链滴

Python3 网络爬虫实战：16、Web 网页基础

作者：zhaolixiang
原文链接：https://ld246.com/article/1567138614609
来源网站：链滴

我们平时用浏览器访问网站的时候，一个个站点形形色色，页面也各不相同，但有没有想过它是何才能呈现出这个样子的？
那么本节我们就来了解一下网页的基本组成、结构、节点等内容。
<h2 id="1--网页的组成">1. 网页的组成</h2>
网页可以分为三大部分，HTML、CSS、JavaScript，我们把网页比作一个人的话，HTML 相当骨架，JavaScript 则相当于肌肉，CSS 则相当于皮肤，三者结合起来才能形成一个完善的网页，下面们分别来介绍一下三部分的功能。
<h2 id="HTML">HTML</h2>
HTML 是用来描述网页的一种语言，其全称叫做 Hyper Text Markup Language，即超文本标语言。网页包括文字、按钮、图片、视频等各种复杂的元素，其基础架构就是 HTML。不同类型的文通过不同类型的标签来表示，如图片用 img 标签表示，视频用 video 标签来表示，段落用 p 标签来示，它们之间的布局又常通过布局标签 div 嵌套组合而成，各种标签通过不同的排列和嵌套才形成了页的框架。
我们在 Chrome 浏览器中打开百度，右键单击审查元素或按 F12 打开开发者模式，切换到 Elements 选项卡即可看到网页的源代码，如图 2-10 所示：


图 2-10 源代码
这就是 HTML，整个网页就是由各种不同的标签嵌套组合而成的，这些不同标签定义的节点元素相互套和组合形成了复杂的层次关系，就形成了网页的架构。
<h2 id="CSS">CSS</h2>
HTML 定义了网页的结构，但是只有 HTML 页面的布局会不美观，可能只是简单的节点元素的列，那么为了让网页看起来更好看一点，在这里就借助于 CSS。
CSS，全称叫做 Cascading Style Sheets，即层叠样式表。“层叠”是指当在 HTML 中引用了数个式文件，并且样式发生冲突时，浏览器能依据层叠顺序处理。“样式”指网页中文字大小、颜色、元间距、排列等格式。
CSS 是目前唯一的网页页面排版样式标准，有了它的帮助，页面才会变得更为美观。
在上图的右侧即为 CSS，例如：
<pre><code class="highlight-chroma">#head_wrapper.s-ps-islite .s-p-top {
  &nsp; position: absolute;
  &nsp; bottom: 40px;
  &nsp; width: 100%;
  &nsp; height: 181px;
}
</code></pre>
这就是一个 CSS 样式，在大括号前面是一个 CSS 选择器，此选择器的意思是选中 id 为 head_wrpper 且 class 为 s-ps-islite 内部的 class 为 s-p-top 的元素。大括号内部写的就是一条条样式规则例如 position 指定了这个元素的布局方式为绝对布局，bottom 指定元素的下边距为 40 像素，width 指定了宽度为 100% 占满父元素，height 则指定了元素的高度。也就是说我们将一些位置、宽度、度等样式配置统一写成这样的形式，大括号括起来，然后开头再加上一个 CSS 选择器，就代表这一样式对 CSS 选择器选中的元素生效，这样元素就会根据此样式来展示了。
所以在网页中，一般会统一定义整个网页的样式规则，写入到 CSS 文件，其后缀名为 css，在 HTML 中只需要用 link 标签即可引入写好的 CSS 文件，这样整个页面就会变得美观优雅。
<h2 id="JavaScript">JavaScript</h2>
JavaScript，简称为 JS，是一种脚本语言，HTML 和 CSS 配合使用，提供给用户的只是一种静的信息，缺少交互性。我们在网页里可能会看到一些交互和动画效果，如下载进度条、提示框、轮播等，这通常就是 JavaScript 的功劳。它的出现使得用户与信息之间不只是一种浏览与显示的关系，而实现了一种实时、动态、交互的页面功能。
JavaScript 通常也是以单独的文件形式加载的，后缀名为 js，在 HTML 中通过 script 标签即可引入

