## 入门

7.24-Monday

1. js无法操作放大缩小和书签。
2. <a href = “javascript:alert('本来该跳转到页面变为了执行alert')" />
3. script可以放在页面任何地方。
4. css外联 link href
5. js外 联 script src
6. js执行顺序从上到下，从左到右。
7. 获取元素
   1. getElementById();
   2. getElementsByTagName();
8. 代码调试
   1. console.log(element);//打印元素
   2. console.dir(element);//元素的所有属性
9. 注意script的放置位置（会导致获取不到元素）。
10. 变量名的规则：
    1. 变量的区分大小写
    2. 允许包含字母、数字、$、\_
    3. 但是第一个字符不可以是数字，不允许包含空格和其他标点符号。
    4. 不能是关键字和保留字
    5. 语义化
11. document.write()
    1. 使用方面
       1. 页面载入过程中，用脚本加入新的页面内容。
       2. 用延时脚本创建本窗口或者新窗口的内容。
    2. 参数一个：字符串参数，
    3. 在载入页面后，浏览器输出流自动关闭。在此之后，任何一个对当前页面进行操作的document.write()方法将打开一个新的输出流，它将清除当前页面内容。因此，如果希望用脚本生成的HTML页面替换当前页面，就必须把HTML内容连接起来赋给一个变量，使用一个doucument.write()方法完成写操作。
    4. document.close();关闭输出流。
    5. 在延时脚本的最后一个document.write()方法后面，必须确保含有document.close()方法，不这样做就不能显示图像和表单。
    6. 并且，任何后面调用的document.write()方法会清除现有内容来写入新值。
12. ctrl+r和f5都是普通刷新，ctrl+f5是强制刷新。
13. 执行代码后才加分号，方法后不加。
14. console
    1. console.error,console.info,console.warn;
    2. console对象的上面的5种方法，都可以使用pirntf()风格的占位符。只支持（%s）、整数（%d或%i）、浮点数（%f）和对象(%o)四种。
    3. 分组显示：

console.group(“第一组：")

console.end();

* 1. console.dir()可以显示一个对象的所有的属性和方法。

1. box.style.height = '';
2. js属性操作修改的是行间样式，别的文件获取不到（如外部的css）。
3. js获取不到自定义的属性。以document.style.index和document.style['index']这种方式获取不到。用js在设置一下，比如box.index = 1;就可以获取到了。
4. 常见的属性： id,class,value,style,innerHTML,src,href,innerTEXT(不建议使用)
5. 国际化的命名方式
   1. 对象命名用大驼峰
   2. 常量全大写
   3. 变量用小驼峰
6. 查看属性的方式
   1. box.style.width;
   2. box.style['width'];
7. console.log(imgEye['src']);//获取到的是本地服务器的绝对路径
8. js修改的是内存中的内容。
9. 获取input的输入内容 value();
10. 给一个元素设置某个属性的属性值

box.style[attr] = val;

1. 解决页面拖动到右边有空白的问题。
   1. position:fixed;脱离所有的文档流。
   2. 隐藏html滚动条显示body滚动条

**html**{  
 **width**:100%;  
 **height**:100%;  
 **overflow**: **hidden**;  
}  
**body**{  
 **width**:100%;  
 **height**:100%;  
 **overflow**: **auto**;

}

兼容IE6时这样使用

1. 居中方法：

position: absolute;

top:0px;

left:0px;

bottom: 0px;

right:0;

margin:auto;

1. select使用

<select name="" id="select1">

<option value="北京">北京</option>

<option value="上海">上海</option>

</select>

### DOM模型概述

1. 基本概念
   1. DOM
      1. DOM是JavaScript操作网页的接口，全称为“文档对象模型"（Document Object Model）。
      2. 它的作用是将网页转为一个JavaScript对象，从而可以用脚本进行各种操作。
      3. JavaScript是最常用的操作DOM的语言。
   2. 节点
      1. **节点类型。**

Document：整个文档树的顶层节点

DocumentType：doctype标签（比如<!DOCTYPE html>）

Element：网页的各种HTML标签（比如<body>、<a>等）

Attribute：网页元素的属性（比如class="right"）

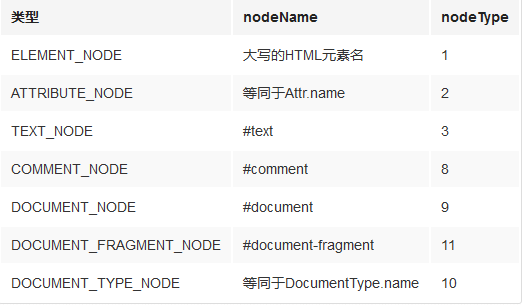
Text：标签之间或标签包含的文本

Comment：注释

DocumentFragment：文档的片段

* 1. 节点树
     1. 一个文档的所有节点，按照所在的层级，可以抽象成一种树状结构。这种树状结构就是DOM
     2. 最顶层的节点就是document节点，它代表了整个文档。文档最高一层的HTML标签，一般是<html>，它构成树结构的根节点，其他HTML标签节点都是它的下级。
     3. 除了根节点，其他节点对周围的节点都存在三种关系。
        1. 父节点关系（parentNode）：直接的那个上级节点。
        2. 子节点关系（childNodes）：直接的下级节点。（firstChild,lastChild）
        3. 同级节点关系（sibing）：拥有同一个父节点的节点。(nextSibling,previousSibing)。

1. 特征相关的属性
   1. Node.nodeName,Node.nodeType
      1. nodeName属性返回节点的名称，nodeType属性返回节点的类型的常数值。



* + 1. 使用nodeType来确认一个节点的类型（常用的元素节点-1，属性节点-2，文本节点-3）
  1. Node.nodeValue
     1. 返回一个字符串 ，表示节点本身的文本值，该属性可读写。
     2. 只有Text节点，Comment节点，XML文档的CDATA节点有文本值，其他节点返回null
  2. Node.textContent
     1. 返回当前节点和所有后代节点的文本内容。（自动忽略标签），可读写。
  3. Node.baseURI
     1. Node.baseURI属性返回一个字符串，表示当前网页的绝对路径。如果无法取到这个值，则返回null。如果无法取到这个值，则返回null。
     2. 浏览器根据这个属性，计算网页上的相对路径URL。
     3. 不同节点都可以调用这个属性（比如document.baseURI和element.baseURI），通常它们的值是相同的。
     4. window.location属性设置当前网址的URL，可以使用<base>标签来改变属性的值。

1. 相关节点的属性
   1. Node.ownerDocument
      1. Node.ownerDocument返回当前节点所在的顶层文档对象，即document对象。
      2. document对象本身的ownerDocument属性，返回null
   2. Node.nextSibling
      1. 返回当前节点后面的第一个同级节点。（包括文本节点和评论节点）
   3. Node.previousSibing
      1. 返回当前节点的前面距离最近的一个同级节点。如果当前节点前面没有同级节点，则返回null。
   4. Node.parentNode
      1. 该属性返回当前节点的父节点。对于一个节点来说，它的父节点只可能是三种类型：element节点、document节点和documentfragment节点。
      2. 下面的代码是如何从父节点移除指定节点。
      3. 对于刚生成的还没插入DOM树的节点它的父节点也是null
   5. Node.parentElement
      1. parentElement属性返回当前节点的父Element节点。如果当前节点没有父元素，或者父元素类型不是Element节点，则返回null。
   6. Node.childNodes
      1. childNodes属性返回一个NodeList集合，成员包括当前节点的所有子节点。
      2. 注意，除了HTML元素节点，该属性返回的还包括Text节点和Comment节点。
      3. 如果当前节点不包括任何节点则返回一个空的NodeList集合，
   7. Node.firstChild,Node.lastChild
      1. firstChild属性返回当前节点的第一个子节点，如果当前节点没有子节点，则返回null
      2. lastChild属性会返回当前节点的最后一个子节点，如果当前节点没有子节点则返回null。
2. 节点对象的方法
   1. Node.appendChild()
      1. 将参数节点作为最后一个子节点插入当前节点。如果参数节点已经存在，就将其移动到新位置。
   2. Node.hasChildNodes()
      1. 用布尔值表示当前节点是否有子节点。
   3. Node.cloneNode()
      1. 方法用于克隆一个节点。
      2. 它接受一个布尔值作为参数，表示是否同时被克隆子节点，默认是false，即不克隆子节点。
   4. Node.insertBefore()
      1. 方法用于将某个节点插入当前节点的指定位置。它接受两个参数，第一个参数是所要插入的节点，第二个参数是当前节点的一个子节点，新的节点将插在这个节点的前面。该方法返回被插入的新节点。
      2. 如果insertBefore方法的第二个参数为null，则新节点将插在当前节点的最后位置，即变成最后一个子节点
   5. Node.removeChild()
      1. 这个方法是在父节点上调用的，不是在被移除的节点上调用的。
      2. eg: divA.parentNode.removeChild(divA);
      3. 一个节点被移出之后，依然可以使用它，比如插入到另一个节点下面。
   6. Node.replaceChild()
      1. 第一个参数是用来替换的新节点，第二个参数是将要被替换走的子节点。
      2. 它返回被替换走的那个节点。
   7. Node.contains()
      1. 接收一个节点作为参数，返回一个布尔值，表示参数节点是否是当前节点的后代节点。
   8. Node.compareDocumentPosition()
      1. 返回一个7个比特位的二进制值，表示参数节点与当前节点的关系。
      2. 
      3. 属性节点集合，obj.attributes
   9. Node.isEqualNode()
      1. 用于检查两个节点是否相等。所谓相等的节点，指的是两个节点的类型相同、属性相同、子节点相同。
   10. Node.normalize()
       1. normailize方法用于清理当前节点内部的所有Text节点。它会去除空的文本节点，并且将毗邻的文本节点合并成一个。
       2. Node.childNodes,documet.queySelectorAll()返回的都是NodeList实例对象。
       3. NodeList实例对象可能是静态集合也可能是动态集合
       4. parent.childNodes返回的就是一个动态集合。
       5. document.querSelecor()返回的就是一个静态集合。
       6. NodeList不是数组，不能使用pop和push之类的数组特有的方法。
       7. NodeList继承链，myNodeList🡪NodeList.prototype🡪Object.prototype🡪null
       8. NodeList转换为数组的方法
          1. Array.prototype.slice.call(div\_list);
       9. 不要使用for…in循环遍历NodeList实例对象，因为for…in循环会将非数字索引的length属性和下面要讲到的item方法，也遍历进去，而且不保证各个成员的顺序。
       10. ES6新增的for of 循环可以正确的遍历NodeList实例对象。
       11. NodeList实例对象的item方法，接受一个数字索引作为参数，返回该索引对应的成员。如果取不到成员，或者索引不合法，则返回null。
   11. HTMLCollection对象
       1. document.links,document.forms,document.images返回的都是HTMLCollections对象。
       2. HTMLCollection实例对象成员只能是Element节点，NodeList实例对象的成员可以包含其他节点。
       3. HTMLCollection实例对象都是动态集合，节点的变化会实力反映在集合中。
       4. HTMLCollection实例对象可以用id属性或name属性来引用节点，NodeList只能使用数字索引引用。
       5. HTMLCollection实例的item方法，可以根据成员的位置参数（从0开始），返回该成员。如果取不到成员或数字索引不合法，则返回undefined。
       6. 由于item方法和namedItem方法，都可以用方括号运算符代替，所以建议一律使用方括号运算符。
3. ParentNode接口，ChlidNode接口
   1. ParentNode接口
      1. Element节点、Document节点和DocumentFragment节点，部署了ParentNode接口。都具有以下四个属性
      2. children
         1. children返回一个动态的HTMLCollection集合，由当前节点的所有Element子节点组成。
      3. firstElementChild
         1. firstElementChild属性返回当前节点的第一个Elment子节点。如果不存在返回null
      4. lastElementChild
      5. childElmentCount
         1. 返回当前节点的所有Element子节点的数目。
   2. ChildNode接口
      1. Element节点、DocumentType节点和CharacterData接口，部署了ChildNode接口。
      2. ChildNode接口用于处理子节点（不限于Element子节点）。
      3. remove()
         1. eg: el.remove();
            1. 这个方法用于移除节点本身，而不是它的父节点。
      4. before(),在当前节点前插入一个同级节点。如果是元素节点就插入元素节点本身，如果是文本就插入文本节点。
      5. after(),类似于before()，只不过是在当前节点后面。
      6. replaceWith()类似于before只不过是替换当前节点。

### string

1. nth-of-type(n)//选择器匹配属于父元素的特定类型的第N个子元素的每个元素
2. nth-child() 选择器，该选择器选取父元素的第 N 个子元素，与类型无关
3. nth-of-type(odd)// 操作奇数序号的元素
4. nth-last-of-type(odd)//操作偶数序号的元素
5. console.log(1+1+'2');//22
6. element = document.querySelector(selectors);
   1. 返回文档中匹配指定的选择器组的第一个元素(使用深度优先先序遍历文档的节点 | 并且通过文档标记中的第一个元素，并按照子节点数量的顺序迭代顺序节点)。
   2. element 是一个 element 对象（DOM 元素）。
   3. selectors 是一个字符串，包含一个或是多个 CSS 选择器 ，多个则以逗号分隔。
7. 换行方式
   1. **word-break**: **break-all**;//只对英文起作用，以字母作为换行依据。
   2. word-wrap:break-word;//只对英文起作用，以单词作为换行依据
   3. white-space：pre-wrap;//只对中文起作用，强制换行
   4. white-space:nowrap;//强制不换行，都起作用
   5. white-sapce:nowrap;overflow:hidden;text-overflow:ellipsis;不换行，超出部分隐藏且以省略号形式出现（部分浏览器支持）
8. 消息小三角实现：
   1. 右

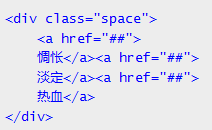
.**r**:**after**{  
 **content**:**" "**;  
 **position**: **absolute**;  
 **border**:10**px solid transparent**;  
 **border-left-color**: **#E1E1E1**;  
 **right**: -20**px**;  
 **top**:10**px**;  
}

* 1. 带边框的小三角(上)

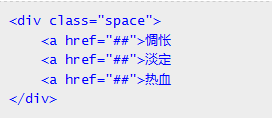
.**bubble-arrow-border**{  
 **left**: 64%;  
 **border**: **solid transparent**;  
 **border-bottom-color**: **#C8CCD3**;  
 **bottom**: 100%;  
 **border-width**: 9**px** 8**px**;  
 **position**: **absolute**;  
 **width**: 0;  
 **height**: 0;  
}

.**bubble-arrow**{  
 **display**: **block**;  
 **border**: **solid transparent**;  
 **border-bottom-color**: **#fff**;  
 **left**: 64%;  
 **bottom**: 100%;  
 **border-width**: 8**px** 7**px**;  
 **margin-left**: 1**px**;  
 **position**: **absolute**;  
 **width**: 0;  
 **height**: 0;  
}

1. [去除inline-block元素间间距](http://www.zhangxinxu.com/wordpress/2012/04/inline-block-space-remove-%E5%8E%BB%E9%99%A4%E9%97%B4%E8%B7%9D/)
   1. 原因：元素之间间距出现的原因就是标签段之间的空白，因此去掉HTML中的空格，自然就是没有了。

* 1. 使用margin负值。
  2. 去掉闭合标签



* 1. 使用font-size:0
  2. 使用letter-spacing（推荐）
  3. 使用word-spacing
  4. 成品方法
     1. 针对block

.yui3-g {

letter-spacing: -0.31em; /\* webkit \*/

\*letter-spacing: normal; /\* IE < 8 重置 \*/

word-spacing: -0.43em; /\* IE < 8 && gecko \*/

}

.yui3-u {

display: inline-block;

zoom: 1; \*display: inline; /\* IE < 8: 伪造 inline-block \*/

letter-spacing: normal;

word-spacing: normal;

vertical-align: top;

}

li {

display:inline-block;

background: orange;

padding:10px;

word-spacing:0;

}

ul {

width:100%;

display:table; /\* 调教webkit\*/

word-spacing:-1em;

}

.nav li { \*display:inline;}

1. 用ID加类布局
2. 基本不用onload一般将script标签放到</body>之前
   1. 这么做有两个好处：
      1. 不需要写onload因为在此时DOM元素已经初始化完毕
      2. 优先加载样式和HTML结构防止脚本加载慢时导致页面显示空白
   2. 需要在页面加载前执行脚本写入onload
3. script标签放到哪里比较合适
   1. 放入head标签然后使用 defer

eg:<script src="script1.js" defer></script>

1. document.querySelector('.d1');//css选择器获取元素 获取第一个
2. document.querySelector('.d1');//css选择器获取元素 获取第一个
3. z-index必须设置一下定位，否则会不起效果。
4. 背景渐变
   1. 渐变是设置在background-image上！！！
   2. linear-gradient(角度, 颜色 位置, 颜色 位置)
   3. repeating-linear-gradient()。
   4. 从左上角到右下角
      1. background-image:linear-gradient(to right bottom, royalblue, #DD5044);
      2. 注意：用空格隔开right和bottom
5. 禁止文本选中 文本选择

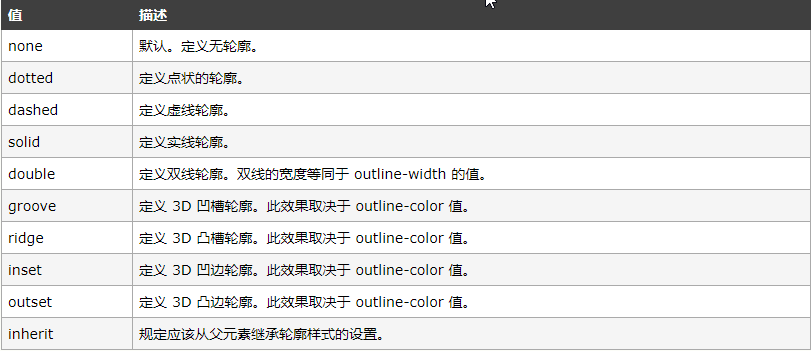
**body**{  
 **-webkit-user-select**: **none**;  
 **-moz-user-select**: **none**;  
 **-ms-user-select**: **none**;  
 **user-select**: **none**;  
}

