**资料来源：网易云课堂视频内容**

**本文作者：王冬**

**qq：20004604**

**CSDN博客：<http://blog.csdn.net/qq20004604>**

**（1）web前端应用范围**

网页、app、甚至是各种pc的软件（以浏览器形式）、虚拟现实效果（hellorun.helloenjoy.com）、

**（2）一个系统的协作**

按顺序来：

①策划；

②交互；

③视觉和前端一起做；

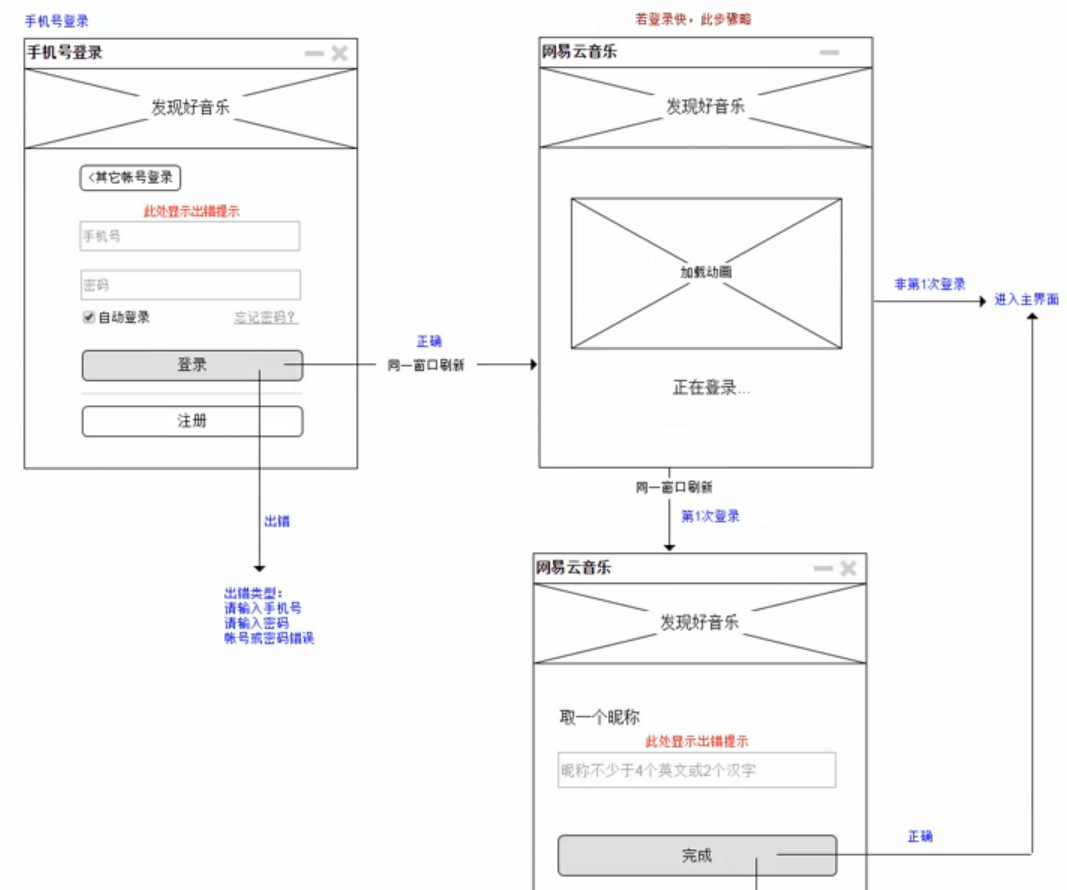
④前端和后端一起做；

⑤最后测试；

**（3）前端的职责**

①将视觉稿和交互稿，通过前端开发，形成UI；

②交互稿示例：



③视觉稿：

简单来说，就是说这个页面是什么样子，随便找一个app，截图某个界面下来，就可以认为是视觉稿；

涉及图标、距离、空间、布局、高、宽、

④前端就是把交互和视觉，实现成UI。

⑤视觉稿——》前端——》页面制作；

⑥交互稿——》前端——》页面逻辑开发；

⑦页面制作能力要求：PS、HTML、CSS；

⑧页面逻辑能力：JavaScript、DOM；

⑨HTML用来定页面的结构；

CSS用于定页面的样式；

JS用来定页面各种内容的响应、事件等；

PS用于切图（把图取出来并使用）；

DOM和HTML有关，定页面结构等；

⑩页面制作的进阶是页面架构；

页面制作 + JavaScript程序设计 的进阶是DOM编程艺术；

页面架构和DOM编程艺术的进阶是产品前端架构；

**（4）PhotoShop（简称PS）之切图**

所谓切图：

①从设计稿中切出自己想要的图（图标、背景图等）；

②在代码中引用；

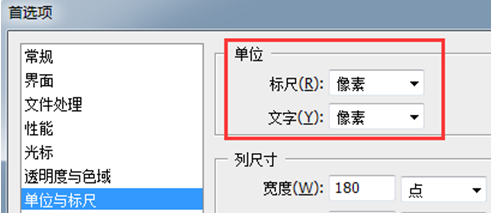
HTML通过img标签引用图片，<img src="路径">

CSS通过background-image:url(路径);

**（5）PS工具**

设置：

①编辑——首选项——单位和标尺——单位（选择像素）；



②跟前端切图相关的面板有：

窗口菜单下的工具、选项、图层（默认打开），还有信息、历史记录；（需要将其拉到右边栏自己习惯的位置）

将其他的没用的都可以关掉；

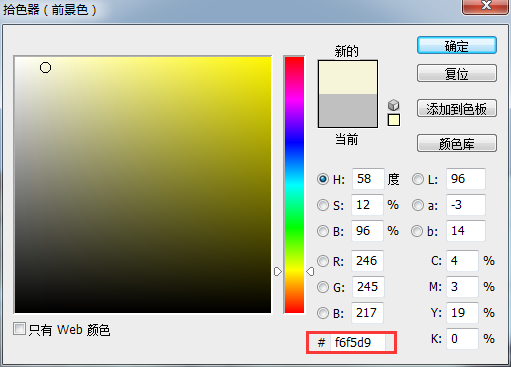
然后通过存储工作区来保存起来。

③切图常用工具：

移动工具、矩形选框工具、魔棒工具、裁剪工具+切片工具、缩放工具（Ctrl+加号或者减号）、取色器；

具体略，只说取色器，如图可以得到色彩的编号





④视图菜单下勾选：对其、标尺、显示额外内容、参考线（需勾选显示额外内容）

**（6）视觉稿的测量**

①图片的height和width；

②内边距和外边距（padding、margin）；

③边框（border）；

④定位（position）；

⑤文字大小（font-size）；

⑥行高；（http://www.cnblogs.com/dolphinX/p/3236686.html）

⑦背景图位置（background）；

关于测量：

①应将画布尽可能放大，减少误差；

②用矩形选框工具，查看信息面板；

③行高：从第一段文字最下面到第二段文字最下面，或者直接查看。

④字体大小（font-size）：文字最上面到最下面。

关于矩形选框工具：

①添加到选区：shift

②从选区中减去：alt

③和选区重合的部分：shift+alt

④选取很大的图时，画一个小框，最左边挨着区域的最左边，再画一个小框，最右边挨着区域的最右边，然后从信息面板可以看到（利用shift）

**（7）视觉图的取色**

靠取色工具取色

①边框色（border-color）

②背景色（background-color）

③文字色（color）；

不会的，请学习ps先……

可以这么做：

①确定是否是纯色（取色器多点几个地方）；

②是否是线性渐变（用魔棒）；

**（8）切图**

目标：

①修饰性（图标、logo、特殊效果的文字、按钮、非纯色背景）；

保存为PNG24（半透明，质量高，IE6不支持半透明）、或者PNG8（都支持全透明）；

②内容型（Banner(横幅)、广告图片、文章配图）；

保存为jpg（因为颜色丰富，可能需要压缩）；

取背景：

①隐藏文字取背景；

②合并在一起的，先在图标上没文字的地方选取一块区域，然后用“自由变换”（Ctrl+T），（对有纹理部分，还需要使用移动工具（+Alt））将这块区域覆盖到文字之上，于是文字没有了，截取即可。

