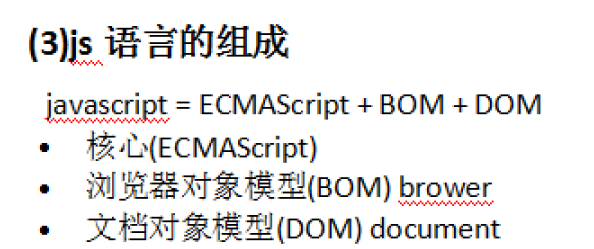
**JavaScript 基础**

**一**

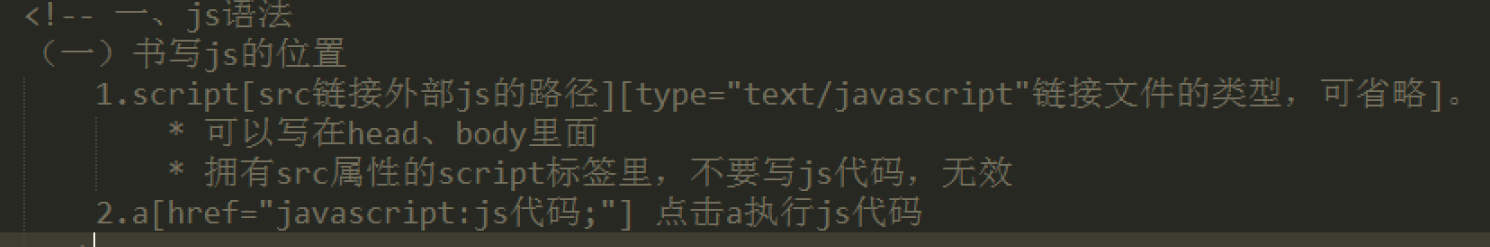


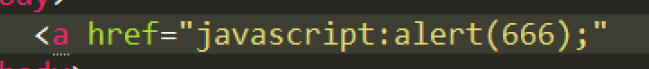
1.书写位置：

有三种：

1. Script 的src 链接外部js文件
2. 写在script标签中 没有src属性
3. 写在<a href=”jsvascript:js代码” ></a> 点击a 链接执行的js代码

例如：





2.JS的的注释：（单行和多行）

//单行注释

/\*\*/ 多行注释 【不能嵌套使用】

3.声明变量及赋值（重点）

1. 如何声明变量（容器）：通过关键字var
2. 对变量赋值
3. 如何同时声明变量及赋值
4. 同时声明多个变量



**补充：命名规则：**变量的命名规则：数字、字母、下划线、（$） [不能以数字开头、不能使用关键字或保留字，严格区分大小写] 【命名建议： 见名知意、驼峰命名规则】

**数据类型：（只有三种：number 、 boolean 、 String）**

1. 数字类型 number var a=10;
2. 字符串类型 String var a=”qzz”;
3. 布尔类型 boolean true/false； var tianqi=true

转义字符：\

输出方法：

1. alert（） 弹窗输出
2. Document.write(); 在页面中写出；往body里面输出内容
3. Console.log() 在控制台输出，多用于调试代码

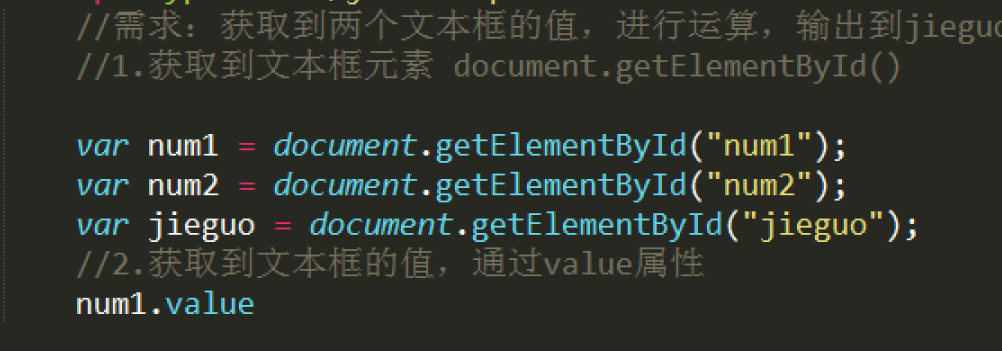




如何进行字符串和变量的拼接（字符串的拼接）？？？

1. 字符串拼接：当加号两侧只要有一侧为字符串，那么代表字符串拼接 【引号引号，加号加号，引号里面加号，加号里面加内】

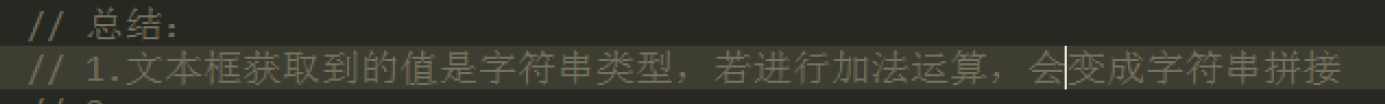
获取文本框的内容的值、通过value

【】

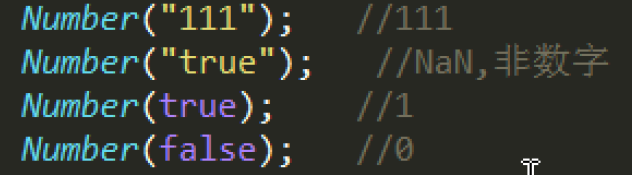
day\_01案例：



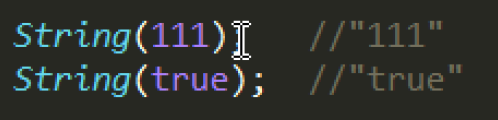
数据类型的转换：



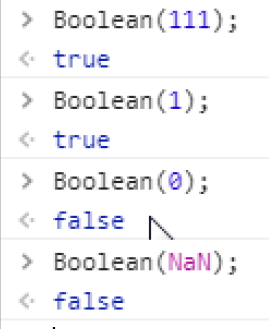
1. 直接转换
   1. Number（）转换为数字 【结果：数字、NaN】

 NaN代表纯数字

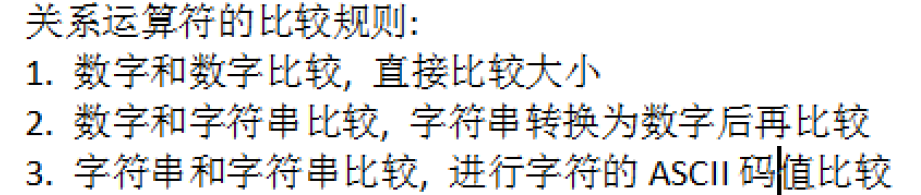
* 1. String（） 转换成字符串



* 1. Boolean（）转换成布尔值【除了 0、空字符串、NaN、null、undefined 之外都是true】

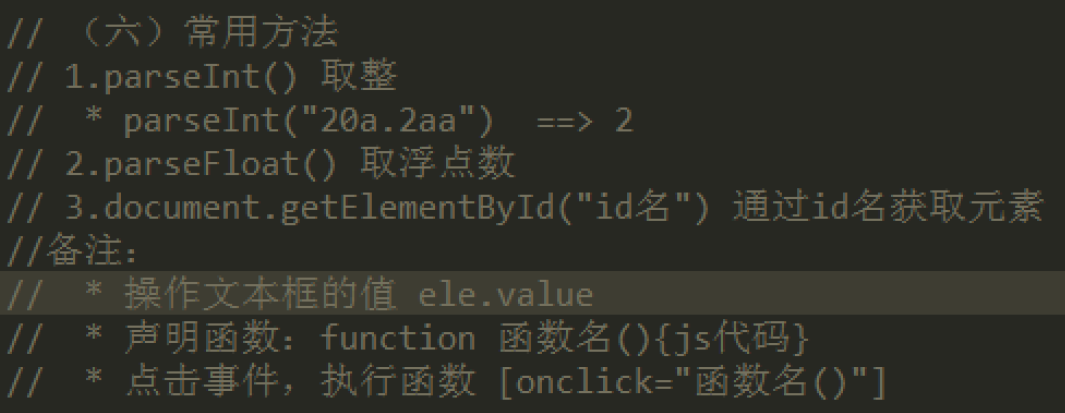


1. 隐式转换
   1. 加法不可以进行隐式转换
   2. 当隐式转换后，运算仍无法进行，会得到NaN
2. 字符串的比较：



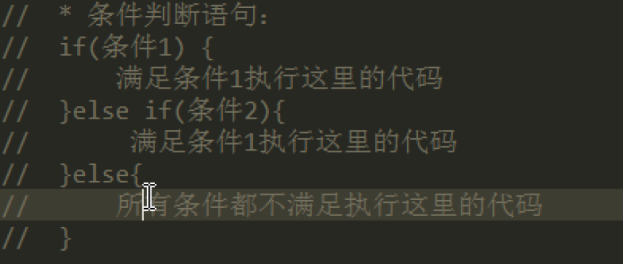
NaN： 非数值。NaN不等于NaN 【isNaN():判断是不是数字】

常用方法：



number.toFiexed(n)保留n位小数

Math.random() 生成[0,1)的随机数，包含0，不包含1 【parseInt(Math.random()\*101)】:生成0~100的随机整；

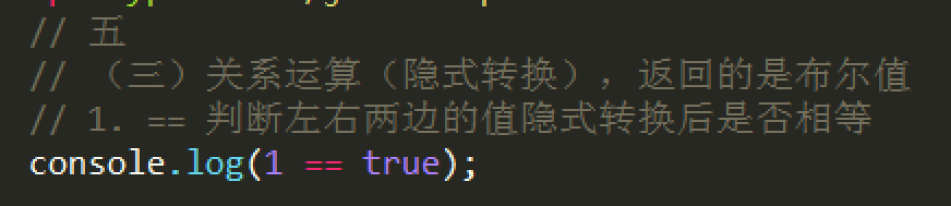


错误：在元素结构产生前，先获取元素会报错，因为代码执行顺序是从上到下的

关系运算： == 和 != 和 === 【判断当两边的值不仅内容，数据类型完全相等时才

为true 】 !==【判断当两边的值不仅内容，数据类型不完全相等时才

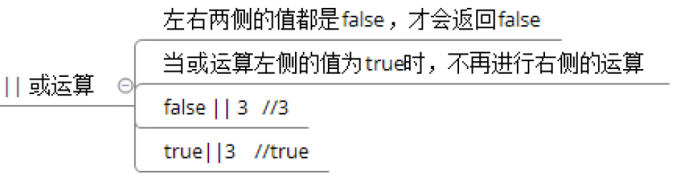
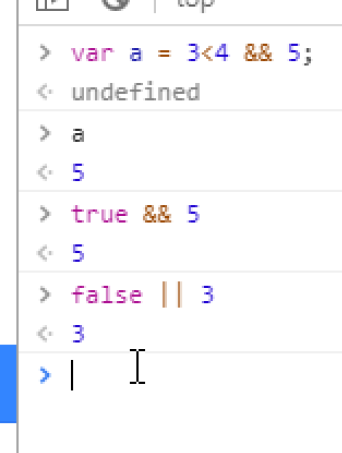
为true】



