

1. 什么是HTML、CSS?

- 是做网站的编程语言。
- 浏览器把代码解析后的样子就是我们看到的网站。可以通过鼠标右键选择查看网页源代码。
- 如何去写代码？写到哪里呢？
 - 一个网站是由N个网页组成的。 每一个网页 .HTML 文件

2. VSCode编辑器

- VS code 下载地址: <https://code.visualstudio.com/>
- 如何安装插件？ 语言包、 open in browser 、 view in browser 、 live server、 px to rem
- 学习编辑器基本使用
 - 设置： 文件-> 首选项-> 设置 (大小、 是否换行 word wrap)

快捷键	功能
ctrl + s	保存
ctrl + a	全选
ctrl + x 、 ctrl + c 、 ctrl + v	剪切、复制、黏贴
ctrl + z 、 ctrl + y	撤销、前进
shift + end	从头选中一行
shift + home	从尾部选中一行
shift + alt + ↓	快速复制一行
alt + ↓ 或 ↑	快速移动一行
tab	向后缩进
tab + shift	向前缩进
alt + 鼠标左键	多光标
ctrl + d	选择相同元素的下一个

3. Chrome浏览器？（谷歌浏览器）

- 下载地址：自行搜索，官网下载
- Chrome占浏览器市场份额：66.88%

4. 深入了解网站开发？

- UI设计师：设计稿

- web前端开发工程师 (H5 开发)
 - 设计稿 -> 代码
 - 数据库里的数据 -> 显示到页面
 - HTML + CSS
 - HTML : 结构, CS : 样式, JS : 行为
- web后端开发工程师

5. web三大核心技术?

HTML + CSS + JavaScript

6. HTML基本结构和属性?

- HTML : 超文本 标记 语言
 - 超文本 : 文本内容 + 非文本内容 (图片、视频、音频等)
 - 标记 : < 单词 >
 - 语言 : 编程语言
- 标记也叫做标签:
 - 单标签 <header>
 - 双标签 <header></header>
- 标签是可以上下排列, 也可以组合嵌套。
- HTML 常见标签属性: <http://www.html5star.com/manual/html5label-meaning/>
- 标签的属性: 来修饰标签的, 设置当前标签的一些功能。


```

```

7. HTML初始代码?

- 每个 .html 文件都有的代码叫做初始代码, 要符合 html 文件的规范写法。
 - ! + tab 键 : 快速的创建 html 的初始代码

```
<!DOCTYPE html> <!-- 文档声明: 告诉浏览器这是一个html文件 -->
<html lang="en"> <!-- lang="en"表示是一个英文网站 lang="zh-cn"表示是一个中文网站 -->
<head>
  <meta charset="UTF-8"> <!-- 元信息: 是编写网页中的一些赋值信息 charset="UTF-8"国际编码 -->
  <title>标题写这里</title> <!-- 设置网页的标题 -->
</head>
<body>
  显示网页内容的区域
</body>
</html>
```

- VsCode 更改每种语言的快速创建方式: 设置-用户代码片段-选择语言

8. HTML注释

写法: `<!-- 注释的内容 -->` 在浏览器中看不到, 只能在代码编辑时看到注释的内容

- 意义:
 1. 把暂时不用的代码注释起来, 方便以后使用
 2. 对开发人员进行提示
- 快速添加与删除注释:
 1. `ctrl + /`
 2. `shift + alt + a`

9. 标题与段落

- 标题: 双标签 `<h1></h1>` ... `<h6></h6>`
 - 在一个网页中, `h1` 标题最重要, 并且一个 `.html` 文件中只能出现一次 `h1` 标签
 - `h5` `h6` 标签在网页中不经常使用。
- 段落: 双标签 `<p></p>`

10. 文本修饰标签

- ``、``
 - 区别:
 1. 写法和展示效果是有区别的, 一个加粗、一个斜体
 2. `strong` 的强调性更强, `em` 的强调性稍弱
 - 下标: `` 文本_{sub}
 - 上标: `` 文本^{sup}
 - 删除文本: `` ~~del~~
 - 插入文本: `<ins></ins>` ins
- 注: 一般情况下, 删除文本都是和插入文本配合使用的。

11. 图片标签

- `img`: 单标签
 - `src`: 引入图片地址
 - `alt`: 当图片出现问题的时候, 可以显示一段友好的提示文字
 - `title`: 提示信息 (所有标签都具有 `title` 属性)
 - `width`、`height`: 图片的大小

12. 路径的引入

- 相对路径 (`.` `..`) : `../` 返回上一级目录, 可以通过多次 `../` 来返回想要的目录
- 绝对路径: 网络路径、本地其他文件

13. 链接标签

- `a`: `<a>`

- `href` 属性：链接的地址
- `target` 属性：可以改变链接打开方式
 - `_self`：默认，在当前页面打开
 - `_blank`：新窗口打开
- `base`：单标签，作用就是改变链接的默认行为。添加在 `<head>`，`<base target="_blank">`

14. 跳转锚点

- 跳转锚点是在本页面进行的操作(也可以跳转到其他页面)，可以跳转到本页面指定位置。

1. `#` 号 + `id` 属性：`a` 跳转到其他标签，给 `a` 链接 `href="#跳转id"`

```
<a href="#tiaozhuan"></a>
<p id="tiaozhuan"></p>
```

2. `#` 号 + `name` 属性：`a` 跳转 `a` 标签，注意 `name` 属性加给的是 `a` 标签

```
<a href="#aa"></a>
<a name="aa"></a>
```

- 邮箱链接：` `
- 下载链接：` `

15. 特殊字符

1. `&` + 字符
2. 解决冲突 左右尖括号、添加多个空格的实现
 - `<`；（左尖括号、小于号）、`>`；（右尖括号、大于号）、` `；（空格符）

16. 列表标签（列表标签夹层不能含有其他标签）

1. 无序列表：`ul` `li` 符合嵌套的规范
2. 有序列表：`ol` `li` 一般用的比较少，可以用无序列表来实现
 - `type` 属性可以更改列表样式，一般不用
 - 有序列表扩展：`start` 让列表从多少开始计数，`reversed` 表示倒序显示，`type` 定义前缀样式

```
<ol start="50" reversed type="v">
  <li>咖啡</li>
  <li>牛奶</li>
  <li value="1">茶</li>
</ol>
```

- `li` 的 `value` 属性：定义单个列表的开始序号（强制更改顺序）

3. 定义列表：`dl` `dt` `dd` 列表项需要添加标题和对标题进行描述的内容

- 注：列表之间可以互相嵌套，形成多层级的列表。

17. 表格标签

- `table`：表格的最外层容器
- `tr`：定义表格行
- `th`：定义表头
- `td`：定义表格单元
- `caption`：定义表格标题（显示在表格上方）
 - 注：之间是有嵌套关系的，要符合嵌套规范。
- 语义化标签：`tHead`、`tBody`、`tFoot`
 - `tBody` 可以出现多次，但是 `tHead`、`tFoot` 只能出现一次。
- 表格属性

属性名	格式	说明
border	<code>border="1"</code>	表格边框
align	<code>align="center"</code>	表格左右对齐的方式
valign	<code>valign="middle"</code>	写在 <code>tr</code> 或 <code>td</code> 单元格内容上下对齐方式
cellpadding	<code>cellpadding="10"</code>	单元格内的空间，类似 <code>padding</code>
cellspacing	<code>cellspacing="10"</code>	单元格之间的空间，类似 <code>margin</code>
rowspan	<code>rowspan="2"</code>	合并行，添加给 <code>td</code>
colspan	<code>colspan="2"</code>	合并列，添加给 <code>td</code>

18. 表单标签

- `form`：最外层容器
- `input`（单标签）标签有一个 `type` 属性，决定是什么控件。
 - `type` 属性：

属性名	描述
<code>text</code>	普通的文本输入框
<code>password</code>	密码输入框
<code>checkbox</code>	复选框
<code>radio</code>	单选框
<code>file</code>	上传文件
<code>submit</code>	提交按钮

属性名	描述
reset	重置按钮
hidden	隐藏的 input，用户看不见，用于存储信息

- checkbox 和 radio 通过 name 绑定分组
- placeholder 会在首次加载时在表单内显示想要的文本
- checked 首次加载时选中，disabled 不可选中状态
- textarea :多行文本框
 - rows : 行数、cols : 列数
- select、option : 下拉菜单
 - selected : 默认选中 disabled : 不可选中 multiple : 可多选
- label : 标注，不会呈现任何效果。含有 for 属性，常绑定在单选框
 - for : for 属性可把 label 绑定到另外一个元素标签的 id 属性。
- 关于表单被选中时会有边框的解决方法

```
input{ outline:none; }
input:focus{ outline:none; }
```

19. div和span

- div : 做一个区域划分的块
- span : 对文字进行修饰的，内联
- 注：默认是没有样式的，要用 css 添加样式

20. CSS基础语法

- 选择器{ 属性1 : 值; 属性2 : 值2 }
 - width : 宽
 - height : 高
 - background-color : 背景颜色 (css 中的背景颜色, html 中有 bgcolor)
- 长度单位 :
 - px : 像素
 - % : 百分比
- css 注释 : /* css注释的内容 */
 - 快捷键 ; shift+alt+a、ctrl+ /

21. CSS样式的引入方式

1. 内联样式

```
<p style="内联"></p>
```

2. 内部样式

```
<head>
  <style> <!-- 内部样式 --> </style>
</head>
```

3. 外部样式

- 引入一个单独的 CSS 文件, `name.css`
- 通过 `link` 标签引入外部资源, `rel` 属性制定资源跟页面的关系, `href` 属性指资源的地址

```
<link rel="stylesheet" href="地址">
```

- `@import` 方式也可以引入 (注: 这种方式是有很多问题的, 不建议使用)

22. CSS中的颜色表示法

1. 单词表示法: `red blue green yellow ...`
 2. 16进制表示法: `#000000`(最小值) `#ffffff`(最大值)
 3. `rgb` 三原色表示法: `rgb(255,255,255)`
 - 取值范围 `0~255`
- 网页提取颜色的谷歌插件下载地址: <https://www.baidufe.com/fehelper>

1. CSS背景样式

- `background-color` : 背景颜色
- `background-image` : 背景图片
 - `background-image:url(背景地址)` , 默认: 会水平垂直都铺满
- `background-repeat` : 背景图片的平铺方式
 - `repeat-x` : x 轴平铺
 - `repeat-y` : y 轴平铺
 - `repeat` : x , y 都进行平铺, 默认值
 - `no-repeat` : 都不平铺
- `background-position` : 背景图片的位置
 - `background-position: x,y`
 - `number (px 、 %)`
 - 单词, x : `left center right` , y : `top center bottom`
- `background-attachment` : 背景图随滚动条的移动方式
 - `scroll` : 默认值, 背景位置是按照当前元素进行偏移
 - `fixed` : 背景位置是按照浏览器进行偏移

2. CSS边框样式

- `border-style` : 边框样式
 - `solid` : 实线
 - `dashed` : 虚线
 - `dotted` : 点线
- `border-width` : 边框大小(边框宽度)
- `border-color` : 边框颜色
 - 格式: `red #f0000 rgb(255,255,255)`
 - 透明颜色: `transparent`
- 边框扩展:
 - `border: 1px solid black;`
 - `border-width: 1px 0;` 这样也可以, 上下有边框, 左右没有。
 - `border-width: 1px(上) 0px(左/右) 1px(下);`
- 边框也可以针对某一条边进行单独设置: `border-left-color`、`border-top-color ...`
- 透明颜色: `transparent`
- 扩展: `outline` (轮廓) 是绘制于元素周围的一条线, 位于边框边缘的外围, 可起到突出元素的作用。
 - 用法和 `border` 相同, 但不可以独立设置一条边。

3. CSS文字样式

- `font-family` : 字体类型
 - 英文、中文，每个中文字体都有英文昵称，宋体：simsun
 - 字体分为衬线体和非衬线体
 - 衬线体棱角多，如宋体；非衬线体比较圆润，如微软雅黑。
 - 添加多字体：

```
font-family:"Source Sans Pro", Helvetica, Arial, sans-serif;
```

 - 字体有空格使用 `" "` 包裹，多字体使用 `,` 隔开
- `font-size` : 字体大小
 - 默认大小：`16px`
 - 注：字体大小一般为偶数
- `font-weight` : 字体粗细
 - 单词：`normal`、`bold` (加粗)
 - 数值：100、200...900，100到500都是正常的，600-900都是加粗
- `font-style` : 字体样式
 - 单词：`normal`、`italic` (斜体)
 - `oblique` 也是表示斜体，用的比较少，一般了解即可。
 - `italic` 带有倾斜属性的字体才可以设置倾斜操作。
 - `oblique` 没有倾斜属性的字体也可以设置倾斜操作。
- `color` : 字体颜色

4. CSS段落样式

- `text-decoration` : 文本修饰
 - `text-decoration: underline dashed #cccccc;`
 - 下划线：`underline`，删除线：`line-through`，上划线：`overline`，不添加任何装饰：`none`
 - 添加多个文本修饰用空格隔开：`text-decoration:line-through overline underline`
- `text-transform` : 文本大小写（针对英文段落）
 - 小写：`lowercase`，大写：`uppercase`，只针对首字母：`capitalize`
- `text-indent` : 文本缩进(首行缩进)
 - `em` 单位：相对单位，`1em` 永远都是跟字体大小相同
- `text-align` : 文本对齐方式
 - 左对齐：`left`，居中：`center`，右对齐：`right`，两端点对齐：`justify`（两端对齐，中间间距自动调整）

