d++;Hyper text markup language

任何一个标签具备的4中属性，class style id title

body 有的属性 text bgcolor\

nav  导航 section 一个区域 header footer article文章 aside  额外的标签 readonly disabled required 必须填写的意思

<input type=”file”> 如果文件上传的话

enctype =”multipart/form-data”

cascading style sheet ccs

!important 优先级 a:link a:visited a:hover 当鼠标滑过的时候

active  鼠标点击的时候 focus 获取焦点的时候w

标准 盒子型 和快捷盒子型

盒子模型是css布局 的基石

网页元素如何显示，和网页之间的关系，css元素规定所有的元素都有盒子的外形和布局空间

border 实线solid 虚线 dotted dashed double outline=none

repeat round平铺 stretch

text-overflow clip ellipsis 省略号

white-space naomal/nowrap

自动换行

dib{word-wrap:break-word}

word-break：break-all 强制英文单词断行

word:break; nomal 正常是允许在字内换行

white-space：nowrap 不换行

行内标签不能设置宽高

filter:alpha(opacity=30);

trident 三叉戟 ie 腾讯 360

beckon壁虎 火狐

presto 迅速的 Opera 渲染速度最快的引擎

webkit safari chrome内核原型 苹果自己的内核

blink Google 和Opera开发的浏览器内核

line-height

clear=both   清除浮动影响

position fixed 的包含快是视窗本身 absolute 是相对于包含快的位置定位，relative 是相对于原来的位置

 同级的话后定义的标签会覆盖先前的标签，

在布局中relative 是占空间的 absolute 是不占空间的

在JavaScript中声明的var声明的变量 会加载时在最开始的时候声明，赋值则会留在原地，而function则会将所有的都提前，包括括号里面的；先变量后方法；如果不加var就开始赋值，进行提前调用的话就会报错，如果对变量没有赋值就进行调用的话会显示undefined 不会报错；如果对var声明后的变量在用const进行声明的时候会报错，而如果再用var声明的话则不会报错；

const 给常量命名，一经命名就不会改变, 会在加载的时候就会加载，但不会有预声明的过程；如果提前调用的话会直接报错，const必须在定义的时候就赋值，否则报错,对于const进行声明的变量如果错误整体都不会显示；

数据类型

1. 原始数据类型， 直接保存在变量本地的数据类型

5大类 Number String Boolean undefined(没有进行赋值) null （不指向任何地址） typeof a 检测a的数据类型

1. 引用数据类型 不直接保存在变量本地的数据类型

number

console.log(Infinity) 输出为Infinity

0/0 NaN 是 number类型的

（2-1.7）.toFixed(2) 是1.3

所占空间大小和文字大小没什么关系

var a=5,b=6;相当于var a=5; var b=6;

string - 自动将两边的转换为数字

字符串内容一旦创建就不能改变，只能新建字符串去赋值；

console.log(1+true) 输出为2

NaN和任何字符进行计算结果都是NaN NaN 和字符串运算时字符串 和方法名运算时方法和NaN

toString(); 括号里面可以有数字，表示以数字的进制

String（）

Number（）

parseInt(string) 从开始将String转换成int之后的就不计了

parseInt(null) NaN

parseInt(undefined) NaN

parseFloat(32adfa) 结果是32；

Boolean()

除了：”” 0 null NaN undefined 其他的都是true；

数据类型比较

1》俩个字符串比较，依次比较两个字符串中的unicode码

如“3”>“10”

2》NaN和任何元素比较 < > =都是false

和任何元素比较！=都是true；

isNaN（）方法判断数据是否是NaN；

如果说X是数字后者可以被隐式转换为数字，返回false

如果说X不是数字后者不可以被隐式转换为数字，返回false

3》undefined==null 结果是true；

==带有隐式转换，===不带有隐式转换

3<<4 左移 3乘上2的4次幂 >>是除以2的几次幂

n+=9; n+=n+=8 结果为24

在for里面var一个变量，相当于在全局设置了一个变量

在for循环上定义一个名字 waiceng：

break waiceng;

跳出外层循环；

break是跳出当前循环；循环指的是for循环，if不算；

continue waiceng;

continue 是跳过continue之后的

debugger;

undefined 定义了没有赋值 对于方法没有return 会出现undefined

var b=3;

function a(){

console.log(b);

b=4;

console.log(b);