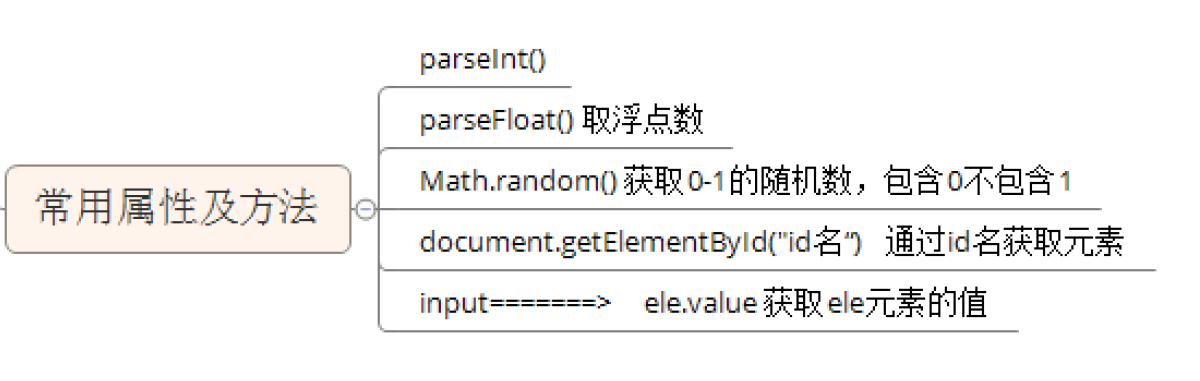
隐式转换后也相等。

逻辑运算：

1. && 逻辑与【运算符的两侧都为true的时候，最终返回true，短路原则：左侧返回false右侧将不执行】
2. || 逻辑或 【】



复习：



Ele.value ==> 只有表单元素有的属性

进制转换：

1. 二进制：0b开头
2. 八进制：0o开头
3. 十进制：
4. 十六进制：ox开头

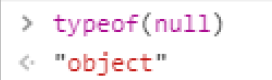
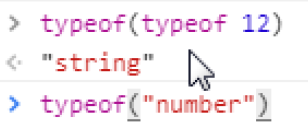
如何进行进制的转换：

1. 十进制转换 【number=100; number.toString(16);转成16进制；toString(8);转成8进制;number是要转换的值】 
2. 多进制转换十进制：

特殊数据类型：

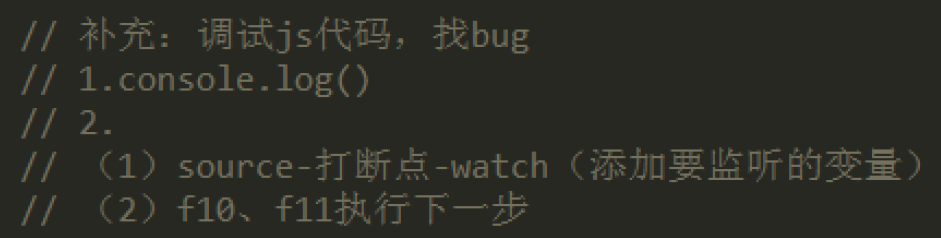
1. null 空对象 数据类型是NULL
2. undefined 数据类型是Undefined； 代表变量声明了，未被赋值；【区分报错： not defined 代表变量未声明】

数据类型的判断：

**Typeof() 方法**  ex: 【页面中获取的元素是一个对象】  typeof(typeof 12) ===>”string”

调试js代码：找bug

1. console.log()
2. 工具台：



【顺序执行、选择分支语句、循环语句】

**判断语句：if switch**

If语句：

1. 单分支if(){}
2. 双分支if(){}else{}
3. 多分支if(){} else if(){}...... else if(){} else{}

Switch 语句 【只有switch才有break 语句】

比较值的时候用的是全等于“===” 只有值的内容和数据类型完全相等的时候

Switch(变量){

Case 值1： ... break;

...

Default: 所有条件都不满足的时候，才执行这里的代码；（break）

}

Switch(true){

Case 关系表达式： ... break；

}

**循环语句：**



1. 三大要素：变量初始化、条件、变量更新；

注意：编写条件的时，要避免出现死循环；

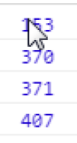
1. While语句：
   1. 变量初始化
   2. While(条件){

条件满足时，执行这里的js的代码；

变量更新；

}

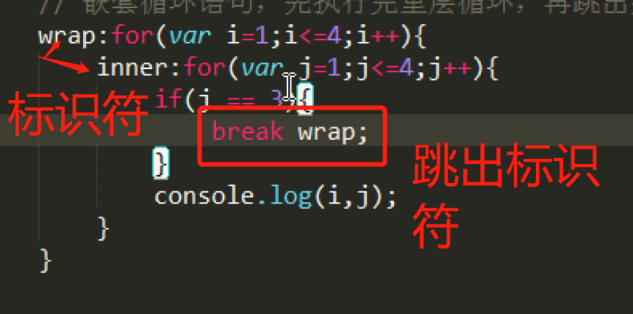
【while(true){ }】

水仙花数：153 = 1\*1\*1+5\*5\*5+3\*3\*3 

**Break 和continue 区别：**

Break：跳出循环；后面的代码将不在执行

Continue:跳出本次循环，进行下一次循环

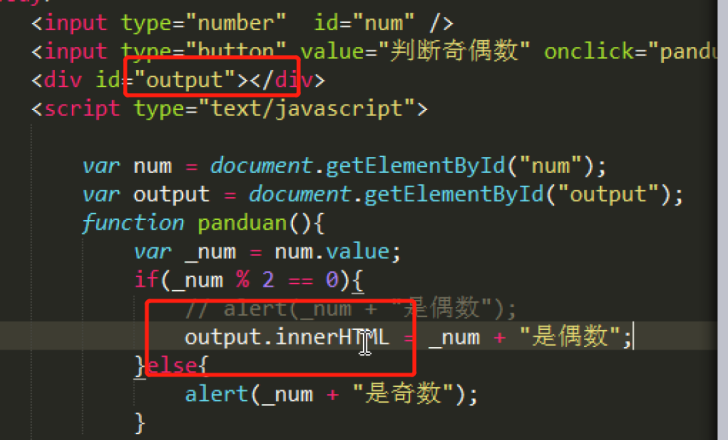


补：若break和continue 后带标识，则跳出标识所在循环

三元运算：条件？条件成立：条件不成立；

InnerHtml：【 ele的ID名.innerHtml = ... 在元素内部添加HTML的代码，**只有双标签才能写，可以添加标签**】



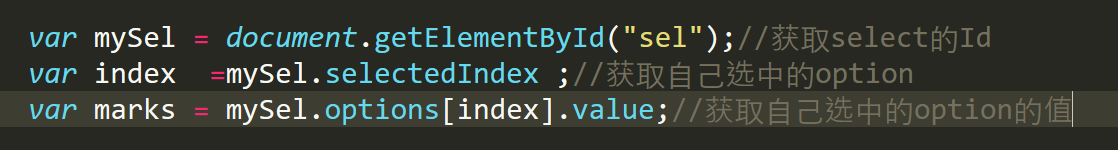
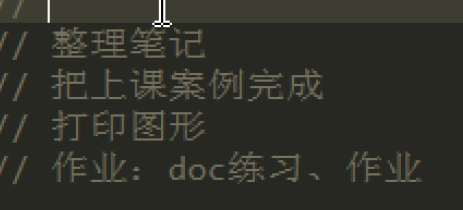
1. 输出到元素里面；
2. 