1. 伪类的颜色被js修改后只需再次置空就可以了eg: this.style.backgroundColor = "";
2. transform: rotate(-70deg);//旋转 默认角度是从下至上
3. 大图不要超过500kb，小图不要超过20kb
4. outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。（比如input周围的篮框）。

可以按顺序设置如下属性：

1. outline-color
2. outline-style
3. outline-width

outline-style的值



1. 居中方式 父级加一个text-align 本身inline-block
2. 用连等来给两个不同元素执行相同函数。
3. 取绝对值Math.abs(number);
4. 有border可以修复父子margin重复问题
5. 取模和取余不一样

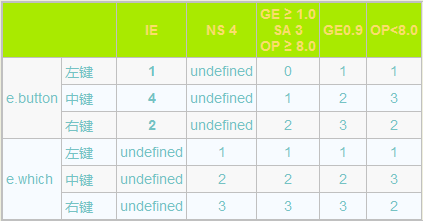
### [鼠标事件总结](http://www.cnblogs.com/rubylouvre/archive/2009/08/24/1552862.html)

1. 常见鼠标事件
2. E.onclick 鼠标点下去并且抬起来之后的一瞬间
3. E.dbclick 鼠标的按钮被按下两次
4. E.contextmenu弹出右键菜单
5. E.mousedown鼠标的按钮被按下
6. E.mouseup鼠标的按钮被释放弹起
7. E.onmouseover 鼠标指针移入元素的一瞬间
8. E.onmouseout 鼠标指针移出元素的一瞬间
9. E.onmousemove 鼠标在目标的上方移动。

# 鼠标事件

DOM3级事件中定义了9个鼠标事件。

* mousedown:鼠标按钮被按下（左键或者右键）时触发。不能通过键盘触发。
* mouseup:鼠标按钮被释放弹起时触发。不能通过键盘触发。
* click:单击鼠标**左键**或者按下回车键时触发。这点对确保易访问性很重要，意味着onclick事件处理程序既可以通过键盘也可以通过鼠标执行。
* dblclick:双击鼠标**左键**时触发。
* mouseover:鼠标移入目标元素上方。鼠标移到其后代元素上时会触发。
* mouseout:鼠标移出目标元素上方。
* mouseenter:鼠标移入元素范围内触发，**该事件不冒泡**，即鼠标移到其后代元素上时不会触发。
* mouseleave:鼠标移出元素范围时触发，**该事件不冒泡**，即鼠标移到其后代元素时不会触发。
* mousemove:鼠标在元素内部移到时不断触发。不能通过键盘触发。、

1. mousedown 和 mouseupk 可以说是click在时间上的细分，顺序是mousedown =>mouseup=>click。
2. 在DOM2.0中，W3C对鼠标事件作了规范，鼠标事件被解析为MouseEvent(用e.constructor == MouseEvent来判断其是否为鼠标事件)，左键点击还是右键点击由MouseEvent的button的属性来判定。
   1. 1：按下左键
   2. 2：按下中键
   3. 3：按下右键

### [Javascript事件](https://wenku.baidu.com/view/400a89f4f61fb7360b4c65ca.html)

1. 事件对象 的获取
   1. IE中事件对象是作为全局对象（window.event）存在的，Firefox中则是作为句柄（handler）的第一个参数传入的。
   2. var evt = window.event || arguments[0];
2. 三种添加事件的方式
   1. 第一种添加事件的方式，直接在html中写js代码
      1. eg: <div onclick="alert(4);">Div1 Element</div>
      2. 获取事件对象：直接使用event（全局对象）即可。
   2. 第二种添加事件的方式，定义一个函数，赋值给html元素的onXXX属性
   3. 第三种添加事件的方式，使用element.onXXX方式
   4. 第四种添加事件的方式，使用addEventListener或IE专有的attachEvent
3. 总结
   1. IE6/7/8支持通过window.event获取对象，通过attachEvent方式添加事件时也支持事件对象作为句柄第一个参数传入
   2. Firefox只支持事件对象作为句柄第一个参数传入
   3. IE9/Opera/Safari/Chrome两种方式都支持

### 一周总结

1. js是操作浏览器内容给的

浏览器内容包括：

1. 元素
2. 浏览器工具条，浏览器属性
3. 创造浏览器的交互行为
4. js内容
   1. 变量：
      1. 作用：用来从存放js数据，保存一些需要重复使用数据
   2. 获取元素方法：
      1. document[E].getElementById();
      2. document[E].getElementsByClassName();
      3. document[E].getElementsByTagName();
      4. document[E].querySelector();//获取的是元素集合中的第0个
      5. document[E].querySelectorAll();//获取的是一个集合，即使id也是一个集合。
5. 添加事件
   1. window[E].onload 监测浏览器或者某些元素是否加载完成
6. 函数
   1. 命名函数定义

function 函数名(参数1,参数2,…){

操作;

}

* 1. 调用

函数名();

* 1. 匿名函数定义
     1. function(参数1,参数2,…){操作;}
  2. 1.立即执行 2.存到变量或者属性方法再执行

1. 属性修改：
   1. id,className,style(集合),value（表单元素内容）,innerHTML,(url，href)获取的是绝对地址。
      1. eg:

<textarea>33333</textarea>

<input value = "222222" />

表单元素建议用value获取内容

1. isFinite(testValue)
   1. 用于检测有限性（finiteness）的值，用来检测一个数字是否是有限数字
   2. 如果参数是NaN,正无穷或者负无穷大，会返回false，其他返回true。
   3. eg:
      1. isFinite(Infinity);//false
      2. isFinite(NaN)
      3. ifFinite(2e64) //true,在更强壮的Number.isInfite(null)中将会得到fasle
      4. isFinite(“0"); //true,在更强壮的Number.isInfite('0')中将会得到false。

## 第二周

1. css代码比js执行慢(具体慢多少看transition)
2. 变量作用域范围
   1. 全局变量
   2. 局部变量
   3. javaScript没有块级作用域，i,j,k作用域 一样（同步执行的变量会冲突，异步执行的代码不会冲突）。

<script type="text/javascript">

function rainman(){

// rainman函数体内存在三个局部变量 i j k

var i = 0;

if ( 1 ) {

var j = 0;

for(var k = 0; k < 3; k++) {

alert( k ); //分别弹出 0 1 2

}

alert( k ); //弹出3

}

alert( j ); //弹出0

}

</script>

1. JavaScript执行一个函数时，它会首先查找所有变量声明.无论是在条件块还是其他构造声明。JavaScript一旦找到所有变量，就会执行函数中的代码。如果函数内部隐式声明变量（即该变量出现在复制表达式的左侧但尚且为使用var进行声明）则它将被创建为全局变量（局部变量会屏蔽掉全局变量即使在声明之前）

<script type="text/javascript">

var x = 1;

function rain(){

alert( x ); //弹出 'undefined'，而不是1

var x = 'rain-man';

alert( x ); //弹出 'rain-man'

}

rain();

</script>

1. 未使用var关键字定义的变量都是全局变量
2. 全局变量都是window对象的属性
3. 通过for循环添加的事件内的循环的变量是最终值。（易错点）
   * 1. 给变量添加属性
4. 在for循环括号内声明的变量是全局变量。
5. 只要函数没有被其他对象调用，默认就是window调用。（例如手动执行）
6. 父元素和子元素不要添加相同的事件（冒泡会导致子元素和父元素的事件一块执行）
7. continue跳出当前循环，继续下个循环。
8. 老是忘写style。
9. 用属性当作判断标志不稳定。
10. [background-clip](http://www.w3school.com.cn/tiy/c.asp?f=css_background-clip)规定背景的绘制区域，值border-box,padding-box,content-box.
11. [background-orgin](http://www.w3school.com.cn/tiy/c.asp?f=css_background-origin) 规定背景图像相对什么位置定位值border-box,padding-box,content-box.
12. 单标签（如img）无法添加after伪类。
13. margin要写全。
14. span的line-height高度盒模型是不显示的。
15. 文字描边 text-shadow。
16. [shadow阴影](https://www.w3cplus.com/content/css3-box-shadow)

E {box-shadow: <length> <length> <length>?<length>?||<color>}

也就是：

E {box-shadow:inset x-offset y-offset blur-radius spread-radius color}

换句说：

对象选择器 {box-shadow:投影方式 X轴偏移量 Y轴偏移量 阴影模糊半径 阴影扩展半径 阴影颜色}

1. [特殊符号表](http://star.gg/special-symbols#star_symbols)
2. b标签被挤下来了，设置行高或者vertical-align解决(因为b标签的vertical-align默认值为 baseline;)
3. vertical-align标签必须有一个参照物（加一个span设置高度为line-height）
4. toString是一个被继承的方法，undefined和null没有该方法。
5. 数字类型的转换方法：parseInt(),parseFloat(),Number().
6. NaN是数字类型。
7. NaN不等于自身。
8. isNaN会自动调用Number()强制转换。
9. parseInt和parseFloat必须是数字开头的字符串。
10. Number()转换字形态的字符串。
11. String() //""
12. String(null); //null
13. String(undefined); //undefined
14. String({}); //[object object]
15. String([]); //""空
16. String([1,2]); //1,2
17. String(true); //true
18. String(''); //""空
19. Number(undefined); //NaN \*
20. Number(null) //0
21. Number(false) //0
22. Number(true) //1
23. Number([]) //0
24. Number('') //0
25. Number([1,2]) //NaN
26. Boolean(undefined) //false
27. Boolean(null) //false
28. Boolean(''); //false
29. Boolean(' '); //true
30. Boolean('abc'); //true
31. Boolean(0); //false
32. Boolean(非零数); //true
33. Boolean(NaN); //false
34. Boolean({}); //true
35. Boolean([]); //true
36. Boolean([1,2]); //true
37. undefined，Null不能转换
38. typeof 123; //number
39. typeof '123'; //string
40. typeof {}; //object
41. typeof []; //object
42. typeof function (){};//function
43. typeof undefined //undefined
44. typeof true; //boolean
45. typeof null; //object
46. null就是一个空对象
47. 占位：
    1. 字符串： var str = '';
    2. 数字：var num = 0;
    3. 对象：var timer = null;
48. console.log(1-'a');//NaN
49. 'a'.charCodeAt();看它的ASCAII码值。
50. span.active , span .active//一个是本身的类， 一个是子元素的类
51. float:right放在float：left后会出现换行。将它放在float:left前可以解决
52. 数组中存储的是经过计算的值
53. parseInt和Number()区别
    1. Number构造函数执行类型转换，parseInt执行解析
    2. parseInt可以识别八进制和十六进制的数字。eg:"010","0xa0","0XA0"
    3. parseInt("10100",2) //20
    4. Number()转换的是整个值，parseInt()和parseFloat()转换的是第一个无效字符串之前的字符串。
    5. 如果被转换的字符串开头有空格会被自动去除。
    6. parseInt()如果参数不是字符串，则会先转为字符串再转化eg:parseInt(1.23)等同于parseInt('1.23')
    7. parseInt()会将''转化为NaN,Number('')转化位0
54. 一元运算符
    1. +"2e1" //20
    2. +"0xf" //15
    3. +"010" //10
55. 6种数据类型
    1. 数字
       1. 最大值 Number.MAX\_VALUE 1.7976931348623157e+308
       2. 最小值 Number.MIN\_VALUE 5e-324
       3. 最大安全值Number.MAX\_SAFE\_INTEGER 9007199254740991

Math.pow(2, 53)-1

* + 1. 最小安全值Number.MIN\_SAFE\_INTEGER -9007199254740991

-(Math.pow(2, 53)-1)

* 1. 字符串
  2. 布尔值 true false
  3. 空 null
  4. 未定义 undefined
  5. 对象 {},[],function

{

"fun":function(){}, //方法

"str":asssd, //属性

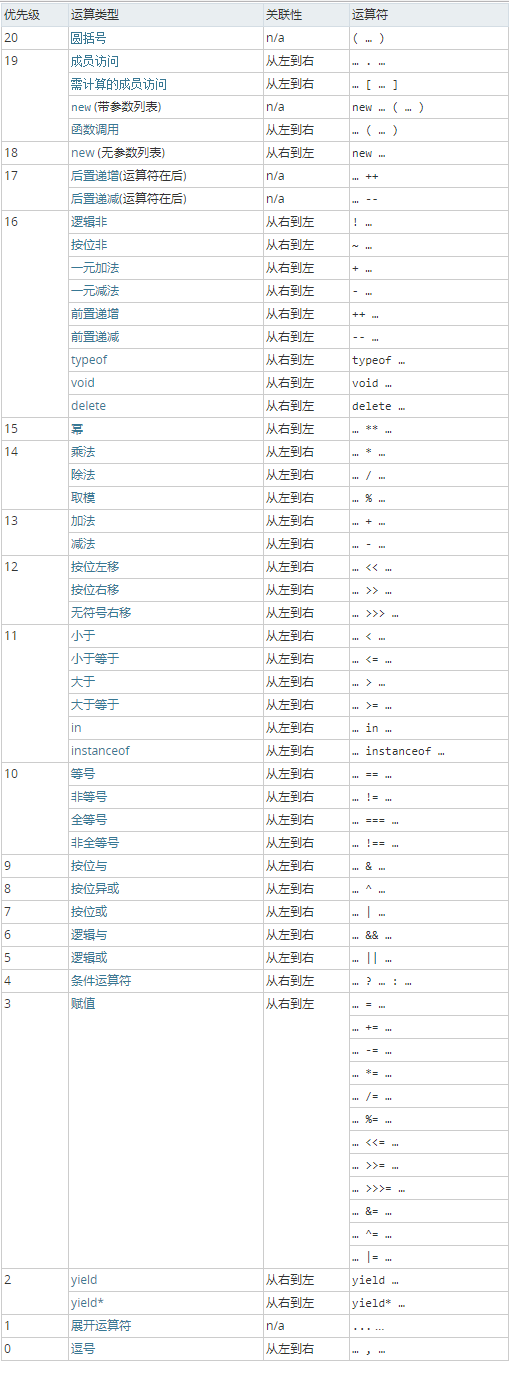
"num":1234,

"bool":false

}

1. typeof
   1. 字符串： 'string'
   2. 数字： 'number'
   3. 布尔： 'boolean'
   4. 空类型： 'object' //特殊
   5. 未定义： 'undefined'
   6. 对象：
      1. {} 'object'
      2. [] 'object'
      3. function(){} 'function'

#### 运算

1. 一元运算（参与运算的值只有一个）
   1. typeof 返回类别名称，返回值是一个小写的字符串
   2. ++ 自增
   3. -- 自减
   4. + 将操作数转为Number类型 console.log(+'10' + 1) //11
   5. - 将操作数转为Number类型并反转正负
2. 浮点数运算有时是不准确的。
3. ++ num == ++ num; //false
4. num ++ == num ++ //false \*
5. Infinity \* 0 //NaN
6. Infinity / Infinity //NaN
7. 0/0 //NaN
8. 0%0 //NaN
9. 取模时，如果被除数是无穷大或者除数是零，结果是NaN
10. null == undefined //true
11. [1,2,3,4] > [1,2,3,3] //true
12. 'abc' > 'abb' //true
13. {} > {} //false 内存地址无法比较
14. {} == {} //false
15. {} < {} //false
16. 运算符优先级
17. 流体布局
    1. display:flex;
    2. justify-content:space-around;

#### 数值

1. 整数和浮点数
   1. JavaScript内部，所有数字都是以64位浮点数形式存储的。
   2. 由于浮点数不是精确的值，所以涉及小数的比较和运算时要特别小心。
      1. 0.1 + 0.2 === 0.3 //false
      2. 0.3 / 0.1 // 2.9999999999999996
      3. ( 0.3 – 0.2 ) === ( 0.2 – 0.1) //false
2. 数值精度
   1. JavaScript浮点数的64个二进制位，  
      第1位：符号位，0表示正数，1表示负数。  
      第2-12位：指数部分  
      第13-64位：小数部分（即有效数字）
   2. 有效数字的第一位默认是1，不保存在64位浮点数中，因此JavaScript提供的有效数字最长位53个二进制位。
   3. 大于等于2的53次方的数值，都无法保持精度。
3. 数值范围
   1. 21024到2-1023 //能够表示的数值范围，超出这个范围显示Infinity或-Infinity ,不保证精度。
   2. –(253-1)到(253-1) //保证精度的数值范围
   3. 指数输入方式：ctrl+shift+=,下标输入方式：ctrl+=
4. 数值的表示法
   1. 科学计数法（123e3,123e-3,-3.13E+12,.1e-23）
      1. 当小数点前的数字多余21位或小数点后的零多余5个会自动转为科学计数法
   2. 数值的进制
      1. 十进制：没有前导0的数值
      2. 八进制：有前缀0o或0O，或有前导0、且只用到0-7
      3. 十六进制：有前缀0x或0X的数值
      4. 二进制：有前缀0b或0B的数值
   3. 根据前导0来判断在ES6中已经废除，但是浏览器目前还支持。
5. 特殊数值
   1. 正零和负零，几乎等价（唯一的区别 1/+0 === 1/-0 //false）一个是+Infinity，一个是-Infinity。
   2. NaN
      1. typeof(NaN) //"number"
      2. isNaN(number);
         1. isNaN(NaN) //true
         2. isNaN('abc') //true, \*注意，isNaN只对数值有效，传入其他会先被自动转换成number
         3. 同理，isNaN({}),isNaN(['zyx']) //true
         4. 因此，使用isNaN之前最好判断是否为number类型。
            1. typeof value === "number" && isNaN(number);
         5. NaN是js中唯一不等于自身值
            1. return value !== value;
   3. Infinity
      1. 定义
      2. Infinity表示"无穷"，用来表示两种场景
      3. Infinity与NaN比较，总是返回false
      4. 运算规则
         1. 0 \* Infinity //NaN
         2. 0 / Infinity //0
         3. Infinity / 0 //Infinity
      5. Infinity > NaN //false
      6. Infinity 与null计算，null会转成0，等同于与0的计算。
      7. Infinity 与 undefined计算，返回的都是NaN
      8. Infinity减去或除以Infinity，得到NaN
      9. isFinite函数检测是否是正常数值，
         1. isFinite(Infinity)
         2. isFinite(-1)
         3. isFinite(true)
         4. isFinite(NaN)
6. 与数值相关的全局方法
   1. parseInt()
      1. 如果第一个字符不能转化位数字（后面跟着数字的正负号除外），返回NaN
      2. parseInt的返回值只有两种可能，不是一个十进制整数，就是NaN
      3. parseInt不认识科学计数法。

