

一、前序知识

1. 认识两位先驱



艾伦·麦席森·图灵

- 二战时期，破译了德军的战争编码 — 英格玛。
- 让二战提前2年结束，拯救了上千万人的生命。
- 设立图灵奖，被后人誉为：**人工智能之父**

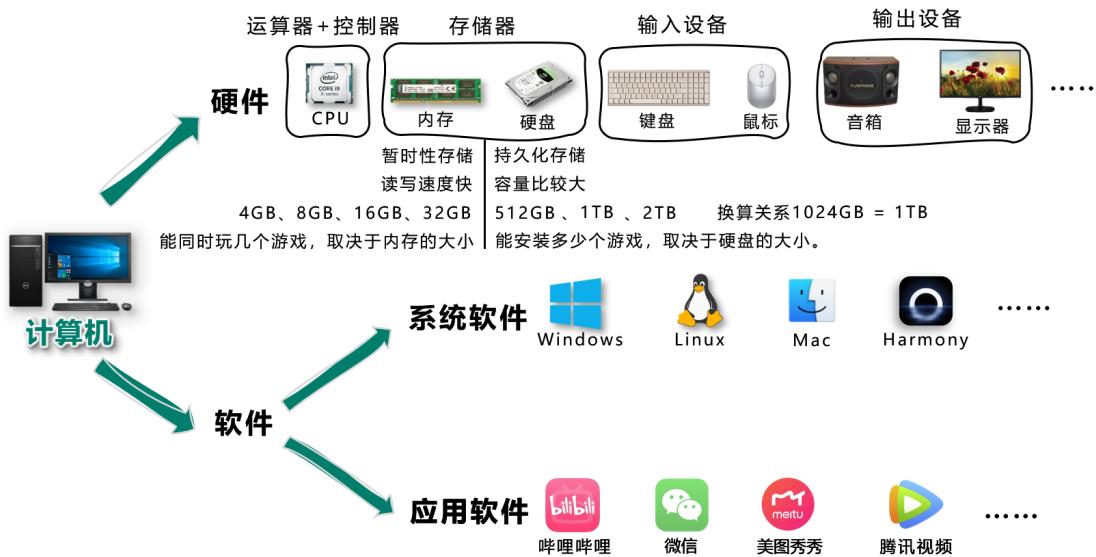


约翰·冯·诺依曼

- 制订了现代计算机标准 —— 冯诺依曼体系结构。
- 提出：计算机要采用二进制、明确计算机组成部分。
- 被后人誉为：**现代计算机之父**

2. 计算机基础知识

1. **计算机**俗称电脑，是现代一种用于高速计算的电子计算机器，可以进行数值计算、逻辑计算，还具有存储记忆功能。
2. 计算机由 **硬件 + 软件** 成：
 - 硬件：看得见摸得着的物理部件。
 - 软件：可以指挥硬件工作的指令。
3. 软件的分类：
 1. 系统软件：Windows、Linux、Android、Harmony等。
 2. 应用软件：微信、QQ、王者荣耀、PhotoShop等。
4. 整体图示：



3. C/S架构与B/S架构

1. 上面提到的应用软件，又分为两大类：

- **C/S架构**，特点：需要安装、偶尔更新、不跨平台、开发更具针对性。
- **B/S架构**，特点：无需安装、无需更新、可跨平台、开发更具通用性。

名词解释：C => client (客户端)、B => browser (浏览器)、S => server (服务器)。

服务器：为软件提供数据的设备（在背后默默的付出）。

2. 前端工程师，主要负责编写 B/S架构中的网页（呈现界面、实现交互）。

备注：大前端时代，我们可以用前端的技术栈，做出一个C/S架构的应用、甚至搭建一个服务器。

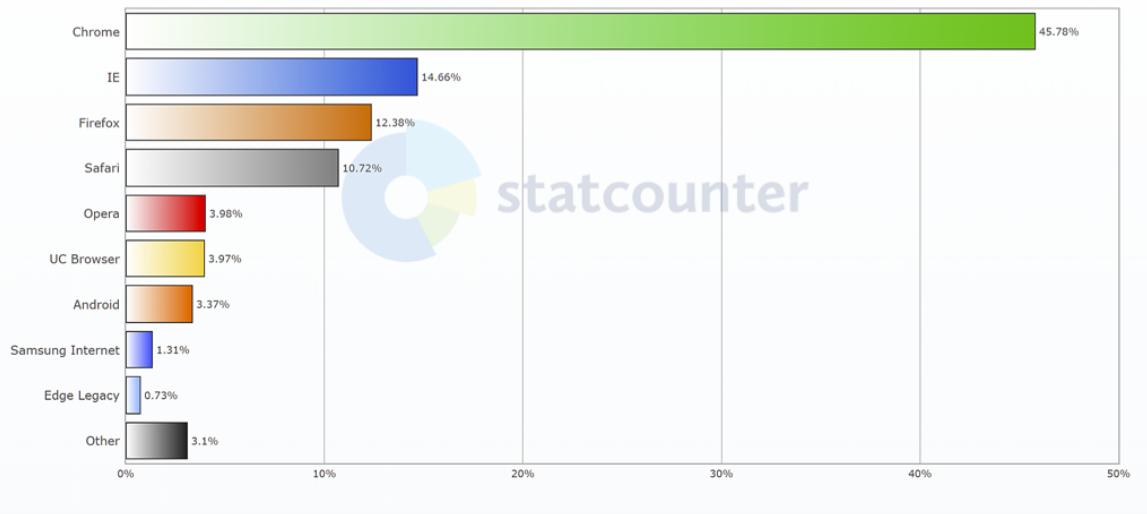
4. 浏览器相关知识

浏览器是网页运行的平台，常见的浏览器有：[谷歌\(Chrome\)](#)、[Safari](#)、[IE](#)、[火狐\(Firefox\)](#)、[欧朋\(Opera\)](#)等，以上这些是常用的五大浏览器。

1.各大浏览器市场份额：

Desktop & Mobile Browser Market Share Worldwide

Aug 2011 - Aug 2021



2. 常见浏览器的内核：



五大浏览器

四大内核



在上述内核的基础上，加上了一些精美的UI界面（皮肤）、实用的小功能。

5. 网页相关概念

1. 网址：我们在浏览器中输入的地址。
2. 网页：浏览器所呈现的每一个页面。
3. 网站：多个网页构成了一个网站。
4. 网页标准：



二、HTML简介

1. 什么是HTML？

全称：HyperText Markup Language（超文本标记语言）。

超文本：暂且简单理解为“超级的文本”，和普通文本比，内容更丰富。

标记：文本要变成超文本，就需要用到各种标记符号。

语言：每一个标记的写法、读音、使用规则，组成了一个标记语言。

2. 相关国际组织（了解）

1. IETF

全称：Internet Engineering Task Force（国际互联网工程任务组），成立于1985年底，是一个权威的互联网技术标准化组织，主要负责互联网相关技术规范的研发和制定，当前绝大多数国际互联网技术标准均出自IETF。官网：<https://www.ietf.org>

2. W3C

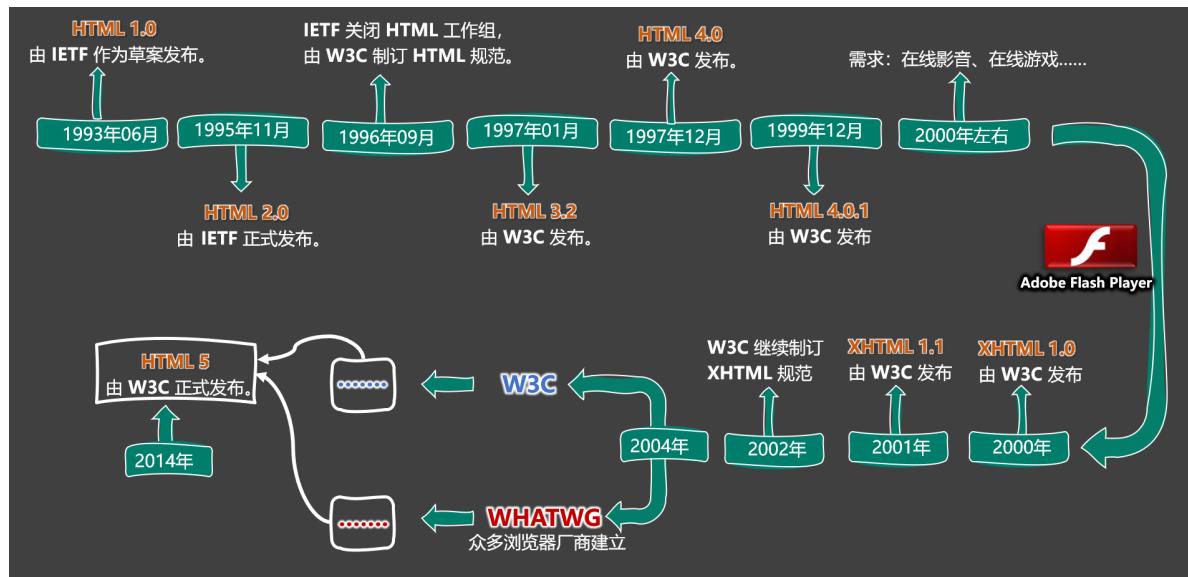
全称：World Wide Web Consortium（万维网联盟），创建于1994年，是目前Web技术领域，最具影响力的技术标准机构。共计发布了200多项技术标准和实施指南，对互联网技术的发展和应用起到了基础性和根本性的支撑作用，官网：<https://www.w3.org>

3. WHATWG

全称：Web Hypertext Application Technology Working Group（网页超文本应用技术工作小组）成立于2004年，是一个以推动网络HTML 5 标准为目的而成立的组织。由Opera、Mozilla基金会、苹果，等这些浏览器厂商组成。官网：<https://whatwg.org/>

3. HTML发展历史（了解）

从 HTML 1.0 开始发展，期间经历了很多版本，目前HTML的最新标准是：HMTL 5，具体发展史如图（了解即可）。



三、准备工作

1. 常用电脑设置

1. 查看文件夹内容的几种布局。
2. 展示文件扩展名（文件后缀）。
3. 使用指定程序打开文件。
4. 配置文件的默认打开方式。

2. 安装Chrome浏览器

1. 下载地址：<https://www.google.cn/chrome>。
2. 若上述地址打不开，或无法安装，请使用资料中的离线安装包。
3. 详细步骤请参考视频。

四、HTML入门

1. HTML初体验

1. 第一步：鼠标右键 => 新建 => 文本文档 => 输入以下内容，并保存。

```
<marquee>尚硅谷，让天下没有难学的技术！</marquee>
```

2. 第二步：修改后缀为 **.html**，然后双击打开即可。

这里的后缀名，使用 **.htm** 也可以，但推荐使用更标准的 **.html**。

3. 程序员写的叫 **源代码**，要交给浏览器进行渲染。
4. 借助浏览器看网页的 **源代码**，具体操作：

在网页空白处：鼠标右键 ==> 查看网页源代码

2. HTML 标签

1. **标签** 又称 **元素**，是HTML的基本组成单位。
2. 标签分为：**双标签** 与 **单标签**（绝大多数都是双标签）。
3. 标签名不区分大小写，但推荐小写，因为小写更规范。
4. 双标签：



示例代码：

```
<marquee>尚硅谷，让天下没有难学的技术！</marquee>
```

5. 单标签：



示例代码：

```
<input>
```

6. 标签之间的关系：并列关系、嵌套关系，可以使用 **tab** 键进行缩进：

```
<marquee>
    尚硅谷，让天下没有难学的技术！
<input>
</marquee>
<input>
```

3. HTML 标签属性

1. 用于给标签提供 **附加信息**。
2. 可以写在：**起始标签** 或 **单标签中**，形式如下：



例如：

```
<marquee loop="1" bgcolor="orange">尚硅谷，让天下没有难学的技术！</marquee>
<input type="password">
```

- 有些特殊的属性，没有属性名，只有属性值，例如：

```
<input disabled>
```

- 注意点：

- 不同的标签，有不同的属性；也有一些通用属性（在任何标签内都能写，后面会详细总结）。
- 属性名、属性值不能乱写，都是W3C规定好的。
- 属性名、属性值，都不区分大小写，但推荐小写。
- 双引号，也可以写成单引号，甚至不写都行，但还是推荐写双引号。
- 标签中不要出现同名属性，否则后写的会失效，例如：

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

4. HTML 基本结构

- 在网页中，如何查看某段结构的具体代码？——点击鼠标右键，选择“检查”。
- 【检查】和【查看网页源代码】的区别：

【查看网页源代码】看到的是：程序员编写的源代码。

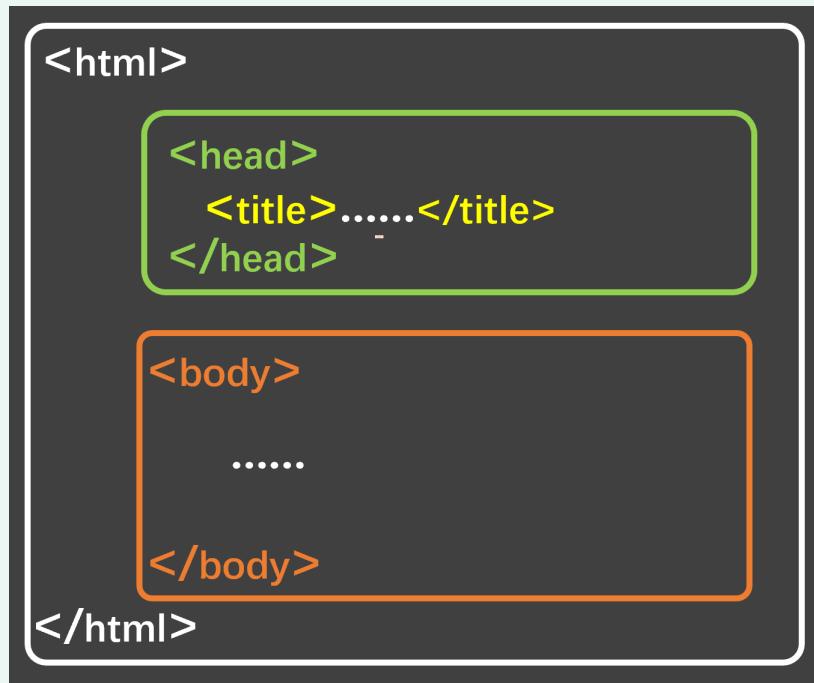
【检查】看到的是：经过浏览器“处理”后的源代码。

备注：日常开发中，【检查】用的最多。

- 网页的基本结构如下：

- 想要呈现在网页中的内容写在 **body** 标签中。
- head** 标签中的内容不会出现在网页中。
- head** 标签中的 **title** 标签可以指定网页的标题。

4. 图示：



5. 代码：

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

5. 安装 VSCode

1. 安装中文语言包。
2. 使用 VSCode 打开文件夹的两种方式。
3. 调整字体大小。
4. 设置主题。
5. 安装图标主题：`vscode-icons`。

备注：详细安装步骤请参考视频。

6. 安装 Live Server 插件

1. 可以更加方便的打开网页。
2. 打开网页的方式更贴近项目上线。
3. 代码出现改动后，可以自动刷新。
4. 根据自己的情况，去配置一下 VSCode 的自动保存。

注意1：务必使用VSCode打开的是文件夹，否则 Live Server 插件无法正常工作！

注意2：打开的网页必须是标准的HTML结构，否则无法自动刷新！

7. HTML 注释

- 特点：注释的内容会被浏览器所忽略，不会呈现到页面中，但源代码中依然可见。
- 作用：对代码进行解释和说明。
- 写法：

```
<!-- 下面的文字只能滚动一次 -->
<marquee loop="1">尚硅谷</marquee>

<!-- 下面的文字可以无限滚动 -->
<marquee>尚硅谷123</marquee>
```

- 注释不可以嵌套，以下这么写是错的（反例）。

```
<!--
我是一段注释
<!-- 我是一段注释 -->
-->
```

8. HTML 文档声明

- 作用：告诉浏览器当前网页的版本。
- 写法：
 - 旧写法：要依网页所用的HTML版本而定，写法有很多。

具体有哪些写法请参考：[W3C官网-文档声明](#)（了解即可，千万别背！）

- 新写法：一切都变得简单了！W3C 推荐使用 HTML 5 的写法。

```
<!DOCTYPE html>
或
<!DOCTYPE HTML>
或
<!doctype html>
```

- 注意：文档声明，必须在网页的第一行，且在 `html` 标签的外侧。

9. HTML 字符编码

- 计算机对数据的操作：
 - 存储时，对数据进行：**编码**。
 - 读取时，对数据进行：**解码**。
- 编码、解码，会遵循一定的规范——**字符集**。
- 字符集有很多中，常见的有（了解）：

1. **ASCII**：大写字母、小写字母、数字、一些符号，共计128个。
2. **ISO 8859-1**：在 ASCII 基础上，扩充了一些希腊字符等，共计是256个。
3. **GB2312**：继续扩充，收录了 6763 个常用汉字、682个字符。
4. **GBK**：收录了的汉字和符号达到 20000+，支持繁体中文。
5. **UTF-8**：包含世界上所有语言的：所有文字与符号。—— 很常用。

4. 使用原则是怎样的？

原则1：存储时，务必采用合适的字符编码。

否则：无法存储，数据会丢失！

原则2：存储时采用哪种方式编码，读取时就采用哪种方式解码。

否则：数据错乱（乱码）！

例如下面文字中，包含有：中文、英文、泰文、缅甸文

```
我爱你  
I love you!  
ສັນຮັກເຮອນະ  
ကျွန်မကို ချစ်တယ်။
```

若使用 **ISO8859-1** 编码存储，在存入的那一刻，就出问题了，因为 **ISO8859-1** 仅支持英文！

为保证所有的输入，都能正常存储和读取，现在几乎全都采用：**UTF-8** 编码。

所以我们编写 **html** 文件时，也都统一用 **UTF-8** 编码。

5. 总结：

- 平时编写代码时，统一采用 **UTF-8** 编码（最稳妥）。
- 为了让浏览器在渲染 **html** 文件时，不犯错误，可以通过 **meta** 标签配合 **charset** 属性指定字符编码。

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

10. HTML 设置语言

1. 主要作用：

- 让浏览器显示对应的翻译提示。
- 有利于搜索引擎优化。

2. 具体写法：

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

3. 扩展知识：**lang** 属性的编写规则（作为一个课外扩展知识，了解即可）。

1. 第一种写法（语言-国家/地区），例如：
 - zh-CN : 中文-中国大陆（简体中文）
 - zh-TW : 中文-中国台湾（繁体中文）
 - zh : 中文
 - en-US : 英语-美国
 - en-GB : 英语-英国
2. 第二种写法（语言—具体种类）已不推荐使用，例如：
 - zh-Hans : 中文-简体
 - zh-Hant : 中文-繁体
3. W3School 上的说明：《[语言代码参考手册](#)》、《[国家/地区代码参考手册](#)》
4. W3C官网上的说明：《[Language tags in HTML](#)》

11. HTML标准结构

- HTML标准结构如下：

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <title>我是一个标题</title>
  </head>
  <body>

  </body>
</html>
```

- 输入 `!`，随后回车即可快速生成标准结构。

生成的结构中，有两个meta标签，我们暂时用不到，可以先删掉。

- 配置 VScode 的内置插件 emmet，可以对生成结构的属性进行定制。
- 在存放代码的文件夹中，存放一个 `favicon.ico` 图片，可配置网站图标。

五、HTML 基础

1. 开发者文档

- W3C官网：www.w3c.org
- W3School：www.w3school.com.cn
- MDN：developer.mozilla.org — 平时用的最多。

2. 排版标签

标签名	标签含义	单 / 双 标签
h1 ~ h6	标题	双
p	段落	双
div	没有任何含义，用于整体布局（生活中的包装袋）。	双

1. h1 最好写一个， h2~h6 能适当多写。
2. h1~h6 不能互相嵌套，例如： h1 标签中最好不要写 h2 标签了。
3. p 标签很特殊！它里面不能有： h1~h6 、 p 、 div 标签（暂时先这样记，后面会说规律）。

3. 语义化标签

- 概念：用特定的标签，去表达特定的含义。
- 原则：标签的默认效果不重要（后期可以通过 CSS 随便控制效果），语义最重要！
- 举例：对于 h1 标签，效果是文字很大（不重要），语义是网页主要内容（很重要）。
- 优势：
 - 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、盲人阅读器等）。

4. 块级元素与行内元素

1. **块级元素**：独占一行（排版标签都是块级元素）。
2. **行内元素**：不独占一行（目前只学了： input ，稍后会学习更多）。
3. **使用原则**：

1. 块级元素 中能写 行内元素 和 块级元素 （简单记：块级元素中几乎什么都能写）。
2. 行内元素 中能写 行内元素，但不能写 块级元素。
3. 一些特殊的规则：
 - h1~h6 不能互相嵌套。
 - p 中不要写块级元素。

备注： marquee 元素设计的初衷是：让文字有动画效果，但如今我们可以通过 CSS 来实现了，而且还可以实现的更加炫酷，所以 marquee 标签已经： 过时了（废弃了），不推荐使用。我们只是在开篇的时候，用他做了一个引子而已，在后续的学习过程中，这些已经废弃的标签，我们直接跳过。

5. 文本标签_常用的

1. 用于包裹：词汇、短语等。
2. 通常写在排版标签里面。
3. 排版标签更宏观（大段的文字），文本标签更微观（词汇、短语）。

4. 文本标签通常都是行内元素。

标签名	标签语义	单 / 双 标 签
em	要着重阅读的内容	双
strong	十分重要的内容（语气比em要强）	双
span	没有语义，用于包裹短语的通用容器	双

生活中的例子：`div` 是大包装袋，`span` 是小包装袋。

6. 文本标签_不常用的

标签名	标签语义	单 / 双 标 签
<code>cite</code>	作品标题（书籍、歌曲、电影、电视节目、绘画、雕塑）	双
<code>dfn</code>	特殊术语，或专属名词	双
<code>del</code> 与 <code>ins</code>	删除的文本【与】插入的文本	双
<code>sub</code> 与 <code>sup</code>	下标文字【与】上标文字	双
<code>code</code>	一段代码	双
<code>samp</code>	从正常的上下文中，将某些内容提取出来，例如：标识设备输出	双
<code>kbd</code>	键盘文本，表示文本是通过键盘输入的，经常用在与计算机相关的手册中	双
<code>abbr</code>	缩写，最好配合上 <code>title</code> 属性	双
<code>bdo</code>	更改文本方向，要配合 <code>dir</code> 属性，可选值： <code>ltr</code> （默认值）、 <code>rtl</code>	双
<code>var</code>	标记变量，可以与 <code>code</code> 标签一起使用	双
<code>small</code>	附属细则，例如：包括版权、法律文本。——很少使用	双
<code>b</code>	摘要中的关键字、评论中的产品名称。——很少使用	双
<code>i</code>	本意是：人物的思想活动、所说的话等等。 现在多用于：呈现字体图标（后面要讲的内容）。	双
<code>u</code>	与正常内容有反差文本，例如：错的单词、不合适的描述等。——很少使用	双
<code>q</code>	短引用——很少使用	双
<code>blockquote</code>	长引用——很少使用	双
<code>address</code>	地址信息	双

备注：

1. 这些不常用的文本标签，编码时不用过于纠结（酌情而定，不用也没毛病）。
2. `blockquote` 与 `address` 是块级元素，其他的文本标签，都是行内元素。
3. 有些语义感不强的标签，我们很少使用，例如：
`small`、`b`、`u`、`q`、`blockquote`
4. HTML标签太多了！记住那些：重要的、语义感强的标签即可；截止目前，有这些：
`h1~h6`、`p`、`div`、`em`、`strong`、`span`

7. 图片标签

1. 基本使用

标签名	标签语义	常用属性	单/双标签
<code>img</code>	图片	<code>src</code> : 图片路径（又称：图片地址）——图片的具体位置 <code>alt</code> : 图片描述 <code>width</code> : 图片宽度，单位是像素，例如： <code>200px</code> 或 <code>200</code> <code>height</code> : 图片高度，单位是像素，例如： <code>200px</code> 或 <code>200</code>	单

总结：

1. 像素（`px`）是一种单位，学到 `CSS` 时，我们会详细讲解。
2. 尽量不同时修改图片的宽和高，可能会造成比例失调。
3. 暂且认为 `img` 是行内元素（学到 `CSS` 时，会认识一个新的元素分类，目前咱们只知道：块、行内）。
4. `alt` 属性的作用：
 - 搜索引擎通过 `alt` 属性，得知图片的内容。——最主要的作用。
 - 当图片无法展示时候，有些浏览器会呈现 `alt` 属性的值。
 - 盲人阅读器会朗读 `alt` 属性的值。

2. 路径的分类

1. **相对路径**：以**当前位置**作为参考点，去建立路径。

已有结构	符号	含义	举例（在 <code>测试.html</code> 中）
	<code>.</code> <code>/</code>	同级	引入【怪兽.jpg】： <code></code>
	<code>/</code>	下一级	引入【喜羊羊.jpg】： <code></code>
	<code>.. /</code>	上一级	引入【奥特曼.jpg】： <code></code>

注意点：

- 相对路径中的 `./` 可以省略不写。
- 相对路径依赖的是当前位置，后期若调整了文件位置，那么文件中的路径也要修改。

2. 绝对路径：以**根位置**作为参考点，去建立路径。

1. 本地绝对路径：`E:/a/b/c/奥特曼.jpg`。（很少使用）
2. 网络绝对路径：`http://www.atguigu.com/images/index_new/logo.png`。

注意点：

- 使用本地绝对路径，一旦更换设备，路径处理起来比较麻烦，所以很少使用。
- 使用网络绝对路径，确实方便，但要注意：若服务器开启了防盗链，会造成图片引入失败。

3. 常见图片格式

1. `jpg` 格式：

概述：扩展名为 `.jpg` 或 `.jpeg`，是一种有损的压缩格式（把肉眼不容易观察出来的细节丢弃了）。

主要特点：**支持的颜色丰富、占用空间较小**、不支持透明背景、不支持动态图。

使用场景：对图片细节**没有极高要求**的场景，例如：网站的产品宣传图等。——该格式网页中很常见。

2. `png` 格式：

概述：扩展名为 `.png`，是一种无损的压缩格式，能够更高质量的保存图片。

主要特点：**支持的颜色丰富、占用空间略大、支持透明背景**、不支持动态图。

使用场景：①想让图片有透明背景；②想更高质量的呈现图片；例如：公司logo图、重要配图等。

3. `bmp` 格式：

概述：扩展名为 `.bmp`，不进行压缩的一种格式，在最大程度上保留图片更多的细节。

主要特点：**支持的颜色丰富、保留的细节更多**、占用空间极大、不支持透明背景、不支持动态图。

使用场景：对图片细节**要求极高的**场景，例如：一些大型游戏中的图片。（网页中很少使用）

4. `gif` 格式：

概述：扩展名为 `.gif`，仅支持256种颜色，色彩呈现不是很完整。

主要特点：支持的颜色较少、**支持简单透明背景、支持动态图**。

使用场景：网页中的动态图片。

5. `webp` 格式：

概述：扩展名为 `.webp`，谷歌推出的一种格式，专门用来在网页中呈现图片。

主要特点：具备上述几种格式的优点，但兼容性不太好，一旦使用务必要解决兼容性问题。

使用场景：网页中的各种图片。

6. `base64` 格式

1. 本质：一串特殊的文本，要通过浏览器打开，传统看图应用通常无法打开。
2. 原理：把图片进行 `base64` 编码，形成一串文本。
3. 如何生成：靠一些工具或网站。
4. 如何使用：直接作为 `img` 标签的 `src` 属性的值即可，并且不受文件位置的影响。
5. 使用场景：一些较小的图片，或者需要和网页一起加载的图片。

图片的格式非常多，上面这些，只是一些常见的、我们前端人员常接触到的。

8. 超链接

主要作用：从当前页面进行跳转。

可以实现：①跳转到指定页面、②跳转到指定文件（也可触发下载）、③跳转到锚点位置、④唤起指定应用。

标签名	标签语义	常用属性	单 / 双 标签
a	超链接	<code>href</code> : 指定要跳转到的具体目标。 <code>target</code> : 控制跳转时如何打开页面，常用值如下： <code>_self</code> : 在本窗口打开。 <code>_blank</code> : 在新窗口打开。 <code>id</code> : 元素的唯一标识，可用于设置锚点。 <code>name</code> : 元素的名字，写在 <code>a</code> 标签中，也能设置锚点。	双

1. 跳转到页面

