

学习资料参考

视频	文档
	w3school HTML 标签手册
	MDN HTML
尚硅谷前端入门html+css零基础教程	W3C HTML Standard
	w3school CSS 教程
	MDN CSS

PART1 Supplements (day1: 0%)

C/S & B/S 架构

C/S架构	B/S架构
Client-Server (客户端-服务器)	Browser-Server (浏览器-服务器)
需要安装；偶尔更新；不跨平台	无需安装；无需更新；可跨平台

- C/S 架构一般用于**大型专业、安全性要求较高的应用**
- 前端工程师主要开发基于 B/S 架构的**网页**，也涉及**微信小程序开发、客户端开发**（React Native 或 uni-app + Vue）、**服务器搭建**（NodeJS）、**数据可视化**（ECHARTS）

浏览器

1. 什么是浏览器内核？内核是浏览器的核心，用于**处理**浏览器所得到的各种**资源**。



2. 五大主流浏览器

浏览器	内核	问世时间
Chrome	Blink (before: webkit)	2008/9
Safari	webkit	2003/1
IE	Trident	1995/8
Firefox	Gecko	2002/9
Opera	Blink (before: Presto)	1995/4

- 五大浏览器、四大内核
- 其他浏览器在四大内核的基础上，添加精美 UI 以及实用功能

网页

1. 一个网站 = 一个或多个网页。
2. 网页的组成部分/网页标准：网页 = **结构** (HTML) + **表现** (CSS) + **行为** (JavaScript) 。

编程软件

- 编程软件：VSCode
- 相关扩展：中文语言包、vscode-icons、Live Server
 - vscode-icons 可以修改文件的图标主题
 - Live Server 可以便捷打开 html 文件，以更加贴近项目上线的方式打开（本地创建一个服务器，将当前文件夹下的内容放到服务器中）
 - 源代码出现改动后，浏览器内容可以**自动刷新**
 - 使用 VSCode 打开的必须是**文件夹**
 - 打开的网页必须是**标准的 HTML 结构**，否则无法自动刷新

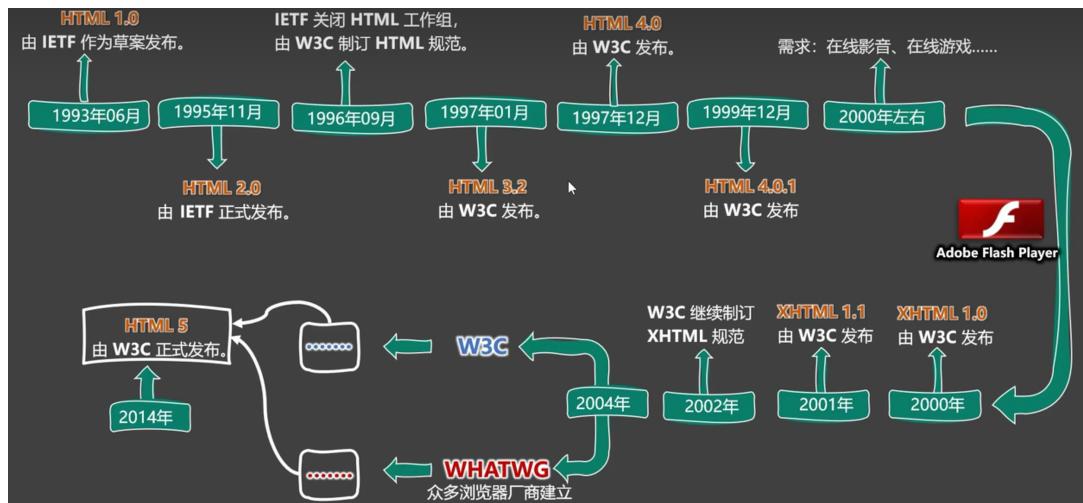
PART2 HTML4

初认识

1. HTML 全称为 HyperText Markup Language，翻译为**超文本标记语言**。

- 超文本：与普通文本相比，内容更加丰富，“超”指超链接
- 标记：将文本变为超文本的符号

2. 发展史



标签

1. **标签**：又称**元素**，是 HTML 的基本组成单位。
2. 标签的分类：**双标签**和**单标签**。



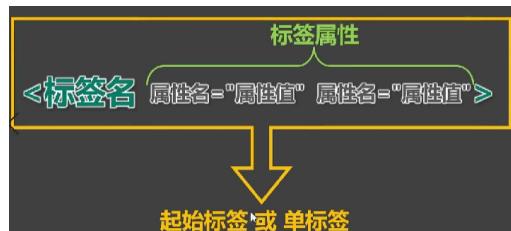
```
<marquee>悟已往之不谏，知来者之可追</marquee>
<input>
```

3. 标签之间的关系：并列关系、嵌套关系。

```
<marquee>
    悟已往之不谏，知来者之可追
    <input>
</marquee>
```

4. 注意：标签名不区分大小写，但小写更规范。

1. 标签属性：是用于给标签提供附加信息的键值对。



```
<marquee loop="1" bgcolor="orange">悟已往之不谏，知来者之可追</marquee>
<input type="password">
```

2. 注意

- 有些特殊的属性没有属性名，只有属性值

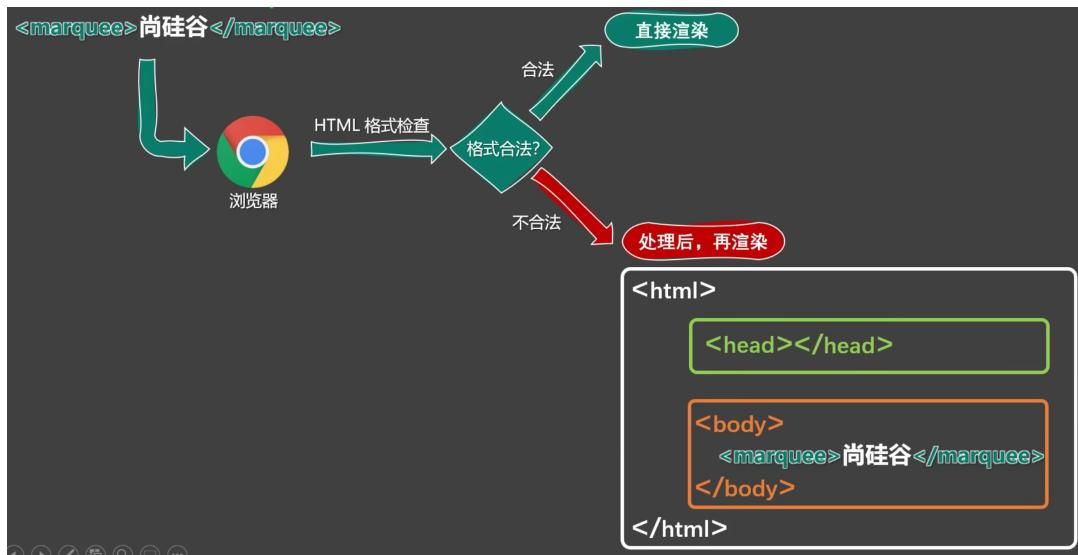
```
<input disabled>
```

- 不同的标签有不同的属性，也有一些通用属性
- 属性名与属性值都是 W3C 预先规定好的
- 属性名与属性值都不区分大小写，但推荐小写
- 属性值可以用双引号、单引号表示，也可以不写，但推荐双引号
- 标签中如果出现同名属性，以第一个为准，后写的会失效

```
<input type="text" type="password">
```

基本结构

1. 源代码是怎样渲染成网页的？浏览器先进行格式检查，合法则直接渲染，否则需要对源代码处理后再进行渲染。



2. 网页的基本结构

```

<html>
  <head>
    <title>网页标题</title>
  </head>
  <body>
    .....
  </body>
</html>

```

- `body` 标签中是要呈现在网页中的内容
- `head` 标签中的内容不会出现再网页中
- `head` 标签中的 `title` 标签用于指定网页的标题

3. 网页开发的两大功能（浏览器右键菜单栏）：检查、查看网页源代码

- 检查：经过浏览器处理后的源代码
- 查看网页源代码：程序员编写的源代码

注释

1. HTML 的注释以 `<!--` 开始，以 `-->` 结束。

```
<!-- 这是注释性文字 -->
```

2. 注意：注释不可以嵌套使用。

文档声明

1. 因为 HTML 有若干版本，因此为了渲染正确，使用**文档声明**告诉浏览器**当前网页的版本**。最新的，也是默认的网页版本是 **HTML5**，其文档声明为 `<!DOCTYPE html>`。
2. 注意：文档声明必须放在**网页的第一行**，在 `html` 标签的外侧。

```
<!DOCTYPE html>
<html>
  <head></head>
  <body></body>
</html>
```

字符编码

1. 计算机存储数据时进行**编码**, 读取数据时进行**解码**, 编码和解码的过程中都遵循一定的规范, 即**字符集**。
2. 数据操作需要遵守两项基本原则
 - 存储时, 必须采用**合适的字符集编码**。
 - 存储和读取数据时, 必须采用**相同的字符集**进行编码和解码。
3. 通过指定字符解码方式, 可以让浏览器在渲染 HTML 文件时不出错误。HTML 通过 `meta` 标签的 `charset` 属性指定字符编码, 一般使用 `UTF-8` 字符集。

```
<head>
  <meta charset="UTF-8" />
</head>
```

语言

1. 设置 HTML 的语言有助于 ①让浏览器显示对应的翻译提示 ②搜索引擎优化。
2. HTML 通过 `html` 标签中的 `lang` 属性设置**网页的语言**。

<code>lang</code> 的取值	语言
<code>zh-CN</code>	简体中文
<code>zh-TW</code>	台湾
<code>zh</code>	中文
<code>en-US</code>	美国英语
<code>en-GB</code>	英国英语
<code>en</code>	英语

```
<html lang="zh-CN">
```

3. `lang` 的参数取值规律为 `lang=语言-国家/地区`。

标准结构

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <title>网页标题</title>
  </head>
  <body>

  </body>
</html>
```

- VSCode 中，可以通过输入 `! + 回车` 快速生成 HTML 的标准结构，并且可以通过内置插件 `emmet` 自定义生成结构的属性
- 通过在存放代码的文件夹中存放一个名为 `favicon.ico` 的图片，可以配置网站图标

排版标签

标签	含义	单/双
<code>h1 ~ h6</code>	标题	双
<code>p</code>	段落	双
<code>div</code>	无含义，用于整体布局	双

- `h1` 不仅仅表示一级标题，还可以看作是重要性的表征，一般只写一个
- [1-排版标签.html](#)

语义化标签 (day2 | 22p: 9.1%)

- 语义化标签**：即用特定的标签，表达特定的含义。这要求我们在使用 HTML 的标签时，关注的应该是**标签的语义**，而不是标签的效果（这一点是由 CSS 来完成的）。如，`h1` 标签的语义为“网页的主要内容”，效果是“文字加粗加大”，我们更应该关注的是前者。
- 关注标签语义的优点
 - 代码结构清晰，可读性强
 - 有利于 SEO（搜索引擎优化）
 - 方便设备解析（如屏幕阅读器等）

块级元素与行内元素

块级元素	行内元素
独占一行	不独占一行
如所有的排版标签	如 <code>input</code> 、 <code>span</code> 等

- 块级元素中能写行内元素和块级元素 (ALMOST NOT ALL)**
 - `h1 - h6` 不能相互嵌套

- `p` 中不能写块级元素
- 行内元素中能写行内元素，但不能写块级元素

文本标签（常用）

标签	语义	单/双
<code>em</code>	需要着重阅读的内容	双
<code>strong</code>	十分重要的内容（比 <code>em</code> 更重要）	双
<code>span</code>	无语义，是用于包裹短语的通用容器	双

- 文本标签用于包括词汇、短语等内容
- 文本标签通常写在排版标签里边
- 排版标签关注于大段的文字，而文本标签关注词汇、短语等内容
- 文本标签通常都是行内元素
- 排版标签中的 `div` 和文本标签中的 `span` 类似，都没有语义，是更加通用的标签

文本标签（不常用）

标签	语义	单/双
<code>cite</code>	(书籍、歌曲、电影等的) 作品标题	双
<code>dfn</code>	特殊术语或专有名词	双
<code>del</code> / <code>ins</code>	删除的文本/插入的文本	双
<code>sub</code> / <code>sup</code>	下标文字/上标文字	双
<code>code</code>	代码片	双
<code>samp</code>	从正常上下文中，将某些内容提取出来（如，标识设备的输出）	双
<code>kdb</code>	键盘文本，表示文本是通过键盘输入的	双
<code>abbr</code>	缩写，常配合 <code>title</code> 属性使用	双
<code>bdo</code>	更改文本方向，常配合 <code>dir</code> 属性使用 (<code>dir=ltr/rtl</code>)	双
<code>var</code>	变量，可以与 <code>code</code> 标签一起使用（即代码中的变量）	双
<code>small</code>	附属细则（如，版权、法律文本）	双
<code>b</code>	摘要中的关键字或评论中的产品名称	双
<code>i</code>	人物的思想活动、所说的话等；现多用于呈现字体图标	双
<code>u</code>	表示与正常内容有反差的文本（如，错误的单词、不合适的描述等）	双
<code>q</code>	短引用	双
<code>blockquote</code>	长引用	双

标签	语义	单/双
address	地址信息	双

```
<abbr title="英雄联盟">LOL</abbr>
<bdo dir="rtl">你是年少的欢喜</bdo>
```

- 上述不常用的文本标签，编写代码时可以不使用
- `blockquote` 和 `address` 是块级元素，其他文本标签都是行内元素
- 语义感不强的标签，如：`small`、`b`、`u`、`q`、`blockquote`，很少使用
- 目前需要记住的重要的、语义感强的标签有：`h1-h6`、`p`、`div`、`em`、`strong`、`span`

图片标签

```
<img src="" alt="" width="" height="">
<!--
1. 标签名: img
2. 语义: 图片
3. 常用属性及其含义
  - src 图片路径
  - alt 图片描述
  - width 图片宽度, 单位是像素, 如 200px
  - height 图片高度, 单位是像素, 如 200px
4. 单标签
-->
```

- `img` 是一个新的元素分类，即既不属于块级元素，也不属于行内元素，可以暂且认为是行内元素。
- `alt` 属性的作用
 - (最主要) 搜索引擎可以通过 `alt` 属性内容，得知图片的内容
 - 当图片无法加载时，有些浏览器会显示 `alt` 属性的值
 - 盲人阅读器等设备会朗读 `alt` 属性的值

相对路径与绝对路径

1. 相对路径：以当前位置为参考点去建立路径。

层级	相对路径
当前路径（目录）	./
上一级路径（目录）	../
上两级路径（目录）	../../
下一级路径（目录）	./child/
下两级路径（目录）	./child/2-levels-down/

- 相对路径中的表示当前路径的 `./` 可以省略不写

- 因为相对路径依赖的是当前位置，如果后期调整了文件位置，那么代码中文件的相对路径也要修改
- [2-相对路径.html](#)

2. **绝对路径**: 以根位置 (如盘符) 为参考点去建立路径。

- **本地绝对路径**: 如 `C:\Users\15787\Pictures\GenshinImpactCloudGame\1292433.jpg`，很少使用
 - 一旦更换设备，路径处理起来比较麻烦，因此很少使用
- **网络绝对路径**: 如 `https://www.miyoushe.com/_nuxt/img/miHoYo_Game.2457753.png`
 - 如果服务器开启了防盗链，则会造成图片引入失败

常见图片格式

jpg	png	bmp	gif	webp	base64
.jpg 或 .jpeg，有损压缩，丢弃肉眼不易观察出来的细节	.png，无损压缩，能够更高质量的保存图片	.bmp，不进行压缩，最大程度保留图片的更多细节	.gif，仅支持 256 种颜色，色彩呈现不是很完整	.webp，由谷歌推出，专用来在网页中呈现图片	本质是一串特殊的文本，需要通过浏览器打开
支持的颜色丰富	支持的颜色丰富	支持的颜色丰富，保留的细节更多	支持的颜色较少	具备前四种格式的优点	原理是把图片进行 base64 编码，形成一串文本
占用的空间较小	占用空间略大	占用空间极大		但是与浏览器的兼容性不好	可作为 img 标签的 src 属性的值
不支持透明背景	支持透明背景	不支持透明背景	支持简单透明背景	使用务必要解决兼容性问题	
不支持动态图	不支持动态图	不支持动态图	支持动态图		
适用于对图片细节没有极高要求的场，如：网站的产品宣传图等。	适用于①想让图片有透明背景或②想要更高质量地呈现图片，如：公司 logo 图、重要配图等。	适用于对图片细节要求极高的场景，如：大型游戏中的图片等。	适用于网页中的动态图片	适用于网页中的各种图片	适用于①一些较小的图片②需要和网页一起加载的图片

超链接

```
<a href="" target=""></a>
<!--
1. 标签名: a
2. 语义: 超链接
3. 常用属性及其含义
    - href 要跳转到的具体位置
    - target 跳转时应该如何打开页面, 取值为,
        -- _self 在本页签中打开
        -- _blank 在新页签中打开
4. 双标签
-->
```

3-超链接.html

1. 跳转到**页面**: 可以使用 `a` 实现**跳转到其他网页**和**跳转到本地网页**两个功能。

```
<!-- 超链接-跳转页面 -->
<!-- 跳转到其他网页 -->
<a href="https://ys.mihoyo.com/cloud/?utm_source=default#/" target="_blank">
    云原神, 启动! </a>
<!-- 跳转到本地网页 -->
<a href=". ./two-levels-up/parent/current/2-相对路径.html" target="_self">托马照片合集, 点击即看! </a>
```

- 代码中的**多个空格和多个回车**, 都会被浏览器解析为**一个空格**
- 虽然 `a` 是行内元素, 但是 `a` 可以包裹除自身外的任何元素

2. 跳转到**文件**: 可以使用 `a` 实现在浏览器中**打开或下载**文件。

```
<!-- 超链接-跳转文件 -->
<!-- 跳转到浏览器可以打开的文件(图片、视频、pdf等) -->
<a href=". ./two-levels-up//托马生日图.jpg">点击查看托马生日照</a>
<!-- 跳转到浏览器不可以打开的文件(zip等), 会触发自动下载 -->
<a href=". ./托马图片合集.7z">点击获取托马照片合集</a>
<!-- 通过 download 属性, 强制触发自动下载, 同时可以通过指定取值设定默认下载文件的名称 -->
<a href=". ./two-levels-up/托马生日图.jpg" download="托马生日照片">点击下载托马生日照</a>
```

- 浏览器**可以直接打开**的文件, 可以通过超链接跳转直接**查看**
- 浏览器**不可以直接打开**的文件, 通过超链接跳转会直接**触发下载**, 下载的文件的默认名与 `href` 中的默认名一样
- 如果想**强制触发下载**要跳转的文件, 可以配合使用 `download` 属性, 属性值为下载文件的名称 (也可以直接通过 `点击下载托马生日照`, 省略 `download` 的值后, 下载文件的名称默认为 `href` 中的文件名称)
- 可以跳转到本地文件, 也可以跳转到在线文件

3. 跳转到**锚点**: 锚点是网页中的一个**标记点**, 当通过某种方式设置好锚点后, 可以使用 `a` 跳转到**本页面的锚点或其他页面的锚点**。

```
<!-- 超链接-跳转锚点 -->
```

```

<!-- 1. 设置锚点 -->
<!-- 通过 a 标签的 name 属性设置锚点 -->
<a name="anchor_a">HHHHHHHHHH</a>
<!-- 任何标签和 id 属性配合设置锚点 -->
<h2 id="anchor_h2">这里是一个标定锚点</h2>
<!-- 2. 跳转锚点 -->
<!-- 通过 # + 锚点名/id 实现跳转到锚点 -->
<!-- 通过在超链接的 href 中添加 #, 浏览器明白此时的超链接是想要跳转到一个锚点 -->
<a href="#anchor_a">调转到HHHHHHHHHH</a>
<a href="#anchor_h2">跳转到“这里是一个标定锚点”</a>
<a href=". ./two-levels-up/parent/current/2-相对路径.html#tomma">跳转到“托马比拼图”</a>
<!-- 3. 其他跳转行为 -->
<!-- 通过 href="#" 实现跳转到网页顶部 -->
<a href="#">回到顶部</a>
<!-- 通过 href="" 刷新页面 -->
<a href="">刷新</a>
<!-- 4. JS 弹窗行为 -->
<a href="javascript:alert('Hello world!');">欢迎世界</a>

```

- **设置锚点**有两种方式：① a 标签配合 name 属性 ② 任意标签配合 id 属性
- 具有 href 属性的 a 标签是**超链接**；具有 name 属性的 a 标签是**锚点**
- name 和 id 都是**区分大小写**的，其中 id 最好不要是数字开头
- **跳转锚点**分为两种情况：①跳转到本页面中的锚点，此时 href 取值为 #name 值 或 #id 值 ② 跳转到其他页面中的锚点，此时 href 取值为 其他页面的路径#name 值 或 其他页面的路径#id 值
- 网页中的导航栏可以通过锚点来实现
- **回到当前网页顶部和刷新当前网页**可以通过 和 来实现
- **执行一段 js 代码**可以通过 来实现

4. 唤起指定应用：可以使用 a 标签唤起设备应用程序。

```

<!-- 超链接-唤起指定应用 -->
<!-- 唤起电话拨号 -->
<a href="tel:10086">给 10086 拨打电话</a>
<!-- 唤起短信发送短信 -->
<a href="sms:10086">给 10086 发送短信</a>
<!-- 唤起邮箱发送邮件 -->
<a href="mailto:10000@qq.com">给 10000@qq.com 发送邮件</a>

```

5. 什么是**超文本**？超文本是一种组织信息的方式，通过超链接将不同空间的文字、图片等各种信息组织在一起，能从当前阅读的内容，跳转到超链接所指向的内容（页面、文件、锚点、应用）。

列表 (day3 | 35p: 14.5%)

4-列表.html

1. **有序列表**：有顺序或侧重顺序的列表。

```
<h2>红烧肉的做法如下</h2>
<ol>
    <li>准备材料，五花肉洗净，切成麻将块大小；</li>
    <li>锅烧热放火麻油，爆香姜片大蒜花椒八角；</li>
    <li>倒入五花肉翻炒至两面微焦，加入料酒或者白酒、酱油、冰糖；</li>
    <li>转入砂锅加适量开水，慢火焖一个小时，要注意经常翻身，一方面均匀上色，另一方面避免猪皮粘锅。出锅之前撒点胡椒粉和盐就可以了。</li>
</ol>
```

- `` 和 `` 配合使用构成有序列表
- ol=ordered list, li=list item, `` 表示有序列表, `` 表示列表项
- `` 中只能放 ``

2. 无序列表：无顺序或不侧重顺序的列表。

```
<h2>西安旅游推荐景点如下</h2>
<ul>
    <li>大雁塔</li>
    <li>华清池</li>
    <li>不夜城</li>
    <li>钟楼</li>
</ul>
```

- `` 和 `` 配合使用构成无序列表
- ul=unordered list, li=list item, `` 表示无序列表, `` 表示列表项
- `` 中只能放 ``

3. 自定义列表：包含术语名称以及术语描述的列表。

```
<h2>前端术语解释</h2>
<dl>
    <dt>HTML</dt>
    <dd>HTML 的全称是 HyperText Markup Language</dd>
    <dd>HTML 中最重要的标签之一是 a</dd>

    <dt>CSS</dt>
    <dd>CSS 的全称是 Cascading Style Sheets</dd>
    <dd>HTML 相当于网页的骨架, CSS 则对网页进行修饰</dd>
</dl>
```

- `<dl>`、`<dt>` 和 `<dd>` 配合使用构成自定义列表
- dl=definition list, dt=definition term, dd=definition description, `<dl>` 表示自定义列表, `<dt>` 表示术语名称, `<dd>` 表示术语描述
- 一个 `<dt>` 可以搭配多个 `<dd>` 使用

4. 列表嵌套：列表中的某项（item），又包含了一个列表。

```
<h2>我想去的几个城市</h2>
<ul>
    <li>成都</li>
    <li>重庆</li>
    <li>
```

```

<span>北京</span>
<ul>
    <li>天安门</li>
    <li>圆明园</li>
    <li>颐和园</li>
    <li>故宫</li>
</ul>
</li>
<li>上海</li>
</ul>

```

- `` 标签最好写在 `` 或 `` 中，不要单独使用

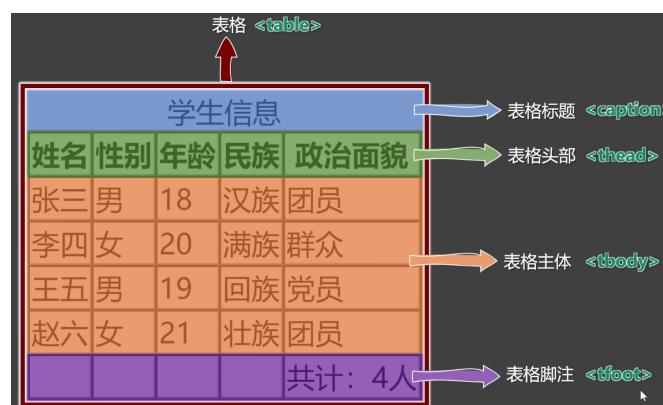
表格

[5-表格.html](#)

基本结构

一个完整的表格由**表格标题、表格头部、表格主体、表格脚注**四部分组成。

标签	语义
<code>table</code>	表格
<code>caption</code>	表格标题
<code>thead</code>	表格头部
<code>tbody</code>	表格主体
<code>tfoot</code>	表格脚注
<code>tr</code>	行
<code>th</code>	(表格头部中的) 单元格
<code>td</code>	(表格主体和表格脚注中的) 单元格



- 表格: `<table></table>`

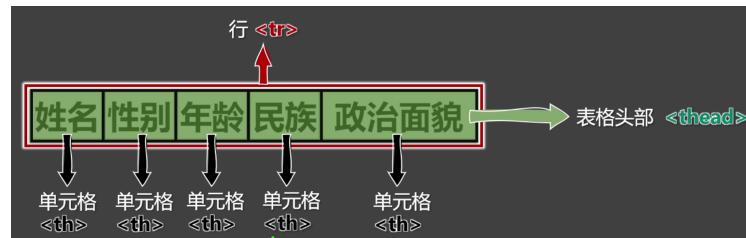
```
<table>
  <caption></caption>
  <thead></thead>
  <tbody></tbody>
  <tfoot></tfoot>
</table>
```

- 表格标题: `<caption></caption>`

```
<caption>学生信息</caption>
```

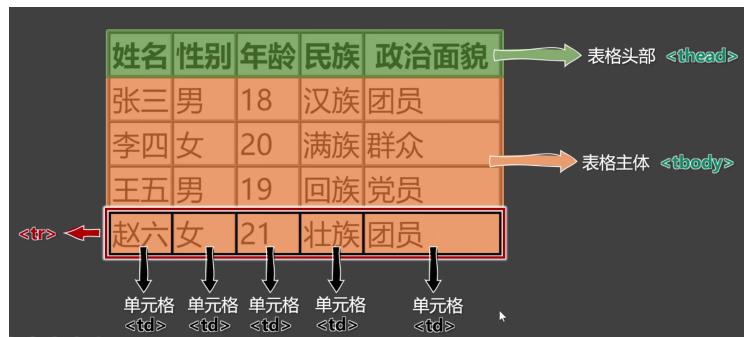
- 表格头部: `<thead></thead>`

```
<thead>
  <tr>
    <th>姓名</th>
    <th>性别</th>
    <th>年龄</th>
    <th>民族</th>
    <th>政治面貌</th>
  </tr>
</thead>
```



- 表格主体: `<tbody></tbody>`

```
<tbody>
  <tr>
    <th>张三</th>
    <th>男</th>
    <th>18</th>
    <th>汉族</th>
    <th>团员</th>
  </tr>
  <tr>
    <th>李四</th>
    <th>女</th>
    <th>20</th>
    <th>满族</th>
    <th>群众</th>
  </tr>
</tbody>
```



- 表格脚注: <tfoot></tfoot>

常用属性

标签	语义	常用属性	单/ 双
table	表格	width 表格宽度 height 表格最小高度 border 表格边框宽度 cellspacing 单元格之间的间距	双
thead	表格头部	height 表格头部高度 align 单元格的水平对齐方式, 可选值为: <code>left</code> 、 <code>center</code> 、 <code>right</code> valign 单元格的垂直对齐方式, 可选值为: <code>top</code> 、 <code>middle</code> 、 <code>bottom</code>	双
tbody	表格主体	同 <code>thead</code>	双
tfoot	表格脚注	同 <code>thead</code>	双
tr	行	同 <code>thead</code>	双
th	表头单元格	width 单元格宽度 height 单元格高度 align 单元格水平对齐方式, 取值同 <code>thead</code> valign 单元格垂直对齐方式, 取值同 <code>thead</code> rowspan 单元格要跨的行数 colspan 单元格要跨的列数	双
td	普通单元格	同 <code>th</code>	双

- `table` 中的 `height` 设置的是**表格的最小高度**, 表格最终高度可能比设置的值大
- `table` 中的 `border` 可以控制**表格边框的宽度**, 但是无法控制单元格边框的宽度
- `th` 和 `td` 设置好宽度后, 其所在列的宽度就确定了
- `th` 和 `td` 设置好高度后, 其所在行的高度就确定了

常用标签 (补充)

标签	语义	单/双
<code>br</code>	换行	单
<code>hr</code>	分隔	单
<code>pre</code>	按原文显示	双

- `br` 和 `hr` 标签，应关注其语义，而不是其带来的效果

表单

[6-表单.html](#)

基本结构

标签	语义	属性	单/ 双
<code>form</code>	表单	<code>action</code> 指定表单的提交地址 <code>target</code> 控制表单提交后，如何跳转，可取值： <code>_self</code> ， <code>blank</code> <code>method</code> 控制表单提交的请求方式，可选值： <code>get</code> 、 <code>post</code>	双
<code>input</code>	输入框	<code>type</code> 设置表单控件的类型，可取值： <code>text</code> 、 <code>password</code> 、 <code>radio</code> 、 <code>checkbox</code> 、 <code>hidden</code> 、 <code>submit</code> 、 <code>reset</code> 、 <code>button</code> 、... <code>name</code> 指定提交数据的名称 <code>value</code> 对于输入框，指定默认输入值的值；对于单选和复选框，是实际提交的数据；对于按钮，是显示在按钮上的文字 <code>maxlength</code> 对于输入框，设置最大可输入长度 <code>checked</code> 用于单选和复选框的默认选中	单
<code>textarea</code>	文本域	<code>name</code> 数据名称 <code>rows</code> 默认显示的行数 <code>cols</code> 默认显示的列数 <code>disabled</code> 设置表单控件不可用	双
<code>select</code>	下拉框	<code>name</code> 数据名称 <code>disabled</code> 设置整个下拉框不可用	双
<code>option</code>	下拉框的选项	<code>value</code> 该选项提交的数据（如不指定，将标签中的内容作为提交数据） <code>selected</code> 默认选中 <code>disabled</code> 设置下拉选项不可用	双
<code>button</code>	按钮	<code>type</code> 设置按钮的类型，可选值： <code>submit</code> （默认）、 <code>reset</code> 、 <code>button</code>	双

标签	语义	属性	单/ 双
label	与表单控件做关联	for 要关联的表单控件的 id	双
fieldset	表单控件分组		双
legend	表单控件分组名称		双