获取select ：自己选中的option

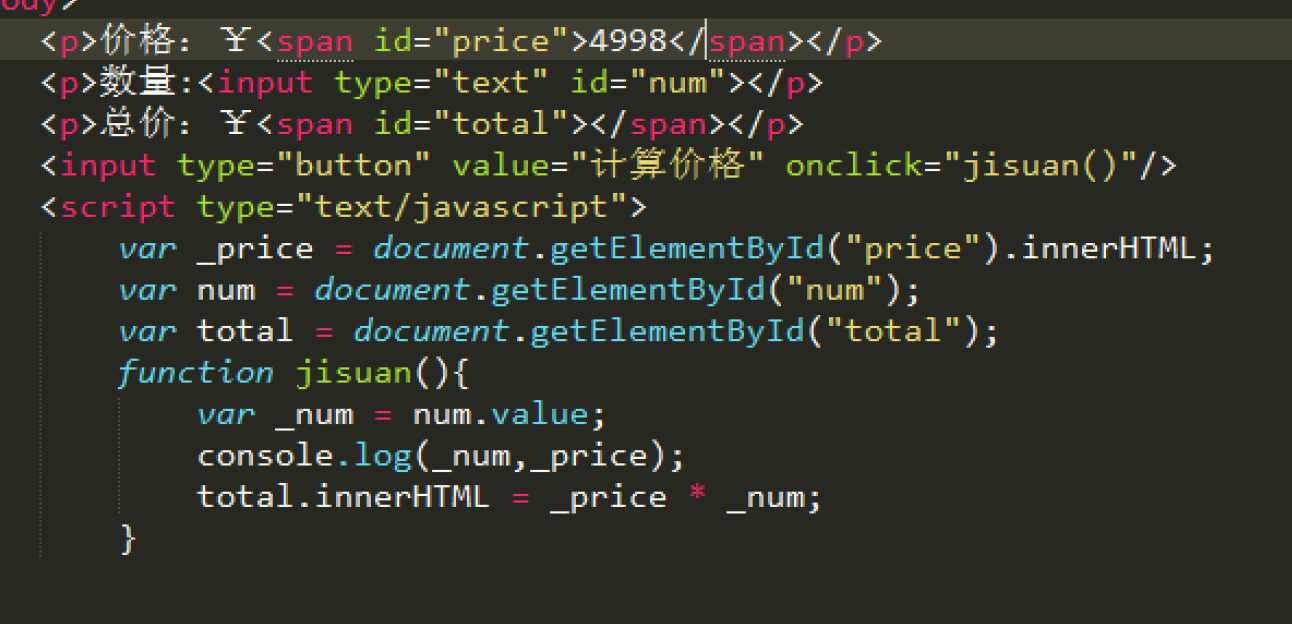
var mySel = document.getElementById("sel");

var index =mySel.selectedIndex ;

var marks = mySel.options[index].value;



作业：



**函数：**

1. **函数的声明：**
   1. **关键字声明： function 函数名（形参）{ 执行的js代码 }**
   2. **赋值式声明： var变量（函数名） = function （形参）{ 执行的js代码 }**
   3. **构造函数 new Function() { }**

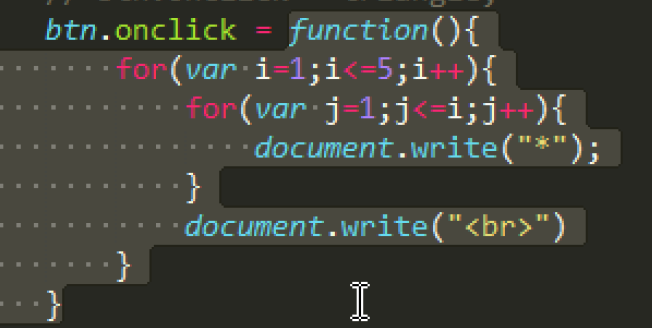
**重复声明函数（同名的函数）：后面的会覆盖前面的（声明提前）**

1. **函数的执行：**
   1. **函数名（） 【去调用这个函数并执行它】**
   2. **事件驱动 onclick=”函数名 ( ) ” 【写在HTML标签中，一般不建议这样使用，结构样式行为三分离】**

**用法： 元素.onclick = 函数名 【这里不加括号】**

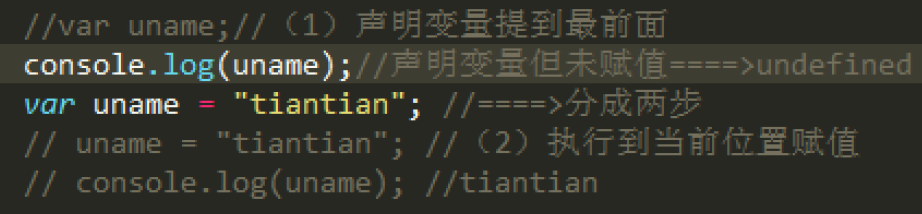


* 1. **函数的分类：**
     1. **内置函数 ：无需自己定义，就能去用**
     2. **自定义函数：自己定义的**
     3. **匿名函数：没有名字的函数 btn.onclick = function(){ JS执行的代码 }**



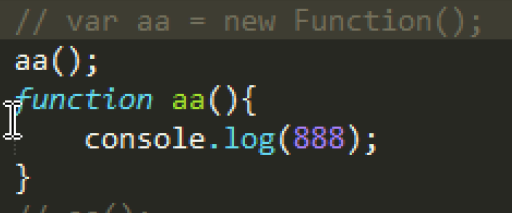
* 1. **声明提前的概念：**
     1. **【在执行JS代码前，会将所有声明提升到当前作用域的最前面】**
     2. **从上到下按照程序的三大流程执行代码：**

【undefined ： 变量声明但未赋值】

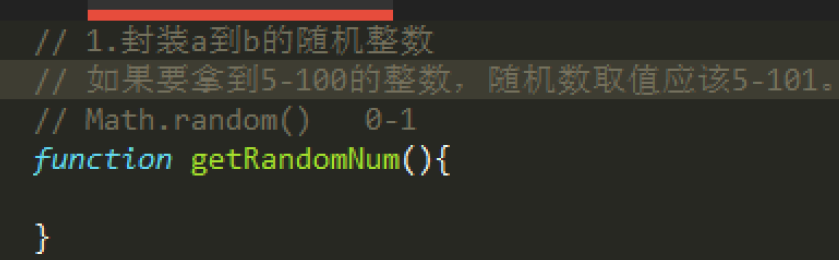


* + 1. **注意：不能在赋值式定义函数前，执行函数，会报错 \*\* is not a function 【原因是声明提前，此时变量为undefined ，是一个值】**
    2. **以下这种方法： 是可以的 ，在关键字声明函数前，执行函数**

**【因为会new Function这样一个对象】**

* + 1. 

**补：随机数：**

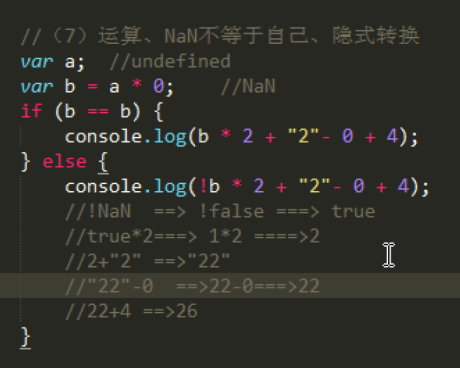
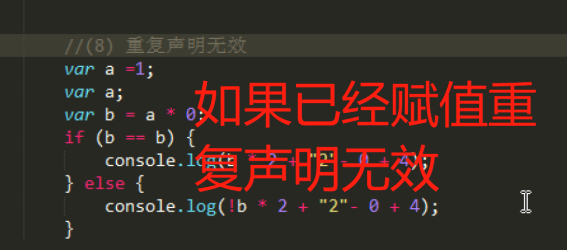


Math.random()\*(max-min+1)+min; 取min 到max的随机数



操作符的优先级：

“（）” > 一元运算符 > 算术运算符 > 关系运算符(大于小于高于等于) > 逻辑运算符（&&>||） > 赋值运算符

**作用域：（当在某个函数内使用某个变量）**

1. 全局作用域：（全局变量）：在函数最外层声明的变量
2. 局部作用域（局部变量）：在函数内声明的变量

作用域链：当函数访问变量时，根据**就近原则**从**内到外**查询变量，这个路径称为作用域链。

**变量的访问规则：【当前函数-->父级函数-->继续上层-->直到最顶层-->not defined】**

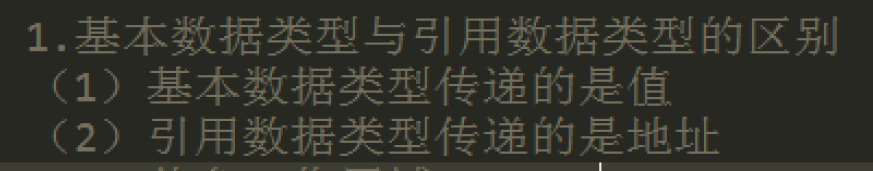
就近原则（如查找变量a）：

\* 使用变量a时先从**当前函数**查找，如果有则可以使用;

\* 如果当前函数无变量a,则往**父级函数**查找，如果找到则使用，并停止查找;

\* 如果在父级函数还是无法找到，则继续**往上一层函数**查找，以此类推;

\* 直到**最顶层(全局作用域)**，如果还是没找到，则报**错误 \*\* is not defined;**



函数的形参与实参：

作用：将函数外部的值传入函数内部

形参：函数定义时的参数（变量）

实参：函数执行时的参数（值） 实参是**基本数据类型**相当于传递的是值 

【arguments】类数组：保存实参的信息

索引：

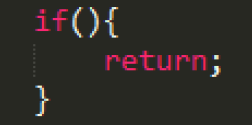


函数的返回值 【return】

作用：

1.将函数内部的值返回到函数外部；外部需要应用变量接受该值

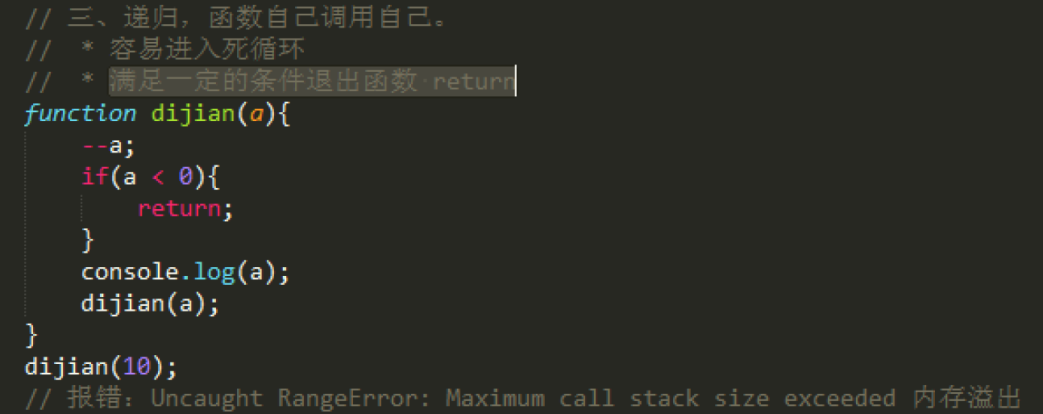
若函数没有书写return，函数默认的返回值为undefined；

1. 跳出函数，终止函数的执行 
2. Return 后面的代码将补执行，只能跳出一层函数

**this:** 当前对象，指的是调用函数的对象，函数大部分手动执行时，this都只想window

（谁调用函数，this指向的就是谁）事件驱动执行的话，this指向驱动事件的那个元素对象

**递归：**函数自己调用自己，容易进入死循环，



**数组：【不确定数组的长度，栈中的内存比较小，数组在栈中存放堆中的地址】**

1. 在JavaScript中，数组中的每个元素的类型是可以不同的；

【是因为JavaScript的弱数据类型决定数组中元素类型可以不同的；  
因为在JavaScript中，定义变量的时候不指定其数据类型，仅仅用一个var来表示当前对象是一个变量，至于其是什么类型的不指定，在后面使用的时候可以赋值不同的数值类型。】

