第 3 章

移动开发常用的 HTML5 标签

3.1	HTML5 文件基本标记	03 - 24
3.2	页面主体标签	25 - 26
3.3	列表	27 - 29
3.4	层	30 - 32
3.5	表格	33 - 33
3.6	多媒体	34 - 36
3.7	图像效果	37 - 37
3.8	文件与拖放	38 - 46
◆	本章作业	47 - 47

◆ HTML5 标记/标签

- 是HTML语言中，最基本的单位。
- **作用**：描述页面元素的内容、格式、位置。
- 告诉浏览器如何显示元素内容。
- 大小写无关，推荐使用**小写字母**。
- HTML5 引入了很多新的标记元素。
- HTML5 移除了很多行内设样式的标记，如big、center、font和basefont等，➜ 使用CSS

◆ HTML5 文档结构

```
<!DOCTYPE html>
<html>
    <head>
      ---
    </head>
    <body>
      ---
    </body>
</html>
```

3-1

◆ 头部标签 <head>

- 所有头部元素的容器
- **作用**：文档相关内容的说明、定义、描述
- 头部定义内容，不会在浏览器正文窗口显示出来
- **包含**：元信息、脚本、样式表等。

<title> … </title>

- 文档的标题。
- 提供页面被添加到收藏夹时显示的标题。
- 显示在搜索引擎结果中的页面标题。

<base href="#/" >

- 为所有链接规定默认地址或默认目标（target）；
- 通常情况下，浏览器会从当前文档的**URL**中提取相应的元素，来填写相对URL中的空白。
- 使用 <base>，浏览器使指定的基本URL来解析所有的相对URL。
- 包括 **<a>、<img>、<link>、<form>** 标签中的URL。
- **例如**：

✓

✓ <base href="img/" />

➜

3-2

<link … >

- 定义文档与外部资源之间的关系。
- 常用于连接样式表。

- **例如**:

✓ <link rel="stylesheet" href="c1.css" />

<script --- >

- 用于定义客户端脚本，比如JavaScript。

<style>

- 用于为HTML文档定义样式信息。

```
<style  type="text/css">
    body { background-color: #40E0D0 }
    .c1  { color: blue }
</style>
```

<meta --- >

- **元数据**，提供关于HTML文档的元数据。
- 不会显示在页面上，但是对于机器是可读的；
- 页面描述、关键词、文档的作者、修改时间、其他数据;
- 可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词）或其他 web 服务；
- 搜索引擎会利用name和content属性来索引页面；
- 可以列举多行；
- <meta> 标签永远位于 head 元素内部；
- 元数据总是以**名称**/**值**的形式被成对传递的。

```
<meta name="#1" content="#2" />
```

- name：名称；
- content：内容；

➢ **name 常用取值**

✓ `<meta name="description" content="web" />`

- 告诉搜索引擎 ➜ 本页面的主要内容；

✓ `<meta name="keywords" content="html5" />`

- 告诉搜索引擎 ➜ 本页面的关键字；

✓ <meta name="author" content="小明,xm@xx.com" />

– 告诉搜索引擎 ➜ 本页面的作者；

✓ <meta name="generator" content="HBuilder" />

– 告诉搜索引擎 ➜ 网站的制作软件；

✓ <meta name="copyright" content="小明" />

– 告诉搜索引擎 ➜ 网站的版权所属；

✓ <meta name="viewport" content="width=device-width, initial-scale=1" />

– 说明页面显示相关设置 ➜ 用于设计移动端网页。

✓ <meta name="robots" content="#" />

- 告诉爬虫 ➜ 哪些页面需要/不需要索引；
- 爬虫：一种按照一定规则，自动抓取网页信息的程序。

#= "none"	➜忽略此网页
= "noindex"	➜不索引此网页
= "nofollow"	➜不继续通过此网页的链接索引搜索其它网页
= "all"	➜将索引此网页与链接索引，等价于index，follow
= "index"	➜爬虫索引此网页
= "follow"	➜爬虫继续通过此页链接索引搜索其它网页

✓ <meta name="renderer" content="#" />

– 渲染器 ➜ 指定浏览器默认以何种方式渲染页面；

#= "webkit"	➜默认webkit内核（IE内核）
= "ie-comp"	➜IE兼容模式 指网页在各种浏览器上的显示效果可能不一致而产生浏览器和网页间的兼容问题。 比如：用户名、密码无法输入，…
= "ie-stand"	➜IE标准模式

```
<meta http-equiv="#1" content="#2" />
```