## 第三周

### 函数

1. 函数命名遵循小驼峰命名法。
2. 函数定义
   1. function abs1(x){

return x;

}

* 1. var abs2 = function(x) {

return x;

};（注意末尾的需要加一个';'）

1. 调用函数
   1. 未传入的参数将收到undefiend。
   2. arguments[i]获取第i个参数。（可以使用arguments.length判断是否传入参数，但是它没有语义化表示所以并不推荐使用，）
   3. 函数默认的返回值undefined。
2. 作用域
   1. 如果不用var声明，默认是全局变量。
   2. 局部变量如果声明在使用之后，则使用就是undefined。
   3. 闭包
      1. 就是通过函数创建一个不被外接访问的作用域。
      2. 优点：保护变量和函数名不被污染
      3. 缺点：闭包的变量不会被自动回收，造成内存溢出。
      4. (function(){  
          window.fn = function(num){  
          return sum() + sum;  
          }  
         })();  
         window.fn(10); //2
3. css动画：
   1. @keyframes 规定动画。
   2. animation 所有动画属性的简写属性，除了 animation-play-state 属性。
   3. animation-name:myfirst; 规定需要绑定到选择器的 keyframe 名称。。
   4. animation-duration:5s; 规定完成动画所花费的时间，以秒或毫秒计。
   5. animation-timing-function:linear; 规定动画的速度曲线。
   6. animation-delay:2s; 规定在动画开始之前的延迟。
   7. animation-iteration-count:infinite; 规定动画应该播放的次数。
   8. animation-direction:alternate; 规定是否应该轮流反向播放动画。
   9. animation-play-state:running; 规定动画是否正在运行或暂停。默认是 "running"。
   10. animation-fill-mode 规定对象动画时间之外的状态。
4. 用hover修改其他元素的样式（伪类选择器）

<div class="container">

<ul class="ul">

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

</div>

<div class="wrap"></div>

.container:hover +.wrap{background:#f00;}

1. CSS实现背景色从上向下渐变  
   div{  
    **position**: **relative**;  
    **z-index**: 2;  
    **display**: **inline-block**;  
    **margin**: 0;  
    **padding**: 3**px** 10**px**;  
    **color**: **#fff**;  
    **background-color**: **rgba**(10,140,210,1);  
    **overflow**: **hidden**;  
   }  
   div:**before**{  
    **display**: **inline-block**;  
    **position**: **absolute**;  
    **top**: -100%;  
    **left**: 0;  
    **z-index**: -1; */\*为什么设置1达不到效果？\*/* **width**: 100%;  
    **height**: 100%;  
    **background-color**: **#005EAC**;  
    **transition**: **ease-out** 0.5**s**;  
    **content**: **''**;  
   }div:**hover**:**before**{  
    **top**: 0;  
   }
2. CSS多行文本溢出省略显示
   1. webkit内核的浏览器   
      **overflow**: **hidden**;  
      **width**: 490**px**;  
      **height**: 90**px**;  
      **text-overflow**: **ellipsis**;  
      **display**: **-webkit-box**;  
      **-webkit-line-clamp**: 4;  
      **-webkit-box-orient**: **vertical**;
   2. 通用方法用after往后加一个…
3. inset是上左灰色，下和右是设置的颜色。
4. overflow:hidden后只要超出父级范围就不显示
5. 伪类设置z-index：-1；后虽然是比所有层级都低，但仍然在父级之中所以会导致这种效果。建议用父级再套一种div的方式去做，设置z-index：-1后会导致3D效果的交叉效果。
6. 外部引入的js是同一个作用域。
7. 先找子集作用域的声明，如果有但是还没有执行赋值语句那么就是undefined，否则就向父级找。(js预解析，js会先收集收有的var和function，先把所有收集的变量和函数存入undefined。函数是直接存储函数体。)
   1. eg:
8. 闭包
   1. 闭包目的是隐藏变量
   2. 开放接口（return或者绑定到window对象上）。
9. string.charAt(index)获取字符串的某个位置的字符。
10. 用isNaN()判断是否是纯数字。
11. for(var i in object){  
     console.log(i);  
    }
    1. i是对象的键名。
12. 开根号：Math.exp(1/3\*Math.log(a))//不精确

##### 定时器

1. 间隔定时器
   1. setInterval(fn,1000);
2. 延时定时器
   1. setTimeout(fn,1000);
3. 字符串方式函数使用
   1. setInterval("fn()",1000); //一直执行
   2. setTimeout("fn(1,3)",1000); //执行依稀
4. 如果要传值的话，建议使用匿名函数的形式。
   1. setInterval(function(){  
       fn(1,3);  
      },1000);
5. 将定时器Id存入变量，以便clearInterval()调用。
6. 定时器封装  
   function autoStart(){  
    var timer = setInterval("fn(1,3),2000");  
   }
7. 每次定时器开启都会有一个新的Id。
8. [定时器因为js单线程执行原因会不准时。会等当前代码块执行完毕后再执行定时器。](http://caibaojian.com/js-event-loop.html)
   1. eg:

**function** *foo*() {  
 **console**.log( **'first'** );  
 setTimeout( ( **function**(){ **console**.log( **'second'** ); } ), 5);  
  
}  
**for** (**var *i*** = 0; ***i*** < 3000; ***i***++) {  
 *foo*();  
}

1. setTimeout(func, 0) 重点是改变了代码流程，把func的执行放到了等待当前的代码执行完毕再执行。
2. 页面切换后，定时器仍然会执行。
3. getComputedStyle(obj)[attr];获取元素属性(会有单位)。
4. call会改变函数内部的this指针的引用对象。
5. 定时器开启后不会自动关闭，会一块执行。
6. 异步执行的代码要用作用域的方式传值。
7. 回调函数，将函数作为参数传入。
8. 对于小数的运动数值，建议先放大倍数。最后设置结果时再缩小。
9. js不能实现动画效果的属性：
10. getComputeStyle获取的是人为设置的属性，无论是css还是行间样式，必须设置才会生效。默认样式可能会为none。
11. top不能用于变量名。
12. 调用定时器前要先清除定时器，否则会有多个定时器。
13. 因为网页坐标的原点在左上角，所以设置高度会导致向下。
14. fn&&fn()如果存在，就执行fn函数。
15. 页面上的时间一般使用事件戳来计算时间。
16. document.write不是重写而是添加在后面
17. 获取的月份比真正的月份数小一（易错）
18. getDay()获取星期几
19. getDate()获取日
20. new Date('August 16,2016 10:29:12')
21. 服务器后台一般以秒为单位计算。
22. **var** countdownDay = parseInt (differTime / 86400000);  
    **var** countdownHour = parseInt (differTime % 86400000 / 3600000);  
    **var** countdownMinutes = parseInt (differTime % 3600000 / (60\*1000));  
    **var** countdownSecond = parseInt (differTime % 60000 / 1000 );  
    ***countdown***.**innerHTML** = countdownDay + **"天"** + countdownHour + **"时"** + countdownMinutes + **"分"** + countdownSecond +**"秒"**;
23. ***date***.setDate(); *//小于1会设置位上一个月的日期****date***.setDate(); *//超过本月天数会设置为下个月的日期。*
24. date不能设置星期
25. 设置时间没有限制，会自动计算。
26. 负数会按照比他大一级的事件单位的上一级的满时间减去相应的时间。
27. date.setDate(0);会设置位上个月的最后一天。
28. 页面重新渲染会导致事件失效。
29. 让规模扩大缩小

**-webkit-transition**: **all** .2**s ease-in**;  
**-moz-transition**: **all** .2**s ease-in**;  
**-ms-transition**: **all** .2**s ease-in**;  
**-o-transition**: **all** .2**s ease-in**;  
**-webkit-transform**: **scale**(1.1);  
**-moz-transform**: **scale**(1.1);  
**-ms-transform**: **scale**(1.1);  
**-o-transform**: **scale**(1.1);

1. 虽然在border-color上能运用rgba()、hsla()设置边框为半透明或完全透明，如果元素设置了背景颜色或背景图片的时候，会直接影响边框的透明颜色效果。特别是，要看到边框底下的内容时。造成这个现象是由于背景图片会延伸到边框底部。要解决这一问题，可以通过CSS3的background-clip来修正。著作权归作者所有。
2. [js如何控制css伪元素内容（before，after）](http://www.cnblogs.com/yunkou/p/4269974.html)
3. [【CSS进阶】伪元素的妙用--单标签之美](https://shimo.im/doc/ONQyy3ZcFaM6hJgY)
4. [获取Object对象的length](http://www.cnblogs.com/zera/archive/2013/03/26/2983123.html)
   1. Object.**prototype**.length = **function** (){  
       **var** count = 0;  
       **for**(**var** i **in this**){  
       count ++;  
       }  
       **return** count - 1;  
      };
5. 添加代码块会导致方法不再是全局方法 //？
6. 字符串
7. charAt();
8. charCodeAt();
9. indexOf(); //查找首次出现的位置，第二个参数指定查找的起始值，未找到返回-1。查找位空返回查找起始值。
10. lastIndexof();//查找最后一次出现的位置，第二个参数指定查找的起始值
    1. **str = "fjksjfk";  
       console**.log(**str**.indexOf(**'f'**)); //0  
       **console**.log(**str**.indexOf(**' '**)); //-1  
       **console**.log(**str**.indexOf(**''**)); //0字符之间有空所以为0  
       **console**.log(**str**.indexOf(**'f'**,2)); //5第二个参数指的是从哪里开始查找
    2. 判断className中是否有某个值
11. slice(startPosition,endPosition);
    1. 截取字符串，包括开始字符但不包括结束位置字符。
    2. 参数可以是负数，如果是负数的话从末尾开始计算。
    3. 返回被截取的字符串
12. [substring和substr和slice区别：](http://www.cnblogs.com/ider/p/js-slice-vs-substr-vs-substring-table.html)
13. substring 与slice 的区别，substring将负数视为0，slice从末尾开始计算。
14. substr与slice 的区别：参数2在slice和substring中表示结束位置，而在substr中代表的则是子串长度；
15. 对于参数1小于参数2的情况，substring最大的不同在于它会交换两个参数再截取子串，substr因第二参数表示的是长度因此并无异常，slice则依然正常搜寻子串始末位置，若开始位置在结尾后边则返回空串。
16. split（）,参数一，分割符，参数二，分割后的String数组长度。
    1. 分割符不填，默认为一个整串。
17. arr.join();参数一，合并数组时添加的分隔符，默认是一个英文的'，'
18. trim 去掉首尾空格。
19. toLowerCase();转为小写
20. toUpperCase();转为大写
21. [首字母大写CSS](https://segmentfault.com/q/1010000003020515)：text-transform: capitalize;
22. user-modify ：read-write-plaintext-only 只能输入纯文本。//仅chrome支持
23. div可输入：contenteditable:plaintext-only; 只能输入纯文本。//仅chrome支持
24. [placeholder样式更改](https://segmentfault.com/q/1010000000397925)：

::-webkit-input-placeholder { /\* WebKit browsers \*/

color: #999;

}

:-moz-placeholder { /\* Mozilla Firefox 4 to 18 \*/

color: #999;

}

::-moz-placeholder { /\* Mozilla Firefox 19+ \*/

color: #999;

}

:-ms-input-placeholder { /\* Internet Explorer 10+ \*/

color: #999;

}

1. z-index必须设置定位。relative或者absolute，fixed都可以
2. 间距，大小有问题一般源于inline-block
3. Math.ceil() 向上取整
4. Math.floor() 向下取整
5. Math.round() 四舍五入
6. Math.abs() 取绝对值
7. Math.random() 取随机数 ,返回0~1之间的小数。
8. 参数有要求的函数会隐式转换为要求的类型。
9. Math.round(-10.1)//-10
10. Math.round(-10.5)//-10
11. Math.round(-10.6)//-11
12. 正： 1-4 往比较小数的舍， 5-9 往比较大的数进
13. 负： -6~-9 往比较小数的舍， -1~-5 往比较大的数进
14. Math.sqrt();
15. Math.pow(x,y) x:底数，y:幂数
16. arr.push(number); 往最后添加变量。
17. arr.reverse(); 颠倒数组序列，会改变原始数据。
18. string.indexOf(); 返回第一个字符串的下标，找不到返回-1
19. 有padding默认样式的元素： ul ol
20. 有margin默认样式的元素： body hr h1~h6 p dl dd
21. 哪些元素是独占一行的： div h1~h6 p hr ul ol dl nav header section footer article aside
22. 哪些元素是一行能显示多个的： a strong em span time mark
23. cssReset

body, h1, h2, h3, h4,h5,h6,hr,dl,dd,p

{

margin: 0;

}

ul, ol{

margin: 0;

padding: 0;

list-style: none;

}

img{

vertical-align: top;

border: none;

}

a{

text-decoration: none;

color: #000;

}

1. p,dt,h标签 里面不能嵌套块属性标签
2. a标签不能嵌套a
3. 内联标签不能嵌套块标签；

#### 数组

1. var arr = new Array(1,2,3,4,5);
2. var arr = new Array(10);单独填写一个数字待变数组的长度。用undefined占位。
3. push往数组最后添加一个元素， pop将数组最后一个元素移除。
   1. array.push(content,[,content,content]);
4. 基本类型传值，复合类型传址。
5. 不想改变元数组使用浅拷贝。声明一个新数组，通过循环挨个赋值。
6. unshift：往数组第一个为位置添加内容。返回值，新数组的长度。
7. shift: 从开始删除一个元素。
8. forEach(callback);
   1. 遍历数组
   2. 参数：回调函数
      1. 回调函数默认的第一个代表数组的值
      2. 回调函数第二个默认参数代表下表。
   3. foreach会过过滤掉无效值。
9. join(参数)
   1. 作用把数组拼接成字符串
   2. 参数：用来拼接成字符串连接符
   3. 默认用逗号拼接。
10. concat
    1. eg：arr1.concat(arr2);
    2. 数组合并。
    3. 参数
    4. 元数据不会发生变换。
    5. 应用：数组合并
11. slice
    1. 截取数组的开始位置
    2. 参数
       1. start 截取的数组开始位置
       2. end 截取的数组的结束位置。截取的位置不包括自己，如果不填默认截取整个数组。
12. splice(start,howmonnnty)
    1. 可去除数组指定位置的元素。
    2. 替换数组中的内容
    3. 参数
       1. index开始的位置
       2. 替换的个数
       3. item替换的内容
    4. 返回值是被替换的内容，
    5. 可作为删除和添加使用‘’
       1. arr1.splice(0,0,4,5); //添加
       2. arr2.splice(0,2,1,2); //删除
13. sort
    1. // 从小到大

var arr3 = [3,2,6,1,9];

arr3.sort(function (a,b) {

return a-b;

});

* 1. // 从大到小

var arr3 = [3,2,6,1,9];

arr3.sort(function (a,b) {

return b-a;

});

* 1. //随机排序

var arr3 = [3,2,6,1,9];

arr3.sort(function (a,b) {

return 0.5 – Math.random;

});

#### [SVG](http://www.cnblogs.com/hupeng/archive/2012/12/21/2828456.html)

1. what:开放标准的矢量图形语
2. why:
   1. 不失真
   2. 体积小
   3. 可以自由修改
   4. 色彩分辨率高，支持滤镜
   5. 跨平台
3. whe re:适合静态图片展示，高保真文档查看和打印的应用场景。

备注：pc端强交互少元素 svg胜出 |

pc端弱交互而强性能 用canvas |

pc端超强性能 webgl-canvas |

移动端弱交互 原生canvas挺好（木有ie的世界真好） |

移动端强交互 刚才提到的一些基于canvas的库不错（svg性能很挫）

1. how:
   1. 包含在<svg>标签中，需要浏览器支持HTML5中的svg标签。
2. explanation:
   1. width和height定义svg的宽高
   2. version定义版本
   3. xmlns属性可定义svg命名空间。
   4. 用<circle>来创建圆，cx和cy定义圆中心,r定义半径。
   5. stroke和stroke-width属性控制如何显示形状和轮廓。
   6. fill属性设置形状的颜色。
3. animate：相当于 CSS 中的 transition
   1. eg:

<rect x="10" y="10" width="200" height="20" stroke="black" fill="none">

<animate

attributeName="width"

attributeType="XML"

from="200" to="20"

begin="0s" dur="5s"

fill="freeze" />

</rect>

1. animateTransform：相当于 CSS 中的 transform
   1. eg：

<rect x="-10" y="-10" width="20" height="20"

style="fill: #ff9; stroke: black;">

<animateTransform attributeType="XML"

attributeName="transform" type="scale"

from="1" to="4 2"

begin="0s" dur="4s" fill="freeze"/>

</rect>

1. animate morph：主要做图形渐变
   1. repeatCount="indefinite",无限循环执行
   2. keyTimes:和CSS中定义的@keyframes一样。通过0-1之间的值，定义每段动画完成的事件。格式为：value;value…。例如：0;.0625;.208333333;.3125;.395833333;.645833333;.833333333;1。从第一个动画，到第二个动画经历的时间比例为6.25%。并且，keyTimes需要和values里面定义的帧数一致。
   3. calMode：用来定义动画具体的插值模型。取值有：discrete|linear[deafault]|paced|spline。具体可以参考MDN。这里我们主要介绍一下spline。该值表示每个动画间使用自定的贝塞尔变换曲线。如果没有特殊要求，使用linear其实已经足够了，不用去定义下面的keySplines属性。
   4. keySplines：该值用来定义动画执行时的贝塞尔曲线。使用格式是通过;来分割每一个值。
   5. values：直接结合attributeName属性，来设置具体的值，每个值之间使用;进行分隔。
   6. where:Morph比较常用于数字的更迭，比如，倒数10s的相关动画。
2. animateMotion
   1. 基本属性和animat差不多，还有keyPoints,rotate,path等。不过，calcMode在AM中的默认属性位paced。
   2. 在定义AM路径时，多种方式会发生覆盖：mpath > path > values > from/to。
   3. AM中的旋转角：默认运动元素的角度是它与坐标轴的初始角度。
   4. rotate属性值
      1. auto：让元素垂直与路径的切线方向运动。不过，如果路径是闭合的话需要注意起始点的位置。
      2. auto-reverse:让元素垂直于路径的切线方向并+180，与auto方向切线对称。
      3. Number：让物体以固定角度的旋转角度运动。这个相当于使用transform：rotate(deg)进行控制。
      4. set:和animate的主要区别是，它仅仅需要to的指定属性，而不需要其他的参考属性，比如from，by等。可以用它来描述一些非number的属性值。
   5. 矩阵动画 抛物线实现 matrix keyframe
   6. 线性动画：
      1. where：过渡屏，等待界面，酷炫的LOGO
      2. stoke:定义笔触的颜色。eg:stroke="green";
      3. stroke-dasharray\*:定义dash和gap的长度。它主要通过使用,来分隔实线和间隔的值。
         1. eg:stroke-dasharray="5,5"表示，按照实线长度5，每个实线间距为5的排布重复下去。
      4. 另外stroke-dasharray 并不局限于只能设置两个值，要知道，它本身的含义是设置最小重复单元，即，dash,gap,dash,gap...。比如，我定义 stroke-dasharray="15, 10, 5" 则相当于，[15,10,5] 为一段。则有：



1. onmclick抬起有延迟，用onmousedown效果更好一点
2. 更改鼠标指针
   1. **\***{  
       **cursor**: **url**(**./cd\_glowsword.cur**),**auto !important**;  
       }  
       **\***:**active**{  
       **cursor**: **url**(**./cd\_glaive.cur**),**auto !important**;  
       }

#### apply,call,bind

1. call和apply是为了改变函数体内部this的指向。
2. JavaScript存在一个特点：函数存在定义时上下文和运行时上下文、以及上下文是可以改动的。
   1. eg:

***apple***.say();  
***apple***.say.call(***banana***);  
***apple***.say.apply(***banana***);

1. call,apply二者作用完全一样，只是接收参数的方式不一样
   1. eg:

var func = function(arg1,arg2) {

}

func.call(this,arg1,arg2);

func.apply(this,[arg1,arg2]);

* 1. this是想要指定的上下文，可以是任何一个js对象（js中一切皆对象），call需要把参数传递进去，apply则是需要将参数放入数组。
  2. 适用条件：
     1. 参数数量明确时使用call
     2. 不确定时使用apply

1. 常用用法
   1. inline的vertical-align属性值是和父级的线对齐
   2. outerHTML获取整个元素的内容（包括标签）。不兼容。
2. 对定时器赋值时要注意，易导致定时器清除不掉。
3. 用innerHTML在获取更改元素，少用DOM操作。
4. 递归， 闭包， 面向对象， 作用域， 预解析。
5. Date.now()获取时间戳。
6. js计算小数精度有问题，先放大再缩小
7. 缩小动画
   1. transform:scale();
8. 回调函数与调用函数同级。
9. 用css写红心：

.**much-heart** {  
 **display**: **inline-block**;  
 **position**: **relative**;   
 **margin**: 0 4**px**;  
 **height**: 10**px**;  
 **width**: 10**px**;  
 **background**: **#ac1D3f**;  
 **-webkit-transform**: **rotate**(45**deg**);  
 **-moz-transform**: **rotate**(45**deg**);  
 **-ms-transform**: **rotate**(45**deg**);  
 **-o-transform**: **rotate**(45**deg**);  
 **transform**: **rotate**(45**deg**);   
}  
  
.**much-heart**::**before**,

.**much-heart**::**after** {  
 **display**: **block**;  
 **content**: **''**;  
 **position**: **absolute**;  
 **margin**: **auto**;  
 **height**: 10**px**;  
 **width**: 10**px**;  
 **border-radius**: 5**px**;  
 **background**: **#AC1D3F**;  
 **top**: -4**px**;  
}  
  
.**much-heart**::**after** {  
 **bottom**: 0;  
 **top**: **auto**;  
 **left**: -4**px**;  
}

1. 甭管以前有没有定时器，开之前先清一下。
2. slice可以将类数组对象转化为数组，例如参数组和NodeList集合。
   1. Array.prototype.slice.call(NodeList);
   2. [].slice.call(arguments)
   3. var unboundSlice = Array.prototype.slice;

var slice = Function.prototype.call.bind(unboundSlice);

slice(arguments);

1. 数组之间追加Array.prototype.push.apply(arr1,arr2);
2. 获取数组中的最大值和最小值
   * 1. var maxInArray = Math.max.apply(null,array);
        1. 这块在调用的时候第一个参数给了一个null,这个是因为没有对象去调用这个方法,我只需要用这个方法帮我运算,得到返回的结果就行,.所以直接传递了一个null过去
     2. var maxInArray = Math.max.call(Math,4,7,9,3,5,1,2);
3. 验证是否为数组
   * 1. Object.prototype.toString.call(obj) === "[object Array]";
   1. 类数组使用数组方法
      1. var domNodes = Array.prototype.slice.call(document.querySelector("div"));

## bind 详解

1. bind() 方法与 apply 和 call 很相似，也是可以改变函数体内 this 的指向。
2. bind()方法会创建一个新函数，称为绑定函数，当调用这个绑定函数时，绑定函数会以创建它时传入 bind()方法的第一个参数作为 this。
3. 在常见的单体模式中，通常我们会使用 \_this , that , self 等保存 this ，这样我们可以在改变了上下文之后继续引用到它。
4. 在Javascript中，多次 bind() 是无效的。
   1. bind() 的实现，相当于使用函数在内部包了一个 call / apply ，第二次 bind() 相当于再包住第一次 bind() ,故第二次以后的 bind 是无法生效的。

## apply、call、bind比较

1. 当你希望改变上下文环境后回调执行，使用bind方法。
2. 立即执行，使用call/apply方法。
3. bind 是返回对应函数，便于稍后调用；apply 、call 则是立即调用 。

#### 获取元素在网页中的位置

1. 元素都有clientHeight和clientWidth属性。这两个属性指的是元素的内容加上padding所占的视觉面积，不包括border和滚动条所占的面积。

因此，document的clientHeight和clientWidth属性，就代表了网页的大小。

1. 获取网页大小：
   1. function getViewport(){

　　　　if (document.compatMode == "BackCompat"){

　　　　　　return {/\*IE6's quirks\*/

　　　　　　　　width: document.body.clientWidth,

　　　　　　　　height: document.body.clientHeight

　　　　　　}

　　　　} else {

　　　　　　return {

　　　　　　　　width: document.documentElement.clientWidth,

　　　　　　　　height: document.documentElement.clientHeight

　　　　　　}

　　　　}

}

备注：clientWidth和clientHeight都是只读属性，不能对它们赋值。

* 1. 通过scrollWidth和scrollHeight获取网页大小。

function getPagearea(){

　　return {

　　　　width: document.documentElement.scrollWidth,

　　　　height: document.documentElement.scrollHeight

　　}

}

* 1. 如果浏览器不出现滚动条，那么网页的clientWidth和scrollWidth的值应该相等， 但是实际上，因为浏览器有不同的处理，所以这两个值未必相等。所以，我们需要取其中较大的值，因此对getPagearea()进行改写。

function getPagearea(){

　　return {

　width: Math.max(document.documentElement.scrollWidth,　　　　　　　　　　　　　　　document.documentElement.clientWidth),

　 height: Math.max(document.documentElement.scrollHeight,　　　　　　　　　　　　　　　　document.documentElement.clientHeight)

　　}

}

1. 获取网页元素的绝对位置
   1. 网页的绝对位置，指的是元素左上角相对于整张网页左上角的坐标。
   2. 首先，每个元素都有offsetLeft和offsetTop属性，表示该元素的左上角和父容器（offsetParent）左上角的距离。所以只要将这两个值进行累加，就可以得到该元素的坐标。（有问题）

　　function getElementLeft(element){

　　　　var actualLeft = element.offsetLeft;

　　　　var current = element.offsetParent;

　　　　while (current !== null && current !== undefined){

　　　　　　actualLeft += current.current.offsetLeft ;

　　　　　　current = current.offsetParent;

　　　　}

　　　　return actualLeft;

　}

　function getElementTop(element){

　　　var actualTop = element.offsetTop;

　　　var current = element.offsetParent;

　　　while (current !== null && current !== undefined){

　　　　　actualTop += current.offsetTop;

　　　　　current = current.offsetParent;

　　　}

　　　return actualTop;

　}

备注：由于在表格和iframe中，offsetParent对象并不等于父容器，所以上面的函数对于表格和iframe中的元素并不适用。

1. 获取网页中的相对位置
   1. 元素的相对位置，指的是该元素左上角相对于浏览器左上角的坐标。
   2. 有了绝对位置，只要将绝对位置减去滚动条滚动的距离就可以了。滚动的距离是document对象的scrollTop和scrollLeft。

function getElementViewLeft(element){

　　var actualLeft = element.offsetLeft;

　　var current = element.offsetParent;

　　while (current !== null){

　　　　actualLeft += current.offsetLeft;

　　　　current = current.offsetParent;

　　}

var elementScrollLeft=document.documentElement.scrollLeft;

　　　　return actualLeft-elementScrollLeft;

}

function getElementViewTop(element){

　　var actualTop = element.offsetTop;

　　var current = element.offsetParent;

　　while (current !== null){

　　　　actualTop += current. offsetTop;

　　　　current = current.offsetParent;

　　}

var elementScrollTop=document.documentElement.scrollTop;

　 return actualTop-elementScrollTop;

}

scrollTop和scrollLeft属性是可以赋值的，并且会立即自动滚动网页到相应位置，因此可以利用它们改变网页元素的相对位置。另外，element.scrollIntoView()方法也有类似作用，可以使网页元素出现在浏览器窗口的左上角。

1. 获取元素位置的快速方法
   1. getBoundingClientRect()方法。它返回一个对象，其中包含了left、right、top、bottom四个属性，分别对应了该元素的左上角和右下角相对于浏览器窗口（viewport）左上角的距离。
   2. 网页元素的相对位置
      1. var X = this.getBoundingClientRect().left;
      2. var Y = this.getBoundingClientRect().top;
   3. 网页元素的绝对位置（相对位置加上滚动距离）
      1. var X= this.getBoundingClientRect().left+document.documentElement.scrollLeft;
      2. var Y =this.getBoundingClientRect().top+document.documentElement.scrollTop;

#### [viewport](https://www.cnblogs.com/2050/p/3877280.html)

1. <meta name="viewport" content="width=device-width, initial-scale=1">
2. initial-scale:初始缩放比例
3. maximum-scale:允许缩放的最大比例
4. minimum-scale:允许缩放的最小比例
5. user-scale:是否允许手动缩放
6. eg:<meta name="viewport" content="width=500,initial-scale=1">
7. 因为某些手机屏幕很小，所以在手机上慎用 position: fixed
8. 移动设备上的viewport就是设备的屏幕上能用来显示我们的网页的那一块区域，在具体一点，就是浏览器上(也可能是一个app中的webview)用来显示网页的那部分区域，但viewport又不局限于浏览器可视区域的大小，它可能比浏览器的可视区域要大，也可能比浏览器的可视区域要小。
9. 移动设备上的浏览器默认的viewport一般是980px.
10. 使用viewportcss中的1px并不等于设备的1px
11. 在苹果的规范中，meta viewport 有6个属性(暂且把content中的那些东西称为一个个属性和值)，如下：
12. 关于缩放以及initial-scale的默认值
    1. visual viewport宽度 = ideal viewport宽度 / 当前缩放值
    2. 当前缩放值 = ideal viewport宽度 / visual viewport宽度
    3. visual viewport的宽度指的是浏览器可视区域的宽度。

|  |  |
| --- | --- |
| width | 设置***layout viewport***  的宽度，为一个正整数，或字符串"width-device" |
| initial-scale | 设置页面的初始缩放值，为一个数字，可以带小数 |
| minimum-scale | 允许用户的最小缩放值，为一个数字，可以带小数 |
| maximum-scale | 允许用户的最大缩放值，为一个数字，可以带小数 |
| height | 设置***layout viewport***  的高度，这个属性对我们并不重要，很少使用 |
| user-scalable | 是否允许用户进行缩放，值为"no"或"yes", no 代表不允许，yes代表允许 |

#### json

1. 定义：是前端与服务端交互的一种格式。
2. json中的引号必须是双引号。
3. JSON,stringify(obj);
   1. 作用：将对象转成Json字符串。
   2. 参数：是一个对象[],{}
4. for … in
   1. 用来循环无序对象。
   2. eg：

var arr = [1,2,3,4,5];

for(var i in jsonobj){

i //键

jsonobj[i] //键值

}

#### offsetHeight,scrollHeight,clientHeight区别

##### offset

1. offsetLeft寻找的是元素border（不包括border）距离定位父级border (不包括border)的left。
2. 行内元素会影响offsetHeight,多一个line-height。
3. body特殊会计算到浏览器边界
4. offsetWidth包括border和padding，不包括margin。
5. chrome默认margin:8px;

##### client

1. clientLeft是当前元素的左边框宽度。
2. clientWidth指的是当前元素不包括border的宽度。

##### scroll

1. scrollLeft滚动条滚动左边的距离。
2. 可以赋值，没有单位。
3. scrollHeight能滚动的高度

##### 总结

E:element

E.offsetLeft E到定位父级的左边距

E.clientLeft 获取E边框宽度

E.scrollLeft E的左滚动位置

E.offsetWidth 获取包括边框，内边距，以及内容的撑开的所有宽度

E.clientWidth 获取除了边框的宽度。

E.scrollWidth获取可以滚动的宽度

#### addEventListener

1. addEventListener会多次绑定匿名函数，触发时会触发多次。
2. 行间样式使用getAttribute

#### 媒体查询

div.sidebar {

width: 300px;

}

@media all and (max-width: 400px) {

// 样式生效当页面宽度小于400px

div.sidebar {

width: 100px;

}

}

#### 属性节点

1. 获取属性节点DOM getAtrrbute();
2. 设置属性节点setAttribute('attribute','setString');
3. 属性节点集合attrbutes();
4. 添加节点
   1. appendChild();//将元素添加到最后
5. insertBefore(argument1，argument2);//argument1插入到argument2之前。
6. parentNode.insertBefore(参数1,null) = appendChild()如果第二个参数为空则往最后添加。
7. 克隆节点不会克隆属性不会克隆方法。

#### 元素分类

1. block元素（块元素）大致有：P|H1|H2|H3|H4|H5|H6|UL|OL|PRE| DL | DIV | NOSCRIPT | BLOCKQUOTE | FORM | HR | TABLE | FIELDSET | ADDRESS(随着html5标准的推进，一些元素将被废除，而一些新的元素将被引入)注意的是并非所有的block元素的默认display属性都是block，像table这种display:table的元素也是block元素。
2. inline元素（内联元素）大致有：#PCDATA（即文本）| TT | I | B | BIG | SMALL|EM | STRONG | DFN | CODE |SAMP | KBD | VAR | CITE | ABBR | ACRONYM|A | IMG | OBJECT | BR | SCRIPT | MAP | Q | SUB | SUP | SPAN | BDO|INPUT | SELECT | TEXTAREA | LABEL | BUTTON
3. <!-- 标题栏logo图标 --><!-- 收藏夹logo图标 -->  
   <link rel="shortcut icon" href="" type="image/x-icon">
4. 花式弹窗 <http://www.jq22.com/yanshi594>
5. [backface-visibility](https://developer.mozilla.org/zh-CN/docs/Web/CSS/backface-visibility) 属性指定当元素背面朝向观察者时是否可见。元素的背面总是透明的，当其朝向观察者时，显示正面的镜像。
6. table.tHead.rows[0].cells[0]; 或者 table.rows[m].cells[n];
   1. table 🡪 通过dom节点获取
   2. table.tFoot 获取头部节点。
   3. table.tFoot 获取底部节点。
   4. table.tBodies获取主体节点（集合）
   5. cells[index] 获取行集合
   6. cells[index] 获取列集合
7. tr 必须放在thead,tbody,tfoot中。
8. colspan="n" 一个单元格等于n个默认单元格大小
9. cellpadding="20" 单元格内边距
10. cellspacing="10" 单元格外边距
11. 表单根据name收集信息。整合成JSON
12. 同组单选框name一样。
13. onreset
14. onsubmit
15. oninput 输入的时候一直的触发
16. onchang 当输入完点击其他位置时触发
17. <input type="hidden"/>隐藏文本框
18. json的key必须使用双引号

#### BOM

浏览器对象

##### 常用方法：

1. [window.open](http://www.w3school.com.cn/jsref/met_win_open.asp)(url,openMode,style);
   1. url
   2. \_blank, \_self ... iframeName
   3. 浏览器窗口的特征（宽， 高， 位置）
2. iframe src=""
   1. id, name, src, frameborder
3. **window**.**navigator**.**userAgent**--- 用户代理信息 （操作系统,浏览器版本,）
4. window.location.href 浏览器地址栏地址
5. window.location.hash
   1. 锚点可以用哈希值。

#### JSON

本身是一个数组或者是对象格式的字符串。

json.parse(JsonStr);//把json转换成js对象

json.stringfy(obj);//把字符串的json对象转换成json字符串。

##### perspective（景深）

##### 弹性盒模型

1. 父级：display：flex;

