**jquery与js区别混淆**

隐式迭代、链式调用、读写一体

jQ有一个index（方法），返回对象的索引，找不到返回-1

var $v=$("#v"); //jQuery对象

var v=$v.get(0); //转DOM对象

`模板字符串${变量}`

入口函数:

1. $(**function**(){
2. // 执行代码
3. });
4. $(document).ready(**function**(){
5. // 执行代码
6. });

操作内容：

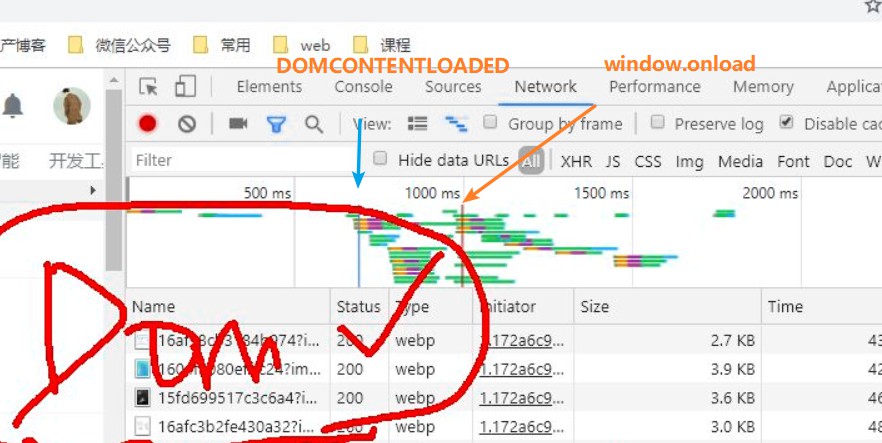
1. $().html()  获取选中标签的内容（包括html标签）
2. $().text()  获取选中的标签的文本内容
3. $().val()   ▶获取表单元素的内容（value）

操作属性：

1. $().attr('属性名')  获取属性值
2. $().attr('属性名','属性值'） 修改属性值
3. $().prop('属性名','属性值') 一般用于表单属性查找
4. 可同时设置多个属性
5. $("#runoob").attr({
6. "href" : "http://www.baidu.com",
7. "title" : "百度"
8. })

操作class：

1. addClass() - 向被选元素添加一个或多个类
2. removeClass() - 从被选元素删除一个或多个类
3. toggleClass() - 对被选元素进行添加/删除类的切换操作
4. document.addEventListener("DOMContentLoaded",**function**(){
5. //执行代码
6. },options:**false**)
7. window.onload = **function** () {
8. // 执行代码
9. }



1. innerHTML  获取选中标签的内容（包括html标签）
2. innerText   获取选中的标签的文本内容
3. nodeValue 只能获取文本节点里的内容
4. 表单元素.value 获取表单元素的内容
5. Node.attribute()  返回所有属性 以对象形式
6. Node.getAttribute(name)  获取属性值
7. Node.setAttribute(name, value)  ▶修改属性值
8. Node.removeAttribute(name)   删除属性

▶js里面改现有 属性！！！直接.来改 ▶box.width= ；▶style是 改样式！！！！！

可同时改多个样式

1. ▶Node. classList.add() 添加class属性
2. Node.classList.remove() 删除class属性
3. Node.classList.toggle() 切换class属性
4. ▶Node.className=’ ’加class属性/改整个class属性

操作样式CSS：

1. $(selector).css({"propertyname":"value","propertyname":"value"})
3. $('div').css('background-color':'red')
4. //css修改属性，如left，默认已带单位

加元素：

1. $(selector).append() - 在被选元素的结尾插入内容（或节点）另一种appendTo($("selector"))[内]
2. $(selector).prepend() -在被选元素的开头插入[内]
3. $(selector).after() - 在被选元素之后插入内容[外]
4. $(selector).before() - 在被选元素之前插入内容[外]
5. $(selector).remove() - 删除被选元素
6. $(selector).empty() - 从被选元素中删除子元素

加事件：

1. $("XXX").on("click", function(){......});
2. $("XXX").on("XX",”li”, function(){}); 事件委托
3. $("XXX").one("XX", function(){......});

显示隐藏：

☩jQ动画…

slideUp(null/speed , fn);

fadeIn(null/speed, fn);

animate({样式} , speed , fn)

clearQueue(true)清空未执行完的 动画列队

gotoEnd()跳转到末状态

1. $("XXX").hide();
2. $("XXX").show();