}

a();

console.log(b); console 显示 3 4 4

var b=3;

function a(){

console.log(b);

var b=4;

console.log(b);

}

a();

console.log(b); console 显示 undefined 4 3

arguments 是接收所有参数的类数组对象；即使在定义的时候定义参数，在调用的到时候arguments也能接收所有的参数；

var 函数名 = new Function(“a”,”b”…..”函数体”);

闭包 ：函数变量可以被隐藏在作用域链之内，看起来像是函数将变量包裹起来；

1. 定义外部函数来封装被保存的局部变量、
2. 定义内层函数 执行对外部函数局部变量的操作

（函数内的数据想被外部获得，只能用return；）

3．外层函数返回内层函数对象，并且外层函数被调用时，把结果保存在 全局变量中

什么时候使用闭包：反复使用局部变量，又避免了全局污染；

闭包的特点：1函数嵌套

2 内层函数使用了外层函数局部变量

3 内层函数对象被返回外部，在全局反复调用；

for (var i = 0; i < 3; i++) {

var a = function(n){

setTimeout(

function(){

alert(n)

},1000)

}

a(i);

}

function outer(){

var n=1;

var inner =function(){

console.log(n++);

}

return inner;

}

var inn = outer();

inn();

inn();

inn();

闭包就是两个方法，内方法调用外方法的局部变量，内方法返回函数的地址，实线方法内外数据的调用；要看是否立即执行如果是就是随之改变的变量；

程序=算法+数据结构

数据结构：程序中数据存储结构，好的数据结构可极大提升程序执行效率，

好的数据结构： 相关数据集中存储，集中管理

数组：一组连续变量组成的集合，统一起一名字；

var 变量名 = 【】

var 变量名 =【“值1”，“值2”】

var 变量名 =new Array（）；

var 变量名 = new Array（n）；

var 变量名 = new Array（“值1”，“值2”）；

var week = null 表示主动释放一个变量引用对象 而不是将引用的对象的值变为null；

typeof arr【】是object; arr instanceof Array;

js数组下标越界不会出错

为不存在的位置取值不会出错也不会增添新元素，返回undefined；

一下结果不是true的是（）

A var a=0;b=-0; B var a = NaN; b=NaN;

C var a=null;b=undefined D var a=[] b=false;

for (var i = 0,a=0; i <3,a<4; i++,a++) {

console.log(a,i)

}第二个中间的逗号表示或，只要其中的一项满足就执行括号里面的

console.log("11"+1-"2"); 输出为109；

console.log("11"+1-undefined); 输出结果为NaN

console.log("11"+1+undefined); 输出结果为111undefined；

console.log("11"+1-null); 输出结果为111；

索引数组

关联数组: 可以自定义下标名称的数组；

在关联数组中length会失效；

想精确定位某个下标位置的时候

var a=[];

a["ha"]=100;

a["he"]=90;

a["aa"]=22;

console.log(a); 输出 [ha: 100, he: 90, aa: 22]

for in

for(var key in a){

console.log(key+a[key]);

}

两个数组相加得到的是字符串

数组通过toString 进行转换的时候会有逗号；所以用join（）；

优化，频繁拼接字符串容易造成内存浪费，先将拼接的字符串放入数组中然后再进行拼接；

var arr2=arr.concat("",arr1,"").join("");//在arr后添加括号内的字符串然后赋值给新的数组；

var arr2=arr.slice(1,2); 取头不取尾 1 2 表示下标；如果第一个值是负的话表示倒数

var arr2 = arr.splice(1,4,"2"); aplice 是在原数组上进行操作减去从1开始的4个数组然后替换成“2”; arr2 是删除的数组 所以可以用splice 代替slice；

reverse 反转 是在原数组上改变

lice concat 不改变原数组，splice reverse sort 改变原数组；slice 是切下的意思，splice是粘结的意思

indexOf（n，start） 查找从start开始第一个n的数组下标;如果没有就返回-1

sort() 默认将一切元素转换成字符串进行排序，原数组改变

// var arr2 = arr.push(234)；从数组的末尾添加返回数组的个数

// var arr2 = arr.pop(); 从数组的末尾删除，返回删除的项

// var arr2 = arr.unshift(0); 从数组的开通添加 返回数组的个数

// var arr2 = arr.shift(); 从数组的末尾天骄，返回删除的项

以上的原数组都改变；

一维数组 下标越界不会报错

二维数组 列下标越界不会报错，行下标越界会报错；

Number String Boolean Array Math Date RegEx Error Function Object Global 内置的11种对象