```
<!-- 跳转其他网页 -->
<a href="https://www.jd.com/" target="_blank">去京东</a>

<!-- 跳转本地网页 -->
<a href=". /10_HTML排版标签.html" target="_self">去看排版标签</a>
```

注意点：

1. 代码中的多个空格、多个回车，都会被浏览器解析成一个空格！
2. 虽然 `a` 是行内元素，但 `a` 元素可以包裹除它自身外的任何元素！

想展示多个回车或空格，怎么办呢？——后面会讲。

2. 跳转到文件

```
<!-- 浏览器能直接打开的文件 -->
<a href=".resource/自拍.jpg">看自拍</a>
<a href=".resource/小电影.mp4">看小电影</a>
<a href=".resource/小姐姐.gif">看小姐姐</a>
<a href=".resource/如何一夜暴富.pdf">点我一夜暴富</a>

<!-- 浏览器不能打开的文件，会自动触发下载 -->
<a href=".resource/内部资源.zip">内部资源</a>

<!-- 强制触发下载 -->
<a href=".resource/小电影.mp4" download="电影片段.mp4">下载电影</a>
```

注意1：若浏览器无法打开文件，则会引导用户下载。

注意2：若想强制触发下载，请使用 `download` 属性，属性值即为下载文件的名称。

3. 跳转到锚点

什么是锚点？——网页中的一个标记点。

具体使用方式：

- 第一步：设置锚点

```
<!-- 第一种方式：a标签配合name属性 -->
<a name="test1"></a>

<!-- 第二种方式：其他标签配合id属性 -->
<h2 id="test2">我是一个位置</h2>
```

注意点：

1. 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。
2. `name` 和 `id` 都是区分大小写的，且 `id` 最好别是数字开头。

- 第二步：跳转锚点

```
<!-- 跳转到test1锚点-->
<a href="#test1">去test1锚点</a>

<!-- 跳到本页面顶部 -->
<a href="#">回到顶部</a>

<!-- 跳转到其他页面锚点 -->
<a href="demo.html#test1">去demo.html页面的test1锚点</a>

<!-- 刷新本页面 -->
<a href="">刷新本页面</a>

<!-- 执行一段js，如果还不知道执行什么，可以留空，javascript:; -->
<a href="javascript:alert(1);">点我弹窗</a>
```

4. 唤起指定应用

通过 `a` 标签，可以唤起设备应用程序。

```
<!-- 唤起设备拨号 -->
<a href="tel:10010">电话联系</a>
<!-- 唤起设备发送邮件 -->
<a href="mailto:10010@qq.com">邮件联系</a>
<!-- 唤起设备发送短信 -->
<a href="sms:10086">短信联系</a>
```

9. 列表

1. 有序列表

概念：有顺序或侧重顺序的列表。

```
<h2>要把大象放冰箱总共分几步</h2>
<ol>
    <li>把冰箱门打开</li>
    <li>把大象放进去</li>
    <li>把冰箱门关上</li>
</ol>
```

2. 无序列表

概念：无顺序或不侧重顺序的列表。

```
<h2>我想去的几个城市</h2>
<ul>
    <li>成都</li>
    <li>上海</li>
    <li>西安</li>
    <li>武汉</li>
</ul>
```

3. 列表嵌套

概念：列表中的某项内容，又包含一个列表（注意：嵌套时，请务必把解构写完整）。

```
<h2>我想去的几个城市</h2>
<ul>
    <li>成都</li>
    <li>
        <span>上海</span>
        <ul>
            <li>外滩</li>
            <li>杜莎夫人蜡像馆</li>
            <li>
                <a href="https://www.opg.cn/">东方明珠</a>
            </li>
            <li>迪士尼乐园</li>
        </ul>
    </li>
</ul>
```

```
<li>西安</li>
<li>武汉</li>
</ul>
```

注意：li 标签最好写在 ul 或 ol 中，不要单独使用。

4. 自定义列表

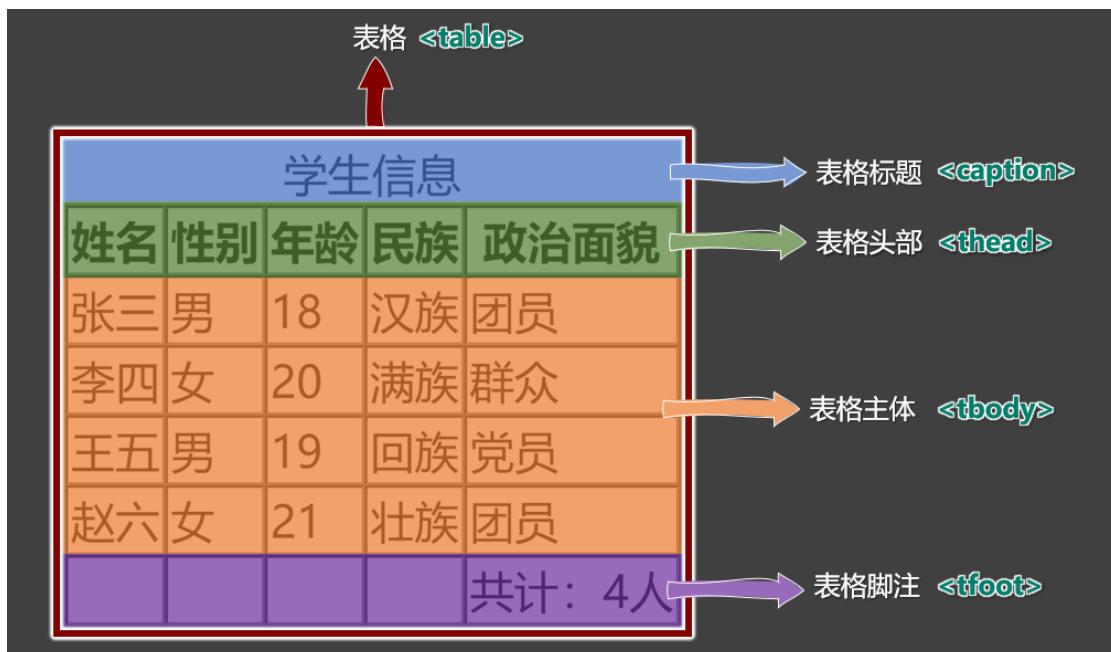
- 概念：所谓自定义列表，就是一个包含术语名称以及术语描述的列表。
- 一个 dl 就是一个自定义列表，一个 dt 就是一个术语名称，一个 dd 就是术语描述（可以有多个）。

```
<h2>如何高效的学习？</h2>
<dl>
    <dt>做好笔记</dt>
    <dd>笔记是我们以后复习的一个抓手</dd>
    <dd>笔记可以是电子版，也可以是纸质版</dd>
    <dt>多加练习</dt>
    <dd>只有敲出来的代码，才是自己的</dd>
    <dt>别怕出错</dt>
    <dd>错很正常，改正后并记住，就是经验</dd>
</dl>
```

10. 表格

1. 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。



- 表格涉及到的标签：

table : 表格

caption : 表格标题

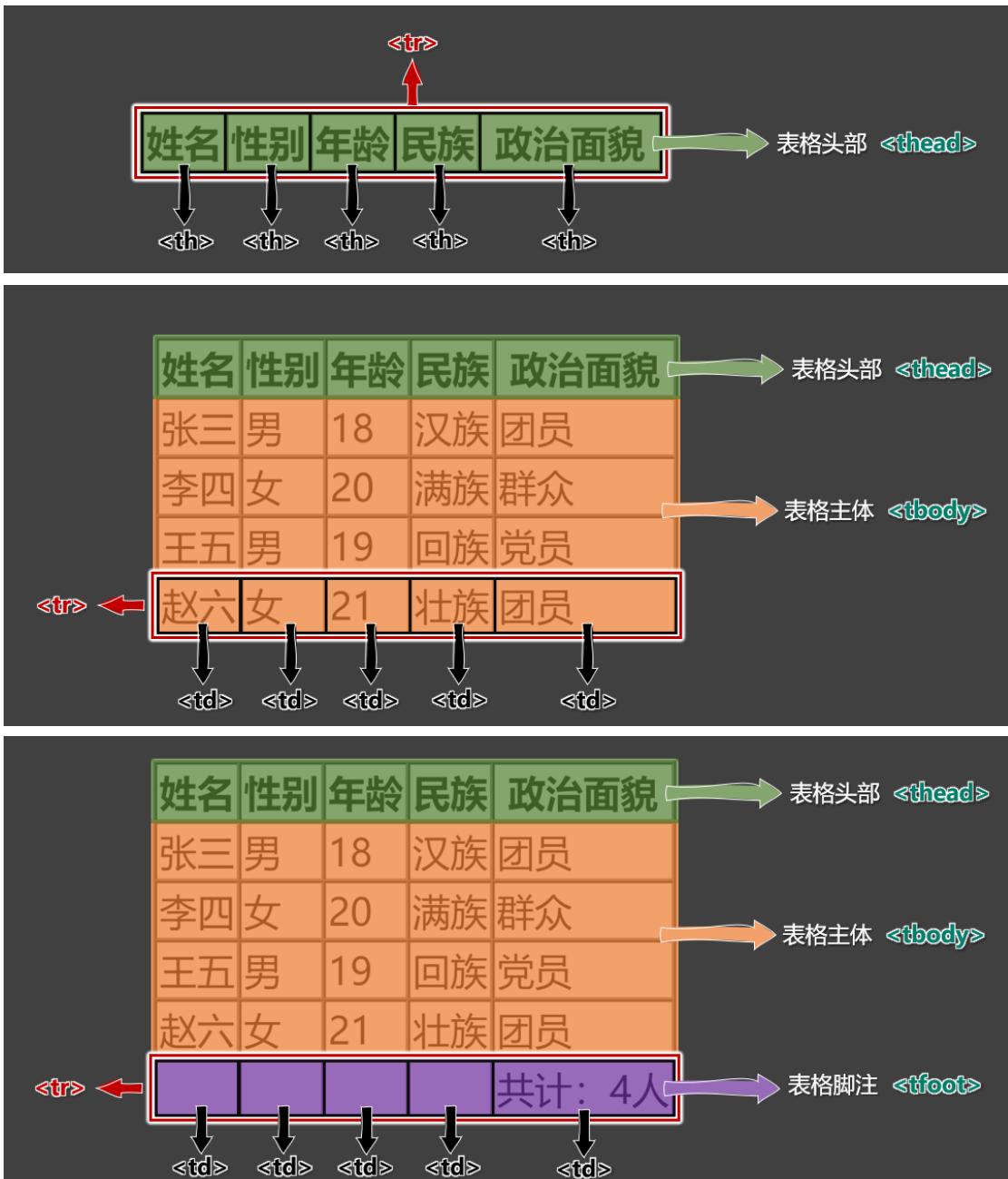
thead : 表格头部

`tbody` : 表格主体

`tfoot` : 表格脚注

`tr` : 每一行

`th`、`td` : 每一个单元格 (备注: 表格头部中用 `th`, 表格主体、表格脚注中用: `td`)



3. 具体编码：

```
<table border="1">
    
    <caption>学生信息</caption>
    
    <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>
    <tr>
        <td>张三</td>
        <td>男</td>
        <td>18</td>
        <td>汉族</td>
        <td>团员</td>
    </tr>
    <tr>
        <td>李四</td>
        <td>女</td>
        <td>20</td>
        <td>满族</td>
        <td>群众</td>
    </tr>
    <tr>
        <td>王五</td>
        <td>男</td>
        <td>20</td>
        <td>回族</td>
        <td>党员</td>
    </tr>
    <tr>
        <td>赵六</td>
        <td>女</td>
        <td>21</td>
        <td>壮族</td>
        <td>团员</td>
    </tr>
</tbody>
<!-- 表格脚注 -->
<tfoot>
    <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td>共计: 4人</td>
    </tr>
</tfoot>
</table>
```

2. 常用属性

标签名	标签语义	常用属性	单 / 双标签
table	表格	<p><code>width</code> : 设置表格宽度。</p> <p><code>height</code> : 设置表格最小高度，表格最终高度可能比设置的值大。</p> <p><code>border</code> : 设置表格边框宽度。</p> <p><code>cellspacing</code> : 设置单元格之间的间距。</p>	双
thead	表格头部	<p><code>height</code> : 设置表格头部高度。</p> <p><code>align</code> : 设置单元格的水平对齐方式，可选值如下：</p> <ol style="list-style-type: none"> 1. <code>left</code> : 左对齐 2. <code>center</code> : 中间对齐 3. <code>right</code> : 右对齐 <p><code>valign</code> : 设置单元格的垂直对齐方式，可选值如下：</p> <ol style="list-style-type: none"> 1. <code>top</code> : 顶部对齐 2. <code>middle</code> : 中间对齐 3. <code>bottom</code> : 底部对齐 	双
tbody	表格主体	常用属性与 <code>thead</code> 相同。	双
tr	行	常用属性与 <code>thead</code> 相同。	双
tfoot	表格脚注	常用属性与 <code>thead</code> 相同。	双
td	普通单元格	<p><code>width</code> : 设置单元格的宽度，同列所有单元格全都受影响。</p> <p><code>height</code> : 设置单元格的高度，同行所有单元格全都受影响。</p> <p><code>align</code> : 设置单元格的水平对齐方式。</p> <p><code>valign</code> : 设置单元格的垂直对齐方式。</p> <p><code code="" rowspan<=""> : 指定要跨的行数。</code></p> <p><code code="" colspan<=""> : 指定要跨的列数。</code></p>	双
th	表头单元格	常用属性与 <code>td</code> 相同。	双

注意点：

1. `<table>` 元素的 `border` 属性可以控制表格边框，但 `border` 值的大小，并不控制单元格边框的宽度，
只能控制表格最外侧边框的宽度，这个问题如何解决？——后期靠 CSS 控制。
2. 默认情况下，每列的宽度，得看这一列单元格最长的那个文字。
3. 给某个 `th` 或 `td` 设置了宽度之后，他们所在的一列的宽度就确定了。
4. 给某个 `th` 或 `td` 设置了高度之后，他们所在的一行的高度就确定了。

3. 跨行跨列

1. **rowspan**：指定要跨的行数。
2. **colspan**：指定要跨的列数。

课程表效果：

项目	上课					活动与休息	
	星期一	星期二	星期三	星期四	星期五	星期六	星期日
上午	语文	数学	英语	英语	物理	数学竞赛	休息
	数学	语文	化学	物理	英语	篮球比赛	
	化学	语文	体育	历史	地理	每周一考	
	体育	化学	语文	数学	英语	社会实践	
下午	语文	英语	数学	物理	数学	英语角	休息
	化学	物理	地理	生物	体育	自由活动	

编写思路：

1-1	1-2				1-7		
2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8
3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8
	4-2	4-3	4-4	4-5	4-6	4-7	
	5-2	5-3	5-4	5-5	5-6	5-7	
	6-2	6-3	6-4	6-5	6-6	6-7	
7-1	7-2	7-3	7-4	7-5	7-6	7-7	7-8
	8-2	8-3	8-4	8-5	8-6	8-7	

11. 常用标签补充

标签名	标签含义	单 / 双 标签
br	换行	单
hr	分隔	单
pre	按原文显示（一般用于在页面中嵌入大段代码）	双

注意点：

- 不要用 `
` 来增加文本之间的行间隔，应使用 `<p>` 元素，或后面即将学到的 CSS `margin` 属性。
- `<hr>` 的语义是分隔，如果不想要语义，只是想画一条水平线，那么应当使用 CSS 完成。

12. 表单

概念：一个包含交互的区域，用于收集用户提供的数据。

1. 基本结构

简单梳理：

标签名	标签语义	常用属性	单 / 双 标签
form	表单	<code>action</code> ：用于指定表单的提交地址（需要与后端人员沟通后确定）。 <code>target</code> ：用于控制表单提交后，如何打开页面，常用值如下： <code>_self</code> ：在本窗口打开。 <code>_blank</code> ：在新窗口打开。 <code>method</code> ：用于控制表单的提交方式，暂时只需了解，在后面 Ajax 的课程中，会详细讲解。	双
input	输入框	<code>type</code> ：设置输入框的类型，目前用到的值是 <code>text</code> ，表示普通文本。 <code>name</code> ：用于指定提交数据的名字，（需要与后端人员沟通后确定）。	单
button	按钮	本小节暂不涉及	双

在本小节，我们先记住表单的整体形式，稍后会对表单控件进行详细讲解。

示例代码：

```

<form action="https://www.baidu.com/s" target="_blank" method="get">
  <input type="text" name="wd">
  <button>去百度搜索</button>
</form>

```

2. 常用表单控件

① 文本输入框

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

常用属性如下：

name 属性：数据的名称。

value 属性：输入框的默认输入值。

maxlength 属性：输入框最大可输入长度。

② 密码输入框

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

常用属性如下：

name 属性：数据的名称。

value 属性：输入框的默认输入值（一般不用，无意义）。

maxlength 属性：输入框最大可输入长度。

③ 单选框

```
<input type="radio" name="sex" value="female">女  
<input type="radio" name="sex" value="male">男
```

常用属性如下：

name 属性：数据的名称，注意：想要单选效果，多个 **radio** 的 **name** 属性值要保持一致。

value 属性：提交的数据值。

checked 属性：让该单选按钮默认选中。

④ 复选框

```
<input type="checkbox" name="hobby" value="smoke">抽烟  
<input type="checkbox" name="hobby" value="drink">喝酒  
<input type="checkbox" name="hobby" value="perm">烫头
```

常用属性如下：

name 属性：数据的名称。

value 属性：提交的数据值。

checked 属性：让该复选框默认选中。

⑤ 隐藏域

```
<input type="hidden" name="tag" value="100">
```

用户不可见的一个输入区域，作用是：提交表单的时候，携带一些固定的数据。

`name` 属性：指定数据的名称。

`value` 属性：指定的是真正提交的数据。

⑥ 提交按钮

```
<input type="submit" value="点我提交表单">  
<button>点我提交表单</button>
```

注意：

1. `button` 标签 `type` 属性的默认值是 `submit`。
2. `button` 不要指定 `name` 属性
3. `input` 标签编写的按钮，使用 `value` 属性指定按钮文字。

⑦ 重置按钮

```
<input type="reset" value="点我重置">  
<button type="reset">点我重置</button>
```

注意点：

1. `button` 不要指定 `name` 属性
2. `input` 标签编写的按钮，使用 `value` 属性指定按钮文字。

⑧ 普通按钮

```
<input type="button" value="普通按钮">  
<button type="button">普通按钮</button>
```

注意点：普通按钮的 `type` 值为 `button`，若不写 `type` 值是 `submit` 会引起表单的提交。

⑨ 文本域

```
<textarea name="msg" rows="22" cols="3">我是文本域</textarea>
```

常用属性如下：

1. `rows` 属性：指定默认显示的行数，会影响文本域的高度。
2. `cols` 属性：指定默认显示的列数，会影响文本域的宽度。
3. 不能编写 `type` 属性，其他属性，与普通文本输入框一致。

⑩ 下拉框

```
<select name="from">
    <option value="黑">黑龙江</option>
    <option value="辽">辽宁</option>
    <option value="吉">吉林</option>
    <option value="粤" selected>广东</option>
</select>
```

常用属性及注意事项：

1. `name` 属性：指定数据的名称。
2. `option` 标签设置 `value` 属性，如果没有 `value` 属性，提交的数据是 `option` 中间的文字；如果设置了 `value` 属性，提交的数据就是 `value` 的值（建议设置 `value` 属性）
3. `option` 标签设置了 `selected` 属性，表示默认选中。

3. 禁用表单控件

给表单控件的标签设置 `disabled` 既可禁用表单控件。

`input`、`textarea`、`button`、`select`、`option` 都可以设置 `disabled` 属性

4. label 标签

`label` 标签可与表单控件相关联，关联之后点击文字，与之对应的表单控件就会获取焦点。

两种与 `label` 关联方式如下：

1. 让 `label` 标签的 `for` 属性的值等于表单控件的 `id`。
2. 把表单控件套在 `label` 标签的里面。

5. fieldset 与 legend 的使用（了解）

`fieldset` 可以为表单控件分组、`legend` 标签是分组的标题。

示例：