子级：

1. x轴居中：justify-content:center;
2. y轴居中:align-content:center;
3. 中间自动补空白(一个在开头一个在最后)：
4. justify-content:space-between;
5. 单独给元素定位弹性盒模型最后：flex-self:flex-end;
6. 用div来布局。
7. 竖着排：flex-wrap:wrap;flex-direction:column;
8. 旋转：transform:rotateY(90deg) 偏移：translateZ(300px);
9. 闭包：子函数访问父函数变量。
10. 用call就可以在回调函数中用this了。
11. @keyframes move{
    1. 100%{
       1. 结束位置参数：transform:rotateX(340deg) roatateY(340deg) roatateZ(360deg);
    2. }
12. }
13. animation: 10s move infination linear infinition;
14. linear-gradient(to bottom right, red, green);
15. -webkit-linear-gradient(to bottom, red, green); 报错，不要加to。
16. 函数名和变量名可以用中文。
17. 设置search会导致刷新，会导致后面代码无法执行。
18. a标签链接顺序： link，visited， hover，active;
19. window可视区尺寸（包括滚动条）
    1. window.innerHeight();
    2. window.innerWidth();
20. 能滚动的高度
    1. document.body.offsetHeight;
21. 计算浏览器滚动条宽度
    1. window.innerWidth.body.offsetWidth;(需要去掉margin) document.documentElement.offsetWidth;(有兼容性问题)
22. 滚动条滚动距离
    1. window.pageXOffset
    2. window.pageYOffset
23. 滚动设置滚动条位置
    1. window.scrollTo(x,y)
24. 回到顶部动画
25. */\*  
     \* obj: 元素  
     \* direct: 方向 'x' 或 'y'  
     \* time: 动画时间  
     \* 动画方式  
     \* fn: 回调函数  
     \* \*/***function** *scrollAnimation*(obj, direct, time, fx, fn){  
     obj.onclick = **function** () {  
     clearInterval(obj.**timer**);  
     **var** startTime = **new** Date().getTime();  
     **var** begin;  
     **if**(direct === **'y'**){  
     begin = **window**.**pageYOffset**;  
      
     }**else if**(direct === **'x'**){  
     begin = **window**.**pageXOffset**;  
     }**else**{  
     **return**;  
     }  
     obj.**timer** = setInterval(**function** () {  
     **var** t = **new** Date().getTime() - startTime;  
     **if**(t >= time){  
     t = time;  
     clearInterval(obj.**timer**);  
     }  
     **if**(direct === **'y'**){  
     **window**.scrollTo(**window**.**pageXOffset**,***Tween***[fx](t,begin,-begin,time));  
     **console**.log(***Tween***[fx](t,begin,begin,time));  
     }**else if**(direct === **'x'**){  
     **window**.scrollTo(***Tween***[fx](t,begin,-begin,time), **window**.**pageYOffset**);  
     }**else**{  
     **return**;  
     }  
     **if**(t>=time&&fn) fn();  
     },30)  
     }  
    }
26. 被内容撑起的高度。
    1. document.body.scrollHeight
27. 元素的可视区域
    1. e.clientHeight(当前元素不包括边框的宽度,不包括滚动条)。
28. 上下左右居中
    1. obj.left= (window.innerHeight – obj.offsetHeight)/2;
29. onresize 当浏览器尺寸变化时触发。
30. onscroll 当浏览器滚动时触发。
31. 轩邈xuanmiao首字母有圆形白底。

**h1**{  
 **display**: **inline-block**;  
 **z-index**: 0;  
 **position**: **relative**;  
 **margin**: 20**px** 0 0 9**px**;  
 **font-size**: 30**px**;  
 **color**: **#fff**;  
  
}  
**h1**:**before**{  
 **z-index**: -1;  
 **position**: **absolute**;  
 **content**: **''**;  
 **display**: **block**;  
 **width**: 24%;  
 **height**: 80%;  
 **background-color**: **#fff**;  
 **border-radius**: 100%;  
 **margin**: 6% 0 0 -8%;  
}  
**h1**::**first-letter** {  
 **margin-right**: 2%;  
 **color**: **#30B8C5**;  
 **font-size**: 120%;  
}

1. slice(0,-1)将最后一个字符移除。
2. input[type="button"]和input[type="text"]不一样长，设置box-sizing
3. 不一样高：

#### 浏览器事件

1. onfocus 获取焦点时触发
2. onblur 失去焦点时触发
3. oninput 输入框内容发生变化时触发
4. onchange 当失去焦点时检测内容是否发生变化，如果发生变化触发。（js更改不会触发）；

##### 键盘事件

1. onkeydown 键盘按下时触发。
2. onkeyup 键盘抬起时触发。
3. focus 让元素获取焦点。
4. blur 让元素失去焦点。
5. select 选中元素内容
6. 事件函数中，第一个参数是事件对象。
7. 鼠标位置（ev.clientX,ev.clientY）。
8. target 触发事件的目标元素。
9. keyCode 按下的键键码
   1. 小键盘1键的键码97
   2. 主键盘上面的1的键码49
   3. enter 13
   4. 左键 37
   5. 上键 38
   6. 右键 39
   7. 下键·· 40
   8. 空格 32
10. ctrlkey 代表ctrl是否被按下(Boolean)
11. shiftkey 代表shift是否被按下(Boolean)
12. altkey 代表alt是否被按下（Boolean）

##### 更改placeholder样式

1. 添加行间方法和样式onfocus="this.style.color='#000';this.value='';" style="color: #f00;"
2. 因为每个浏览器的CSS选择器都有所差异，所以需要针对每个浏览器做单独的设定。

::-webkit-input-placeholder { /\* WebKit browsers \*/

color: #999;

}

:-moz-placeholder { /\* Mozilla Firefox 4 to 18 \*/

color: #999;

}

::-moz-placeholder { /\* Mozilla Firefox 19+ \*/

color: #999;

}

:-ms-input-placeholder { /\* Internet Explorer 10+ \*/

color: #999;

}

input::-webkit-input-placeholder, textarea::-webkit-input-placeholder {

color: #666;

}

input:-moz-placeholder, textarea:-moz-placeholder {

color: #666;

}

input::-moz-placeholder, textarea::-moz-placeholder {

color: #666;

}

input:-ms-input-placeholder, textarea:-ms-input-placeholder {

color: #666;

}。

1. 问题及解决
   1. */\*Firefox在私有属性里面额外设置了边框和留白，去掉即可。\*/***button**::**-moz-focus-inner**,  
      **input**[**type**=**"button"**]::**-moz-focus-inner** {  
       **border**: **none**;  
       **padding**: 0;  
      }
2. demo
   1. 多一写padding会导致ie下出现问题

<!DOCTYPE **html**>  
<**html**>  
<**head**>  
 <**meta name="viewport" content="width=device-width"**>  
 <**title**>Demo Page</**title**>  
 <**style**>  
  
 .**btn-box** {  
 **margin**: 30**px**;  
 **border**: 1**px dashed #0094ff**;  
 }  
  
 .**standard-btn** {  
 **display**: **inline-block**;  
 **border**: **none**;  
 **padding**: 5**px** 10**px**;  
  
 **font**: 20**px**/30**px "Microsoft YaHei"**, **sans-erif**;  
 **vertical-align**: **middle**;  
 **text-decoration**: **none**;  
  
 **color**: **white**;  
 **background-color**: **#fe532d**;  
 }  
  
 */\*Firefox在私有属性里面额外设置了边框和留白，去掉即可。\*/* **button**::**-moz-focus-inner**,  
 **input**[**type**=**"button"**]::**-moz-focus-inner** {  
 **border**: **none**;  
 }  
  
 </**style**>  
</**head**>  
<**body**>  
<**div class="btn-box"**>  
 <**input class="standard-btn" type="text" placeholder="Input Text Tag"**>  
 <**div class="standard-btn"**>Div Tag</**div**>  
 <**a href="#" class="standard-btn"**>Anchor Tag</**a**>  
 <**input class="standard-btn" type="button" value="Input Button Tag"** >  
 <**span class="standard-btn"**>Span Tag Button</**span**>  
 <**button class="standard-btn"**>Normal Button Tag</**button**>  
</**div**>  
</**body**>  
</**html**>

* 1. <!DOCTYPE **html**>  
     <**html**>  
      <**head**>  
      <**meta charset="UTF-8"**>  
      <**title**></**title**>  
      <**link rel="stylesheet" type="text/css" href="../css/talk.css"**/>  
      <**style**>  
      **\***{  
      **padding**: 0;  
      **margin**: 0;  
      **border**: **none**;  
      **font-size**: 0;  
      **vertical-align**: **top**;  
      }  
      .**box**{  
      */\*position: relative;\*/* **height**: 40**px**;  
      **width**: 660**px**;  
      **margin**: 100**px auto**;  
      **border**: 2**px solid #000000**;  
      }  
      .**bbox**{  
      */\*position: absolute;\*/  
      /\*float: left;\*/* **display**: **inline-block**;  
      **width**: 100**px**;  
      **height**: 40**px**;  
      **background-color**: **red**;  
      **text-align**: **center**;  
      **line-height**: 40**px**;  
      **margin-right**: 10**px**;  
      **font-size**: 16**px**;  
      }  
      </**style**>  
      </**head**>  
      <**body**>  
      <**section class="box"**>  
      <**div class="bbox"**>test btn</**div**>  
      <**a href="" class="bbox"**>test btn</**a**>  
      <**input class="bbox" type="text" value="test btn"** />  
      <**input class="bbox" type="button" value="test btn"** />  
      <**span class="bbox"**>test btn</**span**>  
      <**button class="bbox"**>test btn</**button**>  
      </**section**>  
        
      *<!--<script src="../js/talk.js"></script>-->* </**body**>  
     </**html**>

1. chrome刷新最低为14ms.
2. 获取选中的option的值 select.options[select.selectedIndex].text
3. pageYoffset和documentElement.scrollTop和body.scrollTop区别：
   1. 有doctype
      1. firefox： 
      2. chrome： 
      3. ie： 
   2. 无doctype
      1. firefox: 
      2. chrome: 
      3. ie: 

### 事件捕获和事件冒泡

1. cancelBubble = true; 阻止事件冒泡
2. 使用on方式绑定的事件阻止冒泡使用ev.cancelBubble = true;
3. 使用addEventListener()添加的事件只能使用ev.stopPropagation();来组织冒泡。（但是该方法在低版本的ie中不支持）。
4. 使用addEvenetListener('click',fn，true);可以为一个事件添加多个方法。
   1. 第三个参数
      1. true：表示在捕获阶段添加事件。
      2. false：表示在冒泡阶段添加事件。
5. removeEventListener('click',fn，true)；
6. 只调用一次的函数
   1. **function** *test1*() {  
       **console**.log(**'1'**);  
       **document**.**documentElement**.removeEventListener(**'click'**,*test1*);  
      }  
      **document**.**documentElement**.addEventListener(**'click'**,*test1*);
7. [background-size](http://www.w3cplus.com/content/css3-background-size): auto || <length> || <percentage> || cover || contain
8. textarea
   1. 去除小三角：resize:none;
   2. 去除黄色的框：outline:none;
9. web标准是：只有跳转的链接才需要用pointer

# [CSS设置滚动条样式](https://segmentfault.com/a/1190000003708894)

1.

transition

1. demo:

a {

transition: background 0.5s ease-in,color 0.3s ease-out;

transition：transform .4s ease-in-out;

}

animation

@keyframes animationname {keyframes-selector {css-styles;}}

animationname 必需。定义动画的名称。

keyframes-selector 必需。定义动画的名称。

合法的值：

1. 0-100%

2. from（与 0% 相同）

3. to（与 100% 相同）

css-styles 必需。一个或多个合法的 CSS 样式属性。

demo: animation: gif 1.4s infinite linear;

@keyframes gif {

0% {

-webkit-transform: rotate(0deg);

transform: rotate(0deg);

}

100% {

-webkit-transform: rotate(360deg);

transform: rotate(360deg);

}

}

#### transform:语法

transform: none|transform-functions|initial|inherit;

<transform-function>表示一个或多个变换函数，以空格分开

demo: transform: rotate(45deg) scale(0.8,1.2) skew(60deg,-30deg);

## 拖拽

function *Drag*(id){  
 var box = document.getElementById(id);  
 box.addEventListener('mousedown',*down*);  
 function *down*(ev){  
 var disX = ev.clientX - this.offsetLeft;  
 var disY = ev.clientY - this.offsetTop;  
   
 document.addEventListener('mousemove',*move*);  
 document.addEventListener('mouseup',*up*);  
 function *up*(ev){  
 document.removeEventListener('mouseup',*up*);  
 document.removeEventListener('mousemove',*move*);  
 }  
 function *move*(ev){  
 box.style.left = ev.clientX - disX + 'px';   
 box.style.top = ev.clientY - disY + 'px';  
 }  
 ev.preventDefault();  
   
 }  
}

#### 禁止浏览器拖拽图片功能

采用jquery:

|  |  |
| --- | --- |
| 1 2 3 | $('body').on('dragstart','img',function(ev){     ev.preventDefault(); }) |

纯js:

|  |  |
| --- | --- |
| 1 2 3 4 5 | document.body.addEventListener('dragstart',function(e){     if(e.target.tagName && e.target.tagName.toLocaleLowerCase()=='img'){         e.preventDefault();     } },false) |

movementX和movementY

imgBox.onmousemove = function (ev) {

console.log(ev.movementX);

}

### [CSS属性值定义语法](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Value_definition_syntax)

|  | **名称** | **描述** | **示例** |
| --- | --- | --- | --- |
| **组合符号** | | | |
|  | 并置 | 各部分必须出现且按顺序出现 | solid <length> |
| && | “与”组合符 | 各部分必须出现，但可以不按顺序 | <length> && <string> |
| || | “或”组合符 | 各部分至少出现一个，可以不按顺序 | <'border-image-outset'> || <'border-image-slice'> |
| | | “互斥”组合符 | 各部分恰好出现一个 | smaller | small | normal | big | bigger |
| [ ] | 方括号 | 强调优先级 | bold [ thin && <length> ] |
| **数量符号** | | | |
|  | 无数量符号 | 恰好一次 | solid |
| \* | 星号 | 零次、一次或多次 | bold smaller\* |
| + | 加号 | 一次或多次 | bold smaller+ |
| ? | 问号 | 零次或一次（即可选） | bold smaller? |
| {A,B} | 大括号 | 至少A次，至多B次 | bold smaller{1,3} |
| # | 井号 | 一次或多次，但多次出现必须以逗号分隔 | bold smaller# |

transitionend 事件会在 CSS transition 结束后触发.

### 阻止默认事件

1. 可以通过ev.preventDefault()阻止默认事件
2. on方法添加的事件可以通过return false；阻止默认事件。
3. addEventListener不能通过return false来阻止默认事件。
4. 应用：右键菜单
5. 右键点击菜单事件oncontextmenu();
6. onmousedown 🡪 ontouchstart 手指触摸屏幕时触发
7. onmouseup 🡪 ontouchend 手指离开屏幕时触发
8. onmousemove 🡪 ontouchmove 手指在屏幕上移动时触发
9. changedTouches 屏幕上总共触摸的手指数
10. targetTouches 接触屏幕的手指数
11. touches 接触屏幕的手指数
12. 移动端a标签会有事件穿透。上面覆盖的元素的onclick会触发a标签的跳转。
13. radiusX，radiusY。
14. a标签内联元素。
15. createRadialGradient(xStart, yStart, radiusStart, xEnd, yEnd, radiusEnd)

|  |  |
| --- | --- |
| **参数** | **描述** |
| xStart, yStart | 开始圆的圆心的坐标。 |
| radiusStart | 开始圆的半径。 |
| xEnd, yEnd | 结束圆的圆心的坐标。 |
| radiusEnd | 结束圆的半径。 |

### [canvas](https://www.w3schools.com/graphics/canvas_reference.asp)

拖拽bug和碰撞检测。

1. 函数声明比变量声明更置顶

#### 鼠标滚动事件

1. **document**.onmousewheel
2. wheelDelta记录的滚动方向和滚动距离
   1. 正数向上滚动，负数向下滚动
   2. 每次滚动120
3. 火狐只能用 obj.addEventListener("DOMMouseScroll",function(){},false);
   1. detail 负数向上，整数向下
   2. 6，7交替
4. input禁止输入readonly="readonly";
5. 多个inline-block因为vertical-align 的对齐会导致设置margin-top失效。inline-block 的margin-top之间会互相影响
6. 解决：使用float
7. 无缝轮播思路
   1. 在回调函数中进行边界处理
   2. 第一张边界处理
      1. 是否等于0
      2. 等于0则跳转倒数第二张
      3. （obj.index = obj.childrenNumber - 2）因为长度本来就大一所以要减2不是减1。
   3. 最后一张边界处理
      1. 是否等于倒数第一张
      2. 等于则跳转到第二张
      3. obj.index = 1;
8. 左面固定宽度右边自适应的处理，右边设置margin-left为左面宽度，在设置float:left;
9. [BFC](http://www.jianshu.com/p/c9858d49a84e)(block formating context)
   1. 形成BFC的情况
      1. float的值不为none
      2. position的值为absolute或fixed
      3. overflow的值不为visible（默认就是visible，所以要改变必须显式设置）
      4. display的值为table-cell, table-caption, flex, inline-flex, inline-block,中的任何一个
      5. 根元素（其中除了inline-flex和inline-block的前后不会强制进行换行以外，其他都会三者都会使元素强制前后换行。）
   2. BFC应用要点
      1. BFC不会和浮动元素重叠
      2. BFC会计算内部浮动元素的高度
      3. 利用上面的父容器会计算内部浮动元素高度的特性，可以使父元素形成闭合浮动，从而让浮动不影响其他元素，达到清除浮动的目的。
      4. BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。
      5. BFC可以解决上下margin重叠的问题：
10. [三栏布局技巧](https://zhuanlan.zhihu.com/p/25070186)
    1. 流体布局  
       用float，main左右加margin  
       缺点：主要内容无法最先加载？
    2. BFC布局  
       左右float并设置margin，main用overflow形成BFC使间距生效  
       缺点：主要内容无法最先加载？
    3. 双飞翼布局  
       左右float，左设置margin：-100%，右设置margin: -220px, 主要内容外面套一个父级并设置左浮动和宽度100%，本身设置左右220px的外边距。  
       缺点：结构复杂，样式不易理解  
       优点，主题内容可以优先加载
    4. 圣杯布局  
       三者的父级左右设置margin，main设置float：left,宽度100%，左右设置float：left，position：relative，左设置left -220px,和margin-left：-100%,右设置right: -220px;margin-left: -200px;  
       缺点：样式不易理解  
       优点，主题内容可以优先加载,HTML 结构相对简单
    5. Flex 布局  
       父级设置display: flex,main设置flex-grow: 1;left设置order: -1;flex: 0 1 200px;,right设置flex: 0 1 200px;margin-left: 20px;  
       优点：很多  
       缺点：浏览器兼容
    6. Table 布局  
       父级设置display：table, 子集设置display:table-cell.  
       优点：很多  
       缺点：无法设置栏间距 ???
    7. 绝对定位布局  
       左右设置绝对定位，main设置左右margin.  
       优点：很多  
       缺点：无法设置栏间距 ???