保存图片：

①可以将选取的合并到一个图层，然后只显示这部分，保存；

②或者拖到新文件中保存；

魔棒工具：

①可以用魔棒工具，然后按alt选取图标（可以智能选）；

平铺：（讲的不太明白）

①以x轴，那么宽填充为100%，切图时要填满；

②y轴同理

切片工具：

①当没办法抠图时，比如效果很炫，那么用切片工具将其切出来（分为几部分），统一保存；

②貌似用的很少；

**（9）保存**

①存图快捷键：Ctrl + C（复制）Ctrl + N（新建）Ctrl + V（粘贴），

②保存类型：

色彩丰富、可能有透明度要求，用jpg（质量60~80，因为web用）；

色彩不丰富，无透明度要求，用PNG-8

半透明要求，PNG-24

有必要的情况下，保存一份PSD文件，在PSD文件上修改，保存，特别是对某些不可逆的操作而言。

③对多图标在同一文件时，导入html的时候貌似是有讲究的（具体下面看）

**（10）图标的修改、移动**

①可以更改画布大小alt + ctrl + c

②画布据说比图片大一点好

③修改PNG8图片时，将PNG8图片的 图像-模式 更改为RGB颜色模式（因为有时候默认不是，导致颜色不正确）

**（11）将图片导入**

如代码：

HTML：

<button class="u-btn">click me</button>

CSS：

①对于单图标在某个图片时：

.u-btn{ background:url(图片链接) no-repeat 0 0;}

②对于多图标在某个图片时：

u-btn{ background:url(图片链接} no-repeat 10px -30px;}

注：

no‑repeat后面的坐标，第一个是x坐标，第二个是y坐标。

在显示框不动的情况下，

-30px表示将原图沿着y坐标向上30px，

10px表示将原图沿着x坐标向右10px；

不理解的自己P个图加到html试试就知道了。

**（12）图片的合并使用**

①将多个图标放到一个图片之中使用

②称为Sprite拼图

③可以减少网络请求，提升网页加载速度

④反正这么干就对了

⑤不同图标之间保留一定的空隙

⑥排列方式：横向、或者纵向

⑦图片的合并：

将同属于一个模块的图片，进行分类合并；

大小、颜色相近的图片进行合并；

同一页的图片合并

有状态（例如“点击关注”和“已关注”）的图片合并

⑧兼容性问题：

为低级版本浏览器存一份低级版本（原是PNG24，低级的是png8）的图片

⑧

**（13）图片的压缩**

①无损压缩工具：Minimage

②有损压缩工具：TinyPng

**（14）开发工具**

文本编辑器：

①Sublime Text（根据个人体验感觉，插件很多，但是用起来上手有点复杂）

②Notepad++（真不觉得好用，太清爽了）

③EditPlus（同上）

④UltraEdit（我正在用的，带自动补完、染色和缩进，虽然不是很完美，但可以用）

IDE（集成开发环境）：

①WebStorm

②InterlliJ IDEA

③Eclipse

**（15）调试**

①可以使用chrome

②IE8以上有调试功能。

关于IE6的兼容效果（安装IETester），来查看IE6的效果

**（16）标签**

①文档声明，用<!DOCTYPE html>，支持html5，首行、顶格写，（有时候不加也没事，但最好加上）；

②<meta charset="utf-8">定义了文档的字符编码（这里是UTF8），一般都是utf-8

<meta name="keywords" content="游戏…">

<meta name="description" content="网易…">

用于描述页面的特征的（对搜索引擎有用）；

<meta name="viewport" content="width=dev-width">

对移动端浏览器有效，具体之后说

③<link rel="shortcut icon" href="xxx.ico">



用于红框处的图标。默认是加载根目录下的ico文件的（但最好还是加上这个吧）

但rel后面的是什么意思？下同

④<link rel="stylesheet" href="xxxx.css">

引入的样式表，这里的rel似乎最好不要改

⑤HTML5的标签（之前用div来定义）

<header>表示头部

<footer>表示尾部

<nav>表示导航栏

<aside>表示比如侧边栏、广告、插入的工具等

主体结构：

<article>表示独立的，可重复的结构，比如帖子、新闻、评论等，一般他包含<header>和<footer>标签，可以嵌套在<article>中

<section>表示文档区域中的一个区域或者一节，比如说内容的专题组，一般带个标题，<sction>之间是有相关性的。

⑥<h1>~<h6>表示标题，重要性依次递减。或者说，大小不同，h1最大，h6最小

⑦<a>超链接标签，指向另一个文档、文档内部锚点、email地址。

属性target="\_self"表示当前窗口，

target="\_blank"是新窗口

target="inner"表示在某个区域内打开这个页面。需要和<iframe frameborder="0" name="inner"></iframe>中的name属性一致（也就是说可以把inner改成其他的）

⑧强调标签：<strong>粗体，<em>斜体

<span>没有任何语义（预设的），因此可以结合class

粗体也可以用<b>，但<strong>更表示强调（虽然我感觉看起来一样）

麻蛋，这斜体和斜体之间有啥区别？

⑨<q>引用，效果是加了双引号

<cite>引用，效果是微斜

**（17）段落相关**

①<p>段落，独立的一行，带换行效果

②<ul>缩进，**无序列表**

<ol>缩进，**有序列表**

可以用<ol type="a" start="2">来控制其有序的顺序（a表示字母——不能用b或aa，2表示第二个，这个表示排序从b开始，到z后变aa）（另外，type用大写的i表示罗马数字）

<dl>不缩，**自定义列表，<dt>不缩，<dd>缩进，**一般是一个<dt>后面跟多个<dd>

<li>可以放在<ul>或者<ol>之中，段落之前带一个点（默认）或者序号

列表可以嵌套。

**HTML代码：**

<ul style="width:100px">

<li>点击点击点击点击</li>

<li>点击点击点击点击</li>

</ul>

<ol style="width:100px">

<li>点击点击点击点击</li>

<li>点击点击点击点击</li>

</ol>

<dl style="width:100px">

<dt>作者（测试测试）</dt>

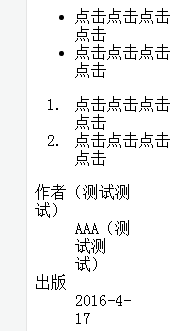
<dd>AAA（测试测试）</dd>

<dt>出版</dt>

<dd>2016-4-17</dd>

</dl>

**效果图：**



<pre>表示可以保留里面的换行符和空格

**HTML代码：**

<pre> abc abc</pre>

<p> abc abc</p>

**效果图：**



③<blockquote>大段的引用（但如果需要像论坛帖子里那种引用的话，还需要加其他样式），例如：

**HTML代码：**

<blockquote>

<pre> abc abc</pre>

<p> abc abc</p>

</blockquote>

**CSS代码：**

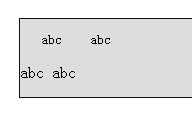
blockquote{

background-color:#ddd;

border:1px double #111;

}

**效果图：**



问题：

①宽度需要调整为自适应（不过还没搞明白怎么调，估计是在具体的块里，用width限制吧）

**（18）嵌入标签**

①<img>，图片标签

例子：

<img src="2.png" alt="显示失败">

效果：加载图片链接为2.png，如果显示失败，根据alt里面来显示（chrome和360默认是图片里面内嵌文字显示，firefox是以文字显示）

②<iframe>，嵌入页面

例子：

<iframe src="http://www.163.com"></iframe>

注意：

* ①外链必须加http://，否则无法加载。
* ②默认大小貌似是有范围的（不是用加载的链接大小来加载），但可以通过加入style属性来调整。
* ③嵌入页面的操作不会导致整个页面的刷新。

③<object>引入外部资源

以网易课堂为例：（因为我自己没有啊）

<object type="application/x-shockwave-flash">

<pargam name="movie" value="http://pdfReader.swf">

</object>

type表示插件的类型的类型；

pargam标签表示插件的参数，

需要用的时候查看：**页面制作——HTML——标签——第5个视频（约2分50秒左右）**

下略

④<video>是插入视频标签

例子：

<video autoplay loop controls="controls" poster="1.png">

<source src="xxxxxxxx.mp4" type="video/mp4">

<track kind="subtitles" src="xxx.srt" label="English">

你的浏览器无法查看

</video>

其中，

①controls表示控制浏览器的进度条，poster是封面，autoplay表示自动播放（不加就不会自动播放），loop表示循环播放

②source一般有多个，用于支持多种浏览器（比如火狐貌似就不支持mp4），因此也可能需要准备多种类型的视频。src是链接，type是类型名（更多可以查看上面蓝字同一视频更后面的内容）

