOf(person1)来确定对象是否存在指向Person.prototype的指针

3)Person.getPrototypeOf(person1).name来取得原型对象中name的值

4)如果我们在实例中添加属性，而该属性与实例中的属性同名， 则该属性将会屏蔽原型中的那个属性，delete person1.name 将会恢复原型属性

5)“name” in person1 ,返回布尔型

6)var o={toString:function(){…};};

7)要想获得对象上可枚举的属性，可以用Object.keys(Person.prototype)

8)function Person (){Person.prototype={name:”xxx”,age:18} }//constructor属于Object而是是Person,可function Person(){Person.prototype={constructor:Person,name:”xxx”,age:18} }显式声明。

9)创建实例后修改原型属性也是可以的，实例中的指针仅指向原型，而不指向构造函数，重写原型切断了现有原型与任何之前的已经存在的对象实例之间的联系，引用的是最初原型。

5.寄生构造函数模式，基本思想是创建一个函数，该函数的作用仅仅是封装创建对象的代码，然后再返回新创建的代码

6.稳妥构造函数模式，稳妥对象指没有公共属性，而且方法也不引用this对象。

7.继承, 接口继承只继承方法签名，实现继承只继承实际的方法

1)ECMAScrip只支持实现继承，继承方法

2)call()方法，实际上是在新创建的SubType实例的环境下调用SuperType的构造函数

3)组合继承，原型式继承

4)寄生式继承：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最后再像真的是它做了所有工作一样返回对象

第七章 函数表达式

1. 定义函数有两种方法，函数声明和函数表达式。
2. 函数声明： function functionName(){…},函数声明提升，在执行代码前会先读取函数声明，意味着可以把函数声明放在调用它的语句后面。
3. 函数表达式：var functionName=function(){…};这种情况下创建的函数是匿名函数，在function关键字后面没有标识符，也叫作拉姆达函数。Name属性使空字符串。

函数表达式在使用前必须先赋值。

1. if(condition){var sayHi=function(){…}} ,最好用函数表达式
2. 递归，在一个函数通过名字调用自身。若，假设Factorial是一个阶乘，则anotherFactorial=Factorial;Factorial=null;anotherFactorial(4);会出错

1)可以使用arguments.callee(num-1),指向正在执行的函数的指针，来代替函数名Factorial(num-1)

2)可以使用命名函数表达式，eg. anotherFactorial=(function f(num){

if (num<=1){return 1;}

else

{return num\*f(num-1);}

});

6.闭包，指的是有权访问另一个函数作用域中的变量的函数。创建闭包的常见方式是，在一个函数内部创建另一个函数。

7.在全局对象中，this等于window，被函数当做某个对象的方法调用时，this等于那个对象。

8.闭包会引用函数的整个活动，即便不直接引用，也会保存一个引用，可以设置为null来回收，不占用内存

9.js不会告诉是否多次声明了同一变量，会对后续的变量声明视而不见，不过会执行后续声明的变量初始化。

10.将函数声明转换为函数表达式，(function (){…})();//可以直接调用函数，不能去掉第一个括号，若为function(){…}();则是错误的

11.在匿名函数中定义的变量，在执行结束后会立即自动销毁。

12.私有变量，任何在函数中定义的变量都可以认为是私有变量，因为不能在函数外部访问这些变量

13.单例，指的是只有一个实例的对象。

第八章 BOM

1. BOM(Brower Object Model,浏览器对象模型)，核心对象是window,表示浏览器的一个实例。
2. Window对象既是js访问浏览器的一个接口，又是ECMAScript规定的global对象，有权访问parseInt()等方法
3. 全局变量不能通过delete操作符删除，直接在window对象的定义的属性可以通过delete删除。
4. Frames：top.frames[0]//始终指向最高层的框架；parent //指向当前框架的上层框架；

