

# HTML自学笔记#2

原创 Zerg\_Wang 于 2019-06-01 16:14:20 发布 97 收藏  
分类专栏： Web Development

编辑 版权



Web Development 专栏收录该内容

0 订阅 3 篇文章

环境：VS Code

插件：HTML Snippets、open in browser

## 表格

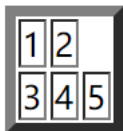
### 内容与边框

通过

声明每个单元格及其内容。
--------------

表格的设置不一定非要n×m，如下图：

```
1 <table border="5">
2   <tr>
3     <td>1</td>
4     <td>2</td>
5   </tr>
6   <tr>
7     <td>3</td>
8     <td>4</td>
9     <td>5</td>
10  </tr>
11 </table>
```



当然，如果在第一行写了三个<td>，但第三个仅仅是<td></td>，则会显示出其边框，无内容：



可通过<table>处的属性border设置外边框，不写则为0，内、外边框都无。若大于0，则外边框按这个值变化，内边框不变。

除了border，属性cellspacing用于设置单元格间距，属性cellpadding设置单元格边距。（即单元格中的内容与单元格边框的距离）

此外，整个table或单个单元格都可通过background属性设置背景、align设置对齐方式等等。

### 表头及标题

假设有3列，则表头可如下设计：

```
1 <tr>
2   <th>Column 1</th>
3   <th>Column 2</th>
4   <th>Column 3</th>
5 </tr>
```

则内容会对应加粗，出现在每一列中间。

如果要表头在列表上端，则要写于其他行之前。

表头不一定要与实际列数相匹配，可自由设计。此外，表头也可以是每行一个，这样一来<th>要在每个<tr>中声明。

标题写在<table>中，但不需要声明<tr>：

```
<caption>我的标题</caption>
```

标题内容会显示在表的上方居中位置，无论这段代码写在何处。

Name	Age	Gender
John	22	F
Mike	41	M

单元格合并

通过colspan（合并相邻列）和rowspan（合并相邻行）进行操作。

```
1 | <tr>
2 |     <th>Name</th>
3 |     <th colspan="2">Phone Number</th>
4 | </tr>
5 | <tr>
6 |     <td>John</td>
7 |     <td>1521336</td>
8 |     <td>1526688</td>
9 | </tr>
```

Name	Phone Number	
John	1521336	1526688

列表

无序列表

以<ul>开头，以<li>定义每一项。

```
1 | <ul>
2 |     <li><h2>Chapter I</h2></li>
3 |     <li><h2>Chapter II</h2></li>
4 | </ul>
```

- Chapter I
- Chapter II

有序列表

以<ol>开头，以<li>定义每一项。每一项开头不再是黑点，而是按顺序排列的数字：

## 1. Chapter I

## 2. Chapter II

可在<ol>处通过属性start定义序号的开始值，默认为1。

若连续定义多个有序表，且start为默认值，则序号不会在表间延续，到下一个表，又从1、2、3开始。

### 自定义列表

以<dl>开头，以<dt>定义每一项。每项开头无黑点也无数字。

此外，在<dt>下还可通过<dd>补充对该项的描述，每项可多个<dd>对其进行描述。

```
1 <dl>
2   <dt>Mike</dt>
3   <dd>27</dd>
4   <dd>worker</dd>
5   <dt>James</dt>
6   <dd>23</dd>
7   <dd>engineer</dd>
8 </dl>
```

Mike  
27  
worker  
James  
23  
engineer

## 代码结构

### 块元素与内联元素

HTML中元素可分为块元素（block level element）以及内联元素（inline element）。

浏览器显示块元素时，通常会在其开始和结束前以一行与其他元素相隔，如<h1>、<p>。内联元素则不会，如<a>、<img>。

### <div>与<span>

这两个相当于C++中的大括号{}，相当于Pascal中的begin和end，逻辑上为一体的一部分代码可以写在一个<div>或<span>中，两者的区别在于，<div>为块元素，<span>为内联元素。

此外，通过在<div>或<span>处的属性设置，可同时将多组代码设置为同一风格，如：

```
1 <div style="color:rgb(125, 17, 134)">
2   <h1>Chapter II</h1>
3   <p>blablabla...</p>
4 </div>
```

## Chapter II

blablabla...

