河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《互联网医疗服务开发》课程

第02章：HTML

冯顺磊

河南中医药大学信息技术学院（智能医疗行业学院）
河南中医药大学信息技术学院智能医疗教研室
https://aitcm.hactcm.edu.cn
2025/9/26

本章概要

- 基础
- 属性
  - 属性结构
  - 公共属性
  - 全局属性
- 元素
  - 结构元素：构建页面骨架
  - 文本与列表元素：填充内容基础
  - 图像与多媒体元素：丰富页面内容
  - 表格元素：架构化数据展示
  - 表单元素：实现用户交互

1. 基础

1.1 文档结构

- HTML 是构建网页的核心语言，负责定义文档结构与内容元素。浏览器访问网站时，本质是从服务器获取 HTML 代码，并将其渲染为可视化页面。
- 无论内容多么复杂，最终均由这一基础框架扩展而来。符合语法规范的网页需遵循特定的基础结构。
- 无论网页内容多么复杂，均由该基础框架拓展而来。值得注意的是，HTML 代码中的缩进、换行等格式，不会影响浏览器对页面的解析与渲染。理论上，所有元素均可压缩为单行代码，其呈现效果与格式化后的代码并无差异。然而，开发者应采用分层缩进的书写方式，以提升源码的可读性与可维护性。

```
1. <!-- HTML5文档类型声明 -->
2. <!DOCTYPE html>
3. <!-- 根元素，设置语言为中文 -->
4. <html lang="zh-CN">

5. <!-- 文档头部，包含元数据和页面信息 -->
6. <head>
7.     <!-- 字符编码设置为UTF-8，支持中文等字符 -->
8.     <meta charset="UTF-8">
9.     <!-- 响应式设计：设置视口宽度为设备宽度，初始缩放比例为1.0 -->
10.    <meta name="viewport" content="width=device-width, initial-scale=1.0">
11.    <!-- 页面标题，显示在浏览器标签页上 -->
12.    <title>页面标题</title>
13. </head>
14.
15. <!-- 文档主体，包含页面可见内容 -->
16. <body>
17.    <!-- 段落元素，显示"Hello World"文本 -->
18.    <p>Hello World</p>
19. </body>
20.
21. </html>
```

HTML 元素在大小写方面并无区分。这意味着，在输入元素时，既能够使用大写字母，也能够使用小写字母。例如，元素 <title> 可以写成 <title>、<TITLE>、<Title>、<TiTlE> 等形式，且均能正常发挥作用。然而，从保持一致性以及增强可读性的角度考量，建议仅使用小写字母。

1. 基础

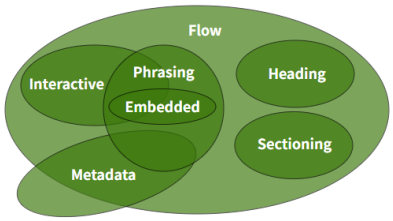
1.2 元素结构

- HTML元素由三个核心组成部分构成：开始标签、内容元素和结束标签，具体结构如下。
  - **开始标签（Opening tag）**：由元素名称（本例为p）及左右尖括号构成，标记元素的起始位置。该标签表示元素作用范围的开始，在段落文本中位于内容起始处。
  - **内容元素（Content）**：元素承载的主体信息，此处即段落文本内容，位于开始标签与结束标签之间。
  - **结束标签（Closing tag）**：在元素名称前添加斜杠（/）进行标识，与开始标签形成对应关系。该标签缺失会造成元素作用范围不明确，是初学者易犯的典型错误，可能导致页面渲染异常。

1. 基础

1.3 元素分类

- 内容分类
  - HTML元素通常被归类于一个或多个内容类别，这些类别依据元素的共性特征进行逻辑分组。需要明确的是，这种分类体系属于非严格性分组（不构成严格的层级关系），其主要作用是为理解和应用这些元素的共性特征及关联规则提供框架，尤其在处理复杂文档结构时具有指导意义。值得注意的是，确实存在部分元素未被纳入现有分类体系的情况。



1. 基础

1.3 元素分类

内容类型	描述	
元数据内容 (Metadata content)	元数据内容（Metadata content）元素主要用于控制文档的呈现方式与交互行为、建立跨文档资源关联以及传递文档的附属信息。 该类别包含以下核心元素：<base>、<link>、<meta>、<noscript>、<script>、<style> 和 <title>。	
流式内容 (Flow content)	流式内容（Flow content）是HTML标准中的宽泛分类，涵盖元素内绝大多数允许包含的元素类型，具体包括标题、分段、短语、嵌入、交互及表单相关元素，同时涵盖、、等嵌入元素，以及、等交互控件。该类别还包含文本节点（不包含仅含空白字符的节点）。 流式元素包括：<a>、<abbr>、<address>、<article>、<aside>、<audio>、<b>、<bdo>、<bdi>、<blockquote>、 、<button>、<canvas>、<cite>、<code>、<data>、<datalist>、<del>、<details>、<dfn>、<dialog>、 、<dl>、、<embed>、<fieldset>、<figure>、<footer>、<form>、<h1>、<h2>、<h3>、<h4>、<h5>、<h6>、<header>、<hgroup>、<hr>、<i>、<iframe>、<img>、<input>、<ins>、<kbd>、<label>、<main>、<map>、<mark>、<math>、<menu>、<meter>、<nav>、<noscript>、<object>、 、<output>、<p>、<picture>、<pre>、<progress>、<q>、<ruby>、<s>、<samp>、<search>、<script>、<section>、<select>、<slot>、<small>、、、<sub>、<sup>、<svg>、<table>、<template>、<textarea>、<time>、<u>、 、<var>、<video>、<wbr>、纯文本。	
	分段内容 （Sectioning content）	分段内容（Sectioning content）作为流式内容的子集，能够在文档大纲中创建独立的分段结构，其核心功能是界定 <header> 元素、<footer> 元素以及标题层级关系的语义范围。 该类别包含以下核心语义元素：<article>、<aside>、<nav> 和 <section>。
	标题内容 （Heading content）	标题内容（Heading content）作为流式内容的子集，用于定义段落标题，其对应段落既可通过显式分段内容元素标记，也可由标题自身隐式定义。 涵盖元素包括：<h1>-<h6>及<hgroup>。

1. 基础

1.3 元素分类

内容类型		描述
流式内容 (Flow content)	短语内容 (Phrasing content)	短语内容（Phrasing content）作为流式内容的子集，专门用于定义文档中文本元素及其标记结构。这类内容通过其有序排列形成段落的基本结构，既规范了文本的语义表达，又确保了文档的逻辑连贯性。 属于此类的元素有：<abbr>、<audio>、<b>、<bdi>、<bdo>、<button>、<canvas>、<cite>、<code>、<data>、<datalist>、<dfn>、、<embed>、<i>、<iframe>、<img>、<input>、<kbd>、<label>、<mark>、<math>、<meter>、<noscript>、<object>、<output>、<picture>、<progress>、<q>、<ruby>、<s>、<samp>、<script>、<select>、<slot>、<small>、、、<sub>、<sup>、<svg>、<template>、<textarea>、<time>、<u>、<var>、<video>、<wbr>、纯文本。
	嵌入内容 (Embedded content)	嵌入内容（Embedded content）作为流式内容的子集，其核心功能在于通过引入外部资源或将异源内容插入文档，实现跨资源整合。具体表现为：将其他格式的标记语言或命名空间内容无缝集成至当前文档架构中。 属于此类的元素有：<audio>、<canvas>、<embed>、<iframe>、<img>、<math>、<object>、<picture>、<svg>、<video>。
	交互内容 (Interactive content)	交互内容（Interactive content）作为流式内容的子集，包含为用户交互而特别设计的元素。 属于此类的元素有：<button>、<details>、<embed>、<iframe>、<label>、<select>、<textarea>。

1. 基础

1.3 元素分类

内容类型		描述
流式内容 (Flow content)	表单相关内容 (Form-associated content)	表单相关内容（Form-associated content）作为流式内容的子集，特指通过form属性显式关联表单所有者的元素集合，可在预期出现流式内容的位置合法使用。表单所有者既可以是直接包含这些元素的<form>标签，也可以是元素通过form属性指定其id值的关联表单。 属于此类的元素有：<button>、<fieldset>、<input>、<label>、<meter>、<object>、<output>、<progress>、<select>、<textarea> 此类包含了几个子类： • 可列举的元素（listed） 在 form.elements 和 fieldset.elements 集合中列举出的元素。包括 <button>、<fieldset>、<input>、<object>、<output>、<select> 和 <textarea>。 • 可标记的元素（labelable） 可以与 <label> 相关联的元素。包括 <button>、<input>、<meter>、<output>、<progress>、<select> 和 <textarea>。 • 可提交的元素（submittable） 包括当表单提交时可以用来组成表单数据的元素。包括 <button>、<input>、<object>、<select> 和 <textarea>。 • 可重置的元素（resettable） 当表单重置时会被重置的元素。包括 <input>、<output>、<select> 和 <textarea>。

1. 基础

1.3 元素分类

- 功能分类
  - 主根元素

元素名称	描述
<html>	作为 HTML 文档的根元素（顶级元素），因此也被称为根节点。文档中所有其他元素必须作为其子元素存在，且严格遵循嵌套层级关系。

  - 元数据元素
    - 元数据（Metadata）包含网页的关键描述信息，涵盖样式规范、脚本逻辑及结构化数据，可辅助相关软件（例如搜索引擎、浏览器等）高效解析与渲染页面内容。针对样式和脚本类元数据，开发者可采用两种实现方式：以内联方式直接嵌入网页文档，或通过外部文件链接引用相关资源。

元素	描述
<base>	指定用于一个文档中包含的所有相对 URL 的根 URL。一份中只能有一个该元素。
<head>	包含文档相关的配置信息（元数据），包括文档的标题、脚本和样式表等。
<link>	指定当前文档与外部资源的关系。该元素最常用于链接 CSS，此外也可以被用来创建站点图标（比如“favicon”样式图标和移动设备上用以显示在主屏幕的图标）。
<meta>	表示那些不能由其它 HTML 元相关（meta-related）元素表示的元数据信息。如：<base>、<link>、<script>、<style> 或 <title>。
<style>	包含文档或者文档部分内容的样式信息，它们会被应用于包含此元素的文档。
<title>	定义文档的标题，显示在浏览器的标题栏或标签页上。它只应该包含文本，若是包含有标签，则它包含的任何标签都将被忽略。

1. 基础

1.3 元素分类

  - 分区根元素

元素名称	描述
<body>	表示文档的内容。文档中只能有一个该元素。

  - 内容分区元素
    - 内容分区元素允许你将文档内容从逻辑上进行组织划分。使用包括页眉（header）、页脚（footer）、导航（nav）和标题（h1~h6）等分区元素，来为页面内容创建明确的大纲，以便区分各个章节的内容。

元素	描述
<address>	表示其中的 HTML 提供了某个人、某些人或某个组织（等等）的联系信息。
<article>	表示文档、页面、应用或网站中的独立结构，旨在成为可独立分配的或可复用的结构，如在发布中，它可能是论坛帖子、杂志或新闻文章博客、用户提交的评论、交互式组件，或者其它独立的内容项目。
<aside>	表示文档的一部分，其内容仅与文档的主要内容间接相关。其通常以侧边栏或标注框（call-out box）的形式出现。
<footer>	表示最近的一个父分段内容或分段的根元素的页脚。<footer> 通常包含该章节作者、版权数据或者与文档相关的链接等信息。
<header>	表示介绍性内容，通常包含一组介绍性的或是辅助导航的实用元素。它可能包含一些标题元素，但也可能包含其它元素，比如 Logo、搜索框、作者名称和其它元素。
<h1>、<h2>、<h3>、<h4>、<h5>、<h6>	表示六个不同的级别的章节标题，<h1> 级别最高，而 <h6> 级别最低。
<main>	呈现了文档正文的主体部分。主体部分由与文档直接相关，或者扩展于文档的中心主题、应用的主要功能部分的内容组成。
<nav>	表示页面的一部分，其目的是在当前文档或其它文档中提供导航链接。导航部分的常见示例是菜单、目录和索引。
<section>	表示 HTML 文档中一个通用独立章节，它没有更具体的语义元素来表示。一般来说会包含一个标题。

1. 基础

1.3 元素分类

- 文本内容元素
  - 使用 HTML 文本内容元素来组织在开标签 <body> 和闭标签 </body> 里的块或章节的内容。这些元素能标识内容的宗旨或结构，而这对于无障碍和搜索引擎优化很重要。

元素	描述
<blockquote>	代表其中的文字是引用内容。通常在渲染时，这部分的内容会有一定的缩进。引文来源的 URL 地址可以使用属性 cite 给出，而来源的文本可以使用 <cite> 元素给出。
<dd>	用来指明一个描述列表（<dl>）元素中先前术语（<dt>）的描述、定义或值。
	一个通用型的流式内容容器，在不使用 CSS 的情况下，其对内容或布局没有任何影响，直到通过某种方式设置样式（例如，将样式直接应用于该元素，或将弹性盒子等布局模型应用于其父元素）。
<dl>	一个包含一组术语（使用 <dt> 元素指定）以及描述（由 <dd> 元素提供）的列表。通常用于展示词汇表或者元数据（键—值对列表）。
<dt>	在描述或定义列表中声明一个术语。该元素仅能作为 <dl> 的子元素出现。通常在该元素后面会跟着一个 <dd> 元素；但多个连续出现的 <dt> 元素也将由出现在它们后面的第一个 <dd> 元素定义。
<figcaption>	描述其父元素 <figure> 里其它内容的标题或图例。
<figure>	表示一段独立的内容，可能包含 <figcaption> 元素定义的标题。该插图、标题和其中的内容通常作为一个独立的引用单元。
<hr>	表示段落级元素之间的主题转换：例如，一个故事中的场景的改变，或一个章节的主题的改变。
	表示列表里的条目。它必须包含在一个父元素里：有序列表（ ）、无序列表（ ）或者菜单（<menu>）。在菜单或者无序列表里，列表条目通常用点排列显示；在有序列表里，列表条目通常在左边显示按升序排列的计数，例如数字或者字母。
<menu>	的语意替换，但被浏览器视为（并向无障碍树暴露为）与 没有区别。它表示了条目的无序列表（使用 表示）。
	表示有序列表，通常渲染为一个带编号的列表。
<p>	表示文本的一个段落。该元素通常表现为通过空行和/或首行缩进与相邻块分隔的文本块。但 HTML 段落可以与任何相关内容（例如，图像或表单字段）构成结构分组。
<pre>	表示预定义格式文本。在该元素中的文本通常按照 HTML 文件中的编排，以非比例或等宽字体的形式展现出来，文本中的空白符都会显示出来。
	表示一系列无序的列表项目，通常渲染为项目符号列表。

1. 基础

1.3 元素分类

- 内联文本语义元素
  - 使用 HTML 内联文本语义定义单词、内容、或任意文字的语义、结构或样式。

元素	描述
<a>	可以通过它的 href 属性创建通向其它网页、文件、电子邮件地址、同一页面内的位置或任何其它 URL 的超链接。
<abbr>	用于代表缩写。
<b>	用于吸引读者的注意到该元素的内容上（如果没有另加特别强调）。这个元素过去被认为是粗体（Boldface）元素，并且大多数浏览器仍然将文字显示为粗体。尽管如此，你不应将 <b> 元素用于显示粗体文字；替代方案是使用 CSS font-weight 属性来创建粗体文字。
<bdi>	告诉浏览器的双向算法将其包含的文本与周围的文本隔离，当网站动态插入一些文本且不知道所插入文本的方向性时，此功能特别有用。
<bdo>	覆盖文本的方向性，使文本以不同的方向渲染呈现出来。
 	在文本中生成一个换行（回车）符号。此元素在写诗和地址时很有用，这些地方的换行都非常重要。
<cite>	用于包含引用作品的标题。这个引用可能是一个根据适当的上下文约定关联引用的元数据的缩写。
<code>	以一种旨在表明其中的内容是计算机代码片段的方式显示其内容。默认情况下，它以用户代理的默认等宽字体显示。
<data>	将指定内容和机器可读的翻译联系在一起。但是，如果内容是与时间或者日期相关的，则一定要使用 time 元素。
<dfn>	用于表示在定义短语或句子的上下文中定义的术语。父级 <p> 元素、<dt>/<dd> 对，或与 <dfn> 元素最近的分区元素被认定为是术语的定义。
	标记出需要用户着重阅读的内容， 元素是可以嵌套的，嵌套层次越深，则其包含的内容被认定为越需要着重阅读。
<i>	用于表现因某些原因需要区分普通文本的一系列文本。例如惯用文本、技术术语、分类名称等。它通常使用斜体表示，这也是该元素（<i>）命名的来源。
<kbd>	表示一段内联文本，文本来自键盘、语音输入或其他文本输入设备的用户输入。按照惯例，用户代理默认使用其默认的等宽字体显示 <kbd> 元素的内容，尽管这不是 HTML 标准强制要求的。
<mark>	表示为引用或符号目的而标记或突出显示的文本，这是由于标记的段落在封闭上下文中的相关性或重要性造成的。

1. 基础

- 内联文本语义元素

元素	描述
<q>	表示一个封闭的并且是短的行内引用的文本。大多数现代浏览器通过将文本包裹在引号内来实现这一点。此元素适用于不需要分段的短文本；对于长的文本的引用请使用 <blockquote> 元素。
<rp>	用于为那些不能使用 <ruby> 元素展示 ruby 注解的浏览器，提供回退的圆括号。一个 <rp> 元素应该包裹一个左括号或右括号，这些括号将包含注解文本的 <rt> 元素包裹起来。
<rt>	指定 ruby 注解的 ruby 文本组件，用于描述东亚字符的发音、翻译或音译信息。该元素始终在 <ruby> 元素中使用。
<ruby>	用来在基础文本上方、下方或一旁展现小注解，通常用于显示东亚字符的发音。它还用于注解其他类型的文本，但这种用法不太常见。
<s>	使用删除线来渲染文本。使用 <s> 元素来表示不再相关或者不再准确的事情。但是当表示文档编辑时，不建议使用 <s>；为此，请酌情使用 del 和 ins 元素。
<samp>	用于标识计算机程序输出，通常使用浏览器缺省的等宽字体来渲染（例如 Courier 或 Lucida Console）。
<small>	代表旁注和小字体，如版权和法律等独立于其样式展示的文本。默认情况下，它将其中的文本使用小一号的字体渲染，例如从 small 到 x-small。
	短语内容的通用行内容器，并没有任何特殊语义。可以使用它来编组元素以达到某种样式意图（通过使用 class 或者 id 属性），或者这些元素有着共同的属性，比如 lang。应该在没有其他合适的语义元素时才使用它。 与 div 元素很相似，但 是一个块级元素而 则是行级元素。
	表示其内容十分重要、严肃或紧迫。浏览器通常用粗体显示。
<sub>	定义因排版原因而应显示为下标的内联文本。下标通常显示得更小且更低。
<sup>	定义因排版原因而应显示为上标的内联文本。上标通常显示得更小且更高。
<time>	表示特定的时间段。可能包括 datetime 属性，以将日期转换为机器可读的格式，从而获得更好的搜索引擎结果或自定义功能（如，提醒）。
<u>	表示一个需要标注为非文本化（non-textual）的内联文本域。默认情况下渲染为一个实线下划线，但可以用 CSS 替换。
<var>	表示数学表达式或编程上下文中的变量名称。尽管该行为取决于浏览器，但通常使用当前字体的斜体形式显示。
<wbr>	一个文本中的位置，其中浏览器可以选择来换行，虽然它的换行规则可能不会在这里换行。

1. 基础

- 内嵌内容
  - 除了常规的多媒体内容，HTML 可以包括各种其他的内容，即使它并不容易交互。

元素	描述
<embed>	将外部内容嵌入文档中的指定位置。此内容由外部应用程序或其他交互式内容源（如浏览器插件）提供。
<iframe>	表示嵌套的浏览上下文。它能够将另一个 HTML 页面嵌入到当前页面中。
<object>	表示引入一个外部资源，这个资源可能是一张图片、嵌入的浏览上下文，亦或是一个插件所使用的资源。
<picture>	通过包含零或多个 <source> 元素和一个 <img> 元素来为不同的显示/设备场景提供图像版本。
<portal>	允许将另一个 HTML 页面嵌入到当前页面中，以便更流畅地导航到新页面。
<source>	为 picture、audio 或 video 元素指定多个媒体资源。这是一个空元素，这意味着它没有内容，也没有封闭标签。它通常用于以多种格式提供相同的媒体内容，以提供不同浏览器的兼容性，因为浏览器对图像文件和媒体文件格式的支持不同。

1. 基础

1.3 元素分类

- SVG 和 MathML
  - 可通过使用 <svg> 或 <math> 标签，将 SVG 或 MathML 内容直接嵌入 HTML 文档。

元素	描述
<svg>	定义新坐标系和视口的容器。它被用作 SVG 文档的最外层元素，但也可用于在 SVG 或 HTML 文档中嵌入 SVG 片段。
<math>	顶级的 MathML 元素。每一个有效的 MathML 实例都必须封装在其中。此外，不能在另一个此类元素中嵌套第二个 <math> 元素，但可以在其中包含任意个其他的子元素。

- 脚本
  - 为了创建动态内容和 Web 应用程序，HTML 支持使用脚本语言，最突出的就是 JavaScript。有一些特定的元素用于支持此功能。

元素	描述
<canvas>	用来通过 canvas scripting API 或 WebGL API 绘制图形及图形动画的容器元素。
<noscript>	定义脚本未被执行时（页面的脚本类型不受支持，或当前浏览器关闭了脚本）的替代内容。
<script>	用于嵌入可执行脚本或数据。这通常用作嵌入或者引用 JavaScript 代码。<script> 元素也能在其他语言中使用，比如 WebGL 的 GLSL 着色器语言和 JSON。

1. 基础

1.3 元素分类

- 编辑标识
  - 这些元素能标示出某个文本被更改过的部分。

元素	描述
<del>	表示一些被从文档中删除的文本内容。比如可以在需要显示修改记录或者源代码差异的情况使用这个标签。<ins> 标签的作用恰恰与此相反：表示文档中添加的文本。
<ins>	表示一些添加到文档中的文本内容。你可以使用 <del> 元素来类似地表示已从文档中删除的文本。

1. 基础

1.3 元素分类

- 表单
  - HTML 提供了众多可协同运用的元素，这些元素能够用于创建表单，供用户填写并提交至网站或应用程序。

元素	描述
<button>	一个可交互元素（可通过用户的鼠标、键盘、手指、声音指令或其他辅助技术激活）。一旦被激活，它就会执行一个动作，例如提交表单或打开对话框。
<datalist>	包含了一组 <option> 元素，这些元素表示其它表单控件可选值。
<fieldset>	用于对 web 表单中的控件和标签（<label>）进行分组。
<form>	表示文档中的一个区域，此区域包含交互控件，用于向 Web 服务器提交信息。
<input>	用于为基于 Web 的表单创建交互式控件，以便接受来自用户的数据。取决于设备和用户代理的不同，表单可以使用各种类型的输入数据和控件。<input> 元素是目前是 HTML 中最强大、最复杂的元素之一，因为它有大量的输入类型和属性组合。
<label>	表示用户界面中某个元素的说明。
<legend>	用于表示其父元素 <fieldset> 的内容标题。
<meter>	用来显示已知范围的标量值或者分数值。
<optgroup>	为 <select> 元素中的选项创建分组。
<option>	用于定义在 select、<optgroup> 或 <datalist> 元素中包含的选项。<option> 可以在弹出窗口和 HTML 文档中的其他项目列表中表示菜单项。
<output>	网站或应用程序可以将计算或用户操作的结果注入其中的容器元素。
<progress>	用来显示一项任务的完成进度，通常情况下该元素显示为一个进度条。
<select>	表示一个提供选项菜单的控件。
<textarea>	表示一个多行纯文本编辑控件，当你希望用户输入一段相当长的、不限格式的文本，例如评论或反馈表单中的一段意见时，这很有用。

1. 基础

1.3 元素分类

- 交互元素
  - HTML 提供了一系列有助于构建交互式用户界面对象的元素。

元素	描述
<details>	创建一个挂件，仅在被切换成“展开”状态时，它才会显示内含的信息。必须使用 <summary> 元素为该部件提供概要或者标签。
<dialog>	表示一个对话框或其他交互式组件，例如一个可关闭警告、检查器或者窗口。
<summary>	用作 details 元素内容的摘要、标题或图例。点击 <summary> 元素会翻转父元素 <details> 的展开和关闭状态。

- Web组件
  - HTML 提供了一系列有助于构建交互式用户界面对象的元素。

元素	描述
<slot>	作为 Web 组件技术套件的一部分，该元素是 web 组件内的占位符，你可以使用你自己的标记填充它，从而让你创建单独的 DOM 树并一起呈现它们。
<template>	一种保存 HTML 的机制，它不会在加载页面时立即渲染，但随后可以在运行时使用 JavaScript 实例化。

