HTML，CSS，JavaScript 是 Web 开发的基础，任何一个网页都离不开 HTML, CSS。HTML 标签组构建了网页的基本框架，而 CSS 则使得网页布局更加合理美观

**HTML 有很多版本**，其中最新的 HTML5 目前最新也最为流行的版本。我们看到的任何一个网页，查看其源码时，能看到类似于 <div></div> 这样的标签，HTML 就由这些标签组成，学习 HTML5 的过程学习各种各种标签，了解其用途和各种属性。HTML 本身非常简单，任何一个网页都是很好的学习例子，所以多看网页源码，多练习就可以快速掌握了。

HTML5 页面有各种标签组成，基本的结构如在前文中已经展示过，现在先介绍下 index.html 页面基本结构:

* <!DOCTYPE HTML> 指示了页面的文档类型是 HTML5 文档；
* <head></head> 定义了文档的头部，描述了这个页面的一些信息，比如标题，编码，引用的 CSS 等信息；
* <body></body> 定义了文档的主体，包含文档的所有内容；

可以看到 HTML5 就是由各类标签组成的，这些标签大部分有一个起始标签和闭合标签，比如 <div> 是起始标签，而 </div> 是闭合标签。起始标签和闭合标签形成了一个容器，又可以包含其他各种标签。有一些标签，可以没有闭合标签，比如上面出现的 <link> 标签。完整的标签列表，可以参考 [https://developer.mozilla.org/en/docs/Web/HTML/Element](https://developer.mozilla.org/en/docs/Web/HTML/Element" \t "https://www.shiyanlou.com/courses/_blank) 文档。

HTML 页面上的所有标签组成了一颗树，比如所有 <body> 标签内的标签都称为 <body> 标签的子元素，<body> 称为父元素。

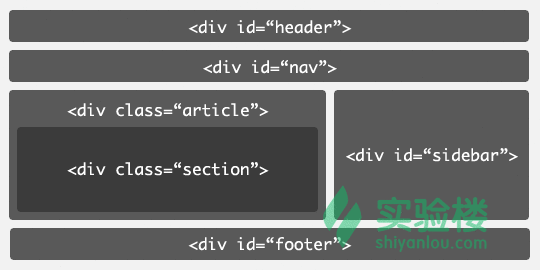
标签经过浏览器渲染显示的时候，可以按默认显示类型分为两种，一种是在行内显示，类似于 <span> 标签，其他大部分的标签都直接显示成一行，比如 <div>, <p> 标签。标签可以包含各种属性，比如 <link> 标签的 href 属性指定了加载的 CSS 文件路径。<div> 标签的 class 属性指定了标签的类名。大部分标签都可有 class 和 id 两个属性，前者指定了标签的类名，后者指定的标签内的标识符，一个 HTML 文档内不能有相同的 id 属性。

一些常用的标签如下表所示：

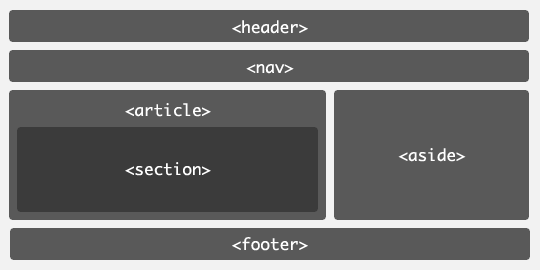
| **标签** | **用途** |
| --- | --- |
| <div> | 最常用的标签，可以包含各种标签，页面布局最常用的标签 |
| <a> | 定义连接 |
| <br> | 插入换行符 |
| <button> | 定义按钮 |
| <dl> | 定义列表 |
| <dt> | 定义列表项 |
| <fieldset> | 定义表单控件组 |
| <form> | 定义表单 |
| <input> | 定义输入框 |
| <ul> | 定义无序列表 |
| <script> | 定义加载的 JavaScript |
| <table> | 定义表格 |
| <img> | 定义图片 |

### 语义标签

HTML5 相对于老版本 HTML 非常好的一点是，可以使用语义标签，所谓语义标签是指从标签名就可以看出标签的用途。比如在前面出现的 <nav> 标签，一看就知道应该实现导航栏功能。使用老版本实现页面时，大部分时候都是通过标签的 class 和 id 属性来指明标签的含义, 比如下图：



