

工具使用

vscode快捷键

- cmd+ / 注释
- cmd + shift + f 格式化
- cmd + alt + 方向键 复制一行
- ul.className>li.className*n 快速生成n个无序列表
- .classname 快速生成div类名为classname
- cmd + shift +k 删除当前行
- shift+opt+↓ 快速复制一行
- ul.li*5>a[href=#]>i.iconfont+{标签内容}
- shift+opt+F 格式化

Typora快捷键

- cmd 1-6 标题
- cmd+opt+t 表格

代码块 esc下方按键 ~~~

```
<html>
</html>
```

单行代码强调 esc下方按键 ..

hello world

斜体 一个* 内容一个 *

粗体 两个* 内容两个 *

无序列表 -加空格

- 无序

有序列表：数字小数点空格

1. 哈哈
2. 图片

```+语言 可生成代码块

cmd+/ 可以查看各种格式的快捷键

尽量不要使用图片，因为没有上传服务器

## git常用命令

---

git config --list 设置账户信息

git config --global user.name 'allsun'

git config --global user.email '869472104@qq.com'

git init

工作区、暂存区、版本库

理解为什么要有暂存区：1.可以针对专门任务统一提交，如都是修bug，然后一起提交；2.批量提交想要修改的文件；3.工作区文件变更，想恢复成暂存区的内容，暂存区相当于变成了历史快照，git restore 文件名

git rm --cached 文件名 删除暂存区的更改

git add

git status -s 查看文件状态

git commit -m "变更内容" 本地仓提交

git ls-files 查看被追踪的文件

git reflog --oneline 查看完整历史

git log --oneline 提交历史记录

git reset --hard 版本号 还有soft、mixed模式

git branch 分支名 创建分支

git merge 分支名 先要切换为要合并的分支，一般切换为master

git rebase 分支名 变基合并

### 理解merge和rebase

- merge保留分支合并后创建新的提交，rebase合并成一条主线
- 两种不同合并模式

合并发生冲突时，手动解决冲突，再commit提交后才能继续使用

git branch -d 分支名 合并完就删除

【merge】保留分叉：

A---B---C (main)

\   \

D---E---F (feature)

【rebase】历史平铺：

A---B---C (main)  
    \  
        D'---E' (feature)

`git checkout` 分支名 head指针切换到该分支，head指针影响暂存区和工作区的代码

`git remote add origin https://github.com/AllSun/note.git` 与远程仓库进行关联

`git remote remove origin` 删除远程仓库

`git push --set -upstream origin master` 关联上游分支

`git pull --rebase` 远程分支和本地分支没有关联关系的合并

`git pull` git pull 远程仓库名 远程分支名：本地分支名

`git push`

## 浏览器渲染

输入 URL → DNS解析 → TCP连接 → 发送HTTP请求 → 响应返回HTML  
→ 浏览器解析HTML → 构建DOM树 → 解析CSS → 构建CSSOM树  
→ DOM + CSSOM → 构建渲染树 → 布局 (Layout) → 绘制 (Paint) → 合成 (Composite) → 页面呈现

## HTML+CSS

### 常用html标签

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <!-- 移动设备上以设备宽度显示 -->
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <!-- seo优化 -->
 <!-- 指定网站的关键词 -->
 <meta name="keywords" content="">
 <!-- 指定网站的描述信息 -->
 <meta name="description" content="">
 <link rel="stylesheet" href="./my.css">
 <title>常见标签及解释</title>
</head>
<body>
 <!--h5 当属性和属性值相同时，则可以用一个单词表示-->
 <audio src="#" controls loop autoplay ></audio>
 <div style="color: red; font-size:20px;">这是 div 标签</div>
```

```

<video src="#" controls loop muted autoplay></video>
加粗
<!-- 双标签和单标签 -->
<!-- 单标签 -->
<!-- src中分为相对路径和绝对路径，绝对路径就是写死地址 -->

斜体
<i>斜体</i>
加粗
5²=25

H₂O
<ins>下划线</ins>
删除线
<p>这个标签会换行，当文本内容塞不下会换行</p>
<hr>这是一条横线分割

<blockquote>长引用</blockquote>
<q>短引用</q>

<dl>
 <dt>第一阶段</dt>
 <dd>css</dd>
 <dd>html</dd>
 <dt>第二阶段</dt>
 <dd>javascript</dd>
 <dd>vue</dd>
</dl>
<div>
 回车不换行，换行当做一个空格处理
 连续多个空格，当做一个空格处理
</div>
<div>
 主要用于布局，划分区域
</div>
用于解决局部部件、文本
<!--h1一般网页中只用一次-->
<h1>h1-h6</h1>
<!-- href:
1、可以是网址
2、可以是工程中的其他文件
3、可以是页面中的元素，如跳转到类名为footer的标签
target:
_self 当前窗口打开
_blank 新窗口打开 -->