1. 基础

- HTML元素按显示特性可分为块级元素与内联元素两大核心类别。
  - 块级元素以独立区块形式呈现：在文档流中始终独占父容器整行空间，其前后内容均会自动换行显示。这类元素主要承担页面框架构建功能，典型应用包括标题(<h1>)、段落(<p>)、列表(<ul>)及结构化区域(<header>/<footer>)。遵循嵌套规则时，块级元素不可作为内联元素的子级，但允许嵌套其他块级元素构成复合结构。
  - 内联元素以文本级标签形式存在，用于修饰块级元素中的局部内容，其显示特征不会引发文本换行。典型应用场景包括：<a>创建交互式链接，与实现语义化强调。这类元素通过包裹文字片段或媒体资源，在不破坏段落连续性的前提下实现精细化排版控制。

- 块级元素的特点：
  - 块级元素始终独占一行，后续元素自动换行显示
  - 支持设置高度、宽度、行高及上下边距等盒模型属性
  - 默认宽度自动继承父容器的100%宽度
- 内联元素的特点：
  - 与相邻元素共处同一行，不强制换行
  - 无法设置高度、宽度及垂直方向边距（上下边距无效）
  - 宽度由内容决定，不可显式定义
  - 元素间存在空白符（空格、换行、制表符）时会产生渲染间隙

1. 基础

1.4 块级元素和内联元素

```
<!-- 内联元素演示 -->
<div>
  <h3>内联元素演示（em标签）：</h3>
  <p><strong>特点：</strong>多个内联元素会在同一行显示，不会换行</p>

  <!-- 内联元素：em标签在同一行显示 -->
  <div>
    <em>第一</em><em>第二</em><em>第三</em>
  </div>

  <p><strong>观察：</strong>三个em元素在同一行显示，没有换行</p>
</div>

<!-- 块级元素演示 -->
<div>
  <h3>块级元素演示（p标签）：</h3>
  <p><strong>特点：</strong>每个块级元素独占一行，会自动换行</p>

  <!-- 块级元素：p标签独占一行 -->
  <p>第四</p>
  <p>第五</p>
  <p>第六</p>

  <p><strong>观察：</strong>三个p元素各自独占一行，自动换行</p>
</div>
```

1. 基础

1.5 空白、特殊字符

- 在HTML元素内容内部，若存在连续空白字符（包括空格、制表符、换行符等），无论其数量与排列形式如何，在网页渲染过程中，HTML解析器会将连续空白字符合并为单个空格进行显示。

- 网页中插入多个空格的方法
  - ：表示不换行空格（non-breaking space），单个实体对应一个半角空格，重复使用该实体可插入多个空格（常用方式）；
  - ：表示半角空格（en space），单个实体等同字母"n"宽度，重复使用可形成连续空格；
  - ：表示全角空格（em space），单个实体等同字母"m"宽度，重复使用可获得更大间距；
  - ：表示窄空格（thin space），约为1/6em宽度的细小间距；
- 所有HTML实体结尾的分号必须保留，该符号是实体代码的必要组成部分；

1. 基础

1.5 空白、特殊字符

```
<!-- 普通空格演示 -->
<div>
  <h3>1. 普通空格演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code><div>狗 狗 很 呆 萌。</div></pre></code>

  <p><strong>显示效果：</strong></p>
  <!-- 普通空格：HTML会忽略多个连续空格，只显示一个空格 -->
  <p>狗 狗 很 呆 萌。</p>

  <p><strong>说明：</strong>每个单词之间只有一个空格，HTML会保持这种格式。</p>
</div>

<!-- 多个空格和换行演示 -->
<div>
  <h3>2. 多个空格和换行演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code><div>狗 狗        很
  呆 萌。</div></pre></code>

  <p><strong>显示效果：</strong></p>
  <!-- 多个空格和换行：HTML会忽略多个连续空格和换行，只显示一个空格 -->
  <p>狗 狗        很
  呆 萌。</p>

  <p><strong>说明：</strong>HTML会忽略多个连续空格和换行符，只显示一个空格。</p>
</div>

<!--  实体演示 -->
<div>
  <h3>3. &amp;nbsp;实体演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code><div>狗&amp;nbsp;狗&amp;nbsp;很&amp;nbsp;呆&amp;nbsp;萌。</div></pre></code>

  <p><strong>显示效果：</strong></p>
  <!-- 使用 实体：每个 都会显示为一个空格，不会被忽略 -->
  <p>狗 狗 很 呆 萌。</p>

  <p><strong>说明：</strong>每个 都会显示为一个空格，不会被HTML合并。</p>
</div>
```

1. 基础

1.5 块级元素和内联元素

- 在 HTML 中，字符 <、>、"、' 和 & 是特殊字符，它们是 HTML 语法自身的一部分，在展示这些字符时必须使用字符引用——表示字符的特殊编码，每个字符引用以符号 & 开始，以分号（;）结束，在HTML中常用的特殊字符如表所示。

字符	名称	实体名称	说明与使用场景
	不断行的空格		在文本中强制插入空格，不会被浏览器合并。
<	小于号	<	必须使用：在页面上显示代码中的 <，避免被解析为标签开始。
>	大于号	>	必须使用：在页面上显示代码中的 >，避免被解析为标签结束。
&	和号	&	必须使用：用于显示 & 本身，避免被解析为实体字符的开始。
“	双引号	"	在标签属性值内显示引号，如 alt="描述"。
©	版权符号	©	用于声明版权所有权，如 © 2023 公司名。
®	注册商标	®	表示已正式注册的商标。
™	商标符号	™	表示商标（注册或未注册均可使用）。
€	欧元符号	€	表示欧元货币单位。
£	英镑符号	£	表示英镑货币单位。
•	圆点列表符	•	常用于自定义列表项前的圆点符号。
—	破折号	—	表示语气或思想的转折，比连字符“-”更长。

1. 基础

1.5 空白、特殊字符

```
<!-- 问题演示 -->
<div>
  <h3>1. 问题演示：直接使用特殊字符</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;p&gt;HTML 中用 &lt;p&gt; 来定义段落元素。&lt;/p&gt;</code></pre>

  <p><strong>显示效果：</strong></p>
  <!-- 错误示例：直接使用小于号和大于号 -->
  <div>
      <p>HTML 中用 <p> 来定义段落元素。</p>
  </div>

  <p><strong>问题：</strong>浏览器将 &lt;p&gt; 解析为HTML标签，而不是文本内容。</p>
</div>

<!-- 解决方案演示 -->
<div>
  <h3>2. 解决方案：使用字符实体</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;p&gt;HTML 中用 &amp;lt;p&amp;gt; 来定义段落元素&lt;/p&gt;</code></pre>

  <p><strong>显示效果：</strong></p>
  <!-- 正确示例：使用字符实体 -->
  <p>HTML 中用 &lt;p&gt; 来定义段落元素</p>

  <p><strong>解决：</strong>使用 &amp;lt; 和 &amp;gt; 实体正确显示HTML标签。</p>
</div>
```

1. 基础

1.6 注释

- 在 HTML 中，注释通过 <!-- 和 --> 标签定义。这类特殊标记内容不会在浏览器界面呈现，但能辅助代码文档化或临时隐藏待调试的代码段。
- 注释不仅能够阐明代码逻辑与实现意图，在团队协作中更可促进成员间的高效沟通。调试阶段，开发者可快速注释特定代码段进行问题定位，这种非破坏性操作极大提升了排错效率。
- 需特别强调的是，注释内容应具备明确的功能指向性，冗杂注释反而会降低代码可读性。由于用户可通过查看网页源代码获取注释内容，因此必须严格避免在注释中写入敏感数据或隐私信息。
- 规范化注释既保障了代码结构的整洁度与可维护性，又为后续迭代开发构建了有效的技术传承路径，是专业级代码工程的重要实践标准。

1. 基础

1.6 注释

```
<!-- 单行注释演示 -->
<div>
  <h3>1. 单行注释演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;!-- 这是一个单行注释 --&gt;</code></pre>

  <p><strong>说明：</strong>单行注释用于简短的解释说明。</p>

  <!-- 这是一个单行注释，用于解释下面的内容 -->
  <p>我在注释外！</p>

  <p><strong>观察：</strong>上面的注释不会显示在页面上，但下面的段落会正常显示。</p>
</div>

<!-- 多行注释演示 -->
<div>
  <h3>3. 多行注释演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;!--
这是一个多行注释。
你可以在这里写很多内容，
而它们都不会在浏览器中显示。
--&gt;</code></pre>

  <p><strong>说明：</strong>多行注释用于较长的说明或临时隐藏大段代码。</p>

  <!--
  这是一个多行注释的演示。
  你可以在这里写很多内容，
  包括：
  - 开发说明
  - 版本信息
  - 待办事项
  - 代码解释
  而它们都不会在浏览器中显示。
  -->

  <p><strong>观察：</strong>上面的多行注释内容不会显示在页面上。</p>
</div>
```

2. 属性

2.1 属性结构

- 属性包含元素的附加信息，这些信息不会直接显示在页面内容中。在示例中，class属性作为标识名称，以便后续为元素设置样式信息。
- 属性必须包含以下要素：
  - 元素名称与属性之间的空格。若元素包含多个属性，则各属性之间需以空格分隔。
  - 属性名称后接等号。
  - 属性值需包裹在一对引号（""）内。

2. 属性

2.2 公共属性

- HTML元素包含的属性数量庞大，由于数量庞大，本文无法详尽列举所有HTML元素的全部属性，重点针对各元素共通的公共属性展开说明。根据功能特征，公共属性通常可归纳为基本属性、语言属性、键盘属性、内容属性及延伸属性等主要类别，主要涵盖元素标识、交互支持、语义标注等核心功能维度。
- 基本属性
  - 基本属性涵盖以下三个核心特征，这些属性为绝大多数元素所共有。

属性名称	描述
class	定义类规则或样式规则
id	定义元素的唯一标识
style	定义元素的样式声明

2. 属性

- 语言属性
  - 语言属性的核心功能在于明确界定网页元素对应的语言类型，包含两个标准化属性。

属性名称	描述
lang	定义元素的语言代码或编码
dir	定义文本的方向，包括ltr和rtl，分别标识从左向右和从右向左

- 键盘属性
  - 键盘属性定义元素的键盘访问方法，包括两个属性。

属性名称	描述
accesskey	定义访问某元素的键盘快捷键
tabindex	定义元素的Tab键索引编号

2. 属性

- accesskey属性通过快捷键组合（Alt+字母）实现指定URL的快速跳转，需注意确保该快捷键与系统或浏览器的默认组合键不发生冲突。

```
<!-- 链接演示 -->
    <div>
        <h3>1. 链接快捷键演示：</h3>
        <p><strong>源代码：</strong></p>
        <pre><code>&lt;a href="http://www.baidu.com" accesskey="s"&gt;百度首页&lt;/a&gt;</code></pre>

        <p><strong>演示：</strong></p>
        <p>
            <a href="http://www.baidu.com" accesskey="s">
                百度首页（快捷键：Alt+S）
            </a>
        </p>

        <p>
            <a href="http://www.bing.com" accesskey="b">
                必应首页（快捷键：Alt+B）
            </a>
        </p>
    </div>
```

2. 属性

- tabindex属性用于定义元素在Tab键导航序列中的位置顺序。该属性允许用户通过Tab键遍历网页中的可交互元素（如链接和表单控件）。导航顺序由tabindex数值大小决定，当焦点定位至链接元素时，此时按下Enter键将触发该链接的默认操作（通常为跳转至目标页面）。

```
<!-- 演示区域：展示不同tabindex值的导航顺序 -->
<div>
  <h3>1. 按Tab键导航顺序演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;a href="https://www.baidu.com" tabindex="1"&gt;百度&lt;/a&gt;
&lt;a href="https://www.hactcm.edu.cn/" tabindex="2"&gt;河南中医药大学&lt;/a&gt;
&lt;a href="https://xxjsxy.hactcm.edu.cn/" tabindex="3"&gt;信息技术学院&lt;/a&gt;</code></pre>

  <p><strong>演示：</strong></p>
  <!-- 第一个链接：tabindex="1" - 最高优先级 -->
  <p>
      <a href="https://www.baidu.com" tabindex="1">
          1. 百度 (tabindex="1")
      </a>
  </p>

  <!-- 第二个链接：tabindex="2" - 第二优先级 -->
  <p>
      <a href="https://www.hactcm.edu.cn/" tabindex="2">
          2. 河南中医药大学 (tabindex="2")
      </a>
  </p>

  <!-- 第三个链接：tabindex="3" - 第三优先级 -->
  <p>
      <a href="https://xxjsxy.hactcm.edu.cn/" tabindex="3">
          3. 信息技术学院 (tabindex="3")
      </a>
  </p>
</div>
```


2. 属性

- 内容属性
  - 内容属性定义元素用于承载与元素内容相关的补充性信息，其核心功能在于提供关键性补充说明，避免因元素本身信息不完整而导致理解偏差，具体而言包含五个核心属性。

属性名称	描述
alt	定义元素的替换文本
title	定义元素的提示文本
longdesc	定义元素包含内容的大段描述信息
cite	定义元素包含内容的引用信息
datetime	定义元素包含内容的日期和时间

2. 属性

2.2 公共属性

- 内容属性
  - HTML元素的alt与title是两个核心属性，分别用于定义替代性文字描述和辅助信息提示，在实际应用中应严格遵循其语义规范，确保属性内容与上下文场景保持高度关联。
  - 当图像无法显示时，必须提供替代文本以替换无法显示的图像，该属性是图像及图像热点区域的必要功能，因此alt属性仅适用于img、area及input元素。在input元素中，alt属性专门用于替换图片类型的提交按钮。
  - title属性用于为元素提供鼠标悬停时的提示信息，这些补充说明内容属于非必要信息，该属性并非必填项。当用户将光标悬停在元素上方时，浏览器将显示预设的提示文本，但需注意该属性禁止应用于特定类型的元素。

```
<!-- 超链接的title属性演示 -->
<div>
  <h3>1. 超链接的title属性演示：</h3>
  <p><strong>源代码：</strong></p>
  <pre><code>&lt;a href="https://www.baidu.com" title="点击访问百度搜索引擎"&gt;百度搜索&lt;/a&gt;</code></pre>

  <p><strong>演示：</strong>将鼠标悬停在下面的链接上，会显示提示信息：</p>

  <!-- 第一个链接：有title属性 -->
  <p>
      <a href="https://www.baidu.com" title="点击访问百度搜索引擎，提供网页、图片、视频等搜索服务">
          百度搜索
      </a>
  </p>

  <!-- 第二个链接：有title属性 -->
  <p>
      <a href="https://www.hactcm.edu.cn/" title="河南中医药大学官方网站，了解学校信息和招生政策">
          河南中医药大学
      </a>
  </p>

  <!-- 对比：没有title属性的链接 -->
  <p>
      <a href="https://www.google.com">
          谷歌搜索（无title属性）
      </a>
  </p>
</div>
```


2. 属性

2.3 全局属性

- 全局属性（global attributes）指可在所有HTML元素中通用的属性集合。开发者可将这些属性应用于任何HTML元素标签，但需注意某些属性在特定元素中可能呈现无效状态。

属性名称	描述
accesskey	accesskey属性用于指定使网页元素获得焦点的快捷键，其属性值必须为单个可打印字符。当用户触发该快捷键时，对应网页元素将立即获得焦点。
class	class属性用于对网页元素进行分类管理。当多个网页元素具有相同的class属性值时，即视为同类元素，可实现样式统一或功能分组。
contenteditable	默认情况下，HTML网页内容不可编辑。通过设置contenteditable属性，可将元素设置为可编辑状态。该属性接受两个有效值： • 当属性值为true或空字符串时，元素内容可被用户编辑 • 当属性值为false时，元素内容不可编辑
contextmenu	规定元素的上下文菜单。上下文菜单在用户点击元素时显示
data-*	data-*属性用于放置自定义数据。如果没有其他属性或元素合适放置数据，就可以放在data-*属性，如data-tip。
dir	dir属性表示文字的阅读方向，有三个可能的值。 • ltr：从左到右阅读，比如英语。 • rtl：从右到左阅读，阿拉伯语、波斯语、希伯来语都属于这一类。 • auto：浏览器根据内容的解析结果，自行决定。
draggable	用于规定元素是否可拖动。这个属性可以应用于任何 HTML 元素，使其能够被用户拖动。默认情况下，链接和图像是可拖动的。
dropzone	规定在拖动被拖动数据时是否进行复制、移动或链接 • copy拖动数据会产生被拖动数据的副本。 • move拖动数据会导致被拖动数据被移动到新位置。 • link拖动数据会产生指向原始数据的链接。

2. 属性

2.3 全局属性

- 全局属性（global attributes）指可在所有HTML元素中通用的属性集合。开发者可将这些属性应用于任何HTML元素标签，但需注意某些属性在特定元素中可能呈现无效状态。

属性名称	描述
hidden	hidden是一个布尔属性，表示当前的网页元素不再跟页面相关，因此浏览器不会渲染这个元素，所以就不会在网页中看到它。
id	id属性是元素在网页内的唯一标识符。比如，网页可能包含多个<p>标签，id属性可以指定每个<p>标签的唯一标识符。
lang	lang属性指定网页元素使用的语言 常见的语言代码：zh（中文）、zh-Hans（简体中文）、en（英语）、en-US（美国英语）、en-GB（英国英语）
spellcheck	浏览器一般会自带拼写检查功能，编辑内容时，拼错的单词下面会显示红色的波浪线。spellcheck属性就表示，是否打开拼写检查。
style	style属性用来指定当前元素的 CSS 样式。
tabindex	网页交互主要依赖鼠标操作，但在特定场景中，用户可能需要或只能使用键盘进行控制。为此浏览器提供了Tab键遍历网页元素的功能：通过连续按动Tab键，焦点将在可交互元素间顺序跳转。当目标元素获得焦点后，用户可执行回车键激活链接、在输入框直接录入文本等后续操作。
title	title属性用来为元素添加附加说明。大多数浏览器中，鼠标悬浮在元素上面时，会将title属性值作为浮动提示，显示出来。
translate	用来规定对应元素的可翻译属性值及其 Text 子节点内容是否跟随系统语言作出对应的翻译变化。 • 空字符串或 yes，意味着网页在进行本地化的时候，对应内容要被翻译。 • no，意味着对应的内容无需做任何翻译。

3. 元素

3.1 结构元素：搭建页面骨架

- HTML的核心设计目标在于构建网页内容的结构化语义体系。作为网页的基础架构组件，结构元素不仅规范了内容的组织形式，更直接决定了浏览器解析引擎的渲染机制，同时还对搜索引擎索引策略、辅助设备兼容性等底层技术支撑产生了深远影响。
- 文档基础结构
  - <!DOCTYPE>
    - <!DOCTYPE>作为HTML文档的首行代码，需严格置于文档最顶端，且要排在注释等其他内容之前，其作用是声明文档所遵循的HTML版本规范。HTML5的文档类型声明语法如下：

```
<!DOCTYPE html>
```

    - 严格来讲，<!DOCTYPE>声明并不属于 HTML 标签，而是用于声明文档类型以及所遵循的 HTML/XHTML 规范版本，以此触发浏览器的标准模式渲染机制。若该声明缺失或格式不符合规范，浏览器会自动切换至 怪异模式（Quirks Mode），这可能会引发跨浏览器兼容性问题，例如盒模型计算规则存在差异，进而致使页面渲染效果与设计预期不符。

3. 元素

- 文档基础结构
  - <!DOCTYPE>
    - HTML4.01 中的 <!DOCTYPE>声明（了解即可）
    - 严格模式（Strict）：强制使用最新标准，禁用已废弃的标签（如 <font>、 ）

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

    - 过渡模式（Transitional）：允许使用已废弃的标签和属性（兼容旧页面）

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

    - 框架集模式（Frameset）：用于包含 <frameset>的页面（已淘汰）

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```


3. 元素

- 文档基础结构
  - <html>
    - 【元素简介】
      - <html>元素作为HTML文档的根元素（Root Element），其核心职责在于封装所有其他HTML内容。文档结构中的<head>、<body>及其子元素均需严格嵌套于该元素内部，从而构成文档树状结构的根基。此元素不仅是文档解析的起始标志，更是浏览器识别文档类型的关键标识，能够触发页面渲染引擎的解析机制。
    - 【语法结构】

```
1. <!DOCTYPE html>
2. <html lang="zh-CN">
3.   <head>...</head>
4.   <body>...</body>
5. </html>
```

    - 开始标签：<html>（可附带文档类型声明等属性）。
    - 结束标签：</html>（需严格闭合，尽管HTML5规范允许省略该结束标签，但为保障跨平台兼容性，建议显式声明）。
    - 内容规范：该标签必须且仅能包含两个直接子元素，即 <head>（用于定义文档元数据）和 <body>（用于承载页面的可视化内容）。

3. 元素

- 文档基础结构
  - <html>
    - 【核心属性】
      - lang
        - 作用：声明网页内容的主要语言，以确保搜索引擎能够正确开展SEO优化，同时辅助屏幕阅读器等无障碍工具精准解析内容。
        - 取值：遵循BCP 47标准，采用ISO 639 - 1语言代码（2位小写字母）与ISO 3166 - 1国家/地区代码（2位大写字母）组合，构成复合语言标签。
        - 示例：<html lang="zh - CN">（简体中文，中国地区）、<html lang="en - US">（英语，美国地区）。
    - 【使用建议】
      - 强制要求包含 <head> 和 <body> 标签
        - <head>：专门用于存储文档元数据，其中涵盖页面标题、字符编码声明、CSS样式表、外部脚本链接等关键信息，且该部分内容不会直接呈现在页面的可视化渲染结果中。

```
<head>
  <meta charset="UTF-8">
  <title>页面标题</title>
  <link rel="stylesheet" href="style.css">
</head>
```

      - <body>：存放页面的可见内容（如文本、图片、链接、表单等），所有用户直接看到的元素都需嵌套在此。

```
<body>
  <h1>欢迎访问</h1>
  <p>这是一段正文内容。</p>
</body>
```


3. 元素

3.1 结构元素：搭建页面骨架

- 文档基础结构
  - <html>
    - 【使用建议】
      - 在 HTML 文档结构里，<html>元素需唯一存在，作为整个文档的根容器，除 <!DOCTYPE>声明外，所有其他元素都应作为其直接或间接子元素进行嵌套。
      - 务必显式声明 <html>标签：尽管现代浏览器支持隐式解析，但显式声明能确保跨浏览器的渲染一致性，且符合 W3C 标准要求。
      - 精准设置 lang属性值：采用 ISO 639-1 标准语言代码（例如中文使用 zh-CN），可有效提升搜索引擎优化效果和无障碍阅读体验。

3. 元素

- 文档基础结构
  - <head>
    - 【元素简介】
      - <head>作为 HTML 文档的控制核心，承载着页面的元数据体系与非可视化配置信息。该元素借助声明式编程，达成三大核心功能：定义文档属性、协调浏览器行为以及构建资源关联网络。其技术特性如下：
      - 声明基础配置（如字符集声明、文档标题定义）
      - 调控渲染机制（例如视口参数设置、缓存策略配置）
      - 加载外部依赖项（像 CSS 样式表预加载、JavaScript 脚本异步注入）
      - 提供结构化元数据（如 SEO 优化标记、无障碍访问语义标注）

3. 元素

- 文档基础结构
  - <head>
    - 【语法结构】
      - <head>元素必须严格嵌套在 <html> 标签内部，并作为<body>元素的同级前置元素（HTML5 规范虽允许部分标签省略，但显式编写可确保最佳兼容性）。典型文档结构示例如下：

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <!-- 元数据和辅助信息 -->
    <title>页面标题</title>          <!-- 必选：定义页面标题 -->
    <meta charset="UTF-8">          <!-- 必选：声明字符编码 -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- 移动端适配 -->
    <link rel="stylesheet" href="style.css"> <!-- 引入 CSS -->
    <style>                          <!-- 内联 CSS（可选） -->
      body { font-family: sans-serif; }
    </style>
    <script src="app.js" defer></script> <!-- 引入 JS（可选） -->
  </head>
  <body>
    <!-- 可见内容（如文本、图片等） -->
  </body>
</html>
```

      - <head>应作为<html>标签的直接子元素，且每个文档中仅可存在一个。
      - <head>与<body>需保持同级关系，两者之间不得插入任何其他元素（如 ）。
      - 虽然HTML5允许省略<head>标签（浏览器解析时会自动补全），但显式编写该标签能大幅提高代码的可读性。

3. 元素

- 文档基础结构
  - <head>
    - 【核心属性】
      - <head>元素本身没有专属属性，但其功能通过子元素实现。
    - 【子元素】
      - <title>：页面标题：页面标题用于定义在浏览器标题栏或标签页上显示的内容，它是 SEO 核心关键词的重要来源。页面标题用于定义在浏览器标题栏或标签页上显示的内容，它是 SEO 核心关键词的重要来源。虽非强制要求，但强烈建议将其作为 <head> 标签的第一个子元素。另外，标题内容应简洁明了，建议控制在 50 - 60 字符，以防因过长而被搜索引擎截断。