Self //始终指向window，与window可以交换使用

1. window属性screenLeft //指窗口相对于屏幕左边的位置，screenTop,screenX,screen
2. window.moveTo(x,y)//接收x,y坐标值；window.moveBy(a,b)//接收的是水平，垂直移动像素值,这两个方法可能被浏览器禁用
3. outerWidth,outerHeight返回浏览器窗口本身大小；innerWidth,innnerHeight返回页面视图大小
4. window.open(),既可以导航到一个特定的URL，也可以打开一个新的浏览器窗口。

var newWin=Window.open(“<https://www.baidu.com>”,”\_blank”,”left=100px,top=100px”);

newWin.close();//关闭打开的窗口，仅适用于用open打开的窗口

1. 新创建的window对象有一个opener属性，保存着打开它的原始窗口对象，将opener属性设置为null,则切断新窗口与原窗口之间的联系，在独立的进程中运行，没有办法恢复
2. 浏览器内置的屏蔽程序阻止弹出窗口，那么window.open()很有可能就返回null
3. js是单线程语言
4. 超时调用：setTimeout(function(){…},1000) //再过1s把当前任务添加到队列中，等前面执行完之后才开始执行，会返回一个数值ID,是计划执行代码的唯一标识符，clearTimeout(TimeoutID) //取消未执行的超时计划 --执行一次
5. 间歇调用，按照指定的时间间隔重复调用，直至被取消或者页面被卸载。

var IntervalID=setInterval(function(){…},1000) //每隔1s执行一次

clearInterval(IntervalID) //取消

14.一般认为使用超时调用来模拟间歇调用是一种最佳模式，因为间歇调用的后一个调用可能在前一个调用还没结束就已经开始了。

eg.

var num=0;var max=10;

function test(){num++;console.log(num);

if (num<max){

var ID=setTimeout(test,1000);

}

else

{clearInterval(ID);}

}

setTimeout(test,1000);

1. alert(“hello”),向用户显示一个系统对话框，其中包含一个指定的文本和一个确定的按钮。用户只能在看完消息后关闭对话框。
2. confirm(“are you sure ?”) //确定返回true,取消返回false
3. prompt(“what is your name ？”) //提示框，返回字符串或者null
4. window.print() //打印当前页 window.find(“key words”) //查找key words,，返回布尔型
5. location对象：

1)location.hash,返回#后面的内容，eg.#content

2)location.host,返回服务器的名称和端口号 eg.www.baidu.com:80

3)location.hostname ,返回不带端口号的服务器名称

4)location.href,返回加载页面的完整URL,location.toString()也返回这个值

5)location.pathname,返回URL中的目录和文件名

6)location.port,返回端口号，eg.8080

7)location.search,返回？开头的字符串，eg.?q=javascript

8)location.protocol,返回协议类型

9)window.location=”url” ; location.href=”url”;location.assign(”url”);

10)每次修改location的属性(hash除外)，页面都会以新的URL加载

11)location.replace(“url”),不能通过返回前一页来返回前一页，后退按钮处于禁用状态 12)location.reload() //重新加载当前页面，可能从缓存中加载，以最有效的方法加载

location.reload(true)//强制从服务器加载

1. navigator对象，识别客户端浏览器的事实标准
2. screen对象，用处不大
3. history对象，history.go(-1) //后退一页 history.go(2) //前进2页

第九章 客户端检测

1. 能力检测：在编写代码前先检测特定浏览器的能力
2. 怪癖检测：识别浏览器的特殊行为，即存在什么缺陷(bug)
3. 用户代理检测：通过检测用户代理字符串来确定实际使用的浏览器navigator.userAgent

第十章 DOM

1. DOM(Document Object Model) ,文档对象模型，是针对HTML和XML的一个API,描绘一个层次化的节点树。
2. 节点之间的关系构成了层次，而所有页面的标记则表现一个特定节点为根节点的树形结构。
3. <htnl>文档元素，是文档的最外层元素。一共有12nodeType,someNode.nodeType
4. someNode.nodeName,对于元素节点，nodeName中保存的始终的都是元素的标签名，而nodeValue的值始终是null
5. 1)查找节点：