但是现在有了 HTML5 的语义标签，可以实现如下：



语义标签实现的同样的功能更加清晰简洁，一些常用的语义标签如下表：

| **标签** | **描述** |
| --- | --- |
| <header> | 表示内容的头部信息，比如一个页面内容的头部； |
| <footer> | 表示内容的尾部信息，可以包含作者，版权信息等内容。 |
| <nav> | 表示导航信息； |
| <section> | 表示节，比如文档的一节内容； |
| <article> | 表示文档内容的一个独立片段， 比如博客条目或报纸上的文章。 |
| <aside> | 表示与页面其他部分略微相关的内容片段。 |

### 表单

表单是网页中必不可少的组件，用户可以通过表单中输入各种数据，然后通过提交表单，将输入的数据提交给网站后台。在 HTML 中，通过 <form> <input> 等标签实现。HTML5 表单的功能进一步加强，可以自定义表单能够接受的数据类型，比如邮箱输入框只能接受邮箱，如果输入非邮箱字符串则不能提交表单。实践见真知，在前文中的 index.html 中输入下面的代码：

<legend>表单示例</legend>

<form action="" method="POST" id="form1">

<input type="text" autofocus="autofocus" required name="auto" placeholder="必填项测试">

<input type="email" placeholder="请输入邮箱" name="mail">

<input type="url" name="url" placeholder="输入正确的网址">

<input type="password" name="password" placeholder="输入密码">

<br>

<br>

<input type="submit" value="提交表单">

</form>

上面的代码，我们通过 <form> 标签定义了一个标签，该表单向网站提交数据时，使用 HTTP POST 方法。接着定义了通过 <input> 标签定义了四个输入框。

第一个输入框通过 required 属性指明了该输入框是必填项；第二个输入框使用 type 属性设置了输入的内容必须是邮件样式的文本，第三个输入框设置了输入的文本必须是 URL 类型的，最后一个输入框为 password 类型，当用户输入内容时，将以星号显示。

使用浏览器打开 index.html 文件后，直接点击 提交表单 可以看到下面的效果：



**CSS**

HTML 实现网页以后就可以使用 CSS 进行控制网页标签的样式和布局。CSS 全名叫层叠样式表，可以用于控制标签如何显示。

在 HTML 页面中使用 CSS 样式有两种方式，一种是通过 <style> 标签直接将 CSS 代码写在页面中，比如下面的代码：

<style>body {background-color: powderblue;}h1 {color: blue;}p {color: red;}</style>

另外一种方式是将 CSS 代码单独写在文件，比如前文中我们创建的 style.css 样式表，然后在 HTML 页面中通过 <link> 标签引用该文件，就像上文中 index.html 内容一样。

在后面的内容中，我们会将 CSS 代码写入 style.css 文件中进行演示。

### 基础语法

CSS 规则由两个主要的部分构成：选择器，以及一个或多个属性值，如下：

selector {

property1: value1;

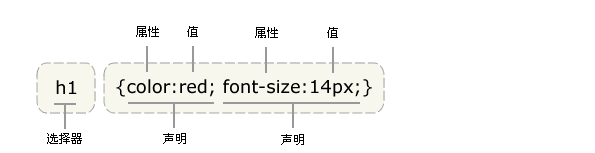
property2: value2;

...

propertyN: valueN;

}

比如：



可以理解为，选择器选择 HTML 页面中相应的标签，然后应用括号中定义的各种属性。把下面的代码写入 style.css 中：

h1 {

color:red;

font-size: 40px;

}

然后把下面代码写入 index.html 中的 <body> 标签内部：

<h1> 实验楼 </h1>

接着刷新浏览器，就可以看到效果：



上面的 CSS 代码，将 <h1> 标签内容设置为红色，并将字体设置为 40px 大小。

可以对选择器进行分组，这样被分组的选择器就可以分享相同的属性设置声明，用逗号将需要分组的选择器分开。在下面的例子中，所有的标题元素都是红的：

h1，h2，h3，h4，h5，h6 {

color：red;

}

HTML 子元素将从父元素继承属性。看看下面这条规则：

body {

color：green;

}

通过 CSS 继承，子元素将继承最高级元素（在本例中是 body）所拥有的属性。所有子元素都显示成绿色。

### 选择器

CSS 有多种选择 HTML 元素的方式，下面我们将一一讲解。

* 派生选择器

通过依据元素在其位置的上下文关系来定义样式，可以使标记更加简洁。派生选择器允许你根据文档的上下文关系来确定某个标签的样式。通过合理地使用派生选择器可以使 HTML 代码变得更加整洁。比方说，你希望列表标签 <li> 中的所有 <strong> 元素变为红色，而不是通常的黑色，可以这样定义一个派生选择器：

li strong{

color: red;

