目录

CONTENTS

模块 1　网页布局设计体验　　/1

1.1　CSS 网页 …… 1
1.1.1　什么是 CSS …… 1
1.1.2　Web 标准 …… 1
1.1.3　网页结构与表现分离实例 …… 2
1.2　编写 CSS 布局的 XHTML …… 3
1.2.1　一个简单的 XHTML 页面实例 …… 4
1.2.2　XHTML 编写的规范 …… 4
1.2.3　CSS 网页的编辑 …… 6
1.2.4　CSS 的使用注释 …… 7
1.2.5　制作一个简单的 CSS 实例 …… 7
1.3　CSS 控制的页面 …… 8
1.3.1　CSS 的引入 …… 8
1.3.2　行内样式 …… 8
1.3.3　内嵌式调用 CSS 样式的实例 …… 8
1.3.4　链接外部 CSS 样式的实例 …… 9
1.3.5　导入外部样式 …… 10
拓展实践 …… 10
习题 …… 11

模块 2　CSS 的实战基础　　/13

2.1　CSS 的基本语法 …… 13
2.1.1　CSS 语法的格式 …… 13
2.1.2　CSS 的取值单位 …… 14
2.2　基本 CSS 选择符 …… 15
2.2.1　标签选择符 …… 15
2.2.2　类选择符 …… 16
2.2.3　ID 选择符 …… 17

2.3 灵活使用 CSS 选择符 …… 19
2.3.1 分组选择符及实例 …… 19
2.3.2 通配选择符 …… 20
2.3.3 包含选择符及实例 …… 20
2.3.4 选择符嵌套及实例 …… 21
2.3.5 标签指定选择符 …… 22
2.3.6 伪类选择符 …… 22
2.4 CSS 继承应用的实例 …… 23
2.5 CSS 样式和选择符优先级 …… 24
2.5.1 CSS 样式的优先级原则 …… 24
2.5.2 CSS 样式的优先级 …… 24
2.5.3 CSS 选择符的优先级实例 …… 25
拓展实践 …… 26
习题 …… 27

模块 3 DIV 布局与 CSS 定位 /30

3.1 盒子模型 …… 30
3.1.1 盒子模型的概念 …… 30
3.1.2 padding、margin、border 设置实例 …… 31
3.1.3 盒子模型尺寸的计算 …… 35
3.1.4 盒子外边距合并实例 …… 36
3.1.5 块级元素与行内元素 …… 38
3.2 页面定位技术 …… 39
3.2.1 CSS 定位机制 …… 39
3.2.2 浮动定位问题 …… 39
3.2.3 清除浮动 …… 42
3.3 CSS 定位属性 …… 45
3.3.1 相对定位 …… 45
3.3.2 绝对定位 …… 47
3.3.3 固定定位 …… 47
3.4 CSS 布局技术 …… 48
3.4.1 CSS 布局理念 …… 48
3.4.2 一列固定宽度居中 …… 49
3.4.3 一列自适应宽度居中 …… 50
3.4.4 二列固定宽度居中 …… 50
3.4.5 二列自适应宽度居中 …… 51
3.4.6 CSS 高度自适应 …… 52
3.4.7 溢出内容 overflow 属性 …… 54

3.5 三列布局实例——唐诗《黄鹤楼》画卷 …… 54
拓展实践 …… 56
习题 …… 57

模块 4 文字段落与列表的应用 /60

4.1 CSS 文字样式 …… 60
4.1.1 字体 …… 60
4.1.2 文字大小 …… 62
4.1.3 颜色 …… 63
4.1.4 样式 …… 64
4.2 文本对齐 …… 64
4.2.1 CSS 的文本属性 …… 64
4.2.2 文本添加修饰实例 …… 64
4.2.3 文本居中排列实例 …… 65
4.2.4 制作 Google 公司 Logo 实例 …… 66
4.2.5 首字下沉图文混排实例 …… 66
4.3 CSS 列表样式 …… 68
4.3.1 列表的类型 …… 68
4.3.2 CSS 的列表属性 …… 69
4.3.3 List 图文混排 …… 70
4.3.4 竖排唐诗文字 …… 71
拓展实践 …… 72
习题 …… 73