A)document.getElementByTagName(“xxx”) //通过标签名称

B)document.getElementByName(“xxx”) //通过元素的name属性

C)document.getElementById(“xxx”) //通过元素的id属性，具有唯一性,只返回第一个

2)创建节点

A)document.createElement(“xxx”) //创建一个具体的元素

B)document.createTextNode(“xxx”) //创建一个文本节点

C)document.createDocumentFragment(“xxx”)

3)添加，删除，替换，插入

appendchChild(“xxx”),removeChild(“xxx”),replaceChild(“xxx”),insertBefore(“xxx”)

1. Document类型,window.document.nodeType==9,document.nodeName=#document

1)document.documentElement,document.childNodes[0] //指向<html>元素

2)document.body 🡺得到的是Object

3)document.title //取得文档标题 document.title=”设置文档标题”

4)document.URL //取得完整URL document.domain //取得域名

document.referrer //取得来源页面的URL

其中只有document.domain可以进行设置，但是不能将设置为URL不包含的域

如果将两个页面的domain设置一样，它们之间就可以进行通信,可以用于frame

5)ie存在getElementById(“id”),会返回name和id的首先出现的元素，尽量不让表单的name的值与id相同

6)document.getElementByTagName(“xxx”),会返回HTMLCollection对象，作为一个动态集合，可以通过item(n)方法进行访问，或者直接使用数组访问方法imgs[n]

7)在IE中，注释也是一个节点

8)document.getElementByName(“xxx”),可以获得单选按钮的所有项，HTMLCollection

9)document.images // 返回所有<img>元素, document.forms //返回所有带有<form>元素,document.links //返回<a>中带有href属性的 ；集合中的项会随文档的更新而更新

10)document.implementation.hasFeature(“XML”,”1.0”) //如果浏览器支持给定名称及其版本的话，则返回true,可以检测Dom功能的名称及其版本号

11)document.write() 会原样写入,document.writeln() 会自动在末尾加\n,

1. Element类型 document.getElementById(“xxx”).nodeType=1

1) document.getElementById(“xxx”).toLowerCase()==”div”

2)每个元素都存在的属性：id,元素在文档的唯一标识；title,有关元素的附加说明信息；lang,元素内容的语言代码，很少使用；dir,语言方向，ltr,rtl；

3)className,与元素的class特性，指css指定的class类，class是ECMAScript保留关键字 eg.div.className 🡺返回div中的属性class值

4)div.getAttribute(“xxx”),div.setAttribute(“name”,”value”),div.removeArrtibute(“xxx”),特性的名称是不区分大小写的，只有公认的特性才会以属性的方式添加到Dom对象中，可以使用div.xxx来访问

5).有两个特殊的特性，虽然有对象的属性名，但是属性的值与通过getAttribute()返回的值并不相同。

A.style, div.getAttribute()返回的是css文本，div.style返回的是对象

B.事件处理的程序(onclick..),getAttribute()返回的是代码字符串；通过属性返回函数

6)添加一个标签

var a=document.createElement(“a”);var aa=document.body.appendChild(a);aa.href=’’

var a=document.createElement(“<a href=’#’>”);var aa=document.body.appendChild(a);

1. Text类型 nodeType=3

document.createTextNode(“hello world”) //创建文本节点

1. Comment类型 nodeType=8
2. <![CDATA[….]]>,内容纯文本，无需转义
3. document.doctype.name ,nodeTpe=10, 出现在<! DOCTYPE html>中的html

12.Attr类型，nodeType=11,var attr=document.createAttribute(“align’);attr.value=’’