```

```

超链接

<!-- 转移字符 -->
<abc>
©

<!-- 表格 -->
<table>
 <caption>板块</caption>
 <thead>
 <th>排名</th>
 <th>股票名</th>
 <th>涨幅</th>
 </thead>
 <tbody>
 <tr>
 <!-- td 当中行合并 rowspan=2 列合并colspan -->
 <td>1</td>
 <td>龙头股份</td>
 <td>10%</td>
 </tr>
 </tbody>
 <tfoot></tfoot>
</table>
<!--表单 checked (针对radio、checkbox) ,multiple(针对file)-->
<input type:"file/checkbox/radio/text/password" checked placeholder="" multiple>
<textarea></textarea>
<label for="man">男</label>
<selection>
 <option>北京</option>
 <option>福州</option>
 <option selected>三明</option>
</selection>

<!-- form 表单区域 -->
<!-- action="" 发送数据的地址 -->
<form action="">
 用户名: <input type="text">

 密码: <input type="password">

 <!-- 如果省略 type 属性, 功能是 提交 -->
 <button type="submit">提交</button>
 <button type="reset">重置</button>
 <button type="button">普通按钮</button>
</form>
</body>

```

</html>

## css知识点

### 常见属性

```
height
width
background-color
color
font-family
font-size
font-style normal/italic
line-height
text-indent 2em 缩进
text-align
text-decoration underline/linethrough
background-image: url(./images/1.png);
background-repeat: no-repeat;
background-position: 50px center; 水平 垂直
background-size: cover/contain
background-attachment: fixed;
```

**font:** 是否倾斜 是否加粗 字号/行高 字体（必须按顺序书写）

**属性值:** 背景色 背景图 背景图平铺方式 背景图位置/背景图缩放 背景图固定（空格隔开各个属性值，不区分顺序）

### 布局规则

四种布局规则 BFC/IFC/FFC/GFC

浮动塌陷

子元素浮动脱离文档流，父元素高度无法感知子元素高度，父元素高度为0

三个解决办法

---增加clearfix

```
.parent::after{
 content: "";
 display: block;
 clear: both;
}
```

---变为BFC

```
.parent{
 overflow: hidden;
 /*或者 overflow: auto;*/
 /*或者 display: flow-root*/
}
```

margin布局陷阱

垂直上会合并取最大值，水平上不会

子元素`margin-top`会穿透父元素

`margin:0 auto`，作用在块元素且有宽度，`inline`无效

`flex`布局中，`margin:auto`，会自动变成可以拉伸的弹簧，自动把元素推到两边或中间

`text-align:center`，只针对行内元素和，`inline-block`元素有效

如果父元素没有设置 `position (relative/absolute/fixed/sticky)`，子元素的 `z-index` 是不会生效的。

标签选择器、类选择器、ID选择器、通配符选择器

复合选择器

- 后代 `div span{}`
- 子代 `div > span{}`
- 交集 `p.box{}`
- 并集 `div,p{}`
- 伪类 `a:`
- 超链接伪类 `a:link /visited/hover/active`
- 伪元素 `a::after/before`

