

## HTML 教程- (HTML5 标准)

### 一、Web 浏览器

### 二、HTML 网页结构

### 三、<!DOCTYPE> 声明

### 四、HTML 属性

### 五、HTML <head> 元素

## HTML 文字处理基础

### 一、基础标题和段落：

大部分的文本结构由标题和段落组成。不管是小说、报刊、教科书还是杂志等。

#### 1、为什么我们需要结构化？

#### 2、为什么我们需要语义？

### 二、列表 Lists

#### 1、无序 Unordered：

无序列表用于标记列表项目顺序无关紧要的列表 — 让我们以早点清单为例。然后就是用 <li> 元素把每个列出的项目单独包裹起来：

#### 2、有序 Ordered

### 三、重点强调

在日常用语中，我们常常会加重某个字的读音，或者用加粗等方式突出某句话的重点。

#### 1、强调

#### 2、非常重要

#### 3、斜体字、粗体字、下划线...

## 建立超链接

### 1、使用title属性添加支持信息

## 2、块级链接

### 3、HTML 链接 - target 属性

注释：请始终将正斜杠添加到子文件夹。假如这样书写链接：  
`href="https://www.runoob.com/html"`，就会向服务器产生两次 HTTP 请求。这是因为服务器会添加正斜杠到这个地址，然后创建一个新的请求，就像这样：  
`href="https://www.runoob.com/html/"`。

## 一、统一资源定位符(URL)与路径(path)快速入门

### 常见的 URL Scheme

#### 1、文档片段(HTML 链接- id 属性)

#### 2、绝对URL和相对URL

#### 3、在下载链接时使用 download 属性

## 二、电子邮件链接

## 高阶文字排版

### 一、描述列表

### 二、引用

#### 1、块引用

#### 2、行内引用

#### 3、引文

### 三、缩略语

### 四、标记联系方式

### 五、上标和下标

### 六、展示计算机代码

### 七、标记时间和日期

### 八、总结

## 文档与网站架构

## 一、文档的基本组成部分

## 二、用于构建内容的 HTML

## 三、HTML 布局元素细节

### 1、无语义元素

### 2、换行与水平分割线

## 四、规划一个简单的网站

## HTML中的图片

### 一、怎样将一幅图片放到网页上？

### 二、通过为图片搭配说明文字的方式来解说图片

### 三、CSS 背景图片

## 视频和音频内容

### web 中的音频和视频

## 从 <object> 到 <iframe> — 其他嵌入技术

## 在页面中添加矢量图像

## 响应式图片

## HTML 表格

### 一、什么是表格？

#### 1、表格如何工作？

### 二、使用 <th> 元素添加标题

### 三、允许单元格跨越多行和列

### 四、为表格中的列提供共同的样式

## HTML表格高级特性和可访问性

### 一、使用 <caption> 为你的表格增加一个标题

### 二、<thead>, <tfoot>, 和 <tbody> 结构

### 三、嵌套表格

#### 1、使用列和行的标题

#### 2、scope 属性

#### 3、id 和标题属性（替代 scope 属性）

#### 构建 CSS 块

#### 样式化文本

#### CSS 布局

### 学习CSS 第一步

#### 什么是CSS?

#### 让我们开始CSS的学习之旅

#### 一、改变元素的默认行为list-style-type

#### 二、选择器：

##### 使用类名class

##### ID选择器 #onething

#### 三、根据元素在文档中的位置确定样式（选择器）

##### 后代选择器

##### 相邻选择符

##### 根据状态确定样式

#### 在文档中应用CSS的三种方法：

##### 一、外部样式表：

##### 二、内部样式表：

##### 三、内联样式：

#### 四、函数

##### CSS transform属性

## 五、@规则

@import

@media

CSS究竟是怎么工作的？

关于DOM

一个真实的DOM案例：

当浏览器遇到无法解析的CSS代码会发生什么

一、层叠

二、优先级

三、继承

一些不能继承的属性

控制继承

重设所有属性值all: initial/inherit/unset

四、理解层叠(选择器优先级)

有三个因素需要考虑，根据重要性排序如下，前面的更重要：

!important

选择器列表

CSS选择器的种类

一、ID选择器

三、属性选择器

1、存否和值选择器

2、子字符串匹配选择器

下个示例展示了这些选择器的用法：

3、大小写不敏感的情况下，匹配属性值

## 四、伪类与伪元素

### 用户行为伪类

备注：一些早期的伪元素曾使用单冒号的语法，所以你可能会在代码或者示例中看到。现代的浏览器为了保持后向兼容，支持早期的带有单双冒号语法的伪元素。

### 把伪类和伪元素组合起来

### 生成带有::before和::after的内容

## 五、运算符

## 六、全局选择器(通配符 \*)

### 使用全局选择器，让选择器更易读

### 选择器参考表

## 七、关系选择器

### 1、后代选择器

### 2、子代关系选择器( > )

### 3、邻接兄弟选择器 ( + )

### 4、通用兄弟 ( ~ )

### 5、使用关系选择器

## 盒模型

### 一、块级盒子 (Block box) 和 内联盒子 (Inline box)

### 二、什么是CSS 盒模型？

#### 1、盒模型的各个部分

#### 2、标准盒模型

#### 3、替代 (IE) 盒模型

#### 4、外边距

##### 4.1外边距折叠

## 5、边框

## 6、内边距

## 三、盒子模型和内联盒子

### 1、使用display: inline-block

## 背景与边框

### 一、CSS的背景样式

#### 1、背景颜色

#### 2、背景图片

##### 2.1、控制背景平铺

##### 2.2、调整背景图像的大小

##### 2.3、背景图像定位

下面是等价的位置关键字：

##### 2.4、渐变背景

##### 2.5、多个背景图像

##### 2.6、背景的可访问性考虑

##### 2.6、背景关联

### 3、背景不透明度通过opacity属性设置

## 3、边框

## 4、圆角

## 为文本添加样式（样式化文本）

## 基本文本和字体样式

### 一、CSS中的文字样式涉及什么？

#### 二、颜色

#### 三、字体种类

## 1、网页安全字体

## 2、默认字体

## 3、字体栈

## 四、字体大小

## 五、字体样式，字体粗细，文本转换和文本装饰

## 六、文字阴影

## 七、文本布局

### 1、文本对齐

### 2、行高

### 3、字母和单词间距

### 4、Font 样式:

### 4、文本布局样式:

## 八、Font 简写

## 样式列表

### 一、处理列表间距

### 二、列表特定样式

#### 1、项目符号位置

#### 2、list-style 速记

### 三、管理列表计数

#### 1、start 属性

#### 2、value 属性

该属性允许设置列表项指定数值，示例如下:



# HTML 教程- (HTML5 标准)

超文本标记语言（英语：HyperText Markup Language，简称：HTML）是一种用于创建网页的标准标记语言。

您可以使用 HTML 来建立自己的 WEB 站点，HTML 运行在浏览器上，由浏览器来解析。

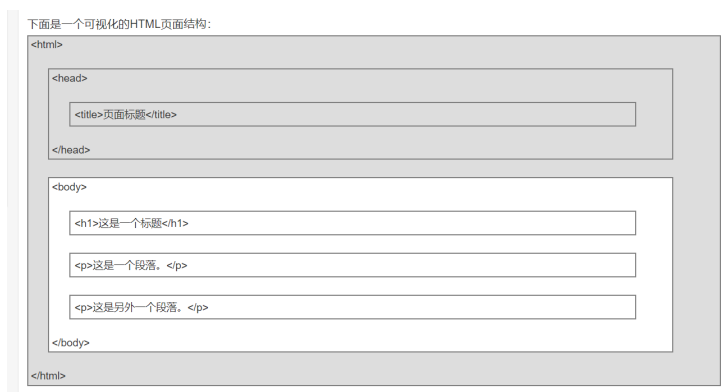
```
1  !DOCTYPE html> <!--声明为 HTML5 文档,有助于浏览器中正确显示网页。网络上有很多
不同的文件，如果能够正确声明HTML的版本，浏览器就能正确显示网页内容。-->
2  <html><!--HTML 页面的根元素-->
3  <head><!--包含了文档的元（meta）数据-->
4  <meta charset="utf-8"><!--定义网页编码格式为 utf-8-->
5  <title>菜鸟教程(runoob.com)</title><!--描述了文档的标题-->
6  <meta http-equiv="refresh" content="30"><!--每30秒钟刷新当前页面-->
7  </head>
8  <body><!--包含了可见的页面内容-->
9  <h1>我的第一个标题</h1>
10 <p>我的第一个段落。</p>
11 </body>
12 </html>
```

## 一、Web 浏览器

Web浏览器（如谷歌浏览器，Internet Explorer，Firefox，Safari）是用于读取HTML文件，并将其作为网页显示。浏览器并不是直接显示的HTML标签，但可以使用标签来决定如何展现HTML页面的内容给用户：

## 二、HTML 网页结构

下面是一个可视化的HTML页面结构：



只有 <body> 区域 (白色部分) 才会在浏览器中显示。

## 三、<!DOCTYPE> 声明

<!DOCTYPE>用来告知 Web 浏览器页面使用了哪种 HTML 版本，声明有助于浏览器中正确显示网页。网络上有很多不同的文件，如果能够正确声明HTML的版本，浏览器就能正确显示网页内容。doctype 声明是不区分大小写的，以下方式均可

```
1 <!doctype html>
2 <!DOCTYPE HTML>
3 <!DOCTYPE html>
4 <!DOCTYPE Html>
```

四、HTML 属性

属性总是以名称/值对的形式出现，比如：name="value"。

下面列出了适用于大多数 HTML 元素的属性：

属性	描述
class	为html元素定义一个或多个类名（className引入），引号里面可以填入多个class属性
id	定义元素的唯一id，引号里只能填写一个，
style	规定元素的行内样式（inline style）
title	描述了元素的额外信息 (作为工具条使用)

五、HTML <head> 元素

<head> 元素包含了所有的头部标签元素。在 <head>元素中你可以插入脚本（scripts），样式文件（CSS），及各种meta信息。

可以添加在头部区域的元素标签为：

<title>	标签定义HTML文档的标题、定义了浏览器工具栏的标题、当网页添加到收藏夹时，显示在收藏夹中。 <title>元素不仅可以显示文本， <b>也可以在左侧显示logo等图片。</b>  显示时，要将<link>标签放入<head>里 <head> <title>鸟教程(runoob.com) <i>这是一个带图片的标签</i> </title> <meta charset="utf-8"> <link rel="shortcut icon" href="images路径\图片名字.jpg" />   <!-- <b>也可以在左侧显示</b> --> </head>
<style>	定义了HTML文档的样式文件引用地址.
<base>	定义页面中所有链接默认的链接目标地址<base href="//www.runoob.com/images/" target="_blank">
<meta>	元素来描述HTML文档的描述，关键词，作者，字符集等描述了一些基本的元数据。
<link>	定义了文档与外部资源之间的关系，通常用于链接到样式表:<link rel="stylesheet" type="text/css">
<script>	用于加载脚本文件，如： JavaScript
<noscript> >	

# HTML 文字处理基础

目的:

学习如何用标记(段落、标题、列表、强调、引用)来建立基础文本页面的文本结构和文本内容。

## 一、基础标题和段落：

大部分的文本结构由**标题**和**段落**组成。不管是小说、报刊、教科书还是杂志等。

#在HTML中，每个段落是通过 `<p>` 元素标签进行定义的, 比如下面这样：

```
1 <p>我是一个段落，千真万确。</p>
```

#每个标题（Heading）是通过“标题标签”进行定义的：搜索引擎使用标题为您的网页的结构和内容编制索引。

```
1 <h1>我是文章的标题</h1>
```

这里有六个标题元素标签——`<h1>`、`<h2>`、`<h3>`、`<h4>`、`<h5>`、`<h6>`。每个元素代表文档中不同级别的内容；

`<h1>` 表示主标题（the main heading），`<h2>` 表示二级子标题（subheadings），

`<h3>` 表示三级子标题（sub-subheadings），等等。

在创建结构时，您只需要记住一些最佳实践：

- 您应该最好只对每个页面使用一次**`<h1>`** — **这是顶级标题**，所有其他标题位于层次结构中的下方。
- 请确保在层次结构中以正确的顺序使用标题。不要使用`<h3>`来表示副标题，后面跟`<h2>`来表示副副标题 - 这是没有意义的，会导致奇怪的结果。
- 在可用的六个标题级别中，每页使用不超过三个，除非您认为有必要使用更多。具有许多级别的文档（即，较深的标题层次结构）变得难以操作并且难以导航。在这种情况下，如果可能，建议将内容分散在多个页面上。

## 1、为什么我们需要结构化？

- 用户在阅读网页时，经常只是阅读开头的标题（我们通常在一个网页上会花费很少的时间 spend a very short time on a web page）。如果用户不能在几秒内看到一些有用的内容，他们很可能会感到沮丧并离开。
- 对您的网页建立索引的搜索引擎将标题的内容，视为影响网页搜索排名的重要关键字。没有标题，您的网页在SEO（搜索引擎优化）方面效果不佳。
- 严重视力障碍者通常不会阅读网页；他们用听力来代替。完成这项工作的软件叫做屏幕阅读器（screen reader）。该软件提供了快速访问给定文本内容的方法。

- 使用CSS样式化内容，或者使用JavaScript做一些有趣的事情，你需要包含相关内容的元素，所以CSS / JavaScript可以有效地定位它。

因此，需要给我们的内容结构标记。

```
1 <h1>静夜思</h1>
2 <p>床前明月光 疑是地上霜</p>
3 <p>举头望明月 低头思故乡</p>
```

## 2、为什么我们需要语义？

我们需要确保使用了正确的元素来给予内容正确的意思、作用以及外形。

```
1 <h1>这是一个顶级标题</h1>
2 <!-- 它给出了包裹在您的页面上用来表示顶级标题的角色（或意义）的文本。
3 它的语义值将以多种方式被使用，比如通过搜索引擎和屏幕阅读器-->
```

在另一方面，你可以让任一元素看起来像一个顶级标题，如下：

```
1 <span style="font-size: 32px; margin: 21px 0;">这是顶级标题吗？</span> //这是没有意义的，因为没有语义。
```

**<span> 元素**，它没有语义。当您想要对它用CSS（或者JS）时，您可以用**它包裹内容**。

## 二、列表 Lists

### 1、无序 Unordered:

无序列表用于标记列表项目顺序无关紧要的列表 — 让我们以早点清单为例。然后就是用 `<li>` 元素把每个列出的项目单独包裹起来：

```
1 <ul>
2   <li>豆浆</li>
3   <li>油条</li>
4 </ul>
```

### 2、有序 Ordered

这个标记的结构和无序列表一样，除了需要用`<ol>` 元素将所有项目包裹，而不是`<ul>`，有序列表需要按照项目的顺序列出来——让我们以一组方向为例：

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

### 三、重点强调

在日常用语中，我们常常会加重某个字的读音，或者用加粗等方式突出某句话的重点。

#### 1、强调

在HTML中我们用<em> (emphasis) 元素来标记这样的情况。这样做既可以让文档读起来更有趣，也可以被屏幕阅读器识别出来，并以不同的语调发出。

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

#### 2、非常重要

<strong> (strong importance) 元素。让文档更加地有用，也可以被屏幕阅读器识别出来，并以不同的语调发出。浏览器默认风格为粗体，但你不应该纯粹使用这个标签来获得粗体风格，为了获得粗体风格，你应该使用<span>元素和一些CSS，或者是 <b> 元素。

```
1 <p>This liquid is <strong>highly toxic</strong>.</p>
2 <p>I am counting on you. <strong>Do not</strong> be late!</p>
```

如有需要你可以将strong元素和em元素嵌套在其他的标签中：

```
1 <p>This liquid is <strong>highly toxic</strong> -if you drink it,
  <strong>you may <em>die</em></strong>.</p>
```

#### 3、斜体字、粗体字、下划线...

- <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.....

---

## 建立超链接

#超链接非常重要 ——它们使互联网成为一个互联的网络。

标目：	学习如何实现一个有效地把多个文件链接在一起的文本链接。
-----	-----------------------------

#### 1、使用title属性添加支持信息

当鼠标指针悬停在链接上时，标题将作为提示信息出现）：“我创建了一个指向Mozilla 主页 的超链接。”

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

## 2、块级链接

你可以将一些内容转换为链接，甚至是块级元素。将一个图像转换为链接，你只需把图像元素放到<a></a>标签中间。

```
1 <a href="https://www.mozilla.org/zh-CN/">
2   
3 </a>
```

## 3、HTML 链接 - target 属性

使用 target 属性，你可以定义被链接的文档在何处显示。

```
1 <a href="https://www.runoob.com/" target="_blank">访问菜鸟教程!</a>
```

## 基本的注意事项 - 有用的提示

**注释：** 请始终将**正斜杠**添加到**子文件夹**。假如这样书写链接：

href="https://www.runoob.com/html"，就会向服务器产生两次 HTTP 请求。这是因为服务器会添加

正斜杠到这个地址，然后创建一个新的请求，就像这样：href="https://www.runoob.com/html/".

## 属性值

值	描述
_blank	在新窗口中打开被链接文档。
_self	默认。在相同的框架中打开被链接文档。
_parent	在父框架集中打开被链接文档。
_top	在整个窗口中打开被链接文档。
framename	在指定的框架中打开被链接文档。

## 一、统一资源定位符(URL)与路径(path)快速入门

统一资源定位符（英文：Uniform Resource Locator，简写：URL）是一个定义了在网络上的位置的一个文本字符串。用于定位万维网上的文档 <https://www.mozilla.org/zh-CN/>.

