JS笔记整理

写程序前先写思路

在配置config的时候，加入这句解决缓存问题

JQ ajax的url是相对于html文件而定

git需要gitignore文件过滤，sass需要在文件名前添加\_用来过滤

gulpfile.js用来进行统一性操作



Dist文件夹用来存放用gulp合并后的东西

判断对象是否jquery对象，用元素.jquery

Ajax获得数据后要遍历数据，再开展工作

构造函数直接调用的就是静态方法

写js之前可以先想想最后如何输出，然后再按照最后输出的参数考虑各步骤

当有参数的函数，尝试新声明变量计算结果，因为就是计算参数里面的东西

用变量保存输出的信息，在用innerHTML=变量比较好

记住页面上只有那个元素或功能出现才能获取，要分析代码发生的顺序

判断对象里面的属性是否存在运用obj【attr】 === undefined？

如果循环里面放函数，则函数里面的变量会获得循环后的值，所以循环里面放事件，先设置自定义属性保存相关属性的值再进入事件函数

onkeyup事件用switch循环方法

元素在隐藏状态时无法获取宽高

表单的提交是浏览器默认行为，可用preventDefault（阻止），判断完条件后要return阻止代码执行，或直接return false

表单验证要用开头结尾

动画：出现抖动肯定是定时器问题，就要分析什么时候清楚哪个定时器

轮播图外层定时器控制target值，内层控制到达target的速度值,利用idx计算target值，speed = target – offset

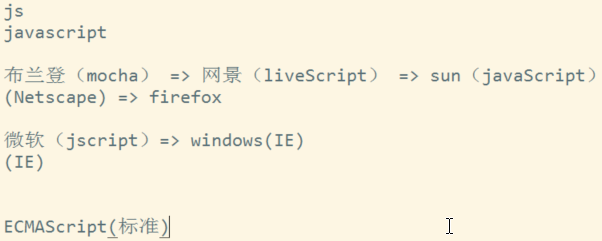
想要元素同步，例如图片转换圆点也高亮，立即要想到数组，第idx个图片切换了，第idx个圆点也要切换，XX.[idx]=xxx

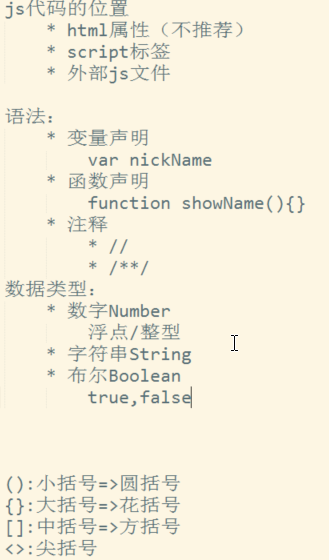
循环里面如果想要某一元素跟循环元素同步运动，需要添加idx标识

生成的元素要绑定时间建议使用事件委托



Javascript=ECMAscript+BOM+DOM





JS运用方式

1 放在head的 <script></script>

2 <script type=”text/javascipt”src=””></script>

3 html属性

JS语法

输出：alert（’’）

变量：数据的容器

单行注释// 多行注释/\* \*/（不能嵌套）

声明一个变量

关键字： var

变量名：name

数据：10

1 var name=10；//把10赋值给变量name

2 var = name;

name=10;

3 var a,b,c;

Var a = 1, b = 2, c = 3;

JS代码规则

变量命名规则

1数字，字母，\_ ,$

2 第一个字符不能为数字

3 不能是关键字或保留字





驼峰命名法

Var myNickName；

变量名尽量接近本意

保持代码缩进

JS语句的末尾尽量加上分号

运算符两边都留一个空格：如var n = 1 + 2;

数据类型：类型都是大写首字母

数字：Number

字符串：String（用单/双引号包住）

布尔类型：Boolean（true/false）

引用数据对象

对象 object

数组 array

特殊数据类型

Null（类型）：null（值）空值

Undefined：undefined

NaN（not a number）： \*不代表任何值，也不等于任何值。\*任何数据和它运算都返回NaN \*数字转换不成功，就会返回NaN

parseInt（）

parseFloat（）

变量通过ID获取元素

var num1 = document.getElementById(‘num1’)（括号里面的num1是元素ID）

声明函数

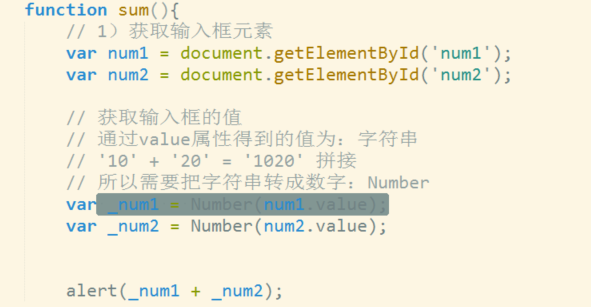
function sum（）｛｝



函数的执行

格式：函数名（）

sum（）



步骤：

第一步：先获取元素

第二步：获取元素里面的值

第三步：把字符串转成数字

第四步：输出值

通过value得到的值为字符串

数据类型的转换

其他类型转数字：Number（）

其他类型转字符串：String（）

其他类型转布尔值：Boolean（）数字转布尔类型只有0是false值

判断数据类型

typeof（变量名）

typeof（null）🡺object

typeof（undefined）🡺undefined

typeof（NaN）🡺number

输出方式

alert（）警告框

document.write（）

confirm（）确认框

prompt（）对话框

console.log（）输出到控制台

赋值运算

+= -= \*= /= %=

var不可以和赋值运算符一起使用

例如var link +=sum；会报错

关系运算（返回布尔值）

==(等于), !=(不等于)

<(小于)、>(大于)、<=(小于等于)、>=(大于等于)

===、恒等于/全等于，比较的时候要求值和类型都相等（不会进行类型隐

式转换）

!==、不全等于

关系运算符的比较规则:   
1. 数字和数字比较, 直接比较大小  
2. 数字和字符串比较, 字符串转换为数字后再比较  
3. 字符串和字符串比较, 进行字符的ASCII码值比较

隐式转换  
如果运算不能进行下去，内部就会尝试进行数据类型的转换  
支持隐式转换的运算：逻辑运算、关系运算、数学运算

Math.random（）获取0-1的小数

逻辑运算符

&&与

||或

！非

条件判断语句

<1>if条件语句

if（条件）｛

｝else if（条件）｛

｝else｛｝

<2>三元运算

格式：条件 ? 条件成立代码 : 条件不成立代码

var a=20;

var b = 50;

var sum = a>b ? a-b : a+b;

<3>switch语句：表达式和case的条件要恒等才会执行（数据类型、数值等都相等）

switch(表达式/true) {

case value1:

//如果表达式的值恒等于value1，代码从这里开始执行

break;

case value2:

//如果表达式的值恒等于value2，代码从这里开始执行

break;

case value3:

//如果表达式的值恒等于value3，代码从这里开始执行

break;

case value4:

//如果表达式的值恒等于value4，代码从这里开始执行

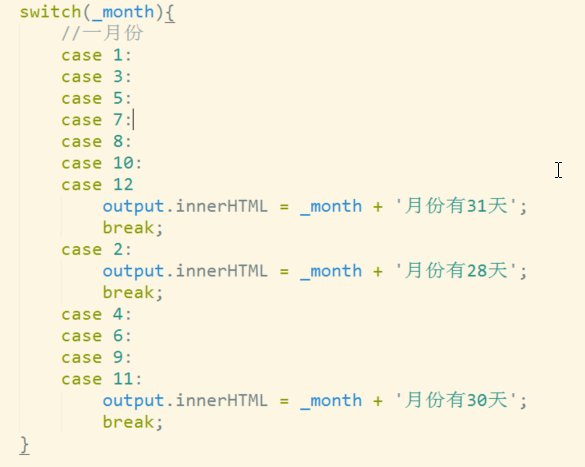
break;

default:

如果以上条件都不成立，默认执行这里的代码

}

case穿透



switch语句在比较值时使用的是全等操作符，因此不会发生类型转换

case: 当符合条件时，会从符合条件的那一条case语句开始，依次顺序执行

break: 跳出switch语句

default: 当所有的case都不满足的情况下会执行defalut下面的语句

循环语句

<1> while循环

//变量初始化

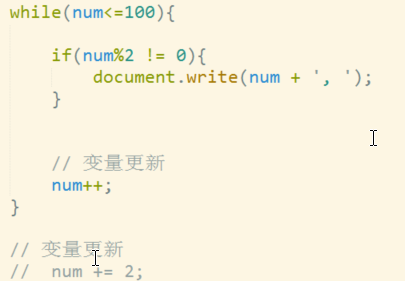
while(条件){

//条件成立就会不断地执行这里的代码，直到条件不成立

//所以这里一般会伴随着变量的更新

}

打印100以内奇数





<2>do…while

//变量初始化

do {

//不管条件是否成立，先执行一次这里的代码，再进行条件判断，如果条件依然成立，则再次执行这里的代码，依此类推

//所以这里一般会伴随着条件的更新

} while(条件)

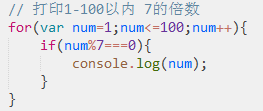
<3>for循环

for(变量初始化; 条件判断; 变量更新){

//循环条件成立，则执行这里的代码

}执行顺序：先初始化，满足条件进行代码执行，再执行变量更新，满足条件再执行代码

\*分号必须写



<4>循环跳转

break：//退出当前整个循环

只能在循环语句中使用和switch中使用

循环体中位于break后的语句不会被执行

在多层循环嵌套中，一个break语句只向外跳一层

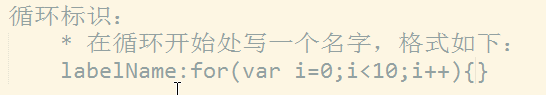
continue：//跳过本次循环，继续下一次循环

只能在循环语句中使用，

结束本次循环（即跳过循环体中下面尚未执行的代码），接着执行下次循环。

label：给循环代码添加标识

break和continue后如果带标识，则跳到标识所在循环





break与continue的区别？

PS：

- 固定次数的循环，用for循环，不固定次数用while循环

- 死循环

死循环就是重复执行代码, 不会停止.