- `line-height` : 定义行高
 - 什么是行高, (文字上边距、文字下边距、文字大小是行高的组成部分) 一行文字的高度, 上边距和下边距的等价关系。
 - 默认行高不是固定值, 而是根据当前字体大小进行变化 (可以理解为: 默认文字上下边距不变)
 - 取值: `number (px) | scale` (比例值, 跟文字大小成比例)
- `letter-spacing` : 字之间的间距
 - `spacing` 是外部距离的意思, `letter` 是文字的意思, 同样还有单元格外边距 `cellspacing` `cell` 是单人牢房的意思
- `word-spacing` : 单词之间的间距 (针对英文段落)
- 英文和数字不自动折行的问题 (连贯性的英文和数字在行尾是不会自动折行的)
 1. `word-break : break-all;` : 非常强烈的折行)
 2. `word-wrap : break-word;` : 不是那么强烈的折行, 会产生一些空白区域
 - `white-space : nowrap;` : 不让内容折行

5. CSS复合样式

- 复合样式的写法是通过空格的方式实现的。复合写法有的是不需要关心顺序的, 例如 `background`、`border`, 有的是需要关心顺序的, 例如 `font`
- 1. `background : red url() repeat 0 0;`
- 2. `border : 1px dotted #颜色`
- 3. `font` :
 - 注: `font` 最少要有两个值 (先写) `size` (后写) `family`
 - `font: weight style size family` ✓
 - `font: style weight size family` ✓
 - `weight style size/line-height family` ✓
 - `font:italic bold 30px/40px 宋体;`
 - `size` 和 `family` 要写在最后, 而且 `size` 要在前
- 尽量不要混写, 如果非要混写, 那么一定要先写复合样式, 再写单一样式, 这样样式才不会被覆盖掉。

6. CSS标签选择器

1. ID 选择器

```
#id{}
```

1. ID 是唯一值, 在一个页面中只能出现一次, 出现多次是不符合规范的。

2. 命名的规范，由字母、下划线、中划线、数字（并且第一个不能是数字）

3. 驼峰写法： `searchButton` (小驼峰) `Searchbutton` (大驼峰)

- 短线写法： `search-small-button`

- 下划线写法： `search_small_button`

2. class 选择器

```
.class{}
```

1. class 选择器可以复用。

2. (标签) 可以添加多个 class 样式。

3. 多个样式的时候，样式的优先级根据 CSS 决定，而不是 class 属性中的顺序。(具有覆盖性)

4. 标签+类的写法。

- 只套用 div 标签中 `class="box"` `div.box{}`

```
<div class="box"> <p class="box">
```

3. 标签选择器(TAG 选择器)

```
div{}
```

- 使用的场景：

- 1. 去掉某些标签的默认样式

- 2. 复杂的选择器当中，如层次选择器

4. 群组选择器(分组选择器)

- 可以通过逗号的方式，给多个不同的选择器添加统一的CSS样式，来达到代码的复用。

```
p,#text,.title{background: red;}
```

5. 通配选择器

```
*{ } -> div,ul,li,p,h1,h2.....{}
```

- 使用的场景：去掉所有标签的默认样式时

6. 层次选择器

名称	代码	描述
后代	<code>M N { }</code>	<code>ul li{ }</code> 、 <code>#list ul li{ }</code> 在 ul 里面去寻找 li
父子	<code>M > N { }</code>	M 的孩子N会继承 M 的样式，但 N 的孩子不会继承样式
兄弟	<code>M ~ N { }</code>	当前 M 下方 的所有 N 标签 <code>div~h2{ }</code> 会找到 div 后面的所有 h2 标签添加样式)
相邻	<code>M + N { }</code>	当前 M 下面相邻的 N 标签 <code>div+p{ }</code> 如果 P 标签在 div 正下方，则会被选中，否则不会被选中

7. 属性选择器

写法	代码	描述
M[属性]{ }	div[class]{ }	所有带有 class 属性的 div 标签
M[属性][属性2]{ }	div[class][id]{ }	同时带有 class 和 id 属性的 div 标签
M[属性=值]{ }	div[class=search]{ }	= ： 完全匹配，名称完全相同
M[属性*=值]{ }	div[class*=search]{ }	*= ： 部分匹配，部分名称相同
M[属性^=值]{ }	div[class^=search]{ }	^= ： 起始匹配，起始名称相同
M[属性\$=值]{ }	div[class\$=search]{ }	\$= ： 结束匹配，结束名称相同

8. 伪类选择器

◦ 常用伪类

伪类	描述
:link	访问前的样式，只能添加给 a 链接
:visited	访问后的样式，只能添加给 a 链接
:hover	鼠标移入时的样式，可以添加给所有标签
:active	鼠标按下时的样式，可以添加给所有标签

- 一般网站都只设置 a{} (link visited active) 和 a:hover
- 其他伪类
 - :after 、 :before ： 通过伪类的方式给元素添加一段文本内容，使用 content CSS 属性。
 - :after 和 :before 属于伪元素，在 h5 中要加两个 : ，为了兼容低版本浏览器可以加一个冒号。
 - ::selection ： 选中文本的样式 伪元素
 - :checked 、 :disabled 、 :focus 都是针对表单元素的。
 - :checked 是（单选、多选都可以）选中后的样式
 - :disabled 是不可选中表单样式，要给表单预设不可选中状态
 - :focus 是获取光标后样式，文本框常见
 - 结构性伪类选择器
 - :nth-of-type(1) :nth-child(n)
 - 角标是从 1 开始的， n 值 表示从 1 到无穷大
 - :nth-of-type() ： 根据括号里面的值选

格式	说明
<code>:nth-of-type(2)</code>	选取第 2 个标签
<code>:nth-of-type(+n+2)</code>	选取大于等于 2 的标签
<code>:nth-of-type(-n+2)</code>	选取小于等于 2 的标签
<code>:nth-of-type(2n)</code>	选取偶数标签, <code>2n</code> 也可以是 <code>even</code>
<code>:nth-of-type(2n-1)</code>	选取奇数标签, <code>2n-1</code> 可以是 <code>odd</code>
<code>:nth-of-type(3n+1)</code>	自定义选取标签, <code>3n+1</code> 表示“隔二取一”

- `:nth-child()` 和 `:nth-of-type()` 差不多, 有一点区别
 - 区别: `type` 只识别要套用的标签, `child` 不识别
 - `type`: 类型, `child`: 孩子
- `:first-of-type`: 第一个套用
- `:last-of-type`: 最后一个套用
- `:nth-last-of-type(2)`: 倒数第二个开始套用
- `:only-of-type`: 唯一一个套用 (其父容器里面只有唯一的标签值)

7. CSS样式继承

- 文字相关的样式可以被继承 (颜色、字体大小、行高、缩进等)
- 布局相关的样式不能被继承 (默认是不能继承的, 但是可以设置继承属性 `inherit` 值)

8. CSS样式优先级

1. 相同样式优先级

- 当设置相同样式时, **后面的优先级较高**, 但不建议出现重复设置样式的情况。

2. 内部样式与外部样式

- **内部样式与外部样式优先级相同**, 如果设置了相同样式, 那么后写的引入方式优先级高。

3. 单一样式优先级

`style`行内 > `id` > `class` > `tag`(标签) > `*` > 继承

4. `!important`

- 提升样式优先级, 非规范方式, 不建议使用。(不能针对继承的属性进行优先级提升)

5. 标签+类与单类

- 标签+类 (`div.box`) > 单类 (`.box`)

6. 群组优先级

- **群组选择器与单一选择器的优先级相同**, 靠后写的优先级越高。

7. 层次优先级

1. 权重比较

`ul li .box p input{}` = `1 + 1 + 10 + 1 + 1`

```
.hello span #elem = 10 + 1 + 100
```

2. 约分比较

```
ul li .box p input{} = li p input{}
```

```
.hello span #elem = #elem
```

9. CSS盒子模型

- 组成: content -> padding -> border -> margin
 - content : 内容区域, 由 width 和 height 组成
 - padding : 内边距 (内填充)
 - 只写一个值: 30px (上下左右)
 - 只写两个值: 30px 40px (上下、左右)
 - 写四个值: 30px 40px 50px 60px (上、右、下、左)
 - margin 同上
 - border : 外部边框
 - margin : 外边距 (外填充)
 - padding、border、margin 都可以单独设置
 - margin-left、margin-right、margin-top、margin-bottom
 - 背景颜色会填充到 border 以内的区域。(content、padding)
 - 文字会在 content 区域。
 - padding 不能出现负值, margin 是可以出现负值的。
 - box-sizing : 盒尺寸, 可以改变盒子模型的展示形态
 - content-box : content, 宽、高只分给 content, 默认值
 - border-box : content padding border (宽、高分给三个)
 - 使用的场景:
 1. 不用再去计算一些值
 2. 解决一些百分比的问题
- 盒子模型的一些问题
 1. margin 叠加问题, 出现在两个上下容器 margin 同时存在的时候。会取上下中值较大的作为叠加的值。
 - 解决方案: 1. BFC 规范; 2. 想办法只给一个元素添加间距
 2. margin 传递问题, 出现在嵌套结构中, 只针对 margin-top
 - 解决方案: 1. BFC 规范 2. 给父容器加边框 3. 把 margin 换成 padding 加给父容器
 - 浮动的元素不存在 margin 传递问题(BFC)
 - 扩展: margin 左右自适应是可以的, 但是上下自适应是不行的。
 - 当给元素脱离文档流后(float, absolute, fixed), margin: 0 auto; margin居中就不起作用了

10. 标签分类

- 按类型
 - `block`：块 `div`、`p`、`ul`、`li`、`h1...h6` ...
 - 独占一行。
 - 支持所有样式。
 - 不写宽的时候，跟父元素的宽相同。
 - 所占区域是一个矩形
 - `inline`：内联，`span`、`a`、`em`、`strong`、`img` ...
 - 挨在一起的
 - 有些样式不支持，例如：`width`、`height`
 - `margin` **左右支持**，相邻的内联元素会隔开
 - `padding` **上下左右都支持**，只是视觉上撑开了元素，和其他元素排列时，会和其他元素区域重叠
 - 宽由内容决定。
 - 所占的区域不一定是矩形。
 - 内联标签之间会有空隙，原因：代码中换行产生的
 - 两个 `span` 写在一行就不会有空隙、或者给父容器设置 `font-size : 0;` 后，给子容器单独设置字体大小
 - `inline-block`：内联块 `input`、`select` ...
 - 挨在一起，但是支持宽高。
 - 布局一般用块标签，修饰文本一般用内联标签**
- 按内容划分标签

名称	描述
flow	流内容
Metadata	元数据
Sectioning	分区
Heading	标题
Phrasing	措辞
Embedded	嵌入式
Interactive	互动的

- 具体访问<http://www.w3.org/tr/html5/dom.html> 在50%位置查看
- 按显示划分
 - 替换元素：浏览器根据元素的标签和属性，来决定元素的具体显示内容。

- `img`、`input` ...
- 非替换元素：将内容直接告诉浏览器，将其显示出来
 - `div`、`h1`、`p` ...

11. 显示框类型

- `display` : 可以改变标签的类型
 - `display: block`、`display: inline`、`display: inline-block`、`display: none`
 - `display: none` : 不占空间的隐藏
 - `visibility: hidden` : 占空间的隐藏
- 扩展：
 - 内联元素居中：给父容器添加 `text-align: center;`
 - 块元素居中： `margin: 0 auto;`
 - 内联块居中：给父容器添加 `text-align: center;`

12. 标签嵌套规范

- 块标签可以嵌套内联标签

```
<div>
  <span></span>
  <a href="#"></a>
</div>
```

- 块嵌套块

```
<div>
  <div></div>
</div>
```

- 特殊：

```
<!-- 错误的写法 -->
<p>
  <div></div>
</p>
```

- 内联不能嵌套块

```
<!-- 错误的写法 -->
<span>
  <div></div>
</span>
```

- 特殊：


```
<!-- 正确的写法: -->
<a href="#">
  <div></div>
</a>
```

13. 溢出隐藏

- `overflow`
 - `visible` : 默认
 - `hidden` : 边框边界直接截断
 - `scroll` : 边框右侧和下侧会出现滚动条, 内容少的时候也会显示滚动条。
 - `auto` : 溢出时右侧会出现滚动条, 内容少的时候不显示滚动条。
- 针对两个轴分别设置: `overflow-x`、`overflow-y`

14. 透明度与手势

- `opacity` : 0 (透明, 占空间) ~ 1 (不透明)
 - IE浏览器: `filter:alpha(opacity=50);`
 - 注: 占空间、所有的内容也会透明
- `rgba()` : 0 ~ 1
 - `background:rgba(255,0,0,0.5)`
 - 可以让指定的样式透明, 而不影响其他样式
- `cursor` : 手势
 - `default` : 箭头
 - `pointer` : 手型
 - 要实现自定义手势:
 1. 准备图片: `.cur`、`.ico`
 2. `cursor : url(./img/cursor.ico),auto;`

15. 最大、最小宽高

- `min-width`、`max-width`
- `min-height`、`max-height`
 - `min-height:200px;` : 最小为 200px, 如果内容溢出, 则会根据内容变大
 - `max-height:200px;` : 最大为 200px, 如果内容较少, 则会根据内容变小
- % 单位: 换算, 以所在文档流父容器的大小进行换算的
- 一个容器怎么适应屏幕的高: 容器加 `height:100%`; `body:100%`; `html:100%`;

```
html,body{height:100%}
.Container{ height:100%}
```

16. CSS默认样式

- 没有默认样式的: `div`、`span`

- 有默认样式的:

- `body : margin:8px;`
- `h1 : margin : 21.440px 0px; font-weight : bold;`
- `p : margin : 16px 0;`
- `ul : margin : 16px 0; padding-left : 40px; list-style : disc;`
- `a : text-decoration: underline;`

- `css reset` (样式重置):

- `*{ margin:0;padding:0;}`
 - 优点: 不用考虑哪些标签有默认的 `margin` 和 `padding`
 - 缺点: 稍微的影响性能
 - `body,p,h1,ul{ margin:0;padding:0;}`
- `ul{ list-style:none;}`
- `a{ text-decoration:none;color:#999;}`
- `a:hover{ color:red;}`
- `img{ display:block;}`
 - 问题的现象: 图片跟容器底部有一些空隙。
 - 内联元素的对齐方式是按照文字基线对齐的, 而不是文字底线对齐的。
 - `vertical-align : baseline;` 基线对齐方式, 默认值
 - `img{ vertical-align : bottom;}` 底线对齐, 解决方式是推荐的

- web前端助手下载: <https://www.baidufe.com/fehelper>

1. photoshop使用

- 组成：菜单项、工具栏、辅助面板
- 快捷键：
 - `ctrl + r`：显示隐藏标尺，在标尺上可以拖拽参考线，可以通过移动工具拖拽回去，也可以在视图菜单中选择清除所有参考线。
 - `alt + 滚轮`：放大缩小
- 图片格式：`jpg`、`png`、`gif`
- `png` 等图片的切图流程
 1. 通过矩形选框工具，选择指定的区域。
 - 微调：`alt` 减少区域，`shift` 增加区域，上下左右键微调
 - 利用参考线记录量取的位置，以便以后测量其他值
 2. `ctrl + c` 复制框选中区域
 2. `ctrl + n` 新建一个psd文件
 3. `ctrl + v` 粘贴复制文件
 4. 存储为 `web` 格式
- 企业切图流程
 - 利用工具快速获取样式
 - 蓝湖：<https://lanhuapp.com>

2. float浮动

- 脱离文档流，沿着父容器靠左或靠右进行排列
 - `float: left`、`right`、`none`
 - **加浮动的元素，会脱离文档流，会沿着父容器靠左或靠右排列**，如果之前已经有浮动的元素，会挨着浮动的元素进行排列
- `float` 注意点
 - **脱离文档流，沿着父容器靠左或靠右进行排列**，三种脱离文档流方法分别是 `float`、`absolute`、`fixed`
 - 只会影响后面的元素的排版。
 - 内容默认提升半层，`float` 下面可以拥有其他元素，但是会将文字挤出去。
 - **默认宽根据内容决定，可以让内联元素支持宽高。**
 - 换行排列。(如果都设置了浮动，第一行最后一个浮动的高是第二行的分界线)
 - 主要给块级元素添加，但也可以给内联元素添加。
- 如何清除浮动
 - 上下排列：`clear` 属性，表示清除浮动的，`left`、`right`、`both`
 - 嵌套排列：
 - 固定宽高：不推荐。不能把高度固定死，不适合做自适应的效果
 - 父元素浮动：不推荐，因为父容器浮动也会影响到后面的元素
 - `overflow: hidden` (BFC 规范)，如果有子元素溢出，则会被截断
 - `display: inline-block` (BFC 规范)

- 设置空标签：不推荐,会多添加一个标签。
- `::after` 伪元素：推荐，是空标签的加强版，目前各大公司的做法。
 - `clear` 属性只会操作块标签，对内联不起作用

3. CSS 定位 (注：定位的高度坍塌只能给父容器加宽度)

- `position` :
 - `static` (默认), `relative`, `absolute`, `fixed`, `sticky`
- `relative` : 相对定位
 - 如果没有定位偏移的量，对元素本身没有任何影响
 - **不使元素脱离文档流**
 - 不影响其他元素布局
 - `left`、`top`、`right`、`bottom` 是**相对于当前元素自身进行偏移**
- `absolute` : 绝对定位
 - 使**元素完全脱离文档流**
 - 使**内联元素支持宽高** (让内联具备块特性)
 - 使**块元素默认宽根据内容决定** (让块元素具备内联的特性)
 - 如果**有定位祖先元素相对于定位祖先元素发生偏移**，没有定位祖先元素相对于整个文档发生偏移 (绝对、相对、固定)
- `fixed` : 固定定位
 - **使元素完全脱离文档流**
 - 使**内联元素支持宽高** (让内联具备块特性)
 - 使**块元素默认宽根据内容决定** (让块元素具备内联的特性)
 - **相对于整个浏览器窗口进行偏移，不受浏览器滚动条的影响** (绝对定位受浏览器影响)
- `sticky` : 粘性定位
 - 在指定的位置，进行粘性操作。
- 定位扩展：定位元素支持 `margin` 和 `padding`，也可以设置 % 宽高 (**这里的百分比对象是 `position` 定位的对象**)
- `float` 和 `position`
 - 当 `float` 和 `position` 混用时，只能指定 `static` 和 `relative`，如果指定了 `absolute` 或者 `fixed`，`float` 的设置将无效。
- `z-index` : 定位层级，默认层级为 0。
 - 嵌套时候的层级问题：同级的层级先比较，如果同级没设置定位，则会找子元素比较
 - 关于覆盖的问题：可以给覆盖的元素设置 `relative`，层级设置的比覆盖本元素的高。
 - `z-index` 仅能在定位元素上奏效

- 扩展：定位居中（有固定宽度）

```
#content {  
  position: absolute;  
  left: 0;  
  right: 0;  
  margin-left: auto;  
  margin-right: auto;  
  width: 100px;  
}
```

```
<body>  
  <div>  
    <div id="content">  
      居中蓄力中  
    </div>  
  </div>  
</body>
```

- 不知道宽度：

```
<body>  
  <div style="position: absolute; left: 50%;">  
    <div style="position: relative; left: -50%; border: dotted red 1px;">  
      没有宽度，照样居中  
    </div>  
  </div>  
</body>
```

- 扩展：内联 和 内联块 居中
 - 给父容器加 `text-align: center;`

4. CSS添加省略号

- `width`：必须有一个固定的宽
- `white-space : nowrap`：不让内容折行
- `overflow : hidden`：隐藏溢出的内容
- `text-overflow : ellipsis` 添加省略号

- 扩展： `white-space: nowrap;` 可以让内联/内联块不换行排列

5. CSS圆角

- `border-radius : 值;`：圆角设置是四个圆在边框与容器相切，圆的半径由值决定
 - 值决定了圆的半径
 - `px`：正常值
 - 写一个值：四个角
 - 写两个值：左上，右下、右上，左下（对角线）

- 写四个值：左上、右上、右下、左下
- px：20px / 40px;
 - 代表与容器相切的圆，x 轴长度为 20px，y 轴长度为 40px。
- %：50% 的时候会切成一个圆，如果容器宽高不相等，则会为椭圆
- 扩展：如何做一个半圆

```
#box1{  
  width: 200px;  
  height: 100px;  
  background: red;  
  border-radius: 200px 200px 0 0;  
}
```

6. PC端的布局

- 通栏：自适应浏览器的宽度
- 版心：固定一个宽度，并且让容器居中

1. HTML与XHTML的区别

- DOCTYPE 文档及编码
- 元素大小写 (XHTML 不允许元素标签大写)
- 属性布尔值 (XHTML 属性的值必须要完整)
- 属性引号 (XHTML 属性的值必须要加引号)
- 图片的alt属性 (XHTML img 中必须加alt属性)
- 单标签的写法 (XHTML 单标签后必须要加/
)
- 双标签闭合

2. strong和b标签、em和i标签

- 表现形态都是 文本加粗 和 文本斜体，区别在于， strong 和 em 是具备语义化的,而b和i不具备语义化
- 注意： strong 、 b 、 em 、 i 、 span 都属于内联

3. 引用标签

标签类型	描述	元素类型
blockquote	引用大段段落解释	块元素，带有上下左右外边距
q	引用小段的短语解释	内联元素，会自动添加双引号
abbr	缩写或首字母缩略词	内联元素，默认带下划点线，删除 title 属性后，点线消失
address	引用文档地址信息	块元素，默认斜体 font-style: italic;
cite	引用著作的标题	内联元素，默认斜体 font-style: italic;

4. iframe嵌套页面 标签

- iframe 元素会创建包含另外一个文档的内联框架（即行内框架）
 - 可以引用其他 html 到当前 html 中显示。
- 主要是利用 iframe 属性进行样式的调节。

属性	描述
frameborder	规定是否显示框架周围边框， 0 表示不显示， 1 表示显示
width	定义 iframe 的宽度
height	定义 iframe 的高度
scrolling	规定是否在 iframe 中显示滚动条， scrolling="no" 表示不显示滚动条

属性	描述
src	规定在 <code>iframe</code> 中引入 URL
srcdoc	规定在 <code>iframe</code> 中显示的页面内容

- 应用场景：
 - 数据传输
 - 共享代码：如：W3school网站的演示示例，改变 `iframe` 的 `src` 来进行区域内容切换
 - 局部刷新
 - 第三方介入

5. br与wbr标签

- `br` 表示换行操作，而 `wbr` 表示软换行操作。 `
` `<wbr/>`
- `wbr` 软换行，添加在较长单词中间，告诉浏览器换行时机
 - 如果单词太长，或者您担心浏览器会在错误的位置换行，那么可以使用 `wbr` 元素来添加 Word Break Opportunity (单词换行时机)

6. pre元与 code

- 针对网页中的程序代码的显示效果。(显示的都是等宽字体)
 - `pre` 不支持回车，`code` 支持回车。

7. map 与 area

- 给特殊图形添加链接，`area` 能添加热区的形状：矩形、圆形、多边形。

```
<img src="" alt="" usemap="#star">
<map name="star">
  <area shape="" coords="" href="" alt="">
</map>
```

- `shape` 指定形状：`rect` 矩形、`circle` 圆、`poly` 多边形
- `coords` 制定坐标：
 - 矩形：`coords="x1 y1 x2 y2"` 矩形左上、右下角坐标
 - 圆形：`coords="x1 y1 r"` 圆心坐标与半径
 - 多边形：`coords="x1 y1 x2 y2 x3 y3..."` 每个拐点的坐标，首尾坐标会相连
- `href` 跳转 URL 地址

8. embed 与 object

- `embed` 和 `object` 都能够嵌入一些多媒体，如flash动画、插件等。基本使用没有太多区别，主要是为了兼容不同的浏览器。
 - `object` 要配合 `param` 使用

- 关于 object 扩展

```
<object data="" type="">
  <!-- 设置文件路径, name值要写对 -->
  <param name="movie" value=".Flash/flash5377.swf">

  <!-- 设置透明度, name值要写对 -->
  <param name="wmode" value="transparent">
</object>
```

9. audio 与 video

- audio 控制音频, video 控制视频, 可通过 controls 属性来显示控件。

属性名	描述
controls	显示控件
autoplay	自动播放
hidden	隐藏
loop	循环
muted	默认静音
preload	如果出现该属性, 则音频在页面加载时进行加载, 并预备播放。 如果使用 "autoplay", 则忽略该属性。
poster	规定视频下载时显示的图像, 或者在用户点击播放按钮前显示的图像。 如果未设置该属性, 则使用视频的第一帧来代替。

- 关于浏览器格式不支持问题
 - <source> 标签添加在 audio、video 内部, src 链接添加在 source 内, 可以设置多个格式在多个 source 单标签, 放入 src 地址

```
<video>
  <source src="" type="video/mp4">
  <source src="" type="video/ogg">
  <source src="" type="video/avi">
</video>
```

- 按从上到下的顺序引入, 第一个能播放 后面的就不需要,
 - 不同的浏览器 支持不同的格式, 总共就这个三种格式 mp4\ogg\avi
- 关于等比缩放的扩展: 如果不想让子元素等比缩放, 给子元素添加 min-width="100%"

10. 文字注解

- `ruby` 、 `rt` 这样一个组合要结合使用

```
<p> 这是一个段 <ruby>落 <rt> luo </rt></ruby> </p>
<!-- 给文字上方添加注音 -->
```

- `bdo` 可以改变文字排列方向

```
<bdo dir="rtl">这是一个段落</bdo> <!-- 从右向左排列 -->
```

- CSS 样式中如何改变文字排列方向

```
span{ direction : rtl; unicode-bidi: bidi-override;}
```

- `unicode-bidi` 属性与 `direction` 属性一起使用，来设置或返回文本是否被重写。
 - `bidi-override` 属性 创建一个附加的嵌入层面。重新排序取决于 `direction` 属性。

11. link标签的扩展

- 添加外部样式

```
<link rel="stylesheet" type="text/css" href="./样式地址">
```

- 添加网址的小图标

```
<link rel="icon" type="/image/x-icom" href="./图标地址">
```

- DNS 预解析（加快网站访问速度）

```
<link rel="dns-prefetch" href="解析网址">
```

12. meta标签扩展学习

- `meta` 添加一些辅助信息
- 网站描述

```
<meta name="description" content= "大连美团网精选大连美食餐厅,酒店预订,电影票">
```

- 添加搜索关键字

```
<meta name="keywords" content="大连美食, 大连酒店,大连团购" >
```

- 选择浏览器内核

```
<meta name="renderer" content= "webkit" >
```

- IE 解决部分兼容问题

```
<meta http-equiv= "X-UA-Compatible" content="ie=edge" >
```

- 重定向：网页三秒后跳转地址

```
<meta http-equiv="refresh" content= "3" url="跳转的地址">
```

- 指定网页在缓存中的过期时间，一旦网页过期，必须到服务器上重新传输。

```
<meta http-equiv=" expires" content= "Wed, 20 Jun 2019 22:33:00 GMT">
```