URL使用路径查找文件。路径指定文件系统中您感兴趣的文件所在的位置

```
1 scheme://host.domain:port/path/filename
```

说明:

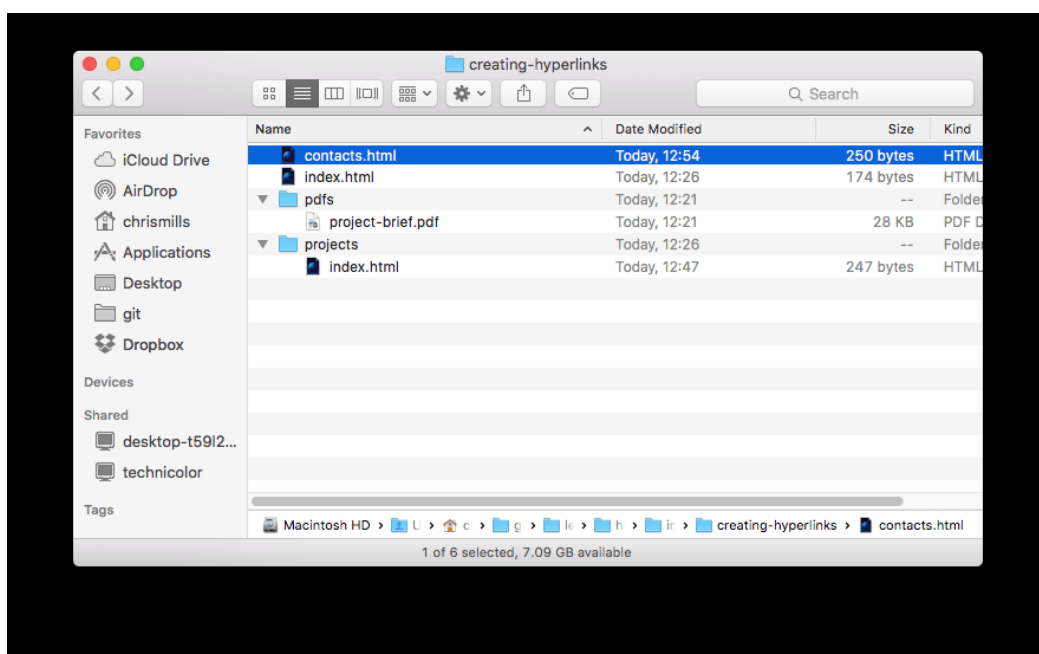
- scheme - 定义因特网服务的类型。最常见的类型是 http
- host - 定义域主机（http 的默认主机是 www）
- domain - 定义因特网域名，比如 runoob.com
- :port - 定义主机上的端口号（http 的默认端口号是 80）

- path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。
- filename - 定义文档/资源的名称

## 常见的 URL Scheme

Scheme	访问	用于...
http	超文本传输协议	以 http:// 开头的普
https	安全超文本传输协议	安全网页，加密所有
ftp	文件传输协议	用于将文件下载或上
file		您计算机上的文件。

此目录结构的根目录称为 creating-hyperlinks。当在网站上工作时，你会有一个包含整个网站的目录。在根目录，我们有一个 index.html 和一个 contacts.html 文件。在真实的网站上，**index.html 将会成为我们的主页或登录页面。**



我们的根目录还有两个目录—— pdfs 和 projects，它们分别包含一个 PDF (project-brief.pdf) 文件和一个 index.html 文件。请注意你可以有两个 index.html 文件，前提是他们不同的目录下，许多网站就是如此。第二个 index.html 或许是项目相关信息的主登录界面。

- **指向当前目录：**如果 index.html（目录顶层的 index.html）想要包含一个超链接指向 contacts.html

<p>要联系某位工作人员，请访问我们的 <a href="contacts.html">联系人页面</a>。</p>

- **指向子目录：**如果 index.html（目录顶层 index.html）想要包含一个超链接指向 projects/index.html，您要使用的 URL 是 projects/index.html：



<p>请访问 <a href="projects/index.html">项目页面</a>。</p>

- **指向上级目录：** 如果你想在projects/index.html中包含一个指向pdfs/project-brief.pdf的超链接，你必须先返回上级目录，然后再回到pdf目录。“返回上一个目录级”使用两个英文点号表示 — .. — 所以你应该使用的URL是 ../pdfs/project-brief.pdf:

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

## 1、文档片段(HTML 链接- id 属性)

超链接除了可以链接到文档外，也可以链接到HTML文档的特定部分（被称为**文档片段**）。

1.首先给要链接到的元素分配一个id属性。例如，如果你想链接到一个特定的标题，可以这样做：

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

2.然后链接到那个特定的id，您可以在URL的结尾使用一个**哈希符号 #** 号指向它，例如：

```
1 <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>
```

你甚至可以在同一份文档下，通过链接文档片段，来链接到同一份文档的另一部分：

```
1 <p>本页面底部可以找到 <a href="#Mailing_address">公司邮寄地址</a>。</p>
```

## 2、绝对URL和相对URL

网络上两个术语，**绝对URL**和**相对URL**（或者称为，**绝对链接**和**相对链接**）：

**绝对URL：** 指向由其在Web上的绝对位置定义的位置。包括 [protocol](#)（协议）和 [domain name](#)（域名）。像下面的例子，如果index.html页面上传到projects这一个目录。并且projects目录位于web服务网站的根目录，web网站的域名为http://www.example.com，可以通过http://www.example.com/**projects/index.html**访问（或者通过http://www.example.com/projects/来访问，因为在没有指定特定的URL的情况下，大多数web服务会默认访问加载index.html这类页面）

**不管绝对URL在哪里使用，它总是指向确定的相同位置。**

**相对URL：** 指向与您链接的文件相关的位置。就像**终点**总是相对于**起点**。例如，从示例文件链接 http://www.example.com/ /index.html 转到相同目录下的一个PDF文件，URL就是文件名URL——例如project-brief.pdf——没有其他的信息要求。如果PDF文件能够在projects的子目录pdfs中访问到，相对路径就是pdfs/project-brief.pdf（对应的绝对URL是http://www.example.com/projects/pdfs/project-brief.pdf）



一个相对URL将指向不同的位置，这取决于它所在的文件所在的位置——例如，把index.html文件从projects目录移动到Web站点的根目录（最高级别，而不是任何目录中），里面的pdfs/project-brief.pdf相对URL将会指向<http://www.example.com/pdfs/project-brief.pdf>，而不是<http://www.example.com/projects/pdfs/project-brief.pdf>

### 3、在下载链接时使用 download 属性

当您链接到要下载的资源而不是在浏览器中打开时，您可以使用 **download 属性**来提供一个默认的保存文件名（译注：此属性仅适用于同源URL）。下面是一个下载链接到Firefox 的 Windows最新版本的示例：

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

## 二、电子邮件链接

使用<a>元素和mailto:

```
1 <a href="mailto:">向 xxx 发邮件</a>
2 <a href="mailto:nowhere@mozilla.org">向 nowhere 发邮件</a>
3 <a href="mailto:nowhere@mozilla.org,nobody@mozilla.org">向 多个人 发邮件</a>
4 <a href="mailto:nowhere@mozilla.org?cc=nobody@mozilla.org">抄送邮件</a>
5 <a href="mailto:nowhere@mozilla.org?cc=nobody@mozilla.org&subject=This%20is%20the%20subject">抄送、密送、主题</a>
```

**注意：**单词之间的空格使用 **%20** 代替，以确保浏览器可以正常显示文本。

---

## 高阶文字排版

预备知识：熟悉 HTML 基础（包含在 [开始学习 HTML](#) 中）、HTML 文本格式（包含在 [HTML 文字处理初步](#) 中）。

目标：学习一些不常见的 HTML 元素标记来使用高级语义功能。

### 一、描述列表

三种类型的列表——**描述列表** (description list)。这种列表的目的是标记一组项目及其相关描述，例如术语和定义，或者是问题和答案等。

描述列表使用与其他列表类型不同的闭合标签——**<dl>**；每一项都用 **<dt>** (description term) 元素闭合。每个描述都用 **<dd>** (description description) 元素闭

合。请注意：一个术语 `<dt>` 可以同时有多个描述 `<dd>`

```
1 <dl>
2   <dt>培根</dt>
3   <dd>整个世界的粘合剂。</dd>
4   <dt>鸡蛋</dt>
5   <dd>一块蛋糕的粘合剂。</dd>
6   <dt>咖啡</dt>
7   <dd>一种浅棕色的饮料。</dd>
8   <dd>可以在清晨带来活力。</dd>
9 </dl>
```

浏览器的默认样式会在**描述列表的描述部分** (description description) 和**描述术语** (description terms) 之间产生缩进。

## 二、引用

HTML也有用于标记引用的特性，至于使用哪个元素标记，取决于你引用的是一块还是一行。

### 1、块引用

一个块级内容（一个段落、多个段落、一个列表等）从其他地方被引用，你应该把它用 `<blockquote>` 元素包裹起来表示，并且在 `cite` 属性里用URL来指向引用的资源。浏览器在渲染块引用时默认会增加缩进，作为引用的一个指示符；如下面的例子就是引用的MDN的`<blockquote>`元素页面：

```
1 <blockquote cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">
2   <p>
3     The <strong>HTML <code><blockquote></code> Element</strong> (or <em>HTML
4     Quotation Element</em>) indicates that the enclosed text is an extended
5     quotation.
6   </p>
7 </blockquote>
```

### 2、行内引用

行内元素用同样的方式工作，除了使用`<q>`元素。下面的标记包含了从MDN`<q>`页面的引用：

```
1 <p>The quote element –
2   <code><q></code> – is <q cite="https://developer.mozilla.org/en-US/docs/
3   Web/HTML/Element/q">intended
4   for short quotations that don't require paragraph breaks.</q>
5 </p>
```

### 3、引文

`cite` 属性内容不会被浏览器显示、也不会被屏幕阅读器阅读，需使用 JavaScript 或 CSS，浏览器才会显示 `cite` 的内容。如果你想要确保引用的来源在页面上是可显示的，更好的方法是为 `<cite>` 元素附上链接： `<cite>` 标签通常表示它所包含的文本对某个参考文献的引用，比如书籍或者杂志的标题。

```
1 <p>根据MDN blockquote页面 <a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">
2 <cite>MDN块引用页面</cite></a>:
3 </p>
4
5 <blockquote cite="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/blockquote">
6 <p>The <strong>HTML <code><blockquote></code> Element</strong> (or <em>H
7 TML Block
8 Quotation Element</em>) 表示随附的文本是扩展引号。</p>
9
10 <p>The quote element – <code><q></code> – is <q cite="https://developer.
11 mozilla.org/en-US/docs/Web/HTML/Element/q">intended
12 for short quotations that don't require paragraph breaks.</q> --
13 <a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/q">
14 <cite>MDN q page</cite>
15 </a>.
16 </p>
```

### 三、缩略语

另一个你在web上看到的相当常见的元素是`<abbr>`——它常被用来包裹一个缩略语或缩写，并且提供缩写的解释（包含在`title`属性中）。

```
1 <p>我们使用<abbr title="超文本标记语言（Hypertext Markup Language）">HTML</a
2 bbr>来组织网页文档。</p>
3 <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
4 在法国巴黎举行。</p>
5 <p><abbr title="美国国家航空航天局（National Aeronautics and Space Administ
6 ration）">NASA</abbr> 做了一些动人心弦的事情。</p>
```

### 四、标记联系方式

HTML有个用于标记联系方式的元素——`<address>`

**定义和用法：**

`<address>` 标签定义文档或文章的作者/拥有者的联系信息。

如果 `<address>` 元素位于 `<body>` 元素内，则它表示文档联系信息。

如果 `<address>` 元素位于 `<article>` 元素内，则它表示文章的联系信息。

<address> 元素中的文本通常呈现为斜体。大多数浏览器会在 address 元素前后添加折行。

```
1 <address>克里斯·米尔斯（Chris Mills），曼彻斯特，严峻的北方，英国</address>
```

但要记住的一点是，<address>元素是为了标记编写HTML文档的人的联系方式，而不是任何其他的内容。因此，如果这是Chris写的文档，上面的内容将会很好。

提示和注释

**提示：** <address> 标签不应该用于描述通讯地址，除非它是联系信息的一部分。

**提示：** <address> 元素通常连同其他信息被包含在 <footer> 元素中。

## 五、上标和下标

化学方程式、和数学方程式时会偶尔使用上标和下标。 <sup> 和<sub>元素可以解决这样的问题。例如：

```
1 <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>
2 <p>如果 x<sup>2</sup> 的值为 9，那么 x 的值必为 3 或 -3。</p>
```

## 六、展示计算机代码

有大量的HTML元素可以来标记计算机代码：

<code>&lt;code&gt;</code>	用于标记计算机通用代码。
<code>&lt;pre&gt;</code>	用于保留空白字符（通常用于代码块）——如果浏览器将忽略它，您将不会在呈现的页面上看到它 <code>&lt;/pre&gt;</code> 标签中，那么空白将会以与你在文本编
<code>&lt;var&gt;</code>	用于标记具体变量名。
<code>&lt;kbd&gt;</code>	用于标记输入电脑的键盘（或其他类型）输入。
<code>&lt;samp&gt;</code>	用于标记计算机程序的输出。

## 七、标记时间和日期

HTML 还支持将时间和日期标记为可供机器识别的格式的 `<time>` 元素。例如：

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

为什么需要这样做？因为世界上有许多种书写日期的格式，上边的日期可能被写成：

<ul style="list-style-type: none"><li>• 20 January 2016</li><li>• 20th January 2016</li><li>• Jan 20 2016</li><li>• 20/06/16</li><li>• 06/20/16</li><li>• The 20th of next month</li><li>• 20e Janvier 2016</li></ul>	但是这些不同的格式不容易被电脑识别 — 假如你想自动抓取上所有事件的日期并将它们插入到日历中， <code>&lt;time&gt;</code> 元素允许上清晰的、可被机器识别的时间/日期来实现这种需求。
---	---

- 2016年1月20日
- And so on

上述基本的例子仅提供了一种简单的可被机器识别的日期格式，这里还有许多其他支持的格式，例如：

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

## 八、总结

到这里你就完成了 HTML 语义文本元素的学习。但要记住，你在本课程中学到的并不是 HTML 文本元素的详细列表 — 我们想要尽量覆盖主要的、通用的、常见的，或者至少是有趣的部分。如果你想找到更多的 HTML 元素，可以看一看我们的[HTML 元素参考](#)

（从 [内联文本语义](#) 部分开始会是一个好的选择）。在下一篇文章中我们将会学习用来组织 HTML 文档不同部分的 HTML 元素。

## 文档与网站架构

[HTML](#) 不仅能够定义网页的单独部分（例如“段落”或“图片”），还可以使用块级元素（例如“标题栏”、“导航菜单”、“主内容列”）来定义网站中的复合区域。本文将探讨如何规划基本的网站结构，并根据规划的结构来编写 HTML。

预备知识：	阅读 <a href="#">开始学习 HTML</a> 、 <a href="#">HTML 文字处理初步</a> 、 <a href="#">创建超链接</a> ，掌握相关基础知识。
学习目标：	会用语义标签来构建文档，会搭建简单的网站结构

# 一、文档的基本组成部分

页眉	通常横跨于整个页面顶部有一个大标题 和/或 一个标志。这是网站的主要一
导航栏	指向网站各个主要区段的超链接。通常用 <b>菜单按钮</b> 、 <b>链接</b> 或 <b>标签页</b> 表示。类别则会让用户感到疑惑，甚至无所适从。许多 web 设计人员认为导航栏是标题人认为，两者独立可以提供更好的 <a href="#">无障碍访问特性</a> ，因为屏幕阅读器可以更
主内容	中心的大部分区域是当前网页大多数的独有内容，例如视频、文章、地图、表
侧边栏	一些外围信息、链接、引用、广告等。通常与主内容相关（例如一个新闻页面可能存在其他的重复元素，如辅助导航系统。
页脚	横跨页面底部的狭长区域。和标题一样，页脚是放置公共信息（比如版权声明容。还可以通过提供快速访问链接来进行 <a href="#">SEO</a> 。

一个“典型的网站”可能会这样布局：



# 二、用于构建内容的 HTML

为了实现语义化标记，HTML 提供了明确这些区段的专用标签，例如：

<code>&lt;header&gt;</code>	页眉
<code>&lt;nav&gt;</code>	导航栏



<code>&lt;main&gt;</code>	主内容。主内容中还可以有各种子内容区段，可用 <code>&lt;article&gt;</code> 、 <code>&lt;section&gt;</code> 和 <code>&lt;div&gt;</code> 等元素表示。
<code>&lt;aside&gt;</code>	侧边栏，经常嵌套在 <code>&lt;main&gt;</code> 中。
<code>&lt;footer&gt;</code>	页脚

### 三、HTML 布局元素细节

- `<main>` 存放每个页面独有的内容。每个页面上只能用一次 `<main>`，且直接位于 `<body>` 中。最好不要把它嵌套进其它元素。
- `<article>` 包围的内容即一篇文章，与页面其它部分无关（比如一篇博文）。
- `<section>` 与 `<article>` 类似，但 `<section>` 更适用于组织页面使其按功能（比如迷你地图、一组文章标题和摘要）分块。一般的最佳用法是：以 **标题** 作为开头；也可以把一篇 `<article>` 分成若干部分并分别置于不同的 `<section>` 中，也可以把一个区段 `<section>` 分成若干部分并分别置于不同的 `<article>` 中，取决于上下文。
- `<aside>` 包含一些间接信息（术语条目、作者简介、相关链接，等等）。
- `<header>` 是简介形式的内容。如果它是 `<body>` 的子元素，那么就是网站的全局页眉。如果它是 `<article>` 或 `<section>` 的子元素，那么它是这些部分特有的页眉（此 `<header>` 非彼 **标题**）。
- `<nav>` 包含页面主导航功能。其中不应包含二级链接等内容。
- `<footer>` 包含了页面的页脚部分。

#### 1、无语义元素

对于一些要组织的项目或要包装的内容，现有的语义元素均不能很好对应。你可能只想将一组元素作为一个单独的实体来修饰来响应单一的用 CSS 或 JavaScript。为了应对这种情况，HTML 提供了 `<div>` 和 `<span>` 元素。应配合使用 `class` 属性提供一些标签，使这些元素能易于查询。

<code>&lt;span&gt;</code>	是一个内联的 (inline) 无语义元素，可用作文本的容器。最好只用于无法找到特定的含义时。如下代码：
---------------------------	--

```
1 <p>国王喝得酩酊大醉，在凌晨 1 点时才回到自己的房间，踉跄地走过门口。<span class="editor-note">[编辑批注：此刻舞台灯光应变暗]</span>.</p>
```