死循环会造成程序卡死甚至崩溃等问题, 所以我们写代码要避免死循环

<5>嵌套循环

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

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

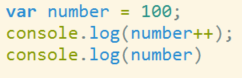
console.log(i,j);

}

}

第二天笔记整理

后置++，得到加1之前的值，number++得到number，但number的值已经加1



前置++，++number得到加1之后的值

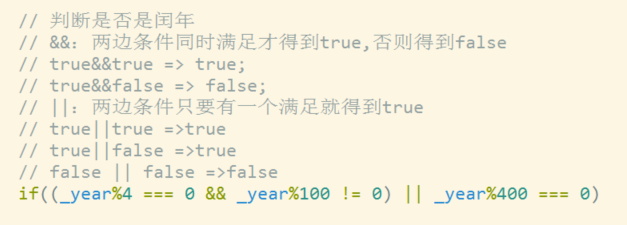


&&逻辑与 两边条件同时满足 返回true

||逻辑或 两边条件只满足一边即 返回true

&&比||的优先级高

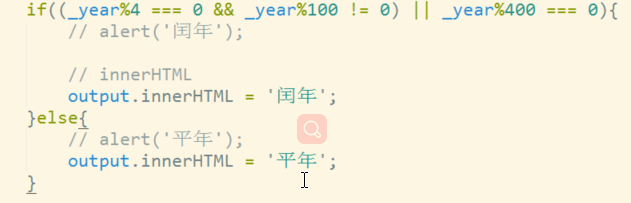
！非



改变元素的内容

InnerHTML

获取后的元素名.innerHTML=’内容’（把内容写入HTML元素里面）



0b+数字二进制

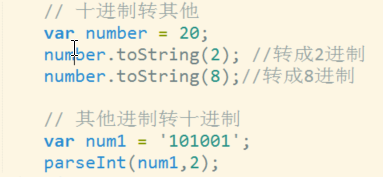
0o+数字八进制

0x+数字十六进制

十进制转其他：number.toString（x,进制）

其他转十进制：var num=‘ 数值 ’；

parseInt（num,进制）；



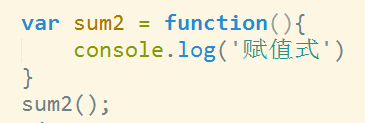
拓展

eval（）函数可计算某个字符串，并执行其中的代码

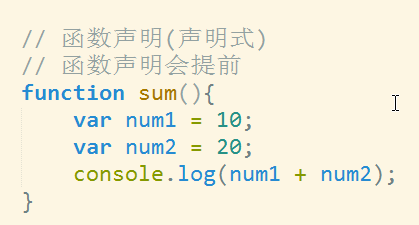
toFixed（num）保留小数点后几位数，num是几位数，返回的是字符串

函数

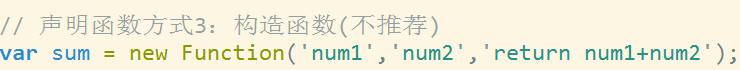
赋值式



声明式：先保存函数，逐行执行代码



构造函数

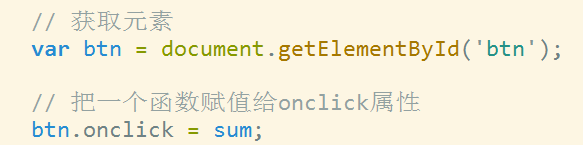


在js中调用html属性：事件驱动

函数的执行

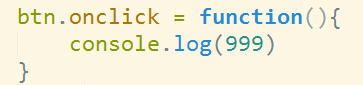
手动调用:

sum();



事件驱动

onclick调用匿名函数



格式：元素.事件 = 函数名;

buton.onclick = sum;

常见事件触发函数

onclick：点击事件

ondblclick：双击事件

onmouseover：鼠标移入事件

onmouseout:鼠标移开事件

onchange：内容改变事件（一般用于表单元素）

onkeyup：键盘按键弹起事件

window.onload页面加载事件

onselect文本框内容被选中事件

onfocus光标聚集

onblur光标离开

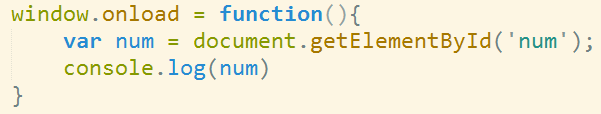
onscroll滚动事件

onreize窗口大小改变时触发

oninput输入的时候触发

\*页面元素加载完成后执行

\*作用：延迟代码执行



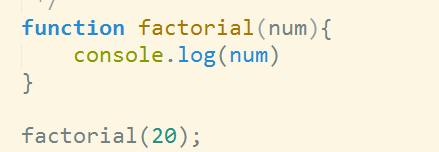
通过js改变样式



注：把css样式里面有-的写法变成驼峰法

函数的参数

function factorial（num）｛｝num就是参数



num：形参（函数定义）相当于在函数内部生成了一个变量

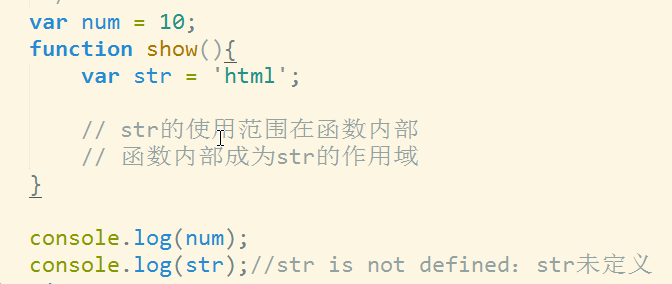
20：实参（函数调用）

函数的参数可以是任意的数据类型

作用域：变量的使用范围

\*局部作用域

\*全局作用域



定论1：局部变量：定义在函数内部的变量只能在函数内使用

定论2：全局变量：定义在函数外的变量能在任意地方使用

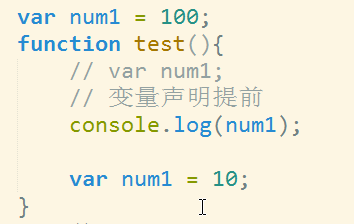
就近原则

1先在函数内部寻找变量

2如果找不到，就到函数外找变量

声明变量不负责，变量默认为undifined

变量声明提前，但是不提前值

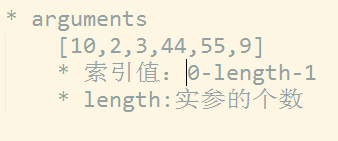


函数字符串长度属性：.length

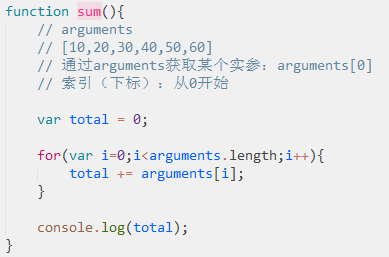
arguments

函数内部隐藏的对象（是一个类数组），保存着实参的信息

length: 实参的数量



callee: 对函数本身的引用

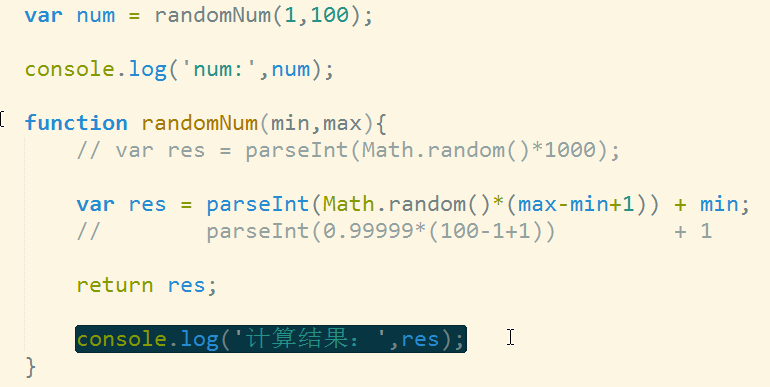


函数执行完成后，默认返回一个undefined

return作用

\*终止函数的执行

\*如果return后面有值，则把这个值返回到函数执行的地方，可理解为强制执行return后面的东西



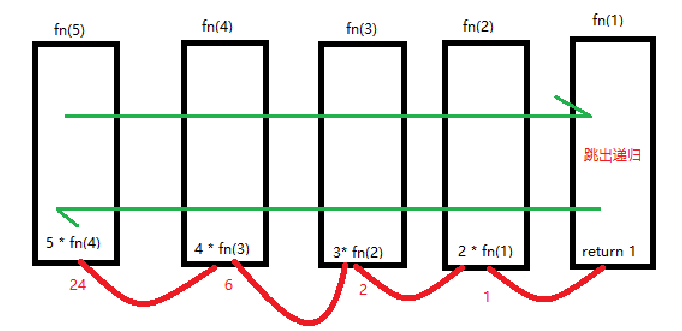
函数的递归

函数可以自己调用自己, 成为函数的递归调用，一定要给一个跳出值

判断递归：就是判断什么时候不需要计算

递归调用的过程:

1. 首先去找临界值，即无需计算，获得的值(一般是返回该值)。
2. 找这一次和上一次的关系(一般从后往前找)
3. 假设当前函数已经可以使用，调用自身计算上一次的运行结果，再写出这次的运行结果
4. 递归示意图：



Js里面的css样式，如果将js变量放到style里面，需要用’+a+’的方式链接，单双引号取决于外部是单引号还是双引号。

函数里面的函数不能直接被调用，如果加了window.onload，要添加window.变量名 = 函数名才能使用

如果函数里面有声明了的东西但是没给任何操作，函数是不会执行的。

# 数组：Array

创建数组方式：

字面量：var arr = [10,20,30];

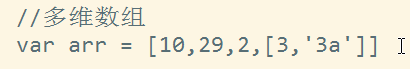
空数组：var arr = []；

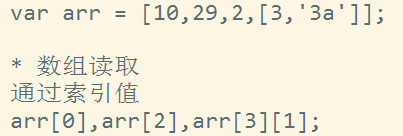
构造函数方法创建数组：var arr = new Aarray（）；

索引值：从0开始

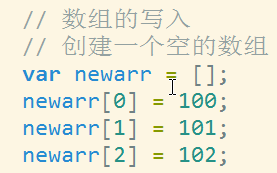
数组的读取（如果索引值没有对应的元素，得到undefined）：arr[值];

多位数组：（数组嵌套数组）

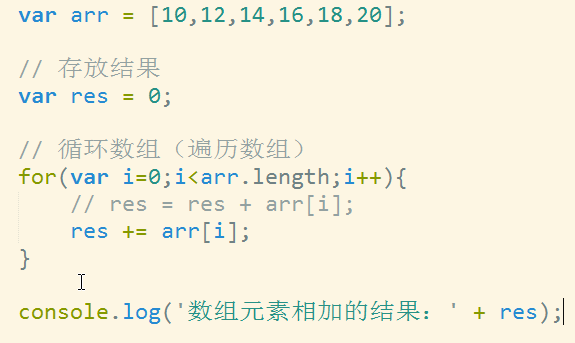


arr[3][1]读取3

数组的写入：



数组遍历：\*字符串可以像字符串一样遍历



数组的方法：

数组push（）：往数组后面加值，括号里面的值就是添加到数组里面的值，添加后返回新的长度，就是按顺序排列。

数字unshift（）：往数组前面加值，与push（）相反。

如果用一个变量接收一个数组，返回的是数组的长度（就是数组内容的数量）

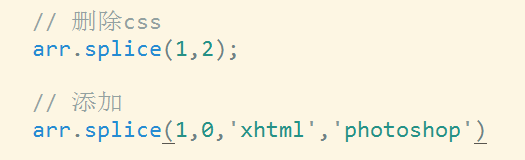


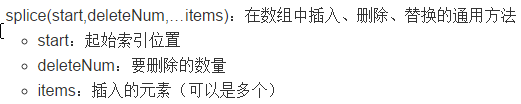
删除：

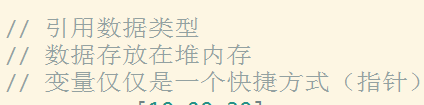
pop（）：删除最后一个元素

shift（）：删除第一个元素

删除的方法，返回被删除的元素，包括splice

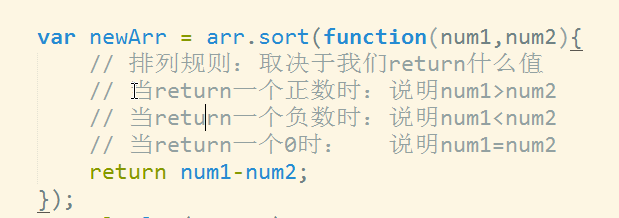
Splice





sort（function（num1,num2）{}）：排序，仅仅排列第一个数字

按顺序排列方法：



reverse（）：颠倒数组的顺序

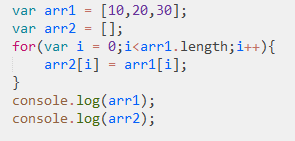
slice（strat，end）：返回从start到end的数的数组，下标到下标，例如数租[0,1,2,3]，返回[0,1,2]，end的值是3（可理解为从一开始数）

join（）：将数组变为字符串类型,括号里面写符号，

concat：将数组合并（用得不多）



遍历赋值



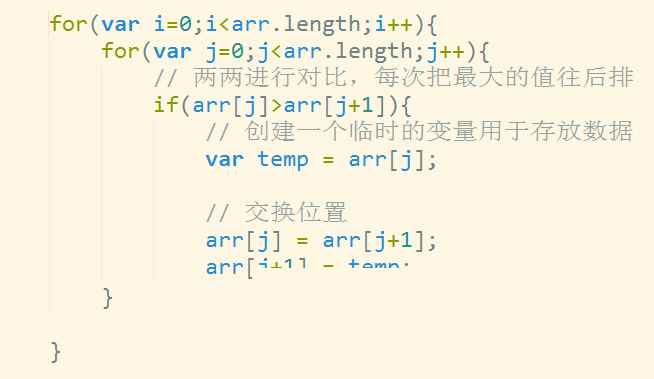
冒泡排序法：每次最大的值往后排

思路：将每个数字都放到数组里，然后将数组的每个数字取值，依次和后

面的数字将比较，如果后面的数字大，那顺序不变，相反那就将后面的数

字和前面的数字交换位置；

这里就需要一个交换赋值：temp;