此外，更可以为每个<div>设置类别，相同类别设置为同一风格，不同类别设置为不同风格，然后在<body>之外可为每一类设计显示风格，例如：

```
1 <html>
2 <head>
3   <style>
4     .fruits {
5       background:red;
```

```

6 |         color:white;
7 |         margin:20px;
8 |     }
9 |     .animals{
10 |         background:rgba(53, 29, 1, 0.829);
11 |         color:white;
12 |         margin:20px;
13 |     }
14 | </style>
15 | </head>
16 | <body>
17 |     <div class="fruits">
18 |         <h1>Apple</h1>
19 |     </div>
20 |
21 |     <div class="animals">
22 |         <h1>Cat</h1>
23 |     </div>
24 | </body>
25 | </html>

```

除了class，也可以使用<div id="animals">这种形式，在<style>处编辑则在类名前加“#”而不是“.”。

此外，还可以替换掉<div>，使用自定义的布局：

```

1 | <!DOCTYPE html>
2 | <html>
3 | <head>
4 |     <style>
5 |         header {
6 |             color:purple;
7 |             text-align:center;
8 |         }
9 |     </style>
10 | </head>
11 | <body>
12 |     <header>
13 |         <h1>My Diary</h1>
14 |     </header>
15 | </body>
16 | </html>

```

## 框架

将当前页面分割为多个框架，使其可显示多个页面，用<frameset>设置：

```

1 | <html>
2 | <frameset cols="50%,50%">
3 |     <frame src="index.html"/>
4 |     <frame src="index2.html"/>
5 | </frameset>
6 | </html>

```

## 页面1

## 页面2

[https://blog.csdn.net/Zerg\\_Wang](https://blog.csdn.net/Zerg_Wang)

通过在<frameset>中添加属性来确定为水平分割（rows）或者垂直分割（cols），还可通过百分比为每个框架分配大小，可写任意个百分比（对应多个页面），只要加起来为100%即可。（其实不为100%也不会报错……）

<frame>设置每个框架内显示的页面，可以为网址，也可以为本地的文档。

框架之间的分割线可用鼠标拖拽来调整页面大小，若要禁用，可在相应<frame>处设置属性noresize="noresize"。

另外要注意的是，<frameset>写于<body>之外，或者说，<frameset>与<body>不能同在，除非使用<noframes>，在<noframes>中使用<body>，从而当浏览器无法正确显示框架时显示出<body>中的相应信息：

```
1 <html>
2 <frameset cols="50%,50%">
3   <frame src="index1.html"/>
4   <frame src="index2.html"/>
5   <noframes>
6     <body>您的浏览器无法处理框架！</body>
7   </noframes>
8 </frameset>
9 </html>
```

若要在当前页面不仅仅有框架，还要求有其他元素，可以使用内联框架，这个是写在<body>之中的：

```
<iframe src="URL"></iframe>
```

src可以是本地网页，也可以是网址、图片等等。在src后面可以使用width、height定义框架大小，frameborder设置边框粗细。

以下为与内联框架的互动：（通过按钮使内联框架内显示不同页面）

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p>原来是页面1</p>
5   <iframe src="index.html" name="new_page"></iframe>
6   <p><a href="index2.html" target="new_page">点击打开页面2</a></p>
7 </body>
8 </html>
```

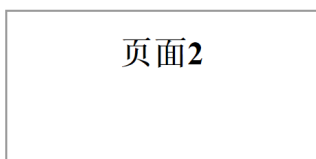
原来是页面1



[点击打开页面2](#)

[https://blog.csdn.net/Zerg\\_Wang](https://blog.csdn.net/Zerg_Wang)

原来是页面1



[点击打开页面2](#)

[https://blog.csdn.net/Zerg\\_Wang](https://blog.csdn.net/Zerg_Wang)

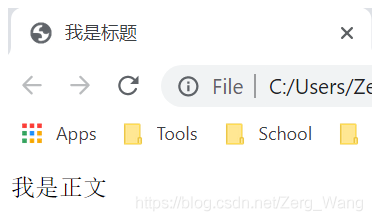
## 头部元素

即<html>中的<head>部分，一般写于<body>之前，提供关于页面本身的信息（如页面标题、默认设置等等）

## <title>元素

规定页面在浏览器标签页所显示的标题，若没有，则会显示本地网页的文件名。

```
1 <html>
2 <head>
3   <title>我是标题</title>
4 </head>
5 <body>
6   <p>我是正文</p>
7 </body>
8 </html>
```



## <base>元素

规定页面上所有链接的默认地址以及默认目标（target）

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <base href="index.html"/>
5   <base target="_blank" />
6 </head>
7
8 <body>
9   <a href="">不写网址也可打开</a>
10 </body>
11 </html>
```

即使在<a>中不设置target，页面也还是会在新标签页打开（因为在<base>）中规定了默认为\_blank。

然而，即使在<base>中规定了默认网址，href却不能省，写成空字符串即可。

若<a>中有具体URL或本地文件，打开网页时会按照<a>中网址来。但有例外：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <base href="https://www.bilibili.com"/>
5   <base target="_blank" />
6 </head>
7
8 <body>
9   <a href="index2.html">不写网址也可打开</a>
10 </body>
11 </html>
```

打开的是不是本地的index2.html，而是http://www.bilibili.com/index2.html，若用绝对地址则不会出现以上情况。

## <link> 元素

定义网页与外部资源之间的关系，常用于连接外部样式表来设计内容显示效果。

## <meta> 元素

定义网页的元数据（metadata），即描述数据的数据（data about data），可使用<meta>定义网页内容简介、作者、修改日期、关键字等，按一定规则定义这些内容可方便搜索引擎对网页的索引。

