**跨浏览器事件**

**跨浏览器添加事件**

|  |
| --- |
| //跨浏览器添加事件      function addEvent(obj,type,fn){          if(obj.addEventListener){              obj.addEventListener(type,fn,false);          }else if(obj.attachEvent){//IE              obj.attchEvent('on'+type,fn);          }      } |

**跨浏览器移除事件**

|  |
| --- |
| //跨浏览器移除事件  function removeEvent(obj,type,fn){      if(obj.removeEventListener){          obj.removeEventListener(type,fn,false);      }else if(obj.detachEvent){//兼容IE          obj.detachEvent('on'+type,fn);      }  } |

**跨浏览器阻止默认行为**

|  |
| --- |
| //跨浏览器阻止默认行为      function preDef(ev){          var e = ev || window.event;          if(e.preventDefault){              e.preventDefault();          }else{              e.returnValue =false;          }      } |

**跨浏览器获取目标对象**

|  |
| --- |
| //跨浏览器获取目标对象  function getTarget(ev){      if(ev.target){//w3c          return ev.target;      }else if(window.event.srcElement){//IE          return window.event.srcElement;      }  } |

**跨浏览器获取滚动条位置**

|  |
| --- |
| //跨浏览器获取滚动条位置，sp == scroll position      function getSP(){          return{              top: document.documentElement.scrollTop || document.body.scrollTop,              left : document.documentElement.scrollLeft || document.body.scrollLeft;          }      } |

**跨浏览器获取可视窗口大小**

|  |
| --- |
| //跨浏览器获取可视窗口大小            function  getWindow () {              if(typeof window.innerWidth !='undefined') {                  return{                      width : window.innerWidth,                      height : window.innerHeight                  }                } else{                  return {                      width : document.documentElement.clientWidth,                      height : document.documentElement.clientHeight                  }              }          }, |

**js 对象冒充**

|  |
| --- |
| <script type="text/javascript">        function Person(name , age){          this.name = name ;          this.age = age ;          this.say = function (){              return "name : "+ this.name + " age: "+this.age ;          } ;      }        var o = new Object() ;//可以简化为Object()      Person.call(o , "zhangsan" , 20) ;      console.log(o.say() );//name : zhangsan age: 20    </script> |

**js 异步加载和同步加载**

异步加载也叫非阻塞模式加载，浏览器在下载js的同时，同时还会执行后续的页面处理。  
在script标签内，用js创建一个script元素并插入到document中，这种就是异步加载js文件了：

|  |
| --- |
| (function() {      var s = document.createElement('script');      s.type = 'text/javascript';      s.async = true;      s.src = '<http://yourdomain.com/script.js>';      var x = document.getElementsByTagName('script')[0];       x.parentNode.insertBefore(s, x);  })(); |

**同步加载**

平常默认用的都是同步加载。如：

|  |
| --- |
| <script src="<http://yourdomain.com/script.js>"></script> |

同步模式又称阻塞模式，会阻止流览器的后续处理。停止了后续的文件的解析，执行，如图像的渲染。浏览器之所以会采用同步模式，是因为加载的js文件中有对dom的操作，重定向，输出document等默认行为，所以同步才是最安全的。  
通常会把要加载的js放到body结束标签之前，使得js可在页面最后加载，尽量减少阻塞页面的渲染。这样可以先让页面显示出来。

同步加载流程是瀑布模型，异步加载流程是并发模型。

**js获取屏幕坐标**

|  |
| --- |
| 获取鼠标坐标      <script type="text/javascript">      function mousePosition(ev){          if(ev.pageX || ev.pageY){              return {x:ev.pageX, y:ev.pageY};          }          return {              x:ev.clientX + document.body.scrollLeft - document.body.clientLeft,              y:ev.clientY + document.body.scrollTop - document.body.clientTop          };      }      function mouseMove(ev){          ev = ev || window.event;          var mousePos = mousePosition(ev);          document.getElementById('xxx').value = mousePos.x;          document.getElementById('yyy').value = mousePos.y;      }      document.onmousemove = mouseMove;  </script>  X:<input id="xxx" type="text"> Y:<input id="yyy" type="text"> |