优化：



增加了在内循环的length-1

选择排序法：顺序跟冒泡相反

# Object：对象

里面用：不用=

创建对象：

字面量：var obj = ｛｝

添加/获取属性值：(对象的属性读取格式)





1 obj[‘name’] = ‘老谢’

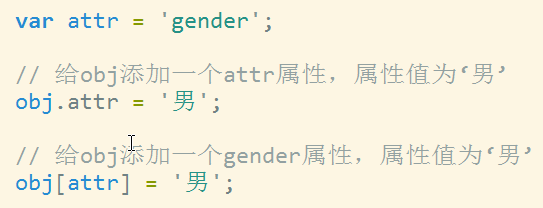
obj[‘age’] = 18

2 通过点属性

obj .name = ‘laoxie’

obj.age = 18

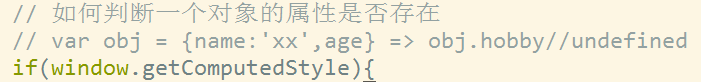
区别：



构造函数建立对象

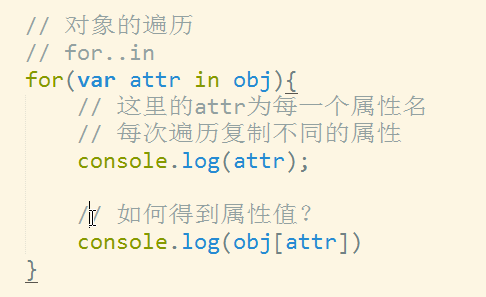
var obj = new Object（）

判断对象的属性是否存在

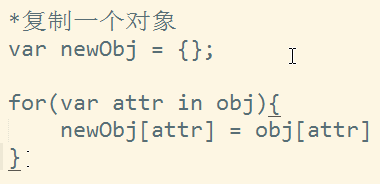


对象的遍历

For…In



复制对象：



# ECMASCRIPT5

## 数组新增方法：

判断是否为数组：Array.isArray（数组）

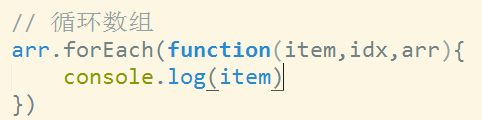
判断属性值是否存在：数组名.indexOf（要查找的内容），&数组名.lastIndexOf（要查找的内容）区别：一个从前往后，一个从后往前

数组名.indexOf（查找的内容，数值（从哪个位置开始找）），会返回元素在数组里面的位置的下标,返回-1表示不存在

### 迭代方法

forEach(fn)

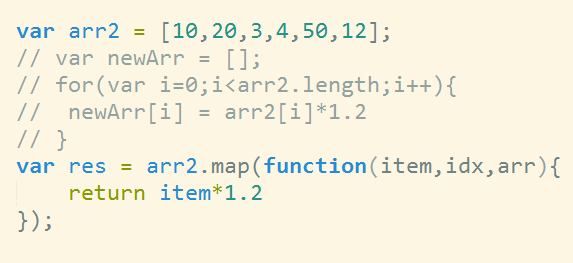
遍历方法，for循环没有太大差别，比for循环方便



map(fn)：能快速生成html结构

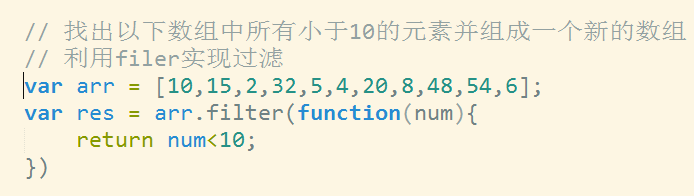
返回每次函数调用的结果组成的数组

返回取决于return了什么东西



filter(fn)

返回该函数会返回 true 的项组成的数组，利用这个方法可对数组元素进行过滤筛选



every(fn)

如果该函数对每一项都返回 true，则返回true

some(fn)

如果该函数对任何一项返回 true，则返回true

以上方法都对数组中的每一项运行给定函数fn,，函数中有三个形参分别为

- item：数组中的每一项,

- index：遍历过程中对应的索引值,

- array：对数组的引用

### 归并方法

这两个方法都会迭代数组中的所有项，然后生成一个最终返回值。

reduce(fn, initVal（初始值）)

reduceRight(fn, initVal（初始值）)

fn(prev,cur,index,array): fn是每一项调用的函数，函数接受4个参数分别是

prev：前一次返回值，

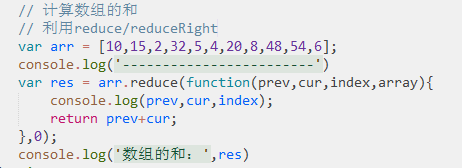
cur：当前值，

index：索引值，

array：当前数组，

函数需要返回一个值，这个值会在下一次迭代中作为prev的值

initVal: 迭代初始值（可选），如果缺省，prev的值为数组第一项



# 字符串的属性和方法

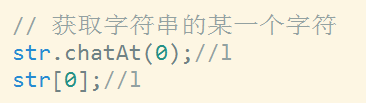
\*字符串可以看成是一个数组，可遍历,但是遍历字符串会把每个字都看成是一个下标，所以会将字符串变成数组后再遍历

属性

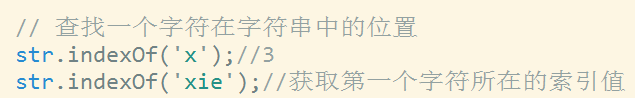
length: 表示字符串的长度，只读（只能读取）

## 字符串的获取方法

charAt(3) //获取下标为3的字符



## 字符串的查找方法



indexOf(“abc”) 从开头向后查找字符串第一次出现的位置,如果没找到返回-1，不支持正则

lastIndexOf(“abc”) 从尾部向前查找字符串第一次出现的位置，如果没找到返回-1，不支持正则

search(str|regExp) 查找字符串第一次出现的位置,支持正则

match(str|regExp) 匹配字符串，返回一个数组，支持正则

index:匹配字符所在的索引

input:表示整个字符串的引用

### replace(str|rgExp,’‘) 替换字符串 支持正则

这里的替换只能执行一次，不能够进行全局匹配，如果需要全局匹配，则应使用正则表达式



## 字符串的截取方法

substring(start[,end])：不包括end所在字符，end省略表示截取到最后

substr(start[,len])：支持负数，len为截取的数量

slice(start,end) ：截取start到end(不包括end)的字符串，支持负数

字符串大小写转换

toLowerCase()：转换成小写

toUpperCase()：转换成大写

## ECMAscript5新增

str[3]//通过下标获取

#### trim()：删除前后所有空格，返回新的字符串

#### charCodeAt(3) //获取下标为3的字符的ASCII(American Standard Code for \* Information Interchange) == > unicode编码一般使用循环

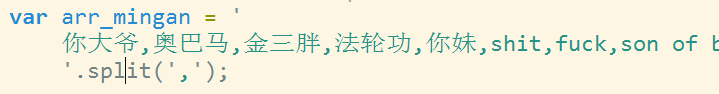
#### String.fromCharCode(94) //编码转换成字符

charCodeAt()/String.fromCharCode()是数组的方法，遍历数组使用

split(分割符)：根据分割字符，把字符串拆分成数组

将字符串拆分后，会按照拆分的符号把字符串拆成一个个item作为数组

🡺



# 正则表达式

创建

var reg = /,/gi /需要查找的内容/

构造函数方法创建：用变量创建表达式一定要用构造函数的方法创建

var reg = new regExp（）;



str表示需要查找的内容

g：表示匹配所有

i：表示不区分大小写

# Math对象方法

属性：Math.PI

round(3.6) //四舍五入取整

ceil(11.3) //12 向上取整

floor(11.8) //11 向下取整

random() //返回0-1之间的随机数（不包括1）

max(num1, num2) //返回较大的数

min(num1, num2) //返回较小的数

abs(num) //绝对值

pow(x,y) //x的y次方

sqrt(num) //开平方根

相关数学知识

三角函数复习

sin(radian)

cos(radian)

tan(radian)

角度与弧度的转换

弧度=角度\*Math.PI/180

三角函数一定要放弧度

勾股定理复习

在直角三角形中，斜边的平方等于直角边的平方和

曲线方程复习(一元二次方程)

# 时间日期



## 创建方法

使用之前都要先创建一个时间

1）创建当前时间的日期和时间

var d = new Date();//得到的是代码执行时的时间（本地时间）new Date（）已经有东西了

2）创建指定日期的时间和日期

var d = new Date("2015/08/22");

var d = new Date(56521211021);//参数为距1970-1-1零时的毫秒数

获取年月日

getFullYear()/setFullYear(2014)

getMonth()/setMonth(8)注意：获取月份是从0开始的，所以要+1

getDate()/setDate(25)



获取星期

getDay() 0-6:星期天-星期一

获取时分秒

getHours()/setHours()

getMinutes()/setMinutes()

getSeconds()/setSeconds()

getMilliseconds()/setMilliseconds();

UTC时间：在以上方法的get/set后加UTC, 如：

set/getUTCFullYear()

get/setUTCHours()

## 日期处理

getTime()：获取某个日期自1970年1月1日0时以来的毫秒数

setTime()：修改日期的毫秒数，对应的日期时间也会修改

toDateString(); 以特定的格式显示星期几、月、日和年

toTimeString(); 以特定的格式显示时、分、秒和时区

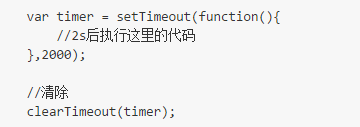
toLocaleDateString(); 以特定地区格式显示年、月、日

toLocaleTimeString(); 以特定地区格式显示时、分、秒

## 延迟与定时器

setTimeout(fn,200)：两百毫秒后执行fn这个函数（只执行一次）,返回一个id标识

clearTimeout(timeoutID)：清除指定id标识的延迟操作



setInterval(fn,30)：每隔30毫秒执行一次fn这个函数,返回一个id标识

clearInterval(intervalID)：清除指定id标识的定时器操作



\*延时器执行完需要清楚，时间需要通过三元变化作补0设置

\*定时器里面暂时应该可以不用循环

## ES5时间方法

Date.parse(“2015-08-24”)//返回指定日期距1970-1-1零时的毫秒数

PS：转换格式默认支持2015-08-24或2015/08/24

Date.now();//返回执行这行代码时距1970-1-1零时的毫秒数

# BOM 浏览器对象模型

全局对象会变成window对象的属性

通过var在全局作用域下声明的变量无法delete

在函数里面如果不使用var声明的话，变量会成为全局对象

## Window对象下的常用属性

innerWidth/innerHeight //表示浏览器窗口可视区域的尺寸

outerWidth/outerHeight //表示整个浏览器窗口的尺寸



## 滚动相关

scrollX/scrollY //获取浏览器窗口滚动条滚动过的距离

scrollTo(x,y) //设置浏览器滚动距离

scrollBy(xnum,ynum) //设置基于当前位置滚动的距离，可以为负数

常用方法(了解)