<code>&lt;div&gt;</code>	是一个块级(block-level)无语义元素，它可用于组合其他 HTML 元素的容器。应加特定的意义时。例如，一个电子商务网站页面上有一个一直显示的购物车组件
--------------------------	--

`<div>` 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式

这里不应使用 `<aside>`，因为它和主内容并没有必要的联系（你想在任何地方都能看到它）。甚至不能用 `<section>`，因为它也不是页面上主内容的一部分。所以在这种情况下就应使用 `<div>`，为满足无障碍使用特征，还应为购物车的标题设置一个可读标签。

```
1 div class="shopping-cart">
2   <h2>购物车</h2>
3   <ul>
4     <li>
5       <p><a href=""><strong>银耳环</strong></a>: $99.95.</p>
6       
7     </li>
8     <li>
9       ...
10    </li>
11  </ul>
12  <p>售价: $237.89</p>
13 </div>
```

**警告：**`<div>` 非常便利但容易被滥用。由于它们没有语义值，会使 HTML 代码变得混乱。要小心使用，只有在没有更好的语义方案时才选择它，而且要尽可能少用，否则文档的升级和维护工作会非常困难。

## 2、换行与水平分割线

<code>&lt;br&gt;</code>	在段落中进行换行，是唯一能够生成多个短行结构（例如邮寄地址或诗歌）的元素。
<code>&lt;hr&gt;</code>	元素在文档中生成一条水平分割线，表示文本中主题的变化（例如话题或场景的改变）。

```
1 <p>原来这唐僧是个慈悯的圣僧。他见行者哀告，却也回心转意道：“既如此说，且饶你这一次。再休无礼。如若仍前作恶，这咒语颠倒就念二十遍！”行者道：“三十遍也由你，只是我不打人了。”却才伏侍唐僧上马，又将摘来桃子奉上。唐僧在马上也吃了几个，权且充饥。</p>
2 <hr>
3 <p>却说那妖精，脱命升空。原来行者那一棒不曾打杀妖精，妖精出神去了。他在那云端里，咬牙切齿，暗恨行者道：“几年只闻得讲他手段，今日果然话不虚传。那唐僧已此不认得我，将要吃饭。若低头闻一闻儿，我就一把捞住，却不是我的人。不期被他走来，弄破我这勾当，又几乎被他打了一棒。若饶了这个和尚，诚然是劳而无功也。我还下去戏他一戏。”</p>
```

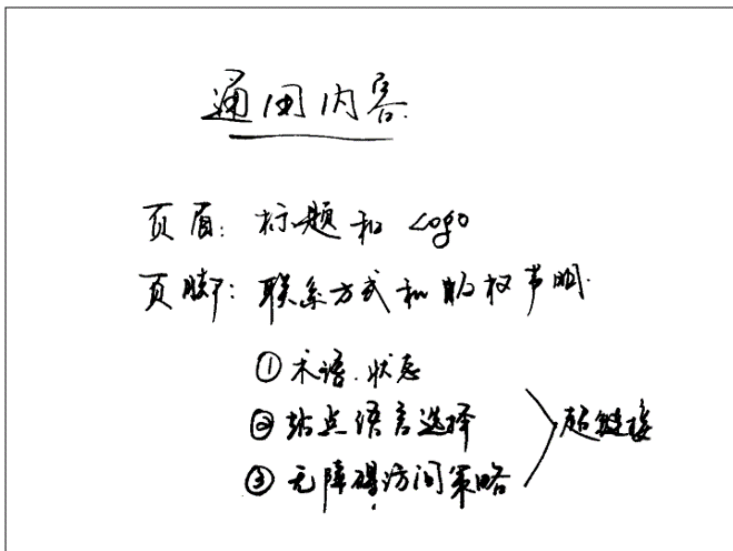
## 四、规划一个简单的网站

在完成页面内容的规划后，一般应按部就班地规划整个网站的内容，要可能带给用户最好的体验，需要哪些页面、如何排列组合这些页面、如何互相链接等问题不可忽略。这些工作称

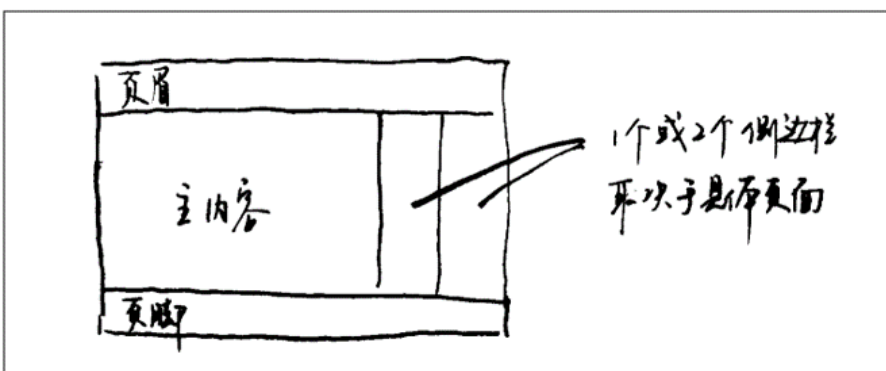


为[信息架构](#)。在大型网站中，大多数规划工作都可以归结于此，而对于一个只有几个页面的简单网站，规划设计过程可以更简单，更有趣！

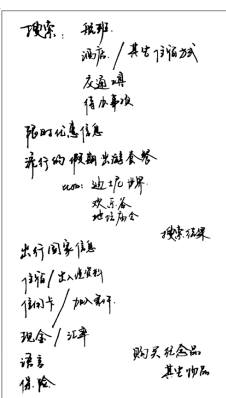
1. 时刻记住，大多数（不是全部）页面会使用一些相同的元素，例如导航菜单以及页脚内容。若网站为商业站点，不妨在所有页面的页脚都加上联系方式。请记录这些对所有页面都通用的内容：



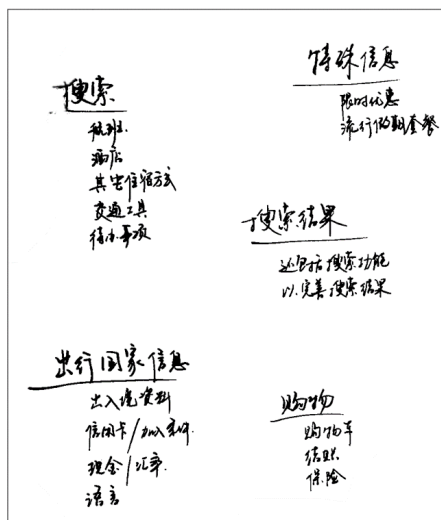
2、接下来，可为页面结构绘制草图（这里与前文那个站点页面的截图类似）。记录每一块的作用：



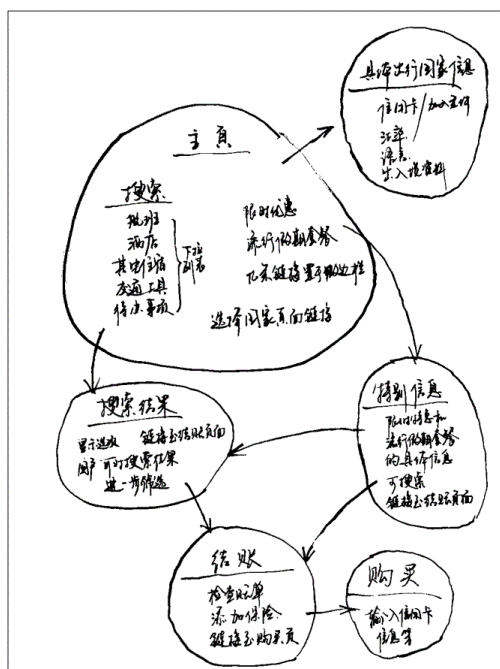
3、下面，对于期望添加进站点的所有其它（通用内容以外的）内容展开头脑风暴，直接罗列出来。



4、下一步，试着对这些内容进行分组，这样可以让你了解哪些内容可以放在同一个页面。这种做法和 **卡片分类法** 非常相似。



5、接下来，试着绘制一个站点地图的草图，使用一个气泡代表网站的一个页面，并绘制连线来表示页面间的一般工作流。主页面一般置于中心，且链接到其他大多数页面；小型网站的大多数页面都可以从主页的导航栏中链接跳转。也可记录下内容的显示方式。



## HTML中的图片

### 一、怎样将一幅图片放到网页上？

`<img>`空元素，它不需要包含文本内容或闭合标签，最少只需要一个 `src`（一般读作其全称 source）。

#### 必需的属性

属性	值	描述
<code>alt</code>	<code>text</code>	规定图像的替代文本。 #它可以派上用场的原因有：1、用户有视力障碍，通过屏幕阅读器来浏览网页、2型时、3、文字描述来给搜索引擎使用，例如搜索引擎可能会将图片的文字描述和

		<p>户关闭的图片显示以减少数据的传输，这在手机上是十分普遍的。</p> <p>#到底应该在 alt 里写点什么呢？这首先取决于为什么这张图片会在这儿，换句话说出来，会少了什么：</p> <ul style="list-style-type: none"> <li>● <b>装饰</b>：如果图片只是用于装饰，而不是内容的一部分，可以写一个空。阅读器不会浪费时间对用户读出不是核心需要的内容。实际上装饰性图文件里，<a href="#">CSS background images</a>才应该用于插入装饰图片，但如果这免，alt=""会是最佳处理方案。</li> <li>● <b>链接</b>：如果你把图片嵌套在&lt;a&gt;标签里，来把图片变成链接，那你还以<a href="#">文本</a>。在这种情况下，你可以写在同一个&lt;a&gt;元素里，或者写在图片的</li> </ul>
src	URL	规定显示图像的 URL。可以是 <b>相对路径</b> 或 <b>绝对URL</b> ，就像 <a href="#">&lt;a&gt;</a> 元素的 href 属性荐，这样做只会使浏览器做更多的工作，例如重新通过 DNS 再去寻找。把图片和 HTML 放在同一个服务器上。
title	text	鼠标滑过时显示的文字提示，用户体验上很重要。当然不必要所有的img方说logo这样比较重要或者说用户会体验到的图片内容建议一定要加此
height、width	像素、百分比	<p>你不应该使用HTML属性来改变图片的大小。如果你把尺寸设定的太大，最终图片小，会在下载远远大于你需要的图片时浪费带宽。如果没有保持正确的宽高比，图把图片放到你的网站页面之前，你应该使用图形编辑器使图片的尺寸正确。</p> <p><b>注意</b>：如果你需要改变图片的尺寸，你应该使用<a href="#">CSS</a>而不是HTML。</p> <p><b>提示</b>：指定图像的高度和宽度是一个很好的习惯。如果图像指定了高度宽保留指定的尺寸。如果没有指定图片的大小，加载页面时有可能会破坏局。</p>
border	xp	设置为0就无边框了

一般建议图片存储在和 HTML 页面同路径的 images 文件夹下（这也是Google推荐的做法，利于[SEO/索引](#)），如下形式：

```
1 
```

**注意**：搜索引擎也读取图像的文件名并把它们计入SEO。因此你应该给你的图片取一个描述性的文件名：`dinosaur.jpg` 比 `img835.png` 要好。

**注意**：像[<img>](#)和[<video>](#)这样的元素有时被称之为**替换元素**，因为这样的元素的内容和尺寸由外部资源（像是一个图片或视频文件）所定义，而不是元素自身。

标签	描述
<a href="#">&lt;img&gt;</a>	定义图像
<a href="#">&lt;map&gt;</a>	定义图像地图
<a href="#">&lt;area&gt;</a>	定义图像地图中的可点击区域

## 二、通过为图片搭配说明文字的方式来解说图片

比如当你有 很多N 张图片和其搭配的 N 段说明文字，那么一段说明文字是和哪张图片有关联的呢？

HTML5 的 `<figure>` 和 `<figcaption>` 元素，为图片提供一个语义容器，在标题和图片之间建立清晰的关联

```
1 <figure>
2   
3   <figcaption>曼彻斯特大学博物馆展出的一只霸王龙的化石</figcaption>
4 </figure>
5 <!--这个 <figcaption> 元素 告诉浏览器和其他辅助的技术工具这段说明文字描述了 <figure> 元素的内容.-->
```

注意 `<figure>` 里不一定要是一张图片，只要是一个这样的独立内容单元：

- 用简洁、易懂的方式表达意图。
- 可以置于页面线性流的某处。
- 为主要内容提供重要的补充说明。

可以是几张图片、一段代码、音视频、方程、表格或别的。

### 三、CSS 背景图片

使用CSS 把图片嵌入网站中（JavaScript也行，不过那是另外一个故事了），这个 CSS 属性 `background-image` 和另其他 `background-*` 属性是用来放置背景图片的。比如，为页面中的所有段落设置一个背景图片，你可以这样做：

```
1 p{
2   background-image:url(images/dinosaure.jpg);/*如果图像对您的内容里有意义，则应使用HTML图像(有语义)。 如果图像纯粹是装饰，则应使用CSS背景图片(无语义)。*/
3 }
```

## 视频和音频内容

预备知识：	基础计算机能力，基础的软件安装，基础的文件处知识，基础的 HTML 知识 (阅读 <a href="#">开始学习 HTML</a> ) 及 <a href="#">HTML 中的图片</a> 。
学习目标：	学习如何在一个网页中嵌入音频和视频，以及如何视频添加字幕。

### web 中的音频和视频

# 从 <object> 到 <iframe> — 其他嵌入技术

## 在页面中添加矢量图像

## 响应式图片

# HTML 表格

标签	描述
<a href="#">&lt;table&gt;</a>	定义表格
<a href="#">&lt;th&gt;</a>	定义表格的表头
<a href="#">&lt;tr&gt;</a>	定义表格的行
<a href="#">&lt;td&gt;</a>	定义表格单元
<a href="#">&lt;caption&gt;</a>	定义表格标题
<a href="#">&lt;colgroup&gt;</a>	定义表格列的组
<a href="#">&lt;col&gt;</a>	定义用于表格列的属性
<a href="#">&lt;thead&gt;</a>	定义表格的页眉
<a href="#">&lt;tbody&gt;</a>	定义表格的主体
<a href="#">&lt;tfoot&gt;</a>	定义表格的页脚

HTML让在web上显示表格数据变的很容易，例如

- 你的学校的教学计划
- 你当地的游泳馆的时刻表
- 或者是关于你最爱的恐龙或足球队的统计数据。

这个模块会教给你所有你需要知道的关于用HTML构建表格数据的知识。本文介绍一些基本的内容，如行和**单元格**、**标题**、使单元格**跨越**多个**列**和**行**，以及如何将列中的所

有单元组合在一起进行样式化。

# 一、什么是表格？

表格是由行和列组成的结构化数据集(表格数据)，它能够使你简捷迅速地查找某个表示不同类型数据之间的某种关系的值。

Person	Age
Chris	38
Dennis	45
Sarah	29
Karen	47

## 1、表格如何工作？

表格的一个特点就是严格. 通过在**行和列的标题之间**进行视觉关联的方法，就可以让信息能够很简单地被解读出来。

Subject	Object		
单数	第一人称		I
	第二人称		you
	第三人称	♂	he
		♀	she
		o	it
复数	第一人称		we
	第二人称		you
	第三人称		they

即使是盲人也可以解析 HTML 表格中的数据，一个成功的 HTML 表格应该做到无论用户是**视力正常**还是**视力受损**，都能提高用户的体验。

## 二、使用 <th> 元素添加标题

[<th>](#)元素 ('th' 代表 'table header').

## 三、允许单元格跨越多行和列

属性	值	描述
colspan	数字	比如, <code>colspan="2"</code> 使一个单元格的宽度是两个单元格。
rowspan	数字	

## 四、为表格中的列提供共同的样式

HTML有一种方法可以定义整列数据的样式信息：

就是 `<col>` 和 `<colgroup>` 元素

元素	描述	属性
<code>&lt;colgroup&gt;</code>	表格列组 (Column Group <code>&lt;colgroup&gt;</code> ) 标签用来定义表中的一组列表。	
<code>&lt;col&gt;</code>	定义用于表格列的属性。定义“列的样式”,每一个 <code>&lt;col&gt;</code> 都会制定每列的样式	<code>span</code>

```
1 <table>
2   <colgroup>
3     <col><!--第一列一定要写-->
4     <col style="background-color: yellow"><!--表示样式应用在第二列-->
5   </colgroup>
6   <tr>
7     <th>Data 1</th>
8     <th>Data 2</th>
9   </tr>
10  <tr>
11    <td>Calcutta</td>
12    <td>Orange</td>
13  </tr>
14 </table>
```

如果你想把这种样式信息应用到每一列(即用来制定你想要让这个样式应用到表格中多少列),我们可以只使用一个 `<col>` 元素,不过需要包含 `span` 属性,像这样:

```
1 <colgroup>
2   <col style="background-color: yellow" span="2">
3 </colgroup>
```

## HTML表格高级特性和可访问性

### 一、使用 `<caption>` 为你的表格增加一个标题

```
1 <table>
2   <caption>这里可以放标题</caption><!--用summary也可以,但是更推荐<caption>可以帮助残疾人-->
3 </table>
```

## 二、<thead>, <tfoot>, 和 <tbody> 结构

```
1 <table>
2   <thead></thead><!--表头:第一行中往往都是每列的标题-->
3   <tbody></tbody><!--正文:注意: <tbody> 总是包含在每个表中, 如果你没有在代码中
   写它, 那就是隐式的, 浏览器为你自动添加了这个标签-->
4   <tfoot></tfoot><!--页脚:一般是最后一行, 往往是对前面所有行的总结,
5   比如, 你可以按照预想的方式将<tfoot>放在表格的底部, 或者就放在 <thead> 的下面。
   (浏览器仍将它呈现在表格的底部)-->
6 </table>
```

## 三、嵌套表格

在一个表格中**嵌套**另外一个表格是可能的, 只要你包含完整的结构, 包括 <table> 元素。**这样通常是不建议的**, 因为这种做法会使标记看上去很难理解, 对使用屏幕阅读的用户来说, 可访问性也降低了。

```
1 <table id="table1">
2   <tr>
3     <th>title1</th>
4     <th>title2</th>
5   </tr>
6   <tr>
7     <td id="nested">
8       <table id="table2">
9         <tr>
10          <td>cell1</td>
11          <td>cell2</td>
12          <td>cell3</td>
13        </tr>
14      </table>
15    </td>
16    <td>cell2</td>
17  </tr>
18  <tr>
19    <td>cell4</td>
20    <td>cell5</td>
21  </tr>
22 </table>
```