注释：  
1.documentElement 属性可返回文档的根节点。  
2.scrollTop() 为滚动条向下移动的距离  
3.document.documentElement.scrollTop 指的是滚动条的垂直坐标  
4.document.documentElement.clientHeight 指的是浏览器可见区域高度

DTD已声明的情况下：

|  |
| --- |
|  |

如果在页面中添加这行标记的话

**IE**

|  |
| --- |
| document.body.clientWidth ==> BODY对象宽度  document.body.clientHeight ==> BODY对象高度  document.documentElement.clientWidth ==> 可见区域宽度  document.documentElement.clientHeight ==> 可见区域高度 |

**Firefox**

|  |
| --- |
| document.documentElement.scrollHeight ==> 浏览器所有内容高度  document.body.scrollHeight ==> 浏览器所有内容高度  document.documentElement.scrollTop ==> 浏览器滚动部分高度  document.body.scrollTop ==>始终为0  document.documentElement.clientHeight ==>浏览器可视部分高度  document.body.clientHeight ==> 浏览器所有内容高度 |

**Chrome**

|  |
| --- |
| document.documentElement.scrollHeight ==> 浏览器所有内容高度  document.body.scrollHeight ==> 浏览器所有内容高度  document.documentElement.scrollTop==> 始终为0  document.body.scrollTop==>浏览器滚动部分高度  document.documentElement.clientHeight ==> 浏览器可视部分高度  document.body.clientHeight ==> 浏览器所有内容高度 |

浏览器所有内容高度即浏览器整个框架的高度，包括滚动条卷去部分+可视部分+底部隐藏部分的高度总和

浏览器滚动部分高度即滚动条卷去部分高度即可视顶端距离整个对象顶端的高度。

综上

1、document.documentElement.scrollTop和document.body.scrollTop始终有一个为0，所以可以用这两个的和来求scrollTop

2、scrollHeight、clientHeight 在DTD已声明的情况下用documentElement，未声明的情况下用body

|  |
| --- |
| clientHeight   在IE和FF下，该属性没什么差别，都是指浏览器的可视区域，即除去浏览器的那些工具栏状态栏剩下的页面展示空间的高度。 |

**PageX和clientX**

PageX:鼠标在页面上的位置,从页面左上角开始,即是以页面为参考点,不随滑动条移动而变化

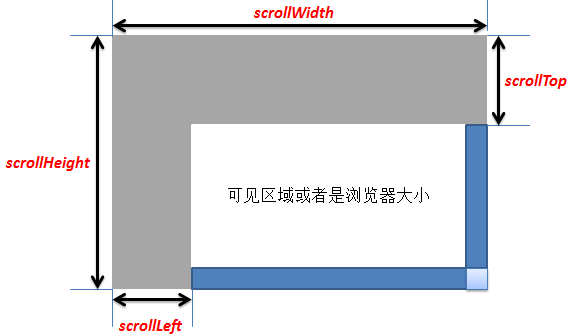
clientX:鼠标在页面上可视区域的位置,从浏览器可视区域左上角开始,即是以浏览器滑动条此刻的滑动到的位置为参考点,随滑动条移动 而变化.

