1.getElementById

通过id获取的是一个，所以没s，通过tagname和classname获取的是集合，所以有s

2.

DOM中class要用className

CSS名规则如下：去除-，-后面的单词首字母大写（如font-size写成fontSize）。

3.

IE中：currentstyle获取css的style；Firefox：getComputedStyle

（1）style 只能获取行间样式，但能设置样式。

（2）getComputedStyle 和 currentStyle 能够获取 行间样式/非行间样式/浏览器默认样式，但存在浏览器兼容问题，且不能设置样式。

6.取消事件冒泡

IE：event.cancelBubble=true

others：event.stopPropagation（）

7.取消默认行为：

IE：event.returnValue = false

others：event.preventDefault（）

8.ctrlKey,shiftKey,altKey

9.

aBtn[i].index=i;

11.

Math.ceil向上取整 Math.floor向下取整 Math.abs绝对值

13.定时器里的this是window对象

可在setInterval中套一层函数，用that=this来引用对象中的this

事件里的this是事件对象（如按钮），解决方法同上

14.继承

Function son(id){

Father.call(this,id);

}

For(i in Father.prototype)

{

Son.prototype[i]=Father.prototype[i];

}

16.

获取表格元素：tBodies、tHead、tFoot、rows、cells

oTab.tBodies[0].row[3].cells[5]

tbody中的第3行第5列

18.

表单：onsubmit onreset

return confirm(‘是否清空’)；

19.事件绑定

IE方式

attachEvent(事件名称, 函数)，绑定事件处理函数

detachEvent(事件名称, 函数)，解除绑定

DOM方式

addEventListener(事件名称,函数, 捕获) preventDefault()取消默认事件

removeEventListener(事件名称, 函数, 捕获)

绑定匿名函数无法解除

IE下事件绑定this指向window

20.String包装类型

concat（）连接字符

trim（）删除字符开头和结尾的空格，返回字符副本

获取类

charAt(index)指定字符

charCodeAt指定字符编码、fromCharCode执行相反操作

查找类

indexOf()、lastIndexOf()、search()正则--查找

match、replace—匹配和替换

截取类

slice(1,3)、substr(1,3)、substring(1,3)

substr的第二个参数是返回字符的个数，slice、substring则是index(0开始)

3个都返回基本数据类型的字符串值，对原始值无影响

其他

split—根据分隔符，拆分数组

toLowerCase、toUpperCase

match（）、search（）、replace（）用于在字符串中匹配模式的方法

localeCompare（参数）比较字符串大小，参数排在前返回1

21.array

从尾部添加push

从尾部删除pop

从头部删除shift

从头部添加unshift

转换、获取

splice（开始位置，长度，元素） 先删除一段元素，再添加一段元素

join 与split用法相反

slice

排序

数字

sort(function(num1,num2){

return num1-num2;

})

汉字

sort(function(str1,str2){

return str1.localeCompare(str2);})

reverse反转

其他

arr1.concat(arr2)数组连接

+连接后的的类型是string

es5数组方法：通过 . 来调用

forEach（数组元素，元素索引，数组）

map（每个数组元素），返回新数组，不修改调用的函数

reduce() 方法接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终计算为一个值。

filter（），跳过稀疏数组中缺少的元素，返回数组总是稠密的。新数组中的元素是通过检查指定数组中符合条件的所有元素。返回布尔值

every（），当且仅当针对数组中的所有元素调用判定函数都返回true，才返回true

some（），当数组中至少有一个元素调用判定函数返回true，就返回true

indexOf（）、lastIndexOf（）搜索整个数组中具有给定值的元素，返回找到的第一个元素的索引，若无则返回-1

判定类型Array.isArray（）

Object.prototype.toString.call(obj)

console.log(Object.prototype.toString.call(123)) //[object Number]  
console.log(Object.prototype.toString.call('123')) //[object String]  
console.log(Object.prototype.toString.call(undefined)) //[object Undefined]  
console.log(Object.prototype.toString.call(true)) //[object Boolean]  
console.log(Object.prototype.toString.call({})) //[object Object]  
console.log(Object.prototype.toString.call([])) //[object Array]  
console.log(Object.prototype.toString.call(function(){})) //[object Function]

typeof检测基本类型：**number**、**string**、**boolean**、**null**（“object“）、**undefined**、function（”function“）

