**不凡学院笔记**

**页面布局注意事项**

1. 左边布局用浮动,右边布局用定位(需要2个以上的元素在同一行的情况下)
2. 在布局前要讲 a,ul 内外边距等标签的准备设置好(a:text-decoration 超链接下划线消失  
   ul: list-style ,ul前置图标消失  
   \*{ margin:0px;  
   padding:0px;} 这些现阶段最基本的一些设置,还有外部类的引用等等)
3. 在给元素进行命名时,要规范化尽量使用一个系列,哪怕最后实在想不出叫什么了也要使得命名有理可查
4. 元素的命名在编写class类时要注意名称有误无错
5. 对于页面要有版心,要有版心,否则会造成页面的缩放或者放大时不协调
6. 对于页面的居中设置要做好,从最基本的文字居中到元素的居中等
7. 对于注册/登录,或者一图片一文字的样式,尽量使用 div 例如 ()QQ图标) QQ
8. 一个元素可以有多个类名,便于使用公共类,便于使用公共类时进行微调修改
9. 在页面布局前要做好对页面整体了解,列:页面分几个模块,模块的大小,图片是否整备,页面的大小,版心的大小
10. 对于触发事件,一个个来,不要急 (:have 当鼠标放在XX上时发生)
11. 页面布局时,从上往下写,不要跳着写
12. 切记心浮气躁
13. div虽好,但不要滥用
14. 在页面布局时推荐使用 浮动,定位,快尽量少用
15. 当页面 进行margin时发生整体的位移,注意高度坍塌的发生,可用overflow尝试解决
16. 在页面布局的模块设计时,可以先用背景颜色将模块整体勾画出来,再逐级设计
17. 某些相似的页面可以重复使用类
18. 在页面设计时整体大小和设计稿尽量不要有误差
19. 注意细节的把控,列如a标签,:haver事件等
20. 对于img的图片要注意../ 是否要加几个,一个../是返回上一级
21. div之间有空隙时,检查是否是因为换行符的原因导致,或使用font-size:0 来解决
22. cursor:pointer 将鼠标变为手的形状
23. border-radiao:50% 变成椭圆

**DOm操作心得**

* 先找到发生事件的元素
* 将相应的元素doucument到script中
* 在script中进行相应的操作
* 将相应的方法进行函数封装优化(通常是先写好无错误在进行封装优化)
* **对于图片轮播切换时要有计数器(用来查看第几张图片)**
* 活动this. :this.可以代表当前的函数/对象等,
* dom说穿了就是if,for的不断应用,数组函数的互相帮助
* 活用true和false,元素不为真即为假
* 不可以图省事而粘贴复制,因为容易出更多的问题,老老实实手打
* 对于打元素的名字时要细心
* 对于for中i的值固定时可以取元素的其他类名进行代替,或者给元素声明一个新的类名,然后就可以进行正常操作
* 活用排它方法,remove('active'),例:
* <!-- 元素声明新的类和排他,增加实例 -->
* for (var i = 0; i < li1.length; i++) {
* li1[i].index = i;
* li1[i].onclick = function () {
* var index = this.index;
* console.log(index);
* for (var j = 0; j < li1.length; j++) {
* li1[j].classList.remove('active');
* }
* li1[index].classList.add('active');
* innerHTML将文本变为标签
* innerTEXT将标签变为文本
* 对于单选,反选,全选等要先思考它们的判断方式,列出大概率,在进行函数封装
* 对于一些if判断中没有相应的指示物(变量),应当自己声明一个值,来充当桥梁的作用
* 对于某些for中的i=固定值,可以视情况赋值给他新的属性
* 对于图片动态生成这种常用的方法要多练习,例
* initImg(ul,imgarr);
* function initImg (el,arr){
* var str ='';
* for(var i = 0; i<arr.length; i++){
* str+='<li style="background-image: url(img/' + arr[i] +')"></li>';
* }
* el.innerHTML = str;
* }
* 在dom中推荐给元素加入新的属性而不是在自己在JS中添加(onclick方法)
* dom操作以简单省事方便为主旨,可以在css样式中给class添加类名或给元素添加新属性为而不是在script中添加
* 对于计时/倒计时等,以倒计时为例
* \*{
* margin: 0px;
* padding: 0px;
* }
* .h1 {
* width: 500px;
* height: 100px;
* margin: 100px auto 50px auto;
* }
* .surplus {
* width: 1000px;
* height: 500px;
* margin: 0 auto;
* text-align: center;
* line-height: 12px;
* font-size: 30px;
* color: orange;
* }
* strong{
* background-color: orange;
* padding: 0 10px;
* color: white;
* border-radius: 4px;
* }
* </style>
* <h1 class="h1">距离培训毕业还有:</h1>
* <div class="surplus">
* <span><span class="day">00</span>天</span>
* <strong><span class="hour">00</span>时</strong>
* <strong><span class="branch">00</span>分</strong>
* <strong><span class="second">00</span>秒</strong>
* </div>
* <script>
* var targetTime = new Date('2019.11.10');
* var dayEl = document.querySelector('.day');
* var hourEl = document.querySelector('.hour');
* var minEl = document.querySelector('.branch');
* var secondEl = document.querySelector('.second');
* // document.querySelector为类需在后面➕.;小数点
* setInterval(function(){
* var presentTime = new Date();
* // console.log(presentTime);
* var difference = targetTime-presentTime;//剩余时间
* // console.log(difference);
* var DAY = 1000\*60\*60\*24;
* var HOUR = 1000\*60\*60;
* var MIN = 1000\*60;
* var SECOND =1000;
* //天时分秒的大小,1千为1秒
* var day = Math.floor(difference / DAY);
* // console.log(day);
* var hour = Math.floor(difference%DAY / HOUR);
* var min = Math.floor(difference%HOUR / MIN);
* var second = Math.floor(difference%MIN / SECOND);
* //计算相差的时间
* dayEl.innerText =digits(day);
* hourEl.innerText = digits(hour);
* minEl.innerText = digits(min);
* secondEl.innerText = digits(second);
* //将所求结果转换成文本,将上面的00替换为转换成的文本
* },1000)
* function digits(num){
* return num >10 ? num : '0'+num ;
* }
* //判断00,是否是2位数,是的话不用改,不是的话要在前面+0保持界面的完整
* </script>
* 写的时候一定要细心细心再细心
* 如果有差错可以使用断点调试和console.log来进行查找错误,但通常推荐使用**console.log**来进行查找错误

**JQuery注意事项**

* 无论页面怎样复杂,都不要急,将页面划分成一个个小部分和血HTML页面一样一步步来
* 在写页面动态时心里要有逻辑,想好先做什么,在做什么,如果实在记不住可以写在纸上
* 写JQ是先将效果呈现出来,先不管美丑,写出来后在修改
* JQ的应用在页面上的操作,说白了便是页面的元素和鼠标的互动
* 注意JQ和DOM写法上的差别,比如function,JQ是写在()里面,而DOM则是写在=后
* 活用变量来代替元素,找下标的操作
* 注意回调函数内部的this发生变化(充满了不确定性,不推荐直接使用this,可以和dom一样用一个标签的其他属性来代替)
  + 在JQ中少使用函数点击事件方法,在CSS元素中使用,onclick = "xxx(this)"来在下面进行函数this下面
* 动画效果使用时第一参数必须有,第二三参数看情况而定
* 在查找时,元素的下标有些已经自动给了,但还有一些元素无法找到,可以使用父元素parent来解决,父元素可以多次使用,比如在一个ul中的li内部还有一个li1,那么,li1.parent = li,li1.parent.parent = ul

**Git操作流程**

1. git init 初始化本地仓库
2. git status 查看文件状态 (红色就是没有添加到暂存区,绿色就是已经添加到了暂存区,绿色红色都没有(目前工作区文件都已提交到了本地仓库))
3. git add 文件名 (将文件添加到 暂存区)/git add . (将所有文件添加到 暂存区)
4. git commit -m'提交信息' 暂存区内容 提交到本地仓库
5. git remote add origin + 网址(git库网址)
6. git push origin master(上传,第一次需要进行输入账号密码)
7. 刷新网页  
   (对于第一次建立2可以不进行,git界面一定要在你要上传的文件夹中打开)
8. GitHub建立新的库,不识别中文需要进行英文

**常见问题**

**1. 权重问题**

* 行内样式(1000)>id选择器(100)>class选择器(10)>标签选择呢器(1)>通配符/继承属性(0)
* 选择器选择时,就近使用,权重高的优先使用
* 比如插入图片,我在li用background(背景图片)代表着我把所有的样式都设置好了,若是我想在li的类里在对background设置则是没有效果,所以若想对background设置则在li使用background-image(background的简写),那么就可以在li的类里对背景进行设置

**2. 权重问题**

* 对于JQ,DOM来说一定要分清自己使用时是类名还是标签名,是类名要在前面加点,不是类名千万不要加.