③文字表示假如无法播放的话，将会显示的内容

④track标签表示引入的字幕

关于<video>兼容性：

https://en.wikipedia.org/wiki/HTML5\_video#Browser\_support

关于<audio>兼容性：

https://en.wikipedia.org/wiki/HTML5\_Audio#Supported\_audio\_coding\_formats

⑤嵌入图片

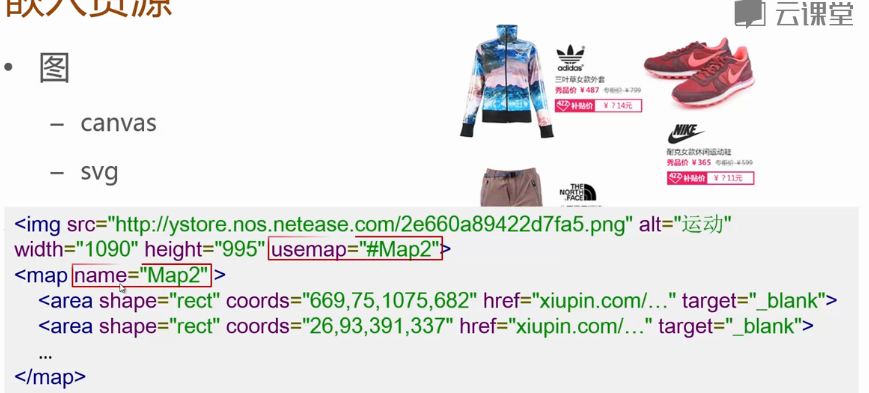
<canvas>基于像素

<svg>基于矢量

以上两个可以用于图表，但这只是容器，具体绘制需要有脚本，并且这是HTML5的标签

<map>可以控制在点击不同区域来执行不同的指令（例如打开不同页面）

具体如图：



<area>是热点区域（该范围之内）rect是矩形，左上和右下两个点。href是将打开的链接，target是新窗口模式打开

**（19）表格**

<table>标签

标题：<caption>

表头：<thead>

* 一行用<tr>，行内的每项用<th>

表格内容：<tbody>

* 一行用<tr>，行内的每项用<th>

表格尾部：<tfoot>

属性：

①<td colspan="2">

加入标签内，表示跨列，=2表示2列（可以理解为合并单元格），写在<tr>标签内，很好理解。

②<th rowspan="2">

表示跨行，跨2行

具体写法是：

<tr><th rowspan="2">跨行</th><th>A</th><th>B</th></tr>

<tr><th>a</th><th>b</th></tr>

<th>写在跨行的起始行的<tr>标签内部，假如跨两行，那么下一个<tr>直接写第二格的内容

③给表格加上边框：（添加css属性）

table{ border-collapse:collapse; }

th,td{ border:1px solid #000;}

**HTML代码：**

<table>

<caption>测试用表格标题</caption>

<thead>

<tr><th>表头</th><th>第一项</th><th>第二项</th></tr>

</thead>

<tbody>

<tr><th>姓名</th><th>王</th><th>冬</th></tr>

<tr><th>爱好</th><th>赚钱</th><th>做好人</th></tr>

<tr><th rowspan="2">跨行</th><th>A</th><th>B</th></tr>

<tr><th>a</th><th>b</th></tr>

</tbody>

<tfoot>

<tr><td colspan="2">无聊的话~~</td><td>就学习</td></tr>

</tfoot>

</table>

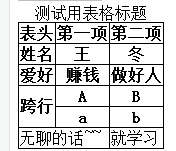
**CSS代码：**

table{ border-collapse:collapse; }

th,td{ border:1px solid #000;}

</style>

**效果图：**



**（20）表单**

①<form>标签

②用于向服务器提交

③样例：<form action="/login" method="post">

action是提交的地址，或者哪个后台的接口；

method是表单提交的方式。

④<fieldset>用于分区，比如提交的表单有多种分区

<legend>是分区的标题

⑤<input>标签

type="file" 是文件（下略type）

checkbox 是多选框

radio 是单选框

text 是文本框（默认）

submit是提交的按钮

reset是重置

以上2个可以用button来替换，加上type="submit"或者是reset

属性name是数据的名称，value是该input的值

属性名：

readonly文本框不能被修改

placehold="" 未输入时的背景文字

value="" 默认文字

hidden内容被隐藏（不显示）

⑥<select>是下拉标签

具体每一项在其内加<option value="数值">内容</option>

<label>标签是说明标签，一般加入for="被说明的id属性"。虽然我并不明白这个for有什么用

⑦selected属性表示默认选中，disable表示不能被选中。

⑧单选／多选框其name属性要一致

⑨下拉框可以加入<optgroup>分组标签，用于将几个分为一组（具体请使用下面的html代码体会）

⑩<textarea></textarea>多行文本

rows是显示的行数

cols是显示的列数（英文字母数）

决定文本框大小，可以不写

**HTML代码：**

<form action="/login" style="width:400px" method="post">

<fieldset>

<legend>测试表单</legend>

<fieldset>

<legend>多选</legend>

<input type="checkbox" value="1" id="c\_1" name="size"><label>多选框1</label>

<input type="checkbox" value="2" checked id="c\_2" name="size"><label>多选框2默认选中</label>

</fieldset>

<fieldset>

<legend>单选</legend>

<input type="radio" value="3" id="r\_1" name="rrr"><label>单选1</label>

<input type="radio" value="4" id="r\_2" checked name="rrr"><label>单选2（默认选中）</label>

</fieldset>

<fieldset>

<legend>文件上传</legend>

<input type="file" id="file">

</fieldset>

<fieldset>

<legend>下拉菜单</legend>

<select id="choice">

<optgroup label="下拉分组">

<option value="1">选项1</option>

<option value="1">选项2</option>

</optgroup>

<option value="1" selected>选项3（默认选中）</option>

</select>

<label for="choice">下拉框选择</label>

</fieldset>

<fieldset>

<legend>文本框</legend>

<label>背景文字：</label><input id="words" placeholder="文本提示，不输入时显示"><br>

<label>默认文字：</label><input id="words" value="默认有文字"><br>

<label>只看不改：</label><input id="words" readonly value="不能被修改"><br>

<label>多行：</label><textarea id="words" rows="5" cols="30">多行</textarea>

</fieldset>

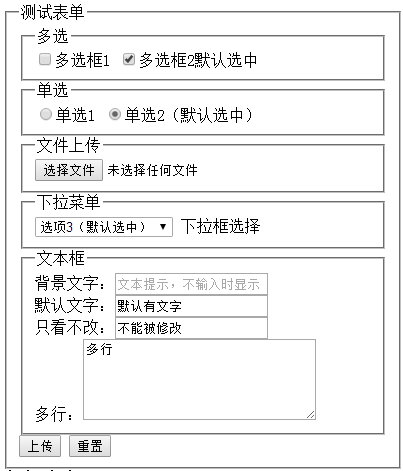
<button type="submit">上传</button>

<button type="reset">重置</button>

</fieldset>

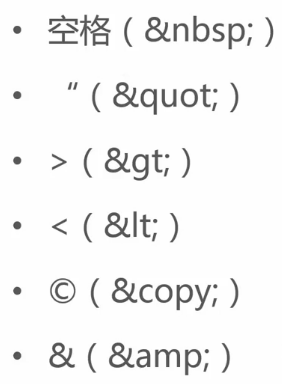
</form>

**效果图：**



**（21）实体字符**

常用的，直接上图：



**（23）样式表**

引入外部样式表：

<link rel="stylesheet" href="xxx.css">

内部样式表：

<style>

xx

</style>

**（24）伪元素选择器**

①开头是2个冒号（CSS3开始）

**（25）选择第一个字母：**

**::first-letter{color:red; }**

效果：

①每行的第一个字母颜色变红。

②多行的话，每一行都有效；

③有效的包括<p><ul><li>等；

④简单的来说，对算是段落的有效；

**（26）选择第一行**

**::first-line{color:red;}**

效果：

①第一行颜色变红；

②有效范围应该跟上面一样吧；

**（27）在某个之前或者之后插入**

**::before{content:"插入的内容";}**

效果：

①在符合选择器的内容之前，插入content后面引号包括起来的；

②如果单独用这个的话，容易出问题（因为是在所有段落之前）；

③所以建议和别的结合使用，例如：

**li::before{content:"ww";}**

**li::after{content:"end";}**

对li标签之前插入“ww”，之后插入“end”

**（28）修改选中内容的样式**

**::selection{color:red;background-color:#ddd}**

效果：

①被选中部分，文字颜色和背景颜色被更改；

②似乎是不能改变文字大小，或者其他的；

**（29）后代、子、兄弟**

**后代：**（即在该标签范围内的所有标签）

①假如要选择class="abc"的所有后代中class=“def"的；那么使用：

**.abc .def{color:red;}**

**子：**（直系的，子的子不算）

①html：

<ul class="abc">

55555

<li>6666</li>

<span>

<li>777</li>

</span>

</ul>

②CSS：

**.abc>li{color:red;}**

③效果：

6666颜色变红，777没变化；

**相邻兄弟（后一个）：**

①html：

<ul>

<li>555</li>

<li class="abc">aaa</li>

<li>666</li>

<span>

<li>777</li>

</span>

<li>888</li>

</ul>

②CSS：

**.abc+li{color:red;}**

③效果：

666颜色变红，其他无变化；

④结论：对符合要求的标签之后的第一个标签，如果符合，则变色。

**相邻兄弟（后面所有）：**

①html：

<ul>

<li>555</li>

<li class="abc">aaa</li>

<span>

<li>777</li>

</span>

<li>666</li>

<li>888</li>

</ul>

②CSS：

**.abc~li{color:red;}**

③效果：666和888颜色变红，之前的555没变化，兄弟的子777也没变化；

即从该元素开始往后的所有符合要求的兄弟元素。

**（30）共享样式**

①语法：

**h1,h2{color:red;}**

②效果：

标签是h1或者是h2的，文字颜色变红。

③分割符号是逗号

**（31）设置字体**

①语法：

**font-family:"Microsoft YaHei";**

②效果：

设置字体为微软雅黑（值表示微软雅黑）；

**（32）继承**

所谓继承，就是指父标签的样式，如果子标签没有特殊设置的话，将使用父标签的

①可以被继承的属性：

color——字体颜色

font——字体（family，size）；

text-align——文字停靠（居中/靠左等）；

list-style——简写属性在一个声明中设置所有的列表属性（大概就是说<ul>等之间的<li>之前的标记相关，如标记是图片？在哪放？等具体参照http://www.w3school.com.cn/cssref/pr\_list-style.asp）；

②非继承属性：

background：背景（颜色等）

border：边框

position：位置

其他

③查看继承：查看相关文档的Inherited属性

**（33）CSS的选择器的优先级**

①根据权重来计算，当样式冲突时，根据权重大小来决定选择哪个

②权重系数：

行内样式：1000 ——如<p style="color:red">xxx</p>

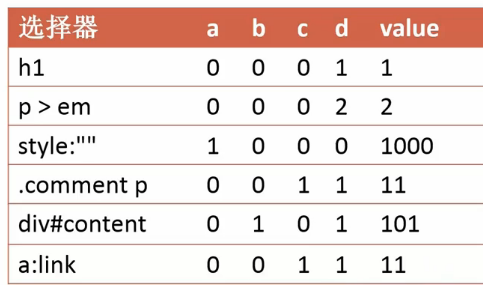
id选择器：100 ——如#menu{color:blue;}

类、伪类、属性选择器：10 ——如 .tooltip{color:green;}

标签选择器、伪元素选择器：1 ——如 p{color:black;}

③当组合使用时，则相加，例如p#menu则表示标签选择器＋id选择器，其权重为101。

示例：



④当优先级一样时，且面对同一个元素，描述同一个样式，那么后面的覆盖前面的。

如果描述不同样式，则合并（即两个都生效）；

如果优先级不一样，描述同一个样式，那么优先级高的覆盖低的。

⑤改变优先级：

<1>改变先后顺序（对优先级相同的有效）；

<2>组合使用选择器（比如加个标签选择器，其总权重提升）；

<3>在属性后面（具体某一个，例如color:red的后面），加关键字“!important”。如：color:red !important 这样，则表示将使用这个。

**（34）文字字体**

①**font-size**：文字大小

②值有：length（具体值）、absolute-size（绝对值，用的少）、relative-size（相对值，用的少）、percentage（百分比）

length：如15px（15px大小），2em（相对父元素属性的2倍大小）

percentage：如200%（类似2em，表示2倍大小）；

③**font-family**：文字类型

④值有：family-name（文字类型名）、generic-family（通用类型）

family-name：如

generic-family：如serif、sans-serif、cursive（草书）、fantasy（幻想体）、monospace（等宽字体）。

⑤当一行样式，font-family的值同时有多个并列时，优先使用前面的，如果前面没有该字体（例如是英文字体，不是中文字体），那么使用后面里靠前的。

也就是说，假如前面的中英文都包含，那么后面的是无效的（如果她是描述中文或者英文的话）；

⑥**font-weight**：字体加粗

值：normal（没变化）、bold（加粗）、bolder（相对很少用）、lighter（很少用）。

值：100、200等一直到900，一般字体只支持400和900

⑦**font-style**：字体斜体

值：normal（正常）、italic（斜体，字体有斜体的话用这个，一般都用这个）、oblique（斜体，强制倾斜，字体不带斜体的话也倾斜，一般不用）

⑧**line-height**：行距

值：normal（非固定值，浏览器决定，一般是1.1~1.2之间）、number（数值，例如1.1）、length（例如35px，或者3em，1em的大小是字体大小）、percentage（百分比，如150%）。