**继承性**：文字属性和文本属性，父元素设置后子元素会继承，`<a>` 标签不会继承父元素字体颜色

**层叠性**：后覆盖前相同属性、不同属性叠加

**优先级**

**css样式冲突**，根据权重进行判断，复杂选择器则根据权重相加比较总和，权重相同，后者生效

`!important >` 行内样式 (1000) `>` id选择器 (100) `>` 类选择器 (10) `>` 标签选择器 (1) `>` 通配符选择器 (0)

```
p {
 color: red !important;
}
```

**标签的表现形式**

1. 块元素

- 独占一行
- 可设置宽高
- 未设置宽度时，宽度默认父元素100%

2. 行内元素 `span label a`

- 可同行展示
- 不可设置宽高
- 宽度由内容撑开
- 上下padding、上下margin不起作用

### 3. 行内块元素 input img button

- 同行展示
- 可设置宽高
- 宽度由内容撑开

转换显示模式：display:block/inline-block/inline

```
<style>
 a{
 /*转换为块元素; inline/inline-block*/
 display:block;
 width:400px;
 height:400px;
 color:red;
 background-color:yellowgreen;
 text-align:center;
 line-height:400px;
 }

 *{
 /*清楚标签默认样式*/
 padding:0;
 margin:0;
 }

 ul{
 /*清楚项目符号*/
 list-style:none;
 list-style-type:none;
 }

 /*转换为行内块的问题
 1, 行内块之间的空隙, 是由结构里的换行符导致
 解决方案:
 (1) 父元素设置
 font-size:0;
 (2) 去掉换行符, div全部放一行, 或者使用注释进行占位
 (3) 设置负边距
 margin:-40px;
 */

</style>
```



# 图文对齐

主要解决行内块与文字对齐方式，也可以用数值

`vertical-align:middle/bottom/baseline;`

行内块元素、行内元素水平居中，针对父元素设置 `text-align:center;`

块元素居中，直接对块元素设置 `margin:0 auto;`

## 常见css属性

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>css常见样式</title>
 <link rel="stylesheet" href="h.css">
 <style>
 /* 类选择器 类名设置可重复，可同时作用，类名最好有实际意义，最常用*/
 .desc {
 width: 200px;
 height: 300px;
 background-color: yellow;
 /* 字体颜色 */
 color: pink;
 font-size: 20px;
 text-align: center;
 /* 行高和高度一致时，文字可以居中 */
 line-height: 300px;
 }
 .detail {
 background-color: antiquewhite;
 }

 /* 伪类选择器 */
 .detail:hover{
 /* 父元素的50% */
 width: 50%;
 /* 4倍字体大小，em当前字体大小 */
 height: 4em;
 background-color: rgb(75, 29, 29);
 }

 /* 标签选择器 作用域太广*/
 div {
 width: 100px;
 height: 100px;
```

```

 background-color: brown;
 }
 /* id选择器 缺点id唯一，不符合html文档规范，不常用*/
 #title {
 width: 100px;
 height: 100px;
 background-color: blue;
 }

 /* 字体属性 ， 了解 无衬线体、衬线体*/
 .font {
 width: 200px;
 height: 200px;
 font-size: 40px;
 /* 粗细 */
 font-weight: bolder;
 font-style: italic;
 /* 可指定多个，没有的话向下溯源 */
 font-family: 'Courier New', Courier, monospace;
 /* font: 40px italic bolder "楷体"; */
 /* 文字不换行 */
 white-space: nowrap;
 /* 指定标签里面的内容超出样式的宽度和高度之后如何处理 */
 overflow: auto/scroll/hidden;
 }

 .text {
 width: 800px;
 height: 200px;
 /* line-height: 200px; */
 font-size: 20px;
 text-align: center;
 /* 首行缩进 */
 text-indent: 2em;
 /* 下划线 */
 text-decoration: underline;
 /* 字符间间距 */
 letter-spacing: 1px;
 /* 单词间间距 */
 word-spacing: 1px;
 /* 文字超出部分隐藏 */
 overflow: hidden;
 /* 溢出部分显示省略号 */
 text-overflow: ellipsis;
 }