数组创建的方式：

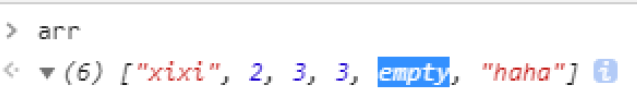
1. 字面量： var arr = [1,2,3..]
2. 使用构造函数创建 var arr = new Array(); var arr = new Array(4); 创建一个长度为4的数组
3. var arr = new Array[“123”,”223”,...] 创建数组时并对元素赋值
4. 【带数值与不带数值的】

**重点（常用的数组方法：8种）：**

（都是小写）

数组的增删改查，数组的索引a[i]

1. 对数组中的元素进行赋值（长度不是固定，如：赋初值的时候，只有3个值，也可以操作a[5]）



1. 如何遍历数组

* push()： 往数组尾部添加一个或多个元素，返回数组新的长度
* pop()：删除数组最后一个元素，返回删除的元素
* unshift()：往数组开头添加一个或多个元素，**返回数组新的长度**
* shift()：删除数组第一个元素，返回删除的元素
* splice(start,deleteNum,…items)：在数组中插入、删除、替换的通用方法
  + start：起始索引位置
  + deleteNum：要删除的数量
  + items：插入的元素（可以是多个）

删除：

修改：

* slice(start[,end])：返回数组的片段或子数组，从start开始到end(不包括end所对应的元素)

如果省略end参数，则截取到数组的最后一项,支持负数

* sort()：将数组中的元素排序，并返回排序后的数组
  + 默认:以字符串的排列方式（转换成ASCII码进行对比）
  + 参数fn（比较函数）：利用fn的返回值来确定数组中两个数的位置（假设两个数为a,b）

Function fn(a,b){

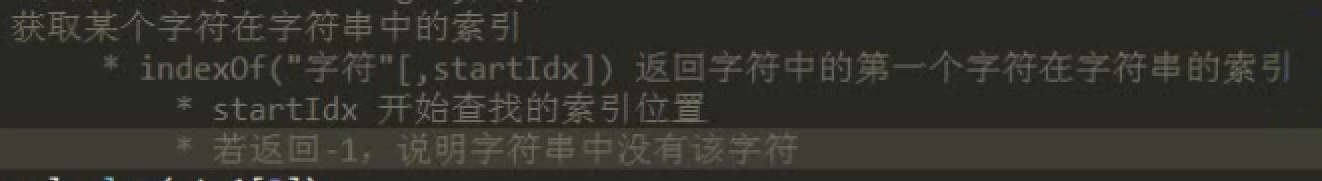
Return a-b;

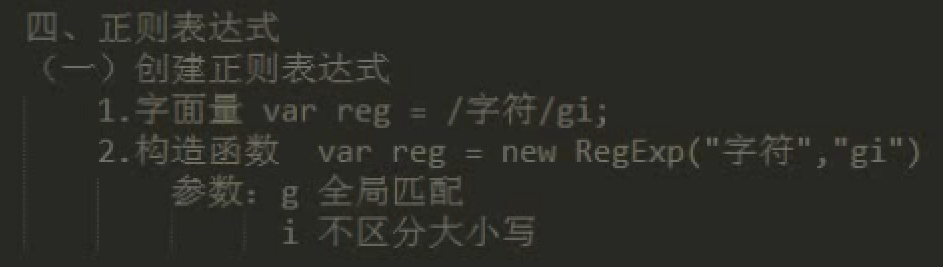
} arr.sort(fn());

* + - 返回负数：确定a在b前，[a,b…]
    - 返回0：不改变现有位置
    - 返回正数：确定a在b后面，[b,a…]
* reverse()：将数组中的元素颠倒顺序，返回逆序后的数组
* join(separator) 返回字符串值，其中包含了连接到一起的数组的所有元素
  + separator为分隔符，默认为逗号

**字符串：**

1. 字符串的创建
   1. 字面量 var str = “”;
   2. 构造函数 var str = new String(“”);
2. 字符串的操作
   1. 通过索引获取字符串的某个字符【str[索引值]==》只能在es5用】【str.charAt(索引值)】
   2. 字符串的长度 str.length 只读
   3. 获取某个字符在字符串中的索引indexOf(“字符”)



1. 字符串的其他方法
   1. replace(str|regExp,’’) 替换字符串  
      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 

# **二、数组对象排序**

1. 如果数组项是对象，我们需要根据数组项的某个属性对数组进行排序，要怎么办呢？其实和前面的比较函数也差不多：

var arr = [{name: "zlw", age: 24}, {name: "wlz", age: 25}];var compare = function (obj1, obj2) {

var val1 = obj1.name;

var val2 = obj2.name;

if (val1 < val2) {

return -1;

} else if (val1 > val2) {

return 1;

} else {

return 0;

}

}

console.log(arr.sort(compare));

**时间对象：（时间对象的创建）**

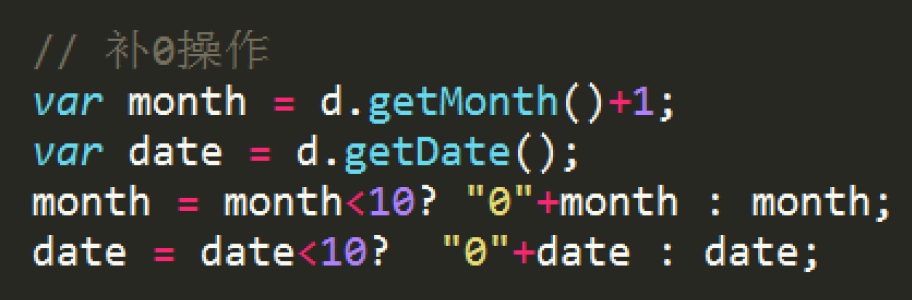
1. 不带参数：返回代码执行的时间（本地的时间）
2. 带参数：返回的是具体的日期
   1. 数字：var d = new Date(20162134) 【数字：距离1970-01-01的毫秒数】
   2. 字符串：var d = new Date(“2020/09/20 12:04:09”) //具体的日期时间有固定的格式【1997-02-18 或 1998/09/17】



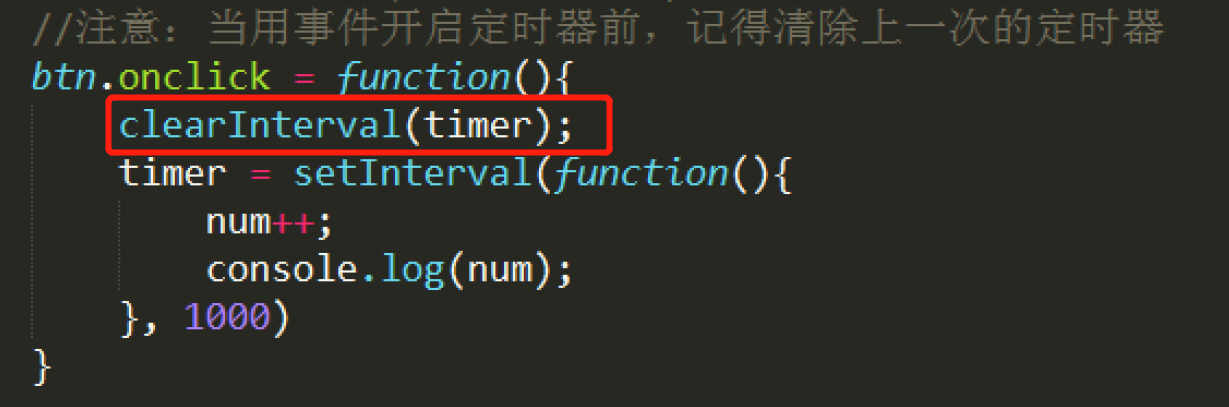
转化日期格式：

1.时间对象的获取方法

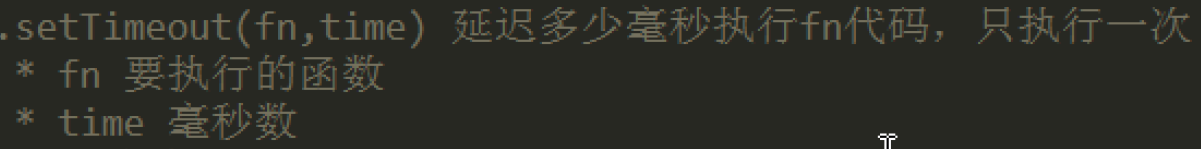
* 1. getFullYear() 返回的是对象的年份
  2. getMonth() 返回的是对象的月份 返回0-11，代表1-12
  3. getDate() 返回的是对象的日期
  4. getDay() 返回的是对象的星期几，返回0-6，代表星期日到星期六
  5. getHours() 返回的是小时数
  6. getMinutes() 返回的是分钟数
  7. getSeconds() 返回的是秒数