- **equiv**：equivalent ➜ 相等、相当于；
- 相当于http的文件头作用。
- 向浏览器传回一些有用信息，帮助正确地显示网页内容
- 服务器向浏览器发送文档时，会先发送许多名称/值对
- 至少发送一个：content-type : **text/html**，告诉浏览器准备接受一个 Html 文档；

✓ `<meta http-equiv="content-Type" content="text/html;charset=utf-8" />`

- 设定网页字符集，便于浏览器解析与渲染页面；
- HTML4 格式
- HTML5 格式，改为：

`<meta charset="utf-8" />`	➜一种Unicode可变长度字符编码，又称万国码，世界通用语言编码
`charset="GB2312"`	➜简体中文
`charset="BIG5"`	➜繁体中文
`charset="ISO-8859-1"`	➜英文

- ✓ <meta http-equiv="expires" content="#" />
  - 设定网页到期时间，过期后必须到服务器上重新传输
- ✓ <meta http-equiv="pragma" content="#" />
  - 禁止浏览器从本地计算机的缓存中访问页面内容
- ✓ <meta http-equiv="refresh" content="#" />
  - 自动刷新并指向新页面，过期后必须到服务器上重新传输
- ✓ <meta http-equiv="set-cookie" content="#" />
  - 如果网页过期，存在本地的cookies会被自动删除

- ✓ <meta http-equiv="cache-control" content="#" />
  - 指定请求和响应遵循的缓存机制
- ✓ <meta http-equiv="X-UA-Compatible" content="#" />
  - 用于告知浏览器以何种版本来渲染页面
- ✓ <meta http-equiv="imagetoolbar" content="#" />
  - 指定是否显示图片工具栏，false/true
- ✓ <meta http-equiv="Window-target" content="#" />
  - 强制页面在当前窗口以独立页面显示
- ✓ ……

✓ `<meta name="#1" content="#2" scheme="#3" />`

- scheme：方案；
- 指定content属性值遵循的格式

`<meta name="date" content="#" scheme="YYYY-MM-DD" />`

- 指定日期格式；

◆ 主体标签 <body>

- 文档主体，用户看到的内容 ➜ 浏览器信息窗口（正文窗口）。

➢ 作用

- 文档的主体，包含文档的所有内容。
- 用户界面，展示页面功能。

➢ 包含

- 文本、图像、视频、表格、表单，…
- 主体元素，有很多自身属性，如：定义页面文字颜色、背景颜色、背景图像等。

<body> 属性（部分）	
属性	**描述**
text	设定页面文字的颜色
bgcolor	设定页面背景的颜色
background	设定页面的背景图像
bgproperties	设定页面背景图像为固定，不随页面的滚动而滚动
link	设定页面默认的链接颜色
alink	设定鼠标正在单击时的链接颜色
vlink	设定访问过后的链接颜色
leftmargin	设定页面的左边距
…	…

➢ **属性：margin**

– 可以设定文档边缘与窗口的距离

margin: 20px;	➜外边距：上=下=左=右=20
margin: 10px 20px;	➜外边距：上=下=10；左=右=20
margin: 10px 20px 30px;	➜外边距：上=10；左=右=20；下=30
margin-top：20px	➜上边距
margin-left：20px	➜左边距
margin-right：20px	➜右边距
margin-bottom：20px	➜下边距

3-4

- **属性：text**
  - 可以设定整个页面默认文字的颜色

```
<body text="#" />
```

  - 颜色的表示方法
    - ✓ **颜色名**：16个 → aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow
    - ✓ **颜色值**：#RRGGBB / #RGB
    - ✓ **Web安全色**：当年大多数计算机仅支持 256 种颜色，216 种 Web 安全色作为 Web 标准被建议使用。
      - → #000000，#003300，#006600，#00CC00，#00FF00
      - → #000033，#33FF33，#990033，#CCFF99，…