 /* 后代选择器 , 有空格 缩小范围 */
 .nav li {
 width: 100px;
 }

```

```
height: 100px;
background-color: aqua;
}

/* 并集选择器 ,逗号隔开*/
.out,
p,.inner{
 background-color: yellowgreen;
}

/* 交集选择器 */
.inner.second{
 background-color: pink;
}

/* 通配符选择器 ,一般用于初始化界面*/
* {
 background-color: #f5f5f5;
}
</style>
</head>
<body>
 <div class="desc">描述信息</div>
 <div class="desc">描述信息</div>
 <div class="desc detail">描述信息</div>
 <div>我是测试组</div>
 <div id="title">id选择器</div>
 <hr>
 <div class="font">接天莲叶无穷碧，映日荷花别样红</div>
 <hr>
 <div class="text">每天花半小时学习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学
习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学
习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学
习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学
习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学习，作一个行业精英每天花半小时学
习，作一个行业精英每天花半小时学习，作一个行业精英</div>
 <hr>
 <ul class="nav">
 <li class="nav-item">首页
 <li class="nav-item">关于我们
 <li class="nav-item">加入我们

 <hr>
 <div class="out">
 <p>每天学一点</p>
 <div class="inner second">真的就学一点</div>
 </div>
</body>
```

```
</html>
```

## 盒子模型

margin 外边距 设置大小，垂直方向会重叠取最大值，水平方向相加

border边框 设置大小、属性，一定要有厚度、样式、颜色

padding 内边距 设置大小

content内容

赋值： 上右下左 、 上/左右/下 、 上下/左右 、 全部

盒子大小= content+padding+border

行内元素上下padding、上下margin不起作用

背景色会填充至padding区域,也会蔓延至边框，不过一般会被边框样式覆盖

```
<style>
 div{
 padding-top:20px;
 padding-right:20px;
 border:5px solid dashed pink;
 }