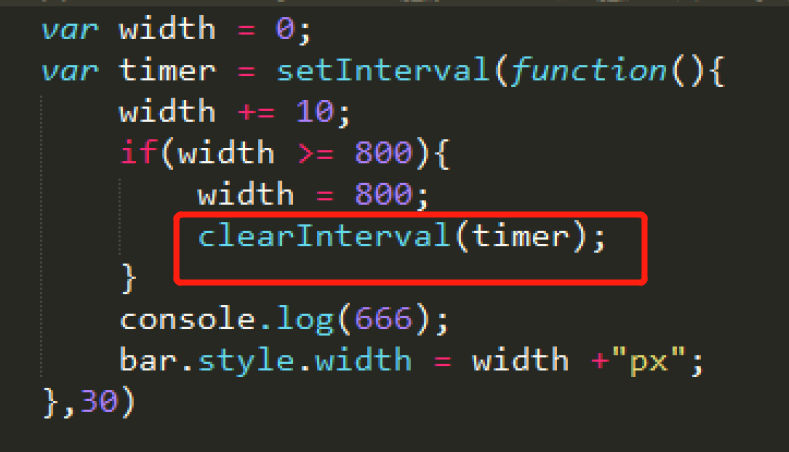
1. 时间和日期的补0操作
   1. 
2. 定时器
   1. setInterval(fn,time) 每隔多少毫秒执行fn代码，当定时器执行的时候才执行的fn代码
      * 1. 【fn:要执行的函数[不加括号] time：每间隔多少毫秒执行fn】



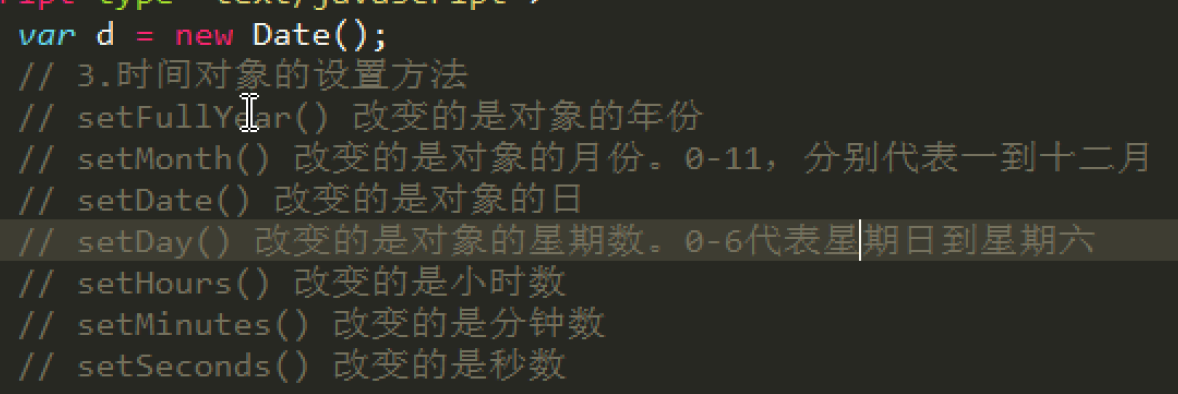


1. Setimeout(fn,time)



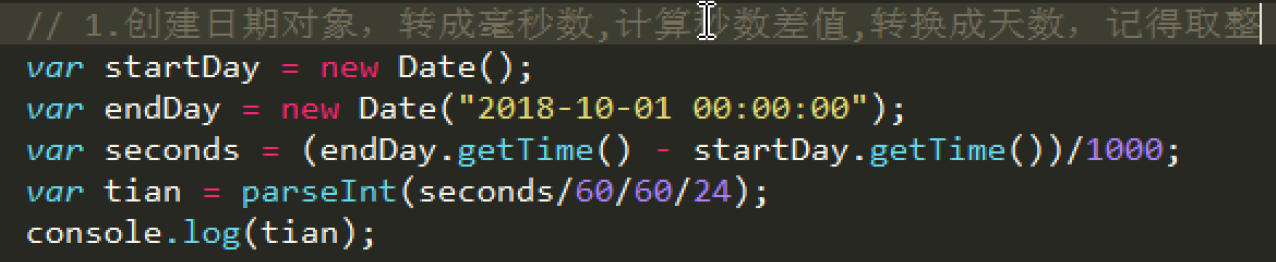
* 1. setInterval(function(){ } , 1000)
  2. 清除定时器 clearInterval(定时器的名字) 【ex：timer】
  3. 

1. 时间对象的设置方法【无法设置星期数】

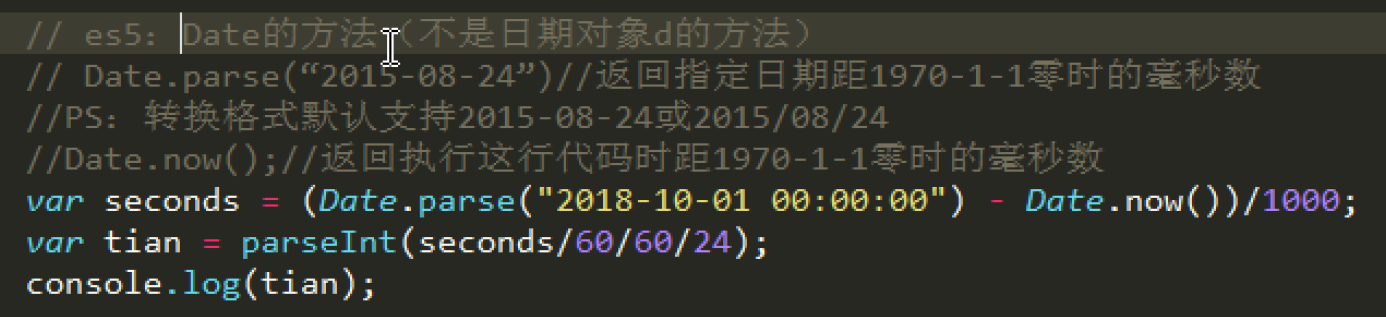




1. 日期处理【判断两个日期的相差天数】
   1. 获取毫秒数 getTime()
   2. 设置毫秒数 getTime()



Es5: Date的方法（不是日期对象的方法）



获取浏览器可视区域的宽度：

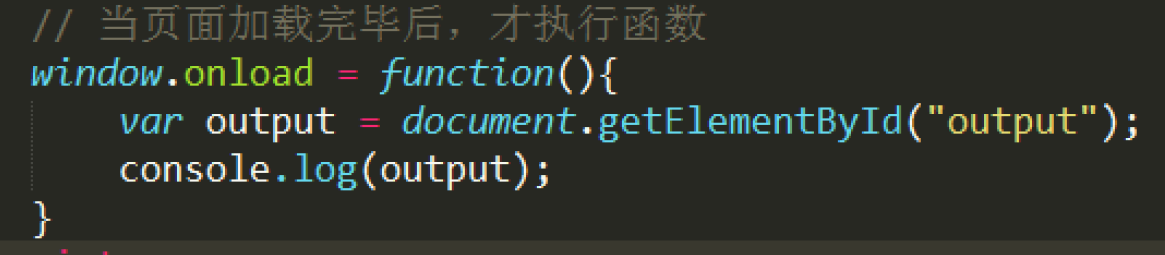


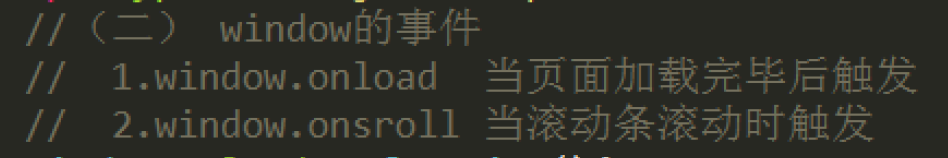
**对象：**



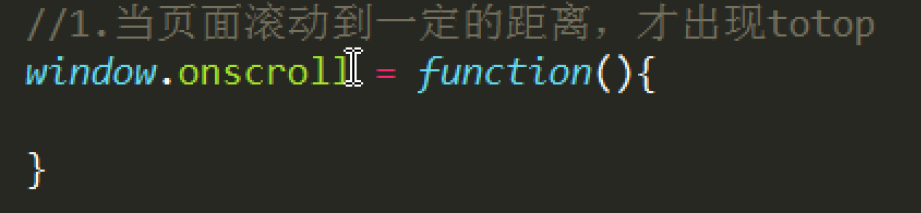
1. window对象
   1. 最顶端的对象
   2. 省略了调用对象的方法，其实都是window的方法【尽量不要对属性及事件，神略window对象的调用】
   3. 定义在全局环境下的变量都会成为window对象的属性【若函数内没有用var声明变量，也是全局变量，也是window的属性】
      1. 命名冲突
      2. 全局污染
   4. 全局变量起名字要避免window已存在的命名（name,top）
   5. 不要对window属性进行赋值【ex:window.innerHTML = 500,以后获取浏览器的可视化区域都为500】
   6. 通过var声明的全局变量，无法用delete删除

注：









判断滚动条滚动的距离：

1. window.scrollX
2. window.scrollY
3. window.scrollTo(x,y) ; 设置浏览器当前的滚动距离
4. window.scrollBy(x,y) ; 设置浏览器基于当前浏览器的滚动距离，可以去负值（向下）