Js 常见错误

SyntaxError 语法错误

ReferenceError 引用错误，找不到变量对象；

TypeError 类型错误 错误的使用了对象中的方法

RangeError 参数超出范围

URLError url路径错误

Eval 调用Eval函数时出错

字符串对象：多个字符组成的一个只读的集合（数组）

凡数组操作中，不修改原对象的API字符串都可以直接用；

转义字符：当字符串内容和程序中特殊符号冲突时或字符串包含功能字符时

用\转化字符意义

\t \n

大小写转换 toUpperCase toLowerCase

str[i] str.charAt(i) str.charCodeAt(i)

str.indexOf("",n)如果第一个里面不写 会返回n；

str.substring(); 与slice（）相同 str.substr(起始位置，n个) slice（开始位置，结束位置）与数组的不同，第一个可以是负值，但此时第二个不能有值

split(“”) 返回的是数组

固定套路，将字符串打散成字符数组 str.split(‘’);

replace(‘’要替换的字符串”,”替换的内容”)替换只能执行一次，需要全局匹配，使用正则

indexOf 和 replace 都查的是第一个

Math对象

1：取整 parseInt() round (四舍五入) Math.ceil()上取整 floor()下取整

2 Math.pow(x,y) x的y次方

3 Math.max(“”,””,””)；Math.min();括号中不能放数组；会返回NaN；

Math.min.apply(Math,arr) 返回数组中较小的数

4 随机数 Math.random();

var aa = new Date(); aa是当前电脑的时间， 月是0到11

console.log(aa.getMonth());

console.log(aa.getDay());

console.log(aa.getFullYear());

console.log(aa.getDate());

console.log(aa.getHours());

console.log(aa.getMinutes());

console.log(aa.getSeconds());

DHTML 动态网页技术

Html css js

7大对象

window 当前浏览器打开的窗口

document 当前正在打开的html文档

navigator 封装了软件的版本信息和部分设置

history 封装了当前窗口成功浏览过的网址历史记录

location； 当前窗口栏中的地址

screen 当前显示器

event 事件对象

HTML|XHTML|DHTML|XML 可扩展的标记语言

Js

Es ECMAScript 欧洲计算机协会

BOM ;浏览器对象模型，用来访问和操作浏览器窗口，没有人来制定标准，但被广泛的支持

DOM; 文档对象模型 w3c组织定义和访问的操作标准，可以对网页对象做任意修改

核心DOM ；所有结构化文档通用API

DOM树 在html文档中DOM将文档视作一颗DOM树

文档中的元素，属性、文本、注释都被看做成一个节点-Node节点

当页面被加载进内存的时候，浏览器为网页创建一个document对象

所有节点都是document的字节点

document 对象封装了网页中所有子节点的增加，删除、修改

[<!DOCTYPE html>, html] 是document的childNodes

document.getElementById("last");

document.getElementsByTagName("li"); 多个s，获取节点的元素

document.getElementsByName(“i");获取相同name节点的元素，并不是所有的变迁都具有name属性，ie8兼容问题

document。getElementByClassName（）f

childNodes 结果是个数组

nodeName用来获取元素节点标签名称

nodeValue 用来获取文本节点文本内容，只用于树的结尾

nodeType 用来判断到底是什么节点（元素 1，文本 3）

空格 回车（回车也是空格）的节点显示是#text 文本的节点是字符串

firstChild 获取当前元素的第一个子节点

lastChild

parentNode

previousSibling 获取当前节点的上一个同级节点

nextSibling

ownerDocument 获取文档根节点,相当于document

元素树：仅由元素组成的书

.children

.firstElementChild

.lastElementChild

.parentElement

previousElementSibling

nextElementSibling

.innerHTML 输出当前元素的子元素 兼容问题， table tr 不支持

如果ab是li ，那么ab.innerHTML 就代表li里的值

如果ab是ul 那么 ab.innerHTML 就是 <li class="aa">1</li>

<li>3</li>

<li class="aa">2</li>

<li id="fourli">4</li>

innerText 火狐不认

.textContent 开始和结束标签之间的内容 ie8 不支持

所以写的时候写成xx.textContent||xx.innerText

var a =document.createElement("input")

appendChild(a)