}

<p>

<strong>我是黑色，因为我不在列表当中，所以这个规则对我不起作用</strong></p><u1>

<li><strong>我是红色。这是因为 strong 元素位于 li 元素内。</li></u1>

分别在 style.css 和 index.html 中输入以上代码，就可以看到效果了。

* ID 选择器

ID 选择器可以选择设置了相同 ID 值的 HTML 元素，ID 选择器以 # 字符开始，比如：

#pid a{

color:#00755f;

}

<p id="pid">

hello css <a href="www.shiyanlou.com">shiyanlou</a></p>

可以将上面代码输入相应的文件查看效果。

* 类选择器

在学习 HTML 的时候，我们知道标签可以定义类 class 属性，CSS 就可以基于该属性值也就是类名进行选择。类选择器以 . 字符开始。如下：

.container {

color: red;

}

<div class="container">

container 容器

</div>

* 属性选择器

HTML 标签可以设置各种属性，CSS 可以利用这些属性进行选择标签。如下:

[title] {

color: red;

}

HTML代码：

<p title="foo">属性和值选择器</p>

如果想根据属性和属性值来选择，还可以使用下面的代码：

[title=foo] {

color: red;

}

### 常用属性

CSS 有许多常用属性，比如设置元素的背景图片，颜色，设置字体大小等。 如下代码：

.container {

color: red;

text-align: center;

font-size: 20px;

font-weight: 400;

}

上面的代码设置类名为 container 的元素的内字体为红色，文字居中，字体大小为 20px, 字的粗细为 400。

下面列出一些常用的属性；

* 文本属性

| **属性** | **用途** |
| --- | --- |
| font | 在一个声明中设置所有字体属性 |
| font-family | 规定文本的字体系列 |
| font-size | 规定文本的字体尺寸 |
| font-weight | 规定字的粗细 |
| color | 设置文本的颜色 |
| direction | 规定文本的书写方向 |
| line-height | 设置行高 |
| text-align | 规定文本的水平对齐方式 |
| text-decoration | 规定添加到文本的装饰效果 |
| text-indent | 规定文本块首行的缩进 |

* 背景属性

| **属性** | **用途** |
| --- | --- |
| background | 在一个声明中设置所有的背景属性 |
| background-attachment | 设置背景图像是否固定或者随着页面的其余部分滚动 |
| background-color | 设置元素的背景颜色 |
| background-image | 设置元素的背景图像 |
| background-position | 设置背景图像的开始位置 |
| background-repeat | 设置是否及如何重复背景图像 |
| background-clip | 规定背景的绘制区域 |
| background-origin | 规定背景图片的定位区域 |
| background-size | 规定背景图片的尺寸 |

* 尺寸属性

| **属性** | **用途** |
| --- | --- |
| height | 设置元素高度 |
| max-height | 设置元素的最大高度 |
| max-width | 设置元素的最大宽度 |
| min-height | 设置元素的最小高度 |
| min-width | 设置元素的最小宽度 |
| width | 设置元素宽度 |

* 外边距属性

| **属性** | **用途** |
| --- | --- |
| margin | 在一个声明中设置所有外边距属性 |
| margin-bottom | 设置元素的下外边距 |
| margin-left | 设置元素的左外边距 |
| margin-right | 设置元素的右外边距 |
| margin-top | 设置元素的上外边距 |

* 边框属性

| **属性** | **用途** |
| --- | --- |
| border | 在一个声明中设置所有的边框属性 |
| border-bottom | 在一个声明中设置所有的下边框属性 |
| border-left | 在一个声明中设置所有的左边框属性 |
| border-right | 在一个声明中设置所有的右边框属性 |
| border-top | 在一个声明中设置所有的上边框属性 |

* 内边距属性

| **属性** | **用途** |
| --- | --- |
| padding | 在一个声明中设置所有内边距属性 |
| padding-bottom | 设置元素的下内边距 |
| padding-left | 设置元素的左内边距 |
| padding-right | 设置元素的右内边距 |
| padding-top | 设置元素的上内边距 |

* 定位属性

| **属性** | **用途** |
| --- | --- |
| bottom | 设置定位元素下外边距边界与其包含块下边界之间的偏移 |
| clear | 规定元素的哪一侧不允许其他浮动元素 |
| display | 规定元素应该生成的框的类型 |
| float | 规定框是否应该浮动 |
| left | 设置定位元素左外边距边界与其包含块左边界之间的偏移 |
| overflow | 规定当内容溢出元素框时发生的事情 |
| position | 规定元素的定位类型 |
| right | 设置定位元素右外边距边界与其包含块右边界之间的偏移 |
| top | 设置定位元素的上外边距边界与其包含块上边界之间的偏移 |
| visibility | 规定元素是否可见 |