```
<fieldset>
    <legend>主要信息</legend>
    <label for="zhanghu">账户：</label>
    <input id="zhanghu" type="text" name="account" maxlength="10"><br>
    <label>
        密码：
        <input id="mima" type="password" name="pwd" maxlength="6">
    </label>
    <br>
    性别：
    <input type="radio" name="gender" value="male" id="nan">
    <label for="nan">男</label>
    <label>
        <input type="radio" name="gender" value="female" id="nv">女
    </label>
</fieldset>
```

6. 表单总结

标签名	标签语义	常用属性
form	表单	<p><code>action</code> 属性：表单要提交的地址。</p> <p><code>target</code> 属性：要跳转的新地址打开位置；值：<code>_self</code>、<code>_blank</code></p> <p><code>method</code> 属性：请求方式，值：<code>get</code>、<code>post</code></p>
input	多种形式的表单控件	<p><code>type</code> 属性：指定表单控件的类型。 值：<code>text</code>、<code>password</code>、<code>radio</code>、<code>checkbox</code>、<code>hidden</code>、<code>submit</code>、<code>reset</code> 等。</p> <p><code>name</code> 属性：指定数据名称</p> <p><code>value</code> 属性： 对于输入框：指定默认输入的值； 对于单选和复选框：实际提交的数据； 对于按钮：显示按钮文字。</p> <p><code>disabled</code> 属性：设置表单控件不可用。</p> <p><code>maxlength</code> 属性：用于输入框，设置最大可输入长度。</p> <p><code>checked</code> 属性：用于单选按钮和复选框，默认选中</p>
textarea	文本域	<p><code>name</code> 属性：指定数据名称</p> <p><code>rows</code> 属性：指定默认显示的行数，影响文本域的高度。</p> <p><code>cols</code> 属性：指定默认显示的列数，影响文本域的宽度。</p> <p><code>disabled</code> 属性：设置表单控件不可用。</p>
select	下拉框	<p><code>name</code> 属性：指定数据名称</p> <p><code>disabled</code> 属性：设置整个下拉框不可用。</p>
option	下拉框的选项	<p><code>disabled</code> 属性：设置拉下选项不可用。</p> <p><code>value</code> 属性：该选项事件提交的数据 (不指定<code>value</code>，会把标签中的内容作为提交数据)</p> <p><code>selected</code> 属性：默认选中。</p>
button	按钮	<p><code>disabled</code> 属性：设置按钮不可用。</p> <p><code>type</code> 属性：设置按钮的类型，值：<code>submit</code>（默认）、<code>reset</code>、<code>button</code></p>
label	与表单控件做关联	<p><code>for</code> 属性：值与要关联的表单控件的ID值相同。</p>
fieldset	表单边框	

13.框架标签

标签名	功能和语义	属性	单 / 双标签
iframe	框架（在网页中嵌入其他文件）	<code>name</code> : 框架名字，可以与 <code>target</code> 属性配合。 <code>width</code> : 框架的宽。 <code>height</code> : 框架的高度。 <code>frameborder</code> : 是否显示边框，值：0 或者1。	双

`iframe` 标签的实际应用：

1. 在网页中嵌入广告。
2. 与超链接或表单的 `target` 配合，展示不同的内容。

14. HTML实体

在 HTML 中我们可以用一种**特殊的形式**的内容，来表示某个**符号**，这种特殊形式的内容，就是 HTML 实体。比如小于号 `<` 用于定义 HTML 标签的开始。如果我们希望浏览器正确地显示这些字符，我们必须在 HTML 源码中插入字符实体。

字符实体由三部分组成：一个 `&` 和一个实体名称（或者一个 `#` 和一个实体编号），最后加上一个分号 `;`。

常见字符实体总结：

	描述	实体名称	实体编号
	空格	 	
<	小于号	<	<
>	大于号	>	>
&	和号	&	&
"	引号	"	"
'	反引号	´	´
¢	分 (cent)	¢	¢
£	镑 (pound)	£	£
¥	元 (yen)	¥	¥
€	欧元 (euro)	€	€
©	版权 (copyright)	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

完整实体列表, 请参考: [HTML Standard \(whatwg.org\)](https://html.spec.whatwg.org/)

15. HTML全局属性

常用的全局属性:

属性名	含义
id	给标签指定唯一标识, 注意: id 是不能重复的。 作用: 可以让 label 标签与表单控件相关联; 也可以与 CSS、JavaScript 配合使用, 。
class	给标签指定类名, 随后通过 CSS 就可以给标签设置样式。
style	给标签设置 CSS 样式。
dir	内容的方向, 值: ltr、 rtl
title	给标签设置一个文字提示, 一般超链接和图片用得比较多。
lang	给标签指定语言, 具体规范和可选值请参考【10. HTML 设置语言】。

完整的全局属性, 请参考: [全局属性 - HTML \(超文本标记语言\) | MDN \(mozilla.org\)](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes)

16.meta 元信息

1. 配置字符编码

```
<meta charset="utf-8">
```

2. 针对 IE 浏览器的兼容性配置。

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

3. 针对移动端的配置 (移动端课程中会详细讲解)

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

4. 配置网页关键字

```
<meta name="keywords" content="8-12个以英文逗号隔开的单词/词语">
```

5. 配置网页描述信息

```
<meta name="description" content="80字以内的一段话，与网站内容相关">
```

6. 针对搜索引擎爬虫配置：

```
<meta name="robots" content="此处可选值见下表">
```

值	描述
index	允许搜索爬虫索引此页面。
noindex	要求搜索爬虫不索引此页面。
follow	允许搜索爬虫跟随此页面上的链接。
nofollow	要求搜索爬虫不跟随此页面上的链接。
all	与 index, follow 等价
none	与 noindex,nofollow 等价
noarchive	要求搜索引擎不缓存页面内容。
nocache	noarchive 的替代名称。

7. 配置网页作者：

```
<meta name="author" content="tony">
```

8. 配置网页生成工具

```
<meta name="generator" content="Visual Studio Code">
```

9. 配置定义网页版权信息：

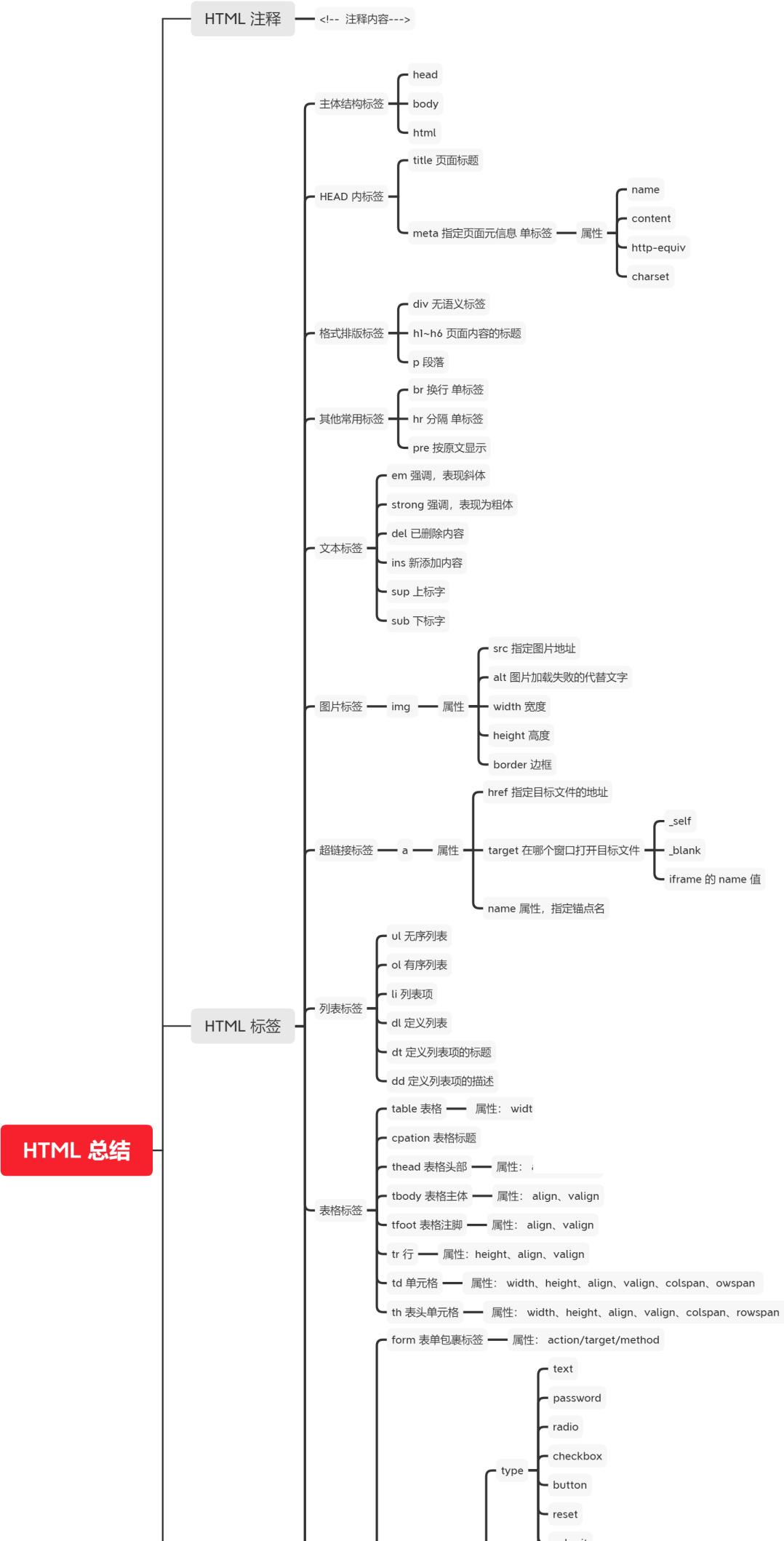
```
<meta name="copyright" content="2023-2027©版权所有">
```

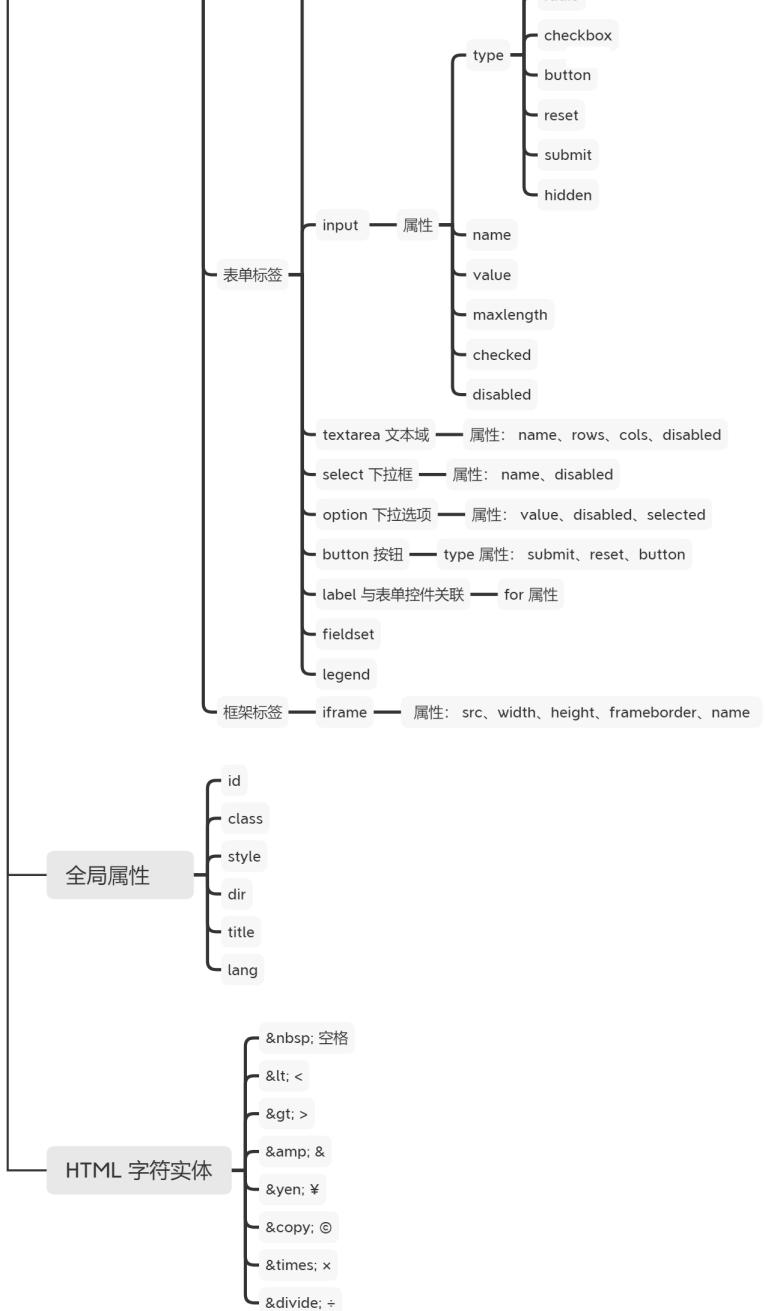
10. 配置网页自动刷新

```
<meta http-equiv="refresh" content="10;url=http://www.baidu.com">
```

完整的网页元信息，请参考：[文档级元数据元素 | MDN](#)

17. HTML总结





一、HTML5简介

1. 什么是HTML5

- HTML5 是新一代的 HTML 标准，2014年10月由万维网联盟（W3C）完成标准制定。
- 官网地址：
 - W3C 提供： <https://www.w3.org/TR/html/index.html>
 - WHATWG 提供： <https://whatwg-cn.github.io/html/multipage>
- HTML5 在狭义上是指新一代的 HTML 标准，在广义上是指：整个前端。

2. HTML5优势

1. 针对 JavaScript，新增了很多可操作的接口。
2. 新增了一些语义化标签、全局属性。
3. 新增了多媒体标签，可以很好的替代 flash。
4. 更加侧重语义化，对于 SEO 更友好。
5. 可移植性好，可以大量应用在移动设备上。

3. HTML5兼容性

- 支持： Chrome、Safari、Opera、Firefox 等主流浏览器。

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

二、新增语义化标签

1. 新增布局标签

标签名	语义	单/双标签
header	整个页面, 或部分区域的头部	双
footer	整个页面, 或部分区域的底部	双
nav	导航	双
article	文章、帖子、杂志、新闻、博客、评论等。	双
section	页面中的某段文字, 或文章中的某段文字 (里面文字通常里面会包含标题)。	双
aside	侧边栏	双
main	文档的主要内容 (WHATWG 没有语义, IE 不支持), 几乎不用。	双
hgroup	包裹连续的标题, 如文章主标题、副标题的组合 (W3C 将其删除)	双

关于 `article` 和 `section` :

- `article` 里面可以有多个 `section`。
- `section` 强调的是分段或分块, 如果你想将一块内容分成几段的时候, 可使用 `section` 元素。
- `article` 比 `section` 更强调独立性, 一块内容如果比较独立、比较完整, 应该使用 `article` 元素。

2. 新增状态标签

2.1 meter 标签

- 语义: 定义已知范围内的标量测量。也被称为 `gauge` (尺度), 双标签, 例如: 电量、磁盘用量等。
- 常用属性如下:

属性	值	描述
<code>high</code>	数值	规定高值
<code>low</code>	数值	规定低值
<code>max</code>	数值	规定最大值
<code>min</code>	数值	规定最小值
<code>optimum</code>	数值	规定最优值
<code>value</code>	数值	规定当前值

2.2 progress 标签

- 语义：显示某个任务完成的进度的指示器，一般用于表示进度条，双标签，例如：工作完成进度等。
- 常用属性如下：

属性	值	描述
max	数值	规定目标值。
value	数值	规定当前值。

3. 新增列表标签

标签名	语义	单/双标签
datalist	用于搜索框的关键字提示	双
details	用于展示问题和答案，或对专有名词进行解释	双
summary	写在 details 的里面，用于指定问题或专有名词	双

```
<input type="text" list="mydata">
<datalist id="mydata">
  <option value="周冬雨">周冬雨</option>
  <option value="周杰伦">周杰伦</option>
  <option value="温兆伦">温兆伦</option>
  <option value="马冬梅">马冬梅</option>
</datalist>
```

```
<details>
  <summary>如何走上人生巅峰？</summary>
  <p>一步一步走呗</p>
</details>
```

4. 新增文本标签

4.1 文本注音

标签名	语义	单/双标签
ruby	包裹需要注音的文字	双
rt	写注音， rt 标签写在 ruby 的里面	双

```
<ruby>
  <span>魑魅魍魉</span>
  <rt>chī mèi wǎng liǎng </rt>
</ruby>
```

4.2 文本标记

标签名	语义	单/双标签
mark	标记	双

注意： W3C 建议 mark 用于标记搜索结果中的关键字。

三、新增表单功能

1. 表单控件新增属性

属性名	功能
placeholder	提示文字（注意：不是默认值， value 是默认值），适用于 文字输入类 的表单控件。
required	表示该输入项必填，适用于 除按钮外 其他表单控件。
autofocus	自动获取焦点，适用于所有表单控件。
autocomplete	自动完成，可以设置为 on 或 off ，适用于 文字输入类 的表单控件。 注意：密码输入框、多行输入框不可用。
pattern	填写正则表达式，适用于 文本输入类 表单控件。 注意：多行输入不可用，且空的输入框不会验证，往往与 required 配合。

2. input 新增属性值

属性名	功能
email	邮箱类型的输入框，表单提交时会验证格式，输入为空则不验证格式。
url	url 类型的输入框，表单提交时会验证格式，输入为空则不验证格式。
number	数字类型的输入框，表单提交时会验证格式，输入为空则不验证格式。
search	搜索类型的输入框，表单提交时不会验证格式。
tel	电话类型的输入框，表单提交时不会验证格式，在移动端使用时，会唤起数字键盘。
range	范围选择框，默认值为 50，表单提交时不会验证格式。
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	日期+时间选择框，默认值为空，表单提交时不会验证格式。

3. form 标签新增属性

属性名	功能
novalidate	如果给 form 标签设置了该属性，表单提交的时候不再进行验证。

四、新增多媒体标签

1. 视频标签

<video> 标签用来定义视频，它是双标签。

属性	值	描述
<code>src</code>	URL地址	视频地址
<code>width</code>	像素值	设置视频播放器的宽度
<code>height</code>	像素值	设置视频播放器的高度
<code>controls</code>	-	向用户显示视频控件（比如播放/暂停按钮）
<code>muted</code>	-	视频静音
<code>autoplay</code>	-	视频自动播放
<code>loop</code>	-	循环播放
<code>poster</code>	URL地址	视频封面
<code>preload</code>	<code>auto / metadata / none</code>	<p>视频预加载，如果使用 <code>autoplay</code>，则忽略该属性。</p> <ul style="list-style-type: none"> • <code>none</code>：不预加载视频。 • <code>metadata</code>：仅预先获取视频的元数据（例如长度）。 • <code>auto</code>：可以下载整个视频文件，即使用户不希望使用它。

2. 音频标签

`<audio>` 标签用来定义音频，它是双标签。

属性	值	描述
<code>src</code>	URL地址	音频地址
<code>controls</code>	-	向用户显示音频控件（比如播放/暂停按钮）
<code>autoplay</code>	-	音频自动播放
<code>muted</code>	-	音频静音
<code>loop</code>	-	循环播放
<code>preload</code>	<code>auto / metadata / none</code>	<p>音频预加载，如果使用 <code>autoplay</code>，则忽略该属性。</p> <ul style="list-style-type: none"> • <code>none</code>：不预加载音频。 • <code>metadata</code>：仅预先获取音频的元数据（例如长度）。 • <code>auto</code>：可以下载整个音频文件，即使用户不希望使用它。

五、新增全局属性（了解）

属性名	功能
contenteditable	表示元素是否可被用户编辑，可选值如下： <code>true</code> ：可编辑 <code>false</code> ：不可编辑
draggable	表示元素可以被拖动，可选值如下： <code>true</code> ：可拖动 <code>false</code> ：不可拖动
hidden	隐藏元素
spellcheck	规定是否对元素进行拼写和语法检查，可选值如下： <code>true</code> ：检查 <code>false</code> ：不检查
contextmenu	规定元素的上下文菜单，在用户鼠标右键点击元素时显示。
data-*	用于存储页面的私有定制数据。

六、HTML5兼容性处理

- 添加元信息，让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等壳浏览器-->
<meta name="renderer" content="webkit">
```

- 使用 `html5shiv` 让低版本浏览器认识 H5 的语义化标签。

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

- 扩展

```
lt 小于
lte 小于等于
gt 大于
gte 大于等于
! 逻辑非
```

- 示例：

```
<!--[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]>IE8及以上可见<![endif]-->
<!--[if lte IE 8]>IE8及以下可见<![endif]-->
<!--[if !IE 8]>非IE8的IE可见<![endif]-->
```

一、CSS基础

1. CSS简介



- CSS 的全称为：层叠样式表 (Cascading Style Sheets)。
- CSS 也是一种标记语言，用于给 HTML 结构设置样式，例如：文字大小、颜色、元素宽高等等。

简单理解：CSS 可以美化 HTML，让 HTML 更漂亮。

核心思想：HTML 搭建结构，CSS 添加样式，实现了：结构与样式的分离。

2. CSS的编写位置

2.1 行内样式

- 写在标签的 style 属性中，(又称：内联样式)。
- 语法：

```
<h1 style="color:red;font-size:60px;">欢迎来到尚硅谷学习</h1>
```

- 注意点：

1. style 属性的值不能随便写，写要符合 CSS 语法规规范，是 名:值；的形式。
2. 行内样式表，只能控制当前标签的样式，对其他标签无效。

- 存在的问题：

书写繁琐、样式不能复用、并且没有体现出：结构与样式分离 的思想，不推荐大量使用，只有对当前元素添加简单样式时，才偶尔使用。

2.2 内部样式

- 写在 html 页面内部，将所有的 CSS 代码提取出来，单独放在 <style> 标签中。
- 语法：

```
<style>
  h1 {
    color: red;
    font-size: 40px;
  }
</style>
```

- 注意点：

1. `<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. 此种写法：样式可以复用、代码结构清晰。

- 存在的问题：

1. 并没有实现：结构与样式完全分离。
2. 多个 `HTML` 页面无法复用样式。

2.3 外部样式

- 写在单独的 `.css` 文件中，随后在 `HTML` 文件中引入使用。

- 语法：

1. 新建一个扩展名为 `.css` 的样式文件，把所有 `CSS` 代码都放入此文件中。

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

2. 在 `HTML` 文件中引入 `.css` 文件。

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

- 注意点：

1. `<link>` 标签要写在 `<head>` 标签中。
2. `<link>` 标签属性说明：
 - `href`：引入的文档来自于哪里。
 - `rel`：(`relation`：关系) 说明引入的文档与当前文档之间的关系。
3. 外部样式的优势：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。
4. 实际开发中，几乎都使用外部样式，这是**最推荐的使用方式！**

3. 样式表的优先级

- 优先级规则：行内样式 > 内部样式 = 外部样式

1. 内部样式、外部样式，这二者的优先级相同，且：后面的会覆盖前面的（简记：“后来者居上”）。

2. 同一个样式表中，优先级也和编写顺序有关，且：后面的会覆盖前面的（简记：“后来者居上”）。

分类	优点	缺点	使用频率	作用范围
行内样式	优先级最高	1. 结构与样式未分离 2. 代码结构混乱 3. 样式不能复用	很低	当前标签
内部样式	1. 样式可复用 2. 代码结构清晰	1. 结构与样式未彻底分离 2. 样式不能多页面复用	一般	当前页面
外部样式	1. 样式可多页面复用 2. 代码结构清晰 3. 可触发浏览器的缓存机制 4. 结构与样式彻底分离	需要引入才能使用	最高	多个页面

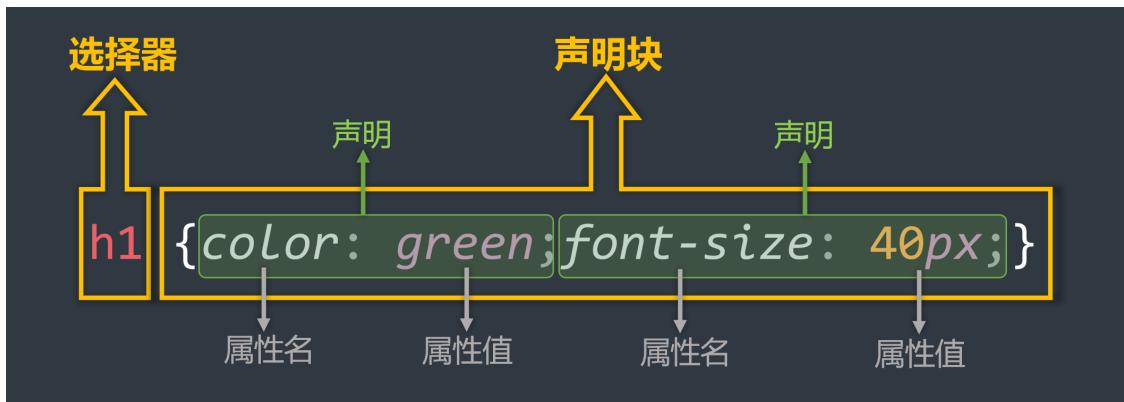
4. CSS语法规规范

CSS 语法规规范由两部分构成：

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

备注1：最后一个声明后的分号理论上能省略，但最好还是写上。

备注2：选择器与声明块之间，属性名与属性值之间，均有一个空格，理论上能省略，但最好还是写上。



- 注释的写法：

```
/* 给h1元素添加样式 */
h1 {
    /* 设置文字颜色为红色 */
    color: red;
    /* 设置文字大小为40px */
    font-size: 40px
}
```

5. CSS代码风格

- 展开风格 —— 开发时推荐，便于维护和调试。

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

- 紧凑风格 —— 项目上线时推荐，可减小文件体积。

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

- 备注：

项目上线时，我们会通过工具将【展开风格】的代码，变成【紧凑风格】，这样可以减小文件体积，节约网络流量，同时也能让用户打开网页时速度更快。

二、CSS选择器

1. CSS基本选择器

1. 通配选择器
2. 元素选择器
3. 类选择器
4. **id** 选择器

1.1 通配选择器

- 作用：可以选中所有的 **HTML** 元素。
- 语法：

```
* {  
    属性名: 属性值;  
}
```

- 举例：

```
/* 选中所有元素 */  
* {  
    color: orange;  
    font-size: 40px;  
}
```

备注：目前来看通配选择器貌似有点鸡肋，但后面清除样式时，会对我们有很大帮助，后面会详细讲。

1.2 元素选择器

- 作用：为页面中 **某种元素** 统一设置样式。
- 语法：

```
标签名 {  
    属性名: 属性值;  
}
```

- 举例：

```
/* 选中所有h1元素 */  
h1 {  
    color: orange;  
    font-size: 40px;  
}  
  
/* 选中所有p元素 */  
p {  
    color: blue;  
    font-size: 60px;  
}
```

- 备注：元素选择器无法实现**差异化设置**，例如上面的代码中，所有的 p 元素效果都一样。

1.3 类选择器

- 作用：根据元素的 class 值，来选中某些元素。

class 翻译过来有：**种类、类别的**含义，所以 class 值，又称：类名。

- 语法：

```
.类名 {  
    属性名: 属性值;  
}
```

- 举例：

```
/* 选中所有class值为speak的元素 */  
.speak {  
    color: red;  
}  
/* 选中所有class值为answer的元素 */  
.answer {  
    color: blue;  
}
```

- 注意点：

1. 元素的 class 属性值不带 .，但 CSS 的类选择器要带 .。
2. class 值，是我们自定义的，按照标准：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，例如：left-menu，且命名要有意义，做到“见名知意”。
3. 一个元素不能写多个 class 属性，下面是**错误示例**：

```
<!-- 该写法错误，元素的属性不能重复，后写的会失效 -->  
<h1 class="speak" class="big">你好啊</h1>
```

4. 一个元素的 class 属性，能写多个值，要用空格隔开，例如：

```
<!-- 该写法正确，class属性，能写多个值 -->
<h1 class="speak big">你好啊</h1>
```

1.4 ID选择器

- 作用：根据元素的 `id` 属性值，来精准的选中某个元素。
- 语法：

```
#id值 {
    属性名: 属性值;
}
```

- 举例：

```
/* 选中id值为earthy的那个元素 */
#earthy {
    color: red;
    font-size: 60px;
}
```

- 注意：
 - `id` 属性值：尽量由字母、数字、下划线(`_`)、短杠(`-`)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` 和 `class` 属性。

1.5 基本选择器总结

基本选择器	特点	用法
通配选择器	选中所有标签，一般用于清除样式。	<code>* {color:red}</code>
元素选择器	选中所有同种标签，但是不能差异化选择。	<code>h1 {color:red}</code>
类选择器	选中所有特定类名（ <code>class</code> 值）的元素——使用频率很高。	<code>.say {color:red}</code>
ID选择器	选中特定 <code>id</code> 值的那个元素（唯一的）。	<code>#earthy {color:red}</code>

2. CSS复合选择器

CSS选择器整体分类两大类：

一、基本选择器：

- ①通配选择器
- ②元素选择器
- ③类选择器
- ④ `ID` 选择器

二、复合选择器：

- ①交集选择器

②并集选择器

③后代选择器

④子元素选择器

.....

