

# 前端培训

郭碧雪

安徽兮克电子科技有限公司

2018.10.11

# 目录

HTML

CSS

HTML+CSS

JavaScript

# HTML

## 概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# 什么是 HTML

HTML：超文本标记语言(HyperText Markup Language)，是一种建立网页文件的语言，透过标记式的指令 (Tag)，将影像、声音、图片、文字等连结显示出来。

说明：HTML 是一种描述性的标记语言；

HTML 文件是文本文件；

HTML 文件必须使用 html 或 htm 作为文件扩展名。

# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# 浏览器内核

不同的浏览器内核有不同的渲染引擎 (Rendering Engine)。不同浏览器内核对网页编写的语法的解释不同，所以同一网页在不同内核的浏览器里渲染效果也可能不同。

Table: 常见的浏览器内核

内核	开发公司	浏览器
Trident	微软公司	IE/Edge 浏览器 360 浏览器（兼容模式）
Gecko	Mozilla 组织	Firefox
Webkit	苹果公司	Safari 360 浏览器（极速模式）
Blink	谷歌和 Opera	Chrome 浏览器

# HTML

概述

浏览器内核

**WEB 标准**

开发工具

基本结构

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# WEB 标准

WEB 标准是一系列制定网页规范的标准的集合。

由 W3C 万维网联盟组织制定。

官网: <https://www.w3.org/>

WEB 标准分为三方面:

- ▶ 结构化标准语言
- ▶ 表现标准语言
- ▶ 行为标准

# HTML

概述

浏览器内核

WEB 标准

**开发工具**

基本结构

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# 常用开发工具

- ▶ 记事本 ( windows )
- ▶ vi ( Linux )
- ▶ TextEdit ( Mac )
- ▶ Sublime Text
- ▶ Visual Studio Code

下载地址: <https://code.visualstudio.com/Download>

# HTML

概述

浏览器内核

WEB 标准

开发工具

**基本结构**

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# 基本结构 |

```
<!DOCTYPE HTML> 文档类型声明
```

```
<html> 根标签
```

```
    <head>
```

头部信息相关内容

```
    </head>
```

```
<body>
```

页面主体相关内容

```
</body>
```

```
</html>
```

# 基本结构 II

## ▶ 文档声明

- ▶ HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

- ▶ HTML 5 <!DOCTYPE html>

- ▶ W3C 组织的 XHTML 标准

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

- ▶ 一些专业的 HTML 网页编辑器（如 dreamweaver 等），会按照操作自动在 HTML 文档头部生成相应的文档类型声明

## ▶ 头部信息

- ▶ 包含辅助性标签，如下

<meta>: 描述文档属性

<title>: 文档标题

<link>: 定义与外部资源的关系

<script>: 定义客户端脚本

# 基本结构 III

## ▶ 页面主体

### ▶ 代码编写区

<p>: 段落标签

<hr/>: 下拉线标签

<img>: 图片标签

### ▶ 注意

- ▶ HTML 不区分大小写，建议小写
- ▶ 书写的规范，注意缩进

Listing 1: 实例

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Hello World</title>
7  </head>
8  <body>
9      <h1>hello World!</h1>
10 </body>
11 </html>
```

# 基本结构 IV

**hello World!**



# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

**基本语法**

字符设定

常用标签

特殊符号

HTML5 新标签

# 基本语法

- ▶ 标签

标记标签，由尖括号包围的关键词。

如 <html><title>

- ▶ 元素

元素指的是从开始标签到结束标签的所有代码。

如 <title>test</title>

- ▶ 属性

标签可以拥有属性，属性为元素提供附加信息。

如 <a href="http://seekswan.com"> 安徽兮克 </a>

# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

**字符设定**

常用标签

特殊符号

HTML5 新标签

# 字符设定

- ▶ XHTML、HTML4.0.1 设置字符集

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

- ▶ HTML5 设置字符集

```
<meta charset="UTF-8" />
```

# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

字符设定

**常用标签**

特殊符号

HTML5 新标签

# 标签说明

- ▶ < 标签符 > 内容 </标签符 >

如: <p> 段落 </p>

大部分标签是由**开始标签**和**结束标签**所组成的，可以在其中插入其他标签和文字。

- ▶ < 标签符 />

如: <img />、<hr />、<br />

部分标签是**自闭合标签**，没办法在内部插入标签或文字，只能定义自身的一些属性。

# 标题标签

共有 6 个级别的标题标签：`<h1>~<h6>`，重要性依次降低。  
h 是 header 的简称。

Listing 2: 实例

```
1 <body>
2   <h1>这是一级标题</h1>
3   <h3>这是三级标题</h3>
4   <h6>这是六级标题</h6>
5 </body>
```

这是一级标题  
这是三级标题  
这是六级标题

# 段落标签

使用< p >来定义段落。  
p 是 paragraph 的简称。

Listing 3: 实例

```
1 <body>
2   <p>水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。
3     予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，
4       亭亭净植，可远观而不可亵玩焉。</p>
5 </body>
```

水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自  
李唐来，世人甚爱牡丹。予独爱莲之出淤泥而不  
染，濯清涟而不妖，中通外直，不蔓不枝，香远益  
清，亭亭净植，可远观而不可亵玩焉。

# 换行与水平线标签

使用 `<br />` 来定义换行，`br` 是 `break` 的简称。

使用 `<hr />` 来定义水平线，`hr` 是 `horizon` 的简称。

Listing 4: 实例

```
1 <body>
2   <hr />
3   <hr />
4   <br />
5   <hr />
6 </body>
```



# 文本格式化标签

加粗标签，可以使用 `<b>` 或 `<strong>`

斜体标签，可以使用 `<i>` 或 `<em>`

下划线标签 `<u>`， 删除线标签 `<s>`

上标标签 `<sup>`， 下标标签 `<sub>`

Listing 5: 实例

```
1 <body>
2     <strong>我是加粗</strong>
3     <em>我是斜体</em>
4     <u>我是下划线</u>
5     <s>我是删除线</s>
6     我是上标<sup>1</sup>, 我是下标<sub>2</sub>
7 </body>
```

我是加粗 我是斜体 我是下划线 我是删除线 我是上  
标<sup>1</sup>，我是下标<sub>2</sub>

# div 标签 |

div ( division ) 标签，用来为文档内大块的内容提供结构和背景，被称为分隔标签，是最常使用的标签之一。

div 标签内可以放入 <body> 标签的任何内部标签：段落文字、表格、列表、图像等。

使用 div 标签划分区域的代码更加具有逻辑性。

# div 标签 II

Listing 6: 实例

```
1 <body>
2     <!-- 第一首诗 -->
3     <h2>离思</h2>
4     <p>曾经沧海难为水，除却巫山不是云。</p>
5     <!-- 第二首诗 -->
6     <h2>行宫</h2>
7     <p>白头宫女在，闲坐说玄宗。</p>
8
9
10    <!-- 第一首诗 -->
11    <div>
12        <h2>离思</h2>
13        <p>曾经沧海难为水，除却巫山不是云。</p>
14    </div>
15    <!-- 第二首诗 -->
16    <div>
17        <h2>行宫</h2>
18        <p>白头宫女在，闲坐说玄宗。</p>
19    </div>
20 </body>
```

# 列表

- ▶ 有序列表
- ▶ 无序列表
- ▶ 自定义列表

# 有序列表 I

- ▶ 从`<ol>`开始到`</ol>`结束；  
`ol` 是 ordered list 的简称，`<li>`即 list ( 列表项 )；
- ▶ 各个列表项有先后顺序，一般采用数字或作为顺序，默认采用数字排序。

Table: type 属性

|   |                   |
|---|-------------------|
| 1 | 数字 1、2、3……        |
| a | 小写 a、b、c……        |
| A | 大写 A、B、C……        |
| i | 小写罗马数字 i、ii、iii…… |
| I | 大写罗马数字 I、II、III…… |

# 有序列表 II

Listing 7: 实例

```
1 <body>
2     <ol>
3         <li>HTML</li>
4         <li>CSS</li>
5         <li>bootstrap</li>
6         <li>ECMAScript</li>
7         <li>jQuery</li>
8     </ol>
9 </body>
```

1. HTML
2. CSS
3. bootstrap
4. ECMAScript
5. jQuery

# 无序列表 |

- ▶ 从<ul>开始到</ul>结束；  
ul 是 unordered list 的简称。
- ▶ 无序列表在实际项目中最常被使用。
- ▶ 各个列表项没有有先后顺序，默认情况下，无序列表的项目符号是点；

Table: type 属性

|        |           |
|--------|-----------|
| none   | 数字不使用项目符号 |
| disc   | 实心圆，默认值   |
| circle | 空心圆       |
| square | 实心方块      |

# 无序列表 II

Listing 8: 实例

```
1 <body>
2     <ul>
3         <li>HTML</li>
4         <li>CSS</li>
5         <li>bootstrap</li>
6         <li>ECMAScript</li>
7         <li>jQuery</li>
8     </ul>
9 </body>
```

- HTML
- CSS
- bootstrap
- ECMAScript
- jQuery

# 定义列表

- ▶ 定义列表由两部分组成：定义条件和定义描述。
- ▶ 从`<dl>`开始到`</dl>`结束；  
dl 是 definition list 的简称。`<dt>`definition term ( 定义名词 ),`<dd>`definition description ( 定义描述 );

Listing 9: 实例

```
1 <body>
2   <dl>
3     <dt>HTML</dt>
4     <dd>控制网页的结构</dd>
5     <dt>CSS</dt>
6     <dd>控制网页的样式</dd>
7     <dt>JavaScript</dt>
8     <dd>控制网页的行为</dd>
9   </dl>
```

|            |         |
|------------|---------|
| HTML       |         |
|            | 控制网页的结构 |
| CSS        |         |
|            | 控制网页的样式 |
| JavaScript |         |
|            | 控制网页的行为 |

# 图像

图像使用标签定义。

img 是 image 的简称。

<img /> 是**自闭合**标签。

<img> 最重要的属性就是src(source): 指定图像的地址。

其次是alt属性: 设置图像的描述信息, 图像加载出错时显示该信息。

title属性: 设置鼠标移动到图像上的提示信息。

Listing 10: 实例

```
1 
```



# 超链接

超链接（hyperlink），使浏览者在各个独立的页面相互跳转；

超链接有外部链接、内部链接、锚点链接等。

超链接使用[标签定义。](#)

**href**属性：链接地址，点击之后跳转的地址。

**target**属性：跳转方式。

Listing 11: 实例

```
1 <a href="http://seekswan.com" target="_self">兮克</a>
```

Table: target 属性

|         |                 |
|---------|-----------------|
| _self   | 默认方式，在当前窗口打开链接  |
| _blank  | 在一个全新的空白窗口中打开链接 |
| _top    | 在顶层框架中打开链接      |
| _parent | 在当前框架的上一层里打开链接  |

# 锚点链接

锚点可以在同一个页面上的不同位置相互跳转，也就是在不同元素之间跳转。

锚点链接的应用场景：在一个很长的页面，方便让用户在页面不同部分间跳转。

使用方法：加 id 或者 name，推荐使用 id

- ▶ 建立锚点目标：

Listing 12: 实例 1

```
1 <div id="test" name="test"></div>
```

- ▶ 创建跳转到 id="test" 的 div 的锚点：

Listing 13: 实例 2

```
1 <a href="#test"></a>
```



# 表格 I

表格可以更清晰地排列数据，所以在网页制作中应用很多。



## 表格 II

Table: 表格标签

|           |                              |
|-----------|------------------------------|
| <table>   | 定义表格                         |
| <tr>      | table row<br>定义表格的行          |
| <td>      | table data cell<br>定义标准单元格   |
| <caption> | 定义表格标题                       |
| <thead>   | table header<br>定义表格头部       |
| <tbody>   | table body<br>定义表格主体         |
| <tfoot>   | table foot<br>定义表格页脚         |
| <th>      | table header cell<br>定义表头单元格 |

# 表格 III

Table: 表格相关属性

|             |                    |
|-------------|--------------------|
| border      | 表格边框， 默认 0         |
| cellspacing | 单元格与其边框之间的间距， 默认 2 |
| cellpadding | 单元格内容与其边框的间距， 默认 1 |
| width       | 表格宽度               |
| height      | 表格高度               |
| align       | 水平对齐方式             |
| colspan     | 合并列， 用于 td 和 th    |
| rowspan     | 合并行， 用于 td 和 th    |

# 表格 IV

Listing 14: 实例 1