百分比和number之间的区别（即200%和2之间的区别）：百分比不受具体某一行（子标签）的style的设置（统一使用，不针对某个而变化）；而number受具体某一行（某个子标签的font-size）的影响（直接继承）。

也就是说，假如父标签里某一个子标签的字体大小不同，number会实时变化，百分比什么时候都一样。

⑨缩写：font: [ <font-style> || <font-weight> ] ? <font-size> [ / <line-height> ] ? <font-family>

简单来说，line-height在具体值前应该加一个“/”。

注意：缩写情况下**font-size**和**font-family**是必填项，必须都有，如果少了的话，则该属性不起作用。

例子：**font: italic bold 40px /200% fantasy;**

表示斜体，加粗，字体大小40px，行距200%，字体幻想体。

⑩**color**：字体颜色

值：

文字（如red）、

六位写法（如#ffffff 表示白色，也可以缩写为三位写法）、

rgb写法（如rgb(0,0,0)表示黑色），

rgba写法（在rgb写法的基础上，最后加一个参数表示透明度，如rgb(0,0,0,0.5)表示黑色，50%透明度，1表示不透明）

color:transparent 表示全透明

**（35）对齐方式**

①**text-align**：值

left（左对齐）；

right（右对齐）；

center（居中）；

justify（两端对齐）（需要超过一行，并且不对最后一行生效。汉字会改变间距，英文会对单词之间的空格进行调整）；

②**vertical-align**：垂直对齐

值：

baseline（基线对齐）；

sub（下标）；

super（上标）；

top（对齐到当前行的最高点）；

text-top（文本的最高点）；

middle（垂直居中）；

bottom和text-bottom（和上面的对应）；

百分比：（参照行高line-height，参照物也是baseline为起点）；

length：（如10px，针对以baseline为起点，往上10px，也可以是负数，则往下）

注意：该属性是子元素的，对自己本身生效，因此同一父元素下面不同子元素可以有不同这样的属性。

③**text-indent**：首行缩进

值：

length（1em是1个文字大小，常用是2em；也可以是10px，表示缩进固定10px值一般不用。如果是个负值，则文字容器外移动，如果很大，则离开浏览器，可以用于SEO，不影响正常使用但可以对搜索引擎生效）；

percentage（百分比，指整个容器宽度，20%则缩进整个容器的20%的宽度）

④**white-space**：空格、换行、tab、是否自动换行

值：

normal：普通（浏览器决定），通常换行、tab、空格会被合并为一个空格，会自动换行；

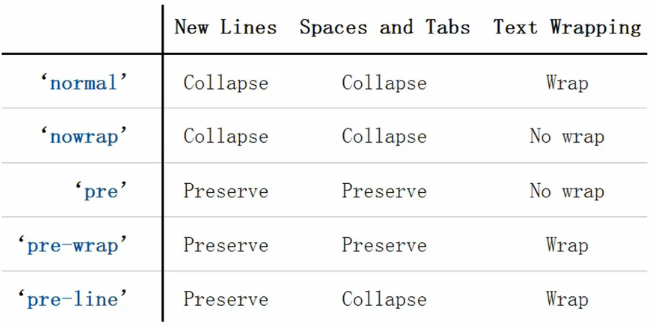
nowrap：在normal的基础上，不换行（如果一行写不下，会超出）；

pre：换行保留，空格和tab会保留，不会自动换行（完整保留在代码中的格式）；

pre-wrap：在pre的基础上，允许自动换行。

pre-line：保留换行，空格、tab合并，允许自动换行。

如图：



⑤**word-wrap**：单词自动换行（但单词会保持完整）

值：normal（不自动换行）、break-word（允许单词自动换行）；

效果：假设本行放不下，那么放到下一行

⑥**word-break**：单词自动换行（单词不保持完整）

值：normal（不变）、keep-all（和normal类似）、break-all（允许任意两个字母断开）。

效果：假如本行放不下，本行放满，剩下的字母放到下一行

**（36）文本修饰**

①**text-shadow**：文字阴影

值：

none（无阴影）；

length（x轴偏移方向，y轴偏移方向，阴影模糊半径(虚化)可省略，再加一个颜色可省略将自动用文字颜色）；

②**text-decoration**：下划线

值：

none（不加线）；

underline（下划线）；

overline（上划线）；

line-throught（中线划线）；

注：三种线可以同时存在，互不干扰，用空格隔开

**（37）文本高级设置**

①**text-overflow**：文字溢出时设置

clip：默认值

ellipsis：（溢出部分设置为三个点）

注意：text-overflow:ellipsis需要结合overflow:hidden和white-space:nowrap一起使用，表示不换行，并且溢出部分将自动隐藏，用“...”替代。

②**cursor**：定义鼠标形状

值：

[<url>,] 图片

auto默认的，自动处理的

default鼠标光标，

none 消失

help 带个问号的

pointer 手型

zoom-in 放大镜

zoom-out 缩小镜

move 一个有四个方向的图标

row-resize/col-resize 拖拽的切分条用的那种图标

例子：

**cursor:pointer; 指定为手型**

**cursor:url(xx.cur), pointer;指定为图片，如果图片失效则为pointer**

这里指的是当鼠标移动到目标区域后的变化

③**inherit**：强制继承

当css的某个属性的参数为inherit时，强制继承其为父元素的值。

原因是，有的css属性，可能不继承父元素的，例如border，那么在子元素中使用border:inherit，将强制子元素使用父元素的border属性

**（38）让一个块水平居中**

**margin:0 auto**

0代表上下边距是0，auto代表左右边距（相对于父元素）为auto，效果是左右自动居中。

**（39）圆角**

①**border-radius：x方向半径{1~4个} /y方向半径{1~4个}**

注意，x和y之间有“/”

这里的两个参数，可以理解为x方向受影响的线段的长度和y方向受影响的线段长度

4个时候，顺序左上右上右下左下

②方向：top-left等四个

③几种情况假设：

<1>border-radius:10px; 四个角的x、y都是10px

<2>border-radius:10px 100px; 表示左上和右下的x、y都是10px，右上和左下的x、y都是100px；

<3>border-radius:10px /100px; 表示四个角的x偏移是10px，y偏移是100px

<4>border-radius:0px 10px 20px 30px /100px 110px 120px 130px; 表示左上x偏移0，y偏移100px，右上10和110，右下20和120，左下30和130（顺时针从左上开始）

<5>border-radius:50%; 四个角的偏移，都是50%的x和y（假设x=y，那么就是一个正圆）

**（40）超出部分的处理**

①**overflow:**

②值：visible 显示（默认）

hidden 隐藏

scroll 滚动（出现2个滚动条，并且无论内容多少都一直有）

auto 自动（比较少没有滚动条，多的话有滚动条）

③单独设置某一个方向的：overflow-x或者是overflow-y某一个方向的x和y轴对超出部分的处理

**（41）盒模型的尺寸**

①**box-sizing:**

②值：content-box（默认，content/文本区）

border-box（指的是border的大小）

③影响：width和height的值影响的是哪个区域：

默认值：影响content区域

border-box：影响border区域

即在border-box的情况下，content区域的大小，需要width和height减去padding四个方向的值后，才能得到。

**（42）盒阴影**

①**box-shadow**

②值：none（默认，外阴影）、inset（内阴影）

③例如：

box-shadow:1px 2px 3px 4px red;

第一个参数指水平偏移（左和右），正值往右，负值往左；

第二个参数指垂直偏移（上和下），正值往下，负值往上；

第三个参数指模糊半径，往里和往外1.5px模糊（以偏移后，再加上第四个参数阴影大小之后的基线为基准线，即这个数据的情况下，x方向阴影最远为1px+4px+3/2px）；

第四个参数指阴影大小，指的是偏移之后，阴影额外扩张的宽度（主要影响的是以偏移后的两个基线，阴影的宽度增加）

第五个参数指阴影的颜色；

④内阴影：

box-shadow:inset 2px 2px 5px 5px red;

盒内阴影，x轴往右偏移2px，y轴往下偏移2px，模糊半径是5px（基于基准线，基准线定义同上），阴影大小为5px，阴影颜色为红色；

⑤多阴影：

box-shadow:5px 2px 10px 10px red,inset 5px 2px 10px 10px blue;

相当于内外都有阴影，具体不细说。

**（43）轮廓线**

①**outline: 颜色 样式 宽度;**

也可以分开设置：**outline-color**等

②说明：绘制于元素周围的一条线，位于边框边缘的外围，可以起到突出元素的作用。但不会占据空间，也不一定是矩形。

③样式：solid实线，dashed虚线，dotted点（方形）

④会压到文字/图像等之上。

**（44）关于兼容性**

可以查阅http://caniuse.com/

**（45）背景颜色**

①**background-color:**

②值：rgb，rgba，英文单词，#加6位/3位表示法

③默认值是：transparent（透明）

**（46）背景图片**

①**background-image:url(地址)**

②值：url(地址)

括号里可以有引号也可以没引号

③可以加多个url属性，中间以逗号分隔，于是两种图片都成为了背景图片。并且前面的，将位于上面，后面的，将位于下面（被覆盖）

④background-color将位于图片之下（最下层）

**（47）背景图片平铺**

①**background-repeat:**

②值：

repeat-x 只x轴方向平铺

repeat-y 只y轴方向平铺

space 平铺，在每张图片之间留出一点空隙（正好能够容下的最大值）

round 平铺，伸缩（正好能够容下，通过缩放的方式来容纳下）

no-repeat 不平铺（只出现一次）

③假设正常能容纳3.5个图片，space容纳3个（留空），round容纳3个或4个（确定是不留空，但具体是缩小还是放大，不知道如何确定）

④background-repeat:no-repeat repeat;