获取下拉框的值和文本：

1. var options=$("#XXX option:selected"); //获取选中的项
2. alert(options.val()); //拿到选中项的值
3. alert(options.text()); //拿到选中项的文本
4. Node.style.propertyname='XXXX'  获取内联样式
5. div.style.backgroundColor = 'red';
6. window.getComputedStyle(elemen) ▶获取css样式

只读

1. Node.appenChild()   在末尾添加子节点，可以是创建的新节点，也可以是已经存在的节点
2. Node.insertBefore(newChild,refChild)   在redChild节点前插入一个新节点
3. Node.insertAfter(newChild,refChild) 在指定的已有的子节点后插入新节点
4. Node.replaceChild(newChild,oldChild)  替换
5. Node.remove ()  删除节点

解绑：target.removeEventListener("click",fn, "捕获/冒泡")

1. dom.addEventListener("click", fn);
2. dom.onclick = function(){..} 解绑把值=null
3. ev.target—事件委托
4. dom.style.display = "none/block";//隐藏不占位
5. dom.style.visibility = "hidden/visible";//隐藏但占位
6. var myselect=document.getElementById("XX"); 拿到select对象：
8. var index=myselect.selectedIndex ; 拿到选中项的索引：selectedIndex代表的是你所选中项的index
10. myselect.options[index].value; 拿到选中项options的value：
12. myselect.options[index].text; 拿到选中项options的text

遍历：

1. $.each() 或arr.each()

DOM创建、克隆：

1. var Div=$("<div>我是DOM</div>")
2. $(selector).clone(true|false) 生成被选元素的副本,包含子节点,文本和属性,括号中的参数为是否要复制时间处理程序

------------------------------------------获取

获取节点：

1. $('selector').parent()  方法返回被选元素的直接父元素
3. $('selector').siblings()  方法返回被选元素的所有同胞元素。
4. $('selector').next()  方法返回被选元素的下一个同胞元素。(只返回一个)
5. $('selector').nextAll()  方法返回被选元素的之后所有的同胞元素
6. $('selector').prev()   方法返回被选元素的上一个同胞元素。(只返回一个)
7. $('selector').prevAll()  方法返回被选元素的之前所有的同胞元素
9. $('selector').children() 方法返回被选元素的所有直接子元素。
10. $('selector').find() 方法返回被选元素的后代元素，一路向下直到最后一个后代。(里面必须传入一个值)
11. end().返回上一级节点
12. forEach()  Array.form(arr)—转真数组遍历

Array.prototype.slice.call() ,

[].forEach.call()

1. Node.createElement("li") 创建一个html标记对象
2. Node.createTextNode('text') 创建一个文本节点
3. Node.cloneNode(true|false) ▶克隆选中的节点,克隆所有后代加true

附：var newImg = new Image() 创建img

-----------------------------------------

1. document.getElementsBy…    获取元素节点
2. document.querySelector (".class")
3. Node.parentNode  返回元素的父元素
5. Node.nextElementSibling 方法返回被选元素的下一个同胞元素。(Element在IE6-8中是underfine）
6. Node.previousElementSibling  方法返回被选元素的上一个同胞元素。
8. Node.children() 获取元素所有子元素
9. Node.firstElementChild()    获取第一个子节点
10. Node.lastElemntChild()  获取最后一个子节点
11. **★有兼容性写法**

JQ过滤节点：

1. $('selectors').first() 获取被选元素的首个元素。
2. $('selectors').last() 获取被选元素的最后一个元素。
3. $('selectors').eq() 获取被选元素中带有指定索引号的元素。
4. $('selectors:gt(n)')  获取被选元素中索引号大于n的所有元素。
5. $('selectors:lt(n)') 获取被选元素中索引号小于n的所有元素。
6. $('selectors:even')  获取被选元素中索引号为偶数的所有元素。
7. $('selectors:odd')  获取被选元素中索引号为奇数的所有元素。
8. $('selectors:empty')  获取被选元素中为空的所有元素。
9. $('selectors:contains(text))  获取被选元素中内容为...的所有元素。
10. $('selectors:not(selector))  获取被选元素不是。。的所有元素。
11. [attribute]、[attribute=value]、[attribute^=value]、[attribute$=value]、[attribute\*=value]
12. selector:checked/:disabled/:selected/:input/:text/:password/:checkbox