### 1、使用列和行的标题

屏幕阅读设备会识别所有的标题, 然后在它们和它们所关联的单元格之间产生编程关联。列和行标题的组合将标识和解释每个单元格中的数据, 以便屏幕阅读器用户可以类似于



视力正常的用户的操作来理解表格。

2、scope 属性

属性	作用
scope	<p>这个枚举属性定义了表头元素 (在&lt;th&gt;中定义) 关联的单元格,屏幕阅读设备会识别化的标记, 并一次读出整列或整行(即标题和单元格之间的联系! )</p> <p>属性定义将表头单元与数据单元相关联的方法。</p> <p>属性标识某个单元是否是列、行、列组或行组的表头。</p> <p>属性不会在普通浏览器中产生任何视觉变化。屏幕阅读器可以利用该属性。</p>

scope属性, 可以添加在<th> 元素中, 用来帮助屏幕阅读设备, 更好地理解那些标题单元格, 这个标题单元格到底是**列标题**呢, 还是**行标题**。比如: 回顾我们之前的支出记录示例, 你可以明确地将列标题这样定义:

```
1 <thead>
2   <tr>
3     <th scope="col">Purchase购买</th>
4     <th scope="col">location</th>
5     <th scope="col">date</th>
6     <th scope="col">Evaluation</th>
7     <th scope="col">Cost (€)</th>
8   </tr>
9 </thead>
```

以及每一行都可以这样定义一个行标题 (如果我们已经使用了 th 和 td 元素)如下:

My spending record

Purchase	Location	Date	Evaluation	Cost (€)
Haircut	Hairdresser	12/09	Great idea	30
Lasagna	Restaurant	12/09	Regrets	18
Shoes	Shoeshop	13/09	Big regrets	65
Toothpaste	Supermarket	13/09	Good	5
SUM				118

```
1 <tbody>
2   <tr>
3     <th scope="row">Haircut</th>
4     <td>Hairdresser</td>
5     <td>12/09</td>
6     <td>Great idea</td>
7     <td>30</td>
8   </tr>
9   <tr>
10    <th scope="row">Lasagna</th>
```

```

11 <td>restaurant</td>
12 <td>12/09</td>
13 <td>regrets</td>
14 <td>18</td>
15 </tr>
16 </tbody>

```

屏幕阅读设备会识别这种结构化的标记，并一次读出整列或整行，比如：

scope 还有两个可选的值：**colgroup** 和 **rowgroup**。这些用于位于多个列或行的顶部的标题。如果你回顾这部分文章开始部分的 "Items Sold August 2016" 表格如下。你会看到 "Clothes" 单元格在 "Trousers", "Skirts", 和 "Dresses" 单元格的上面。这几个单元格都应该被标记为 (<th>), 但是 "Clothes" 是一个位于顶部且定义了其他三个子标题的标题。因此 "Clothes" 应该有一个 scope="colgroup" 属性，而另外三个子标题应该有 scope="col" 属性。

Items Sold August 2016

		Clothes			Accessories	
		Trousers	Skirts	Dresses	Bracelets	Rings
Belgium	Antwerp	56	22	43	72	23
	Gent	46	18	50	61	15
	Brussels	51	27	38	69	28

```

1 <table>
2 <caption>Items Sold August 2016</caption>
3 <tbody>
4 <tr>
5 <td></td>
6 <td></td>
7 <th colspan="3" scope="colgroup">Clothes</th>
8 <th colspan="2" scope="colgroup">Accessories</th>
9 </tr>
10 <tr>
11 <td></td>
12 <td></td>
13 <th scope="col">Trousers</th>
14 <th scope="col">Skirts</th>
15 <th scope="col">Dresses</th>
16 <th scope="col">Bracelets</th>
17 <th scope="col">Rings</th>
18 </tr>
19 <tr>
20 <th rowspan="3" scope="rowgroup">Belgium</th>
21 <th scope="row">Antwerp</th>

```

```

22 <td>56</td>
23 <td>22</td>
24 <td>43</td>
25 <td>72</td>
26 <td>23</td>
27 </tr>
28 <tr>
29 <th scope="row">Gent</th>
30 <td>46</td>
31 <td>18</td>
32 <td>50</td>
33 <td>61</td>
34 <td>15</td>
35 </tr>
36 </tbody>
37 </table>

```

### 3、id 和标题属性（替代 scope 属性）

如果要替代 **scope** 属性，可以使用 **id** 和 **headers** 属性来创造标题与单元格之间的联系。使用方法如下：

1. 为每个<th> 元素添加一个唯一的 id 。
2. 为每个 <td> 元素添加一个 headers 属性。每个单元格的headers 属性需要**包含它从属于的所有标题的id**，之间用空格分隔开。

这会给你的HTML表格中每个单元格的位置一个明确的定义。像一个电子表格一样，通过 **headers** 属性来定义属于哪些行或列。为了让它工作良好，表格同时需要列和行标题。回到我们的花费成本示例，前两个片段可以重写为：

#### My spending record

How I chose to spend my money

Purchase	Location	Date	Evaluation	Cost (€)
Haircut	Hairdresser	12/09	Great idea	30
Lasagna	Restaurant	12/09	Regrets	18
Shoes	Shoeshop	13/09	Big regrets	65
Toothpaste	Supermarket	13/09	Good	5
SUM				118

```

1 <thead>
2 <tr>
3 <th id="purchase">Purchase</th>
4 <th id="location">Location</th>
5 <th id="date">Date</th>

```

```
6 <th id="evaluation">Evaluation</th>
7 <th id="cost">Cost (€)</th>
8 </tr>
9 </thead>
10 <tbody>
11 <tr>
12 <th id="haircut">Haircut</th>
13 <td headers="location haircut">Hairdresser</td>
14 <td headers="date haircut">12/09</td>
15 <td headers="evaluation haircut">Great idea</td>
16 <td headers="cost haircut">30</td>
17 </tr>
18 ...
19 </tbody>
20 <--注意：这个放进为标题单元格和数据单元格之间创造了非常精确的联系。但是这个方法
    使用了大量的标记，所以容错率比较低。
21 使用 scope 的方法对于大多数表格来说，也够用了。-->
```

---

# CSS

本主题包含以下模块，建议按顺序阅读这些模块。你应该从第一个模块开始。

**构建 CSS 块**

**样式化文本**

**CSS 布局**

---

## 学习CSS 第一步 在此模块

1. [什么是CSS?](#)
2. [开始学习CSS](#)
3. [CSS代码是如何组织的](#)
4. [CSS是如何工作的](#)
5. [开始使用你的新知识](#)

CSS（层叠样式表）用于设置和布置网页 - 例如，更改内容的字体，颜色，大小和间距，将其拆分为多个列，或添加动画和其他装饰功能。包括它如何工作的基础知识，语法是什么样的，以及如何开始使用它来为HTML添加样式。

---

# 什么是CSS?

预备知识:	基本的计算机知识, <a href="#">安装基础软件</a> , <a href="#">文件处理</a> 的基 知识, 还有HTML基础 (学习 <a href="#">HTML概述</a> 。)
目标:	了解什么是CSS。

# 让我们开始CSS的学习之旅

预备知识:	基本的计算机知识, <a href="#">安装基础软件</a> , <a href="#">文件处理</a> 的基 知识, 还有HTML基础 (学习 <a href="#">HTML概述</a> 。)
目标:	理解 HTML文档链接CSS文档的几个基本套路, 并能运用所学的CSS, 做些文字上的格式化操作。

```
1 <link rel="stylesheet" href="styles.css">
2 属性rel: 让浏览器知道有CSS文档存在（所以需要遵守CSS样式的规定），属性 href 指
   定，寻找CSS文件的位置。
```

## 一、改变元素的默认行为list-style-type

属性 `list-style-type` 可以设置列表元素的 marker（比如圆点、符号、或者自定义计数器样式）。

标题：默认使用大号粗体；列表：旁总有项目符号。这是因为浏览器自带一个包含默认样式的样式表，默认对任何页面有效。

```
1 li{
2   list-style-type:none; /*移除自带项目符号*/
3   list-style-type:square/*实心方块*/
4 }
```

属性	值	描述
list-style-type	none	不显示列表项的标记。
	square	实心方块
	disc	● 实心圆点 (默认值)
	circle	○ 空心圆点
	upper-roman	大写罗马数字(I, II, III, IV, V, 等。)

	upper-latin	大写字母
	lower-latin	小写字母

## 二、选择器：

CSS 给我们提供了几种定位元素的方法：

### 使用类名class

让所有class为special的元素应用如下样式：

```
1 .special{
2   color:green;
3   font-weight:bold;
4 }
```

HTML 元素选择器跟类一起出现，标签名+类名

```
1 p.special{
2   color:orange;
3 }
```

选中每个 special 类的 li 元素和special类的span元素：

```
1 li.special,span.special{
2   color:orange;
3 }
```

### ID选择器 #onething

## 三、根据元素在文档中的位置确定样式（选择器）

### 后代选择器

可以选择作为某元素后代的元素。也叫**包含选择符的选择器**：是指仅选择嵌套在<li> 元素内的<em>如下：

```
1 li em{
2   color:red;
3 }
```

### 相邻选择符

在两个选择器之间添加一个 + 号 (成为 **相邻选择符**)

```
1 h1 + p{
2   font-size: 200%;
3 }
```

### 根据状态确定样式

当我们修改一个链接的样式时我们需要定位（针对） `<a>`（锚）标签。取决于是否是未访问的、访问过的、被鼠标悬停的、被键盘定位的，亦或是正在被点击当中的状态，这个标签有着不同的状态。

伪类	描述
<code>a:link</code>	未访问过的链接，即正常状态下样式，如果 href 属性为空、为"#"符号，则显示visited样式,如果为"#"或者字符串、javascript:void(0)表达式则显示:li
<code>a:visited</code>	被访问过的链接
<code>a:hover</code>	链接被鼠标悬停的时候的样式

```
1 a:link{color:pink;}/*正常状态下的样式，即没有访问之前的样子，*/
2 a:visited{color:green;}/*被访问过的样式*/
3 a:hover{text-decoration:none;}/*鼠标悬停在链接上的样式*/
```

**需要注意**，对一个实际的站点，需要让浏览者知道“链接是链接”。为了让浏览者注意到一段文字中的某些部分是可点击的，最好**保留link状态下的下划线**。这是下划线的本来作用。你的访客可能在一台使用鼠标和键盘操作的计算机前，也可能正在使用带有触摸屏的手机，或者正在使用屏幕阅读软件读出文档内容，或者他们需要使用更大的字体，或者仅仅使用键盘浏览站点。一个朴素的HTML文档一般来说对任何人都是可以无障碍访问的，**当你开始为它添加样式，记得不要破坏这种可访问性**。

# 如何构建CSS

## 在文档中应用CSS的三种方法：

### 一、外部样式表：

外部样式表是指将CSS编写在扩展名为.css 的单独文件中，并从HTML<link> 元素引用它的情况：

```
1 <head>
2 <meta charset="utf-8">
```

```

3 <title>My CSS experiment</title>
4 <link rel="stylesheet" href="styles.css">
5 </head>

```

## 二、内部样式表：

将CSS放在HTML文件<head>标签里的<style>标签之中。但该方法 and 外部样式表比起来更加低效。

```

1 <head>
2 <meta charset="utf-8">
3 <title>My CSS experiment</title>
4 <style>
5 p{
6 </style>
7 </head>

```

## 三、内联样式：

内联样式表存在于HTML元素的style属性之中。其特点是每个CSS表只影响一个元素：

```

1 <body>
2 <h1 style="color: blue;background-color: yellow;border: 1px solid
3 black;">Hello World!</h1>
4 <p style="color:red;">This is my first CSS example</p>
5 </body>

```

**注意：**除非你有充足的理由，否则不要这样做！它难以维护（在需要更新时，你必须在修改同一个文档的多处地方），并且这种写法将文档结构和文档表现混合起来了，这使得代码变得难以阅读和理解。将不同类型的代码分开存放在不同的文档中，会让我们的工作更加清晰。

## 四、函数

一个函数由**函数名**和一些**括号**组成，其中放置了该函数的**允许值**

```

1 .box {
2 padding: 10px;
3 width: calc(90% - 30px);/*此框的宽度为包含块宽度的90%，减去30像素，我们无法
4 提前计算它，只是在CSS中输入值，因为我不知道90%会是什么*/
5 background-color: rebeccapurple;
6 color: white;
7 }

```

函数名称	语法	描述
calc()函数	/* property: calc(expression) */ width: calc(100% - 80px);	用一个表达式作为它的参数，用这个的结果作为值。这个表达式可以是任



		操作符的组合，采用标准操作符处理简单表达式。
rotate()函数	<pre> .box {   margin: 30px;   width: 100px;   height: 100px;   background-color: rebeccapurple;   transform: rotate(0.8turn) }</pre>	<p>函数定义了一种将元素围绕一个定点(<code>transform-origin</code>属性指定)旋转的转换。指定的角度定义了旋转的角度，角度为正，则顺时针方向旋转，否则逆时针方向旋转。旋转180°也被称为点反射。元素旋转的固定点 - 如上所述 - 也称为<b>原点</b>。这默认为元素的中心，但你可使用<b><code>transform-origin</code></b>属性设置自己的转换原点。</p>

## CSS transform属性

该属性允许你**旋转**，**缩放**，**倾斜**或**平移**给定元素。这是通过修改CSS视觉格式化模型的坐标空间来实现的。

只能转换由盒模型定位的元素。根据经验，如果元素具有`display: block`，则由盒模型定位元素。

	描述	参数值
		<code>transform: matrix(1, 2, 3, 4, 5, 6);</code>

## 五、@规则

### @import

有规则名和值，要将额外的样式表导入主CSS样式表，可以使用@import:

```
1 @import 'styles2.css';
```

### @media

使用**媒体查询**来应用CSS，仅当某些条件成立(例如，当屏幕分辨率高于某一数量，或屏幕宽度大于某一宽度时)。

```

1 body{background-color:pink;}/*粉红色的背景色*/
2 @media (min-width: 30 em){
3   body{background-color: blue;}/*该部分仅适用于视口大于30em的浏览器。如果浏览器的宽度大于30em，则背景色将为蓝色。*/
4 }
```

---

---

# CSS如何运行

前置知识:	基础计算机知识、基本软件安装、简单文件知识、HTML基础
目标:	理解浏览器如何加载CSS和HTML、浏览器遇到无法解析的CSS会发生什么

---

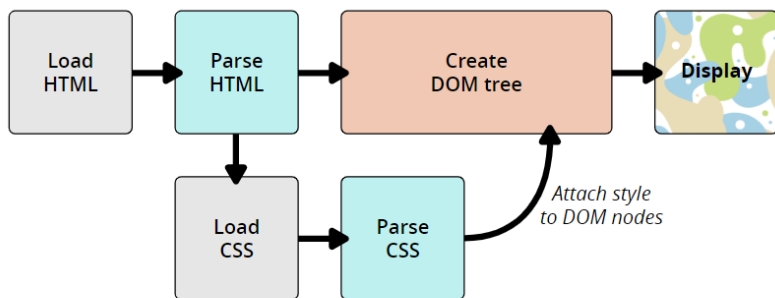
## CSS究竟是怎么工作的？

当浏览器展示一个文件的时候，它必须兼顾文件的内容和文件的样式信息，下面是它处理文件的标准的流程。下面的步骤是浏览加载网页的简化版本，而且不同的浏览器在处理文件的时候会有不同的方式，但是下面的步骤基本都会出现：

1. 浏览器载入HTML文件（比如从网络上获取）。
2. 将HTML文件转化成一个DOM（Document Object Model），DOM是文件在计算机内存中的表现形式，下一节将更加详细的解释DOM。
3. 接着，浏览器会拉取该HTML相关的大部分资源，比如嵌入到页面的**图片、视频和CSS样式**。JavaScript则会稍后进行处理，简单起见，同时此节主讲CSS，所以这里对如何加载JavaScript不会展开叙述。
4. 浏览器拉取到CSS之后会进行解析，根据选择器的不同类型（比如element、class、id等等）把他们分到不同的“桶”中。浏览器基于它找到的不同的选择器，将不同的规则（基于选择器的规则，如元素选择器、类选择器、id选择器等）应用在对应的DOM的节点中，并添加节点依赖的样式（这个中间步骤称为渲染树）。
5. 上述的规则应用于渲染树之后，渲染树会依照应该出现的结构进行布局。
6. 网页展示在屏幕上（这一步被称为着色）。

结合下面的图示更形象：

结合下面的图示更形象：



## 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。

### 一个真实的DOM案例：

以下列HTML代码为例：

```
1 <p>
2   Let's use:
3   <span>Cascading</span>
4   <span>Style</span>
5   <span>Sheets</span>
6 </p>
```

在上例这个DOM中，<p>元素对应了父节点，它的子节点是：一个text节点和三个对应了<span>元素的节点，SPAN节点同时也是他们中的Text节点的父亲节点。

```
1 P父亲
2 └─ "Let's use:" 他的父节点是p
3 └─ SPAN
4   └─ "Cascading" 他的父节点是span
5   └─ SPAN
6     └─ "Style" 他的父节点是span
7     └─ SPAN
8       └─ "Sheets" 他的父节点是span
```

上面就是浏览器怎么解析之前那个HTML片段——它生成上图的**DOM树形结构**并将它按照如下输出到浏览器：

## Let's use: Cascading Style Sheets

接下来让我们看看添加一些CSS到文件里加以渲染，同样使用上面的HTML代码：  
以下为CSS代码：

```
1 span {  
2   border: 1px solid black;  
3   background-color: lime;  
4 }
```

浏览器会解析HTML并创建一个DOM，然后解析CSS。唯一的选择器是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。