可是悲剧的是,PageX只有FF特有,IE则没有这个，所以在IE下使用这个：

PageY=clientY+scrollTop-clientTop;(只讨论Y轴,X轴同理,下同)

scrollTop代表的是被浏览器滑动条滚过的长度

offsetX:IE特有,鼠标相比较于触发事件的元素的位置,以元素盒子模型的内容区域的左上角为参考点,如果有boder`,可能出现负值

只有clientX和screenX 皆大欢喜是W3C标准.其他的,都纠结了.  
最给力的是，chrome和safari一条龙通杀!完全支持所有属性  


**js拖拽效果**

|  |
| --- |
| <!doctype html>      <style type="text/css">          #login{              height: 100px;              width: 100px;              border: 1px solid black;              position: relative;              top:200px;              left: 200px;              background: red;          }      </style> |

offsetTop 返回的是数字，而 style.top 返回的是字符串，除了数字外还带有单位：px。

**js获取图片原始大小尺寸**

|  |
| --- |
| var img = $("#img\_id"); // Get my img elem  var pic\_real\_width, pic\_real\_height;  $("<img/>") // Make in memory copy of image to avoid css issues      .attr("src", $(img).attr("src"))      .load(function() {          pic\_real\_width = this.width;   // Note: $(this).width() will not          pic\_real\_height = this.height; // work for in memory images.      }); |

**js循环遍历数组**

|  |
| --- |
| <script>         //循环遍历数组         var animals = ["cat",'dog','human','whale','seal'];         var animalString = "";         for(var i = 0;i<animals.length;i++){             animalString += animals[i] + " ";         }         alert(animalString);  //输出数组里的每个项  </script> |

**遍历二维数组**

|  |
| --- |
| <script>   var arr=[[0,0,0,0,0,0],[0,0,1,0,0,0],[0,2,0,3,0,0],[0,0,0,0,0,0]];   for(var i=0;i<arr.length;i++){   //遍历每一个具体的值   for(var j=0;j<arr[i].length;j++){   document.writeln(arr[i][j]+" ");   }   document.writeln("<br/>");   }   </script> |

**阻止表单重复提交**

有两种方法可以解决：一是提交之后，立刻禁用点击按钮；第二种就是提交之后取消后续的表单提交操作。

|  |
| --- |
| document.getElementById("btn").disabled = true;//第一次提交后，将按钮禁用 |

这种方式只能用于通过提交按钮防止重复提交，还可以使用如下方式：

|  |
| --- |
| var flag = false;//设置一个监听变量  if(flag ==true)return;//退出事件  flag = true;//表示提交过一次了 |

**符串部分**

**在字符串中查找子字符串**

|  |
| --- |
| <script type="text/javascript">      var test = 'Welcome to my blog!';      var value = 'blog';      var subValue = test.indexOf(value);      console.log(subValue);//14,子字符串的索引  </script> |

**Number和Math部分**

数字可以是一个直接量，也可以是一个对象，但是Math对象不同，他没有构造函数，并且其所有的属性和方法都是直接通过这个对象来访问的

**把十进制转化为一个十六进制值**

|  |
| --- |
| var num = 255;  console.log(num.toString(16));//ff |

js中，十进制数字以0x开头，八进制数字总是以0开头

**随进产生颜色**

|  |
| --- |
| <script type="text/javascript">      function randomVal(val){          return Math.floor(Math.random()\*(val + 1));      }        function randomColor(){          return 'rgb(' + randomVal(255) + ',' + randomVal(255) + ',' + randomVal(255) + ')';      }  </script> |

目前，所有浏览器都支持RGB表示法和十六进制表示法，除了IE7，它只支持十六进制表示法

**在角度和弧度之间转换**

|  |
| --- |
| var rad = degrees\*(Math.PI/180);    var degrees = rad\*(180/Math.PI); |

**数组部分**

**创建多维数组**

|  |
| --- |
| <script type="text/javascript">      var arrayLength = 3;//设置数组长度        //创建数组      var multiArray = new Array(arrayLength);      for(var i =0;i<multiArray.length;i++){          multiArray[i] = new Array(arrayLength);      }        //给第一个数组索引添加项      multiArray[0][0] = 'phone';      multiArray[0][1] = 'book';      multiArray[0][2] = 'TV';        //第二个      multiArray[1][0] = 2;      multiArray[1][1] = 1;      multiArray[1][2] = 98;        //第三个      multiArray[2][0] = ['java','python'];      multiArray[2][1] = ['js','C++'];      multiArray[2][2] = ['Haskell','php'];  </script> |