系统对话框

alert(msg)//弹出对话框

confirm(msg)//弹出警告框，返回布尔值

prompt(msg,default)//弹出输入框，返回消息或null

\*\*三个方法都是中断代码的执行

open(url,name,[options]) : 打开一个新窗口并返回新 window 对象

name:不命名会每次打开新窗口，命名的第一次打开新窗口,之后在这个窗口中加载

options为字符串：'width=400,height=400,top=200,left=200'

opener父窗口对象，通过open方法打开的窗口才有opener对象

close(): 关闭窗口

print(): 调出打印对话框

## 属性对象

### document(核心): 文档对象，让我们可以在js脚本中直接访问页面元素(DOM)

### history(重要): 历史对象,包含窗口的浏览历史，可以实现后退

属性：

length 返回浏览器历史列表中的 URL 数量。

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 列表中的某个具体页面，支持负数。

history.go(2)：向前两个页面；

history.go(-2)：后退两个页面

### location(重要):

location是BOM最有用的对象之一，保存着当前窗口中加载文档的相关信息，还提供一些导航功能，它是个很特别的对象，既是window的属性，也是document的属性

属性：

hash 设置或返回从井号 (#) 开始的 URL（锚）==>哈希值。

href 设置或返回完整的 URL。可用于跳转

search 设置或返回从问号 (?) 开始的 URL（查询部分）。

encodeURI()：转码//百分号表示转码

decodeURI()：解码

PS：修改以上属性(hash除外)都会刷新当前页面，并生成历史纪录

## 方法：

### reload() 重新加载当前文档，带参数true表示不使用缓存刷新页面。

### navigator（了解）:

导航对象, 包含所有有关访问者浏览器的信息，通常用于检测浏览器类型

appName 浏览器名称

appVersion 浏览器版本

platform 操作系统

userAgent 用户代理信息，通过该属性可以获取浏览器及操作系统信息

#### geolocation 获取地理位置信息

# window对象常用事件

onload //页面资源全部加载完成后触发这个事件

onbeforeunload //在页面关闭时触发这个事件

onscroll//滚动窗口滚动条时触发

document.documentElement:表示html元素

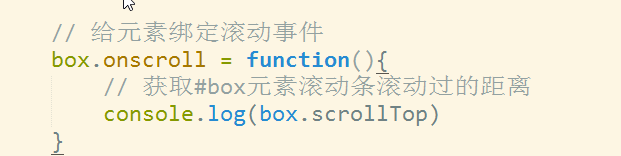
document.body: 表示body元素

document.documentElement.scrollTop/Left 表垂直/水平的滚动条，向下/右滚动的距离（FF、Opera和IE），可以设置

document.body.scrollTop（） 代表垂直的滚动条，向下滚动的距离（chrome和IE的quirks ,可以设置

mode兼容模式）

## 获取元素的滚动条

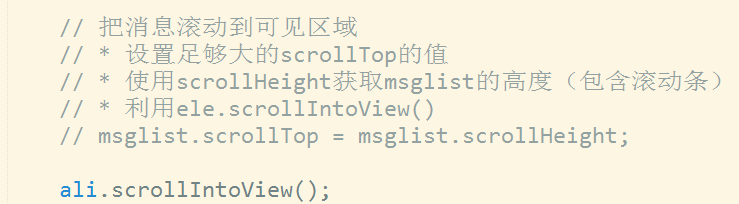


onresize //窗口大小改变时触发

scrollHight滚动条的高度

focus（），使用textarea后，使发送消息之后光标不会消失

### 把消息滚动到可视区域

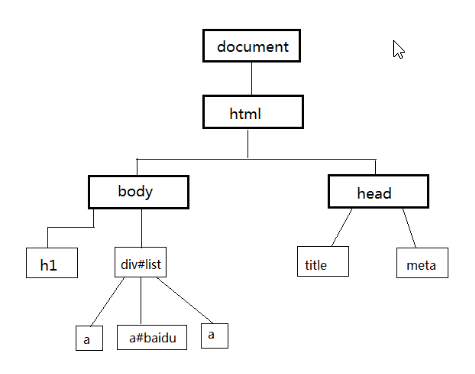


isNaN（需要查看的元素）返回值true/false检查是否数字

# DOM:节点必须插入节点，而不是元素，元素才能绑定事件，元素也可理解为是一个节点

Window

|



## 节点node：

元素节点🡸🡺1

属性节点（在元素节点当中）🡸🡺2

文本节点🡸🡺3

## 获取节点

通过id获取：document.getElementById(‘’)//速度最快

通过标签名获取：上级ID名/document/body.getElementsByTagName(‘’)【】

通过类名获取：getElementsByClassName(‘’) )【】

//通过标签名和类名获取，得到一个数组，并可在任意元素节点下调用

通过name属性获取：document.getElementsByName(‘’)【】

//通常用于表单

注意：如果确定元素存在，但是返回null或[]，一定是代码执行顺序的问题

## 节点属性

nodeType：用于表明节点的类型(1,2,3)与上面对应

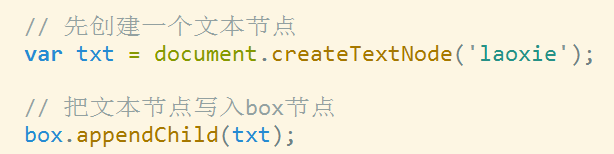
nodeName：节点名称

nodeValue：节点的值

## 创建节点

创建元素节点：document.creatElement(‘标签名如div’)

创建文本节点：document.creatTextNode



## 插入节点：

往里面后面加parent.appendChild（‘节点’）

往前面加parent.insertBefore(new,node)//new是要插入东西//node是要插入的地方



## 复制&替换

cloneNode(boolean) 复制节点，为true是深复制。

parent.replaceChild(new,old) 将old节点替换为new节点

## 删除：

parent.removeChild(ele) 删除（并返回）当前节点parent的指定子节点ele。

判断：

parent.hasChildNodes() 判断当前节点是否拥有子节点,返回布尔值

ele.hasAttribute(attr) 判断当前元素是否拥有attr属性，返回布尔值

## 利用节点关系获取其他节点

### 获取父级节点

ele.parentNode 得到ele元素的父节点

### 获取子节点

ele.childNodes 得到ele元素的全部子节点列表（类数组）

ele.firstChild 获得ele元素的第一个子节点（里面的）

ele.lastChild 获得ele元素的最后一个子节点

### 获取兄弟节点

ele.nextSibling 获得ele元素的下一个兄弟节点//有可能获取的不是元素节点

ele.previousSibling 得到ele元素的上一个兄弟节点

### 获取各种元素节点

parentElement 获取父级节点元素

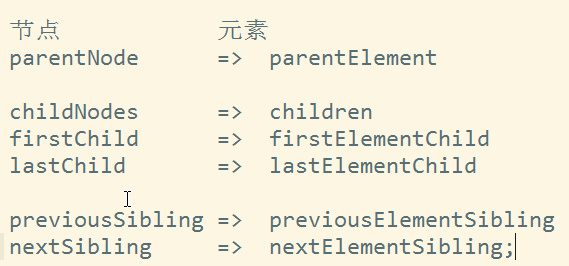
children 获取元素的全部子元素节点 🡺返回类数组

firstElementChild 获取第一个子节点元素

lastElementChild 获取最后一个子节点元素

previousElementSibling 获取前一个节点元素

nextElementSibling 获取下一个节点元素



## this：给谁绑定了事件，谁就是this

## 常用属性

innerText 设置或获取位于元素标签内的文本（不支持html字符串拼接）

outerHTML 设置或获取元素及其内容（包含html代码），把东西写入，但是不保留原来的东西，相当于替换了

outerText 设置(包括标签)或获取(不包括标签)元素的文本

## 元素方法

添加获取非标准属性，要在自定义属性前添加data-

ele.getAttribute(attr) //获取元素的属性值（自定义属性获取）



ele.setAttribute(attr,val); //设置元素的属性



ele.removeAttribute(attr) //删除属性attr

ele.hasAttribute(attr) //判断是否存在属性attr

## 盒模型相关

ele.offsetTop: 自身元素离<定位父级>元素顶部的距离。

ele.offsetLeft: 自身元素离<定位父级>元素左边的距离。

以上两个属性如果没定位的父级，则相对于根元素html的距离

ele.offsetWidth: 自身元素的宽度（border + padding + content）

ele.offsetHeight: 自身元素的高度（border + padding + content）

## 获取css样式（非内联样式）

得到当前元素计算后的所有样式

getComputedStyle(ele,pseudo).想要获取的样式/[想要获取的样式] （标准）

ele:要获取样式的元素

pseudo:伪元素样式字符(可选)，可获取伪元素样式



ele.currentStyle （IE8-）

判断兼容性



# table对象(了解)

table对象属性&方法

rows 返回包含表格中所有行的一个数组

tBodies 返回包含表格中所有 tbody 的一个数组

insertRow(index) 在表格中插入一个新行。

deleteRow(index) 从表格删除一行。

tr对象属性&方法

cells 返回包含表格中所有单元格的一个数组

rowIndex 返回该行在表中的位置

sectionRowIndex 返回在 tBody 、tHead 或 tFoot 中行的位置。

insertCell(index) 在一行中的指定位置插入一个空的列

deleteCell(index) 删除行中的指定的单元格

td/th对象属性&方法

cellIndex 返回单元格在表格行的单元格集合中的位置。

# 事件

格式：元素.on + 事件名

## 鼠标事件

onclick 当用户点击某个对象时调用的事件。

ondblclick 当用户双击某个对象时调用的事件。

onmousedown 鼠标按钮被按下。

onmouseup 鼠标按键被松开。

onmouseover 鼠标移到某元素之上。

onmouseout 鼠标从某元素移开。

onmousemove 鼠标被移动时触发。

mouseenter 在鼠标光标从元素外部移动到元素范围之内时触发。这个事件不冒泡

mouseleave 在位于元素上方的鼠标光标移动到元素范围之外时触发。这个事件不冒泡，

oncontextmenu 鼠标右键菜单展开时触发。

PS：click = mousedown + mouseup, dblclick = click\*2(短时间内两次单击);

执行顺序：mouseover=>mouseenter; mouseout => mouseleave

键盘事件

onkeydown 某个键盘按键被按下。

onkeyup 某个键盘按键被松开。

onkeypress 键盘<字符键>被按下,而且如果按住不放的话，会重复触发此事件。

UI事件

onload 页面元素（包括图片多媒体等）加载完成后

onbeforeunload 用户退出页面。

onresize 窗口或框架被重新调整大小。

表单事件

onselect 输入框文本被选中。

onblur 元素失去焦点时触发。

onfocus 元素获得焦点时触发。

onchange 元素内容被改变，且失去焦点时触发。

onreset 重置按钮被点击。

onsubmit 确认按钮被点击。

oninput 输入字符时触发

其他事件

# Event对象

事件处理函数元素.on事件 = function（e）｛｝

什么是event对象

事件执行过程中的状态，用来保存当前事件的信息对象

如何获取event对象

标准：事件处理函数的第一个参数

div.onclick = function(e){}

IE8-：window.event

获取event对象的兼容写法: e = e || window.event;

## Event对象详情

鼠标/键盘事件Event属性

### e.button 返回当事件被触发时，哪个鼠标按钮被点击。

W3C标准

0: 代表鼠标按下了左键

1: 代表按下了滚轮

2: 代表按下了右键

IE8-（IE8以下的浏览器）

1鼠标左键， 2鼠标右键， 3左右同时按， 4滚轮， 5左键加滚轮， 6右键加滚轮， 7三个同时



e.which/e.keyCode

对于keypress事件，该属性声明了被敲击的键生成的 Unicode 字符码(ascii码)