```
1. <title>互联网医疗服务开发课程</title>
```


3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <meta>：元数据元素：<meta>作为 HTML 中极为灵活的元数据元素，主要用于描述页面的基础信息（例如字符集、作者、关键词等），或者对浏览器的行为进行控制（如视口设置、缓存策略）。
      - 核心属性
        - name：用于定义元数据的类型（例如 viewport、keywords、description）。
        - content：表示元数据的具体值，需与 name 属性配合使用。
        - http-equiv：可模拟 HTTP 响应头（例如 X-UA-Compatible、refresh）。
        - charset：用于声明字符编码，这是 HTML5 的简化写法，可替代 http-equiv="Content-Type"。
        - 字符编码：用于声明字符集，推荐使用 UTF - 8，因其支持多语言，可有效防止乱码。

```
1. <meta charset="UTF-8">
```

        - 视口配置（移动端）：对移动端页面的缩放与布局进行控制（采用 “width=device-width” 可使页面宽度与设备宽度保持一致）。

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

3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <meta>：元数据元素：<meta>作为 HTML 中极为灵活的元数据元素，主要用于描述页面的基础信息（例如字符集、作者、关键词等），或者对浏览器的行为进行控制（如视口设置、缓存策略）。
        - 关键词（SEO）：声明页面关键词（该方式已逐渐被搜索引擎弱化）。

```
1. <meta name="keywords" content="课程，Web，互联网">
```

        - 描述（SEO）：（重要）此为声明页面摘要，其可能会显示于搜索引擎结果之中。

```
1. <meta name="description" content="互联网医疗服务开发课程建设">
```

        - HTTP 头部等价：模拟 HTTP 头部设置，例如强制 Internet Explorer（IE）浏览器使用其最新内核进行页面渲染。

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

        - 刷新/重定向：5 秒后将自动跳转至指定 URL（不建议滥用此功能）。

```
1. <meta http-equiv="refresh" content="5;url=https://example.com">
```


3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <link>：引入外部资源。<link>标签用于链接外部资源（非页面内容），其常见用途包括引入 CSS 样式表或网站图标（Favicon）。
        - 核心属性：
        - rel：该属性用于定义当前文档与链接资源之间的关系，为必填项。
        - stylesheet：此值用于链接 CSS 样式表，是 rel 属性最常用的取值。
        - icon：用于指定网站图标（Favicon）。
        - preload：用于预加载资源，可优化性能，适用于字体、脚本等资源。
        - href：该属性指定外部资源的 URL 路径，为必填项。
        - type（可选）：该属性用于指定资源的 MIME 类型，例如 text/css 表示 CSS 资源，在现代浏览器中此属性可省略。

```
1. <!-- 引入外部 CSS -->
2. <link rel="stylesheet" href="styles.css" type="text/css">

3. <!-- 设置网站图标（Favicon） -->
4. <link rel="icon" href="/favicon.ico" type="image/x-icon">

5. <!-- 预加载关键字体（提升渲染速度） -->
6. <link rel="preload" href="font.woff2" as="font" type="font/woff2" crossorigin>
```

3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <style>：内联 CSS 样式。<style>标签用于在 <head>标签中直接嵌入 CSS 样式代码，此方法适用于少量样式的情况。使用时需通过 type 属性声明样式类型为 “text/css”，不过在 HTML5 中该声明可省略。需要注意的是，通过 <style>标签定义的样式仅对当前页面有效。

```
<style>
  body {
    font-family: "微软雅黑", sans-serif;
    margin: 0;
  }
  h1 {
    color: #333;
  }
</style>
```


3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <base>：设定基准 URL。<base>标签用于为页面中所有相对 URL 定义基准地址，此设定仅对 <a>、<img>、<link>等标签的相对路径产生影响。值得注意的是，一个文档中最多只能存在一个 <base>标签，且该标签需置于 <head>标签的最前端，以避免后续标签引用时出现未生效的情况。
      - 核心属性
        - href：基准 URL，为必填属性。
        - target（可选）：用于指定链接的打开方式，例如使用 _blank 可在新窗口中打开链接。

```
<base href="https://www.example.com/path/" target="_blank">
<!-- 实际效果：<a href="about.html"> 会指向 https://www.example.com/path/about.html -->
```

3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <script>：引入 JavaScript 脚本。<script>标签用于在 <head> 中嵌入或引入 JavaScript 代码，该代码主要负责控制页面的交互逻辑。
      - 核心属性
        - src（可选）：此属性用于指定外部 JS 文件的 URL 路径，若要引入外部文件，则必须填写该属性。
        - type（可选）：该属性用于指定脚本类型，通常为 text/javascript，在 HTML5 中此属性可省略。
        - async/defer（可选）：这两个属性用于控制脚本的加载时机，有助于优化页面性能。async：使用该属性时，脚本将进行异步加载，加载完成后会立即执行，不过可能会打乱脚本的执行顺序。defer：使用该属性时，脚本同样进行异步加载，但会延迟到 HTML 解析完成后再执行，从而保证脚本按顺序执行。

```
<!-- 引入外部 JS 文件（带 defer，确保 DOM 加载完成后执行） -->
<script src="app.js" defer></script>

<!-- 内联 JS 代码 -->
<script>
  console.log("页面加载完成！");
</script>
```


3. 元素

- 文档基础结构
  - <head>
    - 【子元素】
      - <noscript>：无脚本支持时的备用内容。<noscript>标签用于定义在浏览器禁用 JavaScript 时显示的备用内容，例如提示用户启用脚本。需要注意的是，该标签需放置在 <head> 或 <body> 标签内，通常在 <head> 中与 <script> 标签配合使用。若浏览器支持 JavaScript，则 <noscript> 标签内的内容将被忽略。

```
<script>
  // 启用 JS 时隐藏提示
  document.write('<style>#no-script-warning { display: none; }</style>');
</script>
<noscript>
  <div id="no-script-warning">
    警告：您的浏览器禁用了 JavaScript，部分功能可能无法使用！
  </div>
</noscript>
```

3. 元素

- 文档基础结构
  - <head>
    - 【使用建议】
      - 必须包含关键子元素，否则可能引发功能异常：<title>：用于定义页面标题，该标题会显示在浏览器标签页，同时也是 SEO 核心关键词的重要来源。<meta charset>：用于声明字符编码，建议采用 UTF - 8，可有效防止页面出现乱码。
      - 内容限制：仅允许包含元数据标签（如 <title>、<meta>、<link>、<style>、<script>、<base>）和脚本支持标签（如内联 CSS/JS）。严禁包含可见内容（如 <p>、 、<h1> 等块级/行内元素）。
      - 子元素顺序：虽无严格顺序要求，但建议按照“基础信息→资源引入→脚本”的逻辑进行排列。例如，将 <title> 和 <meta charset> 置于最前端，使用 <link> 引入 CSS 置于中间，<script> 放置在底部。建议显式声明 <head> 标签。尽管 HTML5 允许省略该标签，但显式编写可增强代码的可读性，避免旧版本浏览器在解析时出现错误。
      - 需优化 <title> 标签内容，确保其简洁明了（建议控制在 50 - 60 字符，避免因过长而被搜索引擎截断），并包含核心关键词（如页面主题、产品名称），以提升 SEO 效果。
      - 应强制声明字符编码，始终采用 <meta charset="UTF - 8">，这是现代网页的标准编码，支持多语言，可有效防止乱码。
      - 进行移动端适配（视口配置）时，必须添加 <meta name="viewport" content="width=device - width, initial - scale = 1.0">，以确保页面在手机、平板等移动设备上能够正确缩放和布局。
      - 需合理管理资源加载：应将 CSS 放置在 <head> 中，以确保在页面渲染前加载样式，避免出现“白屏”现象。JS 应尽量放置在 <head> 底部，或使用 defer/async 属性，以避免阻塞 HTML 解析。
      - 应减少内联样式和脚本的使用。内联 CSS/JS 仅适用于少量临时样式或脚本，对于全局样式和复杂逻辑，建议通过外部文件引入，以便于维护和缓存。
      - 应避免使用冗余元标签，若非必要，请勿添加重复或过时的元标签。例如，keywords 已被搜索引擎弱化，description 更为重要。

3. 元素

- 文档基础结构
  - <body>
    - 【元素简介】
      - <body>作为 HTML 文档的主体元素，用于承载页面中所有可见内容，即用户直接浏览并与之交互的部分。它是 <html>元素的直接子元素，与 <head>元素呈“兄弟”关系，其核心价值主要体现在以下方面：
      - 界定页面的核心内容，涵盖文本、图片、链接、表单、视频等各类元素。
      - 充当用户交互的核心，诸如点击、滚动、输入等操作均在此展开。
      - 支持语义化标签，例如 <header>、<main>、<article>等，有助于提升页面的可访问性（无障碍）和搜索引擎优化（SEO）效果。

3. 元素

- 文档基础结构
  - <body>
    - 【语法结构】
      - <body>元素必须嵌套于 <html>元素内部，且应位于 <head>元素之后（尽管 HTML5 允许省略部分标签，但建议显式编写）。

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <title>页面标题</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <!-- 所有可见内容 -->
    <header>页面头部</header>
    <main>
      <h1>主标题</h1>
      <p>段落内容...</p>
      <img src="image.jpg" alt="图片描述">
    </main>
    <footer>页面底部</footer>
  </body>
</html>
```

      - 在 HTML 文档中，依据 HTML5 的严格规定，必须且只能存在一个 <body> 元素。
      - <body> 与 <head> 属于“兄弟关系”，二者之间不得插入其他元素（例如 ）。
      - HTML5 虽允许省略 <body> 标签，浏览器会自动补全，但为提升代码的可读性，强烈建议显式编写该标签。

3. 元素

- 文档基础结构
  - <body>
    - 【核心属性】
      - <body>元素支持 HTML 的全局属性（例如 id、class、style）以及部分事件处理属性（用于响应页面生命周期事件）。以下为常用属性：

属性	说明	示例
id	为 body指定唯一标识（页面中不可重复，用于 CSS/JS 定位）。	<body id="home-page">
class	为 body指定一个或多个类名（用于批量 CSS 样式或 JS 选择）。	<body class="dark-mode mobile">
style	直接为 body添加内联 CSS 样式（控制页面整体外观）。	<body style="background: #f5f5f5;">
lang	声明页面内容的语言（与 html的 lang重复时可省略，但建议统一）。	<body lang="zh-CN">

3. 元素

- 文档基础结构
  - <body>
    - 【核心属性】
      - <body>支持通过事件属性绑定 JavaScript 代码，以响应页面加载、卸载等关键事件。尽管在现代开发中，更推荐使用 addEventListener 方法进行绑定，但属性写法依然保持兼容。

属性	说明	示例
onload	页面完全加载完成后触发（包括图片、样式表等资源）。	<body onload="initApp()">
onunload	页面即将卸载时触发（如关闭标签页、跳转页面）。	<body onunload="logExit()">
onresize	窗口尺寸变化时触发（常用于响应式布局调整）。	<body onresize="adjustLayout()">
onscroll	页面滚动时触发（用于滚动动画或懒加载）。	<body onscroll="lazyLoadImages()">

3. 元素

3.1 结构元素：搭建页面骨架

- 文档基础结构
  - <body>
    - 【使用建议】
      - 在一个 HTML 文档中，依据 HTML5 的强制规定，仅允许存在一个 <body> 元素。若违反此规则，浏览器会自动进行纠正或报错。
      - <body> 元素仅可包含可见内容元素（例如文本、 、<p>、<img>、<a> 等）以及语义化标签（如 <header>、<section>）。严禁包含元数据类元素（像 <title>、<meta>、<link>），此类元素必须置于 <head> 标签内。
      - 非空元素必须正确闭合，例如 <body></body>。尽管 HTML5 允许省略结束标签，但为避免解析错误，建议显式编写结束标签。
      - 建议运用 HTML5 语义化标签（如 <header>、<nav>、<main>、<article>、<aside>、<footer>）来组织内容，以此替代大量无实际意义的 标签，从而提升页面的可访问性（对屏幕阅读器友好）和搜索引擎优化（SEO）效果。
      - 应尽量通过外部 CSS 文件或 <style> 标签统一管理样式，仅在必要场景（如调试）下使用 style 属性添加临时样式。
      - 在现代开发中，推荐采用 JavaScript 的 addEventListener 方法绑定事件（如 DOMContentLoaded），而非直接使用 onload 等属性，以增强代码的可维护性。
      - 适配无障碍访问：
        - 若页面无明确标题，需为 <body> 添加 aria-label 或 aria-labelledby 属性，助力屏幕阅读器理解页面用途。
        - 要确保内容顺序符合逻辑（如标题在前，正文在后），防止因 CSS 浮动或定位致使屏幕阅读器读取混乱。

3. 元素

- 语义化结构元素
  - 在早期的 HTML 开发中，页面结构的区域划分通常采用 （通用容器）搭配 id 或 class 的方式，例如 。然而，这种方式缺乏语义性，浏览器和屏幕阅读器无法理解诸如 “header” 这类标识的具体含义，进而导致搜索引擎优化（SEO）效果不佳，且对无障碍访问的支持较弱。HTML5 引入了语义化结构标签，它能够通过标签本身来表达内容的含义，这一特性使其成为现代 Web 开发的核心规范。
  - 语义化元素的核心优势主要体现在以下三个方面：
    - 优化 SEO：搜索引擎能够借助语义元素（如 <article>）识别页面的核心内容（例如一篇博客），从而提升该内容在搜索结果中的排名。
    - 增强无障碍性：屏幕阅读器（如 JAWS、NVDA）可以依据语义标签朗读页面内容，如 “导航区域” “文章标题” 等，有助于视障用户理解页面结构。
    - 提升可维护性：语义化标签使代码的可读性更强，例如看到 <nav> 就能判断其为导航栏，从而降低了团队协作过程中的沟通成本。

3. 元素

- 语义化结构元素

标签	语义定义	使用场景示例	技术注意事项
<header>	页面或区域的头部，通常包含标题、Logo、主导航	博客首页的顶部区域（含Logo和导航）	一个页面可包含多个<header>（如文章详情页的<article>顶部也可使用）
<nav>	导航栏，包含指向页面内其他部分或外部资源的链接	页面顶部的主导航（如“首页”“归档”“关于”）	导航链接推荐使用 <a>结构（符合无障碍标准），避免直接使用 包裹链接
<main>	页面的主内容区域，代表文档的核心内容	博客首页的文章列表或文章详情页的正文	一个页面只能有一个<main>；不可嵌套在<article>或<section>中
<article>	独立、可复用的内容块，即使脱离上下文仍有独立意义	一篇博客文章、一条新闻、用户评论	内容需具备“独立性”（如单独复制到其他网站仍可理解）
<section>	分组相关的内容块，通常包含标题	文章的“引言”“优势”“总结”章节	若内容块无独立标题（如装饰性区块），应使用 而非<section>
<aside>	与主内容间接相关的辅助内容	“热门标签”“作者简介”“广告”	内容可独立于主内容存在（如删除主内容后仍有意义）
<footer>	页面或区域的底部，通常包含版权、备案、联系方式等信息	博客首页的底部（含版权声明、备案号）	一个页面可包含多个<footer>（如文章详情页的<article>底部也可使用）

3. 元素

- 语义化结构元素
  - <header>
    - 【元素简介】
      - <header>作为 HTML5 引入的语义化元素，用于定义文档或区域的头部区域（Header Region）。其核心功能是通过语义化标记，精准界定页面或内容块的“头部”部分，该部分包含标题、导航及简介等内容，而并非像传统的 那样仅作为样式容器。此元素的核心价值主要体现在以下几个方面：：
      - 语义明确：能够让浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）快速理解内容的结构和功能。
      - 结构清晰：以之替代传统无意义的 ，可提升代码的可读性和可维护性。
      - 无障碍友好：有利于屏幕阅读器识别内容的逻辑层次，例如识别出“这是文章的头部”。界定页面的核心内容，涵盖文本、图片、链接、表单、视频等各类元素。

3. 元素

- 语义化结构元素
  - <header>
    - 【语法结构】
      - <header>为块级元素，默认情况下会占满父容器的宽度，常作为父容器来嵌套其他元素，如标题、导航、简介等。其典型结构如下：

```
<!-- 页面级头部（网站顶部） -->
<header class="site-header">
  <h1>网站名称</h1>
  <nav>
    <ul>
      <li><a href="/">首页</a></li>
      <li><a href="/about">关于</a></li>
    </ul>
  </nav>
</header>

<!-- 文章级头部（文章标题区） -->
<article>
  <header class="article-header">
    <h2>文章标题</h2>
    <p class="meta">作者：张三 · 发布时间：2025-09-18</p>
  </header>
  <p>文章正文内容...</p>
</article>
```

      - <header>可作为 <body>、<article>、<section>等元素的子元素，并且一个文档中允许存在多个 <header>。
      - 通常情况下，<header>位于内容块的顶部，然而其语义优先级高于位置，即便调整位置，仍可用 <header>进行标记。
      - <header>内部可包含标题（<h1> - <h6>）、导航（<nav>）、简介（<p>）等内容，但不建议直接包含主要内容，主要内容应置于<main>或 <article>中。

3. 元素

- 语义化结构元素
  - <header>
    - 【核心属性】
      - <header>元素继承了 HTML 的全局属性，并且可通过 ARIA（无障碍富互联网应用）属性提升可访问性。为增强屏幕阅读器的可访问性，可添加以下 ARIA 属性。

属性	说明	示例
aria-label	为 <header>添加描述性元素（当内容无法明确表达用途时使用）。	<header aria-label="网站导航头部">...</header>
aria-labelledby	关联其他元素的 ID，通过元素内容描述<header>的用途（更灵活）。	<header aria-labelledby="header-title">...</header>（配合 <h1 id="header-title">标题</h1>）

3. 元素

- 语义化结构元素
  - <header>
    - 【使用建议】
      - 语义适用性
        - 仅适用于标记“头部”区域，诸如页面顶部、文章标题区、卡片头部等，不适用于主要内容区（主要内容应使用 <main> 或 <article> 元素）。
        - 应避免使用 <header> 包裹页脚（页脚应使用 <footer> 元素）、侧边栏（侧边栏应使用 <aside> 元素）等无关内容。
      - 嵌套规则
        - 可作为 <body>、<article>、<section>、 等元素的子元素，但需遵循逻辑，例如 <article> 内的 <header> 代表文章头部。
        - 内部可包含标题（<h1> - <h6>）、导航（<nav>）、文本（<p>）等，但不建议直接包含交互组件，如按钮、表单（除非其为头部功能的一部分）。
      - 一个文档中可包含多个 <header>，如页面级头部与文章级头部，但要确保每个 <header> 对应明确的逻辑区域。
      - HTML5 之前的浏览器（如 IE9 以下）不支持 <header> 元素，而现代浏览器（如 Chrome、Firefox、Edge 等）均已实现完美支持。
      - 优先选用语义化元素，而非样式容器，建议使用 <header> 替代 ，以明确标识区域的“头部”语义，而非仅将其作为 CSS 样式的钩子。
      - 合理组织内部内容：
        - 需对页面级别的内容进行合理组织。

3. 元素

- 语义化结构元素
  - <header>
    - 【使用建议】
      - 合理组织内部内容：
        - 需对页面级别的内容进行合理组织。

```
<header class="site-header">
  <h1>我的博客</h1>
  <nav>
    <a href="/">首页</a> |
    <a href="/posts">文章</a> |
    <a href="/about">关于</a>
  </nav>
</header>
```

        - 文章层面的头部包含标题、作者、发布时间等元信息。

```
<article>
  <header class="post-header">
    <h2>HTML5 语义化指南</h2>
    <p class="post-meta">作者：李四 · 2025-09-18 · 阅读时长：8分钟</p>
  </header>
  <p>正文内容...</p>
</article>
```


3. 元素

- 语义化结构元素
  - <header>
    - 【使用建议】
      - 合理组织内部内容：
        - 利用 ARIA 提升无障碍性：当 <header> 内容表意不明确（如仅包含图标或抽象装饰）时，可借助 aria-label 或 aria-labelledby 进行补充阐释。

```
<!-- 图标 + 文字的头部 -->
<header aria-label="网站导航">
  <svg>...</svg> <!-- 导航图标 -->
  <span>菜单</span>
</header>
```

3. 元素

- 语义化结构元素
  - <nav>
    - 【元素简介】
      - <nav> 作为 HTML5 引入的语义化元素，精准定义了文档或区域的导航链接区域（Navigation Region）。其核心功能是通过语义化标记，清晰地标识页面中的“导航功能区”，像主导航菜单、目录、面包屑等，而非像传统的 那样仅作为样式容器。该元素的核心价值主要体现在以下几个方面：
      - 语义明确：有利于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）快速识别该区域的“导航”功能。
      - 结构清晰：用 <nav> 元素取代传统无意义的 ，能大幅提高代码的可读性和可维护性。
      - 无障碍友好：便于屏幕阅读器识别导航区域，使用户可借助语音指令实现快速跳转。

3. 元素

- 语义化结构元素
  - <nav>
    - 【语法结构】
      - <nav> 属于块级元素，默认情况下会占满父容器的宽度。它通常作为父容器，用于嵌套其他与导航相关的元素（如链接列表、按钮等）。其典型结构如右侧所示。
        - <nav>元素可作为 <header>、<main>、<aside>、 等元素的子元素，并且一个文档中能够包含多个 <nav>。
        - 此元素一般用于包裹一组相关链接，例如主导航、次级导航、页内目录、分页等。
        - 其内部可以包含文本，如“导航”标题；列表，如 /<ol>；按钮，如折叠菜单的触发按钮等。不过，不建议直接包含非导航内容，像广告、无关文本等。
    - 【核心属性】
      - <nav>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!-- 页面级主导航（常见于 header 内） -->
<header class="site-header">
  <nav class="main-nav">
    <ul>
      <li><a href="/">首页</a></li>
      <li><a href="/posts">文章</a></li>
      <li><a href="/about">关于</a></li>
      <li><a href="/contact">联系</a></li>
    </ul>
  </nav>
</header>

<!-- 文章内目录导航（次级导航） -->
<article>
  <nav class="toc-nav" aria-label="文章目录">
    <h3>目录</h3>
    <ul>
      <li><a href="#section1">第一章</a></li>
      <li><a href="#section2">第二章</a></li>
      <li><a href="#section3">第三章</a></li>
    </ul>
  </nav>
  <h2 id="section1">第一章内容...</h2>
  <p>正文...</p>
</article>

<!-- 移动端折叠导航（配合 JS 交互） -->
<nav class="mobile-nav">
  <button class="menu-toggle">三</button>
  <ul class="nav-links" hidden>
    <li><a href="/">首页</a></li>
    <li><a href="/posts">文章</a></li>
  </ul>
</nav>
```

3. 元素

- 语义化结构元素
  - <main>
    - 【元素简介】
      - <main> 作为 HTML5 引入的语义化核心元素，用于精确界定文档或应用的主要内容区域（Main Content Region）。其关键意义在于，通过语义化标记，清晰地标识出页面中最核心、高度相关且具有唯一性的内容，诸如文章主体、商品详情、功能操作区等，而非仅仅作为样式容器使用，这与传统的 存在显著差异。该元素的核心价值主要体现在以下几个方面：
      - 语义清晰：有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）快速识别该区域为页面的“核心内容”。
      - 结构合理：替代传统缺乏语义的 ，提高代码的可读性和可维护性。
      - 无障碍适配：方便屏幕阅读器直接定位核心内容，用户能够通过导航实现快速跳转。
      - SEO 增强：搜索引擎会优先抓取 <main> 内的内容，将其认定为页面核心信息，从而提升关键内容的相关性评分。

3. 元素

- 语义化结构元素
  - <main>
    - 【语法结构】
      - <main> 属于块级元素，默认情况下会占满父容器的宽度。它通常作为 <body> 的直接子元素，或者嵌套于少量语义化容器（如 ）中，但需确保逻辑合理。其典型结构如右侧。
        - 根据 HTML5 的强制规定，在一个 HTML 文档中必须且只能有一个 <main> 元素，否则浏览器会自动纠正或报错。
        - <main> 元素应直接包含页面的核心内容，如文章主体、功能操作区等，而不应包含导航、页脚、侧边栏等辅助内容。
        - 嵌套限制：
        - 此元素既可以作为 <body> 的直接子元素，也可以嵌套在无语义的 元素中，但应避免多层嵌套。
        - 不得将 <main> 元素嵌套在 <article>、<section>、<aside>、<nav>、<header>、<footer> 等语义化元素内，因为这些元素的内容应直接归属于 <main> 元素或独立存在。
    - 【核心属性】
      - <main>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <title>页面标题</title>
  </head>
  <body>
    <header>页面头部（导航、标题等）</header>
    <nav>主导航菜单</nav>

    <!-- 主内容区域（唯一） -->
    <main id="primary-content" class="main-content">
      <!-- 核心内容块（如文章、商品详情等） -->
      <article>
        <h1>文章标题</h1>
        <p>文章正文内容...</p>
      </article>

      <!-- 其他核心内容（如侧边栏不属于 main） -->
      <section class="related-posts">
        <h2>相关文章</h2>
        <ul>...</ul>
      </section>
    </main>

    <footer>页面底部（版权、联系方式等）</footer>
  </body>
</html>
```