模块 5 背景与图片的应用 /76

5.1 页面的背景图 …… 76
5.1.1 背景颜色与背景图片 …… 76
5.1.2 前景图与背景图 …… 76
5.2 图片样式 …… 77
5.2.1 CSS 图片边框 …… 77
5.2.2 图片对齐方式 …… 80
5.2.3 图文混排效果 …… 81
5.3 背景图片的应用实例 …… 82
5.3.1 背景渐变效果实例 …… 83
5.3.2 导航按钮实例 …… 84
5.3.3 横格信纸实例 …… 85
5.4 CSS 样式圆角布局研究 …… 86
5.4.1 以四角小图实现圆角效果实例 …… 87

5.4.2 以画线方式实现圆角效果实例 …… 88
5.4.3 多图片组合布局圆角效果实例 …… 90
5.5 背景图片的 CSS Sprite 技术 …… 91
拓展实践 …… 93
习题 …… 94

模块 6 表格与表单的样式应用 /96

6.1 表格控制 …… 96
6.1.1 表格标记 …… 96
6.1.2 表格隔行变色实例 …… 100
6.1.3 鼠标经过时变色的表格实例 …… 102
6.2 表单控制 …… 103
6.2.1 表单标记 …… 104
6.2.2 登录框表单的应用实例 …… 107
6.2.3 网民调查问卷实例 …… 109
拓展实践 …… 111
习题 …… 113

模块 7 超链接与导航栏的样式应用 /116

7.1 超链接 CSS 样式 …… 116
7.1.1 定义超链接 CSS 样式 …… 116
7.1.2 浮雕式超链接的效果 …… 118
7.2 超链接制作导航栏 …… 120
7.2.1 CSS 样式实现垂直导航栏 …… 121
7.2.2 利用背景图片制作立体导航菜单 …… 122
7.2.3 制作带下拉子菜单的导航菜单 …… 124
拓展实践 …… 127
习题 …… 127

模块 8 CSS 控制其他元素的显示 /130

8.1 CSS 静态滤镜 …… 130
8.1.1 透明度滤镜 …… 131
8.1.2 模糊滤镜 …… 132
8.1.3 透明色滤镜 …… 133
8.1.4 投射阴影滤镜 …… 134
8.1.5 对称变换滤镜 …… 135
8.1.6 光晕滤镜 …… 136
8.1.7 灰度滤镜 …… 137

8.1.8 反色滤镜 …… 138
8.1.9 遮罩滤镜 …… 139
8.1.10 阴影滤镜 …… 140
8.1.11 X 射线滤镜 …… 141
8.1.12 波浪滤镜 …… 141
8.1.13 一个使用颜色渐变滤镜的实例 …… 142
8.2 控制滚动条的显示 …… 143
8.2.1 滚动条样式的属性 …… 144
8.2.2 控制滚动条样式的实例 …… 144
8.3 CSS 布局中的兼容问题 …… 145
8.3.1 兼容问题的由来 …… 145
8.3.2 兼容问题的解决 …… 146
8.3.3 一个关于浏览器显示差异的实例 …… 147
拓展实践 …… 149
习题 …… 150

模块 9 CSS 3.0 的新技术与应用 /152

9.1 CSS 3.0 简介 …… 152
9.1.1 CSS 3.0 技术的发展 …… 152
9.1.2 CSS 3.0 中的新特性 …… 152
9.2 进入 CSS 3.0 样式新世界 …… 153
9.2.1 用 CSS 3.0 实现圆角效果实例 …… 153
9.2.2 CSS 3.0 边框、阴影实例 …… 154
9.2.3 CSS 3.0 文本描边及阴影效果实例 …… 155
9.2.4 CSS 3.0 表格美化效果实例 …… 156
9.2.5 CSS 3.0 按钮动画过渡效果实例 …… 157
9.2.6 CSS 3.0 多列布局排版实例 …… 158
拓展实践 …… 159
习题 …… 161