```
<!-- 以下代码可以实现：点击对应按钮后，到百度或京东搜索文本框中写入的内容 -->
<form action="https://www.baidu.com/s" target="_blank">
    <input type="text" name="wd">
    <button>通过百度搜索</button>
</form>
<form action="https://search.jd.com/search" target="_self">
    <input type="text" name="keyword">
    <button>通过京东搜索</button>
</form>
```

- 表单的提交地址（form-target）需要与后端人员沟通后确定，例如百度搜索的提交地址是 `https://www.baidu.com/s`，而京东搜索的提交地址为 `https://search.jd.com/search`
- 表单的提交方式（form-method）会在 Ajax 的学习中涉及
- 提交数据的名字（input-name）需要与后端人员沟通后确定

常用控件

控件	语义	相关属性	单 双
<code><input type="text"></code>	文本输入框	<code>name</code> 数据名称 <code>value</code> 输入框的默认值 <code>maxlength</code> 输入框的最大可输入长度	单
<code><input type="password"></code>	密码输入框	<code>name</code> 数据名称 <code>value</code> 输入框的默认值 <code>maxlength</code> 输入框的最大可输入长度	单
<code><input type="radio"></code>	单选框	<code>name</code> 数据名称 <code>value</code> 提交的数据值 <code>checked</code> 默认选中该单选框	单
<code><input type="checkbox"></code>	复选框	<code>name</code> 数据名称 <code>value</code> 提交的数据值 <code>checked</code> 默认选中该复选框	单

控件	语义	相关属性	单 双
<code><input type="hidden"></code>	隐藏域	<code>name</code> 数据名称 <code>value</code> 提交的数据值	单
<code><input type="submit" value="提交"></code>	提交按钮	<code>value</code> 指定按钮的文字	单
<code><button type="submit">提交</button></code>	提交按钮		双
<code><input type="reset" value="重置"></code>	重置按钮	<code>value</code> 指定按钮的文字	单
<code><button type="reset">重置</button></code>	重置按钮		双
<code><input type="button" value="norm"></code>	普通按钮	<code>value</code> 指定按钮的文字	单
<code><button type="button">norm</button></code>	普通按钮		双
<code><textarea name="msg"></textarea></code>	文本域	<code>name</code> 数据名称 <code>rows</code> 默认显示的行数 <code>cols</code> 默认显示的列数	双
<code><select name="from"></select></code>	下拉框	<code>name</code> 数据名称	双
<code><option value="bj">北京</option></code>	下拉框的项	<code>value</code> 提交的数据值 <code>selected</code> 默认选中该选项	双

- 密码输入框 (`<input type="password">`) 一般不使用 `value` 属性，无意义
- 多个单选框 (`<input type="radio">`) 的 `name` 属性的取值必须相同，才能实现单选功能
- 单选框 (`<input type="radio">`) 的 `checked` 属性没有值，仅仅表明有该属性的单选框被选中
- 隐藏域 (`<input type="hidden">`) 是一个用户不可见的区域，其作用是提交表单时，携带一些固定的数据
- 除了 `name` 属性以外，必须指定单选框和复选框的 `value` 属性的值
- 下拉框 (`<select name="from"></select>`) 中只能写 `option` 标签
- 下拉框的项 (`<option value="bj">北京</option>`) 最好设置 `value` 值，否则提交的数据就是 `option` 标签中的值
- 下拉框的项 (`<option value="bj">北京</option>`) 的 `selected` 属性没有值，仅仅表明有该属性的项目被选中

```

<h2>Person Information Sheet</h2>
<form action="https://search.jd.com/search">
  Account: <input type="text" name="acct" value="admin" maxlength="10"> <br>
  Password: <input type="password" name="pwd" maxlength="8"> <br>
  Gender:

```

```

<input type="radio" name="gender" value="male" checked>Male
<input type="radio" name="gender" value="female">Female <br>
Language:
<input type="checkbox" name="language" value="c">C
<input type="checkbox" name="language" value="java">Java
<input type="checkbox" name="language" value="python" checked>Python <br>
<input type="hidden" name="from" value="jd">
Preferred Salary:
<select name="salary">
    <option value="level-1">10k+</option>
    <option value="level-2">13k+</option>
    <option value="level-3" selected>15k+</option>
    <option value="level-4">18k+</option>
</select> <br>
Self-Assessment:
<textarea name="msg" cols="12" rows="3">Make a assessment of yourself here</textarea> <br>
    <button type="submit">confirm</button>
    <button type="reset">reset</button>
    <button type="button">verify</button>
</form>

```

禁用控件

给表单控件的标签添加 `disabled` 属性可以禁用该控件，禁用后用户将无法在网页中对其进行操作。

label 标签（控件关联）

1. `label` 标签可以将文字（或其他内容）与表单控件相关联。关联之后，点击文字，与之对应的表单控件就会获取焦点。
2. 关联方式
 - 给表单控件设置 `id` 属性，并且让 `label` 标签的 `for` 属性的值等于表单控件的 `id` 属性的值

```

<label for="act">
    Account:
</label>
<input id="act" type="text" name="acct" value="admin" maxlength="10">
<br>

```

- 将表单控件和文字（或其他内容）整体放在 `label` 标签里边

```

<label>
    Password: <input type="password" name="pwd" maxlength="8">
</label> <br>

```

fieldset/legend 标签（控件分组）

- `fieldset` 标签为表单控件分组，`legend` 标签设置分组的标题

```

<h2>Person Information Sheet</h2>
<form action="https://search.jd.com/search">

```

```

<fieldset>
    <legend>Majority</legend>
    <label for="act">
        Account:
    </label>
    <input id="act" type="text" name="acct" value="admin" maxlength="10">
<br>
    <label>
        Password: <input type="password" name="pwd" maxlength="8">
    </label> <br>
    Gender:
    <input type="radio" name="gender" value="male" checked>Male
    <input type="radio" name="gender" value="female">Female <br>
    Language:
    <input type="checkbox" name="language" value="c">C
    <input type="checkbox" name="language" value="java">Java
    <input type="checkbox" name="language" value="python" checked>Python <br>
    <input type="hidden" name="from" value="jd">
</fieldset>
<fieldset>
    <legend>Supplement</legend>
    Preferred Salary:
    <select name="salary">
        <option value="level-1">10k+</option>
        <option value="level-2">13k+</option>
        <option value="level-3" selected>15k+</option>
        <option value="level-4">18k+</option>
    </select> <br>
    Self-Assessment:
    <textarea name="msg" cols="12" rows="3">Make a assessment of yourself here</textarea> <br>
</fieldset>

    <button type="submit">confirm</button>
    <button type="reset">reset</button>
    <button type="button">verify</button>
</form>

```

框架标签

[7-框架标签.html](#)

```

<iframe src="" name="" width="" height="" frameborder=""></iframe>
<!--
1. 标签名: iframe
2. 语义: 框架
3. 常用属性及其含义
    - name 框架名字, 可以与 target 属性配合使用
    - width 框架的宽度
    - height 框架的高度
    - frameborder 是否显示边框, 可取值: 0 或 1
4. 双标签
5. 应用
    - 在网页中嵌入一个普通网页/广告网页
-->

```

- 在网页中嵌入一个浏览器可以打开的文件（图片、视频、pdf 等）
- 将 `iframe` 的 `name` 与超链接或表单的 `target` 配合，实现：超链接点击或表单提交后跳转到网页中嵌入的 `iframe` 框架中

-->

```

<h2>通过 iframe 嵌入一个普通网页</h2>
<iframe src="https://www.bilibili.com" frameborder="1" width="600" height="300">
</iframe>
<br>

<h2>通过 iframe 嵌入一个广告网页</h2>
<iframe src="https://pic3.zhimg.com/v2-8065c386354970c985325383711b337f_720w.webp?source=d6434cab" frameborder="1"
width="600" height="300"></iframe>
<br>

<h2>通过 iframe 嵌入浏览器可以打开的文件（图片、视频、pdf 等）</h2>
<iframe src="./two-levels-up/托马生日图.jpg" frameborder="1" width="600"
height="300"></iframe>

<h2>通过 iframe 的 name 属性与表单或超链接的 target 属性关联，可以让超链接（或表单提交后）要
跳转的页面在 iframe 框架中打开</h2>
<a href="https://www.bilibili.com" target="container">点击访问哔哩哔哩</a>
<a href="https://www.taobao.com" target="container">点击访问淘宝</a> <br>
<form action="https://search.jd.com/search" target="container">
  <label for="txt">
    在京东商城搜索商品
  </label>
  <input type="text" name="keyword" id="txt">
  <input type="submit" value="搜索">
</form>
<iframe name="container" frameborder="1" width="600" height="300"></iframe>

```

字符实体

1. **HTML 实体**：在 HTML 中，某些字符是预留的，如 `<`、`>`，如果想要**正确显示预留字符**，则必须在 HTML 源代码中使用**字符实体** (character entities) 。
2. 字符实体的形式：`&实体名称；` 或 `&#实体编号；`
3. 常见的字符实体（详细的字符实体的名称或编号见 [HTML Standard](#)）

显示结果	描述	实体名称	实体编号
	空格	<code>&nbsp;</code>	<code>&#160;</code>
<code><</code>	小于号	<code>&lt;</code>	<code>&#60;</code>
<code>></code>	大于号	<code>&gt;</code>	<code>&#62;</code>
<code>&</code>	和号	<code>&amp;</code>	<code>&#38;</code>
<code>"</code>	引号	<code>&quot;</code>	<code>&#34;</code>
<code>'</code>	撇号	<code>&apos;</code> (IE不支持)	<code>&#39;</code>

显示结果	描述	实体名称	实体编号
¢	分 (cent)	¢	¢
£	镑 (pound)	£	£
¥	元 (yen)	¥	¥
€	欧元 (euro)	€	€
§	小节	§	§
©	版权 (copyright)	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

全局属性

全局属性是所有 HTML 元素共有的属性，它们可以用于所有元素，即使属性可能对某些元素不起作用。
(关于全局属性的详细信息见 [全局属性 - HTML \(超文本标记语言\) | MDN](#))

全局属性	含义
<code>id</code>	标签的唯一标识
<code>class</code>	标签的类名
<code>style</code>	内联样式，直接给标签指定 CSS 样式
<code>dir</code>	指定标签中文本方向，可取值 <code>ltr</code> 、 <code>rtl</code>
<code>title</code>	给标签设置一个文字提示，多用于超链接和图片
<code>lang</code>	指定标签内容的语言

meta 标签

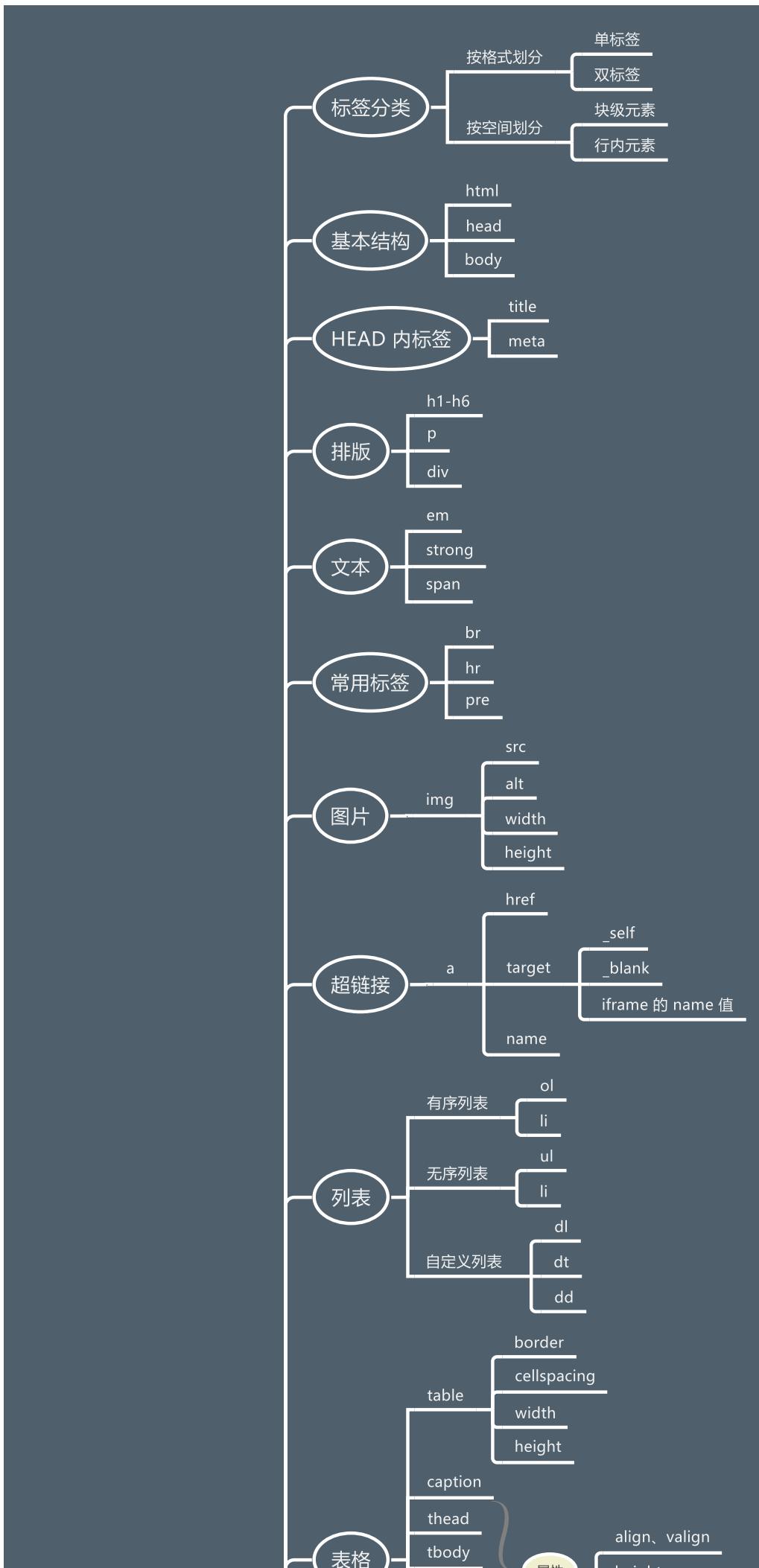
`<meta>` 标签用于配置网页元信息（基本信息）。

代码	含义
<code><meta charset="utf-8"></code>	配置字符编码
<code><meta http-equiv="X-UA-Compatible" content="IE=edge"></code>	针对 IE 浏览器的兼容性配置
<code><meta name="viewport" content="width=device-width, initial-scale=1.0"></code>	针对移动端的配置

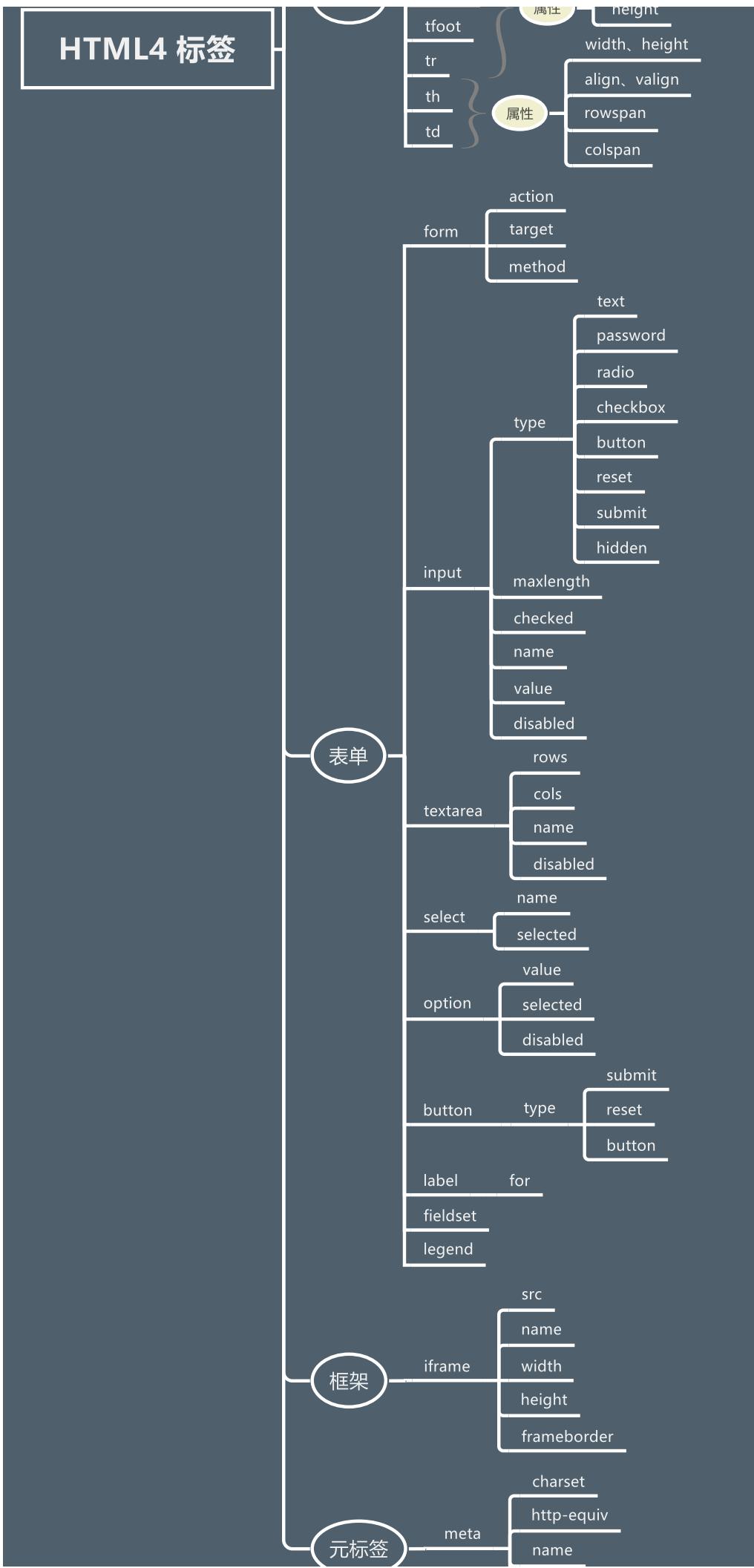
代码	含义
<code><meta name="keywords" content="8-12个以英文逗号隔开的单词/词语"></code>	配置网页关键字
<code><meta name="description" content="80字以内的一段话，与网站内容相关"></code>	配置网页描述性信息
<code><meta name="robots" content="针对不同情况，有不同的特定取值"></code>	针对搜索引擎爬虫配置
<code><meta name="author" content="tony"></code>	配置网页作者
<code><meta name="generator" content="vs Code"></code>	配置网页生成工具
<code><meta name="copyright" content="2023-2027©所有"></code>	配置版权信息
<code><meta http-equiv="refresh" content="10;url=https://www.baidu.com"></code>	配置网页自动刷新

SUMMARY of HTML4

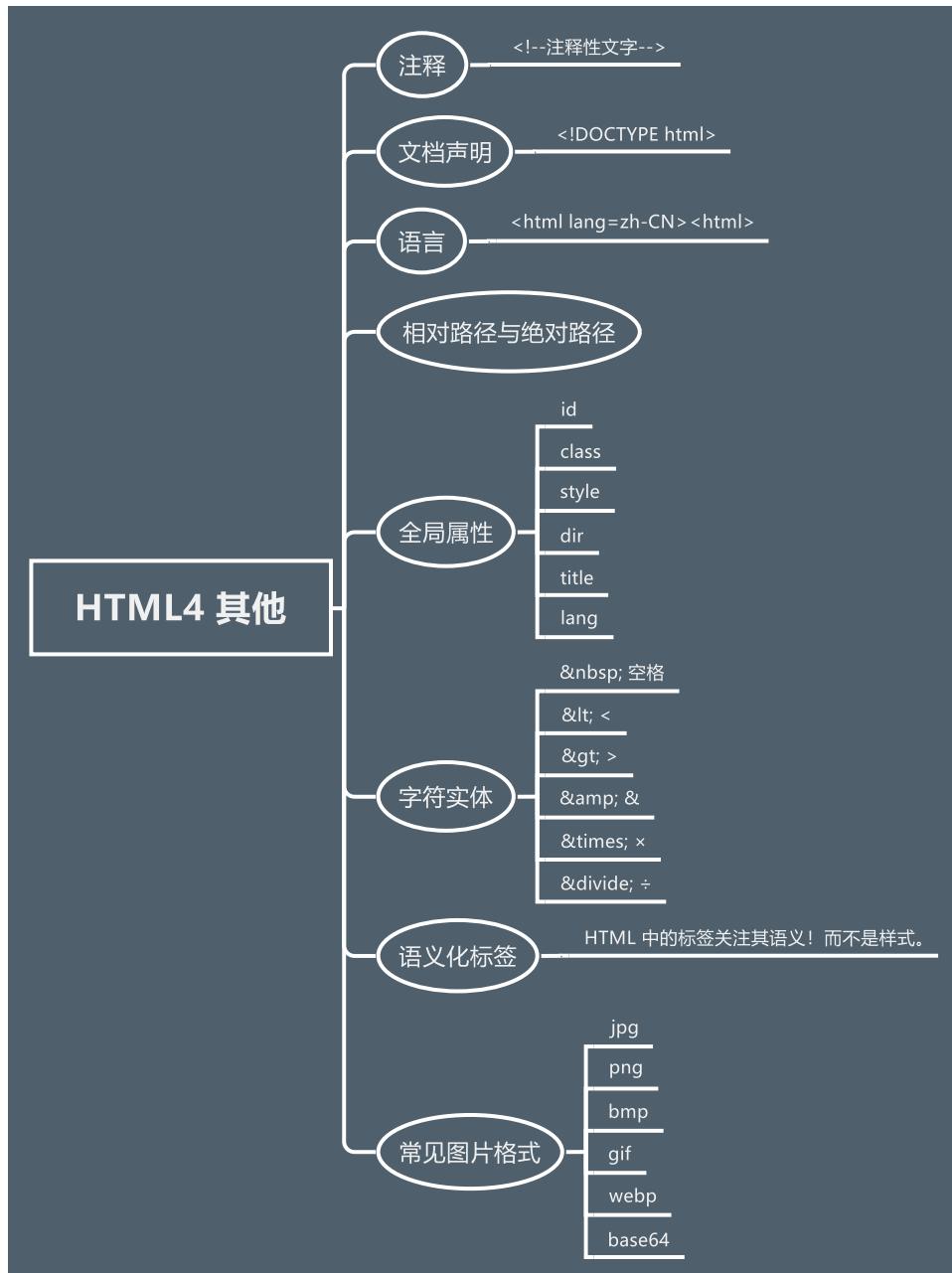
HTML4 标签



HTML4 标签



HTML4 其他



PART3 CSS2

初认识

1. CSS 全称为 Cascading Style Sheets, 翻译为层叠样式表。

- 层叠: CSS 给 HTML 设置样式像化妆一样, 是一层一层的
- 样式: 文字大小、颜色、元素宽高等
- 表: 列表

2. HTML 负责结构, CSS 添加样式, 二者实现了结构与样式的分离。

编写位置

行内样式

1. 行内样式：又称为**内联样式**，是写在标签的 `style` 属性中的样式表。

```
<h1 style="color: red; font-size: 60px">悟已往之不谏，知来者之可追</h1>
```

- `style` 属性的值要符合 **CSS 语法规规范**，即 `名:值;` 的形式
- 行内样式表只能控制当前标签的样式，对其他标签无效

2. 不足之处

- 书写繁琐、样式**不能复用**
- **无法体现结构与样式分离的思想**
- 只有对当前元素添加简单样式时，才偶尔使用，不推荐大量使用

内部样式

1. 内部样式：是写在 HTML 页面内部，将所有 CSS 代码提取出来，单独放在 `style` 标签中的样式表。

```
<head>
  <style>
    h1 {
      color: red;
      font-size: 40px;
    }
  </style>
</head>
<body>
  <h1 style="color: red; font-size: 60px">悟已往之不谏，知来者之可追</h1>
</body>
```

- `style` 标签理论上可以放在 HTML 文档的任何地方，但一般都放在 `head` 标签中

2. 优点与不足之处

- 优点：样式**可以复用**、代码结构清晰
- 不足之处
 - **没有实现结构与样式完全分离**
 - 多个 HTML 页面无法服用样式

外部样式

1. 外部样式：单独写在 `.css` 文件中，随后在 HTML 文件中通过 `link` 标签引入使用的样式表。

```
/* CSS 文件(xxx.css) */
h1 {
  color: red;
  font-size: 40px;
}
```

```

<!-- HTML 文件 -->
<head>
    <link rel="stylesheet" href="./xxx.css">
</head>
<body>
    <h1 style="color: red; font-size: 60px">悟已往之不谏，知来者之可追</h1>
</body>

```

- `link` 标签要写在 `head` 标签中
- `<link rel="" href="">`
 - `rel` 是 `relation` 的缩写，说明引入的文档与当前文档之间的关系
 - `href` 指定要引入的文档的位置

2. 优点

- 样式可以复用、结构清晰
- 可触发浏览器的缓存机制，提高访问速度
- 实现了结构与样式的完全分离
- 实际开发中，几乎都使用外部样式，是最推荐的使用方式

样式表优先级

1. 优先级规则：行内样式 > 内部样式 = 外部样式

- 内部样式、外部样式优先级相同，并且后来的样式会覆盖掉前面的同名样式

```

/* CSS 文件(xxx.css) */
h1 {
    color: red;
    font-size: 60px;
}

```

```

<!-- HTML 文件 -->
<head>
    <link rel="stylesheet" href="./xxx.css">
    <style>
        h1 {
            color: green;
        }
    </style>
</head>
<body>
    <h1 style="color: red; font-size: 60px">悟已往之不谏，知来者之可追</h1>
</body>
<!-- 网页上的文字样式应该是 green 60px -->

```

- 同一个样式表中，后来的样式也会覆盖掉前面的同名样式

```

<h1 style="color: red; font-size: 60px; color: green">悟已往之不谏，知来者之
可追</h1>
<!-- 网页上的文字样式应该是 green 60px -->

```

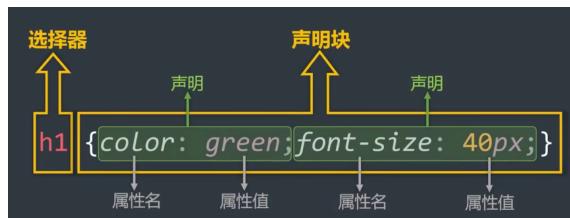
2. 三种样式表的对比

	行内样式	内部样式	外部样式
优点	优先级最高	样式可复用 代码结构清晰	样式可多页面复用 代码结构清晰 可触发浏览器缓存机制 结构与样式彻底分离
缺点	结构与样式未分离 代码结构混乱 样式不能复用	结构与样式未彻底分离 样式不能多页面复用	需要引入才能使用
使用频率	使用频率很低	一般	最高
作用范围	当前标签	当前页面	多个页面

语法规规范

1. CSS 样式由两部分组成：选择器和声明块

- **选择器**：负责找到要添加样式的元素
- **声明块**：负责设置具体的样式，由一个或多个**声明**组成（声明的格式为：`属性名: 属性值;`）



2. 注意

- 声明块中的最后一个声明的分号可以省略，但推荐写上
- 选择器与声明块之间，属性名与属性值之间，均有一个空格，可以省略，但推荐写上

注释

CSS 的注释以 `/*` 开始，以 `*/` 结束。

```
/* 这是注释性文字 */
```

代码风格

- **展开风格**：开发时推荐使用，便于维护和调试

```
h1 {  
    color: red;  
    font-size: 40px;  
}
```

- **紧凑风格**：项目上线时使用，可以减小文件体积

```
h1{color:red;font-size:40px}
```

基本选择器

8-基本选择器

基本选择器	语法	特点
通配选择器	*	选中所有标签
元素选择器	element-name	选中 element-name 的标签
类选择器	.class-name	选中类名为 class-name 的标签 (使用频率高)
id 选择器	#id-value	选中 id 值为 id-value 的单个标签

通配选择器

1. 通配选择器：*，可以选择页面中的所有 HTML 元素。

2. 语法