3. 元素

- 语义化结构元素
  - <article>
    - 【元素简介】
      - <article>作为 HTML5 引入的语义化核心元素，用于定义文档、页面或应用中独立、完整且可复用的内容块（Standalone Content）。其核心功能是借助语义化标记，精准标识一段“自包含”的内容（例如博客文章、新闻条目、评论、论坛帖子、商品详情等），使该内容在结构上与页面其他部分相互独立，同时支持跨上下文复用（如分享、转载）。其核心价值主要体现在以下方面：
      - 语义明确：便于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）迅速识别内容的“独立性”与“完整性”。
      - 结构清晰：以 <article>元素取代传统无意义的 元素，可增强代码的可读性与可维护性。
      - 无障碍友好：有助于屏幕阅读器识别内容的独立区块，方便用户通过导航快速跳转。
      - SEO 优化：搜索引擎会将 <article>内的内容视作独立主题，从而提高关键信息的抓取权重。

3. 元素

- 语义化结构元素
  - <article>
    - 【语法结构】
      - <article> 为块级元素，默认会占满父容器的宽度。它常作为父容器，嵌套诸如 <header>、<section>、<footer> 等其他语义化元素，用于组织内容的标题、正文以及元信息等。其典型结构如右侧。
        - <article>可作为 <main>、<section>、 等元素的子元素，一个文档中可包含多个<article>。
        - 内容应具备独立性与完整性，即便脱离页面上下文，用户仍能领会其核心要义，例如单独发布的一篇博客文章。
        - 其内部可嵌套其他语义化元素，如 <header>用于表示内容头部，<section>用于划分章节，<footer>用于包含元信息或操作按钮。
    - 【核心属性】
      - <article>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!-- 博客文章（独立内容块） -->
<article class="blog-post" id="post-123">
  <header class="post-header">
    <h1>HTML5 语义化指南：从入门到精通</h1>
    <p class="post-meta">作者：张三 · 发布时间：2025-09-18 · 阅读时长：10分钟</p>
  </header>

  <section class="post-content">
    <h2>什么是语义化？</h2>
    <p>语义化是指通过 HTML 元素明确内容的含义（如用 `&lt;article&gt;` 表示文章，而非 `&lt;div&gt;`）...</p>

    <h2>为什么需要语义化？</h2>
    <p>语义化能提升 SEO、无障碍访问和代码可维护性...</p>
  </section>

  <footer class="post-footer">
    <div class="tags">
      <a href="/tag/html5">HTML5</a>
      <a href="/tag/semantic">语义化</a>
    </div>
    <button class="like-btn">点赞（123）</button>
  </footer>
</article>

<!-- 评论区块（独立可复用内容） -->
<article class="comment" id="comment-456">
  <header class="comment-header">
    <img src="user-avatar.jpg" alt="用户头像" class="avatar">
    <h3 class="comment-author">李四</h3>
    <time datetime="2025-09-18T14:30">2025-09-18 14:30</time>
  </header>
  <p class="comment-text">这篇文章写得非常清晰！语义化的实际应用场景讲得很到位。</p>
</article>
```

3. 元素

- 语义化结构元素
  - <section>
    - 【元素简介】
      - <section>作为 HTML5 引入的语义化结构元素，用于定义文档或应用中与主题相关且逻辑独立的区域或章节（Thematic Section）。其核心功能在于通过语义化标记，依据主题或功能对页面内容进行精准划分，形成清晰的区块，如文章章节、产品特性介绍、页面功能模块等，从而提升内容的可读性、可维护性以及无障碍访问体验。其核心价值具体如下：
      - 语义明晰：有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）快速理解该区域的“主题性”与“逻辑独立性”。
      - 结构合理：替代传统无实际语义的 ，使代码更符合人类阅读习惯。
      - 无障碍支持：协助屏幕阅读器识别内容分区，使用户能够通过导航迅速跳转至目标章节。
      - SEO 强化：搜索引擎可借助 <section>识别页面的内容结构（如章节标题），提高内容相关性的抓取权重。

3. 元素

- 语义化结构元素
  - <section>
    - 【语法结构】
      - <section>作为块级元素，默认情况下会占满父容器的宽度。通常，它作为父容器嵌套于其他语义化元素（例如 <main>、<article>、 ）之中，其内部能够包含标题（<h1>-<h6>）、段落（<p>）、列表（<ul>/<ol>）等内容。其典型结构如右侧。
        - <section>元素可作为 <main>、<article>、 等元素的子元素，且一个文档中可包含多个 <section>。
        - <section>区域内的内容需围绕同一主题，如“产品功能”“文章章节”“用户信息”等，且在逻辑上应独立于其他区域。
        - 建议 <section>包含标题（<h1>-<h6>），以便屏幕阅读器和用户能够迅速理解该区域的主题。若无标题，<section>可能会被视作“无意义分区”。
    - 【核心属性】
      - <section>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!-- 博客文章（独立内容块） -->
<article class="blog-post" id="post-123">
  <header class="post-header">
    <h1>HTML5 语义化指南：从入门到精通</h1>
    <p class="post-meta">作者：张三 · 发布时间：2025-09-18 · 阅读时长：10分钟</p>
  </header>

  <section class="post-content">
    <h2>什么是语义化？</h2>
    <p>语义化是指通过 HTML 元素明确内容的含义（如用 `&lt;article&gt;` 表示文章，而非 `&lt;div&gt;`）...</p>

    <h2>为什么需要语义化？</h2>
    <p>语义化能提升 SEO、无障碍访问和代码可维护性...</p>
  </section>

  <footer class="post-footer">
    <div class="tags">
      <a href="/tag/html5">HTML5</a>
      <a href="/tag/semantic">语义化</a>
    </div>
    <button class="like-btn">点赞（123）</button>
  </footer>
</article>

<!-- 评论区块（独立可复用内容） -->
<article class="comment" id="comment-456">
  <header class="comment-header">
    <img src="user-avatar.jpg" alt="用户头像" class="avatar">
    <h3 class="comment-author">李四</h3>
    <time datetime="2025-09-18T14:30">2025-09-18 14:30</time>
  </header>
  <p class="comment-text">这篇文章写得非常清晰！语义化的实际应用场景讲得很到位。</p>
</article>
```

3. 元素

- 语义化结构元素
  - <aside>
    - 【元素简介】
      - <aside>作为 HTML5 引入的语义化辅助元素，用于定义文档或页面中与主内容相关但逻辑独立、可分离的辅助信息（Supplementary Content）。其核心功能是通过语义化标记，精准界定页面中的“辅助区域”（如侧边栏、相关链接、注释、广告、作者信息等），使该区域在结构上与主内容相互独立又紧密关联。其核心价值主要体现在以下方面：
      - 语义清晰：有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）快速识别该区域为“辅助信息”，而非主内容。
      - 结构合理：使用 <aside>元素取代传统无意义的 ，使代码更符合语义化规范。
      - 无障碍性佳：便于屏幕阅读器识别辅助区域，用户可通过导航跳过或聚焦该区域，提升阅读效率。
      - SEO 优化显著：搜索引擎可借助 <aside>元素识别页面的辅助信息（如相关链接、元素），增强主内容的相关性。

3. 元素

- 语义化结构元素
  - <aside>
    - 【语法结构】
      - <aside>为块级元素，默认状态下会占满父容器的宽度。它通常作为子元素嵌套于 <main>、<article>、 等父元素之中，其内部能够包含文本、链接、列表、图片等内容。其典型结构如右侧。
        - <aside>可作为 <main>、<article>、 等元素的子元素，且一个文档中可包含多个 <aside>。
        - 其内容需与主内容存在逻辑关联，例如文章的“相关链接”、商品的“用户评价”，但应独立于主内容的核心流程。
        - 具有可分离性，即便移除 <aside>，主内容的核心信息依然完整，例如侧边栏的“相关链接”不会影响对文章正文的理解。
    - 【核心属性】
      - <aside>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!-- 页面侧边栏（主内容外的辅助信息） -->
<main>
  <article>主内容（文章正文）...</article>
</main>
<aside class="sidebar" aria-label="文章相关信息">
  <h3>相关链接</h3>
  <ul>
    <li><a href="/related-post1">相关文章 1</a></li>
    <li><a href="/related-post2">相关文章 2</a></li>
  </ul>

  <h3>作者信息</h3>
  <img src="author-avatar.jpg" alt="作者头像" class="author-avatar">
  <p class="author-name">张三</p>
  <p class="author-bio">前端开发者，专注 HTML/CSS/JS 技术分享。</p>
</aside>

<!-- 文章内注释（主内容中的补充信息） -->
<article>
  <h1>HTML5 语义化指南</h1>
  <p>正文内容...</p>

  <aside class="note" aria-label="重要说明">
    <p>注意：语义化元素需根据内容实际用途选择，避免滥用。</p>
  </aside>
</article>

<!-- 页面广告（与主内容相关的推广信息） -->
<div class="page-container">
  <main>主内容...</main>
  <aside class="advertisement" aria-label="推广信息">
    <h3>限时优惠</h3>
    <p>新用户注册即送 100 元优惠券！</p>
    <button>立即注册</button>
  </aside>
</div>
```

3. 元素

- 语义化结构元素
  - <footer>
    - 【元素简介】
      - <footer>作为 HTML5 引入的语义化结构元素，主要用于精准定义文档、页面或内容区域的底部区域（Footer Region）。其核心功能是通过语义化标记，精确标识页面或内容块的“页脚”部分，涵盖版权信息、联系方式、相关链接、备案号等内容，使该部分在结构上与主内容相互独立，同时高效传递与区域相关的元信息。其核心价值具体如下：
      - 语义精准：利于浏览器、搜索引擎（SEO）及辅助技术（如屏幕阅读器）快速识别该区域为“页脚”，清晰明确其承载着与区域相关的补充信息。
      - 结构明晰：使用该元素取代传统无意义的 ，使代码更符合语义化规范，大幅提高代码的可维护性。
      - 无障碍适配：方便屏幕阅读器识别页脚区域，用户可利用导航功能迅速跳转至该区域，对视力障碍用户极为便利。
      - SEO 增强：搜索引擎可通过 <footer>识别页面的元信息（如版权、备案号），进而补充主内容的相关性。

3. 元素

- 语义化结构元素
  - <footer>
    - 【语法结构】
      - <footer> 作为块级元素，默认情况下会占满父容器的宽度。通常，它会被嵌套于 <body>、<main>、<article>、<section> 等元素的末尾，作为这些父容器的一部分。其内部可容纳文本、链接、列表、图标等内容。
        - <footer>可作为 <body>、<main>、<article>、<section>、 等元素的子元素，一个文档中允许包含多个 <footer>。
        - <footer>通常位于父容器的最底部，例如作为<body>的最后一个子元素，但在语义上其优先级高于位置，即便是调整了位置，仍可使用<footer>进行标记。
        - <footer>内的内容需与所在父容器高度相关，例如页面页脚可包含网站信息，文章页脚可包含文章元信息。
    - 【核心属性】
      - <footer>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```html
<!-- 页面级页脚（网站底部） -->
<body>
  <header>页面头部</header>
  <nav>主导航</nav>
  <main>主内容...</main>

  <!-- 页面页脚 -->
  <footer class="site-footer" aria-label="网站页脚">
    <div class="footer-container">
      <div class="copyright">
        © 2025 腾讯元宝. 保留所有权利 | 备案号：粤ICP备12345678号
      </div>
      <div class="footer-links">
        <a href="/about">关于我们</a>
        <a href="/privacy">隐私政策</a>
        <a href="/contact">联系我们</a>
      </div>
      <div class="social-media">
        <a href="https://weibo.com" aria-label="微博"><svg>...</svg></a>
        <a href="https://wechat.com" aria-label="微信"><svg>...</svg></a>
      </div>
    </div>
  </footer>
</body>

<!-- 文章页脚（文章元信息） -->
<article>
  <h1>HTML5 语义化指南</h1>
  <p>正文内容...</p>

  <footer class="article-footer" aria-label="文章页脚">
    <p class="meta">发布时间：2025-09-18 · 阅读时长：10分钟 · 作者：张三</p>
    <div class="article-tags">
      <a href="/tag/html5">HTML5</a>
      <a href="/tag/semantic">语义化</a>
    </div>
  </footer>
</article>

<!-- 卡片页脚（功能操作区） -->
<div class="product-card">
  <h3>智能音箱</h3>
  <p>支持语音交互、音乐播放...</p>

  <footer class="card-footer" aria-label="商品操作区">
    <button>加入购物车</button>
    <button>立即购买</button>
  </footer>
</div>
```

3. 元素

- <footer>
  - 【语法结构】
    - <footer> 作为块级元素，默认情况下会占满父容器的宽度。通常，它会被嵌套于 <body>、<main>、<article>、<section> 等元素的末尾，作为这些父容器的一部分。其内部可容纳文本、链接、列表、图标等内容。
      - <footer>可作为 <body>、<main>、<article>、<section>、 等元素的子元素，一个文档中允许包含多个 <footer>。
      - <footer>通常位于父容器的最底部，例如作为<body>的最后一个子元素，但在语义上其优先级高于位置，即便是调整了位置，仍可使用 <footer>进行标记。
      - <footer>内的内容需与所在父容器高度相关，例如页面页脚可包含网站信息，文章页脚可包含文章元信息。
  - 【核心属性】
    - <footer>元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<!-- 页面级页脚（网站底部） -->
<body>
  <header>页面头部</header>
  <nav>主导航</nav>
  <main>主内容...</main>

  <!-- 页面页脚 -->
  <footer class="site-footer" aria-label="网站页脚">
    <div class="footer-container">
      <div class="copyright">
        © 2025 腾讯元宝. 保留所有权利 | 备案号：粤ICP备12345678号
      </div>
      <div class="footer-links">
        <a href="/about">关于我们</a>
        <a href="/privacy">隐私政策</a>
        <a href="/contact">联系我们</a>
      </div>
      <div class="social-media">
        <a href="https://weibo.com" aria-label="微博"><svg>...</svg></a>
        <a href="https://wechat.com" aria-label="微信"><svg>...</svg></a>
      </div>
    </div>
  </footer>
</body>

<!-- 文章页脚（文章元信息） -->
<article>
  <h1>HTML5 语义化指南</h1>
  <p>正文内容...</p>

  <footer class="article-footer" aria-label="文章页脚">
    <p class="meta">发布时间：2025-09-18 · 阅读时长：10分钟 · 作者：张三</p>
    <div class="article-tags">
      <a href="/tag/html5">HTML5</a>
      <a href="/tag/semantic">语义化</a>
    </div>
  </footer>
</article>

<!-- 卡片页脚（功能操作区） -->
<div class="product-card">
  <h3>智能音箱</h3>
  <p>支持语音交互、音乐播放...</p>

  <footer class="card-footer" aria-label="商品操作区">
    <button>加入购物车</button>
    <button>立即购买</button>
  </footer>
</div>
```


案例演示

- 案例：语义化结构页面
  - 构建一个个人技术博客首页，实践HTML结构元素的核心用法。该案例覆盖文档基础结构、元信息元素与语义化结构元素的综合应用，帮助学习者掌握“从0到1”搭建规范页面框架的能力。
  - 【案例目标】
    - 熟练使用文档基础结构元素（<!DOCTYPE>、<html>、<head>、<body>）；
    - 掌握元信息元素（<meta>、<title>）的配置方法；
    - 理解并应用语义化结构元素（<header>、<nav>、<main>、<article>、<aside>、<footer>）；
    - 避免常见结构错误（如滥用 、标题层级混乱）。

3. 元素

- 文本与列表元素作为HTML中最基础且常用的内容填充工具，肩负着将离散信息整合为结构化文本内容的重任。无论是博客文章、产品说明，还是新闻资讯，文本与列表元素都是信息传递的核心载体。
- 标题
  - <h1>-<h6>
    - 【元素简介】
      - <h1>至 <h6>作为 HTML 中的标题元素（Heading Elements），用于构建文档或内容的层次化标题体系。这些元素利用数字 1-6 来表示标题的重要性级别（<h1>最为重要，<h6>相对次要），是 HTML 语义化的重要组成部分，其核心价值主要体现在以下方面：
      - 样式控制：可以使用 CSS 为不同级别的标题设置样式，如字体大小、颜色和边距等，以统一页面的视觉标准。
      - 语义清晰：通过层级关系明确内容的逻辑结构，如“主标题→子标题→细节”，有助于浏览器、搜索引擎（SEO）和辅助技术（如屏幕阅读器）理解内容的重要程度。
      - SEO 优化：搜索引擎通过标题层级识别页面的核心主题，其中 <h1>通常被视为页面主关键词的载体，从而提高内容相关性的抓取权重。
      - 无障碍适配：屏幕阅读器（如 NVDA、VoiceOver）支持根据标题层级进行导航，用户能够快速跳转到目标章节，这对视力障碍用户尤为重要。
      - 样式基础：标题元素默认具有浏览器内置样式，如字体大小、加粗效果等，是页面排版的基础结构。

3. 元素

- 标题
  - <h1>-<h6>
    - 【语法结构】
      - <h1>至 <h6>属于块级元素（默认情况下会占满父容器的宽度），通常会作为父容器嵌套于内容区域（例如 <article>、<section>、<main>）之中，用于标记内容的标题或关键节点。
        - 标题级别应从 <h1>开始，按照逐级递减的顺序排列，如 <h1>→ <h2>→ <h3>，严禁出现跳级情况，例如从 <h1>直接过渡到 <h3>。
        - 建议每个页面仅设置一个 <h1>标题。虽然 HTML5 支持多个 <h1>，但从 SEO 和无障碍访问的最佳实践来看，推荐仅使用一个 <h1>标题，以此作为页面核心主题的标识。
        - 标题内容需简洁明了，精准体现其下方内容的核心要点，如 <h2>标题应概括后续段落的主题。
    - 【核心属性】
      - <h1> 至 <h6> 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性，无专属属性。

```
<!-- 页面主标题（最重要） -->
<h1>HTML5 语义化完全指南</h1>

<!-- 子标题（次重要，通常对应主内容的章节） -->
<h2>什么是语义化？</h2>
<p>语义化是指通过 HTML 元素明确内容的含义...</p>

<h2>为什么需要语义化？</h2>
<p>语义化能提升 SEO、无障碍访问和代码可维护性...</p>

<!-- 子子标题（三级标题，细化章节内容） -->
<h3>语义化的核心原则</h3>
<ul>
  <li>使用正确的元素描述内容（如 `&lt;article&gt;` 表示文章）</li>
  <li>避免滥用标题元素（如用 `&lt;h1&gt;` 仅为了大字体）</li>
</ul>

<!-- 四级标题（更细粒度的内容划分） -->
<h4>示例：正确的标题层级</h4>
<p>...</p>
```

3. 元素

- 标题
  - <h1>-<h6>
    - 【使用建议】
      - 标题级别应严格遵循从高到低的顺序依次递减（例如，从 <h1>到 <h2>再到 <h3>），严禁跳过中间级别（如从 <h1>直接过渡到 <h3>）。
      - 建议每个页面仅设置一个 <h1>元素（虽然 HTML5 允许存在多个，但从 SEO 最佳实践角度出发，一个 <h1>元素更为适宜），该元素用于标识页面的核心主题，如文章标题或产品名称。
      - 标题内容需简洁明了，精准反映其下方内容的核心要点，避免冗长繁杂或表意模糊。
      - 禁止单纯为实现 CSS 样式效果（如大字体、加粗）而使用标题元素，可采用 、元素或 CSS 类来达成相应样式。
      - 标题元素可与 <article>、<section>等语义化元素协同使用，以清晰界定内容的层次结构。

```
<article>
  <h1>文章标题</h1>
  <section>
    <h2>章节标题</h2>
    <p>正文内容...</p>
  </section>
</article>
```


3. 元素

- 段落与强调
  - <p>
    - 【元素简介】
      - <p>作为 HTML 中的段落元素（Paragraph Element），主要用于界定文档或内容中的独立文本段落（Paragraph）。它属于块级元素（Block-Level Element），在页面中默认换行显示（会自动在前后添加换行符），是文本内容最基础的容器之一，其核心价值如下：
      - 语义明确：利用 <p>标记文本内容的“段落边界”，有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）准确理解文本结构，例如清晰区分“这是第一段”“这是第二段”。
      - 排版基础：其默认的块级特性使其成为页面文本布局的重要工具，常用于文章正文、说明文字等的排版。
      - 无障碍适配性良好：屏幕阅读器（如 NVDA、VoiceOver）能够通过 <p>识别段落分隔，便于用户通过导航快速跳转至目标段落。

3. 元素

- 段落与强调
  -
    - 【语法结构】
      - 作为块级元素，常作为父容器嵌套于内容区域（例如 <article>、<section>、<main>）之中，其内部可容纳文本内容、行内元素（像 、<a>、）或其他块级元素（如 、<ol>）。
      - 元素会依据浏览器的默认样式，自动在前后添加换行符，故而多个连续的 元素会呈垂直排列。
      - 内容限制：
        - 建议包含文本内容或行内元素（例如 、<a>）。
        - 尽管语法层面允许嵌套块级元素（如 、 ），但不建议这么做，因为这会破坏段落的语义，进而导致布局混乱。
      - 不建议创建无内容的空 元素（如 ），此类操作会造成语义浪费，并对无障碍体验产生负面影响。
    - 【核心属性】
      - 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），无专属属性。

```
<!-- 基础段落（纯文本） -->
<p>这是一段普通的文本内容，用于描述某个主题或观点。</p>

<!-- 包含行内元素的段落 -->
<p>这是一段包含<a href="/link">超链接</a>、<strong>加粗文本</strong>和<em>斜体文本</em>的段落。</p>

<!-- 包含块级元素的段落（不推荐，但语法允许） -->
<p>这是一段包含列表的段落：
  <ul>
    <li>列表项 1</li>
    <li>列表项 2</li>
  </ul>
</p>
```


3. 元素

- 段落与强调
  -
    - 【使用建议】
      - 语义正确性：仅用于标记独立的文本段落，如文章正文、说明文字、评论内容等，不适用于布局容器，例如包裹图片、按钮等非文本内容的容器。
      - 内容完整性：段落内容应完整且具有明确意义，避免出现碎片化文本。
      - 避免空段落：不建议创建无实际内容的空 元素（如 ），此类空元素会干扰屏幕阅读器的导航逻辑，且对 SEO 毫无价值。
      - 合理嵌套：仅允许嵌套行内元素，如 、<a>、 等，不建议嵌套块级元素，如 、 等，否则会破坏段落的语义结构。
      - 与其他语义化元素的配合：段落通常需与标题（<h1>-<h6>）、列表（<ul>、<ol>）、引用（<blockquote>）等元素协同使用，以明确内容结构。

```
<article>
  <h1>HTML5 语义化指南</h1>
  <p>语义化是 HTML 的核心特性之一...</p>
  <h2>核心元素</h2>
  <ul>
    <li><code>&lt;article&gt;</code>：独立内容块</li>
    <li><code>&lt;section&gt;</code>：主题分区</li>
  </ul>
</article>
```

3. 元素

- 段落与强调
  - 、
    - 【元素简介】
      - 和 是 HTML 中的语义化行内元素，用于通过语义而非样式标记文本的重要性或强调程度。它们的核心价值在于：
      - 语义明确：表示内容的“重要性”（如关键结论、警告信息）；表示内容的“强调”（如情感表达、重点突出）。
      - 无障碍友好：屏幕阅读器（如 NVDA、VoiceOver）会根据元素语义调整朗读方式（如 读作“重要”，读作“强调”）。
      - 样式灵活：默认样式（加粗、斜体）可通过 CSS 自定义，避免依赖浏览器默认样式。

3. 元素

- 段落与强调
  - 、
    - 【语法结构】
      - 和 均为行内元素（Inline Element），默认不换行，通常嵌套在块级元素（如 、 、<h1>-<h6>）中，用于标记文本中的关键或强调部分。
        - 不会强制换行，可与文本、其他行内元素（如 <a>、）混合使用。
        - 的语义强度高于 （“重要”>“强调”）。
        - 禁止仅为了加粗或斜体样式使用 或 （应通过 CSS 类实现）。不建议创建无内容的空 元素（如 ），此类操作会造成语义浪费，并对无障碍体验产生负面影响。
    - 【核心属性】
      - 和 继承了 HTML 的全局属性（适用于所有元素的通用属性），同时可通过 ARIA（无障碍富互联网应用）属性增强可访问性，无专属属性。

```
1. <!-- 段落中的重要内容（strong） -->
2. <p>请务必注意：<strong>此操作将永久删除数据，无法恢复！</strong></p>

3. <!-- 强调情感或重点（em） -->
4. <p>我<u>非常</u>喜欢 HTML5 的语义化元素，尤其是 <em>strong</em> 和
   <em>em</em>！</p>