### [BFC与IFC概念理解+布局规则+形成方法+用处](https://segmentfault.com/a/1190000009545742)

1. tabindex 全局属性 是个整数，表示元素（如果可聚焦）是否能够接受输入焦点。 如果它应该参与键盘序列导航，那么就是它的位置。它可以设为多种值：
   1. tabindex=负值 (通常是tabindex=“-1”)，表示元素是可聚焦的，但是不能通过键盘导航来访问到该元素，用JS做页面小组件内部键盘导航的时候非常有用。
   2. tabindex="0" ，表示元素是可聚焦的，并且可以通过键盘导航来聚焦到该元素，它的相对顺序是当前处于的DOM结构来决定的。
   3. tabindex=正值，表示元素是可聚焦的，并且可以通过键盘导航来访问到该元素；它的相对顺序按照tabindex 的数值递增而滞后获焦。如果多个元素拥有相同的 tabindex，它们的相对顺序按照他们在当前DOM中的先后顺序决定。
2. 根据键盘序列导航的顺序，值为 0 、非法值、或者没有 tabindex 值的元素应该放置在 tabindex 值为正值的元素后面。
3. 如果我们在 <div> 上设置了 tabindex 属性，它的子元素内容不能使用箭头键来滚动，除非我们在内容上也设置 tabindex。
4. 注:tabindex 的最大值不应超过 32767。如果没有指定，它的默认值为-1。
5. 绝对定位相对于相对定位父级pading的边界
6. 如何在chrome里实现小于12px的文字
   1. 先用js判断是否为chrome （至今还没听说有区分safari 和 chrome 的 css hack ）var isChrome = !!window.chrome;
   2. 再用-webkit-transform: scale( ) 缩小到合适值。
7. -webkit-text-size-adjust这属性现在的一般用处是防止iPhone在坚屏转向横屏时放大文字
8. 面包屑导航是一个次级导航系统，可以展示用户在网站或应用中的位置。这个术语来源于童话《糖果屋》，故事中的主人公在森林里用面包屑做标记来防止迷路。
   1. 优点
      1. 减少不必要的步骤
         1. 面包屑导航最实用的一点便是可以帮助用户更快地访问上级网页，无需借助浏览器的“返回”按钮和顶级导航栏。
      2. 占用空间少
         1. 面包屑导航仅由文本和链接组成的一行内容构成，因此占用的页面空间非常小。这样的好处是当内容过载时它的功能也不会受到影响。
      3. 用户体验很好
         1. 用户可能会忽略这个小控件，但是他们从来不会误解或在使用上遇到问题。
   2. 注意
      1. 不要用面包屑导航来替代顶级导航
      2. 不要给当前页面的导航文字添加链接
      3. 使用分隔符
      4. 设置合适的大小和padding属性
      5. 不要让它们看起来很突出
      6. 不要在移动端网站上使用
   3. 一句话总结
      1. 面包屑导航的宗旨是让用户可以更方便地访问你的网站。它看似简单，但却可以很好地优化用户体验，增加用户粘性。
9. ie版本曲风hack技巧

.ie6\_7\_8{

color:blue; /\*所有浏览器\*/

color:red\9; /\*IE8以及以下版本浏览器\*/

\*color:green; /\*IE7及其以下版本浏览器\*/

\_color:purple; /\*IE6浏览器\*/

}

1. [了解flex](http://www.w3cplus.com/blog/666.html)
2. storm调出terminal快捷键 alt+f12
3. 数据库🡪数据逻辑处理（拼装数据结构）🡪重构数据结构🡪重构数据结构（根据业务逻辑）🡪根据数据结构渲染页面
4. js设置有'-'的自定义属性用setAttribute。
5. 闭包理解
   1. **var *name*** = **"The Window"**;  
      **var *object*** = {  
       **name** : **"My Object"**,  
       getNameFunc : **function**(){  
       **return function**(){  
       **return this**.**name**;  
       };  
       }  
      };

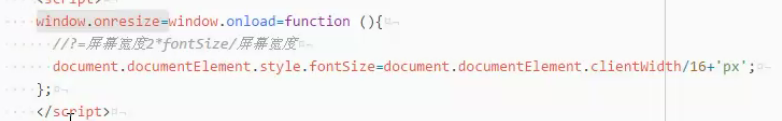
alert(***object***.getNameFunc()()); // The Window

* 1. **var *name*** = **"The Window"**;  
     **var *object*** = {  
      **name** : **"My Object"**,  
      getNameFunc : **function**(){  
      **var** that = **this**;  
      **return function**(){  
      **return** that.name;  
      };  
      }  
     };  
     alert(***object***.getNameFunc()());// My Object

1. var ev = ev||event; //兼容写法
2. 鼠标键位

左,中,右：

* 1. chrome： 0, 1, 2
  2. ie: 0, 1, 2
  3. 火狐: 0, 1, 2
  4. ie8 1, 4, 2

1. 移动端淘宝m站
   1. viewport,rem,响应式
   2. 懒加载，数据交互，webSocket,Ajax,jsonp,
   3. 封装
   4. 混合式App
      1. 自己写oc,java
      2. 现成框架PhoneGap(国外)、 APICloud（国内）
2. 先vue后angular
3. meta name= "viewport" content = "width=device-width"
4. 移动端设备自适应
5. window.onload = function(){

//?= 屏幕宽度2\*fontsize/屏幕宽度

document.documentElement.style.fontSize = documentElement.clientWidth/320/根子体+"px";

}

1. 
2. @media()条件{  
    代码  
   }
3. demo:
   1. @media(max-width:500){  
       .box{background:red}  
      }
4. 响应式：小型网站
5. 多套网站：大型网站
6. pc [www.taobao.com](http://www.taobao.com)
7. mobile m.taobao.com
8. touch事件demo
9. 
10. onmouseenter,onmouseleave在子集里面不会触发onmouseleave ,mouseenter是不冒泡的
11. onmouseover,onomouseout在子集里面里面会触发onmouseout事件。mouseover是会冒泡的
12. 在 IE9 之前的 IE 中，可以通过设置事件对象的 returnValue 属性为 false 来达到同样的效果,IE9 之前的IE不支持 stopPropagation() 方法
13. window.event?window.event.cancelBubble=true:event.stopPropagation()
14. phpstorm多行合并一行
    1. 按Ctrl+Shift+J快捷键，所有多余的字符(空格，引号和加号)被去掉了
15. **使用默认变量替代短路运算或条件**
    1. Bad:

function createMicrobrewery(name) {

const breweryName = name || 'Hipster Brew Co.'

// ...

}

* 1. Good:

function createMicrobrewery(breweryName = 'Hipster Brew Co.') {

// ...

}

* 1. Firefox 设置不再询问是否关闭多个标签页
     1. 在地址栏输入about:config?filter=browser.tabs.warnoncloseothertabs 然后双击，设置为false。
  2. js获取字符串最后一位
     1. str.charAt(str.length – 1)
     2. str.substr(str.length-1,1)
     3. spstr = str.split(“”);

spstr[spstr.length-1];

* + 1. str.replace(/^(.\*[n])\*.\*(.|n)$/g, "$2");
  1. 注意slice返回的是数组
  2. <div abc="1"><div/>,div.abc和div["abc"]无法获取abc的值。
  3. 非空对象（元素，｛｝，[]）可以设置属性和方法，如果没有设置过可以设置为undefined。
  4. 通过get方式和children获取元素是动态集合。
  5. 通过querySelectorAll获取元素是静态集合

1. **var a = 10;  
   function** *fn*() {  
    **var** b = ***c*** = 20;  
   }  
   *fn*();  
   delete **this**.a;  
     
   alert(**this**.a);  
   alert(**this**.b);  
   alert(**this**.c);

第一次弹10，第二次弹undefined，第三次弹20.

1. 自动挂在window上的属性无法用delete删除。（eg:var a = 10;）
2. 在对象上找不到属性不会报错，显示undefined。
3. 系统自带属性不能够被覆盖。设置无效。
4. **var *a*** = 10;  
   alert(***a***++ === ++***a***); //false 10 12  
   alert(***a***++ === ***a***++); //false 10 11  
   alert(++***a*** === ***a***++); //true 11 11
5. *只能将第一个元素和第三个元素的className更改为"new\_class".  
   /\*\*通过getElementsByClassName()获取到的元素集合temp是动态的，所以我们对元素任何的更改都会直接反映到对应的NodeList中；  
    刚开始temp长度为3，也就是max为3，这里i=0的时候，更改了temp[0]的className为"new\_class"，所以temp的长度马上发生变化，max变为2；  
    继续循环，i=1的时候，temp[1]实际上是没变化前的temp[2]。此时又更改了temp[1]的className为"new\_class"，所以temp的长度又发生变化，max变为1；  
    继续循环，i=2的时候，不满足条件，循环结束；  
    所以temp[0],temp[2]的className都变为"new\_class"，而temp[1]没改变\*\*/***var *temp*** = **document**.getElementsByClassName(**"class"**);  
   **for** (**var *i***=0, ***max***=***temp***.**length**; ***i***<***max***; ***i***++) {  
    ***temp***[***i***].**className** = **"new\_class"**;  
   }
   1. 函数嵌套函数。
   2. 子集去使用父级的环境变量。scope
   3. 好处：局部变量可以在父级一直使用。
6. 滚轮事件函数
   1. **返回值为true则滚轮向上，返回值为fasle滚轮向下。  
      function** *addWheel*(obj,fn){  
       */\*火狐\*/* **if**(obj.onmousewheel === **undefined**){  
       obj.addEventListener(**"DOMMouseScroll"**,*fn1*);  
       }**else**{  
       */\*其他浏览器\*/* obj.addEventListener(**"mousewheel"**,*fn1*);   
       }  
       **function** *fn1*(ev) {  
       **var** o = **true**;  
       */\*chrome\*/* **if**(ev.**wheelDelta**){  
       o = ev.**wheelDelta** > 0 ? **true** : **false**;  
       }**else**{  
       o = ev.**detail** < 0 ? **true** : **false**;  
       }  
       fn(o);  
       }  
      }
7. 到了目标阶段，目标元素绑定事件之后，执行时按照绑定的先后顺序来的，并不是先捕获后冒泡，
8. 事件模型从window开始，window结束。
9. 简述事件模型的过程：  
   当某个事件被触发的时候，那么会从window由上而下流向目标元素（捕获），到达目标元素之后，从目标元素由下而上流向window的整个过程（冒泡）成为事件模型也称为事件流。
10. [关于setInterval和setTImeout中的this指向问题](http://www.cnblogs.com/zsqos/p/6188835.html)
11. setInterval和setTimeout中传入函数时，函数中的this会指向window对象
12. 只有空字符串被split分割之后才会是空数组
13. 反转字符串

* var str = '12345';

Array.prototype.map.call(str, function(x) {

return x;

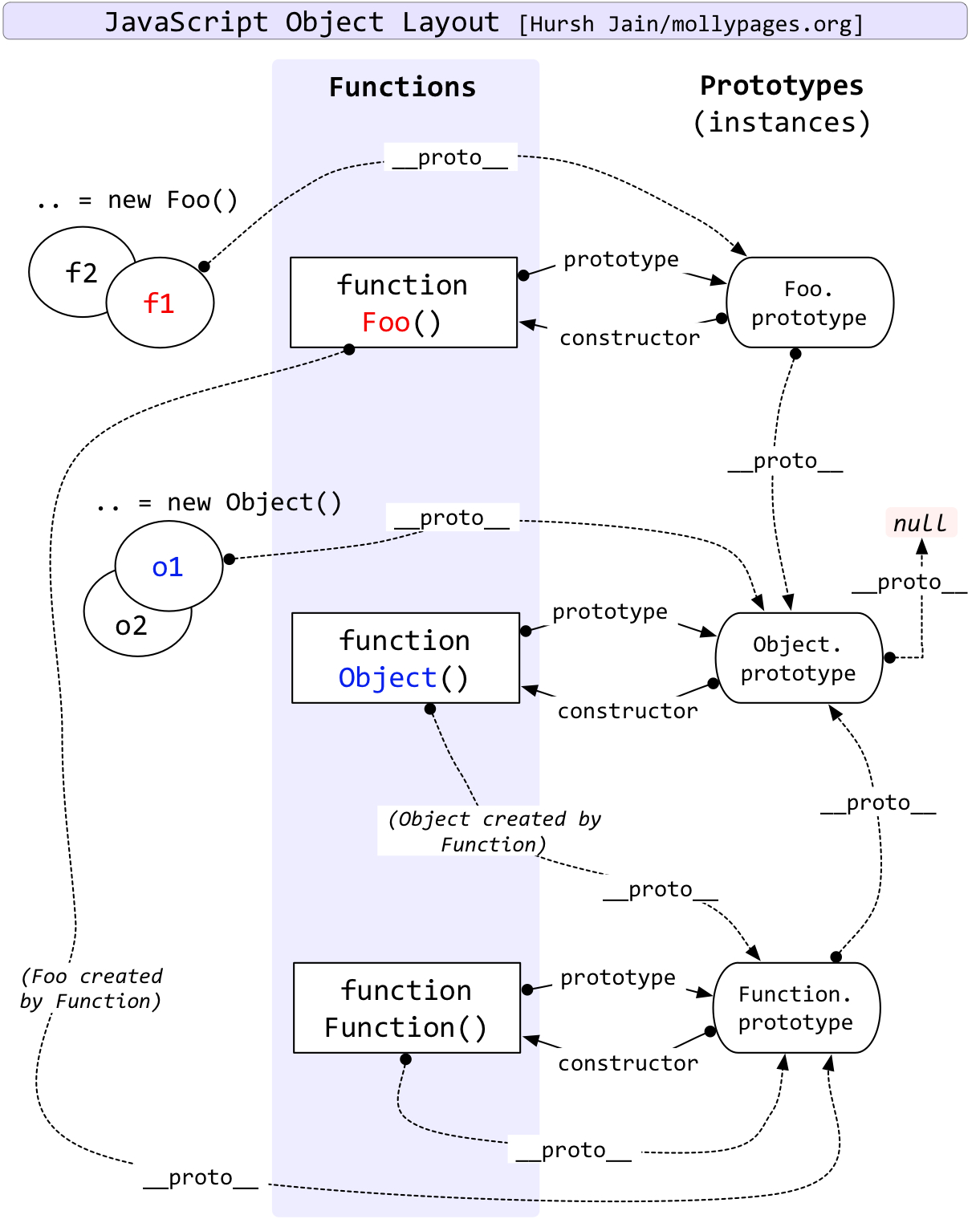
}).reverse().join('');

* 'xuanmiao'.split('').reverse().join('')

1. Number(" ")为0， isNaN(" ") 为false；
2. 找数字， if(!isNaN(parseInt(str[i])))//是数字
3. str.search() 查找字符串指定字符的位置,找不到返回-1。
4. str.match() 把正则匹配到的字符提取出来放到数组中。没有返回null
   1. 如果至匹配到一个，数组中会有index（匹配到的字符的索引）和input(要匹配的字符串)，但是length为1.
5. string.replace(字符串||正则，字符串||回调函数)。
   1. 回调函数参数
      1. 第一个参数，匹配到的字符
      2. 第二个参数，索引
      3. 第三个，整个字符串
   2. 第一个参数：匹配的字符串，第二个参数：替换的字符串
   3. 返回替换后的字符
      * 1. 应用：敏感词替换  
           string.replace(/敏感词|敏感词/g,function($0){  
           　　var temp = "";  
            for(var i=0; i < $0.length; i++){  
            temp += "\*";  
            }  
            return temp;  
           })
6. 最小值 -Infinity

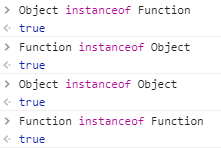
## 面向对象

1. 一种编程思想，把具有相同特征的代码归为一类，把细节描述挂在原型链上的编程方法。
2. new：函数一元运算符， 专门用来运算的符号。会返回一个对象。
3. prototype是函数的一个属性
4. prototype的值是一个对象。
5. \_\_proto\_\_原型链：实例化对象与构造函数的桥梁，只要是对象都有原型链。



1. 对象没有原型有原型链
2. 函数既有原型又有原型链
3. 构造函数的原型只给实例化对象使用
4. 对象找链 –-> 构造函数找原型 🡪 找找值下的链
5. \_\_proto\_\_ === .prototype //true
6. 实例化对象 p 🡪构造函数的原型🡪Object的原型🡪报错
7. Object.say 🡪 Object.\_\_proto\_\_ 🡪Function.prototype 🡪 Object.prototype 🡪 报错
8. 在函数是大多是window，方法上大多是寄主