1. 复合选择器建立在基本选择器之上，由多个基础选择器，通过不同的方式组合而成。
2. 复合选择器可以在复杂结构中，快速而准确的选中元素。

2.1 交集选择器

- 作用：选中同时符合多个条件的元素。

交集有并且的含义（通俗理解：即……又……的意思），例如：年轻且长得帅。

- 语法：**选择器1选择器2选择器3...选择器n { }**
- 举例：

```
/* 选中：类名为beauty的p元素，为此种写法用的非常多！！！！ */
p.beauty {
    color: blue;
}
/* 选中：类名包含rich和beauty的元素 */
.rich.beauty {
    color: green;
}
```

- 注意：

- 有标签名，标签名必须写在前面。
- id**选择器、通配选择器，理论上可以作为交集的条件，但实际应用中几乎不用 —— 因为没有意义。
- 交集选择器中**不可能出现两个元素选择器**，因为一个元素，不可能即是 **p** 元素又是 **span** 元素。
- 用的最多的交集选择器是：元素选择器配合类名选择器，例如：**p.beauty**。

2.2 并集选择器

- 作用：选中多个选择器对应的元素，又称：**分组选择器**。

所谓并集就是或者的含义（通俗理解：要么……要么……的意思），例如：给我转10万块钱或者我报警。

- 语法：**选择器1, 选择器2, 选择器3, ... 选择器n { }**

多个选择器通过，连接，此处，的含义就是：或。

- 举例：

```
/* 选中id为peiqi, 或类名为rich, 或类名为beauty的元素 */
#peiqi,
.rich,
.beauty {
    font-size: 40px;
    background-color: skyblue;
    width: 200px;
}
```

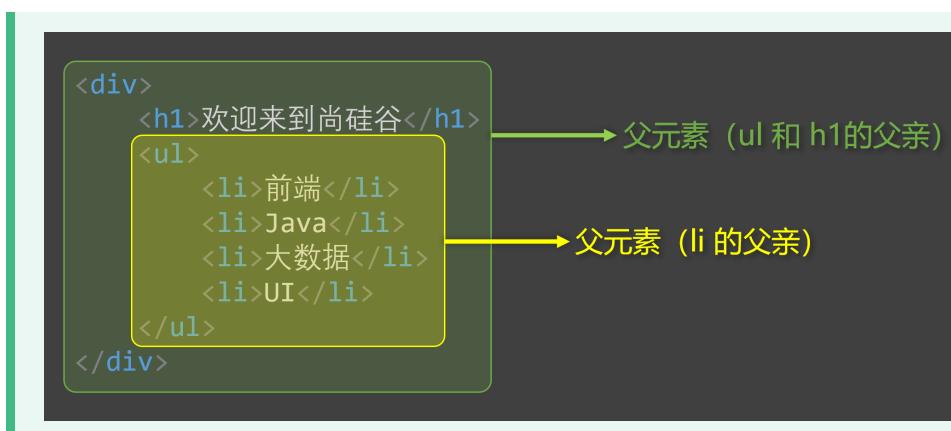
- 注意：

1. 并集选择器，我们一般竖着写。
2. 任何形式的选择器，都可以作为并集选择器的一部分。
3. 并集选择器，通常用于集体声明，可以缩小样式表体积。

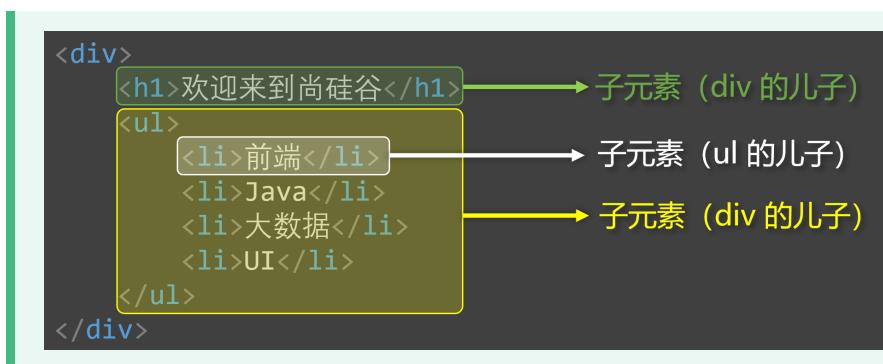
2.3 HTML元素间的关系

分为：①父元素、②子元素、③祖先元素、④后代元素、⑤兄弟元素。

1. 父元素：直接包裹某个元素的元素，就是该元素的父元素。



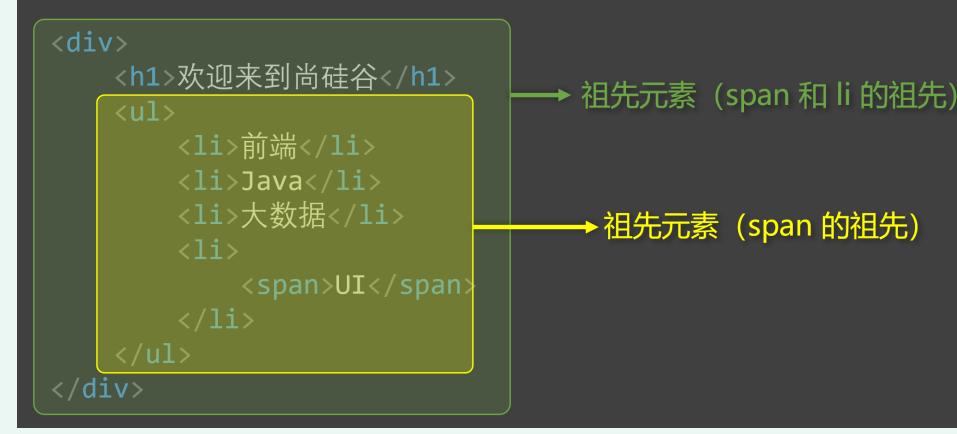
2. 子元素：被父元素直接包含的元素（简记：儿子元素）。



3. 祖先元素：父亲的父亲……，一直往外找，都是祖先。

备注：父元素，也算是祖先元素的一种。

例如：张三的父亲，也算是张三的祖先，但一般还是称呼：父亲。

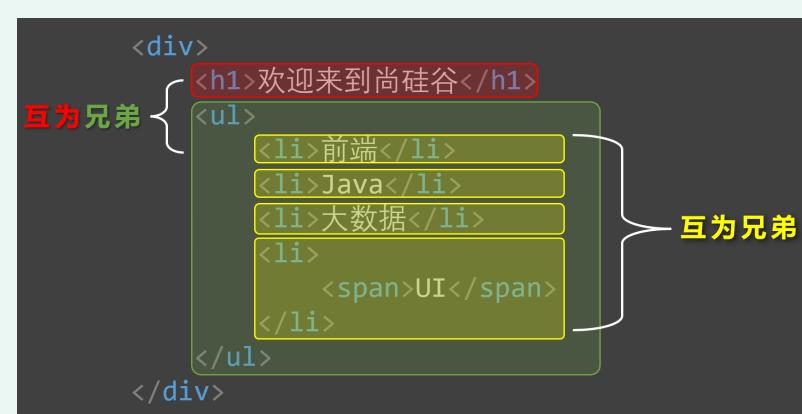


4. 后代元素：儿子的儿子……，一直往里找，都是后代。

备注：子元素，也算是后代元素的一种。
例如：张三的儿子，也算是张三的后代，但一般还是称呼：儿子。



5. 兄弟元素：具有相同父元素的元素，互为兄弟元素。



2.4 后代选择器

- 作用：选中指定元素中，符合要求的后代元素。
- 语法：选择器1 选择器2 选择器3 选择器n {} （先写祖先，再写后代）

选择器之间，用空格隔开，空格可以理解为：“xxx 中的”，其实就是后代的意思。

选择器 1234....n，可以是我们之前学的任何一种选择器。

- 举例：

```

/* 选中ul中的所有li */
ul li {
    color: red;
}

/* 选中ul中所有li中的a */
ul li a {
    color: orange;
}

/* 选中类名为subject元素中的所有li */
.subject li {
    color: blue;
}

/* 选中类名为subject元素中的所有类名为front-end的li */
.subject li.front-end {
    color: blue;
}

```

- 注意：

- 后代选择器，最终选择的是后代，不选中祖先。
- 儿子、孙子、重孙子，都算是后代。
- 结构一定要符合之前讲的 HTML 嵌套要求，例如：不能 p 中写 h1 ~ h6。

2.5 子代选择器

- 作用：选中指定元素中，符合要求的子元素（儿子元素）。（先写父，再写子）

子代选择器又称：子元素选择器、子选择器。

- 语法：选择器1 > 选择器2 > 选择器3 > 选择器n {}

选择器之间，用 > 隔开，> 可以理解为：“xxx 的子代”，其实就是儿子的意思。

选择器 1234....n，可以是我们之前学的任何一种选择器。

- 举例：

```

/* div中的子代a元素 */
div>a {
    color: red;
}

/* 类名为persons的元素中的子代a元素 */
.persons>a{
    color: red;
}

```

- 注意：

- 子代选择器，最终选择的是子代，不是父级。
- 子、孙子、重孙子、重重孙子.....统称后代！，子就是指儿子。



2.6 兄弟选择器

- 相邻兄弟选择器：
 - 作用：选中指定元素后，符合条件的**相邻兄弟**元素。

所谓相邻，就是紧挨着他的下一个，简记：睡在我下铺的兄弟。

- 语法：`选择器1+选择器2 { }` 。
- 示例：

```
/* 选中div后相邻的兄弟p元素 */
div+p {
  color:red;
}
```

- 通用兄弟选择器：
 - 作用：选中指定元素后，符合条件的**所有兄弟**元素。（简记：睡在我下铺的所有兄弟）
 - 语法：`选择器1~选择器2 { }` 。
 - 实例：

```
/* 选中div后的所有的兄弟p元素 */
div~p {
  color:red;
}
```

- 注意：两种兄弟选择器，选择的是**下面**的兄弟。

2.7 属性选择器

- 作用：选中属性值符合一定要求的元素。
- 语法：
 - `[属性名]` 选中**具有**某个属性的元素。
 - `[属性名="值"]` 选中包含某个属性，且属性值**等于**指定值的元素。
 - `[属性名^="值"]` 选中包含某个属性，且属性值以**指定的值开头**的元素。
 - `[属性名$="值"]` 选中包含某个属性，且属性值以**指定的值结尾**的元素。
 - `[属性名*=“值”]` 选择包含某个属性，属性值**包含**指定值的元素。
- 举例：

```
/* 选中具有title属性的元素 */
div[title]{color:red;}

/* 选中title属性值为atguigu的元素 */
div[title="atguigu"]{color:red;}

/* 选中title属性值以a开头的元素 */

```

```
div[title^="a"]{color:red;}  
/* 选中title属性值以u结尾的元素 */  
div[title$="u"]{color:red;}  
  
/* 选中title属性值包含g的元素 */  
div[title*= "g"]{color:red;}
```

2.8 伪类选择器

- 作用：选中特殊状态的元素。

如何理解“伪”？— 虚假的，不是真的。

如何理解“伪类”？— 像类(`class`)，但不是类，是元素的一种特殊状态。

- 常用的伪类选择器：

一、动态伪类：

- `:link` 超链接**未被访问**的状态。
- `:visited` 超链接**访问过**的状态。
- `:hover` 鼠标**悬停**在元素上的状态。
- `:active` 元素**激活**的状态。

什么是激活？——按下鼠标不松开。

注意点：遵循 **LVHA** 的顺序，即：`link`、`visited`、`hover`、`active`。

- `:focus` 获取焦点的元素。

表单类元素才能使用 `:focus` 伪类。

当用户：点击元素、触摸元素、或通过键盘的“`tab`”键等方式，选择元素时，就是获得焦点。

二、结构伪类

- 常用的：

- `:first-child` 所有兄弟元素中的**第一个**。
- `:last-child` 所有兄弟元素中的**最后一个**。
- `:nth-child(n)` 所有兄弟元素中的**第n个**。
- `:first-of-type` 所有**同类型**兄弟元素中的**第一个**。
- `:last-of-type` 所有**同类型**兄弟元素中的**最后一个**。
- `:nth-of-type(n)` 所有**同类型**兄弟元素中的**第n个**。

关于 `n` 的值：

- `0` 或 不写：什么都选不中 — 几乎不用。
- `n`：选中所有子元素 — 几乎不用。
- `1~无穷的整数`：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`：选中的是前 3 个。

- 了解即可：

1. `:nth-last-child(n)` 所有兄弟元素中的倒数第 n 个。
2. `:nth-last-of-type(n)` 所有同类型兄弟元素中的倒数第n个。
3. `:only-child` 选择没有兄弟的元素（独生子女）。
4. `:only-of-type` 选择没有同类型兄弟的元素。
5. `:root` 根元素。
6. `:empty` 内容为空元素（空格也算内容）。

三、否定伪类：

`:not(选择器)` 排除满足括号中条件的元素。

四、UI伪类：

1. `:checked` 被选中的复选框或单选按钮。
2. `:enable` 可用的表单元素（没有 `disabled` 属性）。
3. `:disabled` 不可用的表单元素（有 `disabled` 属性）。

五、目标伪类（了解）

`:target` 选中锚点指向的元素。

六、语言伪类（了解）

`:lang()` 根据指定的语言选择元素（本质是看 `lang` 属性的值）。

2.9 伪元素选择器

- 作用：选中元素中的一些特殊位置。
- 常用伪元素：

- `::first-letter` 选中元素中的第一个文字。
- `::first-line` 选中元素中的第一行文字。
- `::selection` 选中被鼠标选中的内容。
- `::placeholder` 选中输入框的提示文字。
- `::before` 在元素最开始的位置，创建一个子元素（必须用 `content` 属性指定内容）。
- `::after` 在元素最后的位置，创建一个子元素（必须用 `content` 属性指定内容）。

3. 选择器的优先级（权重）

通过不同的选择器，选中相同的元素，并且为相同的样式名设置不同的值时，就发生了样式的冲突。

到底应用哪个样式，此时就需要看优先级了。

- 简单描述：

行内样式 > ID选择器 > 类选择器 > 元素选择器 > 通配选择器。

- 详细描述：

1. 计算方式：每个选择器，都可计算出一组权重，格式为：(a, b, c)

- `a` : ID 选择器的个数。
- `b` : 类、伪类、属性 选择器的个数。
- `c` : 元素、伪元素 选择器的个数。

例如：

选择器	权重
ul>li	(0, 0, 2)
div ul>li p a span	(0, 0, 6)
#atguigu .slogan	(1, 1, 0)
#atguigu .slogan a	(1, 1, 1)
#atguigu .slogan a:hover	(1, 2, 1)

2. 比较规则：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，例如：

- (1, 0, 0) > (0, 2, 2)
- (1, 1, 0) > (1, 0, 3)
- (1, 1, 3) > (1, 1, 2)

3. 特殊规则：

1. 行内样式权重大于所有选择器。

2. !important 的权重，大于行内样式，大于所有选择器，权重最高！

4. 图示：

格式: (a , b , c)



a : ID选择器的个数



b : 类、伪类、属性 选择器的个数



c : 元素、伪元素 选择器的个数



通配选择器权重为 (0,0,0)

*	div	ul>li	div ul>li>span a
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 * 3 (1,3,0)	(1,a,b,c)	(1,?,a,b,c)

三、CSS三大特性

1. 层叠性

- 概念：如果发生了样式冲突，那就会根据一定的规则（选择器优先级），进行样式的层叠（覆盖）。

什么是样式冲突？ —— 元素的**同一个样式名**，被设置了**不同的值**，这就是冲突。

2. 继承性

- 概念：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**。
- 规则：优先继承**离得近**的。
- 常见的可继承属性：

text-?? , font-?? , line-?? 、 color

- 备注：参照MDN网站，可查询属性是否可被继承。

3. 优先级

- 简单聊：`!important` > 行内样式 > ID选择器 > 类选择器 > 元素选择器 > * > 继承的样式。
- 详细聊：需要计算权重。

计算权重时需要注意：**并集选择器的每一个部分是分开算的！**

四、CSS常用属性

1. 像素的概念

- 概念：我们的电脑屏幕是，是由一个一个“小点”组成的，每个“小点”，就是一个像素（px）。
- 规律：像素点越小，呈现的内容就越清晰、越细腻。



注意点：如果电脑设置中开启了缩放，那么就会影响一些工具的测量结果，但这无所谓，因为我们工作中都是参考详细的设计稿，去给元素设置宽高。

2. 颜色的表示

2.1 表示方式一：颜色名

- 编写方式：直接使用颜色对应的英文单词，编写比较简单，例如：
 - 红色：red
 - 绿色：green
 - 蓝色：blue
 - 紫色：purple
 - 橙色：orange
 - 灰色：gray

.....

- 颜色名这种方式，表达的颜色比较单一，所以用的并不多。
- 具体颜色名参考MDN官方文档：

2.2 表示方式二：rgb 或 rgba

- 编写方式：使用 红、黄、蓝 这三种光的三原色进行组合。
 - r 表示 红色
 - g 表示 绿色
 - b 表示 蓝色
 - a 表示 透明度
- 举例：

```
/* 使用 0~255 之间的数字表示一种颜色 */
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 */

/* 混合出任意一种颜色 */
color:rgb(138, 43, 226) /* 紫罗兰色 */
color:rgba(255, 0, 0, 0.5); /* 半透明的红色 */

/* 也可以使用百分比表示一种颜色（用的少） */
color: rgb(100%, 0%, 0%); /* 红色 */
color: rgba(100%, 0%, 0%, 50%); /* 半透明的红色 */
```

- 小规律：

1. 若三种颜色值相同，呈现的是灰色，值越大，灰色越浅。
2. `rgb(0, 0, 0)` 是黑色，`rgb(255, 255, 255)` 是白色。
3. 对于 `rgba` 来说，前三位的 `rgb` 形式要保持一致，要么都是 0~255 的数字，要么都是百分比。

2.3 表示方式三：HEX 或 HEXA

HEX 的原理同与 `rgb` 一样，依然是通过：红、绿、蓝色 进行组合，只不过要用 6位（分成3组）来表达，

格式为：#rrggbb

每一位数字的取值范围是：0 ~ f，即：(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f)

所以每一种光的最小值是：00，最大值是：ff

```

color: #ff0000; /* 红色 */
color: #00ff00; /* 绿色 */
color: #0000ff; /* 蓝色 */
color: #000000; /* 黑色 */
color: #ffffff; /* 白色 */

/* 如果每种颜色的两位都是相同的，就可以简写 */
color: #ff9988; /* 可简为: #f98 */

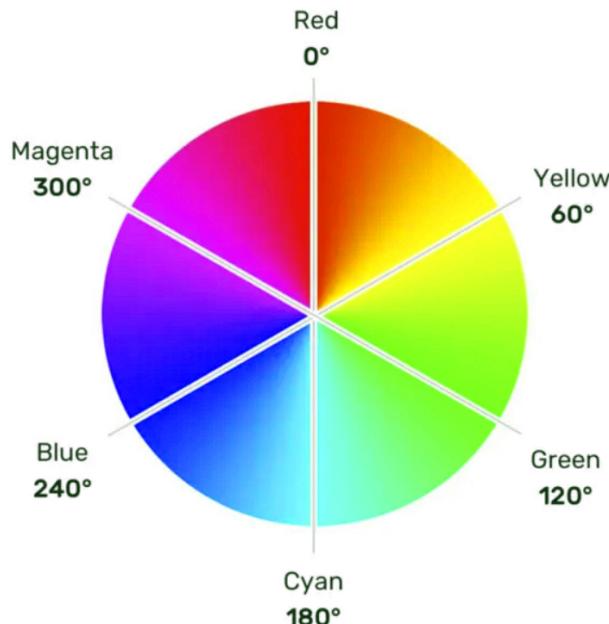
/* 但要注意前三位简写了，那么透明度也要简写 */
color: #ff998866; /* 可简为: #f986 */

```

注意点：IE 浏览器不支持 HEXA，但支持 HEX。

2.4 表示方式四：HSL 或 HSLA

- HSL 是通过：色相、饱和度、亮度，来表示一个颜色的，格式为：`hsl(色相, 饱和度, 亮度)`
 - 色相：取值范围是 0~360 度，具体度数对应的颜色如下图：



- 饱和度：取值范围是 0%~100%。 (向色相中对应颜色中添加灰色， 0% 全灰， 100% 没有灰)
- 亮度：取值范围是 0%~100%。 (0% 亮度没了，所以就是黑色。 100% 亮度太强，所以就是白色了)
- HSLA 其实就是在 HSL 的基础上，添加了透明度。

3. CSS字体属性

3.1 字体大小

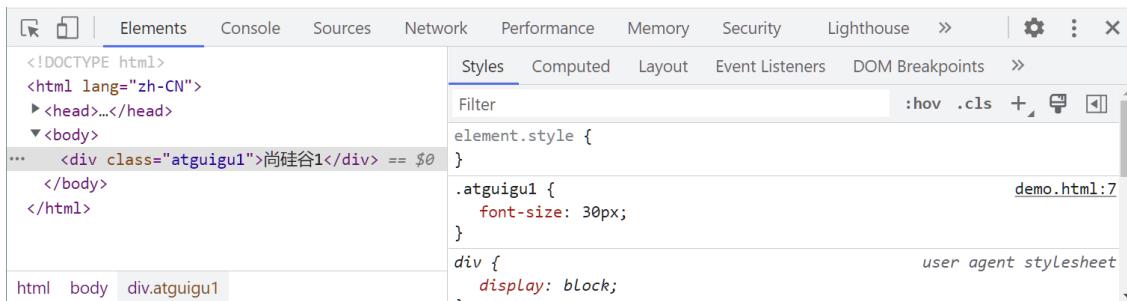
- 属性名：`font-size`
- 作用：控制字体的大小。
- 语法：

```
div {  
    font-size: 40px;  
}
```