5. <!-- 嵌套使用（合理场景） -->
6. <p>警告：<strong>未保存的修改 <em>可能</em> 丢失</strong>，请及时提交。</p>
```

3. 元素

- 链接
  - <a>
    - 【元素简介】
      - <a>元素，即 HTML 中的超链接元素（Anchor Element），主要用于创建可点击的链接。这些链接既可以指向页面内的其他位置（即锚点），也可以指向外部资源，如网页、文件、邮件地址等。作为万维网（Web）的核心元素之一，它通过 href 属性明确目标资源的位置，使用户能够通过点击操作在不同资源之间实现跳转。其核心价值主要体现在以下几个方面：
      - 资源连接：该元素可实现页面间的导航、文件下载以及邮件发送等功能，是 Web 信息交互的基础支撑。
      - 语义明确：借助 href 属性，能够精准定位目标资源（例如 https://example.com），有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）准确理解链接的用途。
      - 无障碍友好：屏幕阅读器（如 NVDA、VoiceOver）会识别 <a>元素并朗读其内容，方便用户通过导航快速进行跳转操作。

3. 元素

- 链接
  - <a>
    - 【语法结构】
      - <a>作为行内元素（Inline Element），默认状态下不会换行（前后无换行符），一般嵌套于块级元素（如 <p>、 、<h1> - <h6>）之内，其内部能够包含文本内容或其他行内元素（例如 、<img>）。
        - 具备不强制换行的特性，可与文本及其他行内元素（如 <img>、）混合使用。
        - href（目标资源 URL）作为 <a>元素的核心属性，若 <a>元素缺失 href属性，则该元素不具备实际功能，此类链接被称为“空链接”。
        - 其内部可容纳任意行内内容，如文本、图标、图片等，但建议内容保持简洁，避免采用复杂布局。

```
<!-- 基础链接（指向外部页面） -->
<a href="https://www.tencent.com" target="_blank">访问腾讯官网</a>

<!-- 页面内锚点链接（跳转到当前页面的某个位置） -->
<a href="#section1">跳转到第一章</a>
<!-- 页面下方 -->
<div id="section1">第一章内容...</div>

<!-- 下载链接（指向文件） -->
<a href="/report.pdf" download>下载报告（PDF）</a>

<!-- 邮件链接（触发邮件客户端） -->
<a href="mailto:contact@example.com">发送邮件</a>

<!-- 包含行内元素的链接 -->
<a href="https://example.com" class="btn">
  <img src="arrow.svg" alt="箭头图标"> 立即访问
</a>
```

3. 元素

3.1 结构元素：搭建页面骨架

- 链接
  - <a>
    - 【核心属性】
      - <a>继承了 HTML 的全局属性（适用于所有元素的通用属性），同时可通过 ARIA（无障碍富互联网应用）属性增强可访问性。除此之外，<a>还元素具备多种属性，可用于精准控制链接的行为、样式及语义，以下为最为常用的核心属性。

属性	说明	示例
href	必需：指定目标资源的 URL（绝对路径或相对路径）。	href="https://example.com"（绝对路径） href="page.html"（相对路径）
target	控制链接的打开方式（默认 _self，在当前窗口打开）。	target="_blank"（新窗口打开） target="_parent"（父窗口打开）
rel	定义当前页面与目标资源的关系类型（影响 SEO 和安全性）。	rel="noopener"（新窗口打开时禁止访问原页面 window.opener，提升安全） rel="nofollow"（告知搜索引擎不跟踪此链接，用于外部广告）
title	为链接添加提示信息（鼠标悬停时显示，提升无障碍体验）。	title="点击访问腾讯官网"
download	（可选）指示浏览器下载资源而非直接打开（仅对文件有效）。	href="/file.zip" download（触发下载）

3. 元素

3.2 文本与列表元素：填充内容基础

- 链接
  - <a>
    - 【使用建议】
      - 优先选用具备实际意义的 href，当 <a> 元素未设置 href 属性或者 href="#"，仅用于触发交互行为（例如触发 JS 事件）时，建议添加 role="button" 并绑定相应事件，防止屏幕阅读器误将其识别为链接。
        - 错误示例：打开弹窗（无 href，屏幕阅读器可能无法准确识别）。
        - 正确示例：打开弹窗（通过 role 和 aria-label 明确语义）。
      - 优化 target 和 rel 属性，使用 target="_blank" 时，务必添加 rel="noopener" 或 rel="noreferrer"，以避免目标页面通过 window.opener 访问原页面，消除安全隐患。
        - 示例：新窗口打开。
        - 针对指向非本站的链接，建议添加 rel="nofollow"，以告知搜索引擎无需跟踪此链接，防止权重流失。
      - 增强无障碍体验
        - 链接内容应简洁确切，避免使用模糊表述（如“点击这里”），应直接说明目标（如“查看用户协议”）。
        - 错误示例：点击这里。
        - 正确示例：查看用户协议。
        - 当链接内容为图标或抽象符号时，利用 aria-label 进行补充说明（如“返回顶部”）。
        - 示例：↑。
    - 合理运用锚点链接，使用 href="#id" 指向当前页面的特定元素（需为目标元素添加 id 属性），以优化长页面的导航体验。

```
<a href="#footer">跳转到页脚</a>
<!-- 页面下方 -->
<footer id="footer">页脚内容...</footer>
```

3. 元素

- 链接
  - <a>
    - 【使用建议】
      - 下载链接优化，对于指向文件（如 PDF、文档）的链接，建议添加 download 属性，以触发浏览器下载操作，而非直接打开文件。
        - 示例：下载报告（PDF）
      - 除非确有必要（如触发 JS 交互），否则不建议使用 href="javascript:void(0);"，因为这可能引发 SEO 问题和无障碍访问障碍。建议优先通过 onclick 绑定事件，并添加 role="button" 和 aria-label。
      - 链接样式与交互
        - 借助 CSS 为链接添加悬停（:hover）和聚焦（:focus）状态效果（如颜色变化、下划线），增强交互的可感知性。

3. 元素

- 列表
  -
    - 【元素简介】
      - 作为 HTML 里的无序列表元素（Unordered List Element），主要用于定义项目符号列表（Bullet List）。其核心功能是利用项目符号（如圆点、方块、空心圆等）对一组并列且无需排序的内容项（诸如选项、分类、特征等）加以标记，是网页中极为常见的列表结构之一。该元素的核心价值主要体现在以下方面：
      - 语义明确：使用 元素标记无序列表，有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）精准识别内容“并列无序”的属性。
      - 结构清晰：与传统采用无意义的 元素模拟列表的做法相比，运用 元素能让代码更符合语义化规范，从而提升代码的可维护性。
      - 样式基础：默认的项目符号样式（如圆点）为页面提供了基础的排版样式，并且能够通过 CSS 进行自定义调整。

3. 元素

- 列表
  -
    - 【语法结构】
      - 作为块级元素（Block - Level Element），默认状态下会占据父容器的整个宽度。此元素内部仅允许包含 <li> 子元素（即列表项），并且支持嵌套其他列表（例如 <ul> 或 ）或行内元素（如 、<a>）。
        - 的直接子元素只能是 （列表项），禁止直接嵌套其他块级元素（如 、<p>）。
        - 内部可包含文本、行内元素（如 、<a>）或嵌套列表（如 、<ol>）。
        - 浏览器默认为 添加项目符号（如圆点），通过 list-style-typeCSS 属性可修改。
    - 【核心属性】
      - 继承了 HTML 的全局属性（适用于所有元素的通用属性），但并无专属属性。

```
<!-- 基础无序列表 -->
<ul>
  <li>列表项 1</li>
  <li>列表项 2</li>
  <li>列表项 3</li>
</ul>

<!-- 包含行内元素的列表项 -->
<ul>
  <li>水果：<span class="fruit">苹果</span>、<span class="fruit">香蕉</span></li>
  <li>蔬菜：<a href="/vegetables">查看详情</a></li>
</ul>

<!-- 嵌套无序列表（子列表） -->
<ul>
  <li>前端技术：
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </li>
  <li>后端技术：
    <ol>
      <li>Java</li>
      <li>Python</li>
    </ol>
  </li>
</ul>
```


3. 元素

- 列表
  -
    - 【使用建议】
      - 仅用 作为直接子元素，<ul>的直接子元素必须是 （列表项），禁止直接嵌套其他块级元素（如 、<p>）。若需包含复杂内容，可将内容包裹在 内。

```
<ul>
  <div>错误：直接使用 div 作为子元素</div> <!-- 违反语义 -->
</ul>
<ul>
  <li><div>正确：li 包裹 div</div></li> <!-- li 作为直接子元素 -->
</ul>
```

      - 避免空列表或无效列表项
        - 禁止创建无内容的空 或 （如 或 ），会干扰屏幕阅读器的导航逻辑。
        - 若列表项内容为空（如占位），建议添加 aria-hidden="true"或通过 CSS 隐藏。

```
<ul>
  <li aria-hidden="true"></li> <!-- 屏幕阅读器忽略 -->
</ul>
```

3. 元素

- 列表
  -
    - 【使用建议】
      - 语义化嵌套列表，嵌套列表（如子列表）需保持逻辑清晰，父列表项与子列表项需有明确关联（如“分类→子分类”）。

```
<ul>
  <li>前端技术：
    <ul>
      <li>HTML</li>
      <li>CSS</li>
    </ul>
  </li>
</ul>
```

      - 避免滥用列表，仅当内容为并列且无顺序的项时使用 （如选项、分类），若内容为段落或长文本，建议使用 <p>或其他块级元素。

3. 元素

- 列表
  -
    - 【元素简介】
      - 作为 HTML 中的有序列表元素（Ordered List Element），主要用于定义编号列表（Numbered List）。其关键作用在于，通过顺序编号（如 1.、2.、3. 或 a.、b.、c. 等）对一组具有逻辑顺序的内容项（如步骤、排名、操作指南等）进行标记，是页面中极为常见的有序内容结构之一。其核心价值主要体现在以下方面：
      - 语义明确：使用 标记有序列表，有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）精准识别内容的“顺序性”与“逻辑层级”。
      - 结构清晰：与传统使用无意义的 模拟列表的方式相比，<ol>使代码更符合语义化规范，进而提升代码的可维护性。
      - 样式基础：默认的编号样式（如阿拉伯数字）为页面提供了基础排版，且可通过 CSS 进行自定义调整。

3. 元素

-
  - 【语法结构】
    - 是块级元素（Block-Level Element），默认占满父容器宽度，内部只能包含 子元素（列表项），支持嵌套其他列表（如 或 ）或行内元素（如 、<a>）。
      - 的直接子元素只能是 （列表项），禁止直接嵌套其他块级元素（如 、<p>）。
      - 内部可包含文本、行内元素（如 、<a>）或嵌套列表（如 、<ol>）。
      - 浏览器默认使用阿拉伯数字（1.、2.、3.）编号，通过 CSS 可修改为字母（a.、b.、c.）或罗马数字（Ⅰ、Ⅱ、Ⅲ）。

```
<!-- 基础有序列表（默认阿拉伯数字编号） -->
<ol>
  <li>第一步：准备材料</li>
  <li>第二步：烹饪食材</li>
  <li>第三步：装盘装饰</li>
</ol>

<!-- 包含行内元素的列表项 -->
<ol>
  <li>水果：<span class="fruit">苹果</span>（价格：5元）</li>
  <li>蔬菜：<a href="/vegetables">查看详情</a></li>
</ol>

<!-- 嵌套有序列表（子列表） -->
<ol>
  <li>前端技术：
    <ol>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ol>
  </li>
  <li>后端技术：
    <ul>
      <li>Java</li>
      <li>Python</li>
    </ul>
  </li>
</ol>

<!-- 自定义起始编号（HTML5 支持） -->
<ol start="5">
  <li>第五项内容</li>
  <li>第六项内容</li>
</ol>
```


3. 元素

- 列表
  -
    - 【核心属性】
      - 继承了 HTML 的全局属性（适用于所有元素的通用属性），同时支持专有属性。

属性	说明	示例
start	定义列表的起始编号（仅对阿拉伯数字有效，HTML5 仍支持）。	<ol start="3">... （从 3 开始编号）

    - 【使用建议】
      - 仅用 作为直接子元素，<ol>的直接子元素必须是 （列表项），禁止直接嵌套其他块级元素（如 、<p>）。若需包含复杂内容，可将内容包裹在 内。
      - 避免空列表或无效列表项
        - 禁止创建无内容的空 或 （如 或 ），会干扰屏幕阅读器的导航逻辑。
        - 若列表项内容为空（如占位），建议添加 aria-hidden="true"或通过 CSS 隐藏。
      - 语义化使用有序列表，仅当内容项有明确顺序要求时使用 （如步骤、排名、操作指南）。

3. 元素

- 列表
  - <dl>
    - 【元素简介】
      - <dl>作为 HTML 中的定义列表元素（Definition List Element），用于明确术语与其解释之间的关联关系（Term - Definition Pair）。其关键作用在于以结构化形式呈现“术语→定义”的映射关系，在词典、术语表以及产品规格说明等场景中应用广泛。它的核心价值主要体现在以下方面：
      - 核心价值：
      - 语义清晰：借助 <dl>标记术语与定义的关系，有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）精准理解内容的“解释性”逻辑。
      - 结构合理：摒弃传统使用无意义的 模拟列表的方式，使代码更贴合语义化规范，增强可维护性。
      - 无障碍体验佳：屏幕阅读器（如 NVDA、VoiceOver）能够识别 <dl>中术语与定义的关系，方便用户通过导航实现快速跳转。
      - 利于 SEO 优化：搜索引擎可通过 <dl>识别术语与解释的关联，提高专业内容相关性的抓取权重。

3. 元素

- 列表
  - <dl>
    - 【语法结构】
      - <dl>是块级元素（Block-Level Element），默认占满父容器宽度，内部由至少一个术语（<dt>）和其对应的定义（<dd>）组成。支持嵌套其他元素（如 <p>、<ul>）或行内元素（如 、<a>）。
        - <dt>（术语）与 <dd>（定义）应一一对应，单个 <dt>可对应多个 <dd>，或多个 <dt>共享一个 <dd>，但逻辑关系需合理。
        - <dl>的直接子元素仅允许为 <dt>或 <dd>，严禁直接嵌套其他块级元素，例如 、<p>。
        - 浏览器默认无特殊样式，需借助 CSS 自定义术语与定义的间距、字体等样式。
    - 【核心属性】
      - <dl>继承了 HTML 的全局属性（适用于所有元素的通用属性），但无专属属性。

```
<!-- 基础定义列表（术语+定义） -->
<dl>
  <dt>HTML</dt>
  <dd>超文本标记语言（HyperText Markup Language），用于描述网页内容的结构和语义。</dd>

  <dt>CSS</dt>
  <dd>层叠样式表（Cascading Style Sheets），用于控制网页的布局和视觉表现。</dd>
</dl>

<!-- 包含复杂内容的定义项 -->
<dl>
  <dt>语义化元素</dt>
  <dd>
    <p>通过 HTML 元素明确内容含义的编码实践，例如：</p>
    <ul>
      <li><code>&lt;article&gt;</code>：独立内容块</li>
      <li><code>&lt;section&gt;</code>：主题分区</li>
    </ul>
  </dd>
</dl>

<!-- 多术语共享一个定义（不推荐，但语法允许） -->
<dl>
  <dt>JavaScript</dt>
  <dt>JS</dt>
  <dd>一种轻量级的解释型编程语言，主要用于网页交互。</dd>
</dl>
```

3. 元素

- 列表
  - <dl>
    - 【使用建议】
      - 严格配对 <dt>与 <dd>，每个 <dt>（术语）应对应至少一个 <dd>（定义），避免“无定义的术语”或“无术语的定义”。

```
<dl>
  <dt>HTML</dt>
  <dd>超文本标记语言...</dd> <!-- 正确：术语+定义 -->
</dl>

<dl>
  <dt>HTML</dt> <!-- 错误：无对应的 dd -->
  <dd>CSS</dd> <!-- 错误：无对应的 dt -->
</dl>
```

      - 避免空列表或无效项
        - 禁止创建无内容的空 <dl>、<dt>或 <dd>（如 <dl></dl>或 <dl><dt></dt></dl>），会干扰屏幕阅读器的导航逻辑。
        - 若定义内容为空（如占位），建议添加 aria-hidden="true"或通过 CSS 隐藏。
      - 语义化嵌套内容，<dd>内部可包含文本、段落（<p>）、列表（<ul>、<ol>）或其他块级元素，用于详细解释术语。

```
<dl>
  <dt>响应式设计</dt>
  <dd>
    <p>一种网页设计方法，使页面能根据设备屏幕尺寸自动调整布局：</p>
    <ul>
      <li>使用弹性布局（Flexbox）</li>
      <li>媒体查询（Media Queries）适配不同屏幕</li>
    </ul>
  </dd>
</dl>
```


3. 元素

3.2 文本与列表元素：填充内容基础

- 列表
  - <dl>
    - 【使用建议】
      - 区分 <dl>与其他列表，<dl>用于术语与定义的关联（如“HTML→超文本标记语言”），而 /<ol>用于无序/有序项目列表（如“步骤1→步骤2”）。

```
<!-- 定义列表（术语+定义） -->
<dl>
  <dt>JavaScript</dt>
  <dd>网页交互编程语言...</dd>
</dl>

<!-- 无序列表（项目列表） -->
<ul>
  <li>步骤1：准备材料</li>
  <li>步骤2：烹饪食材</li>
</ul>
```

3. 元素

- 列表
  - <dl>
    - 【使用建议】
      - 区分 <dl>与其他列表，<dl>用于术语与定义的关联（如“HTML→超文本标记语言”），而 /<ol>用于无序/有序项目列表（如“步骤1→步骤2”）。

```
<!-- 定义列表（术语+定义） -->
<dl>
  <dt>JavaScript</dt>
  <dd>网页交互编程语言...</dd>
</dl>

<!-- 无序列表（项目列表） -->
<ul>
  <li>步骤1：准备材料</li>
  <li>步骤2：烹饪食材</li>
</ul>
```


案例演示

- 案例：前端学习入门指南
  - 通过构建一个“前端学习入门指南”页面，实践HTML文本与列表元素的核心用法。该案例覆盖标题层级、段落、强调、链接、无序列表、有序列表、定义列表等元素的综合应用，帮助学习者掌握“如何用文本与列表清晰传递信息”的核心技能。
  - 【案例目标】
    - 熟练使用标题元素（<h1>-<h6>）定义内容层级；
    - 掌握段落（<p>）与强调元素（、）的内容组织；
    - 理解链接（<a>）的跳转逻辑与最佳实践；
    - 学会用列表元素（<ul>、<ol>、<dl>）结构化展示信息；
    - 避免常见错误（如标题层级混乱、列表项无意义）。

3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 在HTML中，图像与多媒体元素是提升页面表现力的核心工具。它们通过整合视觉与听觉信息，能够显著增强用户对内容的理解，优化交互体验。无论是用于产品展示、新闻资讯呈现，还是教育类页面搭建，图像与多媒体元素均为传递信息的关键载体。
- 图像
  - <img>
    - 【元素简介】
      - <img>作为 HTML 中的图像嵌入元素（Image Element），主要用于在网页中展示图片。它属于替换元素（Replaced Element），由浏览器依据 src 属性加载外部图像资源进行渲染，本身并不包含文本内容，不过可借助 alt 属性提供替代信息，其核心价值体现在以下方面。
      - 视觉呈现：借助图片（如产品图、插图、图表）传递信息，增强内容的吸引力与可读性。
      - 用户体验：通过增强页面交互性（如按钮图标、装饰元素），降低用户对文字的理解成本。
      - 无障碍支持：利用 alt 属性为视障用户（屏幕阅读器）提供图像描述，以符合 WCAG 标准。
      - SEO 优化：搜索引擎可通过 alt 文本理解图片内容，进而提升页面的相关性排名。

3. 元素

3.2 文本与列表元素：填充内容基础

- 图像
  - <img>
    - 【语法结构】
      - <img>为自闭合元素（无需闭合符号），属于行内块元素（Inline-Block），默认呈“基线对齐”显示，可能与文本底部存在间隙。该元素的核心依赖于 src（图像路径）和 alt（替代文本）属性，且支持嵌套于块级元素或行内元素之中。
        - 必选属性包含 src（图像 URL）以及 alt（替代文本，在 HTML5 中为可选属性，但强烈建议填写）。
        - 具备行内块特性，默认与文本基线对齐，底部可能出现间隙，可通过 display: block 或 vertical-align: top 消除。
        - 浏览器会并行加载图像，然而过多未经优化的图片会对页面性能产生影响，需结合懒加载、压缩等技术加以处理。

```
<!-- 基础图像（必选属性 src 和 alt） -->
<img src="logo.png" alt="公司logo">

<!-- 指定宽高（单位：像素或百分比） -->
<img src="banner.jpg" alt="活动横幅" width="1200" height="400">

<!-- 鼠标悬停提示（title 属性） -->
<img src="product.jpg" alt="智能手表" title="点击查看详情">

<!-- 响应式图像（srcset + sizes） -->
<img
  src="small.jpg"
  srcset="small.jpg 480w, medium.jpg 768w, large.jpg 1200w"
  sizes="(max-width: 600px) 480px, (max-width: 1000px) 768px, 1200px"
  alt="响应式风景图"
>

<!-- 跨域图像（CORS 设置） -->
<img src="https://other-domain.com/image.jpg" crossorigin="anonymous"
  alt="跨域图片">

<!-- 图像映射（关联 <map> 元素） -->
<img
  src="world-map.jpg"
  alt="世界地图"
  usemap="#country-map"
>
<map name="country-map">
  <area shape="rect" coords="0,0,100,100" href="/asia" alt="亚洲">
</map>
```

3. 元素

- 图像
  - <img>
    - 【核心属性】
      - <img> 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。同时，<img>支持多种专有属性，用于控制图像的显示、加载行为和语义。
      - 必选属性

属性	说明	示例	注意事项
src	必需：指定图像资源的 URL（绝对路径或相对路径）。	src="images/photo.jpg"	若路径错误，图像无法加载，alt文本会显示（或空白）。
alt	推荐：为图像提供替代文本（当图像无法加载或视障用户访问时显示）。	alt="红色跑车在公路上行驶"	避免空 alt=""（除非图像仅为装饰），否则屏幕阅读器会忽略。

      - 显示控制属性

属性	说明	示例	注意事项
width	定义图像的显示宽度（单位：像素 px或百分比 %）。	width="600"（600px） width="50%"（父容器宽度的50%）	仅改变显示尺寸，不影响原始文件大小；建议与height同时设置以避免布局抖动。
height	定义图像的显示高度（单位：像素 px或百分比 %）。	height="400"（400px）	同上。
style	通过 CSS 自定义图像样式（如边框、圆角、透明度）。	style="border-radius: 8px; opacity: 0.8;"	推荐用 CSS 类统一管理样式，避免内联样式冗余。

3. 元素

3.2 文本与列表元素：填充内容基础

- 图像
  - <img>
    - 【核心属性】
      - 交互与提示属性

属性	说明	示例	注意事项
title	定义鼠标悬停时的提示文本（气泡框）。	title="点击下载高清图"	仅在鼠标悬停时显示，对屏幕阅读器支持有限，不可替代 alt。
loading	控制图像的懒加载行为（HTML5 新增）。	loading="lazy"（懒加载） loading="eager"（立即加载）	浏览器兼容性：Chrome 77+、Firefox 75+、Edge 79+；默认值 auto（自动判断）。

      - 响应式图像属性

属性	说明	示例	注意事项
srcset	定义多组图像源及其像素密度（用于响应式适配不同屏幕）。	srcset="small.jpg 480w, medium.jpg 768w, large.jpg 1200w"	480w表示图像的固有宽度为 480px；浏览器根据屏幕宽度选择最合适的图像。
sizes	定义图像在不同屏幕宽度下的显示尺寸（配合 srcset使用）。	sizes="(max-width: 600px) 480px, 800px"	描述图像在视口中的显示宽度，帮助浏览器选择最优 srcset图像。

3. 元素

3.2 文本与列表元素：填充内容基础

- 图像
  - <img>
    - 【核心属性】
      - 高级属性

属性	说明	示例	注意事项
crossorigin	控制跨域图像的 CORS（跨域资源共享）设置（用于画布操作等场景）。	crossorigin="anonymous"（匿名请求） crossorigin="use-credentials"（携带 Cookie）	若图像来自第三方域且需操作画布（如滤镜），需设置此属性，否则画布会被污染。
usemap	关联 <map>元素创建图像映射（可点击区域）。	usemap="#image-map"	<map>元素需定义 name属性（值为usemap的值去掉 #）。
decoding	控制图像解码方式（优化渲染性能，HTML5新增）。	decoding="async"（异步解码） decoding="sync"（同步解码）	适用于大图像，避免阻塞主线程；默认值auto（浏览器自动选择）。

3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 图像
  - <img>
    - 【使用建议】
      - 始终提供有意义的 alt文本，alt应准确描述图像内容或功能（如“用户个人资料头像”“产品价格对比图”），避免模糊表述（如“图片1”），