13. HTML5新语义化标签

H5标签	描述
header	页眉
footer	页脚
main	主体
hgroup	标题组合
nav	导航
article	独立的内容
aside	辅助信息的内容
section	区域
figure	描述图像或视频
figcaption	描述图像或视频的标题部分
details / summary	文档细节/文档标题

- 注：header、footer、main 在一个网页中只能出现一次
- datalist：选项列表 (用在文本输入框，可以结合 js，提示用户近期输入的内容)

```
<input type="text" list="tishi">
<datalist id="tishi">
  <option value="abbr"></option>
  <option value="asssd"></option>
</datalist>
```

- progress / meter：定义进度条/定义度量范围
 - meter 标签属性: min max value low high
- time：定义日期或时间
 - 我在 <time datetime="2010-02-14">情人节</time> 有个约会。

- `mark` : 带有记号的文本

14. 表格的扩展学习

- 隐藏空单元：`empty-cells:hide;`
- 添加单线：`border-collapse:collapse;`
 - `border-collapse:collapse;` 让表格显示单线边框，两个相邻的单元格只会有一个边框线
- 单元格斜线：`border / rotate` (设置超大边框和文字定位)
- 列分组：`colgroup / col`

15. 表单扩展学习

- 美化表单控件： 1. `label + :checked` ; 2. `position + opacity`
- 新的 `input` 控件

input新增的type属性值	描述
<code>email</code>	电子邮箱地址输入框
<code>url</code>	网址输入框
<code>number</code>	数值输入框
<code>range</code>	滚动条
<code>date / month / week</code>	日期控件， <code>date</code> 具备年月日， <code>month</code> 具备年月， <code>week</code> 具备年和周
<code>search</code>	搜索框
<code>color</code>	颜色控件
<code>tel</code>	电话号码输入框，在移动端会默认调起数字键盘
<code>time</code>	时间控件

- 新的表单属性:

属性名	说明
<code>autocomplete</code>	自动完成 (提示之前输入的信息 默认: <code>on</code> 开启 / <code>off</code> 关闭)
<code>autofocus</code>	获取焦点 (加载时光标在控件上)
<code>required</code>	不能为空
<code>method</code>	数据传输方式 <code>GET</code> , 安全性低, 适合用在查询; <code>POST</code> , 安全性高

属性名	说明
enctype	数据传输类型 字符串：enctype="application/x-www-form-urlencoded" 二进制数据流：可提交文件 enctype="multipart/form-data"
name / value	数据的键值对
pattern	使用在需要验证的 from 控件中，正则表达式验证是否符合规则

- pattern 属性要在有 form 表单时，并且点击该 form 表单的提交按钮后才会验证。
- 扩展标签：
 - fieldset：表单内元组分组(将内部标签包裹)
 - legend：为 fieldset 元素定义标题
 - optgroup：定义选项组

16. BFC规范

- 触发 BFC 规范的元素，可以形成一个独立的容器，不受到外界的影响，从而解决布局问题。
- BFC (Block Formatting Context) 叫做“块级格式化上下文”
 1. 内部的盒子会在垂直方向，一个个地放置；
 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Box的上下 margin 会发生重叠；
 3. 每个元素的左边，与包含的盒子的左边相接触，即使存在浮动也是如此；
 4. BFC 的区域不会与 float 重叠；
 5. 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，反之也是如此；
 6. 计算 BFC 的高度时，浮动元素也参与计算。
 - 触发BFC 规范
 1. 根元素(html)；
 2. float 的属性不为 none ；
 3. position 为 absolute 或 fixed ；
 4. display 为 inline-block , table-cell (表格单元格), table-caption (表格标题), flex , grid ；
 5. overflow 不为 visible
- BFC特性及应用
 - 解决 margin 叠加问题
 - 解决 margin 传递问题
 - 解决浮动问题
 - 解决 float 覆盖问题

1. 浏览器前缀

- css3 兼容不同的浏览器，针对旧的浏览器做兼容，新浏览器基本不需要添加前缀

浏览器	内核	前缀
IE	Trident	-ms-
Firefox	Gecko	-moz-
Opera	Presto	-o-
Chrome	Webkit	-webkit-
Safari	Webkit	-webkit-

2. transition过渡

- 当 属性值被改变时，过渡就会起作用。
- transition-property：规定设置过渡效果 css 属性名称

```
transition-property:width,height,all;
```

 - 可以单选一个属性，也可以选择 ALL 所有
- transition-duration：过渡需要的时间，规定完成过渡效果需要多少秒或毫秒
- transition-delay：过渡效果的延迟，定义过渡效果何时开始
 - 延迟：数值为正数
 - 提前：数值为负数,如果过渡需要时间为 3s，但是延迟为 -2s，过渡效果就会只显示最后一秒的动画
- transition-timing-function：规定速度效果的速度曲线

属性值	描述
linear	匀速
ease (默认值)	逐渐慢下来
ease-in	加速
ease-out	减速
ease-in-out	先加速后减速

- cubic-bezier (贝塞尔曲线)：<https://cubic-bezier.com>

3. transform变形

- `translate` : 位移

```
transform : translate( x , y );
```

- `translateX` `translateY` `translateZ` (3D)

- `scale` : 缩放 , 值是一个比例值, 正常大小就是 1 , 会以当前元素中心点进行缩放

```
transform : scale( 宽 , 高 )
```

- `scaleX` `scaleY` `scaleZ` (3D)

- `rotate` : 旋转 , 旋转的值, 单位是角度 `deg` 弧度 `rad`

```
transform : rotate( 角度 )
```

- `rotateX` : 3D

- `rotateY` : 3D, `Y` 轴旋转, 会按 `transform-origin` 基点, 顺时针旋转

- `rotateZ` : 和 `rotate` 是等价关系, 正值按顺时针旋转, 负值按逆时针旋转

- `skew` : 斜切

```
transform : skew( x , y )
```

- `skewX` : 单位也是角度(`deg`), 正值向左倾斜, 负值向右倾斜

- `skewY`

- `transform` 注意事项:

1. 变形操作不会影响到其他元素。

2. 变形操作只能添加给块元素, 不能添加给内联元素

3. 复合写法, 可以添加做个变形操作。

- 执行是有顺序的, 先执行后面的操作, 再执行前面的操作。

- `translate` 会受到 `rotate`、`scale`、`skew` 的影响

- `transform-origin` : 设置基点 `x` `y` `z` (3d)

- `transform-origin` : `center center` (默认值), 可以设置数值和单词

- `transform` 在 HTML 中, 可以拥有很多相同属性。依次执行

- `transform: rotateX(30deg) rotateX(10deg);`

- 先旋转 `10deg` , 再旋转 `30deg` . 属于依次执行

4. animation动画

animation 属性	描述
<code>animation-name</code>	设置动画的名字 (自定义的)
<code>animation-duration</code>	设置动画的持续时间
<code>animation-delay</code>	动画的延迟

animation 属性	描述
animation-iteration-count	动画的重复次数,默认就是 1 , infinite 无限次数
animation-timing-function	动画的运动形式 ease 、 linear 、 ease-in ...
animation-play-state	控制动画是否暂停 running 让动画运动起来, paused 让动画停下来

- 复合写法: `animation: mybox 4s 1s infinite linear running;`

1. 运动结束后, 默认情况下会停留在起始位置。

2. `@keyframes` : `from -> 0%` ; `to -> 100%`

```
@keyframes mybox{
  0%{ transform: translate(0,0);}
  25%{ transform: translate(200px,0);}
  50%{ transform: translate(200px,200px);}
  75%{ transform: translate(0,200px);}
  100%{ transform: translate(0,0);}
}
```

- `animation-fill-mode` : 规定动画播放之前或之后,其动画效果是否可见。
 - `none` (默认值) : 在运动结束后,回到初始位置,在延迟情况下,让 0% 在延迟后生效。
 - `backwards` : 在延迟情况下, 让 0% 延迟时生效。(在延迟时就执行 0% 的效果)
 - `forwards` : 在运动结束之后, 停到结束位置。
 - `both` : `backwards` 和 `forwards` 同时生效。
- `animation-direction` : 属性定义是否应该轮流反向播放动画。
 - `alternate` : 一次正向(0%~100%),一次反向(100%~0%)
 - `reverse` : 永远都是反向,从 100%~0%
 - `normal` (默认值) : 永远都是正向, 0%~100%
 - 只有播放两次及以上次数的动画, `alternate` 才会生效。
- 扩展 : 通过 `document.styleSheets[0]` 可以获取 当前页面的 style 样式

```
var style = document.styleSheet[0]
style.insertRule( rule, index )
/* rule : 要添加到样式表的规则。
   <1>'@keyframes box{...}'
   <2>'#box{ ... }'
   index: 要把规则插入或附加到 cssRules 数组中的位置。*/

style.rules.length // 查看已有样式条数
style.deleteRule(index) // 删除对应下标的 样式表规则

style.rules // 所有的属性
```


5. animate.css动画库

- 官网地址: <https://daneden.github.io/animate.css/>
- 基本使用:
 - `animated` : 基类, 基础的样式, 每个动画效果都需要加
 - `infinite` : 动画的无限次数

6. transform3D相关属性

- `transform` :
 - `rotateX()` : 正值向上翻转,
 - `rotateY()` : 正值向右翻转
 - `translateZ()` : 正值向前, 负值向后
 - `scaleZ()` : 立体元素的厚度
 - X轴方向, 网页右方; Y轴方向, 网页下方; Z轴方向, 面向自己
 - 关于旋转的正负: 手变成拳头, 大拇指竖直, 指向对应轴的方向, 其他四个手指弯曲的方向就是正
 - 3D写法
 - `scale3d()` : 三个值 `x` `y` `z`
 - `translate3d()` : 三个值 `x` `y` `z`
 - `rotate3d()` : 4个值 `0|1` (`x` 轴是否旋转) `0|1` (`y` 轴是否旋转) `0|1` (`z` 轴是否旋转) `deg`
- 3D相关属性
 - `perspective` (景深): 离屏幕多远的距离去观察元素, 值越大幅度越小。(景深要加给父元素)
 - `perspective` 相当于在屏幕前架设一个相机, 我们看到的画面, 是浏览器射影在相机里的画面, 值越大, 相机离浏览器页面越远
 - `perspective-origin` : 景深-基点位置, 观察元素的角度
 - 景深基点, 就是摄像机的位置
 - `perspective` 景深和 `perspective-origin` 基点位置 要加在同一个元素上才会生效
 - `transform-origin` : `x` `y` `z`
 - `transform-origin: center center -50px;` (`z` 轴只能写数值)
 - `transform-style` : 3D空间
 - `flat` (默认值 2D)、`preserve-3D` (3D, 产生一个3D空间)
 - `backface-visibility` : 背面隐藏 (可以理解为, 物体的背面看不到)
 - `hidden`、`visible` (默认值)
- 3D扩展: 如果景深(`perspective`)为 `d`, `z` 轴宽度为 `z`, 缩放的数值 `d/(d-z)` 是成比例的

7. 背景扩展

- `background-size` : 背景图的尺寸大小
 - `cover` : 覆盖 (背景图覆盖填充整个容器, 等比放大, 会溢出容器)
 - `contain` : 包含 (背景图等比放大, 但不会溢出容器)
 - `background-size : x , y;` 可以设置数值/百分比, 百分比是按照容器大小来定, 而不是图片原始大小。(也可以 `background-size : x , auto;`)
- `background-origin` : 背景图的填充位置
 - `padding-box` (默认)、`border-box`、`content-box`
- `background-clip` : 背景图的裁切方式
 - `border-box` (默认)、`padding-box`、`content-box`
 - 扩展: 以文字为路径, 裁切背景, 可以形成**渐变文字** (IE不支持)
 - `-webkit-background-clip: text;` : 谷歌前缀支持
 - `-moz-background-clip: text;` : 火狐前缀支持
- 注: 这些属性也可以改变背景填充色
- 在复合样式中, 先写 `origin` (位置), 再写 `clip` (裁切)
 - `background: url() no-repeat position/size origin clip;`

8. css3渐变

- `linear-gradient` : 线性渐变

```
background-image: linear-gradient(to top, red 25%, blue 75%,);  
/* 方向从下到上, 在75%到25%执行渐变, 其他地方是纯色 */
```

- `to top` : 是从下到上的渐变, 也可以设置 `to left top`, 从右下到左上
 - 也可以设置角度, `0deg` 在下方, 正值按顺时针旋转
- 扩展: 如何在一个背景里面添加两个不渐变的背景色

```
background-image: linear-gradient(red 50%, blue 50%,)
```

- 添加三个不渐变的背景色

```
background-image: linear-gradient(red 40%, blue 40% 60%, yellow 60%)
```

- `radial-gradient` : 径向渐变

```
background-image: radial-gradient(center, shape, size, start-color, ..., last-color);
```

- `center` : 渐变起点的位置, 可以为百分比, 默认是图形的正中心。
- `shape` : 渐变的形状

- `ellipse` 表示椭圆形, `circle` 表示圆形。默认为 `ellipse`, 如果元素形状为正方形的元素, 则 `ellipse` 和 `circle` 显示一样。
- `size`: 渐变的大小, 即渐变到哪里停止, 它有四个值。
 - `closest-side`: 最近边; `farthest-side`: 最远边; `closest-corner`: 最近角; `farthest-corner`: 最远角
- 正常情况下设置颜色的填充就好

```
background-image: radial-gradient(red 5%, green 15%, blue 60%);
```