- 注意点：

1. Chrome 浏览器支持的最小文字为 12px，默认的文字大小为 16px，并且 0px 会自动消失。
2. 不同浏览器默认的字体大小可能不一致，所以最好给一个明确的值，不要用默认大小。
3. 通常以给 body 设置 font-size 属性，这样 body 中的其他元素就都可以继承了。

- 借助控制台看样式：



3.2 字体族

- 属性名：font-family
- 作用：控制字体类型。
- 语法：

```
div {  
    font-family: "STCaiyun", "Microsoft YaHei", sans-serif  
}
```

- 注意：

1. 使用字体的英文名字兼容性会更好，具体的英文名可以自行查询，或在电脑的设置里去寻找。
2. 如果字体名包含空格，必须使用引号包裹起来。
3. 可以设置多个字体，按照从左到右的顺序逐个查找，找到就用，没有找到就使用后面的，且通常在最后写上 serif（衬线字体）或 sans-serif（非衬线字体）。
4. windows 系统中，默认的字体就是微软雅黑。

3.3 字体风格

- 属性名：font-style
- 作用：控制字体是否为斜体。
- 常用值：
 1. normal：正常（默认值）
 2. italic：斜体（使用字体自带的斜体效果）
 3. oblique：斜体（强制倾斜产生的斜体效果）

实现斜体时，更推荐使用 italic。

- 语法:

```
div {  
    font-style: italic;  
}
```

3.4 字体粗细

- 属性名: `font-weight`

- 作用: 控制字体的粗细。

- 常用值:

- 关键词

1. `lighter` : 细
2. `normal` : 正常
3. `bold` : 粗
4. `bolder` : 很粗 (多数字体不支持)

- 数值:

1. `100~1000` 且无单位, 数值越大, 字体越粗 (或一样粗, 具体得看字体设计时的精确程度)。
2. `100~300` 等同于 `lighter` , `400~500` 等同于 `normal` , `600` 及以上等同于 `bold` 。

- 语法:

```
div {  
    font-weight: bold;  
}  
  
div {  
    font-weight: 600;  
}
```

3.5 字体复合写法

- 属性名: `font` , 可以把上述字体样式合并成一个属性。

- 作用: 将上述所有字体相关的属性复合在一起编写。

- 编写规则:

1. 字体大小、字体族必须都写上。
 2. 字体族必须是最后一位、字体大小必须是倒数第二位。
 3. 各个属性间用空格隔开。
- 实际开发中更推荐复合写法, 但这也不是绝对的, 比如只想设置字体大小, 那就直接用 `font-size` 属性。

4. CSS文本属性

4.1 文本颜色

- 属性名: `color`
- 作用: 控制文字的颜色。
- 可选值:
 1. 颜色名
 2. `rgb` 或 `rgba`
 3. `HEX` 或 `HEXA` (十六进制)
 4. `HSL` 或 `HSLA`

开发中常用的是: `rgb/rgba` 或 `HEX/HEXA` (十六进制)。

- 举例:

```
div {  
    color: rgb(112, 45, 78);  
}
```

4.2 文本间距

- 字母间距: `letter-spacing`
- 单词间距: `word-spacing` (通过空格识别词)
- 属性值为像素 (`px`) , 正值让间距增大, 负值让间距缩小。

4.3 文本修饰

- 属性名: `text-decoration`
- 作用: 控制文本的各种装饰线。
- 可选值:
 1. `none` : 无装饰线 (常用)
 2. `underline` : 下划线 (常用)
 3. `overline` : 上划线
 4. `line-through` : 删除线

可搭配如下值使用:

1. `dotted` : 虚线
2. `wavy` : 波浪线
3. 也可以指定颜色

- 举例:

```
a {  
    text-decoration: none;  
}
```

4.4 文本缩进

- 属性名: `text-indent`。
- 作用: 控制文本首字母的缩进。
- 属性值: `css` 中的长度单位, 例如: `px`
- 举例:

```
div {  
    text-indent: 40px;  
}
```

后面我们会学习 `css` 中一些新的长度单位, 目前我们只知道像素(`px`)。

4.5 文本对齐_水平

- 属性名: `text-align`。
- 作用: 控制文本的水平对齐方式。
- 常用值:
 1. `left` : 左对齐 (默认值)
 2. `right` : 右对齐
 3. `center` : 居中对齐
- 举例:

```
div {  
    text-align: center;  
}
```

4.6 细说 `font-size`

1. 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。

例如: `font-size` 设为 `40px`, 最终呈现的文字, 可能比 `40px` 大, 也可能比 `40px` 小。

2. 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。

4.7 行高

- 属性名: `line-height`
- 作用: 控制一行文字的高度。
- 可选值:
 1. `normal` : 由浏览器根据文字大小决定的一个默认值。
 2. 像素(`px`)。
 3. 数字: 参考自身 `font-size` 的倍数 (很常用)。
 4. 百分比: 参考自身 `font-size` 的百分比。
- 备注: 由于字体设计原因, 文字在一行中, 并不是绝对垂直居中, 若一行中都是文字, 不会太影响观感。

- 举例：

```
div {  
    line-height: 60px;  
    line-height: 1.5;  
    line-height: 150%;  
}
```

- 行高注意事项：

1. `line-height` 过小会怎样？—— 文字产生重叠，且最小值是 `0`，不能为负数。
2. `line-height` 是可以继承的，且为了能更好的呈现文字，最好写数值。
3. `line-height` 和 `height` 是什么关系？
 - 设置了 `height`，那么高度就是 `height` 的值。
 - 不设置 `height` 的时候，会根据 `line-height` 计算高度。

- 应用场景：

1. 对于多行文字：控制行与行之间的距离。
2. 对于单行文字：让 `height` 等于 `line-height`，可以实现文字垂直居中。

备注：由于字体设计原因，靠上述办法实现的居中，并不是绝对的垂直居中，但如果一行中都是文字，不会太影响观感。

4.8 文本对齐_垂直

1. **顶部**：无需任何属性，在垂直方向上，默认就是顶部对齐。
2. **居中**：对于单行文字，让 `height = line-height` 即可。

问题：多行文字**垂直居中**怎么办？—— 后面我们用定位去做。

3. **底部**：对于单行文字，目前一个临时的方式：

让 `line-height = (height × 2) - font-size - x`。

备注：`x` 是根据字体族，动态决定的一个值。

问题：垂直方向上的底部对齐，更好的解决办法是什么？—— 后面我们用定位去做。

4.9 vertical-align

- 属性名：`vertical-align`。
- 作用：用于指定**同一行元素之间**，或**表格单元格**内文字的**垂直对齐方式**。
- 常用值：
 1. `baseline`（默认值）：使元素的基线与父元素的基线对齐。
 2. `top`：使元素的**顶部与其所在行的顶部**对齐。
 3. `middle`：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。
 4. `bottom`：使元素的**底部与其所在行的底部**对齐。

特别注意：`vertical-align` 不能控制块元素。

5. CSS列表属性

列表相关的属性，可以作用在 `ul`、`ol`、`li` 元素上。

CSS 属性名	功能	属性值
<code>list-style-type</code>	设置列表符号	常用值如下： <code>none</code> ：不显示前面的标识（很常用！） <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> 的里面 <code>outside</code> ：在 <code>li</code> 的外边
<code>list-style-image</code>	自定义列表符号	<code>url(图片地址)</code>
<code>list-style</code>	复合属性	没有数量、顺序的要求

6. CSS表格属性

1. 边框相关属性（其他元素也能用）：

CSS 属性名	功能	属性值
<code>border-width</code>	边框宽度	<code>CSS</code> 中可用的长度值
<code>border-color</code>	边框颜色	<code>CSS</code> 中可用的颜色值
<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	没有数量、顺序的要求

注意：

- 以上 4 个边框相关的属性，其他元素也可以用，这是我们第一次遇见它们。
- 在后面的盒子模型中，我们会详细讲解边框相关的知识。

2. 表格独有属性（只有 `table` 标签才能使用）：

css 属性名	功能	属性值
table-layout	设置列宽度	<code>auto</code> : 自动, 列宽根据内容计算 (默认值)。 <code>fixed</code> : 固定列宽, 平均分。
border-spacing	单元格间距	CSS 中可用的长度值。 生效的前提: 单元格边框不能合并。
border-collapse	合并单元格边框	<code>collapse</code> : 合并 <code>separate</code> : 不合并
empty-cells	隐藏没有内容的单元格	<code>show</code> : 显示, 默认 <code>hide</code> : 隐藏 生效前提: 单元格不能合并。
caption-side	设置表格标题位置	<code>top</code> : 上面 (默认值) <code>bottom</code> : 在表格下面

以上 5 个属性, 只有表格才能使用, 即: `<table>` 标签。

7. CSS背景属性

css 属性名	功能	属性值
background-color	设置背景颜色	符合 CSS 中颜色规范的值。 默认背景颜色是 <code>transparent</code> 。
background-image	设置背景图片	<code>url(图片的地址)</code>
background-repeat	设置背景重复方式	<code>repeat</code> : 重复, 铺满整个元素, 默认值。 <code>repeat-x</code> : 只在水平方向重复。 <code>repeat-y</code> : 只在垂直方向重复。 <code>no-repeat</code> : 不重复。
background-position	设置背景图位置	通过关键字设置位置: 写两个值, 用空格隔开 水平: <code>left</code> 、 <code>center</code> 、 <code>right</code> 垂直: <code>top</code> 、 <code>center</code> 、 <code>bottom</code> 如果只写一个值, 另一个方向的值取 <code>center</code> 通过长度指定坐标位置: 以元素左上角, 为坐标原点, 设置图片左上角的位置。 两个值, 分别是 <code>x</code> 坐标和 <code>y</code> 坐标。 只写一个值, 会被当做 <code>x</code> 坐标, <code>y</code> 坐标取 <code>center</code>
background	复合属性	没有数量和顺序要求

8. CSS鼠标属性

CSS 属性名	功能	属性值
cursor	设置鼠标光标的样式	pointer : 小手 move : 移动图标 text : 文字选择器 crosshair : 十字架 wait : 等待 help : 帮助

扩展：自定义鼠标图标

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

五、CSS盒子模型

1. CSS 长度单位

- px : 像素。
- em : 相对元素 font-size 的倍数。
- rem : 相对根字体大小，html标签就是根。
- % : 相对父元素计算。

注意：CSS 中设置长度，必须加单位，否则样式无效！

2. 元素的显示模式

• 块元素 (block)

又称：块级元素

特点：

- 在页面中**独占一行**，不会与任何元素共用一行，是从上到下排列的。
- 默认宽度：撑满**父元素**。
- 默认高度：由**内容**撑开。
- 可以通过** CSS 设置宽高。

• 行内元素 (inline)

又称：内联元素

特点：

- 在页面中**不独占一行**，一行中不能容纳下的行内元素，会在下一行继续从左到右排列。
- 默认宽度：由**内容**撑开。

- 3. 默认高度：由**内容**撑开。
- 4. **无法**通过 CSS 设置宽高。

• 行内块元素 (inline-block)

又称：内联块元素

特点：

- 1. 在页面中**不独占一行**，一行中不能容纳下的行内元素，会在下一行继续从左到右排列。
- 2. 默认宽度：由**内容**撑开。
- 3. 默认高度：由**内容**撑开。
- 4. **可以**通过 CSS 设置宽高。

注意：元素早期只分为：**行内元素、块级元素**，区分条件也只有一条：“是否独占一行”，如果按照这种分类方式，行内块元素应该算作行内元素。

3. 总结各元素的显示模式

• 块元素 (block)

1. 主体结构标签：`<html>`、`<body>`
2. 排版标签：`<h1>` ~ `<h6>`、`<hr>`、`<p>`、`<pre>`、`<div>`
3. 列表标签：``、``、``、`<dl>`、`<dt>`、`<dd>`
4. 表格相关标签：`<table>`、`<tbody>`、`<thead>`、`<tfoot>`、`<tr>`、`<caption>`
5. `<form>` 与 `<option>`

• 行内元素 (inline)

1. 文本标签：`
`、``、``、`<sup>`、`<sub>`、``、`<ins>`
2. `<a>` 与 `<label>`

• 行内块元素 (inline-block)

1. 图片：``
2. 单元格：`<td>`、`<th>`
3. 表单控件：`<input>`、`<textarea>`、`<select>`、`<button>`
4. 框架标签：`<iframe>`

4. 修改元素的显示模式

通过 CSS 中的 `display` 属性可以修改元素的默认显示模式，常用值如下：

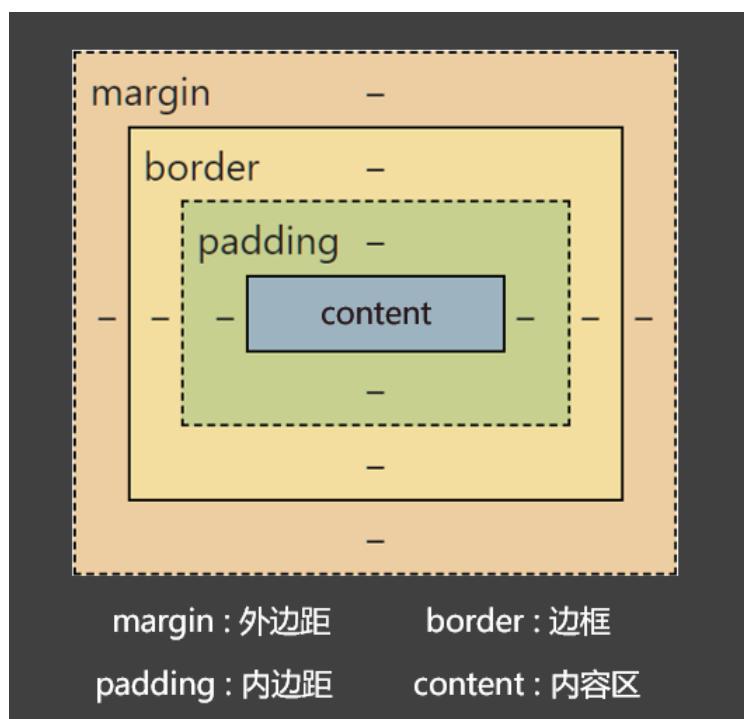
值	描述
none	元素会被隐藏。
block	元素将作为 块级元素 显示。
inline	元素将作为 内联元素 显示。
inline-block	元素将作为 行内块元素 显示。

5. 盒子模型的组成

CSS 会把所有的 **HTML** 元素都看成一个**盒子**，所有的样式也都是基于这个盒子。

1. **margin (外边距)**： 盒子与外界的距离。
2. **border (边框)**： 盒子的边框。
3. **padding (内边距)**： 紧贴内容的补白区域。
4. **content (内容)**： 元素中的文本或后代元素都是它的内容。

图示如下：



盒子的大小 = **content** + **左右 padding** + **左右 border**。

注意：外边距 **margin** 不会影响盒子的大小，但会影响盒子的位置。

6. 盒子内容区 (content)

CSS 属性名	功能	属性值
<code>width</code>	设置内容区域宽度	长度
<code>max-width</code>	设置内容区域的最大宽度	长度
<code>min-width</code>	设置内容区域的最小宽度	长度
<code>height</code>	设置内容区域的高度	长度
<code>max-height</code>	设置内容区域的最大高度	长度
<code>min-height</code>	设置内容区域的最小高度	长度

注意：

`max-width`、`min-width` 一般不与 `width` 一起使用。

`max-height`、`min-height` 一般不与 `height` 一起使用。

7. 关于默认宽度

所谓的默认宽度，就是**不设置 width 属性时**，元素所呈现出来的宽度。

总宽度 = 父的 `content` – 自身的左右 `margin`。

内容区的宽度 = 父的 `content` – 自身的左右 `margin` – 自身的左右 `border` – 自身的左右 `padding`。

8. 盒子内边距 (padding)

CSS 属性名	功能	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	长度，可以设置 1~4 个值

`padding` 复合属性的使用规则：

1. `padding: 10px;` 四个方向内边距都是 `10px`。
2. `padding: 10px 20px;` 上 `10px`，左右 `20px`。（上下、左右）
3. `padding: 10px 20px 30px;` 上 `10px`，左右 `20px`，下 `30px`。（上、左右、下）
4. `padding: 10px 20px 30px 40px;` 上 `10px`，右 `20px`，下 `30px`，左 `40px`。（上、右、下、左）

注意点：

1. `padding` 的值不能为负数。
2. 行内元素 的左右内边距是没问题的，上下内边距不能完美的设置。
3. 块级元素、行内块元素，四个方向内边距都可以完美设置。

9. 盒子边框 (border)

css 属性名	功能	属性值
<code>border-style</code>	边框线风格 复合了四个方向的边框风格	<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-width</code>	边框线宽度 复合了四个方向的边框宽度	长度, 默认 <code>3px</code>
<code>border-color</code>	边框线颜色 复合了四个方向的边框颜色	颜色, 默认黑色
<code>border</code>	复合属性	值没有顺序和数量要求。
<code>border-left</code> <code>border-left-style</code> <code>border-left-width</code> <code>border-left-color</code>	分别设置各个方向的边框	同上
<code>border-right</code> <code>border-right-style</code> <code>border-right-width</code> <code>border-right-color</code>		
<code>border-top</code> <code>border-top-style</code> <code>border-top-width</code> <code>border-top-color</code>		
<code>border-bottom</code> <code>border-bottom-style</code> <code>border-bottom-width</code> <code>border-bottom-color</code>		

边框相关属性共 20 个。

`border-style`、`border-width`、`border-color` 其实也是复合属性。

10. 盒子外边距_margin

CSS 属性名	功能	属性值
<code>margin-left</code>	左外边距	CSS 中的长度值
<code>margin-right</code>	右外边距	CSS 中的长度值
<code>margin-top</code>	上外边距	CSS 中的长度值
<code>margin-bottom</code>	下外边距	CSS 中的长度值
<code>margin</code>	复合属性，可以写 1~4 个值，规律同 <code>padding</code> (顺时针)	CSS 中的长度值

10.1 margin 注意事项

- 子元素的 `margin`，是参考父元素的 `content` 计算的。（因为是父亲的 `content` 中承装着子元素）
- 上 `margin`、左 `margin`：影响自己的位置；下 `margin`、右 `margin`：影响后面兄弟元素的位置。
- 块级元素、行内块元素，均可以完美地设置四个方向的 `margin`；但行内元素，左右 `margin` 可以完美设置，上下 `margin` 设置无效。
- `margin` 的值也可以是 `auto`，如果给一个块级元素设置左右 `margin` 都为 `auto`，该块级元素会在父元素中水平居中。
- `margin` 的值可以是负值。

10.2 margin 塌陷问题

什么是 `margin` 塌陷？

第一个子元素的上 `margin` 会作用在父元素上，最后一个子元素的下 `margin` 会作用在父元素上。

如何解决 `margin` 塌陷？

- 方案一：给父元素设置不为 0 的 `padding`。
- 方案二：给父元素设置宽度不为 0 的 `border`。
- 方案三：给父元素设置 css 样式 `overflow:hidden`

10.3 margin 合并问题

什么是 `margin` 合并？

上面兄弟元素的下外边距和下面兄弟元素的上外边距会合并，取一个最大的值，而不是相加。

如何解决 `margin` 塌陷？

无需解决，布局的时候上下的兄弟元素，只给一个设置上下外边距就可以了。

11. 处理内容溢出

CSS 属性名	功能	属性值
overflow	溢出内容的处理方式	visible : 显示, 默认值 hidden : 隐藏 scroll : 显示滚动条, 不论内容是否溢出 auto : 自动显示滚动条, 内容不溢出不显示
overflow-x	水平方向溢出内容的处理方式	同 overflow
overflow-y	垂直方向溢出内容的处理方式	同 overflow

注意：

1. `overflow-x`、`overflow-y` 不能一个是 `hidden`，一个是 `visible`，是实验性属性，不建议使用。
2. `overflow` 常用的值是 `hidden` 和 `auto`，除了能处理溢出的显示方式，还可以解决很多疑难杂症。

12. 隐藏元素的方式

方式一：visibility 属性

`visibility` 属性默认值是 `show`，如果设置为 `hidden`，元素会隐藏。

元素看不见了，还占有原来的位置（元素的大小依然保持）。

方式二：display 属性

设置 `display:none`，就可以让元素隐藏。

彻底地隐藏，不但看不见，也不占用任何位置，没有大小宽高。

13. 样式的继承

有些样式会继承，元素如果本身设置了某个样式，就使用本身设置的样式；但如果本身没有设置某个样式，会从父元素开始一级一级继承（优先继承离得近的祖先元素）。

会继承的 css 属性

字体属性、文本属性（除了 `vertical-align`）、文字颜色 等。

不会继承的 css 属性

边框、背景、内边距、外边距、宽高、溢出方式 等。

一个规律：能继承的属性，都是不影响布局的，简单说：都是和盒子模型没关系的。

14. 默认样式

元素一般都些默认的样式，例如：

1. `<a>` 元素：下划线、字体颜色、鼠标小手。
 2. `<h1> ~ <h6>` 元素：文字加粗、文字大小、上下外边距。
 3. `<p>` 元素：上下外边距
 4. ``、`ol` 元素：左内边距
 5. `body` 元素： `8px` 外边距（4个方向）
-

优先级：**元素的默认样式 > 继承的样式**，所以如果要重置元素的默认样式，选择器一定要直接选择器到该元素。

15. 布局小技巧

1. 行内元素、行内块元素，可以被父元素当做文本处理。

即：可以像处理文本对齐一样，去处理：行内、行内块在父元素中的对齐。

例如：`text-align`、`line-height`、`text-indent` 等。

2. 如何让子元素，在父亲中 **水平居中**：

- 若子元素为**块元素**，给父元素加上：`margin:0 auto;`。
- 若子元素为**行内元素、行内块元素**，给父元素加上：`text-align:center`。

3. 如何让子元素，在父亲中 **垂直居中**：

- 若子元素为**块元素**，给子元素加上：`margin-top`，值为：(父元素 `content` - 子元素盒子总高) / 2。
- 若子元素为**行内元素、行内块元素**：

让父元素的 `height = line-height`，每个子元素都加上：`vertical-align:middle;`。

补充：若想绝对垂直居中，父元素 `font-size` 设置为 0。

16. 元素之间的空白问题

产生的原因：

行内元素、行内块元素，彼此之间的换行会被浏览器解析为一个空白字符。

解决方案：

1. **方案一：** 去掉换行和空格（不推荐）。
2. **方案二：** 给父元素设置 `font-size:0`，再给需要显示文字的元素，单独设置字体大小（推荐）。

17. 行内块的幽灵空白问题

产生原因：

行内块元素与文本的基线对齐，而文本的基线与文本最底端之间是有一定距离的。

解决方案：

2. **方案一：**给行内块设置 `vertical`，值不为 `baseline` 即可，设置为 `middle`、`bottom`、`top` 均可。
3. **方案二：**若父元素中只有一张图片，设置图片为 `display:block`。
4. **方案三：**给父元素设置 `font-size: 0`。如果该行内块内部还有文本，则需单独设置 `font-size`。

六、浮动

1. 浮动的简介

在最初，浮动是用来实现文字环绕图片效果的，现在浮动是主流的页面布局方式之一。



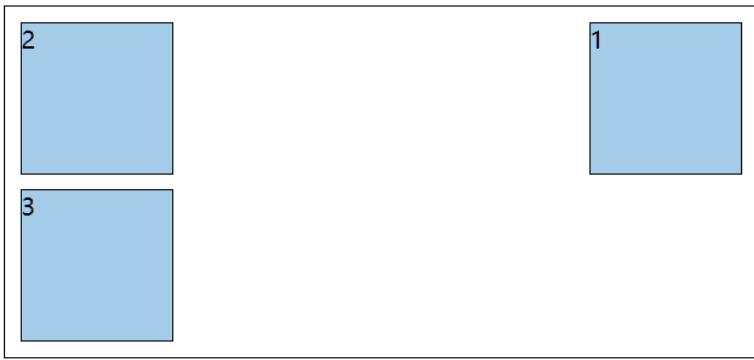
Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga, esse eveniet. Soluta quos beatae aperiam velit voluptatibus at veniam minima mollitia nostrum maxime eveniet eos, totam facilis animi laudantium eius placeat earum aspernatur rerum voluptatum, sunt repellat? Quidem recusandae iusto velit incident veritatis quasi, labore voluptatem odit debitis eum ullam vero illum tempore dignissimos voluptatibus eius cumque, optio nulla assumenda! Magnam quisquam laborum doloribus eveniet minima eum sit! Repellat, quibusdam. Illum, consequuntur blanditiis fugiat quasi reiciendis unde voluptate rem sequi explicabo, temporibus facilis quod tenetur voluptates aliquid aperiam, ducimus sint. Similique mollitia illum aliquid eos vero suscipit necessitatibus dolore maxime?