```
<!-- 差，过于笼统 -->
<img src="logo.png" alt="logo">
<!-- 好，明确信息 -->
<img src="logo.png" alt="XX科技公司官方LOGO">
```

      - 优化图像加载性能
        - 对非首屏图像使用 loading="lazy"，延迟加载直到用户滚动到附近（提升首屏加载速度）。

```
<img src="below-fold.jpg" alt="底部图片" loading="lazy">
```

        - 通过 srcset和 sizes为不同屏幕提供适配尺寸（避免大图像在小屏幕上浪费带宽）。

```
<img
  src="small.jpg"
  srcset="small.jpg 480w, medium.jpg 768w, large.jpg 1200w"
  sizes="(max-width: 600px) 480px, 800px"
  alt="响应式风景图"
>
```

3. 元素

- 图像
  - <img>
    - 【使用建议】
      - 使用工具（如 ImageOptim、Squoosh）压缩图像，优先选择现代格式（WebP、AVIF）替代 PNG/JPEG（相同质量下体积更小）。
      - 避免布局抖动（Layout Shift），通过 width和 height属性声明图像的固有尺寸（或通过 CSS 预留空间），避免图像加载后因尺寸变化导致页面布局偏移。

```
.image-container {
  width: 600px;
  height: 400px;
  background: #f0f0f0; /* 加载前占位 */
}
.image-container img {
  width: 100%;
  height: 100%;
  object-fit: cover; /* 控制图像填充方式 */
}
```

      - 提升无障碍体验，确保 alt文本对视障用户有意义（如“图表显示2023年销售额增长20%”比“图表”更实用），title仅在鼠标悬停时显示，且部分屏幕阅读器可能忽略，不可替代 alt。
      - 图像映射的合理使用，仅当需要为图像的不同区域添加链接（如地图热点、示意图标注）时使用 <map>和 usemap，避免滥用（普通链接更易维护）。
      - 响应式设计的适配，通过 sizes属性结合媒体查询，为小屏幕提供更小的图像（如手机端使用 480px 宽的图片），减少数据流量消耗。

3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 视频与音频元素：<video>
  - 【元素简介】
    - <video>作为 HTML 里的视频嵌入元素（Video Element），主要用于在网页中实现视频文件的播放功能。它属于替换元素（Replaced Element），浏览器会依据 src 或 <source>属性来加载外部视频资源并进行渲染，同时支持用户对播放操作进行控制，例如暂停、音量调节等。其核心价值主要体现在以下几个方面：
    - 多媒体体验：可直接在网页中播放诸如产品演示、教程、宣传片等视频内容，有效提升网页内容的丰富度以及用户的参与度。
    - 跨平台兼容：支持 MP4、WebM、Ogg 等主流视频格式，并能通过 <source>元素适配不同的浏览器。
    - 用户控制：默认情况下，可通过 controls 属性启用播放/暂停、进度条、音量调节等控件，方便用户操作。
    - 无障碍支持：借助 <track>元素添加字幕，能为听障用户提供文本描述，符合 WCAG 标准。

3. 元素

- 视频与音频元素：<video>
  - 【语法结构】
    - <video>是块级元素（Block-Level Element），默认占满父容器宽度，内部可包含 <source>（指定视频源）、<track>（字幕/元数据）等子元素，支持通过属性控制播放行为。
      - 必须通过 src（单个源）或 <source>（多个源）指定视频文件，否则无法播放。
      - 具有块级特性：默认占满父容器宽度，可通过width/height调整显示尺寸，建议同时设置以保持宽高比。
      - 可嵌套 <source>（多格式适配）、<track>（字幕）、文本（备用提示）。

```
<!-- 基础视频播放（必选 src 或 source） -->
<video
  src="demo.mp4"
  controls
  width="800"
  height="450"
  poster="preview.jpg"
  preload="metadata"
>
  <!-- 备用视频格式（兼容不支持 MP4 的浏览器） -->
  <source src="demo.webm" type="video/webm">
  <source src="demo.ogg" type="video/ogg">

  <!-- 字幕轨道（可选） -->
  <track
    src="subtitles.vtt"
    kind="subtitles"
    label="中文字幕"
    srclang="zh-CN"
    default
  >

  <!-- 浏览器不支持 video 时的提示 -->
  您的浏览器不支持 HTML5 视频，请升级浏览器或 <a href="demo.mp4">下载视频</a>。
</video>
```


3. 元素

- 视频与音频元素：<video>
  - 【核心属性】
    - <video> 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。同时，<video>支持多种属性，用于控制视频的加载、播放行为和显示效果。
    - 基础属性

属性	说明	示例	注意事项
src	可选：直接指定视频文件的 URL（若使用 <source>，此属性可省略）。	src="demo.mp4"	仅支持单个视频源，推荐配合 <source>提供多格式。
controls	可选：显示浏览器默认的视频控制条（播放/暂停、进度条、音量等）。	controls	移动端浏览器可能默认隐藏控制条（需用户交互触发显示）。

    - 播放控制属性

属性	说明	示例	注意事项
autoplay	可选：视频加载完成后自动播放（部分浏览器需用户交互后生效）。	autoplay	浏览器策略限制：多数浏览器禁止自动播放有声视频（需 muted配合）。
loop	可选：视频播放结束后循环重播。	loop	适用于背景视频或循环广告。
muted	可选：视频初始静音（配合 autoplay使用，避免被浏览器拦截）。	muted	移动端可能强制静音，需用户手动取消。
preload	可选：定义视频预加载策略（减少首屏加载时间）。	preload="metadata"（加载元数据） preload="auto"（自动预加载）	取值：none（不预加载）、metadata（加载时长/尺寸）、auto（自动）。

3. 元素

3.2 文本与列表元素：填充内容基础

- 视频与音频元素：<video>
  - 【核心属性】
    - 显示与交互属性

属性	说明	示例	注意事项
width	定义视频的显示宽度（单位：像素 px或百分比 %）。	width="800"（800px） width="50%"（父容器宽度的50%）	仅改变显示尺寸，不影响原始文件大小；建议与 height同时设置以保持宽高比。
height	定义视频的显示高度（单位：像素 px或百分比 %）。	height="450"（450px）	同上。
poster	可选：视频加载前或暂停时显示的预览图（URL）。	poster="preview.jpg"	提升用户体验，避免空白屏幕。
controlslist	可选：自定义控制条显示的按钮（如隐藏“全屏”按钮）。	controlslist="nodownload nofullscreen"	仅部分浏览器支持（如 Chrome），需配合 controls使用。

    - 高级属性

属性	说明	示例	注意事项
crossorigin	控制跨域视频的 CORS（跨域资源共享）设置（用于画布操作等场景）。	crossorigin="anonymous"（匿名请求）	若需通过画布（<canvas>）处理视频（如截图），需设置此属性。
disablePictureInPicture	可选：禁用画中画模式（部分浏览器支持）。	disablePictureInPicture	适用于不允许用户小窗播放的场景。
playsinline	可选：强制视频在页面内播放（而非全屏，iOS 专用）。	playsinline	解决 iOS 自动全屏播放的问题。

3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 视频与音频元素：<video>
  - 【使用建议】
    - 提供多格式视频源（<source>元素），不同浏览器支持的视频格式不同（如 Chrome 支持 WebM，Safari 支持 MP4），通过<source>元素指定多个格式可确保兼容性。

```
<video controls>
  <source src="demo.mp4" type="video/mp4">
  <source src="demo.webm" type="video/webm">
  <source src="demo.ogg" type="video/ogg">
</video>
```

    - 优化视频加载性能
      - 根据视频重要性设置 preload（如首屏视频用 preload="auto"，非首屏用 preload="metadata"）。
      - 对非首屏视频使用 loading="lazy"（HTML5 新增），延迟加载直到用户滚动到附近。

```
<video src="below-fold.mp4" controls loading="lazy">
```

      - 使用工具（如 FFmpeg、HandBrake）压缩视频，优先选择现代格式（WebM/VP9 或 MP4/H.265），相同质量下体积更小。
    - 提升用户体验
      - 设置 poster属性，避免视频加载前显示空白

```
<video src="demo.mp4" controls poster="preview.jpg">
```

      - 通过 controlslist隐藏不必要的按钮（如“下载”），保持界面简洁

```
<video controls controlslist="nodownload">
```

3. 元素

- 视频与音频元素：<video>
  - 【使用建议】
    - 处理自动播放限制，浏览器通常禁止自动播放有声视频（需用户交互触发），若需自动播放，需满足以下条件之一：
      - 视频静音（muted属性）
      - 用户已与页面交互（如点击按钮）
    - 无障碍支持（字幕与元数据），通过 <track>元素添加字幕（.vtt格式），提升听障用户的观看体验。

```
<track
  src="subtitles-zh.vtt"
  kind="subtitles"
  label="中文字幕"
  srclang="zh-CN"
  default
>
<track
  src="subtitles-en.vtt"
  kind="subtitles"
  label="英文字幕"
  srclang="en"
>
```


3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 视频与音频元素：<video>
  - 【使用建议】
    - 通过 width和 height属性设置视频的显示尺寸，或使用 CSS 百分比适配不同屏幕。

```
video {
  width: 100%; /* 宽度占满父容器 */
  height: auto; /* 高度自动适配，保持宽高比 */
}
```

    - 当浏览器不支持 <video>时，显示备用提示（如下载链接）

```
<video>...</video>
<p>您的浏览器不支持 HTML5 视频，请 <a href="demo.mp4">下载视频</a>。</p>
```

3. 元素

- 视频与音频元素：<audio>
  - 【元素简介】
    - <audio>作为 HTML 中的音频嵌入元素（Audio Element），主要用于在网页中实现音频文件的播放功能。它属于替换元素（Replaced Element），浏览器会依据 src 或 <source> 属性加载外部音频资源并进行渲染，同时支持用户对播放操作进行控制，如暂停、调节音量以及跳转进度等。其核心价值体现在以下几个方面：
    - 多媒体体验：借助该元素可直接在网页中播放各类音频，如背景音乐、语音提示、有声书等，有效提升网页内容的丰富度与用户的参与度。
    - 跨平台兼容：支持 MP3、WAV、OGG 等主流音频格式，并可通过 <source> 元素实现对不同浏览器的适配。
    - 用户控制：默认情况下，通过 controls 属性启用播放/暂停、音量调节、进度条等控件，方便用户操作。
    - 无障碍支持：与字幕（<track> 元素）相结合，可为听障用户提供文本描述，符合 WCAG 标准。

3. 元素

- 视频与音频元素：<audio>
  - 【语法结构】
    - <audio>是块级元素（Block-Level Element），默认占满父容器宽度，内部可包含 <source>（指定音频源）、<track>（字幕/元数据）等子元素，支持通过属性控制播放行为。
      - 必须通过 src（单个源）或 <source>（多个源）指定音频文件，否则无法播放。
      - 具备块级特性，默认占满父容器宽度，可通过 CSS 调整控件样式，如宽度、位置）。
      - 可嵌套 <source>（多格式适配）、<track>（字幕）、文本（备用提示）。

```
<!-- 基础音频播放（必选 src 或 source） -->
<audio
  src="music.mp3"
  controls
  preload="metadata"
  autoplay
  muted
>
  <!-- 备用音频格式（兼容不支持 MP3 的浏览器） -->
  <source src="music.ogg" type="audio/ogg">
  <source src="music.wav" type="audio/wav">

  <!-- 字幕轨道（可选，如音频描述） -->
  <track
    src="description.vtt"
    kind="descriptions"
    label="音频描述"
    srclang="zh-CN"
    default
  >

  <!-- 浏览器不支持 audio 时的提示 -->
  您的浏览器不支持 HTML5 音频，请升级浏览器或 <a href="music.mp3">下载音频</a>。
</audio>
```

3. 元素

- <audio>
  - 【核心属性】
    - <audio> 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。同时，<audio>支持多种属性，用于控制音频的加载、播放行为和显示效果。
    - 基础属性

属性	说明	示例	注意事项
src	可选：直接指定视频文件的 URL（若使用 <source>，此属性可省略）。	src="music.mp3"	仅支持单个音频源，推荐配合 <source>提供多格式。
controls	可选：显示浏览器默认的音频控制条（播放/暂停、进度条、音量等）。	controls	移动端浏览器可能默认隐藏控制条（需用户交互触发显示）。

    - 播放控制属性

属性	说明	示例	注意事项
autoplay	可选：音频加载完成后自动播放（部分浏览器需用户交互后生效）。	autoplay	浏览器策略限制：多数浏览器禁止自动播放有声视频（需 muted配合）。
loop	可选：音频播放结束后循环重播。	loop	适用于背景视频或循环广告。
muted	可选：音频初始静音（配合 autoplay使用，避免被浏览器拦截）。	muted	移动端可能强制静音，需用户手动取消。
preload	可选：定义音频预加载策略（减少首屏加载时间）。	preload="metadata"（加载元数据） preload="auto"（自动预加载）	取值：none（不预加载）、metadata（加载时长/尺寸）、auto（自动）。

3. 元素

- 视频与音频元素：<audio>
  - 【核心属性】
    - 显示与交互属性

属性	说明	示例	注意事项
width	定义音频的显示宽度（单位：像素 px或百分比 %）。	width="800"（800px） width="50%"（父容器宽度的50%）	仅影响控件样式，不影响音频文件本身。
style	通过 CSS 自定义音频控件样式（如颜色、布局）。	style="margin: 1rem 0; background: #f0f0f0;"	推荐用 CSS 类统一管理样式，避免内联样式冗余。

    - 高级属性

属性	说明	示例	注意事项
crossorigin	控制跨域音频的 CORS（跨域资源共享）设置（用于画布操作等场景）。	crossorigin="anonymous"（匿名请求）	若需通过画布（<canvas>）处理音频（如分析波形），需设置此属性。
disablePictureInPicture	可选：禁用画中画模式（部分浏览器支持）。	disablePictureInPicture	适用于不允许用户小窗播放的场景（音频通常无画中画，此属性多用于视频）。

3. 元素

- 视频与音频元素：<audio>
  - 【使用建议】
    - 提供多格式音频源（<source>元素），不同浏览器支持的音频格式不同（如 Chrome 支持 MP3，Safari 支持 WAV），通过<source>元素指定多个格式可确保兼容性。

```
<audio controls>
  <source src="music.mp3" type="audio/mpeg">
  <source src="music.ogg" type="audio/ogg">
  <source src="music.wav" type="audio/wav">
</audio>
```

    - 优化音频加载性能
      - 配置预加载策略，根据音频重要性设置 preload（如首屏背景音乐用 preload="auto"，非首屏用 preload="metadata"）。
      - 实现懒加载，对非首屏音频使用 loading="lazy"（HTML5 新增），延迟加载直到用户滚动到附近。

```
<audio src="background-music.mp3" controls loading="lazy">
```

      - 使用工具（如 FFmpeg、Audacity）压缩音频，优先选择现代格式（MP3/AAC 或 Ogg/Vorbis），相同质量下体积更小。
    - 提升用户体验
      - 浏览器通常禁止自动播放有声音频，若需自动播放，需设置 muted属性（用户可手动取消静音）

```
<audio autoplay muted loop>
  <source src="bg-music.mp3" type="audio/mpeg">
</audio>
```

      - 通过 CSS 调整控件样式（如隐藏默认控件，自定义播放按钮）

```
audio::-webkit-media-controls-panel {
  background: #f8f9fa; /* 自定义控件背景 */
}
```


3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 视频与音频元素：<audio>
  - 【使用建议】
    - 无障碍支持（字幕与元数据）。
      - 通过 <track>元素添加音频描述（如盲人用户需要的场景说明），提升听障用户的体验

```
<track
  src="audio-description.vtt"
  kind="descriptions"
  label="音频场景描述"
  srclang="zh-CN"
  default
>
```

    - 响应式设计适配，通过 width属性或 CSS 百分比调整控件宽度，适配不同屏幕尺寸

```
audio {
  width: 100%; /* 宽度占满父容器 */
  max-width: 400px; /* 最大宽度限制 */
}
```

    - 当浏览器不支持 <audio>时，显示备用提示（如下载链接）

```
<audio>...</audio>
<p>您的浏览器不支持 HTML5 音频，请 <a href="music.mp3">下载音频</a>。</p>
```

3. 元素

3.3 图像与多媒体元素：丰富页面内容

- 媒体与说明：<figure>与<figcaption>
  - 【元素简介】
    - <figure> 与 <figcaption> 是 HTML5 引入的语义化元素组合，用于标记文档中独立且可分离的内容块，如图像、图表、代码片段、引用等，并借助 <figcaption> 为其添加标题或说明。
    - <figure>：用于定义一个独立的内容块（块级元素），该内容与主内容在逻辑上相关，但可独立存在，即移除后不影响主内容的核心信息。
    - <figcaption>：用于定义 <figure> 的标题或描述（块级元素），通常嵌套于 <figure> 内部，旨在明确内容块的主题或提供补充说明。
    - 其核心价值主要体现在以下方面：
    - 语义明确：通过 <figure> 标记独立内容块，<figcaption> 提供标题，有助于浏览器、搜索引擎（SEO）以及辅助技术（如屏幕阅读器）理解内容的“独立性”与“上下文”。
    - 结构清晰：取代传统使用无意义的 模拟内容块的方式，使代码更符合语义化规范，增强可维护性。
    - 无障碍友好：<figcaption> 为视障用户（屏幕阅读器）提供内容说明，符合 WCAG（网页内容可访问性指南）标准。
    - SEO 优化：搜索引擎可通过 <figure> 和 <figcaption> 识别内容的独立性和相关性，如示例代码、图表说明等，从而提升页面内容的可信度和排名。

3. 元素

3.2 文本与列表元素：填充内容基础

- 媒体与说明：<figure>与<figcaption>
  - 【语法结构】
    - <figure> 为块级元素，默认情况下会占满父容器的宽度。该元素内部可容纳任意内容，例如文本、图像、代码、引用等，并且必须与 <figcaption> 搭配使用（ <figcaption> 为可选元素，但强烈建议使用）。
      - <figcaption>必须作为 <figure>的直接子元素，通常置于内容的上方或下方。
      - <figure>的内容应与主内容存在逻辑关联，且具备独立性，即便移除该内容，主内容依然完整。
      - <figcaption>虽为可选元素，但建议始终添加。空的<figcaption>并无实际意义，可能会对无障碍体验造成干扰。
  - 【核心属性】
    - <figure>和 <figcaption>元素均继承了 HTML 的全局属性（即适用于所有元素的通用属性）。

```
<!-- 基础结构：figure + 内容 + figcaption -->
<figure class="chart">
  <img src="sales-chart.png" alt="2023年销售额增长趋势图">
  <figcaption>图1：2023年各季度销售额对比（单位：万元）</figcaption>
</figure>

<!-- 复杂内容块（代码示例） -->
<figure class="code-example">
  <pre><code>
    // JavaScript 示例：数组去重
    const uniqueArray = arr => [...new Set(arr)];
  </code></pre>
  <figcaption>示例1：使用 Set 实现数组去重</figcaption>
</figure>

<!-- 包含引用的内容块 -->
<figure class="quote">
  <blockquote>
    “语义化元素是 HTML 的未来，它让网页内容更易被机器和人类理解。”
  </blockquote>
  <figcaption>—— Web 标准委员会，《HTML5 语义化指南》</figcaption>
</figure>
```

3. 元素

- 媒体与说明：<figure>与<figcaption>
  - 【使用建议】
    - <figcaption>是 <figure>的“官方搭档”，用于明确内容块的主题或说明，如图像标题、代码示例描述。

```
<figure>
  <img src="logo.png" alt="公司LOGO">
  <figcaption>XX科技公司官方LOGO</figcaption> <!-- 明确说明内容 -->
</figure>
```

    - <figure>内容需与主内容逻辑相关但可独立，如移除后主内容仍完整。

```
<article>
  <h1>HTML5 语义化指南</h1>
  <p>正文内容...</p>
  <figure>
    <img src="semantic-tags.png" alt="语义化元素示意图">
    <figcaption>图2：HTML5 常用语义化元素结构</figcaption>
  </figure>
</article>
```

    - <figure>适用于以下典型场景，<figcaption>需进行针对性描述：
      - <figcaption>阐明图像的主题或数据含义，例如“图1：2023年销售额对比”。
      - <figcaption>详述代码的功能或用途，如“示例1：使用 Set 实现数组去重”。
      - <figcaption>标注引用的来源或作者，如“—— Web 标准委员会，《HTML5 语义化指南》”。
    - 优化可访问性
      - 确保 <figcaption>内容简洁且表意明确，例如“图1：2023年销售额对比”比“图片”更具实际意义。
      - 若 <figure>内容本身清晰明确（如代码块），可省略 <figcaption>，但可通过 aria - label 进行补充说明（可选）。

案例演示

- 案例：前端学习入门指南
  - 构建一个“智能手表产品展示页”，实践HTML图像与多媒体元素的核心用法。该案例覆盖静态图片、响应式图片、视频嵌入、音频说明、结构化媒体组织等场景，帮助学习者掌握“如何用多媒体元素提升页面表现力”的核心技能。。
  - 【案例目标】
    - 熟练使用<img>标签展示静态图片，实现响应式加载与懒加载；
    - 掌握<video>标签嵌入视频，支持多源兼容与控制条；
    - 理解<audio>标签嵌入音频说明；
    - 学会用<figure>+<figcaption>结构化组织媒体内容；
    - 优化多媒体元素的无障碍性与性能。

3. 元素

3.4 表格元素：结构化数据展示

- 表格（Table）作为 HTML 中结构化数据展示的关键工具，以行列对齐的方式清晰呈现数据关系，可用于展示统计数据、产品规格、财务报表等内容。相较于列表、段落等元素，表格凭借其严格的行列结构，能够直观地反映数据之间的逻辑关联，例如“行代表记录，列代表属性”。
- 表格的基础结构：<table>
  - 【元素简介】
    - <table>是 HTML 中的表格标签（Table Element），用于在网页中结构化展示数据（如统计表格、日程表、数据对比表等）。它是语义化标签的核心成员之一，通过行（<tr>）、列（<th>/<td>）的二维结构清晰呈现数据关系，其核心价值：
    - 语义明确：通过 <table>标记结构化数据，帮助浏览器、搜索引擎（SEO）和辅助技术（如屏幕阅读器）理解数据的“行列关系”和“逻辑分组”。
    - 数据展示：以表格形式直观呈现多维度数据（如时间-事件、产品-参数），提升信息可读性。
    - 无障碍友好：配合 <caption>（标题）和 ARIA 属性，为视障用户（屏幕阅读器）提供表格上下文，符合 WCAG 标准。
    - 样式可控：通过 CSS 灵活控制表格边框、间距、对齐方式等，适应不同设计需求。

3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<table>
  - 【语法结构】
    - <table>是块级元素（默认占满父容器宽度），内部通过子元素定义表格的结构和内容。
      - <table>直接子元素通常为 <caption>、<thead>、<tbody>、<tfoot>，顺序不限，但 <thead>通常在最前。
      - <tr>（行）是表格的水平单元，内部可包含 <th>（表头单元格）或 <td>（数据单元格）。
      - <thead>（表头）、<tbody>（表体）、<tfoot>（表尾）用于逻辑分组，如表头定义列名，表体为核心数据，表尾为汇总，提升可访问性。

```
<table class="sales-table">
  <caption>2023年各季度销售额统计（单位：万元）</caption>
  <thead>
    <tr>
      <th scope="col">产品</th>
      <th scope="col">Q1</th>
      <th scope="col">Q2</th>
      <th scope="col">Q3</th>
      <th scope="col">Q4</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>手机</td>
      <td>120</td>
      <td>150</td>
      <td>180</td>
      <td>200</td>
    </tr>
    <tr>
      <td>平板</td>
      <td>80</td>
      <td>100</td>
      <td>120</td>
      <td>150</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>总计</td>
      <td>200</td>
      <td>250</td>
      <td>300</td>
      <td>350</td>
    </tr>
  </tfoot>
</table>
```

3. 元素

- 表格的基础结构：<table>
  - 【核心属性】
    - <table>支持多种属性，用于控制表格的显示、结构和语义。

属性	说明	示例	注意事项
border	定义表格边框宽度（单位：像素 px）。	border="1"（1px 边框）	HTML5 不推荐使用（建议通过 CSS border属性控制）。
cellpadding	定义单元格内容与边框的间距（单位：像素 px）。	cellpadding="8"（内边距 8px）	HTML5 不推荐使用（建议通过 CSS padding控制）。
cellspacing	定义单元格之间的间距（单位：像素 px）。	cellspacing="2"（间距 2px）	HTML5 不推荐使用（建议通过 CSS border-spacing控制）。
width	定义表格的显示宽度（单位：像素 px或百分比 %）。	width="800"（800px） width="90%"（父容器宽度的90%）	仅改变显示尺寸，不影响原始数据结构；建议通过 CSS 控制。
summary	HTML5 新增：为表格提供简要描述（用于无障碍访问，屏幕阅读器朗读）。	summary="2023年各季度销售额统计表"	替代 <caption>的补充说明（<caption>更适合可见标题）。

3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<table>
  - 【使用建议】
    - 语义化使用表格
      - 避免用 <table>做页面布局（如导航栏、卡片排列），布局应使用 或 CSS Grid/Flexbox。
      - 确保表格的行（<tr>）和列（<th>/<td>）与数据的逻辑维度一致，如“时间-指标”“产品-销量”。
    - 优化可访问性
      - 为表格添加可见标题（如“2023年销售额统计”），帮助所有用户（包括视障用户）快速理解表格用途。