</style>
```

**margin嵌套崩塌**：子元素设置的margin-top作用到了父元素身上

1. 给父元素设置 `overflow:hidden;`
2. 父元素添加一个极小的padding-top;

**脑子里要有底线、基线、中线、顶线轮廓图**

利用border属性 `border-bottom` /`border-left/right` 画出三角形

利用 `border-radius: 50px/50%;` 画圆形

**怪异盒子模型（IE盒子模型）**

box-sizing

盒子大小不会受padding和border影响，只会挤压content

## 背景属性

`background-color`

`background-image:url();`

平铺 `background-repeat: no-repeat;`

`background-position:4px 4px;` 背景图片位置距离盒子左边和顶部的像素

`background-attachment:scroll/fixed` 背景图片随着盒子滚动/背景图片固定

`background-size:10px 10px/10px auto/cover/contain` 背景图片大小宽高, `cover`填满盒子图片不完整, `contain`图片完整, 但盒子没填满

**雪碧图**: 把所有图片都放一张图片上, 减少网络请求次数, 使用 `background-position` 来确认

## 透明属性

`opacity:0-1;` 作用所有元素及内容

`rgba(255,12,11,0.5)` 仅作用颜色

## 文档流

从左到右, 从上往下

`display :none` 从结构中消失, 不占据位置, 下方元素上移

`display:block` 显示

`opacity:0` 消失, 依旧占据位置, 可触发事件 (常用)

`visibility:hidden/visible` 消失, 但占据位置, 可触发事件

`position: absolute; top: -9999px;` 将元素移出视口, 仍占空间

## 浮动

1. 浮动: 脱离文档流, 实现页面布局, 专门用于块元素同行展示的问题, 空间不够换行, 左浮动不会改变布局顺序, 右浮动会

`float:left/right/none`

2. 元素浮动后, 会脱离文档流, 不占据位置, 下方元素上移, 文字不会和浮动元素产生交集, 如产生交集, 会环绕在浮动元素周围
3. 浮动元素会脱离文档流, 因此无法撑起父元素的高度, 当父元素没有指定高度时, 会塌陷, 对布局造成影响
  - 严格指定父元素高度
  - 给父元素添加 `overflow:hidden;`, 经常使用
  - 在容器所有浮动元素的末尾, 追加一个空的元素块, 对其设置 `clear:both;`
4. 块元素浮动之后, 不设置宽度, 由内容撑开, 一般需指定宽度; 行内元素浮动之后, 可设置宽高, 上下 `margin` 也生效
5. 浮动元素同行展示, 摆不下时自动换行
6. 使用浮动时, 在一个容器里, 尽量都使用浮动

# 定位

定位主要用于解决文档流、浮动无法解决的页面布局问题

正常使用顺序：文档流 > 浮动 > 定位

定位：把元素定到指定位置

`position: relative/absolute/fixed` 三者区别在于参照物不同

`left/right/top/bottom: 10px;` 释义：元素的左边距离参照物的左边距离10个像素

1. `relative` 参照物为自身原本的位置，没有脱离文档流，占据原来位置，一般辅助定位，不常用，一般使用绝对定位。开启定位之后，元素层级得到提升，高于原文档流元素，如果方位未进行设置，则元素在原位
2. `fixed` 参照物为浏览器视口，脱离文档流，层级提高，高于文档流元素，网页滚动，不会随着滚动，一般应用“联系我们”
3. `absolute` 要先确定参照物，脱离文档流，层级提高，高于文档流元素
  - 参照物为距离自己最近的具备定位属性的祖先元素或html文档，从自身出发，一层层向上溯源，找不到，以html文档为参照
  - 一般以父元素作为参照物，如果父元素没有定位属性，则添加 `relative`，因为 `relative` 不会影响任何东西
4. `sticky` 滚动到一定位置时，不会再随着网页滚动
5. `relative` 和 `fixed` 块元素开启之后，不写宽度时，宽度由内容撑开；行内元素开启后，可设置宽高

`z-index: 999` 层级属性，如果父元素也有设置，则无法突破父元素的限制，如果没有，则和外界元素比较；定位元素可以使用此元素，数值大的在最上面展示

## Flex布局

基本概念：水平轴、交叉轴

水平方向

`justify-content: center/flex-start/flex-end/space-between/space-around/space-between` (元素之间相等距离) / `space-around` (元素之间与容器之间相等距离) / `space-around` (元素与容器周围之间保持相等距离)

垂直方向

`align-items: center/flex-start/flex-end/baseline` (容器基线位置) / `stretch` (拉伸填满)

注意点：1. 当改变列方向时，`justify-content` 控制垂直方向，`align-content` (`align-items`) 控制水平方向

2. 当改变行列方向后：`flex-start/flex-end` 也调转了

决定行之间的距离

`align-content: flex-start/flex-end/center/space-between/space-around/space-around` / `stretch`

`flex-direction: row/row-reverse/column/column-reverse`

`flex-wrap: nowrap` (挤一行) / `wrap` (多行) / `wrap-reverse`

`flex-flow: row/column` (2选1) `wrap/wrap-reverse` (2选1)

`order`: 值为±整数，向前后移动

`align-self`: 作用单个元素，同`align-items`的值

## Grid布局

```
display:grid;
grid-template-columns: 1fr 1fr 1fr;
grid-template-rows
gap
grid-column
grid-row
grid-template-areas /*类似画个草图*/
grid-area /*草图的子元素*/
justify-items /*控制子元素在容器内水平方向的对齐方式*/
align-items
justify-content /*控制容器在浏览器内水平方向的对齐方式*/
align-content
```

## 移动端布局

第一种

@media 媒体查询

Flexbox 和 Grid 布局

相对单位如 vw, vh, em, rem, %

第二种 视口适配

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

第三种 响应式图片

```
<picture>
 <source media="(max-width: 600px)" srcset="image-small.jpg">
 <source media="(min-width: 601px)" srcset="image-large.jpg">

</picture>
```

第四种flex布局

第五种百分比布局、单位适配

## H5部分

---

1. 语义化标签：根据网页布局建立不同的标签，利于SEO优化

例如：<header>、<nav>、<footer>

2. 多媒体标签