9. 字体图标

- `font-face` 是 CSS 中的一个模块, 他主要是把自己定义的 Web 字体嵌入到你的网页中。
- 好处:
 1. 可以非常方便改变图标大小(`font-size`)和颜色(`color`)
 2. 放大后不会失真
 3. 减少请求次数和提高加载速度
 4. 简化网页布局
 5. 减少设计和前端工程师的工作量
 6. 可使用计算机没有提供的字体
- 使用:

```
@font-face{
  font-family : 定义字体名称
}
```

- 通常都是使用 `class` 分别给每个字体图标分开标识, 使用时引用 `class`
- 像 `.woff` 等文件都是做兼容平台处理的 `mac`、`linux`
- 使用字体图标方法:
 - 把图标添加至阿里巴巴项目, 下载项目, 项目文件放入 `css` 文件夹, 具体操作看 `demo.html`
- 如何自定义图标:
 - <https://icomoon.io/app>: 在线生成字体图标, `.css`文件内路径要确认
- 阿里 `iconfont` 使用
 1. 把需要的图标加入购物车添加至项目
 2. 下载项目到桌面
 3. 解压到项目目录
 4. 如果只用字体图标而不用彩色图标, 只需要 `iconfont.css` 和一些兼容平台的文件 (`.eot/.svg/.ttf/.woff/.woff2`)

1. text-shadow (针对文字)

```
text-shadow: color x y blur, [color x y blur]...
```

- `color` : 阴影颜色, `blur` : 模糊半径
- 阴影默认颜色和文字颜色相同
- 多阴影操作作用逗号隔开
- 阴影 `x` 轴设置正值, 就是下方阴影, 设置负值就是上方阴影, `y` 同理

2. box-shadow (针对容器)

```
box-shadow: X Y blur spread color inset;  
box-shadow: 10px 10px 2px 5px red inset;
```

- `spread` : 阴影范围扩大
- `inset` : 外阴影(默认), `outset` 外阴影
- 盒子阴影的默认颜色是黑色。

3. mask 遮罩

- 遮罩用的图片: 透明区域遮罩在图片上, 图片上的内容也会边透明

mask
url
repeat
X
Y
w
h
多遮罩

```
-webkit-mask: url('./image/自制/mask4.png') no-repeat right center/200px 200px;
```

- `mask` 目前还没有标准化, 要添加浏览器前缀。
- 默认是x、y都平铺
- `mask` 可以设置 位置、大小、`-webkit-mask-size`; 也可以设置多遮罩, 用两个 `URL()`, 用逗号隔开

4. box-reflect 倒影

```
box-reflect: <direction> <offset> <mask-box-image>;
```

- `direction` : `above` 指定倒影在对象的上边, `below` 下边, `left` 左边, `right` 右边
- `offset` : 倒影与对象之间的间隔 (px %)
- `mask-box-image` :
 - `url` 使用绝对或相对地址指定遮罩图像

- `linear-gradient` 使用线性渐变创建遮罩图像
- `radial-gradient` 使用径向(放射性)渐变创建遮罩图像
- `box-reflect` 目前还没有标准化，要添加浏览器前缀。
- 扩展：`scaleX(-1)` x 轴水平翻转，`scaleY(-1)` y 轴垂直翻转，`scale(-1)` x, y 都翻转。

5. blur模糊

```
filter : blur(2px);
```

6. calc计算

- 四则运算

```
width:calc(100% - 100px); /* 符号两边要用空格。 */
```

7. column 分栏布局

属性	描述
<code>column-count</code>	分栏的个数
<code>column-width</code>	分栏的宽度
<code>column-gap</code>	分栏的间距
<code>column-rule</code>	分栏的边线
<code>column-span</code>	合并分栏

- 注：`column-count` 个数和 `column-width` 宽度不能一起设置

8. 伪元素概念

- 伪元素本质上是创建了一个有内容的虚拟容器。这个容器不包含任何 `DOM` 元素，但是可以包含内容，另外开发者还可以为伪元素定制样式。
- 伪类(`:`) 和 伪元素(`::`) 的区别
 - 数量：
 - 伪类可以同时存在多个拼接：`input:first-child:focus{}`
 - 伪元素只能存在一个：`input::after{}`
 - 位置：
 - 伪元素只能在最后面：`input:checked::after{}`
- 常见的伪元素选择器

伪元素	描述
<code>::first-letter</code>	选择元素文本的第一个字

伪元素	描述
<code>::first-line</code>	选择元素文本的第一行
<code>::before</code>	在元素内容的最前面添加新内容
<code>::after</code>	在元素内容的最后面添加新内容
<code>::selection</code>	匹配用户被用户选中或者处于高亮状态的部分
<code>::placeholder</code>	匹配占位符的文本，只有元素设置了 <code>placeholder</code> 属性时，该伪元素才能生效

9. css Hack

- 用服务器环境测试
- Hack 分类
 1. CSS 属性前缀法
 - 属性前缀法是在 CSS 样式属性名前加上一些只有特定浏览器才能识别 Hack 前缀，以达到预期的页面展现效果。

前缀标识	兼容浏览器	写法
<code>_</code>	IE6	<code>_background:blue;</code>
<code>+ 、 *</code>	IE6/7	<code>+background:blue;</code>
<code>\9</code>	IE6/7/8/9	<code>background:blue\9;</code>
<code>\0</code>	IE8/9/10/11	<code>background:blue\0;</code>

2. 选择器前缀法
 - 选择器前缀法是针对一些页面表现不一致，或需要特殊对待的浏览器，在 CSS 选择器前加上一些只有某些特定浏览器才能识别的 Hack 前缀。

前缀标识	兼容浏览器	写法
<code>*html</code>	IE6	<code>*html .box{ width:100px;height:100px;}</code>
<code>*+html</code>	IE7	<code>*+html .box{ width:100px;height:100px;}</code>
<code>:root</code>	IE9及现代浏览器	<code>:root .box{ width:100px;height:100px;}</code>

3. IE条件注释法(了解)
 - IE10以上已经不支持注释法

前缀标识	兼容浏览器	写法(是写在 body 内容区域)
<code><--[if IE]>...<![endif]></code>	IE	<code><--[if IE]> <div></div> <![endif]></code>
<code><--[if IE 7]>...<![endif]></code>	IE7	

前缀标识	兼容浏览器	写法(是写在 <code>body</code> 内容区域)
<code><--[if lte IE 7]>...<![endif]></code>	IE7及以下	
<code><--[if gte IE 7]>...<![endif]></code>	IE7及以上	
<code><--[if ! IE 7]>...<![endif]></code>	非IE7	

- IE低版本常见 bug
 - `opacity` 透明度 IE8 及以下版本不识别
 - 解决方法: `filter:alpha(opacity=50);`
 - IE6 下的双边距 bug (同时拥有浮动和外边距的块元素会有双倍外边距)
 - 解决方法: `_display:inline;`
 - IE6 下的最小高度 bug (IE6 最小高度为 19px)
 - 解决方法: `overflow:hidden;`
 - IE9 及以下版本 图片链接有边框问题(`<a> `)
 - 解决方法: `border:none;`

10. 渐进增强 与 优雅降级

- 渐进增强: 针对低版本的浏览器进行构建页面, 保证最基本的功能, 然后再针对高级浏览器进行效果、交互等改进和追加功能, 以达到更好的用户体验。
 - 先兼容IE6/7... 然后进行界面优化做更高级的界面
- 优雅降级: 一开始就构建完整的功能, 然后再针对低版本的浏览器进行兼容。
 - 先做更高级的界面, 然后想办法吧IE6/7...也做成这种界面 (可以用图片遮罩)

11. 网页布局扩展

- `margin` 扩展: (内联元素支持 `margin` 左右 和 `padding` 上下左右(`padding` 会覆盖))
 - 没有固定宽时,负值的 `margin-left` 和 `margin-right` 都是可以增加宽度, 正值的相反(减少宽度);
 - 在有固定宽时, `margin-left` 和 `margin-right` 不会增加宽度, 只会产生位移。
 - 注意:
 - `margin-top` 为负值不会增加高度, 只会产生向上位移
 - `margin-bottom` 为负值不会产生位移, 会减少自身的供 `css` 读取的高度
 - 在有固定宽度和高度的时候, `margin-left\right\top` 都会产生位移, 此位移不脱离文档流, 但是会让所有跟随的元素都会被位移
 - `margin-bottom` 位移: 一个父级元素内有两个内联块子元素, 两个子元素设置宽高, 第一个子元素设置 `margin-bottom` 负值, 第二个子元素不设置, 就会看到第一个子元素向下位移。
- 版心, 通栏布局是最基础
- 等高布局
 - 利用 `margin-bottom` 负值和 `padding-bottom` 正值配合
- 三列布局, 左右固定, 中间自适应

1. BFC 方式：左边的左浮动，右边的右浮动，中间内容加 `overflow:hidden`;
2. 定位
3. 浮动（双飞翼布局、圣杯布局）
 - 双飞翼布局：先写中间部分（宽度要 `100%`，不然浮动后的宽度是根据内容决定的），三个容器都浮动，设置左侧右侧 `margin` 负值，设置中间左右 `margin` 的值
 - 圣杯布局：把父容器设置 `margin` 左右值，然后同理，最后用 `transform: translate()` 或者 `position: relative`; 偏移
4. flex 弹性

1. flex弹性 盒模型

- 2009年提出，现在所有浏览器都支持
- flex 属性

作用在flex容器上	作用在flex子项上
flex-direction	order
flex-wrap	flex-grow
flex-flow	flex-shrink
justify-content	flex-basis
align-items	flex
align-content	align-self

- 作用在父容器上的属性
- flex-direction： 用来控制子项整体布局方向，是从左往右还是从右往左，是从上往下还是从下往上。
 - row： 默认值，显示为行。方向为当前文档水平流方向，默认情况下是从左往右
 - row-reverse： 显示为行。但方向和 row 属性值是反的
 - column： 显示为列
 - column-reverse： 显示为列。但方向和 column 属性值是反的。
- flex-wrap： 用来控制子项整体单行显示还是换行显示。
 - nowrap： 默认值，表示单行显示，不换行。
 - nowrap 不会换行。如果子元素宽度大于容器，会压缩子元素，不让子元素溢出；如果压缩的只剩文字和内容，则会溢出。
 - wrap： 宽度不足换行显示
 - wrap-reverse： 子元素从最后一列开始排列，宽度不足向上换行。
- flex-flow: direction wrap
 - flex-flow 属性是 flex-direction 和 flex-wrap 的缩写，表示 flex 布局的 flow 流动性。**第一个值表示方向，第二个值表示换行**，中间用空格隔开。
- justify-content： 决定每行**主轴 方向上子项的对齐和分布方式**。主轴上有多余空间起作用
 - 主轴： 根据 flex-direction 决定。

代码	对齐方式
flex-start	默认值，表现为起始位置对齐。
flex-end	表现为结束位置对齐。

代码	对齐方式
center	表现为居中对齐。
space-between	两端对齐，多余的空白间距在元素中间分配(between 的意思是在...之间)
space-around	around 是环绕的意思， 每个 flex 子项两侧都环绕互不相干扰的等宽空白间距 最终视觉上 边缘两侧的空白只有中间空白宽度的一半。
space-evenly	evenly 是匀称、平等的意思。 视觉上每个 flex 子项两侧的空白间距完全相等。

- align-items：控制副轴上子项在每一行的对齐方式,需要行内拥有多余空间
 - align-items 中的 items 指的就是 flex 子项们，因此 align-items 指的就是 flex 子项们相对于 flex 容器在行内侧轴方向上的对齐方式。

代码	对齐方式
stretch	默认值, flex 子项拉伸,如果子项没有固定高度，则高度和此行高度相等。
flex-start	表现为容器顶部对齐，如果没有固定高，高度根据内容决定。
flex-end	表现为容器底部对齐,如果没有固定高，高度根据内容决定。
center	表现为 垂直居中对齐,如果没有固定高，高度根据内容决定。
baseline	基于项目的行内文字的基线对齐

- align-content 决定每行副轴 方向上子项的对齐和分布方式。副轴上有多余空间起作用
 - align-content 可以看成和 justify-content 相似且对立的属性，如果所有 flex 子项只有一行，则 align-content 属性是没有任何效果的。（align-content 是针对侧轴的）

代码	对齐方式
stretch	默认值。每一行子元素都拉伸，如果共有两行flex元素，则每一行拉伸高度是50%。
flex-start	表现为起始位置对齐。
flex-end	表现为结束位置对齐。
center	表现为居中对齐。

代码	对齐方式
space-between	表现为两端对齐。
space-around	每一行上下都享有独立不重叠的空间。
space-evenly	每一个元素都完全上下等分。