## 当浏览器遇到无法解析的CSS代码会发生什么

- 书写CSS规则的过程中遇到了无法理解的属性或者值
- 在你书写了错误的CSS代码（或者误拼写）
- 又或者当浏览器遇到对于它来说很新的，还没有支持的CSS代码的时候同样会发生（直接忽略）
- 或者使用最新版本的浏览器。
- 当浏览器遇到无法解析的选择器的时候，他会直接忽略整个选择器规则，然后解析下一个CSS选择器。

答案就是浏览器什么也不会做，继续解析下一个CSS样式！

你可以为同一个元素指定多个CSS样式，来解决有些浏览器不兼容\不支持新CSS特性的问题（比如指定两个width）。举例来说，一些老的浏览器不接收calc()(calculate的缩写，CSS3新增，为元素指定动态宽度、长度等，注意此处的动态是计算之后得一个值)作为一个值。我可能使用它结合像素为一个元素设置了动态宽度（如下），老式的浏览器由于无法解析忽略这一行；新式的浏览器则会把这一行解析成像素值，并且覆盖第一行指定的宽度，如下代码：

```
1 .box{  
2   width: 500px;  
3   width: calc(100% - 50px);  
4 }
```

# CSS构建

这个模块着眼于 级联 和 继承，所有可供使用的选择器类型，单位，尺寸，背景、边框样式，调试，等等等等。

本文目标是，在你进一步了解 为 [为文本添加样式](#) 和 [CSS布局](#) 之前，为你提供一个助你写出合格CSS和理解所有基本理论的工具箱。

## 层叠与继承

理解CSS的一些最基本的概念——**层叠**、**优先级**和**继承**——这些概念决定着如何将CSS应用到HTML中，以及如何解决冲突。理解这些东西将为您以后节省很多痛苦！望您仔细阅读本节，并在继续下一步学习之前，确保您是否理解了这些概念。

预备知识:	基本的计算机知识， <a href="#">安装基本的软件</a> ， <a href="#">文件处理</a> 基础知识，HTML基础知识（如果不了解HTML，请移步 <a href="#">习HTML入门</a> ），以及CSS如何工作的基本常识（如要了解CSS，请移步 <a href="#">学习CSS第一步</a> 。）
目标:	学习层叠、优先级，以及在CSS中继承是如何工作的。

**继承**的概念，默认情况下，一些css属性继承当前元素的父元素上设置的值，有些则不继承。这也可能导致一些和期望不同的结果。

### 一、层叠

Stylesheets **cascade** (**样式表层叠**) 层叠的意思就是“继承”、“权重”、“覆盖”，通过良好的层级命名更好的实现效果，更少的代码，更多的功能，css规则的顺序很重要；当应用两条同级别的规则到一个元素的时候，写在后面的就是实际使用的规则。**层叠指的是样式的优先级，当产生冲突时以优先级高的为准。**

```
1 h1{color:red;}
2 h1{color:blue;} /*两个h1的规则有相同的优先级，所以顺序在最后的生效。*/
```

### 二、优先级

浏览器是根据优先级来决定，当多个规则有**不同选择器**，对应**相同的元素**的时候，需要使用哪个规则。它基本上是一个衡量选择器具体选择哪些区域的尺度：

- 一个元素选择器不是很具体 — 会选择页面上该类型的所有元素 — 所以它的优先级就会低一些。

- 一个类选择器稍微具体点 — 它会选择该页面中有特定 `class` 属性值的元素 — 所以它的优先级就要高一点。

```
1 .main-heading{color:red}; /*类选择器有更高的优先级，因此就会被应用——即使元素选择器顺序在它后面。*/
2 h1{color:green;}
3 <h1 class="main-heading">This is my heading.</h1>
```

## 类选择器 > 标签（元素）选择器

# 三、继承

继承也需要在上下文中去理解 —— 一些设置在父元素上的css属性是可以被子元素继承的，有些则不能。

例如：设置一个元素的 `color` 和 `font-family`，每个在里面的元素，也都会有相同的属性，除非你直接在元素上设置属性。

```
1 body {
2   color: blue;
3 }
4 span {
5   color: black; /*如果span不设置该规则的话，他默认继承body的blue */
6 }
```

## 一些不能继承的属性

例：在一个元素上设置 `width` 50%，所有的后代不会是父元素的宽度的50%，故而不继承。如果这个也可以继承的话，CSS就会很难使用了！哪些属性属于默认继承很大程度上是由常识决定的。

## 控制继承

CSS 为控制继承提供了四个特殊的通用属性值。每个css属性都接收这些值。

通用属性值	描述
<code>inherit</code>	设置该属性会使子元素属性和父元素相同。实际上，就是 "开启继承".
<code>initial</code>	表示初始值的意思。即设置属性值和浏览器默认样式相同。如果浏览器默认样式为 <code>inherit</code> 。
<code>unset</code>	不固定值，当前元素浏览器或用户设置的CSS忽略，然后如果是具有继承特性的如color, 则使用继承值； 如果是没有继承特性的CSS属性，如background-color, 则使用初始值。 意思是：将属性重置为自然值，也就是如果属性是自然继承那么就是 <code>inherit</code> ，
<code>revert</code>	新的属性, <code>revert</code> ，只有很少的浏览器支持。

```

1 <style>
2   body {color: green;}
3   .my-class-1 a {color: inherit;}
4   .my-class-2 a {color: initial;}
5   .my-class-3 a { color: unset;}
6   a{color:red;}
7 </style>
8 <ul>
9   <li>Default <a href="#">link</a> color</li>
10  <li class="my-class-1">Inherit the <a href="#">link</a> color</li>
11  <li class="my-class-2">Reset the <a href="#">link</a> color</li>
12  <li class="my-class-3">Unset the <a href="#">link</a> color</li>
13 </ul>

```

## 重设所有属性值all: initial/inherit/unset

属性	取值	解释
<b>all</b> 简写属性 将除却 <code>unicode-bidi</code> 与 <code>direction</code> 之外的所有属性重设至其初始值，或继承值。	initial	该元素所有的属性都取默认值（即初始值）
	inherit	其值，继承父元素的属性。
	unset	所有属性都相当于不设置值，默认可继承的继承，
		来源地址: <a href="https://www.php.cn/css-tutorial-45">https://www.php.cn/css-tutorial-45</a>

```

1 <style>
2   body {
3     color: green;
4     border:4px solid red;
5   }
6   blockquote{background-color:black; border:5px solid yellow;}
7   .fix-this{
8     color:blue;
9     all:initial;/*该元素所有的属性值取默认值（初始值）*/
10  }
11 </style>
12 <blockquote><p>This blockquote is styled</p></blockquote>

```

```
13 <blockquote class="fix-this"><p>This blockquote is not styled</p></blockquote>
```

## 四、理解层叠(选择器优先级)

有三个因素需要考虑，根据重要性排序如下，前面的更重要：

### 1. 重要程度

**2. 优先级：**在了解了顺序的重要性后，会发现，有些规则在最后出现，但是却应用了前面的规则。这是因为前面的有更高的优先级 — 它范围更小，因此浏览器就把它选择为元素的样式。就像“类选择器”的权重大于“元素选择器”，因此类上定义的属性将覆盖应用于元素上的属性。

不同类型的选择器有不同的分数值，把这些分数相加就得到特定选择器的权重，然后就可以进行匹配。一个选择器的优先级可以说是由四个部分相加 (分量)，是个十百千 — 四位数的四个位数：

- **千位：**如果声明在 `style` 的属性 (内联样式) 则该位得一分。这样的声明没有选择器，所以它得分总是1000。
- **百位：**选择器中包含ID选择器则该位得一分。
- **十位：**选择器中包含类选择器、属性选择器或者伪类则该位得一分。
- **个位：**选择器中包含元素、伪元素选择器则该位得一分。

**警告：**在进行计算时不允许进行进位，例如，20 个类选择器仅仅意味着 20 个十位，而不能视为 两个百位，也就是说，无论多少个类选择器的权重叠加，都不会超过一个 ID 选择器。

选择器	千位	百位	十位	个位	优先级
<code>h1</code>	0	0	0	1	0001
<code>h1 + p::first-letter</code>	0	0	0	3	0003
<code>li &gt; a[href*="en-US"] &gt; .inline-warning</code>	0	0	2	2	0022
<code>#identifier</code>	0	1	0	0	0100
内联样式	1	0	0	0	1000

**3. 资源顺序：**如果你有超过一条规则，而且都是相同的权重，那么最后面的规则会应用。可以理解为后面的规则覆盖前面的规则，直到最后一个开



始设置样式。

我们从下往上，看看浏览器是如何决定该应用哪个css规则的。

```
1 <style>
2 /* specificity: 0101 */
3 #outer a {
4   background-color: red;
5 }
6 /* specificity: 0201 */
7 #outer #inner a {
8   background-color: blue;
9 }
10 /* specificity: 0104 */
11 #outer div ul li a {
12   color: yellow;
13 }
14 </style>
15 <div id="outer" class="container">
16   <div id="inner" class="container">
17     <ul>
18       <li class="nav"><a href="#">One</a></li>
19       <li class="nav"><a href="#">Two</a></li>
20     </ul>
21   </div>
22 </div>
```

## !important

!important 是个特殊的 CSS 可以用来覆盖所有上面所有优先级计算，不过需要很小心的使用 — `!important`。用于修改特定属性的值，能够覆盖普通规则的层叠。如下图例子：

看看这个例子，有两个段落，其中一个有ID。

This is a paragraph.

One selector to rule them all!

```
#winning {  
  background-color: red;  
  border: 1px solid black;  
}  
  
.better {  
  background-color: gray;  
  border: none !important;  
}  
  
p {  
  background-color: blue;  
  color: white;  
  padding: 5px;  
}
```

```
<p class="better">This is a paragraph.</p>  
<p class="better" id="winning">One selector to rule them all!</p>
```

**注：**覆盖 `!important` 唯一的办法就是另一个 `!important` 具有相同优先级而且顺序靠后，或者更高优先级。了解 `!important` 是为了在阅读别人代码的时候知道有什么作用。但是，强烈建议除了非常情况不要使用它。`!important` 改变了层叠的常规工作方式，它会使调试 CSS 问题非常困难，特别是在大型样式表中。

## 简而言之-相互冲突的声明

相互冲突的声明将按以下顺序适用，后一种声明将覆盖前一种声明：

1. 用户代理样式表中的声明(例如，浏览器的默认样式，在没有设置其他样式时使用)。
2. 用户样式表中的常规声明(由用户设置的自定义样式)。
3. 作者样式表中的常规声明(这些是我们web开发人员设置的样式)。
4. 作者样式表中的 `!important` 声明
5. 用户样式表中的 `!important` 声明

对于web开发人员的样式表来说，覆盖用户样式表是有意义的，因此设计可以按预期进行，但是有时用户充足的理由覆盖web开发人员样式，正如上面提到的一这可以通过在他们的规则中使用 `!important` 来实现。

# CSS选择器

[CSS](#)中，选择器用来指定网页上我们想要样式化的[HTML](#)元素。有CSS选择器提供了很多种方法，所以在选择要样式化的元素时，我们可以做到很精细的地步。

学习前提：	计算机的基本知识， <a href="#">安装了基础软件</a> ， <a href="#">处理文件</a> 的基本知识，HTML基础（学习 <a href="#">HTML介绍</a> ）以及对CS作方式的了解（学习 <a href="#">CSS初步</a> ）
目标：	详细学习CSS选择器的工作方式。

CSS选择器是CSS规则的第一部分。它是元素和其他部分组合起来告诉浏览器，哪个HTML元素应当是被选为应用规则中的CSS属性值的方式。选择器所选择的元素，叫做“**选择器的对象**”。

```
h1 {  
  color: blue;  
  background-color: yellow;  
}  
  
p {  
  color: red;  
}
```

## 选择器列表

如果你有多个使用相同样式的CSS选择器，那么这些单独的选择器可以被混编为一个“**选择器列表**”，规则就可以应用到所有的单个选择器上了。例如，如果我的h1和.special类有相同的CSS，可以把它们写成两个分开的规则。

```
1 h1 {  
2   color: blue;  
3 }  
4 .special {  
5   color: blue;  
6 }
```

我也可以将它们组合起来，在它们之间加上一个逗号，变为**选择器列表**。

```
1 h1,  
2 .special {  
3   color: blue;  
4 }
```

空格可以在逗号前或后，如果每个选择器都**另起一行**，会更好读些。当你使用选择器列表时，如果任何一个选择器无效（存在语法错误），那么整条规则都会被忽

略。

# CSS选择器的种类

## 一、ID选择器

```
1 #unique { } /*id选择器: */
```

**备注：**ID所指特定，会优先于大多数其他选择器。所以很难处理它们。大多数情况下，给一个元素加个类，而不是使用ID，会更好。不过要是ID是唯一一种，指定这个元素的方式的话——也许是因为你没法访问标记标记，因此不能编辑——这种方式可行。

## 二、类型选择器（也叫标签选择器或元素选择器）、类

```
1 h1 { } /*类型选择器也叫做“标签名选择器”或者是“元素选择器”，指向所有HTML元素<h1>。*/
2 .box { } /*它也包含了一个class的选择器*/
```

## 三、属性选择器

用属性选择器来选中带有特定属性的元素。

### 1、存否和值选择器

这些选择器允许基于一个元素自身是否存在（例如[href](#)）或者基于各式各样的按属性值的匹配，来选取元素。

```
1 a [title]{} /*这组选择器根据，一个元素上的某个标签的属性的存在以选择元素的不同方式*/
2 a[href="https://example.com"]{} /*一个有特定值的标签属性是否存在来选择*/
```

选择器	示例
<code>[attr]</code>	<code>a[title]</code>
<code>[attr=value]</code>	<code>a[href="https://example.com"]</code>
<code>[attr~=value]</code>	<code>p[class~="special"]</code>
<code>[attr =value]</code>	<code>div[lang = "zh"]</code>

下例中，看这些选择器是怎样使用的：

- 使用`li[class]`，我们就能匹配任何有class属性的选择器。这匹配了除了第一项以外的**所有项**。
- `li[class="a"]`匹配带有一个a类的选择器，不过不会选中一部分值为a而另一部分是另一个用空格隔开的值的类，它选中了**第二项**。
- `li[class~="a"]`会匹配一个a类，不过也可以匹配一系列用空格分开、包含a类的值，它选中了**第二和第三项**。

```

1  li[class]{
2    font-size:200%;
3  }
4  li[class="a"]{
5    background-color:yellow;
6  }
7  li[class~="a"]{
8    color: red;
9  }
10 <h1>Attribute presence and value selectors属性存在和值选择器</h1>
11 <ul>
12 <li>Item 1项目1</li>
13 <li class="a">Item 2项目2</li>
14 <li class="a b">Item 3项目3</li>
15 <li class="ab">Item 4项目4</li>
16 </ul>

```

## 2、子字符串匹配选择器

这些选择器让更高级的属性的值的子字符串的匹配变得可行。例如，如果你有`box-warning`和`box-error`类，想把开头为“box-”字符串的每个物件都匹配上的话，你可以用`[class^="box-"]`来把它们两个都选中。

选择器	示例
<code>[attr^=value]</code>	<code>li[class^="box-"]</code>
<code>[attr\$=value]</code>	<code>li[class\$="-box"]</code>
<code>[attr*=value]</code>	<code>li[class*="box"]</code>

下个示例展示了这些选择器的用法：

- `li[class^="a"]`匹配了任何值开头为a的属性，于是匹配了前两项。

- `li[class$="a"]` 匹配了任何值结尾为a的属性，于是匹配了第一和第三项。
- `li[class*="a"]` 匹配了任何值的字符串中出现了a的属性，于是匹配了所有项。

```
1 li[class^="a"] {
2   font-size: 200%;
3 }
4 li[class$="a"] {
5   background-color: yellow;
6 }
7 li[class*="a"] {
8   color: red;
9 }
10 <h1>Attribute substring matching selectors</h1>
11 <ul>
12   <li class="a">Item 1</li>
13   <li class="ab">Item 2</li>
14   <li class="bca">Item 3</li>
15   <li class="bcabc">Item 4</li>
16 </ul>
```

### 3、大小写不敏感的情况下，匹配属性值

想在大小写不敏感的情况下，匹配属性值，可以在闭合括号之前，使用 `i` 值。这个标记告诉浏览器，要以大小写不敏感的方式匹配ASCII字符。没有了这个标记的话，值会按照文档语言对大小写的处理方式，进行匹配——HTML中是大小写敏感的。

```
1 <h1>Case-insensitivity</h1>
2 <ul>
3   <li class="a">Item 1</li>
4   <li class="A">Item 2</li>
5   <li class="Ab">Item 3</li>
6 </ul>
7
```

---

## 四、伪类与伪元素

选择器被称为**伪类**和**伪元素**。这一类选择器的数量众多，通常用于很明确的目的。

## 伪类和伪元素参考列表[https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building\\_blocks/Selectors/Pseudo-classes\\_and\\_pseudo-elements](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements)

- **伪类**它用于选择处于特定状态的元素（用来样式化一个元素的特定状态），比如：当它们是这一类型的第一个元素时，或者是当鼠标指针悬浮在元素上面的时候。伪类就是开头为冒号的关键字：

```
1 a:hover{} /*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*/
```

```
1 article p:first-child { /*选中文章中的第一个子元素*/
2   font-size: 120%;
3   font-weight: bold;
4 }
5 p:last-child{ /*:last-child CSS 伪类代表父元素的最后一个子元素。*/
6   background-color:green;
7 }
8 <article>
9   <p>a garlic bulb一头大蒜.</p>
10  <p> this is a corn.</p>
11 </article>
```

### 用户行为伪类

一些伪类只会在用户以某种方式和文档交互的时候应用。这些**用户行为伪类**，有时叫做**动态伪类**，表现得就像是一个类在用户和元素交互的时候加到了元素上一样。案例包括：

<a href="#">:hover</a>	只会在用户将指针挪到元素上的时候才会激活，一般就是链接
<a href="#">:focus</a>	只会在用户使用键盘控制，选定元素的时候激活
visited	
link	

- **伪元素**选择一个元素的某个部分而不是元素自己；它表现得是像你往标记文本中加入全新的HTML元素一样，而不是向现有的元素上应用类。伪元素开头为双冒号`::`。

```
1 P::first-line{} /*是会选择第一个元素中的第一行，类似<span>包在了第一个被格式化的行外面，然后选择这个<span>。*/
```