对于keydown和keyup事件，它指定了被敲击的键的虚拟键盘码。虚拟键盘码可能和使用的键盘的布局相关。

e.altKey 返回当事件被触发时，”ALT” 是否被按下。

e. ctrlKey 返回当事件被触发时，”CTRL” 键是否被按下。

e. shiftKey 返回当事件被触发时，“Shift” 键是否被按下。

### 光标位置信息

clientX /clientY 光标相对于浏览器可视区域的位置，也就是浏览器坐标。

screenX/screenY 光标指针相对于电脑屏幕的水平/垂直坐标。

pageX/pageY:鼠标相对于文档的位置（包括滚动条滚动的距离，clientX+window.scrollX，IE不支持）

offsetX,offsetY: 光标相对于事件源对象的相对偏移量。

事件源对象：触发事件的对象

## Event公共属性

e.type：被触发的事件的类型(一般用于判断)

e.target: 返回触发此事件的元素（事件的目标元素）。

兼容性写法var target = e.target || e.srcElement

e.currentTarget: 其事件处理程序当前正在处理事件的那个元素

IE事件Event属性（ie8-）

e.cancelBubble：如果事件句柄想阻止事件传播到包容对象，必须把该属性设为 true（同e.stopPropagation()）。

returnValue: 如果设置了该属性，它的值比事件句柄的返回值优先级高。把这个属性设置为fasle，可以取消发生事件的源元素的默认动作（同preventDefault()）。

srcElement: 对于生成事件的 Window 对象、Document 对象或 Element 对象的引用（同target）。

## 事件冒泡

e.stopPropagation（）;阻止事件冒泡

什么是事件冒泡：

在一个对象上触发某类事件（如onclick事件），那么click事件就会沿着DOM树向这个对象的父级传播，从里到外，直至它被处理程序处理，或者事件到达了最顶层（document/window）

事件源对象：触发事件的元素

标准：event.target

IE8-：event.srcElement

事件源对象在事件传播过程中不会改变

事件委托，利用冒泡的原理将事件给父级绑定，利用target和srcElment

阻止事件的传播

标准：event.stopPropagation();

IE8-：event.cancelBubble = true;

1）不是所有的事件都能冒泡。以下事件不冒泡：blur、focus、load、unload…。

2）冒泡到最顶层的目标不同。大部分浏览器到window对象，IE8-到document对象

阻止浏览器默认行为

链接跳转

表单提交

右键菜单…

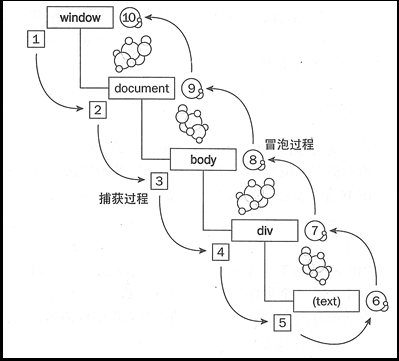
标准：event.preventDefault();

IE8-：event.returnValue = false;

注意：

页面事件绑定数量越多，越影响性能（速度越慢）

事件传播：事件捕获与事件冒泡



# 事件监听与捕获(反冒泡)

传统的事件绑定方式，只能在冒泡方式中执行函数的代码,同名的事件会被覆盖

## 事件监听器

多个同名事件不会被覆盖

可以在捕获阶段实行事件处理函数

格式：元素.addEventListener(‘’事件名,事件处理函数,是否捕获‘’)

target.addEventListener("click", fn, false);

可以绑定多个处理函数在一个对象上, 执行顺序按照绑定的顺序来

不同元素事件执行顺序跟html结构有关

相同元素事件执行顺序跟绑定先后有关

第三个参数是否使用捕获(反向冒泡)，默认false，为冒泡

IE8-的事件监听器：

格式：元素.attachEvent(on+事件名,事件处理函数)

target.attachEvent("onclick",fun);

可以绑定多个函数在一个对象上, 执行顺序按照绑定的反序

### 移除

标准：removeEventListener(‘click’,fn, true)传入的参数fn要跟添加时一样(同一个函数)，否则不能移除事件

IE8-：detachEvent(‘onclick’,fun)，传入的参数fun要跟添加时一样，否则不能移除事件

## 拖拽效果

拖拽的原理

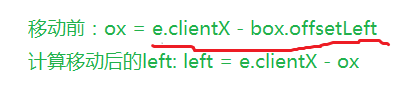
鼠标按下且移动鼠标时，改变当前元素的top,left值

拖拽的思路

给目标元素添加onmousedown事件

拖拽的前提是目标元素设置css定位

记录按下鼠标位置与元素左上角的偏移量offsetX，offsetY



当onmousedown发生以后，此刻给document添加onmousemove事件

意味着此刻鼠标在网页的移动都将改变目标元素的位置

在onmousemove事件中，设定目标元素的left和top：

公式

目标元素的left = 鼠标的clientX – offsetX

目标元素的top = 鼠标的clientY– offsetY

给目标元素添加onmouseup事件，清空document的onmousemove事件

意味着拖动过程中鼠标在网页的任意位置松开鼠标，都会放弃拖拽的效果

# Cookie

http明文通信，不加密

https比较安全，用于注册密码等私人信息的传输，都会请求完之后断开

socket会一直连着服务器，用于聊天室等

## cookie的概念

cookie 是客户端与服务器端进行通讯使用的一个能够在浏览器本地化存储的技术

PS：chrome不支持本地文件的cookie读写

## cookie的应用场景

七天免登陆

7天内访问网站无需输入密码

购物车信息

添加到购物车后，取到购物车页面，商品信息依然存在

商品浏览记录

用户每浏览一个商品就会保留商品的浏览记录

## cookie的获取和设置

//设置

document.cookie = 'name=laoxie';

//获取

var cookies = document.cookie;

一次只能写入一个cookie

获取时得到所有cookie，多个cookie之间用分号+空格（’; ‘）来隔开

## 写入cookie

格式：document.cookie = ‘cookie名 = cookie值’一次只能写入一个cookie

生成的cookie是一个字符串，可以利用字符串的方法转变为数组，通过；和空格进行拆分，然后将数组里面的每个值都拆成一个数组，再分别提取他们的属性和属性值，就可以提取到cookie



## cookie的组成

cookie value一定要是字符串

cookie由键值对形式的文本组成，完整格式如下：

document.cookie = name=value[;expires=date][;path=路径][;domain=域名]

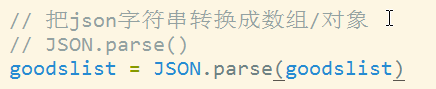
//[]表示该值是可选

name=value: 为你要保存的键值对(必选)

利用decodeURI解码中文字符

利用json保存多条信息





expires=date: 表示cookie的失效时间, 默认是浏览器关闭时失效(可选)

//设置7天内生效的cookie

var date = new Date();

date.setDate(date.getDate() +7);

document.cookie = "user=laoxie;expires=" + date;

path=路径: 访问路径, 默认为当前文件所在目录(可选) path=/表示设置cookie所在的目录为根目录

cookie只能在设置的路径及它的子目录下使用

domain=域名: 访问域名, 限制在该域名下访问(可选)

没有设置则默认为当前域名

## cookie的删除

利用设置过期时间达到删除的效果。

## cookie的封装

封装cookie的增删改查

## XSS攻击

跨域脚本攻击(Cross Site Scripting)

## JSON字符串的规则：

1 属性必须使用双引号

2 字符串必须使用双引

3 属性值不能使用函数

4 不能有多余的逗号

5不能存在注释

因为cookie里面的数组有对象，所以用JSON来变成数组再读取

# 正则表达式

文本替换和匹配功能，表单验证

[]表示一个字符，里面的内容是或的关系

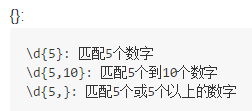
-表示一个范围，例如a-z a到z

^表示非

\d表示一个数字

\D表示非数字

｛num｝数量词



\b表示单词边界

\s表示空格

|表示或,只有一个的情况下用[]

^表示以开头

$表示以结束

+表示一个或一个以上{1,}

\*表示0个或0个以上{0,}

?表示0个或1个{0,1}

\w表示数字字母下划线

.表示换行符以外的字符,如果要在正则中使用.符号，要在前面加转义符\

（）表示分组

在正则外$1表示第一个分组

在正则外$n表示第n个分组

在正则内\n写在分组后面，表示对第n组的引用

可在分组后任一个位置

## 正则表达式的属性和方法

测试正则表达式用test方法,返回布尔值

格式：正则表达式.test(字符串//要测试的东西)

用<正则表达式>测试<字符串>是否匹配,返回true/false

测试正则表达式exec方法

/xx/.exec(字符串)

以上都只表示一个，要多个必须加入数量词

支持正则表达式的字符串方法

search

返回第一次匹配时所在的索引值,如果匹配不到则返回-1

match

默认匹配字符串，返回一个数组

0:所匹配的字符

index:匹配第一个字符所在的索引

input:对字符串的引用

全局匹配(g)，返回一个匹配所有字符串数组

如果匹配不到则返回null

replace

替换字符串

split

'a,b ,c , d, e'.split(/\s\*,\s\* /);

document.onreadystatechange = function(){}

if(document.readyState === interactive)

if(document.readyState === complete

# ES5

## DOM文档加载的步骤为

解析HTML结构。

加载外部脚本和样式表文件。

解析并执行脚本代码。

DOM树构建完成。//DOMContentLoaded

加载图片等外部文件。

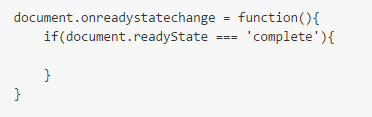
页面加载完毕。//window.onload

## document事件

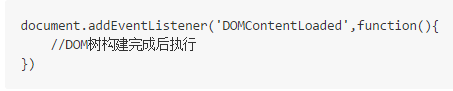
## readystatechange事件

interactive

complete



## DOMContentLoaded事件



PS:以上事件用以取代window.onload事件

## Es5的严格模式

在js头部加入‘’use strict‘’即可

## ES5获取页面元素方式

元素.querySelector（’css选择器’）

querySelectorAll（’css选择器’）返回一个类数组

## 函数扩展

bind（）



## 获取class列表属性

classList：js 的方法

length : class类名的个数

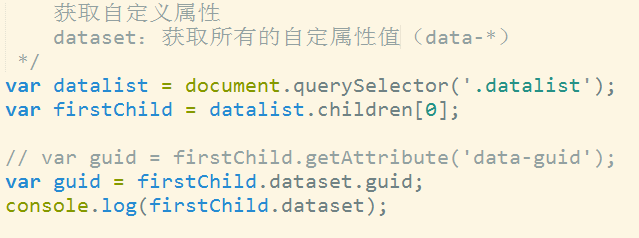
add(诶名) : 添加class方法

remove(类名) : 删除class方法

toggle(类名) : 切换class方法

contains(类名):是否含有某个类,返回布尔值

## dataset获取所有自定义属性值



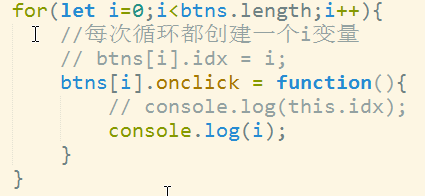
# ES6：可直接用花括号来规定作用域

## 变量声明

### Let

1 let声明不会提升

2 块级作用域



3 不允许同作用于内声明多个同名变量

### Const

### 后面的名字需要全大写



1 声明后无法修改值

2 块级作用域

3 const不允许想同作用域多次声明同一变量

### 解构

#### 数组

1）var [a,b,c] = [1,2,3] //a=1,b=2,c=3

2）var [a,[[b],c]] = [10,[[20],30]] //a=10,b=20,c=30

3）var [a,...b] = [1,2,3,4] //a=1,b=[2,3,4];...表示剩余参数

#### 对象



1）var {a,b}={a:’html’,b:’css’}

2）变量必须与对象属性名相同，否则为undefined

3）如果变量名与属性名不相同，则必须写成以下格式才能取到值

var {a:test} ={a:'html',b:'css'} //test=>html

##### 解构失败：

var [a]=[],[b]=1,[c]='jiegou',[d]=false //a,b,c,d都得到undefined

##### 指定默认值：

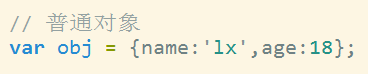
var [a=true]=[]

var {a=10} = {}

解构同样适用于let和const

PS:解构只能用于数组和对象，如果解构不成功，变量会返回undefined，但如果对undefined和null解构则会报错

解构成功会使用后面赋予的值，不成功则会返回默认值（有的话）或undefined



### 解构用途

#### 交换变量值

var [x,y] = [y,x];