```
* {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意：通配选择器在清楚样式时有很大作用。

元素选择器

1. 元素选择器：标签名，可以选择页面中的特定标签的所有元素。

2. 语法

```
标签名 {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意：元素选择器的缺点是无法实现差异化设置。

类选择器

1. 类选择器：.类名，可以选择页面中有特定 class 值 (类名) 的所有元素。

2. 语法

```
.类名 {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意

- 元素的 `class` 值是我们自定义的，一般来说，不要使用纯数字、不要使用中文、**尽量使用英文与数字的组合，若由多个单词组成，使用 - 做连接**
- 一个元素**不能写多个 `class` 属性**，但是一个元素的 `class` 属性**可以写多个值**，用空格隔开

```
<!-- 错误示例 -->  
<h1 class="motto" class="quote">悟已往之不谏，知来者之可追</h1>  
<!-- 正确示例 -->  
<h1 class="motto quote">悟已往之不谏，知来者之可追</h1>
```

id 选择器

1. id 选择器: `#id值`，可以精准选中页面中某个有特定 `id` 值的元素。

2. 语法

```
#id值 {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意

- 元素的 `id` 值尽量由**字母、数字、下划线（_）、短横线（-）**组成，最好以字母开头，不要包含空格，**区分大小写**
- 一个元素只能拥有一个 `id` 属性，多个元素的 `id` 属性值不能相同
- 一个元素可以同时拥有 `id` 值和 `class` 属性

复合选择器

[9-复合选择器（交并集）](#)

[10-复合选择器（子后代、兄弟）](#)

[11-复合选择器（属性）](#)

[12-复合选择器（动态伪类）](#)

[13-复合选择器（结构伪类-常用）](#)

[14-复合选择器（其他伪类）](#)

[15-复合选择器（伪元素）](#)

复合选择器	语法	特点
交集选择器	<code>selector_1 selector_2 ... selector_n</code>	选中 n 个选择器选中的元素交集
并集选择器	<code>selector_1, selector_2, ..., selector_n</code>	选中 n 个选择器选中的元素并集
后代选择器	<code>selector_1 selector_2 ... selector_n</code>	选中指定元素中符合要求的后代元素
子代选择器	<code>selector_1>selector_2>...>selector_n</code>	选中指定元素中符合要求的子代元素
兄弟选择器	<code>selector_1+selector_2 or selector_1~selector_2</code>	选中指定元素的一个下方直接相邻兄弟 or 所有下方兄弟
属性选择器	<code>[属性名="属性值"] or more</code>	选中具有某个属性且具有特定值的元素
伪类选择器	动态伪类、结构伪类、否定伪类	

交集选择器

1. **交集选择器:** `selector_1 selector_2 ... selector_n`, 可以选择同时符合多个条件的元素。

2. 语法

```
selector_1 selector_2 ... selector_n {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}
```

3. 注意

- 如果元素选择器是交集选择器的一部分，则元素选择器必须写在最前面
- id 选择器也可以作为交集选择器的一部分，但是实际应用中几乎不用，没有任何意义
- 交集选择器中不可能出现两个元素选择器
- 元素选择器配合类名选择器是用的最多的交集选择器

并集选择器

1. **并集选择器:** `selector_1, selector_2, ..., selector_n`, 可以选择符合任意条件的元素。

2. 语法

```

selector_1,
selector_2,
...,
selector_n {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

```

3. 注意

- 任何选择器都可以作为并集选择器的一部分
- 并集选择器通常用于**集体声明**, 可以缩小样式表的体积

元素之间的关系

关系	图示
父元素 : 直接包裹某个元素的元素, 就是该元素的父元素	<pre> <div> <h1>欢迎来到尚硅谷</h1> 前端 Java 大数据 UI </div> </pre>
子元素 : 被父元素直接包裹的元素 (儿子元素)	<pre> <div> <h1>欢迎来到尚硅谷</h1> 前端 Java 大数据 UI </div> </pre>
祖先元素 : 父亲的父亲....., 一直向外找, 都是祖先。	<pre> <div> <h1>欢迎来到尚硅谷</h1> 前端 Java 大数据 UI </div> </pre>
后代元素 : 儿子的儿子....., 一直向里找, 都是后代。	<pre> <div> <h1>欢迎来到尚硅谷</h1> 前端 Java 大数据 UI </div> </pre>
兄弟元素 : 具有相同父元素的元素, 互为兄弟元素。	<pre> <div> <h1>欢迎来到尚硅谷</h1> 前端 Java 大数据 UI </div> </pre>

- 父元素也算是祖先元素的一种, 但是一般还是称呼为父元素
- 子元素也算是后代元素的一种, 但是一般还是称呼为子元素

后代选择器 (day4|71p: 26.1%)

1. **后代选择器**: `selector_1 selector_2 ... selector_n`, 选中指定元素中符合要求的**后代元素**。

2. 语法

```
selector_1 selector_2 ... selector_n {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意

- 这里的空格, 语义上可以理解为“xxx中的”, 即**后代**
- `selector` 既可以是**基本选择器**, 也可以是**交并集选择器**
- 后代选择器最终选择的是**后代**, **儿子**、**孙子**、**重孙子**等都是后代
- HTML 结构一定要符合 HTML 的嵌套要求, 例如不能在 `p` 标签中写 `h1 ~ h6` (此时后代选择器 `p h1` 是不生效的)

子代选择器

1. **子代选择器**: `selector_1>selector_2>...>selector_n`, 选中指定元素中符合要求的**子元素**。

2. 语法

```
selector_1>selector_2>...>selector_n {  
    属性名1: 属性值1;  
    属性名2: 属性值2;  
    ...  
    属性名n: 属性值n;  
}
```

3. 注意

- 子代选择器最终选择的是**子代**
- 后代包含子代, 范围更广



兄弟选择器

1. **相邻兄弟选择器**: `selector_1+selector_2`, 选中指定元素中符合要求的**相邻兄弟元素**。这里的**相邻**, 指的是紧挨着**指定元素的下一个元素**。

2. **通用兄弟选择器**: `selector_1~selector_2`, 选中指定元素中符合要求的**所有兄弟元素**。这里的**所有兄弟元素**, 指的是**紧挨着指定元素下的所有元素**。

3. 语法

```

selector_1+selector_2 {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

selector_1~selector_2 {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

```

4. 注意：两种兄弟选择器，选择的都是其下边的兄弟元素。

属性选择器

1. 属性选择器

- **[属性名]** 选中具有某个属性的元素
- **[属性名="值"]** 选中具有某个属性，并且属性值等于指定值的元素
- **[属性名^="值"]** 选中具有某个属性，并且属性值以指定值开头的元素
- **[属性名\$="值"]** 选中具有某个属性，并且属性值以指定值结尾的元素
- **[属性名*="值"]** 选中具有某个属性，并且属性值包含指定值的元素

2. 语法

```

[属性名] {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

[属性名="值"] {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

[属性名^="值"] {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

[属性名$="值"] {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

```

```

}

[属性名*= "值"] {
    属性名1: 属性值1;
    属性名2: 属性值2;
    ...
    属性名n: 属性值n;
}

```

伪类选择器

1. **伪类**: 像类, 但不是类, 是**元素特殊状态**的一种描述。

2. **伪类选择器**: 选中某种特殊状态的元素。

动态伪类

伪类选择器	含义
<code>selector:link</code>	标签 (一般是超链接) 未被访问 的状态
<code>selector:visited</code>	标签 (一般是超链接) 访问过 的状态
<code>selector:hover</code>	鼠标 悬停 在标签上的状态
<code>selector:active</code>	标签 激活 (即按下鼠标不松开) 的状态
<code>selector:focus</code>	标签 获取焦点 的状态

- 前四个伪类选择器, 务必遵循 LVHA 的顺序编写样式表, 即: `link`、`visited`、`hover`、`active`

LVHA 原因分析, 以超链接 `a` 为例

- 如果不触碰超链接, 则处于 `link/visited` 状态
- 如果鼠标悬停在超链接上, 则处于 `link/visited` 和 `hover` 状态
- 如果鼠标点击超链接不松开, 则处于 `link/visited` 和 `hover` 和 `active` 状态
- 基于上述分析, 我们在使用伪类选择器时, 最好遵循 LVHA 的顺序

- 只有表单元素才能使用 `:focus` 伪类, 当点击元素、触摸元素、`tab` 键选择等方式都可以让元素获取焦点

结构伪类 (常用)

伪类选择器	含义
<code>selector:first-child</code>	<code>selector</code> 选中的标签中是 其父亲的第一个儿子 的标签
<code>selector:last-child</code>	<code>selector</code> 选中的标签中是 其父亲的最后一个儿子 的标签
<code>selector:nth-child(n)</code>	<code>selector</code> 选中的标签中是 其父亲的第n个儿子 的标签
<code>selector:first-of-type</code>	<code>selector</code> 选中的标签中是 其父亲的第一个特定类型 (同类型) 儿子 的标签

伪类选择器	含义
<code>selector:last-of-type</code>	<code>selector</code> 选中的标签中是其父亲的最后一个特定类型（同类型） 儿子的标签
<code>selector:nth-of-type(n)</code>	<code>selector</code> 选中的标签中是其父亲的第n个特定类型（同类型）儿子的标签

- 【HARD-TO-COMPREHEND】前三种伪类选择器的次序是在**所有兄弟元素**中的，后三种伪类选择器的次序是在**所有同类型兄弟元素**中的；这两类伪类选择器的不同是，比较的范围不同
- n 的不同取值对应不同的情况（括号里的内容的标准写法为 $an+b$ ）

n=?	含义
0 或不写	什么都选不中
n	选中所有子元素
1~ $+\infty$	选中对应序号的子元素
$2n$ 或 even	选中序号为偶数的子元素
$2n+1$ 或 odd	选中序号为奇数的子元素
$-n+3$	选中前三个子元素

- 通过代码及其渲染结果举例

```
<!-- 网页结构：这一部分是不变的，然后我们根据不同的 CSS 观察网页的效果变化 -->
<div>
  <span>选择器举例</span>
  <ul>
    <li>伪类选择器</li>
    <li>动态伪类</li>
    <li>结构伪类</li>
    <li>否定伪类</li>
    <li>UI伪类</li>
    <li>目标伪类</li>
    <li>语言伪类</li>
  </ul>
  <ul>
    <li>复合选择器</li>
    <li>并集选择器</li>
    <li>交集选择器</li>
    <li>属性选择器</li>
    <li>子代选择器</li>
    <li>后代选择器</li>
    <li>兄弟选择器</li>
  </ul>
</div>
```

CSS	效果	解释
无	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	没有样式时候的网页
<pre>div ul:first-child { color: ■ red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	<p><code>div ul</code> 选中 <code>div</code> 的后代元素 <code>ul</code>，此时可以找到两个 <code>ul</code> 元素； <code>:first-child</code> 要求从这两个 <code>ul</code> 元素中找到是其父元素的第一个子元素，因为其父元素 <code>div</code> 的第一个子元素是 <code>span</code>，因此无法找到符合要求的 <code>ul</code>； 最终没有选中任何元素</p>
<pre>div ul:last-child { color: ■ red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	<p><code>div ul</code> 选中 <code>div</code> 的后代元素 <code>ul</code>，此时可以找到两个 <code>ul</code> 元素； <code>:last-child</code> 要求从这两个 <code>ul</code> 元素中找到是其父元素的最后一个子元素，因为其父元素 <code>div</code> 的最后一个子元素是 <code>ul</code>，因此可以找到符合要求的 <code>ul</code>； 最终选中了最后一个 <code>ul</code> 元素</p>
<pre>div ul:nth-child(2) { color: ■ red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	<p><code>div ul</code> 选中 <code>div</code> 的后代元素 <code>ul</code>，此时可以找到两个 <code>ul</code> 元素； <code>:nth-child(2)</code> 要求从这两个 <code>ul</code> 元素中找到是其父元素的第二个子元素，因为其父元素 <code>div</code> 的最后一个子元素是 <code>ul</code>，因此可以找到符合要求的 <code>ul</code>； 最终选中了第一个 <code>ul</code> 元素</p>
以上三种伪类选择器	其次序都是基于	父元素里边的所有子元素排的

CSS	效果	解释
以下三种伪类选择器	其次序都是基于	父元素里边的所有同类型的子元素排的
<pre>div ul:first-of-type { color: red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 <ul style="list-style-type: none"> • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	<pre>div ul 选中 div 的后代元素 ul, 此时可以 找到两个 ul 元素; :first-of-type 要求从这两个 ul 元素中找 到是其父元素的第一个 ul 子元素; 最终选中了第一个 ul 元素</pre>
<pre>div ul:last-of-type { color: red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 <ul style="list-style-type: none"> • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	原因显而易见
<pre>div ul:nth-of-type(2) { color: red; }</pre>	<p>选择器举例</p> <ul style="list-style-type: none"> • 伪类选择器 • 动态伪类 • 结构伪类 • 否定伪类 • UI伪类 • 目标伪类 • 语言伪类 <ul style="list-style-type: none"> • 复合选择器 • 并集选择器 • 交集选择器 • 属性选择器 • 子代选择器 • 后代选择器 • 兄弟选择器 	原因显而易见

结构伪类 (不常用)

伪类选择器	含义
<code>selector:nth-last-child(n)</code>	<code>selector</code> 选中的标签中是其父亲的倒数第n个儿子的标签
<code>selector:nth-last-of-type(n)</code>	<code>selector</code> 选中的标签中是其父亲的倒数第n个特定类型 (同类型) 的儿子的标签
<code>selector:only-child</code>	<code>selector</code> 选中的标签中没有兄弟的元素

伪类选择器	含义
<code>selector:only-of-type</code>	<code>selector</code> 选中的标签中 没有特定类型（同类型）兄弟的元素
<code>:root</code>	根元素
<code>selector:empty</code>	<code>selector</code> 选中的标签中 内容为空 的元素（注意空格也算内容）

其他伪类

伪类	形式	含义
否定伪类	<code>:not(选择器)</code>	表示 排除括号中条件 的元素
UI伪类	<code>:checked</code>	表示被选中的复选框或单选按钮
	<code>:enabled</code>	表示 可用的表单元素 （即么有 <code>disabled</code> 属性）
	<code>:disabled</code>	表示 不可用的表单元素 （即有 <code>disabled</code> 属性）
目标伪类	<code>:target</code>	选中锚点 指向 的元素
语言伪类	<code>:lang()</code>	根据 指定的语言 选择元素

伪元素选择器

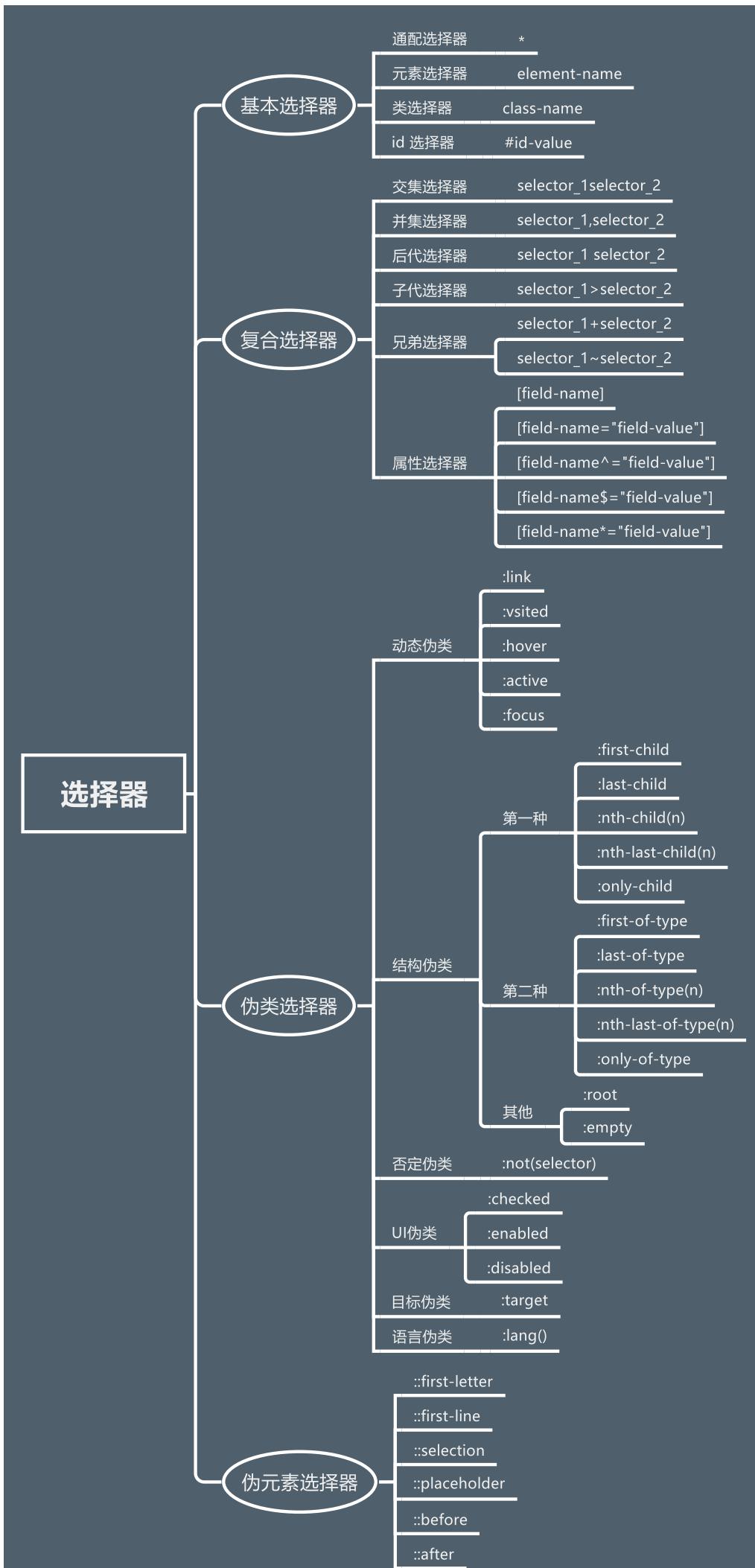
1. **伪元素**: 很像元素，但不是元素，是元素中的一些**特殊位置**。

2. **伪元素选择器**: 可以选中元素中的一些特殊位置。

- 伪类以 `:` 开始，伪元素以 `::` 开始

伪元素	含义
<code>::first-letter</code>	选中元素的 第一个文字
<code>::first-line</code>	选中元素的 第一行
<code>::selection</code>	选中 被鼠标选中 的内容
<code>::placeholder</code>	选中输入框的 提示文字
<code>::before</code>	在 元素最开始的位置 ，创建一个子元素（用 <code>content</code> 指定内容）
<code>::after</code>	在 元素最后的位置 ，创建一个子元素（用 <code>content</code> 指定内容）

SUMMARY of SELECTORS



选择器优先级

1. **样式冲突**: 当通过**不同选择器**, 选中**相同的元素**, 并且为**相同的样式名**设置**不同的值**时, 就发生了**样式冲突**, 此时就需要根据**选择器的优先级**决定到底应用哪个样式。其中, 当两个选择器优先级相同时, 则遵从“**后来居上**”的原则, 应用位于较下边的样式。

2. 简单的说, 有以下优先级: **!important>行内样式>ID选择器>类选择器>元素选择器>通配选择器>继承的样式**。

- 其中 **!important** 是写在一个**声明之后, 分号之前**, 表明该声明最重要

```
.slogan{  
    color: purple !important;  
}
```

- 继承的样式**指的是, 任何标签会默认继承其祖先元素的样式

3. 对于复杂的复合选择器, 我们采用①**计算权重**, ②**依次比较**的方式进行优先级比较。

- 权重的计算**: (a, b, c) , a 是 ID 选择器个数, b 是类、伪类、属性选择器个数, c 是元素、伪元素选择器个数



- 优先级比较**: 按照**从左到右**的顺序, 依次比较大小, 当前位胜出后, 后面的不再对比 (如 $(1,0,0)>(0,2,2)$, $(1,1,0)>(1,0,3)$)

4. 权重计算举例

通配选择器 (0,0,0)	元素/伪元素 * 1 (0,0,1)	元素/伪元素 * 2 (0,0,2)	元素/伪元素* 5 (0,0,5)
类/伪类/属性 * 1 (0,1,0)	类/伪类/属性 * 1 元素/伪元素 * 1 (0,1,1)	类/伪类/属性 * 2 元素/伪元素 * 1 (0,2,1)	类/伪类/属性 * 2 元素/伪元素 * 2 (0,2,2)
ID选择器* 1 (1,0,0)	ID选择器* 1 类/伪类/属性 * 1 (1,1,0)	ID选择器* 1 类/伪类/属性 * 1 元素/伪元素 * 1 (1,1,1)	ID选择器* 1 类/伪类/属性 * 1 元素/伪元素 * 2 (1,1,2)
#atguigu	#atguigu .slogan	#atguigu .slogan::before	#atguigu p .slogan::before
#atguigu>.slogan [title]	#atguigu>.slogan [title]:first-child	style = ""	!important
ID选择器* 1 类/伪类/属性 * 2 (1,2,0)	ID选择器* 1 类/伪类/属性 * 8 (1,3,0)	(1,a,b,c)	(1,?,a,b,c)

三大特性

- 层叠性：如果发生了样式冲突，那就会根据一定的规则（选择器的优先级），进行样式的层叠（覆盖）。
- 继承性：元素会自动拥有其父元素或其祖先元素上所设置的某些样式。
 - 优先级继承离得近的祖先元素的某些样式
 - 有些属性是可以继承的，有些属性是不可以继承的
 - 常见的可继承属性有：`text-???`, `font-???`, `line-???`, `color`, ...
- 优先级：简单的说，`!important`>行内样式>ID选择器>类选择器>元素选择器>通配选择器>继承的样式；复杂的复合选择器则需要通过权重计算进行优先级比较。
 - 并集选择器的每一个部分的权重都是分开计算的

像素和颜色 (day5 | 88p: 32.8%)

- 像素 (pixel)：图像中的最小的元素点，具有特定的位置及颜色值，每个像素的颜色是由强度不同的红、绿、蓝三原色组合而成的。

光的三原色：红、绿、蓝

颜料的三原色：红、黄、蓝

- 颜色的表示

- 颜色名：使用颜色对应的英文单词表示颜色，如 red、green、blue、purple 等

```
color: red;
```

- 这种颜色表达方式比较单一，使用并不多

- 具体颜色名需要参考 MDN 官方文档，不能胡写
- rgb/rgba：使用**红、绿、蓝**这三种光的三原色进行组合来表示具体颜色，其中 r 表示红色， g 表示绿色， b 表示蓝色， a 表示透明度

```

color: rgb(255, 0, 0); /* 红色 */
color: rgb(0, 255, 0); /* 绿色 */
color: rgb(0, 0, 255); /* 蓝色 */
color: rgb(0, 0, 0); /* 黑色 */
color: rgb(255, 255, 255); /* 白色 */

color: rgb(100%, 0%, 0%); /* 红色 */

color: rgba(255, 0, 0, 0.5); /* 半透明的红色 */
color: rgba(100%, 0%, 0%, 50%); /* 半透明的红色 */

```

- 若 $r=g=b$ ，则呈现是灰色，三者取值越大，则灰色越浅
- $\text{rgb}(0,0,0)$ 是黑色， $\text{rgb}(255,255,255)$ 是白色
- rgb 可以用 0~255 的数字表示，也可以用 0%~100% 的百分比表示，一定要统一使用
- a 可以用 0~1 的数字表示，也可以用 0%~100% 的百分比表示
- HEX/HEXA：原理同 rgb/rgba ，不过是使用**十六进制**，用六个数字表示一个特定颜色，两个数字为一组，三组分别对应红、绿、蓝三原色

```

color: #ff0000; /* 红色 */
color: #00ff00; /* 绿色 */
color: #0000ff; /* 蓝色 */

color: #ff9988; /* 可简写为 #f98 */
color: #ff998866; /* 可简写为 #f986 */

```

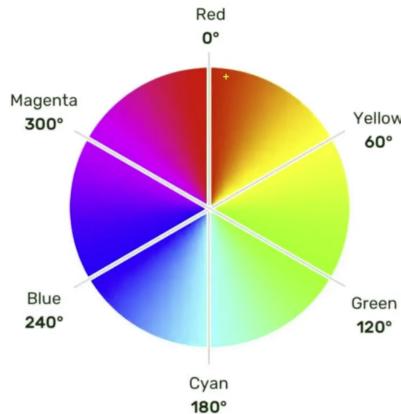
- IE 浏览器不支持 HEXA，但支持 HEX
- 如果要引入透明度，则使用八个数字表示一个颜色，两个数字为一组，每组取值范围为 00~ff
- 如果每组的两个数字相同，则可以进行简写，每组用一个数字表示颜色，取值范围为 0~f，如 ffaa99 可简写为 fa9，ffaa8811 可简写为 fa11
- HSL/HSLA：使用**色相、饱和度、亮度**三者进行组合来表示具体颜色，其中 H 表示色相环上的角度，S 表示饱和度，L 表示亮度，a 表示透明度

```

color: hsl(0, 100%, 50%); /* 红色 */
color: hsla(0, 100%, 50%, 50%); /* 半透明红色 */

```

- 色相：取值范围是 0~360 度，具体每个度数对应的颜色如下色相图



- 饱和度：取值范围是 0%~100%
- 亮度：取值范围是 0%~100%

常用字体属性

1. 字体大小 `font-size`

- 每个浏览器都会有**默认字体大小**和**最小支持字体大小**，且都是可以在浏览器设置页面更改的，例如，Chrome 默认最小支持文字大小为 12px，默认字体大小为 16px
- 如果给标签设置的字体大小<浏览器最小支持字体大小，则浏览器只显示其最小支持的字体大小
- 可以通过给 `body` 设置 `font-size` 属性，从而控制整个页面的默认字体大小（CSS 的三大特性之继承性）
- 注：通过开发者工具的 `styles` 选项卡，可以看到所选元素的样式和继承样式等信息

2. 字体族 `font-family`

- 字体可以分为**衬线字体** (`serif`) 和**非衬线字体** (`sans-serif`)，实际开发中多使用非衬线字体
- 设置标签的字体时，最好使用字体的**英文名称**，并且用**双引号**包裹，多字体之间用**逗号**隔开
- 给标签设置多个字体时，浏览器会按照**从左到右**的顺序逐个查找，系统中找到对应字体则使用，否则往后继续查找
- 给标签设置多个字体时，通常以（不带双引号）`serif` 或 `sans-serif` 结尾，代表一类字体，意思是：如果浏览器没有找到前边的字体，则**从系统中的衬线/非衬线字体中找一个字体渲染网页**
- windows 系统中的**默认字体是微软雅黑**

3. 字体风格 `font-style` (可取值: `normal`、`italic`、`oblique`)

- `normal` 是默认值，表示**正常**；`italic` 是斜体，特指**字体自带**的斜体效果；`oblique` 是斜体，表示**强制倾斜**得到的斜体效果
- 开发中推荐使用 `italic` 实现斜体效果，浏览器会先去找改字体库是否存在相关文字的斜体，如果存在则使用，否则将将相关文字强制倾斜来实现斜体效果

4. 字体粗细 `font-weight` (可取值: `lighter`、`normal`、`bold`、`bolder`、`100~1000`)

- `normal` 是默认值，表示**正常**；`lighter` 表示**细**；`bold` 表示**粗**；`bolder` 表示**很粗**
- `100~1000` 的取值是无单位的，**数值越大则字体越粗**（也有可能一样粗）；`100~300` 等同于 `lighter`、`400~500` 等同于 `normal`、`600` 及以上等同于 `bold`
- 与字体风格类似，如果字体设计的过程中只设计了三种粗细，则无论怎么调整，最终只能渲染出三种字体的粗细效果（例如 `bold` 和 `bolder` 的效果是相同的，`100` 和 `300` 的效果是相同的）

5. 字体的复合属性 `font`

- 字体的复合属性即将字体**大小、族、风格、粗细**合并成一个属性
- 字体的复合属性要求
 - 必须写上**字体大小、字体族**
 - **字体族必须是最后一位、字体大小必须是倒数第二位**
 - 字体风格与字体粗细则是可选的，可以不写，写了也没有顺序要求
 - 各个属性之间使用**空格**分开

```

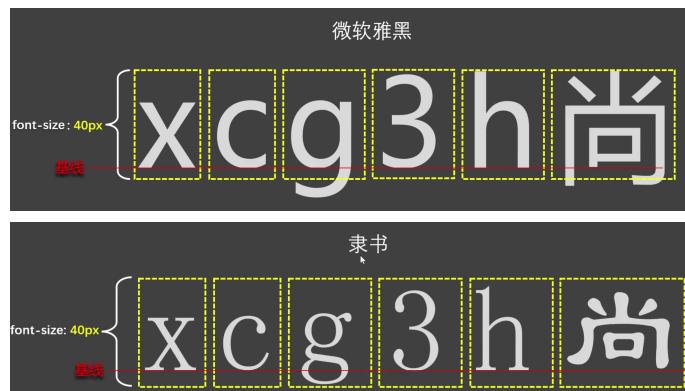
div {
  font-size: 40px;
  font-family: "STCaiyun", "Microsoft YaHei", sans-serif;
  font-style: italic;
  font-weight: bold;
}

h2 {
  font: bold italic 100px "STCaiyun", "Microsoft YaHei", sans-serif;
}

```

从字体设计角度讨论 font-size

1. 设计师怎么设计字体（简单说）？每个字体都是放在一个**字体设计框**中进行设计的；其中 X 最为特殊，其底部与字体设计框的**基线重合**。



2. `font-size` 是什么？`font-size` 可以理解为**字体设计框的高度**，当我们调整 `font-size` 时，其实是调整字体设计框的高度，与之同时，字体设计框的宽度也等比例变化，字体于是也被放大了。