13. DOM操作技术

1)动态脚本

A.动态加载的外部js文件能立即运行 <script type=”text/javascript” src=”a.js”></script>

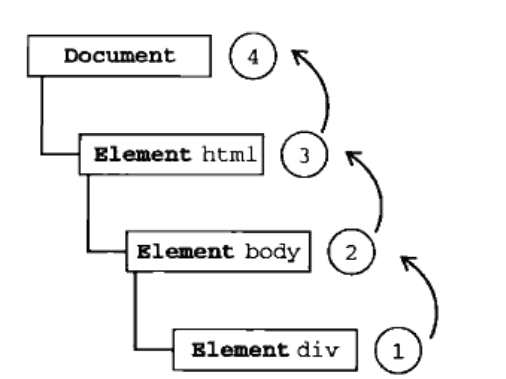
B.行内方式加入js代码，直接在页面上写

2)动态样式

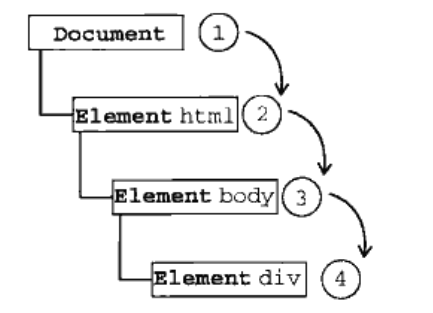
<link rel=”stylesheet” type=”text/css” href=”style.css”>

第十三章 事件

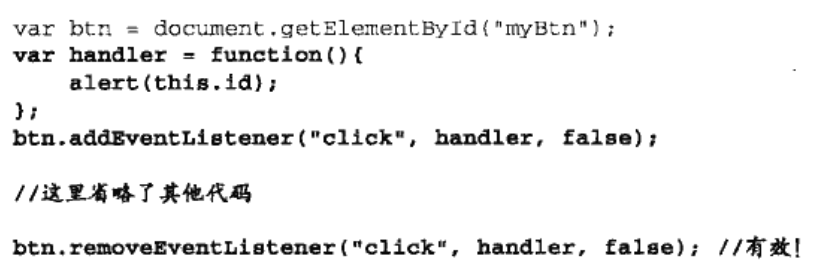
1. js与HTML的交互是通过事件的。
2. 事件流描述的是从页面接收事件的顺序。
3. IE的事件流是事件冒泡，即由内往外冒，由最具体的元素逐级向上传播到较为不具体的节点(文档)



1. 事件捕获：不太具体的节点应该更早接收到事件，而最具体的节点应该最后才接收到事件。事件捕获的目的是在事件到达预定目标之前捕获它。

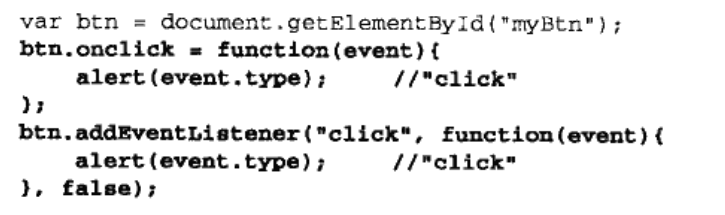


1. DOM2级事件规定的事件流包括3阶段：事件捕获阶段🡺处于目标阶段🡺事件冒泡阶段，首先发生的是事件捕获，为截获事件提供了机会；然后是实际的目标接收到事件，最后一阶段是事件冒泡，可以在这一阶段对事件作出响应。
2. 事件，就是用户和浏览器自身执行的某种动作，事件处理的程序以”on”开头
3. HTML事件处理程序，直接在标签内部进行，eg.<a onclick=”alert(“aa”)”>aa</a>
4. 在HTML中指定事件处理程序有两个缺点：a.存在时差问题，用户可能在元素一出现就触发相应事件，但当时事件处理程序有可能不具备执行条件，如函数在未定义之前就执行，js在最后才加载；b.作用域在不同浏览器中可能导致不同的结果。HTML代码与js代码紧密耦合。
5. js指定事件处理程序
6. 必须取得要操作对象的引用 var btn=document.getElementById(“myBtn”);
7. btn.onclick=function(){…;this.id;} //this 引用当前元素btn
8. btn.addEventListener(“click”,function(){….},false) //false,表示在冒泡阶段调用事件处理程序，true,在捕获阶段调用事件处理程序