#### 函数返回多个值

function example(){

return [1,2,3]

}

接收

var [x,y,z] = example();

#### 定义函数形参

函数的参数定义方式, 不用再考虑参数的顺序

function test({x,y,z}){}

//传参

test({x:10,y:20,z:30})

//参数可以设置默认值

fuction test({x=10,y=20,z}){}

### 字符串扩展

#### 字符串方法

includes

判断是否包含某个字符，返回布尔值

'html5'.includes('html');//true

startsWith/endsWith

是否以某一字符开头/结尾

let str='google';

str.startsWith('g'); //true

str.endsWith('le'); //true

repeat(n)

得到字符串重复n次后的结果，n可以为小数，但不能为负数

'laoxie'.repeat(2);//laoxielaoxie

字符串模板, 使用反引号`表示

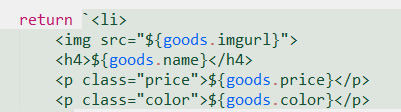
你可以通过一种更加美观、更加方便的方式向字符串中插入变量

格式：${变量|函数}，

`你好，我的名字叫${username},接下来是我的自我介绍：${introduce

()}`

模板字符串中所有的空格、新行、缩进，都会原样输出在生成的字符串中



### 数组扩展

#### 遍历

for...of

var arr = [10,12,18,30]

for (var value of arr) {

console.log(value);

}

这是最简洁、最直接的遍历数组元素的语法，只要预到一个数满足条件就会跳出循环

这个方法避开了for-in循环的所有缺陷

for…of跟for-in的区别很明显，就是直接取值，而不再取下标了

与forEach()不同的是，它可以正确响应break、continue和return语句

for-of循环不支持普通对象

for-of循环也可以遍历其它的集合

DOM节点

字符串

Set/Map集合 map有点类似对象 set有点类似数组

### 对象扩展

Object.assign(obj1,obj2,…objN);

合并对象

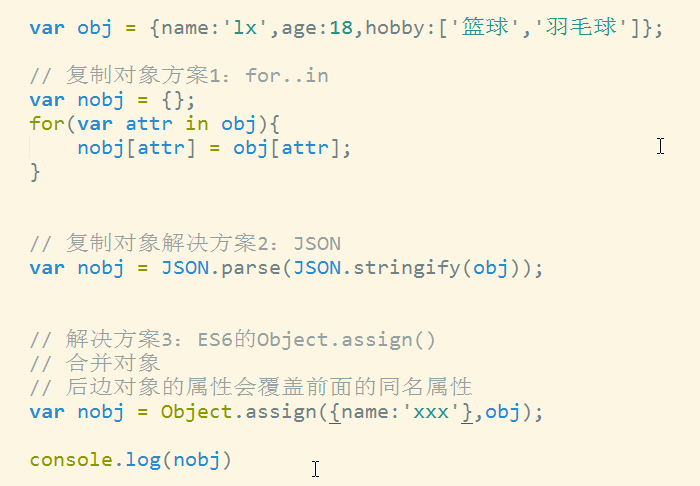
Object.assign({a:1},{b:2},{b:4,c:3}); //{a:1,b:4,c:3}

只支持浅拷贝（对于引用类型，只拷贝引用）,就是只拷贝快捷方式

忽略不可枚举属性

对象复制

Object.assign（）



## 箭头函数

当函数只有一个代码时，等于箭头后面是return值

无参时写法：var sum = （）=>100

一个参数：var sum = num =>100

多个参数：var sum = （num1，num2）=> num1+num2

多行代码

Var sum = （num1，num2） =>｛

Num1++；

Num2--;

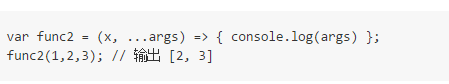


｝

如果需要返回对象，要加上小括号

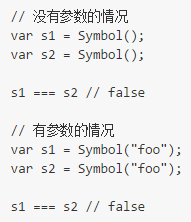


剩余参数



箭头函数没有自己的this值，里面的this继承自外面的作用域

## Symbol数据类型



注意：

1 Symbol函数前不能使用new命令，否则会报错

2 Symbol函数可以接受一个字符串作为参数，表示对Symbol实例的描述，主要是为了标识和区分，对调式非常有用

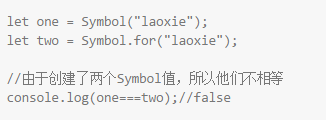
3 Symbol值不能与其他类型的值进行运算

### 常用方法

Symbol.for()

首先在全局中搜索已登记的Symbol值，如果有，就返回这个Symbol值，否则就新建并返回一个以该字符串为名称的Symbol值

直接用Symbol()方法创建的Symbol值不会被登记



Symbol.keyFor()

获取被登记的Symbol值

直接使用Symbo()创建的Symbol值的键不会被登记，所以也就获取不到

用途

给对象创建私有属性，而且不用担心属性名冲突

作为属性名的Symbol

var mySymbol = Symbol();

// 第一种写法

var a = {};

a[mySymbol] = 'Nani';

// 第二种写法（注意加方括号，否则回被当作普通属性）

var a = {

[mySymbol]: 'Nani'

};

// 第三种写法

var a = {};

Object.defineProperty(a, mySymbol, { value: 'Nani' });

// 以上写法都得到同样结果

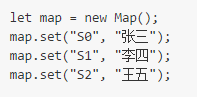
a[mySymbol] // "Nani"

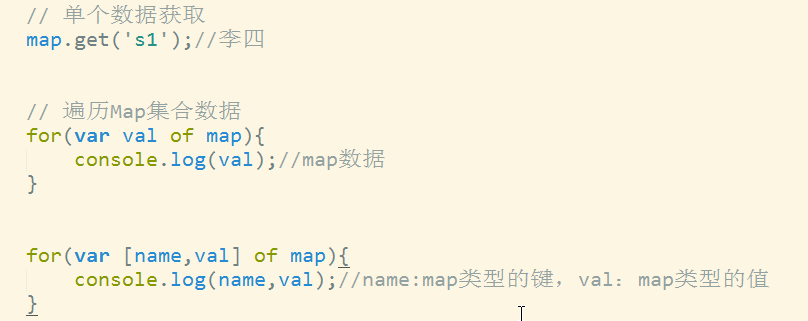
Symbol值作为对象属性名时，不能用点运算符。

不能用for…in、for…of遍历，也无法用Object.keys()、Object.getOwnPropertyNames()获取

## Map映射

var map = new Map（）；





第二种方法返回的是三个数组

键也叫名

方法

Map.prototype.keys() 获取所有键

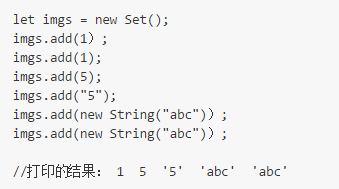
Map.prototype.values() 获取所有值

Map.prototype.entries() 获取所有键值对，返回类数组

## Set集合

var set = new Set（）；

Set集合，本质上就是对数组的一种包装



set集合是默认去重复的，但前提是两个添加的元素严格相等，所以5和”5”不相等，两个new出来的字符串不相等



# 动画

动画函数里面有四个参数，元素，属性，目标值，函数（用来链式运动）

不断改变对象的属性以产生动画的效果

## 盒模型

### offset

offsetTop: 当前元素离<定位父级>元素顶部的距离。

offsetLeft: 当前元素离<定位父级>元素左边的距离。

以上两个属性如果没定位的父级，则相对于根元素html的距离

offsetWidth: 当前元素的宽度（border + padding + content）

offsetHeight: 当前元素的高度（border + padding + content）

### client

clientTop: 当前元素上边框的宽度（border-top）

clientLeft: 当前元素左边框的宽度（border-left）

clientWidth: 当前元素宽度（padding + content，不包括border）

clientHeight: 当前元素高度（padding + content，不包括border）

### scroll

scrollTop: 当前元素<垂直滚动条>滚动过的距离

scrollLeft: 当前元素<水平滚动条>滚动过的距离

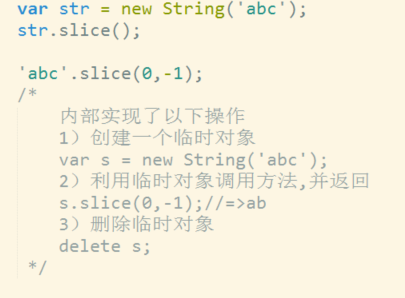
scrollWidth: 当前元素滚动宽度（不包括边框）

scrollHeight: 当前元素滚动高度（不包括边框）

PS: scrollWidth,scrollHeight浏览器间有兼容性问题

# 面向对象

Q1为什么字符串能够调用方法，过程



构造函数：构造函数，实例，原型对象

实例都指向原型对象

约定：首字母大写

function Person｛｝

与普通函数的唯一区别：调用方式

new Person（）

普通函数执行完没有return值的话会返回undefined；

构造函数即使没有return值，执行完会返回一个对象

实例可以直接调用原型对象的方法

构造函数执行过程

1 创建一个object对象

2 将构造函数中的this指向这个对象

3执行构造函数中的代码

4 返回创建的对象

通过new构造函数生成的对象称为实例

普通函数执行时，this指向调用函数的对象

构造函数执行时，this指向实例（内部创建的对象）

var obj = new Person｛｝，实例指obj

如何获取原型对象

1 通过构造函数获取

Person.prototype

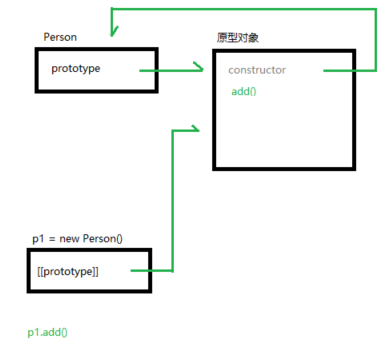
2 通过实例获取

var p1 = new Person（）；

私有：p1.\_\_proto\_\_

ES5标准:：Object.getPrototypeOf（p1）





原型对象的方法可以给所有实例对象共享

## this

函数中的this作为JS的关键字，有特殊的含义，代表了当前对象，而当前对象是谁，由函数执行时所处的环境来决定，谁调用函数，谁就是this  
实际开发中，利用构造函数添加属性，利用原型对象添加方法

匿名函数的自运行：

1（function（）｛｝）（）；

2（function（）｛｝）

3+/-/!function（）｛｝前面是一元运算符

可以传参

每个对象里面设置方法时，都设置一个初始化方法：init：function（）｛｝

1生成html结构

2绑定事件

## 判断

判断原型和实例的关系（返回布尔值）

constructor: 一般用于判断该实例是否由某一构造函数生成

实例.constructor == Student //true

instanceof: 检测某个对象是不是某一构造函数的实例

实例 instanceof Student //true

实例 instanceof Object //true

isPrototypeOf: 判断当前对象是否为实例的原型

原型对象.isPrototypeOf(实例) //true

## ES5对象扩展(了解)

### 属性特性

Object.create(prototype)

### 以指定的原型创建对象

Object.defineProperty(object, propertyname, descriptor)

### 对指定的对象的一个属性设置属性特性

#### 值属性

configurable

可配置性，控制着其描述的属性的修改，表示能否修改属性的特性

enumerable

可枚举性，表示能否通过for-in遍历得到属性

writable

可写性，表示能否修改属性的值

value

数据属性，表示属性的值。默认值为undefined

#### 访问器属性

configurable

enumerable

get

在读取属性时调用的函数。只指定get则表示属性为只读属性。默认值为undefined

set

在写入属性时调用的函数。只指定set则表示属性为只写属性。默认值为undefined

Object.defineProperties(object, descriptors)