**获取尺寸位置：**

**1.Js获取浏览器高度和宽度**

document.documentElement.clientWidth ==> 浏览器可见区域宽度   
document.documentElement.clientHeight ==> 浏览器可见区域高度   
document.body.clientWidth ==> BODY对象宽度   
document.body.clientHeight ==> BODY对象高度

**Jq获取浏览器高度和宽度**

$(window).width() ==> 浏览器可见区域宽度  
$(window).height() ==> 浏览器可见区域高度   
$(document).height() ==> 页面文档的高度  
$(document.body).height() ==> BODY对象高度

第一种情况就是宽高都写在样式表里，就比如#div1{width:120px;}。这中情况通过#div1.style.width拿不到宽度，而通过#div1.offsetWidth才可以获取到宽度。

第二种情况就是宽和高是写在行内中，比如style="width:120px;"，这中情况通过上述2个方法都能拿到宽度。

**2.Js获取对象的高度和宽度**  
obj.clientWidth = width + padding   ==> 获得包括内边界（padding）的元素宽度  
obj.offsetHeight = height + padding + border ==> 获得包括内边界(padding)和边框(border)的元素高度

**Jq获取对象的高度和宽度**

obj.innerWidth() ==> 获得包括内边界（padding）的元素宽度，  
obj.outerWidth() ==> 获得包括内边界(padding)和边框(border)的元素宽度  
obj.outerWidth(true) ==> 获得包括外边界(margin)的元素宽度  
w同一个元素应该是：width()<=innerWidth()<=outerWidth()<=outerWidth(true);

**3.Js 获取对象的相对高度和宽度**

obj.offsetLeft ==> 元素相对于父元素的left  
obj.offsettop ==> 元素相对于父元素的top  
obj.scrollLeft ==> 设置或获取位于对象左端滚动的距离  
obj.scrollTop ==> 设置或获取位于对象顶端滚动的距离

▶Node.getBoundingClientRect()==>左上角、右下角相、xy，width/height[border-box]

**Jq 获取对象的相对高度和宽度**

obj.offset().left ==> 元素相对于文档的left  
obj.offset().top ==> 元素相对于文档的top  
obj.scrollLeft() ==> 设置或返回对象相对滚动条左侧的偏移。  
obj.scrollTop() ==> 设置或返回对象相对滚动条顶部的偏移。

图片轮播:改scrollleft 或 定位改left.

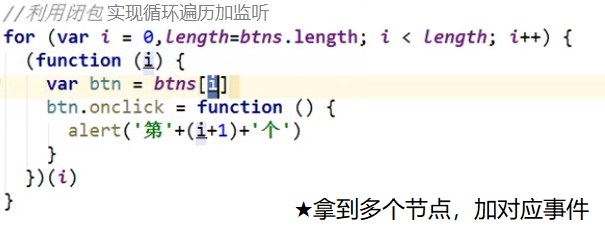
**•ECMAScript262**

解决方案：

1、添加this.index

2、使用es6的let语法

3、使用(function(i){})(i)形成闭包



闭包：函数套函数，是内部和外部沟通的桥梁。

闭包作用：1.可以读取到其他函数内部的变量。

2.可以将变量保存在内存中。

闭包优点：能避免使用全局变量，防止污染，使用全局变量很危险。

闭包缺点：闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题。

•Object

•遍历对象，i返回属性名 ==> for i in obj



••Function

this/call、apply、bind：

test.call(this, test1, test2)

test.apply(this, [test1, test2]) --可变形参

•立即可执行函数 ==> ( function(){} )()

•arguments用于保存全部实参-实参和argument共享

• callee属性，指向当前执行的函数

• caller 属性，指向调用当前函数的函数

• length属性，argument的长度

|  |
| --- |
| **function** fn() {  console.log(**this**.length) }; **var** length = 10; **var** obj = {  length:5,  method:**function** (fn) {  fn(); /\* 10 \*/  arguments[0](); /\* 2 \*/  } }; obj.method(fn,1); |