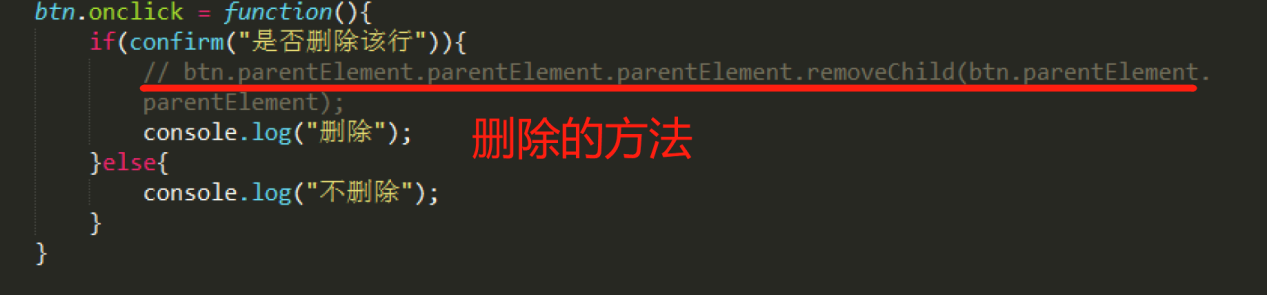
弹窗：



1. comfirm() 确认框



删除操作：

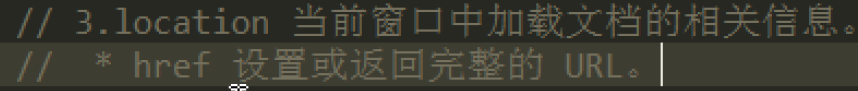
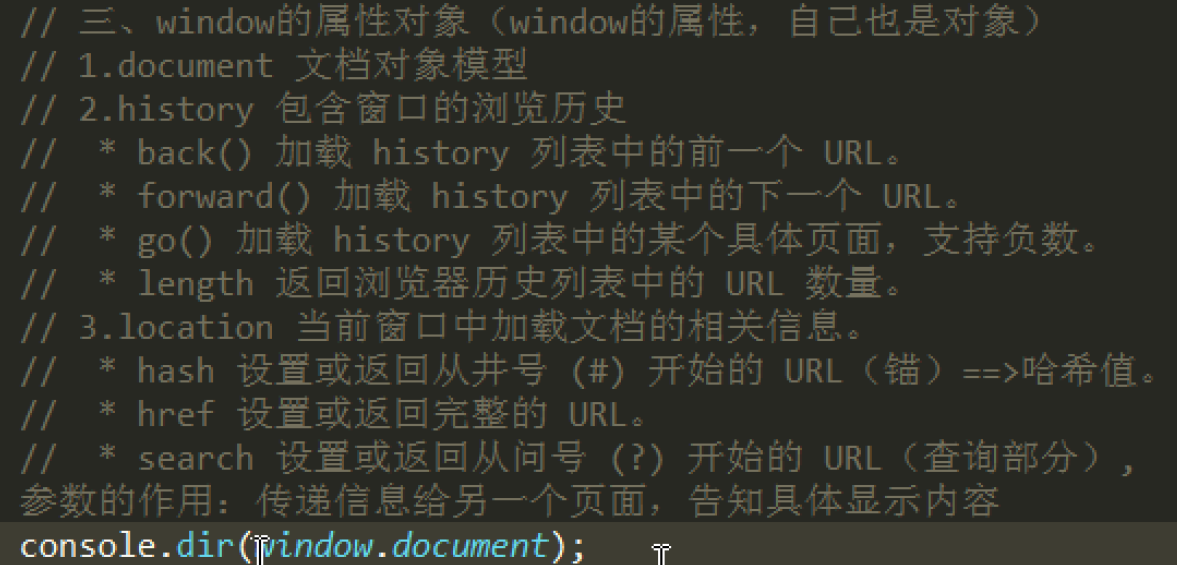


open() 打开浏览器

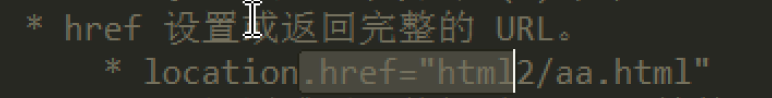
close（）关闭窗口

print() 跳出打印框

属性对象：【既是window属性，也是对象】

1. document 文档对象模型
2. History 包含浏览器的【移动端会用得上】
3. console.dir()更深结构的
4. 
5. 