```
1 <table border="1px">
2     <caption>前端课程表</caption>
3     <tr>
4         <th>课程名</th>
5         <th>作用</th>
6     </tr>
7     <tr>
8         <td>HTML</td>
9         <td>控制网页的结构</td>
10    </tr>
11    <tr>
12        <td>CSS</td>
13        <td>控制网页的样式</td>
14    </tr>
15 </table>
```

前端课程表

| 课程名  | 作用      |
|------|---------|
| HTML | 控制网页的结构 |
| CSS  | 控制网页的样式 |

# 表格 V

合并行使用 td 标签的rowspan属性，而合并列则用到 td 标签的colspan属性。

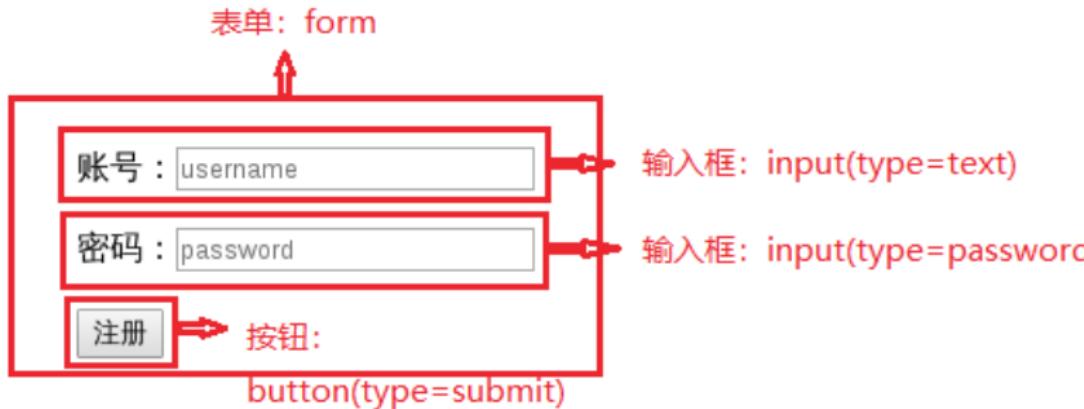
Listing 15: 实例 2

```
1 <table border="1px">
2   <tr>
3     <th>课程名</th>
4     <th>作用</th>
5   </tr>
6   <tr>
7     <td rowspan="2">HTML</td>
8     <td>标记语言</td>
9   </tr>
10  <tr>
11    <td>控制网页的结构</td>
12  </tr>
13  <tr>
14    <td colspan="2">CSS</td>
15  </tr>
16 </table>
```

| 课程名  | 作用      |
|------|---------|
| HTML | 标记语言    |
| CSS  | 控制网页的结构 |
| CSS  |         |

# 表单 |

表单的作用是在客户端收集用户信息；  
文本框、下拉菜单、按钮等是常见的表单元素。



如图，表单由`<form>`标签来定义，`<input />`、`<textarea>`、`<select>`元素包含在内。

## 表单 II

Table: 表单标签

|            |             |
|------------|-------------|
| <form>     | 定义表单        |
| <fieldset> | 将表单内的相关元素分组 |
| <input>    | 定义输入域       |
| <textarea> | 定义文本域       |
| <label>    | 定义控制的标签     |
| <select>   | 定义下拉选择列表    |
| <option>   | 定义下拉列表的选项   |
| <button>   | 定义按钮        |

Table: form 属性

|         |                |
|---------|----------------|
| action  | 传递地址           |
| method  | 传递方法: GET/POST |
| enctype | 提交表单的编码方式      |
| target  | 目标窗口的打开方式      |

# <input> 标签 |

input 标签是自闭合标签，用法 <input type="" />。

Table: type 属性

|             |                                      |
|-------------|--------------------------------------|
| type        | 表单类型                                 |
| placeholder | 输入值之前显示的提示 ( type 为 text/password )  |
| value       | 表单的值 ( type 为 text/search/password ) |
| autofocus   | 是否自动获得焦点                             |
| multiple    | 是否允许输入一个以上的值 ( type 为 file )         |

## <input> 标签 II

Table: input 的 type 属性

|          |                       |
|----------|-----------------------|
| hidden   | 隐藏域，一般用来保存不用显示在页面上的数据 |
| button   | 可点击按钮                 |
| checkbox | 复选框                   |
| file     | 文件上传按钮                |
| password | 密码字段，该字段中的字符被掩码       |
| radio    | 单选框                   |
| checkbox | 复选框                   |
| reset    | 重置按钮                  |
| submit   | 提交按钮                  |
| text     | 单行的输入字段               |

## <textarea> 标签

上面介绍的 <input type="text" /> 是单行文本框。

多行文本框使用 <textarea>。 textarea 的两个重要的属性：

- ▶ cols: 控制输入字符的长度
- ▶ rows: 控制输入字符的行数

1

```
<textarea rows="5" cols="50">多行文本框内容</textarea>
```

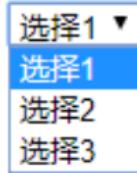
多行文本框内容

# 下拉列表 <select> 标签 |

下列列表的特点跟列表一样，如无序列表适用 <ul> 和 <li> 配合使用，下拉列表使用 <select> 和 <option> 配合。

Listing 16: 实例 1

```
1   <select>
2     <option>选择 1</option>
3     <option>选择 2</option>
4     <option>选择 3</option>
5   </select>
```

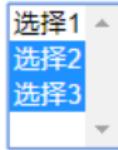


## 下拉列表 <select> 标签 II

下列列表还有一个特有的属性 multiple，跟 input 一样，它也是指定下列列表是否可以多选。

Listing 17: 实例 2

```
1  <select multiple="multiple">
2      <option>选择1</option>
3      <option>选择2</option>
4      <option>选择3</option>
5  </select>
```



# 表单属性

Table: 表单属性

|           |   |
|-----------|---|
| name      | 与服务器通信时使用的名称  |
| value     | 提交的值，不同类型的用法不同<br>"button"、"reset"、"submit" 定义显示文本<br>"text"、"password"、"hidden" 定义输入的初始值<br>"checkbox"、"radio" 定义与输入相关的值 |
| checked   | radio 和 checkbox 默认选中   |
| selected  | select 的 option 默认选中  |
| readonly  | text、password 和 textarea 只读   |
| disabled  | 禁用  |
| maxlength | 控制输入的最大字符数量   |

# 表单综合实例

Listing 18: 实例

```
1 <form action="login.cgi" method="post" style="width:240px">
2   <fieldset>
3     <legend>用户注册</legend>
4     昵称: <input type="text" placeholder="name" /><br />
5     密码: <input type="password" placeholder="password" /><br />
6     性别:
7     <label><input type="radio">男</label>
8     <label><input type="radio">女</label><br />
9     签名: <textarea placeholder="签名"></textarea><br /><br />
10    <button type="submit">注册</button>
11  </fieldset>
12 </form>
```

用户注册

昵称：

密码：

性别： 男  女

签名：

# 块元素和行内元素 |

- ▶ 块元素 (block)：在浏览器显示时独占一行的元素。  
如：div、h1-h6、p、hr等
- ▶ 行内元素 (inline)：在浏览器显示时和其他行内元素显示在同一行的元素。  
如：strong、i、img等
- ▶ 块元素可以容纳行内元素和块元素。
- ▶ 行内元素可以容纳行内元素，无法容纳块元素，否则可能会产生无法预计的效果。
- ▶ 行内元素和块元素可以相互转换。
- ▶ 块元素可以设置高度和宽度，行内元素的高度和宽度由元素的内容所决定。

# 块元素和行内元素 II

Listing 19: 实例

```
1 <body>
2   <div>我是块级元素</div>
3   <p>我是块级元素</p>
4   <h1>我是块级元素</h1>
5   <hr />
6
7   <a>我是行内元素</a>
8   <span>我是行内元素</span>
9   <i>我是行内元素</i>
10  <strong>我是行内元素</strong>
11  <label>我是行内元素</label>
12 </body>
```

我是块级元素

我是块级元素

# 我是块级元素

---

我是行内元素 我是行内元素 我是行内元素 **我是行内元素** 我是行内元素

# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

字符设定

常用标签

**特殊符号**

HTML5 新标签

# 特殊符号对照图

|   |          |   |          |   |          |   |          |   |         |
|---|----------|---|----------|---|----------|---|----------|---|---------|
| ' | &acute;  | © | &copy;   | > | &gt;     | μ | &micro;  | ® | &reg;   |
| & | &amp;    | ° | &deg;    | i | &iexcl;  |   | &nbsp;   | » | &raquo; |
| : | &brvbar; | ÷ | &divide; | ¿ | &iquest; | ¬ | &not;    | § | &sct;   |
| • | &bull;   | ½ | &frac12; | « | &laquo;  | ¶ | &para;   | ” | &uml;   |
| , | &cedil;  | ¼ | &frac14; | < | &lt;     | ± | &plusmn; | × | &times; |
| ¢ | &cent;   | ¾ | &frac34; | — | &macr;   | ” | &quot;   | ™ | &trade; |
|   |          |   |          |   |          |   |          |   |         |
| € | &euro;   | £ | &pound;  | ¥ | &yen;    |   |          |   |         |
|   |          |   |          |   |          |   |          |   |         |
| „ | &bdquo;  | … | &hellip; | · | &middot; | › | &rsaquo; | ¤ | &ordf;  |
| ^ | &circ;   | “ | &ldquo;  | — | &mdash;  | ’ | &rsquo;  | ◦ | &ordm;  |
| † | &dagger; | ‘ | &lsquo;  | – | &ndash;  | , | &sbquo;  | ” | &rdquo; |
| = | &Dagger; | ‘ | &lsquo;  | % | &permil; |   | &shy;    | ˜ | &tild;  |

# 作业 1

## 1. 实现如图所示的文字效果。



## 2. 实现如图所示的列表效果。

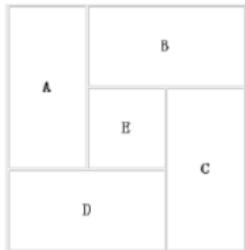
- JavaScript
  - 1. 第一章
    - const
    - let
  - 2. 第二章
    - function
    - object
- Java
  - 1. 第一章
    - class
    - package
  - 2. 第二章
    - private
    - public

## 3. 实现如图所示的列表效果，其中下划线是超链接。



## 作业 II

4. 编写出实现如图所示效果的代码。



5. 编写代码实现如图的出差报销表。

| 姓 名 |    | 职务   | 出差事由  |    | 金 额 |      |    |     |
|-----|----|------|-------|----|-----|------|----|-----|
| 起日  | 止日 |      | 项 目   | 张数 | 金 额 | 项 目  | 天数 | 金 额 |
|     |    |      | 火车费   |    |     | 途中补助 |    |     |
|     |    |      | 汽车费   |    |     | 住勤补助 |    |     |
|     |    |      | 市内交通费 |    |     | 夜间乘车 |    |     |
|     |    |      | 住宿费   |    |     | 其 它  |    |     |
|     |    |      | 邮电费   |    |     |      |    |     |
|     |    |      | 小 计   |    |     | 小 计  |    |     |
| 合 计 |    | (大写) | 仟     | 佰  | 拾   | 元    | 角  | 分   |

# 作业 III

## 6. 编写出实现如图所示的个人信息表。

|                                     |   |  |
|-------------------------------------|---|--|
| 1. 用户登录名和密码                         |   |  |
| 用户名:                                | <input type="text"/>  | 检测用户名 5-15位,请使用英文(a-z,A-Z),数字(0-9)                         |
| 密 码:                                | <input type="password"/>  | 5-15位,请使用英文(a-z,A-Z),数字(0-9);<br>密码不能与登录名相同;标记,难猜.         |
| 再次输入密码:                             | <input type="password"/>  | 两次输入的密码必须一致.   |
| 2. 姓名和联系方式                          |   |  |
| 真实姓名:                               | <input type="text"/>  | <input checked="" type="radio"/> 男 <input type="radio"/> 女 |
| 电子邮箱:                               | <input type="text"/>  | 非常重要!<br>这是客户与您联系的首选方式,请填写真实.                              |
| 固定电话:                               | <input type="text"/>  | 区号+电话号码  |
| 公司所在地:                              | <input type="button" value="北京"/> <input type="button" value="东城"/> |  |
| 街道地址:                               | <input type="text"/>  | 填写县,街道,门牌号   |
| 传真号码:                               | <input type="text"/>  |  |
| 手机号码:                               | <input type="text"/>  |  |
| 邮编(编码):                             | <input type="text"/>  |  |
| 3. 公司名称和主营业务                        |   |  |
| 贵公司名称:                              | <input type="text"/>  | 请填写在工商局注册的公司/商号全称;<br>无商号的个体经营者填写执照上的姓名,如:张三(个体经营)         |
| 你的职称:                               | <input type="text"/>  |  |
| 主营行业:                               | <input type="button" value="电子电工"/>                                 | 请正确选择,您会收到该行业,该产品的供求信息                                     |
| 主营产品/服务:                            | <input type="text"/>  | 3个主要相关产品名/服务名,最少填一个,例如:太阳纸,布料                              |
| 公司网址:                               | <input type="text"/>  |  |
| <input type="button" value="确认提交"/> |   |  |

# HTML

概述

浏览器内核

WEB 标准

开发工具

基本结构

基本语法

字符设定

常用标签

特殊符号

HTML5 新标签

# 语义化标签

HTML5 新增了一些语义化标签使得页面结构化。

Table: HTML5 语义化标签

|           |             |
|-----------|-------------|
| <header>  | 定义文档的头部区域   |
| <footer>  | 定义文档的尾部区域   |
| <nav>     | 定义文档的导航     |
| <section> | 定义文档的节和区段   |
| <article> | 定义页面独立的内容区域 |
| <aside>   | 定义页面的侧边栏内容  |
| <dialog>  | 定义对话框或其实框   |

# 增强型表单

HTML5 拥有多个新的表单 input 输入类型。这些新特性提供了更好的输入控制和验证。

Table: HTML5 新增 input 类型

|          |               |
|----------|---------------|
| color    | 颜色选择器         |
| email    | 邮件地址的输入域      |
| search   | 搜索的输入域        |
| tel      | 电话号码的输入域      |
| url      | url 地址的输入域    |
| date     | 日期选择器         |
| datetime | 选择日期 (UTC 时间) |

# audio |

HTML5 使用 `<audio></audio>` 标签实现音频播放。  
在 `<audio></audio>` 之间嵌入文本，当浏览器不支持时，文本以提示语显示。

audio 在不同浏览器中的兼容情况如下：

|            | IE 9 | Firefox 3.5 | Opera 10.5 | Chrome 3.0 | Safari 3.0 |
|------------|------|-------------|------------|------------|------------|
| Ogg Vorbis |      | √           | √          | √          |            |
| MP3        | √    |             |            | √          | √          |
| Wav        |      | √           | √          |            | √          |

Table: audio 属性

|          |                              |
|----------|------------------------------|
| src      | 定义音频文件的路径                    |
| autoplay | 自动播放                         |
| controls | 显示默认播放控件                     |
| loop     | 循环播放                         |
| preload  | 页面初始化时就加载音频， autoplay 会覆盖此属性 |

# audio II

## Listing 20: 实例

