Learning

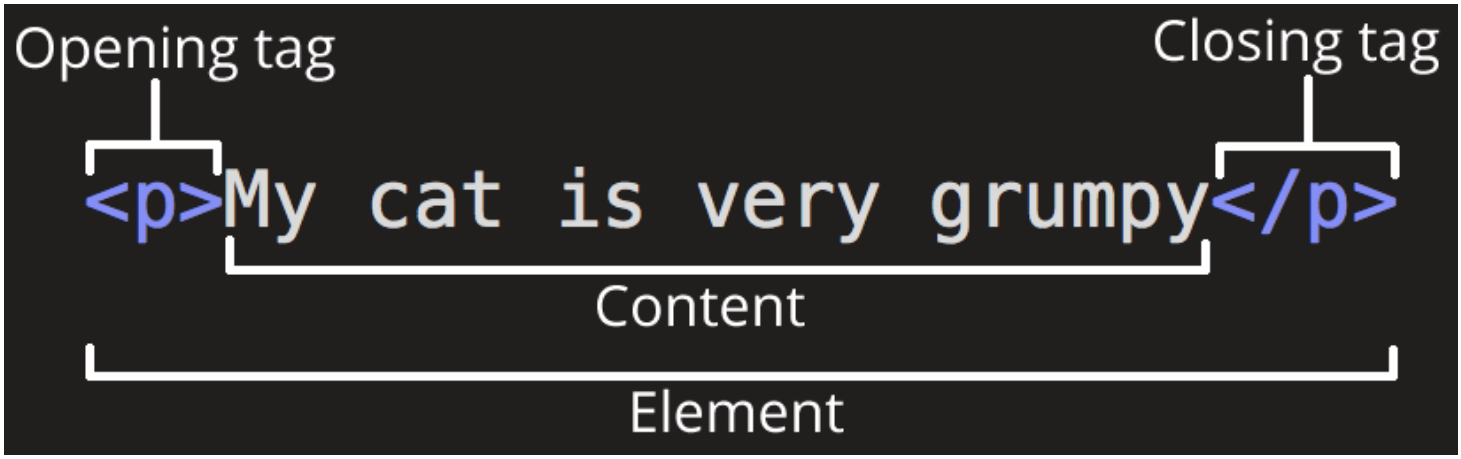
HTMLearning

HTML是一种超文本标记语言，用于告知浏览器组织页面，有一系列的元素构成，通过元素包围不同部分的内容使以一定的方式呈现或工作，标签能够为文字或图片添加超链接等等设置

使用<p><p>包围文字形成段落

标签不区分大小写

基础剖析



使用开始标签与结束标签包围内容，这一整个结构就称之为元素

嵌套

在元素放在别的元素之中，称之为嵌套

```
<p>Hello <strong>World</strong> !</p>
```

块级元素和内联元素

块级元素在页面中以块的形式展现 —— 相对于其前面的内容它会出现在新的一行，其后的内容也会被挤到下一行展现。块级元素通常用于展示页面上结构化的内容，例如段落、列表、导航菜单、页脚等等。一个以 block 形式展现的块级元素不会被嵌套进内联元素中，但可以嵌套在其它块级元素中。
内联元素通常出现在块级元素中并环绕文档内容的一小部分，而不是一整个段落或者一组内容。内联元

素不会导致文本换行：它通常出现在一堆文字之间，例如超链接元素 `<a>` 或者强调元素： `<em>` 和 `<strong>` 。

空元素

部分元素只拥有一个标签，用于在说在所在位置插入、嵌入内容，如元素 `<img>` 用于在该位置插入一张指定的图片

如：

```
<img src="/HTMLLearning/images/test.jpg">
```

属性

属性包括元素的额外信息，如上图中，class属性给元素赋值了一个识别的名字（id），可以用来识别此元素的样式和其他信息

一个属性必须包含如下内容：
一个空格，在属性和元素名称之间（如果已经有一个或多个属性，就与前一个属性之间有一个空格）。
属性名称，后面跟着一个等于号。
一个属性值，由一对引号 ("") 引起来。

属性名	解释
href	超链接的web地址
title	超链接上的额外信息
target	呈现方式

布尔属性

有时你会看到没有值的属性，它是合法的。这些属性被称为布尔属性，他们只能有跟它的属性名一样的属性值。例如disabled 属性，他们可以标记表单输入使之变为不可用 (变灰色)，此时用户不能向他们入任何数据。

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

<!-- 使用 disabled 属性来防止终端用户输入文本到输入框中 -->
<input type="text" disabled>

<!-- 下面这个输入框没有 disabled 属性，所以用户可以向其中输入 -->
<input type="text">
```

格式

添加对应值时需要加上引号，单引号与双引号均可，但需要避免混用
HTML中所有文本的多个连续空格在渲染时都将视作为一个空格

特殊字符

在HTML中，某些字符时特殊字符，会被语法所调用，因此使用字符引用替代

原始字符	等价字符引用
<	<
>	>
"	&qout;
'	'
&	&amp

HTML注释

在HTML中使用 `<!-->` 与 `-->` 包括起来作为注释语言