2. 元素浮动后的特点

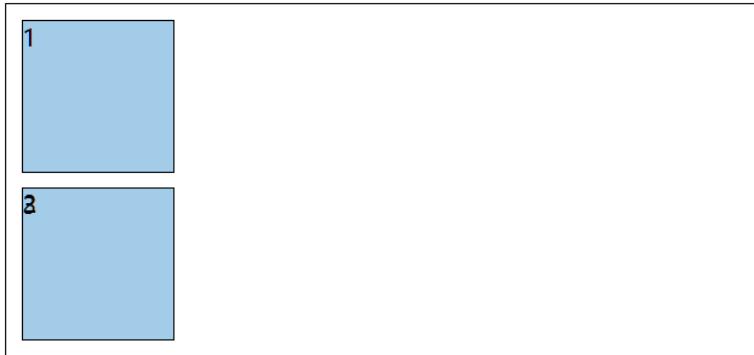
1. **脱离文档流。**
2. **不管浮动前是什么元素，浮动后：默认宽与高都是被内容撑开（尽可能小），而且可以设置宽高。**
3. **不会独占一行，可以与其他元素共用一行。**
4. **不会 `margin` 合并，也不会 `margin` 塌陷，能够完美的设置四个方向的 `margin` 和 `padding`。**
5. **不会像行内块一样被当做文本处理（没有行内块的空白问题）。**

3. 浮动小练习

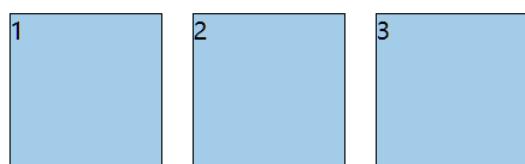
练习1：盒子1右浮动，效果如下



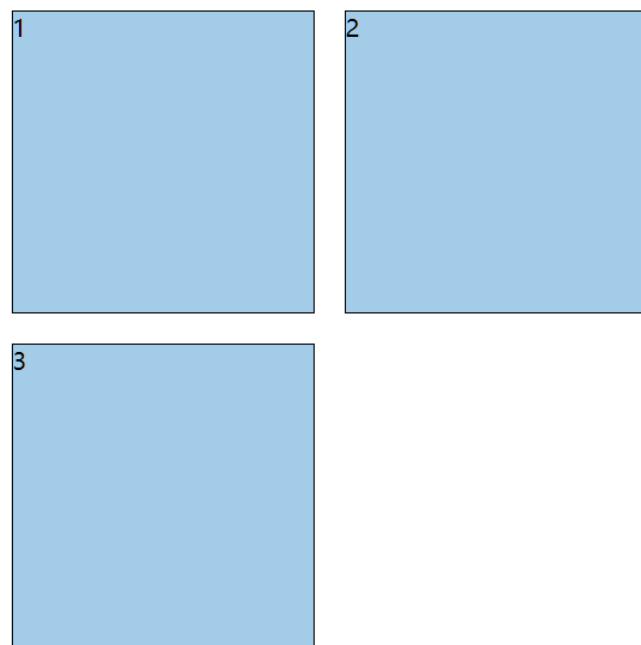
练习2：盒子1左浮动，效果如下



练习3：所有盒子都浮动，效果如下



练习4：所有盒子浮动后，盒子3落下来，效果如下



练习5：所有盒子浮动后，盒子3卡住了，效果如下

4. 解决浮动产生的影响

4.1 元素浮动后会有哪些影响

对兄弟元素的影响：后面的兄弟元素，会占据浮动元素之前的位置，在浮动元素的下面；对前面的兄弟无影响。

对父元素的影响：不能撑起父元素的高度，导致父元素高度塌陷；但父元素的宽度依然束缚浮动的元素。

4.2 解决浮动产生的影响（清除浮动）

解决方案：

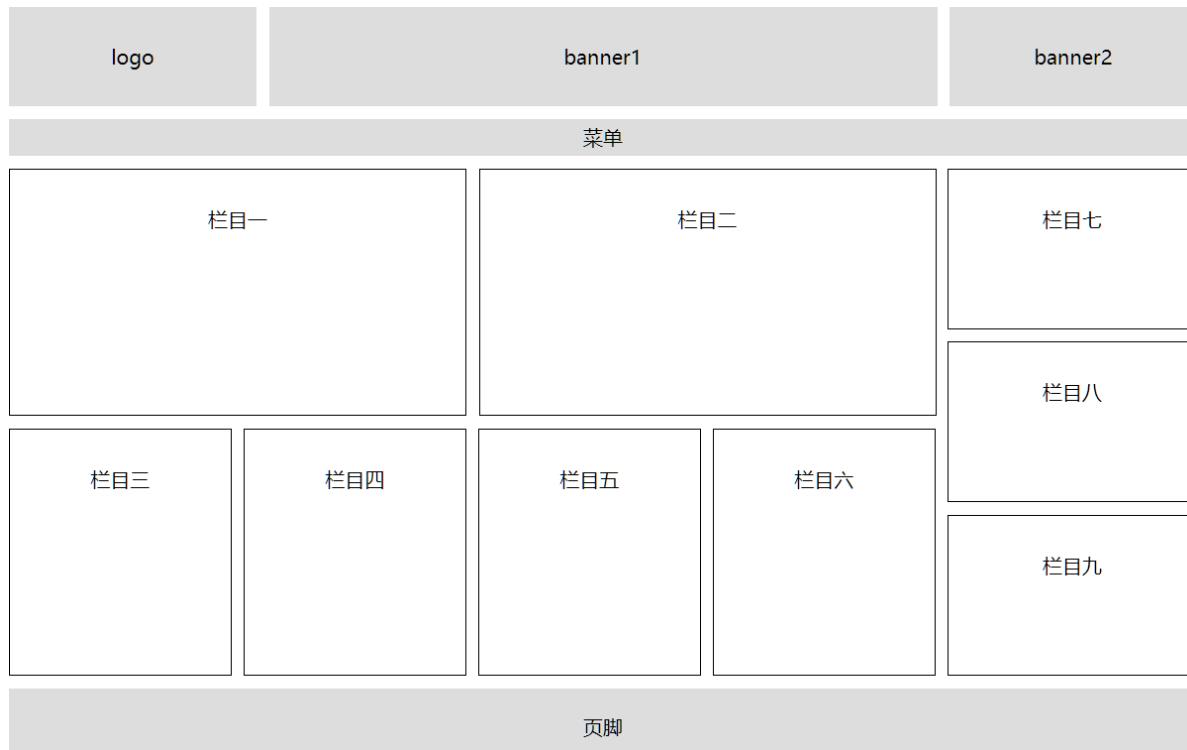
1. 方案一：给父元素指定高度。
2. 方案二：给父元素也设置浮动，带来其他影响。
3. 方案三：给父元素设置 `overflow:hidden`。
4. 方案四：在所有浮动元素的最后面，添加一个块级元素，并给该块级元素设置 `clear:both`。
5. **方案五：**给浮动元素的父元素，设置伪元素，通过伪元素清除浮动，原理与方案四相同。====> 推荐使用

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

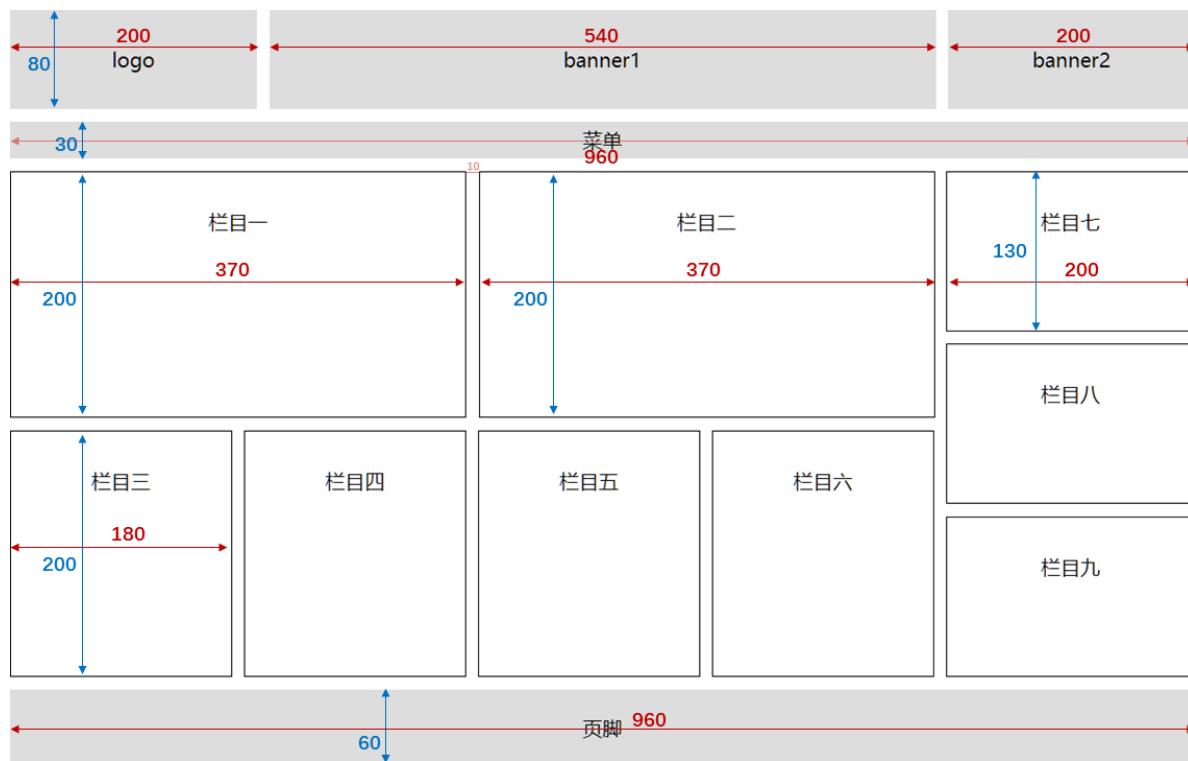
布局中的一个原则：设置浮动的时候，兄弟元素要么全都浮动，要么全都不浮动。

5. 浮动布局小练习

整体效果如下：



具体标注如下：



6. 浮动相关属性

CSS 属性	功能	属性值
<code>float</code>	设置浮动	<code>left</code> : 设置左浮动 <code>right</code> : 设置右浮动 <code>none</code> : 不浮动, 默认值
<code>clear</code>	清除浮动 清除前面兄弟元素浮动元素的响应	<code>left</code> : 清除前面左浮动的影响 <code>right</code> : 清除前面右浮动的影响 <code>both</code> : 清除前面左右浮动的影响

七、定位

1. 相对定位

1.1 如何设置相对定位?

- 给元素设置 `position: relative` 即可实现相对定位。
- 可以使用 `left`、`right`、`top`、`bottom` 四个属性调整位置。

1.2 相对定位的参考点在哪里?

- 相对自己原来的位置

1.3 相对定位的特点:

- 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。
- 定位元素的显示层级比普通元素高, 无论什么定位, 显示层级都是一样的。

默认规则是:

- 定位的元素会盖在普通元素之上。

- 都发生定位的两个元素，后写的元素会盖在先写的元素之上。
- `left` 不能和 `right` 一起设置，`top` 和 `bottom` 不能一起设置。
- 相对定位的元素，也能继续浮动，但不推荐这样做。
- 相对行为的元素，也能通过 `margin` 调整位置，但不推荐这样做。

注意：绝大多数情况下，相对定位，会与绝对定位配合使用。

2. 绝对定位

2.1 如何设置绝对定位？

- 给元素设置 `position: absolute` 即可实现绝对定位。
- 可以使用 `left`、`right`、`top`、`bottom` 四个属性调整位置。

2.2 绝对定位的参考点在哪里？

- 参考它的包含块。

什么是包含块？

- 对于没有脱离文档流的元素：包含块就是父元素；
- 对于脱离文档流的元素：包含块是第一个拥有定位属性的祖先元素（如果所有祖先都没定位，那包含块就是整个页面）。

2.3 绝对定位元素的特点：

- 脱离文档流，会对后面的兄弟元素、父元素有影响。
- `left` 不能和 `right` 一起设置，`top` 和 `bottom` 不能一起设置。
- 绝对定位、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，也能通过 `margin` 调整位置，但不推荐这样做。
- 无论是什么元素（行内、行内块、块级）设置为绝对定位之后，都变成了定位元素。

何为定位元素？——默认宽、高都被内容所撑开，且能自由设置宽高。

3. 固定定位

3.1 如何设置为固定定位？

- 给元素设置 `position: fixed` 即可实现固定定位。
- 可以使用 `left`、`right`、`top`、`bottom` 四个属性调整位置。

3.2 固定定位的参考点在哪里？

- 参考它的视口

什么是视口？——对于 PC 浏览器来说，视口就是我们看网页的那扇“窗户”。

3.3 固定定位元素的特点

- 脱离文档流，会对后面的兄弟元素、父元素有影响。
- `left` 不能和 `right` 一起设置，`top` 和 `bottom` 不能一起设置。
- 固定定位和浮动不能同时设置，如果同时设置，浮动失效，以固定定位为主。

4. 固定定位的元素，也能通过 `margin` 调整位置，但不推荐这样做。
 5. 无论是什么元素（行内、行内块、块级）设置为固定定位之后，都变成了定位元素。
-

4. 粘性定位

4.1 如何设置为粘性定位？

- 给元素设置 `position:sticky` 即可实现粘性定位。
- 可以使用 `left`、`right`、`top`、`bottom` 四个属性调整位置，不过最常用的是 `top` 值。

4.2 粘性定位的参考点在哪里？

- 离它最近的一个拥有“滚动机制”的祖先元素，即便这个祖先不是最近的真实可滚动祖先。

4.3 粘性定位元素的特点

- 不会脱离文档流，它是一种专门用于窗口滚动时的新的定位方式。
- 最常用的是 `top` 值。
- 粘性定位和浮动可以同时设置，但不推荐这样做。
- 粘性定位的元素，也能通过 `margin` 调整位置，但不推荐这样做。

粘性定位和相对定位的特点基本一致，不同的是：粘性定位可以在元素到达某个位置时将其固定。

5. 定位层级

1. 定位元素的显示层级比普通元素高，无论什么定位，显示层级都是一样的。
2. 如果位置发生重叠，默认情况是：后面的元素，会显示在前面元素之上。
3. 可以通过 CSS 属性 `z-index` 调整元素的显示层级。
4. `z-index` 的属性值是数字，没有单位，值越大显示层级越高。
5. 只有定位的元素设置 `z-index` 才有效。
6. 如果 `z-index` 值大的元素，依然没有覆盖掉 `z-index` 值小的元素，那么请检查其包含块的层级。

6. 定位的特殊应用

注意：

1. 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。
2. 发生相对定位后，元素依然是之前的显示模式。
3. 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。

让定位元素的宽充满包含块

1. 块宽想与包含块一致，可以给定位元素同时设置 `left` 和 `right` 为 `0`。
2. 高度想与包含块一致，`top` 和 `bottom` 设置为 `0`。

让定位元素在包含块中居中

- 方案一：

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

- 方案二：

```
left: 50%;  
top: 50%;  
margin-left: 负的宽度一半;  
margin-top: 负的高度一半;
```

注意：该定位的元素必须设置宽高！！！

八、布局

1. 版心

- 在 PC 端网页中，一般都会有一个固定宽度且水平居中的盒子，来显示网页的主要内容，这是网页的版心。
- 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。



2. 常用布局名词

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

3. 重置默认样式

很多元素都有默认样式，比如：

1. `p` 元素有默认的上下 `margin`。
2. `h1~h6` 标题也有上下 `margin`，且字体加粗。
3. `body` 元素有默认的 `8px` 外边距。
4. 超链接有默认的文字颜色和下划线。
5. `ul` 元素有默认的左 `padding`。
6.

在早期，元素默认样式，能够让我们快速的绘制网页，但如今网页的设计越来越复杂，内容越来越多，而且很精细，这些默认样式会给我们绘制页面带来麻烦；而且这些默认样式，在不同的浏览器上呈现出来的效果也不一样，所以我们需要重置这些默认样式。

方案一：使用全局选择器

```
* {
  margin: 0;
  padding: 0;
  .....
}
```

此种方法，在简单案例中可以用一下，但实际开发中不会使用，因为 `*` 选择的是所有元素，而并不是所有的元素都有默认样式；而且我们重置时，有时候是需要做特定处理的，比如：想让 `a` 元素的文字是灰色，其他元素文字是蓝色。

方案二：reset.css

选择到具有默认样式的元素，清空其默认的样式。

经过 `reset` 后的网页，好似“一张白纸”，开发人员可根据设计稿，精细的去添加具体的样式。

方案三：Normalize.css

`Normalize.css` 是一种最新方案，它在清除默认样式的基础上，保留了一些有价值的默认样式。

- 官网地址：<http://necolas.github.io/normalize.css/>

相对于 `reset.css`，`Normalize.css` 有如下优点：

1. 保护了有价值的默认样式，而不是完全去掉它们。
2. 为大部分HTML元素提供一般化的样式。
3. 新增对 `HTML5` 元素的设置。
4. 对并集选择器的使用比较谨慎，有效避免调试工具杂乱。

备注：`Normalize.css` 的重置，和 `reset.css` 相比，更加的温和，开发时可根据实际情况进行选择。

一、CSS3 简介

1. CSS3 概述

- CSS3 是 CSS2 的升级版本，它在 CSS2 的基础上，新增了很多强大的新功能，从而解决一些实际面临的问题。
- CSS3 在未来会按照模块化的方式去发展：<https://www.w3.org/Style/CSS/current-work.html>
- CSS3 的新特性如下：
 - 新增了**更加实用的选择器**，例如：动态伪类选择器、目标伪类选择器、伪元素选择器等等。
 - 新增了**更好的视觉效果**，例如：圆角、阴影、渐变等。
 - 新增了**丰富的背景效果**，例如：支持多个背景图片，同时新增了若干个背景相关的属性。
 - 新增了**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没有安装的字体。
 - 增强了**颜色**，例如： HSL 、 HSLA 、 RGBA 几种新的颜色模式，新增 opacity 属性来控制透明度。
 - 增加了**2D 和 3D 变换**，例如：旋转、扭曲、缩放、位移等。
 - 增加**动画与过渡效果**，让效果的变换更具流线性、平滑性。
 -

2. CSS3私有前缀

2.1 什么是私有前缀

如下代码中的 `-webkit-` 就是私有前缀

```
div {  
    width:400px;  
    height:400px;  
    -webkit-border-radius: 20px;  
}
```

2.2 为什么要有私有前缀

- W3C 标准所提出的某个 CSS 特性，在被浏览器正式支持之前，浏览器厂商会根据浏览器的内核，使用私有前缀来测试该 CSS 特性，在浏览器正式支持该 CSS 特性后，就不需要私有前缀了。
- 举个例子：

```
-webkit-border-radius: 20px;  
-moz-border-radius: 20px;  
-ms-border-radius: 20px;  
-o-border-radius: 20px;  
border-radius: 20px;
```

- 查询 CSS3 兼容性的网站：<https://caniuse.com/>

2.3 常见浏览器私有前缀

- **Chrome 浏览器:** `-webkit-`
- **Safari 浏览器:** `-webkit-`
- **Firefox 浏览器:** `-moz-`
- **Edge 浏览器:** `-webkit-`
- 旧 **Opera 浏览器:** `-o-`
- 旧 **IE 浏览器:** `-ms-`

注意：

我们在编码时，不用过于关注浏览器私有前缀，不用绞尽脑汁的去记忆，也不用每个都去查询，因为常用的 **CSS3** 新特性，主流浏览器都是支持的，即便是为了老浏览器而加前缀，我们也可以借助现代的构建工具，去帮我们添加私有前缀。

二、CSS3 基本语法

1. CSS3 新增长度单位

1. **rem** 根元素字体大小的倍数，只与根元素字体大小有关。
2. **vw** 视口宽度的百分之多少 `10vw` 就是视口宽度的 `10%`。
3. **vh** 视口高度的百分之多少 `10vh` 就是视口高度的 `10%`。
4. **vmax** 视口宽高中大的那个的百分之多少。（了解即可）
5. **vmin** 视口宽高中小的那个的百分之多少。（了解即可）

2. CSS3 新增颜色设置方式

CSS3 新增了三种颜色设置方式，分别是：`rgba`、`hsl`、`hsla`，由于之前已经详细讲解，此处略过。

3. CSS3 新增选择器

CSS3 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；这些在 **CSS2** 中已经详细讲解，此处略过。

4. CSS3 新增盒模型相关属性

4.1. `box-sizing` 怪异盒模型

使用 `box-sizing` 属性可以设置盒模型的两种类型

可选值	含义
<code>content-box</code>	<code>width</code> 和 <code>height</code> 设置的是盒子内容区的大小。 (默认值)
<code>border-box</code>	<code>width</code> 和 <code>height</code> 设置的是盒子总大小。 (怪异盒模型)

4.2. `resize` 调整盒子大小

使用 `resize` 属性可以控制是否允许用户调节元素尺寸。

值	含义
<code>none</code>	不允许用户调整元素大小。 (默认)
<code>both</code>	用户可以调节元素的宽度和高度。
<code>horizontal</code>	用户可以调节元素的宽度。
<code>vertical</code>	用户可以调节元素的高度。

4.3. `box-shadow` 盒子阴影

使用 `box-shadow` 属性为盒子添加阴影。

- 语法:

```
box-shadow: h-shadow v-shadow blur spread color inset;
```

- 各个值的含义:

值	含义
<code>h-shadow</code>	水平阴影的位置，必须填写，可以为负值
<code>v-shadow</code>	垂直阴影的位置，必须填写，可以为负值
<code>blur</code>	可选，模糊距离
<code>spread</code>	可选，阴影的外延值
<code>color</code>	可选，阴影的颜色
<code>inset</code>	可选，将外部阴影改为内部阴影

- 默认值: `box-shadow:none` 表示没有阴影

- 示例:

```
/* 写两个值，含义：水平位置、垂直位置 */
```

```
box-shadow: 10px 10px;
```

```
/* 写三个值，含义：水平位置、垂直位置、颜色 */
```

```
box-shadow: 10px 10px red;
```

```
/* 写三个值，含义：水平位置、垂直位置、模糊值 */
box-shadow: 10px 10px 10px;

/* 写四个值，含义：水平位置、垂直位置、模糊值、颜色 */
box-shadow: 10px 10px 10px red;

/* 写五个值，含义：水平位置、垂直位置、模糊值、外延值、颜色 */
box-shadow: 10px 10px 10px 10px blue;

/* 写六个值，含义：水平位置、垂直位置、模糊值、外延值、颜色、内阴影 */
box-shadow: 10px 10px 20px 3px blue inset;
```

4.4. opacity 不透明度

- **opacity** 属性能为整个元素添加透明效果，值是 0 到 1 之间的数字，0 是完全透明，1 表示完全不透明。

opacity 与 **rgba** 的区别？

opacity 是一个属性，设置的是整个元素（包括元素里的内容）的不透明度。

rgba 是颜色的设置方式，用于设置颜色，它的透明度，仅仅是调整颜色的透明度。

5. CSS3 新增背景属性

5.1. background-origin

- 作用：设置背景图的原点。
- 语法
 1. **padding-box**：从 **padding** 区域开始显示背景图像。——默认值
 2. **border-box**：从 **border** 区域开始显示背景图像。
 3. **content-box**：从 **content** 区域开始显示背景图像。

5.2. background-clip

- 作用：设置背景图的向外裁剪的区域。
- 语法
 1. **border-box**：从 **border** 区域开始向外裁剪背景。——默认值
 2. **padding-box**：从 **padding** 区域开始向外裁剪背景。
 3. **content-box**：从 **content** 区域开始向外裁剪背景。
 4. **text**：背景图只呈现在文字上。

注意：若值为 **text**，那么 **background-clip** 要加上 **-webkit-** 前缀。

5.3. background-size

- 作用：设置背景图的尺寸。
- 语法：
 - 用长度值指定背景图片大小，不允许负值。

```
background-size: 300px 200px;
```
 - 用百分比指定背景图片大小，不允许负值。

```
background-size: 100% 100%;
```
 - auto**：背景图片的真实大小。——默认值
 - contain**：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，再将完整背景图片包含在容器内，但要注意：可能会造成容器里部分区域没有背景图片。

```
background-size: contain;
```
 - cover**：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，但要注意：背景图片有可能显示不完整。——相对比较好的选择

```
background-size: cover;
```

5.4. background 复合属性

- 语法：

```
background: color url repeat position / size origin clip
```

注意：

 - origin** 和 **clip** 的值如果一样，如果只写一个值，则 **origin** 和 **clip** 都设置；如果设置了两个值，前面的是 **origin**，后面的 **clip**。
 - size** 的值必须写在 **position** 值的后面，并且用 **/** 分开。