**3.定时器**

* setTimeout();延时执行一次
* setInterval();循环执行多次,每隔X毫秒执行一次
* 关闭循环计时:setTimeout(c=>{  
  clearTimeout(变量);},时间),先清除:setTimeout(c)再清除 clearTimeout(),C为循环计时的名称变量
* clearTimeout();清除计时器

**4.children**

* children:DOM中遍历下面子元素,里面的所有元素

**HTML基础**

**2019.7.23-2019.7.26**

**1. html:超文本标记语言**

**2. head:给浏览器识别用,辅助浏览页面(不会再浏览器中显示)**

**3. meta:设置网页的元数据**

**4. title:设置页面的标题,在浏览器头部tab中显示**

**5. body:设置网页的主体,网页的所有内容都在body中**

**6. h1-h6:标题标签,逐级变小**

**7. br换行**

**8: hr:水平线换行**

**9. 路径:**

* 相对路径:同一级目录,直接引用(常用)
* 绝对路径:将路径信息完整的填入(不推荐)

**10. a:超链接**

* href:跳转地址, {电子邮件: href:"mailto:邮件地址"(默认打开电脑电子邮件客户端)}
* target:self/blank 当前页面跳转/新的页面跳转
* alt:不会再图片中显示,会在图片无法显示时进行提示(提示内容为alt值)
* 在页面某处设置id,可以使用a标签到达设置的位置即一键到底or一键到顶的实现

**11. 图片使用关系(png,jpg)**

* 效果一致:使用体积小的
* 效果不一致:使用效果好的

**12. 列表**

* ul:无序列表,list-style:none(清除默认图标)
* ol:有序列表,dl,dt,dd:自定义列表

**13. 字体样式:**

* strong:加粗,
* em:斜体,
* ins:下划线,
* del:删除线,
* suo:上标,
* sub:下标

**14. 转以符:**

* -&nbsp:空格
* &lt:小于号
* &gt:大于号

**15. 表单:input type:**

* text:单行文本框
* textarea:多行文本输入框
* password:密码输入框
* radiao:单选框
* checkbox:多选框
* option:下拉框
* submit 提交按钮
* 表单属性:
  + value:表单内容
  + name:表单名称
  + readonly:只读属性,内容不可更改
  + disabled:禁用表单的值,在提交会被舍弃
  + checked默认被选中

**16. 选择器**

* 标签选择器(列入p)
* id选择器:\*
* class选择器:.

**17. 在样式冲突时必定会有一个样式生效**

**18. 权重**

* 行内样式(1000)>id选择器(100)>class选择器(10)>标签选择呢器(1)>通配符/继承属性(0)
* 选择器选择时,就近使用,权重高的优先使用
* 比如插入图片,我在li用background(背景图片)代表着我把所有的样式都设置好了,若是我想在li的类里在对background设置则是没有效果,所以若想对background设置则在li使用background-image(background的简写),那么就可以在li的类里对背景进行设置

**19. 字体缩写:**

* font:ltalic bolder 25px/2 "黑体";
* 字体: 倾斜 加粗 25像素大/2行高 黑体

**20. 标签表现形式: display(修改元素的性质):**

* black:设为块元素 宽高有效,自动换行(跑,div,劜...)
* inline:设为行内元素 宽高无效,不自动换行(span,a...)
* inline-black:设为行内块元素 宽高有效,同行不会自动换行(img,button,input...)

**21. span:主要用来设置字体的样式(无任何语意)**

**22. div:块元素,主要用来设置页面的布局**

**23. p:不可以包含其他元素块**

**24. 样式继承:背景相关的不会被继承**

**25. 在页面布局时多使用浮动(块元素容易造成不对齐,但易于对页面进行垂直分割)**

* float:left/right(左/右浮动)

**26. 先简写再单写(内外边距边框,不然容易造成覆盖)**

**27. 盒子**

* margin:遵循上右下左的顺时针顺序(外边距)  
  auto:一般只给水平方向的margin使用
* padding:影响盒子所占空间的大小(内边距)

**28. (补充)对于相邻的兄弟元素,垂直外边距会取最值,而不是相加.**

* 发生重叠需要两元素垂直且相邻(父子情况)
* 对于一个页面而言,去除margin或padding是必要的,因为有些元素会有默认值
* 对于水平方向的边距会进行求和而不是重叠(内联元素).

**29. 隐藏:**

* visibity:hidden 隐藏元素,元素小时候依旧在原地占位
* display:none 隐藏元素实际上将元素剔除,使元素没有占位

**30. overflow:hidden 隐藏超出盒子的内容(解决高度坍塌)**

**31. 定位 position**

* 相对定位:relative 相当于自己对自己的移动,即原本位置A移动会把自身当做参照物,它不会改变元素的特性,并且在不设定的情况下不会继承元素的宽度,他不会覆盖文本流中的元素
* 绝对定位:absolate 他是相对于窗口或者有relative属性的元素为参照物,进行定位,它会以"优先靠近,优先使用"为原则(relative) 子绝父相
* 固定定位 fixed 类似网站中的联系电话(IT培训机构的那种)是以整个HTML文档为参照物  
  **( 关于固定定位:如果要将一个元素固定到一个页面最左或最右,可以设置 top/buttom 50% left/right 50%. 但如果只是将元素固定到视口中心需要对元素左右边距进行延长:延长距离=版心距离+元素离页面的边缘距离)**
* 粘性定位: sticky 是相对定位和固定定位的混合,以固定的阀值为界限,在固定定位和相对定位间切换(阀值自己设置)
* z-indes 可以将定位的元素优先级进行变化,0-9逐级变大,默认0

**32. 行距: line-height**

**33. 在页面布局时以标准的方式为优先,其次浮动,最后定位,页面是一个个元素累计搭建的,在布局时不要急,一步步来**

**34. 页面布局通常准行左浮动右定位**

**35. 对于代码极多情况需要仔细对比**

**36. 表格**

- 在网页中可以通过表格来表示一些格式化的数据

- 表格相关的标签

- <table> 用来创建一个表格

- <tr> 表示表格中的一行

- <th> 表示表头中的单元格

- <td> 表示表格中的单元格

- 属性：

colspan 横向的合并单元格

rowspan 纵向的合并单元格

- 例子：

<table>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

</table>

- 长表格

- <thead> 表格的头部

- <tbody> 表格的主体

- 注意：如果表格中没有写thead tbody tfoot，浏览器会自动向table中添加一个tbody

并且将所有的tr都放到tbody中，tr是tbody的子元素，不是table的子元素

- <tfoot> 表格的底部

**37. 表单**

- 表单可以将用户的信息提交到服务器中

- <form>

- 用来创建一个表单

- 属性：

action:需要一个服务器地址，提交表单时表单中的内容将会被提交到该地址

- 表单项

- <input />

- 它可以根据不同的type属性值，生成不同的表单项

- type="text" 文本框 <input type="text" name="" />

- type="password" 密码框 <input type="password" name="" />

- type="radio" 单选按钮 <input type="radio" name="" value="" checked="checked" />

- type="checkbox" 多选框 <input type="checkbox" name="" value="" checked="checked" />

- type="submit" 提交按钮 <input type="submit" value="按钮上的文字" />

- type="reset" 重置按钮 <input type="reset" value="按钮上的文字" />

- type="button" 普通按钮 <input type="button" value="按钮上的文字" />

- <select>

- 下拉列表

- <select name="">

<option value="" selected="selected"></option>

<option value=""> </option>

<option value=""></option>

</select>

- <button>

- 按钮功能input那几个按钮一样，但是它们要灵活一些

<button type="submit">按钮的文字</button>

<button type="reset">按钮的文字</button>

<button type="button">按钮的文字</button>

**JavaScript基础**

**2019.8.5-2019.8.9**

**1.在前端里,HTML是页面的框架,CSS为页面进行布局,JS实现页面的动态功能,它是一门直译式脚本语言**

**2. js的写入方式可以用外部引用,也可以卸载内部,但是需要使用scprit**

**3. 变量:**

* 可以使字母,数字,下划线
* 但不能是数字开头
* 不能使用关键字,保留字(JS里面的命名,例:div)
* 要严格区分大小写(大小写不同便是2个不同的元素)
* 推荐使用驼峰命名法(例:getEementByld, 即首字母小写,后面的单词开头大写)
* 取名时尽量语义化,看到名字就能知道它代表的意思

**4. JS输出方式**

* alert();弹窗(警告弹出框)
* console.log()控制台输出
* prompt()弹窗(用户输入语句)

**5. 算数运算符 (先乘除后加减,有括号先算括号中的):**

* 任何非Number的值要先转换成Number再运算
* 任何值与NaN运算结果都是NaN
* 任何值与String运算,要先转换成String再拼串(例:  
  a=1+2+'3'→3+'3'=33)
* 任何值做-,\*,/时都会自动转换成Number

**6. typeof()可以用来获取变量的类型**

**7. 变量的类型:**

* String:字符串,字符串相乘没有结果显示为NaN
* Number:数值{NaN (string*string/string*Number时会出现):Not a Number); isNaN(): :任何不能被转换为数值的值，都会让这个函数返回 true。}
* Boolean:布尔值 只有2个 true真,fase假
* undefind:声明了一个变量，但是没有赋值（例如：var a;），此时它的值就是 undefined。  
  null:空值(在被typeof时返回的是object(对象))

**8. 变量转换:**

* 其他值转换String:
* toString();
* 变量+" ";
* String();(函数)
* 其他值转换成Number:
  + Number();
  + parseInt():转换整数,自爆流字符节开头的的数字取整,专用于字符串,但不包括开头的0,
  + parseFloat:获取小数和小数后面的值
* 转换成Boolean:Boolean() 函数
  + 数字→布尔值:除了0,NaN都是true
  + 字符串→布尔值:除了空串(" "),都是ture
  + 对象,null,undefind转换后为ture

**9. 自增:**

* 例: var a=1;
  + a++ :var a1=a++;此时a1=1,即a为原来的自增前的值
  + ++a :var a2=++a;此时a2=2.即a为自增后的值
* a++,++a.都会自增同样的值,但是表达值却不同
  + a++是先将a的值赋予a在自增
  + ++a是先将自增后的值赋予a再自增(可以参考桌面文件夹不凡学院/Js基础/2019.8.5/学习/变量)
* 自减原理同自减

**10. 比较:**

* == 判断是否等于(不严谨,例 "true"==true)
* === 全等于(完全等于,有你无我)
* !反
* 规则:
  + 对于非数值比较先转换成数字在比较
  + 都是字符串时,不用转换以Unicode编码比较(同一位置开始比较,例:"56">"123" true)
  + 任何值与NaN比较都为fase(假);