```
<!-- <p>here i am</p>-->
here i am
```

剖析HTML文档

```
<!DOCTYPE html>                          //声明文档类型
<html>                                   //根元素，包含整个页面
  <head>                                 //关键词/页面描述等信息，不出现在正文之中的信息
    <meta charset="utf-8">               //使用字符集
    <title>我的测试站点</title>          //页面标题
  </head>
  <body>                                 //主体显示内容
    <p>这是我的页面</p>
  </body>
</html>
```

head标签与元数据

head标签

head标签内的内容不会显示在页面之中，用于保存页面内的元数据

example：添加标题 使用 `<title>` 元素

元数据 <meta>元素

元数据：描述数据的数据

修改字符编码

example：字符集使用 `utf-8` 编码避免出现乱码的情况

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

添加作者和描述

example：使用 `name` 和 `content` 属性指定元素类型与内容

```
<meta name="author" content="Chris Mills">
<meta name="description" content="The MDN Web Docs Learning Area aims to provide
complete beginners to the Web with all they need to know to get
started with developing web sites and applications.">
```

通过在head内的description能够在搜索引擎内寻找到网站

添加自定义图标

使用 `favicon` 标签自定义浏览器icon

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

icon放置在image文件夹下

使用CSS和JavaScript丰富HTML元素

CSS使用<link>元素，Jscript使用<script>元素

<link>

通常位于文档头部，拥有两个属性 `rel` 与 `href`

```
<link rel="stylesheet" href="my-css-file.css">
```

`rel=stylesheet` 表明这是文档的样式表
`href` 包含样式表的文件路径

<script>

非强制放于 `head` 之中，包含 `src` 属性指向需要加载的JavaScript文件路径，同时加上 `defer` 以在HTML加载完之后加载HTML

```
<script src="my-js-file.js" defer></script>
```

设定主语言

使用 `<html lang="zh_CN">` 设置语言为中文

同时也可以针对分段设置为不同的语言

```
<p>Japanese example: <span lang="ja">ご飯が熱い。</span>.</p>
```

文字处理

排版处理

通过使用 `<p>` 元素标签对段落进行定义

```
<p>这是一个段落</p>
```

使用“标题标签”定义标题

```
<h1>这是一个标题</h1>
```

其中标题元素一共有六个标签，从h1到h6，分别代表不同层级的内容，需要严明各段的内容，便于阅读
层级演示文件

文字格式处理

斜体强调

使用 `<em>` 标记斜体字表强调语义

```
<p>I am <em>glad</em> you weren't <em>late</em>.</p>
```

此外还可以通过 `<span>` 和CSS，以及 `<i>` 达到斜体风格的效果

粗体非常重要

使用 `<strong>` 标记粗体字强调重要性

```
<p>This liquid is <strong>highly toxic</strong>.</p>

<p>I am counting on you. <strong>Do not</strong> be late!</p>
```

此外还可以通过 `<span>` 和CSS，以及 `<b>` 达到斜体风格的效果

补充

关于 `<b>`，`<i>`，`<u>` 三个元素，仅影响表象而不含有语义，适用于在没有更加合适的元素的情况下

`<i>` 被用来传达传统上用斜体表达的意义：外国文字，分类名称，技术术语，一种思想……
`<b>` 被用来传达传统上用粗体表达的意义：关键字，产品名称，引导句……
`<u>` 被用来传达传统上用下划线表达的意义：专有名词，拼写错误……

列表

无序列表

无序列表用于记录顺序无关紧要的项目
使用 `<ul>` 元素开始记录内容，并使用 `<li>` 将所有单个项目包围

```
<ul>
  <li>豆浆</li>
  <li>油条</li>
  <li>豆汁</li>
  <li>焦圈</li>
</ul>
```

有序列表

有序列表需要按照项目的顺序列出来，有强顺序关联性
结构类似于无序列表，不同之处在于需要使用 `<ol>` 将项目包裹

```
<ol>
  <li>沿着条路走到头</li>
  <li>右转</li>
  <li>直行穿过第一个十字路口</li>
  <li>在第三个十字路口处左转</li>
  <li>继续走 300 米，学校就在你的右手边</li>
</ol>
```

练习：列表个人练习

嵌套列表

在一个列表之中嵌入另一个列表是同样可行的

超链接

使用超链接将文档连接到任何其他文档
使用 `<a>` 元素，并给予一个 `href` 属性以创建链接

```
<p>chilck
<a href="https://www.google.com"> here </a>to jumo to Google
</p>
```

title属性

添加title属性氪补充链接信息，包括即将跳转的网页等信息（鼠标放置在该链接上显示的内容）

块级链接