模块 10 制作个人博客页面 /162

10.1 构架设计制作 …… 162
10.1.1 功能需求分析与界面设计 …… 162
10.1.2 页面结构的规划 …… 162
10.1.3 切割网页的效果 …… 164
10.1.4 页面主体框架内容 …… 166
10.2 制作页面内容 …… 167
10.2.1 制作页面头部 Banner 的内容与导航 …… 167

10.2.2 制作页面导航左侧列表…… 169
10.2.3 制作页面主体内容部分…… 173
10.2.4 制作页面底部内容…… 178
10.3 页面细节整体调整…… 179
10.4 页面的浏览与兼容测试…… 180
拓展实践…… 182
习题…… 183

模块 11 企业网站制作 /184

11.1 企业网站构架…… 184
11.1.1 功能需求分析与界面设计…… 184
11.1.2 页面结构的规划…… 185
11.1.3 切割网页效果图…… 185
11.1.4 页面主体框架内容…… 187
11.2 制作页面内容…… 187
11.2.1 制作 Banner 内容与导航 …… 187
11.2.2 制作页面左侧列表…… 189
11.2.3 制作页面主体内容…… 191
11.2.4 制作页面图片展示…… 192
11.2.5 制作页面底部内容…… 193
11.3 页面细节整体调整…… 194
11.4 页面的浏览与兼容测试…… 194
拓展实践…… 195
习题…… 196

附录 CSS 常用属性 /197

参考文献 /203

模块 1

网页布局设计体验

学习目标

- 了解 CSS 的概念；
- 了解 XHTML 编写的规范，会编写网页；
- 会使用 CSS 样式表控制网页；
- 会对 CSS 样式表添加注释；
- 熟悉网页开发工具。

1.1 CSS 网 页

1.1.1 什么是 CSS

CSS(Cascading Style Sheet，层叠样式表，简称为样式表)是 W3C 组织制定的用于控制网页内容显示效果的一种置标语言。

CSS 通过结构语言控制页面中内容部分的显示效果。通过使用 CSS，可以控制页面中的背景、文本、布局方式等，并使页面更加美观。CSS 语言是一种置标语言，不需要编译，可以直接由浏览器执行(属于浏览器解释型语言)。

1.1.2 Web 标准

Web 标准是在 W3C 组织下建立起来的，目的是用于建立更加规范的页面。使用 Web 标准制作的页面，可以完成页面结构与表现的分离，使页面具有更大的扩展性和简洁性。网页主要由 3 部分组成：结构(Structure)、表现(Presentation)和行为(Behavior)。对应的标准也分 3 个方面：结构标准语言主要包括 XHTML 和 XML，表现标准语言主要包括 CSS，行为标准主要包括对象模型(如 W3C DOM)、ECMAScript 等。

1. 结构标准语言

在 Web 标准中规定的结构语言有两种：一种为 XML 语言；另一种为 XHTML 语言。

XML(eXtensible Markup Language，可扩展置标语言)具有强大的扩展性，可以用于网络数据的转换和描述。XML 具有简洁有效、易学易用、开放的国际化标准、高效可扩充的特点。

XHTML(eXtensible HyperText Markup Language,可扩展超文本置标语言)是在HTML 4.0 的基础上,用 XML 的规则对其进行扩展建立起来的语言。XHTML 作为HTML 向 XML 的过渡,具有更加严格的代码规范,也具有更大的扩展性。

在实际的网页制作过程中,通常使用 XHTML 语言作为页面的结构语言。

2. 表现标准语言

在 Web 标准中,表现标准语言使用 CSS,目前推荐遵循的是 W3C 制定的 CSS 2.0。在 Web 标准中,推荐使用 CSS 对页面进行布局和美化。使用 CSS 与 XHTML 语言相结合,能够实现网页结构与表现相分离,给网站的维护和管理带来了极大的方便。

3. 行为标准

在 Web 标准中,行为标准分为两个部分:一部分为 DOM;另一部分为 ECMAScript。使用 Web 标准中提供的行为,可以很好地完成页面的交互,其具体内容如下所述。