```
1 <body>
2     <audio controls="controls" autoplay="auto">
3         <source src="../media/test.ogg" type="audio/ogg">
4         <source src="../media/test.mp3" type="audio/mpeg">
5         Your browser does not support the &lt;audio&gt; element.
6     </audio>
7 </body>
```



# video |

HTML5 使用 `<video></video>` 标签实现视频的播放。  
`<video>` 目前支持的三种视频格式：ogg、mp4、webM

`<video>` 在各种浏览器上的兼容性情况如下：

| 格式     | IE   | Firefox | Opera | Chrome | Safari |
|--------|------|---------|-------|--------|--------|
| Ogg    | No   | 3.5+    | 10.5+ | 5.0+   | No     |
| MPEG 4 | 9.0+ | No      | No    | 5.0+   | 3.0+   |
| WebM   | No   | 4.0+    | 10.6+ | 6.0+   | No     |

Table: video 常用属性

|          |          |
|----------|----------|
| autoplay | 自动播放     |
| controls | 显示默认播放控制 |
| loop     | 循环播放     |
| width    | 设置播放器宽度  |
| height   | 设置播放器高度  |

# video II

## Listing 21: 实例

```
1 <body>
2     <video width="400" height="240" controls autoplay>
3         <source src="../media/test.mp4" type="video/mp4"></source>
4         <source src="../media/test.ogg" type="video/ogg"></source>
5         <source src="../media/test.webm" type="video/webm"></source>
6         Your browser does not support the &lt;video&gt; element.
7     </video>
8 </body>
```



# svg |

HTML5 支持 svg(可缩放矢量图形), 是基于 XML 语法的图像格式。它的本质是文本文件, 体积小, 不会失真。

svg 代码都放在顶层标签 <svg> 之中。

<svg></svg> 中可以使用下面标签。

Table: svg 常用元素

|          |           |
|----------|-----------|
| line     | 直线        |
| polyline | 折线        |
| rect     | 矩形        |
| circle   | 圆形        |
| ellipse  | 椭圆        |
| polygon  | 多边形       |
| path     | 支持任意路径的定义 |

# svg ||

## Listing 22: 实例

```
1 <body>
2     <svg width="300" height="400">
3         <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" style="fill:
4             white;stroke:#ff6600;stroke-width:4" />
5     </svg>
6     <svg width="300" height="400">
7         <circle cx="100" cy="100" r="50" style="fill:#ff6600">
8             <animate attributeName="r" attributeType="XML" from="50" to="80" begin="0
9                 s" dur="2s" />
10            </circle>
11        </svg>
12        <svg width="300" height="440">
13            <polygon points="122 59,72 205,194 114,49 114,171 205" style="fill: #ff6600;">
14        </svg>
</body>
```



# CSS

概述

引用方式

选择器

优先级

基本样式

盒子模型

文档流

布局与排版

CSS3 新特性

# 什么是 CSS

HTML 控制网页的结构，而 CSS 控制网页的外观。

**层叠样式表** ( Cascading Style Sheet )，也成为级联样式表。

通过一组格式设置规则，控制 HTML 元素在浏览器呈现的样式，  
比如字体大小、颜色、背景色等。

说明：

CSS 是一种标记语言，直接由浏览器解释执行。

CSS 文件是一个文本文件，且必须以 css 为扩展名。

# CSS

概述

引用方式

选择器

优先级

基本样式

盒子模型

文档流

布局与排版

CSS3 新特性

# 引用方式 I

在 HTML 中引用 CSS 的方式一共有三种：

- ▶ 内联样式表
- ▶ 内部样式表
- ▶ 外部样式表

## 1. 内联样式表

使用标签的style属性把 CSS 代码和 HTML 放在同一处。

如下：

Listing 23: 实例

```
1 <body>
2   <p style="color: red; background-color:#000;"> 安徽兮克 </p>
3 </body>
```

# 引用方式 II

## 2. 内部样式表

在 `<head></head>` 中使用 `<style>` 标签将样式表包围起来。

如下：

Listing 24: 实例

```
1 <head>
2   <meta charset="UTF-8">
3   <title>Document</title>
4   <style>
5     p {
6       color: red;
7       background-color: #000;
8     }
9   </style>
10  </head>
11
12  <body>
13    <p> 安徽夸克 </p>
14  </body>
```

# 引用方式 III

## 3. 外部样式表

样式表在单独文件中定义，并且在 `<head></head>` 标签对中使用 `link` 标签来引用。

如下：

Listing 25: 实例

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4     <head>
5         <meta charset="UTF-8">
6         <title>Document</title>
7         <link href="style.css" rel="stylesheet" type="text/css" />
8     </head>
9
10    <body>
11        <p> 安徽夸克 </p>
12    </body>
13
14 </html>
```

为了使代码更加整洁并且易于管理，在正式的项目中一般都使用外部样式表。

# CSS

概述

引用方式

**选择器**

优先级

基本样式

盒子模型

文档流

布局与排版

CSS3 新特性

# 元素的 id 和 class

id 和 class 是 HTML 元素中两个最基本的公共属性。

- ▶ id 属性

id 属性被赋予了标识页面元素的唯一身份。如果一个页面出现了多个相同 id 属性取值，CSS 选择器就会失效。

## Listing 26: 实例

```
1 <p id="xk">安徽夸克</p>
```

- ▶ class 属性

class，顾名思义就是“类”，它的思想跟 c、java 等编程语言的“类”相似。

跟 id 不同，在同一个页面可以设置多个相同的 class，并赋予它相同的样式，从而减少冗余代码，增加效率。

## Listing 27: 实例

```
1 <p class="xk xk1">安徽夸克</p>
```

# 什么是选择器

Listing 28: 实例

```
1 <div>安徽夸克</div>
2 <div>安徽夸克</div>
3 <div>安徽夸克</div>
```

如果想对上面的 3 个 div 设置不同的样式，需要做的就是选中这个 div 然后设置样式。

选择器就是选中目标的方式。

# 基本选择器

- ▶ 通配选择器：\*，选择 HTML 文档内的所有元素。
- ▶ 元素选择器：以元素名作为选择器。  
如：div、p、h1、img 等
- ▶ 类选择器：以元素的 class 值作为选择器。  
如：`<div class="xk_div"></div>`，选择器为.xk\_div
- ▶ ID 选择器：以元素的 id 值作为选择器。  
如：`<div id="xk_div"></div>`，选择器为 #xk\_div

# 常用选择器

- ▶ 后代选择器：作用在后代包含此选择的所有元素。  
如：选择器 `div p`，选择的是 `div` 下所有的 `p` 元素。
- ▶ 子代选择器：作用在子代包含此选择的所有元素。  
如：选择器 `div>p`，选择的是 `div` 下的子 `p` 元素。
- ▶ 相邻选择器：选中该元素的下一个兄弟元素。  
如：选择器 `div+p`，选择的是 `div` 的兄弟 `p` 元素。
- ▶ 群组选择器：选中该元素的下一个兄弟元素。  
如：选择器 `div, p`，选择的是 `div` 元素和 `p` 元素。

# 伪类选择器

伪类用于向某些选择器添加特殊的效果。

语法: selector : pseudo-class property: value 锚伪类

在浏览器中，链接的不同状态都以不同的方式显示，这些状态包括：活动状态，已被访问状态，未被访问状态，和鼠标悬停状态。

如下：

Listing 29: 实例

```
1      a:link {color: #FF0000;}          /* 未访问的链接 */
2      a:visited {color: #00FF00;}        /* 已访问的链接 */
3      a:hover {color: #FF00FF;}          /* 鼠标移动到链接上 */
4      a:active {color: #0000FF;}        /* 选定的链接 */
```

# CSS

概述

引用方式

选择器

**优先级**

基本样式

盒子模型

文档流

布局与排版

CSS3 新特性

# 优先级

如果对同一个元素定义了不同的样式表，浏览器会按照一定的优先级规则去解析样式。

- ▶ 不同的样式表来源，内联样式 > 内部样式 > 外部样式 > 浏览器用户自定义样式 > 浏览器默认样式
- ▶ 后面加载的样式优先级高
- ▶ 不同的样式选择器，ID 选择器 > 类选择器 > 标签选择器
- ▶ 包含层级关系时，近层的优先级高于远层的。
- ▶ !important 的优先级最高

# CSS

概述

引用方式

选择器

优先级

**基本样式**

盒子模型

文档流

布局与排版

CSS3 新特性

# 文字样式 I

类似于 Microsoft Word 和 Office，CSS 也提供了对文字样式的修改。

Table: 文字样式

font-family	字体类型
font-size	字体大小
font-weight	字体粗细
font-style	字体斜体
color	文字颜色

# 文字样式 II

Listing 30: 实例

```
1 <title>Document</title>
2 <style>
3     p {
4         color: #333;
5         font-size: 20px;
6         font-weight: bold;
7         font-style: italic;
8     }
9     </style>
10 </head>
11
12 <body>
13     <p>
14         水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。
15         予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，
16         亭亭净植，可远观而不可亵玩焉。
17     </p>
18 </body>
```

水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，亭亭净植，可远观而不可亵玩焉。

# RGBA 颜色

RGB 是一种色彩标准，由红 (R)、绿 (G)、蓝 (B) 的变化以及相互叠加来得到各种颜色。而 RGBA 在 RGB 的基础上加了一个透明度通道 Alpha。

用法：

Listing 31: 实例

```
1 div{  
2     color: rgb(0, 0, 0);  
3     background-color: rgba(255, 255, 255, .2);  
4 }
```

# 段落样式 I

对文字样式进行调整的同时，往往还伴随着对段落样式的调整。

Table: 段落样式

letter-spacing	字距
line-height	行高
text-align	对齐方式
text-indent	缩进
text-decoration	下划线、删除线、顶划线

# 段落样式 II

Listing 32: 实例

```
1  <style>
2      p {
3          letter-spacing: 5px;
4          line-height: 2;
5          text-align: center;
6          text-indent: 30px;
7          text-decoration: underline;
8      }
9  </style>
10 </head>
11
12 <body>
13     <p>
14         水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。
15         予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，  

16         亭亭净植，可远观而不可亵玩焉。
17     </p>
```

水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，亭亭净植，可远观而不可亵玩焉。

# 边框样式 |

任何块元素和行内元素都可以设置边框属性。

Table: 边框样式

border-width	边框的宽度
border-style	边框的外观
border-color	边框的颜色

也可以border: 1px solid red;

边框也可以分开定义，如：

border-left: 1px dashed blue;

# 边框样式 II

Listing 33: 实例

```
1  <style>
2      p {
3          border: 10px dashed #000;
4      }
5  </style>
6 </head>
7
8 <body>
9     <p>水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。  
    予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，  
    亭亭净植，可远观而不可亵玩焉。</p>
10 </body>
11
12 </html>
```

水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊。自李唐来，世人甚爱牡丹。予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，亭亭净植，可远观而不可亵玩焉。

# 背景样式 |

背景样式主要包括背景颜色和背景图像。

Table: 背景样式

background	设置背景
background-color	设置背景颜色
background-image	设置背景图像
background-repeat	设置背景图像的显示方式
background-position	设置背景图像在元素的位置
background-clip	设置背景的绘制区域
background-attachment	设置背景图像是否随内容而滚动