3. 重要结论

- 由于字体设计原因，**文字最终呈现的大小，并不一定与 `font-size` 的值一致**，可能大，也可能小
- 通常情况下，文字相对字体设计框，并不是垂直居中的，**通常都靠下一些**。

常用文本属性

1. 文本颜色 `color` (可选值：颜色名、rgb/rgba、HEX/HEXA、HSL/HSLA)

2. 文本间距

- **字母间距** `letter-spacing` (单位是 px)
- **单词间距** `word-spacing` (单位是 px)
- 注
 - 浏览器通过**空格**识别单词

- 正值让间距增大，负值让间距缩小

3. 文本修饰 `text-decoration` (可选值: `none`、`underline`、`overline`、`line-through`；可搭配 `dotted`、`wavy` 和颜色使用)

- `none` 表示无装饰线；`underline` 表示下划线；`overline` 表示上划线；`line-through` 表示删除线
- `dotted` 表示虚线；`wavy` 表示波浪线
- 属性值举例：每种线都可以指定其线型和颜色等样式
 - `overline dotted green`：绿色上划虚线
 - `underline wavy red`：红色下划波浪线
 - `line-through`：删除线
 - `none`：无任何线（可用于删除超链接 `a` 标签的下划线）

4. 文本缩进：`text-indent`（单位是 px）

- `text-indent` 取值一般是 `font-size` 的两倍，可以实现缩进两个文字的效果

5. 文本水平对齐 `text-align` (可选值: `left`、`right`、`center`)

- `left` 是默认值，表示左对齐；`right` 表示右对齐；`center` 表示居中对齐

6. 文本行高 `line-height` (可选值: `normal`、具体的像素、数字、百分比)

- `normal` 表示浏览器根据文字大小决定的一个默认值；具体的像素表示：直接设置行高（px）；数字表示：设置行高为文字大小（`font-size`）的倍数；百分比表示：设置行高为文字大小的百分比
- 行与行之间是紧紧贴在一起的
- 最好不要让行高等于字体大小 (`line-height==font-size`)！由于字体设计的过程中，有些文字可能会超出字体设计框，因此如果行高等于字体大小，则可能会发生“字体打架”的情况
- 由于字体设计的原因，文字在一行中并不是绝对垂直居中

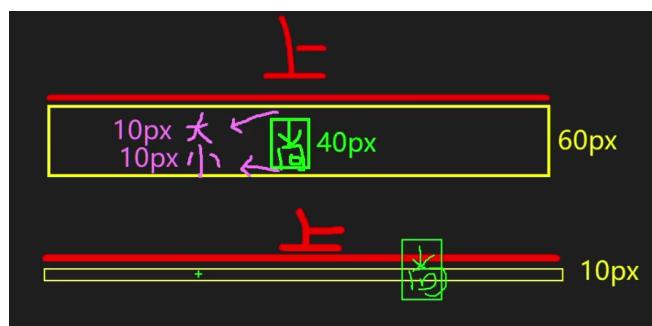
7. 文本垂直对齐（暂时用行高办法实现）

- 置顶：默认情况就是顶部对齐，不做讨论
- 居中：对于单行文字，让 `height` 等于 `line-height`
- 底部：对于单行文字，让 `height × 2 - font-size × x` 等于 `line-height`，其中 `x` 是根据特定的字体，动态确定的一个值
- 注：垂直方向上的对齐，下面会用定位的方式去实现

进一步讨论 `line-height`

1. 关于行高的三点讨论

- `line-height` 过小会怎样？



- 文字重叠
- 同时行高最小值是 0，不能是负数
- `line-height` 的继承性
 - 行高是可继承的属性
 - 为了更好的呈现文字，最好让行高等于一个数值（文字大小的倍数），这样的话，虽然子代元素的行高与祖先元素的行高相同，但是子代元素会根据其文字大小（`font-size`）来计算自己的行高
- `line-height` 和 `height` 之间的关系
 - 如果已经设置了 `height`，则高度就是 `height`
 - 如果没有设置 `height`，则高度就是 `line-height`

2. 行高的应用场景

- 调整多行文字行与行的间距
- 单行文字的垂直居中：让 `height` 等于 `line-height`

元素垂直对齐 vertical-align

`vertical-align`，可选值：`baseline`、`top`、`middle`、`bottom`，用于指定①同一行元素之间或②表格单元格内文字的垂直对齐方式。

- `baseline` 默认值，表示使元素的基线与其父元素的基线对齐
- `top` 表示使元素的顶部与其所在行的顶部对齐
- `bottom` 表示使元素的底部与其所在行的底部对齐
- `middle` 表示使元素的中部与父元素的基线加上父元素字母 x 的一半对齐



列表相关属性

列表相关属性见下表，只可以作用在 `ul`、`ol`、`li` 元素上。

属性名	作用	属性值
<code>list-style-type</code>	设置列表符号	<code>none</code> 不显示列表符号 <code>square</code> 实心方块 <code>disc</code> 圆形 <code>decimal</code> 数字 <code>lower-roman</code> 小写罗马字 <code>upper-roman</code> 大写罗马字 <code>lower-alpha</code> 小写字母 <code>upper-alpha</code> 大写字母
<code>list-style-position</code>	设置列表符号的位置	<code>inside</code> 在 <code>li</code> 里边 <code>outside</code> 在 <code>li</code> 外边
<code>list-style-image</code>	自定义列表符号	<code>url(图片地址)</code>
<code>list-style</code>	列表复合属性	上述三个属性的复合属性，没有数量和顺序要求

```
list-style-type: decimal;
list-style-position: inside;
list-style-image: url("../images/haha.png");
list-style: decimal inside url("../images/haha.png");
```

边框相关属性

边框相关属性见下表，可以用在很多元素上。

属性名	作用	属性值
<code>border-width</code>	边框宽度	CSS 中的长度值
<code>border-color</code>	边框颜色	CSS 中的颜色值
<code>border-style</code>	边框风格	<code>none</code> 默认值 <code>solid</code> 实线 <code>dashed</code> 虚线 <code>dotted</code> 点线 <code>double</code> 双实线
<code>border</code>	边框复合属性	上述三个属性的复合属性，没有数量和顺序要求

```
border-width: 2px;
border-color: green;
border-style: solid;
border: 2px green solid;
```

表格独有属性

表格独有属性见下表，只有 `table` 标签才能使用。

属性名	作用	属性值
<code>table-layout</code>	设置列宽度	<code>auto</code> 默认值，表示根据内容自动计算列宽 <code>fixed</code> 表示固定列宽，均分
<code>border-spacing</code>	单元格间距	CSS 中的长度值 (生效前提：单元格边框不能合并)
<code>border-collapse</code>	合并单元格边框	<code>collapse</code> 合并 <code>separate</code> 默认值，表示不合并
<code>empty-cells</code>	隐藏没有内容的单元格	<code>show</code> 默认值，表示显示 <code>hide</code> 表示隐藏 (生效前提：单元格边框不能合并)
<code>caption-side</code>	设置表格标题位置	<code>top</code> 默认值，表示在上面 <code>bottom</code> 表示在下边

```
table-layout: fixed;
border-spacing: 50px;
border-collapse: collapse;
empty-cells: hide;
caption-side: top;
```

背景相关属性

背景相关属性见下表，

属性名	作用	属性值
<code>background-color</code>	设置背景颜色	CSS 中的颜色值，默认为 <code>transparent</code> ，即透明
<code>background-image</code>	设置背景图片	<code>url(图片地址)</code>
<code>background-repeat</code>	设置背景重复方式	<code>repeat</code> 默认值，表示背景图重复铺满整个元素； <code>repeat-x</code> 表示只在水平方向重复 <code>repeat-y</code> 表示只在垂直方向重复 <code>no-repeat</code> 表示不重复
<code>background-position</code>	设置背景图位置	要么通过 关键字 设置位置；要么通过 坐标值 指定位置
<code>background</code>	复合属性	上述四个属性的复合属性，没有数量和顺序要求

- 背景图片如果太大，则会选择一部分填充元素；如果太小，则会通过重复填充元素
- 关于背景图位置的设置
 - **关键字：**两个用空格隔开的**关键字**，一个控制水平方向上的位置，一个控制垂直方向上的位置
 - 水平位置关键字：`left`、`center`、`right`
 - 垂直位置关键字：`top`、`center`、`bottom`

- 注：如果只写一个关键字，则默认另一个位置的关键字取值为 `center`（例如 `center` 表示中间，`left` 表示水平左边垂直中间，`right` 表示水平右边垂直中间，`top` 表示水平中间垂直上边，`bottom` 表示水平中间垂直下边）
- 坐标值：以元素的左上角为原点，上边框为 x 轴，下边框为 y 轴建立坐标系；在坐标系中设置要填充的左上角的位置，分别是 x 值和 y 值，单位是 CSS 中的长度单位
 - 注：第一个数字表示 x 坐标取值，第二个数字表示 y 坐标的取值，如果只写一个数字，则会被认为是 x 坐标，y 坐标取值 `center`
- 关于复合属性的使用注意点：复合属性（`background`）中，如果不设置 `background-color` 的值，其实是默认设置了是 `transparent` 的取值，因此如果在 `background` 上边已经设置了 `background-color` 的话，此时背景颜色会被覆盖为透明色

```
background-color: skyblue;
background-image: url("../images/haha.png");
background-repeat: no-repeat;
background-position: center;
background: url("../images/haha.png") no-repeat center;
/* 此时的背景色为透明色，因为 background 中如果有设置背景色的话，自动认为设置了 transparent */
```

鼠标相关属性

鼠标相关属性见下表，如果给某个元素设置了该属性后，当鼠标光标进入该元素范围后，就会改变样式。

属性名	作用	属性值
<code>cursor</code>	设置鼠标光标的样式	<code>pointer</code> 小手 <code>move</code> 移动图标 <code>text</code> 文字选择器 <code>crosshair</code> 十字架 <code>wait</code> 等待 <code>help</code> 帮助

自定义鼠标光标：`cursor("../arrow.png"), pointer`

常用长度单位

单位	含义
<code>px</code>	像素
<code>em</code>	相对于当前元素或其祖先元素的 <code>font-size</code> 的倍数
<code>rem</code>	相对于根元素（ <code>html</code> ）的 <code>font-size</code> 的倍数
<code>%</code>	相对于其父元素对应属性的百分比

- `indent` 属性可以直接使用 `2em` 实现首行缩进两个文字

```

width: 10em; /* 当前元素的宽度是其字体大小的十倍 */
height: 10rem; /* 当前元素的高度是根元素字体大小的十倍 */
font-size: 150%; /* 当前元素的字体大小是其父元素字体大小的 1.5 倍 */

```

元素的显示模式

1. 元素的显示模式：块元素、行内元素、行内块元素，具体区别见下表。

块元素 block	行内元素 inline	行内块元素 inline-block
又称块级元素	又称内联元素	又称内联块元素
在页面中独占一行，不与任何元素共用一行，从上到下排列	在页面中不独占一行，一行中无法容纳下的行内元素，会在下一行继续从左到右排列	在页面中不独占一行，一行中无法容纳下的行内块元素，会在下一行继续从左到右排列
默认宽度撑满父元素	默认宽度由内容撑开	默认宽度由内容撑开
默认高度由内容撑开	默认高度由内容撑开	默认高度由内容撑开
可以通过 CSS 设置宽高	无法通过 CSS 设置宽高	可以通过 CSS 设置宽高

2. 常用元素的显示模式总结

块元素 block	行内元素 inline	行内块元素 inline-block
主体结构标签 html、body		图片标签 img
排版标签 h1-h6、hr、p、pre、div	文本标签 br、em、strong、sup、sub、del、ins	单元格标签 td、th
列表标签 ul、ol、li、dl、dt、dd	a、label	表单控件 input、textarea、select、button
表格相关标签 table、tbody、thead、tfoot、tr、caption		框架标签 iframe
表单相关标签 form、option		

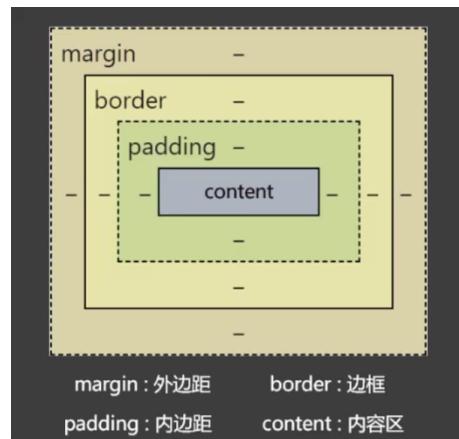
3. 修改元素的显示模式：通过 `display` 属性可以修改元素的默认显示模式。

属性值	作用
<code>none</code>	元素会被隐藏
<code>block</code>	元素将作为块级元素显示
<code>inline</code>	元素将作为内联元素显示
<code>inline-block</code>	元素将作为行内块元素显示

盒子模型

组成

1. 盒子模型：CSS 把所有的 HTML 元素都看成一个盒子，所有的样式也都是基于这个盒子。



2. 盒子模型的组成

- **外边距** margin: 盒子与外界的距离
- **边框** border: 盒子的边框
- **内边距** padding: 紧贴内容的补白区域
- **内容** content: 元素中的文本或后代元素

3. 注意事项

- 盒子的宽度 = 内容 (content) + 左右内边距 (padding) + 左右边框 (border)
- 外边距 (margin) 不会影响盒子的大小，但是会影响盒子的位置

```
width: 400px; /* 内容区的宽 */
height: 400px; /* 内容区的高 */
padding: 20px; /* 设置内边距 */
border: 10px solid transparent; /* 设置边框 */
margin: 10px; /* 设置外边距 */
```

内容区

视图：`body` 标签铺满了整个浏览器的视图

盒子模型的内容区共涉及了**两类（宽度和高度）**，六个属性。

属性名	作用	属性值
<code>width</code>	设置内容区的宽度	长度
<code>max-width</code>	设置内容区的最大宽度	长度
<code>min-width</code>	设置内容区的最小宽度	长度
<code>height</code>	设置内容区的高度	长度
<code>max-height</code>	设置内容区的最大高度	长度
<code>min-height</code>	设置内容区的最小高度	长度

- 三个宽度属性一般不一起使用，三个高度属性也一般不一起使用

默认宽度

1. 默认宽度：不设置 `width` 属性时，元素呈现出来的宽度
2. 总宽度 = 父元素的内容区宽度 (content) - 自身左右的外边距 (margin)
3. 内容区宽度 = 父元素的内容区宽度 (content) - 自身左右的外边距 (margin) - 自身左右的边框 (border) - 自身左右的内边距 (padding)

内边距

盒子模型的内边距共涉及了五个属性，八种用法。

属性名	作用	属性值
<code>padding-top</code>	上内边距	长度
<code>padding-right</code>	右内边距	长度
<code>padding-bottom</code>	下内边距	长度
<code>padding-left</code>	左内边距	长度
<code>padding</code>	复合属性	四种使用方式

- `padding` 属性的四种使用规则

属性名:属性值	作用
<code>padding:10px</code>	四个方向的内边距都是 <code>10px</code>
<code>padding: 10px 20px</code>	上下内边距 <code>10px</code> , 左右内边距 <code>20px</code>
<code>padding: 10px 20px 30px</code>	上内边距 <code>10px</code> , 左右内边距 <code>20px</code> , 下内边距 <code>30px</code>
<code>padding: 10px 20px 30px 40px</code>	上内边距 <code>10px</code> , 右内边距 <code>20px</code> , 下内边距 <code>30px</code> , 左内边距 <code>40px</code>

- `padding` 的取值不能是负数
- 行内元素的左右边距可以完美设置，但是上下内边距的设置存在问题（会出现不占位的问题）
- 块级元素和行内块元素的四个方向的内边距都可以完美设置

边框

盒子模型的边框共涉及了五类，二十个属性。

属性名	作用	属性值
<code>border-style</code>	复合属性，指定四个方向的边框风格	<code>none</code> 默认值，表示什么也没有 <code>solid</code> 表示实线 <code>dashed</code> 表示虚线 <code>dotted</code> 表示点线 <code>double</code> 表示双实线 (类同于 <code>padding</code> 也有四种使用方式)
<code>border-width</code>	复合属性，指定四个方向的边框宽度	长度，默认 3px (类同于 <code>padding</code> 也有四种使用方式)
<code>border-color</code>	复合属性，指定四个方向的边框颜色	颜色，默认黑色 (类同于 <code>padding</code> 也有四种使用方式)
<code>border</code>	复合属性，指定四个方向的边框风格、宽度、颜色	风格、宽度、颜色 (风格一定要指定，其他两个可选；三个值的顺序任意) (只能同时指定四个方向的边框)
<code>border-left-style</code>	设置左边框风格	同上
<code>border-left-width</code>	设置左边框宽度	同上
<code>border-left-color</code>	设置左边框颜色	同上
<code>border-left</code>	复合属性，设置左边框的风格、宽度、颜色	同上
<code>border-right-style</code>	设置右边框风格	同上
<code>border-right-width</code>	设置右边框宽度	同上
<code>border-right-color</code>	设置右边框颜色	同上
<code>border-right</code>	复合属性，设置右边框的风格、宽度、颜色	同上
<code>border-top-style</code>	设置上边框风格	同上
<code>border-top-width</code>	设置上边框宽度	同上
<code>border-top-color</code>	设置上边框颜色	同上
<code>border-top</code>	复合属性，设置上边框的风格、宽度、颜色	同上

属性名	作用	属性值
<code>border-bottom-style</code>	设置下边框风格	同上
<code>border-bottom-width</code>	设置下边框宽度	同上
<code>border-bottom-color</code>	设置下边框颜色	同上
<code>border-bottom</code>	复合属性，设置下边框的风格、宽度、颜色	同上

外边距

盒子模型的外边距共涉及五个属性，八种用法，类同内边距。

属性名	作用	属性值
<code>margin-top</code>	上外边距	长度
<code>margin-right</code>	右外边距	长度
<code>margin-bottom</code>	下外边距	长度
<code>margin-left</code>	左外边距	长度
<code>margin</code>	复合属性	四种使用方式，类同 <code>padding</code>

关于 `margin` 的进一步讨论 (day6 | 123p: 46.2%)

1. `margin` 的注意事项

- 子元素的 `margin` 是参考父元素的 `content` 计算的（子元素的盒子模型包含在父元素的内容区）
 - | 这里我们理解：盒子模型=外边距+边框+内边距+内容区；盒子=边框+内边距+内容区
- 块级元素和行内块元素可以完美设置四个方向的 `margin`；但是行内元素，只能完美设置其左右的 `margin`，上下的 `margin` 设置无效（类似 `padding`）
- 给块级元素设置 `margin: 0 auto`（即左右 `margin` 为 `auto`），表示该块级元素在父元素中水平居中
- `margin` 可以取负值

2. `margin` 塌陷问题：给父元素中的第一个子元素设置上边距 or 最后一个子元素设置下外边距，此时外边距会被父元素“抢走”，也就是成为了父元素的外边距。该问题是一个历史遗留问题。

- 解决方式1 (not good)：给父元素添加不为零的 `border` 属性
- 解决方式2 (not good)：给父元素添加不为零的 `padding` 属性
- 解决方式3：给父元素设置样式 `overflow: hidden`

3. `margin` 合并问题：上下相邻的两个元素，上面的元素的下外边距和下面的元素的上外边距会合并，此时这两个盒子之间的距离为 `max{上边元素的下外边距, 下边元素的上外边距}`。这个问题无需解决。

内容溢出的处理

当我们在固定大小的 `div` 中填充过多的文本之后，就会产生溢出现象，具体可通过 `overflow` 属性处理。

属性名	作用	属性值
<code>overflow</code>	设置溢出内容的处理方式	<code>visible</code> 默认值，表示显示溢出内容 <code>hidden</code> 表示隐藏溢出内容 <code>scroll</code> 表示显示滚动条，无论内容是否溢出 <code>auto</code> 表示自动设置，当内容溢出时显示滚动条，不溢出则不显示
<code>overflow-x</code>	设置水平溢出内容的处理方式	同 <code>overflow</code>
<code>overflow-y</code>	设置垂直溢出内容的处理方式	同 <code>overflow</code>

- `overflow-x` 和 `overflow-y` 不可以一个是 `hidden` 一个是 `visible`
- `overflow-x` 和 `overflow-y` 都是实验性属性，不建议使用
- `overflow` 常用的属性值是 `hidden` 和 `auto`，除了可以处理溢出的显示方式之外，还可以解决许多疑难杂症（比如 `margin` 塌陷问题）

元素的隐藏

让元素隐藏有两种属性可以实现：`visibility` 和 `display`，二者的区别就是，`visibility` 隐藏元素后，元素看不见了，但是还会占有原来的位置；`display` 隐藏元素后，是彻彻底底地隐藏，不但看不见，也不占用任何位置。

属性名	作用	属性值
<code>visibility</code>	隐藏元素（仍占位）	<code>show</code> 默认值，表示显示元素 <code>hidden</code> 表示隐藏元素
<code>display</code>	隐藏元素（不占位）	<code>none</code> 表示隐藏元素 <code>block</code> 表示将元素作为块级元素显示 <code>inline</code> 表示将元素作为内联元素显示 <code>inline-block</code> 表示将元素作为行内块元素显示

样式的继承

1. 样式的继承：有些样式是元素可以从祖先元素那里继承的，如果元素本身设置了某个样式，就是用本身设置的样式；如果元素本身没有设置某个样式，就会优先继承离得近的祖先元素的对应样式。
2. 继承和不继承的 CSS 属性

继承的 CSS 属性	不继承的 CSS 属性
字体属性	边框、内边距、外边距、宽高
文本属性 (除了 <code>vertical-align</code>)	背景
文字颜色	溢出方式
...	...

- 能继承的属性，都是不影响布局的，即和盒子模型没有关系

元素的默认样式

- 我们可以通过开发者工具看到一个元素的默认样式、外部样式、行内样式、内联样式、继承样式等信息。

```

element.style {
    color: ■ aqua;          行内样式
}
a {
    font-family: "华文新魏";  内部样式
}
a {
    background-color: ■ pink; 外部样式
}
a:-webkit-any-link {
    color: -webkit-link;
    cursor: pointer;
    text-decoration: ▶ underline;
}                                     user agent stylesheet

Inherited from div
div {
    height: 400px; ①           继承样式
    width: 400px; ①           亮色的才表示是继承的样式;
    font-size: 30px;            暗色的表示是父元素的样式，但没有继承过来
    background-color: ■ gray;
}

Inherited from div
div {                                亮色划线表示该属性
    height: 400px; ①           从祖先元素继承来了,
    width: 400px; ①           但是被更高优先级的
    font-size: 30px;            同名属性的取值顶掉了
    background-color: ■ gray;
    color: ■ aqua;           ②
}

```

- 元素一般都有一些默认样式，从优先级的角度来考虑：**元素默认样式 > 元素继承的样式**，因此如果要修改元素的默认样式，一定要通过选择器直接选择该元素。

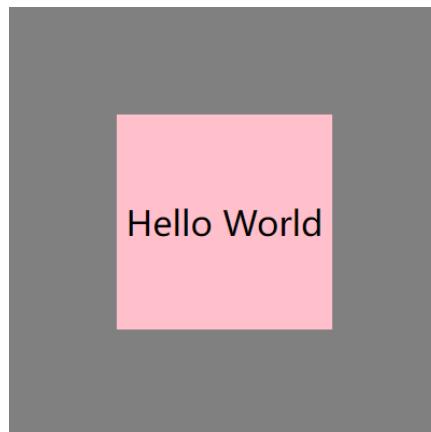
元素	默认样式
a	下划线；字体颜色；鼠标小手
h1 - h6	文字加粗、文字大小、上下外边距
p	上下外边距
ul、ol	左内边距
body	四周 8px 的外边距 (解释了为什么 body 没有铺满整个浏览器视口)

布局小技巧

16-三种布局练习

1. 三种布局效果的实现

- 效果一：内 `div` 在外 `div` 中居中；内 `div` 中的文字在内 `div` 中居中



```
<div class="outer">
    <div class="inner">Hello world</div>
</div>
```

```
.outer {
    width: 200px;
    height: 200px;

    background-color: gray;

    overflow: hidden;
}

.inner {
    width: 100px;
    height: 100px;

    background-color: pink;

    margin: 0 auto;
    /* 实现块元素在块元素中的水平居中 */
    margin-top: 50px;
    /* 实现块元素在块元素中的垂直居中（在父元素中搭配 overflow: hidden 使用） */

    text-align: center;
    /* 实现文本水平对齐 */
    line-height: 100px;
    /* 实现（单行）文本的垂直对齐 */
}
```

- 效果二：内 `span` 在外 `div` 中居中



Hello World

```
<div class="outer2">
    <span class="inner2">Hello world</span>
</div>
```

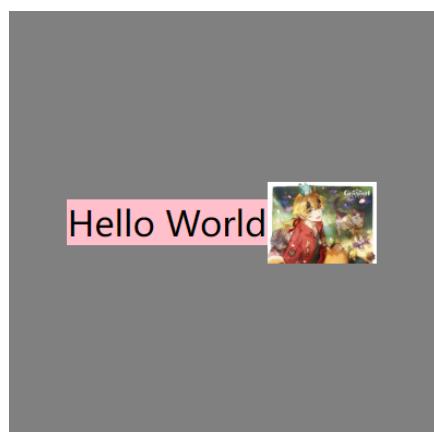
```
.outer2 {
    width: 200px;
    height: 200px;

    background-color: gray;

    text-align: center;
    /* 实现块元素中的行内元素水平居中 */
    line-height: 200px;
    /* 实现块元素中的行内元素（单行内容）垂直居中 */
}

.inner2 {
    background-color: pink;
}
```

- 效果三：span 和 img 在块元素 div 中居中



Hello World

```
<div class="outer3">
    <span class="inner3">Hello world</span>
</div>
```

```
.outer3 {
    width: 200px;
    height: 200px;
```

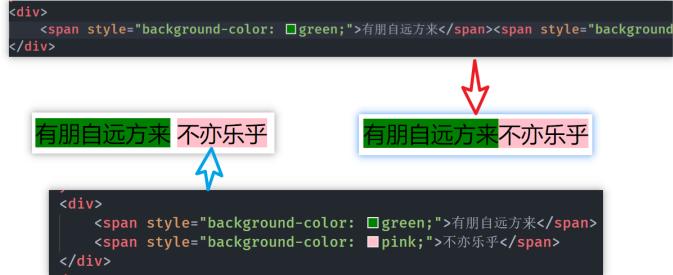
```
background-color: gray;  
  
font-size: 0px;  
  
text-align: center;  
line-height: 200px;  
  
}  
  
.inner3 {  
background-color: pink;  
vertical-align: middle;  
font-size: 16px;  
}  
  
img {  
width: 50px;  
vertical-align: middle;  
}
```

2. 布局技巧总结

- 行内元素、行内块元素可以被父元素当作文本处理
 - 可以像处理文本对齐一样，去处理行内、行内块元素的对齐
 - 可以在父元素中使用 `text-align: center` 让行内元素和行内块元素在父元素中水平居中
 - 可以在父元素中使用 `text-align=line-height` 让行内元素和行内块元素在父元素中垂直居中
 - 类似的，也可以在父元素中使用 `text-indent` 属性，给行内元素和行内块元素设置缩进
- 如何让子元素在父元素中水平居中
 - 子元素为块元素：子元素加上 `margin: 0 auto`
 - 子元素为行内元素或行内块元素：父元素加上 `text-align: center`
- 如何让子元素在父元素中垂直居中
 - 子元素为块元素：子元素加上 `margin-top: (父元素的 content - 子元素的盒子高度) ÷ 2`
 - 子元素为行内元素或行内块元素：让父元素的 `line-height` 等于其 `height`；每个子元素都加上 `vertical-align: middle`（如果想要绝对，垂直居中，需要设置父元素的 `font-size: 0`）

元素之间的空白问题

1. 元素之间的空白问题：行内元素、行内块元素，彼此之间的换行和空格会被浏览器解析为一个空白字符。



2. 空白解决方式

- 方式一：源代码中去除换行和空格（不推荐）
- 方式二：父元素中设置 `font-size: 0`，再在要显示文字的子元素中，单独设置字体大小（推荐）

行内块的幽灵空白问题

1. 行内块的幽灵空白问题：行内块元素与父元素的文本默认进行基线对齐，而文本的基线与文本最底部之间是有一定距离的。



2. 空白解决方式

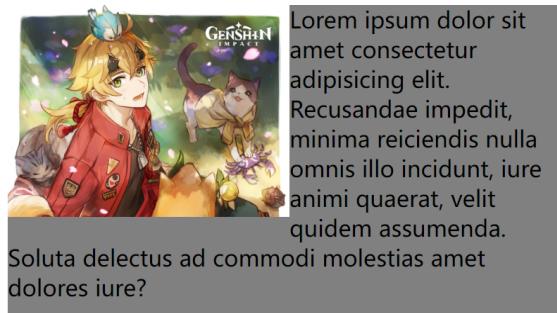
- 方式一：给行内块设置 `vertical-align=middle/bottom/top`（调整 `top` 文字会往上跑，调整 `bottom` 文字会往下跑，调整 `middle` 文字会往中间跑）
- 方式二：若果父元素只有一张照片，则设置图片 `display: block`（因为块级元素不涉及这种空白问题）
- 方式三：给父元素设置 `font-size: 0`，如果该行内还有文本，则需要单独设置 `font-size`