**11. Date:**

* 时间对象,是一个构造函数,需要new出来一个实例对象才能使用,可以用来了解计算机的时间 例var date1 = new Date();  
  或var date2 = new Date("2017/09/06 09:00:00");(兼容性最强)
* .getTime:时间戳,具有唯一,是从格林威治标准时间到打钱日期所花费的时间毫秒数  
  获取日期和时间
* Date 对象 有如下方法，可以获取日期和时间：
  + getDate() 获取日 1-31
  + getDay() 获取星期 0-6（0 代表周日，1 代表周一）
  + getMonth() 获取月 0-11（0 代表一月）
  + getFullYear() 获取年份
  + getHours() 获取小时 0-23
  + getMinutes() 获取分钟 0-59
  + getSeconds() 获取秒 0-59
  + getMilliseconds() 获取毫秒 （1s = 1000ms）

**12. Match:**

* 不是构造函数,不需要创建对象,属于工具类,封装数字运算相关的属性和方法
* Math.abs() 返回绝对值
* Math.floor() 向下取整（向小取）
* Math.ceil() 向上取整（向大取）
* Math.round() 四舍五入取整（正数四舍五入，负数五舍六入）
* Math.random() 生成 0-1 之间的随机数 不包含 0 和 1
* Math.max(x, y, z) 返回多个数中的最大值
* Math.min(x, y, z) 返回多个数中的最小值
* Math.pow(x,y) 返回 x 的 y 次幂
* Math.sqrt() 对一个数进行开方运算

**13. 逻辑运算:**

* & 与 :一个为假,所有都为假
* || 或;一个为真所有为真
* ! 反/非
* 非布尔值进行与,或运算时,会先转换为布尔值,然后再运算,单反毁结果是原值(决定整个表达式真假的那个值)

**14. 三元运算符:**

* 表达式?如果表达式结果为 true 执行这里的代码:如果表达式结果为 false 执行冒号后面的代码;  
  1)例:

例5 > 3 ? "5大于3" : "5不大于3";

// 返回'5大于3'

var count = 30;

var price = count > 20 ? 6 : 7;

console.log(price);

// 可以理解为 if else 的另一种写法

**15. if,switch为条件判断语句,条件分置于句**

**16. for,while为循环语句**

* while为先判断再运行
* do while为先运行在判断

**17. break:**

* 用来停止switch语句或整个循环语句
* 用来终止离它最近的循环语句

**18. continue:**

* 用来调过此次循环
* 只会对默认离它最近的循环起作用

**19. 对于if,for的应用,混合应用**

* 要活用逻辑运算符
* 对于列出某一数字能够整除的数字,以var==i和for将其内的数都列出来进行取余,余数为0即成立

var a =prompt("请输入");

for(var i=1;i<=a;i++){

if(a%i=0){

console.log(i);

}

}

* 对于取余a%b==c要灵活运用
* 对于求和求乘积要添加累加器(累加器说白了就是先声明一个值为0的量然后将变量I加到上面)例

求和 var a=0;

for(var i=1;i<=a;i++){

a=a+i;(a+=i);

console.log(a);

}

在求和时a=0不影响初始值,为1影响

求乘积时a=0则无效(a=a\*i)只能为1

* 对于判断质数要声明一个量,当做计数器,在if中一次判断+1(++a),for则进行将数全部列举  
  1). 以求质数为列,他只有2个因数,因此计数器最大且只能为2,符合a=2的数则是质数  
  1.Array:数组
* 内置对象,普通对象们可以根据属性名获取值
* 需要使用索引,获取元素的值
* 数组当中没一个元素都有对应的下标(索引).下标从0开始
* 数组的元素可以使任意的数据类型
* 数组通常是表达同一类数据的集合
* 数组具有增删改查四种基本操作()都是通过数组下标进行操作)

例

var arr=[1,2,3,4](下标0-3)

增: arr[4]=5; 在下标位4出添加元素5

push():在最后加上一个元素(数组的末尾)

unshift()在数组第一个元素之前添加一个元素.

二者都是修改了原数组,返回值为修改后的数组长度

查:console.log(arr[3]); 控制台输出4,即获取了下标3的值=4

改: arr[1]=33; 即将下表1的值由2变为33(不仅能修改数组的值也

可以修改数组的长度length)

例:原长度为4,修改为6后,多出的2个长度值为null

反之,若从4改为3 ,则会删除多出的长度,从后向前删除

删: shift():删除数组第一个元素

pop():删除数组最后一个元素.

二者都是修改了原数组,并且返回值都是删除的那个元素

**20. 数组**

* 遍历数组:
  + 查找元素需要用到的下标
  + 遍历下标(0-length)
  + 应用下标arr[i]来进行判断
* 对于数组的平均数,需要先对数组进行求和在平均
* 对于数组求和,要先遍历数组,先将下标找出再将值拿出来然后应用求和的思想定义一个指,然后var a=a+arr[i];

**21.冒泡排序:**

* 会重复地走访要排序的元素列,一次比较两个相邻的元素(它们的顺序,从大到小,首字母从A-Z,如果有错误则会把他们交换过来,
* 走访元素的工作室重复地进行,直到没有相邻的元素需要交换即元素已经完成排序)
* 越大的元素会经由交换慢慢浮到数列的顶端(升序或降序)

例: var arr = [23, 45, 12, 9, 47, 11];

冒泡排序

// for (var i = 0; i < arr.length - 1; i++) {

// if (arr[i] > arr[i + 1]) {

// // 顺序错误

// // 交换位置

// // 先把之前arr[i]的值保存下来

// var temp = arr[i];

// arr[i] = arr[i + 1]; // arr[i]已经发生了改变

// arr[i + 1] = temp;

// }

// }

// 这段代码 要执行 arr.length-1次

冒泡排序优化

for (var j = 1; j < arr.length; j++) {

// 外层 for 循环 控制 轮次

for (var i = 0; i < arr.length - j; i++) {

// 内层 for 循环 负责 找出本轮最大值

if (arr[i] > arr[i + 1]) {

// 顺序错误

// 交换位置

// 先把之前arr[i]的值保存下来

var temp = arr[i];

arr[i] = arr[i + 1]; // arr[i]已经发生了改变

arr[i + 1] = temp;

}

}

}

console.log(arr);

**22. 函数:function**

* 命名规定和变量的命名规定是一样的
* 参数可选
* 函数表达式:将匿名函数的赋值给一个变量
* 形参:在函数的()中来指定一个或多个形参function(a,b,c),多形参之间使用逗号隔开,声明形参就相当于在函数内部声明了对应的变量,并不赋值
* 实参:在调用函数式可以在()中定义实参arr(1);此时1就是实参.实参会赋值给函数中对应的形参(形参,实参的个数相同)

例: function arr(str){

alert(str);

}

arr(1)

str形参,1实参

* return:函数的返回值
  + return之后的代码不会再执行m,函数遇到return会执行结束
  + 不是所有的函数都有返回值
  + 但是返回值很重要
  + 返回值可以是任意的数据类型，可以是对象，也可以是函数。
* fn()：调用函数。相当于获取了函数的返回值。
* fn：函数对象。相当于直接获取了函数对象。

**23. 作用域**

* 全局作用域:里面的变量都是全局变量,在页面的任意位置都可以访问
* 局部作用域(局部变量):在局部作用域中声明的变量,函数内部的声明的变量,只能在内部使用,若在函数要访问全局变量可以使用window对象(window.)
* 变量遵守就近原则,先找自身作用域,找不到再找父级作用域查找(上一级作用域),弱势全局作用域也没有找到便会报错

**24. 递归**

* 必须要有条件,没有条件它会一直地跪下去,无法进行输出
* 条件是让函数内部满足他时停止然后输出

**25. 声明**

* 函数声明放在上面会在函数代码执行前执行即undefined
* 变量声明若不使用var,即变量成为全局变量,声明提前
* 若使用var,表明变量在代码执行前已经被声明但没有赋值

例: console.log(a);//undefind,被声明没有赋值

var a=10;

console.log(b);//bis not defind 没有声明提升

b=2;

console.log(b);//2 b=全局变量

**26. 函数里的代码不调用是不会执行**

**27. 对象:用来封装信息,用来描述一个事物**

**28. 构造函数:**

* 和普通函数一笔没构造函数首字母大写,在使用时用new
* 在构造函数中,this的指向对象是创造出的实例对象
* 遍历对象可以使用 in 来进行,在输出时必须加[]

**29. JSON:**

* 数组,对象等是不可以进行传输的,但是可以借助JSON的2个方法来进行传输
  + JSON.Stringfy();将数组,对象等变为文本
  + JSON.porse();将文本变为数组,对象等

**30. 数组的高级对象**

* forEach():遍历数组,对多科协三个参数,第一个必须要有,另外两个可以根据需要来使用  
  1). 即 forEach(item,index,arr);,item:遍历数组的元素,index:遍历数组的元素对应的下标,arr你遍历的数组
* 翻转数组:reverse
* 截取数组(slice):
  + 例:arr.slice(a,b);a时开始的索引,b是结束的索引
  + slice并不包括b的值,即输出的是a-b-1的对应的下标的元素
  + arr.slice(0)即复制数组,并不影响原数组
* 删除或替换数组中的元素:splice()
  + arr.splice(a,b,c)替换:a从哪个索引开始替换,b从替换开始的地方a开始,要替换几个元素.C:参数,即修改的值可以一个或多个
  + 删除:即arr.splice(a,b),不写后面的参数,即表示删除
* 返回元素在宿主的索引:
  + indexof:从前开始匹配
  + lastIndexof:从后开始匹配
  + 若是输入一个数组中找不到的元素,则返回-1
* 字符串的高级方法
  + charAt();返回指定位置的字符(从0开始),charCodeAT():和charAt一样不过是返回指定位置的字符的Uncode编码
  + .indexof("a"):返回"a"在元素中的位置由前→后开始,lastIndexof("a"),一样只是是由后→前开始
  + 截取:slice(a,b):a-b-1的元素,substr(a,b),a-b的元素
  + toUpperCase:字符串大变小,toLowerCase()字符串小变大
  + concat:连接字符串,例:var a=b.concat(c);
* 排序sort
  + 当不带有参数是以unicode编码进行排序
  + .带参数时,按照指定的排序规则进行(参数可以是函数)