DOM(Document Object Model,文档对象模型)是网页与 Script(或程序语言)之间进行沟通的桥梁,是访问页面中标准组件的一种方法。

ECMAScript 是 ECMA(European Computer Manufacturers Association,欧洲计算机制造商协会)制定的标准脚本语言,用来完成页面的交互。

1.1.3 网页结构与表现分离实例

CSS 的作用就是通过结构语言控制页面中内容部分的显示效果。通过使用 CSS,可以控制页面中的背景、文本、布局方式等。使用 CSS 控制页面表现的好处在于,同样的内容与结构,可以使用不同的表现形式。只定义不同的 CSS 样式,在不更改页面内容和结构的前提下,可以更换页面的布局结构。图 1-1 是一个表单结构内容的网页,通过定义不同的 CSS 样式而控制页面的表现后,其显示效果如图 1-2 和图 1-3 所示。

图 1-1 无样式的网页内容

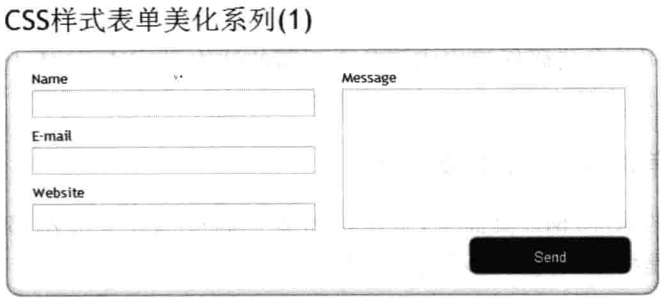

图 1-2　相同结构表单样式 1

图 1-3　相同结构表单样式 2

采用 CSS 样式，页面内容和显示样式分离，便于程序员、UI 设计人员协同开发，使用 CSS 样式设置页面的格式，可以将格式控制代码单独存放，对页面样式进行统一管理。采用样式，只要修改 CSS 文件，就可改变整个网站的风格特色，避免一个一个地修改网页，这大大减少了重复工作量。

采用 CSS 样式对网站建设的好处如下。

(1) 文件下载与页面显示速度更快。

(2) 用户能够通过样式选择定制自己的表现界面。

(3) 代码更简洁，成本降低，内容能被更广泛的设备所访问(包括屏幕阅读机、手持设备、搜索机器人、打印机、电冰箱等)。

(4) 更少的代码和组件，容易维护，改版方便，不需要变动页面内容。

(5) 更容易被搜索引擎搜索到。

(6) 所有页面都能提供适于打印的版本。

1.2　编写 CSS 布局的 XHTML

现在通常使用的 XHTML 版本是 W3C 组织 2000 年年底发布的 XHTML 1.0。XHTML 用来定义页面中的各种元素，如页面中的图片、文本、超链接、列表及标题等

内容。

XHTML 使用各种代码标记构成了整个页面的结构,但 XHTML 代码本身只用来构成页面的结构,并不直接显示。

1.2.1 一个简单的 XHTML 页面实例

在一个简单的 XHTML 页面中,页面的所有代码可以分为显示部分和不显示部分(显示和不显示,是根据浏览器显示内容区分的)。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>网页标题填写在这里</title>
<link href="css/main.css" rel="stylesheet" type="text/css" />
                                                    /*CSS样式文件引用方式*/
<meta name="Description"  content="网站描述的内容" />
<meta name="keywords" content="网站的关键字" />
</head>
<body>
  网页中的内容显示区
</body>
</html>
```

针对上面的网页,XHTML 网页文档结构主要有 3 个标签——<html>、<head>、<body>,主体结构如下。