第一个参数是x轴，第二个参数是y轴

⑤background-repeat:no-repeat, repeat;

注意有逗号分隔，第一个针对第一张图片，第二个针对第二张图片（针对有多个背景图片时）

**（48）背景图片的滚动**

①**background-attachment**

②值：

scroll 默认值，当文本有滚动条时，背景图片保持不变（不滚动）

local 当文本滚动时，背景图片跟着滚动

fixed 很少用，参照物是整个窗口。

**（49）背景图片的位置**

①**background-position**

②值：

两个参数：

background-position:10px 20px x轴（往右）偏移10px，y轴（往下）偏移20px

background-position:20% 40% x轴偏移20%，y轴40%（图片的x轴20%，y轴50%的那个点，位于整个容器x轴20%，y轴50%那个点，即两个点重合后，图片的位置），当50% 50%时，图片居中

background-position:center center; 图片居中

top相当于y轴0%，buttom相当于y轴100%

一个参数：

background-position:left x轴靠左，y轴居中，相当于x轴0%，y轴50%（center）

描述的参数符合，另一个参数自动center

四个参数：

background-position:left 10px top 20px; 图片最左边靠容器最左边10px，图片最上边靠容器最上边20px。

和上面的参照点不同

③**只显示图片的某一部分**：

假设图标在相对于图片左上角的x轴往右100px处，y轴0px处开始，那么这么写：

background-position:-100px, 0px;

于是，图片只显示图标部分（即从图片左上角往右100px处开始显示）；

y轴方法同理

**（50）线性渐变**

①**linear-gradient() 这是background的参数**

②括号里的值：

**角度：**（渐变方向，默认从上往下to buttom）to right buttom就是从左上往右下。也可以写为0deg（0度，从下往上），45deg（顺时针转45度，注意，是以x轴和y轴的中间为中心——应该）；

**渐变的颜色：**第一个值默认为在0%的位置，最后一个值默认为100%的位置，中间值默认为平分后的位置，如red, blue, green这样。也可以写为red, blue 20%, green，那么蓝色将位于20%位置，绿色位于100%位置

③例子：

background-image: linear-gradient(red, blue)

图片从上往下，从红色渐变到蓝色

background-image: linear-gradient(to right top, red, blue)

从左下往右上，从红色往蓝色渐变

background-image: linear-gradient(rgba(0, 0, 0, 0) 0%, rgba(0, 0, 0, 0) 50%, rgba(0, 0, 0, 0.5) 100%);

这个是一个50%开始到100%，从透明渐变为灰色

**（51）镜像渐变**

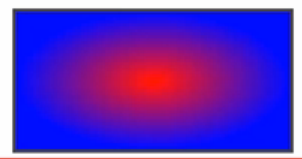
①**radial-gradient**

②例子：

**最近边**

<1>background-image: radial-gradient( closet-side, red, blue);

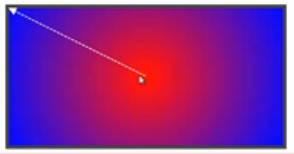
从中心到最近那条边（准确的说是四个边），从红色渐变到蓝色（如果是长方形的话，那么渐变的是一个椭圆渐变）；



**圆形**

<2> background-image: radial-gradient( circle, red, blue);

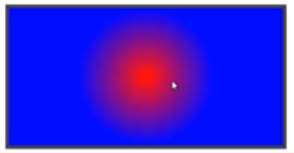
圆形的（默认值是farthest-corner，最远的那个角），从中心往最远的那个角，从红色渐变到蓝色（渐变图形是一个圆形，但会导致某部分没有完全渐变结束）



**圆形带半径**

<3> background-image: radial-gradient(circle 100px, red, blue);

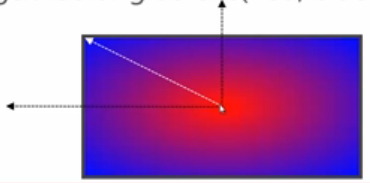
圆形的，渐变距离到100px，从红到蓝



**参数只有两个渐变颜色**

<4> background-image: radial-gradient(red, blue);

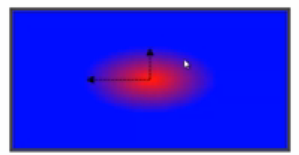
默认的是椭圆，默认的大小是farthest-corner，即从中心往四周是红到蓝（事实上，x轴和y轴都没有渐变结束，因为事实上是一个椭圆，并且最远的角在椭圆上，因此，椭圆的x和y轴都比矩形要大）



**带渐变半径**

<5>background-image: radial-gradient(100px 50px, red, blue);

椭圆，x半径100px，y半径50px，从红色渐变到蓝色



**更改渐变中心点**

<6> background-image: radial-gradient(100px 50px at 0 0, red, blue);

默认中心点为居中，这里的中心点指定为x=0，y=0。



**多色渐变**

<7> background-image: radial-gradient(red, blue 20%, white);

椭圆，从红色开始，20%（针对farthest-corner）渐变到蓝色，最后到白色；

**（52）带repeating的渐变**

①**repeating-\*-gradient**

②渐变都是以gradient结尾的。可以给其加前缀repeat-

③如：

background-image: repeating-linear-gradient(red, blue 20px, red 140px);

方向to buttom，初始红色，20px渐变到蓝色，第140px渐变到红色，然后再过20px（第160px）渐变到蓝色（依次类推）；

background-image: repeating-radial-gradient(red, blue 20px, red 40px);

镜像渐变，从红色，20px到蓝色，40px到红色，依次类推。

**（53）图片渐变范围**

①**background-origin:**

②值：

padding-box 默认渐变范围，即其角和边，指的都是padding-box的

border-box 以border-box为范围

content-box 以content-box为范围

**（54）图片显示范围**

①**background-clip:**

②值：

border-box 默认范围

padding-box 图片范围为padding-box

content-box 只在content范围显示

③联合使用

{ background-origin: content-box;

background-clip: content-box; }

图片显示范围是content-box，并且只在这个范围内渐变（比如平铺等）

**（55）图片大小**

①**background-size:**

②可以是具体大小，

或者是百分比（相对于容器），

auto（自动，使用本来）；

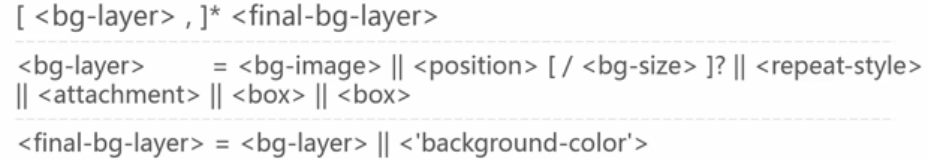
可以是1~2个值，宽度和高度，1个值的话默认是宽度，高度为auto。

cover 撑满容器（实际可能会有一部分在容器之外）

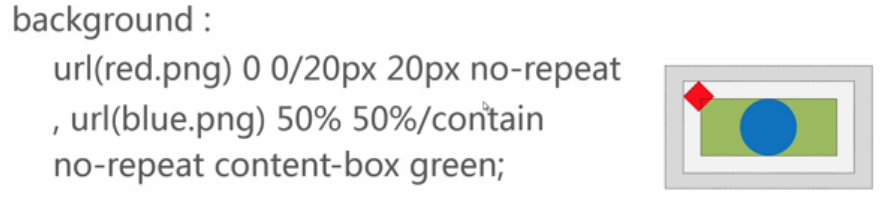
contain 尽可能大，实际可能有一部分容器是没有图片的。

**（56）background缩写**

①好麻烦啊，贴图拉倒



②例子：



**（57）布局**

①元素的摆放模式；

**（58）显示display**

①**display**

设置元素的显示方式；

②值：

**block**  块级元素，默认宽度为父元素的宽度，可设置width和height，换行显示（相对于前序元素、后续元素，也就是说，其他元素，会换行来显示），默认为block的标签有div、p、h1~h6、ul、form

**inline** 行级元素，默认宽度是内容宽度（文字的宽度），不可设置宽高，同行显示（即多个inline的元素，他们将同行显示），元素内部可以换行。默认为inline的标签有span、a、label、cite、em等

**inline-block** 默认宽度是内容宽度，可设置宽高，同行显示，不能在行内换行（只能整块换行），另外假如两边是inline，那么设置inline-block元素的vertical-align:middle;并且设置高度，会导致两边的inline元素垂直居中（但inline-block的文本不会垂直居中）；

**none** 不显示元素，其他元素会占用其位置

|  |  |  |  |
| --- | --- | --- | --- |
| **display值** | **默认宽度** | **可设置宽高** | **起始位置** |
| **block** | 父元素的宽度 | 可 | 换行 |
| **Inline** | 内容宽度 | 不可 | 同行 |
| **Inline-block** | 内容宽度 | 可 | 同行 |

③**display:none**和**visibility:hidden**的区别

前者的话，后续元素会占用该元素的位置；

后者，后续元素不会占用该元素的位置（即原来的元素位置被保留）；

**（59）块级元素水平居中**

①**margin:0 auto;**

②效果：上下不变，水平方向自动居中。

③**实现居中导航**：

首先，设置外层容器水平居中（通过text-align:center;）；

其次，把内部每个元素，设置display:inline-block;（至少是inline但其不能设置宽高）；

最后，此时各个元素被连接在一起了，需要设置各个元素之间的距离（通过margin:0 10px设置间距为20px）；

给个样本：

HTML代码：

<ul class="top">

<li class="mid">11111</li>

<li class="mid">aaaaaaa</li>

<li class="mid">bbbbbb</li>

<li class="mid">cccccc</li>

</ul>

CSS代码：

.top

{

text-align:center;

background-color:#0ec;

}

.mid

{

display:inline-block;

background:#f35;

height:100%;

font-size:30px;

margin:0 10px;