# 背景样式 II

Listing 34: 实例

```
1 <style>
2     div {
3         background: #000;
4         background-image: url(../../../../img/seeker.png);
5         background-repeat: repeat-x;
6         height: 300px;
7         background-position: 40px 20px;
8     }
9 </style>
10 </head>
11 <body>
12     <div>
13     </div>
14 </body>
```



# 列表样式

列表常用的样式有

Table: 列表常用样式

list-style-type	设置列表项符号
list-style-image	设置列表项符号图像 list-style-image:url();

Table: list-style-type 取值

none	去除列表项默认的符号
disc	无序列表列表项实心圆
circle	无序列表列表项空心圆
square	无序列表列表项实心方块

# 表格样式

前端课程表	
课程名	作用
HTML	控制网页的结构
CSS	控制网页的样式

如图所示，在表格的应用中，往往需要设置单元格之间的间隙，这时就要用到 `table` 元素的 `border-collapse` 样式。

Table: `border-collapse` 取值

separate	默认值，边框分开，不合并
collapse	合并单元格

如果不合并单元格的话，想要设置单元格间的间距，使用 `border-spacing`，如：`border-spacing: 10px;`

# 内容溢出元素样式 I

overflow 属性规定当内容溢出元素框时的处理方式。

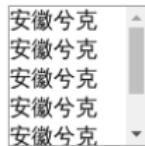
Table: overflow 取值

visible	默认值，内容不会被修剪，会出现实现元素框内
hidden	内容会被修剪，且超出内容不可见
scroll	内容会被修剪，浏览器出现滚动条查看超出内容
auto	如果内容被修剪，浏览器会出现滚动条
inherit	从父元素继承 overflow 属性的值

# 内容溢出元素样式 II

Listing 35: 实例

```
1 <style>
2     .box {
3         width: 100px;
4         height: 100px;
5         overflow: auto;
6         border: 1px solid #999;
7     }
8 </style>
9 </head>
10
11 <body>
12     <div class="box">
13         安徽兮克<br />安徽兮克<br />安徽兮克<br />安徽兮克<br />安徽兮克<br />
14         安徽兮克<br />安徽兮克<br />
15     </div>
```



# CSS

- 概述
- 引用方式
- 选择器
- 优先级
- 基本样式
- 盒子模型**
- 文档流
- 布局与排版
- CSS3 新特性

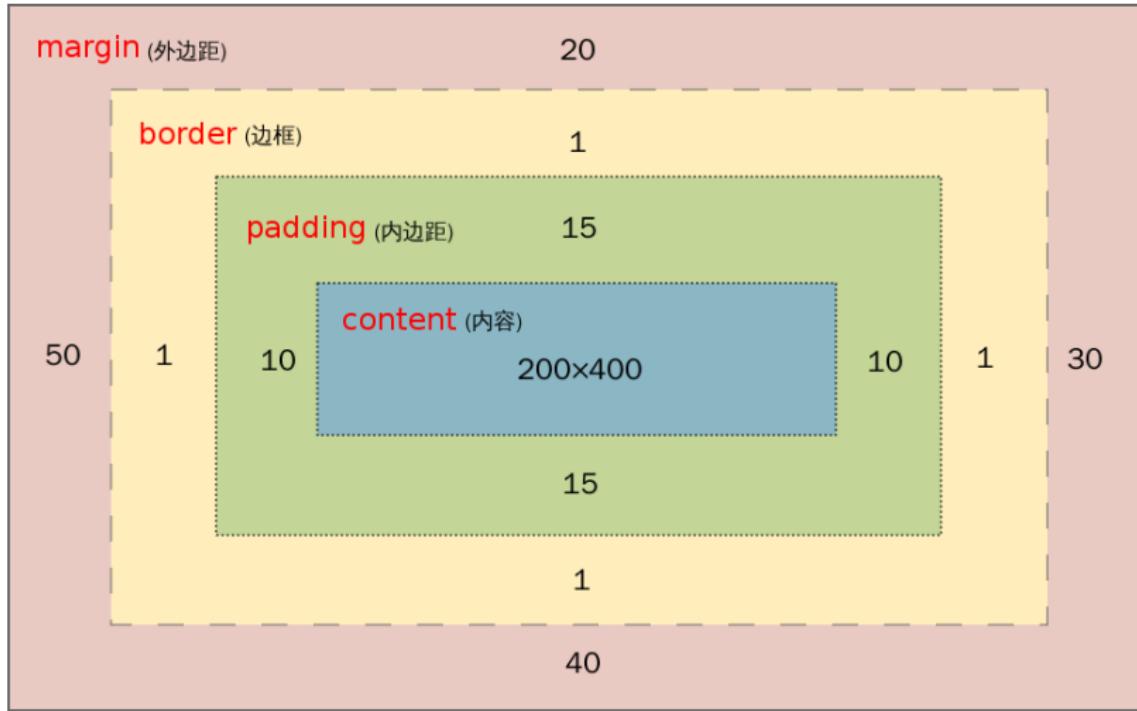
# 盒子模型 |

盒子模型是 html+css 中最核心的基础知识，是进行页面排版和布局的基础。

盒子模型可以理解为，从盒子顶部俯视得到一个平面图，盒子里装的东西，相当于盒子模型的内容（content），东西与盒子直接的空隙，理解为盒子模型的内边距（padding），盒子本身的厚度，相当于盒子模型的的边框（border），盒子与其他盒子的距离相当于盒子模型的外边距（margin）。

如下图所示：

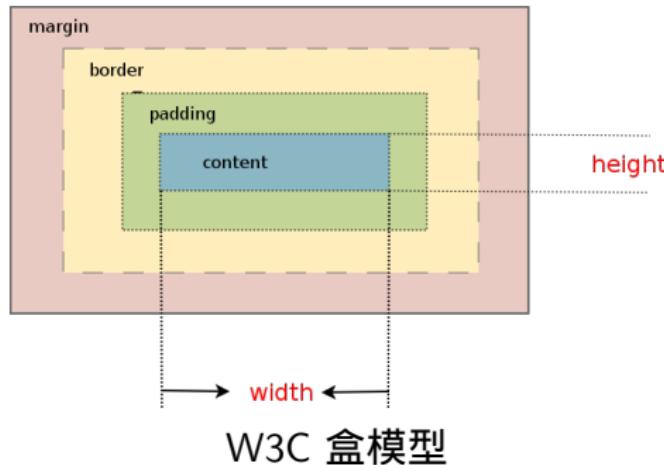
# 盒子模型 II



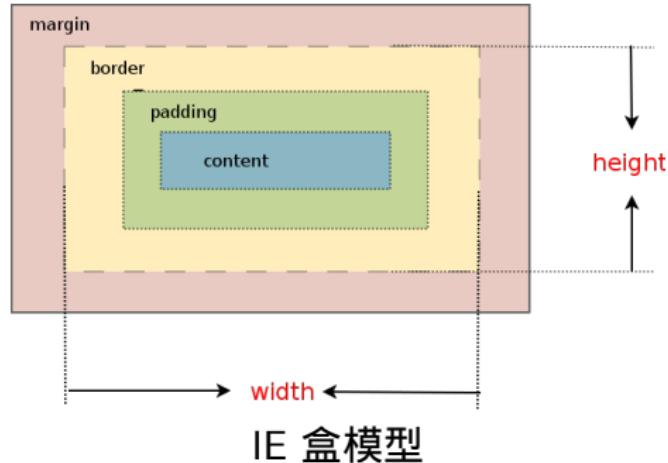
盒子模型示意图

# 盒子模型 III

CSS 盒子模型分为 IE 盒模型和 W3C 盒模型。其实，IE 盒模型是怪异模式下的盒子模型，而 W3C 盒模型是标准模式下的盒子模型。两者的差别如下。



## 盒子模型 IV



从上面两个图可以看出：

W3C 盒模型  $\text{width}/\text{height} = \text{content};$

IE 盒模型  $\text{width}/\text{height} = \text{content} + \text{padding} + \text{border}.$

## 盒子模型 V

为了兼容不同浏览器的兼容性，CSS3 新增了一个属性 box-sizing。

Table: box-sizing 取值

content-box	默认值，遵循的是 W3C 盒模型
border-box	遵循的是 IE 盒模型
inherit	继承父元素的属性

# 盒子模型 VI

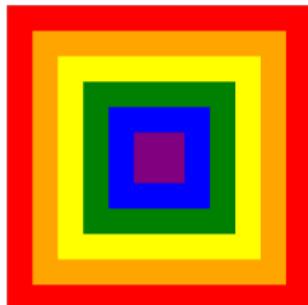
Listing 36: 实例

```
1 <style>
2     body { background-color: #000; }
3     .box {
4         width: 100px; height: 50px;
5         margin-top: 50px;
6         margin-left: 50px;
7         padding: 10px 0 0 10px;
8         border: 10px solid blue;
9         background-color: yellow;
10        box-sizing: content-box;
11    }
12    </style>
13 </head>
14 <body>
15     <div class="box">安徽兮克</div>
16 </body>
```



# 作业 1

- 用盒子模型实现下面效果。



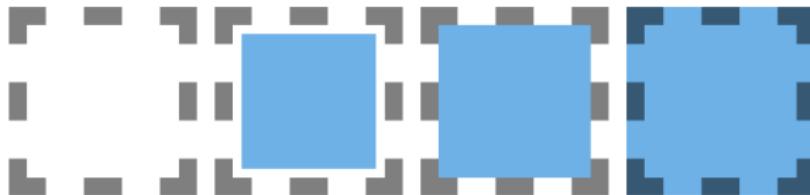
- 用背景样式的方法将左边的图完成右边的效果。

1	2	5
3	4	6
7	8	9

1	9	5	7
2			6
3			8

## 作业 II

3. 用背景样式和盒子模型实现下面四种图形效果。



# CSS

概述

引用方式

选择器

优先级

基本样式

盒子模型

文档流

布局与排版

CSS3 新特性

# 标准文档流 |

标准文档流，就是按照一定规矩排列的，默认的就是元素会从上至下，从左至右排列，块元素独占一行，行内元素会和其他行内元素共享一行。

改变行内元素和块级元素的方法是使用 `display` 属性。

Table: `display` 取值

<code>none</code>	元素不会被显示
<code>block</code>	元素显示为块级元素
<code>inline</code>	元素显示为行内元素
<code>inline-block</code>	行内块元素
<code>inherit</code>	继承父元素的 <code>display</code> 属性

# 标准文档流 II

Listing 37: 实例

```
1   .box {
2       width: 50px;
3       height: 50px;
4       font-size: 16px;
5       color: #fff;
6       background-color: #000;
7       text-align: center;
8       line-height: 50px;
9       display: inline-block;
10      }
11     .box_container {
12         font-size: 0;
13     }
14   </style>
15 </head>
16 <body>
17   <div class="box_container">
18     <div class="box box1">1</div>
19     <div class="box box2">2</div>
20     <div class="box box3">3</div>
21   </div>
```

1      2      3

# 脱离标准文档流布局

为了更加随心所欲的控制页面的布局，有时候需要脱离标准文档流。

css 提供了两种脱离标准文档流的方式：浮动布局，定位。

# CSS

概述

引用方式

选择器

优先级

基本样式

盒子模型

文档流

**布局与排版**

CSS3 新特性

# 浮动布局 I

浮动可以理解为让某个 `div` 元素脱离标准流，漂浮在标准流之上，跟标准流不在一个层次上。

使用`float`来定义浮动。

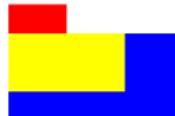
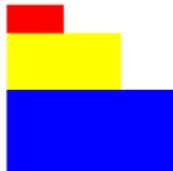
Table: `float` 取值

left	向左浮动
right	向右浮动
none	不浮动

# 浮动布局 II

Listing 38: 实例

```
1  <style>
2      .div1{
3          width: 60px; height: 30px; background-color: red;
4      }
5      .div2{
6          float: left; width: 120px; height: 60px; background-color: yellow;
7      }
8      .div3{
9          width: 180px; height: 90px; background-color: blue;
10     }
11    </style>
12 </head>
13 <body>
14     <div class="div1"></div>
15     <div class="div2"></div>
16     <div class="div3"></div>
17 </body>
```



## 浮动布局 III

在大部分的应用场景下，需要的效果是元素浮动在左侧或右侧但是又不会漂浮在下一个元素上面。这时就需要清除浮动。

`clear: both;` 清除浮动用于浮动元素的下一个元素上面，或者常常使用`:after` 伪类。



# 浮动布局 IV

Listing 39: 实例

```
1  <style>
2      .div1{
3          width: 60px; height: 30px; background-color: red;
4      }
5      .div2{
6          float: right; width: 120px; height: 60px; background-color: yellow;
7      }
8      .div3{
9          clear: both; width: 180px; height: 90px; background-color: blue;
10     }
11  </style>
12 </head>
13 <body>
14     <div class="div1"></div>
15     <div class="div2"></div>
16     <div class="div3"></div>
17 </body>
```