在页面中的那一长串：

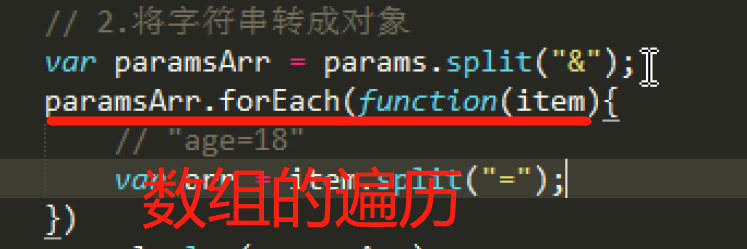


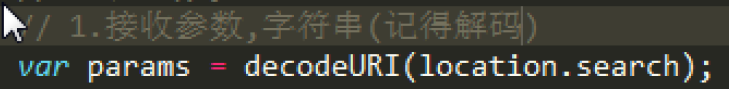
Location:当前窗口中加载文档的相关信息，hash锚点，哈希值#aaa

seach

This.getAttribute(“属性名”) 【获取自定义HTML属性值】

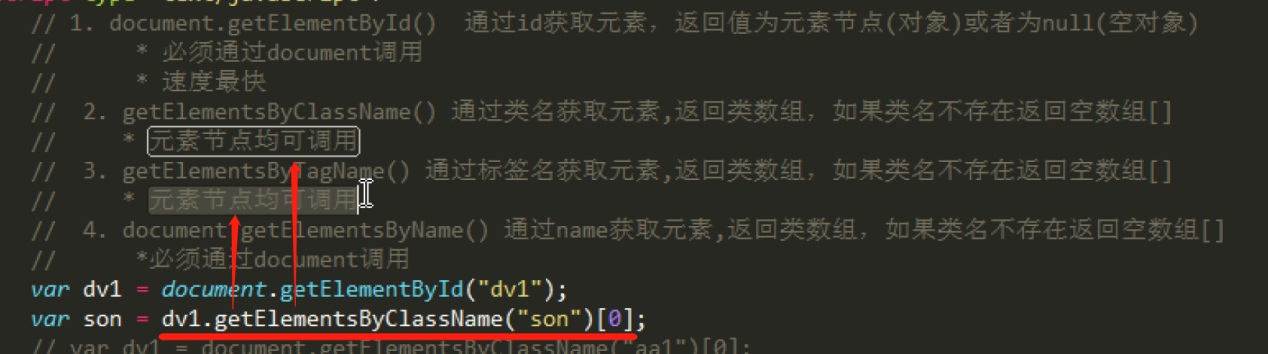


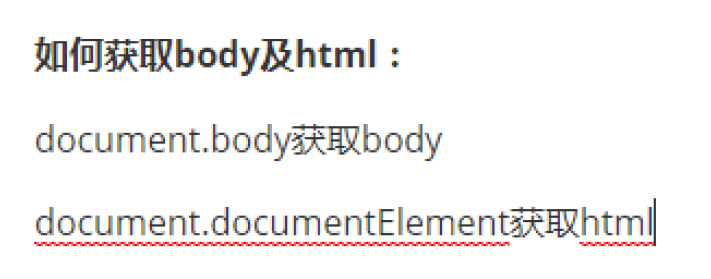


【最好手动转码】

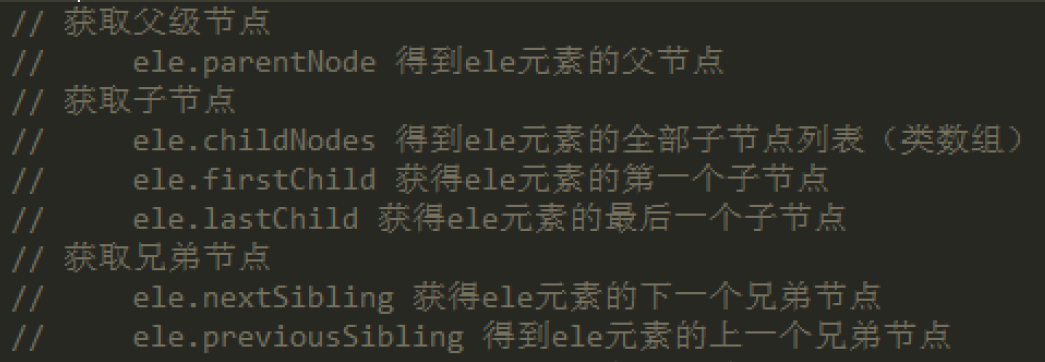
JS实现div在屏幕居中，页面尺寸改变的时候也居中



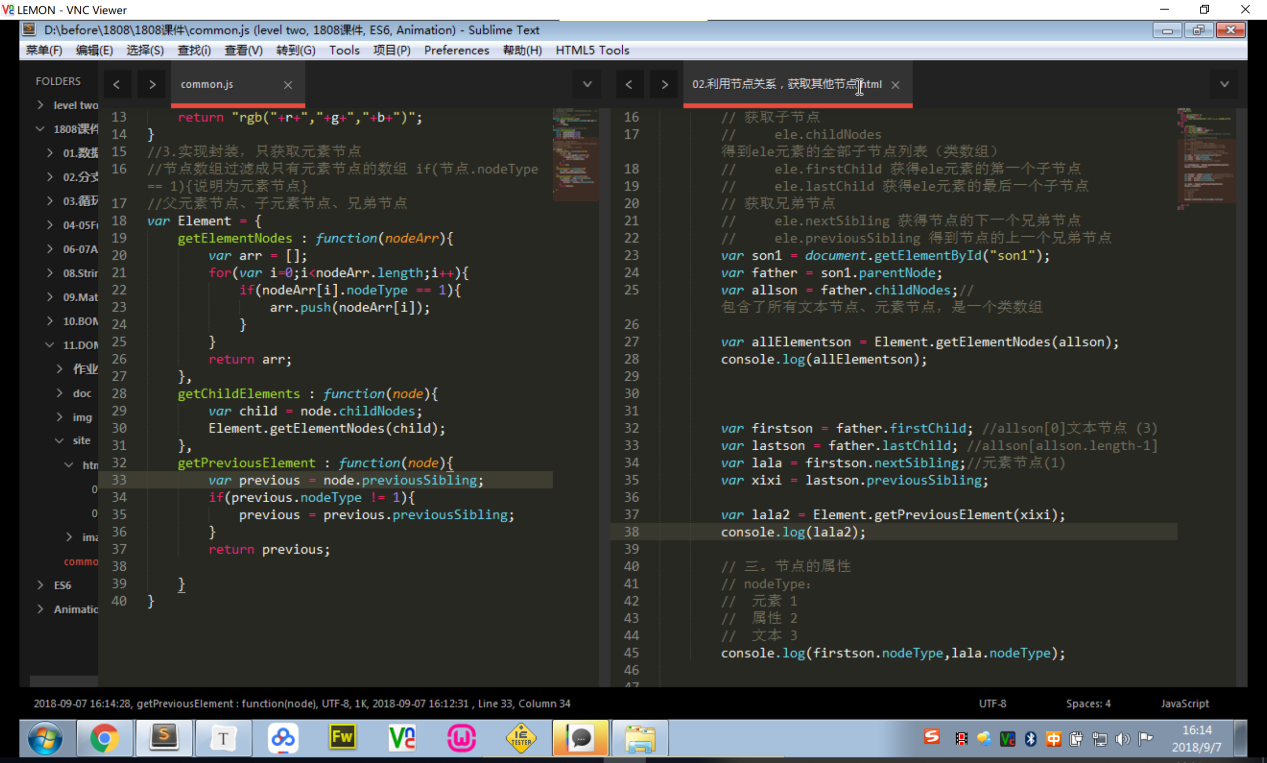


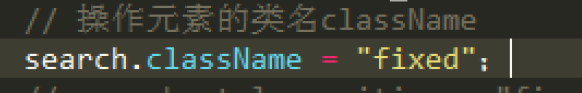


**利用节点关系、获取其他节点：（包含元素节点、文本节点）**

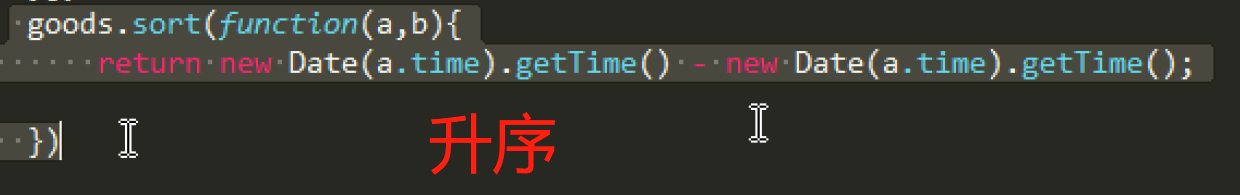


节点的属性 nodeType 【 元素节点的nodeType是1，文本节点的nodeType是3，属性节点的nodeType是2 】

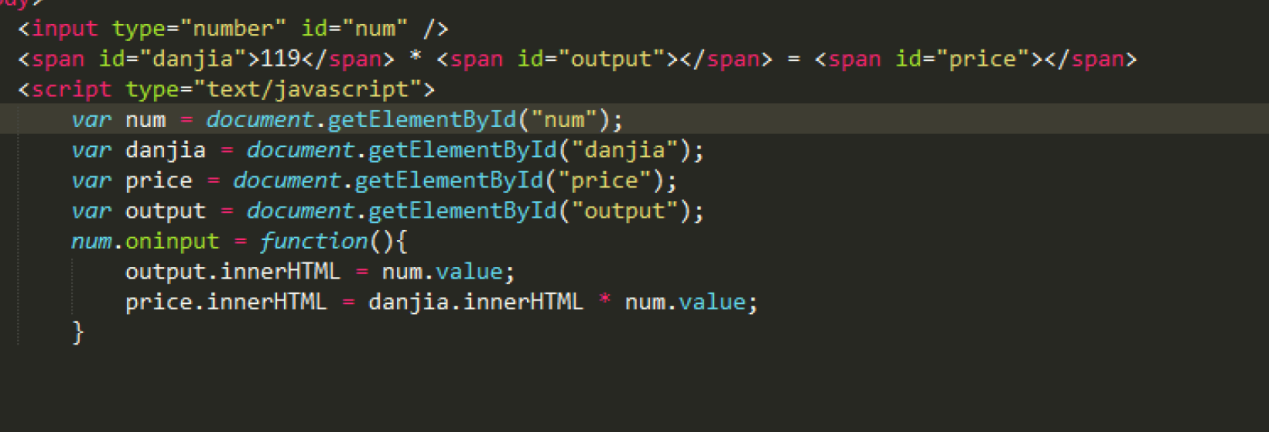


操作元素的类名

转成时间对象



项目：



报错：

很有可能你的符号错了，中英文符号，或者多了少了括号。

不能读取空的内部HTML。这个报错是一个初学者的问题。实际上，在页面的HTML结构中，innerHTML是有实际的值并可以在console进行获取查询到。

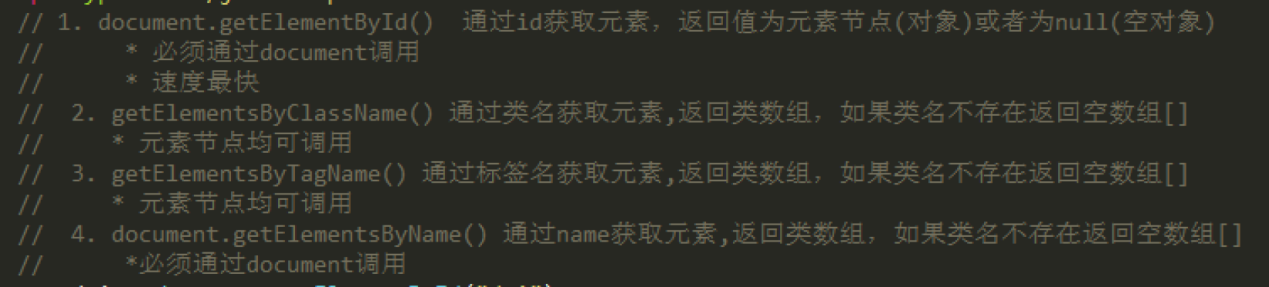
问题：script中关于DOM部分放在和body标签前。

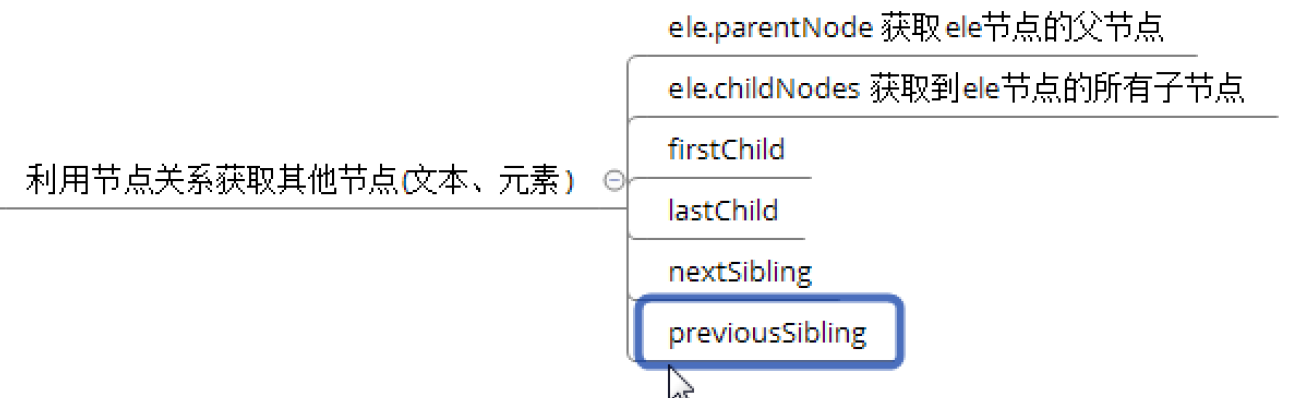
根据浏览器的渲染原理，HTML代码从上到下执行代码，当浏览器JS解析器解析到script并进行DOM操作，下面的DOM结构还没有进行搭建。那么就会提示不能正确读取内部的HTML信息。