以上只是列举常用的 CSS 属性，还有一些不常用的属性没有列出。可以看到 CSS 属性非常丰富，设计 HTML 元素样式的方方面面。这些属性也不必一次全部掌握，后面遇到的时候再查询相关文档即可。完整的 CSS 属性列表可以参考 [https://developer.mozilla.org/en-US/docs/Web/CSS/Reference](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference" \t "https://www.shiyanlou.com/courses/_blank) 。

### 框模型

在上一节中，我们看到可以设置一个元素的外边距，边框和内边距属性，这些属性就构成了框模型。

一个 HTML 标签元素经过浏览器渲染显示在页面上后，最内部分是实际的内容，字节包含内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框，边框以外是外边距，外边距默认是透明的，因此不会遮挡任何元素。先让我们看下例子：

在 style.css 中，输入下面的代码:

\* {

margin: 0;

padding: 0;

}

.box {

width: 70px;

height: 30px;

margin: 60px;

border-style: solid;

border-width: 50px;

padding: 50px;

}

上面的代码，先通过 \* 选择器设置所有元素的外边距和内边距为 0。这是因为浏览器默认情况下会设置元素的内边距和外边距，为了不影响效果所以这里首先将其清零。接着设置了 .box 元素的宽度和高度，然后设置了内边距，边框和外边距。其中设置了边框为实线，宽度为 50px。px 是像素单位，表示一个像素的大小。设置内边距，边框和外边距不会影响元素内容的尺寸，但是会增加元素框的总尺寸。