对指定的对象的一组属性设置属性特性

Object.getOwnPropertyDescriptor(object, propertyname)

返回属性特性

Object.keys(object)

返回对象所有可枚举属性的名称

Object.getOwnPropertyNames(object)

返回所有属性的名称（哪怕说是不能枚举的属性）

# 闭包

1函数嵌套函数

2作用域（全局变量和局部变量）：变量的访问规则

3垃圾回收机制：js内部不断扫描内存，并清理无引用对象

结论

闭包是指有权访问另一函数作用域中的变量的函数

闭包可以访问函数内部的局部变量，并让长期驻内存。

由于闭包会携带包含它的作用域，因此会比其他函数占用更多的内存，过多使用会影响性能

# 原型链

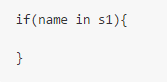
查找机制：

查找对象中的属性，（没有） ->原型对象中的属性，（没有）->原型对象里的原型对象中的属性，（没有）->抛出错误。

## 对象属性的遍历与判断

for…in：遍历对象中的所有属性, 无论该属性存在于实例中还是原型中

in：只要通过对象能够访问到属性就返回true, 无论该属性存在于实例中还是原型中



实例.hasOwnProperty(属性)：检测一个属性是存在于实例中(true)，还是存在于原型中(false)

检测一个属性是否存在于原型中



## 继承

继承的特点:

1. 子类拥有父类所有的属性和方法（代码复用）；

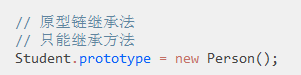
2. 子类可以扩展自己的属性和方法（更灵活）；

3. 子类可以重写父类的方法

## 原型链继承

prototype

原型链属性(方法)的查找过程：实例–>原型对象–>原型对象… –> Object.prototype



把原型对象执行父函数的实例，实现原型链继承，继承父元素的方法

PS:原型链继承法一般用于继承方法

## 借用构造函数

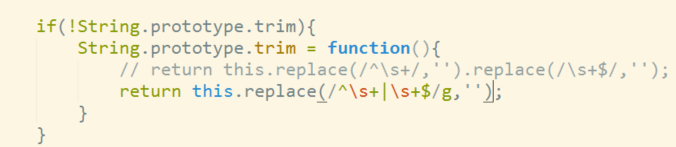
call：使用父类函数的call（子类对象,参数1，参数2，参数3......） 最常见

格式：构造函数.call(this,参数1,参数2,参数3......)

作用

借用：可用来借用不同的方法

扩展原生方法，例如扩展ie8以下trim（）；







把object的this指向arr，相当于arr调用了object的原型的tostring方法

apply：使用父类函数的apply（子类对象，[参数1，参数2，参数3]）

格式：父类构造函数.apply(子类当前对象,[参数1,参数2,参数3......])//第二个参数为数组

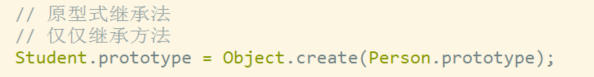
call与apply的唯一区别：传参方式不同

借用构造函数法只能继承构造函数里的属性/方法，不能继承原型对象里的方法

借用构造函数法一般用于继承属性



如果执行的时候传入了参数，而原函数没有参数，则可以传入argument



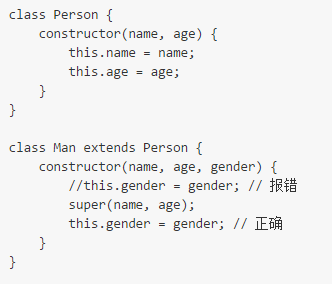
将父函数的原型对象赋值给student的原型对象

# ES6创建对象方法





## 继承

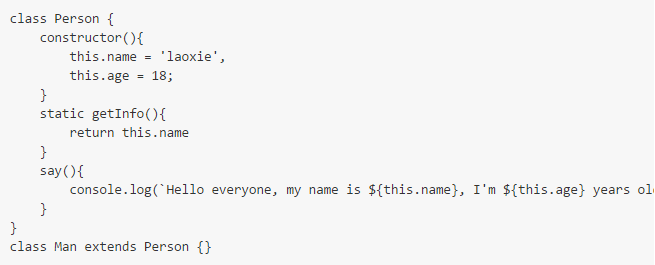


子类继承了父类，在子类构造函数中必须调用super方法。

子类的constructor方法没有调用super之前，不能使用this关键字，否则报错，而放在super方法之后就是正确的。

# 静态方法

如果在一个方法前，加上static关键字，这就称为“静态方法”



静态方法方法不会被实例继承，而是直接通过类来调用Person.getInfo()

父类的静态方法，可以被子类继承Man.getInfo()

# Jquery

Jquery方法只有jquery对象能使用，在使用前，先写$（function（）｛｝）来延迟代码执行

## jQuery实现原理

jQuery构造函数

jQuery实例对象

jQuery('#box');//得到jquery对象（实例）

## 延迟代码执行

jQuery(document).ready(fn)

此方法传入一个匿名函数;

页面DOM渲染完成时执行，用来替代window.onload;

简写方式:$(function(){});

编写jquery代码只需两步

1. 选择元素

2. 操作元素

## 选择器

选择页面中的元素，得到jQuery实例对象

ID选择器$(“#save”)

类选择器$(“.class”)

标签选择器$(“div”)

复合选择器 $(“div,span,p.myClass”)

属性选择器$(‘[id=box]’)

$(‘li[data-index]’):获取有data-index属性的所有元素

$(‘li[data-index!=10]’):data-index属性不等于10的元素,css目前未支持

$(‘li[data-index^=10]’):data-index属性以10开头的元素

$(‘li[data-index$=10]’):data-index属性以10结尾的元素

$(‘li[data-index\*=10]’):data-index属性包含10的元素

## 表单选择器$(‘:input’)

:submit //匹配所有提交按钮

:reset //匹配所有重置按钮

:button //匹配所有按钮

:radio //匹配所有单选按钮

:checkbox //匹配所有复选按钮

:text //匹配所有的单行文本框

:password //匹配所有密码框

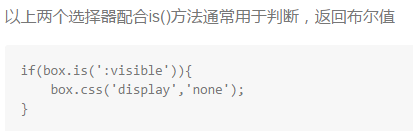
:selected //获取已选择的option元素

:checked //匹配所有被选中的元素(复选框、单选框等，select中的option)

可见性

:hidden //匹配所有不可见元素(display:none)，或者type为hidden的元素

:visible //匹配所有可见元素



## 常用操作

jquery对象与原生对象的转换

jquery转原生:

get(0)/[0]获取集合中第一个DOM节点

get()不传参得到集合中所有的dom节点

原生转jquery: $(dom);

判断是否为jquery对象

var box = $('#box');

if(box.jquery){

}

判断一个jquery对象是否存在(是否能获取到元素)

length

转成原生对象再判断

## 筛选

### 基本筛选

:odd/:even,:gt(n)/:lt(n) 筛选范围（索引支持负数）

:contains(奥巴马) 筛选出包含“奥巴马”这三个字的元素

### 筛选方法

first()/last(): 获取集合中第一个/最后一个元素

eq(index|-index): 获取第N个元素,n支持负数（表示从后面查找）

filter(expr|obj|ele|fn): 筛选出与指定表达式匹配的元素集合。这个方法用于缩小匹配\* 的范围。用逗号分隔多个表达式

map(fn): 将一组元素转换成其他数组（不论是否是元素数组）

slice(start,[end]): 选取一个从start到end(不包括end)匹配的子集

has(expr|ele): 保留包含特定后代的元素，去掉那些不含有指定后代的元素。

not(expr|ele|fn): 删除与指定表达式匹配的元素

## 查找

利用当前元素去查找其他元素

### 查找子元素

find(expr|obj|ele): 查找后代元素

children([expr]): 取得匹配元素的所有子元素。(原生js:children)

### 查找父级元素

parent([expr]): 获取父元素

parents([expr]): 取得所有父级元素

closest(expr|obj|ele): 从元素本身开始，逐级向上级元素匹配，并返回最先匹配的元素

offsetParent(): 返回第一个有定位属性(absolute,relative,fixed)\* 的父元素,如果没有定位父级，则返回html元素

### 查找兄弟元素

next([expr]): 返回下一个同辈元素 ==> nextElementSibling

prev([expr]): 获取前一个同辈元素 ==> previousElementSibling

nextAll([expr]): 获取当前元素之后所有的同辈元素

prevAll([expr]) 获取当前元素之前所有的同辈元素

siblings([expr]) 获取当前元素的所有兄弟元素（除自身以外的所有兄弟元素 = \* prevAll + nextAll）

# jQuery动画

## 基本动画效果

### 显示隐藏：show()/hide()

hide(duration)通过改变元素的高度、宽度、和不透明度，直到这三个属性值到0

show(duration)通过改变元素的高度、宽度、和不透明度，直至内容完全可见

### 滑动（通过改变高度）

slideDown([speed,callback])：

显示元素

不断改变高度，直到样式内设定的值

slideUp([speed,callback])：

不断改变高度，直到0

隐藏元素

slideToggle([speed,callback])

当元素隐藏时调用slideDown()，当元素显示时调用slideUp()

### 淡入淡出（通过改变不透明度）

fadeIn:

1)显示元素

2)不断改变透明度直到1

fadeOut:

1)不断改变透明度直到0

2)隐藏元素

fadeToggle([speed,callback])

fadeTo([[speed],opacity,[fn]]) 不断改变透明度opacity，直到设定的值，并在动画完成后可选地触发一个回调函数。

PS：jQuery动画由三种预设速度slow,normal,fast（600，400，200）

### 自定义动画

animate (params,[speed],[fn])

:animated

获取正在执行动画的元素，一般与is()方法配合使用，用于判断元素是否处于动画状态

### 动画队列

一个元素上的动画：

当animate中存在多个属性时，动画同时发生

当同一个元素链式调用animate时，动画是按顺序发生(队列)

多个元素上的动画：

默认情况下，动画同时发生

回调函数内的动画等到当前动画执行完后才接着执行

stop([clearQueue],[jumpToEnd])

不加参数：停止当前元素所有《正在运行》的动画。

clearQueue:值为true时，清除队列

jumpToEnd:值为true时，跳到当前动画的最后一帧

delay(duration)

设置一个延时来推迟执行队列中之后的动画。

duration:延迟的时间

# DOM节点操作

## 增删改

创建jquery对象

$('<div/>');

$('<div>生成一个div</div>');

元素添加

## 内部添加（添加为子元素）

append(content|obj|ele|fn): 在元素内部最后面追加内容（后置）

prepend: 向元素内部增加内容（前置）

appendTo,prependTo

## 外部插入内容（添加为兄弟元素）

after: 在元素后面插入内容

before: 在元素前面插入内容

insertAfter,insertBefore

如果页面上已经存在了要添加的元素，append,prepend,after,before会把元素移动到页面相应的位置

## 元素删除

remove(); 删除元素, 虽然元素从文档中删除了，但js内部依然保留对它引用

empty(); 清空内容

## 元素复制

clone([Event[,deepEvent]])

Event：（true 或 false）是否复制元素的行为，默认为false

deepEvent: （true 或 false）是否复制子元素的行为，默认为Even的值

# 盒模型属性

.offset():获取匹配元素相对于根元素的偏移量

.offset().left/.offset().top

返回一个对象，包含当前元素的top,left值

position():获取匹配元素相对(有定位属性)父元素的偏移量，如果没有定位父级，则相对于根元素(html)，返回一个对象，包含当前元素的top,left值。

width(v) = width; //取值/赋值,当传入v时，相当于css(‘width’,v);

innerWidth() = width + padding; <==> clientWidth

outerWidth() = width + padding + border; <==> offsetWidth

outerWidth(true) = width + padding + border + margin;