```
<body>
<audio src="" autoplay controls muted></audio>
/*多浏览器支持*/
<audio>
 <source src="" />
</audio>
</body>
```

3. input标签type类型添加

number、email、number、color、date、week

属性

required(必填)/novalidate(关闭表单验证)

4. 新增元素

progress(进度条)/datalist

5. 引入外部字体

```
<style>
@font-face{
 font-family:"bf";
 src: url(ziti.ttf);
}
p {
 font-family:"bf";
}
</style>
```

6. 字体图标技术，矢量图保证图标清晰

1. 引入外部字体图标文件
2. 使用<i>/<span>等标签通过类名引入字体图标文件，引用图标
3. 通过类名选择器选择，通过font-size、color等属性设置图标大小

## CSS3部分

---

1. 取消了私有化前缀，如属性设置不用在强调-webkit
2. 新增选择器



- 相邻兄弟选择器: `.classname1+classname2{}` 跟在class1后面的所有class2会被设置样式
- 子元素选择器: `.box>div.inner{}` 类名为box底下div类名为inner的样式设置
- 属性选择器: `p[title]{}/p[class^='a']` (以a开头) 直接定位
- 伪类选择器

**超链接伪类:** `link` 未被访问、`visited` 访问过的状态、`active` 点击瞬间的状态、`hover` 悬浮的状态

`a:active {}`

**表单伪类:** `focus`、`checked`

**目标伪类:** 跳转到改页面的样式, `target`

**结构伪类:** `item:nth-child(3/3n)/last-child/first-child`

**伪元素:** 是行内元素, 给选中的元素添加样式, 如第一个字幕颜色; 元素中的文本被选中后样式, `after/before/selection`

- 文字阴影

`text-shadow:` 水平偏移、垂直偏移、模糊度、阴影颜色;

- 盒子阴影

`box-shadow:` x y 模糊半径 扩展半径 颜色

- 元素过渡

`transition:0.4s;`

`transform:rotate(45deg);` 控制元素的移动、旋转、缩放、倾斜

`transform:translate(-50%,50%);` 垂直水平居中

- 动画效果多看一些demo学习, 要用的时候再看

## html+css工程demo笔记

工程结构

----CSS

reset.css

iconfont.css/wtff

工程主项.css

---image

index.html

`left:50%`用于水平居中, 然后元素自身再左移动自身一般的宽度, 实现居中

三角形制作, 让content宽高为0, 边框要有像素, 取消一边的像素, 然后设置为对称两边的为透明transparent

水平居中, 移动端用弹性盒子模型, pc端可是用margin-top平分上下空间

`outline:none`

`.search~.search-box` 波浪号选择器 兄弟选择器, 选择后面所有叫search-box的元素

wrap作为公共样式，可作为全局定义，后续子界面可直接继承

浮动影响，父元素高度未知，子元素溢出，使用伪对象 ::after 清除影响

```
.play{} 绘制圆角技巧
```

```
.clear::after{content: "";display: block;clear: both;} 清除浮动影响，作为公共样式
```

```
position:relative;top:10px; 自己小范围移动
```

```
cursor:pointer; 鼠标变小手
```

## 移动web

### 平移

`transform:translate(x轴移动距离, y轴移动距离)` 一个值的时候，默认X轴

```
transform:translateX()
```

```
transform:translateY()
```

```
transform:translateZ()
```

```
transform:translate3d()
```

### 定位居中

第一种,需要自己计算盒子的大小的一半

```
position:absolute;
left:50%;
right:50%;
margin-left:-100px;
margin-top:-200px;
height:400px;
width:200px;
```

第二种，自动计算

```
position:absolute;
left:50%;
top:50%;
transform:translate(-50%,-50%) //百分比都是基于盒子尺寸
```

### 旋转

```
transform:rotate
```

### 转换原点

`transform-origin`:水平原点位置、垂直原点位置 (left、top、right、bottom、center)

## 多重转换

`transform:translate() rotate()`

## 缩放

`transform:scale()`

## 倾斜

`transform:skew(90deg)`

## 渐变

渐变是多个颜色逐渐变化的效果，一般用于设置盒子背景

分类：

- 线性渐变



- 径向渐变



## 线性渐变