例: arr.sort(function(a,b){

return a-b;

})

a-b:从小到大排序,反之从大到小排序

其中当结果>0时元素位置交换,=0,<0时元素位置都不会变动

并且对于对象是可以进行设置条件(即和函数一般操作)

例: console.log(arr2);

var users = [{

name: '张三',

age: 20,

height: 176

},

{

name: '李四',

age: 19,

height: 175

},

{

name: '王五',

age: 22,

height: 172

},

{

name: '赵六',

age: 18,

height: 174

},

{

name: '陈琦',

age: 18,

height: 178

},

]

// 按照年龄从小到大排序

// 年龄相同时 按身高从大到小排

users.sort(function (a, b) {

if (a.age === b.age) {

return b.height - a.height

} else {

return a.age - b.age;

}

})

console.log(users);

**31. JavaScript**

* JavaScript负责页面中的的行为。
* 它是一门运行在浏览器端的脚本语言。
* JS的编写的位置

1.可以编写到标签的指定属性中

<button onclick="alert('hello');">我是按钮</button>

<a href="javascript:alert('aaa');">超链接</a>

2.可以编写到script标签中 \*\*\*\*\*

<script type="text/javascript">

//编写js代码

</script>

3.可以将代码编写到外部的js文件中，然后通过标签将其引入 \*\*\*\*\*

<script type="text/javascript" src="文件路径"></script>

* 输出语句
* alert("要输出的内容");
* 该语句会在浏览器窗口中弹出一个警告框
* document.write("要输出的内容");
* 该内容将会被写到body标签中，并在页面中显示
* console.log("要输出的内容");
* 该内容会被写到开发者工具的控制台中
* 基本的语法
  + 注释
  + 单行注释: //注释内容
  + 多行注释:/*注释内容*/
* JS严格区分大小写
* JS中每条语句以分号(;)结尾
* JS中会自动忽略多个空格和换行，所以我们可以利用空格和换行对代码进行格式化。
* 字面量和变量
  + 字面量
    - 字面量实际上就是一些固定的值，比如 1 2 3 4 true false null NaN "hello"  
      字面量都是不可以改变的。
      * 由于字面量不是很方便使用，所以在JS中很少直接使用字面量
  + 变量
    - 变量可以用来保存字面量，并且可以保存任意的字面量
    - 一般都是通过变量来使用字面量，而不直接使用字面量，而且也可以通过变量来对字面量进行一个描述
    - 声明变量
      * 使用var关键字来声明一个变量  
        var a;  
        var b;  
        var c;
      * 为变量赋值  
        a = 1;  
        b = 2;  
        c = 3;
      * 声明和赋值同时进行 \*\*\*\*\*  
        var d = 456;  
        var e = 789;
* 标识符
  + 在JS中所有的可以自主命名的内容，都可以认为是一个标识符，  
    是标识符就应该遵守标识符的规范。
  + 比如：变量名、函数名、属性名
  + 规范：
    - 标识符中可以含有字母、数字、\_、$
    - 标识符不能以数字开头
    - 标识符不能是JS中的关键字和保留字
    - 标识符一般采用驼峰命名法  
      xxxYyyZzz

**32. 文档流**

* 所有的元素默认情况下都是在文档流中存在的
* 文档流是网页的最底层
* 元素在文档流中的特点：
* 块元素
  + 默认宽度是父元素的全部
  + 默认高度被内容（子元素）撑开
  + 在页面中自上而下垂直排列
* 内联元素  
  1.默认高度和宽度都被内容撑开  
  2.在页面中自左向右水平排列，如果一行不足以容下所有的元素  
  则换到下一行继续从左向右

**33. 浮动**

* 使用float来设置元素的浮动
* 可选值：
  + none 默认值，元素不浮动，就在文档流中
  + left 元素向页面的左侧浮动
  + right 元素向页面的右侧浮动
* 浮动特点：
  + 元素设置浮动以后，会完全脱离文档流，并向页面的左上或右上浮动。  
    直到遇到父元素的边框或其他的父元素时则停止浮动。
  + 如果浮动元素上边是一个没有浮动的块元素，元素不会超过该块元素。
  + 浮动元素的浮动位置不能超过他上边浮动的兄弟元素，最多一边齐
  + 浮动元素不会覆盖文字，文字会围绕在浮动元素的周围，所以可以通过浮动来实现文字环绕图片的效果。
* 浮动以后元素会完全脱离文档流，脱离文档流以后元素会具有如下特点：
  + 块元素不独占一行
  + 块元素的宽度和高度都被内容撑开
  + 元素不在文档流占用位置
  + 内联元素会变成块元素

**34高度塌陷**

* 在文档流中元素的高度默认被子元素撑开，当子元素浮动时，子元素会脱离文档流，  
  此时将不能撑起父元素的高度，会导致父元素的高度塌陷。父元素高度塌陷会导致其他元素的位置上移，导致页面的布局混乱。
* 可以通过开启元素的BFC来处理高度塌陷的问题。
  + BFC叫做Block Formatting Context
  + 它是一个隐含属性，默认情况是关闭，当开启以后元素会具有如下的特性：
  + 父元素的垂直外边距不会和子元素重叠
  + 开启BFC的元素不会被浮动元素覆盖
  + 父元素可以包含浮动的子元素 \*\*\*\*\*\*
* 开启BFC的方式很多：
  + 设置元素浮动
  + 设置元素绝对定位
  + 设置元素为inline-block
  + 将元素的overflow设置为一个非默认值
* 一般我们采取副作用比较小的方式  
  overflow:hidden;

**35. 定位**

* 通过定位可以将元素摆放到页面的任意位置
* 使用position来设置元素的定位
  + 可选值：
    - static 默认值 元素不开启定位
    - relative 开启元素的相对定位
    - absolute 开启元素的绝对定位
    - fixed 开启元素的固定定位
* 相对定位  
  1.相对于元素自身在文档流中的位置进行定位  
  2.相对定位的元素不会脱离文档流，定位元素的性质不会改变，块还是块，内联还是内联  
  3.如果不设置偏移量，元素不会发生任何的变化  
  4.会提升元素的层级
* 绝对定位  
  1.相对于离它最近的开启了定位的祖先元素进行定位，如果祖先元素都没有开启定位则相对于浏览器窗口进行定位。  
  2.绝对定位会使元素完全脱离文档流，会改变元素的性质，内联变成块元素，块元素的宽度被内容撑开  
  3.绝对定位的元素如果不设置偏移量，元素的位置不会发生变化  
  4.会提升元素的层级
* 固定定位
  + 固定定位也是一种绝对定位，它的大部分特点都和绝对定位是相同的。
  + 不同的是：
  + 固定定位永远相对于浏览器窗口进行定位
  + 固定定位会固定在浏览器的指定的位置，不会随页面一起滚动
* 偏移量
  + 当元素开启了定位以后，可以通过四个偏移量来设置元素的位置  
    top：相对于定位位置的顶部的偏移量  
    bottom：相对于定位位置的底部的偏移量  
    left：相对于定位位置的左侧的偏移量  
    right：相对于定位位置的右侧的偏移量
* 一般只需要使用两个值即可给元素进行定位  
  top left  
  top right  
  bottom left  
  bottom right
* 偏移量也可以指定一个负值，如果是负值则元素会向相反的方向移动
  + 层级
    - 当元素开启定位以后，可以通过z-index来设置层级，  
      它需要一个正整数作为参数，值越大层级越高，层级越高越优先显示
    - 如果层级一样，则后边的会盖住前边的，父元素永远都不会盖住子元素。
    - 文档流 < 浮动 < 定位

**36. 元素的透明**

* 使用opacity来设置元素的透明度
  + 需要一个0-1之间的值
  + 0 表示完全透明
  + 1 表示完全不透明
* IE8及以下的浏览器不支持该样式，需要使用如下方式来设置
  + filter:alpha(opacity=透明度);
  + 需要一个0-100之间的值
  + 0 表示完全透明
  + 100 表示完全不透明

**37. 添加原型对象**

* Prototype往函数的原型对象里添加属性方法等
* 原型对象,比如多个函数内部都有一个一样的方法或者属性,就可以把它们统一放在原型对象中,也可以把原型对象理解成一个公共仓库,,

**Dom基础**

**2019.8.13**

**1. Dom: Document object model,为文本提供了结构化表示,文档对象型,操作网页上的元素的API,如让盒子变宽,图像轮播,计时器等**

* **dom常用事件**

|  |  |
| --- | --- |
| **属性** | **当一下情况发生时,出现此事件** |
| onabort | 图像加载被中断 |
| onblur | 元素失去焦点 |
| onchange | 用户改变域的内容 |
| onclick | 鼠标点击某个对象 |
| ondblclick | 鼠标双击某个对象 |
| onerror | 当加载文档或图像时发生某个错误 |
| onfocus | 元素获得焦点 |
| onkeydown | 某个键盘的键被按下 |
| onkeypress | 某个键盘的键被按下或按住 |
| onkeyup | 某个键盘的键被松开 |
| onload | 某个页面或图像被完成加载 |
| onmousedown | 某个鼠标按键被按下 |
| onmouseout | 鼠标从某元素移开 |
| onmouseover | 鼠标被移到某元素之上 |
| onmouseup | 某个鼠标按键被松开 |
| onreset | 重置按钮被点击 |
| onresize | 窗口或框架被调整尺寸 |
| onselect | 文本被选定 |
| onsubmit | 提交按钮被点击 |
| onunload | 用户退出页面 |