**备注：**一些早期的伪元素曾使用单冒号的语法，所以你可能会在代码或者示例中看到。现代的浏览器为了保持后向兼容，支持早期的带有单双冒号语法的伪元素。

```
1 article p::first-line{/*把两段的在浏览器中显示的第一行都选中了*/
2   font-size:120%;
3   color:red;}
4 <article>
5   <p>Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.</p>
6   <p>Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.</p>
7 </article>
```

---

## 把伪类和伪元素组合起来

如果你想让**第一段的第一行**加粗，你需要把`:first-child`和`::first-line`选择器放到一起。

```
1 /*它会选择<article>元素里面的第一个<p>元素的第一行。*/
2 article p:first-child::first-line {
3   font-size: 120%;
4   font-weight: bold;
5 }
6 article>
7   <p>Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.</p>
8   <p>Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.</p>
9 </article>
```

---



## 生成带有::before和::after的内容

::before和after是一组**特别的伪元素**，它们和content属性一同使用，使用CSS将内容插入到文档中。用这些插入一个文本字符串，和在下面的实时示例里那样。试着改变content属性的文本值，看看输出是怎么改变的。文本插入到了元素的末尾或开头。

**contents**是CSS 属性用于在元素的 `::before` 和 `::after` 伪元素中插入内容。使用content 属性插入的内容都是匿名的**可替换元素**。

```
1 .box::before{
2   content:"This should show before the other content这应该在其他内容之前显示;"
3 }
4 .box::after{
5   content:"插入在组后面;"
6 }
7 <p class="box">我的HTML页面中的内容</p>
```

从CSS插入文本字符串，我们并不会在Web浏览器上经常这么做，因为对于一些屏幕阅读器来说，文本是不可见的，而且对于未来别人的查找和编辑也不是很方便。

这些伪元素的更推荐的用法是插入一个图标，例如下面的示例加入的一个小箭头，作为一个视觉性的提示，而且我们并不希望屏幕阅读器读出它。

```
1 .box::after {
2   content: " ➡ "
3 }
4 <p class="box">Content in the box in my HTML page.</p>
```

这些伪元素经常用于插入空字符串，可以像页面上的其他元素被样式化。

下个示例，用 ::before伪元素加入了个空字符串。把它设为display: block，以让它可以用width和height进行样式化。然后用CSS像任何元素那样样式化。你可以摆弄CSS，改变它的外观和行为。

```
1 .box::before {
2   content: "";
3   display: block;
4   width: 100px;
5   height: 100px;
6   background-color: rebeccapurple;
```

```
7 border: 1px solid black;
8 }
9 <p class="box">Content in the box in my HTML page.</p>
```

`::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”，无论什么时候你看到了这些选择器，都要看下`content`属性，以了解文档中添加了什么。

## 伪类和伪元素参考列表

[https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building\\_blocks/Selectors/Pseudo-classes\\_and\\_pseudo-elements](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements)

### 五、运算符

```
1 article > p {} /*这一组选择器可以将其他选择器组合起来，更复杂的选择元素。该示例
用运算符(>)选择了<article>元素的初代子元素。*/
```

### 六、全局选择器(通配符 \*)

全局选择器，由星号(\*)代指的，它选中了文档中的所有内容（或者是父元素中的所有内容，比如，它紧随在其他元素以及邻代运算符之后的时候）。下例中，我们已经用全局选择器，移去了所有元素上的外边距。这就是说，和浏览器以外边距隔开标题和段的方式，默认加上的样式不同的是，每个物件都紧紧地挨在一起，我们不能那么容易就看清楚不同的段。

```
1 <style>
2   *{margin:0;}
3 </style>
4 <h1>Universal selector</h1>
5 <p>Veggies es bonus vobis, proinde vos postulo essum magis <span>kohlrabi
welsh onion</span> daikon amaranth tatsoi tomatillo
6   melon azuki bean garlic.</p>
7 <p>Gumbo beet greens corn soko <strong>endive</strong> gumbo gourd. Parsl
ey shallot courgette tatsoi pea sprouts fava bean collard
8   greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea pea
nut soko zucchini.</p>
```

#### 使用全局选择器，让选择器更易读

```
1 article :first-child{}
2 article:first-child{}/*用作<article>元素选择器的一个兄弟选择器*/
3 /*上面两个看上去都差不多，有空格和每空格*/
```

但是这会和`article:first-child`混淆，而后者选择了作为其他元素的第一子元素的`<article>`元素。

为了避免这种混淆，我们可以向`:first-child`选择器加入**全局选择器**，这样选择器所做的事情很容易就能看懂。选择器正选中`<article>`元素的*任何*第一子元素如下：

```
1 article *frist-child{ /*选择器正选中<article>元素的任何第一子元素：*/
```

## 选择器参考表

可以把这个作为回头的参考，以后需要查询文献中提到的选择器的时候，或者是在你广义上实验CSS的时候。

选择器	示例
<a href="#">类型选择器</a>	<code>h1 { }</code>
<a href="#">通配选择器</a>	<code>* { }</code>
<a href="#">类选择器</a>	<code>.box { }</code>
<a href="#">ID选择器</a>	<code>#unique { }</code>
<a href="#">标签属性选择器</a>	<code>a[title] { }</code>
<a href="#">伪类选择器</a>	<code>p:first-child { }</code>
<a href="#">伪元素选择器</a>	<code>p::first-line { }</code>
<a href="#">后代选择器</a>	<code>article p</code>
<a href="#">子代选择器</a>	<code>article &gt; p</code>
<a href="#">相邻兄弟选择器</a>	<code>h1 + p</code>
<a href="#">通用兄弟选择器</a>	<code>h1 ~ p</code>

## 七、关系选择器

最后一种选择器被命名为**关系选择器**（Combinator），这是因为它们在其他选择器之间和其他选择器与文档内容的位置之间建立了一种有用的关系的缘故。

### 1、后代选择器

后代选择器——典型用单个空格（）字符——组合两个选择器，比如，第二个选择器匹配的元素被选择，如果他们有一个祖先（父亲，父亲的父亲，父亲的父亲的父亲，等等）元素匹配第一个选择器。选择器利用后代组合符被称作后代选择器。

```
1 .box p { /* 只会匹配处于带有 .box 类的元素里面的 <p> 元素。 */
2   color: red;
3 }
4 <div class="box"><p>Text in .box</p></div>
5 <p>Text not in .box</p>
```

## 2、子代关系选择器( > )

子代关系选择器是个大于号（`>`），只会在选择器选中直接子元素的时候匹配。继承关系上更远的后代则不会匹配。例如，只选中作为`<ul>`的直接子元素的`<li>`元素：

```
1 ul > li { border-top: 5px solid red; } /* 只选中为 <ul> 的直接子元素的 <li> 元素，给了
   它们一个顶端边框。 */
2 <ul>
3   <li>Unordered item</li>
4   <li>Unordered item
5     <ol>
6       <li>Item 1</li>
7       <li>Item 2</li>
8     </ol>
9   </li>
10 </ul>
```

## 3、邻接兄弟选择器 ( + )

邻接兄弟选择器（`+`）用来选中恰好处于另一个在继承关系上同级的元素旁边的物件。例如，选中所有紧随`<p>`元素之后的`<img>`元素：

```
1 p + img { }
```

常见的使用场景是，改变紧跟着一个标题的段的某些表现方面，就像是我下面的示例那样。这里我们寻找一个紧挨`<h1>`的段，然后样式化它。

```
1 h1 + p { color: red; }
2 <article>
3   <h1>A heading</h1>
4   <p>Veggies es bonus vobis, proi.</p>
5   <p>Gumbo beet greens corn soko endive gumbo gourd.</p>
6 </article>
```

---

## 4、通用兄弟 (~)

如果想选中一个元素的兄弟元素，即使它们不直接相邻，也可以使用通用兄弟关系选择器 (~)。要选中所有的<p>元素后任何地方的<img>元素，我们会这样做：

```
1 p ~ img{}
```

示例中，选中了所有的 <h1>之后的<p>元素，虽然文档中还有个 <div>，其后的<p>还是被选中了。

```
1 h1 ~ p {
2   font-weight: bold;
3   background-color: #333;
4   color: #fff;
5   padding: .5em;
6 }
7 <article>
8   <h1>A heading</h1>
9   <p>I am a paragraph.</p>
10  <div>I am a div</div>
11  <p>I am another paragraph.</p>
12 </article>
```

## 5、使用关系选择器

用关系选择器，将学到的选择器组合起来，如果要选出文档中的一部分。例如想选中为<ul>的直接子元素的带有“a”类的列表项的话，用下面的代码。

```
1 ul >li[class="a"]{}
```

注意：建立一长列选中文档中，很明确的部分的选择器的时候，小心一些。这些CSS规则难以复用，因为让选择器在表示标记文本中的元素的相对位置上过于明确。建立简单的一个类，然后把它应用到有需求的元素上，经常会是更好的做法。不过话说回来，如果你需要让你的文档变换一下样式，但是没法编辑HTML（也许是因为它由CMS生成）的话，你的关系选择器的知识会派上用场。

---

---

# 盒模型

在 CSS 中，所有的元素都被一个个的“盒子（box）”包围着，理解这些“盒子”的基本原理，是我们使用CSS实现准确布局、处理元素排列的关键。

学习目标（Objective）：	学习盒模型的基本理论，了解盒装模型的工作原理 了解盒模型与替代模型的区别以及如何进行切换。
------------------	--

# 一、块级盒子（Block box） 和 内联盒子（Inline box）

CSS 中广泛地使用两种“盒子”—— **块级盒子 (block box)** 和 **内联盒子 (inline box)**。这两种盒子会在**页面流**（page flow）和**元素之间的关系**方面表现出不同的行为：

- 1、一个被定义成块级的（block）盒子会表现出以下行为：
- 盒子会在内联的方向上扩展并占据父容器在该方向上的所有可用空间，在绝大多数情况下意味着盒子会和父容器一样宽；
  - 每个盒子都会换行；
  - `width` 和 `height` 属性可以发挥作用；
  - 内边距（padding），外边距（margin） 和 边框（border） 会将其他元素从当前盒子周围“推开”

常见的默认块级元素block box	<h1>、<p>、<ul>、<ol>等
--------------------	---------------------

- 2、如果一个盒子对外显示为 `inline`，那么他的行为如下：
- 盒子不会产生换行；
  - `width` 和 `height` 属性将不起作用（不能指定width、height）；
  - 垂直方向的内边距、外边距以及边框会被应用，但是**不会**把其他处于 `inline` 状态的盒子推开；
  - 水平方向的内边距、外边距以及边框会被应用，而且也**会把**其他处于 `inline` 状态的盒子推开。

常见的默认内联元素inline box	a、span、img、strong、em等
---------------------	-----------------------

通过对盒子`display` 属性，来控制盒子的外部显示类型。

<code>display</code> 属性可以设置改变元素的内部和外部显示类型（块级还是内联），这将会改变它与布局中的其他元素的显示方式。	其值： block、 flex inline-f grid
---	---

### 3、补充: 内部和外部显示类型

解释下**内部**和**外部**显示类型。

box盒模型有一个外部显示类型，来决定盒子是块级还是内联。

box盒模型还有内部显示类型，决定了盒子内部元素是如何布局的。默认情况下是按照[正常文档流](#)布局，也意味着它们和其他块元素以及内联元素一样。

3.1如何更改内部显示类型：

通过使用类似 `flex` 的 `display` 属性值来更改内部显示类型。如果设置 `display: flex`，在一个元素上，外部显示类型是 `block`，但是内部显示类型修改为 `flex`。该盒子的所有直接子元素都会成为flex元素，会根据[弹性盒子 \(Flexbox\)](#) 规则进行布局。

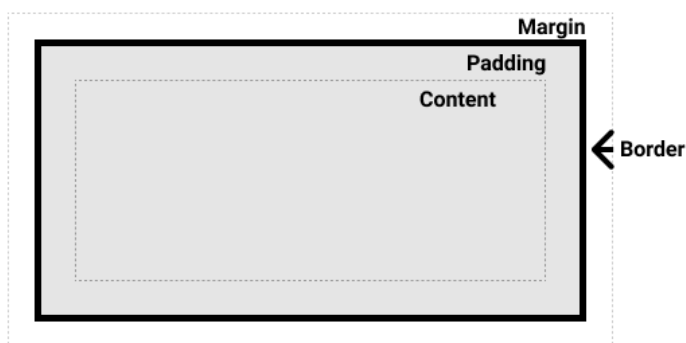
## 二、什么是CSS 盒模型？

完整的 CSS 盒模型应用于**块级盒子**，**内联盒子**只使用盒模型中定义的部分内容。模型定义了盒的每个部分——margin, border, padding, and content——合在一起就可以创建我们在页面上看到的内容。

### 1、盒模型的各个部分

CSS中组成一个块级盒子需要：

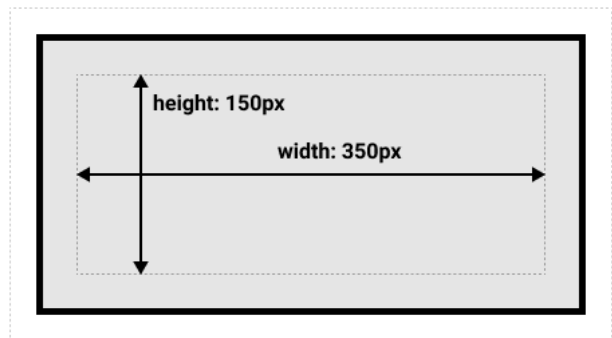
- **Content box:** 这个区域是用来显示内容，大小可以通过设置 `width` 和 `height`；
- **Padding box:** 包围在内容区域外部的空白区域；大小通过 `padding` 相关属性设置；
- **Border box:** 边框盒包裹内容和内边距。大小通过 `border` 相关属性设置；
- **Margin box:** 这是最外面的区域，是盒子和其他元素之间的空白区域。大小通过 `margin` 相关属性设置。



### 2、标准盒模型

在标准模型中，如果给盒设置 `width` 和 `height`，实际设置的是 `content box`。padding 和 border +设置的**宽、高=等于**整个盒子的大小。见下图。假设定义了 `width`, `height`, `margin`, `border`, and `padding`:

```
.box {  
  width: 350px;  
  height: 150px;  
  margin: 25px;  
  padding: 25px;  
  border: 5px solid black;  
}
```

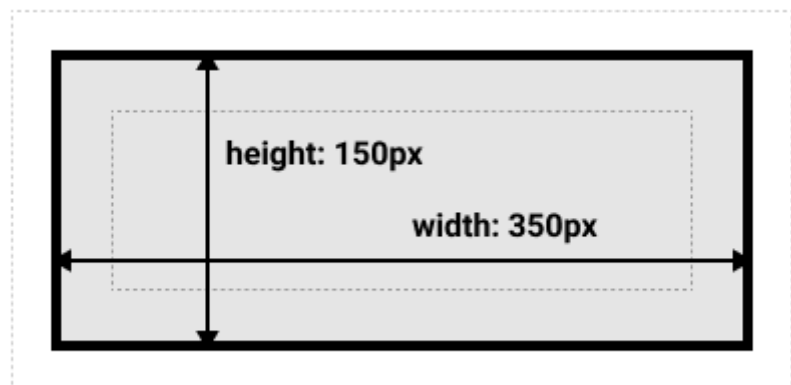


如果使用标准模型 **宽度** = 410px (350 + 25 + 25 + 5 + 5), **高度** = 210px (150 + 25 + 25 + 5 + 5), 即padding+ border + content box。

**注: margin** 不计入实际大小 —— 当然，它会影响盒子在页面所占空间，但是影响的是盒子外部空间。**盒子的范围到边框为止** —— 不会延伸到margin。

### 3、替代 (IE) 盒模型

你可能会认为盒子的大小还要加上**边框**和**内边距**，这样很麻烦，确实如此! 因为这个原因，css还有一个替代盒模型。使用该模型，所有**宽度**都是**可见宽度**，所以内容宽度是该宽度**减去**边框和填充部



分。使用上面相同的样式得到 (width = 350px, height = 150px)。

默认浏览器会使用**标准模型**。如需要使用替代模型，可以通过为其设置：

**box-sizing: border-box;**来实现。该属性告诉浏览器使用 `border-box` 来定义区域，从而设定您想要的大小。

```
1 .box{box-sizing: border-box;}
```



如果希望所有元素都使用替代模式，而且确实很常用，设置 `box-sizing` 在 `<html>` 元素上，然后设置如果你希望所有元素都使用替代模式，而且确实很常用，设置 `box-sizing` 在 `<html>` 元素上，然后设置所有元素继承该属性，正如下面的例子。如果想要深入理解，请看 [the CSS Tricks article on box-sizing](#)。，正如下面的例子。想深入理解，请看 [the CSS Tricks article on box-sizing](#)。

```
1 html{
2   box-sizing: border-box;
3 }
4 *, ::before, ::after{box-sizing: inherit;} /*所有元素继承该属性*/
```

**注：**一个有趣的历史记录——Internet Explorer默认使用替代盒模型，没有可用的机制来切换。（译者注：IE8+ 支持使用`box-sizing` 进行切换）。

---

## 4、外边距

外边距是盒子周围一圈看不到的空间。它会把其他元素从盒子旁边推开。外边距属性值可以为正也可以为负。设置负值会导致和其他内容重叠。无论使用标准模型还是替代模型，外边距总是在计算可见部分后额外添加。

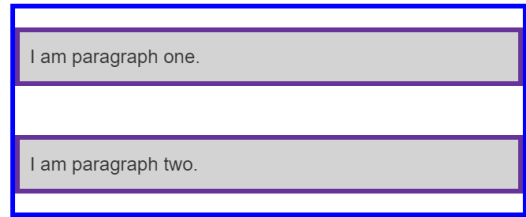
使用`margin`属性一次控制一个元素的所有边距，或者每边单独使用等价的普通属性控制：

- [margin-top](#)
- [margin-right](#)
- [margin-bottom](#)
- [margin-left](#)