```
<table>
  <caption>2023年各季度销售额（单位：万元）</caption>
  ...
</table>
```

      - 为 <th>标记作用域（col或 row），帮助屏幕阅读器识别表头与数据的关联。

```
<tr>
  <th scope="col">季度</th> <!-- 列表头 -->
  <th scope="col">Q1</th>
</tr>
```

    - 性能优化
      - 减少嵌套表格（如 <table>内部再嵌套 <table>）或过多行/列（如超过 100 行），以免影响加载速度。
      - 对大数据量的表格（如数千行），使用分页或懒加载。

3. 元素

- 表格的基础结构：<caption>
  - 【元素简介】
    - <caption>是 HTML 中用于为表格添加标题的标签（Block-Level Element），核心作用是为 <table>定义语义化的标题文本，描述表格的内容或用途。它是表格结构的重要组成部分，不仅能提升页面的可读性，还能增强可访问性，如屏幕阅读器可朗读标题，帮助视障用户理解表格内容，其核心价值在于：
    - 语义化标识：明确表格的用途（如“用户订单统计表”“月度销售数据”），帮助浏览器、搜索引擎（SEO）和辅助技术理解表格上下文。
    - 可访问性支持：屏幕阅读器（如 NVDA、VoiceOver）会优先朗读 <caption>的内容，为用户提供表格的上下文信息。
    - 视觉引导：通过样式设计（如字体加粗、居中显示），引导用户快速定位表格的核心内容。

3. 元素

- 表格的基础结构：<caption>
  - 【语法结构】
    - <caption>必须嵌套在 <table>标签内部，通常位于 <table>的开头（紧接在 <table>之后），用于定义表格的全局标题。
      - 一个 <table>最多包含一个 <caption>（HTML 规范允许，但通常仅需一个）。
      - <caption>可位于 <table>内的任意位置（如开头或结尾），但推荐放在开头（符合阅读习惯，屏幕阅读器优先读取）。
      - <caption>可包含文本、内联元素（如 、、图标等），但需保持简洁。
  - 【语法结构】
    - <caption>继承 HTML 全局属性（如 id、class、style），无专属属性。

```
<table>
  <caption>2023年10月用户注册统计</caption> <!-- 表格标题 -->
  <thead>
    <tr>
      <th>日期</th>
      <th>注册人数</th>
      <th>活跃用户</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>2023-10-01</td>
      <td>120</td>
      <td>85</td>
    </tr>
    <tr>
      <td>2023-10-02</td>
      <td>150</td>
      <td>110</td>
    </tr>
  </tbody>
</table>
```

3. 元素

- 表格的基础结构：<caption>
  - 【使用建议】
    - 为每个表格添加 <caption>，内容应简洁明了，例如“2023年销售额统计”，避免使用如“表格1”这类模糊表述。
    - 默认情况下，<caption>位于表格顶部，可通过 CSS 的 caption - side: bottom 属性将其调整至底部，需注意，屏幕阅读器仍会按照文档顺序进行朗读。
    - 若表格内容复杂，例如包含多维度数据，可在 <caption>中补充关键信息，如“数据来源：财务部”。

3. 元素

- 表格的基础结构：<thead>与<th>
  - 【元素简介】
    - <thead>和 <th>是 HTML 表格（<table>）中用于定义表头内容的核心标签，共同构成表格的头部区域，用于描述表格列的含义或行的分组信息。
    - <thead>：定义表格的表头容器（Block-Level Element），通常包裹表格的表头行（<tr>），位于 <table>内部、<tbody>（表体）之前。其作用是明确表格的列标题或行分组标题，帮助用户快速理解表格结构。
    - <th>：定义表头容器中的表头单元格（Inline-Level Element），是 <thead>或 <tr>的子元素，用于放置每列（或每行）的标题文本（如“姓名”“年龄”“销售额”）。
    - 其核心价值在于：
    - 结构清晰：通过 <thead>和 <th>明确区分表头与表体，使表格结构更易读（尤其对于复杂表格）。
    - 可访问性提升：屏幕阅读器（如 NVDA）会优先朗读 <th>的内容，帮助视障用户理解列/行的含义。
    - 样式可控：通过 CSS 自定义表头样式（如背景色、字体加粗），与表体内容形成视觉区分。

3. 元素

- 表格的基础结构：<thead>与<th>
  - 【语法结构】
    - <thead>是表格的表头容器，内部可包含一个或多个 <tr>（表头行），每个 <tr>内包含多个 <th>（表头单元格）。
      - <thead>必须是 <table>的直接子元素，且位于 <tbody>或 <tfoot>（表尾）之前。
      - <th>通常位于 <thead>内的 <tr>中（定义列标题），也可位于 <tbody>内的 <tr>中（定义行标题，需配合 scope="row"）。
      - 虽然 HTML 规范允许省略 <thead>（直接在 <table>内放置 <tr>作为表头），但推荐显式使用 <thead> 以提升语义化和可访问性。

```
<table>
  <!-- 表头容器 -->
  <thead>
    <!-- 表头行 -->
    <tr>
      <th scope="col">姓名</th> <!-- 列标题（作用域为列） -->
      <th scope="col">年龄</th>
      <th scope="col">注册时间</th>
    </tr>
  </thead>

  <!-- 表体（数据内容） -->
  <tbody>
    <tr>
      <td>张三</td>
      <td>28</td>
      <td>2023-01-15</td>
    </tr>
    <tr>
      <td>李四</td>
      <td>32</td>
      <td>2023-03-20</td>
    </tr>
  </tbody>
</table>
```


3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<thead>与<th>
  - 【核心属性】
    - <thead>和 <th>均继承 HTML 全局属性（如 id、class、style）。
    - <thead>核心属性

属性	说明	示例	注意事项
id	为 <thead>指定唯一标识（用于 CSS 样式或 JavaScript 操作）。	id="userTableHeader"	推荐使用有意义的名称（如 salesHeader）。
class	为 <thead>添加 CSS 类（用于样式控制）。	class="table-header"	配合 CSS 定义背景色、字体等（如 background: #f5f5f5）。
aria-label	无障碍属性：为屏幕阅读器提供表头的描述（当表头内容不明确时）。	aria-label="用户信息表表头"	增强视障用户的理解（如表头行包含多个 <th>时）。

- <th>核心属性

属性	说明	示例	注意事项
scope	定义表头单元格的作用域（col列 / row行 / colgroup列组 / rowgroup行组）。	<th scope="col">姓名</th>（列标题）<th scope="row">2023年</th>（行标题）	默认值 scope="col"（列标题），需根据实际场景设置（如行标题用 row）。
id	为 <th>指定唯一标识（用于 CSS 样式或 JavaScript 操作）。	id="nameHeader"	推荐使用有意义的名称（如 ageHeader）。
class	为 <th>添加 CSS 类（用于样式控制）。	class="header-cell"	配合 CSS 定义对齐方式（如 text-align: center）或字体大小。
aria-label	无障碍属性：为屏幕阅读器提供表头单元格的补充描述（当文本不够明确时）。	<th scope="col" aria-label="用户姓名">姓名</th>	增强视障用户的理解（如标题文本简短时）。
colspan	定义表头单元格跨列的数量（合并多列标题）。	<th colspan="2">基本信息</th>（合并两列）	需确保跨列后的表格结构合理（避免列数不匹配）。
rowspan	定义表头单元格跨行的数量（合并多行标题）。	<th rowspan="2">季度数据</th>（合并两行）	需确保跨行后的表格结构合理（避免行数不匹配）。

3. 元素

- 表格的基础结构：<thead>与<th>
  - 【使用建议】
    - 无论表格多简单，都应通过 <thead>明确标记表头区域（即使只有一行表头）。这有助于屏幕阅读器和搜索引擎理解表格结构。
    - 合理设置 <th>的 scope属性
      - 列标题：默认 scope="col"（无需显式设置），表示该 <th>是某一列的标题（如“姓名”对应“姓名”列）。
      - 行标题：若 <th>用于行标题（如分组行的标题），需设置 scope="row"（如“2023年”对应某一行数据）。
    - 避免 <th>与 <td>混淆，<th>仅用于表头（列/行标题），<td>用于表体（数据内容）。混淆两者会导致语义错误（如屏幕阅读器无法正确识别标题）。
    - 当需要合并多列或多行标题时，通过 colspan（跨列）或 rowspan（跨行）属性实现，但需确保表格结构的一致性，如合并两列后，后续行的 <td>数量需相应减少，要合理使用跨列/跨行标题。

3. 元素

- 表格的基础结构：<tbody>与<tr>、<td>
  - 【元素简介】
    - <tbody>、<tr>和 <td>是 HTML 表格（<table>）中组织数据内容的核心标签，共同构成表格的主体结构，用于清晰展示数据的行与列。
    - <tbody>：定义表格的主体容器（Block-Level Element），用于包裹表格的所有数据行（<tr>），位于 <thead>（表头）之后、<tfoot>（表尾）之前。其作用是明确区分表格的“数据区域”与“表头/表尾区域”，提升结构语义化。
    - <tr>：定义表格的数据行（Block-Level Element），是 <tbody>或 <thead>的子元素，每个 <tr>代表表格中的一行数据，内部可包含多个 <td>（数据单元格）或 <th>（表头单元格）。
    - <td>：定义数据行中的数据单元格（Inline-Level Element），是 <tr>的子元素，用于存放具体的数据内容（如文本、数值、图片等）。

3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<tbody>与<tr>、<td>
  - 【语法结构】
    - 三者需嵌套使用，构成完整的表格数据区域。
      - <thead>必须是 <table>的直接子元素，且位于 <tbody>或 <tfoot>（表尾）之前。
      - <th>通常位于 <thead>内的 <tr>中（定义列标题），也可位于 <tbody>内的 <tr>中（定义行标题，需配合 scope="row"）。
      - 虽然 HTML 规范允许省略 <thead>（直接在 <table>内放置 <tr>作为表头），但推荐显式使用 <thead> 以提升语义化和可访问性。

```
<table>
  <!-- 表头容器 -->
  <thead>
    <tr>
      <th scope="col">姓名</th>
      <th scope="col">年龄</th>
      <th scope="col">注册时间</th>
    </tr>
  </thead>

  <!-- 表格主体（数据区域） -->
  <tbody>
    <!-- 数据行 1 -->
    <tr>
      <td>张三</td>
      <td>28</td>
      <td>2023-01-15</td>
    </tr>

    <!-- 数据行 2 -->
    <tr>
      <td>李四</td>
      <td>32</td>
      <td>2023-03-20</td>
    </tr>
  </tbody>

  <!-- 表尾容器（可选） -->
  <tfoot>
    <tr>
      <td colspan="2">总计</td>
      <td>2人</td>
    </tr>
  </tfoot>
</table>
```


3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<tbody>与<tr>、<td>
  - 【核心属性】
    - <thead>和 <th>均继承 HTML 全局属性（如 id、class、style）。
    - <tbody>核心属性

属性	说明	示例	注意事项
id	为 <thead>指定唯一标识（用于 CSS 样式或 JavaScript 操作）。	id="userTableHeader"	推荐使用有意义的名称（如 salesHeader）。
class	为 <thead>添加 CSS 类（用于样式控制）。	class="table-header"	配合 CSS 定义背景色、字体等（如 background: #f5f5f5）。
aria-label	无障碍属性：为屏幕阅读器提供表头的描述（当表头内容不明确时）。	aria-label="用户信息表表头"	增强视障用户的理解（如表头行包含多个 <th>时）。

    - <tr>核心属性

属性	说明	示例	注意事项
id	为 <tr>指定唯一标识（用于 CSS 样式或 JavaScript 操作）。	id="row-1"	推荐使用有意义的名称（如 row-2023）。
class	为 <tr>添加 CSS 类（用于样式控制）。	class="highlight-row"	配合 CSS 定义行高亮（如 background: #f0f8ff）。
align	已废弃：定义行内内容的水平对齐方式（left/center/right）。	align="center"	现代用法推荐通过 CSS text-align替代。
valign	已废弃：定义行内内容的垂直对齐方式（top/middle/bottom）。	valign="middle"	现代用法推荐通过 CSS vertical-align替代。
aria-rowindex	无障碍属性：定义行的序号（从 1 开始，供屏幕阅读器读取）。	aria-rowindex="2"	需与实际行号一致（通常自动生成）。

3. 元素

- 表格的基础结构：<tbody>与<tr>、<td>
  - 【核心属性】
    - <td>核心属性

属性	说明	示例	注意事项
id	为 <td>指定唯一标识（用于 CSS 样式或 JavaScript 操作）。	id="name-1"	推荐使用有意义的名称（如 age-2023）。
class	为 <td>添加 CSS 类（用于样式控制）。	class="text-center"	配合 CSS 定义对齐方式（如 text-align: right）。
colspan	定义单元格跨列的数量（合并多列）。	colspan="2"（合并两列）	需确保跨列后的表格列数匹配（如合并两列后，后续行的 <td>数量需调整）。
rowspan	定义单元格跨行的数量（合并多行）。	rowspan="2"（合并两行）	需确保跨行后的表格行数匹配（如合并两行后，后续行的 <td>需跳过对应位置）。
headers	无障碍属性：关联表头单元格的 id（逗号分隔），帮助屏幕阅读器理解数据与表头的对应关系。	headers="name age"（关联 id="name"和 id="age"的 <th>）	提升视障用户对数据列的理解（需配合 <th>的 id使用）。
scope	仅用于 <th>，但 <td>无此属性（需注意区分）。	—	—

3. 元素

- 表格的基础结构：<tbody>与<tr>、<td>
  - 【使用建议】
    - 所有数据行（非表头/表尾）必须包裹在 <tbody>中（即使只有一行），这是 HTML 语义化的基本要求。
    - 每个 <tr>代表一行数据，内部 <td>的数量应与表头 <th>的数量一致（除非使用 colspan/rowspan合并单元格）。
    - 通过 <td headers="header-id">关联表头 <th id="header-id">，帮助屏幕阅读器理解数据对应的列标题。

3. 元素

- 表格的基础结构：<tfoot>（可选）
  - 【元素简介】
    - <tfoot>是 HTML 表格（<table>）中用于定义表尾内容的容器标签（Block-Level Element），核心作用是包裹表格的汇总行或统计信息（如总计、平均值、合计值等）。它是表格结构的重要组成部分，通过明确区分“数据主体”与“汇总信息”，帮助用户快速理解数据的整体情况，同时提升表格的语义化和可访问性。

3. 元素

3.4 表格元素：结构化数据展示

- 表格的基础结构：<tfoot>（可选）
  - 【语法结构】
    - <tfoot>必须嵌套在 <table>标签内部，且位于 <tbody>（表体）之后（若存在 <tfoot>，需在 <tbody>之后、</table>之前）。
      - <tfoot>必须位于 <tbody>之后（若表格同时包含 <thead>、<tbody>、<tfoot>，顺序应为 <thead>→ <tbody>→ <tfoot>）
      - <tfoot>内部通常包含汇总行（如总计、平均值），通过 <tr>（行）和 <td>（单元格）组织内容，支持 colspan（跨列）或 rowspan（跨行）合并单元格。
      - <tfoot>仅用于存放汇总信息，不可放置普通数据行（普通数据行应放在 <tbody>中）。
  - 【核心属性】
    - <tfoot> 元素继承了 HTML 的全局属性（即适用于所有元素的通用属性），并且能够借助 ARIA（无障碍富互联网应用）属性来提升可访问性。

```
<table>
  <!-- 表头容器 -->
  <thead>
    <tr>
      <th scope="col">姓名</th>
      <th scope="col">年龄</th>
      <th scope="col">注册时间</th>
    </tr>
  </thead>

  <!-- 表格主体（数据区域） -->
  <tbody>
    <tr>
      <td>张三</td>
      <td>28</td>
      <td>2023-01-15</td>
    </tr>
    <tr>
      <td>李四</td>
      <td>32</td>
      <td>2023-03-20</td>
    </tr>
  </tbody>

  <!-- 表尾容器（汇总行） -->
  <tfoot>
    <tr>
      <td colspan="2">总计</td> <!-- 合并两列 -->
      <td>2人</td> <!-- 汇总数据 -->
    </tr>
  </tfoot>
</table>
```

3. 元素

- 表格的基础结构：<tfoot>（可选）
  - 【使用建议】
    - 汇总行通常需要跨列显示（如“总计”覆盖多列数据），通过 colspan属性合并单元格，使表格结构更紧凑。
    - 提升可访问性
      - 屏幕阅读器友好：通过 aria-label提供表尾的补充描述（如“用户信息统计汇总”），帮助视障用户理解表尾内容的用途。
      - 关联表头与汇总数据：若汇总数据对应特定列，可通过 <td headers="header-id">关联表头 <th id="header-id">，增强语义关联。

案例演示

- 案例：图书库存与借阅管理表
  - 构建企业级「图书库存与借阅管理表」，展示图书的多维度核心数据，包括分类、ISBN、书名、作者、库存状态、借阅记录、出版信息。。
  - 【案例目标】
    - 掌握HTML表格的语义化结构（<table>及子标签的正确使用）
    - 实现多级表头与跨行跨列合并（colspan/rowspan）
    - 用scope、aria-*属性强化无障碍访问
    - 展示结构化数据的清晰组织方式

3. 元素

- 表单（Form）作为 HTML 中实现用户交互的核心工具，能够收集用户输入的各类信息，包括文本、密码、选择项等，并与后端服务器进行交互，从而实现数据提交、用户认证、问卷调查等功能。在注册登录、商品下单以及信息反馈等场景中，表单均是连接用户与系统的关键桥梁。
- 表单容器：<form>
  - 【元素简介】
    - <form>是 HTML 中用于包裹用户输入控件的容器元素（Block-Level Element），是表单交互的核心结构。其核心作用是收集用户输入的数据（如文本、选择、文件等），并通过 action和 method属性将数据提交到服务器端处理脚本（如 PHP、Node.js），实现数据的传递与存储，其核心价值在于：
    - 数据收集与提交：作为表单的容器，<form>是用户输入数据与服务器端处理的桥梁，所有输入元素（如 <input>、<select>）必须包裹在 <form>内才能被提交。
    - 语义化标识：通过 <form>标记表单区域，帮助浏览器、搜索引擎（SEO）和辅助技术（如屏幕阅读器）理解页面的交互逻辑。
    - 功能扩展：支持文件上传、表单验证、自定义提交行为等高级功能（通过 enctype、novalidate等属性或 JavaScript 实现）。

3. 元素

- 表单容器：<form>
  - 【语法结构】
    - <form>内部可包含输入元素（如 <input>、<select>）、辅助元素（如 <label>、<button>）、文本内容或嵌套结构（如 <fieldset>分组）。
      - action（提交目标 URL）和 method（提交方式）是<form>的核心属性，缺一不可（若省略 method，默认值为 GET）。
      - <form>内部可包含任意表单相关元素，如输入框、选择框、文本域、按钮等，以及文本内容（如提示语）。
      - 复杂表单可通过 <fieldset>和 <legend>分组，如“个人信息”“联系方式”，提升可访问性。

```
<form action="/submit" method="POST" enctype="multipart/form-data">
  <!-- 输入元素与元素 -->
  <label for="username">用户名：</label>
  <input type="text" id="username" name="username" required>

  <label for="password">密码：</label>
  <input type="password" id="password" name="password" minlength="6">

  <!-- 下拉选择 -->
  <label for="gender">性别：</label>
  <select id="gender" name="gender">
    <option value="male">男</option>
    <option value="female">女</option>
  </select>

  <!-- 多行文本 -->
  <label for="bio">个人简介：</label>
  <textarea id="bio" name="bio" rows="4" maxlength="200"></textarea>

  <!-- 提交按钮 -->
  <button type="submit">提交</button>
</form>
```

3. 元素

- 表单容器：<form>
  - 【核心属性】
    - 必选属性

属性	说明	示例
action	必需：指定表单提交的目标 URL（服务器端处理脚本的路径）。	action="/api/submit"（提交到 /api/submit接口）
method	必需：定义数据提交方式（GET或 POST）。	method="POST"（推荐敏感数据提交）

    - 可选属性

属性	说明	示例	注意事项
enctype	定义表单数据的编码类型（仅 POST方法有效）。	enctype="multipart/form-data"（文件上传）	默认值 application/x-www-form-urlencoded（普通表单）。
name	为表单指定名称（用于 JavaScript 或服务器端标识）。	name="userForm"	避免与 id重复，推荐使用有意义的名称。
autocomplete	控制浏览器自动完成功能（on或 off）。	autocomplete="off"（禁用密码自动完成）	敏感字段（如密码）建议禁用，非敏感字段启用（提升用户体验）。
novalidate	禁用 HTML5 内置表单验证（需配合 JavaScript 自定义验证）。	novalidate	仅在需要完全自定义验证逻辑时使用。
target	定义提交后页面的打开方式（如 _blank新窗口）。	target="_blank"（提交后新窗口打开）	通常与 action配合使用。

3. 元素

- 表单容器：<form>
  - 【使用建议】
    - 复杂表单可通过 <fieldset>和 <legend>分组，如“个人信息”“联系方式”，提升可访问性。
    - 每个输入元素必须通过 <label>关联（for属性匹配 id），帮助视障用户（屏幕阅读器）识别输入用途。
    - 优先使用 HTML5 内置验证属性,如 required、minlength、pattern，减少客户端脚本依赖。
    - 若需上传文件，设置 enctype="multipart/form-data"，并添加 <input type="file">元素。
    - 通过 CSS 控制表单宽度,如 max-width: 600px，适配移动端输入，如增大字体、间距等。

3. 元素

- 表单容器：<input>
  - 【元素简介】
    - <input>是 HTML 中最灵活、最常用的输入元素（Inline-Level Element），通过 type属性定义不同的输入类型（如文本、密码、复选框、文件上传等），用于收集用户输入的各类数据（文本、数值、日期、文件等）。它是表单交互的核心组件，广泛应用于登录、注册、搜索、数据提交等场景，其核心特点在于：
    - 类型多样：通过 type属性支持 20+ 种输入类型（如 text、password、checkbox、file等），覆盖几乎所有用户输入需求。
    - 语义化友好：不同 type对应不同输入场景（如 email自动验证邮箱格式，date弹出日期选择器），提升用户体验。
    - 验证灵活：支持 HTML5 内置验证（如 required、minlength）和自定义验证（通过 pattern正则表达式）。

3. 元素

- 输入元素：<input>
  - 【语法结构】
    - <input>是自闭合元素，通过属性定义类型、名称、值等。

```
<input
  type="输入类型"       <!-- 必需，定义输入类型（如 text、password） -->
  id="唯一标识"         <!-- 推荐，用于关联 <label> 或 JavaScript 操作 -->
  name="提交键名"       <!-- 必需，提交到服务器时的参数名 -->
  value="初始值"        <!-- 可选，输入框的初始内容 -->
  required              <!-- 可选，标记为必填（未填时阻止提交） -->
  minlength="6"         <!-- 可选，最小输入长度（仅 text/password 等有效） -->
  maxlength="20"        <!-- 可选，最大输入长度 -->
  pattern="\d{11}"      <!-- 可选，正则表达式验证（仅 text 等有效） -->
  placeholder="提示文本" <!-- 可选，输入前的提示文字 -->
  disabled              <!-- 可选，禁用输入（不可编辑、不可提交） -->
  readonly              <!-- 可选，只读（可查看但不可编辑） -->
>

<!-- 文本输入 -->
<input type="text" id="username" name="username" placeholder="请输入用户名">

<!-- 密码输入 -->
<input type="password" id="password" name="password" minlength="6" required>

<!-- 复选框 -->
<input type="checkbox" id="agree" name="agree" value="yes" checked>

<!-- 单选框 -->
<input type="radio" id="male" name="gender" value="male">
<input type="radio" id="female" name="gender" value="female">