# 浮动布局 V

Listing 40: 文字环绕

```
1 <style>
2     .float>img {
3         float: left; width: 20px;
4     }
5 </style>
6 </head>
7 <body>
8     <div class="float">
9         
10        <p>
11            安徽兮克 —— 安徽兮克 —— 安徽兮克 —— 安徽兮克 —— 安徽兮克 —— 安徽兮克
12        </p>
13    </div>
14 </body>
```



安徽兮克——安徽兮克——安徽兮克——安徽兮  
克——安徽兮克——安徽兮克——安徽兮克——  
安徽兮克——安徽兮克——安徽兮克——安徽兮



安徽兮克——安徽兮克——安徽兮克——安徽  
兮克——安徽兮克——安徽兮克——安徽兮克  
——安徽兮克——安徽兮克——安徽兮克——安  
徽兮克

# 浮动布局 VI

Listing 41: float 布局

```
1 <style>
2     .float{
3         width: 50px; height: 50px; color: #333;
4         line-height: 50px; text-align: center;
5         border: 1px solid #666;
6     }
7     .float1, .float2{
8         float:left;
9     }
10    .float3{
11        float: right;
12    }
13 </style>
14 </head>
15 <body>
16     <div class="float float1">1</div>
17     <div class="float float2">2</div>
18     <div class="float float3">3</div>
19 </body>
```



# 定位 I

使用 **position** 来定义定位。

Table: position 取值

static	默认值，没有定位
fixed	固定定位，相对于浏览器窗口进行定位
relative	相对定位，相对于正常位置进行定位
absolute	绝对定位，相对于 static 之外的元素进行定位
inherit	从父元素继承

# 定位 II

Listing 42: position: relative

```
1  <style>
2      .box{ height: 200px; }
3      .relative{
4          width: 50px; height: 50px; background-color: #000;
5          position: relative; top: 10px; left: 10px;
6      }
7  </style>
8  </head>
9  <body>
10     <div class="box">
11         <div class="relative"></div>
12     </div>
13 </body>
```



# 定位 III

Listing 43: position: fixed

```
1 <style>
2     .box{ height: 1000px; }
3     .fixed{
4         width: 50px; height: 50px; background-color: #000;
5         position: fixed; top: 10px; left: 10%;
6     }
7 </style>
8 </head>
9 <body>
10    <div class="box">
11        <div class="fixed"></div>
12    </div>
13 </body>
```



# 定位 IV

Listing 44: position: absolute

```
1 <style>
2     .box{
3         width: 100px; height: 100px; background-color: #000;
4         position: relative;
5     }
6     .absolute{
7         color: #fff;
8         position: absolute; right: 0;bottom: 0;
9     }
10    </style>
11 </head>
12 <body style="background-color:#eee;">
13     <div class="box">
14         <div class="absolute">absolute</div>
15     </div>
16 </body>
```



# CSS

- 概述
- 引用方式
- 选择器
- 优先级
- 基本样式
- 盒子模型
- 文档流
- 布局与排版
- CSS3 新特性

# CSS3 新增选择器 |

CSS3 新增了属性选择器：以元素的属性作为选择器。

如：选择器为 `[name="xk_check"]`，选择的是所有 name 为 `xk_check` 的元素。

Table: CSS3 新增结构伪类

<code>:empty</code>	选择含有空元素的元素
<code>:not()</code>	选择除某个元素之外的所有元素
<code>:nth-child(n)</code>	选择第 n 个子元素
<code>:nth-child(2n/even)</code>	选择第偶数个子元素
<code>:nth-child(2n+1/odd)</code>	选择第奇数个子元素
<code>:nth-last-child(n)</code>	选择倒数第几个元素
<code>:first-child</code>	选择第一个子元素
<code>:last-child</code>	选择最后一个子元素

# CSS3 新增选择器 II

Table: CSS3 新增表单伪类

:checked	选择选中的表单元素
:enable	选择启用的表单元素
:disabled	选择禁用的表单元素
:focus	选择元素输入后具有焦点

Table: CSS3 新增伪类元素

::before	在元素之前插入内容
::after	在元素之后插入内容

# CSS3 新增选择器 III

Listing 45: :empty 选择器实例

```
1 <style>
2     table,tr,td {
3         border:1px solid #eeeeee;
4     }
5     td {
6         width:60px;height:60px;
7         line-height:60px;color: #ffffff;
8         text-align:center;
9         background-color: #ecb22e;
10    }
11    td:empty{
12        background-color: #000;
13    }
14 </style>
15 </head>
16 <body>
17     <table>
18         <tr><td>1</td><td>2</td></tr>
19         <tr><td>3</td><td></td></tr>
20     </table>
```

1	2
3	

# CSS3 新增选择器 IV

## Listing 46: ::after 选择器实例

```
1 <style>
2     .box {
3         position: relative;
4         width: 100px; height: 30px;
5         background-color: #e3eaea;
6     }
7     .box::after {
8         position: absolute;
9         bottom: 0;
10        left: 40px;
11        content: "";
12        bottom: -8px;
13        border-left: 8px solid transparent;
14        border-right: 8px solid transparent;
15        border-top: 8px solid #f1c909;
16    }
17 </style>
18 </head>
19 <body>
20     <div class="box">安徽兮克</div>
21 </body>
```

安徽兮克

# CSS3 浏览器私有前缀

由于 CSS3 很多属性尚未成为 W3C 标准的一部分，因此每种内核的浏览器都只能识别带有搜有前缀的 CSS3 属性。

Table: CSS3 浏览器私有前缀

-webkit-	chrome、safari
-moz-	firefox
-ms-	IE
-o-	opera

# CSS3 新特性 |

## ▶ 透明度

CSS3 使用 opacity 属性设置元素的不透明程度。

语法： opacity: value|inherit;

value: 规定不透明度。从 0.0 (完全透明) 到 1.0 (完全不透明)。默认为 1。

Listing 47: 实例

```
1 <style>
2     img{
3         opacity: .2;
4         filter:alpha(opacity=20); /* 针对 IE8 以及更早的版本 */
5     }
6 </style>
7 </head>
8 <body>
9     
```



# CSS3 新特性 II

## ▶ 圆角

CSS3 使用 border-radius 属性设置边框圆角。

Listing 48: 实例

```
1 <style>
2     div{
3         border-radius: 10px;
4         border-radius: 10px 20px;
5         border-radius: 10px 5px 20px 15px;
6         border-top-left-radius: 10px
7     }
8 </style>
```

圆角也可以通过百分比来控制。

## ▶ 阴影

### ▶ 边框阴影

CSS3 使用 border-shadow 属性设置边框阴影。

语法: box-shadow: h v blur spread color inset;

其中 h-水平阴影值；v-垂直阴影值，可为负值；blur-模糊距离，可选；spread (可选)，表示阴影尺寸；color-阴影的颜色，可选；inset-外阴影或内阴影，可选，默认外阴影。

# CSS3 新特性 III

## Listing 49: 实例

```
1 <style>
2     .box{
3         width: 100px; height: 100px; display: inline-block;
4         border: 1px solid #000;
5     }
6     .box1{
7         box-shadow: 15px 5px 20px red;
8         -webkit-box-shadow: 15px 5px 20px red;
9         -moz-box-shadow: 15px 5px 20px red;
10        -o-box-shadow: 15px 5px 20px red;
11    }
12    .box2{
13        box-shadow: 15px 5px 20px red inset;
14        -webkit-box-shadow: 15px 5px 20px red inset;
15        -moz-box-shadow: 15px 5px 20px red inset;
16        -o-box-shadow: 15px 5px 20px red inset;
17    }
18 </style>
```



# CSS3 新特性 IV

## ▶ 字体阴影

CSS3 使用 text-shadow 属性设置字体阴影。

语法: text-shadow: h v blur color;

其中 h-水平阴影值; v-垂直阴影值; blur-模糊距离, 可选;  
color-阴影的颜色, 可选。

Listing 50: 实例

```
1 <title>Document</title>
2 <style>
3     .box{
4         text-shadow: 10px 6px 2px #333;
5         -webkit-text-shadow: 10px 6px 2px #333;
6         -moz-text-shadow: 10px 6px 2px #333;
7         -o-text-shadow: 10px 6px 2px #333;
8     }
9     </style>
10 </head>
11
12 <body>
13     <div class="box">安徽兮克</div>
```

安徽兮克  
安徽兮克

# CSS3 新特性 V

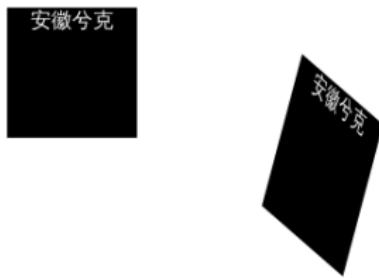
## ▶ 变形

CSS3 使用 transform 属性设置变形效果，如缩放（scale）、定位（translate）、倾斜（skew）、旋转（rotate）等。

Listing 51: 实例

```
1 <style>
2     .box{
3         width: 100px; height:100px; background: #000; color: #fff;
4         text-align: center; display: inline-block; margin-right: 100px;
5     }
6     .box2{
7         -webkit-transform: rotate(9deg) scale(.7, 1.2) translate(0px, 60px)
8             skew(-10deg, 20deg);
9         -moz-transform: rotate(9deg) scale(.7, 1.2) translate(0px, 60px)
10            skew(-10deg, 20deg);
11         -ms-transform: rotate(9deg) scale(.7, 1.2) translate(0px, 60px)
12            skew(-10deg, 20deg);
13         -o-transform: rotate(9deg) scale(.7, 1.2) translate(0px, 60px) skew
14             (-10deg, 20deg);
15         transform: rotate(9deg) scale(.7, 1.2) translate(0px, 60px) skew
16             (-10deg, 20deg);
17     }
18 </style>
19 </head>
20 <body>
21     <div class="box box1">安徽夸克</div>
22     <div class="box box2">安徽夸克</div>
```

# CSS3 新特性 VI



## ▶ 动画

CSS3 的 animation 属性可以用来定义动画。

animation 属性是一个简写属性，用于设置六个动画属性。

Table: 动画属性

animation-name	动画名称，和 keyframe 绑定
animation-duration	规定完成动画的时间
animation-timing-function	规定动画的速度曲线
animation-delay	规定动画开始之前的延迟
animation-iteration-count	规定动画应该播放的次数
animation-direction	规定是否轮流反向播放

# CSS3 新特性 VII

Table: 过渡效果的时间曲线

ease	默认值，慢速开始，中间变快，慢速结束
linear	匀速运动
ease-in	慢速开始
ease-out	慢速结束
ease-in-out	慢速开始，慢速结束
cubic-bezier(n, n, n, n)	自定义贝塞尔时间曲线，数值为 0-1

# CSS3 新特性 VIII

## Listing 52: 实例

```
1 <title>Document</title>
2 <style>
3   a {
4     font-size: 12px;
5     transition: font-size .4s ease;
6   }
7
8   a:hover {
9     font-size: 18px;
10  }
11 </style>
12 </head>
13
14 <body>
15   <a>安徽夸克</a>
```

CSS3 新增属性 @keyframes 定义关键帧。

## Listing 53: 语法

```
1 两帧动画：
2   @keyframes {
3     from{设置开始属性} to{设置结束属性}
4   }
5 多帧动画：
6   @keyframes{
7     0%{设置属性} 50%{设置属性} 100%{设置属性}
8   }
```



# CSS3 新特性 IX

## Listing 54: 实例

```
1   .loading {
2     -moz-animation: spin 1.6s infinite linear;
3     -o-animation: spin 1.6s infinite linear;
4     -webkit-animation: spin 1.6s infinite linear;
5     animation: spin 1.6s infinite linear;
6   }
7   @-moz-keyframes spin {
8     0% { -moz-transform: rotate(0deg) }
9     100% { -moz-transform: rotate(359deg) }
10  }
11  @-webkit-keyframes spin {
12    0% { -webkit-transform: rotate(0deg) }
13    100% { -webkit-transform: rotate(359deg) }
14  }
15  @-o-keyframes spin {
16    0% { -o-transform: rotate(0deg) }
17    100% { -o-transform: rotate(359deg) }
18  }
19  @-ms-keyframes spin {
20    0% { -ms-transform: rotate(0deg) }
21    100% { -ms-transform: rotate(359deg) }
22  }
23  @keyframes spin {
24    0% { transform: rotate(0deg) }
25    100% { transform: rotate(359deg) }
26  }
27  </style>
28 </head>
29 <body>
30   
```

# CSS3 新特性 X

## ▶ 演变

CSS3 背景渐变可以让你在两个或多个颜色中平稳的过渡。  
CSS3 定义了两种类型的渐变：

### ▶ 线性渐变

CSS3 使用 `linear-gradient` 属性设置向下/向右/向左/向右/对角的线性渐变。

语法：`background: linear-gradient(direction, color1, color2);`

### Listing 55: 实例

```
1   .box {  
2       width: 400px;  
3       height: 200px;  
4       display: inline-block;  
5       margin-right: 100px;  
6   }  
7   .box1 {  
8       background: red;  
9       background: -moz-linear-gradient(top, red, blue);  
10      background: -webkit-linear-gradient(top, red, blue);  
11      background: -o-linear-gradient(top, red, blue);  
12      background: linear-gradient(to bottom, red, blue);
```