### ES6

1. 类：  
   **class** Person{  
    constructor(name,age,sex){  
    **this**.**name** = name;  
    **this**.**age** = age;  
    **this**.**sex** = sex;  
    }  
    say(){  
    **console**.log(**this**.**name**);  
    }  
   }  
   **let** p = **new** Person(**"xuanmiao"**,0,**"man"**);  
   p.say();
2. 箭头函数：（箭头函数不能自执行） **let** *fn1* = (a,b) => a;  
   **let** *fn2* = (a,b) => a+b;  
   **let** *fn3* = (a,b,c) =>{  
    **let** c = a+b;  
    **return** a+b+c;  
   };  
   */\*箭头函数中的this跟父级走\*/*
   1. eg:  
      **let** arr = [2,2,3,4,5];  
      arr.forEach((e,i) =>{  
       **console**.log(e,i);  
      })
3. 非空对象才可以设置属性和方法。
4. 包装对象：
   1. 当去调用简单类型的属性或者方法的时候，系统会将简单类型转成复合类型，使用对象的属性和方法，调用完之后再销毁。
   2. 包装对象的属性只能读不能写。
5. alert会自动将参数转为String。
6. obj -> [Object Object] 第二个Object代表构造函数。
7. toString转进制 Number(数字).toString(进制数);
8. fn.call();可以调用函数
9. call:
   1. 是函数的一个方法，有若干个参数
   2. 第一个参数:改变this指向
   3. 第二个往后就是实参的个数。
10. apply：
    1. 参数一：改变this指向。
    2. 参数二：数组
11. bind（修改this指向，但是不会调用返回一个新的函数）：
    1. 修改this指向。（利用其返回的函数调用）
    2. 第二个及其后的参数，就是实参。
12. 判断数据类型为数组
    1. !!arr.push
    2. Object.prototype.toString.call(arr) === "[Object Array]"
13. **let *lis*** = **document**.getElementsByTagName(**'li'**);  
    ***lis*** = Array.**prototype**.slice.call(***lis***);  
      
    ***lis***.forEach((e,i)=>{  
     ele.onclick = **function** () {  
     alert(i);  
     }  
    });
14. forEach()
    1. 回调函数
    2. 改变this指向。
15. 找到数组中最小的数字是?
    1. 排序第0个
    2. Math.min.apply(null,arr);
16. constructor
    1. 实例化对象上的一个属性，指向构造函数。
17. instanceof
    1. 实例化对象 instanceof 构造函数（该实例化对象是否有构造函数构造出来的）
    2. 返回布尔值
    3. eg: （js不稳定性）
18. hasOwnProperty
    1. 是否是自身的属性（prototype上的不算）
    2. obj.hasOwnProperty('属性名')；
19. 严格模式下this 会变为undefined
20. 继承
    1. *// 属性继承（私有）（父类 F 子类S）--类式继承***function** *F*() {  
       }  
       **function** *S*(){  
        *F*.call(**this**);  
       }*//在S()实例化时 通过call将F中的this指向S()的实例化对象,该实例化对象可以通过地址获取到F的所有私有属性和方法*
    2. *//（公有）拷贝继承*
    3. *//（公有）原型继承***function** *PaoHui*(){}  
       *PaoHui*.**prototype** = *F*.**prototype**;  
       *S*.**prototype** = **new** *PaoHui*();
    4. *//子类私有继承父类私有***function** *S*(){F.call(**this**)}  
       *//继承公有并不改变this。  
       S*.**prototype** = Object.create(F.**prototype**,{**constructor**:{value:*S*}})
21. 深度克隆（deepClone）(如果对象中的内容为简单内容，那么直接复制即可，但若对象中有对象类型，因为赋值即赋址，故再次调用遍历，知道不是复合类型为止。)
    1. var arr2 = JSON.parse(JSON.stringify(arr));
       1. JSON.parse + JSON.stringify 这种方法的clone不会clone 对象内部的函数指针，假如a{ b:1 ,c:function(e){} } ，其中函数c是不会被复制的，clone之后就是 a{ b:1 }，
       2. JSON的克隆不会克隆NAN,undefined, function,Data
    2. function *deepClone*(o) {  
        if(typeof(o) !== "object" || o===null)return o;  
        let newObj = o instanceof Array ? [] : {}  
        for (let key of Object.keys(o)) {  
        newObj[key] = typeof o[key] === 'object' ? *deepClone*(o[key]): o[key]  
        }  
        return newObj  
       }
22. 图片模糊：用before覆盖filter:blur(3px);
23. document.activeElement属性用于管理DOM焦点，保存着当前获得焦点的元素
24. [注意]该属性IE浏览器不支持
25. [焦点事件和焦点元素，焦点管理](http://www.cnblogs.com/xiaohuochai/p/5874447.html)
26. ES6不能使用prototype修改原型。
27. let（声明的变量不存在window上，存在于一个不可见的块的作用域中，这个块理论上是Web页面中运行的所有JS代码的外层块。）
    1. let重复定义会报错
    2. const a = 1;  
       let ***a*** = 1;  
       **console**.log(**window**.a); //undefined
    3. **console**.log(***a***);//报错  
       let ***a*** = 1;
28. const

//不推荐这么干

const a = {a:'a'};

//重新赋值行不通

a = {a:'b'};

//黑科技

a.a = 'b'

1. obj中key可以是["num" + 变量]。
2. 组件网站 vux ant
3. Object.assign(obj,obj2);// 浅拷贝obj复制给obj2，返回一个对象。
4. 在使用 typeof 运算符时采用引用类型存储值会出现一个问题，无论引用的是什么类型的对象，它都返回 "object"。ECMAScript 引入了另一个 Java 运算符 instanceof 来解决这个问题。instanceof 运算符与 typeof 运算符相似，用于识别正在处理的对象的类型。
   1. eg:

console.log(Object instanceof Object);//true

console.log(Function instanceof Function);//true

console.log(Number instanceof Number);//false

console.log(String instanceof String);//false

console.log(Function instanceof Object);//true

console.log(Foo instanceof Function);//true

console.log(Foo instanceof Foo);//false

1. 自定义事件：
   1. 滚轮,wx.addEventListener('left)
   2. 在某个条件下触发（基于系统事件来判断）
   3. addEvent(document,'上滚',function(){})
   4. onclick🡪keydown + keyup
   5. 在某个对象上定义一个自定义属性（对象），用来存储“元素”上绑定的时间函数，当事件函数被触发时（基于系统事件并满足一定条件）循环事件名下的所有函数，依次调用。
   6. 订阅

{

'click':[funciton(){},function(){}]

'onmouse':[function(){}]

}

* 1. 发布：trigger(自定义事件名);//触发器
  2. eg：

//

function addEvent(obj,eventName,fn){

//如果是第一次绑定那么创建一个新的对象，如果之前已经绑定过了，用之前的对象。

obj.event = obj.event||{};

//第一次绑定obj.event这个对象创建一个key值为事件名，，value值为空数组。存在的话就还是这个对象//

obj.event[eventName] = obj.event[eventName] || {};

obj.event[eventName].push(fn);

}

//触发器

function trigger(obj,eventName){

//看看这个对象下是否绑定该事件名，没有返回

if(!obj.event[eventNmae])return;

//有则依次循环事件名对应的数组,依次调用。

obj.event[eventName].forEach((e,i) => {

e.call(obj);

});

}

let num = 0;

document.onclick = function(){

num ++;

setTimeout(function(){

num = 0;

},2000);

if(num === 3 ){

trigger(this,"点击三次");

num = 0;

}

}

1. obj.classList.add("read");

### AJAX

1. JS调用异步通讯组件并使用格式化的数据来更新web页面上的内容或操作过程被称作AJAX。
2. 判断是否为数组：
   1. Object.prototype.toString.call([]) === '[object Array]';
   2. [] instanceof Array
3. Object.assign() 方法用于将所有可枚举属性的值从一个或多个源对象复制到目标对象。它将返回目标对象。
4. ajax
   1. 创建一个ajax对象（new XMLHttpRequest）
   2. 填写地址 open()
      1. XMLHttpRequest对象的open()方法有3个参数，第一个参数指定是GET还是POST，第二个参数指定URL地址，第三个参数指定是否使用异步，默认是true，所以不用写。
      2. 因为浏览器的同源策略， URL的域名必须和当前页面完全一致（完全一致的意思是，域名要相同（www.example.com和example.com不同），协议要相同（http和https不同），端口号要相同（默认是:80端口，它和:8080就不同）。有的浏览器口子松一点，允许端口不同，大多数浏览器都会严格遵守这个限制。）
      3. 请求外域的方法
         1. 一是通过Flash插件发送HTTP请求
         2. 二是通过在同源域名下架设一个代理服务器来转发
         3. 第三种方式称为JSONP（只能用GET请求，并且要求返回JavaScript）（实际上是利用了浏览器允许跨域引用JavaScript资源）
         4. CORS（Cross-Origin Resource Sharing）
            1. 只要响应头Access-Control-Allow-Origin与本域相同，或者是\*，本次请求就可以成功。
            2. 简单请求包括GET、HEAD和POST（POST的Content-Type类型仅限application/x-www-form-urlencoded、multipart/form-data和text/plain）
   3. 发送给服务器 send()
      1. GET请求不需要参数，POST请求需要把body部分以字符串或者FormData对象传进去。
   4. 等待 onload
   5. 接收 responseText
5. 中文转码
   1. encodeURI（）中文转url编码
   2. decodeURI()url编码转中文
6. alert(ajax.status) //浏览器状态
7. post(ie会将中文自动转为url编码)
8. get和post区别：（[本质上](https://zhuanlan.zhihu.com/p/22536382)）
   1. GET把参数包含在URL中，POST通过request body传递参数。。
   2. GET产生的URL地址可以被Bookmark，而POST不可以。
   3. GET请求会被浏览器主动cache，而POST不会，除非手动设置。
   4. GET请求只能进行url编码，而POST支持多种编码方式。
   5. GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
   6. GET请求在URL中传送的参数是有长度限制的，而POST么有。
   7. GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。
9. get方式（）
   * 1. 通过地址栏进行数据的传输，.php?user=xuanmiao&password=0(open('get',url+字段拼接))
     2. 体积较小（大多数浏览器通常都会限制url长度在2K个字节，，而（大多数）服务器最多处理64K大小的url。）（要支持IE，则最大长度为2083byte，若只支持Chrome，则最大长度 8182byte）
     3. 相对不安全
     4. 在写中文的时候，IE下会把中文转成URI编码，会引起报错。（通过encodeURI转一下）
10. post方式(跟用户信息相关，较大体积的)
    * 1. 服务器进行传输（拼接的字段要在send中）
      2. 理论上没有大小限制，但是后端会做出限制
      3. 相对就安全一些
      4. 在send前加请求头ajax.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
11. [for in,Object.keys,Object.getOwnPropertyNames区别](http://yanhaijing.com/javascript/2015/05/09/diff-between-keys-getOwnPropertyNames-forin/)
    1. for in会输出 **自身以及原型链上 可枚举的属性** 。
    2. 如果仅想输出 **自身的 可枚举的属性** 可以借助 hasOwnProperty。可以过滤掉原型链上的属性。  
       for (var key in child) {   
        if (child.hasOwnProperty(key)) { console.log(key); }   
       }
    3. Object.keys是es5中新增的，用来获取对象 **自身 可枚举的属性键** 。
    4. Object.getOwnPropertyNames也是es5中新增的方法，用来获取对象 **自身的 全部属性名** 。
12. for of 遍历可枚举对象上的值
13. 查看某个对象属性：Object.getOwnPropertyNames(Object).sort().forEach(function (val) {console.log(val, '\n')});
14. defineProperty
    1. Object.defineProperty() 方法直接在一个对象上定义一个新属性，或者修改一个已经存在的属性， 并返回这个对象。

### [JavaScript中对象的属性和方法](http://yanhaijing.com/javascript/2015/05/08/member-of-object/)

## [JavaScript原生数组函数](http://yanhaijing.com/javascript/2014/01/17/fun-with-javascript-native-array-functions/)

1. link引入css时没有rel不生效
2. 为了统一集合类型，ES6标准引入了新的iterable类型，Array、Map和Set都属于iterable类型。
3. [Promise](http://www.alloyteam.com/2014/05/javascript-promise-mode/)
4. Array.from() :对伪数组或可迭代对象(包括arguments Array,Map,Set,String...)转换成数组对象。
5. delete
   1. delete 操作符用于删除对象的某个属性。
   2. 参数
      1. object 对象的名称，或计算结果为对象的表达式。
      2. property 要删除的属性。
   3. 返回值
      1. 对于所有情况都是true，除非属性是一个自己的不可配置属性，在这种情况下，非严格模式返回 false。
   4. 异常
      1. 在严格模式下，如果是对象自身的不可配置属性，会抛出Global\_objects/SyntaxError。
6. onreadystatechange五个阶段
   1. 0 🡪（未初始化）还没有调用send方法
   2. 1 🡪（载入）已调用send方法，正在发送请求
   3. 2 🡪（载入完成） send()方法执行完成，已经接收到
   4. 3 🡪（交互）正在解析响应内容
   5. 4 🡪（完成）响应内容解析完成，可以在客户端调用了。
7. onreadystatechange每一阶段都执行一次，如果把onreadystatechange放在放在send之前，那么可以监听到第二阶段，但是意义不大，因为4之前的步骤，每个阶段都拿不到数据，之后在第5步的时候前端才可以拿到数据。
8. ajax.readyState查看当前到了第几步。
9. eg:

if(ajax.readyState == 4){

console.log()  
}

1. HTTP状态码
   1. 1xx（临时响应）
   2. 2xx （成功）
   3. 3xx （重定向
   4. 4xx（请求错误）
   5. 5xx（服务器错误）
2. 常见的状态码
   1. 200 - 服务器成功返回网页
   2. 404 - 请求的网页不存在
   3. 503 - 服务不可用
3. eval
   1. 使用json格式不对转换
   2. eg:  
       let ***str*** = **"{'name':'xuanmiao'}"**;  
       ***str*** = eval(**'('** + ***str*** + **')'**);
   3. 代码注入攻击（有些数据会传给服务器）不安全
   4. 替代：str = new Function('','return' + str)();
4. [JSX](http://www.jianshu.com/p/7e872afeae42)（可以理解为在JS中编写与XML类似的语言）
   1. **目的**：是通过各种编译器将标记编译成标准的JS语言。
   2. 引入babel
      1. babel的JSX解析器会把JSX转化成JS语法
      2. babel会把es6语法转化成es5语法
   3. JSX 允许 HTML 与 JavaScript 的混写。
   4. <script> 标签的 type 属性为 text/babel，这是React 独有的 JSX 语法，跟 JavaScript 不兼容。凡是在页面中直接使用 JSX 的地方，都要加上 type="text/babel"。
   5. ReactDOM.render 是 React 的最基本方法，将模板转为HTML语言，并插入指定的 DOM 节点。
   6. 特点
      1. 类XML语法容易接受，结构清晰
      2. 增强JS语义
      3. 抽象程度高，屏蔽DOM操作，跨平台
      4. 代码模块化
   7. JSX本身就和XML语法类似，可以定义属性以及子元素。唯一特殊的是可以用大括号来加入JavaScript表达式。遇到 HTML 标签（以 < 开头），就用 HTML 规则解析；遇到代码块（以 { 开头），就用 JavaScript 规则解析。
   8. 在编写JSX时，在 { } 中不能使用语句（if语句、for语句等等），但可以使用求值表达式，这本身与JSX没有多大关系，是JS中的特性，它是会返回值的表达式。我们不能直接使用语句，但可以把语句包裹在函数求值表达式中运用。
   9. 没法在JSX中使用 if-else 语句， 不过可以采用三元操作表达式
5. 当前一次请求的地址和后一次请求的地址是一样的，那么可能会出现缓存（特别是IE下的get请求特别容易出现缓存）。
6. 上传文件
   1. 文件信息：file.files
   2. FormData对象
      1. formData.append('file',file.fies[0]) 异步上传一个二进制文件。
7. ajax.upload.onprogress=function(ev){} 进行监听数据上传进度的，服务器收到一次数据便会回调该函数。
   1. ev.loaded 当前传输的进度，ev.total总大小。
8. 跨域
   1. 同源策略
      1. 一种约定，他是浏览器最核心也是最基本的安全功能，如果缺少了同源策略那么浏览器的正常功能都会收到影响。
      2. 源：协议，端口，域名
      3. 协议：http,https,file,ftp
      4. 跨域：不同协议，端口，域名，
      5. 解决方案
         1. 服务器开权限header('Access-Control-Allow-Origin:\*');
         2. 服务器代理
            1. 服务器文件能访问别的域下的资源，如果服务器文件与自己的页面时同源，我就能访问别的域下的资源了。
         3. iframe
         4. jsonp
            1. JSON + Padding
            2. 函数名 + 括号 括号中带有数据
            3. 数据：格式

fn([])

fn({})

ajax数据没有函数名

* + - * 1. 原理:

首先：jsonp数据格式是一个函数名+括号的。

然后，全局需要定义一个与后端函数名一致的函数

最后，当需要数据的时候，创建一个script标签去请求，获取数据

1. arr.findeIndex(e => e === x); 找到条件成立的索引。
2. 判断图片是否加载成功：
   1. var img = new Image();
   2. var img = document.createElement('img');
   3. img.src = "";
   4. img.onload = function(){  
       alert('success');  
      }
   5. img.onerror = function(){  
       alert('fail');  
      }
3. console.time('计时器');  
   代码  
   console.timeEnd('计时器');  
   计算出time和timeEnd中间代码执行时间。
4. 将类数组转换为数组Array.from();
5. window.onresize 浏览器缩放时动态变化
6. dom（结构）加载完成后（图片加载之前）加载js：