例如：
<pre><code class="highlight-chroma"><script src="jquery-2.1.0.js"></script>
</code></pre>
因此综上所属，HTML 定义了网页的内容和结构，CSS 描述了网页的布局，JavaScript 定义了页的行为。
这就是网页的三大基本组成。
<h2 id="2--网页的结构">2. 网页的结构</h2>
我们首先用一个例子来感受一下 HTML 的基本结构。新建一个文本文件，名称可以自取，后缀为 html，内容如下：
<pre><code class="highlight-chroma"><!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>This is a Demo</title>
  </head>
  <body>
    <div id="container">
      <div class="wrapper">
        <h2 class="title">Hello World</h2>
        <p class="text">Hello, this is a paragraph.</p>
      </div>

    </div>
  </body>
</html>
</code></pre>
这就是一个最简单的 HTML 实例，开头是 DOCTYPE 定义了文档类型，其次最外层是 html 标，最后还有对应的结尾代表标签闭合，其内部是 head 标签和 body 标签，分别代表网页头和网页体它们也分别需要尾标签表示闭合。head 标签内定义了一些页面的配置和引用，如：
<pre><code class="highlight-chroma"><meta charset="UTF-8">
</code></pre>
它指定了网页的编码为 UTF-8。
title 标签则定义了网页的标题，会显示在网页的选项卡中，不会显示在网页的正文中。body 标签内是在网页正文中显示的内容，div 标签定义了网页中的区块，它的 id 是 container，这是一个非常常的属性，且 id 的内容在网页中是唯一的，我们可以通过 id 来取到这个区块。然后在此区块内又有一个 div 标签，它的 class 为 wrapper，这也是一个非常常用的属性，经常与 CSS 配合使用来设定样式。后此区块内部又有一个 h2 标签，这代表一个二级标题，另外还有一个 p 标签，这代表一个段落，它二者内部直接写入相应的内容即可在网页重呈现出来，它们也有各自的 class 属性。
我们将代码保存之后在浏览器中打开该文件，可以看到如下内容，如图 2-11 所示：

图 2-11 运行结果
可以看到在选项卡上显示了 This is a Demo 字样，这是我们在 head 里面的 title 里定义的文字，它示在了网页选项卡里。而网页正文是 body 标签内部定义的各个元素生成的，图中可以看到网页中显

了二级标题和段落。
如上实例便是网页的一般结构，一个网页标准形式都是 html 标签内嵌套 head 和 body 标签，head 内定义网页的配置和引用，body 内定义网页的正文。
<h2 id="3--节点及节点关系">3. 节点及节点关系</h2>
在 HTML 中，所有标签定义的内容都是节点，它们构成了一个 HTML DOM 树。
我们先看下什么是 DOM，DOM 是 W3C（万维网联盟）的标准。
DOM，英文全称 Document Object Model，即文档对象模型。它定义了访问 HTML 和 XML 文档标准：
<blockquote>
W3C 文档对象模型 （DOM） 是中立于平台和语言的接口，它允许程序和脚本动态地访问和更文档的内容、结构和样式。
</blockquote>
W3C DOM 标准被分为 3 个不同的部分：

核心 DOM - 针对任何结构化文档的标准模型
XML DOM - 针对 XML 文档的标准模型
HTML DOM - 针对 HTML 文档的标准模型

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

整个文档是一个文档节点
每个 HTML 元素是元素节点
HTML 元素内的文本是文本节点
每个 HTML 属性是属性节点注释是
注释节点 HTML

DOM 将 HTML 文档视作树结构，这种结构被称为节点树，如图 2-12 所示：

图 2-12 节点树
通过 HTML DOM，树中的所有节点均可通过 JavaScript 进行访问，所有 HTML 节点元素均可被修，也可以被创建或删除。
节点树中的节点彼此拥有层级关系。我们常用 parent（父）、child（子）和 sibling（兄弟）等术语于描述这些关系。父节点拥有子节点，同级的子节点被称为兄弟节点。
在节点树中，顶端节点被称为根（root），除了根节点之外每个节点都有父节点，同时可拥有任意数的子节点或兄弟节点。
图 2-13 展示了节点树以及节点之间的关系：