```
<a href="https://www.mozilla.org/zh-CN/">
  <img src="mozilla-image.png" alt="链接至 Mozilla 主页的 Mozilla 标志">
</a>
```

URL与path

指向文件目录的差异

文件路径	差异
指向本级目录	直接应用文件
指向子目录	添加子目录前缀
指向上级目录	添加 `../` 返回上级目录

引导到文档片段

在标题端使用特定的 `id` 值，并在链接中在尾部添加 `#+文档id`

绝对路径与相对路径

一般而言，绝对路径=绝对路径地址+相对路径

绝对路径

绝对路径中是直接指向的路径，可以直接通过域名进行访问

相对路径

相对路径指向不同的位置，取决于当前所在的位置

连接到非HTML资源

下载文件:使用 `download` 属性

```
<a> href="zuiding.top" download="test.txt">点击下载（非链接）
</a>
```

电子邮件链接:使用 `<a>` 与 `mailto:` 结合

```
<a href="mailto:zuiding024@gmail.com">发送邮件</a>
```

电子邮件也可添加详细信息，通过添加subject,c,body内容，其中使用 `?` 分割主URL与参数值，使用 `&` 分割不同参数值

高阶文本排版

描述列表

标记单子项目及其相关描述，通过使用 `<dl>` 包裹，并在每一项中都是用 `<dt>` 闭合，每一个描述使用 `<dd>` 闭合

```
<dl>
  <dt>内心独白</dt>
    <dd>戏剧中，某个角色对自己的内心活动或感受进行念白表演，这些台词只面向观众，而其他角色不会听到。</dd
  <dt>语言独白</dt>
    <dd>戏剧中，某个角色把自己的想法直接进行念白表演，观众和其他角色都可以听到。</dd>
  <dt>旁白</dt>
    <dd>戏剧中，为渲染幽默或戏剧性效果而进行的场景之外的补充注释念白，只面向观众，内容一般都是角色的感受、
</dl>
```

HTML会自动在其之间形缩进，使代码整洁规范

实例

引用

标记引用，不会被显示出来，需要通过使用JS，CSS才能够显示 `cite` 的内容

块级引用

使用 `blockquote` 包裹表示，并在 `cite` 属性内用URL指向引用的资源地址

```
<blockquote cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">
  <p>The <strong>HTML <code>&lt;blockquote&gt;</code> Element</strong> (or <em>HTML Block
  Quotation Element</em>) indicates that the enclosed text is an extended quotation.</p>
</blockquote>
```

在渲染块引用时会增加缩进所谓引用的标志

行内引用

大体相同，使用 `<q>` 元素，其中将作文普通文本放入引号表示引用

```
<p>The quote element — <code>&lt;q&gt;</code> — is <q cite="https://developer.mozilla.org/en-US/
for short quotations that don't require paragraph breaks.</q></p>
```

缩略语

使用 `<abbr>` 包裹一个缩写，并提供解释（包含在 `title` 中）

example:

```
<p>我们使用 <abbr title="超文本标记语言（Hyper text Markup Language）">HTML</abbr> 来组织网页文档。<

<p>第 33 届<abbr title="夏季奥林匹克运动会">奥运会</abbr>将于 2024 年 8 月在法国巴黎举行。</p>
```

标记联系方式

使用 address 记录地址

ex

```
<address>
  <p>Chris Mills, Manchester, The Grim North, UK</p>
</address>
or
<address>
  <p>Page written by <a href="../authors/chris-mills/">Chris Mills</a>.</p>
</address>
```

上标与下标

在一些需要使用到上下标的场景中，使用 <sup> 与 <sub> 标签解决
其中 sub 是下标， sup 是上标

```
<p>咖啡因的化学方程式是 C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>。</p>
<p>如果 x<sup>2</sup> 的值为 9，那么 x 的值必为 3 或 -3。</p>
```

展示计算机代码

代码标签	解释
<code>	标记计算机通用代码
<pre	保留空白字符>
var	标记具体变量名
<kbd>	标记输入电脑的键盘输入（输入示例）
<samp>	标记计算机程序的输出

标记时间和日期

统一事件类型，以便使用不同时间浏览形式的用户均能正确阅读时间，并且允许插入到日历之中
使用 <time> 元素标记
ex:

```
<time datetime="2016-01-20">2016 年 1 月 20 日</time>
```

ex2:

```
<!-- 标准简单日期 -->
<time datetime="2016-01-20">20 January 2016</time>
<!-- 只包含年份和月份-->
<time datetime="2016-01">January 2016</time>
<!-- 只包含月份和日期 -->
<time datetime="01-20">20 January</time>
<!-- 只包含时间，小时和分钟数 -->
<time datetime="19:30">19:30</time>
<!-- 还可包含秒和毫秒 -->
<time datetime="19:30:01.856">19:30:01.856</time>
<!-- 日期和时间 -->
<time datetime="2016-01-20T19:30">7.30pm, 20 January 2016</time>
<!-- 含有时区偏移值的日期时间 -->
<time datetime="2016-01-20T19:30+01:00">7.30pm, 20 January 2016 is 8.30pm in France</time>
<!-- 调用特定的周 -->
<time datetime="2016-W04">The fourth week of 2016</time>
```

文档与网页结构

文档的基本组成部分

页眉： 横跨页面顶部的大标题或者标志
导航栏：指向各个主要区段的超链接
主内容：网站的独有内容
侧边栏：外围辅助信息
页脚：页面底部的额外信息

标签	解释
<header>	页眉
<nav>	导航栏
<main>	主内容
<aside>	侧边栏
<footer>	页脚

HTML嵌入

嵌入图片

使用 `<img>` 属性并且使用一个 `src` 标签指向所需要的图片路径，以此导入图片

备选文本

`alt` 用于对图片的文字描述，在图片因某些原因无法显示时出现对应的文字提示

宽度和高度

调整图片的高度和宽度，提前为后续的HTML文本提供空间，利于页面的加载

图片标题

给图片添加 `title` 属性，提供更多的支持信息
（鼠标聚焦图片时出现的提示行）

CSS背景图片

使用CSS为背景添加图片，但是在语义上不存在任何的含义，没有备选文本，无法被识别，这是同HTML图片的区别所在

嵌入音频和视频

使用 `video` 和 `audio` 标签添加视频和音频

插入视频

`src` ： 同 `<img>`
`controls` ： 用于控制视频和音频的回放功能
其他标签： 一般称作后备内容，在浏览器不支持 `<video>` 标签时显示该内容，使之能够回退

媒体文件的内容一共需要分成多个层次进行传输，包括音频层，视频层，文字层。其中不同的编解码器的格式与用户层面的浏览器各不相同，需要适时选择恰当的编解码器进行操作

新特性：

标签	解释
`width` & `height`	视频的尺寸
`autoplay`	使加载后立刻播放
`loop`	循环播放
`muted`	默认关闭声音
`poster`	指向图像URL，用于播放广告

标签	解释
preload	用于缓冲较大文件，分为3个值（如下）
"none"	不缓冲
"auto"	页面加载后缓存媒体
"metadata"	仅缓冲文件元数据

嵌入音频

<audio> 用法上与 <video> 几乎完全相同，仅存在边框不同的差别

差别

<audio> 标签不支持 weight/height 和 poster 标签，因为没有视觉内容

重置媒体

在JavaScript中调用 load() 方法来重置媒体，如果有多个 <source> 标签指定的媒体来源，浏览器会从选择来源开始重新加载

音轨增删

通过监控媒体源色的音频轨道来侦测音轨被添加或者删除。
通过监听 AudioTrackList(en-US) 对象的 addtrack 时间，可以对音轨的增加做出响应。

显示音轨文本

通过天降音频内容的文本内容来帮助并不想听到或者无法听到的人符组阅读
即：外挂字幕

其他嵌入技术

使用 <iframe> 将整个web网页嵌入到另一个网页之中
该嵌入链接可以在网站的直接分享中找到

属性	解释
allowfullscreen	是否能够设置为全屏
frameborder	是否绘制边框（参数在0/1中选择）
src	指向路径

属性	解释
`width` & `height`	宽高限制
`sandbox`	安全性设置

安全性设置

1.只在有必要时嵌入信息，避免非版权的信息
2.使用HTTPS加密保护
3.始终使用 `sandbox` 属性
4.配置CSP指令
5. `<embed>` 和 `<object>` 元素

	`<embed>`	`<object>`
嵌入内容的网址	src	data
嵌入内容的准确媒体类型	type	type
由插件控制的框的高度和宽度（以 CSS 像素为单位）	height width	height width
名称和值，将插件作为参数提供	具有这些名称和值的 ad hoc 属性	单标签 `<param>` 元素，包含在内 `<object>`
独立的 HTML 内容作为不可用资源的回退	不支持（ `<noembed>` 已过时）	包含在元素 `<object>` 之后 `<param>`

嵌入矢量图形

矢量图形的定义：……
使用指令构建的图形，不会存在放大失真的问题不会出现马赛克化

SVG

SVG是描述矢量图形的XML语言，通过不同的元素来定义图像的形状

`<img>` 引用

使用 `<img>` 元素直接嵌入SVG图片，但是无法成JS和CSS对象

`<svg>` 引用

使用 `<svg></svg>` 标签闭合内容，内含svg代码

优点

1.减少加载时间
2.允许通过CSS修改样式
3.使用CSS交互
4.包含 `<a>` 标签，成为超链接

缺点

1.只适用于在一个地方使用，避免资源密集型维护
2.会增加HTML文件大小
3.不会缓存内联SVG文件
4.因为支持过时的浏览器会增加开销成本

`<iframe>` 插入

同 `<iframe>` 操作