```
<html>
  <head>
  …
  </head>
  <body>
  …
  </body>
</html>
```

<html>标签表示网页文档,也可以说是 HTML 文档源代码的分隔符。

<head>标签表示网页头部信息。在<head>标签中包含网页语言编码、样式、网页标题等信息。

<body>标签表示网页主体信息。

1.2.2 XHTML 编写的规范

1. 文档类型

当我们用 Dreamweaver 新建一个 HTML 文档时,查看源代码,会发现代码最上部有这样一段代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

这段代码标明本文档是过渡类型。另外，文档还有框架和严格类型，目前一般都采用过渡类型。在制作页面时，建议大家一定要保留这句话，删除它后可能引起某些样式表失效或其他意想不到的问题。

2. 名字空间

在标准的 HTML 文档中，标签还会包含如下代码。

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

xmlns 属性是 XHTML namespace 的缩写，中文翻译为“名字空间”。XHTML 是 HTML 向 XML 过渡的标识语言，它需要定义名字空间。

3. 语言编码

在标准的 HTML 文档中，还会包含如下代码。

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

它标示文档的语言编码。这里的 gb2312 向浏览器指明，本文档采用简体中文编码。还有一种常用的编码是 UTF-8 编码，它是国际通用的编码。不管我们采用哪种编码，有一点就是所包含的 CSS 样式表和其他文件也必须和该文档的编码一样，否则会出现乱码。

另外，在符合标准的 HTML 文档中，还要注意以下几点。

(1) XHTML 区分大小写。所有的 XHTML 元素和属性的名字都必须使用小写。否则文档将被 W3C 校验为无效的。例如，<head>{...}</head>是正确的，<BODY>{...}</BODY>是不正确的。

(2) 在 XHTML 中，标签在页面中都必须结束。成对的标签以“/标签名”结束；单一的标签在本身的结尾打上“/”来结束。例如，下面成对的标签。

```
<p>每一个打开的标签都必须关闭。</p>
<h1>XHTML 不接受不关闭的标签。</h1>
```

下面是单一的标签。

```
<br />
<img src="webstandards.gif" />
```

(3) 给所有属性值加引号。

在 HTML 中，不需要给属性值加引号，但是在 XHTML 中，它们必须被加引号。例如，应写成 height="100"，而不能写成 height=100。

(4) 为图片添加 alt 属性

alt 属性指定了当图片不能显示的时候就显示供替换文本，这样做对正常用户可有可无，但对纯文本浏览器和使用屏幕阅读机的用户来说是至关重要的。只有添加了 alt 属性，代码才会被 W3C 正确性校验通过。注意，要添加有意义的 alt 属性，下面的写法毫无

意义。

```
<img src="logo_unc_120x30.gif" alt="logo_unc_120x30.gif">
```

正确的写法如下。

```
<img src="logo_unc_120x30.gif" alt="UNC 公司标志,点击返回首页">
```

我们可以利用 W3C 提供免费校验服务(validator. w3. org),发现错误后逐个修改。在后面的资源列表中我们也提供了其他校验服务和对校验进行指导的网址,可以作为 W3C 校验的补充。

1.2.3 CSS 网页的编辑

CSS 与 HTML 一样都是一种标识语言,在任何文本编辑器中都可以打开或编辑。

CSS 文件为纯文本文件,所以一般的文字处理工具都可以编辑 CSS 代码。例如,有时以使用"记事本"工具打开 CSS 文件。在实际制作的过程中,推荐使用网页设计软件 Dreamweaver 来编辑 CSS 代码。使用 Dreamweaver 编辑 CSS 代码,不但能方便地查阅代码的行数,还有属性设置自动提示功能,可以方便地添加各种 CSS 属性,图 1-4 所示为添加颜色值。

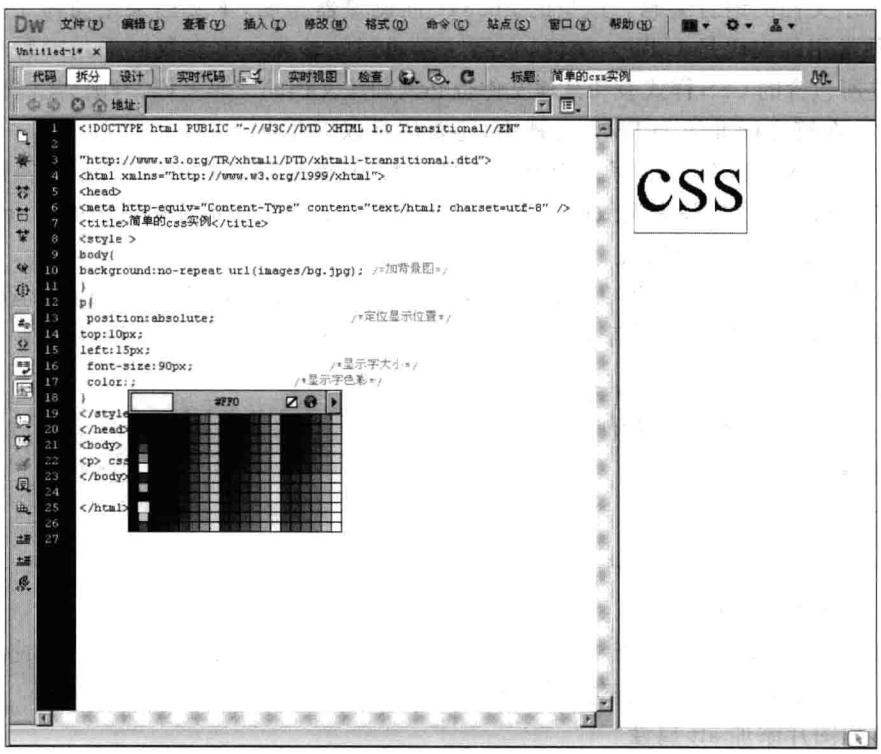

图 1-4 使用 Dreamweaver 编辑 CSS 代码

1.2.4 CSS 的使用注释

为了方便阅读和容易维护，可利用注释语句进行注释，CSS 使用“/ * 注释语句 * /”来进行注释。还可利用增加空格来美化 CSS 源代码排列，例如，对于上面可以进行如下行间注释，直接写于属性值后面。