```

1 <html>
2 <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4     <meta name="author" content="ZergWang" />
5     <meta name="keywords" content="simple, naive" />
6     <meta name="description" content="Just a simple page" />
7 </head>
8 <body>
9     <p>这是一个简单的网页</p>
10 </body>
11 </html>

```

第一个<meta>定义了网页内容的形式和编码方式。

通过http-equiv，可以设置网页的重定向：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta http-equiv="Refresh" content="5;url=index.html" />
5 </head>
6 <body>
7     <p>该网页不存在，5秒后会跳转到首页</p>
8     <p>若未跳转，请点击：<a href="index.html">首页</a></p>
9 </body>
10 </html>

```

该网页不存在，5秒后会跳转到首页

若未跳转，请点击： [首页](#)

## 表单

<form>，用于收集用户输入。

### <input>元素

属性type定义用户的输入类型，有单行的文本输入框，但有两种形式：type="text"的用户可见自己的输入；type="password"的不可见。

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4     <form>
5         <p>请输入用户名： </p>
6         <input type="text" /><br/>
7         <p>请输入密码： </p>
8         <input type="password" />
9     </form>
10 </body>
11 </html>

```

请输入用户名：

123

请输入密码：

...

[https://blog.csdn.net/Zerg\\_Wang](https://blog.csdn.net/Zerg_Wang)

单选按钮 ( type="radio" )

```

1 <form>
2     <input type="radio" name="sex" />男

```

```

3 |      <br> 4 |      <input type="radio" name="sex" />女
5 | </form>

```



复选框 ( type="checkbox" )

```

1 | <form>
2 |   <input type="checkbox" />1
3 |   <br/>
4 |   <input type="checkbox" />2
5 | </form>

```

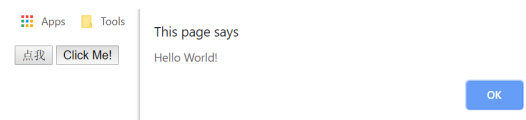


按钮 ( type="button" ) , 这里写了一个点击后弹出“Hello World”提示信息的按钮。实际上, 单独的<button>元素也有, 写法略有不同。

```

1 | <form>
2 |   <button onclick="alert('Hello World!')">点我</button>
3 |   <input type="button" onclick="alert('Hello World!')" value="Click Me!" />
4 | </form>

```

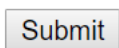


向表单处理程序提交表单的按钮 ( type="submit" ) 。

```

1 | <form>
2 |   <input type="submit" />
3 | </form>

```



<input>不仅在输入方式上多样, 在辅助用户输入方面也有不少功能强大的属性:

value属性: 规定输入字段的初始值。

```

1 | <form>
2 |   <p>请输入</p>
3 |   <input type="text" value="Zerg" />
4 | </form>

```

请输入

readonly属性: 输入框内的内容由value确定, 无法由用户更改。

```
<input type="text" value="Zerg" readonly />
```

disabled属性: 输入框被禁用, 其值不会被提交。disabled下框变成灰色, 内容无法被选中。( readonly下可被选中)



```
<input type="text" value="Zerg" disabled />
```

请输入

Zerg

maxlength属性：规定用户输入内容的最大长度。例如下面设为5，则用户在这个框内只能输入5个字符（无论中英文）。

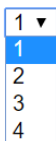
```
<input type="text" maxlength="5" />
```

HTML5为<input>加入了大量输入类型，功能极为强大，也加入多种属性，可对用户的输入进行更为灵活、全面的辅助，此处暂不列举。

## <select>元素

与<input>一样也是输入框，为下拉列表形式。

```
1 <form>
2   <select>
3     <option>1</option>
4     <option>2</option>
5     <option>3</option>
6     <option>4</option>
7   </select>
8 </form>
```



## <fieldset>元素

用于组合表单中的数据，其中<legend>元素用于为<fieldset>定义标题。

```
1 <form>
2   <fieldset>
3     <legend>用户登录</legend>
4     <p>请输入用户名: </p>
5     <input type="text" /><br/>
6     <p>请输入密码: </p>
7     <input type="password" />
8   </fieldset>
9 </form>
```



[https://blog.csdn.net/zerg\\_flying](https://blog.csdn.net/zerg_flying)

## action属性

定义提交表单时执行的动作，若无该属性，则默认为点击submit后打开当前页面。一般所执行的内容通过脚本规定。

```
<form action="action.php">
```

要使用户输入内容能正确提交且被脚本所用，一般获取用户输入的元素<input>、<select>要配有相应的name属性，使得脚本可以通过此进行对应。

## method属性

规定提交表单时所用的 HTTP 方法。有GET和POST两种。

```
<form action="action.php" method="GET">
```

## 参考资料：

<http://www.w3school.com.cn/>