5.5. 多背景图

CSS3 允许元素设置多个背景图片

```
/* 添加多个背景图 */
background: url(..../images/bg-lt.png) no-repeat,
            url(..../images/bg-rt.png) no-repeat right top,
            url(..../images/bg-lb.png) no-repeat left bottom,
            url(..../images/bg-rb.png) no-repeat right bottom;
```

6. CSS3新增边框属性

6.1 边框圆角

- 在 CSS 中，使用 `border-radius` 属性可以将盒子变为圆角。
- 同时设置四个角的圆角：

```
border-radius:10px;
```

- 分开设置每个角的圆角（几乎不用）：

属性名	作用
<code>border-top-left-radius</code>	设置左上角圆角半径： 1.一个值是正圆半径， 2.两个值分别是椭圆的 <code>x</code> 半径、 <code>y</code> 半径。
<code>border-top-right-radius</code>	设置右上角圆角半径： 1.一个值是正圆半径， 2.两个值分别是椭圆的 <code>x</code> 半径、 <code>y</code> 半径。
<code>border-bottom-right-radius</code>	设置右下角圆角半径： 1.一个值是正圆半径， 2.两个值分别是椭圆的 <code>x</code> 半径、 <code>y</code> 半径。
<code>border-bottom-left-radius</code>	设置左下角圆角半径： 1.一个值是正圆半径， 2.两个值分别是椭圆的 <code>x</code> 半径、 <code>y</code> 半径。

- 分开设置每个角的圆角，综合写法（几乎不用）：

```
border-radius: 左上角x 右上角x 右下角x 左下角x / 左上y 右上y 右下y 左下y
```

6.2 边框外轮廓（了解）

- `outline-width`：外轮廓的宽度。
- `outline-color`：外轮廓的颜色。
- `outline-style`：外轮廓的风格。
 - `none`：无轮廓
 - `dotted`：点状轮廓
 - `dashed`：虚线轮廓
 - `solid`：实线轮廓
 - `double`：双线轮廓
- `outline-offset` 设置外轮廓与边框的距离，正负值都可以设置。

注意：`outline-offset` 不是 `outline` 的子属性，是一个独立的属性。

- `outline` 复合属性

```
outline:50px solid blue;
```

7. CSS3新增文本属性

7.1 文本阴影

- 在 `CSS3` 中，我们可以使用 `text-shadow` 属性给文本添加阴影。
- 语法：

```
text-shadow: h-shadow v-shadow blur color;
```

值	描述
<code>h-shadow</code>	必需写，水平阴影的位置。允许负值。
<code>v-shadow</code>	必需写，垂直阴影的位置。允许负值。
<code>blur</code>	可选，模糊的距离。
<code>color</code>	可选，阴影的颜色

默认值：`text-shadow:none` 表示没有阴影。

7.2 文本换行

- 在 `CSS3` 中，我们可以使用 `white-space` 属性设置文本换行方式。
- 常用值如下：

值	含义
<code>normal</code>	文本超出边界自动换行，文本中的换行被浏览器识别为一个空格。 (默认值)
<code>pre</code>	原样输出，与 <code>pre</code> 标签的效果相同。
<code>pre-wrap</code>	在 <code>pre</code> 效果的基础上，超出元素边界自动换行。
<code>pre-line</code>	在 <code>pre</code> 效果的基础上，超出元素边界自动换行，且只识别文本中的换行，空格会忽略。
<code>nowrap</code>	强制不换行

7.3 文本溢出

- 在 `CSS3` 中，我们可以使用 `text-overflow` 属性设置文本内容溢出时的呈现模式。
- 常用值如下：

值	含义
<code>clip</code>	当内联内容溢出时，将溢出部分裁切掉。 (默认值)
<code>ellipsis</code>	当内联内容溢出块容器时，将溢出部分替换为 ...。

注意：要使得 `text-overflow` 属性生效，块容器必须显式定义 `overflow` 为非 `visible` 值，`white-space` 为 `nowrap` 值。

7.4 文本修饰

- CSS3 升级了 `text-decoration` 属性，让其变成了复合属性。

```
text-decoration: text-decoration-line || text-decoration-style || text-decoration-color
```

- 子属性及其含义：

- `text-decoration-line` 设置文本装饰线的位置
 - `none`：指定文字无装饰（默认值）
 - `underline`：指定文字的装饰是下划线
 - `overline`：指定文字的装饰是上划线
 - `line-through`：指定文字的装饰是贯穿线
- `text-decoration-style` 文本装饰线条的形状
 - `solid`：实线（默认）
 - `double`：双线
 - `dotted`：点状线条
 - `dashed`：虚线
 - `wavy`：波浪线
- `text-decoration-color` 文本装饰线条的颜色

7.5 文本描边

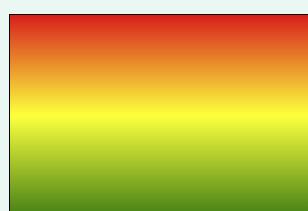
注意：文字描边功能仅 `webkit` 内核浏览器支持。

- `-webkit-text-stroke-width`：设置文字描边的宽度，写长度值。
- `-webkit-text-stroke-color`：设置文字描边的颜色，写颜色值。
- `-webkit-text-stroke`：复合属性，设置文字描边宽度和颜色。

8. CSS3 新增渐变

8.1 线性渐变

- 多个颜色之间的渐变，默认从上到下渐变。



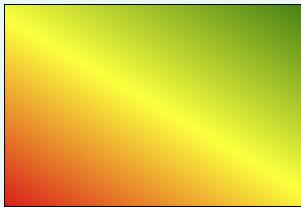
```
background-image: linear-gradient(red, yellow, green);
```

- 使用关键词设置线性渐变的方向。



```
background-image: linear-gradient(to top, red, yellow, green);  
background-image: linear-gradient(to right top, red, yellow, green);
```

- 使用角度设置线性渐变的方向。



```
background-image: linear-gradient(30deg, red, yellow, green);
```

- 调整开始渐变的位置。



```
background-image: linear-gradient(red 50px, yellow 100px, green 150px);
```

8.2 径向渐变

- 多个颜色之间的渐变， 默认从圆心四散。 (注意：不一定是正圆，要看容器本身宽高比)



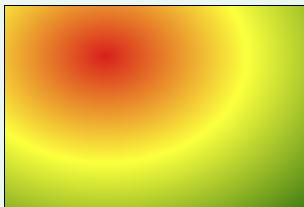
```
background-image: radial-gradient(red, yellow, green);
```

- 使用关键词调整渐变圆的圆心位置。



```
background-image: radial-gradient(at right top, red, yellow, green);
```

- 使用像素值调整渐变圆的圆心位置。



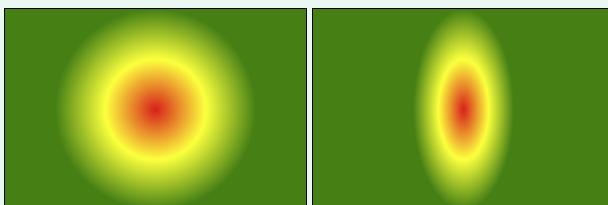
```
background-image: radial-gradient(at 100px 50px, red, yellow, green);
```

- 调整渐变形状为正圆。



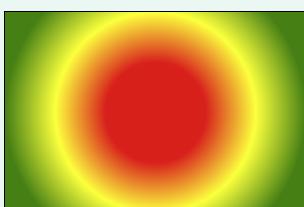
```
background-image: radial-gradient(circle, red, yellow, green);
```

- 调整形状的半径。



```
background-image: radial-gradient(100px, red, yellow, green);  
background-image: radial-gradient(50px 100px, red, yellow, green);
```

- 调整开始渐变的位置。



```
background-image: radial-gradient(red 50px, yellow 100px, green 150px);
```

8.3 重复渐变

无论线性渐变，还是径向渐变，在没有发生渐变的位置，继续进行渐变，就为重复渐变。

- 使用 `repeating-linear-gradient` 进行重复线性渐变，具体参数同 `linear-gradient`。
- 使用 `repeating-radial-gradient` 进行重复径向渐变，具体参数同 `radial-gradient`。

我们可以利用渐变，做出很多有意思的效果：例如：横格纸、立体球等等。

9. web 字体

9.1 基本用法

可以通过 `@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。

- 语法（简写方式）

```
@font-face {  
    font-family: "情书字体";  
    src: url('./方正手迹.ttf');  
}
```

- 语法（高兼容性写法）

```
@font-face {  
    font-family: "atguigu";  
    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*/  
        url('webfont.svg#webfont') format('svg'); /* iOS 4.1- */  
}
```

9.2 定制字体

- 中文的字体文件很大，使用完整的字体文件不现实，通常针对某几个文字进行单独定制。
- 可使用阿里 Web 字体定制工具：<https://www.iconfont.cn/webfont>

9.3 字体图标

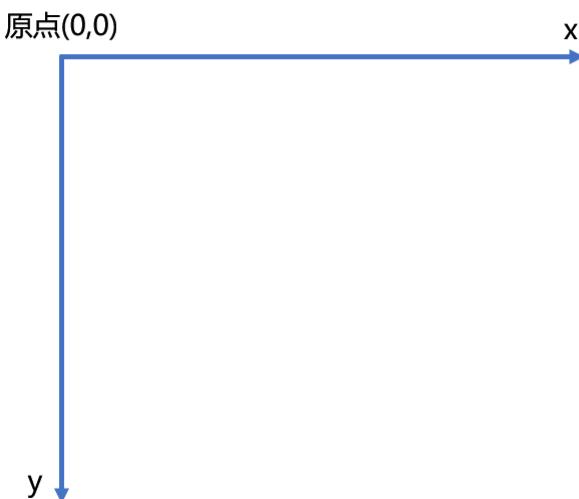
- 相比图片更加清晰。
- 灵活性高，更方便改变大小、颜色、风格等。
- 兼容性好，`IE` 也能支持。

字体图标的具体使用方式，每个平台不尽相同，最好参考平台使用指南，视频中我们是以使用最多的阿里图标库作为演示。

阿里图标官网地址：<https://www.iconfont.cn/>

10. 2D变换

前提：二维坐标系如下图所示



10.1. 2D位移

2D 位移可以改变元素的位置，具体使用方式如下：

- 先给元素添加 **转换属性** `transform`
- 编写 `transform` 的具体值，相关可选值如下：

值	含义
<code>translateX</code>	设置水平方向位移，需指定长度值；若指定的是百分比，是参考自身宽度的百分比。
<code>translateY</code>	设置垂直方向位移，需指定长度值；若指定的是百分比，是参考自身高度的百分比。
<code>translate</code>	一个值代表水平方向，两个值代表：水平和垂直方向。

- 注意点：

- 位移与相对定位很相似，都不脱离文档流，不会影响到其它元素。
- 与相对定位的区别：相对定位的百分比值，参考的是其父元素；定位的百分比值，参考的是其自身。
- 浏览器针对位移有优化，与定位相比，浏览器处理位移的效率更高。
- `transform` 可以链式编写，例如：

```
transform: translateX(30px) translateY(40px);
```

- 位移对行内元素无效。
- 位移配合定位，可实现元素水平垂直居中

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

10.2. 2D缩放

2D 缩放是指：让元素放大或缩小，具体使用方式如下：

1. 先给元素添加 转换属性 `transform`
2. 编写 `transform` 的具体值，相关可选值如下：

值	含义
<code>scaleX</code>	设置水平方向的缩放比例，值为一个数字， <code>1</code> 表示不缩放，大于 <code>1</code> 放大，小于 <code>1</code> 缩小。
<code>scaleY</code>	设置垂直方向的缩放比例，值为一个数字， <code>1</code> 表示不缩放，大于 <code>1</code> 放大，小于 <code>1</code> 缩小。
<code>scale</code>	同时设置水平方向、垂直方向的缩放比例，一个值代表同时设置水平和垂直缩放；两个值分别代表：水平缩放、垂直缩放。

3. 注意点：

1. `scale` 的值，是支持写负数的，但几乎不用，因为容易让人产生误解。
2. 借助缩放，可实现小于 `12px` 的文字。

10.3. 2D旋转

2D 旋转是指：让元素在二维平面内，顺时针旋转或逆时针旋转，具体使用方式如下：

1. 先给元素添加 转换属性 `transform`
2. 编写 `transform` 的具体值，相关可选值如下：

值	含义
<code>rotate</code>	设置旋转角度，需指定一个角度值(<code>deg</code>)，正值顺时针，负值逆时针。

注意：`rotateZ(20deg)` 相当于 `rotate(20deg)`，当然到了 3D 变换的时候，还能写：
`rotate(x, x, x)`

10.4. 2D扭曲（了解）

2D 扭曲是指：让元素在二维平面内被“拉扯”，进而“走形”，实际开发几乎不用，了解即可，具体使用方式如下：

1. 先给元素添加 转换属性 `transform`
2. 编写 `transform` 的具体值，相关可选值如下：

值	含义
<code>skewX</code>	设置元素在水平方向扭曲，值为角度值，会将元素的左上角、右下角 拉扯。
<code>skewY</code>	设置元素在垂直方向扭曲，值为角度值，会将元素的左上角、右下角 拉扯。
<code>skew</code>	一个值代表 <code>skewX</code> ，两个值分别代表： <code>skewX</code> 、 <code>skewY</code>

10.5. 多重变换

多个变换，可以同时使用一个 `transform` 来编写。

```
transform: translate(-50%, -50%) rotate(45deg);
```

注意点：多重变换时，建议最后旋转。

10.6. 变换原点

- 元素变换时，默认的原点是元素的中心，使用 `transform-origin` 可以设置变换的原点。
- 修改变换原点对位移没有影响，对旋转和缩放会产生影响。
- 如果提供两个值，第一个用于横坐标，第二个用于纵坐标。
- 如果只提供一个，若是像素值，表示横坐标，纵坐标取 `50%`；若是关键词，则另一个坐标取 `50%`

1. `transform-origin: 50% 50%`，变换原点在元素的中心位置，百分比是相对于自身。——默认值
2. `transform-origin: left top`，变换原点在元素的左上角。
3. `transform-origin: 50px 50px`，变换原点距离元素左上角 `50px 50px` 的位置。
4. `transform-origin: 0`，只写一个值的时候，第二个值默认为 `50%`。

11. 3D变换

11.1. 开启3D空间

重要原则：元素进行 `3D` 变换的首要操作：**父元素**必须开启 `3D` 空间！

使用 `transform-style` 开启 `3D` 空间，可选值如下：

- `flat`：让子元素位于此元素的二维平面内（`2D` 空间）——默认值
- `preserve-3d`：让子元素位于此元素的三维空间内（`3D` 空间）

11.2. 设置景深

何为景深？——指定观察者与 `z=0` 平面的距离，能让发生 3D 变换的元素，产生透视效果，看来更加立体。

使用 `perspective` 设置景深，可选值如下：

- `none`：不指定透视——（默认值）
- `长度值`：指定观察者距离 `z=0` 平面的距离，不允许负值。

注意：`perspective` 设置给发生 3D 变换元素的父元素！

11.3. 透视点位置

所谓透视点位置，就是观察者位置；默认的透视点在元素的中心。

使用 `perspective-origin` 设置观察者位置（透视点的位置），例如：

```
/* 相对坐标轴往右偏移400px， 向下偏移300px（相当于人蹲下300像素，然后向右移动400像素看元素） */
perspective-origin: 400px 300px;
```

注意：通常情况下，我们不需要调整透视点位置。

11.4. 3D 位移

3D 位移是在 2D 位移的基础上，可以让元素沿 `z` 轴位移，具体使用方式如下：

1. 先给元素添加 转换属性 `transform`
2. 编写 `transform` 的具体值， 3D 相关可选值如下：

值	含义
<code>translateZ</code>	设置 <code>z</code> 轴位移，需指定长度值，正值向屏幕外，负值向屏幕里，且不能写百分比。
<code>translate3d</code>	第1个参数对应 <code>x</code> 轴，第2个参数对应 <code>y</code> 轴，第3个参数对应 <code>z</code> 轴，且均不能省略。

11.5. 3D 旋转

3D 旋转是在 2D 旋转的基础上，可以让元素沿 `x` 轴和 `y` 轴旋转，具体使用方式如下：

1. 先给元素添加 转换属性 `transform`
2. 编写 `transform` 的具体值， 3D 相关可选值如下：

值	含义
rotateX	设置 x 轴旋转角度，需指定一个角度值(<code>deg</code>)，面对 x 轴正方向：正值顺时针，负值逆时针。
rotateY	设置 y 轴旋转角度，需指定一个角度值(<code>deg</code>)，面对 y 轴正方向：正值顺时针，负值逆时针。
rotate3d	前 3 个参数分别表示坐标轴： <code>x</code> , <code>y</code> , <code>z</code> , 第 4 个参数表示旋转的角度，参数不允许省略。 例如： <code>transform: rotate3d(1, 1, 1, 30deg)</code> , 意思是： <code>x</code> 、 <code>y</code> 、 <code>z</code> 分别旋转 30 度。

11.6. 3D 缩放

3D 缩放是在 2D 缩放的基础上，可以让元素沿 z 轴缩放，具体使用方式如下：

- 先给元素添加 **转换属性** `transform`
- 编写 `transform` 的具体值， 3D 相关可选值如下：

值	含义
<code>scaleZ</code>	设置 z 轴方向的缩放比例，值为一个数字， 1 表示不缩放，大于 1 放大，小于 1 缩小。
<code>scale3d</code>	第1个参数对应 x 轴，第2个参数对应 y 轴，第3个参数对应 z 轴，参数不允许省略。

11.7. 多重变换

多个变换，可以同时使用一个 `transform` 来编写。

```
transform: translateZ(100px) scaleZ(3) rotateY(40deg);
```

注意点：多重变换时，建议最后旋转。

11.8. 背部可见性

使用 `backface-visibility` 指定元素背面，在面向用户时是否可见，常用值如下：

- `visible` : 指定元素背面可见，允许显示正面的镜像。-- 默认值
- `hidden` : 指定元素背面不可见

注意： `backface-visibility` 需要加在发生 3D 变换元素的自身上。

12. 过渡

过渡可以在不使用 Flash 动画，不使用 JavaScript 的情况下，让元素从一种样式，平滑过渡为另一种样式。

12.1. transition-property

- 作用：定义哪个属性需要过渡，只有在该属性中定义的属性（比如宽、高、颜色等）才会以有过渡效果。
- 常用值：
 - none：不过渡任何属性。
 - all：过渡所有能过渡的属性。
 - 具体某个属性名，例如：width、height，若有多个以逗号分隔。

不是所有的属性都能过渡，值为数字，或者值能转为数字的属性，都支持过渡，否则不支持过渡。

常见的支持过渡的属性有：颜色、长度值、百分比、z-index、opacity、2D 变换属性、3D 变换属性、阴影。

12.2. transition-duration

- 作用：设置过渡的持续时间，即：一个状态过渡到另外一个状态耗时多久。
- 常用值：
 - 0：没有任何过渡时间——默认值。
 - s 或 ms：秒或毫秒。
 - 列表：
 - 如果想让所有属性都持续一个时间，那就写一个值。
 - 如果想让每个属性持续不同的时间那就写一个时间的列表。

12.3. transition-delay

- 作用：指定开始过渡的延迟时间，单位：s 或 ms

12.4. transition-timing-function

- 作用：设置过渡的类型
- 常用值：
 - ease：平滑过渡——默认值
 - linear：线性过渡
 - ease-in：慢→快
 - ease-out：快→慢
 - ease-in-out：慢→快→慢
 - step-start：等同于 steps(1, start)
 - step-end：等同于 steps(1, end)
 - steps(integer, ?)：接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是 start 或 end，指定每一步的值发生变化的时间点。第二个参数默认值为 end。
 - cubic-bezie (number, number, number, number)：特定的贝塞尔曲线类型。

在线制作贝赛尔曲线：<https://cubic-bezier.com>

12.5. transition 复合属性

- 如果设置了一个时间，表示 `duration`；如果设置了两个时间，第一是 `duration`，第二个是 `delay`；其他值没有顺序要求。

```
transition:1s 1s linear all;
```

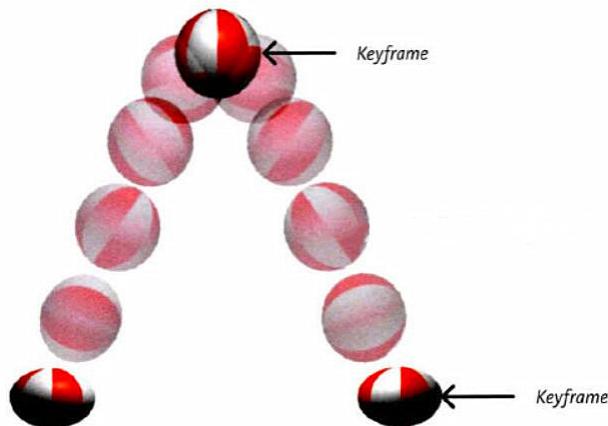
13. 动画

13.1. 什么是帧

- 一段动画，就是一段时间内连续播放 `n` 个画面。每一张画面，我们管它叫做“帧”。一定时间内连续快速播放若干个帧，就成了人眼中所看到的动画。同样时间内，播放的帧数越多，画面看起来越流畅。

13.2. 什么是关键帧

- 关键帧指的是，在构成一段动画的若干帧中，起到决定性作用的 `2-3` 帧。



13.3. 动画的基本使用

- 第一步：定义关键帧（定义动画）

1. 简单方式定义：

```
/*写法一*/
@keyframes 动画名 {
    from {
        /*property1:value1*/
        /*property2:value2*/
    }
    to {
        /*property1:value1*/
    }
}
```

2. 完整方式定义：

```
@keyframes 动画名 {
    0% {
        /*property1:value1*/
    }
    20% {
        /*property1:value1*/
    }
    40% {
        /*property1:value1*/
    }
    60% {
        /*property1:value1*/
    }
    80% {
        /*property1:value1*/
    }
    100% {
        /*property1:value1*/
    }
}
```

- 第二步：给元素应用动画，用到的属性如下：

1. `animation-name`：给元素指定具体的动画（具体的关键帧）
2. `animation-duration`：设置动画所需时间
3. `animation-delay`：设置动画延迟

```
.box {
    /* 指定动画 */
    animation-name: testKey;
    /* 设置动画所需时间 */
    animation-duration: 5s;
    /* 设置动画延迟 */
    animation-delay: 0.5s;
}
```

4. 动画的其他属性

- `animation-timing-function`，设置动画的类型，常用值如下：

1. `ease`：平滑动画——默认值
2. `linear`：线性过渡
3. `ease-in`：慢→快
4. `ease-out`：快→慢
5. `ease-in-out`：慢→快→慢
6. `step-start`：等同于 `steps(1, start)`
7. `step-end`：等同于 `steps(1, end)`
8. `steps(integer, ?)`：接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是 `start` 或 `end`，指定每一步的值发生变化的时间点。第二个参数默认值为 `end`。
9. `cubic-bezie` (`number, number, number, number`)：特定的贝塞尔曲线类型。

- `animation-iteration-count`，指定动画的播放次数，常用值如下：

- 1. `number` : 动画循环次数
- 2. `infinite` : 无限循环

- `animation-direction` , 指定动画方向, 常用值如下:

- 1. `normal` : 正常方向(默认)
- 2. `reverse` : 反方向运行
- 3. `alternate` : 动画先正常运行再反方向运行, 并持续交替运行
- 4. `alternate-reverse` : 动画先反运行再正方向运行, 并持续交替运行

- `animation-fill-mode` , 设置动画之外的状态

- 1. `forwards` : 设置对象状态为动画结束时的状态
- 2. `backwards` : 设置对象状态为动画开始时的状态

- `animation-play-state` , 设置动画的播放状态, 常用值如下:

- 1. `running` : 运动(默认)
- 2. `paused` : 暂停

5. 动画复合属性

只设置一个时间表示 `duration` , 设置两个时间分别是: `duration` 和 `delay` , 其他属性没有数量和顺序要求。

```
.inner {  
    animation: atguigu 3s 0.5s linear 2 alternate-reverse forwards;  
}
```

备注: `animation-play-state` 一般单独使用。

14. 多列布局

作用: 专门用于实现类似于报纸的布局。

“问责”也要“负责”

——为基层减负，为实干撑腰⑤

本报评论部

通过强化责任追究，约束不作为、整治乱作为，从而唤醒责任意识、激发担当精神，这才是问责的价值指向

以法纪为准绳严肃问责，以事实为依据规范问责，以问题为靶心精准问责，以容错为原则慎重问责，才能起到问责一个、警醒一片的功效，才能克服“多干多错、不干不错”的心态，才能激发担当尽责、奋发有为的精神

