HTML简单学习：

自存一些语法：

文字呈现<p>：

My cat is very grumpy

强调:

My cat is **very** grumpy.

插入图片<img>:

注意此时没有返回值</img>，这些统称为 Void elements（末尾的 / 也可以不添加）。

通往外链<a>：

```
  <p >A link to <a href="https://www.bilibili.com/" title="bilibili" target="_blank" > my favorite website.</a> </p>
```

A link to my favorite website.

布尔属性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 />

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

用一个例子来继续：

```
<!doctype html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8" />
    <title>我的测试站点</title>
  </head>
  <body>
    <p>这是我的页面</p>
  </body>
</html>
```

这是我的页面

<!DOCTYPE html>: 声明文档类型

历史遗留物，一般直接用

<meta charset="utf-8">:

这个元素代表了不能由其他 HTML 元相关元素表示的元数据，这里的 charset 属性将你的文档的字符集设置为 UTF-8，其中包括绝大多数人类书面语言的大多数字符。一般直接用。

<title></title>:

这设置了页面的标题，也就是出现在该页面加载的浏览器标签中的内容。当页面被加入书签时，页面标题也被用来描述该页面。

<body></body>:

包含了你访问页面时*所有*显示在页面上的内容，包含文本、图片、视频、游戏、可播放音频轨道等等。

等价字符引用：

原义字符	等价字符引用
<	<
>	>
"	"
'	'
&	&

注释：

```
<!-- <p>我在注释内！</p> -->
```

元数据：<meta> 元素

字符编码：

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

在搜索引擎中的 description：

```
<meta
  name="description"
  content="The MDN Web Docs site
  provides information about Open Web technologies
  including HTML, CSS, and APIs for both Web sites and
  progressive web apps." />
```

link 元素经常位于文档的头部，它有 2 个属性，rel="stylesheet" 表明这是文档的样式表，而 href 包含了样式表文件的路径：

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

<script> 元素也应当放在文档的头部，并包含 src 属性来指向需要加载的 JavaScript 文件路径，同时最好加上 defer 以告诉浏览器在解析完成 HTML 后再加载 JavaScript。

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

添加图标（记得先把图片放进同目录文件中，不一定要用.ico，有些浏览器.gif啥的也可以）：

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

在不同的地方可以用不同的图标（暂时应该用不到）：

```
<!-- 含有高分辨率 Retina 显示屏的 iPad Pro: -->
<link
  rel="apple-touch-icon"
  sizes="167x167"
  href="/apple-touch-icon-167x167.png" />
<!-- 三倍分辨率的 iPhone: -->
<link
  rel="apple-touch-icon"
  sizes="180x180"
  href="/apple-touch-icon-180x180.png" />
<!-- 没有 Retina 的 iPad、iPad mini 等: -->
<link
  rel="apple-touch-icon"
  sizes="152x152"
  href="/apple-touch-icon-152x152.png" />
<!-- 二倍分辨率的 iPhone 和其他设备: -->
<link rel="apple-touch-icon" href="/apple-touch-icon-120x120.png"
/>
<!-- 基本图标 -->
<link rel="icon" href="/favicon.ico" />
```

标题和段落

段落<p>

标题<h1><h2>...<h6>

块引用 <blockquote>：

```
<p>这是块引用：</p>
<blockquote
```

```
  cite="https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/blockquote">
  <p>
    <strong>HTML <code>&lt;blockquote&gt;</code> 元素</strong>（或<em>HTML 块级引用元素</em>）表示所附文本为扩展引用。
  </p>
</blockquote>
```

这是块引用：

> **HTML `<blockquote>` 元素**（或*HTML 块级引用元素*）表示所附文本为扩展引用。

注意： cite="https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/blockquote"仅仅用于开发者了解出处，并不做实际的链接导出。

缩略语 `<abbr>`:

没啥用，就是让你自己标出缩略语的，你可以在里面放title让自己知道原来是什么。

```
<p>
  第 33 届<abbr title="夏季奥林匹克运动会">奥运会</abbr>已于 2024 年 7 月在法国巴黎举行。
</p>
```

标记联系方式 `<address>`:

好像也没啥用，理论上用来标注地址啥的。(默认会变成斜体)

```
<address>
  <p>
    Chris Mills<br />
    Manchester<br />
    The Grim North<br />
    UK
  </p>

  <ul>
    <li>Tel: 01234 567 890</li>
    <li>Email: me@grim-north.co.uk</li>
  </ul>
</address>
```