.insertBefore(b,a)

父容器.insertBefore(新增加元素，旧元素)把新元素放到旧元素之前，父容器需要有旧元素；

removeChild(a);

replaceChild(新增加的元素，旧元素)

操作元素样式属性

1 通过HTMLElement提供的标准属性

对象.Style.属性值 = “”

input.style.background="yellow"

2 通过setAttribute getAttribute

input.setAttribute("type","button");

function getStyle(obj,attr){

if(obj.currentStyle){

return obj.currentStyle[attr];(ie兼容)

}else{

return getComputedStyle(obj,false)[attr];（正常获取）

}

}

样式有3种内联 内嵌 外部 style只能适用于内联的 而currentStyle是只适用于ie 而getComputedStyle(obj,pseudoElt)[attr] 对内嵌和外部都适用 其中的pseudoElt参数的意思是after before 之类的伪类，如果不用伪类的话用null；

针对闭包问题ECMAScript 6 新出了一个let

var 的作用域是最近的函数块，let的作用域是最近的封闭块；for（）属于封闭块，如果在for中var 一个变量是全局变量，而let不是，不属于全局变量、

例如

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

setTimeout(

function(){

alert(i)

},i\*1000)；

}是显示0123，而用var的话显示4444 要想显示0123需要以下

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

var b = function(n){

setTimeout(

function(){

alert(n)

},1000);

}

b(i);

}

var bb = document.getElementsByTagName("li");

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

bb[i].ab = i;

bb[i].onclick=function(){

console.log(this.ab);

}

}

var c =setInterval(fun,seconds);

clearInterval(c)

function c(){

console.log(1);

setTimeout(c,1000);

}

setTimeout(c,1000);

延迟加载js的思想

<script type="text/javascript">

function downloadJSAtOnload() {

var element = document.createElement("script");

element.src = "defer.js";

document.body.appendChild(element);

}

if (window.addEventListener)

window.addEventListener("load", downloadJSAtOnload, false);

else if (window.attachEvent)

window.attachEvent("onload", downloadJSAtOnload);

else window.onload = downloadJSAtOnload;

</script>

浏览器窗口的内部高度 window。innerHeight

var w=window.innerWidth

|| document.documentElement.clientWidth

|| document.body.clientWidth;//下面两个是兼容问题ie

关闭当前窗口 window.close() window.open();

screen screen.availHeight 减去界面特性，比如窗口任务栏 整个浏览器的最大高度

location location.href 返回当前页面的url <file:///Users/apple/Desktop/web/html/1.html>

location.pathname 属性返回 URL 的路径名。

/Users/apple/Desktop/web/html/1.html

location.assign(“http://www.”)

Prompt(“文本”，“默认值”)

confim（“文本”）

事件，用户和web页面进行交互时，解释器会自动创建相应的event对象，用来描述事件信息

事件处理函数：相应事件调用的函数；

3中事件

1. 鼠标事件
2. 键盘事件 onkeydown onkeyup
3. 状态事件 onload

绑定事件：

1. html 《标签 onclick=“js语句”》
2. js 对象。onclick = 函数对象
3. DOM标准情况 addEventListene（“事件名”，函数对象，是否在捕获阶段触发）

Ie8 对象。attachEvent（“on事件名”，函数对象）；

元素目标对象 var src = event.srcElement||event.target

事件周期；从浏览器捕获事件后，一直到最后一个事件触发完所经历的过程

3个阶段：

1. 捕获（捕获由外到内） 从最还曾html开始由外向内逐层记录每个元素绑定的事件处理函数，到目标元素为止
2. 事件触发 ，自动执行目标元素绑定的事件处理函数
3. 事件冒泡？ 从目标元素父元素开始逐层向上执行每层处理函数，直到html

event对象 作为第一个参数获取来

获取：var e = arguments[0]||window.event (ie8)

function（event）

html 《标签 onclick = “fun(event)”;>

var div =document.querySelectorAll(“div”);

取消事件冒泡

if(e.stopPropagation){

e.stopPropagation();

}else{

e.cancelBubble =true; (取消ie的事件冒泡)