• 作用在子项上的属性

- `order` : 可以通过设置 `order` 改变某一个 `flex` 子项的排序位置。所有 `flex` 子项默认 `order` 属性值为 `0` , 可以小于 `0` , **值越小排序越靠前**。
- `flex-grow` : 属性中 `grow` 是扩展的意思, 扩展的就是 `flex` 子项所占据的宽度, 扩展所**侵占**的空间就是除去元素外的**剩余空白间隙**。默认值为 `0` , 最小值为 `0` , 整个空白空间整合是 `1` 。
- `flex-shrink` : 属性中的 `shrink` 是收缩的意思, `flex-shrink` 主要处理**当 `flex` 容器空间不足的时候, 单个元素的收缩比例**, 默认值为 `1` , `0` 表示不收缩, 最小值为 `0` 。
若想不让元素在空间不足时收缩, 把 `flex-shrink` 设置为 `0` 。
- `flex-basis` : `flex-basis` 定义了**在分配剩余空间之前的默认元素大小**。
 - `flex-basis` 优先级大于宽。设置为 `auto` 后, 如果有固定宽, 则宽度就是固定宽, 如果没有固定宽, 则根据内容决定宽度。
- `flex` : `flex` 属性是 `flex-grow` 、 `flex-shrink` 、 `flex-basis` 的缩写。
 - 属性值仅有一个数字时, 该值为 `flex-grow` 的值, 默认 `flex-shrink` 为 `1` , `flex-basis` 为 `0%` ;
- `align-self` : **控制某一个子项行内垂直对齐方式**。
 - 和 `align-items` 一样, 只不过 `align-self` 是作用在单一子项上
 - `align-self` : 属性允许单个项目有与其他项目不一样的对齐方式, 可覆盖 `align-items` 属性。
- `flex` 特性:
 1. 当内容宽度大于容器宽度时, 内容不会换行, 而是自己调节内容 (多余) 宽度来适应容器。
 2. 子容器高度默认和父容器相等。
 3. 子容器如果不设置宽度, 则宽度根据内容分配。

2. Grid 网格布局

- `Grid` 网格布局是一个二维的布局方法, 纵横两个方向总是同时存在。

作用在grid容器上	作用在grid子项上
grid-template-columns	grid-column-start
grid-template-rows	grid-column-end
grid-template-areas	grid-row-start
grid-template	grid-row-end
grid-column-gap	grid-column
grid-row-gap	grid-row
grid-gap	grid-area
justify-items	justify-self
align-items	align-self
place-items	place-self
justify-content	
align-content	
place-content	

• 作用在 grid 容器上

- grid-template-columns 和 grid-template-rows
 - 对网格进行纵横划分，形成二维布局。单位可以是像素,百分比,自适应以及 fr 单位(网格空间比例单位)。
 - fr 单位，最小值为 0，宽和高默认值为 1，可以设置大于 1，情况和 flex-grow 分配剩余空间一样。

```
grid-template-columns:1fr 2fr 1fr;
/* 表示容器列数为3，容器的宽分为4fr（4份） 中间的列占两份 */

grid-template-rows:.1fr .1fr .2fr;
/* 表示分为3行，总行高占容器高的40%(.4fr)。*/
```

- 有时候，我们网格划分是很规律的，如果要添加多个纵横网格时，可以利用 repeat() 语法进行简化操作。

```
grid-template-rows:repeat( 3 , 1fr )
/* grid-template-rows:repeat( 行数 , 每一行的大小, 可以是百分比, 像素 ) */
```

- `grid-template-areas` 和 `grid-template`

- `area` 是区域的意思，`grid-template-areas` 就是给网格划分区域的。此时 `grid` 子项只要使用 `grid-area` 属性指定其隶属于哪个区域。
 - 用法：在父容器进行划分区域，子容器指定区域名。(子元素添加: `grid-area:区域名;`)
 - `grid-template-areas` **不允许形成特殊图形，只能形成矩形。**
 - `grid-template-areas` 的**命名不允许数字开头。**

```
.grid{
  display: grid;
  width: 300px;
  height: 300px;
  grid-template-rows: repeat(3,1fr);
  grid-template-columns: repeat(3,1fr);
  grid-template-areas:
    "name1 name2 name3"
    "name4 name4 name5"
    "name6 name7 name7";
}
.grid>div:nth-child(1){
  grid-area: name1; /* 使用时不用加"" */
  background: red;
}
```

- `grid-template` 是 `grid-template-rows` , `grid-template-columns` , `grid-template-areas` 属性的缩写。

```
grid-template:
"name1 name2 name3" 1fr
"name4 name4 name5" 1fr
"name6 name7 name7" 2fr
/1fr    1fr    1fr;
```

- `grid-template-columns` 和 `grid-template-rows` 、 `grid-template-areas` 、 `grid-template` 只是指定网格区域，`grid-area: name1;` 指定当前元素入驻哪个网格，**在此网格内的对齐方式由 `justify-items` 和 `align-items` 指定。**

- `grid-column-gap` (纵向) 和 `grid-row-gap` (横向)

- 用来定义网格中网格间隙的尺寸。(注意，间隙在设置对齐方式时，不会被覆盖)
 - `grid-gap` 属性是 `grid-column-gap` 和 `grid-row-gap` 的缩写。
 - `grid-gap` :横向 纵向;

- `justify-items` 和 `align-items`

- `justify-items` 指定了**网格内元素的水平呈现方式**，是水平拉伸显示，还是左中右对齐。
- `align-items` 指定了**网格内元素的垂直呈现方式**，是垂直拉伸，还是上中下对齐。

代码	对齐方式
<code>stretch</code>	默认值, <code>grid</code> 子项拉伸,如果子项没有固定高度, 则高度和网格相等。
<code>start</code>	表现为容器顶部对齐, 并且高度根据内容决定。
<code>end</code>	表现为容器底部对齐,并且高度根据内容决定。
<code>center</code>	表现为 垂直居中对齐,并且高度根据内容决定。
<code>baseline</code>	基于项目的第一行文字的基线对齐

- `place-items` 可以让 `align-items` 和 `justify-items` 属性写在单个声明中。

```
place-items: align-items justify-items; /* 先纵向, 在横向 */
```

- `justify-content` 和 `align-content`

- `justify-content` 指定了网格整体的水平分布方式。需要网格外部有空余
- `align-content` 指定了网格的垂直分布方式。需要网格外部有空余

代码	对齐方式
<code>start</code>	表现为起始位置对齐。(默认值)
<code>end</code>	表现为结束位置对齐。
<code>center</code>	表现为居中对齐。
<code>space-between</code>	表现为两端对齐。
<code>space-around</code>	每一行上下都享有独立不重叠的空间。
<code>space-evenly</code>	每一个元素都完全上下等分。

- `place-content` 可以让 `align-content` 和 `justify-content` 属性写在单个声明中。

```
place-content: align-content justify-content; /* 先纵向, 在横向 */
```

- `items` 是针对网格中的内容排列, `content` 是针对网格的排列

• 作用在子项上的属性

- `grid-column-start` : 水平方向上占据的起始位置
 - 这里的起始位置是网格的 纵向线 起始线数值为 1
- `grid-column-end` : 水平方向上占据的结束位置 (`span` 属性)
- `grid-row-start` : 垂直方向上占据的起始位置
 - 这里的起始位置是网格的 横向线 起始线数值为 1
- `grid-row-end` : 垂直方向上占据的结束位置 (`span` 属性)

- 注: `span` 属性只在 `end` 结束位置中拥有

```
grid-column-start:3; /* 从第三条网格纵线开始 */
grid-column-end: span 2; /* 向后延伸两格距离 */
```

- `grid-column` : `grid-column-start` 和 `grid-column-end` 的缩写
 - `grid-column:start / end` (或`span`属性); 这里的 `/` 是要写在值中间的
- `grid-row` : `grid-row-start` 和 `grid-row-end` 的缩写
- `grid-area` : 表示当前网格所占用的区域, 名字和位置两种表示方法。
 - 名字: `grid-area:a1;`
 - 位置: `grid-area: 2 / 3 / 4(或span属性) / 4(或span属性)`
 - `grid-area: y起始 / x起始 / y结束 / x结束`
- `justify-self` : 单个网格元素的水平对齐方式。
- `align-self` : 单个网格元素的垂直对齐方式。
- `place-self` : `align-self` 和 `justify-self` 的缩写。
 - `place` 目前都是先纵向 `y`, 在横向 `x`
- `content` 控制的是网格的位置, `items` 控制的是网格内元素的对齐位置, `self` 是设置在子项上单独控制这个子项内元素的对其位置

3.移动端模拟器

- 切换平台之后一定要重新刷新网页。

4.PC和移动端的网页。

- 大一点的网站都是分开开发的, `PC` 端一套代码, 移动端一套代码。
- <https://www.taobao.com> -> 后端检测当前设备, `pc` 端访问的和手机端访问的网址不同 (重定向) 。

5. viewport 视口(在移动端才会有)

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

- 在移动端 `viewport` 视口就是浏览器显示页面内容的屏幕区域。在 `viewport` 中有两种视口, 分别表示为, `visual viewport` (可视窗口)和 `layout viewport` (布局视口)。
- `visual viewport` 固定大小跟屏幕大小相同, 在上面;而 `layout viewport` 可改变大小, 在下面。 `layout viewport` 默认大小为 `980` 像素, 可通过 `document.documentElement.clientWidth` 获取。
- 现代网页需要将 `layout viewport` 设置成跟 `visual viewport` 同等大小, 方便网页制作。
- `viewport` 设置

content 属性值	解释
width	设置 layout viewport 的宽度特定值，device-width 表示设备宽。
height	设置 layout viewport 的高度固定值，一般不进行设置。
initial-scale	设置页面的初始缩放。（设置为 1.0）
Minimum-scale	设置页面最小缩放。
maximum-scale	设置页面最大缩放。
user-scalable	设置页面能否缩放。（no 表示不允许）

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<!-- 在 content属性内书写 -->
```

6.移动端适配方案

1. 百分比布局，也叫流式布局。代表网站 优酷、百度、天猫、腾讯。
 - 宽度使用 % 百分比定义，但是高度和文字大小等大都是用 px 来固定，所以在大屏幕的手机下显示效果会变成有些页面元素**宽度被拉的很长，但是高度、文字大小还是和原来一样**
 - 流式布局中，文字大小是不会发生变化的。
 - 流式布局中，图片会根据设备进行等比缩放。
 - 好处：大屏幕下显示更多内容。
 - 坏处：宽屏下比例会有些不协调。
2. 等比缩放布局，也叫 rem 布局。代表网站 网易、爱奇艺、淘宝、美团。
 - em：是一个相对单位，1em 等于当前元素或父元素 font-size 的值。
 - rem：是一个相对单位，1rem 等于根元素(html)的 font-size 的值。
 - vw / vh：把屏幕分为100份，1vw 等于 1% 屏幕的宽，1vh 等于 1% 屏幕的高。（具体看下方单位）
 - 通过动态设置 html 的 font-size 和属性的 rem 可以实现动态布局
 - 动态设置 font-size :通过 JS 或通过 vw
 - 要给 body 重置一下 font-size : 16px;
3. 响应式布局 (利用媒介查询和一套网页适配不同的设备，适合中小型的项目)
 - 布局特点：每个屏幕分辨率下面会有一个布局样式，即元素位置和大小都会变。
 - 利用媒体查询，即 media queries，可以针对不同的媒体类型定义不同的样式，从而实现响应式布局。
 - @media 可以用如下几种方式定义: @media not screen and (max-width: 500px){}

```
<link rel="stylesheet" href="print.css" type="text/css" media="print" />
```



```
@import url("print.css") screen;
```

```
@media print { body { font-size: 12px; } }
```

```
<style media="print">body{font-size:12px;}</style>
```

■ 媒体类型

名称	描述
all	用于所有设备
print	用于打印机和打印预览
screen	用于电脑屏幕，平板电脑，智能手机等。
speech	应用于屏幕阅读器等发声设备

■ 运算符：AND 与，NOT 取反，OR 或。

■ 常见选项：

- min-width 、 max-width

orientation:portrait (纵向)、 orientation:landscape (横向)

```
@media screen and (min-width:960px) and (max-width:1200px){  
  body{background:yellow;}  
  /* 表示浏览器宽度大于等于960px且小于等于1200px时使用样式。*/  
}
```

■ 响应式代码写到要适配的 css 后面。

扩展：html中的单位

单位	描述
px	绝对单位，页面按精确像素展示
em	相对单位，1em=父节点字体大小
rem	相对单位，相对根节点 html 的字体大小来计算，chrome 强制最小字体为12号
vw	viewpoint width，视窗宽度，1vw 等于视窗宽度的 1%。
vh	viewpoint height，视窗高度，1vh 等于视窗高度的 1%。
vmin	vw 和 vh 中较小的那个。
vmax	vw 和 vh 中较大的那个。

单位	描述
%	百分比
in	寸
cm	厘米
mm	毫米
pt	point , 大约 1/72 寸
pc	pica , 大约 6pt , 1/6 寸

- 关于图片问题扩展

```
<picture>
  <source media="(max-width:800px)" srcset="./11.jpg">
  
</picture>
```

- 当浏览器宽度小于等于 800px ,图片显示会变换。

1. Bootstrap4 框架

- Bootstrap 是最受欢迎的 HTML \ CSS \ JS 框架，用于开发响应式布局、移动设备优先的web项目。(以下都是bootstrap4)
 - 特色：
 1. 响应式布局。
 2. 基于 flex 的栅格系统。
 3. 丰富的组件和工具方法。
 4. 常见交互使用。
 - 官网:<https://getbootstrap.com>
 - 中文:<https://getbootstrap.net/>
 - 下载完成原码后, dist 文件夹是 bootstrap 内容, scss 文件夹是关于源码的文件
 - bootstrap.css 、 bootstrap-grid.css 、 bootstrap-reboot.css 比较重要
 - bootstrap.css 包含 bootstrap-grid.css 和 bootstrap-reboot.css
 - bootstrap-grid.css 是栅格系统(flex)
 - bootstrap-reboot.css 是重置 css (重置默认样式)
- Containers
 - 屏幕断点: 默认(100%) | >=576px | >=768px | >=992px | >=1200px
 - container 居中，适配不同宽度的 max-width 尺寸。（版心）

container其他格式	描述
container-sm	和container断点相同
container-md	减少了 576 断点
container-lg	减少了 768 断点
container-xl	减少了 992 断点

- container-fluid ： 通栏(宽度 100%)

2. bootstrap-grid

- 要布局在 container 容器中