}

④兼容性：

display:inline-block;在ie6、ie7下只有设置在默认显示方式为inline的元素上才会生效，请实现兼容ie6、ie7的通用的方式。

1. div{
2. display:inline-block;
3. \*display:inline;
4. \*zoom:1
5. }

ps：等我用的时候再搞懂算了，这里留个备份。

**（60）定位**

①**position**

②值：

**static：**默认，无定位

**relative**：相对定位，仍在文档流之中（其他的元素不变），参照物相对于原来的位置进行定位。会压在其他的元素上面。作用是为**绝对定位元素**提供**参照物**。

**absolute**：绝对定位，脱离文档流（后续元素会在绝对定位元素的前序元素的后面），默认宽度是内容宽度，参照物是第一个定位的祖先/根元素（根元素是html元素），如果祖先没有定位元素，那么会导致其参照物是html。

（应用场景：比如轮播图，通过相对定位——背景图和绝对定位——标题、轮播按钮结合）

**fixed**：固定定位，脱离文档流，默认宽度是文本宽度，参照物是视窗（我觉得可以认为是浏览器可视范围），假如其bottom:0px，浏览器上下有滚动条——html高度大于浏览器可视范围高度，那么这个元素将一直位于浏览器可视范围的底部（而不是html的最底部））。**（IE6不兼容）**

应用场景：固定顶栏（无论怎么滚动，都在浏览器的上方，同理也有底栏）；

③位置属性（描述所在位置）：

top、right、bottom、left

只有在设置position属性的前提下，才能生效

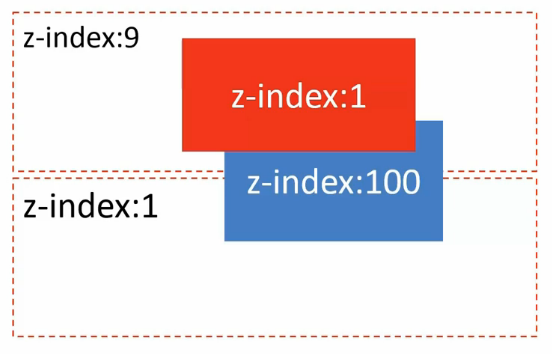
假设4个方向都被设置，会导致元素被撑开。

z-index：越高的值，排序更前（即高的覆盖低的），默认为0。

④**z-index的栈** 概念：

祖先元素的z-index的值更高的，将覆盖祖先元素的z-index的值更低的元素。（需要假设这两个祖先元素是在同一个父元素下）。

如图：



如果z-index一样的话，文档流在更后面的，将覆盖在前面的上面。

**（61）遮罩**

①利用固定定位：position:fixed

②另外需要设置z-index，

一般设置为：

{

position:fixed;

z-index:999;

top:0;

left:0;

height:100%;

width:100%;

background-color:#000;

opacity:0.3;

}

**（62）三行自适应布局**

①指顶栏最上、底部的最下、中间的内容区高度自适应；

②例子：

.top

{

position:absolute;

top:0; /\*上为0\*/

left:0;

height:100px; /\*顶部的高度\*/

width:100%;

background-color:#eee;

}

.body

{

position:absolute;

bottom:100px; /\*和下方距离100px\*/

top:100px; /\*和上方距离100px\*/

left:0; /\*由于设置了四个方向所以撑满高度\*/

right:0; /\*由于设置了四个方向所以撑满宽度\*/

background-color:#ccc;

overflow:auto;

}

.foot

{

position:absolute;

bottom:0; /\*和下方距离0\*/

left:0;

height:100px; /\*顶部的高度\*/

width:100%;

background-color:#aaa;

}

注意：body不能高度和宽度设置为100%（因为会相对于其父元素<body>标签）

③于是可以在中间的class="body"里面设置高度为5000px（那么中间部分会出现滚动条）；

④需要注意的是：

<1>中间部分若过短的话，会导致可能和下方foot留有空白；

<2>若想让下方自适应，那么中间的元素（？或者所有子元素？）设置为绝对定位（脱离文档流）会导致下方直接跟到head后面去。

**（63）对IE6、IE7不兼容display:inline-block的解决办法**

#obj{

display:inline-block;

\*display:inline;

\*zoom:1;

}

**（64）浮动float**

①**float**

②值：

left 向左浮动

right 向右浮动

③特点：

默认宽度——内容宽度（但可以通过width和height设置）；

脱离文档流（非完全脱离）；

向指定方向一直移动（根据值），移动到不能移动为止

④float的元素，在同一个文档流（所以同一个文档流中的float元素，不会重叠）；

float，对元素而言，脱离文档流；对内容，在文档流（不会遮挡内容）；

为了说明上面那点，来一个代码：

<html>

<head>

<meta charset=utf-8"/>

<style>

.a{

width:500px;

height:150px;

background-color:green;

}

#b{

float:left;

width:50px;

height:50px;

background-color:#ccc;

}

#c{

width:50px;

height:100px;

background-color:#aaa;

}

</style>

</head>

<body>

<div class="a">

<div id="b">

ABC

</div>

<div id="c">

FEDCBA

</div>

</div>

</body>

</html>

b的高度是50px；c的高度是100px。在没有float的情况下，b和c的高度正好撑满了外面的div（因为div是块级元素，所以默认情况下他们不会同行显示）；

若完全脱离文档流，那么会导致b和c重叠，即无法看到c（把float改为position:absolute;）或者b（两个都加上position:absolute;）的内容。

然而，使用float的实际情况如图：



A挡住了B的一部分，但是并没有挡住B的内容（注意，是内容FEDCBA，而不是content区域）；

⑤所以可以利用float会使内容不会被遮挡的特性，来达成某些目的。

**潜在问题：**假设float元素在A块中，但由于其脱离文档流的特性，其高度／宽度不会被Ａ块所考虑。假设float元素的高度比A块要大，到了B块的范围内了。

会导致float元素覆盖B块的区域，B块的内容将避开该区域。

如图：



**⑥解决占用后续元素位置空间的办法：**

**clear**

值：

**both、left、right、none（默认）、inherit（继承）**

特点：

应用于float的后续元素＋块级元素；

方法一：

在float元素后，你想要清除浮动的地方，加一个块，然后给其附上clear:both的css属性。这里是一个加空块的示例：

HTML：<div class="clearfloat"></div>

CSS：

.clearfloat{

width:0;

clear:both;

overflow:hidden;

visibility:hidden;

}

举个例子，假设有A块和B块，然后A块有子块a1块和a2块（div）。而a1块是float元素。（注意，除a1块和B块之外，都没有设置高度）

正常情况下：a2块位于a1块右边（行级元素），a1块因为设置css样式时高度比较高，占用了B块的一部分（遮挡了部分B块的区域）；

《1》在a2块前加入这个标签，于是a2块出现在a1块的下方（浮动无效），A块的float元素没有遮挡B块，A块和B块的正常的相对位置。另外，a1块后a2块前，是属于A块的部分。

《2》在A块后加入这个标签，a2块依然位于a1块后面（浮动有效），但是B块不再被遮挡。另外，A块后，B块前有一部分，其属于A、B的父元素的区域。

《3》在a2块中加入这个标签（内容前和内容后两种情况），内容前则a2的内容在a1的下方，内容后则a2的内容在a1的右方。另外，这两种情况，a1块后都是a2块的部分。

**总结：**

也就是说，在 **不设置** 浮动元素的父元素、兄弟元素的 **高度** 情况下，其父元素、兄弟元素的高度，可以根据 **clearfloat所在的标签** 来 **自适应** 。

（注：若设置的话，则不能自适应）

附HTML代码，可以自己试试放在不同的情况。

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

width:500px;

height:200px;

background-color:#ddd;

}

.A{

width:100px;

background-color:blue;

outline:2px dashed blue;

}

.a1{

height:100px;

float:left;

background-color:#ccc;

}

.a2{

background-color:green;

}

.clearfloat {

content:".";

display:block;

height:0;

clear:both;

overflow:hidden;

visibility:hidden;

}

.B{

width:100px;

height:100px;

background-color:Yellow;

}

</style>

</head>

<body>

<div class="main">

<div class="A">

<div class="a1">

aaa

</div>

<div class="a2">

bbbb

<br class="clearfloat"/>

</div>

</div>

<div class="B">

ppp

</div>

</div>

</body>

</html>

方法二：

设置一个类class="clearfloat "，然后利用伪类选择器“:after”，来给该样式的块后面添加一个块级元素。当你想要清除某个块的float浮动时，给其添加一个clearfloat的class即可。

例如CSS样式：

.clearfloat:after{

content:".";

display:block;

height:0;

clear:both;

overflow:hidden;

visibility:hidden;

}

你想要清除A块之后的浮动（达成上面《2》的目的），在A块的标签class属性中，像这样<div class="A **clearfloat**">添加绿色字体部分即可。

如果想要达成上面《1》中的目的，那么在a2的样式中添加clear:both这条CSS属性即可。

注意，不知为何，在float元素中添加.clearfloat，然后利用其after是无效的。可能是因为after依然属于float元素吧。

**总结：**

也就是说，如果通过加样式的方法，若是想从哪个元素之后，float的浮动不生效，那么就在其的class加入上述代码（float元素本身不可加）。

若是从哪个元素开始，float浮动不生效，那么就把伪类选择器的after改为before，并在这个元素的class中加入clearfloat即可。

⑦对于IE不支持伪类选择器after的处理方式：

添加：

.clearfloat{zoom:1;}来处置（**具体不懂，用的时候再说**）

⑧应用场景：

对于三行两列布局（两列浮动），对中间那行的标签，添加clearfloat的样式。

**（65）弹性布局flex**

**flex对IE9及以下不兼容，主要应用于移动web页面中**

①**display: flex;**

②在父元素中设置display: flex;，那么其子元素就可能是弹性元素。

弹性元素有：

《1》子元素中的块级元素、float元素，行级元素；

非弹性元素有：

《1》脱离文档流的元素（position:absolute），孙元素；

③**flex-direction:** 弹性的方向

值：

row 默认，行（从左到右）

row-reverse 反向行（从右到左）

column 列（从上到下）

column-reverse 反向列（从下到上）

④如代码：

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

width:500px;

height:200px;

background-color:#ddd;

display:flex;

flex-direction:row;