**2. 事件的三要素: 事件源,时间,事件驱动(即: 提出的问题,拿出解决的方案名,拿出具体的解决方案)**

**3. 获取事件源**

* 通过ID获得单个标签
* 通过标签名获得标签数组
* 通过类名获得标签数组
* 通过Class获取的是类数组(不是真的数组),它有下标有长度,能用for循环,遍历,但是不具备数组的通常方法(pop,shift,jion等)

**4. 在HTML中一切都是借点,整个HTML文档也是一个节点,所有的节点都是Object**

* 文本节点:标签中的文字,
* 属性节点:标签的属性
* 元素节点:HTML标签

**5. 创建节点**

* 新的标签(元素节点) = document.createElement('标签名');
* 插入节点:重复插入同一个节点,它的剪切效果只保留一个
  + 父节点(插入):appendchid(新的子节点),父节点最后插入一个新的子节点
  + 父节点.inserBefore(新的子节点,作为参考的子节点),在参考节点前插入一个新的节点,如果参考的节点为null,将在父元素最后插入一个子节点
  + 在父节点前+document(删除,插入时)
* 删除节点:
  + 父节点.removrechild(子节点); 用父节点删除子节点,必须指定删除的那个子节点
  + 删除自己:nodel.parentNode.removrchild(npdel);
* 复制节点:
  + 要复制的节点.cloneNode();不带参数/带参数 false：只复制节点本身，不复制子节点。
  + 要复制的节点.cloneNode(true);带参数 true：既复制节点本身，也复制其所有的子节点。

**6. 节点的属性**

* 获取节点属性  
  1.)元素节点.属性;/ 元素节点[属性];  
  2.) 元素节点.getAttribute(属性名称);
* 设置节点的属性
  + 元素节点.属性名 = 属性值
  + 元素节点.setAttribute(属性名, 新的属性值);
* 删除节点  
  1.) 元素节点.removreAttribute(属性名);
* **获取节点的属性值和设置节点的属性值，都有两种方式，但这两种方式是有区别的:**
  + 方式一的元素节点.属性和元素节点[属性]:绑定的属性值不会出现在标签上。
  + 方式二的get/set/removeAttribut: 绑定的属性值会出现在标签上。

**7. 类名操作(H5新加操作)**

* 追加类名:元素.ClassList.add('类名');可以追加多个类,用逗号隔开
* 删除类名:元素.ClassList.remove('类名');
* 判断类名是否存在(true/false):元素.ClassList.contains('类名');
* 判断类名是否存在,有则删除,无则添加:元素.ClassList.toggle('类名');
* 获取第[i]个类的名字:元素.ClassList.item[i];(假使i=1,则获取下标为1的类名)

**8. 表单内容操作**

* 元素[i].value = "新的属性值";
* 内容操作
  + innerText:获取标签内部所有文本,对于双臂和标签里的内容,不识别标签,直接把内容当做文本
  + innerHTML:获取元素内部所有标签,对于双臂和标签的内容,当成新的标签(常用来动态生成)
  + 动态生成实例
* var imgarr = ['m1.jpg', 'm2.jpg', 'm3.jpg', 'm4.jpg', 'm5.jpg'];
* var ul = document.getElementsByTagName('ul')[0];
* initImg(ul,imgarr);
* function initImg (el,arr){
* var str ='';
* for(var i = 0; i<arr.length; i++){
* str+='<li style="background-image: url(img/' + arr[i] +')"></li>';
* }
* el.innerHTML = str;
* }

**9.定时器**

* setTimeout();延时执行一次
* setInterval();循环执行多次,每隔X毫秒执行一次
* 关闭循环计时:setTimeout(c=>{  
  clearTimeout(变量);},时间),先清除:setTimeout(c)再清除 clearTimeout(),C为循环计时的名称变量
* clearTimeout();清除计时器

**10. 立即执行函数技巧:**

* 内部的this定义不明确,最好在函数外声明一个新的类(同级别)来代表i,然后函数内部才能使用,对于JQ里面则是声明一个新的变量

**BOM**

**2019.8.27**

**1. offsetWidth,offsetHeight:检测盒子自身的宽高**

**2. offsetLeft,offsetTop:检测距离有定位属性的盒的左/上的距离**

**3. offsetLeft/Top与style.left/top的区别在于前者只可以读,后者可以读写**

**4. offsetParent （检测父系盒子中带有定位的父盒子节点）**

**5. 动画**

* 匀速动画:
  + 即在每个一个固定时间移动一次距离,每次移动的距离相等
  + 动画操作流程
    - 设置计时器(每隔多久执行一次)
    - 获取移动目标的pffsetleft或者offsettop,a
    - 设置步长(每次移动的距离,因为要考虑返回是的步长,所以用三元运算符来进行判断.比如:正值向右,负值便向左),b
    - 运算,a+=b;
    - 赋值,将a付给盒子的style.left/top
    - 关闭计时器(进行判断,若是总长度-走的长度<=步长那么遍关闭定时器,走的步长固定位总的长度),使用绝对值判断
* 缓动动画
  + 在每一个固定时间内,移动一次距离,每次移动的距离都不相等,下一次移动的距离,为剩下的距离几分之几(通常默认十分之几,10是最佳运动)
  + 计算公式:步长 = (总长度-盒子的offsetLeft/Top)/10;
  + 盒子的offsetLeft/Top可以当做起始位置,即,我从起点以缓动动画的样式跑800米,假使进行滚动形式,可以将总长度设置为0,或者body的长度,可以完成一键到顶或底的操作
  + 因为缓动动画的特性,步长逐步变小,无限接近0但又不可能等于0,故步长最后一步因该是我们来进行判断给它赋值+-1,判断条件是步长是否在(0,1)之间,又因为要返回故为[-1,1]之间的区间,步长不为0.
  + 判断公式: if(Math.abs(步长)<1){s步长p = step > 0 1:Math.floor(步长);}
  + 其余的操作和匀速动画相当
* 图片轮播
  + 在进行图片轮播时要注意首尾图片相等,即:有5张图片,那么我要在最后添加一张和第一张一摸一样的图片来充当过渡图
  + 在图片轮播,图片移动过程中要注意图片的下标,即0转换为图片数组长度,图片数组长度-1=0,同时将他们的style.left转换为相等的长度
  + 图片轮播使用函数时需要用到计时器,但会有双击晃动问题,这是因为下一个计时器启动上一计时器还存在,因此需要有删除计时器操作(在计时器前后各使用clearInterval删除就行),在这里因为要在全局使用故后面全局调取方便,可以给ele代表的标签添加一个类名即:ele.??
  + 在轮播需要自动切换时,只需要给button绑定点击事件并放在计时器中便可,但要注意,计时器要在全局使用,因此不要在局部进行声明,需要在外部声明一个变量,而且要让这个变量等与null(可以不赋值null,但在后期可能会出错)
  + 对于轮播的小数点,在其函数内部的this是固定值,因此需要小数点的元素重新声明一个类名index即var index = this.index;,在在后面声明varindex= this.index;此时index为小数点下标,
  + 然后再执行排他,增加操作,此时排他操作可以使用  
    document.querySelector('类名/标签名').classList.remove('active');来代替以前的for循环
  + 切换按钮绑定小数点与小数点操作类似,只需在内部执行排他,增加操作.
  + 推荐使用时,先写出来再封装函数

**6. 事件Event**

* 可在函数里直接使用Event,也可以在function(e)里添加e,后面可直接使用e.来代替event
* 鼠标事件
  + e.pageX/Y:获取鼠标点击的相对于页面的位置(即鼠标点击区域和浏览器的距离)
  + .clientX/Y:获取鼠标点击的相对于可视区域的位置(在一定程度上和page差不多,但是在页面有覆盖的情况下会产生不同)
  + e.screenX/Y:获取鼠标点击的相对于屏幕的位置(会跟着屏幕变化而变化)
  + e.offsetX/Y:表示鼠标指针位置相对于触发事件的对象的位置(就是鼠标点击的地方和鼠标点击区域的盒子的距离)
* 键盘事件
  + event.key:键盘按键码,即我在键盘敲击哪个按键就可以输出,比如我敲击a那么使用这个方法就可以在控制台输出a
  + event.keyCode:按键按键,二进制代码,比如我敲击回车键,则可以在控制台输出13,即Enter = 13
* 鼠标按键事件(鼠标左中右三键)
  + 使用event.button(鼠标点击按键),可在控制台输出,通常使用onmousedown事件来测试
    - 0 您点击了鼠标左键
    - 1 您点击了鼠标中键
    - 2 您点击了鼠标右键

**7. 事件冒泡和阻止冒泡**

* 阻止a标签默认跳转
  + event.preventDefault ? event.preventDefault() : event.returnValue = false;
* 阻止事件冒泡
  + event.stopPropagation ? event.stopPropagation() : event.cancelBubble = true;

**8. 常用鼠标事件**

* onmousedown :鼠标按键按下 **a.**
* onmousemove:鼠标在区域内移动 **b.**
  + 例如:box.onmousemove只能在box区域内移动
  + window.onmousemove.可以在整个页面内移动
  + 即box事件内的鼠标判定条件出了box是失效的只有重新回到box内才有效
* onmouseup:鼠标按键松开**a.**
* 通常的流程是以**aba**为顺序进行的

**9. 鼠标跟随事件注意事项**

* 图像一定要开启定位
* 看需求图像以那个位置为中心跟随鼠标移动,但总体是图像移动后距离页面的左/上的边距=鼠标距离页面的距离-图像距离页面的左/上距离.
  + 即:获取鼠标相对于绑定事件的元素的位置 e.pageX/Y - 绑定事件的元素距离页面的左/上边距(offsetLeft/Top)