**instanceof** **运算符**用来测试一个对象在其原型链中是否存在一个构造函数的 prototype 属性。

引用类型：Object、Array、Date、RegExp、Function、基本包装类型（Boolean、Number、String）、单体内置对象（Global、Math）

2+.DOM操作

创建新节点：createDocumentFragment、createElement、createTextNode

添加、移除、替换、插入：

appendChild（1.先移出父级 2.添加到新的父级）、removeChild、replaceChild、insertBefore

复制节点：cloneNode（true），true为深复制

获取、设置（修改）、删除属性：getAttribute、setAttribute、removeAttribute

获取子节点

firstChild、lastChild、childNodes[0]（childNodes.item(0)）会获取空白节点，不推荐

firstElementChild、lastElementChild、children[0]取代上述属性

新增API，可以通过Document和Element类型的实例调用

querySelector()、querySelectorAll()接收一个CSS选择符参数

querySelectorAll()返回一个NodeList对象，可以用[]和item()取得元素

22.变量

parseInt()、parseFloat()

number()字符串严格转换成数字

isNaN()判断是不是NaN

匈牙利表示法

o>object

a>array

f>float

i>integer

s>string

23.

ES5中没有块级作用域，可以用闭包模仿

ES6中用let定义变量，在for、while、if中实现块级作用域

const，变量指向的那个内存地址不得改动，对于基本类型来说，就是常量，对于引用类型，则是指针不变，对象或数组的内容可变。

24.事件委托

var ev = ev || window.event;

var target = ev.target || ev.srcElement;

取代用for循环

提高性能、之后添加的子元素都有事件

25.函数声明转换为函数表达式

function aaa(){ var a=3; alert(a); }

(function (){ var a=3; alert(a); })();

26.apply&call

call函数和apply方法的第一个参数都是要传入给当前对象的对象，及函数内部的this。后面的参数都是传递给当前对象的参数。apply()是函数对象的一个方法，它的作用是改变函数的调用对象，它的第一个参数就表示改变后的调用这个函数的对象。因此，this指的就是这第一个参数。

<script>  
   var func=new function(){this.a="func"}  
    var myfunc=function(x){  
        var a="myfunc";  
        alert(this.a);  
        alert(x);  
    }  
    myfunc.call(func,"var");  
</script>

弹出func和var

对于apply和call两者在作用上是相同的，但两者在参数上有区别的。  
对于第一个参数意义都一样，但对第二个参数：  
apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。  
如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3])  
同时使用apply的好处是可以直接将当前函数的arguments对象作为apply的第二个参数传入

27.正则表达式

search查找 substring获取子字符串 charAt获取某个字符 split分割字符

match 获取匹配的字符 replace替换字符

js风格 new RegExp(”a”,”i”)

perl风格 /a/i

选项

i=ignore g=global全部的

量词

{n,m}最少n次，最多m次

{n, }最少n次，最多不限

{ ,m}最少不限，最多m次

{n}正好n次

+等于｛1，｝

\*等于｛0，｝

？等于｛0,1｝

字符类

任意字符：[abc] 例子 1[abc]2----1a2 1b2 1c2

范围：[0-9]、[a-z] 例子 id[0-9]----id0、id7

排除：[^a] 除了a以外所有

转义字符：

. 任意字符（包括所有字符）

\d=digital数字 \w [a-z0-9\_] \s space空格

\D=digital除了数字 \W 除了[a-z0-9\_] \S 除了space空格

^行首 $行尾

[\u4e00]-[\u9fa5]汉字

28.offset scroll client

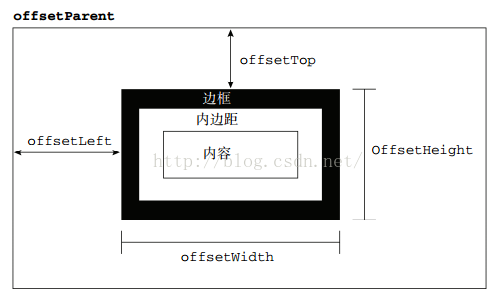
offsetWidth和offsetHeight用来获得对象的大小，由自身宽高、内边距、边框组成。