```
<iframe src="triangle.svg" width="500" height="500" sandbox>
    <img src="triangle.png" alt="Triangle with three unequal sides" />
</iframe>
```

响应式图片

通过对不同用户不同分辨率的差分处理，实现不同用户均能得到良好的阅读体验

不同的尺寸

使用 `<srcset>` 和 `<sizes>` 属性

使用以提供更多额外的资源图像和提示

```
<img
  srcset="elva-fairy-480w.jpg 480w, elva-fairy-800w.jpg 800w"
  sizes="(max-width: 600px) 480px,
         800px"
  src="elva-fairy-800w.jpg"
  alt="Elva dressed as a fairy" />
```

`<srcset>`：文件名+空格+固有宽度（单位：w），设置使用逗号分隔
`<sizes>`：一个媒体条件+空格+条件为真时的表达式
即：根据if语句选择对应的执行代码块

不同分辨率

```
<img srcset="elva-fairy-320w.jpg, elva-fairy-480w.jpg 1.5x, elva-fairy-640w.jpg 2x"
     src="elva-fairy-640w.jpg"
     alt="Elva dressed as a fairy" />
```

浏览器自动选择

美术设计

通过美术设计在恰当时显示不同尺寸的图频，通过 `<picture>` 实现

```
<picture>
  <source media="(max-width: 799px)" srcset="elva-480w-close-portrait.jpg" />
  <source media="(min-width: 800px)" srcset="elva-800w.jpg" />
  <img src="elva-800w.jpg" alt="Chris standing up holding his daughter Elva" />
</picture>
```

表格

基础

标签

表格中内容都包含在 `<table>` 标签之中，其中每一个最小的内容容器（即单元格）都是通过 `<td>` 元素创建的
使用 `<tr>` 来达成换行的目的
表格测试

标题栏

使用 `<th>` 添加标题（特殊单元格）
标题测试

横跨多个单元格

通过 clospan 和 rowspan 实现跨越多行多列的单元格
跨越单元格测试

单元格样式

使用 <col> 和 <colgroup> 使所有的列数据样式一致，避免单独设置
创建 <colgroup> 容器，并在容器内添加 <col> 元素（colgroup定义在table下方）
col 定义方法依照每行的顺序决定，如果需要设置第二行及以后，必须在此之前添加对应的 col 空元素
还可以通过设定 span 值，来规定样式应用到表格的列数
col 样例测试

表格的高级特性

标题

使用 <caption> 为表格增加一个标题
将 <caption> 放在 <table> 元素下面，为表格带来标题
有助于大致了解表格所包含的内容

表格结构

添加表头、页脚、正文部分

<thead>	表头	在头部位置
<tfoot>	页脚	在表格底部
<tbody	正文	在head或者foot下面

三者均需要嵌套在 <table> 之中。
结构测试

嵌套表格

在表格内嵌套另一个表格样式，这在一般情况下会带来阅读问题，但另一层面上，也更方便的解释引用信息

`scope`，`id`，`headers`

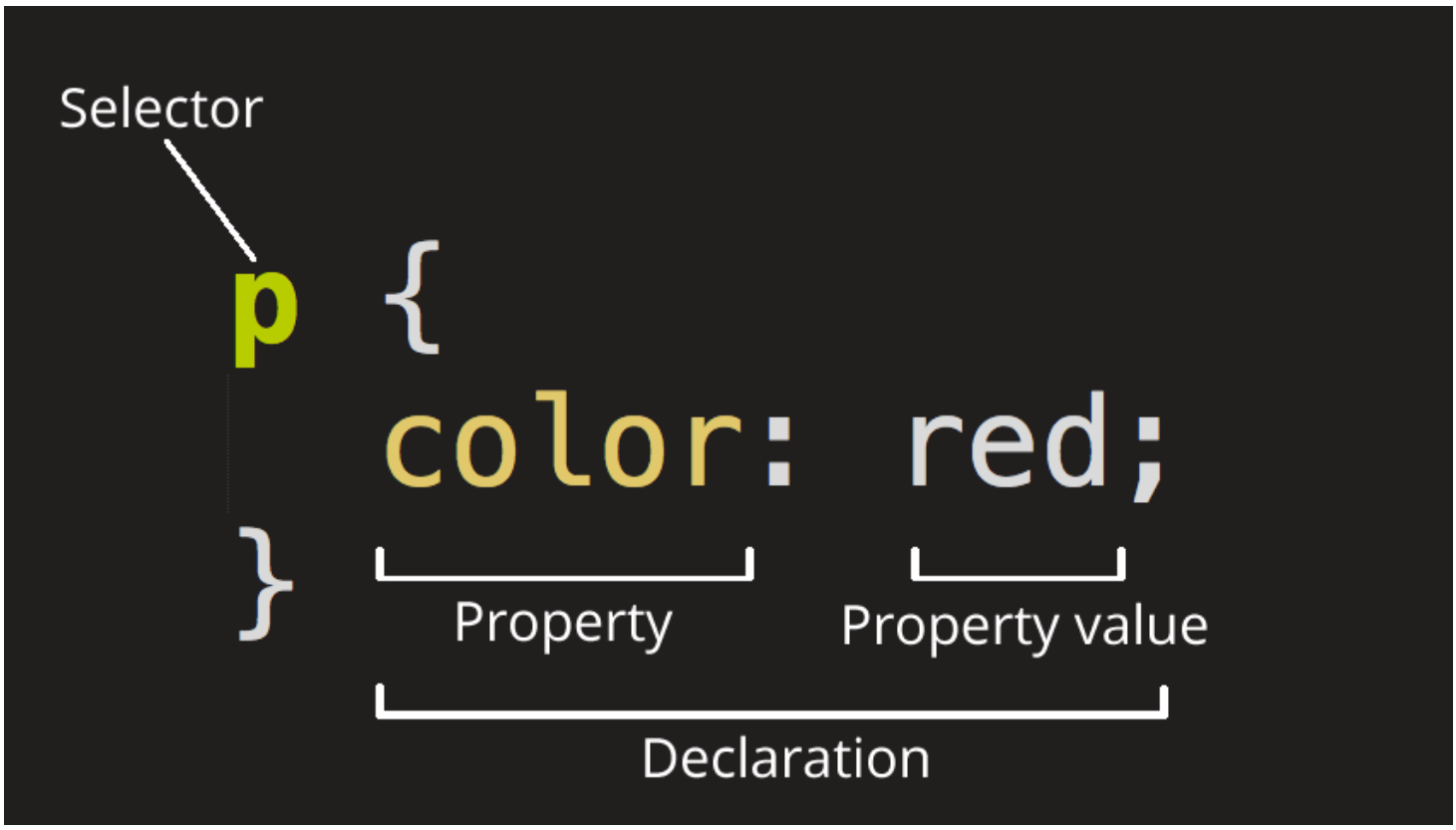
通过 `scope` 属性的定义，可以规定单元格的行标题或列标题，便于阅读器阅读
同理，使用 `id` 和 `headers` 属性结合可以创造标题和单元格之间的联系，进而规定每个单元格的所属关系

CSS

CSS基础

CSS是一种样式表语言，用于为HTML元素添加样式

更改元素的默认行为：使用css添加css规则，以此修改浏览器的默认样式，达成所希望的样式



选择器

基础选择

使用选择器达头表明修改的元素，然后添加声明，声明使用 `{}` 包围，并且在内部添加对应的属性和值