**10. 鼠标拖拽事件(有2种方法)**

* 第一种
  + 鼠标到盒子左/上的距离 = 鼠标未移动时对于页面的距离- 盒子未移动时对于页面的距离
  + 盒子移动的左/上距离 = 鼠标移动后的左/上距离 - 鼠标到盒子的距离
* 第二种
  + 盒子移动的左/上距离 = 盒子移动后鼠标对于左上的距离 - 盒子未移动时鼠标对于页面的距离
  + 盒子移动后对于左/上页面的距离 = 盒子移动的左/上距离+ 盒子未移动时对于左/上页面的距离
* 盒子在拖拽时是否限制在某一固定区域
  + 即找出盒子移动的极值(maxX/Y,minX/Y)
  + 判断盒子移动后的距离是否超出或小于极值,超出赋予最大极值,小于赋予最小极值
  + 在判断时不要偷懒图省事,注意X,Y的变化

**11. 放大镜效果注意事项**

* 一张图片,分别设置在2个box内,第一张作为参考物,第二张作为放大效果(此时第二张的大小设置为第一张的N倍,看需求设置)
* 第二张的移动距离 = N倍的第一张里面的放大镜(盒子)移动距离
  + 示例:
* <!-- img1第二张图片,nowX/Y第一张图片里放大镜移动的距离 -->
* <!-- img1.style.left = -2 \* targetX + "px";
* img1.style.top = -2 \* targetY + "px"; -->
* <!-- 2种方式可以设成left/top也可以设成marginLeft/Top -->
* img1.style.marginLeft = -2 \* nowX + 'px';
* img1.style.marginTop = -2 \* nowY + 'px';

**12. 鼠标形状:cursor:move,鼠标指针变为"十"形状**

**13. scroll**

* onscroll滑动事件
* scrollWidth/scrollWidth获取元素的宽高(在具有滚动效果的情况下才能使用,没有的话不能使用)
* **overflow: auto;滚动条属性**
* 有box1在box内部的2个盒子
  + 目标盒子的宽高(box) = 内容高度(里面盒子的宽高) + padding(左右一个padding,上下2个padding,没有滚动属性overflow: auto时无法获取)
  + 里面盒子的宽高(box1) = 本身
* scrollLeft:实时获取页面卷入的宽度 指定的元素卷入的宽度,即被覆盖的宽度,兼容写法及常用写法
  + function scl() {
  + return document.documentElement.scrollLeft || window.pageXOffset || document.body.scrollLeft
  + }
* scrollTop:实时获取页面卷入的高度 指定的元素卷入的高度,即被覆盖的高度
  + function sct() {
  + return document.documentElement.scrollTop || window.pageYOffset || document.body.scrollTop;
  + }
* scroll(下,y):x,y代表页面或者目标元素的2个滚动条,可以通过改变x,y的值来进行调整滚动条
* 对于滚动条改变导航栏样式的操作,要根据条件找出公因式为后面的换算做准备
  + 出题:有导航栏a,现要求:当页面高度卷入300px时,a的高度从150px变为100px,透明度从1变成0.2
  + 即此时可以发现 0→300,150→100,1→0.2,公因式可为  
    50/300 = 1/6 = 页面卷入高度/6(高度变化)  
    0.8/300 = 页面卷入高度/375(透明度变化)

**JQuery基础**

**2019.8.15**

**1.JQuery:一个快速简介的JS框架,它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互。**

* 三大版本
  + 1.x 使用最广泛,若没有特殊要求,开发项目多使用它
  + 2.x 使用最少
  + 3.x 最新版本,只支持最新的浏览器
* JQ要先引用才能使用,它的引用顺序在第一顺位
* JQ与Dom方法相差不大,很多Dom去掉on就可以在JQ中使用
* JQ→Dom:var = a1;var = a;*a*;a[0]
* Dom→JQ: var = a1;var = a;*a*;(a1)
* JQ的基本选择器与css选择其一摸一样,$('类名').css();

**2.JQ选择器**

* **jQuery的基本选择器**

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **用法** |
| $('#demo') | 选择 id 为 demo 的元素 | $('#demo').css('color','red') |
| $('.demo') | 选择class为demo的所有元素 | $('.demo').css('color','red') |
| $('div') | 选择所有 div 标签元素 | $('div').css('color','red') |
| $('\*') | 选择所有标签元素 | $('\*').css('color','red') |
| $('.arr.arr-left') | 选择同时具有 arr 和 arr-left 类名的元素 | $('.arr.arr-left').css('color','red') |

* **jQuery 的其它选择器**

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **用法** |
| 空格 | 后代选择器 | $('div span').css('color','red') |
| > | 子代选择器 | $('div>span').css('color','red') |
| + | 紧邻选择器 | $('div+p').css('color','red') |
| ~ | 兄弟选择器 | $('div~p').css('color','red') |

* **属性选择器**

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **用法** |
| $('a[href]') | 具有 href 属性的 a 标签 | $('a[href]').css('color','red') |
| $('a[href='baidu']') | href 属性为'baidu'的 a 标签 | $('a[href='baidu']').css('color','red') |
| $('a[href!='baidu']') | href 属性不为'baidu'的 a 标签,包括不具有 href 属性的 a 标签 | $('a[href!='baidu']').css('color','red') |
| $('a[href^='www']') | href 属性以'www'开头的 a 标签 | $('a[href^='www']').css('color','red') |
| $('a[href$='cn']') | href 属性以'cn'结尾的 a 标签 | $('a[href$='cn']').css('color','red') |
| $('a[href\*='i']') | href 属性包含'i'的 a 标签 | $('a[href\*='i']').css('color','red') |
| $('a[href][title='内容']') | 具有 href 属性且 title 属性为'内容'的 a 标签 | $('a[href][title='内容']').css('color','red') |

* **基本筛选选择器**

| **符号** | **说明(index 从 0 开始)** | **用法** |
| --- | --- | --- |
| :eq(index) | 匹配一个给定索引值的元素 | $('li:eq(1)').css('color','red') |
| :gt(index) | 匹配所有大于给定索引值的元素 | $('li:gt(1)').css('color','red') |
| :lt(index) | 匹配所有小于给定索引值的元素 | $('li:lt(2)').css('color','red') |
| :odd | 匹配所有索引值为奇数的元素 | $('li:odd').css('color','red') |
| :even | 匹配所有索引值为偶数的元素 | $('li:odd').css('color','red') |
| :first | 获取匹配的第一个元素 | $('li:first').css('color','red') |
| :last | 获取匹配的最后一个元素 | $('li:last').css('color','red') |

* **其他选择器**

|  |  |  |
| --- | --- | --- |
| **符号** | **说明(index 从 0 开始)** | **用法** |
| :empty | 匹配所有不包含子元素或者文本的空元素 | $('li:empty') |
| :contains(text) | 匹配包含给定文本的元素 | $('li:contains('john')') |

**3. 类名操作**

* addClass(); 向被选元素添加一个或多个类
* removeClass(); 从被选元素删除一个或多个类
* toggleClass(); 对被选元素进行添加/删除类的切换操作
* hasClass(); 判断被选元素是否存在类

**4. 筛选过滤/查找**

* eq(index);
* find();
* **siblings(); 除了自己以外的所有兄弟节点,常与addClass()向元素添加1或多个类,removeClass();向元素删除1或多个类使用**
* children(); 所有孩子节点
* next(); 下一个兄弟节点
* nextAll(); 后面的所有兄弟节点
* nextUntil();后面的兄弟节点,直到...
* prev();上一个兄弟节点
* prevAll();前面的兄弟元素
* prevUntil();
* parent(); 父节点
* parents(); 所有父节点
* parentsUntil();

**5.属性操作 .attr()**

* 获取属性   $('img').attr('src')  获取 img 的 src 属性值
* 设置属性   $('img').attr({src:'text.jpg',alt:'sorry'})
* removeAttr() $('img').removeAttr('src') 删除 src 属性

**6.html 代码/文本/值**

* 可以取值,设值
* html()   ('p').html()(′*p*′).*html*()("p").html('html 代码')
* text()   ('p').text()(′*p*′).*text*()('p').text('内容')
* value()   ('input').value()(′*input*′).*value*()('input').value('姓名')
* ♣ prop() ('input').prop('checked',false/true) 专门用来设置选中状态

**7.动画**

* 隐藏(hide)/显示动画(show)
* 滑入滑出动画
* 淡入淡出动画  
  -**自定义动画  
  例:$(selector).animate(styles,speed,ease,callback)  
  第一个参数表示：要执行动画的 CSS 属性（必选）  
  第二个参数表示：执行动画时长（可选）  
  第三个参数表示：动画执行完后立即执行的回调函数（可选）  
  ease:运动函数:swing:先慢后快再慢,linear:线性匀速**
* 停止动画
  + stop(); 停止当前动画
  + stopAll:是否全部停止动画(停止队列中所有动画),默认 false
  + goToEnd: 是否将停止的动画,停在当前动画的最后一个状态

**8.节点**

* append:在被选中元素内部的最后一个子元素/内容后插入
* appendTo:把新的元素追加到选中的元素中
* prepend:在被选元素内部第一个子元素/内容前面插入
* affter:被选元素前作为兄弟元素插入
* 删除节点:html/empty,二者没有区别,remove连自己一起删除
* 复制节点: .clone()复制JQ对象(不复制内容),clone[i]复制JQ对象连里面的下标为i的内容也复制

**9.事件绑定**

* 简单事件绑定:
  + click(handler)单击事件
  + blur(handler) 失去焦点事件
  + mouseenter(handler) 鼠标进入事件
  + mouseleave(handler)鼠标离开事件
  + dbclick(handler)双击事件
  + change(handler)改变事件，如：文本框值改变，下来列表值改变等
  + focus(handler) 获得焦点事件
  + keydown(handler)键盘按下事件
  + 更多<http://www.w3school.com.cn/jquery/jquery_ref_events.asp>
* delegate方式（特点：性能高，支持动态创建的元素）
  + 给匹配到的元素绑定事件，对支持动态创建的元素有效  
    $(".parentBox").delegate("p", "click", function(){});
  + 与前两种方式最大的优势：减少事件绑定次数提高效率，支持动态创建出来的元素绑定事件！
* on方式（兼容zepto(移动端类似jQuery的一个库)，建议使用的方式）
  + 给匹配的元素绑定事件，包括了上面所有绑定事件方式的优点语法：
  + 与以前的click形式的事件相比将click放入on()里面使用起来更加灵活