上标<sup>和下标<sub>:

咖啡因的化学方程式是 $C_8H_{10}N_4O_2$。

如果 x^2 的值为 9，那么 x 的值必为 3 或 -3。

计算机代码:

- <code>：用于标记计算机通用代码。
- <pre>：用于保留空白字符（通常用于代码块）——如果文本中使用了缩进或多余的空白，浏览器将忽略它，你将不会在渲染的页面上看到它。但是，如果你将文本包含在 <pre></pre> 标签中，那么空白将会以与你在文本编辑器中看到的相同的方式渲染出来。
- <var>：用于标记具体变量名。
- <kbd>：用于标记输入电脑的键盘（或其他类型）输入。
- <samp>：用于标记计算机程序的输出。

标记时间和日期<time>：

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

超链接

一般就是长这样，title里面放的东西，用户鼠标移上去可以看到：

```
<p>
  我创建了一个指向
  <a
    href="https://www.mozilla.org/zh-CN/"
    title="了解 Mozilla 使命以及如何参与贡献的最佳站点。">
    Mozilla 主页</a
  >的超链接。
</p>
```

我创建了一个指向

Mozilla 主页的超链接。

标题啥的大块的也都可以变成连接。

图片也可以：

```
<a href="https://developer.mozilla.org/zh-CN/">
  <img src="mdn_logo.svg" alt="MDN Web 文档" />
</a>
```

值得一提的是，在HTML中，返回上一级目录用的是..：

```
<p>点击打开<a href="../pdfs/project-brief.pdf">项目简介</a>。</p>
```

还可以连接文档片段，只需要先设置id，然后用#访问就可以了，像这样：

```
<h2 id="Mailing_address">邮寄地址</h2>
```

```
<p>
  要提供意见和建议，请将信件邮寄至<a
href="contacts.html#Mailing_address"
    >我们的地址</a
  >。
</p>
```

如果养放下载链接，可以用download：

```
<a
  href="https://download.mozilla.org/?product=firefox-latest-
ssl&os=win64&lang=zh-CN"
  download="firefox-latest-64bit-installer.exe">
  下载最新的 Firefox 中文版 - Windows（64 位）
</a>
```

图片<img>

一般长这样：

```
<img src="images/dinosaur.jpg" alt="恐龙" />
```

alt：图片的文本描述，如果图片挂了他们就会显示出来；

width，height：图片的宽度和高度；

title： 提供图片的更多信息，这些信息会在光标移动在上面的时候显示;

<figure>,<figcaption>为图片提供一个语义容器，在说明文字和图片之间建立清晰的关联（感觉就是在图片底下放了行字），但<figure>底下不一定是字，其他多媒体形式也可以的。

```
  <figure>
  <img
    src="images/dinosaur.jpg"
    alt="The head and torso of a dinosaur skeleton;
            it has a large head with long sharp teeth"
    width="400"
    height="341"
    title="A T-Rex on display in the Manchester University Museum"
/>

  <figcaption>
    A T-Rex on display in the Manchester University Museum.
  </figcaption>
</figure>
```

音视频

<vedio>

src：和 img 里面的一样，直接放链接。

controls：用 controls 属性来让视频或音频包含浏览器自带的控制界面。

<video> 元素内的段落：后备内容，当浏览器不支持 <video> 元素的时候，就会显示这段内容，借此我们能够对旧的浏览器提供回退。

```
<video src="rabbit320.webm" controls>
  <p>
    你的浏览器不支持 HTML 视频。可点击<a href="rabbit320.mp4">此链接</a>观看。
  </p>
</video>
```

为了增加可拓展性，我们可以这样：

```
<video controls>
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
  <p>你的浏览器不支持此视频。可点击<a href="rabbit320.mp4">此链接</a>观看</p>
</video>
```