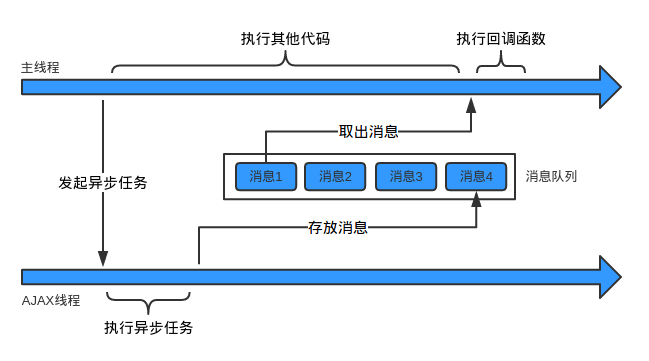
document.addEventListener("DOMContentLoaded",function(){  
 console.log("dom加载完成")  
})

1. cookies：
   1. 用来匹配后端的sessionID的。
   2. 从而保证用户只要在当前的浏览器访问页面，登录信息就不回丢失。
   3. cookies是在服务器运行的。
2. document.cookie = "key=value";（可读可写）
3. 设置cookie
   1. 每个域下大小限制有个数限制。（后端技术）
   2. 默认生命周期：到浏览器关闭为止。
   3. 设置生命周期： document.cookie = "name1 = hehe;expires=date"
4. */\*清除所有cookie\*/***function** *clearAllCookie*() {  
    **var** keys = **document**.**cookie**.match(/[^ =;]+(?=\=)/g);  
    **if**(keys) {  
    **for**(**var** i = keys.**length**; i--;)  
    **document**.**cookie** = keys[i] + **'=0;expires='** + **new** Date(0).toUTCString()  
    }  
   }  
     
   */\*设置cookie\*/***function** *setCookie*(key,val,time){  
    **if**(time){  
    **var** d = **new** Date();  
    d.setDate(d.getDate() + time);  
    **console**.log(d);  
    **document**.**cookie** = key+ **"="** + val + **";expires="** + d;  
    }**else**{  
    **document**.**cookie** = key + **"="** +val;  
    }  
   }  
     
   */\*获取cookie值\*/***function** *getCookie*(key){  
    **var** arr = **document**.**cookie**.split(**";"**);  
    **for**(**var** i=0,len=arr.**length**;i<len;i++){  
    **var** item = arr[i].**split**(**"="**);  
    **return** item[1];  
    }  
     
   }  
     
   */\*清除某个cookie\*/***function** *clearCookie*(key,val) {  
    *setCookie*(key,val,-1);  
   }
5. 设置cookie
6. 获取cookie
7. 如果要存储对象类型的数据，那么浏览器会默认调用toString 方法。  
   因此用JSON.stringify(obj);把对象转为字符串
8. localStorage
   1. 本地存储每个域名5mb
   2. 不清除，一直在
   3. localStorge.setItem();
   4. localStorge.geItem();
   5. localStorge.removeItem();
   6. localStorge.clear()
9. onstorage
   1. 本域名，本浏览器的数据发生变化时触发。
10. 查看数组中是否有某个数据includes(val);
11. filter找一个数组中想要的元素的。 相等要， 不等过滤。

## node.js

1. node.js特点
   1. 事件驱动
   2. 非阻塞式IO（异步）
   3. 单线程
2. 缺点
   1. 可靠性低
   2. 单进程，单线程，只支持单核CPU，不能充分的利用多核CPU服务器。（不适合高cpu运算场景）
3. Node.js应用场景
   1. 单页面，多AJAX请求应用，如Gmail
   2. 流式数据，如实时文件上传系统transloadit
   3. 准实时应用系统，如聊天系统，微博系统
4. 不适合场景
   * 1. CPU使用率较重、IO使用率较轻的应用-如视频编码、人工智能，
5. node支持多进程的方式
   1. 开启多个进程，每个进程绑定不同的端口，用反向代理服务器如nginx做负载均衡。
   2. 多进程绑定在同一个端口侦听。在Node.js中，提供了进程间发送“文件句柄” 的功能。

### 同步与异步：

1. 同步和异步关注的是消息通信机制。
2. 同步：如果在函数A返回的时候，调用者就能够得到预期结果(即拿到了预期的返回值或者看到了预期的效果)，那么这个函数就是同步的。  
    异步：如果在函数A返回的时候，调用者还不能够得到预期结果，而是需要在将来通过一定的手段得到，那么这个函数就是异步的。  
    同步可以保证顺序一致，但是容易导致阻塞；异步可以解决阻塞问题，但是会改变顺序性。改变顺序性其实也没有什么大不了的，只不过让程序变得稍微难理解了一些 :)
3. 事件循环机制：  
   
4. 阻塞和非阻塞关注的是程序在等待调用结果（消息，返回值）时的状态.

### 创建服务器



1. 创建文件服务器
2. 文件操作
   1. 写入
   2. 读取

fs.readFile(**'1.txt'**, **function**(err,data){  
 **if**(err){  
 **console**.log(err);  
 }**else**{  
 **console**.log(data.toString());  
 }  
});

1. 解构赋值:
   1. 数组解构（左右都是数组）
      1. var a = 10;

var b = 20;

var [b,a] = [a,b];

a // 20

b // 10

* + 1. let [,,c] = [1,[2],3]; //c为3
    2. let [x = 30, y] = [, 20]; // x为30
    3. let [x = f()] = [1];
    4. let [x=y,y =1] = []; //报错
    5. var [x=y,y =1] = []; // console.log(x) // undefined
  1. 对象解构
     1. var {width,height,left,top} = obj;
     2. let {foo:f} = {foo:1} //为foo取了一个别名，foo没了，访问foo报错
     3. var{x:y=3} = {x:5}; //x 🡺 报错

1. 扩展运算符
   1. …[1,2,3] 🡺 1,2,3
2. queryString将search转为对象。
3. response.end()之后不能再response.write()。
4. npm(Node Package Manager,node包管理器)是node.js默认的软件包管理系统 。
5. node.js是一个能够在服务器端运行JavaScript的开发源代码、跨平台JavaScript运行环境。

## express

### 简介

Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。

### 特性

1. 可以设置中间件来响应 HTTP 请求。
2. 定义了路由表用于执行不同的 HTTP 请求动作。
3. 可以通过向模板传递参数来动态渲染 HTML 页面。

### 安装

1. 测试各个镜像站的速度：nrm test
2. 切换：nrm use cnpm
3. npm init
4. npm i express -S 或 npm install express –save
5. 采用 --save 选项安装的 Node 模块已添加到 package.json 文件中的 dependencies 列表。 今后运行 app 目录中的 npm install 将自动安装依赖项列表中的模块。
6. 路由：通过访问指定地址来访问数据
7. import命令具有提升效果，会提升到整个模块的头部，首先执行。
8. 把能转为ES5的代码转化为ES5代码
9. Babel是一个广泛使用的转码器，可以将ES6代码转为ES5代码，从而在现有环境执行。
10. babel-cli (commend line interface)
11. npm init -y //package.json初始化默认配置
12. webpack-dev-server
13. mix-blend-mode CSS 属性描述了元素的内容应该与元素的直系父元素的内容和元素的背景如何混合。
14. 某些组件有自己的行为，比如input，你输入内容也可以使页面变化，但是不受react控制，如果给input的onChang事件函数重新赋值一个函数，这样就受到react控制，并通过setState来改变页面状态
15. this的本质就是：this跟作用域无关的，只跟执行上下文有关。
16. map方法，接收一个回调函数，回调函数里面接收的参数是：（元素，索引，数组），回调函数的返回值会替代掉原先的元素，返回一个新数组，原先的数组并没有变化。
17. 当把数组卸载jsx中是，react会自动展开它。
18. 如果直接将元素数组放在jsx中，jsx会警告，添加唯一的key来消除警告。原因：唯一的key用于virtualDOM比较，提高react性能。
19. Windows的cmd命令查询指定端口占用的进程并关闭

以端口8080为例：

* 1. 1.查找对应的端口占用的进程：netstat  -aon|findstr  "8080"    ，找到占用8080端口对应的程序的PID号：
  2. 2.根据PID号找到对应的程序：tasklist|findstr "PID号"    ，找到对应的程序名
  3. 3.结束该进程：taskkill /f /t /im 程序名

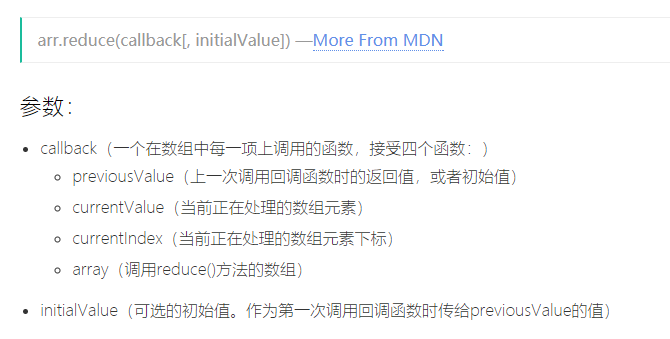
1. +new Data()获取时间戳
2. window.c = function(){console.log(1)}  
   function *c*(){console.log(2)}  
   *c*(); //20
3. 去除滚动条：overflow: hidden;
4. 上传图片的数据在file中的第0个
5. files.change()

#### html5的拖拽

1. ondragstart 事件：当拖拽元素开始被拖拽的时候触发的事件，此事件作用在被拖曳元素上
2. ondragenter 事件：当拖拽元素进入目标元素的时候触发的事件，此事件作用在目标元素上
3. ondragleave 事件：当拖拽元素移出目标元素的时候触发的事件，此事件作用在目标元素上
4. ondragover 事件：拖拽元素在目标元素上移动的时候触发的事件，此事件作用在目标元素上
5. ondrop 事件：被拖拽的元素在目标元素上同时鼠标放开触发的事件，此事件作用在目标元素上
6. ondragend 事件：当拖拽完成后触发的事件，此事件作用在被拖拽元素上
7. 注意：
8. 如果要使用放开的事件，那么前提是在document上把ondragover的默认行为阻止掉。

#### jquery上传图片

#### 移动端

1. 难点在适配
2. 事件不一样
   1. ontouchstart 按下
   2. ontouchmove 移动
   3. ontouchend 抬起
3. 使用事件绑定来解决模拟器上的事件不起作用的问题
4. 使用PC上的事件比移动端上的事件慢一点（300左右）
5. 在移动端上不要使用PC事件，统一使用touch事件
6. 移动端的move和end事件不用套在start里面
7. 在document上禁止缩放和橡皮筋效果和默认的选中框ev.preventDeafault()，这时input不能被选中，在input上阻止冒泡(ev.stopPropagation)
8. 交互： css3 h5 canvas， 后台管理目前最热，
9. 事件对象
   1. changedTouches
   2. targetTouches
   3. touches
10. changedTouches:在元素身上同时出现的触摸点
11. targetTouches：当前对象上触摸点的列表，可以是同时出现的也可以不是同时出现的。
12. touches：当前屏幕上触摸点的列表
13. 一般使用changeTouches[0].pageX
14. chrome 获取addEventListener添加的事件处理函数getEventListener
15. 什么时候会出现undefined
    1. 函数没有返回值
    2. 定义变量未赋值
    3. 对象无指定属性
16. map,forEach, fliter都可以遍历
17. [reduce](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce)(function(item1, item2){return item1 + item2}, 100)
    1. item1：上一次的返回值，item2: 下一项
    2. 如果有初始值，item1是初始值，没有item1则是第一项
    3. 100为初始值
    4. eg: 累加  
       arr.reduce()
    5. 
18. js出现的单词中要么是变量，要么是字符串，否则报错
19. 去除指定元素 arr.splice(i,1)
20. 属性：
    1. writeable 默认false， 可否被修改
    2. configurable 可否
    3. emumable 可否
    4. iterater 可枚举
21. 函数的调用方式
    1. 直接调用this => window
    2. 事件触发调用 this=> 触发事件的元素
    3. new 调用 this=> 通过函数创建的实例
    4. 定时器调用 this=>window
    5. call,apply调用 call,apply第一个参数
22. 严格模式下，this在非严格模式下指向window的全部指向undefined
23. 箭头函数中this，绑定在定义的时候的作用域上。this的值是这个箭头函数所在的作用域中的this
24. window.onhashchange 当哈希变化时触发
25. 数据持久化-读取（localStorage）
    1. let data = JSON.parse(localStorage.getItem('list')) || []
26. 数据持久化-存储
    1. localStorage.setItem('list', JSON.stringify(this.list))
27. vue 深度监听数据  
    watch: {

list: {

deep: true,

handler: function () {

// 数据持久化-存储

localStorage.setItem('list', JSON.stringify(this.list))

}

}

}

1. promise
   1. Promise是异步编程的一种解决方案
   2. 语法：
      1. new Promise(function(){  
          异步操作  
         })
   3. 三种状态：
      1. peding
      2. fulfilled
      3. rejected
2. 多页应用
   1. 每次切换页面的时候 重新加载资源
3. 单页应用
   1. 只有一个页面，切换链接的时候并不会重新刷新页面，不会重复的加载页面
   2. 缺点：
      1. 首次加载时间过长(代码切分)
      2. 每次前进后退按钮都会重新加载资源
4. 带问号search值 , 不带问号queryString

### 打包上线

1. 服务器的跟路径: /www
2. 上线的项目xx: /www/xx/
   1. xx文件下有项目文件
      1. index.html
      2. /static/
         1. /css/
         2. /js/
   2. 在使用npm run build打包的时候，修改两处
      1. /config/index.js 文件中

build

asssetsPublicPath: "/xx/"

* + 1. 修改路由访问的根路径
       1. /router

index.js文件中

配置项中写上：

base: '/xx/'

* 1. 在访问项目，刷新后出现404
  2. 打开apache的配置 http.conf
     1. 打开模块mod-rewrite （去掉#）
  3. 在根目录上/www目录下创建文件.htaccess文件

内容：

<IfModule mod\_rewrite.c>

RewriteEngine On

RewriteBase /

RewriteRule ^index\.html$ - [L]

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule . /xx/index.html [L]

</IfModule>

* 1. 注意：xx为项目所在目录

### 如何拿接口

ajax服务器代理

const request = require('request')  
const express = require('express')  
const app = express();  
  
app.use(function (req, res, next) {  
 res.header("Access-Control-Allow-Origin", "\*");  
 res.header('Access-Control-Allow-Methods', 'PUT, GET, POST, DELETE, OPTIONS');  
 res.header("Access-Control-Allow-Headers", "X-Requested-With");  
 res.header('Access-Control-Allow-Headers', 'Content-Type');  
 next();  
})  
  
app.get('/list', function (req, res) {  
 let url = 'https://www.smartisan.com/product/spus?page\_size=20&category\_id=60&page=1&sort=sort'  
  
 request({  
 url: url,  
 gzip: true  
 }, function (err, response, body) {  
 //err 错误信息  
 // reponse 响应头  
 // body响应的数据  
 res.send(body)  
 })  
})  
  
app.listen(3000)

1. 将数据和模板结合在一起，然后交给浏览器渲染
2. 前端套模板
   1. 优点：可以前后端分离，后端只提供接口，
   2. 缺点：大量套页面的逻辑放在前端，渲染的比较慢
3. 后端套模板：
   1. 优点：请求的页面已经套好数据了，渲染速度非常快
   2. 缺点：前后职责划分不清晰

### 精读前后端渲染之争

1. 前端渲染的优势
   1. 局部刷新。无需每次都进行完整页面请求
   2. 懒加载。如在页面初始时只加载可视区域内的数据，滚动后rp加载其它数据，可以通过 react-lazyload 实现
   3. 富交互。使用 JS 实现各种酷炫效果
   4. 节约服务器成本。省电省钱，JS 支持 CDN 部署，且部署极其简单，只需要服务器支持静态文件即可
   5. 天生的关注分离设计。服务器来访问数据库提供接口，JS 只关注数据获取和展现
   6. JS 一次学习，到处使用。可以用来开发 Web、Serve、Mobile、Desktop 类型的应用
2. 前端渲染的劣势
   1. 主要面临的问题有两个 SEO、首屏性能
3. 后端渲染的优势
   1. 首屏性能强
   2. 方便SEO优化
   3. 服务端渲染不用关心浏览器兼容性问题（随着浏览器发展，这个优点逐渐消失）
4. 判断是否滚动到文档底部

//滚动条在Y轴上的滚动距离  
export function *getScrollTop*() {  
 let bodyScrollTop = 0,  
 documentScrollTop = 0;  
 if (document.body) {  
 bodyScrollTop = document.body.scrollTop;  
 }  
 if (document.documentElement) {  
 documentScrollTop = document.documentElement.scrollTop;  
 }  
 return (bodyScrollTop - documentScrollTop > 0) ?  
 bodyScrollTop :  
 documentScrollTop;  
}  
//文档的总高度  
export function *getScrollHeight*() {  
 return document.documentElement.scrollTop == 0 ?  
 document.body.scrollHeight :  
 document.documentElement.scrollHeight  
}  
//浏览器视口的高度  
export const *getWindowHeight* = () => (document.documentElement.scrollTop == 0 ?  
 document.body.clientHeight :  
 document.documentElement.clientHeight);  
  
//判断鼠标是否滚动到文档底部  
export const *isScrollToDocumentBottom*=()=> {  
 //浏览器视口的高度  
 const windowHeight = *getWindowHeight*();  
 const scrollTop = *getScrollTop*() + windowHeight; // scrollTop视口底部距离文档顶部的距离 = getScrollTop()滚动条在Y轴上的滚动距离+ windowHeight() 浏览器视口的高度  
 //文档的总高度  
 return scrollTop === *getScrollHeight*()  
}

1. 移动端,响应式,rem,dpi

var dpr = 1 / window.devicePixelRatio;  
document.write('<meta name="viewport" content="width=device-width,initial-scale=' + dpr + ',minimum-scale=' + dpr + ',maximum-scale=' + dpr + ',user-scalable=no" />');  
var w = document.documentElement.clientWidth / 10;  
var html = document.getElementsByTagName('html')[0];  
html.style.fontSize = w + 'px';

1. phpstorm在行和列选择模式之间切换 Shift+Alt+Insert