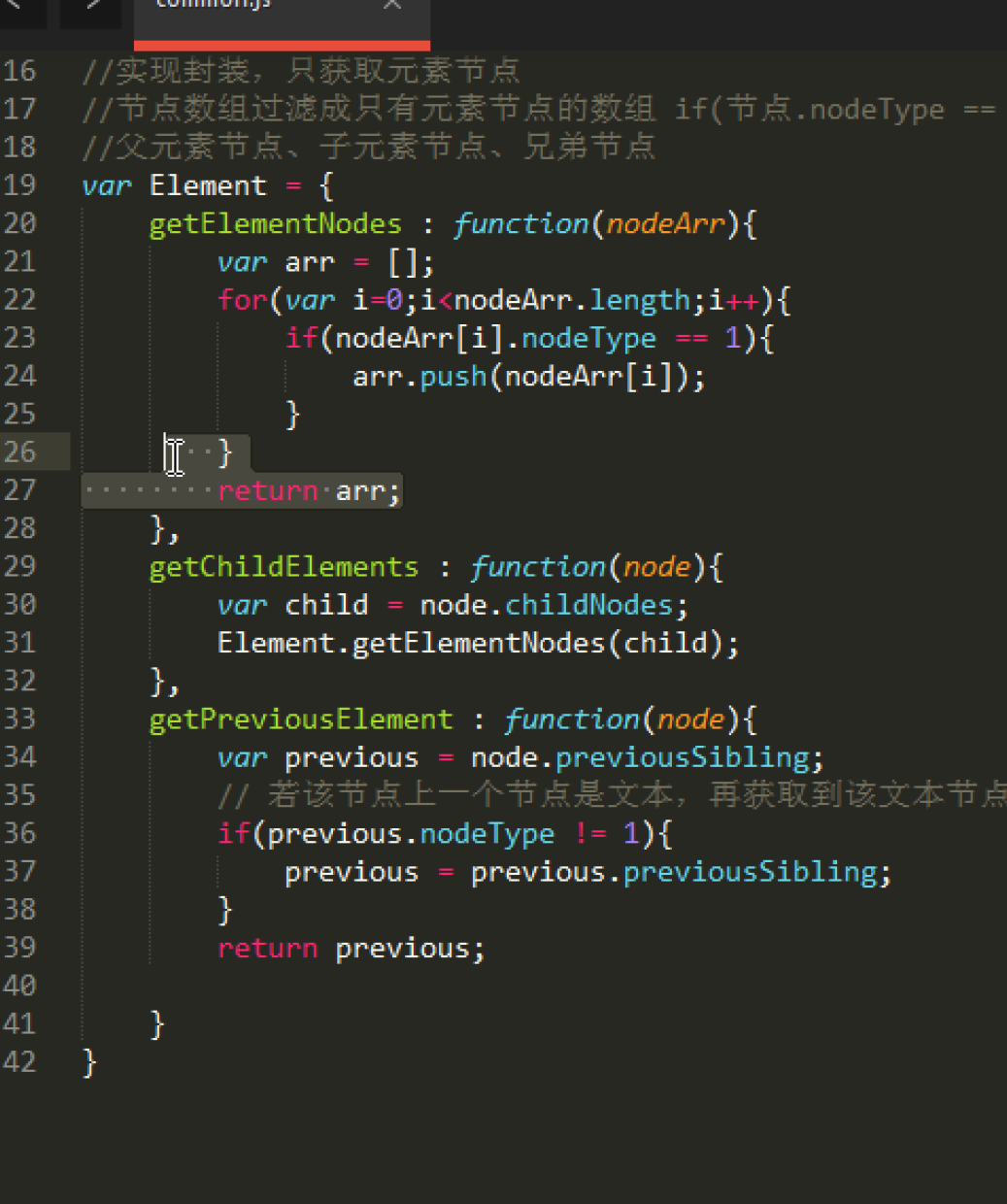
解决思路：将涉及DOM读取部分的script放在DOM结构后面。</body>标签前面。或者在前置的script标签写明：window.onload等，确保DOM树加载完毕后再执行这段script代码。

Dom对象：

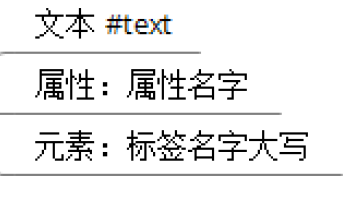
获取元素节点：document.getElementById()返回元素节点对象或null



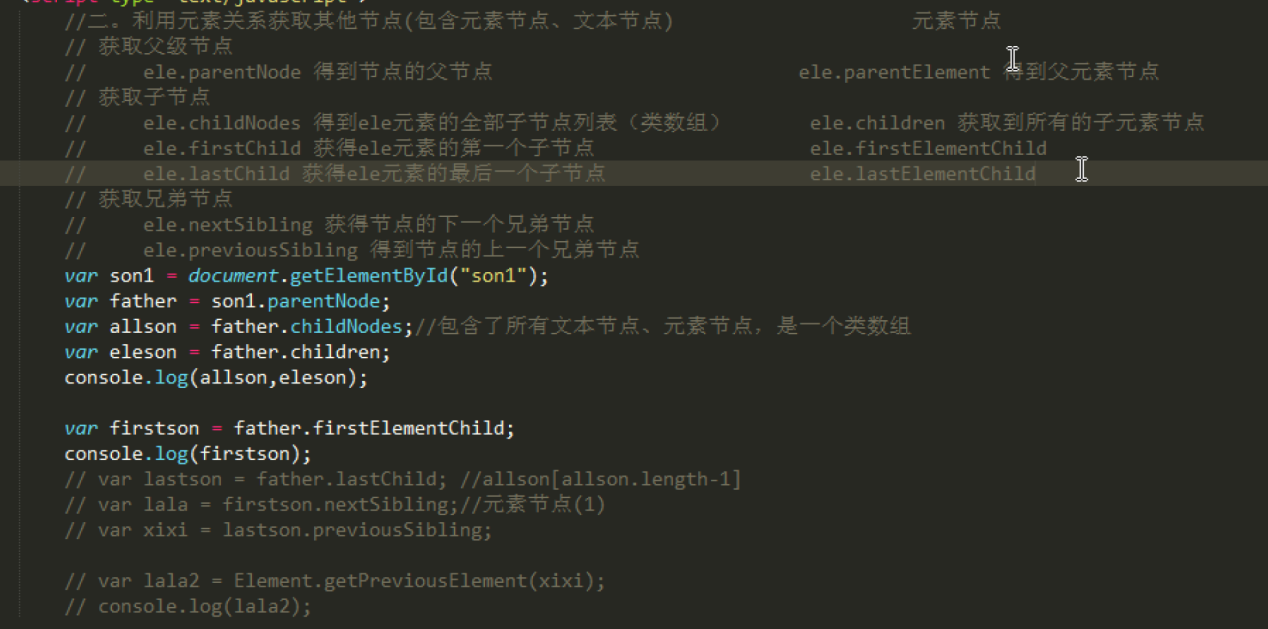


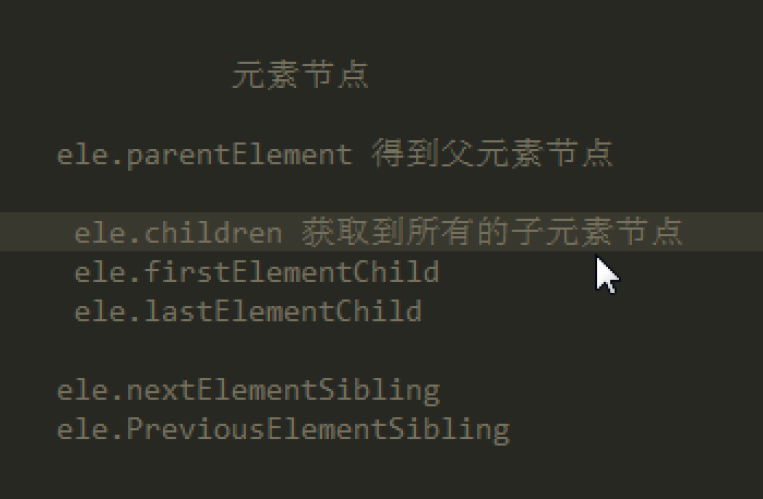
封装实现过滤，只获取元素节点

节点的三大属性：nodeType【节点的类型：文本返回值3；属性返回值2；元素的返回值1；】

nodeName 【节点名字：返回节点的名字【】】

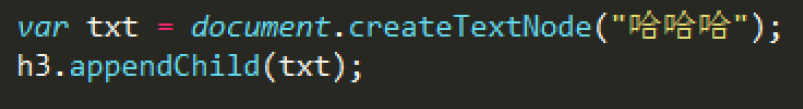
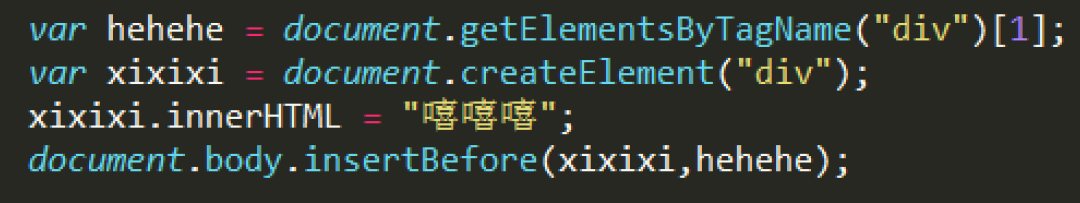
nodeValue【文本：返回文本内容，属性：属性值，元素：很多空格和null】





1. 重点：节点的增删改查
   1. 节点的创建与插入
      1. 创建指定的元素节点：**Documnet.createElement(“标签名”);**

比如：Documnet.createElement(“h3”)

* + 1. 节点的插入：**parent.appendChild(ele)** 给父元素添加**最后**一个子元素
    2. 创建文本节点 ：Documnet.createTextNode(“啦啦啦”); 【了解一下就就可以】
    3. 创建属性节点：var a = Documnet.createAttribute(“id”); a.nodeValue = “id名” h3.setAttributeNode(“”) 【h3.id = “wangcai”】
    4. 在指定的节点插入【insertBeefore】
    5. 
    6. 节点的删除：parent.removeChild();
    7. 节点的复制：jiedian.cloneNode(Boolean复制程度)【取值：false 浅复制（只能复制当前行无法复制里面的内容，不常用），true 深复制：这个标签里的所有的东西都会呗复制】

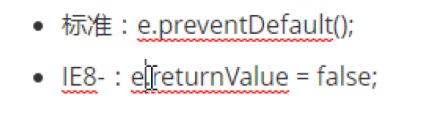
补： --》判断这个数组里是否有\_msg 这个值



1. 节点的属性和方法：
   1. HTML的标准属性：DOM节点均可通过.或[] 获取
   2. DOM操作标准属性：HTML





1. 浏览器的默认行为：【】
   1. a标签自动跳转
   2. 拖拽鼠标文字被选中
   3. Submit自动提交
   4. 右键菜单