### 4.1外边距折叠

如果有两个外边距相接的元素，这些外边距将合并为一个外边距，即最大的单个外边距的大小。

如下例，有两个段落。顶部段落的页 `margin-bottom`为50px。第二段的 `margin-top` 为30px。因为外边距折叠的概念，所以框之间的实际外边距是50px，而不是两个外边距的总和。



```
1 .one {  
2   margin-bottom: 50px;  
3 }  
4 .two {  
5   margin-top: 30px;  
6 }  
7 <div class="container">  
8   <p class="one">I am paragraph one.</p>  
9   <p class="two">I am paragraph two.</p>  
10 </div>
```

可以通过将第2段的 `margin-top` 设置为0来测试它。两个段落之间的可见边距不会改变——它保留了第一个段落 `margin-bottom` 设置的50像素。

有许多规则规定了什么时候外边距会折叠，什么时候不会折叠。相关信息，请参阅 [mastering margin collapsing](#)。现在首先要记住，外边距会折叠这个事情。如果你用外边距创建空间而没有得到你想要的效果，那这可能就是这个原因。

## 5、边框

边框：是在边距和填充框之间绘制的。在标准的盒模型中，边框的大小将添加到框的宽度和高度。在替代盒模型中，边框的大小会使内容框更小，因为它会占用一些可用的宽度和高度。

5.1使用 `border` 属性一次设置所有四个边框的宽度、颜色和样式。如果分别设置每边的宽度、颜色和样式，可以使用：

- `border-top`
- `border-right`
- `border-bottom`
- `border-left`

5.2设置所有边的颜色、样式或宽度，请使用以下属性：

- [border-width](#)
- [border-style](#)
- [border-color](#)

5.3设置单边的颜色、样式或宽度，可以使用最细粒度的普通属性之一：

• <a href="#">border-top-width</a>	• <a href="#">border-top-style</a>	• <a href="#">border-top-color</a>
• <a href="#">border-right-width</a>	• <a href="#">border-right-style</a>	• <a href="#">border-right-color</a>
• <a href="#">border-bottom-width</a>	• <a href="#">border-bottom-style</a>	• <a href="#">border-bottom-color</a>
• <a href="#">border-left-width</a>	• <a href="#">border-left-style</a>	• <a href="#">border-left-color</a>

---

## 6、内边距

内边距位于边框和内容区域之间。与外边距不同，不能有负数量的内边距，所以值必须是>=0正的值。应用于元素的任何背景都将显示在**内边距后面**，内边距通常用于将内容推离边框。

使用[padding](#)简写属性控制元素所有边，或者每边单独使用等价的普通属性：

- [padding-top](#)
- [padding-right](#)
- [padding-bottom](#)
- [padding-left](#)

任何元素上的内边距都可以更改，并在其边界和元素内部的任何内容之间留出空间。

---

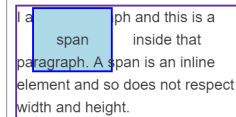
## 三、盒子模型和内联盒子

有些属性也可以应用于内联盒子，例如由<span>元素创建的内联盒子。对<span>应用宽度、高度、边距、边框和内边距。**宽度和高度被忽略**。外边距、内边距和边框是生效的，但它们不会改变其他内容与内联盒子的关系，因此内边距和边框会与段落中的其他单词重叠。

### 1、使用display: inline-block

**display: inline-block** 表示它在内联和块之间提供了一个中间状态。一旦给元素设置该属性会产生以下效果：

- 设置 `width` 和 `height` 属性会生效。它只会变得比其内容更大。
- `padding`, `margin`, 以及 `border` 会推开其他元素。
  - 不会切换到新行。



...ph and this is a  
span inside that  
paragraph. A span is an inline  
element and so does not respect  
width and height.

```
span {  
  margin: 20px;  
  padding: 20px;  
  
  background-color: lightblue;  
  border: 2px solid blue;  
}
```

```
<p>  
  I am a paragraph and this is a <span>span</span> inside that paragraph. A span is  
  an inline element and so does not respect width and height.  
</p>
```

## 背景与边框

使用CSS背景和边框来做一些具有一些创造性的事情。渐变、背景图像和圆角，背景和边框的巧妙运用是CSS中许多样式问题的答案。

目标：	学习如何在盒模型中使用背景和边框。
-----	-------------------

### 一、CSS的背景样式

CSS `background` 属性是许多普通背景属性的简写。**CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。**如果在样式表中发现了一个复杂的背景属性，可能会觉得难以理解，因为可以同时传入这么多值。

```
1 .box {  
2   background: linear-gradient(105deg, rgba(255,255,255,.2) 39%, rgba(51,5  
6,57,1) 96%) center center / 400px 200px no-repeat,  
3   url(big-star.png) center no-repeat, rebeccapurple;  
4 }
```

#### 1、背景颜色

`background-color` 属性定义任何元素的背景颜色。背景色扩展到元素的内容和内边距的下面。`background-color` 不能继承，其默认值是 `transparent`。`transparent` 有“透明”之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其祖先元素的背景才能可见。

```
1 .box {background-color: #567895;}  
2 h2 {background-color: black;  
3   color: white;}  
4 span{background-color: rgba(255,255,255,.5);}
```

## 2、背景图片

`background-image`属性允许在元素的背景中显示图像。有两个方框——一个是比方框大的背景图像，另一个是星星的小图像。

如果指定了背景图像，又指定了背景颜色，则图像将显示在颜色的顶部（即上面）。

```
1 background-image:url(图片路径);
```

### 2.1、控制背景平铺

<code>background-repeat</code>  属性用于控制图像的平铺行为。可用的值是：	<code>no-repeat</code> — 不重复。
	<code>repeat-x</code> — 水平重复
	<code>repeat-y</code> — 垂直重复
	<code>repeat</code> — 在两个方向重复

### 2.2、调整背景图像的大小

`background-size`属性可以设置长度或百分比值，来调整图像的大小以适应背景。如果图片比盒子大的话，多出来的部分会被剪裁。如果为了更好地显示图片的全部可以使用该属性。如果您的图像小于盒子，可以更改background-repeat的值来重复图像。

<code>background-size</code>  属性可以设置、长度或百分比值、关键字来调整图像的大小以适应背景。 <a href="https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-size">https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-size</a>	● <code>cover</code> — 浏览器将使图像足够大，使它完全覆盖了些图像可能会跳出盒子外。
	● <code>contain</code> — 浏览器将使图像的大小适合盒子内。：同，则可能在图像的任何一边或顶部和底部出现间隙
	一个值: 这个值指定图片的宽度，图片的高度隐式的
	两个值： 第一个值指定图片的宽度，第二个值指定
逗号分隔的多个值：设置多重背景	

### 2.3、背景图像定位

`background-position`属性允许您选择背景图像，显示在其应用到的盒子中的位置。它使用的坐标系中，框的左上角是(0,0)，框沿着水平(x)和垂直(y)轴定位。**注意：**默认的背景位置值是(0,0)。

<code>background-position</code> 背景图像定位属性	其值：百分比and长度值，
---	---------------

[https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building\\_blocks/Backgrounds\\_and\\_borders](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders)

也可以使用像top和right这样的关键字。

也可以混合使用关键字，长度值以及百分比

还可以使用4-value语法来指示到盒子的某些边的距离

**注意：** background-position是background-position-x和background-position-y的简写，它们

下面是等价的位置关键字：

单一关键字	等价的关键字
center	center center
top	top center 或 center top
bottom	bottom center 或 center bottom
right	right center 或 center right
left	left center 或 center left

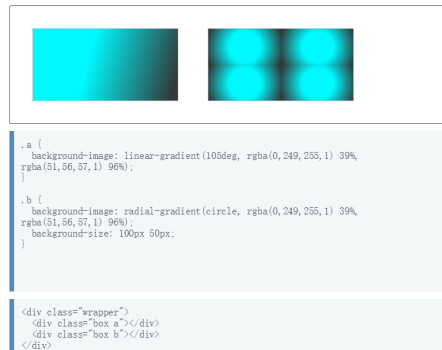
## 2.4、渐变背景

渐变——当它用于背景时——就像图像一样，也可以使用background-image属性设置。

您可以在MDN的<gradient>数据类型页面上了解更多关于渐变的不同类型以及使用它们可以做的事情。使用渐变的一个有趣方法是，使用web上可用的许多CSS渐变生成器之一，比如[这个](#)

<https://cssgradient.io/>您可以创建一个渐变，然后复制并粘贴生成它的源代码。

右边图中例子：在这两个盒子里，我们分别有一个**线性梯度**，它延伸到整个盒子上，还有一个**径向梯度**，它有一个固定的大小，因此会重复。



## 2.5、多个背景图像

在单个属性值中指定多个background-image值，用逗号分隔每个值。例如：可以指定多个背景图像，如此一来背景图像会互相重叠，背景将与最后列出的背景图像层，在堆栈的底部。**背景图像在列表中最先出现的在顶端。**

```
1 background-image: url(image1.png), url(image2.png), url(image3.png), url
  (image1.png);
2 /*其他 background-* 属性也可以有值，和background-image类似*/
```

```
3 background-repeat: no-repeat, repeat-x, repeat;
4 background-position: 10px 20px, top right;
5 background-attachment: fixed;
```

**注意：**渐变可以与常规的背景图像很好地混合在一起。

上述代码中，不同属性的每个值，将与其他属性中相同位置的值匹配，例如，上面的image1对应background-repeat值是no-repeat。但是，当不同的属性具有不同数量的值时（简单讲：有4个url图片，background-repeat或position值只有3个或者更少时），较少数量的值会循环在上面的例子中有四个背景图像，但，只有两个背景位置值时。前两个位置值将应用于前两个图像，然后它们将再次循环—image3将被赋予第一个位置值，image4将被赋予第二个位置值。



```
1 .box {
2   background-image: url(star.png), url(big-star.png);
3   background-repeat: no-repeat, repeat-x;
4   background-position: center 13px ;
5   background-size: ;
6   background-attachment: fixed;
7 }
8 <div class="wrapper">
9   <div class="box"></div>
10 </div>
```

## 2.6、背景的可访问性考虑

当你把文字放在背景图片或颜色上面时，应该注意你有足够的对比度，让文字对访客来说是清晰易读的。如果指定了一个图像，并且文本将被放置在该图像的顶部，您还应该指定一个background-color，以便在图像未加载时文本也足够清晰。

**注意：**屏幕阅读器不能解析背景图像，因此背景图片应该只是纯粹的装饰；任何重要的内容都应该是HTML页面的一部分，而不是包含在背景中。

## 2.6、背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。可以通过 [background-attachment](#) 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的（fixed），因此不会受到滚动的影响：



```

1 body {
2   background-image:url(/i/eg_bg_02.gif);
3   background-repeat:no-repeat;
4   background-attachment:fixed;
5 }

```

<b>background-attachment</b> 属性决定背景图像的位置是在视口内固定，或者随着包含它的区块滚动。	<b>fixed</b>	表示背景相对于视口固定。即使一个元素拥有滚动柄
	scroll	默认值。背景图像会随着页面其余部分的滚动而移动。
	inherit	规定应该从父元素继承 background-attachment 属性的

### 3、背景不透明度通过opacity属性设置

1)、opacity: 0.5; 属性参数的 " 不透明度 " 是以数字表示，从 0.0 至 1.0 都可以，完全透明是 0.0，完全不透明是 1.0。数字越大越不透明。

```
1 opacity:0.5;
```

2)、另一种方法在GRB 背景颜色属性值中设置，括号中最后一个值代表透明度，所谓RGBA颜色，就是RGB三原色加Alpha如下：

```
1 background:rgba(0,0,0,0.5)
```

RGB缺点：背景上面如果有文字的话，文字不透明。这种方法的兼容性不好，不兼容ie浏览器。

### 3、边框

边框如何影响盒子的大小，如何创造性地使用边界，当使用CSS向元素添加边框时，**border**简写属性，可为一个框的四个边设置，边框的颜色、宽度和样式。如下几种写法：

```

1 .box{border-top: 1px solid black;}/*可以设置一边*/
2 .box{border-width:2px;}/*简写等价与上面的写法*/
3   border-style:solid;
4   border-color:black;
5 /*也可以使用更加细粒度的属性如下：*/
6   border-top-width:
7   border-top-style:
8   border-top-color:

```



```
9  }
10
```

## 4、圆角

<code>border-radius</code> 属性,与方框的每个角相关的长边来实现方框的圆角	其值：两个长度或百分比作为值。第一个值定义水平半径，第二个值定义垂直半径。	在很多情况下，将只传递一个值，这两个值都将使用。
--	---------------------------------------	--------------------------

例如，要使一个盒子的四个角都有10px的圆角半径：

```
1  .box{border-radius: 10px;}
```

或使右上角的水平半径为1em，垂直半径为10%：

```
1  .box{border-top-right-radius:1em 10%;}
```

---

---

---

# 为文本添加样式（样式化文本）

了解如何设置字体、粗细、斜体、行还有字符间距、阴影以及文本的其他特征。我们将通过会在您的网页中应用自定义字体、样式化列表以及链接

## 基本文本和字体样式

目的:	了解在网页上设计文本所需的基本属性和技术。包括设置字体粗细（font weight）、字体缩写（font shorthand）、文本排列（text alignment）和其他的效果，还有行）。
-----	--

### 一、CSS中的文字样式涉及什么？

元素中的文本是布置在元素的**内容框中**。以内容区域的左上角作为起点 (或者是右上角，是在 RTL 语言的情况下)，一直延续到行的结束部分。一旦达到行的尽头，它就会进到下一行，然后，再接着下一行，直到所有内容都放入了盒子中。文本内容表现地像一些内联元素，被布置到相邻的行上，除非到达了行的尽头，否则不会换行，或者想强制地换行使用 `<br>` 元素。

用于样式文本的 CSS 属性通常可以分为两类：

- **字体样式:** 作用于字体的属性, 会直接应用到文本中, 比如使用哪种字体, 字体的大小是怎样的, 字体是粗体还是斜体, 等等。
- **文本布局风格:** 作用于文本的间距以及其他布局功能的属性, 比如, 允许操纵行与字之间的空间, 以及在内容框中, 文本如何对齐。

## 二、颜色

`color` 属性设置选中元素的前景内容的颜色, 接受任何合法的 [CSS 颜色单位](#)

```
1 p{color:;}
```

## 三、字体种类

`font-family` 属性, 浏览器指定一个字体 (或者一个字体的列表), 将字体应用到选中的元素上。浏览器只会把在, 当前机器上可用的字体应用到, 当前正在访问的网站上, 如果字体不可用, 就会用浏览器默认的字体代替。

```
1 font-family:arial;
```

### 1、网页安全字体

网络安全字体定义: 字体可用性, 几乎所有最常用的操作系统 (Windows, Mac, 最常见的Linux发行版, Android和iOS版本) 中都能找到, 可以大胆使用。下面的字体是网页安全的, 至少对于现在来说 (它们中的许多都非常流行, 这要感谢微软在90年代末和21世纪初期的倡议[Core fonts for the Web](#)):

字体名称	泛型
Arial	sans-serif
Courier New	monospace
Georgia	serif
Times New Roman	serif
Trebuchet MS	sans-serif
Verdana	sans-serif

**注意:** 在各种资源中, [cssfontstack.com](http://cssfontstack.com) 网站维护了一个可用在 Windows 和 Mac 操作系统上使用的网页安全字体的列表, 这可以帮助决策网站的安全性。

## 2、默认字体

5 个常用的字体名称: `serif`, `sans-serif`, `monospace`, `cursive`, 和 `fantasy`.

名称	定义	示例
<code>serif</code>	有衬线的字体 (衬线一词是指字体笔画末端的小装饰, 存在于某些印刷体字体中)	My big red elephant
<code>sans-serif</code>	没有衬线的字体。	My big red elephant
<code>monospace</code>	每个字符具有相同宽度的字体, 通常用于代码列表。	My big red elephant
<code>cursive</code>	用于模拟笔迹的字体, 具有流动的连接笔画。	My big red elephant
<code>fantasy</code>	用来装饰的字体	<b>My big red elephant</b>

## 3、字体栈

由于无法保证, 想在网页上使用的字体的可用性 (甚至一个网络字体可能由于某些原因而出错), 可以一个**字体栈 (font stack)**浏览器就有多种字体可以选择, `font-family`属性, 其值用逗号分离的字体名称组成。如下:

```
1 font-family: Arial, Times, Verdana, "Courier New"; /*浏览器依次检查, 都没有, 就使用浏览器默认字体*/
```

如果字体列表都没有可用字体, 那么段落将被赋予浏览器的默认衬线字体 - 通常是Time New Roman - 这对于 `sans-serif` 字体是不利的!