- **属性：bgcolor**
  - 可以设定页面的背景颜色。

- **属性：background**
  - 可以设定页面的背景图片。

background=img_url	
bgproperties="fixed"	→背景图像固定，不随内容滚
bgproperties=""	→背景图像随内容滚

- **标记：** `<img src="#" --- />`
  - 显示图像
- **标记：** `<a href="#1" target="#2"> #3 </a>`
  - 定义超链接
- **标记：** `<!-- 注释内容 -->`
  - 用于在源文档中插入注释

◆ **文字格式**

– 包括字体、对齐、大小属性、颜色属性、设置段落、换行、居中、缩进、水平标记线等

➢ **标记：<font>**

– 规定文本的字体、字体尺寸、字体颜色，➜ 使用样式代替！

✓ **属性**

– face=#	➜文本的字体
– size=#	➜文本的大小

3-5

- ✓ **标记：<p>**
  - 段落
- ✓ **标记：<pre> --- </pre>**
  - 定义预格式化的文本，按原始排列显示
- ✓ **标记：
**
- ✓ **标记：<hr/>**
- ✓ **字符实体：→**

显示	实体
<	&lt;
>	&gt;
&	&amp;
…	…

◆ 列表

✓ 有序列表

```
<ol start=# type=# reversed >
    <li>表项1</li>
    <li> … </li>
</ol>
```

- start= 起始序号
- type = 列表的标记类型 →
- reversed： 降序

type="1"	数字
type="A"	大写字母
type="a"	小写字母
type="I"	大写罗马字母
type="i"	小写罗马字母

✓ 无序列表

```
<ul  type=# >
    <li>表项1</li>
    <li>  ---  </li>
</ul>
```

– type =列表的标记类型 ➡

type="disc"	● 默认
type="circle"	○
type="square"	■

✓ **菜单列表**

```
<menu  type=# label=# >
    <li>菜单1</li>
    <li>  ---  </li>
</menu>
```

- **label** =可见标签
- **type** =菜单类型 → **popup | toolbar**

◆ **标记 **

– 用于页面布局

```
<div class=# >
    ---
</div>
```

– **class** =样式表 ➔ 修饰、格式化、定位

3-6

◆ **标记 <iframe>**

- 创建包含另外一个文档的内联框架（即行内框架）
- 可用于组合其他Html元素的容器

```
<iframe name=# src=# sandbox=# srcdoc=#
        height=# width=# seamless >
    ---
</iframe>
```

– **src**	→规定在 <iframe> 中显示的文档的 URL
– **Sandbox**	→对 <iframe> 内容定义一系列额外的限制
– **srcdoc**	→规定 <iframe> 看起来像是父文档中的一部分
– **seamless**	→看起来像包含文档一部分→没有边框和滚动条

◆ **标记 <layer>**

– 在页面上精确地定位一个层

– 可以出现在文档的任何地方

```
<layer class=# >
    ---
</layer>
```

✓ NetScape定义层的标签 →

◆ **表格** <table>

– 由行、列组成

<table border=# cellpadding=# cellspacing=# >

<tr>

<th> # </th> <td> # </td>

</tr>

<tr> --- </tr>

</table>

– cellpadding →内容与单元边框的空白

– cellspacing →单元格间距（像素）

3-7

◆ **多媒体**

– 页面上的音效、音乐、视频和动画。

➢ **音频 <audio>**

– 声音，支持格式：mp3、wav、ogg

```
<audio controls>
    <source  src=#  type="audio/wav" autoplay loop muted />
    浏览器不支持此标签时，提示文字
</audio>
```

controls →显示控制条	autoplay → 自动播放
loop → 循环播放	muted → 静音

- **视频 <video>**

– 视频，格式：avi、wmv、mpg、mov、rm、swf、flv、mp4、ogg

– 可以使用DOM进行控制

```
<video width=# height=# poster=# controls muted  loop
    preload=# >
    <source  src=#  type="video/mp4" />
    浏览器不支持此标签时，提示文字
</video>
```

– **poster** = **url**	→ 视频播放之前显示的图片(海报)
– **preload**=**none**|**auto**	→ 视频是否预加载

✓ 事件 <video>

– play()	→ 播放
– pause()	→ 暂停
– load()	→ 加载

3-8

◆ **图像 <img>**

– 图像格式：jpg、gif、png、bmp

****

– **src = url**	→ 设置图像的 URL
– **alt** = 文本	→ 在图像无法显示时，替代文本
– **title** = 文本	→ 鼠标在图像上悬浮时，显示的文本