# CSS3 新特性 XI

```
13         filter: progid:DXImageTransform.Microsoft.gradient(
14             startColorstr='red', endColorstr='blue', GradientType=0
15         );
16     }
17     .box2 {
18         background: -moz-linear-gradient(180deg, red, blue);
19         background: -webkit-linear-gradient(-180deg, red, blue);
20         background: -o-linear-gradient(180deg, red, blue);
21         background: linear-gradient(180deg, red, blue);
22     }
23     </style>
24 </head>
25
26 <body>
    <div class="box box1"></div>
    <div class="box box2"></div>
```



重复的线性渐变：repeating-linear-gradient

## CSS3 新特性 XII

- ▶ 径向渐变

语法: background: radial-gradient(position, shape size,  
start-color, ..., last-color);

更多实例: <https://www.zhangxinxu.com/wordpress/2017/11/css3-radial-gradient-syntax-example/>

重复的线性渐变: repeating-linear-gradient

# 作业

- 完成下面斑马背景的表格。

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

- 做出如下新闻列表效果。

[http://seekswan.com/web\\_news.htm](http://seekswan.com/web_news.htm)

- 完成知识中心的鼠标划过的动态效果。

[http://seekswan.com/web\\_support.htm](http://seekswan.com/web_support.htm)

- 实现百度首页的布局

# HTML+CSS 综合实例



## 轻巧的存储解决方案

不需专属硬盘会员，不需牺牲性能换取速度，大量且无拘无束的数据存储都可轻松达成。  
一台专业NAS抵过数十台U盘。



### 集中存储 智能备份

您可以将Windows/Mac/Linux平台的  
文件备份至NAS集中管理、云端同步。

备份文件  
增量文件  
差异文件  
本地文件  
远程文件



### 设置不同的访问和读写权限

智能为不同的用户分配不同的存储空间。读写否  
权限。成为账号对称授权，管理更方便。

权限  
账号  
授权

# 作业

## 1. 完成案例的布局

# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

文档对象

事件处理

## 概述

学习 JavaScript 之前，了解一下什么是 ECMA。

<http://www.ecma-international.org>

ECMAScript 制定了脚本语言的规范，JavaScript 是 ECMAScript 规范的一种实现。

JavaScript 是世界上最流行的脚本语言，网页的交互逻辑都是它控制的。

JavaScript 使网页由静态转变为动态。

注意：

JavaScript 文件是一种文本文件，并以 js 为后缀名。  
JavaScript 区分大小写。

# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

文档对象

事件处理

# 引用方式 I

JavaScript 有下面两种引用方式。

- ▶ 内部引用

Listing 56: 实例

```
1 <title>Document</title>
2 <script type="text/javascript">
3     var id = "xk";
4 </script>
5 </head>
6 <body>
7     <div id="xk">测试</div>
8
9     <script type="text/javascript">
10        document.getElementById(id).onclick = function(){
11            alert(1);
12        }
13    </script>
14 </body>
```

Javascript 代码可以嵌入文档的任意位置。

# 引用方式 II

## ▶ 外部引用

Listing 57: 实例

```
1   <script src="common.js" type="text/javascript"></script>
2 </head>
3 <body>
4   <div id="xk">测试</div>
5   <script src="index.js" type="text/javascript"></script>
6 </body>
```

为了使代码更加整洁且易于管理，在正式的项目中一般都使用外部调用的方法。

# JavaScript

概述

引用方式

**基本语法**

流程控制

函数

对象

浏览器对象

文档对象

事件处理

# 标识符

标识符，也就是一个名称，在变量和函数等都需要定义一个名称来供使用。

注意：

第一个字符必须是字母/下划线(\_)/美元符号(\$)。

不能包含空格、加号、等于号等特殊符号。

不能和 JavaScript 中用于其他目的的关键字同名。

# 关键字

关键字指的是在 JavaScript 语言中有特定含义，是 JavaScript 语法中的一部分的那些字。

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

\* 标记的关键字是 ECMAScript5 中新添加的。

# 常量

常量，指不能改变的量，常量值从定义时固定，到程序结束。

常量主要用于为程序提供固定精准的值，如数值、真 (true)，假 (false) 等。

ES6 之前并没有定义声明常量的方式，ES6 标准中引入了新的关键字 const 来定义常量。

语法：const 常量名 = 值

Listing 58: 声明常量

```
1 const a = 5;
```

注意：

Javascript 中每行结尾的分号可有可无，最好的编程习惯是加上分号。

# 变量

变量，值在程序运行的过程中，其值是可以改变的。

在变量使用之前需要先声明变量。声明变量使用关键字var

语法：

1. 声明：var 变量名；

2. 赋值：变量名 = 值；

或者声明变量并初始化值：

var 变量名 = 值；

Listing 59: 定义变量

```
1 var xk = "xk";
```

Listing 60: 定义多个变量

```
1 var xk1 = "xi",  
2     xk2 = "ao",  
3     xk3 = "yan";
```

良好的编程习惯是，在代码开始处，统一对需要的变量进行声明。



# 变量的作用域

变量的作用域指该变量在程序中的有效范围，也就是定义变量的区域。

在 JavaScript 中，根据变量的作用域，可以将变量分为全局变量和局部变量。

全局变量：在主程序中定义，有效范围为定义开始到程序结束。

局部变量：在函数中定义，有效范围只在函数内。

Listing 61: 全局变量和局部变量

```
1 var xk_g_test = "test"; // 全局变量
2 function xkTest() {
3     var test = "test"; // 局部变量
4 }
```

# 数据类型 |

基本数据类型如下：

- ▶ 数字型 ( Number )

- ▶ 整形

- 如：0, 1, 2, 3, 4...

- ▶ 浮点型

- 如：0.1, 0.20, 2.300...

## Listing 62: 实例

```
1 var number = 1, number_float = 0.12;
```

- ▶ 字符串型 ( String )

- 如："a", "xike", "xk\_abc" ...

## Listing 63: 实例

```
1 var string1 = "xk", string2 = 'x_k';
```

字符串类型赋值时使用单引号或双引号都可。

## 数据类型 II

- ▶ 布尔类型 ( Boolean )  
只有两个值： true 和 false

Listing 64: 实例

```
1 var bool1 = true, bool2 = false;
```

- ▶ 对象

比如 String、 Date、 Array 等等。

对象只是带有属性和方法的特殊数据类型。

# 数据类型 III

特殊数据类型如下：

- ▶ 空值 (null)

JavaScript 中的关键字 null，表示空值，系统不会给它分配内存空间。

null 不等于"" 或 0。

Listing 65: 实例

```
1 var empty = null;
```

- ▶ 未定义值 (undefined)

如果一个变量虽然已经声明了，但是并没有进行赋值，那么这个变量的数据类型就是 undefined，表示这是一个未定义数据类型的变量。

# 数据类型 IV

## ▶ 转义字符

JavaScript 提供一些特殊符号的转义字符。

Table: 转义字符

\b	退格
\f	走纸换页
\n	换行
\r	回车
\t	横向跳格 (制表符)
\'	单引号
\"	双引号
\\	反斜杠

# 运算符 I

## ▶ 算数运算符

Table: 算数运算符

+	加
-	减
*	乘
/	除
%	求余
++	自增
--	自减

Listing 66: 实例

```
1 document.write(1 + 2);
2 document.write("安徽" + "兮克");
```

算术运算符两边都是数值，若“+”运算中存在字符或字符串，会自动将数值型转成字符串类型并将字符串拼接。

# 运算符 II

## ▶ 比较运算符

对操作数进行比较，返回 true 或 false

Table: 算数运算符

<	小于
>	大于
<=	小于等于
>=	大于等于
==	是否等于
!=	是否不等于

# 运算符 III

## ▶ 赋值运算符

Table: 赋值运算符

=	等于, 如 $a=b$
+=	加等于, 如 $a+=b$ 等价于 $a=a+b$
-=	减等于, 如 $a-=b$ 等价于 $a=a-b$
*=	乘等于, 如 $a*=b$ 等价于 $a=a*b$

## ▶ 逻辑运算符

往往用于和比较运算符一起使用在 if、while 等语句中

Table: 赋值运算符

&&	与, 如 $a < b \&\& b > c$ 表示 $a < b$ 为真且 $b > c$ 为真
	或, 如 $a < b    b > c$ 表示 $a < b$ 为真或 $b > c$ 为真
!	非, 如 $!a$ 表示若 $a$ 真则返回 false, 若 $a$ 假则返回 true

# 运算符 IV

## ▶ 条件运算符

用于条件的赋值运算。

语法：变量 = (条件) ? 值 1: 值 2

如果条件为真则变量等于值 1，否则等于值 2

### Listing 67: 实例

```
1 var str = (age < 18) ? "未成年人" : "成人";
```

## ▶ typeof 运算符

用于返回操作数的数据类型。

### Listing 68: 实例

```
1 var str1 = "未成年人";
2 document.write(typeof (str1));
```

# JavaScript

概述

引用方式

基本语法

**流程控制**

函数

对象

浏览器对象

文档对象

事件处理

# 流程控制

- ▶ 顺序结构

JavaScript 代码是按照从上到下、从左到右的顺序执行。

- ▶ 选择结构

代码按照指定的逻辑条件来决定执行的顺序。

- ▶ 循环结构

代码按照指定的逻辑条件来决定是否重复执行某一段程序。

# 循环结构 I

下面的常用的集中循环结构的方法：

- ▶ while 语句

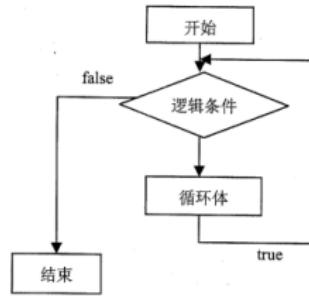
用法是：

```
while(条件表达式语句)
```

```
{
```

执行语句块；

```
}
```



# 循环结构 II

Listing 69: 实例

```
1 var i = 0, sum = 0;
2 while (i < 11) {
3     sum += i;
4     i++
5 }
```

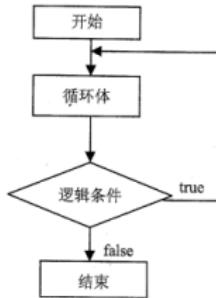
- ▶ do while 语句

用法是：

```
do
{
    执行语句块;
}
```

while(条件表达式语句);

# 循环结构 III



Listing 70: 实例

```
1 var i = 0, sum = 0;
2 do {
3     sum += i;
4     i++;
5 } while (i < 11)
```

# 循环结构 IV

## ▶ for 语句

用法是：

for(初始化表达式; 循环条件表达式; 循环后的操作表达式)

{

    执行语句块;

}

Listing 71: 实例

```
1 var sum = 0;  
2 for (var i = 0; i < 11; i++) {  
3     sum += i;  
4 }
```

# 跳转语句

- ▶ break 语句

只能在循环语句或 switch 语句中使用，退出最内层的循环或 switch 语句。

- ▶ continue 语句

只能在循环中使用，转而执行下一次循环。

- ▶ return 语句

用法：

`return 表达式;`

`return` 只能在函数中使用，当函数执行到 `return` 时函数执行终止，且将返回值传给调用函数。



# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

文档对象

事件处理

# 什么是函数

函数，就是一系列代码的集合，为了完成某一个会重复使用的特定功能。

Listing 72: 函数

```
1 function getAddSum(num) {  
2     var sum = 0;  
3     for (var i = 0; i < num + 1; i++) {  
4         sum += i;  
5     }  
6     return sum;  
7 }
```

说明：

函数只被定义一次，但是可能被执行或调用任意次。

同一个函数不要重复定义，否则只有最后一次的定义有效。

# 函数定义

函数使用function关键词来定义。

函数的定义一般有两种方式：

- ▶ 1. function 函数名 (参数 1, 参数 2...)

```
{  
    //代码块  
}
```

- ▶ 2. var 函数名 = function(参数 1, 参数 2...)

```
{  
    //代码块  
}
```

一般采用第一种方法定义函数。

# 函数调用

- ▶ 函数名 + 参数

Listing 73: 实例

```
1     funClick();
```

- ▶ 事件中

Listing 74: 实例 1

```
1     <div onclick="funClick()">安徽兮克</div>
```

Listing 75: 实例 2

```
1     window.onload = showStart;
```

- ▶ 链接调用

Listing 76: 实例

```
1     <a href="javascript:funClick()">安徽兮克</a>
```

# 特殊函数

- ▶ 嵌套函数
- ▶ 递归函数
- ▶ 内置函数



# 嵌套函数

JS 可以在函数体中定义新的函数，这个新的函数称之为嵌套函数。

Listing 77: 实例

```
1 function isArrSumLess(arr1, arr2) {  
2     function sum(arr) {  
3         var arrSum = 0;  
4         for (var i = 0; i < arr.length; i++) {  
5             arrSum += arr[i];  
6             return arrSum;  
7         }  
8     }  
9     return sum(arr1) < sum(arr2);  
10 }
```

嵌套函数的作用域只在函数体内。

# 递归函数

递归函数，就是在函数内调用本函数。不过要注意，处理不当容易造成死循环，所有尽量避免使用。  
所有只有在特定的情况下会使用，比如阶乘。

Listing 78: 实例

```
1 function fact(num) {  
2     if (num <= 1) {  
3         return 1;  
4     } else {  
5         return num * fact(num - 1);  
6     }  
7 }
```

# 内置函数

- ▶ 常规函数

`alert`: 警告对话框

`confirm`: 确认对话框

`prompt`: 带有文本框的对话框

- ▶ 常用函数

`parseInt`: 转换为数字

`parseFloat`: 转换为浮点型数字

`toString`: 转换为字符串

- ▶ 字符串函数

`substring`、`slice`、`substr`: 字符串截取函数

`toLowerCase`: 转为小写

`toUpperCase`: 转为大写

字符串转码: `escape`、`unescape`、`encodeURL`、`decodeURL`、  
`base64Encode`、`base64Decode`...