浮动

浮动的简介及特点

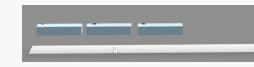
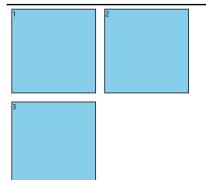
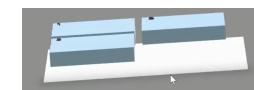
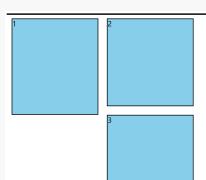
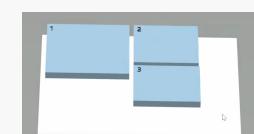
17-浮动小练习

1. 什么是浮动？浮动最初用于实现文字环绕图片效果，现在作为一种页面布局的方式，用于控制元素在页面上的位置。



2. 浮动怎么实现？通过 `float` 属性实现浮动：`left` 表示向左浮动，`right` 表示向右浮动，`none` 是默认值，表示不浮动。
3. **文档流**：HTML 中元素排列和布局的基本规则，如块级元素会独占一行或一块区域，从上至下排列，而行内元素则在同一行内按照它们的出现顺序排列。（可以理解文档流就是没有浮动的元素处于的那个层级、地面；浮动的元素相当于进入了一个更高的层级，physically）
4. 浮动的形象示例

描述	平面	三维
起初：三个块元素依次放置在文档流中		
第二个块元素浮动，此时其他两个块元素依次放置在文档流中		
第二个和第三个块元素浮动，第一个元素放置在文档流中		
起初：三个块元素（盒子）依次放置在一个块元素的父元素中，且都在文档流中		
盒子一右浮动		
盒子一左浮动		

描述	平面	三维
所有盒子左浮动		
所有盒子左浮动，盒子三落下来		
所有盒子左浮动，盒子三落下来但是被卡住		

元素浮动后的特点

1. 元素脱离文档流
2. 不管浮动前是什么元素，**浮动后默认宽与高都是被内容撑开**，同时也**可以设置宽高**
3. **不会独占一行**，可以与其他元素共用一行
4. 不会 `margin` 合并，也不会 `margin` 坍塌
5. **能够完美设置四个方向的 `margin` 与 `padding`****
6. 不会像行内块一样被当作文本处理

浮动产生的影响

1. 影响一（对兄弟元素的影响）：当元素浮动后，**后边的兄弟元素会占据浮动元素之前的位置**，处于浮动元素的下边（文档流中）；对前边的兄弟元素无影响。
2. 影响二（对父元素的影响）：**浮动元素无法撑起父元素的高度**，导致父元素高度塌陷；但是此时父元素的宽度依然束缚着浮动的元素。

浮动影响的解决方式

总共有五种方式解决浮动产生的影响，方案及相对的优缺点见下表。

序号	方案	限制
1	给父元素指定高度	可以解决父元素高度塌陷的问题（影响二）；但是没有解决影响一
2	给父元素也设置浮动	类同上；此时的副作用：父元素的兄弟元素会顶上来
3	给父元素设置 <code>overflow: hidden</code>	可以解决影响一和影响二；但是：要求全部兄弟元素都是浮动的
4	在所有浮动元素的最后边，添加一个块级元素，并给该块级元素设置 <code>clear: both</code>	可以解决影响一和影响二；但是：要求这个新加的块级元素前边的元素都是浮动元素，且这个新加的块级元素不能是浮动元素，且这个新加的块级元素是块元素或行内块元素

序号	方案	限制
5	给浮动元素的父元素设置伪元素，通过伪元素清除浮动（推荐）	同上，方案五是对方案四的改进

- 方案五设置伪元素的代码如下

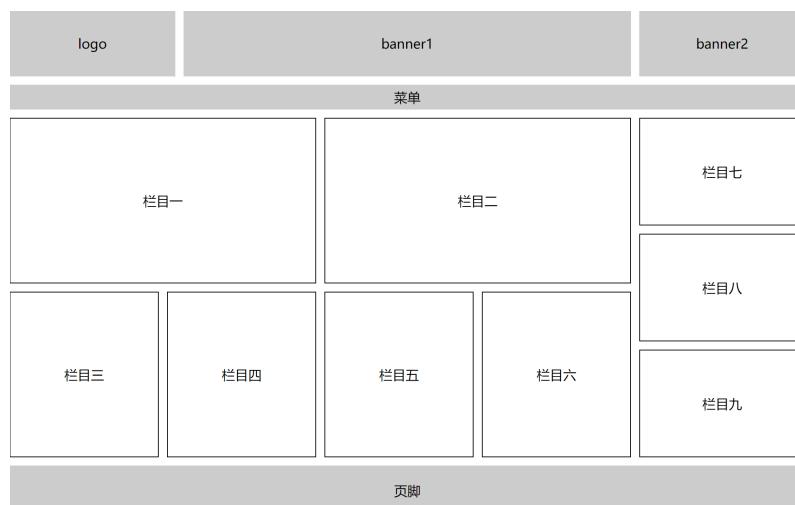
```
.parent::after {
    content: "";
    display: block;
    clear: both
}
```

- 布局原则：**设置浮动的时候，兄弟元素要么全都浮动，要么全都不浮动

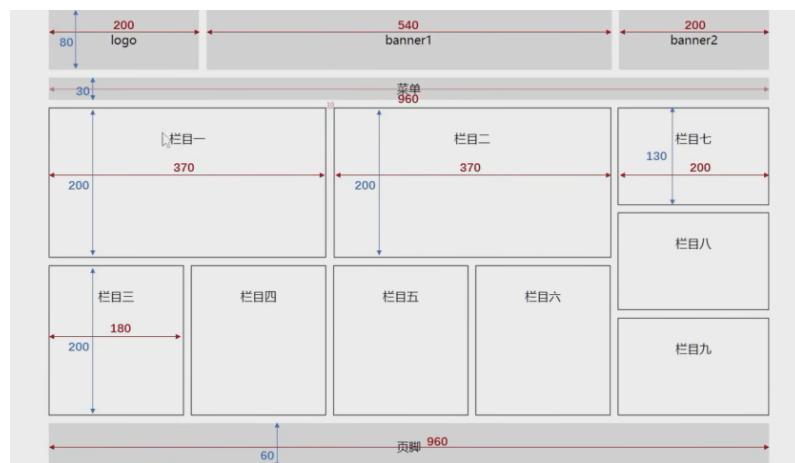
浮动布局练习

18-浮动布局练习

1. 要求效果

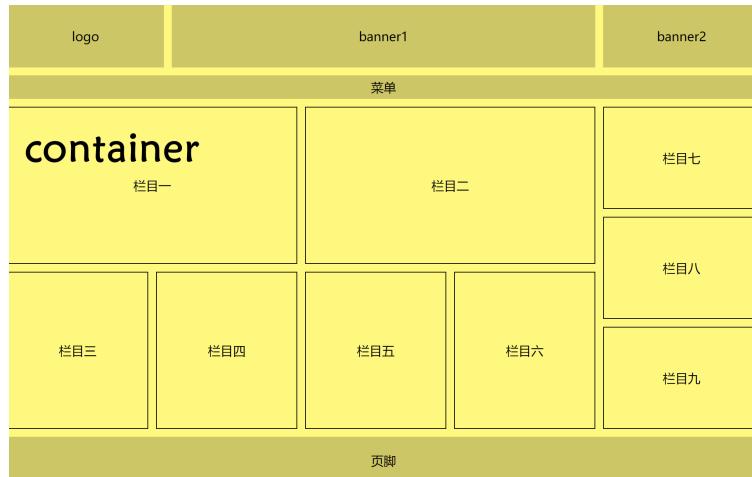


2. 数据标注

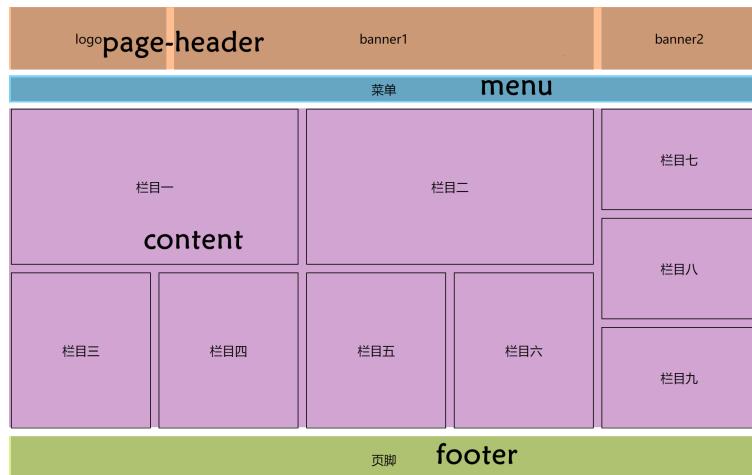


3. 问题分析

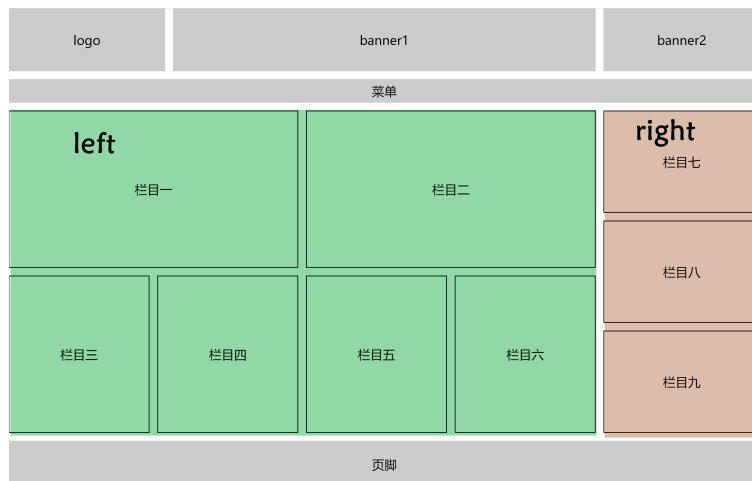
- o container (页面的祖先元素：设置整个页面的最大宽度、居中整个页面、默认文本居中)



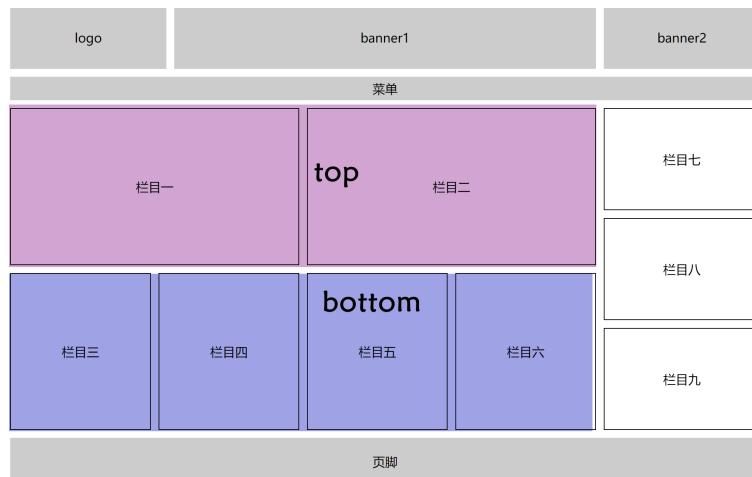
- page-header、menu、content、footer



- left、right



- top、bottom

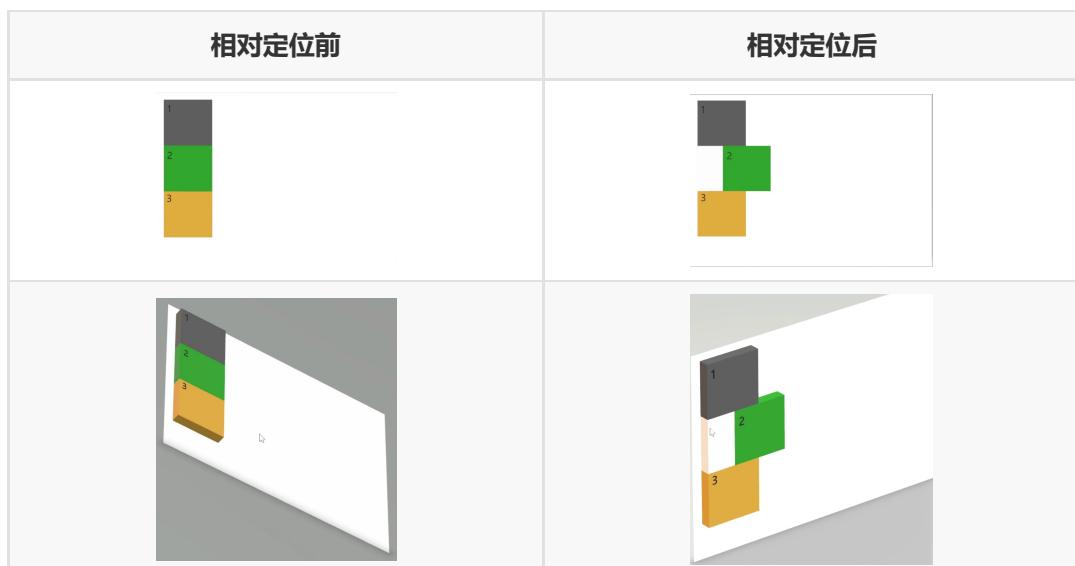


- 其中，header 里边的三个块元素、content 里边的 left 和 right、top 里边的两个块元素、bottom 里边的四个块元素都是浮动元素；注意要在浮动元素的块元素上进行浮动影响的消除

定位 (day7 | 139p: 53.3%)

相对定位

- 如何设置相对定位？通过给元素设置 `position: relative` 即可以实现相对定位。 (step1)
- 如何利用相对定位调整位置？通过给元素设置 `left`、`right`、`top`、`bottom` 四个属性调整位置，属性取值为长度。 (step2)
- 相对定位：**指的是元素相对于自己原来的位置改变位置。
- 相对定位的特点
 - 相对定位的元素**不会脱离文档流**，元素位置的变化，只是视觉效果上的变化，不会对其他元素产生任何影响



- 相对定位的元素的**显示层级**比普通元素高；各种定位的元素的显示层级都是一样的
 - 定位的元素会覆盖在普通元素之上
 - 如果两个元素都发生定位，则后定位的元素会覆盖在先定位的元素之上
 - 注：区分浮动和定位浮动的元素脱离文档流，但是定位的元素没有脱离文档流
- `left` 和 `right` 属性不能一起设置；`top` 和 `bottom` 属性不能一起设置
- 相对定位的元素，也能继续浮动，也能通过 `margin` 调整位置，但是**不推荐**这样做

5. 相对定位的应用场景

- 对元素位置进行微调
- 和绝对定位配合使用

绝对定位

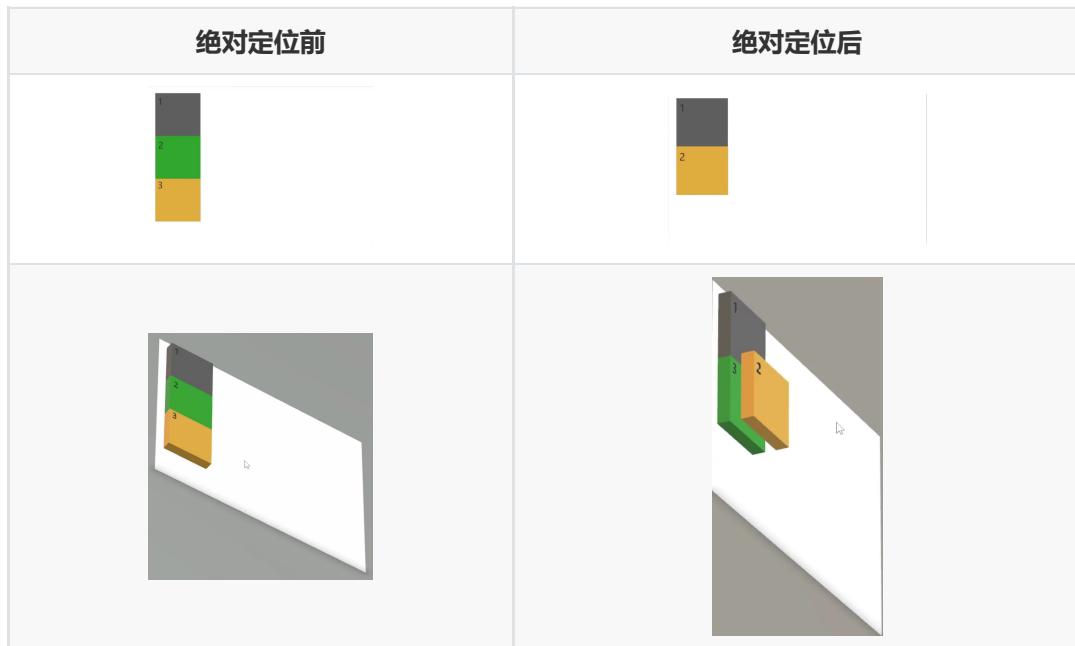
- 如何设置绝对定位？通过给元素设置 `position: absolute` 即可以实现绝对定位。 (step1)
- 如何利用绝对定位调整位置？通过给元素设置 `left`、`right`、`top`、`bottom` 四个属性调整位置，属性取值为长度。 (step2)
- 绝对定位：**指的是元素**参考自己的包含块**来改变位置。

包含块

- 如果元素没有脱离文档流，**父元素**就是它的包含块
- 如果元素脱离了文档流，**第一个拥有定位属性的祖先元素**就是它的包含块；如果所有祖先元素都没定位，那么包含块就是整个页面

4. 绝对定位的特点

- 绝对定位的元素**脱离文档流**，对后面的兄弟元素、父元素都有影响



- `left` 和 `right` 属性不能一起设置；`top` 和 `bottom` 属性不能一起设置
- 绝对定位和浮动**不能同时设置**，如果同时设置，浮动失效，**以定位为主**
- 绝对定位的元素，也可以通过 `margin` 调整位置，但是**不推荐这样做**；同时，只有设置了 `top` 属性，此时 `margin-top` 属性才生效，其他三个方向的 `margin` 类似
- 无论是什么元素（行内、行内块、块级）设置为绝对定位后，都变成了**定位元素**

定位元素：默认宽、高都被内容撑开；但可以自由设置宽高

例如：`span` 变成定位元素后，也可以设置宽高

- 绝对定位和相对定位配合使用：即给父元素设置相对定位，此时子元素绝对定位，父元素就是子元素的包含块

5. 绝对定位的应用场景：将一个元素盖到另外一个元素身上，如京东的分类菜单栏。



The screenshot shows the JD.com mobile website's navigation bar at the top. Below it, a vertical category list is displayed as an absolute-positioned element. The categories include '家用电器', '手机/运营商/数码', '电脑/办公', '家居/家具/家装/厨具', '男装/女装/童装/内衣', '美妆/个护清洁/宠物', '女鞋/箱包/钟表/珠宝', '男鞋/运动/户外', '房产/汽车/汽车用品', '母婴/玩具乐器', '食品/酒类/生鲜/特产', '艺术/礼品鲜花/农资绿植', '医药保健/计生情趣', '图书/文娱/教育/电子书', '机票/酒店/旅游/生活', '众筹/白条/保险/企业金融', '安装/维修/清洗/二手', and '工业品'. This list is positioned over the main content area, demonstrating the effect of absolute positioning.

```
/* 上述功能的一个简单实现 */
.parent {
    position: relative;
}

.child {
    position: absolute;
    transition: 1s all linear;
}

.parent:hover .child {
    left: 220px;
}
```

固定定位

1. 如何设置固定定位？通过给元素设置 `position: fixed` 即可以实现固定定位。 (step1)
2. 如何利用固定定位调整位置？通过给元素设置 `left`、`right`、`top`、`bottom` 四个属性调整位置，属性取值为长度。 (step2)
3. **固定定位**：指的是元素**参考自己的视口**来改变位置。
 - 视口 (viewport)：对于浏览器，视口就是网页内容在浏览器中可见的区域大小。
4. 固定定位的特点：和绝对定位完全相同。
5. 固定定位的应用场景：网页小广告。

粘性定位

1. 如何设置粘性定位？通过给元素设置 `position: sticky` 即可以实现粘性定位。 (step1)
2. 如何利用粘性定位调整位置？通过给元素设置 `left`、`right`、`top`、`bottom` 四个属性调整滚动时粘住的位置，属性取值为长度。 (step2) (最常用是 `top` 属性)
3. 粘性定位：指的是元素**参考离它最近的一个拥有“滚动机制”的祖先元素** (即使这个祖先不是最近的真实可滚动的祖先) 来粘住不动。

什么叫“即使这个祖先不是最近的真实可滚动的祖先”？例如：在开启了粘性定位元素的父元素中设置了 `overflow: scroll`，此时该子元素的参考位置就是它的父元素，因为该父元素拥有所谓的滚动机制，哪怕不能滚动（即不是真实可滚动的）

4. 粘性定位的特点
 - 粘性定位的元素**不会脱离文档流**，是一种专门用于窗口滚动时的新的定位方式
 - 粘性定位最常用的是 `top` 属性
 - 粘性定位的元素，也能继续浮动，也能通过 `margin` 调整位置，但是**不推荐**这样做
 - 粘性定位和相对定位的特点基本一致，不同点在于，粘性定位可以在元素到达某个位置时将其固定，并且在其父元素离开对应位置时一起离开。

定位的层级

[19-定位的层级](#)

1. 定位元素的显示层级比普通元素高
2. 定位元素（无论什么定位）的显示层级都是一样的
3. 如果多个元素都是定位元素，且发生重叠，那么后面的元素会显示在前面的元素之上
4. CSS 的 `z-index` 属性可以调整定位元素的显示层级
 - `z-index` 的属性值是没有单位的数字，值越大，显示层级越高
 - 只有定位元素设置 `z-index` 元素才有效
 - 如果一个 `z-index` 大的定位元素，没有覆盖掉 `z-index` 小的定位元素，那么大概是因为这个 `z-index` 大的定位元素的父元素的缘故，因为父元素的 `z-index` 会影响其子元素的显示层级

定位的一个注意点

[20-绝对定位注意点](#)

绝对定位的元素是越过其包含块的 `padding` 的，即当绝对定位的元素脱离文档流后，是在其包含块的 `padding` + `content` 范围内布局的。



```
<div class="outer">
  <div class="inner">
  </div>
</div>
```

```
.outer {
  width: 400px;
  height: 200px;
  background-color: skyblue;
  padding: 50px;
  border: 5px black solid;
  position: relative;
}

.inner {
  width: 100px;
  height: 100px;
  background-color: pink;
  position: absolute;
  left: 0px;
```

```
    top: 0px;
    border: 5px red dotted;
}
```

定位的特殊应用

[21-定位的特殊应用1](#)

[22-定位的特殊应用1](#)

1. 注意

- 发生**固定定位、绝对定位**后，元素都变成了**定位元素**：默认宽高被内容撑开，且依然可以设置宽高
- 发生相对定位之后，元素依然是之前的显示模式（元素在文档流中的表现方式或布局方式，如块级元素、行内元素、行内块元素等）
- 以下所说的特殊应用，只针对**绝对定位和固定定位**的元素，不包括相对定位的元素。

2. 特殊应用一：让（没有设置宽高的）定位元素充满包含块

- 定位元素的宽充满包含块 `left: 0; right: 0;`
- 定位元素的高充满包含块 `top: 0; bottom: 0;`
- 定位元素充满包含块 `left: 0; right: 0; top: 0; bottom: 0;`

3. 特殊应用二：让（设置了宽高的）定位元素在包含块中居中

- 方式一（推荐）：设置 `left`、`right`、`margin` 可以水平居中；设置 `top`、`bottom`、`margin` 可以垂直居中；都设置则水平垂直居中

```
left: 0;
right: 0;
top: 0;
bottom: 0;
margin: auto;
```

- 方式二

```
top: 50%;
left: 50%;
margin-left: -(定位元素盒子宽的一半);
margin-top: -(定位元素盒子高的一半);
```

注意：笔记中说盒子模型是包含外边距、边框、内边距、内容区的；如果只说盒子，则一般而言是在盒子模型的基础上，略去外边距。这里的**定位元素盒子的宽=内容区 + 内边距 + 边框**。

布局

版心

- 网页的版心（container）是一个**固定宽度且水平居中**的盒子，用于显示网页的主要内容
- 版心的宽度一般是 960~1200 像素之间
- 版心可以是一个，也可以是多个

常用类名

类名	含义
<code>topbar</code>	顶部导航条
<code>header</code> 、 <code>page-header</code>	页头
<code>nav</code> 、 <code>navigator</code> 、 <code>navbar</code>	导航
<code>search</code> 、 <code>search-box</code>	搜索框
<code>banner</code>	横幅、广告、宣传图
<code>content</code> 、 <code>main</code>	主要内容
<code>aside</code> 、 <code>sidebar</code>	侧边栏
<code>footer</code> 、 <code>page-footer</code>	页脚

重置默认样式

1. 为什么要重置默认样式？现在的网页设计更加复杂，默认样式会给页面绘制带来麻烦；默认样式在不同浏览器上呈现的效果不一样。
2. 重置默认样式的方案一：全局选择器
 - 开发中不会使用，可以在写一些 demo 时使用
 - 为什么不会使用？`*` 选择的是所有元素，但是不是所有的元素都有默认样式；我们更希望重置默认样式时，做一些特殊处理，如更改 `a` 的颜色等
3. 重置默认样式的方案二：reset.css
 - 这种方式可以选择到具有默认样式的元素，**清空**其默认样式，此时的网页更像是“一张白纸”
4. 重置默认样式的方案三：normalized.css
 - 这种方式在清楚默认样式的基础上，**保留了一些有价值的默认样式**
 - 官网：<http://necolas.github.io/normalize.css>
 - 这种方式，相较于 reset.css，有以下优点
 - 保留了有价值的默认样式，更加温和，而不是完全去掉
 - 为大部分 HTML 元素提供了一般化样式
 - 新增了对 HTML5 元素的设置
 - 对并集选择器的使用比较谨慎，可以有效避免调试工具杂乱

PART4 尚品汇

[尚品汇](#)

```

<!-- 引入页签图标 -->
<link rel="shortcut icon" href="../../favicon.ico" type="image/x-icon">
<!-- 引入重置样式 -->
<link rel="stylesheet" href="./css/reset.css">
<!-- 引入样式 -->
<link rel="stylesheet" href="./css/index.css">

```

顶部导航条

尚品汇欢迎您 | [请登录](#) | [免费注册](#) | [我的订单](#) | [我的购物车](#) | [尚品汇会员](#) | [企业采购](#) | [关注尚品汇](#) | [合作招商](#) | [商家后台](#)

- 首先用一个大的盒子，并放置一个版心
 - 版心分为左侧的欢迎区和右侧的导航区，分别浮动于版心的左边和右边
 - 欢迎区通过 `span` 和 `a` 的组合实现
 - 导航区通过一个无序列表实现，其中无序列表的每一项就是对应的导航里的每一项
 - 将所有列表项左浮动实现一行的导航内容

头部


 搜索

- 首先用一个大的盒子，并放置一个版心
 - 版心分为左侧的 **logo 区**和右侧的**搜索区**，分别浮动于版心的左边和右边
 - logo 区域简单加一个图片即可
 - 搜索区通过一个**表单**实现，包含一个**输入框**和一个**按钮控件**

主导航

[全部商品分类](#)

[尚品超市](#) [优惠券](#) [买啥](#) [尚品家电](#) [尚品生鲜](#) [PLUS会员](#) [进口好物](#) [品牌闪购](#) [拍卖](#) [五金商店](#)

- 首先用一个大的盒子，并放置一个版心
 - 版心分为左侧的**文字区**和右侧的**菜单区**，依次都浮动于版心的左边
 - 文字区可以通过一个 `div` 简单实现
 - 菜单区可以通过**无序列表**实现，然后将无序列表的全部子元素左浮

内容区 (day8 | 152p: 61.4%)

手机/运营商/数码

电脑/办公

家具/家居/家装/厨具

男装/女装/童装/内衣

美妆/个护清洁/宠物

女鞋/箱包/钟表/珠宝

男鞋/运动/户外

房产/汽车/汽车用品

母婴/玩具乐器

食品/酒类/生鲜/特产

艺术/礼品鲜花/农资绿植

医药保健/计生情趣

图书/文娱/教育/电子书

机票/酒店/旅游/生活

众筹/白条/保险/企业金融

安装/维修/清洗/二手

首发

4键有线机械键盘

键帽 京造月影黄轴

RE

NEW -ARRIVAL

键帽

人体工学设计

键线分离

尚品快报

[特惠] 毛衣+直筒裤，才是YYDS。

[特惠] 超大容量的满足感来啥子于。

[特惠] 每日一件，衣柜满到塞不下。

[特惠] 媳妇儿再买，我就疯给他看。

话费

火车票

加油卡

白条

电影票

酒店

理财

机票

礼品卡

彩票

游戏

众筹

- 首先用一个大的盒子，并放置一个版心
 - 版心分为左侧的**导航区**、中间的**条幅区**和右侧的**其他区**，依次浮动于版心的左边
 - 侧边导航可以通过一个**无序列表**实现，每个列表元素包含**超链接**和对应的**二级菜单**
 - 二级菜单初始隐藏，当鼠标悬浮在对应的列表项区域，二级菜单通过绝对定位显示到正确的地方
 - 条幅区可以通过一个 `div` 简单实现
 - 其他区分为上边的**信息区**和下方的其他**导航区**
 - 信息区域可以通过一个简单的 `div` 和**无序列表**实现
 - 导航区可以通过三个**无序列表**实现

秒杀区



楼层

家用电器		热门 大家电 生活电器 厨房电器 应季电器 空气/净水 高端电器			
节能补贴	4K电视	全球购 食品水饮 满199减20 		每满200减20元 烘焙节1元抢购	8.20家电宅购节 415L双开门3799 
空气净化器	IH电饭煲			每满200减20元 烘焙节1元抢购	每满200减20元 烘焙节1元抢购 
滚筒洗衣机	电热水器				每满200减20元 烘焙节1元抢购 
三星曲面电视 抽奖送好礼 					