<!-- 文件上传 -->
<input type="file" id="avatar" name="avatar" accept="image/*">

<!-- 提交按钮 -->
<input type="submit" value="提交">
```

3. 元素

- 输入元素：<input>
  - 【核心属性】
    - 通用属性

属性	说明	示例	注意事项
type	必需：定义输入类型（决定输入框的交互方式和验证规则）。	type="text"（文本输入） type="password"（密码输入） type="checkbox"（复选框）	常见类型：email（邮箱）、tel（电话）、date（日期）、file（文件）。
id	为输入元素指定唯一标识（用于 <label>关联或 JavaScript 操作）。	id="username"	必须与 <label for>属性值一致（显式关联元素）。
name	必需：定义输入元素提交到服务器时的键名（与 <form>的 name无关）。	name="username"（提交时参数名为 username）	同一表单中同名 name的输入会合并为数组（如复选框）。
value	定义输入的初始值（仅部分类型有效，如 text、hidden、checkbox）。	value="默认文本"（文本输入初始显示） value="yes"（复选框默认选中）	复选框/单选框需通过 checked属性设置选中状态（value为提交值）。
required	HTML5 验证：标记输入为必填（未填写时阻止提交，浏览器提示“请填写此字段”）。	required	适用于所有输入类型（如 text、email、file）。
minlength	HTML5 验证：定义输入的最小字符数（仅 text、password等有效）。	minlength="6"（密码至少6位）	最大长度用 maxlength（如 maxlength="20"）。
maxlength	HTML5 验证：定义输入的最大字符数（包括空格和换行）。	maxlength="200"（最多200字）	超出限制时无法输入，浏览器会提示。
pattern	HTML5 验证：定义输入的正则表达式（仅 text等有效）。	pattern="\d{11}"（11位数字，如手机号）	需配合 title属性说明格式（如 title="请输入11位手机号"）。
placeholder	定义输入的提示文本（输入前显示，输入后消失）。	placeholder="请输入邮箱"	不支持换行，推荐简短提示（如“手机号：138-XXXX-XXXX”）。
disabled	禁用输入元素（不可编辑、不可提交）。	disabled	禁用后 name不会被提交到服务器。
readonly	输入元素只读（可查看但不可编辑）。	readonly	值仍会被提交到服务器（与 disabled不同）。
autocomplete	控制浏览器自动完成功能（on或 off）。	autocomplete="off"（禁用密码自动完成）	敏感字段（如密码）建议禁用，非敏感字段启用（提升用户体验）。

3. 元素

- 输入元素：<input>
  - 【核心属性】
    - 类型专属属性

类型	专属属性	说明	示例
checkbox	checked	标记复选框为默认选中状态（仅对第一个匹配项有效）。	<input type="checkbox" checked>
radio	checked	标记单选框为默认选中状态（仅对第一个匹配项有效）。	<input type="radio" checked>
file	accept	限制上传文件类型（如 image/*仅允许图片）。	<input type="file" accept="image/png">
number	min/max	定义数值的最小/最大值（如 min="0"、max="100"）。	<input type="number" min="0" max="100">
date	min/max	定义日期的最小/最大值（如 min="2023-01-01"）。	<input type="date" min="2023-01-01">

3. 元素

- 输入元素：<input>
  - 【使用建议】
    - 根据输入内容选择合适类型（如 email自动验证格式，date弹出日期选择器），减少用户输入错误。
    - 每个 <input>必须通过 <label for="id">关联（如 <label for="username">用户名：</label>），提升可访问性。
    - 优先使用 HTML5 内置验证（如 required、pattern），复杂验证（如跨字段验证）需配合 JavaScript。
    - 限制文件类型（accept="image/png,image/jpeg"）和大小（通过 JavaScript 验证），避免服务器压力。

3. 元素

- 辅助元素：<label>
  - 【元素简介】
  - <label>是输入元素的辅助元素（Inline-Level Element），用于为 <input>、<select>、<textarea>等元素提供描述性文本，帮助用户（尤其是视障用户）理解输入的用途，其核心价值在于：
  - 可访问性提升：屏幕阅读器（如 NVDA、VoiceOver）会朗读 <label>的内容，明确告知用户当前操作的目标（如“用户名输入框”），避免视障用户混淆输入用途。
  - 用户体验优化：清晰的元素文本可减少用户输入错误（如避免因输入框无提示而填错信息）。
  - 键盘导航支持：配合 accesskey属性，可通过键盘快捷键快速聚焦输入元素，提升操作效率。

3. 元素

- 辅助元素：<label>
  - 【语法结构】
    - <label>有两种关联方式：
    - 显式关联：通过 for属性匹配输入元素的 id（推荐）。
    - 隐式关联：将输入元素直接包裹在 <label>内部（适用于简单结构）。
      - 推荐使用 for和 id显式关联，避免隐式关联可能导致的可访问性问题（如嵌套多层元素时，屏幕阅读器可能无法正确识别）。
      - for属性的值必须与输入元素的 id完全一致（区分大小写），否则关联失效。

```
<!-- 显式关联（推荐） -->
<label for="username">用户名：</label>
<input type="text" id="username" name="username">

<!-- 隐式关联 -->
<label>
  密码：
  <input type="password" id="password" name="password">
</label>
```


3. 元素

- 辅助元素：<label>
  - 【核心属性】

属性	说明	示例	注意事项
for	显式关联：指定要关联的输入元素的 id（必须与输入元素的 id一致）。	for="username"	必须与输入元素的 id唯一对应。
accesskey	定义键盘快捷键（通过 Alt/Ctrl+ 字符聚焦输入元素）。	accesskey="u"（按 Alt+U 聚焦）	字符需唯一，避免与浏览器默认快捷键冲突（如 Ctrl+S保存）。
hidden	隐藏元素（仅用于屏幕阅读器，视觉上不可见）。	hidden	仅在特殊场景使用（如屏幕阅读器需要但用户无需查看）。

  - 【使用建议】
    - 始终使用 for和 id显式关联元素和输入元素，避免隐式关联导致的可访问性问题。
    - 元素文本需清晰描述输入用途（如“请输入手机号”比“手机号：”更友好）。

3. 元素

- 辅助元素：<select>与<option>
  - 【元素简介】
    - <select>和 <option>是 HTML 中用于创建下拉选择框的核心元素，主要用于让用户从多个预设选项中选择一个或多个值，广泛应用于表单数据收集（如性别、国家、分类选择等）。
    - <select>：定义一个块级容器（Block-Level Element），用于包裹一组选项（<option>），并通过 name属性标识提交到服务器时的键名。它支持单选（默认）或多选（通过 multiple属性开启）。
    - <option>：定义 <select>中的具体选项（Inline-Level Element），每个 <option>对应一个可选值，其 value属性是提交到服务器的实际数据，元素文本是用户可见的选项描述。
    - 其核心价值在于：
    - 数据收集高效：通过预设选项限制用户输入范围，减少无效数据（如强制选择“男/女”而非自由输入）。
    - 交互友好：下拉框节省页面空间，用户可通过点击展开选项列表，操作便捷。
    - 语义化支持：配合 label元素关联，提升可访问性（屏幕阅读器可朗读选项内容）。

3. 元素

- 辅助元素：<select>与<option>
  - 【语法结构】
    - <select>内部包含多个 <option>，支持单选或多选（通过 multiple属性）

```
<label for="country">国家：</label>
<select id="country" name="country" required>
  <option value="">请选择国家</option> <!— 默认空选项 —>
  <option value="china">中国</option>
  <option value="usa">美国</option>
</select>

<!— 多选下拉框 —>
<select id="hobbies" name="hobbies" multiple size="3">
  <option value="reading">阅读</option>
  <option value="sports">运动</option>
</select>
```

3. 元素

- 辅助元素：<select>与<option>
  - 【核心属性】
    - <select>和 <option>均继承 HTML 全局属性（如 id、class、style）
    - <select>核心属性

属性	说明	示例	注意事项
name	必需：定义提交到服务器时的键名（与 <input>的 name作用相同）。	name="country"	同一表单中同名 name的 <select>会合并为数组（多选时）。
id	为下拉框指定唯一标识（用于 <label>关联或 JavaScript 操作）	id="country"	必须与 <label for>属性值一致（显式关联）。
multiple	允许选择多个选项（需配合 size属性显示可见选项数）。	multiple	用户需按住 Ctrl（Windows）或 Cmd（Mac）选择多个选项。
size	定义下拉框可见的选项数（仅 multiple有效，否则显示所有选项）。	size="3"（显示3个选项）	若 size小于选项总数，自动添加滚动条。
required	HTML5 验证：标记下拉框为必填（未选择时阻止提交）。	required	需确保至少有一个非空 <option>（如默认空选项 value=""无效）。
disabled	禁用下拉框（不可选择）。	disabled	禁用后 name不会被提交到服务器。

3. 元素

- 辅助元素：<select>与<option>
  - 【核心属性】
    - <option>核心属性

属性	说明	示例	注意事项
value	必需：定义选项提交到服务器时的实际值（若未指定，默认取元素文本内容）。	value="china"（提交“china”）	若元素文本为“中国”，但 value未设置，则提交“中国”。
selected	定义选项为默认选中状态（仅对第一个匹配项有效）。	selected	若多个 <option>有 selected，仅第一个生效。
disabled	禁用选项（不可选择）。	disabled	禁用后选项不可见或不可选（取决于浏览器样式）。
label	HTML5 新增：为选项定义“屏幕阅读器专用元素”（覆盖元素文本内容）。	<option value="china" label="中国（简体）">中国</option>	提升视障用户的可访问性（屏幕阅读器朗读 label内容）。

3. 元素

- 辅助元素：<select>与<option>
  - 【使用建议】
    - 添加一个空值选项（如 <option value="">请选择</option>），避免用户误选（尤其必填场景）。
    - value应使用简洁的标识（如 china），元素文本使用用户友好的描述（如“中国”）。
    - 多选下拉框需通过文本（如“按住 Ctrl 选择多个”）或图标提示用户操作方式。
    - 为 <select>添加 <label>关联（for属性匹配 id），屏幕阅读器会朗读选项数量和当前选中项。
    - 样式自定义：通过 CSS 调整下拉框样式（如 appearance: none移除默认样式），但需保留基本交互功能（如展开/收起）。

3. 元素

- 辅助元素：<textarea>
  - 【元素简介】
    - <textarea>是 HTML 中用于多行文本输入的双元素（Inline-Level Element），允许用户输入大段连续文本（如评论、个人简介、备注等）。它是表单数据收集中处理长文本的核心工具，支持换行、滚动和自定义尺寸，是用户与网页交互的重要组件。
  - 【语法结构】
    - <textarea>是双元素，通过属性定义行数、列数和初始值。

```
1. <label for="bio">个人简介：</label>
2. <textarea id="bio" name="bio" rows="4" cols="50" maxlength="200" placeholder="请输入1-200字简介"></textarea>
```

    - 双元素闭合：<textarea>必须以 </textarea>闭合，不可自闭合（如 <textarea />不符合规范）。
    - 初始内容：元素内的文本（如 Hello World）会作为用户输入前的默认值显示。
    - 属性顺序：属性顺序不影响功能，但推荐按 id、name、rows、cols、maxlength等逻辑顺序排列。

3. 元素

- 辅助元素：<textarea>
  - 必选属性

属性	说明	示例
name	必需：定义输入内容提交到服务器时的键名（与 <input>的 name作用相同）。	name="bio"（提交时键名为 bio）
id	为 <textarea>指定唯一标识（用于 <label>关联或 JavaScript 操作）。	id="bio"（与 <label for="bio">关联）

  - 可选属性

属性	说明	示例	注意事项
rows	定义文本区域的可见行数（垂直高度）。	rows="4"（显示4行）	实际高度可能因字体大小调整，建议配合 CSS height属性。
cols	定义文本区域的可见列数（水平宽度）。	cols="50"（显示50字符宽度）	实际宽度可能因字体大小调整，建议配合 CSS width属性。
maxlength	HTML5 验证：定义输入的最大字符数（包括空格和换行）。	maxlength="200"（最多200字）	超出限制时无法输入，浏览器会提示。
minlength	HTML5 验证：定义输入的最小字符数（包括空格和换行）。	minlength="10"（至少10字）	未满足时浏览器会提示。
placeholder	定义输入的提示文本（输入前显示，输入后消失）。	placeholder="请输入简介"	不支持换行，推荐简短提示。
disabled	禁用文本区域（不可编辑、不可提交）。	disabled	禁用后 name不会被提交到服务器。
readonly	文本区域只读（可查看但不可编辑）。	readonly	值仍会被提交到服务器。
wrap	定义换行方式（soft软换行/hard硬换行）。	wrap="soft"（默认，换行符不提交）	hard会提交换行符（需服务器支持）。

3. 元素

3.5 表单元素：实现用户交互

- 辅助元素：<textarea>
  - 【使用建议】
    - 根据内容长度设置 rows和 cols（如 rows="5"适用于长文本），避免过小导致滚动频繁。
    - 使用 maxlength限制最大字符数（如评论不超过 500 字），减少服务器压力。

3. 元素

- 辅助元素：<button>
  - 【元素简介】
    - <button>是 HTML 中用于创建可点击交互按钮的双标签元素（Inline-Level Element），核心作用是触发用户操作（如提交表单、打开链接、执行 JavaScript 函数等）。它是网页中最常见的交互组件之一，广泛应用于表单提交、模态框控制、功能触发等场景，其核心特点：
    - 高度灵活：可包含文本、图标或其他 HTML 元素（如 <i>、<svg>），支持自定义内容。
    - 交互多样：通过 type属性定义按钮类型（提交、普通、重置），或通过 onclick等事件绑定自定义逻辑。
    - 语义化支持：配合 aria-*属性（如 aria-label）提升可访问性，帮助视障用户理解按钮功能。
  - 【语法结构】
    - <button>是双标签（需显式闭合），内部可包含文本、图标或其他 HTML 元素

```
<!-- 基础文本按钮 -->
<button type="button" onclick="handleClick()">点击我</button>

<!-- 包含图标的按钮 -->
<button type="submit">
  <i class="icon-submit"></i> 提交表单
</button>

<!-- 禁用状态的按钮 -->
<button type="button" disabled>不可点击</button>
```


3. 元素

- 辅助元素：<button>
  - 通用属性

属性	说明	示例	注意事项
type	定义按钮类型（决定默认行为）。	type="button"（普通按钮） type="submit"（提交表单） type="reset"（重置表单）	默认值 type="submit"（在 <form>内时），需显式设置 type="button"避免意外提交。
id	为按钮指定唯一标识（用于 JavaScript 操作或 CSS 样式）。	id="submitBtn"	推荐使用有意义的名称（如 deleteBtn、modalOpenBtn）。
class	为按钮添加 CSS 类（用于样式控制）。	class="primary-btn"	配合 CSS 定义按钮颜色、尺寸、悬停效果等。
disabled	禁用按钮（不可点击、不可触发事件）。	disabled	禁用后按钮变灰（默认样式），cursor: not-allowed。
aria-label	无障碍属性：为屏幕阅读器提供按钮的描述（覆盖可见文本）。	aria-label="关闭模态框"	提升视障用户的可访问性（如按钮仅含图标时）。
aria-disabled	无障碍属性：反映按钮的禁用状态（与 disabled属性同步）。	aria-disabled="true"	当 disabled为 true时，建议同步设置此属性。

3. 元素

- 辅助元素：<button>
  - 功能专属属性（事件绑定）

属性	说明	示例	注意事项
onclick	绑定鼠标点击事件（执行 JavaScript 函数）。	onclick="showModal()"	推荐通过 addEventListener绑定事件（更灵活，支持移除）。
onmouseover	绑定鼠标悬停事件。	onmouseover="this.style.color='red'"	需谨慎使用，可能影响性能（推荐 CSS :hover替代）。
onkeydown	绑定键盘按下事件（如 Enter键触发）。	onkeydown="if(event.key==='Enter') submitForm()"	提升键盘用户的操作体验。

3. 元素

- 辅助元素：<button>
  - 【使用建议】
    - 明确按钮类型（type属性）
      - type="button"：普通按钮（默认行为为无操作，需通过 onclick绑定逻辑）。
      - type="submit"：提交表单（仅在 <form>内有效，触发表单提交到 action地址）。
      - type="reset"：重置表单（仅在 <form>内有效，清空所有输入字段）。
    - 注意：在 <form>内未显式设置 type时，<button>默认类型为 submit，可能导致意外提交（如点击按钮时触发表单提交）。
    - 提升可访问性
      - 若按钮仅含图标（如 ），需通过 aria-label="删除项目"为屏幕阅读器提供描述。
      - 确保按钮可通过 Tab键聚焦（默认支持），并响应 Enter或 Space键触发点击事件（无需额外代码，浏览器默认支持）。
      - 设置 disabled属性时，同步更新 aria-disabled="true"，确保屏幕阅读器正确识别。
    - 事件绑定的最佳实践
      - 避免内联 onclick：推荐通过 JavaScript 的 addEventListener绑定事件（如 button.addEventListener('click', handleClick)），便于事件管理和移除。
      - 防重复提交：提交表单的按钮（type="submit"）可通过 disabled属性禁用（如点击后禁用，防止重复提交）。

```
1. const submitBtn = document.getElementById('submitBtn');
2. submitBtn.addEventListener('click', async () => {
3.   submitBtn.disabled = true; // 禁用按钮
4.   await fetch('/submit');    // 模拟提交
5.   submitBtn.disabled = false; // 提交完成后启用
6. });
```

3. 元素

- 辅助元素：<progress>
  - 【元素简介】
    - <progress>是 HTML5 引入的进度指示标签（Block-Level Element），用于可视化展示任务的完成进度（如文件上传、下载、数据处理等）。它通过动态更新数值或进度条样式，直观反馈用户操作的进展状态，是提升用户体验的重要组件，其核心价值在于：
    - 用户反馈：明确告知用户任务进度（如“上传中：50%”），减少用户等待焦虑。
    - 语义化支持：通过 max和 value属性定义进度范围，帮助屏幕阅读器（如 NVDA）理解进度含义。
    - 灵活定制：可通过 CSS 自定义进度条样式（颜色、尺寸、动画等），适配不同设计需求。
  - 【语法结构】
    - <progress>是块级容器，通过 max（总进度）和 value（当前进度）属性定义进度范围和当前状态。

```
<!-- 基础进度条 -->
<progress id="uploadProgress" max="100" value="50"></progress>

<!-- 结合文本显示 -->
<div class="progress-container">
  <progress id="downloadProgress" max="100" value="30"></progress>
  <span id="progressText">30%</span>
</div>
```


3. 元素

- 辅助元素：<progress>
  - 【核心属性】
    - <progress>继承 HTML 全局属性（如 id、class、style），以下是其特有的核心属性。

属性	说明	示例	注意事项
max	必需：定义进度的总范围（数值或百分比）。	max="100"（总进度100%） max="500"（总进度500单位）	必须大于 0，否则进度条无法显示。
value	必需：定义当前进度值（需介于 0和 max之间）。	value="50"（当前进度50%） value="250"（当前进度250/500）	若 value超出 0或 max，进度条会显示为满或空。
aria-label	无障碍属性：为屏幕阅读器提供进度条的描述（如“文件上传进度”）。	aria-label="文件上传进度"	提升视障用户的可访问性（必加，尤其当进度条无文本说明时）。
aria-valuenow	无障碍属性：同步 value的当前值（供屏幕阅读器读取）。	aria-valuenow="50"	需与 value属性值一致（通常通过 JavaScript 动态更新）。
aria-valuemin	无障碍属性：定义进度的最小值（默认 0）。	aria-valuemin="0"	通常无需显式设置（默认已支持）。
aria-valuemax	无障碍属性：定义进度的最大值（默认等于 max属性）。	aria-valuemax="100"	通常无需显式设置（默认已支持）。

3. 元素

- 辅助元素：<progress>
  - 【使用建议】
    - 显式设置 max和 value
      - 必须为 max设置合理的总进度值（如 max="100"表示百分比，max="500"表示文件大小单位），避免默认值（max="1"）导致的进度显示异常。
      - value需动态更新（通过 JavaScript），确保与实际任务进度同步（如上传文件时，value随已上传字节数变化）。
    - 进度条本身仅显示图形化进度，建议搭配文本（如“50%”）或图标（如“上传中”）明确说明进度含义，避免用户误解。

```
<div class="progress-wrapper">
  <progress id="uploadProgress" max="100" value="50"></progress>
  <span class="progress-text">50% 已上传</span>
</div>
```

    - 通过 JavaScript 监听任务进度（如文件上传的 progress事件），实时更新 value属性，实现进度条动态变化

```
<progress id="uploadProgress" max="100" value="0"></progress>
<span id="progressText">0%</span>
<script>
  const uploadProgress = document.getElementById('uploadProgress');
  const progressText = document.getElementById('progressText');
  let currentProgress = 0;
  // 模拟上传进度（实际场景中替换为真实的上传事件）
  const uploadInterval = setInterval(() => {
    currentProgress += Math.random() * 10; // 随机增加进度
    if (currentProgress > 100) currentProgress = 100;

    uploadProgress.value = currentProgress;
    progressText.textContent = `${Math.round(currentProgress)}%`;

    if (currentProgress === 100) {
      clearInterval(uploadInterval);
      progressText.textContent = '上传完成！';
    }
  }, 500);
</script>
```


3. 元素

- 表单分组：<fieldset>与 <legend>
  - 【元素简介】
    - <fieldset>和 <legend>是 HTML 中用于表单分组的语义化元素，核心作用是将逻辑相关的表单元素（如输入框、选择框等）包裹为一个整体，并为该组提供明确的标题描述。
    - <fieldset>：定义一个块级容器（Block-Level Element），用于包裹一组逻辑相关的表单元素（如“个人信息”“联系方式”）。它通过默认的边框样式（浏览器默认）或 CSS 自定义样式，视觉上区分不同分组。
    - <legend>：定义 <fieldset>的标题（Legend），是 <fieldset>的子元素，用于为分组提供语义化的描述（如“用户基本信息”）。它是屏幕阅读器（如 NVDA、VoiceOver）识别分组内容的关键元素。
    - 其核心价值在于：
    - 语义明确：通过 <fieldset>和 <legend>标记表单分组，帮助浏览器、搜索引擎（SEO）和辅助技术（如屏幕阅读器）理解表单的结构和逻辑。
    - 可访问性提升：屏幕阅读器会朗读 <legend>的内容，明确告知用户当前操作的分组（如“用户基本信息：姓名、邮箱”），避免用户混淆。
    - 样式可控：通过 CSS 自定义 <fieldset>的边框、内边距等样式，统一表单分组的视觉风格。

3. 元素

- 表单分组：<fieldset>与 <legend>
  - 【语法结构】
    - <fieldset>是块级容器，内部可包含任意表单元素（如 <input>、<select>、<textarea>）和 <legend>（必须作为直接子元素）。

```
<form>
  <!-- 分组1：基本信息 -->
  <fieldset>
    <legend>基本信息</legend> <!-- 分组标题 -->
    <label for="username">用户名：</label>
    <input type="text" id="username" name="username" required>

    <label for="email">邮箱：</label>
    <input type="email" id="email" name="email" required>
  </fieldset>

  <!-- 分组2：联系方式 -->
  <fieldset>
    <legend>联系方式</legend> <!-- 分组标题 -->
    <label for="phone">手机号：</label>
    <input type="tel" id="phone" name="phone" pattern="\d{11}" required>

    <label for="address">地址：</label>
    <textarea id="address" name="address" rows="3" maxlength="200"></textarea>
  </fieldset>
</form>
```

    - <legend>必须是 <fieldset>的直接子元素：否则屏幕阅读器无法正确关联标题与分组内容。
    - 分组逻辑：<fieldset>应包裹逻辑相关的表单元素（如同一类信息或同一功能模块），避免随意分组（如将姓名和密码分到不同组）。

3. 元素

- 表单分组：<fieldset>与 <legend>
  - 【核心属性】
    - <fieldset>和 <legend>均继承 HTML 全局属性（如 id、class、style），但 <fieldset>有专属属性，<legend>无专属属性。
    - <fieldset>核心属性

属性	说明	示例	注意事项
disabled	禁用整个分组（分组内所有表单元素不可编辑、不可提交）。	disabled	禁用后，分组内元素的 name不会被提交到服务器。
name	为分组指定名称（用于 JavaScript 或服务器端标识）。	name="userInfo"	推荐使用有意义的名称，避免与 id重复。

3. 元素

- 表单分组：<fieldset>与 <legend>
  - 【使用建议】
    - 强制使用 <legend>，每个 <fieldset>必须包含一个 <legend>（文本内容），否则屏幕阅读器无法识别分组的用途（如“用户基本信息”）。
    - 语义化分组
      - 按逻辑分组：将同一类信息（如“个人信息”“支付信息”）或同一功能模块（如“注册步骤1”“注册步骤2”）包裹在 <fieldset>中。
      - 避免过度分组：无需为每个输入框单独分组（如单个输入框不需要 <fieldset>），仅在需要明确区分的复杂表单中使用。
    - 提升可访问性
      - 屏幕阅读器友好：<legend>的内容需简洁明确（如“联系方式：手机号、地址”），避免模糊表述（如“信息”）。
      - 键盘导航：确保 <fieldset>内的表单元素可通过 Tab键顺序聚焦，<legend>本身不参与焦点，但会引导用户理解分组上下文。
    - 结合表单验证，若需临时禁用一组输入（如用户未勾选同意条款时禁用提交），可通过 <fieldset disabled>禁用整个分组，无需逐个禁用输入元素。

案例演示

- 案例：产品用户反馈表单
  - 构建产品用户反馈表单，收集用户对产品的意见与建议。
  - 【案例目标】
    - 掌握HTML表单核心元素（<form>、<input>、<select>等）的用法
    - 实现用户输入验证与交互反馈
    - 设计清晰的无障碍表单结构
    - 模拟真实场景下的用户数据提交流程

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所