图 2-13 节点树及节点关系
本段参考 W3SCHOOL，链接：http://www.w3school.com.cn/htmldom/dom_nodes.asp。
<h2 id="4--选择器">4. 选择器</h2>
我们知道网页由一个个节点组成，CSS 选择器会根据不同的节点设置不同的样式规则，那么我们样来定义是哪些节点呢？
在 CSS 中是使用了 CSS 选择器来定位节点的，例如上例中有个 div 节点的 id 为 container，那么我就可以用 CSS 选择器表示为 #container，# 开头代表选择 id，其后紧跟 id 的名称。另外如果我们选择 class 为 wrapper 的节点，便可以使用 .wrapper，. 开头代表选择 class，其后紧跟 class 的名。另外还有一种选择方式是根据标签名筛选，例如我们想选择二级标题，直接用 h2 即可选择。如上最常用的三种选择表示，分别是根据 id、class、标签名筛选，请牢记它们的写法。
另外 CSS 选择器还支持嵌套选择，各个选择器之间加上空格分隔开便可以代表嵌套关系，如 #container .wrapper p 则代表选择 id 为 container 内部的 class 为 wrapper 内部的 p 节点。另外如果不加格则代表并列关系，如 div#container .wrapper p.text 代表选择 id 为 container 的 div 节点内部的 c

ass 为 wrapper 节点内部的 class 为 text 的 p 节点。这就是 CSS 选择器，其筛选功能还是非常强大。
另外 CSS 选择器还有一些其他的语法规则，在这里整理如下：</p>
<table>
<thead>
<tr>
<th>选择器</th>
<th>例子</th>
<th>例子描述</th>
</tr>
</thead>
<tbody>
<tr>
<td>.class</td>
<td>.intro</td>
<td>选择 class="intro" 的所有节点。</td>
</tr>
<tr>
<td>#id</td>
<td>#firstname</td>
<td>选择 id="firstname" 的所有节点。</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>选择所有节点。</td>
</tr>
<tr>
<td>element</td>
<td>p</td>
<td>选择所有 p 节点。</td>
</tr>
<tr>
<td>element,element</td>
<td>div,p</td>
<td>选择所有 div 节点和所有 p 节点。</td>
</tr>
<tr>
<td>element element</td>
<td>div p</td>
<td>选择 div 节点内部的所有 p 节点。</td>
</tr>
<tr>
<td>element>element</td>
<td>div>p</td>
<td>选择父节点为 div 节点的所有 p 节点。</td>
</tr>
<tr>
<td>element+element</td>
<td>div+p</td>
<td>选择紧接在 div 节点之后的所有 p 节点。</td>
</tr>
<tr>
<td>[attribute]</td>

<td>[target]</td>
<td>选择带有 target 属性所有节点。 </td>
</tr>
<tr>
<td>[attribute=value]</td>
<td>[target=blank]</td>
<td>选择 target="blank" 的所有节点。 </td>
</tr>
<tr>
<td>[attribute~=value]</td>
<td>[title~=flower]</td>
<td>选择 title 属性包含单词 "flower" 的所有节点。 </td>
</tr>
<tr>
<td>:link</td>
<td>a:link</td>
<td>选择所有未被访问的链接。 </td>
</tr>
<tr>
<td>:visited</td>
<td>a:visited</td>
<td>选择所有已被访问的链接。 </td>
</tr>
<tr>
<td>:active</td>
<td>a:active</td>
<td>选择活动链接。 </td>
</tr>
<tr>
<td>:hover</td>
<td>a:hover</td>
<td>选择鼠标指针位于其上的链接。 </td>
</tr>
<tr>
<td>:focus</td>
<td>input:focus</td>
<td>选择获得焦点的 input 节点。 </td>
</tr>
<tr>
<td>:first-letter</td>
<td>p:first-letter</td>
<td>选择每个 p 节点的首字母。 </td>
</tr>
<tr>
<td>:first-line</td>
<td>p:first-line</td>
<td>选择每个 p 节点的首行。 </td>
</tr>
<tr>
<td>:first-child</td>
<td>p:first-child</td>
<td>选择属于父节点的第一个子节点的每个 p 节点。 </td>
</tr>
<tr>