- 首先用一个大的盒子，并放置一个版心
 - 版心分为上边的**导航区**和下边的**信息区**
 - 导航区通过 `span` 左浮和 `ul` 右浮实现
 - 信息区的五个大 `div` 通过左浮实现

页脚



- 首先用一个大的盒子，并放置一个**版心**
 - 版心分为上边的**菜单区**、中间的**横线**和下边的**菜单区**
 - 上边的菜单区可以通过 `ul` 联合浮动实现
 - 下边的菜单区同理
 - 中间的横线可以通过 `div` 实现

PART5 HTML5

简介

1. 概述：HTML5 是新一代的 HTML 标准，2014 年 10 月由万维网（W3C）完成标准制定。

广义上，HTML5 指的是整个前端；狭义上，HTML5 指的是新一代的 HTML 标准

2. 优点

- 针对 JavaScript，新增了很多**可操作的接口**
- 新增了一些**语义化标签、全局属性**
- 新增了**多媒体标签**，很好替代 flash
- 更加侧重**语义化**，对 SEO 更友好
- 可移植性好，可以大量应用在移动设备上
- 支持 Chrome、Safari、Opera、Firefox 等主流浏览器

IE 浏览器必须是 9 以上的版本才支持 HTML5，且 IE9 仅支持部分 HTML5 新特性

新增布局标签

标签名	语义	单/双
<code>header</code>	整个页面或部分区域的 头部	双
<code>footer</code>	整个页面或部分页面的 底部	双
<code>nav</code>	导航	双
<code>article</code>	文章、帖子、杂志、新闻、博客、评论等	双
<code>section</code>	页面或文章中的某段文字（通常包含标题）	双

标签名	语义	单/双
<code>aside</code>	侧边栏	双
<code>main</code>	文档的主要内容 (WHATWG 没语义; IE 不支持; 几乎不用)	双
<code>hgroup</code>	包裹连续的标题 (如文章的主标题、副标题的组合; 被 W3C 删除)	双

- 这些标签没有什么特殊效果，强调的是其语义，相当于有语义的 `div`
- `article` 与 `section` 有什么区别？
 - `article` 里面可以有多个 `section`
 - `section` 强调的是分段或分块，如果想将一块内容分成几段，可以使用 `section` 元素
 - `article` 比 `section` 更强调独立性，一块内容如果比较独立、完整，则应该使用 `article` 元素

```
<article>
  <h2>xxxxxxxxxxxxxx</h2>
  <section>
    <h3>xxxxxxxxxxxxxx</h3>
    <p>xxxxxxxxxxxxxx</p>
  </section>
  <section>
    <h3>xxxxxxxxxxxxxx</h3>
    <p>xxxxxxxxxxxxxx</p>
  </section>
</article>
```

```
<aside>
  <nav>
    <ul>
      <li><a href="#">xxxxxxxxxx</a></li>
      <li><a href="#">xxxxxxxxxx</a></li>
      <li><a href="#">xxxxxxxxxx</a></li>
    </ul>
  </nav>
</aside>
```

新增状态标签

23-h5状态标签

标签	语义	单/双
<code>meter</code>	定义已知范围内的标量测量	双
<code>progress</code>	显示某个任务完成的进度的指示器	双

1. 第一种状态：`meter` 标签，又被称为 `gauge`（尺度），常用于表示电量、磁盘用量等。

属性	作用	取值
high	高值	数值
low	低值	数值
max	最大值	数值
min	最小值	数值
optimum	最优化	数值
value	当前值	数值

- 使用情形一 (两状态: 安全、警告)

```
<meter max="100" min="0" value="90" low="10" high="20" optimum="15">
</meter>
```

- 当 $value \in [low, high]$, 表示安全状态, 此时显示绿色; 否则表示警告状态, 此时显示黄色

- 使用情形二 (两状态: 安全、警告、危险)

```
<meter max="100" min="0" value="90" low="10" high="20" optimum="90">
</meter>
```

- 当 $value \in [0, low]$, 表示危险状态, 此时显示红色;
- 当 $value \in [low, high]$, 表示危险状态, 此时显示黄色;
- 当 $value \in [high, max]$, 表示安全状态, 此时显示绿色

- 情形一和情形二的区别在于 `optimum` 所在空间的不同, 其所在空间永远是安全的绿色区间
- 使用情形二就是手机电量的颜色的显示逻辑

2. 第一种状态: `progress` 标签, 一般用于表示进度条, 如**工作完成进度**等。

属性	作用	取值
max	目标值	数值
value	当前值	数值

```
<progress max="100" value="70"></progress>
```

新增列表标签

24-h5列表标签

标签	语义	单/双
<code>datalist</code>	搜索框的关键字提示	双
<code>details</code>	展示问题和答案, 或对专有名词进行解释	双

标签	语义	单/双
summary	写在 details 里边，用于指定问题或专有名词	双

1. 第一种列表： `datalist` & `option` 标签，需要将表单的 `list` 属性和 `datalist` 的 `id` 配合使用 (`list = id`) 。

```
<form action="#">
  <input type="text" list="my-data">
  <button>Search</button>
</form>
<datalist id="my-data">
  <option value="周杰伦">周杰伦</option>
  <option value="周冬雨">周冬雨</option>
  <option value="马冬梅">马冬梅</option>
  <option value="温兆伦">温兆伦</option>
</datalist>
```

2. 第二种列表： `details` & `summary` 标签。

```
<details>
  <summary>A 开头的单词有哪些? </summary>
  <p>Abandon</p>
</details>
```

新增文本标签

25-h5文本标签

标签	语义	单/双
ruby	包裹需要注音的文字	双
rt	写在 ruby 里边，是注音内容	双
mark	标记关键字	双

1. 第一种文本（文本注音）：`ruby` & `rt` 标签， `rt` 里的注音对 `ruby` 的文字进行标注。

```
<ruby>
  <span>你好世界</span>
  <rt>ni hao shi jie</rt>
</ruby>
```

2. 第二种文本（文本标记）：`mark` 标签，用于标记搜索结果中的关键字。

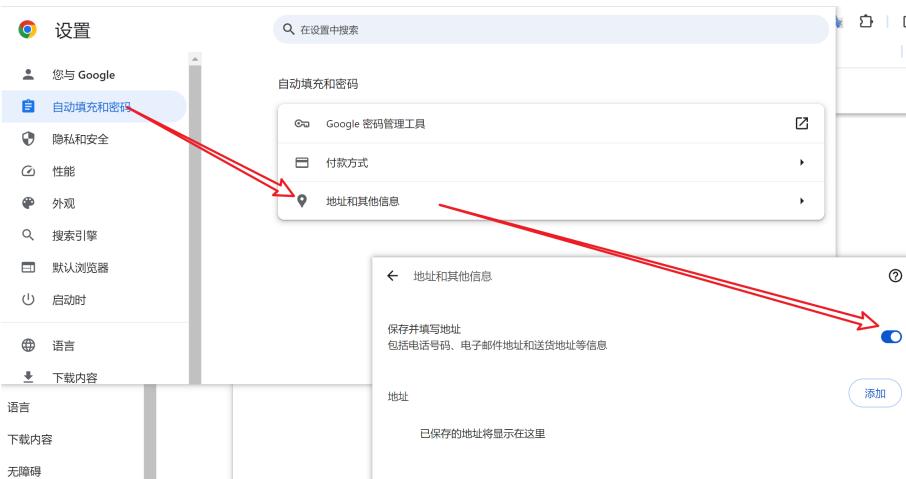
```
<p>Lorem ipsum dolor sit <mark>amet</mark>, consectetur adipisicing elit.
Accusamus reiciendis fuga incidunt nisi fugit ad ut saepe at a repellendus.
</p>
```

新增表单控件属性

26-h5表单控件属性

属性名	作用
<code>placeholder</code>	提示文字 (适用 文字输入类 的表单控件)
<code>required</code>	必填项 (适用 除按钮外其他表单控件)
<code>autofocus</code>	自动获取焦点 (适用 所有表单控件)
<code>autocomplete</code>	自动完成 (适用 文字输入类表单控件 , 密码输入框、多行输入框不可用; 可选值 <code>on</code> 或 <code>off</code>)
<code>pattern</code>	正则表达式 (适用 文本输入类表单控件 , 多行输入框不可用)

- `pattern` 对于**空的输入框不会验证**, 因而往往与 `required` 配合使用
- `required` 对于单选框, 表示必须选择一个; 对于复选框, 表示必须至少选择这个
- `autocomplete=on` 属性的正确使用, 可能需要打开浏览器中的以下设置



新增 input 标签 type 属性值

27-h5input-type属性值

下表为 `input` 标签的 `type` 属性 HTML5 中的新增属性值, 用于实现不同功能。

属性值	作用
<code>email</code>	邮箱 类型输入框 (表单提交时会验证格式, 输入为空时不验证格式)
<code>url</code>	url 类型的输入框 (表单提交时会验证格式, 输入为空时不验证格式)
<code>number</code>	数字 类型的输入框 (表单提交时会验证格式, 输入为空时不验证格式; 还可以设置 <code>step</code> 、 <code>min</code> 、 <code>max</code> 属性, 意思是步长、最低值、最大值)
<code>search</code>	搜索 类型的输入框 (表单提交时 不会 验证格式)
<code>tel</code>	电话 类型的输入框 (表单提交时 不会 验证格式, 移动端输入时会唤起数字键盘)

属性值	作用
range	范围选择框（默认值为 50，表单提交时不会验证格式；还可以设置 value、min、max 属性，意思是初始值、最低值、最大值）
color	颜色选择框（默认值为黑色，表单提交时不会验证格式）
date	日期选择框（默认值为空，表单提交时不会验证格式）
month	月份选择框（默认值为空，表单提交时不会验证格式）
week	周选择框（默认值为空，表单提交时不会验证格式）
time	时间选择框（默认值为空，表单提交时不会验证格式）
datetime-local	时间日期选择框（默认值为空，表单提交时不会验证格式）

新增 form 标签属性

属性值	作用
novalidate	表单提交时不再进行验证，此时所有的校验规则失效

```
<form action="#" novalidate></form>
```

新增音视频标签

标签	语义	双/单
video	视频	双
audio	音频	双

1. 视频： video 标签

属性名	作用	属性值
src	视频地址	url 地址
width	视频播放器的宽度	像素值
height	视频播放器的高度	像素值
controls	向用户显示视频控件（播放/暂停按钮等）	无
muted	静音	无
autoplay	自动播放	无
loop	循环播放	无
poster	视频封面	url 地址

属性名	作用	属性值
preload	视频预加载 (当设置 <code>autoplay</code> 属性时, 自动忽略 <code>preload</code> 属性)	<code>none</code> 不预加载视频 <code>metadata</code> 仅预先获取视频的元数据 <code>auto</code> 下载整个视频文件 (即使用户不希望使用)

- 只有静音 (`muted`) 的视频, 自动播放 (`autoplay`) 才会生效, 这是浏览器对用户的一个保护
- 有时候非静音视频也可以自动播放, 这涉及到浏览器的“媒体参与度”这一策略 (如 `chrome://media-engagement/`), 只有得分较高的网站允许非静音视频的自动播放

Origin	Sessions	Sessions with playback	Last Playback	Is High	Score ▾
https://www.bilibili.com	183	109	2024-03-18T11:07:44.252Z	Yes	0.60
https://www.youtube.com	12	5	2024-03-14T15:08:31.234Z	No	0.25
https://twitter.com	98	10	2024-03-09T17:25:38.891Z	No	0.10
https://act.mihoyo.com	6	2	2024-01-22T14:20:43.216Z	No	0.10
https://s.weibo.com	15	1	2024-03-16T02:39:15.659Z	No	0.05
https://tv.cctv.com	1	1	2024-03-12T13:25:29.947Z	No	0.05
https://www.ted.com	2	1	2024-02-20T08:55:29.222Z	No	0.05
https://www.xvideos.com	6	1	2024-03-09T17:25:48.119Z	No	0.05
https://ys.mihoyo.com	10	1	2024-01-18T11:13:33.274Z	No	0.05

```
<video src="xxx.mp4" controls muted loop poster="xxx.png" preload="auto">
</video>
```

2. 音频: `audio` 标签

属性名	作用	属性值
<code>src</code>	音频地址	url 地址
<code>controls</code>	向用户显示音频控件 (播放/暂停按钮等)	无
<code>muted</code>	静音	无
<code>autoplay</code>	自动播放	无
<code>loop</code>	循环播放	无
<code>preload</code>	音频预加载 (当设置 <code>autoplay</code> 属性时, 自动忽略 <code>preload</code> 属性)	<code>none</code> 不预加载音频 <code>metadata</code> 仅预先获取音频的元数据 <code>auto</code> 下载整个音频文件 (即使用户不希望使用)

```
<audio src="xxx.mp3" controls loop preload="auto"></audio>
```

新增全局属性

属性名	作用	属性值
<code>contenteditable</code>	表示元素是否可以被用户编辑	<code>true</code> : 可编辑 <code>false</code> : 不可编辑
<code>draggable</code>	表示元素是否可以被拖动	<code>true</code> : 可拖动 <code>false</code> : 不可拖动
<code>hidden</code>	隐藏元素 (和 <code>display: none</code> 作用一样)	无
<code>spellcheck</code>	是否对元素进行拼写和语法检查	<code>true</code> : 检查 <code>false</code> : 不检查
<code>contextmenu</code>	规定元素的上下文菜单，用户鼠标右键点击元素时显示	
<code>data-*</code>	用于存储页面的私有定制数据	

- `spellcheck` 属性的正确使用，可能需要打开浏览器中的以下设置



兼容性处理

为了解决旧版本浏览器（如 IE8）对新 HTML5 标准不兼容的问题，我们主要有以下几种策略。

1. 添加元信息，使浏览器处于最优渲染模式

```
<!-- 设置 IE 总是使用最新的文档模式进行渲染 -->
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

```
<!-- 优先使用 webkit (Chromium) 内核进行渲染，针对 360 等双核浏览器 -->
<meta name="renderer" content="webkit">
```

2. 使用 `html5shiv.css` 让低版本浏览器认识 H5 的语义化标签（以下是特殊语法，表示如果 IE 版本小于 9，则引入 `.css` 文件）

```
<!--[if lt ie 9]>
<script src="./html5shiv.js"></script>
<![endif]-->
```

- 上述语法举例

- <!--[if IE 8]>仅IE8可见<![endif]-->
- <!--[if gt IE 8]>仅IE8以上可见<![endif]-->
- <!--[if lt IE 8]>仅IE8以下可见<![endif]-->
- <!--[if gte IE 8]>仅IE8及以上可见<![endif]-->
- <!--[if lte IE 8]>仅IE8及以下可见<![endif]-->
- <!--[if !IE 8]>非IE8可见<![endif]-->

PART6 CSS3 (day9 | 171p: 73.1%)

简介

1. CSS3 概述：CSS3 是 CSS2 的升级版本，在 CSS2 的基础上，新增了许多强大的新功能，在未来会按照模块化的方式去发展。

2. CSS3 新特性

- 更加实用的选择器：动态伪类、目标伪类、伪元素选择器等
- 更好的视觉效果：圆角、阴影、渐变等
- 丰富的背景效果：支持多个背景图片、新增背景相关属性
- 全新的布局方案：弹性盒子
- 新增了 web 字体：可以显示用户电脑上未安装的字体
- 增强颜色：HSL、HSLA、RGBA 颜色模式，opacity 控制不透明度
- 增加 2D/3D 变换：旋转、扭曲、缩放、位移等
- 增加动画与过渡效果
-

3. CSS3 私有前缀

- 是什么？-webkit-border-radius: 20px 中的 -webkit- 就是一个私有前缀
- 为什么？当 W3C 标准提出某个 CSS 特性后，浏览器厂商会先对该特性进行测试，测试通过后再正式支持该 CSS 特性；测试阶段，浏览器会使用私有前缀来测试该 CSS 特性，即在对应的属性名前加上对应的私有前缀，不同的浏览器内核对应的私有前缀不同；例如，在测试阶段，浏览器只会识别如 -webkit-border-radius 这样的属性，而不会支持 border-radius 这样的属性

浏览器	私有前缀
Chrome	-webkit-
Safari	-webkit-
Firefox	-moz-

浏览器	私有前缀
Edge	-webkit-
旧Opera	-o-
旧IE	ms

◦ 怎知道？通过 <https://caniuse.com/> 查询对应 CSS3 特性的兼容性

◦ 注意

- HTML5 基本 IE9 及以上及其他浏览器都是支持的，但是 CSS3 可能有部分模块有些浏览器不支持
- 前缀不一定和浏览器使用的内核一定对应
- 可以通过下述方式使用一些测试属性，如果测试阶段，自动识别第一行代码，忽略第二行；如果正式支持，忽略第一行代码，自动识别第二行

```
-webkit-border-radius: 20px;
border-radius: 20px;
```

- 常用的 CSS3 新特性，主流浏览器都是支持的
- 可以利用现代构建工具（如 webpack）去帮助添加私有前缀

新增长度单位

长度单位	含义
rem	根元素字体大小的倍数（只与根元素字体大小有关）
vw	视口宽度的百分之几
vh	视口高度的百分之几
vmax	max{视口高度，视口宽度} 的百分之几
vmin	min{视口高度，视口宽度} 的百分之几

- vw 在移动端开发中使用较多

新增颜色设置方式

CSS3 中新增的三种颜色设置方式为：rgba、hsb、hsba

新增选择器

CSS3 中新增的选择器有：动态伪类、目标伪类、语言伪类、UI 伪类、结构伪类、否定伪类、伪元素选择器

新增盒子模型属性

[28-c3盒子模型属性](#)

属性	作用	属性值
box-sizing	设置盒模型的两种类型	<code>content-box</code> 默认值，此时 <code>width</code> 和 <code>height</code> 设置的是盒子内容区的大小 <code>border-box</code> 此时 <code>width</code> 和 <code>height</code> 设置的是盒子的总大小，又称怪异盒模型
resize	控制是否允许用户调节元素尺寸 (需要配合 <code>overflow: hidden/scroll/auto</code> 使用)	<code>none</code> 默认值，表示不允许用户调整元素大小 <code>horizontal</code> 表示允许用户调节元素的宽度 <code>vertical</code> 表示允许用户调节元素的高度 <code>both</code> 表示允许用户调节元素的高度和宽度
box-shadow	为盒子添加阴影 (根据属性值数量和值的不同，有六种写法)	<code>none</code> 默认值，表示没有阴影 <code>h-shadow</code> 表示阴影的水平位置，可以取负值 <code>v-shadow</code> 表示阴影的垂直位置，可以取负值 <code>blur</code> 可选，表示模糊距离 <code>spread</code> 可选，表示阴影的外延值 <code>color</code> 可选，表示阴影的颜色 <code>inset</code> 可选，表示外部阴影变为内部阴影 (前五个属性值必须有具体值， <code>inset</code> 本身就是属性值)
opacity	为整个元素添加不透明效果	0~1 之间的数值，0 表示完全透明，1 表示完全不透明

- `rgba` 是颜色的设置方式，用于设置颜色，其透明度仅仅指的是颜色的透明度；`opacity` 是一个属性，设置的是整个元素（包含其中的内容）的不透明度

新增背景属性

29-c3背景属性

属性	作用	属性值
background-origin	设置背景图的原点 (图像的左上角)	<code>padding-box</code> 默认值，表示以 <code>padding</code> 左上角为原点开始显示背景图像 <code>border-box</code> 表示以 <code>border</code> 左上角为原点开始显示背景图像 <code>content-box</code> 表示以 <code>content</code> 左上角为原点开始显示背景图像
background-clip	设置背景的向外裁剪的区域 (这里的背景包括背景颜色和背景图片)	<code>border-box</code> 默认值，表示从 <code>border</code> 开始外裁背景 <code>padding-box</code> 表示从 <code>padding</code> 开始外裁背景 <code>content-box</code> 表示从 <code>content</code> 开始外裁背景 <code>text</code> 表示背景图只显示在文字上

属性	作用	属性值
<code>background-size</code>	设置背景图的尺寸 (五种设置方式)	长度值宽 长度值高 指定背景图片大小, 不允许负值 百分比宽 百分比高 指定背景图片大小, 不允许负值 <code>auto</code> 默认值, 表示背景图片的真实大小 <code>contain</code> 表示将背景图片等比缩放, 使背景图片的宽或高与容器的宽或高相等, 再将完整的背景图放在容器内 (可能会造成容器里部分区域没有背景图片) <code>cover</code> 表示将背景图片等比缩放, 直到完全覆盖容器, 图片会尽可能全的显示在元素上, 但此时图片可能会显示不完整 (相对较好的选择)
<code>background</code>	复合属性	属性值顺序为 <code>color url repeat position / size origin clip</code>

- `background-clip` 如果取值为 `text`, 则可能需要加上私有前缀为 `-webkit-background-clip: text`
- `background` 作为复合属性时, 对属性值的顺序有一定的要求, 但是前三个 `color`、`url`、`repeat` 的顺序和有无是随意的; 如果 `origin` 和 `clip` 取值是一样的, 则可以只写一个值; `size` 的值必须写在 `position` 之后, 并且用 `/` 分开
- 可以使用 `background` 引入多个背景图片, 语法举例如下 (可以理解是使用了四遍复合属性 `background`, 除了不能设定 `background-color` 外, 其余属性都可以使用)

```
background: url("xxx.png") no-repeat left top,
            url("xxx.png") no-repeat right top,
            url("xxx.png") no-repeat left bottom,
            url("xxx.png") no-repeat right bottom
```

新增边框属性

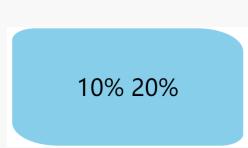
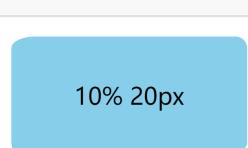
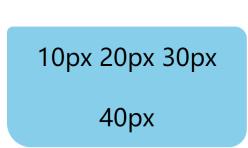
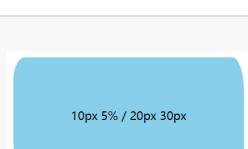
30-c3边框属性

属性	作用	属性值
<code>border-radius</code>	同时设置盒子四个角的圆角	可以是 1~8 个值或百分比, 具体见下述分析
<code>border-top-left-radius</code>	设置盒子左上角圆角半径	可以是 1~2 个值或百分比, 具体见下述分析
<code>border-top-right-radius</code>	设置盒子右上角圆角半径	可以是 1~2 个值或百分比, 具体见下述分析
<code>border-bottom-left-radius</code>	设置盒子左下角圆角半径	可以是 1~2 个值或百分比, 具体见下述分析
<code>border-bottom-right-radius</code>	设置盒子右下角圆角半径	可以是 1~2 个值或百分比, 具体见下述分析
<code>outline-width</code>	外轮廓的宽度	长度值

属性	作用	属性值
<code>outline-color</code>	外轮廓的颜色	长度值
<code>outline-style</code>	外轮廓的风格	<code>none</code> 默认值，表示无轮廓 <code>dotted</code> 表示点状轮廓 <code>dashed</code> 表示虚线轮廓 <code>solid</code> 表示实线轮廓 <code>double</code> 表示双线轮廓
<code>outline-offset</code>	设置外轮廓与边框的距离（该属性不是 <code>outline</code> 的子属性，是一个独立的属性）	长度值，可以是负值，也可以是正值
<code>outline</code>	复合属性	可选 <code>width</code> 、 <code>color</code> 、 <code>style</code> ，无使用顺序，和必选属性

- 要理解圆角相关的使用语法，则必须理解：一个圆角是由一对 (x, y) 唯一决定的
- 定义左上角圆角为 $A(x, y)$ ，右上角圆角为 $B(x, y)$ ，右下角圆角为 $C(x, y)$ ，左下角圆角为 $D(x, y)$ ，则关于 `border-radius` 的各种用法、解释及渲染效果如下表（部分）（具体用法见 <https://developer.mozilla.org/en-US/docs/Web/CSS/border-radius>）
- 一个参数 α 意思是设置四个圆角；
- 两个参数 $\alpha \beta$ 意思是 α 设置左上和右下圆角、 β 设置左下和右上圆角
- 三个参数 $\alpha \beta \gamma$ 意思是 α 设置左上圆角、 β 设置右上和左下圆角、 γ 设置右下圆角
- 四个参数 $\alpha \beta \gamma \omega$ 意思是 α 设置左上圆角、 β 设置右上圆角、 γ 设置右下圆角、 ω 设置左下圆角
- α / β 意思是 α 设置四个圆角的 x 值， β 设置四个圆角的 y 值
- $\alpha_1 \alpha_2 / \beta$ 意思是 $\alpha_1 \alpha_2$ 按照两个参数的规则设置四个圆角的 x 值， β 设置四个圆角的 y 值
- $\alpha_1 \alpha_2 \alpha_3 / \beta_1 \beta_2 \beta_3 \beta_4$ 意思是 $\alpha_1 \alpha_2 \alpha_3$ 按照三个参数的规则设置四个圆角的 x 值， $\beta_1 \beta_2 \beta_3 \beta_4$ 按照四个参数的规则设置四个圆角的 y 值
- 如果 α 是设置一个圆角的百分比，则使其 $[x, y] = \alpha * [width, height]$ ；如果是一个数值，则使其 $x=y=\alpha$
- 总而言之，参数数量不同（对应的圆角不同），参数类型不同（设置圆角方式不同），是否有 $/$ （圆角的 x ， y 是否单独设置）等，有许多种类型，但是规则很简单，只需要理解以下示例，那么都可以归纳总结了（最重要的是明白：①不同参数数量对应的圆角不同②百分比怎么修改圆角③ $/$ 的含义是什么④圆角由两个参数确定）

<code>border-radius</code> 语法/ 代码	解释	效果
<code>10px</code>	四个圆角： $x=y=10px$	<code>10px</code>

<code>border-radius</code> 语法/ 代码	解释	效果
<code>10%</code>	四个圆角: $[x,y]=10\%[width,height]$	
<code>10px 20px</code>	圆角 A、C: $x=y=10px$ 圆角 B、D: $x=y=20px$	
<code>10% 20%</code>	圆角 A、C: $[x,y]=10\%*[width,height]$ 圆角 B、D: $[x,y]=20\%*[width,height]$	
<code>10% 20px</code>	圆角 A、C: $[x,y]=10\%*[width,height]$ 圆角 B、D: $x=y=20px$	
<code>10px 20px 30px</code>	圆角 A: $x=y=10px$ 圆角 B、D: $x=y=20px$ 圆角 C: $x=y=30px$	
<code>10px 20px 30px 40px</code>	圆角 A: $x=y=10px$ 圆角 B: $x=y=20px$ 圆角 C: $x=y=30px$ 圆角 D: $x=y=40px$	
<code>10px 5% / 20px 30px</code>	圆角 A、C: $x=10px, y=20px$ 圆角 B、D: $x=5\%*width, y=30px$	
...

- 关于 `border-top-left-radius` 的各种用法如下表，其余三个对应的属性用法类同

<code>border-top-left-radius</code> 语法/代码	解释
<code>10px</code>	左上角 $x=y=10px$
<code>10px / 20px</code>	左上角 $x=10px, y=20px$
<code>10%</code>	左上角 $[x,y]=10\%*[width,height]$
<code>10% / 20%</code>	左上角 $x=10\%*width, y=20\%*height$

- 边框外轮廓不占位置，不属于盒子模型的内容，可以形象理解为盒子发出来的光

新增文本属性

属性	作用	属性值
<code>text-shadow</code>	给文本添加阴影	<code>h-shadow</code> 必须项，数值，表示阴影的水平位置，允许负值 <code>v-shadow</code> 必须项，数值，表示阴影的垂直位置，允许负值 <code>blur</code> 可选项，数值，表示模糊距离 <code>color</code> 可选项，数值，表示阴影的颜色 <code>none</code> ，默认值，表示没有阴影
<code>white-space</code>	设置文本换行方式	<code>normal</code> 默认值，表示文本超出边界后自动换（文本中的换行被浏览器识别为一个空格） <code>pre</code> 与 <code>pre</code> 标签相同，表示按原文显示 <code>pre-wrap</code> 表示在 <code>pre</code> 的基础上，超出边界后文本自动换行 <code>pre-line</code> 表示在 <code>pre</code> 的基础上，超出边界后文本自动换行，且只识别文本中的换行，并忽略每行前后的空格 <code>nowrap</code> 表示强制不换行
<code>text-overflow</code>	设置文本溢出时的呈现模式 (必须有 <code>overflow: hidden/scroll/auto</code> 和 <code>white-space: nowrap</code> ，改标签才能发挥作用)	<code>clip</code> 默认值，表示当内联内容溢出时，将溢出部分裁切掉 <code>ellipsis</code> 表示当内联内容溢出时，将溢出部分替换为 ...
<code>text-decoration</code>	复合属性，用于文本装饰	可选 <code>line</code> 、 <code>style</code> 、 <code>color</code> ，无使用顺序，和必选属性
<code>text-decoration-line</code>	设置文本装饰线的位置	<code>none</code> 默认值，表示文字无装饰 <code>underline</code> 表示下划线 <code>overline</code> 表示上划线 <code>line-through</code> 表贯穿线（删除线）
<code>text-decoration-style</code>	设置文本装饰线的形状	<code>solid</code> 实线 <code>double</code> 双线 <code>dotted</code> 点状线 <code>dashed</code> 虚线 <code>wavy</code> 波浪线
<code>text-decoration-color</code>	设置文本装饰线的颜色	颜色值

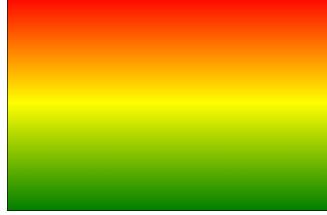
属性	作用	属性值
-webkit-text-stroke	复合属性，用于文本描边	可选 width、color 无顺序要求，和必选属性
-webkit-text-stroke-width	设置文本描边的宽度	长度值
-webkit-text-stroke-color	设置文本描边的颜色	颜色值

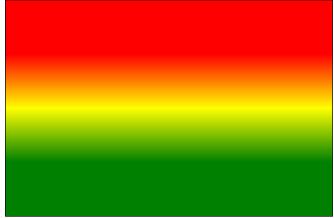
新增渐变 (day10 | 177p: 80.2%)

32-c3渐变效果

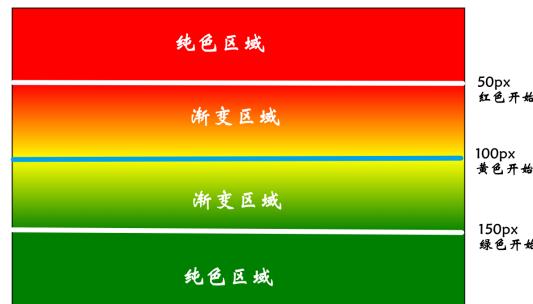
- 渐变：本质上就是将渐变的颜色作为元素的背景图片，可以通过 background-image 属性实现渐变效果。渐变分为线性渐变和径向渐变，此外在这二者的基础上可以实现重复渐变。
- 线性渐变：借助 linear-gradient() 实现。