**10.事件解绑**

* off()解绑事件
  + .off() 方法移除用.on()绑定的事件处理程序
  + off(); 解除所有绑定的事件
  + off('click'); 解绑所有绑定的 click 事件，元素本身的事件不会被解绑
  + off( 'click', “xx” );
  + off( 'click', “xx” ,fn);

**11. 坐标值操作**

* offset()获取元素关于相对于文档的位置,可以进行内部设置
* position()获取元素离元素最近具有定位属性的父元素的位置,不可以进行设置,只能获取
* scrolltop()获取网页卷进去的高度(即一个元素够长够宽,我在网页进行上下滑动网页覆盖的长度),可以是垂直或者水平方向
  + 可以对其设置固定值来进行一键到底或者一键到顶的操作

**12. 宽高**

* width(),height(),括号无参数获取元素的宽高(不包括border和padding)
* width(x),height(x),有参数则是为元素设置宽高, x:数值

**13. 事件对象event**

* event.data 传递给事件处理程序的额外数据
* event.currentTarget 事件绑定的对象(事件源), 和 this 相同
* event.pageX 鼠标相对于文档左部边缘的位置
* event.target 实际触发事件的对象， 不一定 === this
* event.type 事件类型： click， mouseover…
* event.which 鼠标的按键类型： 左 1 中 2 右 3
* event.keyCode 键盘按键代码

**14. 默认事件和冒泡**

* 以a元素为例,它有默认跳转的事件,通常有三种方式解决(不想让他跳转)
  + 在href中输入#
  + 在href中输入JavaScript:;
  + 在事件原script操作中加上 event.preventDefault();
* 冒泡事件:在一些操作中,点击父元素会连带内部的子元素一起点击,为了解决
  + 可在子元素中运行中输入event.stopPropagation();来解决

**15. 链式编程**

* 原理：return this; 调用”任何”一个方法都是返回了对象本身
* $('.box').text(); // 返回值 是 文本内容
* 通常情况下，只有设置操作才能把链式编程延续下去。因为获取操作的时候，会返回获取到的相应的值，无法返回 this。
* end()可以让链式编程中断
* end(); 结束当前链最近的一次过滤操作，并且返回匹配元素之前的状态。(猜想,若是一个事件突然转变可以使用,即原本一直往前冲现在想要往后跑,则可以使用end来给他停下,然后在进行往回跑的操作)

**16. each**

* 大部分情况下是不需要使用 each 方法的，因为 jQuery 的隐式迭代特性。
* 如果要对每个元素做不同的处理，这时候就用到了 each 方法
* 作用：遍历 jQuery 对象集合，为每个匹配的元素执行一个函数
* $(selector).each(function(index,element){});
* Element 是一个 js 对象，需要转换成 jquery 对象

**17. 多库共存**

* .noConflict();让 jQuery 释放对.*noConflict*();让*jQuery*释放对的控制权，让其他库能够使用$符号，达到多库共存的目的。此后，只能使用 jQuery 来调用 jQuery 提供的方法

**18. JQ插件库操作流程**

* 可参考http://learn.jquery.com/plugins/basic-plugin-creation/
* 任何一种jq插件的使用过程
  1. 下载插件库
  2. 在页面引入插件的css或者字体图片等（如果有的话）
  3. 在页面引入jQuery.js
  4. 在页面引入插件的js文件（core）
  5. 在页面通过插件的api初始化插件 即可使用（通过查看相应的API）

**19. eq补充**

* eq():当$(‘li:eq(1)’)时代表的是选择li的第一个元素,而当$(‘li)’.eq(i)时则是代表一种方法,选择li下标为i的元素

# H5新增+CSS3属性

**2019.8.19-2019.8.22**

##### 1. 新增input type类型

* email 限制用户输入必须为Email类型
* tel 手机号码
* url 限制用户输入必须为URL类型
* number 只能输入数字
* search 具有搜索意义的表单属性
* range 范围 滑动条
* color 拾色器
* time 时间
* date 选取日、月、年
* datetime 选取时间、日、月、年（UTC 时间）(移动支持)
* datetime-local 选取时间、日、月、年（本地时间）
* month 选取月、年
* week 选取周和年
* 部分类型是针对移动设备生效的，且具有一定的兼容性，在实际应用当中可选择性的使用。

##### 2. 表单

* 事件
  + oninput 用户输入内容时触发，可用于移动端输入字数统计
  + oninvalid 验证不通过时触发
* 网址: <input type="url" required oninvalid="alert('请填写此字段')"><br>
* tabindex (控制 input 标签按 tab 键获取到焦点的顺序)
* 姓名: <input type="text" tabIndex="1"> <br>
* 年龄: <input type="number" tabIndex="3"> <br>
* 电话: <input type="tel" tabIndex="2"> <br>
* 地址: <input type="text" tabIndex="4">
* 属性
  + autofocus 获取焦点
  + autocomplete 自动完成，用于表单元素，也可用于表单自身(on/off)
  + form 指定表单项属于哪个form，处理复杂表单时会需要 一般一个页面只有一个form
  + novalidate 关闭验证，可用于form标签
  + required 必填项
  + pattern 正则表达式 验证表单
  + autocomplete 是否保存用户输入值 默认为on，关闭提示选择off
  + formaction 主要应用于表单内某元素提交地址与整个表单提交地址不同，进行单个地址覆盖 如：
* <input type="submit" formaction="xxx.action">

##### 3. 多媒体

* 视频:video
* 音频:audio
* autoploy:自动播放,controls:是否默认开启播放控件,lap:循环播放,preload:预加载,widith:窗口宽度,height:窗口高度
* 对于多媒体文件,有的浏览器不兼容

解决方案:例

<audio>

<source src="素材/小手拍拍.mp3">

<source src="素材/小手拍拍.wav">

<source src="素材/小手拍拍.ogg">

</audio>

<video controls>

<source src="素材/movie.ogg">

<source src="素材/movie.mp4">

<source src="素材/movie.webm">

您的浏览器不支持

</video>

##### 4. DOM扩展

* 获取元素  
  document.querySelector("selector") html5新选择器，参数是css选择器参数,选择选中的第一个  
  document.querySelectorAll("selector") 选择多个  
  类似 jQuery 选择器，可惜出现的太晚了
* 类名操作  
  Node.classList.add('class') 添加class  
  Node.classList.remove('class') 移除class  
  Node.classList.toggle('class') 切换class，有则移除，无则添加  
  Node.classList.contains('class') 检测是否存在class  
  非常好用 但是出现的太晚了 。。。
* 自定义属性:  
  在HTML5中我们可以自定义属性，其格式如下 data-\*=""，例如:  
  data-info="我是自定义属性"，通过Node.dataset['info'] 我们便可以获取到自定义的属性值。  
  Node.dataset是以类对象形式存在的  
  当我们如下格式设置时，则需要以小驼峰格式才能正确获取  
  data-my-name="mm"，获取Node.dataset['myName']

##### 5. CSS3,浏览器兼容前缀

* Chrome: -webkit-  
  Firefox: -moz-  
  IE: -ms-  
  IOS: -webkit-  
  Opera: -o-  
  safari: -webkit-
* 在大多数情况下，当你需要使用CSS3规范中的属性且需要使用一个前缀的时候，针对所使用的浏览器，添加上面的前缀就可以了
* 对于那些功能需要添加前缀，可以浏览: [http://shouldiprefix.com/，处理添加前缀可以使用Autoprefixer插件，使用方式查看https://github.com/postcss/autoprefixer](http://shouldiprefix.com/%EF%BC%8C%E5%A4%84%E7%90%86%E6%B7%BB%E5%8A%A0%E5%89%8D%E7%BC%80%E5%8F%AF%E4%BB%A5%E4%BD%BF%E7%94%A8Autoprefixer%E6%8F%92%E4%BB%B6%EF%BC%8C%E4%BD%BF%E7%94%A8%E6%96%B9%E5%BC%8F%E6%9F%A5%E7%9C%8Bhttps:/github.com/postcss/autoprefixer)

##### 6. 属性选择器

* E[attr] 选择包含attr属性的所有元素
* E[attr=mydemo] 选择属性attr的值等于mydemo字符的所有元素
* E[attr\*=mydemo] 选择属性attr的值任意位置包含mydemo字符的所有元素
* E[attr^=mydemo] 选择属性attr的值开始位置包含mydemo字符的所有元素
* E[attr$=demos] 选择属性attr的值结束位置包含mydemo字符的所有元素

##### 7. 伪类选择器

* E:first-child 第一个子元素
* E:last-child 最后一个子元素
* E:nth-child(n) 第n个子元素
* E:nth-last-child(n) 同E:nth-child(n) 相似，只是倒着计算
* n是支持简单表达式的
* 注意: n 是从1开始的自然数 E:nth-child(0) 无效,E代表的是所选元素的父元素,(例: p:first-child p不是代表的p而是代表的p的父元素,即,p父元素的第一个元素,而不是p的第一个元素)
* E:target()可以做页面的一键到底/一件到顶的操作,E为所到元素的ID名字

##### 8. 伪元素

* E::before、E::after 默认行内元素 content 属性必须写
* E::first-letter文本的第一个字母或字
* E::first-line 文本第一行
* E::selection 被选中的文本
* ":" 与 "::" 用于区分伪类和伪元素