其他特性：

```
<video
  controls
  width="400"
  height="400"
  autoplay
  loop
  muted
  preload="auto"
  poster="poster.png">
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
  <p>你的浏览器不支持此视频。可点击<a href="rabbit320.mp4">此链接</a>观看</p>
</video>
```

width,height： 设定长宽，但是注意无论使用哪种方式，视频都会保持它原始的长宽比——也叫做**纵横比**。如果你设置的尺寸没有保持视频原始长宽比，那么视频边框将会拉伸，而未被视频内容填充的部分，将会显示默认的背景颜色。

autoplay： 自动播放（不建议）。

loop： 循环播放（不建议）。

muted： 播放时默认关闭声音。

poster： 这个属性指向了一个图像的 URL，这个图像会在视频播放前显示（belike 开屏广告）。

preload： 这个属性被用来缓冲较大的文件，有三个值可选：

- "none"：不缓冲文件
- "auto"：页面加载后缓存媒体文件
- "metadata"：仅缓冲文件的元数据

<audio>

和 <vedio> 没啥区别（只有一些细微差别比如边框）。

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3" />
  <source src="viper.ogg" type="audio/ogg" />
  <p>你的浏览器不支持该音频，可点击<a href="viper.mp3">此链接</a>收听。
</p>
</audio>
```

除了不支持 width,height 和 poster ， 其他组件应该可以照搬。

调试

调试用网站：https://validator.w3.org/

实战中遇到的补充（待续）

<input>

基本语法（自闭合）：

```
<input type="类型" name="字段名" value="默认值" />
```

name 主要也是方法编写者查看，与 id 不用的的是，id 全局唯一，而 name 可以方便调试而在语法相同的情况下设置相同的 name。而 class 一般用于分组的命名。

常见类型及示例：

类型（type）	功能说明	示例代码
text	单行文本输入框	<input type="text" name="username" />
password	密码输入框（隐藏字符）	<input type="password" name="pwd" />
email	只能输入邮箱地址	<input type="email" name="useremail" />
number	只能输入数字	<input type="number" min="1" max="100" />
date	日期选择器	<input type="date" />

类型（type）	功能说明	示例代码
file	上传文件	<input type="file" />
checkbox	多选框	<input type="checkbox" />
radio	单选框	<input type="radio" name="gender" value="男" />
submit	提交按钮	<input type="submit" value="提交表单" />
button	普通按钮	<input type="button" value="点击我" />

关于预设文本，有两种， placeholder 和 value ：

属性	功能	是否提交给服务器
value	输入框里的**真实内容**，用户可以修改	✅ 是
placeholder	显示在输入框里的一段**灰色提示文本**，用户一输入就消失	❌ 否

```
<!-- 有默认值 -->
<input type="text" value="默认用户名" />

<!-- 有提示但没有默认值 -->
<input type="text" placeholder="请输入用户名" />
```

required ： 布尔属性，表示该字段是必填项，用户不能留空提交表单（在form里面添加）。

<button>

`<button>` 是 HTML 中创建按钮的标签，比 `<input type="button">` 更灵活，因为它可以包含**任意 HTML 内容**（比如图标、文本、甚至图片）。

基本语法：

```
<button type="类型">按钮文字</button>
```

type 属性常见取值：

类型	功能
button	普通按钮，不提交表单（用于 JS 控制）
submit	提交表单（默认类型）
reset	重置表单

创建无序链表

基本语法：

```
<ul>
  <li>第一项</li>
  <li>第二项</li>
  <li>第三项</li>
</ul>
```

可以用 CSS 修改列表符号：

```html
<ul style="list-style-type: square;">
  <li>方块项</li>
  <li>也是方块项</li>
</ul>
```

常用的 `list-style-type` 有：

- `disc`（默认实心圆）
- `circle`（空心圆）
- `square`（方块）
- `none`（不显示符号）

* 和<form>容器*

在 HTML 中，用一个 `<div>` 元素**作为容器**，把已有的四个组件（比如按钮、输入框、标题、列表等）**包在一起**，形成一个结构上的整体。

简单来说就是把你要包在一起的东西放在`<div></div>`中间，起到一个大括号的作用（？）。

像 `<div>` 一样，`<form>` 也可以**包裹其他元素**，但它除了结构上的作用，还具备**专门处理表单提交**的功能。

一般会把`input` 和 `button` 包在一起：

```html
<form action="/submit" method="post">
  <input type="text" name="username" placeholder="输入用户名" />
  <button type="submit">提交</button>
</form>
```

输入用户名 提交

这个结构中：

- `action="/submit"` 表示点击提交按钮后，数据会发送到 `/submit`；
- `method="post"` 指的是以 POST 的方式提交数据；
- `name="username"` 是表单字段的名字，提交时会作为键。