# 作业

- 联合国世界卫生组织对年龄的划分标准，44岁以下为青年；45岁至59岁为中年人。60岁至89岁为老年人；90岁以上为长寿老年人。请据此编写代码。
- 利用循环在控制台打印出所有的“水仙花数”。“水仙花数”是指一个3位数，其各位数字立方和等于该数的本身。例如，153就是一个水仙花数，因为 $153=1^3+5^3+3^3$ 。
- 通过函数创建表格，参数是行和列。
- 利用循环在控制台打印出下面图形。

```
*  
***  
*****  
===== *  
***  
*****  
===== *  
**  
***  
===== *  
**  
***  
*****  
===== *  
**  
***  
***  
*
```

# JavaScript

概述

引用方式

基本语法

流程控制

函数

**对象**

浏览器对象

文档对象

事件处理

# 对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...

对象是带有属性和方法的特殊数据类型。

属性是与对象相关的值。访问对象属性的语法是：

objectName.propertyName

方法是能够在对象上执行的动作。调用对象方法的语法  
是:objectName.methodName()



# 字符串对象 |

- ▶ length 属性

获取字符串的长度。语法：

stringObject.length

字符串对象的主要方法如下：

- ▶ indexOf()

获取指定字符串首次出现的位置。语法：

stringObject.indexOf(字符串)

- ▶ lastIndexOf()

获取指定字符串最后出现的位置。语法：

stringObject.lastIndexOf(字符串)

- ▶ search()

获取指定字符串出现的位置。语法：

stringObject.search(字符串)

# 字符串对象 II

## ▶ match()

从字符串中索引指定的值。语法：

stringObject.match(字符串)

Listing 79: 实例

```
1  var str = "abcdefgabc";
2  console.log("length:");
3  console.log(str.length);
4  console.log("indexOf:");
5  console.log(str.indexOf("a"));
6  console.log("lastIndexOf:");
7  console.log(str.lastIndexOf("a"));
8  console.log("search:");
9  console.log(str.search("a"));
10 console.log("match:")
11 console.log(str.match("a"));
12 str.replace()
```

```
length:
10
indexOf:
0
lastIndexOf:
7
search:
0
match:
▶ ["a", index: 0, input: "abcdefgabc"]
```

>

# 字符串对象 III

- ▶ **test()**

检测一个字符串是否匹配某个模式，一般和正则表达式一起使用。语法：

`regexp.test(字符串)`

`regexp` 正则表达式模式或正则表达式对象。

- ▶ **substring()**

提取字符串中的某一部分字符串。用法：

`字符串.substring(n, m)`

`n` 为开始位置，`m` 为结束位置。

开始和结束位置都是大于等于 0 的整数。

- ▶ **replace()**

用某些字符串替换另一些字符，常和正则表达式一起使用。

语法：

`stringObject.replace(原字符串, 更改字符串)`

## 字符串对象 IV

- ▶ `charAt()`

获取字符串中的某一个字符。语法：

`stringObject.charAt(n)`

`n` 为数字，表示字符串中的第几个字符。

- ▶ `concat()`

连接 2 个或多个字符串。语法：

`字符串 1.concat(字符串 2, 字符串 3,..., 字符串 n)`

- ▶ `split()`

把字符串分割成字符串数组。用法：

`字符串.split(分隔符)`

# 什么是数组

数组是值的有序集合。每个值叫做数组的元素，每个元素在数组中都有一个位置，位置以数字表示，称为索引或下标。

说明：

数组中的元素可以是数字、字符串等不同数据类型。  
数组下标从 0 开始。

# 创建数组 |

- ▶ 数组直接量  
有下面两种方式创建。
  - ▶ 没有元素的数组

```
1 var xk_arr1 = [];
```

- ▶ 有指定元素的数组

```
1 var xk_arr2 = [2, 3, 5, 7, 11];
2 var xk_arr3 = [1.1, true, "a"]
```

这是创建数组最简单的方法。

- ▶ 调用构造函数 Array()  
有下面三种方式调用构造函数。
  - ▶ 调用时没有参数

```
1 var xk_arr1 = new Array();
```

创建一个没有元素的空数组。



# 创建数组 II

- ▶ 调用时有一个数值参数

```
1 var xk_arr2 = new Array(8);
```

创建一个指定长度的数组。

- ▶ 调用时有一个非数组的参数指定多个数组元素

```
1 var xk_arr3 = new Array("a");
2 var xk_arr4 = new Array(1.1, true, "a");
```

创建一个有指定元素的数组。

说明：

数组元素之间用“，”分隔。

如果数组的最后只有逗号没有元素，那么不算是数组元素。

忽略的数组元素被赋予 undefined 值。

```
1 var arr1 = [1, , false, , "a", null];
```

# 数组元素的读写 |

使用 [ ] 操作符来访问数组中的元素。

读写数组元素的语法：

数组名 [n]

n 为数组索引。

```
1 var a = ["hello"];
2 var value = a[0];
3 a[1] = 3.14;
4 i = 2;
5 a[i] = 3;
6 a[i + 1] = "world";
7 a[a[i]] = value;
```

说明：

数组是对象的特殊形式。使用方括号访问数组元素就行访问对象的属性一样。

所有的索引都是属性名，但只有 0 到  $2^{32}-2$  之间的整数属性名才是索引。

事实上数组索引是对象属性的一种特殊类型。

## 数组元素的读写 II

数组不会有“越界”，当查询不存在的属性时不会报错，只会得到 undefined 值，类似对象。

Listing 80: 实例

```
1 a[-1.2] = true;      // 创建一个名为 "-1.2" 的属性
2 a["1000"] = 0;       // 数组的第 1001 个元素
3 a[1.00]             // 和 a[1] 相等
```

# 数组的属性和方法

数组最重要的属性就是 `length`, 用于获取数组的长度。

语法：数组名`.length`

数组常用的方法如下：

Table: 数组方法

<code>slice()</code>	获取数组中的某段数组元素
<code>unshift()</code>	在数组开头添加元素
<code>push()</code>	在数组末尾添加元素
<code>shift()</code>	删除数组的第一个元素
<code>pop()</code>	删除数组的最后一个元素
<code>toString()</code>	将数组转换为字符串
<code>join()</code>	将数组元素连接成字符串
<code>concat()</code>	多个数组连接成字符串
<code>sort()</code>	数组元素正向排序
<code>reverse()</code>	数组元素反向排序

# 数组遍历

使用 for 循环是遍历数组最常见的方法。

Listing 81: 实例

```
1 for (var i = 0; i < arr.length; i++) {  
2     // 跳过null、undefined和不存在的元素  
3     if (!arr[i]) continue;  
4 }
```

# Math 对象 |

Math 对象提供了很多数学常量和数学方法。

Math 对象的属性如下：

Table: Math 对象属性

E	算数常量 e
PI	圆周率
SQRT1_2	2 的平方根的倒数
SQRT2	2 的平方根

## Math 对象 II

Math 对象方法如下：

Table: Math 对象方法

abs(x)	绝对值
ceil(x)	上舍入
fool(x)	下舍入
round(x)	四舍五入
max(x, y)	x 和 y 的最大值
min(x, y)	x 和 y 的最小值
pow(x, y)	x 的 y 次幂
random()	返回 0 到 1 之间的随机数

# 日期对象 |

JavaScript 提供了日期对象 (`Date`) 来操作日期和时间。

创建日期对象必须使用 `new` 语句。语法：

```
var 日期对象名 = new Date();
```

```
1 var xk_d1 = new Date();
2 console.log(xk_d1);
```

```
Fri Mar 15 2019 16:15:00 GMT+0800 (CST)
```

日期对象的方法一共分为三类：获取日期和时间、设置日期和时间、转换日期时间字符串。

Table: 获取日期时间

<code>getFullYear()</code>	返回表示年份的 4 位数字
<code>getFullMonth()</code>	返回月份 (0 到 11 )
<code>getDate()</code>	返回日 (1 到 31 )
<code>getHours()</code>	返回小时 (0 到 23 )
<code>getMinutes()</code>	返回分钟 (0 到 59 )
<code>getSeconds()</code>	返回秒数 (0 到 59 )

# 日期对象 II

Table: 设置日期时间

setFullYear()	设置年、月、日
setFullMonth()	设置月、日
setDate()	设置日数
setHours()	设置时、分、秒、毫秒
setMinutes()	设置分、秒、毫秒
setSeconds()	设置秒、毫秒

Table: 日期时间转换

toString()	转为普通字符串
toUTCString()	转为时间时间格式的字符串
toLocaleString()	转为本地时间格式的字符串

# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

文档对象

事件处理



# 什么是 window 对象

一个浏览器窗口就是一个 window 对象。

window 对象就是用来操作“浏览器窗口”打开、关闭、控制大小、位置的一个对象。



# 计时器 |

定时器在图片轮播、在线时钟、广告弹窗等地方有很多用途。

定时器的实现有下面两种方法：

- ▶ `setTimeOut()`

实现函数或代码在指定的时间之后执行。

语法如下：

```
var 变量名 = setTimeOut(code, time);
```

```
1 window.onload = function () {  
2     setTimeout(alert("Welcome"), 200)  
3 }
```

说明：

`code` 可以是一段代码，也可以是一个可调用的函数名。

`time`，表示要过多长时间才执行 `code` 的内容，单位为毫秒。

`setTimeOut()` 返回一个值，这个值可以传递给

`clearTimeOut()` 用于取消这个函数的执行。

# 计时器 II

- ▶ `setInterval()`

实现函数或代码在指定的时间间隔里重复调用。

语法如下：

```
var 变量名 = setInterval(code, time);
```

```
1 var n = 5, t;
2 window.onload = function () {
3     t = setInterval("count()", 500);
4 }
5 function count() {
6     console.log(n)
7     if (n > 0) {
8         n--;
9     } else {
10        clearInterval(t);
11    }
12 }
```

说明：

`setInterval()` 返回一个值，这个值可以传递给 `clearInterval()` 来取消执行。

# 对话框

window 对象提供了 3 个方法来向用户显示简单的对话框。

- ▶ alert()

向用户显示一条消息并等待用户关闭对话框。

- ▶ confirm()

向用户显示一条消息，要求用户单击确定或取消，并返回一个布尔值。

- ▶ prompt()

向用户显示一条消息，并等待用户输入字符串并返回那个字符串。

```
1 do {  
2     var name = prompt("What is your name?");  
3     var flag = confirm("You entered '" + name + "'.\n" + "Click ok to proceed  
        or cancel to re-enter.")  
4 } while (!flag)  
5 alert("Hello, " + name);
```

# 打开和关闭窗口 |

## ▶ 打开窗口

使用 window 对象的 open() 方法可以打开一个新的浏览器窗口。

语法如下：

```
window.open(URL, 窗口名称, 参数);
```

说明：

url：打开窗口的地址

窗口名称：window 对象的名称，如果指定的是一个已经存在的窗口则不会打开新窗口。

参数：以逗号分隔的列表，包含大小和各种属性。

# 打开和关闭窗口 II

Table: window.open 的参数

top	窗口距离屏幕顶部的距离
left	窗口距离屏幕左边的距离
width	窗口的宽度
height	窗口的高度
scrollbars	是否显示滚动条
resizeable	窗口大小是否固定
toolbar	浏览器工具栏
menubar	浏览器菜单栏
location	浏览器地址栏

# 打开和关闭窗口 III

## ▶ 关闭窗口

使用 close() 方法可以关闭窗口。

语法如下：

window 对象.open(URL, 窗口名称, 参数);