1. IE事件处理程序attachEvent(),detachEvent()

btn.attachEvent(“onclick”,function(){…});

7.DOM事件对象，在触发DOM上的某个事件的时候，会产生一个事件对象event，这个对象包含着所有与事件相关的信息。 

只有事件处理程序执行期间，event对象才会存在；一旦事件处理程序执行完毕，event对象就会被销毁

9.IE中的事件对象，event=window.event

10.事件类型 document.implementation.hasFeature(“MouseEvent”,”3.0”)

1)UI事件

A.load，当前页面加载后触发；unload，当前页面完全卸载后触发

B.select,当用户选择文本框中的内容时触发

C.resize,当浏览器窗口变化大小时就会触发

D.scroll,是在window对象上发生的，实际上表示的是页面上相应元素的变化，scrollLeft,scrollTop

2)焦点事件，focus、blur不冒泡

A.blur,失去焦点时触发

B.focus,获得焦点时触发

3)鼠标与滚轮事件

A.click 左键单击，或键盘enter; B.dblclick 双击 C. mouseover 鼠标悬停

D.mouseout 鼠标离开 E. event.pageX ,event.pageY 鼠标在页面中的位置

F.shiftKey,ctrlKey,altKey,metaKey 按下则为true,否则为false

G.event的button属性：0，左键；1，滚轮；2，右键；

H.offsetX,光标相对于目标元素边界的X坐标

I.mousewheel 滚轮滚动

4)键盘与文本事件

A.keydown,用户按下任意键；keypress,按下字符键触发；keyup,释放键盘上的键时

B.键与键码，每个键都有一个对应的keycode

C.event.keyIndentifier,返回U+0000

5)DOM3新事件，textInput事件，在可编辑区域才能触发；event.data,用户输入的字符

6)复合事件，用于处理IME的输入序列

7)变动事件，DOM中某一部分的发生变化时给出提示

8)HTML，beforeunload事件

9)事件委托，对事件处理程序过多的解决方案，利用事件冒泡，只指定一个事件处理程序，做事件委托只需要在DOM树中尽可能最高的层次上添加一个事件处理程序

10)移除事件，btn.onclick=null,确保内存可以再次被利用

11)事件模拟

第十四章 表单脚本

1.document.forms可以取得页面中所有表单，最好设置name属性，可以通过document.forms[“name”]进行获取对应表单元素

2.表单提交：

A.<input type=”submit” value=”submitForm”>

B.<button type=”submit”>submit</button>

C.<input type=”image” src=”./1.gif”>

D.在js中可以先获得表单对象，再利用form1.submit();调用submit()不会触发submit事件

1. 重复提交表单的解决办法：

A.第一次提交表单后就禁用提交按钮

B.利用onsubmit事件处理程序取消后续的表单提交操作

Eg.document.forms[0].elements[0].disabled = true;

document.getElementById('btn').disabled = true;//两种方法都可以实现禁用

1. 表单重置

A.<input type=”reset” value=”reset” />

B.<button type=”reset”>reset</button>

C.js中，form.reset(),会触发reset事件

6.表单字段

document.getElementById(“form1”).elements[“name”],或数组索引方式访问

1. 禁用元素 属性disable=true
2. Focus(),blur(),autofocus(HTML5)
3. <textarea rows=”25” cols=”25”>initial text</textarea>不能指定最大字符数
4. 选择文本

var textbox=document.forms[0].elements[“textbox1”];textbox.select();

textbox.value //显示内容;textbox.setSelectionRange(0,3) //显示截取内容

selectText(textbox,0,3) //选择前3个字符

1. pattern.test(“text content”) //测试文本内容是否符合pattern的
2. 剪切板：window.clipboardData对象 getData(),setData(),clearData()
3. HTML5为type=”text”,添加了pattern属性
4. 选择框移除选项 selectBox.removeChild(selectBox.options[0]);

selectBox.remove(0); selectBox.options[0]=null

14.富文本编辑，所见即所得