**offsetHeight和style.height的区别**

1> style.height只能获取行内样式**，**offsetHeight可以获取行内样式和内嵌样式

2> style.height是字符串（而且带单位），offsetHeight是数值

3> style.height可读可写，offsetHeight是只读属性



offsetLeft和offsetTop用来获得对象的位置，到距离自身最近的（带有定位的）父元素的距离，若没有则以body为准。

**offsetLeft和style.left的区别**

1> style.left只能获取行内样式，offsetLeft 可以获取行内样式和内嵌样式

2> offsetLeft 只读，style.left可读可写

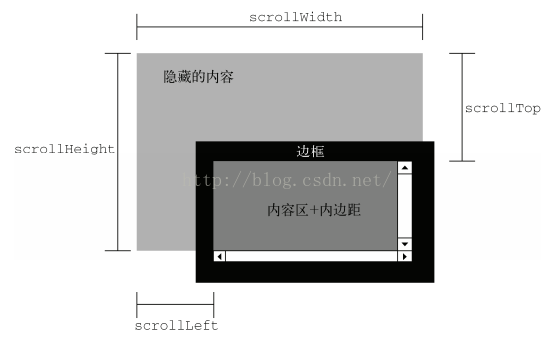
3> offsetLeft是数值，style.left是字符串并且有单位px

4> 如果没有加定位，style.left获取的数值可能是无效的

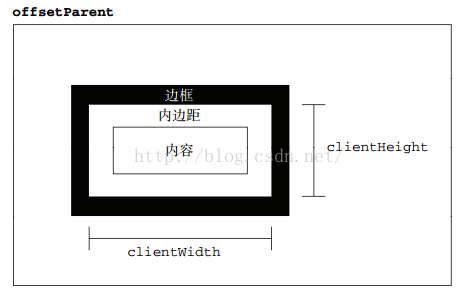
5> 最大区别在于offsetLeft以border左上角为基准，style.left以margin左上角为基准

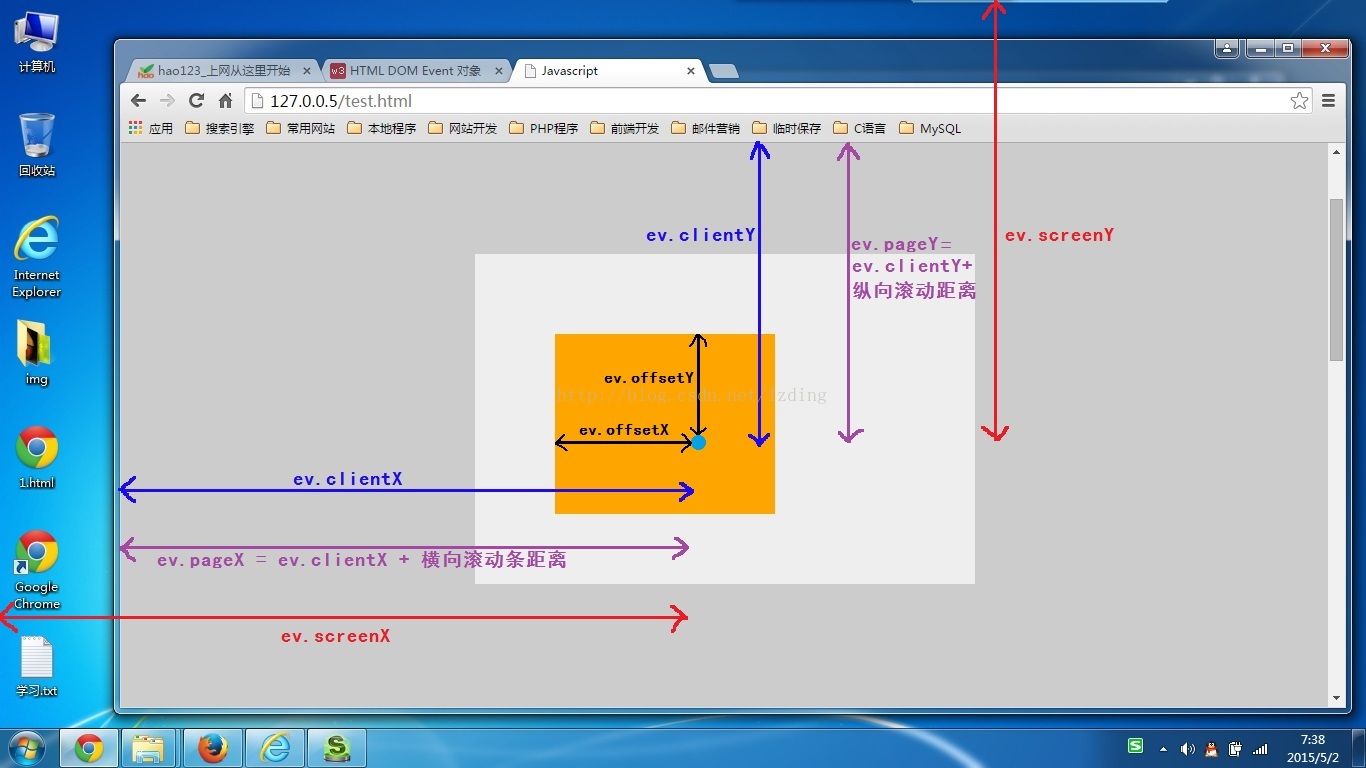
scrollHeight和scrollWidth对象内部实际内容的高度、宽度，包括内容和内边距，不包括边框。