}

.a{

background:red;

margin:5px;

text-align:center;

line-height:50px;

}

</style>

</head>

<body>

<div class="main">

<div class="a">

A

</div>

<div class="a">

B

</div>

<div class="a">

C

</div>

</div>

</body>

</html>

解释：

《1》排列方向会依次排列，如果排列方向的宽高（比如方向row则为width）大于总宽高（3个元素，每个width为200px，总width为300px），那么其宽度将自适应（从200px变为100px）

《2》非排列方向会撑满高度。（不设置其宽高的情况下，比如方向row元素的height未设置）

《2》方向有点类似float的顺序，

⑤**flex-flow** 换行设置

值：

nowrap 不换行（默认）

wrap 换行

wrap-reverse 反向换行

解释：

《1》假设排列方向row，总宽度300px，单个元素宽度200px。总高度100px，单个高度未设置。

《2》不换行的情况下，其宽度自适应为100px；

《3》wrap的情况下，宽度为200px（每行只能放一个），高度为自适应（33.3px），顺序从上往下；

《4》wrap-reverse的情况下，宽度为200px，高度自适应（33.3px），顺序是从下往上。

《5》假设单个高度设置为100px，那么只能容纳一个，剩下的将溢出（到父块外面去了）。若设置overflow:hidden，那么超出部分将不会显示。

如代码：

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

width:300px;

height:100px;

background-color:#ddd;

display:flex;

flex-direction:row;

flex-wrap:wrap;

}

.a{

background:red;

margin:5px;

width:200px;

text-align:center;

}

</style>

</head>

<body>

<div class="main">

<div class="a">

A

</div>

<div class="a">

B

</div>

<div class="a">

C

</div>

</div>

</body>

</html>

⑥**flex-flow:** 理解为flex-direction和flex-wrap的缩写

值：

是flex-direction和flex-wrap的值，可以写一个，也可以写两个；

默认值是：

flex-flow:row nowrap;

⑦**order** 顺序

值：number，是一个数字，初始值是0

排列的时候，先排列order值小的，再排列order值大的。

假设一个的order属性的值设置为1，那么值为0的按顺序排列之后，才会排列值为1的。

值可以是负值。

⑧**flex-basis** flex元素的初始宽高（推测是根据方向决定是width还是height）

值：

main-size 主轴方向的宽高（例如row则是元素内容的宽度）

width 也可以设置为像100px这样的。

如果值是0的话，则没有宽高，但文字会溢出显示。

⑨**flex-grow** 弹性空间分配

值：number 默认值为0，可以是1或者更多

效果：将多余空间，按照flex-grow的属性值占全部flex-grow的和的比例，分配给各个元素。

假设总宽度是200px，3个元素的宽度为50px，那么只占用了150px（剩余50px没有被使用）。

若其中一个元素设置flex-grow:1; 那么这个元素的宽度将变为50px（初始）+50px（未被占用的）=100px的宽度。

若其中两个都是值1，那么这2个元素分别拿走25px的宽度；

若一个是值4，一个值是1，那么值4的拿走40px，值1的拿走10px；

如果基数是0，两个值1，一个值2，那么值2的宽度（100px）是值1宽度（50px）的两倍。

⑩**flex-shrink**  对于剩余空间为负的值的处理

值：number 默认为1

假如3个元素每个元素宽度为200px，总宽度为300px，那么因为自适应，则每个元素变为100px（不换行的情况下）。三个元素分别减少了100px（平分超出的300px）

假设其中两个是默认值，另外一个值为2。那么值为2的元素，减少了150px（剩余50px），值为1的两个元素，分别减少75px（剩余125px）

也就是说，值越大的，减少的越多。

【11】**flex:**  缩写

值：flex-grow flex-shrink flex-basis 多、少、基础

默认值：0 1 main-size

可以不全写，但若写一个，则默认是flex-grow的值

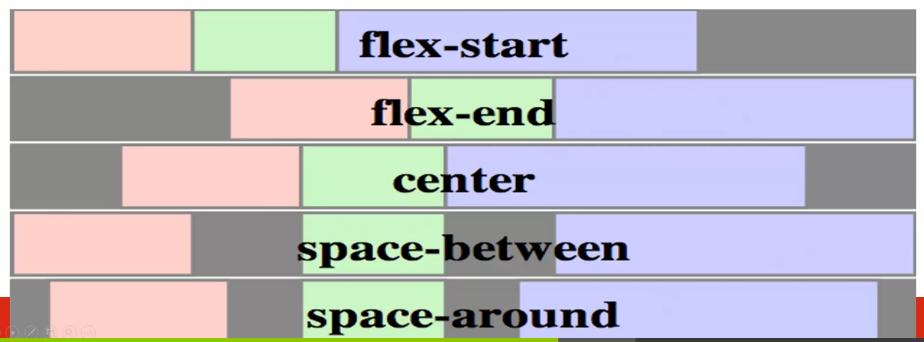
**（66）flex对齐**

**和flex有关，是父元素的属性**

①justify-content 设置主轴方向的对齐方式

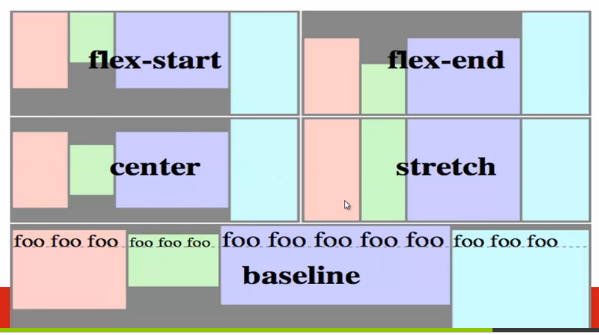
值：

如图：（灰色部分为剩余空间）



②align-items: 设置辅轴方向的对齐方式

值：如图（默认值是flex-start，需要设置子元素高度后才有效）



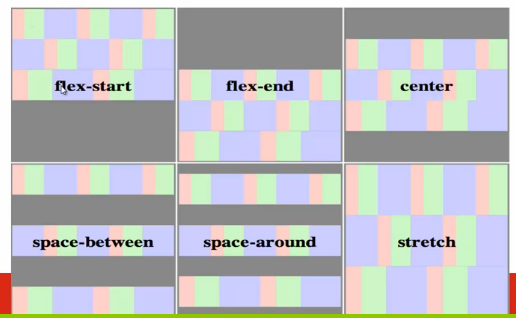
③align-self:

设置单个元素（非父元素属性），在辅轴方向的对齐方式。

值参考align-items

④align-content: 对于多行元素的辅轴方向对齐方式

值如图：



**（67）三行两列自适应布局**

①首先设置三行（中间一行自适应）的布局；

《1》body和html设置为height:100%

《2》body设置display:flex; 然后设置head和foot的高度（由于flex的特性，其宽度将自适应）；

《3》设置中间部分的main的CSS属性为flex:1（这里的1表示对多余空间的分配，全部被中间拿走），于是中间高度撑满；

此时三行布局完成。

②设置两列自适应；

《1》首先确定中间这一行的宽度如何，是定宽，还是为自适应。自适应则不需要设置（因为其父元素默认非主轴宽高为自适应），如果是定宽，则设置其宽度。

《2》决定两列的布局方式，是铺满，或者左、中、右有缝隙；

《3》其次确定左右列的情况，是定宽，还是按比例宽（比如左1右2）；

《4》定宽设置flex-basis属性，按比例宽度则设置flex-grow。

《5》此时，基本两列的情况已定。

③示例代码：

上下两行撑满，中间那行定宽且居中，左右两行根据比例分配宽度，且中间有缝隙（10px）

<html>

<head>

<meta charset=utf-8"/>

<style>

html,body{

height:100%;

}

body{

display:flex;

flex-flow:column;

}

.head,.foot{

background:#ddd;

line-height:100px;

}

.main{

flex:1;

display:flex;

width:1000px;

justify-content:center;

align-self:center;

}

.left{

flex:2;

margin-right:10px;

background:red;

}

.right{

flex:1;

background:blue;

}

</style>

</head>

<body>

<div class="head">

head

</div>

<div class="main">

<div class="left">

left

</div>

<div class="right">

right

</div>

</div>

<div class="foot">

foot

</div>

</body>

</html>

**（68）变形**

①**transform:**

②值：

none（默认，无变形、动画）

transform-function（变形动画）

③**rotate(<angle>)** 旋转，值为角度，正值是顺时针旋转，负值是逆时针方向

例子：transform:rotate(45deg); 旋转45度

④**translate(x,y)**  移动第一个参数是x轴移动，第二个参数是y轴移动（第二个可以不写）

值：

例如100px 移动100px（右或下）；

30% 移动当前物体30%的宽／高（假设元素width:200px，第一个参数100%则相当于移动200px）

例如：

**transform:translate(510px,100%);**

往右偏移510px，往下偏移100%元素的高度的距离

指定方向偏移：

**translateX** 后面加X表示X方向偏移

**translateY** 加Y表示Y方向偏移

⑤**scale(x,y)**  放大、缩小

参数：为x、y轴放大的倍数（可以是小数），若只写一个，表示对x、y轴方向都有效。

《1》这个放大倍数对内容区（比如文字）有效。

《2》负数表示镜像，例如x为-1表示左右方向镜像颠倒，y为-1表示上下方向镜像颠倒。

对单方向起效：

**scaleX** X方向

**scaleY** Y方向

⑥**skew()** 倾斜

参数为x轴和y轴的倾斜角度。

第一个参数是y轴向x轴的倾斜角度（正值逆时针，负值顺时针），

第二个参数是x轴向y轴的倾斜角度（正值顺时针，负值逆时针）；

假如一个值为0，一个值为90deg，此时两周重合（图片消失），90deg变为180deg，则和初始一样（说明两轴没有方向，固定y轴向上x轴向右的部分为正值坐标）；