##### 9. 阴影

* 文本阴影
  + text-shadow: h-shadow(x) v-shadow(y) blur(模糊半径) color(颜色)
  + 水平偏移量 正值向右 负值向左( h-shadow(x))
  + 垂直偏移量 正值向下 负值向上(v-shadow(y) )
  + 模糊半径是不能为负值
  + 可以有多个影子，用逗号隔开
* 盒子阴影
  + box-shadow: h-shadow(x) v-shadow(y) blur(模糊半径) spread(扩展范围) color(颜色) inset(是否内嵌,可省略)

##### 10. 盒模型

* 通常给盒子设置border时会从外部出现border导致元素变大,为了能让元素不会因为设置border属性而变化可以使用:
* box-sizing:border-box/content-box,来解决,即border会在元素内部出现不会导致元素再变大,但会压制元素内部空间(通常使用:border-box)
* content-box: 对象的实际宽度等于设置的 width 值和 border、padding 之和 (默认方式)
* border-box： 对象的实际宽度就等于设置的 width 值，即使定义有 border 和 padding 也不会改变对象的实际宽度，即 ( Element width = width )

##### 11. 旋转/倾斜元素

* tronsform:rotate(); deg:度数 旋转多少度

##### 12. 圆角

* border-radius:,可有上下左右四个属性,可通过对其设置来获取圆等形状,若给长方形设置150px则为半圆

##### 13. 边框图片 border-image

* border-image-source 图片来源
* border-image-slice图片边框向内偏移量 不写单位,默认像素,也可以是百分比
* border-image-width边框宽度
* border-image-outset 边框图像区域超出边框的量
* border-image-repeat 图像边框是否应平铺(repeated)、铺满(rounded)或拉伸(stretched 默认) 可以写2个值,一个代表水平方向,一个代表垂直方向
* 可整体缩写(border-image: url(border.png) 30/ 27px/ 0.99 0.97 1.03 0.98 round round;)

##### 14. 渐变

* 线性渐变:
  + 沿着某条直线朝一个方向产生渐变效果,以上下左右四个角为原点以及对角
  + 语法: background: linear-gradient(direction, color-stop1, color-stop2, ...);
  + 例**:background: linear-gradient(to right, blue 20%, red 70%);**
  + direction: 方向 可以是 角度(10deg)顺时针 也可以是 to top/left/right/bottom
  + background: linear-gradient(to right ,red 60%,orange 80%);颜色后面可以跟百分比,表示从百分几开始渐变
  + 渐变的兼容写法 方向是相反的 而且不带to  
    background: -webkit-linear-gradient(right ,red 60%,orange 80%);
* 重复渐变:
  + 以上下左右四个角为原点以及对角
  + 语法:background: radial-gradient((shape? size? (at position?))?, start-color, ..., last-color)
  + 例:background: radial-gradient(circle at 10% 20%, red, yellow, green);
  + shape 渐变形状 : circle | ellipse(椭圆)
  + size 渐变大小:
  + closest-side（指定径向渐变的半径长度为从圆心到离圆心最近的边）
  + closest-corner （指定径向渐变的半径长度为从圆心到离圆心最近的角）
  + farthest-side（指定径向渐变的半径长度为从圆心到离圆心最远的边）
  + farthest-corner（指定径向渐变的半径长度为从圆心到离圆心最远的角）  
    也可指定大小: 需要注意的是 若渐变形状为圆形，则该渐变大小只能设一个数且不能为百分数，而椭圆既可以为具体数值也可以为百分数
  + 只传一个值默认形状为圆形,传入的是半径大小,不能为百分比;  
    传两个值则代表形状为椭圆形,第一个是x轴半径,第二个是y轴半径  
    圆心位置参数一定要置于radial-gradient()第一个参数的末尾，顺序千万不能放反了

##### 15. 背景图片加强

* 例:background: url("img/1.jpg") no-repeat left top, url(img/2.jpg) no-repeat right top;
* 以逗号分隔可以设置多背景，可用于自适应局
  + right top以右上为参考点
* background-size设置背景图片的尺寸
  + 是需要根据高度还是宽度适配
  + width height 直接设置宽高 百分比
  + cover会自动调整缩放比例，保证图片始终填充满背景区域，如有溢出部分则会被隐藏。
  + contain会自动调整缩放比例，保证图片始终完整显示在背景区域(整个背景图片完整显示在背景区域.)。
  + **不会对图片产生拉伸变形(图片比例保持不变),但是cover 会有部分内容看不到 contain 保证所有内容都内看到**
* background-origin(原点，起点)设置背景定位的原点
  + border-box以边框做为参考原点；
  + padding-box以内边距做为参考原点；
  + content-box以内容区做为参考点；
* background-clip设置背景区域 clip(裁切)
  + border-box 裁切边框和内容区
  + padding-box 裁切padding和内容区
  + content-box 裁切内容区

##### 16. 过渡 transition

* 可作用于普通的 CSS 属性（如 background 、opacity ...），也可以作用于 CSS3 出现时新引入的
* 过渡效果需要触发,一个开始一个结束,即开始状态和过渡状态,以class=d为例
  + d{}时是开始状态即元素的初始形态
  + d:hover{}则是元素触发后的结束状态
* transition:简写属性，用于在一个属性中设置四个过渡属性。
  + transition: 属性 持续时间 运动函数 延迟时间(可以省略)
  + transition: all .4s ease 1;所有能改动的属性,花费4S时间从开始状态到结束状态,延迟1S后执行
* transition-property:规定应用过渡的 CSS 属性的名称。
* transition-duration:定义过渡效果花费的时间。默认是 0。
* transition-delay设置过渡延时 超过时间后执行动画.默认是0(即不添加时间)
* transition-timing-function:规定过渡效果的时间曲线。默认是 "ease"。(元素的行走路线通常不需要改,使用ease即可)

##### 17. 2D转换(transform)

* 2D变换是以X轴Y轴来进行变换的
* 2D转换是位置上变化(位移) ,水平方向,垂直方向
* 通常是和transition一起使用,即transition在元素开始状态定义变动的属性,时间等,transform在元素被处触发后的hover里定义元素点击及点击后的状态
* 效果只做一遍,做完后返回原本的状态
* 属性transform
  + translate(x, y):移动,可以改变元素的位置，x、y可为负值；可单独写translateX和translateY
  + rotate(deg):旋转,可以对元素进行旋转，正值为顺时针，负值为逆时,也可以单独写成rotateX(),与rotateY();
  + scale(x, y):缩放,可以对元素进行水平和垂直方向的缩放，x、y的取值可为小数，不可为负值；
  + skew:让元素沿着X轴Y轴角度倾斜.例:transform:skew(20deg, 30deg);可单独写skewX(x-angle),skewY(y-angle)
* 方法
  + transform-origin: right bottom; :旋转中心 右下

##### 18. 3D转换

* 3D转换是X轴Y轴Z轴的变化
* 它的方法与2D的方法近似,很多都可以使用
* **transform-style: preserve-3d; 开启3D效果,如果没有这个是没3D效果的,使用3D时必须在父元素设置**
* backface-visibility: visible;开启3D效果后,背面是否能见visible/hidden(可见/不可见)
* 影深(立体感) perspective 数值越大离得越远效果越和原来没区别,数值越小离得越近,效果越震撼(近则立体感好,远则立体感不好).
  + 作为一个属性，设置给父元素，作用于所有3D转换的子元素
  + 作为transform属性的一个值，做用于元素自身
  + perspective:数值;

##### 19. Web字体

* 在link中引入字体的CSS样式
* 在style中使用 @font-face {  
  font-family: 'web';给要调用的字体样式命名  
  src: url('webfont.ttf');引入字体样式,字体样式可以有多种  
  src: url('webfont.woff');  
  src: url('webfont.woff2');}  
  将下载字体引入html
* 在需要使用字体的类中使用字体(web)

**BFC**

**什么情况下可以让元素产生BFC(块格式化上下文)**

display 属性为 block, list-item, table 的元素，会产生BFC.  
以上盒子具有BFC条件了，就是说有资质了，但是怎样触发才会产生BFC，从而创造这个封闭的环境呢？  
同样，要给这些元素添加如下属性就可以触发BFC。

* float属性不为none
* position为absolute或fixed
* display为inline-block, table-cell, table-caption, flex, inline-flex
* overflow不为visible。

**BFC元素所具有的特性**

BFC布局规则特性：

1. 在BFC中，盒子从顶端开始垂直地一个接一个地排列.
2. 盒子垂直方向的距离由margin决定。属于同一个BFC的两个相邻盒子的margin会发生重叠  
   3.在BFC中，每一个盒子的左外边缘（margin-left）会触碰到容器的左边缘(border-left)（对于从右到左的格式来说，则触碰到右边缘）。
3. BFC的区域不会与浮动盒子产生交集，而是紧贴浮动边缘。
4. 计算BFC的高度时，自然也会检测浮动或者定位的盒子高度。  
   它是一个独立的渲染区域,并且与这个区域外部毫不相干。  
   白话文： 孩子在家里愿意怎么折腾都行，但是出了家门口，你就的乖乖的，不能影响外面的任何人。

**BFC的主要用途**

BFC能用来做什么？  
(1) 清除元素内部浮动  
只要把父元素设为BFC就可以清理子元素的浮动了，最常见的用法就是在父元素上设置overflow: hidden样式，对于IE6加上zoom:1就可以了。  
主要用到

计算BFC的高度时，自然也会检测浮动或者定位的盒子高度。

(2) 解决外边距合并问题  
外边距合并的问题。  
主要用到

盒子垂直方向的距离由margin决定。属于同一个BFC的两个相邻盒子的margin会发生重叠

属于同一个BFC的两个相邻盒子的margin会发生重叠，那么我们创建不属于同一个BFC，就不会发生margin重叠了。  
(3) 制作右侧自适应的盒子问题  
主要用到

普通流体元素BFC后，为了和浮动元素不产生任何交集，顺着浮动边缘形成自己的封闭上下文

**BFC 总结**

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。包括浮动，和外边距合并等等，因此，有了这个特性，我们布局的时候就不会出现意外情况了。