scrollLeft和scrollTop显示内容区到整个内容区的距离



clientWidth和clientHeight包括内容区和内边距，不包括边框





29.

innerHTML 设置或获取位于对象起始和结束标签内的 HTML

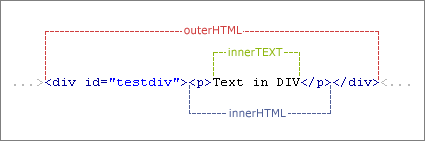
outerHTML 设置或获取对象及其内容的 HTML 形式

innerText 设置或获取位于对象起始和结束标签内的文本

outerText 设置(包括标签)或获取(不包括标签)对象的文本

innerText和outerText在获取时是相同效果，但在设置时，innerText仅设置标签内的文本，而outerText设置包括标签在内的文本。

innerHTML和outerHTML、innerText与outerText的不同之处在于：   
　　1）、innerHTML与outerHTML在设置对象的内容时包含的HTML会被解析，而innerText与outerText则不会。   
　　2)、在设置时，innerHTML与innerText仅设置标签内的文本，而outerHTML与outerText设置包括标签在内的文本。



30.同步与异步

Js只有一条主线程

**同步任务指的是，在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务；**

**异步任务指的是，不进入主线程、而进入"任务队列"（task queue）的任务，只有等主线程任务执行完毕，"任务队列"开始通知主线程，请求执行任务，该任务才会进入主线程执行。**

**具体来说，异步运行机制如下：**

**（1）所有同步任务都在主线程上执行，形成一个执行栈（execution context stack）。  
（2）主线程之外，还存在一个"任务队列"（task queue）。只要异步任务有了运行结果，就在"任务队列"之中放置一个事件。  
（3）一旦"执行栈"中的所有同步任务执行完毕，系统就会读取"任务队列"，看看里面有哪些事件。那些对应的异步任务，于是结束等待状态，进入执行栈，开始执行。  
（4）主线程不断重复上面的第三步。**

31.检测屏幕旋转

window.orientation===’-90’||window.orientation===’90’

32.

a["城市"]="北京";

a["面积"]=16800;

a["人口"]=1600;

在Javascript语言中，关联数组就是对象，对象就是关联数组。这一点与php语言完全不同，在php中，关联数组也是数组。

33.闭包中的this指向window

因为this指向调用函数的对象，闭包是函数内的函数，严格模式下为undefind

34. 也就是说如果某个microtask任务被推入到执行中，那么当主线程任务执行完成后，会循环调用该队列任务中的下一个任务来执行，直到该任务队列到最后一个任务为止。而事件循环每次只会入栈一个macrotask,主线程执行完成该任务后又会检查microtasks队列并完成里面的所有任务后再执行macrotask的任务。

整体的js代码这个macrotask先执行，同步代码执行完后有microtask执行microtask，没有microtask执行下一个macrotask，如此往复循环至结束

microtasks: process.nextTick, Promises, Object.observe, MutationObserver

macrotasks: script(整体代码),setTimeout, setInterval, setImmediate, I/O, UI rendering