```
<div class="container">
  <div class="row">
    <div class="col-sm-6"></div>
    <div class="col-sm-6"></div>
  </div>
</div>
```

- row 是一行,行中可以有多列
- col 是一列,列后可以跟 sm , md , lg , xl ;决定了该列什么时候的宽度为什么
 - 只写 col ,列将不会有断点,任何分辨率下都会平分
 - 行的宽被分成 12 份,可以在列的断点后 跟数字决定分给这个列多大的宽度

- 不给列设置断点，列将会平分行的宽度
- 列默认带有左右 15px 边距
- `.w-100`：宽度 100%
- 根据内容自适应宽度
 - `col-auto` 或 `col-断点-auto`
 - `col-auto`：一直都是根据内容决定宽度
 - `col-断点-auto`：在断点之前(宽度 100% 之前)都是根据内容决定宽度
- 可以设置多个 `col-断点-长度` 来设置不同设备下,断点的宽

```
<div class="row g-0">
  <div class="col-md-6 col-lg-4 col-xl-3">aaa</div>
  <div class="col-md-6 col-lg-4 col-xl-3">aaa</div>
  <div class="col-md-6 col-lg-4 col-xl-3">aaa</div>
  <div class="col-md-6 col-lg-4 col-xl-3">aaa</div>
</div>
```

- `.row-cols-3`

```
<div class="row row-cols-3"> </div>
```

- 一行放三个列
- `row-cols-md-4`：在 768 之上 一行可以放三个列
- `row-cols-lg-4`：在992之上 一行可以放三个列
- 如果在一行内子 `div` 定义的栅格总数超过 12 列，Bootstrap会在保留列完整的前提下，将无法平行布局的多余列，重置到下一行，并占用一个完整的新行。
- 作用在行的 `class`

class	描述
<code>row-cols-3</code>	一行放三个列
<code>align-items-start</code>	行中元素贴上
<code>align-items-center</code>	行中元素居中
<code>align-items-end</code>	行中元素贴下
<code>justify-content-start</code>	行内元素贴左
<code>justify-content-center</code>	行内元素居中
<code>justify-content-end</code>	行内元素贴右
<code>justify-content-around</code>	元素间隔相等对齐
<code>justify-content-between</code>	两端对齐

- 作用在列中的属性

class	描述
<code>col-auto</code>	根据内容自适应宽度
<code>align-self-start</code>	在行内上部对齐
<code>align-self-center</code>	在行内居中对齐
<code>align-self-end</code>	在行内底部对齐
<code>order-frist</code>	将此列排序提升为第一
<code>order-last</code>	排在最后
<code>order-1</code>	数值越小,优先级越大 (取值为0-5) 如未定义排序优先级, 将不会改变位置
<code>offset-md-3</code>	控制间距, 决定列左边 空出几个栅

- margin: `mr/ml/mt/mx/my` - `auto|0~5` , padding同理
- 嵌套: 可以在 `col` 中嵌套 `row` (也是等分12份)

3. bootstrap-content(内容)

1. 将样式重置 -> 阅读原码 查阅重置样式

* 表单新属性hidden: `<input type="text" hidden>` 相当于 `display:none;`

2. 排版

- `h1-h6` : 可以将其他标签字体大小和间距显示成 `h1-h6` ,但不改变 `display` ;
 - `<p class="h1">这是测试</p>`
- `display - 1~4` : 文字更醒目

4. bootstrap-Components(组件)

1. `alert` : `<div class="alert alert-primary">`
2. `badge` (徽章)

```
<div class="alert alert-success">
测试文本
<span class="badge badge-light">new</span>
</div>
```

- 徽章只是将容器变成合适的大小,需要通过 `text-light` \ `bg-light` 调控颜色

3. `breadcrumb` (导航)

```
<div class="bg-info"> <!-- 背景色 -->
  <ol class="breadcrumb px-5">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Library</a></li>
    <li class="breadcrumb-item active">Data</li>
  </ol>
</div>
```

4. button

```
<button type="button" class="btn btn-primary">Primary</button>
```

5. 按钮组(将多个按钮连接)

```
<div class="btn-group btn-group-lg">
  <button type="button" class="btn btn-secondary">Left</button>
  <button type="button" class="btn btn-secondary">Middle</button>
  <button type="button" class="btn btn-secondary">Right</button>
</div>
```

- btn-group-sm\lg: 添加在父元素,调控按钮大小 (只有 sm 和 lg 两种)

6. card (卡片): 图文描述 加 按钮

7. 滚动条: 滑动滚动条会实时滑动内容

- 安装 vscode 组件 bootstrap 4 ; 输入 b4 可以查看相关组件提示

5. Utilities(公共样式)

1. border :添加边框

- border : 默认1px浅灰色边框(#dee2e6)
- border - top\end\bottom\start - 0~5
- border - success/warning... : 设置颜色
- 圆角: rounded - circle/pill/sm/lg/top/bottom/left/right , 或只写 rounded

2. clearfix : 添加在父元素清除浮动

```
<div class="clearfix">
  <div class="float-left">float</div>
</div>
```

3. 对齐处理: text-center、text-end、text-start

4. 定

位: position-static、position-relative、position-absolute、position-fixed、position-sticky

5. 浮动: float-left、float-right、float-none

- float - sm\md\lg\xl - left\right\none

6. display : d-inline、d-block、d-inline-block、d-flex、

- d - sm\md\lg\xl - block

7. top/bottom/start/end - 0/50/100

- top:50% 定位使用, 只有三种值

8. overflow - auto\hidden

- 9. 响应式图片 `mw-100 + h-auto`
- 10. `text`bg``
 - `text - primary\info\success...`
 - `bg - primary\info\success...`
- 11. 宽和高
 - `w - 25\50\75\100\auto` : 高同理
- 12. 隐藏文字: `text-hide`

- 更多查阅文档中文:<https://getbootstrap.net/>

6. 个人网页博客制作

- bootstrap 样板网页:<https://getbootstrap.com/docs/5.1/examples/>

7. Sass和Less

- Sass 和 Less 都属于 CSS 预处理器, CSS 预处理器定义了一种新的语言,其基本思想是,以一种专门的编程语言,为 CSS 增加了一些编程的特性,如:变量、语句、函数、继承等概念。将 CSS 作为目标生成文件,然后开发者就只要使用这种语言进行 CSS 的编码工作。
 - CSS 预处理器: Sass less stylus
 - less 官网: <http://lesscss.org/>
 - Less 中文: <http://lesscss.cn/>
 - VSCode 插件: Easy LESS
 - Sass 官网: <https://sass-lang.com/>
 - Sass 中文: <https://sass-lang.cn/>
 - VSCode 插件: Easy Sass
 - 文件名.less -> 生成 CSS 文件
 - 文件名.scss -> 生成对应的 CSS 和 min.css 文件

8. Sass和Less的基本语法

- 注释
- 变量、插值、作用域
- 选择器嵌套、伪类嵌套、属性嵌套(Sass)
- 运算、单位、转义、颜色
- 函数
- 混入、命名空间(Less), 继承
- 合并、媒体查询
- 条件、循环
- 导入...

9.注释、变量、插值、作用域

- 注释: Less 和 Sass 单行注释都不会被编译,多行注释会被编译, sass 的 min.css 文件注释不编译
- 变量:

- Less: @变量名 : 值;

```
@number: 123px;
.box{
    width: @number;
}
```

- Sass: \$变量名: 值;

```
$number: 123px;
```

- 插值:类似字符串拼接

- Less: @{变量名}

```
@{key}: margin;
@i: 2;
.box@{i}{
    @{key}: auto;
}
.box2{
    margin: auto;
}
```

- Sass

```
#{$变量名}
$key:margin;
$i: 3;
.box#{i}{
    #{key}:10rem;
}
```

- 作用域:

- Less : 就近原则

```
@number: 200px;
.box{
    width: @number; // 100px;
    @number: 100px;
    margin: @number; // 100px;
}
```

- Sass : 先后顺序

```
$number: 200px;
.box{
    width: $number; // 200px;
    $number: 100px;
    margin: $number; // 100px;
}
```

10. Sass和Less选择器嵌套、伪类嵌套、属性嵌套(Sass)

- 选择器嵌套: Less和Scss同理


```
ul{
  list-style: none;
  li:first-child{
    text-align: left;
  }
  li{
    text-align: center;
    p{
      color: red;
    }
  }
}
```

◦ 生成

```
ul { list-style: none; }
ul li:first-child { text-align: left; }
ul li { text-align: center; }
ul li p { color: red; }
```

• 伪类嵌套： Less和Scss同理 `&:hover`

```
ul{
  &:hover{
    color: red;
    li{
      color: red;
    }
  }
}
```

◦ 生成

```
ul:hover { color: red; }
ul:hover li { color: red; }
```

• 属性嵌套： **Sass 拥有**

```
.box5{
  font : {
    size: 20px;
    weight:bold;
  };
  background : {
    color: red;
    repeat:repeat-x;
  };
}
```

◦ 生成

```
.box5 {  
  font-size: 20px;  
  font-weight: bold;  
  background-color: red;  
  background-repeat: repeat-x;  
}
```

11. Sass和Less运算、单位、转义、颜色

- 运算

- Less: `less` 中进行运算时,会以前面的单位为标准

```
@num : 100px;  
.box4{  
  width: @num * 3; // 300px  
  height: @num + 10em; // 110px  
  margin: 10em + @num; // 110em  
}
```

- **Sass中不同单位是不能运算的**

- 转义

- Less

```
padding: ~"20px / 1.5" // ~为拒绝转义字符,引号内的内容不会运算
```

- Scss

```
padding: (20px / 1.5) // '/'默认进行分割,使用小括号进行运算
```

- 颜色: Scss和Less同理

```
color: #010203 *2; // #020406; 颜色也会进行运算
```

- 以下情况Sass和Less可以运算 `2 + 1px`、`1px + 2`

12. Sass和Less函数

```
// Sass自定义函数  
@function sum($m,$n){  
  @return $m+$n;  
}  
sum(3px,2px)
```

13. Sass和Less混入、命名空间(Less)、继承

- 混入

- Less: `.类名{ .类名 }`

```
.show{ display: 'block'; }
.box{
  width: 90%;
  .show; // display:'block';
}
```

- 在标签名后加 `()` , `.show(){}` 之后,此标签样式不会被渲染

```
.hide(@color){
  color: @color;
}
.box{
  color : .hide(blue);
}
```

- Sass : `@mixin 类名{ }`

```
@mixin show{
  display:block;
}
.box{
  @include show;
}
```

- Sass的混入不会渲染，可以传参

```
@mixin hide($color){
  display: none;
  color: $color;
}
.box{
  @include hide(blue);
}
```

- 命名空间(Less)： `#名称{ 类名{ } }`

```
.show{ width:20px; }
#nm(){
  .show(@h){
    width:10px;
    height: @h;
  }
}
.box{
  .show; // width: 20px;
  #nm.show(20px); // width:10px; height:20px;
}
```

- 继承

- Less: `&:extend(类名)`

```
.line{ display:inline; }
.box2{ &:extend(.line); }
.box3{ &:extend(.line); }
```

- 结果:

```
.line, .box2, .box3 {  
  display: inline;  
}
```

- Scss: @extend 类名

```
.line{ display: block; }  
.box6{ @extend .line; }  
.box7{ @extend .line; }
```

- 结果:

```
.line, .box6, .box7 {  
  display: block;  
}
```

- sass占位符 %, 此处样式不会被渲染

```
%line{ display: block; }  
.box7{ @extend %line; }
```

- 结果:

```
.box7 { display: block; }
```

14. Sass和Less合并、媒体查询

- 合并

- Less

```
.box9{  
  background+: url();  
  background+: contain;  
  transform+_: scale();  
  transform+_:translate();  
}
```

- 结果:

```
.box9 {  
  background: url(), contain;  
  transform: scale() translate();  
}
```

- 使用 + 会让属性 隔开, 使用 +_ 会让属性 空格 隔开

- Scss

```

$background:(
  a:url(),
  b:red
);
$transform:(
  a:scale(2),
  b:rotate(30deg)
);

.box9{
  background: map-values($background);
  transform: zip(map-values($transform)...);
}

```

■ 结果:

```

.box9 {
  background: url(), red;
  transform: scale(2) rotate(30deg);
}

```

• 媒体查询: Less和Scss一样

```

.box10{
  width:100px;
  @media all and (min-width:768px) {
    width: 200px;
  }
}
.box11{
  width:100px;
  @media all and (min-width:768px) {
    width: 200px;
  }
}

```

- 以上会生成单独的 @media ,和在Less中使用以下方式没有区别

```

@media all and (min-width:768px) {
  .box10{ }
  .box11{ }
}

```

15. Sass和Less条件、循环、导入

- 条件(尽量使用js,此条件不能实时渲染)
 - Less

```

@num:40;
.get(@cn) when ( @cn > 4 ){
    width: 100px + @cn;
}
.get(@cn) when (@cn <= 4){
    width: 10px + @cn;
}

.box12{
    .get(@num); -> 140px;
}

```

- 类似于 Less混入, 后面 when 相当于条件判断,判断成功就渲染 (Less混入也可以进行传参)
- 可以存在多个相同名的条件判断,组成 if...else

◦ Scss

```

$count:4;
.box12{
    @if($count > 4){
        width: 100px + $count;
    }@else if($count <=3 ){
        width: $count + 20px;
    }@else{
        width: $count + 10px;
    }
}

```

• 循环

◦ Less -> 通过 条件 + 递归