```
body {                                                    /* 页面基本属性 */
  font-size: 12px; color: #CCCCCC;
}
p {background-color:  #FF00FF; /* 段落文本基础属性 */ }
```

另外，还有整段注释，分别在开始及结束地方加入注释，例如：

```
/* =====搜索条===== */
.search {
    border:1px solid #fff;
    background:url(../images/icon.gif) no-repeat #333;
}
/* =====搜索条结束===== */
```

1.2.5 制作一个简单的 CSS 实例

在本实例中，页面内容只有“CSS”3 个字母。通过定义页面中各种元素的 CSS 样式，以及利用修饰的素材图片，可以使页面显示出丰富的效果。在实例中使用的代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>简单的 CSS 实例</title>
<style >
body{
    background:no-repeat url(images/bg.jpg);              /* 加背景图 */
}
p{
  position:absolute;                                      /* 定位显示位置 */
  top:10px;
  left:15px;
  font-size:90px;                                         /* 显示字的大小 */
  color:#FF0;                                             /* 显示字的色彩 */
}
</style>
</head>
<body>
<p>css </p>
```

```
</body>
</html>
```

以上代码中,页面的内容为 p 元素中包含的“CSS”字母。通过使用 CSS 样式,定义了内容的背景和位置。代码运行后的显示效果如图 1-5 所示。

图 1-5 一个简单的 CSS 实例效果

由于在这段代码中,具体的样式分为两个部分,即 body 部分和 p 部分。在 body 部分,定义了页面的背景图片;在 p 部分,定义了文本内容的大小和位置。以上定义的两个样式,通过代码中元素的名称 body 和 p 与页面具体的内容联系在一起。如果取消以上定义的样式,页面将在左上角只显示黑色“CSS”文本内容。

1.3 CSS 控制的页面

1.3.1 CSS 的引入

在页面中使用 CSS 样式,通常有 3 种方法,即行内样式、内部样式和链接外部样式。每一种方法的添加位置和格式都有所区别,并且具有不同的优先级,其中推荐使用的方法是链接外部样式。下面详细讲解这 3 种添加 CSS 样式的方法。

1.3.2 行内样式

在行内添加 CSS,就是将 CSS 样式添加在页面的元素标签中,此时样式的作用范围为元素标签开始和结束之间。行内添加的 CSS 样式具有最高的优先级。

```
<p style=" font-size:  36px; line-height:  100px;">这是一个使用行内样式的实例。
</p>
```