# 事件

常用事件方法

## 鼠标事件

click([[data],fn]) //点击时触发 click = mousedown + mouseup

dblclick([[data],fn]) //双击事件 dblclick = 2\*click

mousedown([[data],fn])

mouseup([[data],fn])

mousemove([[data],fn])

mouseout([[data],fn])

mouseover([[data],fn])

mouseenter([[data],fn]) //事件不会冒泡

mouseleave([[data],fn]) ) //事件不会冒泡

## 键盘事件

keydown([[data],fn]) //键盘按下时触发

keypress([[data],fn])//字符按键

keyup([[data],fn]) //键盘弹起时触发

表单事件

blur([[data],fn]) //失去焦点时触发

focus([[data],fn]) //获得焦点

change([[data],fn]) //值改变并失去焦点时触发

submit([[data],fn])

## 其他事件

resize([[data],fn]) //元素大小改变时触发

scroll([[data],fn]) //滚动时触发

## jquery事件绑定与移除

on(type,[selector],fn)

selector: 把本来绑定给selector的事件委托给它的父级

事件命名空间, 自定义事件（对事件加以细分）

格式：事件类型.自定名字

一次性绑定多个事件，事件之间以空格隔开

支持自定义事件的绑定

$(ele).on('laowang',function(){})

触发自定义事件：$(ele).trigger(‘laowang’);

triggerHandle（），但不会执行浏览器默认动作，也不会产生事件冒泡。

off: 清除绑定事件

off(‘click’);//清除当前元素的点击事件

off();//清除当前元素所有事件

off(‘click mouseover’) 一次性清除多个事件，事件之间以空格隔开

off(‘click.output’) 清除命名空间事件

在事件后面添加.名字，就能为事件命名，叫命名空间事件，一般在需要移除的时候才需要添加

## 其他事件方法

hover(over[,out])

over:鼠标移入时执行

out:鼠标移出时执行

hover方法内部使用mouseenter + mouseleave来实现效果

trigger(type): 手动触发事件（即使事件没有发生，也能执行事件处理函数）

triggerHandler(type): 这个方法会触发指定的事件类型上所有绑定的处理函数。但不会执行浏览器默认行为，也不会产生事件冒泡

阻止浏览器默认行为

event.preventDefault();

阻止冒泡

event.stopPropagation();

两者一起阻止：

return false;

# jQuery的ajax方法

$.ajax(settings)

type:请求类型，默认GET

url:数据请求地址（API地址）

data:发送到服务器的数据对象，格式：{Key:value}。

success:请求成功时回调函数。

dataType:设定返回数据的格式，json, jsonp, text(默认), html, xml, script

async：是否为异步请求，默认true

$.get(url,[data],[fn],[dataType]) // type:’get’

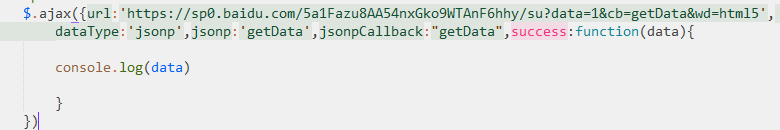
$.post(url,[data],[fn],[dataType]) // type:’post’

$.getJSON(url,[data],[fn]) // type:’get’, dataType:’json’

$.getScript(url,[callback]) // type:’get’, dataType:’script’

load(url,[data],[callback]) 载入远程 HTML 文件代码并插入页面中。

Jsonp方法：jsonpCallback，回调函数名称



# 插件

了解插件

什么是插件

插件(Plugin)也称为jQuery的扩展。以jQuery核心代码为基础编写的符合一定规范的应用程序。通过js文件的方式引用。

插件分类

UI类、表单及验证类、输入类、特效类、Ajax类、滑动类、图形图像类、导航类、综合工具类、动画类等等

自定义插件的规范（降低各种插件之间的冲突，减少错误机率）

## 编写规范

命名规范：jquery.插件名.js

插件中的this应该指向jQuery实例

使用this.each()迭代元素,为了实现多个调用

插件内部必须返回jQuery实例(this)，便于链式调用

所有的对象法附加在jQuery.fn对象(jQuery.prototype)上面，所有全局函数附加在jquery上

所有的方法或插件必须用分号结尾，避免出问题,为了保证插件的安全性，也可以在插件开始的地方加分号

避免插件内部的$冲突，请传递jQuery($并不是总等于jQuery，另外其他js框架也可能使用$)

## 常用插件：

jqueryUI //官方插件，功能多且全面，官网：www.jqueryui.com

jquery.validation //表单验证插件,官网：https://jqueryvalidation.org/

jquery.easyUI //是一组基于jQuery的UI插件集合,Demo：http://www.jeasyui.net/

bootstrap //是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目，官网：http://v3.bootcss.com/

highcharts

<https://www.hcharts.cn>

## 使用插件

引入插件的步骤

引入jquery.js文件，必须在所有插件之前引入

引入插件

引入插件相关文件，比如样式、语言包等

## 编写jquery插件

插件形式分为3类：

封装对象方法插件，扩展原型对象方法

封装全局函数插件，如$.each,$.map,$.makeArray…

选择器插件(类似于.find())

自定义插件的规范（降低各种插件之间的冲突，减少错误机率）

命名规范：jquery.插件名.js

插件中的this应该指向jQuery实例

使用this.each()迭代元素,为了实现多个调用

插件内部必须返回jQuery实例(this)，便于链式调用

所有的对象法附加在jQuery.fn对象(jQuery.prototype)上面，所有全局函数附加在jquery上

所有的方法或插件必须用分号结尾，避免出问题,为了保证插件的安全性，也可以在插件开始的地方加分号

避免插件内部的$冲突，请传递jQuery($并不是总等于jQuery，另外其他js框架也可能使用$)

最终格式：;(function($){$.fn.插件名=function(){}})(jQuery);

编写插件常用方法

$.extend([d],target,obj1,obj2,…,[objN]) //扩展对象或jQuery对象本身。用来扩展jQuery全局函数

用obj1-objN对象来扩展target对象

只有一个参数时，用来扩展jQuery全局函数：$.extend({‘hobby’:’撸串’})

$.ajax,$.map,$.each

d:是否深拷贝,布尔类型（true,false）

$.fn.extend() //扩展 jQuery元素集合提供新的方法（扩展jQuery原型对象，通常用来制作插件）。

# 常用jQuery实例方法

写在jQuery原型对象中的方法，通过jQuery实例调用

css(attr[,val]): 获取/改变元素style属性（内联样式）

取值：css(attr),css([‘color’,’text-align’]) <==> getComputedStyle[attr]

赋值：css(attr,val),css({attr:val});

val(v): 获取/设置匹配表单元素的值（等同于原生js中的value属性）

## 取值：

input.val()

## 赋值：

input.val(v)

v:字符串

v:数组（一般用于单选/复选框的勾选）

v:函数function(idx,val){ return 值}//函数内部一定要返回值

html(): （等同于原生js中的innerHTML）

取值div.html()：取得第一个匹配元素的html内容

赋值div.html(‘:’)：设置匹配元素的内容

text(): 取得所有匹配元素的文本内容。

addClass()/removeClass(): 添加/删除类,支持多个类同时添加或删除

toggleClass(): 如果存在（不存在）就删除（添加）类。

hasClass(‘con’): 判断当前元素是否包含con这个类，返回布尔值（不支持多个类进行判断）

eq(n) 获取第N个jquery对象（元素）,n支持负数（表示从后面查找）

index():获取当前元素在同辈元素中的索引值

$(this).index();

## 显示/隐藏

show()：显示元素

hide()：隐藏元素

带参数：同时改变width,height,opacity的动画

is(expr|obj|ele|fn) 根据选择器、DOM元素或 jQuery 对象来检测匹配元素集合，其中如果有一个元素符合这个给定的选择器表达式就返回true。如果没有元素符合，或者表达式无效，都返回false。

attr(name[,val]) 设置/获取元素的属性值（用于一般属性）

prop(attr[,val]) 获取/设置布尔类型属性（用于布尔类型属性）

获取：获取在匹配的元素集中的第一个元素的属性值。

赋值：给集合中所有元素属性赋值

val为函数

$(':checkbox').prop('checked',function(idx,oldVal){

return !oldVal;

}

each(function(idx,ele){}) //用于遍历jquery对象

return true;// 跳过当前循环，进入下一个循环（等效原生js中得continue）

return false;// 退出整个each循环（等效原生js中得break）

jquery大部分方法的共性：

无参数时为取值，带参数时为赋值

取值：取得第一个匹配元素的值

赋值：设置所有匹配元素的值

隐式迭代（隐式遍历）：看不见的遍历，大部分的jquery方法都支持隐式迭代

## 常用jQuery全局方法

$.each(arr|obj,callback)：通用遍历方法，用于遍历对象和数组。

callback(idx,item) =>item需要用$，交给jq处理

$.map(arr|obj,callback)：根据现有数组生成一个新的数组，新数组的元素为callback内return的值

callback(item,idx)

$.type(n)：检测参数n的数据类型

$.makeArray(obj) //将类数组对象转换为数组。

$.parseJSON(json) //接受一个JSON字符串，返回解析后的对象。类似原生js中的JSON.parse

$.inArray(value,array,[fromIndex]) //确定value在数组array中的位置，从0开始计数(如果没有找到则返回 -1 )，一般用于判断数组中是否包含某一字符。

serialize()/serializeArray() : 只能在form表单中使用，并且表单元素必须有name属性

## 常用插件

jqueryUI //官方插件，功能多且全面，官网：**www.jqueryui.com**

jquery.validation //表单验证插件,官网：**https://jqueryvalidation.org/**

jquery.easyUI //是一组基于jQuery的UI插件集合,Demo：**http://www.jeasyui.net/**

bootstrap //是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目，官网：**http://v3.bootcss.com/**

highcharts

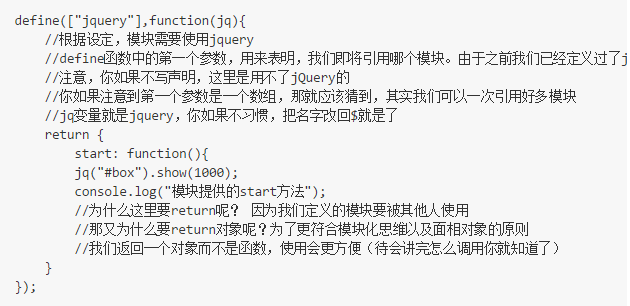
**https://www.hcharts.cn**

# 模块化开发

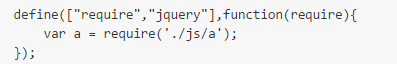
写好js文件后，先统一载入home.js里面，在引用到页面以页面的代码

RequireJS

定义模块



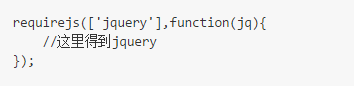
需要依赖的定义方式



引入



加载模块



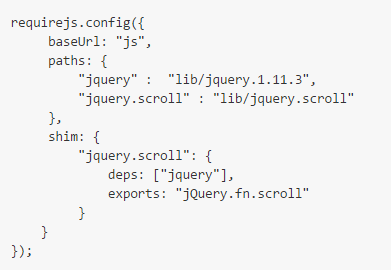
如果define了一个模块，require的回调函数的参数就是define后的模块的内容

需要配置config.js



shim配置(了解)

有一些插件，本身return出任何对象或函数，只是对某个框架的扩展，例如 jquery.scroll.js 该怎么实现模块化加载，答案是shim配置



# 常见错误搜集：



因为是undefined，没有forearch的方法报错



一种是tolowerCase写错了，另一种是前面的东西不是字符串，这里指【‘aa’】

当没报错又没效果，就要在控制台打印看看效果

Unexpected of JSON input 原因JSON.Parse中的字符串不符合规则