例如transform:skew(-45deg); 向右倾斜45度

单一轴则为skewX()（y轴旋转）或者是skewY()（x轴旋转）

⑥

《1》假如即需要移动，又需要旋转。

那么应该**先移动**，**后旋转**。

假如先旋转后移动，那么后续移动是根据旋转后的x、y轴坐标而移动的。（可能和预期有偏差）。

《2》假设有放大、缩小，然后移动，那么移动的距离，会受放大缩小的倍数所影响。

⑦设置坐标轴的原点（x=0，y=0的位置）；

**transform-origin:**

值：

默认值：50% 50%（表示中心点）

0 0 左上角（此时旋转是以左上角为原点进行旋转）

20% 相当于20% 50%，第一个值是x轴的，第二个值是y轴的

right 50px 20px 第一个值是x方向（最右），y轴往下移动50（不是从中心点），z轴沿正方向移动了20px。

**z轴方向的移动请看下面。**

**（69）透视**

①**perspective:**

②值：

none 默认

<length> 比如100px，这个值指观察点里物体的距离（值越小，透视效果越明显）

备注：设置于父元素

③**perspective-origin**  透视的角度

值：

50% 50% 默认正中间

left bottom 左下角透视

50% -800px 水平正中间，纵向往上800px的位置（从top开始往上800px）

right 相当于100% 50%

④**translate3d(x, y, z)**  位移（三个方向的）

x、y同普通的

z只能是数值，比如1000px，正值靠近，负值靠远

**translateZ(参数)**  z轴方向的移动

⑤**scale3d(x, y, z)**  放大缩小（三个方向的）

x、y轴同普通的

z轴的变化正常需要配合其他的才能显示，比如：

transform:scale3d(1,1,5)translateZ(150px);

相当于移动了750px

代码：

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

top:300px;

left:500px;

position:absolute;

font-size:100px;

width:500px;

height:100px;

background:#ddd;

text-align:center;

perspective:1000px;

perspective-origin:left;

}

.right{

background:blue;

width:100px;

height:100px;

transform:scale3d(1,1,5)translateZ(150px);

}

</style>

</head>

<body>

<div class="main">

<div class="right">

right

</div>

</div>

</body>

</html>

⑥**rotate3d(x, y, z, 角度)**  以从0,0,0到参数的点为轴，进行3d旋转

transform:rotate3d(1, 0, 0, 45deg); 相当于以x轴为轴进行旋转

transform:rotate3d(0, 0, 1, 45deg); 以z轴进行旋转（顺时针）

具体来说，从参数给的坐标点，看向原点，然后以这个状态进行顺时针旋转。

注意：  
《1》y轴的正数为靠近观测点，负数为远离。

《2》值用1或者100没区别（因为只是表达方向）；

⑦**transform-style** 应用于父元素，设置其子元素是否继承3d

值：

flat 默认（扁平化）

preserve-3d 3d化

⑧**backface-visibility** 背面是否可见（设置于具体某个元素）

值：

visible 默认，背面可见

hidden 背面不可见

这里指的是，假如一个元素有正面有反面（反面通过沿y轴旋转得到），该元素是半透明的，那么默认情况下，是能看到反面元素的。

假设属性设置为hidden，那么背面元素将不可见。

⑧注意，以上在低版本IE中不支持。

⑨代码示例：

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

top:300px;

left:500px;

position:absolute;

font-size:100px;

width:500px;

height:100px;

background:#ddd;

text-align:center;

}

.out{

perspective:1000px;

perspective-origin:50% 50%;

transform-style:preserve-3d;

width:100px;

}

.A{

position:absolute;

background:blue;

width:100px;

height:100px;

transform:rotate3d(0,1,0,0deg);

backface-visibility:hidden;

}

.B{

background:yellow;

width:100px;

height:100px;

transform:rotate3d(0,1,0,180deg);

backface-visibility:hidden;

}

.out:hover{

transform:rotate3d(0,1,0,180deg);

}

</style>

</head>

<body>

<div class="main">

<div class="out">

<div class="A">

A

</div>

<div class="B">

B

</div>

</div>

</div>

</body>

</html>

**（70）动画transition**

**特点：手动触发、A状态—》B状态**

①**transition-property:** 动画起效的属性

值：

none 默认，没有

all 所有的属性

left 只对left有过渡效果

left, color 两个值有效果（可以添加多个）

②**transition-duration:** 动画时间

值：

0s 瞬间（默认）

1s 1秒钟；

500ms 500毫秒

依次类推

③**transition-timing-function** 动画的时间函数（需要预设transition-duration时间）

值：

ease 两头慢，中间快

linear 线性的，速度一致

ease-in 开始慢

ease-out 结束慢

ease-in-out 开始和结束的时候慢（比ease幅度更大）

cubic-bezier(四个参数) 贝塞尔曲线，不懂

step-start

step-end

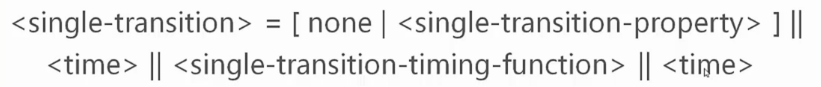
steps(<数字> ,start/end); 比如steps(5,start)就是5步达到目的（参数2表示时间的开始或者结束执行这一步）

④**transition-delay:** 延迟执行动画时间

值：

同transition-duration，决定在多少秒后执行动画时。

⑤transition 以上的综合写法



第一个时间是动画时间，第二个时间（必须有两个时间）是延迟时间。

例如：

transition:left 2s ease 1s, color 2s; 表示属性是左，动画2秒，动画效果是ease，延迟1秒执行。属性颜色，动画2秒立刻执行。

transition:2s 表示2秒动画，所有属性都有效，立刻执行。

**（71）动画animation**

**特点：可自动、可多段、可自定义**

①**animation-duration** 动画时间

0s 不执行动画

1s 动画执行1秒

依次类推，可以是毫秒

②**animation-timing-function** 动画函数

和**transition-timing-function**的值是一样的

③**animation-iteration-count** 动画执行次数

值：

infinite 无限执行

number 具体次数（默认是1）

例子：

animation-iteration-count:1, infinite 第一个动画执行一次，第二个动画无限执行（因为可能同时执行多个动画）

④**animation-direction** 动画执行方向

值：

normal 默认（按定义执行）

reverse 反向执行

alternate 往返动画（先正再反）

alternate-reverse 往返动画（先反后正）

⑤**animation-play-state** 动画状态（正在执行或者停止）

值：

running 执行

paused 停止

比如鼠标移动上去（hover）时停止执行动画；

⑥**animation-delay** 动画延迟执行

值：

0 默认，立刻执行

1s 1s执行

依次类推，可以是毫秒

⑦**animation-fill-mode** 动画开始／结束时是否保持当时的状态

值

none 默认，不保持

backwards 开始时保持动画第一帧状态

forwards 动画结束时保持最后一帧状态

both 都保持

简单来说，backwards，让动画目标在开始前，保持在其开始时的位置（这个不是很明显）。但forwards可以让动画在结束时保持在最后一帧状态（不会返回到动画初始状态）。一般都用both

⑧**animation** 综合使用

值：

none 无动画

aaa 2s ease 0s 1 normal none running这一串表示动画名称是aaa，动画时间是2s，动画形式是ease，0s是delay，1是播放次数，normal表示动画执行方向，none表示动画开始/结束时的状态，running表示动画状态。

bbb 2s 表示动画名称是bbb，动画时间2秒，其他使用默认值

⑨**@keyframes** 关键帧的定义

形式：

@keyframes aaa { from {第一帧状态} to{最后一帧状态} }

或者是（多段的）

@keyframes bbb { 0% {初始状态} 50%{中间状态} 100%{结束状态} }

或者是（多个同样的）

@keyframes ccc { 0%, 75% {初始状态A} 50%{中间状态B} 25%, 100%{结束状态C} } 效果是A-C-B-A-C

⑩定义完后就可以调用了。

比如animation:aaa 1s both; both表示保持开始结束状态

animation:bbb 2s both, ccc 3s; bbb执行2秒（保持）同时ccc执行3秒（不保持）

注意：如果要移动的话，需要加position:absolute;的设置，使其脱离文档流。

附代码：

<html>

<head>

<meta charset=utf-8"/>

<style>

.main{

top:300px;

left:500px;

position:absolute;

width:500px;

height:100px;

background:#555;

text-align:center;

}

@keyframes aaa{

0%{left:0}

50%{left:200px}

100%{left: 400px}

}

@keyframes bbb{

0%{color:red}

50%{color:blue}

100%{color:green}

}

@keyframes ccc{

0%, 75%{width:20px;height:20px;}

50%{width:100px;height:100px;}

25%, 100%{width:200px;height:200px;}

}

.a{

position:absolute;

width:20px;

height:20px;

background:#ddd;

animation:aaa 2s both,bbb 2s both, ccc 2s both;

}

.b{

position:absolute;

width:20px;

height:20px;

background:#aaa;

animation:aaa 2s both,bbb 2s, ccc 2s both 1s alternate infinite;

}

</style>

</head>

<body>

<div class="main">

<div class="out">

<div class="a">

A

</div>

<div class="b">

B

</div>

</div>

</div>

</body>

</html>

（72）transition:width的兼容性测试

div

{

width:100px;

height:100px;

background:blue;

//改变宽度 在2s内改变 线性改变 延迟1s后改变

transition:width 2s linear 1s;

-moz-transition:width 2s linear 1s; /\* Firefox 4 \*/

-webkit-transition:width 2s linear 1s; /\* Safari and Chrome \*/

-o-transition:width 2s linear 1s; /\* Opera \*/

}

div:hover

{

width:300px;

}

div

{

transform: skew(30deg,20deg);

-ms-transform: skew(30deg,20deg); /\* IE 9 \*/

-webkit-transform: skew(30deg,20deg); /\* Safari and Chrome \*/

-o-transform: skew(30deg,20deg); /\* Opera \*/

-moz-transform: skew(30deg,20deg); /\* Firefox \*/

}