– 图像特效 → 采用CSS实现

◆ 文件对象

– 允许Web应用程序，异步读取存储在用户计算机上的文件内容，使用 File 或 Blob 对象指定要读取的文件或数据。

```
<input  type="file" accept="image/*, .pdf "  multiple
        capture="camera"  />
```

– accept =	→ 可接受的文件类型
– multiple	→ 用户能否多选
– capture = camera|camcorder|microphone	→ 文件来自：相机、摄像机、录音

– 提供了上传文件功能 → 上传文件到服务器。

3-9

➢ 相关对象

– File对象	负责处理文件形式的二进制数据 → 操作本地文件
– FileList对象	File对象的网页表单接口
– FileReader对象	负责将二进制数据读入内存内容
– URL对象	用于对二进制数据生成URL

◆ **拖拽事件**

- 抓取对象以后拖到另一个位置，任何元素都可拖放。
- 拖拽文件时，JS可以监听到，并进行处理。

➢ 拖拽时，鼠标动作触发的事件：

dragstart	鼠标点中元素并开始移动时，就会触发
dragenter	处于拖曳状态的鼠标，第一次进入被赋予该事件的元素时触发
dragleave	拖拽过程中会持续不断地触发
dragover	处于拖曳状态的鼠标移动经过被赋予该事件的元素时触发。会不断触发，要慎用

➢ 拖拽时，**被拖曳元素**会触发的事件：

– **drag**	在拖曳源触发，会不断地触发
– **drop**	在释放元素时触发，拖动上传功能需要注意：
	①drop事件会往父元素冒泡，因此需阻止它冒泡；
	②文件drop后会有下载默认动作，需要阻止默认行为；
	③阻止dragover的默认行为
– **dragend**	拖曳操作结束后触发，不管拖曳操作成功与否

3-10

- ✓ **dataTransfer 对象**
- – 在所有的拖放事件中都提供了一个数据传输对象dataTransfer，主要是用于在源对象和目标对象之间传递数据；
- ✓ **setData(format, data)**
- – 设置拖拽中要传递的数据，向dataTransfer中对象中存入数据。
- ✓ **getData(format)**：获得拖拽事件中传递的数据。
- ✓ **clearData()**：清空dataTransfer中存储的数据。
- ✓ **setDragImage(element, x, y)**
- – 用于在拖放操作过程中，修改鼠标指针所指向的图像。

Event 对象

- 代表事件的状态。
- 比如：事件在发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态，等。

标准Event方法	描 述
initEvent()	初始化新创建的 Event 对象的属性。
preventDefault()	通知浏览器不要执行与事件关联的默认动作。
stopPropagation()	不再派发事件。

标准Event属性	描 述
bubbles	返回布尔值，指示事件是否是起泡事件类型。
cancelable	返回布尔值，指示事件是否可拥可取消的默认动作。
currentTarget	返回其事件监听器触发该事件的元素。
eventPhase	返回事件传播的当前阶段。
target	返回触发此事件的元素（事件的目标节点）。
timeStamp	返回事件生成的日期和时间。
type	返回当前 Event对象表示的事件的名称。

➢ 拖放过程

– 主要步骤:

(1) 设置拖放

```
<img draggable="true" />
```

(2) 拖放目标

```
function drag (ev) {
    ev.dataTransfer.setData("Text", ev.target.id)
```

– 设置被拖数据的数据类型和值

– 数据类型→"Text"，id值→被拖动元素的id

(3) 放到何处

- ondragover 事件规定在何处放置被拖动的数据。

(4) 放置目标

```
function drop (ev) {
    ev.preventDefault();          //避免浏览器对数据的默认处理
    d1=ev.dataTransfer. getData("Text") ;    //获得被拖数据
    ev.target.appendChild(document.getElementById(d1));
                                  //把被拖元素追加到放置元素（目标元素）中
```

本章作业

◆ **课后作业**		**第96页**
1	**选择题：**	(1)(2)(3)(4)(5)
2	**填空题：**	(1)(2)(3)(4)
3	**简答题：**	(1)(2)
◆ **上机练习**		
➢	设计页面：	实验1、2、3
➢	编写程序：	实现拖放（第4题，参照课堂示例）