以上的代码使用 style 属性将 CSS 样式添加在 p 元素中。其中 CSS 样式的含义为,定义字体大小为 36 像素,行高为 100 像素。这种在标签内以 style 标记的为行内样式。

行内样式只针对标签内的元素有效,因其没有和内容相分离,所以不建议使用。

1.3.3 内嵌式调用 CSS 样式的实例

内嵌式调用 CSS 样式或内部样式,就是通过在页面的 style 元素中添加 CSS 代码定义 CSS 样式,调用相应样式。内嵌式调用 CSS 样式优先级低于行内样式。

一个使用内嵌式调用 CSS 样式的实例,其代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>文档标题部分</title>
<style>
div{
    font-size: 48px;
    line-height100px; color: white; background-color: black;
}
</style>
</head>
<body>
<div>这是一个使用内嵌式样式的实例。</div>
</body>
</html>
```

以上代码中，通过在 head 元素中定义 style 元素，完成 CSS 样式的定义。其中的 CSS 样式除了定义字体大小和行高之外，还定义了字体的颜色和背景颜色。代码运行后的显示效果如图 1-6 所示。

这是一个使用内嵌式样式的实例。

图 1-6 内嵌式调用 CSS 样式效果

这种形式即内部样式表，它是以＜style＞和＜/style＞结尾，写在源代码的＜head＞标签内。这样的样式表只能针对本页有效，不能作用于其他页面。

1.3.4 链接外部 CSS 样式的实例

链接外部 CSS 样式，就是将 CSS 样式定义在一个外部的文件中，然后通过使用 link 元素，在页面中调用这个外部文件。独立的 CSS 文件是后缀名为. css 的文本文件，可以保存在磁盘的任意位置。链接外部 CSS 样式优先级较低，低于行内样式。

一个链接外部 CSS 样式的实例，其代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>标题部分</title>
<link href="style/css.css" type="text/css" rel="stylesheet" />
</head>
<body>
<div>这是一个使用链接外部 CSS 样式的实例。</div>
</body>
</html>
```

以上代码中，使用 link 元素，链接了一个名为 css. css 的外部样式文件，这个外部样式文件中定义的 CSS 代码如下。

```
div{
    font-size: 48px; line-height: 100px; color: white; background-color: black;
```

```
    border: 15px solid gray;
}
```

在这段 CSS 代码中,除了定义字体的大小、行高、颜色和背景颜色之外,还定义了元素的边框为 15 像素的灰色实线。代码运行后的显示效果如图 1-7 所示。

图 1-7 链接外部 CSS 样式效果

这种形式是把 CSS 单独写到一个 CSS 文件内,然后在源代码中以 link 方式链接。它的好处是不但本页可以调用,其他页面也可以调用,是最常用的一种形式。

1.3.5 导入外部样式

先来看下面这条代码。

```
@import url("/css/global.css");
```

导入外部样式是以@import url 标记所链接的外部样式表,它一般常用在另一个样式表内部。如 layout.css 为主页所用样式,那么我们可以把全局都需要用的公共样式放到一个 global.css 的文件中,然后在 layout.css 中以@import url("/css/global.css")的形式链接全局样式,这样就使代码达到很好的重用性。

拓展实践

1. 用 CSS 定义标签元素外观

使用 CSS 来确定标签元素的外观。我们通过 CSS,可以对<p>文本进行变大、粗体、变彩色等操作,还可对<p>进行定位等。我们使用 CSS 来确定元素的外观,例如,<h1>能变成红色、8 像素的小字。

```
h1 {color:#F00; font-size: 8px; }
```

2. 常用的重要标签

:网页结构化标签,它负责划分网页结构,并负责包含不同的模块。

:行内文本和对象包含标签,它负责修饰行内对象样式。

<p>:主要负责管理段落文本内容的组织和管理。

<h>:标题标签,根据重要性分为 6 个级别,hi 表示一级标题,一般页面中最好只包含一个一级标题。

、<ol>和<li>:项目列表标签,主要负责网页内同类信息的列表。

<img>:主要负责网页内图片的显示。