```
p {      //选择器
  color:red;     属性：值，使用冒号赋值，并在结尾添加分号
  width:500px;
  border:1px solid black
}
```

多重选择

在选择器时可选择多个元素添加相同的样式

需要使用css时，需要在html的 `<head>` 语句中加上

```
<link rel="stylesheet" href="style.css">
```

class标签选择

使用类名：在html元素中定义其 `class` 类名，并选中类名即可，在css修改时需要在前面加上 . 以表示引用

```
<ul>
  <li>项目一</li>
  <li class="special">项目二</li>
  <li>项目 <em>三</em></li>
</ul>
```

```
.special {
  color: orange;
  font-weight: bold;
}
```

属性选择

通过对元素上某个标签的属性是否存在进行选择

```
a[href="https://example.com"] { }
```

存否和值选择

选择是否含有元素名的选项

选择器	示例	描述
`[attr]`	`a[title]`	匹配含有该属性的元素
`[attr=value]`	`a[href="……"]`	匹配该属性为value的元素

选择器	示例	描述
[attr~=value]	p[class~="special"]	属性为value，或者至少有一个匹配的元素
[attr|=value]	div[lang|="zh"]	匹配属性为attr，且值初始为value， 紧跟着连字符的元素

子字符串匹配选择器

选择字符包含的元素选项

选择器	示例	描述
[attr]^=value	li[class^="box-"]	匹配带有一个名为attr的属性的元素， 其值开头为value子字符串。
[attr$=value]	li[class$="-box"]	匹配带有一个名为attr的属性的元素， 其值结尾为value子字符串
[attr*=value]	li[class*="box"]	匹配带有一个名为attr的属性的元素， 其值的字符串中的任何地方， 至少出现了一次value子字符串。

忽略大小写匹配选择

在大括号闭合前加上 `i` 即可表明为忽略大小写匹配