```
/* basic grammar */
background-image: linear-gradient([direction], color1 [start-place], color2 [start-place], color3 [start-place], ...)
```

代码	解释	效果
linear-gradient(red, yellow, green)	默认从上到下进行颜色渐变：红→黄→绿	
linear-gradient(to right top, red, yellow, green);	设置向右上角进行颜色渐变：红→黄→绿	
linear-gradient(to left, red, yellow, green);	设置向左进行颜色渐变	
linear-gradient(45deg, red, yellow, green)	将从上到下进行的颜色渐变（红→黄→绿）顺时针旋转 45°	

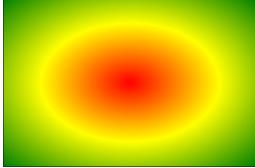
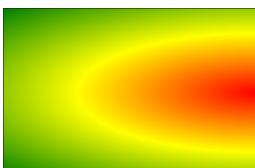
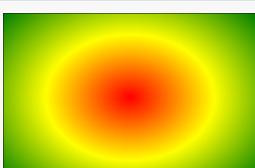
代码	解释	效果
<code>linear-gradient(red 50px, yellow 100px, green 150px)</code>	设置每种颜色在渐变中出现的起始位置	

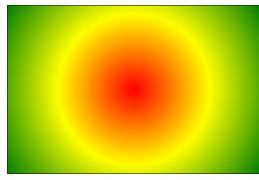
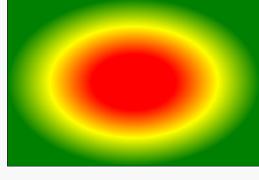
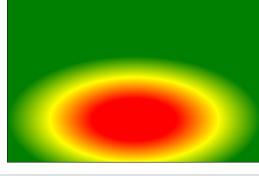
- 我们可以设置参与线性渐变的**颜色**、用关键词设置线性渐变的**方向**（默认是从上到下）、使用**数值**设置每种颜色**开始渐变的位置**
- 注：当我们调整颜色开始渐变的位置之后，元素会存在部分区域是纯色非渐变的，如当我们使用样式 `background-image: linear-gradient(red 50px, yellow 100px, green 150px)` 之后，情形如下



3. 径向渐变：借助 `radial-gradient()` 实现。

```
/* basic grammar */
background-image: radial-gradient([center-radius] ,[center-position] ,color1
[start-place], color2 [start-place], color3 [start-place], ...);
```

代码	解释	效果
<code>radial-gradient(red, yellow, green)</code>	多个颜色之间的渐变， 默认从圆心四散（不一 定是正圆，关键是看容 器本身宽高比）	
<code>radial-gradient(at right top, red, yellow, green)</code>	通过关键词调整渐变的 圆心位置	
<code>radial-gradient(at 300px 100px, red, yellow, green)</code>	通过数值调整渐变的圆 心位置	
<code>radial-gradient(200px 150px, red, yellow, green)</code>	通过数值调整渐变的圆 心半径	

代码	解释	效果
<code>radial-gradient(circle, red, yellow, green)</code>	调整渐变的形状为正圆	
<code>radial-gradient(red 50px, yellow 100px, green 150px)</code>	调整每种颜色在渐变中出现的起始位置	
<code>radial-gradient(100px 50px at 150px 150px, red 50px, yellow 100px, green 150px)</code>	复合使用	

- 我们可以设置参与径向渐变的**颜色**、用关键词或数值设置径向渐变的**圆心位置**、用数值设置径向渐变的**圆心半径**、使用数值设置每种颜色**开始渐变的位置**

- 注：类似线性渐变，当我们调整颜色开始渐变的位置之后，元素会存在部分区域是纯色非渐变的

4. 重复渐变：借助 `repeating-linear-gradient()` 或 `repeating-radial-gradient()` 实现。

- 这里的**重复**：指的是在没有发生渐变的区域，重新开始渐变
- 相应的，所谓需要使用重复的线性渐变或径向渐变的关键是在于**是否存在纯色区域**，只有该渐变具有纯色区域，使用重复渐变才有效

web 字体

[33-c3Web字体](#)

1. Why & What? 当我们设置网页字体时，可能会面对一个问题，那就是：如果我们想要使用某个字体，但是用户没有下载该字体，那么我们如何才能让用户在网页上呈现我们想要的字体呢？—— Web 字体（即在网页中引入本地或服务器中的字体文件）

2. Web 字体的基本用法：通过 `@font-face` 指定字体的具体地址，浏览器会自动下载该字体，这样就不依赖用户电脑上的字体了。

- 简写写法

```
@font-face {
    font-family: "自定义的字体名";
    src: url("本地字体文件路径/网络服务器上字体文件网址");
}
```

此时若我们需要使用相应字体，应该在对应元素的 `font-family` 属性赋予我们自定义的字体名

- 高兼容性写法

```

@font-face {
    font-family: 'webfont';
    font-display: swap;
    src: url('webfont.eot'); /* IE9 */
    src: url('webfont.eot?#iefix') format('embedded-opentype'), /* IE6-
IE8 */
    url('webfont.woff2') format('woff2'),
    url('webfont.woff') format('woff'), /* chrome、firefox */
    url('webfont.ttf') format('truetype'), /* chrome、firefox、opera、
Safari, Android, iOS 4.2+*/
    url('webfont.svg#webfont') format('svg'); /* iOS 4.1- */
}

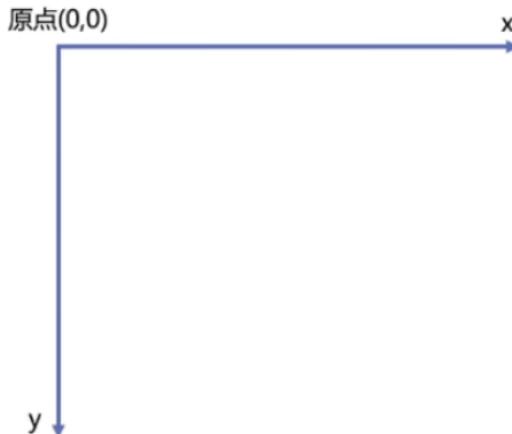
```

3. 定制字体：因为中文的字体文件通常很大，而实际中我们不会使用字体文件中的全部文字，因此我们采用对某些文字进行单独定制的方式，引入更小的字体资源文件。[阿里 Web 字体定制工具](#)
4. 字体图标：实质上是一种文字，而不是图片，是一种用于看而不是读的文字。其优势有：相比图片更加清晰；灵活性高，可以方便改变大小、颜色、风格等；兼容性好，[IE](#) 也可支持。[阿里字体图标库](#)

2D 变换

[34-c3多重变换](#)

变换：是指将 HTML 中的元素进行位移、缩放、旋转、扭曲等操作，2D 变换建立在二维坐标系下，3D 变换建立在三维坐标系下，而变换的实现，我们都是通过 `transform` 属性实现。`transform` 相关的变换都不能用在行内元素上。



1. 位移：2D 变换下的位移可以改变元素的位置，元素的 `transform` 属性的取值情况如下。

属性值	作用
<code>translateX()</code>	设置元素水平方向位移，长度值 or 百分比（参考自身宽度）
<code>translateY()</code>	设置元素垂直方向位移，长度值 or 百分比（参考自身高度）
<code>translate()</code>	一个值，设置元素水平方向的位移 两个值，设置元素水平和垂直方向的位移

- 位移与相对定位相似：**不脱离文档流**；不会影响到其他元素
- 位移与相对定位区别：相对定位的百分比值参考其父元素，位移的百分比值参考自身
- 浏览器针对位移有优化，因此与定位相比，**浏览器处理位移的效率更高**

- `transform` 属性也可以链式编写指定元素水平和垂直方向上的位移 `transform: translateX(30px) translateY(40px)`
- 位移操作对**行内元素无效**
- 位移配合绝对定位，可以实现元素垂直水平居中，代码如下

```
.box {
  position: absolute;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
}
```

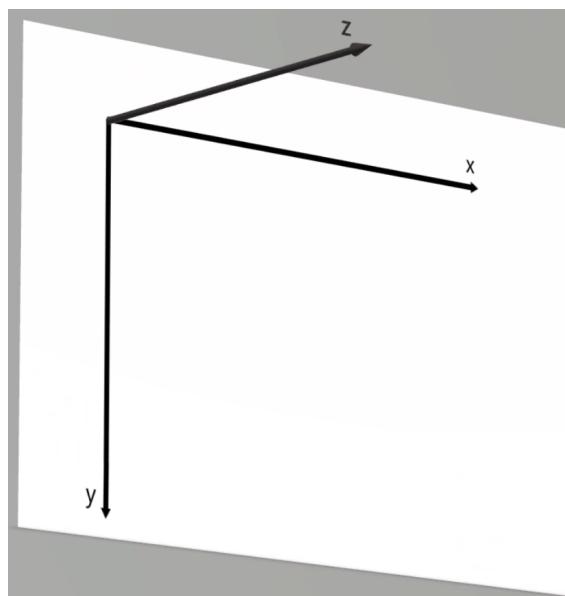
2. 缩放：2D 变换下的缩放可以改变**元素的大小**，元素的 `transform` 属性的取值情况如下。

属性值	作用
<code>scaleX()</code>	设置元素水平方向的缩放比例，数值（1 表示不缩放，大于 1 表示放大，小于 1 表示缩小）
<code>scaleY()</code>	设置元素垂直方向的缩放比例，数值（1 表示不缩放，大于 1 表示放大，小于 1 表示缩小）
<code>scale()</code>	一个值，同时设置元素水平和垂直方向的缩放比例 两个值，分别设置元素水平和垂直方向的缩放比例

- `scale` 的取值可以是负值，但一般不使用
- 借助缩放，可以实现显示小于浏览器规定最小字体（如 `12px`）的文字

3. 旋转：2D 变换下的旋转可以让元素**顺时针或逆时针旋转**，元素的 `transform` 属性的取值情况如下。

属性值	作用
<code>rotateZ</code>	设置元素按 Z 轴的旋转角度，角度值 <code>deg</code> ，正值表示顺时针旋转，负值表示逆时针旋转
<code>rotate</code>	一个值，同 <code>rotateZ</code> 用法

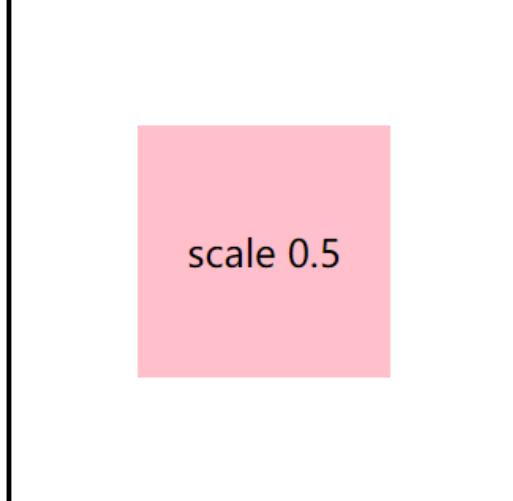
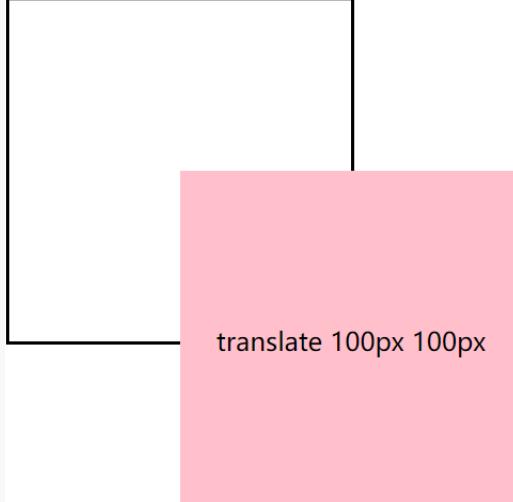


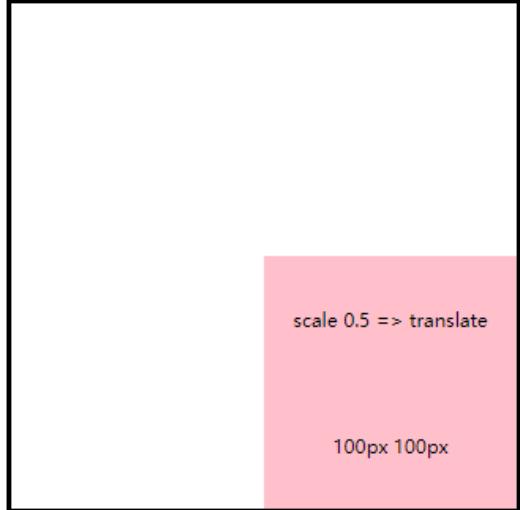
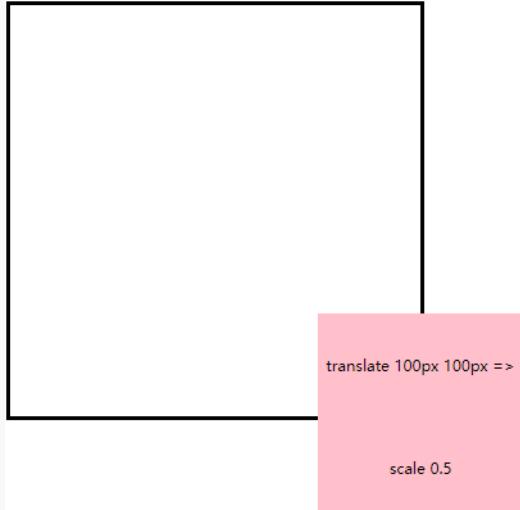
4. 扭曲：2D 变换下的扭曲让元素在二维平面被拉扯，进而走形，实际开发中几乎不用，元素的 `transform` 属性的取值情况如下。

属性值	作用
<code>skewX</code>	设置元素在水平方向的扭曲，角度值 <code>deg</code> ，会将元素的左上角和右下角拉扯
<code>skewY</code>	设置元素在垂直方向的扭曲，角度值 <code>deg</code> ，会将元素的左上角和右下角拉扯
<code>skew</code>	一个值，设置元素在水平方向上的扭曲 两个值，设置元素在水平和垂直方向上的扭曲

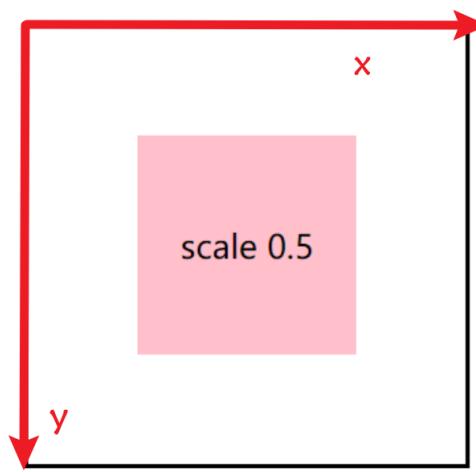
5. 多重变换：2D 变换下的多重变换，指的是让元素通过 `transform` 属性实现位移、缩放、旋转、扭曲。

- 先位移再缩放 ≠ 先缩放再位移

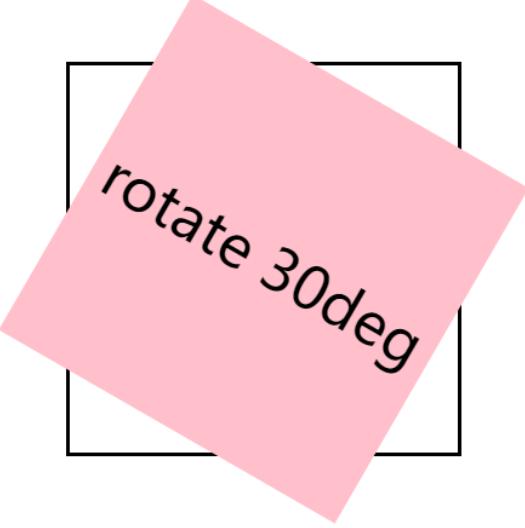
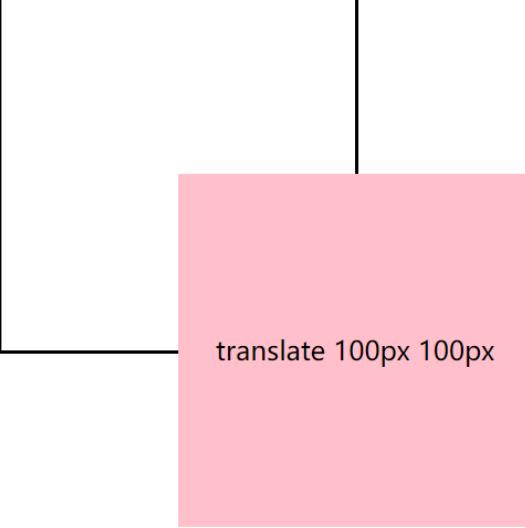
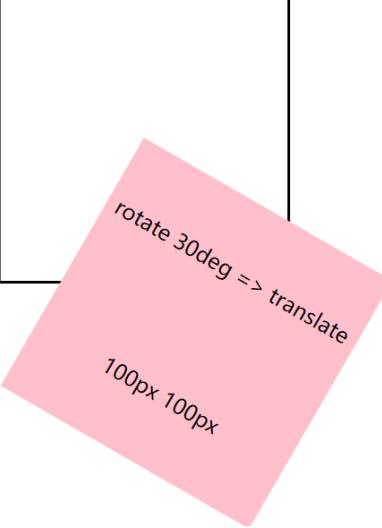
代码	效果
<code>transform: scale(0.5)</code>	
<code>transform: translate(100px, 100px)</code>	

代码	效果
<pre>transform: scale(0.5) translate(100px, 100px)</pre>	
<pre>transform: translate(100px, 100px) scale(0.5)</pre>	

- 整个变换的过程中，二维坐标系没有改变

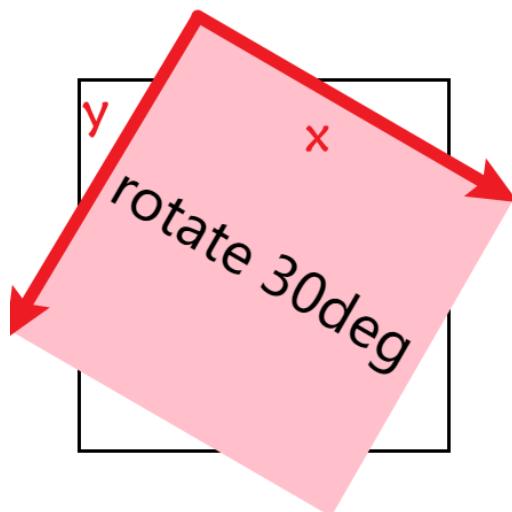


- 位移变换的操作是，对于给定的 (x, y) ，将元素盒子的左上角移动到二维坐标系中的 (x, y) 处
- 缩放变换的操作是，对于给定的缩放比例，以元素盒子的水平垂直中心为原点进行缩放
 - 先位移再旋转 \neq 先旋转再位移

代码	效果
<pre>transform: rotate(30deg)</pre>	 <p>A pink rectangular element is shown rotated 30 degrees counter-clockwise relative to its parent element. The rotation is applied directly to the element.</p>
<pre>transform: translate(100px, 100px)</pre>	 <p>A pink rectangular element is shown moved 100px horizontally and 100px vertically relative to its parent element. The translation is applied directly to the element.</p>
<pre>transform: rotate(30deg) translate(100px, 100px)</pre>	 <p>A pink rectangular element is shown rotated 30 degrees counter-clockwise and then moved 100px horizontally and 100px vertically relative to its parent element. The transformation is applied sequentially, resulting in a combined effect where the element is both rotated and translated.</p>

代码	效果
<pre>transform: translate(100px, 100px) rotate(30deg)</pre>	

- 整个变换的过程中，二维坐标系被旋转破坏掉了



- 旋转变换的操作是，对于给定的角度值，以元素盒子的水平垂直中心为原点进行顺时针或逆时针旋转；与之同时，二维坐标系也发生了旋转
 - 多重变换的使用注意：因为旋转变换对二维坐标系的破坏性，因此使用多重变换时，最后再进行旋转
6. 变换原点：元素变换时默认以元素的中心为原点作为变换原点进行变换，使用 `transform-origin` 属性可以设置变换原点，该属性的取值情况如下。

属性取值举例	作用
<code>transform-origin: 30% 40%</code>	变换原点的 x 坐标为自身宽度的 30%； y 坐标为自身高度的 40%
<code>transform-origin: left top</code>	变换原点在元素盒子的左上角
<code>transform-origin: 30px 40px</code>	变换原点的坐标为 (30px, 40px)
<code>transform-origin: 30%</code>	变换原点的 x 坐标为自身宽度的 30%； y 坐标为自身高度的 50%

属性取值举例	作用
<code>transform-origin: left</code>	变换原点在元素盒子的左中点 <code>left center</code>
<code>transform-origin: bottom</code>	变换原点在元素盒子的中下点 <code>center bottom</code>
<code>transform-origin: 30px</code>	变换原点的坐标为 <code>(30px, 50%*自身高度)</code>

- 我们可以用百分比、像素值、关键字设置变换原点的位置；如果是两个值，则分别设置变换原点水平和垂直的位置；如果是一个值，则设置变换原点水平位置，垂直位置默认为自身高度的 `50%`（关键字除外，因为可以识别出是 `x` 方向，还是 `y` 方向）
- 修改变换原点对位移变换没有影响，**对旋转变换和缩放变换有影响**

3D 变换

3D 变换：与 2D 变换不同，3D 变换建立在三维坐标系下，同时为了对元素进行 3D 变换，其父元素必须**开启 3D 空间**；此外，我们也需要学习理解**景深**这一概念，并给父元素设置景深（可选）；我们也可以通过在父元素中修改**透视点位置**（可选）。

- 开启 3D 变换：必须在需要 3D 变换的元素的父元素中设置！！！

```
transform-style: preserve-3d;
/* 通过在父元素中设置 transform-style 属性，我们可以开启 3D 变换
   1. 可选值 flat：默认值，表示让子元素位于此元素的二维平面内（2D 空间）
   2. 可选值 preserve-3d：表示让子元素位于此元素的三维空间内（3D 空间）
*/
```

- 设置景深：景深是指观察者与 `z=0` 平面的距离，设置景深，可以让发生 3D 变换的元素产生透视效果，看起来更加立体。我们可以借助 `perspective` 属性设置景深，同时，该属性也必须设置给需要 3D 变换的元素的父元素！！！

```
perspective: 100px;
/* 通过在父元素中设置 perspective 属性，我们可以设置观察者与 z=0 平面的距离
   1. none：默认值，表示不设置景深（不指定透视）
   2. 长度值：表示观察者距离 z=0 平面的距离，不允许是负值
*/
```

- 修改透视点位置：透视点位置就是观察者的位置，默认的透视点位置是元素的中心。我们可以借助 `perspective-origin` 属性设置透视点位置，同时，该属性也必须设置给需要 3D 变换的元素的父元素！！！

```
perspective-origin: 400px 300px;
/* 通过在父元素中设置 perspective-origin 属性，我们可以设置观察者位置（透视点位置）
   通常是两个值，第一个值表示水平相对距离，第二个值表示垂直相对距离；
   0px 0px 即元素盒子的中心，200px 300px 表示从元素盒子中心向右移动 200px，向下移动
   300px
*/
```

- 位移：3D 位移是在 2D 位移的基础上，可以让元素沿着 `z` 轴位移，也是通过 `transform` 属性实现，属性取值如下。

属性值	作用
<code>translateZ()</code>	设置 <code>z</code> 轴位移，长度值，正值表示向屏幕外位移，负值表示向屏幕里位移（不能写百分比）
<code>translate3d()</code>	第一个参数对应 <code>x</code> 轴位移，第二个参数对应 <code>y</code> 轴位移，第三个参数对应 <code>z</code> 轴位移（三个参数均不能省略）

2. 旋转：3D 旋转是在 2D 旋转的基础上，可以让元素沿着 `x` 轴和 `y` 轴旋转，也是通过 `transform` 属性实现，属性取值如下。

属性值	作用
<code>rotateX()</code>	设置元素按照 <code>x</code> 轴的旋转角度，角度值 <code>deg</code> ，面对 <code>x</code> 轴：正值表示顺时针旋转，负值表示逆时针旋转
<code>rotateY()</code>	设置元素按照 <code>y</code> 轴的旋转角度，角度值 <code>deg</code> ，面对 <code>y</code> 轴：正值表示顺时针旋转，负值表示逆时针旋转
<code>rotate3d()</code>	共四个参数，前三个参数分别表示 <code>x</code> , <code>y</code> , <code>z</code> 坐标轴是否旋转，第四个参数表示旋转的角度（四个参数均不能省略）

```
transform: rotate3d(1, 1, 1, 30deg) /* 元素绕 x、y、z 轴分别旋转三十度 */
```

3. 缩放：3D 缩放是在 2D 缩放的基础上，可以让元素沿着 `z` 轴缩放，也是通过 `transform` 属性实现，属性取值如下。

属性值	作用
<code>scaleZ()</code>	设置元素在 <code>z</code> 轴方向的缩放比例，数值（1 表示不能缩放，大于 1 表示放大，小于 1 表示缩小）
<code>scale3d()</code>	第一个参数对应元素 <code>x</code> 轴方向的缩放比例，第二个参数对应元素 <code>y</code> 轴方向的缩放比例，第三个参数对应元素 <code>z</code> 轴方向的缩放比例（三个参数不允许省略）

- 可以将 `scaleZ` 理解为修改元素的厚度，但是元素没有厚度，所以相应的，浏览器会根据 `scaleZ` 的值修改景深，起到一个类似的效果

4. 多重变化：同样的，3D 多重变换时，因为旋转会破坏坐标系，故而**最后旋转**。

5. 变换原点：与 2D 变换设置变换原点的方式相同，语法为：`transform-origin: x y`。我们讨论变换原点对于旋转变换的影响。

- `rotateZ` 以过 `(x, y)` 垂直于 `z=0` 平面的直线为旋转轴
- `rotateX` 以过 `(x, y)` 垂直于 `y=0`，在 `z=0` 平面的直线为旋转轴
- `rotateY` 以过 `(x, y)` 垂直于 `x=0`，在 `z=0` 平面的直线为旋转轴

6. 背部可见性：当旋转元素过 `90deg` 后，我们可以看见元素的背部内容，通过语法 `backface-visibility: hidden` 可以设置元素背部内容不可见。

过渡 (day11|181p: 88.2%)

过渡：在不使用 Flash 动画、不使用 JavaScript 的情况下，让元素从一种样式，以一种特殊的效果变成另一种样式。

属性名	作用	属性值
<code>transition-property</code>	定义需要过渡的属性 (只有在该属性中定义的属性才可会有过渡效果)	<code>none</code> 表示任何属性都不过渡 <code>all</code> 默认值，表示全部属性都可以过渡 <code>具体某个属性名</code> 表示特定的属性可以过渡 (多个属性名之间逗号分隔)
<code>transition-duration</code>	设置过渡的持续时间 (即一个样式过渡到另一个样式耗时多久)	<code>0</code> 默认值，表示没有任何过渡时间 时间值 单位为 <code>s</code> 或 <code>ms</code> 时间列表 一个值，表示让所有过渡属性持续时间都相同；多个值，以逗号分隔，表示给每个过渡属性设置不同的持续时间
<code>transition-delay</code>	设置过渡开始前的延迟时间	时间值 单位为 <code>s</code> 或 <code>ms</code>
<code>transition-timing-function</code>	设置过渡的类型	<code>ease</code> 默认值，表示平滑过渡 <code>linear</code> 表示线性过渡 <code>ease-in</code> 表示过渡从慢→快 <code>ease-out</code> 表示过渡从快→慢 <code>ease-in-out</code> 表示过渡从慢→快→慢 <code>step-start</code> 等同 <code>steps(1, start)</code> <code>step-end</code> 等同 <code>steps(1, end)</code> <code>steps(integer, start/end)</code> 接受两个参数的步进函数；第一个参数必须是正整数，表示过渡的步数；第二个参数可以是 <code>start</code> 和 <code>end</code> ，默认是 <code>end</code> ，表示过渡每一步发生变化的时间点 <code>cubic-bezier(number, number, number, number)</code> ：特定的贝塞尔曲线类型
<code>transition</code>	复合属性	语法为： <code>transition-duration transition-property transition-delay transition-timing-function</code>

- 只有值为数字或可以转为数字的属性，才支持过渡，如：颜色、长度值、百分比、`z-index`、`opacity`、2D 变换属性、3D 变换属性、阴影等
- `ease` 和 `ease-in-out` 类型的过渡都是从慢→快→慢，但是 `ease` 更加平滑一些
- `steps(integer, start/end)` 步进函数的 `start` 表示，对于一段过渡时间，开始时刻，瞬间过渡到对应状态，再等待这段时间结束；`end` 表示，等这一段过渡时间结束的一刻，瞬间过渡到对应状态
- `cubic-bezier(number, number, number, number)` 可以利用[在线制作贝塞尔曲线](#)，并粘贴对应的代码设置过渡类型

- `transition` 复合属性，如果设置了一个时间，则表示 `transition-duration`；如果设置了两个时间，则第一个表示 `transition-duration`，第二个表示 `transition-delay`；其他属性值没有顺序要求；所有属性值没有必须的要求

动画

36-c3动画

1. 动画：在一段时间内连续播放 n 个画面。
2. 帧：动画中的每一张画面。一定时间内连续快速播放若干个帧就形成了动画，相同时间内，播放的帧数越多，画面看起来就越流畅。
3. 关键帧：构成一段动画的若干帧中，起决定性作用的几帧。
4. 动画的定义语法：通过关键帧定义动画
 - 方式一：设定开始和结束的关键帧