```
background-image: linear-gradient(
 渐变方向,
 颜色1 终点位置,
 颜色2 终点位置,

);
```

# 径向渐变

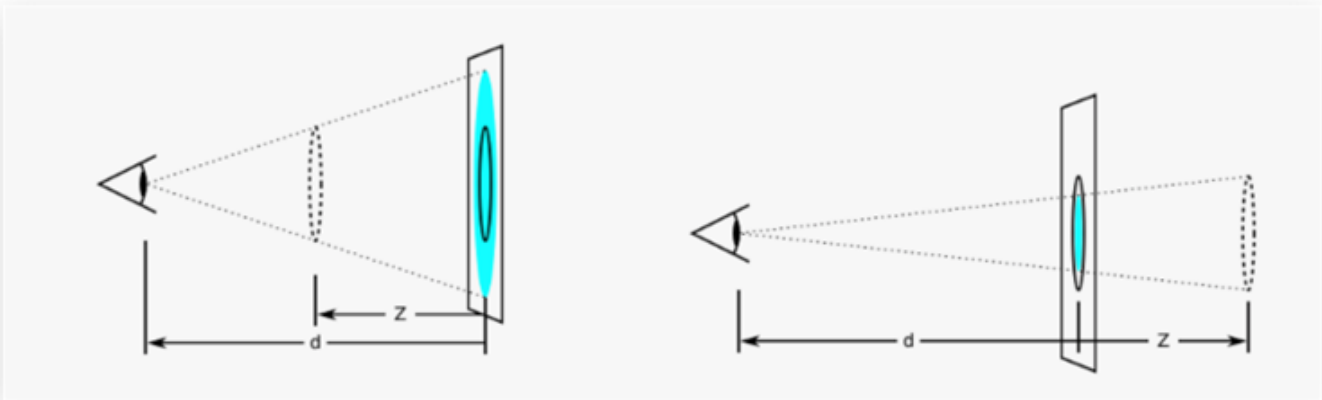
```
background-image: radial-gradient(
 半径 at 圆心位置,
 颜色1 终点位置,
 颜色2 终点位置,

);
```

# 视距

作用：指定了观察者与 Z=0 平面的距离，为元素添加透视效果

```
perspective:800-1200
```



# 动画

定义

```
/* 方式一 */
@keyframes 动画名称 {
 from {}
 to {}
}

/* 方式二 */
@keyframes 动画名称 {
 0% {}
 10% {}

 100% {}
}
```

使用

`animation`: 动画名称 动画花费时长;

## 适配相关

二倍图、屏幕分辨率、逻辑分辨率、（等比适配rem、vw）、flex布局仅适配宽度

视口=网页显示区域

## 媒体查询

条件成立，执行相关CSS

```
@media (width:320px) {
 html {
 background-color: green;
 }
}
```

## 媒体查询-完整写法

**@media** 关键词 媒体类型 **and**（媒体特性）{ CSS代码 }

### 关键词 / 逻辑操作符

- and
- only
- not

### 媒体类型

媒体类型用来区分设备类型

- screen: 屏幕设备
- 打印预览: print
- 阅读器: speech
- 不区分类型: all

### 媒体特性

- 视口宽高: width / height
- 视口最大宽高: max-width ; max-height
- 视口最小宽高: min-width; min-height
- 屏幕方向: orientation
  - portrait: 竖屏

- landscape: 横屏

## 媒体查询-外部CSS

```
<link rel="stylesheet" media="逻辑符 媒体类型 and (媒体特性)" href="xx.css">
```

## rem

网页分成10等份

动态计算rem