<td>:before</td>
<td>p:before</td>
<td>在每个 p 节点的内容之前插入内容。</td>
</tr>
<tr>
<td>:after</td>
<td>p:after</td>
<td>在每个 p 节点的内容之后插入内容。</td>
</tr>
<tr>
<td>:lang(language)</td>
<td>p:lang</td>
<td>选择带有以 "it" 开头的 lang 属性值的每个 p 节点。</td>
</tr>
<tr>
<td>element1~element2</td>
<td>p~ul</td>
<td>选择前面有 p 节点的每个 ul 节点。</td>
</tr>
<tr>
<td>[attribute^=value]</td>
<td>a[src^="https"]</td>
<td>选择其 src 属性值以 "https" 开头的每个 a 节点。</td>
</tr>
<tr>
<td>[attribute$=value]</td>
<td>a[src$=".pdf"]</td>
<td>选择其 src 属性以 ".pdf" 结尾的所有 a 节点。</td>
</tr>
<tr>
<td>[attribute*=value]</td>
<td>a[src*="abc"]</td>
<td>选择其 src 属性中包含 "abc" 子串的每个 a 节点。</td>
</tr>
<tr>
<td>:first-of-type</td>
<td>p:first-of-type</td>
<td>选择属于其父节点的首个 p 节点的每个 p 节点。</td>
</tr>
<tr>
<td>:last-of-type</td>
<td>p:last-of-type</td>
<td>选择属于其父节点的最后 p 节点的每个 p 节点。</td>
</tr>
<tr>
<td>:only-of-type</td>
<td>p:only-of-type</td>
<td>选择属于其父节点唯一的 p 节点的每个 p 节点。</td>
</tr>
<tr>
<td>:only-child</td>
<td>p:only-child</td>
<td>选择属于其父节点的唯一子节点的每个 p 节点。</td>
</tr>

<tr>
<td>:nth-child(n)</td>
<td>p:nth-child</td>
<td>选择属于其父节点的第二个子节点的每个 p 节点。</td>
</tr>
<tr>
<td>:nth-last-child(n)</td>
<td>p:nth-last-child</td>
<td>同上，从最后一个子节点开始计数。</td>
</tr>
<tr>
<td>:nth-of-type(n)</td>
<td>p:nth-of-type</td>
<td>选择属于其父节点第二个 p 节点的每个 p 节点。</td>
</tr>
<tr>
<td>:nth-last-of-type(n)</td>
<td>p:nth-last-of-type</td>
<td>同上，但是从最后一个子节点开始计数。</td>
</tr>
<tr>
<td>:last-child</td>
<td>p:last-child</td>
<td>选择属于其父节点最后一个子节点每个 p 节点。</td>
</tr>
<tr>
<td>:root</td>
<td>:root</td>
<td>选择文档的根节点。</td>
</tr>
<tr>
<td>:empty</td>
<td>p:empty</td>
<td>选择没有子节点的每个 p 节点（包括文本节点）。</td>
</tr>
<tr>
<td>:target</td>
<td>#news:target</td>
<td>选择当前活动的 #news 节点。</td>
</tr>
<tr>
<td>:enabled</td>
<td>input:enabled</td>
<td>选择每个启用的 input 节点。</td>
</tr>
<tr>
<td>:disabled</td>
<td>input:disabled</td>
<td>选择每个禁用的 input 节点</td>
</tr>
<tr>
<td>:checked</td>
<td>input:checked</td>
<td>选择每个被选中的 input 节点。</td>

```
</tr>
<tr>
<td>:not(selector)</td>
<td>p:not</td>
<td>选择非 p 节点的每个节点。</td>
</tr>
<tr>
<td>::selection</td>
<td>::selection</td>
<td>选择被用户选取的节点部分。</td>
</tr>
<tr>
<td>另外还有一种比较常用的选择器是 XPath，此种选择方式在后文会详细介绍。</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<h2 id="5--结语">5. 结语</h2>
<p>本节介绍了网页的基本结构和节点关系，了解了这些内容我们才有更加清晰的思路去解析和提取页内容。</p>
```