```
@keyframes animation-name {
  from {
    /* value of properties */
  }
  to {
    /* value of properties */
  }
}
```

- 方式二：除了设定开始和结束的关键帧外，还可以设置动画中的任意阶段的关键帧

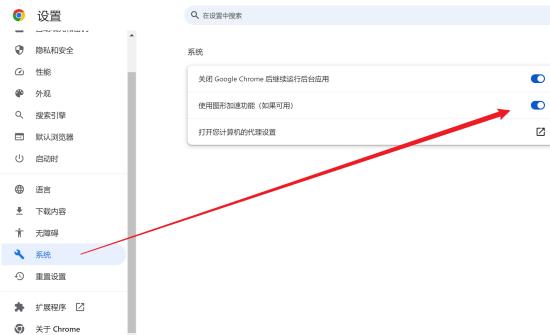
```
@keyframes animation-name {
  0% {
    /* value of properties */
  }
  20% {
    /* value of properties */
  }
  60% {
    /* value of properties */
  }
  80% {
    /* value of properties */
  }
  100% {
    /* value of properties */
  }
}
```

- 方式三：方式一和方式二混用，但不推荐

属性名	作用	属性值
<code>animation-name</code>	给元素应用指定的动画	通过 <code>@keyframes</code> 定义的动画名

属性名	作用	属性值
<code>animation-duration</code>	设置动画的持续时间	时间值，单位为 <code>s</code> 或 <code>ms</code>
<code>animation-delay</code>	设置动画开始前的延迟时间	时间值，单位为 <code>s</code> 或 <code>ms</code>
<code>animation-timing-function</code>	设置动画的类型	<p><code>ease</code> 默认值，表示平滑过渡 <code>linear</code> 表示线性过渡 <code>ease-in</code> 表示过渡从慢→快 <code>ease-out</code> 表示过渡从快→慢 <code>ease-in-out</code> 表示过渡从慢→快→慢 <code>step-start</code> 等同 <code>steps(1, start)</code> <code>step-end</code> 等同 <code>steps(1, end)</code> <code>steps(integer, start/end)</code> 接受两个参数的步进函数；第一个参数必须是正整数，表示过渡的步数；第二个参数可以是 <code>start</code> 和 <code>end</code>，默认是 <code>end</code>，表示过渡每一步发生变化的时间点 <code>cubic-bezier(number, number, number, number)</code>：特定的贝塞尔曲线类型</p>
<code>animation-iteration-count</code>	设置动画的播放次数	<code>number</code> 动画的循环次数 <code>infinite</code> 表示动画无限循环
<code>animation-direction</code>	指定动画的方向	<code>normal</code> 默认值，表示正常方向（即从开始关键帧到结束关键帧） <code>reverse</code> 表示反方向 <code>alternate</code> 表示先正方向，再反方向，交替持续 <code>alternat-reverse</code> 表示先反方向，再正方向，交替持续
<code>animation-fill-mode</code>	设置动画之外的状态（即不发生动画的时候）	<code>forwards</code> 设置元素为动画结束时的状态 <code>backwards</code> 设置元素为动画开始时的状态
<code>animation-play-status</code>	设置动画的播放状态	<code>running</code> 默认值，表示播放 <code>paused</code> 表示暂停
<code>animation</code>	复合属性	语法为： <code>animation-name animation-duration</code> <code>animation-delay animation-timing-function</code> <code>animation-iteration animation-direction</code> <code>animation-fill-mode animation-play-status</code>

- 有时候应用动画，刚开始的短暂停时间内可能会出现不完美的渲染，此时可以在浏览器中打开硬件加速模式，或能解决



- `animation` 复合属性，如果设置了一个时间，则表示 `animation-duration`；如果设置了两个时间，则第一个表示 `animation-duration`，第二个表示 `animation-delay`；其他属性值没有顺序要求；所有属性值没有必须的要求
- `animation-play-status` 一般单独使用，不在复合属性中设置
- 动画与过渡的区别？？
 - 动画不需要任何触发条件；过渡需要触发条件，并且当触发条件撤销后，会恢复过渡前的状态
 - 动画可以在变化的过程中添加新的样式（关键帧）；过渡只能设置元素过渡后的状态

多列布局

37-c3多列布局

1. 文字多列布局：专门用于实现类似报纸的布局。

属性名	作用	属性值
<code>column-count</code>	指定列数	数字
<code>column-width</code>	指定列宽	长度
<code>columns</code>	复合属性	语法为： <code>column-count column-width</code> ，没有数量和顺序要求；实际列数等于 <code>count</code> 和根据 <code>width</code> 计算得到的 <code>count</code> 之间的较小值
<code>column-gap</code>	指定列边距	长度
<code>column-rule-style</code>	设置列间边框的风格	与 <code>border-style</code> 取值相同，如 <code>dashed</code> 、 <code>dotted</code> 、 <code>solid</code> 等
<code>column-rule-width</code>	设置列间边框的宽度	长度
<code>column-rule-color</code>	设置列间边框的颜色	颜色
<code>column-rule</code>	复合属性	语法为： <code>column-rule-width column-rule-color column-rule-style</code> ，没有顺序要求，但是三个参数都得写

属性名	作用	属性值
<code>column-span</code>	设置当前元素是否跨列（常用于标题）	<code>none</code> 默认值，表示不跨列 <code>all</code> 表示跨列

- `column-span` 属性必须给要跨列的元素单独添加，如 `h1`；其他属性则需要给放置内容的容器去添加

2. 图片多列布局：设置 `column-count` 属性即可

伸缩盒模型

简介

1. **伸缩盒模型**：英文为 Flexible Box，又称**弹性盒子**，是 W3C 于 2009 年提出的一种新的盒子模型。
2. **优势**

- 伸缩盒模型可以轻松控制：元素分布方式、元素对齐方式、元素视觉顺序等
- 目前，除了部分 IE 浏览器不支持，其他浏览器均已全部支持伸缩盒模型
- 伸缩盒模型的出现演变出了一套新的布局方案，即 `flex` 布局
 - 传统布局：基于传统盒模型，依靠 `display`、`position`、`float` 属性实现布局
 - 因为传统布局不能很好呈现在移动设备上，`flex` 布局在移动端应用较广泛

伸缩容器与伸缩项目

1. **伸缩容器**：开启了 `flex` 布局的元素，有两种方式开启这种布局（通过 `display` 属性实现）。

- 方式一：`display: flex`
- 方式二：`display: inline-flex`

方式二比较少用，因为 `inline-flex` 将元素变为伸缩容器的同时，使得其也有了行内块的特性，此时两个伸缩容器之间可能会因为 HTML 源代码中的换行在网页中呈现空格效果

2. **伸缩项目**：伸缩容器的所有子元素。

- 伸缩容器的**子元素**是伸缩项目，其孙子元素、重孙子元素等后代元素，都不是伸缩项目
- 无论原来是什么类型的元素（块、行内块、行内），一旦成为了伸缩项目，全都会**块状化**（可以设置宽高，也可以由内容撑起宽高）
- 一个元素可以同时是伸缩容器和伸缩项目

主轴与侧轴

1. **主轴** (`main axis`)：伸缩容器中的**伸缩项目沿主轴排列**，主轴默认是**水平的**，默认方向是：左→右（左边是起点，右边是终点）。

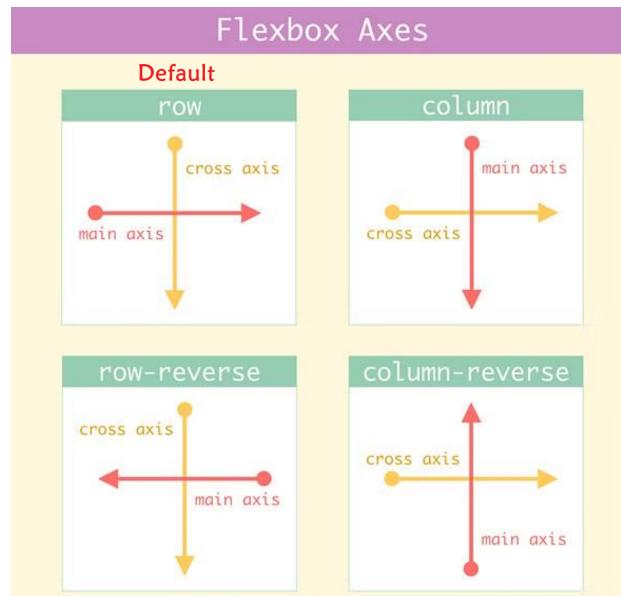
2. **侧轴** (`cross axis`)：与**主轴垂直**，侧轴默认是**垂直的**，默认方向是：上→下（上边是起点，下边是终点）

3. **主轴方向**：通过 `flex-direction` 属性，可以修改伸缩容器的主轴方向。

- 当我们修改了主轴方向后，侧轴方向也随之改变

- 我们不关注原点这一概念，关心的是主轴和侧轴的方向，从而便于元素布局与对齐等

属性值	作用
<code>row</code>	默认值，表示主轴方向：左→右 (→)
<code>row-reverse</code>	表示主轴方向：右→左 (←)
<code>column</code>	表示主轴方向：上→下 (↓)
<code>column-reverse</code>	表示主轴方向：下→上 (↑)



主轴换行方式

我们使用 `flex-wrap` 属性来修改伸缩容器中主轴方向上元素的换行方式。

属性值	作用	效果举例
<code>nowrap</code>	默认值，表示 不换行 （此时若伸缩项目过多则会压缩宽度）	
<code>wrap</code>	表示 自动换行 （此时换行的伸缩项目不一定会抵着上一行项目放置，而可能会有一定的间距）	
<code>wrap-reverse</code>	表示 自动反向换行 （此时第二行元素会从上边开启）	

复合属性：主轴方向&换行

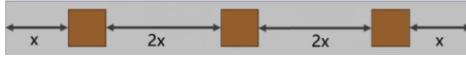
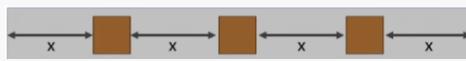
我们可以使用复合属性 `flex-flow` 同时设置主轴方向和元素在主轴方向上的换行方式，两个属性值没有顺序和必须要求，语法如下。

```
flex-flow: flex-direction flex-wrap;
```

一般来说不是用这个复合属性，而是使用其对应的两个子属性，更加见名知意

主轴对齐方式

我们可以使用 `justify-content` 属性调整元素在主轴方向上的对齐方式。

属性值	作用	效果举例
<code>flex-start</code>	默认值，表示元素从主轴的起始位置 紧凑对齐	
<code>flex-end</code>	表示元素从主轴的结束位置紧凑对齐	
<code>center</code>	表示元素居中紧凑对齐	
<code>space-between</code>	表示元素分散对齐，伸缩项目之间间距相等，两边伸缩项目距伸缩容器边缘无间距，最常用	
<code>space-around</code>	表示元素分散对齐，伸缩项目之间间距相等，两边伸缩项目距伸缩容器边缘为项目之间间距一半	
<code>space-evenly</code>	表示元素分散对齐，伸缩项目之间间距和两边伸缩项目距伸缩容器边缘间距均相等	

侧轴对齐方式

我们可以使用 `align-items` 和 `align-content` 两个属性调整元素在侧轴方向上的对齐方式，specifically，我们需要分伸缩容器中有一行伸缩项目还是多行伸缩项目这两种情况进行讨论。

- 情况一：伸缩容器中只有一行伸缩项目，我们使用 `align-items` 属性设置元素在侧轴方向上的对齐方式。

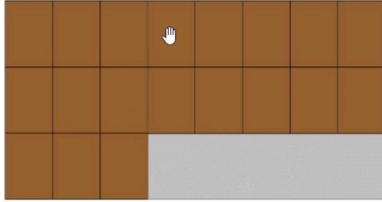
属性值	作用	效果举例
<code>flex-start</code>	表示元素与侧轴的起始位置对齐	
<code>flex-end</code>	表示元素与侧轴的结束位置对齐	
<code>center</code>	表示元素与侧轴的中点对齐	
<code>baseline</code>	表示元素之间第一行文字的基线对齐（小写 x 底边对齐）	

属性值	作用	效果举例
<code>stretch</code>	默认值，表示将元素 垂直拉伸到整个父容器 （前提是伸缩项目未设置高度）	

注：这里的元素指的就是伸缩项目

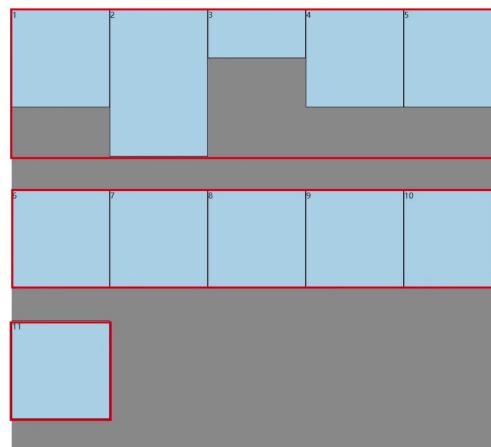
2. 情况二：伸缩容器中存在**多行伸缩项目**，我们使用 `align-content` 属性设置元素在侧轴方向上的对齐方式。

属性值	作用	效果举例
<code>flex-start</code>	表示元素与侧轴的 起始位置紧凑对齐	
<code>flex-end</code>	表示元素与侧轴的 结束位置紧凑对齐	
<code>center</code>	表示元素与侧轴的 中点紧凑对齐	
<code>space-between</code>	表示每行元素 分散对齐 ，每行伸缩项目之间的间距是相等的，上下两行伸缩项目与伸缩容器边缘无间距	
<code>space-around</code>	表示每行元素 分散对齐 ，每行伸缩项目之间的间距是相等的，且是上下两行伸缩项目与伸缩容器边缘间距的二倍	
<code>space-evenly</code>	表示每行元素 分散对齐 ，每行伸缩项目之间的间距和上下两行伸缩项目与伸缩容器边缘间距均相等	

属性值	作用	效果举例
stretch	默认值，表示将元素垂直拉伸占满整个侧轴（前提是伸缩项目未设置高度）	

注

- 这里的元素指的就是伸缩项目
- 这里的一行元素的高度，以这一行中最高的元素为准



伸缩盒模型应用：实现元素的水平垂直居中

1. 方式一

```
/* 父元素样式 */
.outer {
    width: 400px;
    height: 400px;
    background-color: #888;

    display: flex;
    justify-content: center;
    align-items: center;
}

/* 子元素样式：实现了在父元素中的水平垂直居中 */
.inner {
    width: 100px;
    height: 100px;
    background-color: orange;
}
```

2. 方式二

```
/* 父元素样式 */
.outer {
    width: 400px;
    height: 400px;
```

```

background-color: #888;

display: flex;
}

/* 子元素样式：实现了在父元素中的水平垂直居中 */
.inner {
  width: 100px;
  height: 100px;
  background-color: orange;

  margin: auto;
}

```

总结之“实现元素水平垂直居中的方式”

1. 方式一：伸缩盒模型-设置子伸缩项目主轴和侧轴方向上居中，[跳转笔记](#)
2. 方式二：伸缩盒模型-设置伸缩项目外边距为 `auto`，[跳转笔记](#)

基准长度

我们可以利用 `flex-basis` 属性设置伸缩项目在主轴方向上的**基准长度**，所谓基准长度，就是浏览器用于计算主轴上是否有多余空间的一个重要属性。

- 当主轴横向时，设置基准长度会让伸缩项目的宽失效，替代为对应的基准长度值
- 当主轴纵向时，设置基准长度会让伸缩项目的高失效，替代为对应的基准长度值
- 默认情况下 `flex-basis: auto`，当我们设置了基准长度后，该伸缩项目的高或者宽就会失效；此外，注意该属性是给伸缩项目设置的，而不是伸缩容器

伸缩性 (day12|193p: 95.3%)

1. 伸 (`grow`)：当伸缩容器有多余空间时，伸缩项目会在主轴方向上拉伸直至占满父元素。我们可以使用 `flex-grow` 属性定义伸缩项目的放大比例，默认值为 `0`，表示：即使主轴存在剩余空间，也不进行拉伸。

拉伸规则：多个拉伸项目以其 `flex-grow` 值，成比例地分配主轴上的剩余空间，即每个伸缩项目拉伸空间的计算方式为：

$$\text{剩余空间} \times \frac{g_i}{\sum_{\text{所有拉伸项目}} g_j}$$

这里记第 i 个伸缩项目的 `flex-grow` 的值为 g_i

例：容器一、容器二、容器三的 `flex-grow` 值分别为 1、2、3，主轴上的剩余空间为 300px，则

- 容器一拉伸空间为 $300 \times \frac{1}{1+2+3} = 50$
- 容器二拉伸空间为 $300 \times \frac{2}{1+2+3} = 100$
- 容器三拉伸空间为 $300 \times \frac{3}{1+2+3} = 150$

2. 缩 (shrink)：当伸缩容器空间不足时，伸缩项目会在主轴方向压缩到父元素内。我们可以使用 `flex-shrink` 属性定义伸缩项目的缩小比例，默认值是 1，表示：如果空间不足，该伸缩项目会根据其压缩规则进行缩小。（注：必须设置 `flex-wrap: nowrap`，否则伸缩项目会因为换行而无法压缩）

压缩规则：多个压缩项目以其 `flex-shrink` 值，并参考自身在主轴方向上的宽度或高度（主轴方向默认时，考虑宽度），分配主轴上要压缩的总空间，即每个伸缩项目压缩空间的计算方式为：

$$\text{压缩空间} \times \frac{w_i s_i}{\sum_{\text{所有伸缩项目}} w_j s_j}$$

这里记第 i 个伸缩项目的宽度为 w_i ，伸缩项目的 `flex-shrink` 的值为 s_i

例：容器一、容器二、容器三的 `width` 值分别为 200、300、200，`flex-shrink` 值分别为 1、1、1，主轴上的压缩空间为 300px（即主轴宽度是 400px，因此压缩空间为 $200+300+200-400=300$ ），则

- 容器一收缩空间为 $300 \times \frac{200 \times 1}{200 \times 1 + 300 \times 1 + 200 \times 1} = 85.71$
- 容器二收缩空间为 $300 \times \frac{300 \times 1}{200 \times 1 + 300 \times 1 + 200 \times 1} = 128.57$
- 容器三收缩空间为 $300 \times \frac{200 \times 1}{200 \times 1 + 300 \times 1 + 200 \times 1} = 85.71$

3. 注意事项

- `flex-grow` 和 `flex-shrink` 都是给伸缩项目设置的属性
- 为什么按照压缩规则计算出来的伸缩项目压缩后的宽度（或高度）与浏览器提供的不一样？这可能是因为当前伸缩项目存在边框，并且设置盒子的宽高以边框记（怪异盒模型），会导致浏览器的计算误差
- 为什么伸缩项目拉伸和压缩的计算方式不同（同甘不共苦）？如果伸缩项目压缩，也和拉伸一样，按照 `flex-shrink` 成比例压缩，这对宽度（或高度）较小的伸缩项目是不公平的，比如宽度（或高度）300px 和 30px 的两个伸缩项目，如果不考虑宽度（或高度），则轻易会导致 30px 的项目压缩太多，从而内容失真，当引入项目本身宽度（或高度）的因素后，压缩之后的各个项目的呈现才能更加均衡

注：当主轴方向水平，我们考虑的是宽度；当主轴方向垂直，我们考虑的是高度

- 压缩是否存在极限？压缩极限就是至少让伸缩项目中的内容得以呈现

复合属性：伸缩&基准长度

我们可以通过 `flex` 复合属性同时设置伸缩项目的 `flex-grow`、`flex-shrink`、`flex-basis`，语法如下，

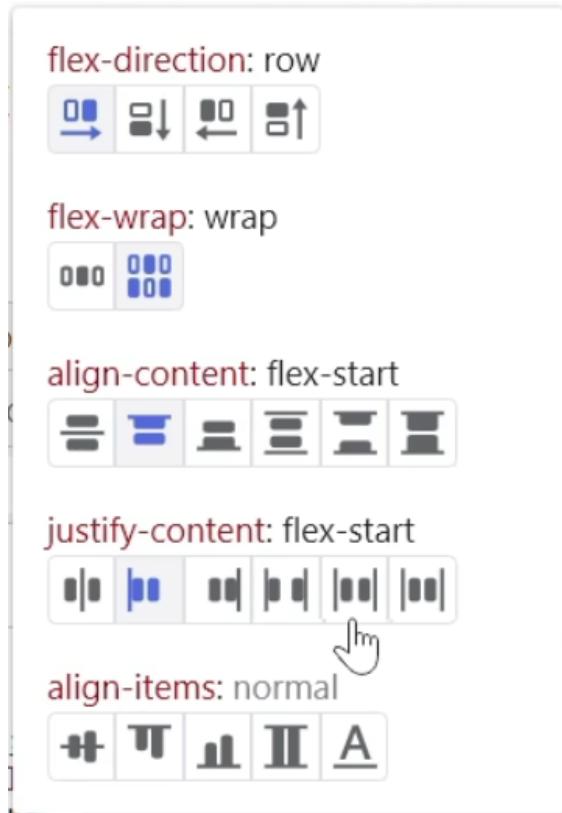
```
flex: flex-grow flex-shrink flex-basis;
```

- `flex` 复合属性中三个子属性的值不可省略，顺序不可修改
- `flex` 复合属性还有以下简写方式

简写方式	对应全写	作用
<code>flex: auto</code>	<code>flex: 1 1 auto</code>	拉伸：√；压缩：√；基准长度：不设置
<code>flex: 1</code>	<code>flex: 1 1 0</code>	拉伸：√；压缩：√；基准长度：0（使用最多）

简写方式	对应全写	作用
<code>flex: none</code>	<code>flex: 0 0 auto</code>	拉伸: ×; 压缩: ×; 基准长度: 不设置
<code>flex: 0 auto</code>	<code>flex: 0 1 auto</code>	拉伸: ×; 压缩: √; 基准长度: 不设置

- 我们可以借助开发者工具，快捷调整弹性盒子布局相关属性



项目排序与单独对齐

- 项目排序：默认情况下，伸缩项目是根据源代码中的顺序在主轴方向上依次排列的，但是我们可以根据 `order` 属性调整伸缩项目出现的视觉顺序。`order` 数值越小，则排列越靠前，默认为 `0`。
- 单独对齐：我们可以通过 `align-self` 调整某个伸缩项目在侧轴方向上的对齐方式，默认值是 `auto`，表示继承父元素的 `align-items` 属性的属性值。`align-self` 属性的取值为 `flex-start`、`flex-end`、`center`、`baseline`、`stretch`。

[38-c3伸缩盒模型案例练习](#)

响应式布局

媒体查询之媒体类型

- 只有在打印机或打印预览时才应用的样式

```
@media print {
    /* CSS properties */
}
```

- 只有在屏幕（电脑屏幕、平板屏幕、手机屏幕）上才应用的样式

```
@media screen {  
    /* css properties */  
}
```

3. 在**所有设备**上都应用的样式

```
@media all {  
    /* css properties */  
}
```

4. 注：通过 `@media media-type {}` 语法，我们可以给一些特定的**设备**设置一些样式，限制只有在对应的设备上对应的样式才生效。

- `@media media-type {}` 是写在 .css 文件或 `<style></style>` 中的
- `@media media-type {}` 中的样式不存在任何优先级，因此，媒体查询中的样式写在所有样式的最后边，保证在特定设备中，媒体查询中的样式因为是后边的可以覆盖掉前边的样式

5. 其他**废弃**使用的媒体类型 [more info](#)

media-type	含义
<u>aural</u>	语音和声音合成器
<u>braille</u>	盲文触摸式反馈设备
<u>embossed</u>	打印的盲人印刷设备
<u>handheld</u>	掌上设备或更小装置，如 PDA、小型电话
<u>projection</u>	投影设备
<u>tty</u>	固定的字符网络，如电报、终端设备和对字符有限制的便携设备
<u>tv</u>	电视和网络电视

媒体查询之媒体特性

1. 在视口宽度为 `800px` 时应用的样式 (`viewport-width=800px`)

```
@media (width: 800px) {  
    /* css properties */  
}
```

2. 在视口宽度小于等于 `700px` 时应用的样式 (`viewport-width≤700px`)

```
@media (max-width: 800px) {  
    /* css properties */  
}
```

3. 在视口宽度大于等于 `900px` 时应用的样式 (`viewport-width≥900px`)

```

@media (min-width: 900px) {
    /* css properties */
}

```

4. 常用的媒体特性 [more info](#)

media-feature	含义
width	视口宽度 (等于, =)
max-width	视口最大宽度 (小于等于, ≤)
min-width	视口最小宽度 (大于等于, ≥)
height	视口高度 (等于, =)
max-height	视口最大高度 (小于等于, ≤)
min-height	视口最小高度 (大于等于, ≥)
device-width	设备屏幕宽度 (等于, =)
max-device-width	设备屏幕最大宽度 (小于等于, ≤)
min-device-width	设备屏幕最小宽度 (大于等于, ≥)
orientation	视口的旋转方向 (判断是否横屏) - portrait 视口纵向, 此时高度大于宽度 - landscape 视口横向, 此时宽度大于高度

媒体查询之运算符

- 在视口宽度大于等于 `700px` 且小于等于 `800px` 时应用的样式 ($700px \leq \text{viewport-width} \leq 800px$)

```

@media (min-width: 700px) and (max-width: 800px) {
    /* css properties */
}

```

- 在屏幕上且视口宽度大于等于 `700px` 且小于等于 `800px` 时应用的样式 ($700px \leq \text{viewport-width} \leq 800px$)

```

@media screen and (min-width: 700px) and (max-width: 800px) {
    /* css properties */
}

```

3. 在视口宽度大于等于 800px 或小于等于 700px 时应用的样式 (`viewport-width≥800px`, `viewport-width≤700px`)

```
@media (max-width: 700px) or (min-width: 800px) {  
    /* css properties */  
}
```

4. 在除了屏幕上应用的样式

```
@media not screen {  
    /* css properties */  
}
```

5. 在且只在屏幕上应用的样式

```
@media only screen {  
    /* css properties */  
}
```

6. 常用的运算符

运算符	含义
and	且
or 或 ,	或
not	否定
only	肯定

`only` 运算符没有什么太大的意义，但是其可以解决 IE 兼容性问题（因为 IE 不支持这个运算符）

常用阈值



四种类型屏幕的媒体查询方式如下,

1. 超小屏幕

```
@media screen and (max-width: 768px) {  
    /* css properties */  
}
```

2. 中等屏幕

```
@media screen and (min-width: 768px) and (max-width: 992px) {  
    /* css properties */  
}
```

3. 大屏幕

```
@media screen and (min-width: 992px) and (max-width: 1200px) {  
    /* css properties */  
}
```

4. 超大屏幕

```
@media screen and (min-width: 1200px) {  
    /* css properties */  
}
```

结合外部样式引入媒体查询样式

1. 方式一：正常方式

```
/* .css 文件 */  
/* css properties */  
/* css properties */  
/* css properties */  
/* css properties */  
@media xxx {  
    /* css properties */  
}
```

```
<!-- .html 文件 -->  
<link rel="stylesheet" href=".//xxx.css">
```

2. 方式二：将 .css 文件内容作为媒体查询样式引入

```
/* .css 文件 */  
/* css properties */  
/* css properties */  
/* css properties */
```

```
<!-- .html 文件 -->  
<link rel="stylesheet" media="xxx" href=".//xxx.css">
```

方式一、二中的 `xxx` 表示具体的媒体查询

BFC

1. 是什么？

- `BFC` 全称为 `Block Formatting Context`，翻译为块级格式上下文，可以理解为元素的一个特异功能。
- 该特异功能默认情况下处于**关闭**状态，当元素满足某些条件后，该特异功能才被激活。
- 激活元素的特异功能，专业来说，该元素创建了 `BFC`（或开启了 `BFC`）

2. 有何用? 开启了 `BFC` 的元素,

- 其子元素不会产生 `margin` 塌陷问题
- 自身不会被其他浮动元素所覆盖
- 子元素浮动时, 自身高度不会塌陷

3. 怎么设置?

- 根元素 `html`
- 浮动元素
- 绝对定位和固定定位的元素
- 行内块元素
- 表格单元格 `table`、`thead`、`tbody`、`tfoot`、`th`、`td`、`tr`、`caption`
- `overflow: hidden/scroll/auto` 的元素 (即不可取值为 `visible`)
- 伸缩项目
- 多列容器
- `column-span: all` 的元素
- `display: flow-root` 的元素

元素开启 `BFC` 副作用最低的一种方式, 唯一缺点是 IE 不支持