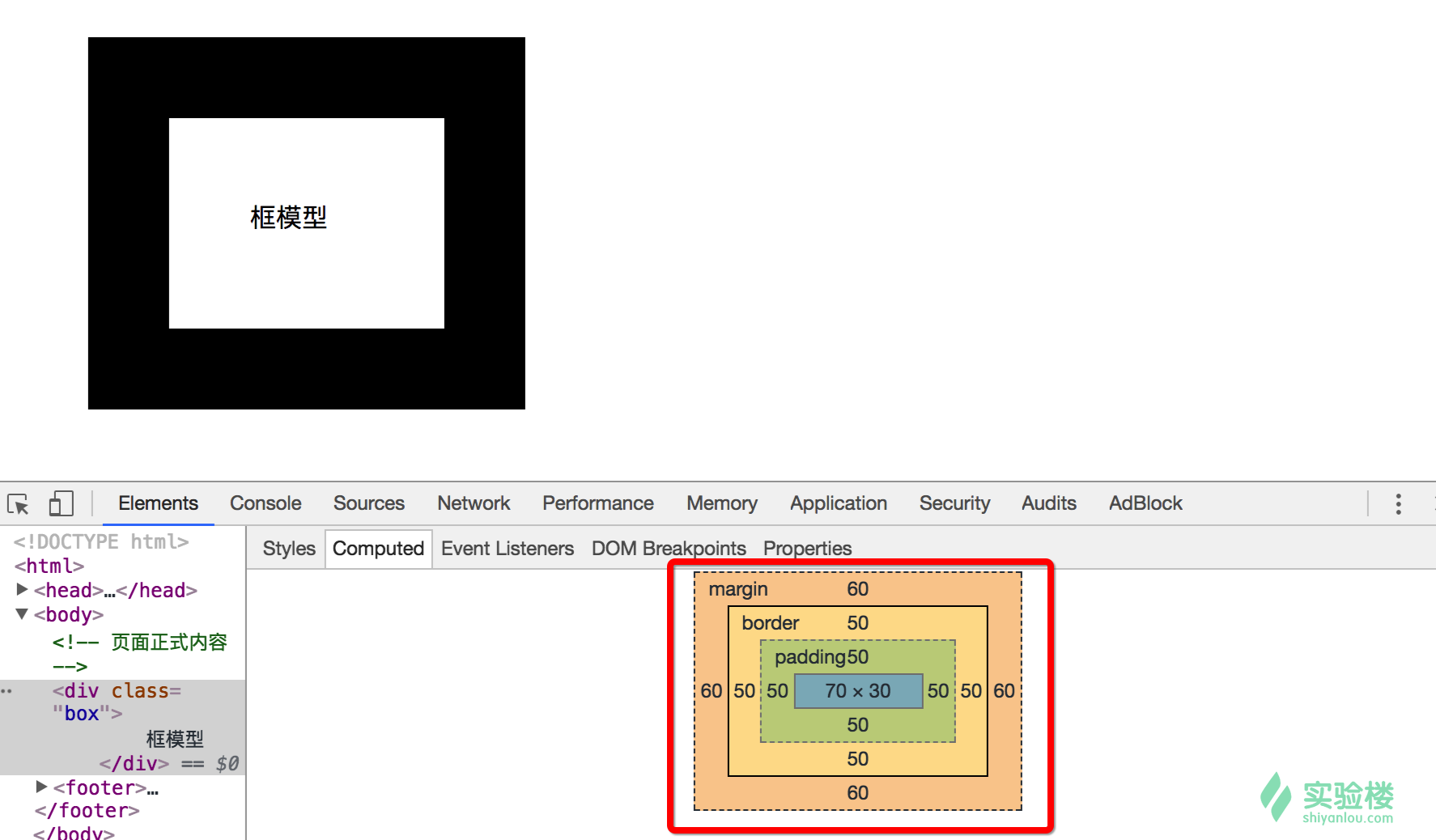
然后在 index.html 文件的 <body> 标签内输入以下代码：

<div class="box">

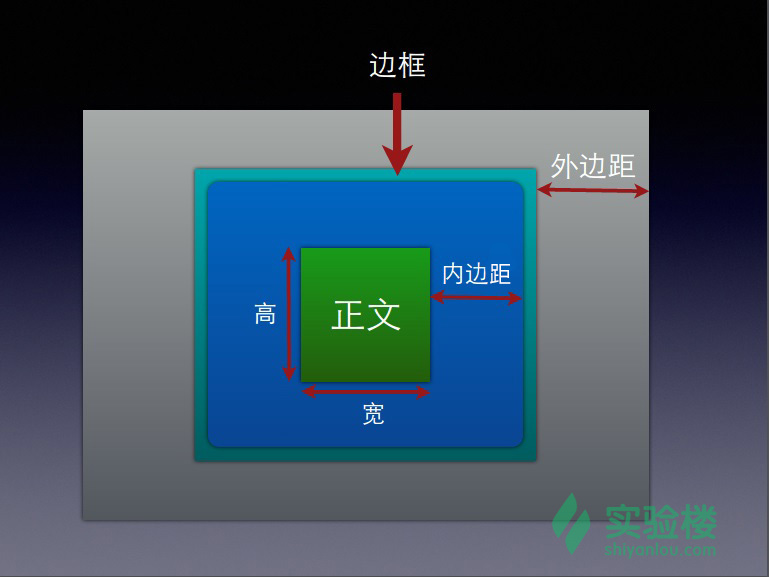
框模型

</div>

刷新浏览器，就可以看到如下效果：



上图中，红框内是浏览器渲染元素后的框模型，可以清晰的看到元素的内外边距，和 CSS 设置的值一样。在 Chrome 浏览器中，可以通过点击右键，然后选择查看元素，查看元素的框模型。下图比较清晰的展示了元素内外边距的关系：



内边距在元素的内容外，边框内。padding 属性用于控制元素的内边距，可以通过下面几个属性分别控制元素的上下左右内边距：

* padding-top
* padding-right
* padding-bottom
* padding-left

也可以直接在 padding 属性中一次性设置上下左右内边距，方法如下：

* .box {padding: 10px;} 设置上下左右的内边距为 10 px;
* .box {padding: 10px 20px 20px 10px} 设置上，右，下，左分别为 10px, 20px, 20px, 10px，按从上内边距开始顺时针方向；
* .box {padding: 5px 10px} 设置上下内边距为 5px, 左右内边距为 10px;
* .box {padding: 5px 4px 3px} 等价于 .box{padding: 5px 4px 3px 4px};

设置内边距的距离可以使用像素单位，也可以使用其他比如 em 字体大小单位, cm 厘米，或者设置成百分数，比如 .box {padding: 50%}。设置成百分数时，最终的内边距宽度是相对于其父元素的宽度计算的，外边距也是同样的计算方式。所以如果父元素的宽度发生改变，它们也会改变。

外边距的设置方法和内边距一致，可以一次性在 margin 属性中指定所有外边距的尺寸，或者通过 margin-top 等标签分别设置上下左右的外边距。

外边距在实际使用中，当两个垂直外边距相遇时，它们会合并成一个外边距。合并后的外边距的高度等于两个发生合并外边距的高度中的较大者，如下图所示：