```
li[class^="a" i]
```

id选择

使用 # 选择id

```
#unique { }
```

伪类选择

这组选择器包含了伪类，用来样式化一个元素的特定状态。例如:hover伪类会在鼠标指针悬浮到一个元素上的时候选择这个元素：

```
a:hover { }
```

它还可以包含了伪元素，选择一个元素的某个部分而不是元素自己。例如，::first-line是会选择一个元素（下面的情况中是 <p> ）中的第一行，类似 包在了第一个被格式化的行外面，然后选择这个 。

```
p::first-line { }
```

特定元素的类选择

除此之外，也能够通过使用 special 选择器选择特定样式进行设置

```
li.special,
span.special {
  color: orange;
  font-weight: bold;
}
```

全局选择

使用 * 进行全局选择

```
* {margin: 0}
```

位置选择

通过对位置的差分进行选择，通过对结构的定义进行选择操作，选择标签的顺序，并且使用空格间隔
例：选择 <li> 元素内的 元素

```
li em{
  color:red;
}
```

另：使用+相连，可使用相邻选择器，选择对应元素

```
<h1>I am a level one heading</h1>

<p>This is a paragraph of text. In the text is a <span>span element</span>
and also a <a href="http://example.com">link</a>.</p>

<p>This is the second paragraph. It contains an <em>emphasized</em> element.</p>

<ul>
    <li>Item <span>one</span></li>
    <li>Item two</li>
    <li>Item <em>three</em></li>
</ul>
```

```
li em {
    color: red;   //第三个列表变为红色（带有em标签）
}

h1 + p {
    font-size: 200%;
    color: red;   //标题后的第一个段落变为红体大字
}
```

根据状态确认样式

在使用 `<a>` 时，需要针对是否访问过进行差分区别
example：

```
a:link {
  color: pink;
}

a:visited {
  color: green;
}
```

同时使用选择器和选择符

```
body h1 + p .special {
  color: yellow;
  background-color: black;
  padding: 5px;
}
```

构建CSS

使用样式表

外部样式表

需要在外部文件中创建一个以 `css` 为后缀的文件，并且在内部编辑，然后从html处引用

```
<link rel="stylecss" href="……">
```

内部样式表

在html文件中的 `<head>` 添加 `<stsyle>` 标签，并且在标签中直接添加css语句
即不另设文件，直接在html中对css进行编写

内联样式

在每个标签中单独添加 `style` 属性进行影响

在文本中使用

使用选择器

```
h1
a:link
.manythings
#onething
*
.box p
.box p:first-child
h1, h2, .intro
```

选择方式

专一性

在html样式中，如果多个同样的选择器对一个元素进行了修改，那么后置的选择器会覆盖靠前的效果
即：后置的选择器优先

但如果出现在不同的选择器之中，会更加偏向更为具体的选择器，比如在类选择器和元素选择器中，类比元素选择器更加具体，能够显示出来

属性和值

通过对可读的属性的赋值实现不同的演示效果

函数

`calc()` 函数

进行简单的计算操作
`width: calc(90%-30px)` 在括号内进行表达式的书写能够进行简单的计算

`transform` 函数

更改元素的相对位置，例如旋转、缩放、倾斜、平移

@rule

特殊规则，用于提供如何表现的指导

```
body {
  background-color: pink;
}

@media (min-width: 30em) {
  body {
    background-color: blue;
  }
}
```

速记属性

部分属性能够在一行内一次性设置多个值，便于简洁代码

```
padding: 10px 15px 15px 5px;

padding-top: 10px;
padding-right: 15px;
padding-bottom: 15px;
padding-left: 5px;
```

```
background: red url(bg-graphic.png) 10px 10px repeat-x fixed;

background-color: red;
background-image: url(bg-graphic.png);
background-position: 10px 10px;
background-repeat: repeat-x;
background-attachment: fixed;
```

注释

使用 /**/ 添加注释信息
同时也可以暂时性禁用部分代码以供测试用途

空白

实际指包括实际空格、制表符和新行在内的空白部分
在实际编辑过程中需要注意格式的问题，确保可读性与共线性

选择

见上
选择器

伪类和伪元素

伪类是选择器的一种，用于选择特定状态的元素
关键字：开头为冒号 `:pseudo-calss-name`

简单伪类

使用 `:first-child` 选择文章中的第一个子元素，以此降低维护难度： `article P:first-child {……}`
同理： `:last-child` 、 `:only-child` 、 `:invalid`

用户行为伪类

又称动态伪类，主要表现在用户与元素交互时
如 `:hover` 、 `focus`

伪元素

以双冒号的形式开头，表现形式为添加额外的html元素
`::pseudo-element-name`

`::first-line` 用于选择第一行字符，即使因设备不同带来的字符改变

结合伪类与伪元素

如果只需要将第一段的第一行加粗，则将伪类和伪元素的两个选择器同时使用即可

生成带有 `::before` 和 `::after` 的内容

有一组特殊的伪元素，用于和 `content` 属性连用，将内容通过css插入到文档中