**排序数组**

|  |
| --- |
| <script type="text/javascript">       var fruits = ['banana','apple','orange','strawberry'];      console.log(fruits.sort());//Array [ "apple", "banana", "orange", "strawberry" ]        var num = [32,43,2,5,-23,0,4];      console.log(num.sort());//Array [ -23, 0, 2, 32, 4, 43, 5 ]  </script> |

Array对象的sort方法会按照字母顺序来排序数组元素。对于数字，是按照字符编码的顺序进行排序

|  |
| --- |
| function compare(a,b){      return a-b;  }  var num = [32,43,2,5,-23,0,4];  console.log(num.sort(compare));//Array [ -23, 0, 2, 4, 5, 32, 43 ] |

**Date日期时间部分**

**js计算时间差**

|  |
| --- |
| var date1=new Date();  //开始时间，当前时间    var date2=new Date(); //结束时间，需传入时间参数  var date3=date2.getTime()-date1.getTime();  //时间差的毫秒数    //计算出相差天数  var days=Math.floor(date3/(24\*3600\*1000));    //计算出小时数  var leave1=date3%(24\*3600\*1000);    //计算天数后剩余的毫秒数  var hours=Math.floor(leave1/(3600\*1000));  //计算相差分钟数  var leave2=leave1%(3600\*1000);        //计算小时数后剩余的毫秒数  var minutes=Math.floor(leave2/(60\*1000));      //计算相差秒数  var leave3=leave2%(60\*1000);      //计算分钟数后剩余的毫秒数  var seconds=Math.round(leave3/1000);    console.log(" 相差 "+days+"天 "+hours+"小时 "+minutes+" 分钟"+seconds+" 秒"); |

**正则部分**

**js实现千分位分隔**

|  |
| --- |
| <script type="text/javascript">      function cc(s){          if(/[^0-9\.]/.test(s)) return "invalid value";          s=s.replace(/^(\d\*)$/,"$1.");          s=(s+"00").replace(/(\d\*\.\d\d)\d\*/,"$1");          s=s.replace(".",",");          var re=/(\d)(\d{3},)/;          while(re.test(s))              s=s.replace(re,"$1,$2");          s=s.replace(/,(\d\d)$/,".$1");          return "￥" + s.replace(/^\./,"0.")      }  </script>  <input type="text"> |

**js判断传入参数是否为质数**

|  |
| --- |
| function fn(input) {    input = parseInt(input,10);    return isPrime(input) ? 'is prime' : 'not prime';  }    function isPrime(input) {    if (input < 2) {      return false;    } else {      for (var i = 2; i <= Math.sqrt(input); i++) {        if (input % i == 0) {          return false;        }      }    }    return true;  } |

**js判断字符串出现最多的字符，并统计次数**

|  |
| --- |
| //js实现一个函数，来判断一个字符串出现次数最多的字符，并统计这个次数        function countStr(str){          var obj = {};          for(var i = 0, l = str.length,k; i < l ;i++){ k = str.charAt(i); if(obj[k]){ obj[k]++; }else{ obj[k] = 1; } } var m = 0,i=null; for(var k in obj){ if(obj[k] > m){                  m = obj[k];                  i = k;              }          }          return i + ':' + m;      } |

本人兼职前端付费技术顾问，如需帮助请加本人微信hawx1993，非诚勿扰  
1.为初学前端而不知道怎么做项目的你指导  
2.指导并扎实你的JavaScript基础  
3.帮你准备面试并提供相关指导性意见  
4.为你的前端之路提供极具建设性的意见  
5.让你的学习更有效率，对知识点的理解更加透彻  
6.不解答可以百度到的答案，也不解答你遇到的bug

付费（每月仅需99）能让我更有耐心和责任心的为你指导，让双方产生更良好的协作

原文地址：<https://segmentfault.com/a/1190000002559158>