“正确对待被问责的干部，对影响期满、表现好的干部，符合条件的，该使用的要使用”“保障党员权利，及时为干部澄清正名，严肃查处诬告陷害行为”“改进谈话和函询工作方法，有效减轻干部不必要的心理负担”……中办近日发出的《关于解决形式主义突出问题为基层减负的通知》，专门拿出一个部分，强调完善问责制度和激励关怀机制，着力解决干部不敢担当作为的问题。这就是对“问责”的正本清源，也是对“负责”的鲜明号召。

动员千遍，不如问责一次。依规治党、依法治国，问责是一个有力抓手。党的十八大以来，从强化问责工作、落实“两个责任”的改革创新，到巩固实践成果、扎紧制度笼子的立规创举，我们党把问责作为管党治党利器，先后对山西煤铁方式腐敗、湖南衡阳破坏选举案、四川南充和辽宁拉票贿选案、陕西秦岭北麓违建别墅问责等严肃问责，问责不主动、追责不尽力的现象大为减少，失责必问、问责必严成为常态。有力推动了管党治党从宽松软走向严紧硬，也让“有权必有责、有责要担当、失责必追究”逐渐成为普遍共识。

与此同时，由于对中央精神和党内法规学习不透彻、领会不深刻，问责泛化

简单化的现象时有发生。比如，4分钟内因没能及时接听脱贫攻坚督查组电话，扶贫干部被公开通报“给予党内警告处分”；扶贫手册中写错两个标点符号，被通报批评……这些执纪简化化、问责粗线条甚至乱问责、错问责、问错责的问题，虽然事后相关处理被撤销，却也造成了一些不良影响。特别是，类似“躺着中枪”的职能式问责、“刚播种就要收获”的计时式问责、只为舆情降温的灭火式问责、对困难不闻不同的机械式问责，不仅会挫伤基层干部的积极性，也无形之中削弱了问责的权威性。

问责只是手段，负责才是目的。通过强化责任追究，约束不作为、整治乱作为，从而唤醒责任意识、激发担当精神，这才是问责的价值指向。正因如此，《通知》指出，“坚持严管和厚爱结合，实事求是、依规依纪依法严肃问责、规范问责、精准问责、慎重问责，真正起到问责一个、警醒一片的效果。”如果说，问责是一把戒尺，那么用好这把尺子既要讲规则，也要讲艺术，既要讲政策，也要有温度。从此前要求运用好监督执纪“四种形态”，到这次要求“严肃、规范、精准、慎重”问责，都体现了严管和厚爱、约束和激励的辩证统一。

既是对党和国家事业的真诚担当，也是对党员干部的真正负责。

换句话说，问责也是一把手术刀，其威力不仅在于刃之锋利，更在于术之高超。认真领会贯彻《通知》精神，以法纪为准绳严问责，以事实为依据规范问责，以问题为靶心精准问责，以容错为原则慎重问责，才能起到问责一个、警醒一片的功效，才能克服“多干多错、不干不错”的心态，才能激发担当尽责、奋发有为的精神。“为担当者担当，为负责者负责。”由此可以理解，为何《通知》明确要求，把“三个区分开来”的要求具体化，正确把握干部在工作中出现失误错误的性质和影响，切实保护干部干事创业的积极性。

习近平总书记强调：“干部就要有担当，有多大的担当才能干多大的事业，尽多大责任才会有多大成就。”“我们的权力是党和人民赋予的，是为党和人民做事情用的，只能用来为党分忧、为国干事、为民谋利。坚持权责一致、严格问责，坚持层层负责、人人担当，为基层干部担当作为撑腰，为干事创业者设置‘减压阀’，我们就能够为基层治理注入源源不断的正能量。”

(本系列评论到此结束)

常用属性如下：

- `column-count`：指定列数，值是数字。
- `column-width`：指定列宽，值是长度。
- `columns`：同时指定列宽和列数，复合属性；值没有数量和顺序要求。
- `column-gap`：设置列边距，值是长度。
- `column-rule-style`：设置列与列之间边框的风格，值与 `border-style` 一致。
- `column-rule-width`：设置列与列之间边框的宽度，值是长度。
- `column-rule-color`：设置列与列之间边框的颜色。
- `column-rule`：设置列边框，复合属性。
- `column-span` 指定是否跨列；值: `none`、`all`。

15.伸缩盒模型

1. 伸缩盒模型简介

- 2009 年，W3C 提出了一种新的盒子模型——Flexible Box（伸缩盒模型，又称：弹性盒子）。
- 它可以轻松的控制：元素分布方式、元素对齐方式、元素视觉顺序……
- 截止目前，除了在部分 IE 浏览器不支持，其他浏览器均已全部支持。
- 伸缩盒模型的出现，逐渐演变出了一套新的布局方案——flex 布局。

小贴士：

1. 传统布局是指：基于传统盒状模型，主要靠：`display` 属性 + `position` 属性 + `float` 属性。
2. `flex` 布局目前在移动端应用比较广泛，因为传统布局不能很好的呈现在移动设备上。

2. 伸缩容器、伸缩项目

- **伸缩容器**: 开启了 `flex` 的元素，就是：伸缩容器。

1. 给元素设置：`display:flex` 或 `display:inline-flex`，该元素就变为了伸缩容器。
2. `display:inline-flex` 很少使用，因为可以给多个伸缩容器的父容器，也设置为伸缩容器。
3. 一个元素可以同时是：伸缩容器、伸缩项目。

- **伸缩项目**: 伸缩容器所有**子元素**自动成为了：伸缩项目。

1. 仅伸缩容器的**子元素**成为了伸缩项目，孙子元素、重孙子元素等后代，不是伸缩项目。
2. 无论原来是哪种元素（块、行内块、行内），一旦成为了伸缩项目，全都会“块化”。

3. 主轴与侧轴

- **主轴**: 伸缩项目沿着主轴排列，主轴默认是水平的，默认方向是：从左到右（左边是起点，右边是终点）。
- **侧轴**: 与主轴垂直的就是侧轴，侧轴默认是垂直的，默认方向是：从上到下（上边是起点，下边是终点）。

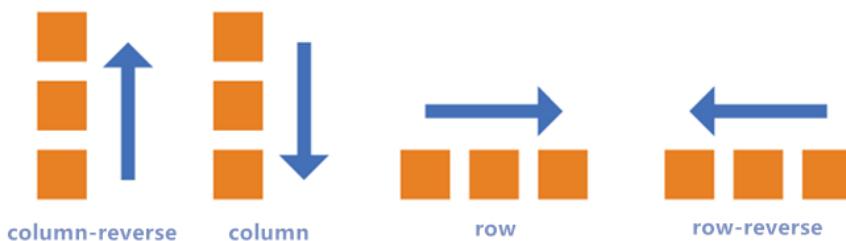
4. 主轴方向

- 属性名：`flex-direction`

- 常用值如下：

1. `row` : 主轴方向水平从左到右 —— 默认值
2. `row-reverse` : 主轴方向水平从右到左。
3. `column` : 主轴方向垂直从上到下。
4. `column-reverse` : 主轴方向垂直从下到上。

`flex-direction`属性



注意：改变了主轴的方向，侧轴方向也随之改变。

5. 主轴换行方式

- 属性名: `flex-wrap`

- 常用值如下:

- `nowrap` : 默认值, 不换行。

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- `wrap` : 自动换行, 伸缩容器不够自动换行。

1	2	3	4	5	6	7	8
9	10	11	12				

- `wrap-reverse` : 反向换行。

9	10	11	12				
1	2	3	4	5	6	7	8

6. flex-flow

- `flex-flow` 是一个复合属性, 复合了 `flex-direction` 和 `flex-wrap` 两个属性。值没有顺序要求。

```
flex-flow: row wrap;
```

7. 主轴对齐方式

- 属性名: `justify-content`

- 常用值如下:

- `flex-start` : 主轴起点对齐。 -- 默认值
- `flex-end` : 主轴终点对齐。
- `center` : 居中对齐
- `space-between` : 均匀分布, 两端对齐 (最常用)。
- `space-around` : 均匀分布, 两端距离是中间距离的一半。
- `space-evenly` : 均匀分布, 两端距离与中间距离一致。

`flex-start`



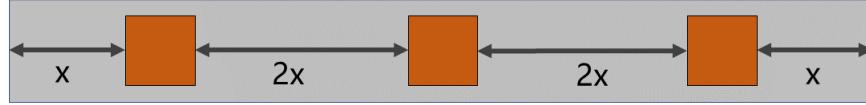
`flex-end`



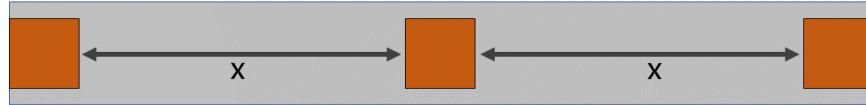
`center`



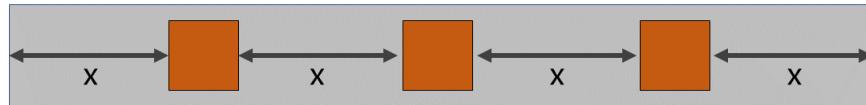
`space-around`



`space-between`



`space-evenly`



8. 侧轴对齐方式

8.1 一行的情况

- 所需属性: `align-items`
- 常用值如下:
 1. `flex-start` : 侧轴的起点对齐。
 2. `flex-end` : 侧轴的终点对齐。
 3. `center` : 侧轴的中点对齐。
 4. `baseline` : 伸缩项目的第一行文字的基线对齐。
 5. `stretch` : 如果伸缩项目未设置高度, 将占满整个容器的高度。— (默认值)

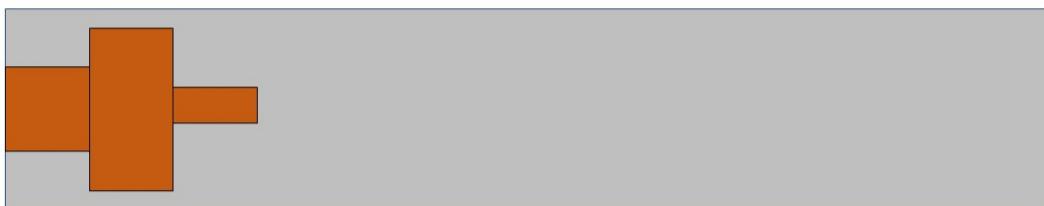
`flex-start`



`flex-end`



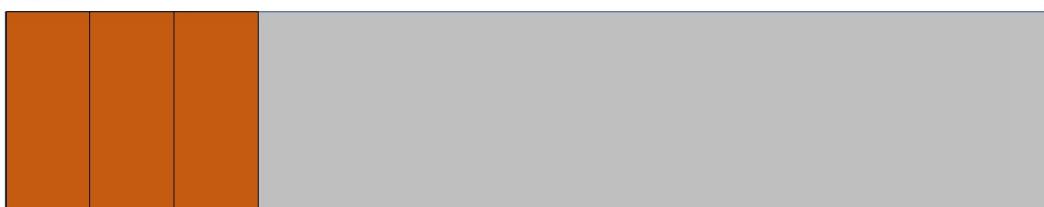
`center`



`baseline`



`stretch`

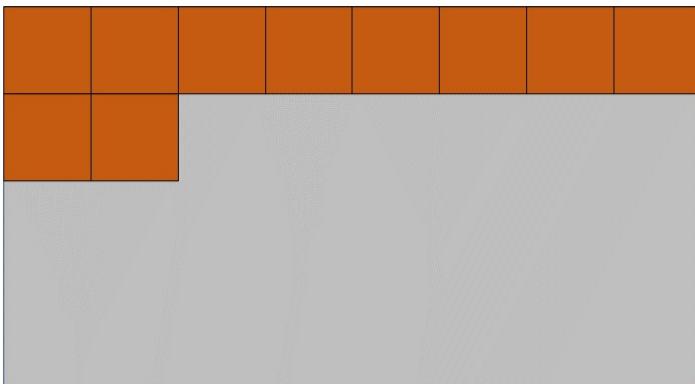


8.2 多行的情况

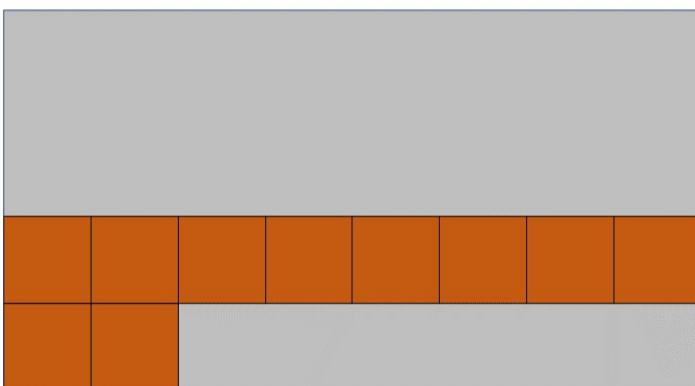
- 所需属性: `align-content`
- 常用值如下:
 1. `flex-start` : 与侧轴的起点对齐。
 2. `flex-end` : 与侧轴的终点对齐。
 3. `center` : 与侧轴的中点对齐。
 4. `space-between` : 与侧轴两端对齐，中间平均分布。
 5. `space-around` : 伸缩项目间的距离相等，比距边缘大一倍。
 6. `space-evenly` : 在侧轴上完全平分。
 7. `stretch` : 占满整个侧轴。——默认值

`flex-start`

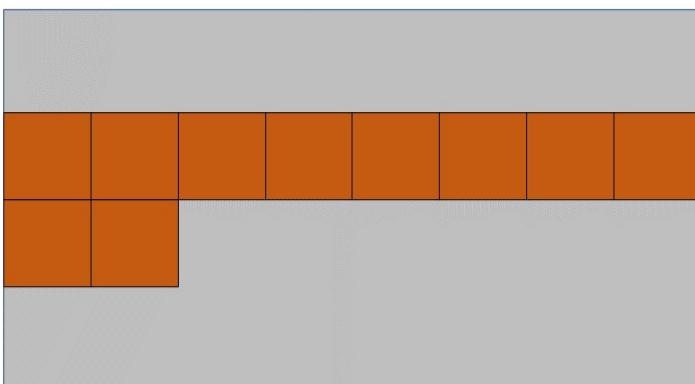
flex-start



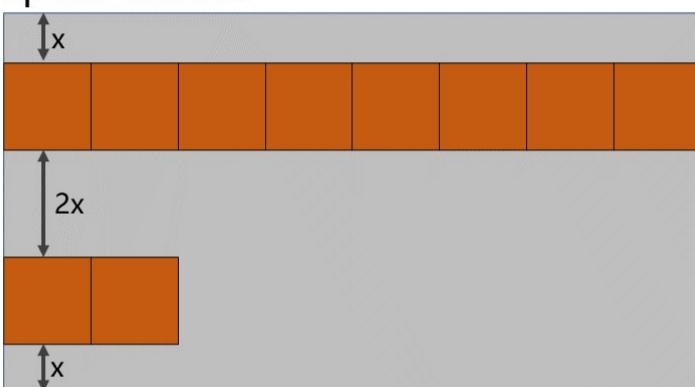
flex-end



center

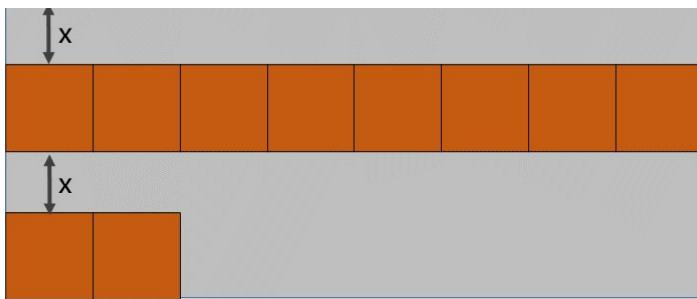


space-around

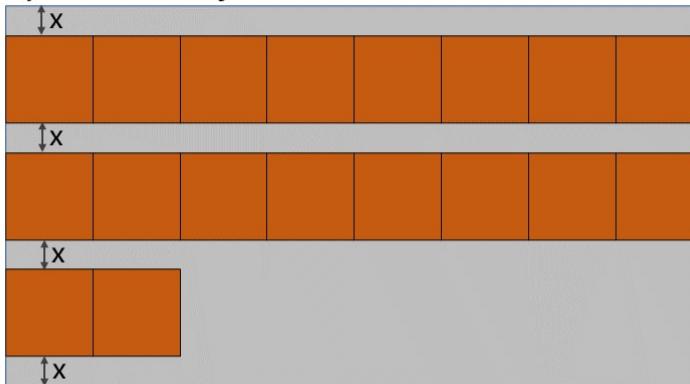


space-between

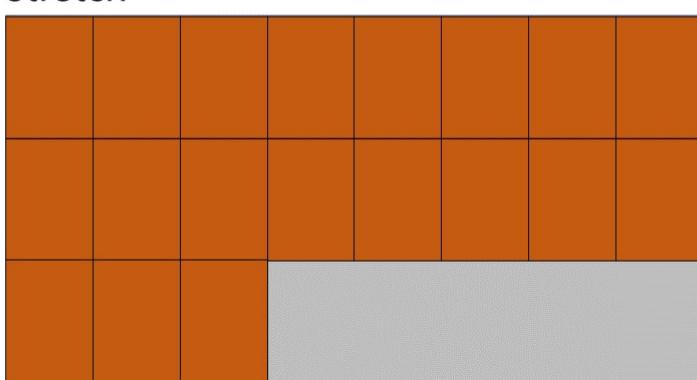




space-around



space-evenly



9.flex 实现水平垂直居中

方法一：父容器开启 `flex` 布局，随后使用 `justify-content` 和 `align-items` 实现水平垂直居中

```
.outer {
  width: 400px;
  height: 400px;
  background-color: #888;
  display: flex;
  justify-content: center;
  align-items: center;
}

.inner {
  width: 100px;
  height: 100px;
  background-color: orange;
}
```

方法二：父容器开启 `flex` 布局，随后子元素 `margin: auto`

```
.outer {  
    width: 400px;  
    height: 400px;  
    background-color: #888;  
    display: flex;  
}  
.inner {  
    width: 100px;  
    height: 100px;  
    background-color: orange;  
    margin: auto;  
}
```

10. 伸缩性

1. flex-basis

- 概念：`flex-basis` 设置的是主轴方向的**基准长度**，会让宽度或高度失效。

备注：主轴横向：宽度失效；主轴纵向：高度失效

- 作用：浏览器根据这个属性设置的值，计算主轴上是否有多余空间，默认值 `auto`，即：伸缩项目的宽或高。

2. flex-grow (伸)

- 概念：`flex-grow` 定义伸缩项目的放大比例，默认为 `0`，即：纵使主轴存在剩余空间，也不拉伸（放大）。
- 规则：
 - 若所有伸缩项目的 `flex-grow` 值都为 `1`，则：它们将等分剩余空间（如果有空间的话）。
 - 若三个伸缩项目的 `flex-grow` 值分别为：`1`、`2`、`3`，则：分别瓜分到：`1/6`、`2/6`、`3/6` 的空间。

3. flex-shrink (缩)

- 概念：`flex-shrink` 定义了项目的压缩比例，默认为 `1`，即：如果空间不足，该项目将会缩小。
- 收缩项目的计算，略微复杂一点，我们拿一个场景举例：

例如：

三个收缩项目，宽度分别为：`200px`、`300px`、`200px`，它们的 `flex-shrink` 值分别为：`1`、`2`、`3`

若想刚好容纳下三个项目，需要总宽度为 `700px`，但目前容器只有 `400px`，还差 `300px`
所以每个人都要收缩一下才可以放下，具体收缩的值，这样计算：

- 计算分母： $(200 \times 1) + (300 \times 2) + (200 \times 3) = 1400$
- 计算比例：
 - 项目一： $(200 \times 1) / 1400 = \text{比例值1}$
 - 项目二： $(300 \times 2) / 1400 = \text{比例值2}$

- 项目三: $(200 \times 3) / 1400 =$ 比例值3
3. 计算最终收缩大小:

- 项目一需要收缩: 比例值1 \times 300
- 项目二需要收缩: 比例值2 \times 300
- 项目三需要收缩: 比例值3 \times 300

11. flex复合属性

`flex` 是复合属性, 复合了: `flex-grow`、`flex-shrink`、`flex-basis` 三个属性, 默认值为 `0 1 auto`。

- 如果写 `flex:1 1 auto`, 则可简写为: `flex:auto`
- 如果写 `flex:1 1 0`, 则可简写为: `flex:1`
- 如果写 `flex:0 0 auto`, 则可简写为: `flex:none`
- 如果写 `flex:0 1 auto`, 则可简写为: `flex:0 auto` —— 即 `flex` 初始值。

12. 项目排序

- `order` 属性定义项目的排列顺序。数值越小, 排列越靠前, 默认为 `0`。

13. 单独对齐

- 通过 `align-self` 属性, 可以单独调整某个伸缩项目的对齐方式
- 默认值为 `auto`, 表示继承父元素的 `align-items` 属性。

16. 响应式布局

媒体查询

1.1 媒体类型

值	含义
<code>all</code>	检测所有设备。
<code>screen</code>	检测电子屏幕, 包括: 电脑屏幕、平板屏幕、手机屏幕等。
<code>print</code>	检测打印机。
<code>aural</code>	已废弃, 用于语音和声音合成器。
<code>braille</code>	已废弃, 应用于盲文触摸式反馈设备。
<code>embossed</code>	已废弃, 用于打印的盲人印刷设备。
<code>handheld</code>	已废弃, 用于掌上设备或更小的装置, 如PDA和小型电话。
<code>projection</code>	已废弃, 用于投影设备。
<code>tty</code>	已废弃, 用于固定的字符网格, 如电报、终端设备和对字符有限制的便携设备。
<code>tv</code>	已废弃, 用于电视和网络电视。

完整列表请参考：<https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media>

1.2 媒体特性

值	含义
<code>width</code>	检测视口 宽度 。
<code>max-width</code>	检测视口 最大宽度 。
<code>min-width</code>	检测视口 最小宽度 。
<code>height</code>	检测视口 高度 。
<code>max-height</code>	检测视口 最大高度 。
<code>min-height</code>	检测视口 最小高度 。
<code>device-width</code>	检测设备 屏幕的宽度 。
<code>max-device-width</code>	检测设备 屏幕的最大宽度 。
<code>min-device-width</code>	检测设备 屏幕的最小宽度 。
<code>orientation</code>	检测 视口的旋转方向 （是否横屏）。 1. <code>portrait</code> ：视口处于纵向，即高度大于等于宽度。 2. <code>landscape</code> ：视口处于横向，即宽度大于高度。

完整列表请参考：<https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media>

1.3 运算符

值	含义
<code>and</code>	并且
<code>, 或 or</code>	或
<code>not</code>	否定
<code>only</code>	肯定

1.4 常用阈值

在实际开发中，会将屏幕划分成几个区间，例如：



1.5 结合外部样式的用法

用法一：

```
<link rel="stylesheet" media="具体的媒体查询" href="mystylesheet.css">
```

用法二：

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

@media screen and (min-width:768px) and (max-width:1200px) {
    /*CSS-Code*/
}
```

17. BFC

1. 什么是BFC

- W3C 上对 BFC 的定义：

原文：Floats, absolutely positioned elements, block containers (such as inline-blocks, table-cells, and table-captions) that are not block boxes, and block boxes with 'overflow' other than 'visible' (except when that value has been propagated to the viewport) establish new block formatting contexts for their contents.

译文：浮动、绝对定位元素、不是块盒子的块容器（如 `inline-blocks`、`table-cells` 和 `table-captions`），以及 `overflow` 属性的值除 `visible` 以外的块盒，将为其内容建立新的块格式化上下文。

- MDN 上对 BFC 的描述：

块格式化上下文 (Block Formatting Context, BFC) 是 Web 页面的可视 CSS 渲染的一部分，是块盒子的布局过程发生的区域，也是浮动元素与其他元素交互的区域。

- 更加通俗的描述：

1. BFC 是 Block Formatting Context (块级格式上下文)，可以理解成元素的一个“特异功能”。

2. 该“**特异功能**”，在默认的情况下处于关闭状态；当元素满足了某些条件后，该“**特异功能**”被激活。
3. 所谓激活“**特异功能**”，专业点说就是：该元素创建了 **BFC**（又称：开启了 **BFC**）。

2. 开启了BFC能解决什么问题

1. 元素开启 **BFC** 后，其子元素不会再产生 **margin** 塌陷问题。
2. 元素开启 **BFC** 后，自己不会被其他浮动元素所覆盖。
3. 元素开启 **BFC** 后，就算其子元素浮动，元素自身高度也不会塌陷。

3. 如何开启BFC

- 根元素
- 浮动元素
- 绝对定位、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` 的值不为 `visible` 的块元素
- 伸缩项目
- 多列容器
- `column-span` 为 `all` 的元素（即使该元素没有包裹在多列容器中）
- `display` 的值，设置为 `flow-root`