345

4555

•Array

1. Array.isArray(arr)  判断
2. arr.push() 后增
3. arr.unshift() 前增
4. arr.pop() 后删
5. arr.shift() 前删
6. arr.concat(arr1) 连接arr
7. arr.join($) 转字符串$间隔
8. arr.toString() 转字符串,间隔
9. arr.indexOf(e) e的下标
10. arr.lastIndexOf(e) e的下标
11. arr.forEach(function(item,index){}) 遍历
12. map(function(item,index){}) 遍历、返回新arr、批量处理(全部+1)
13. every(function(item,index){ 返true/false
14. if(item>0){
15. return true
16. }
17. })
18. some() 返回true/false
19. filter() 过滤、返回新数组
20. slice(1,4) 截取，左开右闭
21. splice(index,删个数，增) 删、增
22. sort(function(a,b){ 排序
23. return a-b
24. })
25. arr.reverse() 倒序

•Date

1. var date = new Date();
3. Date.now();       // 返回值：1970年1月1日午夜与当前日期和时间之间的毫秒数
4. date.getTime()
5. date.getFullYear()      // 年
6. date.getMonth()+1       // 月 （0 - 11）
7. date.getDate())         // 日 （1 - 31）
8. date.getHours()         // 小时 （0 - 23）
9. date.getMinutes()       // 分钟 （0 - 59）
10. date.getSeconds()       // 秒 （0 - 59）
11. date.getDay()           //星期（0-6）

•Math

1. Math.floor();       // 向下取整（下舍去）
2. Math.ceil();        // 向上取整（上进入）
3. Math.round();       // 四舍五入
4. Math.abs();         // 取绝对值
5. Math.cos();         // 余弦值
6. Math.sin();         // 正弦值
7. Math.max();         // 比较（取大）
8. Math.min();         // 比较（取小）
9. Math.pow(x,y);      //x\*\*y
10. Math.sqrt()         //开放
11. 弧度=角度\*Math.PI/180

•String(包装类)

1. indexOf("a",5)       //找a,从5开始
2. lastIndexOf("a",5)
3. toUpperCase()       //大写
4. toLowerCase()       //小写
5. trim()           //去前后空格
6. split(" ")          //按空格拆成数组
7. slice(1,a)       //截取,左闭右开

•Number(包装类)

1. isNaN      //判断
2. parseInt   //转整形
3. isInteger  //判断
4. parseFloat //转浮点

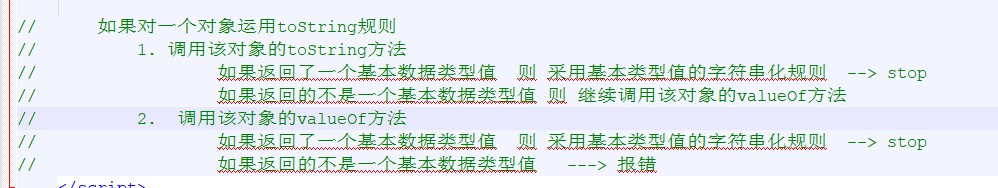
•强制类型转化

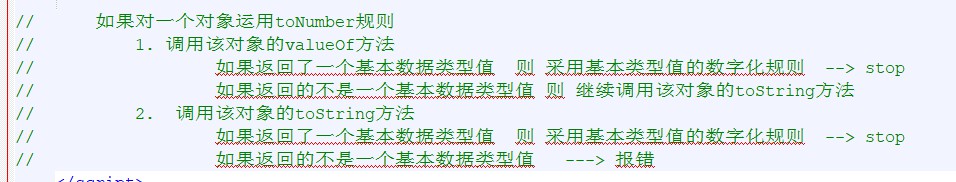
1. 转成string类型： +（字符串连接符）

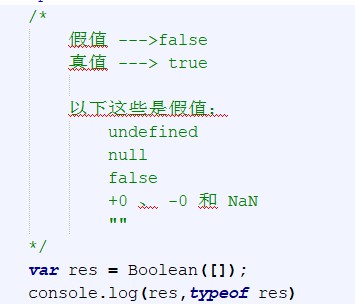
2.转成number类型：++/--(自增自减运算符) + - \* / %(算术运算符) > < >= <= == != === !=== (关系运算符)