```
(function () {
 const setRem = () => {
 const html = document.documentElement;
 const width = html.clientWidth;
 html.style.fontSize = (width / 10) + 'px'; // 1rem = 视口宽度 / 10
 };

 setRem();
 window.addEventListener('resize', setRem);
})();
```

使用

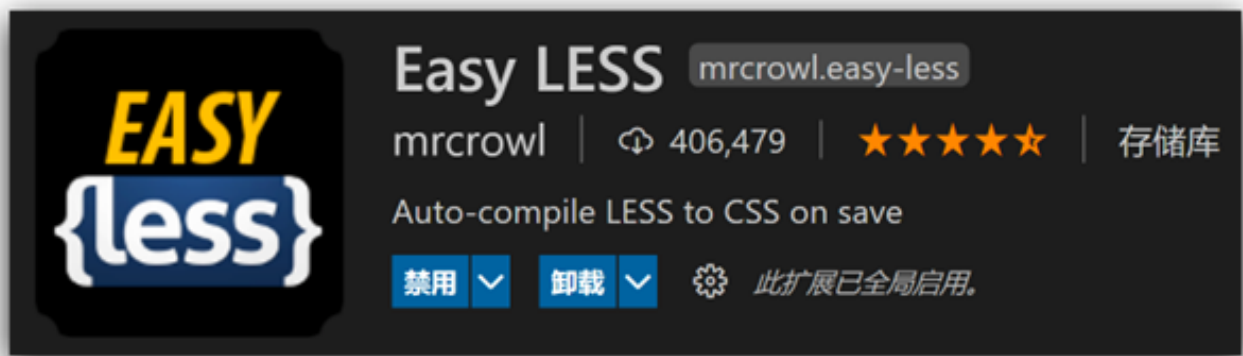
```
.box {
 width: 3rem; /* 3/10 屏宽 */
 height: 2rem; /* 2/10 屏高 */
 font-size: 0.5rem; /* 根据 html 设置动态缩放 */
}
```

## less

Less是一个CSS预处理器, Less文件后缀是.less。扩充了 CSS 语言, 使 CSS 具备一定的逻辑性、计算能力

注意: 浏览器不识别 Less 代码, 目前阶段, 网页要引入对应的 CSS 文件

VS Code 插件: Easy LESS, 保存 less文件后自动生成对应的 CSS 文件



## 注释

- 单行注释
  - 语法：// 注释内容
  - 快捷键：ctrl + /
- 块注释
  - 语法：/\* 注释内容 \*/
  - 快捷键：Shift + Alt + A

## 运算

- 加、减、乘直接书写计算表达式
- 除法需要添加 小括号 或 .
- 表达式存在多个单位以第一个单位为准

```
.box {
 width: 100 + 50px;
 height: 5 * 32px;

 width: (100 / 4px);
 height: 100 ./ 4px;
}
```

## 嵌套



作用：快速生成后代选择器

```
.父级选择器 {
 // 父级样式
 .子级选择器 {
 // 子级样式
 }
}
```

```
.father {
 color: red;
 .son {
 width: 100px;
 }
}
```

提示：用 & 表示当前选择器，不会生成后代选择器，通常配合伪类或伪元素使用

```
.father {
 color:  red;
 &:hover {
 color:  green;
 }
}
```

```
.father {
 color:  red;
}
.father:hover {
 color:  green;
}
```

## 变量

概念：容器，存储数据

作用：存储数据，方便使用和修改

语法：

- 定义变量：@变量名: 数据;
- 使用变量：CSS属性: @变量名;

```
// 定义变量
@myColor: pink;
// 使用变量
.box {
 color: @myColor;
}
a {
 color: @myColor;
}
```



## 导入

作用：导入 less 公共样式文件

语法：导入: @import “文件路径”;

提示：如果是 less 文件可以省略后缀

```
@import './base.less';
@import './common';
```

## 导出

写法：在 less 文件的第一行添加 // out: 存储URL

提示：文件夹名称后面添加 /

```
// out: ./index.css
// out: ./css/
```

## 禁止导出

写法：在 less 文件第一行添加: // out: false

```
1 // out:false
```

## vw和vh基本使用

vw和vh是相对单位，相对视口尺寸计算结果，经常2个搭配一起用

- vw: viewport width (1vw = 1/100视口宽度)
- vh: viewport height (1vh = 1/100视口高度)