}

如果用addEventListener 方法 第三个参宿是true的话，此时用上面的方法，只会执行最外面的事件

事件坐标：相对于浏览器显示区域坐标 clientX clientY 是相对于浏览器

相对于元素左上角位置坐标 offsetX offsetY 是相对于当前div

offserWidth 宽度 offsetHeight 高度

clientWidth clientHeight

clientWidth 包含 width padding

offsetWidth 包含 border width padding

offsetLeft 找的是定位的父元素距离左边的距离

var obj={

name:”aa”，

“ba cbc”:11，

}

obj.name obj[“name”] 使用后一种能取到不加引号的 定义属性的时候后面加逗号 属性是aaa或者是“aaa”是没有什么区别的

var a ={} 和var a =[]区别 a[0]=33;显示不同 前者显示0：33 后者只显示33,而且用a.0则没用

obj.aa = setInterval(function(){},100); console.log(obj.aa)结果是1；

如果再写一遍结果就是2；

如果一个对象的函数赋给另一个对象的函数的话，里面的this指的是新的对象的this。如果里面写setTimeout（）的话不用写window

function start(){

alert(this)

}

start.apply(obj);

aa.apply(obj,[2,3,4]);

aa.call(obj,2,3,4)

区别方法aa的参数的传递不同

function Abc(name,age){

this.name = name;

this.age = age;

}

var aa = new Abc("aa",4);

var person ={};

Object.defineProperty(person,"name",{

configurable:false,//是否可以删除

writable:false,//是否可以更改

value:"fadfa"

});

person.name="after";

delete person.name;

console.log(person.name);

var obj ={};

Object.defineProperties(person,{

name:{

set：function（value）{

this.\_name = “aa”‘;

},

get:function(){

this.\_name = “bb”;

}

age:{}

});

var aa = {

get age(){

if (this.\_age>18) {

return this.\_age;

}else{

return "aaa";

}

},

set age(val){

this.\_age = val;

}

}

aa.age=17;

console.log(aa.age);

var obj = {

a:"aa",

b:"bb"

}

obj.hasOwnProperty("a");

console.log(“a” in obj);

// 确定原型和实例的关系

// instanceof 操作符，

// 只要用这个操作符来测试实例与原型链中出现过的构造函数

// 结果就会返回 true

function Car(){

}

function Bmw(){

}

Bmw.prototype = new Car(); //继承

var X5 = new Bmw()

// console.log(X5 instanceof Bmw); //true

// console.log(X5 instanceof Car); //true

// 第二种方式

// 使用 isPrototypeOf() 方法，同样，只要是原型链中出现过的原型，

// 都可以说是该原型所派生的实例的原型，因此 isPrototypeOf()

// 方法也会返回 true

console.log(Bmw.prototype.isPrototypeOf(X5));

str.split(/\s+/);

str.indexOf(“cd”);

str.search(“cd”);

ser.search(/cd/);

var str = "aabaca";

console.log(str.split(/a+/));

输出结果["", "b", "c", ""]

indexOf 没有不支持正则，如果没有就返回—1；

var str ="小电影，aaaaa小电影";

1. console.log(str.match(/小电影/));
2. Array[1]
   1. 0:"小电影"
   2. index:0
   3. input:"小电影，aaaaa小电影"
   4. length:1

console.log(str.match(/aa/g));

1. Array[2]
   1. 0:"aa"
   2. 1:"aa"
   3. length:2

console.log(str.match("小电影"));

与1一样；

var aa = function(){

return "aaa";

}

console.log(str.replace("小电影",aa));

var reg = /abc/g;

var str = "asdaffabc";

console.log(reg.exec(str));

console.log(reg.test(str));

var demo = prompt("不住徐输入空格");

alert(!(/\s/.test(demo)));

var reg = /[a-zA-Z0-9]/;

console.log(reg.test('中国'));

var reg = /[^a-z]/; //代表有其中的一个就为true //里加^没什么用；

console.log(reg.test("ABC"));

//分界 ^以什么开头

var reg = /^abc/;//判断是否以abc开头

console.log(reg.test('abcaa'));

//$ 以什么结尾

var reg = /xioabai$/;

console.log(reg.test("aaaxioabai"));

var reg = /\d/; //[0-9]

console.log(reg.test("aafaf19"));

var reg = /\D/; //[^0-9]

console.log(reg.test("aaaa11"));

string.trim() 去掉首尾两端的空格