3. 转成boolean类型：!（逻辑非运算符）

<https://blog.csdn.net/itcast_cn/article/details/82887895>







Object.prototype.toString---结果 “[object Objecct]”----> String类型

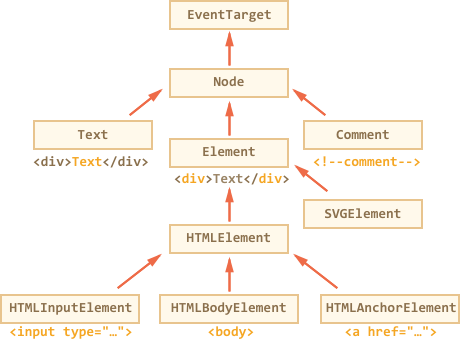
Array.prototype.toString ---结果 “1,2,3”

function () {}.tosring ==> ‘ function () {} ’



**•DOM**

•DOM树



•事件

▶console.log（对象）---需要看成一个异步代码

异步代码

•鼠标事件：

click 在元素上按下并释放任意鼠标按键

dblclick 在元素上双击鼠标按钮

mouseover 指针移到有事件监听的元素或者它的子元素内

mouseout 指针移出元素，或者移到它的子元素上

mousedown 在元素上按下任意鼠标按钮

mousemove 指针在元素内移动时持续触发

mouseup 在元素上释放任意鼠标按键

mouseenter 指针移到有事件监听的元素内（没有冒泡）

mouseleave 指针移出元素范围外（没有冒泡）

contextmenu 右键点击 (右键菜单显示前).

 •滚轮事件:

ie/chrome : onmousewheel(dom0)

event.wheelDelta

上：120

下：-120

firefox : DOMMouseScroll 必须用(dom2的标准模式)

event.detail

上：-3

下：3

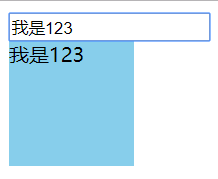
•响应事件（▶input和▶change）：

实时响应

失焦时才响应

|  |
| --- |
| $("input[type = text]").on("input",**function** () {  $(".box").html($("input[type = text]").val()) }); |

|  |
| --- |
| $("input[type = text]").on("change",**function** () {  $(".box").html($("input[type = text]").val()) }); |



 •事件对象:

clientX 鼠标点击位置在X轴上距离视口的距离

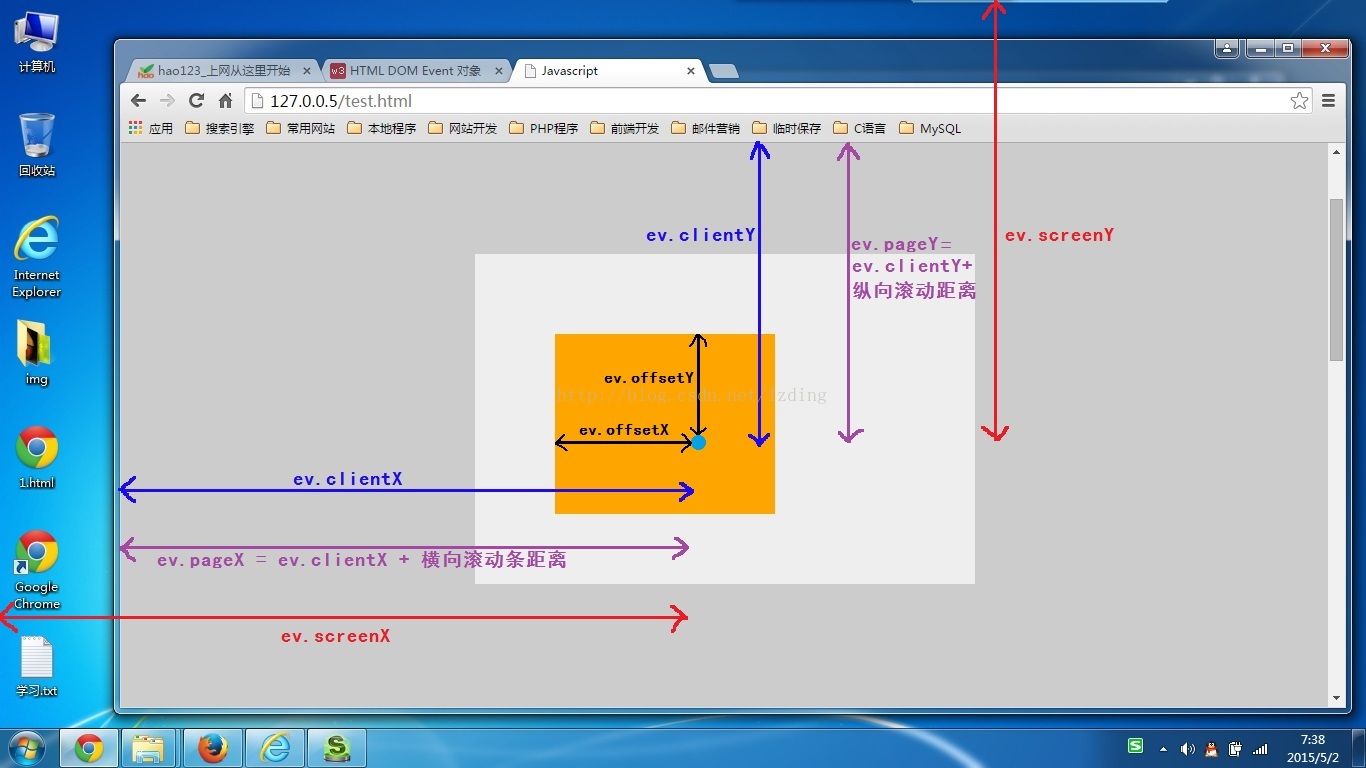
clientY 鼠标点击位置Y轴上距离视口的距离

preventDefault() 阻止事件的默认行为

stopPropagation() 阻止事件的传播

target 事件委托 **★有兼容性写法**

window.onload = **function**(){  
 **var** ul = document.querySelector("ul");  
 ul.onclick = **function**(ev){ //事件委托给父元素  
 ev = ev || event;  
 console.log(ev.target.innerHTML) //这个会打印对应内容  
 }  
}



/

？

•响应式缩放:

1. window.onresize = function(){}响应式缩放，缩放太小后会出现背景图不够大靠左显示，给content一个浅色的背景色
2. ctrl + "-"缩小
3. ctrl + "+"放大
4. ctrl + "0"正常

**•BOM**

**•判断浏览器** **★有兼容性写法**

控制台打印浏览器相关信息

|  |
| --- |
| window.navigator.userAgent.toLowerCase();//将浏览器信息获取，并转成小写 |

//判断是ie、火狐、chrome浏览器

|  |
| --- |
| **function** isBrowser(){  **var** agent=navigator.userAgent.toLowerCase();  console.log(agent);  **if**(agent.indexOf('chrome')>0){  alert("chrome浏览器")  }  **if**(agent.indexOf('firefox')>0){  alert("firefox浏览器")  }  **if**(agent.indexOf('trident')>0){  alert("IE浏览器")  } } isBrowser(); |

//判断是qq还是Chrome浏览器(上面代码qq,搜狗运行时结果是"Chrome浏览器")

|  |
| --- |
| **function** isBrowser(){  **var** agent=navigator.userAgent.toLowerCase();  console.log(agent);  System=**function**(){  **if**(agent.indexOf('qqbrowser')>0){//判断是qq浏览器还是其它浏览器  **return** alert("qq浏览器")  }  **if**(agent.indexOf("se 2.x")>0){  **return** alert("搜狗浏览器")  }  alert('chrome浏览器')  };  System();  **if**(agent.indexOf('firefox')>0){  alert("firefox浏览器")  }  **if**(agent.indexOf('trident')>0){  alert("IE浏览器")  } } isBrowser(); |

360浏览器奇葩

//application/vnd.chromium.remoting-viewer 可能为360特有 通过\_mine判断是否是360代码qq,搜狗运行时结果是"Chrome浏览器")

|  |
| --- |
| **function** isBrowser(){  **var** agent=navigator.userAgent.toLowerCase();  console.log(agent);  System=**function**(){  **if**(agent.indexOf('qqbrowser')>0){//判断是qq浏览器还是其它浏览器  **return** alert("qq浏览器")  }  **if**(agent.indexOf("se 2.x")>0){  **return** alert("搜狗浏览器")  }  **var** is360 = \_mime("type", "application/vnd.chromium.remoting-viewer");   **if** (is360) {  **return** alert("360浏览器")  }   //检测是否是谷歌内核(可排除360及谷歌以外的浏览器)  //测试mime  **function** \_mime(option, value) {  **var** mimeTypes = navigator.mimeTypes;  console.log(mimeTypes);  **for** (**var** mt **in** mimeTypes) {  **if** (mimeTypes[mt][option] == value) {  **return true**;  }  }  **return false**;  }  alert('chrome浏览器')  };  System();  **if**(agent.indexOf('firefox')>0){  alert("firefox浏览器")  }  **if**(agent.indexOf('trident')>0){  alert("IE浏览器")  } } isBrowser() |