## 四、字体大小

字体大小 ( `font-size` 属性设置) 可以取大多数这些单位的值 (如百分比 [percentages](#)), 最常用的单位:

- `px` (像素): 将像素的值赋予给你的文本。这是一个绝对单位, 它导致了在任何情况下, 页面上的文本所计算出来的像素值都是一样的。
- `em`: 1em 等于我们设计的当前元素的父元素上设置的字体大小, 可以使用`em`调整任何东西的大小, 不只是文本。可以有一个单位全部都使用 `em` 的网站, 这样维护起来会很简单。
- `rem`: 效果和 `em` 差不多, 除了 1`rem` 等于 HTML 中的根元素的字体大小, (i.e. `<html>`), 而不是父元素。可以更容易计算字体大小, 但遗憾的是, `rem` 不支持 Internet Explorer 8 和以下的版本。如需要支持较老的浏览器, 可以坚持使用`em` 或 `px`, 或者是 [polyfill](#) 就像 [REM-unit-polyfill](#). (这个单位在“CSS的值和单位”一节也有讲解)。

元素的 `font-size` 属性是从该元素的父元素继承的。所以这一切都是从整个文档的根元素——`<html>` 开始，浏览器的 `font-size` 标准设置的值为 16px。比如 `<h1>` 元素有一个 2em 的默认值，所以它的最终大小值为 32px。1em=16px

更改嵌套元素的字体大小时，事情会变得棘手。比如，有一个 `<article>` 元素，然后设置它的 `font-size` 为 1.5em (通过计算，可以得到大小为 24px)，然后想让 `<article>` 元素中的段落获得一个计算值为 20px 的大小，那么你应该使用多少 em。用20除以24=0.8333333em

```
1 <!-- document base font-size is 16px -->
2 <article> <!-- If my font-size is 1.5em -->
3   <p>My paragraph</p> <!-- How do I compute to 20px font-size? --><!--将 e
m 的值设置为 20/24, 或者 0.8333333em.-->
4 </article>
```

一般将文档(document)的基础 `font-size` 设置为10px是个不错的主意，之后的计算会变得简单，所需要的 (r)em 值就是想得到的像素的值除以 10，而不是 16。样式表的指定区域列出所有`font-size`的规则集是一个好主意，这样它们就可以很容易被找到，如下：

```
1 html {
2   font-size: 10px;
3 }
4 h1 {
5   font-size: 2.6rem;
6 }
7 p {
8   font-size: 1.4rem;
9   color: red;
10  font-family: Helvetica, Arial, sans-serif;
11 }
```

五、字体样式，字体粗细，文本转换和文本装饰

4 种常用的属性来改变文本的样子：

属性	说明	可能的值
<code>font-style:</code>	用来打开和关闭文本 italic (斜体)。你很少会用到这个属性，除非你因为一些理由想将斜体文字关闭斜体状态。	<ul style="list-style-type: none"><li>● <code>normal</code> (斜体关闭)</li><li>● <code>italic</code>: 设置为斜体状态来模</li></ul>

		<ul style="list-style-type: none"> <li>● <code>oblique</code>: 本, 也就中。</li> </ul>
<code>font-weight</code> :	设置文字的粗体大小。这里有很多值可选 (比如 <code>-light</code> , <code>-normal</code> , <code>-bold</code> , <code>-extrabold</code> , <code>-black</code> , 等等), 不过事实上你很少会用到 <code>normal</code> 和 <code>bold</code> 以外的值:	<ul style="list-style-type: none"> <li>● <code>normal</code>: 度。</li> <li>● <code>lighter</code>: 其父元素值, 如果粒度控制</li> </ul>
<code>text-transform</code> :	设置要转换的字体。	<ul style="list-style-type: none"> <li>● <code>none</code>:</li> <li>● <code>upper</code>:</li> <li>● <code>lower</code>:</li> <li>● <code>capitalize</code>: 写。</li> <li>● <code>full-width</code>: 定宽度的字符和亚对齐。</li> </ul>
<code>text-decoration</code> :	设置/取消字体上的文本装饰 (主要使用此方法, 在设置链接时取消, 设置链接上的默认下划线。)	<ul style="list-style-type: none"> <li>● <code>none</code>:</li> <li>● <code>underline</code>:</li> <li>● <code>overline</code>:</li> <li>● <code>line-through</code>: over the</li> </ul>
<p><code>text-decoration</code> 可以接受多个值, 如果想要同时添加多个装饰值, 比如 <code>text-decoration: underline wavy red</code>; 注意 <code>text-decoration</code> 是缩写形式, 它由 <code>text-decoration-line</code>, <code>text-decoration-style</code> 和 <code>text-decoration-color</code> 组成。用这些属性值的组合来创建有趣的效果, 比如 <code>text-decoration: line-through red wavy</code>;</p>		

## 六、文字阴影

文本应用阴影, 使用 `text-shadow` 属性。这最多需要 4 个值, 如下例所示:

```
text-shadow: 4px 4px 5px red;
```

属性	值: 说明, <b>注意:</b> 正偏移值向右移动阴影, 负偏移值左右移动阴影, 例如		
<code>text-shadow</code> 文本应用阴影	1. 阴影与原始文本的水平偏移, 可以使用大多数的 CSS 单位 <code>length and size units</code> , 但是 <code>px</code> 是比较合适的。这个值必须指定。	2. 阴影与原始文本的垂直偏移; 效果基本上就像水平偏移, 除了它向上/向下移动阴影, 而不是左/右。这个值必须指定。	3. 模糊半径 - 更高的值意味着阴影分散得更广泛。如果不包含此值, 则默认为 0, 这意味着没有模糊。可以使用大多数的 CSS 单位 <code>length and size units</code> 。


多种阴影以逗号分隔的多个阴影值，将多个阴影应用于同一文本，例如：

```
1 text-shadow: 1px 2px 1px red;  
2 0px 4px 1px rgba(0,0,0,0.5),  
3 4px 4px 5px rgba(0,0,0,0.7),  
4 0px 0px 7px rgba(0,0,0,0.4);
```

## 七、文本布局

### 1、文本对齐

<a href="#">text-align</a>	属性用来控制文本如何和它所在的内容盒子对齐	● <code>left</code> : 左对齐文本。
		● <code>right</code> : 右对齐文本。
		● <code>center</code> : 居中文字
		<code>justify</code> : 使文本展开，改变单词之用，它可以看起来很可怕。特别是这个，你也应该考虑一起使用别的
<a href="#">hyphens</a>	属性 <code>hyphens</code> 告知浏览器在换行时如何使用连字符连接单词。可以完全阻止使用连字符，也可以控制浏览器什么时候使用，或者让浏览器决定什么时候使用。	<code>none</code> 换行时单词不会被打断，甚至在单行。
		<code>manual</code> 仅当单词中的字符暗示换行时
		<code>auto</code> 浏览器可以按照选择使用的任何会，如建议换行机会所述，应优先于自

注意：自动设置的行为取决于正确标记的语言，以便可以选择适当的连字符规则。您必须使用`lang HTML`

### 2、行高

`line-height` 属性设置文本每行之间的高，可以接受大多数单位。[length and size units](#)，也可以设置一个无单位的值，作为乘数，通常这种是比较好的做法。无单位的值乘以 `font-size` 来获得 `line-height`。当行与行之间拉开空间，正文文本通常看起来更好更容易阅读。推荐的行高大约是 1.5–2 (双倍间距。) 所以文本行高设置为字体高度的1.5倍，你可以使用这个：

```
1 line-height: 2;
```

### 3、字母和单词间距

--	--

<code>letter-spacing</code> 属性	设置你的文本中的字母与字母之间的间距
<code>word-spacing</code> 属性	设置单词与单词之间的间距

不会经常使用它们，但是可能可以通过它们，来获得一个特定的外观，或者让较为密集的文字更加可读。它们可以接受大多数单位 [length and size units](#)。

#### 4、Font 样式:

- `font-variant`: 在小型大写字母和普通文本选项之间切换。
- `font-kerning`: 开启或关闭字体间距选项。
- `font-feature-settings`: 开启或关闭不同的 [OpenType](#) 字体特性。
- `font-variant-alternates`: 控制给定的自定义字体的替代字形的使用。
- `font-variant-caps`: 控制大写字母替代字形的使用。
- `font-variant-east-asian`: 控制东亚文字替代字形的使用, 像日语和汉语。
- `font-variant-ligatures`: 控制文本中使用的连写和上下文形式。
- `font-variant-numeric`: 控制数字, 分式和序标的替代字形的使用。
- `font-variant-position`: 控制位于上标或下标处, 字号更小的替代字形的使用。
- `font-size-adjust`: 独立于字体的实际大小尺寸, 调整其可视大小尺寸。
- `font-stretch`: 在给定字体的可选拉伸版本中切换。
- `text-decoration-position`: 指定下划线的排版位置, 通过使用 `text-decoration-line` 属性的 `underline` 值。
- `text-rendering`: 尝试执行一些文本渲染优化。

#### 4、文本布局样式:

- `text-indent`: 指定文本内容的第一行前面应该留出多少的水平空间。
- `text-overflow`: 定义如何向用户表示存在被隐藏的溢出内容。
- `white-space`: 定义如何处理元素内部的空白和换行。
- `word-break`: 指定是否能在单词内部换行。
- `direction`: 定义文本的方向 (这取决于语言, 并且通常最好让HTML来处理这部分, 因为它是和文本内容相关联的。)
- `hyphens`: 为支持的语言开启或关闭连字符。



- `line-break`: 对东亚语言采用更强或更弱的换行规则。
- `text-align-last`: 定义一个块或行的最后一行，恰好位于一个强制换行前时，如何对齐。
- `text-orientation`: 定义行内文本的方向。
- `word-wrap`: 指定浏览器是否可以在单词内换行以避免超出范围。
- `writing-mode`: 定义文本行布局为水平还是垂直，以及后继文本流的方向。

## 八、Font 简写

许多字体的属性也可以通过 `font` 的简写方式来设置。这些是按照以下顺序来写的：`font-style`, `font-variant`, `font-weight`, `font-stretch`, `font-size`, `line-height`, and `font-family`。

```
1 font:italic normal bold 3em/1.5 Helvetica, Arial, sans-serif;
```

在所有这些属性中，只有 `font-size` 和 `font-family` 是一定要指定的。

## 样式列表

[List列表](#) 大体上和其他文本一样。

目标:	熟悉与列表相关的样式和最佳实践
-----	-----------------

### 一、处理列表间距

### 二、列表特定样式

列表的一般间距，一些列表具有的特定属性。从三个属性开始了解，这三个属性可以在 `<ul>` 或 `<ol>` 元素上设置：

- `list-style-type`：设置用于列表的项目符号的类型，例如无序列表的方形或圆形项目符号，或有序列表的数字，字母或罗马数字。
- `list-style-position`：设置在每个项目开始之前，项目符号是出现在列表项内，还是出现在其外。
- `list-style-image`：允许您为项目符号使用自定义图片，而不是简单的方形或圆形。这个属性在控制项目符号的**位置**，**大小**等方面是有限的。您最好使用 `background` 系列属性。



## 1、项目符号位置

`list-style-position` 规定列表中列表项目标记的位置：inside项目符号出现在列表项内，默认值为 outside项目符号位于列表项之外。

```
1 list-style-position: inside/outside;
```

## 2、list-style 速记

如下三种属性的速记属性 可以用`list-style` 的简写方法：

```
1 ul {
2   list-style-type: square;
3   list-style-image: url(example.png);
4   list-style-position: inside;
5 }
6 list-style: square url(example.png) inside; /*简写卸载一行中*/
```

属性值可以任意顺序排列，可以设置一个，两个或者三个值（该属性的默认值为 disc, none, outside），如果指定了 type 和 image，如果由于某种原因导致图像无法加载，则 type 将用作回退。

## 三、管理列表计数

有时，您可能想在有序列表上进行不同的计数方式。例如：从1以外的数字开始，或向后倒数，或者按步或多于1计数。

### 1、start 属性

该属性允许从1 以外的数字开始计数。示例如下：

```
1 <ol start="4">
2   <li>Toast pitta, leave to cool, then slice down the edge.</li>
3   <li>Fry the halloumi in a shallow, non-stick pan, until browned on both sides.</li>
4   <li>Wash and chop the salad.</li>
5   <li>Fill pitta with salad, humous, and fried halloumi.</li>
6 </ol>
```

	属性	值描述
<ol> </ol>	start	一个整数值属性，指定了列表编号的起始值。此属性型 type 属性可能指定为了罗马数字编号等其他类型的母 "d" 或者罗马数字 "iv" 开始，都应当使用 start="4
	type	设置编号的类型： ● a 表示小写英文字母编号 ● A 表示大写英文字母编号 ● i 表示小写罗马数字编号

		● <b>I</b> 表示大写罗马数字编号 ● <b>1</b> 表示数字编号（默认）
	reversed	此布尔值属性指定列表中的条目，是否是倒序排列的

## 2、value 属性

该属性允许设置列表项指定数值，示例如下：

```
1 <ol>
2   <li value="2">Toast pitta, leave to cool, then slice down the edge.</li>
3   <li value="4">Fry the halloumi in a shallow, non-stick pan, until browned on both sides.</li>
4   <li value="6">Wash and chop the salad.</li>
5   <li value="8">Fill pitta with salad, humous, and fried halloumi.</li>
6 </ol>
```

# 样式化链接

目的:	学习如何将样式应用到链接状态，以及如何使用链实现常见的 UI 功能，比如导航菜单。理解利用伪有效地建立链接状态是很重要的，以及如何链接添加样式来实现常用的功能
-----	---

## 一、链接状态

- **Link (没有访问过的)**: 这是链接的默认状态，当它没有处在其他状态的时候，它可以使用 `:link` 伪类来应用样式。
- **Visited**: 这个链接已经被访问过了(存在于浏览器的历史纪录), 它可以使用 `:visited` 伪类来应用样式。
- **Hover**: 当用户的鼠标光标刚好停留在这个链接，它可以使用 `:hover` 伪类来应用样式。
- **Focus**: 一个链接当它被选中的时候 (比如通过键盘的 Tab 移动到这个链接的时候，或者使用编程的方法来选中这个链接 `HTMLElement.focus()`) 它可以使用 `:focus` 伪类来应用样式。
- **Active**: 一个链接当它被激活的时候 (比如被点击的时候)，它可以使用 `:active` 伪类来应用样式。

## 1.1 默认的风格

# A link to the Mozilla homepage

观察上面链接默认样式，有如下特性：

- 链接具有下划线。
- 未访问过的 (Unvisited) 的链接是蓝色的。
- 访问过的 (Visited) 的链接是紫色的。
- 悬停 (Hover) 在一个链接的时候鼠标的光标会变成一个小手的图标。
- 选中 (Focus) 链接的时候，链接周围会有一个轮廓，你应该可以按 tab 来选中这个页面的链接 (在 Mac 上，你可能需要使用 *Full Keyboard Access: All controls* 选项，然后再按下 `Ctrl + F7`，这样就可以起作用)
- 激活 (Active) 链接的时候会变成红色 (当你点击链接时，请尝试按住鼠标按钮。)

**链接的样式不应该与用户预期的相差太大，应该至少：**

- 为链接使用下划线，但是不要在其他内容上也用下划线，以作区分。如果不想要带有下划线的链接，那至少要用其他方法来**高亮突出链接**。
- 当用户悬停或选择 (hover 或者 focused) 时，使链接有相应的变化，并且在链接被激活(active) 的时候，变化会有一些不同。可以使用以下 CSS 属性关闭/更改默认样式：
  - `color` 文字的颜色
  - `cursor` 鼠标光标的样式，你不应该把这个关掉，除非你有非常好的理由。
  - `outline` 文字的轮廓 (**轮廓有点像边框**，唯一的区别**是边框占用了盒模型的空间，而轮廓没有**；它只是设置在背景图片的顶部)。outline 是一个有用的辅助功能，所以在把它关掉之前考虑清楚；你至少应该将悬停 (hover) 状态的样式同时应用到选中 (focus) 状态上

**注意：**你不仅仅只限于上述属性来把样式应用到你的链接上，你可以用任何你喜欢的属性，就是不要搞得太疯狂！

## 1.2 将样式应用到一些链接

如下这几个规则的顺序是有意义的，因为链接的样式是建立在另一个样式之上的，比如，第一个规则的样式也会后面的规则中生效，一个链接被激活 (activated) 的时候，它也是处于悬停 (hover) 状态的。如果搞错了顺序，就不会产生正确的效果。要记住这个顺序，可以尝试这样帮助记忆：

**LoVe Fears HAte.**

```
1  body{
2    width: 300px;
3    margin: 0 auto;
4    font-size: 1.2rem;
5    font-family: sans-serif;
6  }
7  p{
8    line-height: 1.4;
9  }
10 a{
11   outline: #00FF00 dotted thick; /*表示“轮廓”有点像边框，唯一的区别是边框占用了盒模型的空间，而轮廓没有； 它只是设置在背景图片的顶部*/
12   padding: 2px 1px 0;
13 }
14 /*正常状态下链接，未访问过*/
15 a:link{
16   color: #265301;
17 }
18 /*点击过链接*/
19 a:visited{
20   color: #437A16;
21 }
22 /*键盘tab选中样式*/
23 a:focus{
24   border-bottom: 1px solid;
25   background: #BAE498;
26 }
27 /*鼠标悬停*/
28 a:hover{
29   border-bottom: 1px solid;
30   background: #CDFEAA;
31 }
32 /*激活*/
33 a:active{
```

```
34 background: #265301;
35 color: #CDFEAA;
36 }
```



- 第一和第二条规则和本次讨论关系不大。
- 第三个规则使用了 `a` 选择器，取消了默认的文本下划线和链接被选中 (focus) 时的轮廓 (outline) (不同浏览器的默认行为可能不同)，并为每个链接添加了少量的内边距 (padding)，所有这一切将在之后变得明确。
- 接着，我们使用 `a:link` 和 `a:visited` 选择器来设置未访问 (unvisited) 链接和访问过 (visited) 的链接的一点颜色上的变化，然后就能分辨开来了。
- 下面两条规则使用 `a:focus` 和 `a:hover` 来设置选中 (focus) 和悬停 (hover) 的链接为不同的背景颜色，再加上一个下划线，使链接更加突出。这里有两点需要注意：
  - 下划线是使用 `border-bottom` 创造的, 而不是 `text-decoration`，有一些人喜欢这样，因为前者比后者有更好的样式选项，并且绘制的位置会稍微低一点，所以不会穿过字母 (比如 字母 g 和 y 底部)。
  - `border-bottom` 的值被设置为 `1px solid`，没有指定颜色。这样做可以使边框采用和元素文本一样的颜色，这在这样的情况下是很有用的，因为链接的每种状态下，文本是不同的颜色。
- 最后，`a:active` 用来给链接一个不同的配色方案，当链接被激活 (activated) 时，让链接被激活的时候更加明显。

### 1.3、在链接中包含图标

常见的做法是在链接中包含**图标**，使链接提供更多关于链接指向的内容的信息。

例如：一个外部链接 (链接指向的不是本站，而是外部站点)。这样的图标通常看起来像一个指向盒子的小箭头，比如，我们会使用[icons8.com](https://icons8.com)上的这个优秀

[的范例](#)。

For more information on the weather, visit our [weather page](#), look at [weather on Wikipedia](#), or check out [weather on Extreme Science](#).

```
1 a[href*="http"] {  
2   background: url('https://mdn.mozillademos.org/files/12982/external-link-  
52.png') no-repeat 100% 0;  
3   background-size: 16px 16px;  
4   padding-right: 19px;  
5 }  
6 <p>For more information on the weather, visit our <a href="weather.html">  
weather page</a>,  
7 look at <a href="https://en.wikipedia.org/wiki/Weather">weather on Wikipe  
dia</a>, or check  
8 out <a href="http://www.extremescience.com/weather.htm">weather on Extrem  
e Science</a>.</p>
```