```
.class ::before {
  content: "this is first line"
}

.class ::after {
  content: "this is last line"
}
```

补充：在部分屏幕阅读器中并不能识别出该添加部分，故并不能够用于添加语句文本，可以用作添加视觉性的小标记。

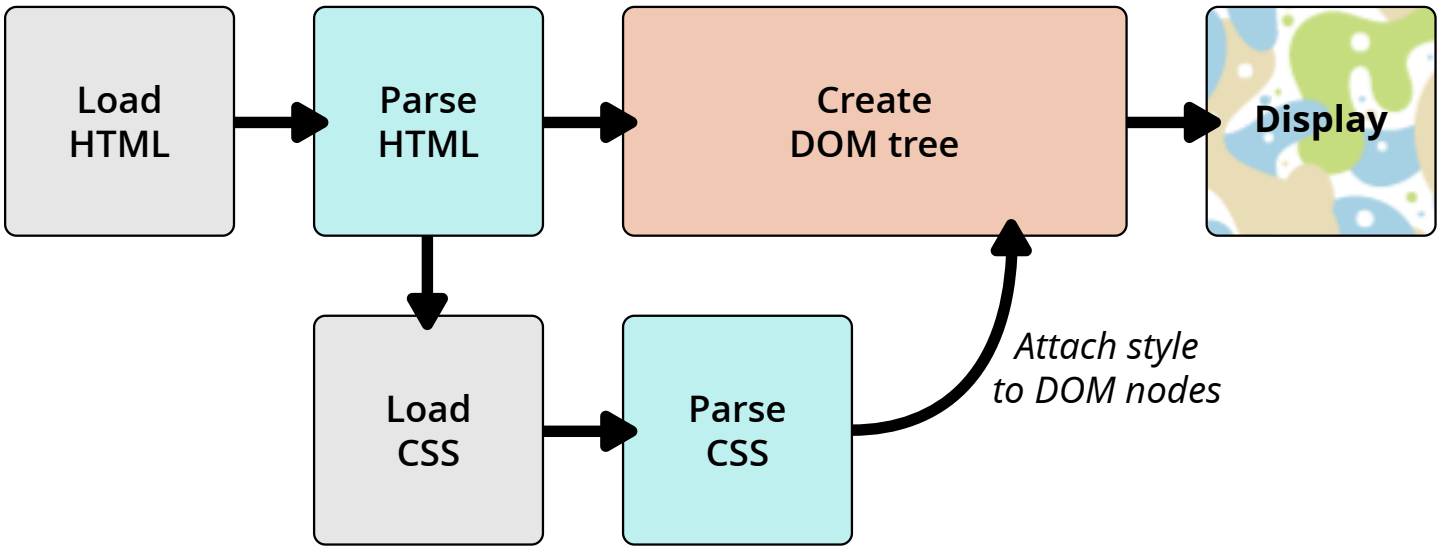
::before和::after伪元素与content属性的共同使用，在 CSS 中被叫做“生成内容”，而且你会见到这种技术被用于完成各种任务。

参考节

列出伪类和伪元素的列表，用作参考
链接

CSS运行方式

1.浏览器载入html文件
2.转换为DOM
3.拉取html相关的资源
4.拉取到css后进行解析，并分配到不同的选择器中，套用不同规则后应用到渲染树
5.开始渲染
6.网页显示



关于DOM

一个 DOM 有一个树形结构，标记语言中的每一个元素、属性以及每一段文字都对应着结构树中的一个节点（Node/DOM 或 DOM node）。节点由节点本身和其他 DOM 节点的关系定义，有些节点有父节点，有些节点有兄弟节点（同级节点）。

对于 DOM 的理解会很大程度上帮助你设计、调试和维护你的 CSS，因为 DOM 是你的 CSS 样式和文件内容的结合。当你使用浏览器 F12 调试的时候你需要操作 DOM 以查看使用了哪些规则。

解析DOM

举例说明：

```
<p>
  Let's use:
  <span>Cascading</span>
  <span>Style</span>
  <span>Sheets</span>
</p>
```

```
P
├─ "Let's use:"
├─ SPAN
|  └─ "Cascading"
├─ SPAN
|  └─ "Style"
└─ SPAN
    └─ "Sheets"
```

应用CSS到DOM

浏览器会解析 HTML 并创造一个 DOM，然后解析 CSS。可以看到唯一的选择器就是span元素选择器，浏览器处理规则会非常快！把同样的规则直接使用在三个 `<span>` 标签上，然后渲染出图像到屏幕。

无法解析的CSS

当遇到无法解析的新CSS时，浏览器会忽略该解析，然后继续解析下一个CSS

为此，可以套用多个CSS样式，在即使无法正常套用解析时，也能够使用另外的解析