//ie版本

|  |
| --- |
| **function** isIE(version){  **var** b = document.createElement("b");  b.innerHTML="<!--[if IE "+version+"]><i></i><![endif]-->";  **return** (b.getElementsByTagName("i").length === 1); } |

**☺代码AA：**

☞遮罩:

|  |
| --- |
| i{  position: absolute;  left: 0;  right: 0;  top: 0;  bottom: 0;  background: rgba(220,155,60,.6); } |

☞判断是否第一次打开页面:

//在刷新的时候，不但window不会被销毁，自定义在window对象上的属性也会被保留。

1. if(!window.name){
2. alert("第一次开这个窗口！name值"+ window.name);
3. window.name = 'lijia';
4. }else{
5. alert('刷新操作 name值：'+ window.name);
6. }

☞节流(起始)

1. //这里是throttle()而不是throttle，调用了一次，返回新的函数给onscroll事件
2. document.onscroll = throttle(function (e) {  //这里的e是形参，真正的实参是下面的argument
3. console.log('滚动了');
4. console.log(this);
5. console.log(e);
6. }, 1000);
8. function throttle(fn, time) {
9. let startTime = 0;
10. return function () {    //每次滚动都会返回一个函数检测时间
11. // console.log(arguments);
12. const endTime = Date.now();
13. if (endTime - startTime >= time) {
14. fn.apply(this,arguments);  //不改的话this指向了window
15. startTime = endTime;
16. }
17. }
18. }

☞防抖（结尾有效）

1. document.onscroll = debounce(function (e) {
2. console.log('运行了');
3. console.log(e);
4. console.log(this)
5. },200);
6. function debounce(fn, time) {
7. let timer = null;
8. return function () {
9. clearTimeout(timer);  //清的是上一次的定时器，清完之后继续运行下面代码
10. // timer = setTimeout(function () {  //普通函数需要把外面的this和argument在外层函数用变量存一下
11. //     fn.apply(this,arguments)
12. // },time)
13. timer = setTimeout(() => {
14. fn.apply(this,arguments)      //箭头函数this指向外层this，因为是document调用，所以就是document了
15. },time)
16. }
17. }

☞兼容性代码

事件对象:

1. box.onclick = function (ev) {
2. ev=ev||window.event;
3. var target = event.target || event.srcElement; /\*support IE\*/
4. };

获取滚动条高度:

1. var scrolltop = document.documentElement.scrollTop || document.body.scrollTop

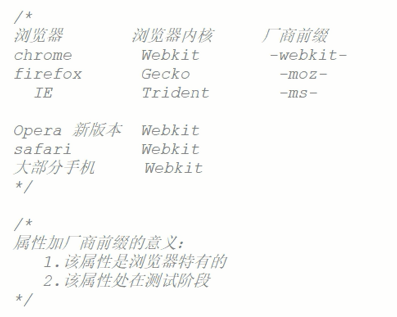
获取子元素:

1. var first = e.firstElementChild || e.firstChild /\*support IE\*/

▶阻止事件默认行为：

1. ev.preventDefault?ev.preventDefault():ev.returnValue = false; /\*support IE\*/

☞选中一个元素时排他法清样式或计算数量、用事件委托绑定事件(新元素) 、JS动画都会用到定时器(clear清定时器)setTimeout(function () {...},1000)/setInterval(function () {...},1000)、图片轮播用Node.onclick()模拟点击行为换图、一般能动的图都是“背景图片、



**☺写项目：**

☞代码优化:

•三元运算：T > 0? 0 : T; 规定T的范围，碰撞检测

•短路运行：box.onmousewheel = fn;

box.addElementListener && box.addElementListener(“DOMMouseScroll”,fn);

•取反实现切换：flag = !flag

•到一定值改变方向: i+=value，if(i>200){ value=-1 };if(i<0){ value+1 }

//单词和数字没有空格默认是一个单词

作者：Alex浪

来源：GitHib

邮箱：https://www. l738367@tom.com (小写L)

版权声明：本文件为作者原创文章，转载请附上出处！