```

@count:0;
.get2(@cn)when(@cn < 3){
    .box-@{cn}{
        width: 10px + @cn;
    }
    .get2((@cn+1));
}
.get2(@count);

```

- 结果:

```

.box-0 { width: 10px; }
.box-1 { width: 11px; }
.box-2 { width: 12px; }

```

◦ Scss : 提供了 for 、 while 循环

```

@for $i from 0 through 2{ // 循环 0,1,2
    .box-#{ $i }{
        width: 10px + $i;
    }
}

```

- 结果:

```
.box-0 { width: 10px; }  
.box-1 { width: 11px; }  
.box-2 { width: 12px; }
```

- 导入：类似于导入模块，可以将其他 `.less`、`.scss` 文件内容,导入本文件中
 - Less和scss相同

```
@import '文件路径.less'
```

1. PostCSS -> 对Css进行工程化处理

- PostCSS 本身是一个功能比较单一的工具。它提供了一种方式用 JavaScript 代码来处理 CSS。利用 PostCSS 可以实现一些工程化的操作,如:自动添加浏览器前缀、代码合并、代码压缩等。
- 官网: <https://postcss.org>
- 中文文档: <https://www.postcss.com.cn/>
- 安装:
 1. 安装 node 环境 (<https://nodejs.org/en/download/>)
 - 通过 cmd node -v 可查看 node 版本确定是否安装成功
 - 可以安装 nvm , 管理 node 版本
 2. npm install postcss-cli -g
 - -g 为全局安装,可以在任意目录下使用 post 命令操作
 - postcss 安装失败解决: npm i postcss autoprefixer@8.0.0 , 版本问题-更新版本
 3. -o 、 -w
 - 进入要转换 css 的目录,在此目录 cmd
 - postcss 要转换文件的路径.css -o 转换好的目标文件.css -w
 - postcss src/demo.css -o dist/demo.css -w
 - 将 src 中的 demo.css 通过 postcss 转换到 dist 中
 - -w 代表实时监听, 修改文件之后不用手动转换.
 - 断开程序: ctrl + c 重复两次(断开实时监听)
 4. postcss.config.js
 - postcss 的配置文件, 在 源文件.css 目录下创建 postcss.config.js , src/postcss.config.js

2. PostCSS常用插件

- 进入官网点击 Plugins (插件), 可以搜索插件, 可以查看插件的使用方法
- 插件安装: 1. 在项目目录 cmd , 2. npm i 插件名
- autoprefixer : 给浏览器自动添加前缀
 - 配置页面:

```
const autoprefixer = require('autoprefixer');
module.exports = {
  plugins : [
    autoprefixer({
      browsers : [' > 0% ',] //给市场份额大于0%的浏览器添加前缀
    })
  ]
}
```

- postcss-import : 合并css

- `src/demo.css` : 使用 `@import './reset.css'`;

- 配置页面:

```
const pcIimport = require('postcss-import');

module.exports = {
  plugins : [
    pcIimport // 放入 pcIimport
  ]
}
```

- `cssnano` , 对 `css` 进行压缩处理, 去除一些不必要的空格和回车, 节省空间

- 配置页面:

```
const cssnano = require('cssnano');

module.exports = {
  plugins : [
    cssnano
  ]
}
```

- `postcss-cssnext` : 处理比较高级的 `css` 语法(一些高级语法, 很多浏览器不支持, 可以使用它来给 `css` 降级)

- 配置页面:

```
const cssnano = require('cssnano');

module.exports = {
  plugins : [
    cssnano
  ]
}
```

- `src/demo.css` (自己编写的)

```
:root{                                /* :root里面定义的是变量 */
  --color:red;
  --vw:100%;
}
div{
  background: var(--color);  /* 通过var(变量名引入) */
  width: var(--vw);
}
```

- `dist/demo.css` (生成的)

```
div{
  background: red;
  width: 100%;
}
```

- **stylelint** : 代码规范检测

- 配置页面:

```
const stylelint = require('stylelint');

module.exports = {
  plugins : [
    stylelint({ // 在此填写, 编写css的规范, 和eslint一样
      "rules" : {
        "color-no-invalid-hex" : true
      }
    })
  ]
}
```

- 代码不规范, 会在 cmd 窗口报错代码不规范的具体位置

- **postcss-sprites** : 精灵图(雪碧图), 默认将 源文件.css 中引入的图片都合成为一张雪碧图,放入 **spritePath** 指定的目录中

- 配置页面:

```
const sprites = require('postcss-sprites');

module.exports = {
  plugins : [
    sprites({
      spritePath: './dist'
    })
  ]
}
```

- **注意**: 以上的配置信息都放在 **module.exports** 对象中的 **plugins** 数组中, 使用 , 隔开

1.CSS架构与文件组织

- 在一个大型项目中,由于页面过多,导致CSS代码难以维护和开发。所以CSS架构可以帮助我们解决文件管理与文件划分等问题。
- 首先要对CSS进行模块化处理,一个模块负责一类操作行为。利用Sass或Lass来实现。
- CSS架构：将不同功能的 .css 文件放入不同的文件夹

文件夹	含义
base	一些初始的通用CSS,如重置默认样式,动画,工具,打印等。 工具：浮动简单封装的 class ,清除浮动的 class 、简单定位的 class 等 打印：需要进行打印机打印的设备样式
components	用于构建页面的所有组件, 如按钮,表单,表格,弹窗等。
layout	用于页面布局的不同部分,如页眉、页脚、弹性布局、网格布局等。
pages	放置页面之间不同的样式, 如首页特殊样式,列表页特殊样式等 一个网站页面特别多,但是重复的样式很多, 这时只需要建立一个 .css 编写其中的特殊样式
themes	应用不同的主体样式时, 如管理员、买家、卖家等
abstracts	放置一些如 变量、函数,响应式等辅助开发的部分
vendors	放置一些第三方独立的CSS文件, bootstrap 、 iconfont 等

- 在文件夹内,建立对应的 css 文件 如 components ->(_button.scss , _form.scss , _table.scss ...), 最后通过 main.scss 将所有样式引入
- 还可以使用vue等框架开发

2.css新特性之 自定义属性

- CSS自定义属性(也称之为 ‘CSS变量’), 在目前所有现代浏览器中都得到了支持。
 - Less: @变量名： 属性值
 - Sass: \$变量名： 属性值
 - 原生CSS定义变量:

```
:root{
  --color:red; /* --属性名 : 值; */
}
.box{
  /* 标签中也可以定义变量 */
  --color : blue;
  color: var(--color); /* blue , 看作用域*/
}
```

- `:root` 相当于文档根元素 `html`
- 老式浏览器不支持 自定义属性的,可以使用 `PostCss` 中 `postcss-cssnext` 进行降级

◦ 计算:

```
:root{
  --number: 100;
}
.box{
  width: calc( var(--number) * 1px )
}
```

◦ 默认值

```
.box{
  width: var(--wid, 100px);
}
```

- 当前面的变量找不到时, 会使用后面的值 (后面的值不能为变量,但是可以为 `var(--变量)`) `color: var(--c,var(--cc));`

- **只能拥有一个备用值**

◦ 作用域: 按照会影响到标签的优先级

- `style` 行间 > `id` > `class` > `tag` (标签) > `*` > 继承
 - `Less` 是就近原则 `Sass` 是先后顺序

1.

```
:root{
  --color: blue;
}
.box{
  color: var(--color);    /* red */
  --color: red;
}
```

2.

```
:root{
  --color: blue;
}
div{
  --color: gray;
}

.box{
  color: var(--color);
}
```

- `class` 为 `box` 的 `div` 颜色为 `gray`,其他的为 `blue`

◦ 在 `:root` 中声明的一些样式, 是默认最高级别的(需要是可继承的样式)

```
:root{
  font-size:12px; /* 在不覆盖的情况下, 所有标签默认字体大小都是12px */
}
```

3.CSS新特性之 shapes

- CSS shapes 布局可以实现不规则的文字环绕效果,需要 配合浮动使用
- 布局

```
<div class="container">
  <div class="float"></div>
  文本文本文本
</div>
```

- Shape 共有两种属性：
 - shape-outside 让文本围在图形外，和 shape-margin 属性一起使用
 - shape-inside 把文本包装在形状的内部，和 shape-padding 一起使用
- 先给 .float 设置 border-radius:50%;
 - 给 .float 设置浮动, 然后通过样式, 控制 外部文本的环绕方式
- shape-outside: 决定 外部文字环绕方式

shape-outside属性值	描述
margin-box	以 margin 区域以外环绕(默认)
border-box	以 border 以外区域环绕
padding-box	以 padding 以外的区域环绕
content-box	以 内容 以外的区域进行环绕
polygon(x1 y1, x2 y2, x3 y3)	多边形， 首尾两点自动相连
circle(i [at x y])	半径长为 i 的圆形， 可以指定圆心位置 x y； 不指定位置默认容器中央为圆心
ellipse(x y [at x2 y2])	椭圆， x 轴半径为 x， y 轴半径为 y 的椭圆； 可以指定圆心位置 x, y； 不指定位置默认容器中央为圆心
inset(10px 20px round 5px)	距离外部盒子的上，下各 10px 距离外部盒子的左，右各 20px 还有半径为 5px 的圆角

- CSS shapes 通过参考盒来定义并且创建,这个盒子用来绘制元素上的形状。默认情况下,使用 margin box 盒模型作为参考,也可以使用 padding-box、border-box、content-box；
 - 简述: 默认使用 margin-box 为画板, margin-box 左上角为 0,0；
 - 图形盒子: 分别是 margin-box，border-box，padding-box 和 content-box。要来指定文字环绕的时候是依照哪个盒子的边缘来计算的。
 - 基本图形函数: 如 circle(), ellipse() 等
 - 图像类: URL 链接图片(要具有透明度的 png), 渐变图像, cross-fade(), element() 等

- 图像类需要服务器环境

- 格式: `shape-outside: circle(50px at 20px) padding-box;`
- `shape-image-threshold: .3;`
 - 决定什么透明度之内的值,可以环绕文字
 - `.5` `50%` 之内透明度的值可以环绕文字
- `clip-path`: 将当前容器, 剪切, 被裁去的部分将隐藏, 不会改变容器大小

取值	描述
<code>circle(i [at x y])</code>	半径长为 <code>i</code> 的圆形, 可以指定圆心位置 <code>x</code> , <code>y</code> ; 不指定位置默认容器中央为圆心
<code>polygon(x1 y1, x2 y2, x3 y3)</code>	多边形, 首尾两点自动相连
<code>ellipse(x y [at x2 y2])</code>	与 <code>shape-outline: ellipse()</code> 相同
<code>inset(10px 20px round 5px)</code>	距离外部盒子的上, 下各 <code>10px</code> 距离外部盒子的左, 右各 <code>20px</code> 还有半径为 <code>5px</code> 的圆角

- 通过以下设置 `shape-outside` 和 `clip-path` , 可以达到 文字环绕容器的效果

```
shape-outside: polygon(0 0,0 100px, 100px 100px);
clip-path: polygon(0 0,0 100px, 100px 100px);
```

- `shape-margin`: 调节容器 和 外部文字之间的间隙

4.CSS新特性之scrollbar (伪元素)

- CSS `scrollbar` 用于实现自定义滚动条样式。

伪元素	描述
<code>::-webkit-scrollbar</code>	定义滚动条宽度
<code>::-webkit-scrollbar-thumb</code>	定义滑块样式
<code>::-webkit-scrollbar-track</code>	定义滑块轨道部分样式

- 伪元素需要给父容器添加, `body` 高 `2000px` , 给 `html` 添加伪元素
- `::-webkit-scrollbar` : 定义滚动条宽度, 横向时使用 `height`

```
body::-webkit-scrollbar{
  width: 10px;
  /* height: 1px; 横向时使用height*/
}
```

- `::-webkit-scrollbar-thumb` : 定义滑块样式

```
body::-webkit-scrollbar-thumb{
  background: red;
  border-radius:100vw;
}
```

- `::-webkit-scrollbar-track` : 定义滑块轨道部分样式

```
body::-webkit-scrollbar-track{
  background: gray;
  box-shadow: inset -1px -1px 10px black;
}
```

- 还可以配合伪类使用

```
body::-webkit-scrollbar-track:hover{}
body::-webkit-scrollbar-track:active{}
```

- 滚动条还有其他的 不常用伪元素,可以查阅资料

5.CSS新特性 scrollsnap

- CSS *ScrollSnap*(CSS滚动捕捉), 允许你在用户完成滚动后 锁定特定的元素或位置。
- `scroll-snap-type` : 添加给父容器, 决定捕捉模式和捕捉轴, 父容器里面的内容需要溢出,产生滚动条才会生效

```
scroll-snap-type: x/y轴 mandatory/proximity;
/* 第一个参数: 捕捉的轴, 第二个参数: mandatory(精准捕捉)/proximity(非精准捕捉)*/
scroll-snap-type: x mandatory;
```

- `scroll-snap-align` : 添加给子元素, 决定子元素被捕捉后, 与父元素的对齐方式

```
scroll-snap-align: center;
/* 可选值三个: start center end 分别于父元素 起始处对齐, 居中对齐, 结束对齐 */
```

- `scroll-padding` : 添加给父元素, 决定 捕捉后距离父元素的 `padding`, 使其具有一定边界
 - 还有 `scroll-padding-left/top/right/bottom` 等
- `scroll-margin` : 添加给子元素, 决定 捕捉后距离父元素的 `margin`, 使其具有一定边界和 `scroll-padding` 差不多

6.css与js结合之 钟表

7.css与js结合之 折叠菜单