Listing 82: 实例

```
1 <body>
2     <button onclick="openWindow()">打开</button>
3     <button onclick="closeWindow()">关闭</button>
4     <script>
5         var win;
6         function openWindow(){
7             win = window.open("http://seekswan.com", "安徽兮克")
8         }
9         function closeWindow(){
10            win.close();
11        }
12    </script>
13 </body>
```

# 浏览历史 |

window 对象中的 history 对象把窗口的浏览历史用文档和文档状态列表的形式表示。

其中，history 对象常用的属性如下：

Table: history 对象属性

length	历史记录的长度
current	当前窗口的 url
next	历史列表的下一个 url
previous	历史列表的前一个 url

## 浏览历史 II

history 对象的常用方法如下：

Table: history 对象方法

go()	进入指定网页
back()	返回上一页
forward()	进入下一页

说明：

history 对象的 back() 和 forward() 方法与浏览器的“后退”和“前进”一样。

go() 接受一个整数参数。

```
1 history.go(-2)      // 后退两个历史记录
```

# 作业

1. 对数组 [1,10,6,1,6,1,2,10,1,3,1,3,4,5,6] 去重。
2. 随机生成 16 个不重复的 200 以内的整数组成数组，并对其从小到大排序后在控制台打印出来。
3. 编写一个返回距离当前日期时间差的函数，如刚刚、几分钟前、几小时前、几天前、几周前、几个月前，参数日期格式为“2018-11-11 14:20”。
4. 一个 60s 的倒计时显示在网页上。

# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

**文档对象**

事件处理

# 文档对象

document 对象，即文档对象，是 window 对象的子对象。  
文档对象，操作的都是 HTML 文档。  
文档对象的属性如下：

Table: document 对象属性

title	文档标题
URL	文档地址
cookie	客户端浏览器的存储值

文档对象的方法如下：

Table: document 对象方法

write()	输入文本到当前打开的文档
writeln()	输入文本到当前打开的文档，并添加换行符
getElementById()	获取某个 id 值的元素



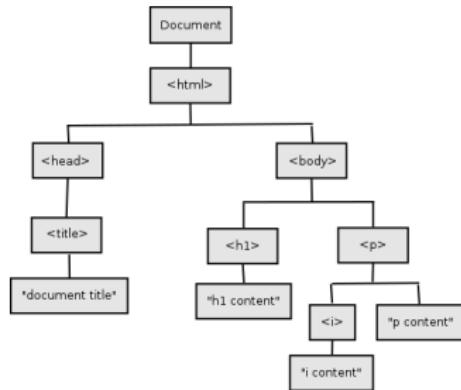
# 文档对象模型

文档对象模型，DOM ( Document Object Model )，它是由 W3C 组织定义的一个标准，代表和操作文档的内容，如添加元素、删除元素、替换元素等。

# DOM 概览 |

DOM 采用树形结构作为分层结构，以树节点形式表示页面中各种元素或内容。

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>document title</title>
6  </head>
7  <body>
8      <h1>h1 content</h1>
9      <p><i>i content</i>p content</p>
10 </body>
11 </html>
```



## DOM 概览 II

借用家谱图联系上图理解父节点、子节点、兄弟节点、后代节点、祖父节点。

说明：

图上的每个框都是文档的一个节点，表示一个 Node 对象。

Document、Element、Text 是 Node 的子类。

Document，是树形的根部节点，代表整个文档。

Element，是 HTML 元素的节点。

Text，是文本节点。

# 选取文档元素 |

为了操作文档中的元素，需要通过某种方式选取 Element 对象。有如下的方法：

- ▶ 通过 ID

`document.getElementById()`

```
1 var e = document.getElementById("xk");
```

- ▶ 通过名字

`document.getElementsByName()`

```
1 var e = document.getElementsByName("xk")[0];
```

- ▶ 通过标签名重写

`document.getElementsByTagName()`

```
1 var e = document.getElementsByTagName("p")[0];
```

# 选取文档元素 II

- ▶ 通过类

document.getElementsByClassName()

```
1 var e = document.getElementsByClassName("xk")[0];
```

- ▶ 通过 css 选择器

返回所有匹配的元素：

document.querySelectorAll()

返回第一个匹配的元素：

document.querySelector()

```
1 var e1 = document.querySelectorAll("[type='checkbox'][0];  
2 var e2 = document.querySelector("[type='checkbox']");
```

# 遍历 DOM I

在 DOM 中，每一个元素节点都被看成一个对象。下面是 DOM 对象节点的属性。

Table: 常用的节点属性

nodeName	返回节点的名字
nodeType	返回节点类型的数字
nodeValue	返回给定节点的当前值

Table: 遍历节点的属性

parentNode	获取父节点
childNodes	获取子节点集合
firstChild	获取第一个子节点
lastChild	获取最后一个子节点
nextSibling	获取下一个兄弟节点
previousSibling	获取上一个兄弟节点
attributes	元素的属性列表



## 遍历 DOM II

```
1 <body>
2   <div><p id="xk"></p></div>
3   <script>
4     var e = document.getElementById("xk");
5     if(e.parentNode){
6       alert("此节点又父节点");
7     }
8   </script>
9 </body>
```

注意：

跟 DOM 有关操作的 js 代码必须在 dom 加载完成后写入才行，如上图放在 `<body>` 标签最下面，如果想放在 `<head>` 内，可以使用 `window.onload` 事件或其他方法确保文档加载完成。

# 节点操作 I

关于 DOM 节点的常用操作如下：

- ▶ 创建节点

DOM 定义了创建 Element 和 Text 对象的方法。如下：

```
document.createElement("元素名");
document.createTextNode("元素内容");
```

```
1 // 创建图像节点
2 var e = document.createElement("img");
3 // 创建文本节点
4 var t = document.createTextNode("安徽兮克");
```

# 节点操作 II

## ▶ 插入节点

插入节点有两种方法：

obj.appendChild(new)，追加到节点的内部。

obj.insertBefore(new, ref)，插入到 ref 节点前。

说明：

obj，表示父节点

new，代表新节点

ref，指定的一个节点

- 安徽  
追加2
- 夯克  
追加1

```
1 <body>
2   <ul id="xk">
3     <li>安徽</li><li id="xike">夯实</li>
4   </ul>
5   <script>
6     var e = document.getElementById("xk");
7     var ct1 = document.createTextNode("追加1");
8     var ct2 = document.createTextNode("追加2");
9     e.appendChild(ct1);
10    e.insertBefore(ct2, document.getElementById("xike"));
11  </script>
12 </body>
```

# 节点操作 III

## ▶ 删除节点

语法: obj.removeChild(oldChild)

说明:

obj, 表示删除元素的父节点

oldChild, 表示删除元素

```
1 <body>
2   <ul id="xk">
3     <li>安徽</li><li id="xike">兇克</li>
4   </ul>
5   <script>
6     var e = document.getElementById("xk");
7     var old = document.getElementById("xike");
8     e.removeChild(old);
9   </script>
10 </body>
```

- 安徽

# 节点操作 IV

## ▶ 复制节点

语法：

obj.cloneNode(bool)

说明：

obj，被复制的节点

bool，布尔值，1 或 true，复制节点本身以及复制该节点下的所有子节点；0 或 false，仅复制节点本身。

```
1 <body>
2   <ul id="xk"><li>安徽</li><li id="xike">不克</li></ul>
3   <script>
4     var e = document.getElementById("xk");
5     var clone = e.firstChild.cloneNode(1);
6     e.appendChild(clone);
7   </script>
8 </body>
```

- 安徽
- 克不
- 安徽

# 节点操作 V

## ▶ 替换节点

语法如下：obj.replaceChild(new, old)

说明：

obj, 被替换节点的父节点

new, 替换后的新节点

old, 要替换的旧节点

```
1 <body>
2   <ul id="xk"><li>安徽</li><li id="xike">兮克</li></ul>
3   <script>
4     var e = document.getElementById("xk");
5     var newNode = document.createTextNode("替换内容");
6     e.replaceChild(newNode, e.firstChild);
7   </script>
8 </body>
```

替换内容

- 兮克

# 修改文档结构

## ▶ 内容操作

获取文档结构: element.innerHTML

获取文本内容: element.innerText

```
1 e.innerHTML = "<h1>111</h1>";  
2 e.innerText = "<h1>111</h1>";
```

## ▶ 属性操作

获取属性: element.getAttribute(" 属性名");

设置属性: element.setAttribute(" 属性名", " 修改值");

```
1 var e = document.getElementById("xk");  
2 var e_class = e.getAttribute("class");  
3 var e_type = e.getAttribute("type");  
4 e.setAttribute("name", "xk");
```

## ▶ 类操作

增加类: element.classList.add(" 类名");

删除类: element.classList.remove(" 类名");

```
1 e.classList.add("active");  
2 e.classList.remove("xk", "xk2");
```



# 修改样式

- ▶ 重写 style 属性

```
element.style.cssText = "修改值";
```

- ▶ 修改单个样式

```
e.style.color = "修改值";
```

```
e.style.backgroundColor = "修改值";
```

```
e.style.marginTop = "修改值";
```

```
1 e.style.cssText = "color: #000; background-color:#fff;";  
2 e.style.color = "#000";  
3 e.style.backgroundColor = "#fff";  
4 e.style.marginTop = 10 + "px";
```

# JavaScript

概述

引用方式

基本语法

流程控制

函数

对象

浏览器对象

文档对象

事件处理

# 什么是事件

事件，是 Web 浏览器通知应用程序发生了什么事情。

页面的交互少不了事件的参与，比如鼠标移动、点击事件、键盘按下事件等。

# 事件调用方式

调用事件的方式有下面两种：

- ▶ 在 js 代码中调用

语法：

```
var 变量名 = document.getElementById(" 元素 ID");
变量名. 事件处理器 = function()
{
    .....
}
```

```
1 var img = document.getElementById("xk_img");
2 img.onload = function(){
3     alert("图片加载完成 ");
4 }
```

- ▶ 在元素中调用

```
1 <body>
2     
3     <script>
4         function imgLoad(){
5             alert("图片加载完成 ");
6         }
7     </script>
8 </body>
```

# 鼠标事件

常用的鼠标事件如下：

Table: 常用的鼠标事件

onclick	鼠标单击事件
ondblclick	鼠标双击事件
onmouseover	鼠标移入事件
onmouseout	鼠标移出事件
onmousedown	鼠标按下事件
onmouseup	鼠标松下事件

# 键盘事件

- ▶ onkeypress

在键盘上的某个字符键被按下到松开的整个过程

- ▶ onkeydown

在键盘上的任意字符都触发。

按下字符键 onkeydown 和 onkeypress 都会触发，但有顺序：  
onkeydown>onkeypress 事件。

- ▶ onkeyup

在键盘上某个键按下之后松开的一瞬触发的事件。

三个事件的执行顺序：onkeydown>onkeypress>onkeyup

# 表单事件

- ▶ onfocus 和 onblur

onfocus 获取焦点触发的事件， onblur 失去焦点触发的事件。  
具有这两个事件的元素有：

- ▶ 单行文本框 text
- ▶ 多行文本框 textarea
- ▶ 下拉列表 select

- ▶ onchange 元素内容发生变化时触发的事件。具有这个事件的元素有：

- ▶ 单行文本框 text
- ▶ 多行文本框 textarea
- ▶ 下拉列表 select
- ▶ 文件上传 file

- ▶ onsubmit

表单提交前触发的事件，常用来表单提交前的验证。

## 编辑事件

- ▶ **oncopy**

在网页上复制内容会触发。往往通过这个事件禁止用户复制网页内容。

- ▶ **oncut**

在网页剪切内容时触发。

- ▶ **onpaste**

在文本框等粘贴内容时触发。

# 页面相关事件

- ▶ **onload**

页面或图像加载完成再执行的事件。除了 window，支持的标签有 body、frame、img 等。

```
window.onload = function()
```

```
{
```

```
.....
```

```
}
```

- ▶ **onresize**

页面改变大小的事件。仅有 window 支持。

```
window.onresize = function()
```

```
{
```

```
.....
```

```
}
```

# 绑定事件 |

之前介绍的绑定事件方式有两种：

- ▶ 在 DOM 元素中直接绑定
- ▶ 在 js 代码中直接绑定

但是的方式不能同时绑定多个事件、使代码耦合在一起。

```
1 <body>
2   <button id="button">点我</button>
3   <script>
4     var btn = document.getElementById("button");
5     btn.onclick = function () {
6       console.log("触发 1");
7     }
8     btn.onclick = function () {
9       console.log("触发 2");
10    }
11  </script>
12 </body>
```

结果只有执行第二个绑定事件。

## 绑定事件 II

- ▶ 使用 attachEvent() 或 addEventListener() 来绑定监听函数  
attachEvent() 是 ie 添加事件处理程序。语法如下：

element.attachEvent(type,listener);

element, HTML 节点, 要绑定事件的对象;

type, 事件名称, 事件前加上 on, 如 “onclick” ;

listener, 监听事件。

其对应的删除绑定事件的方法是 detachEvent()。

addEventListener() 是 W3C 支持的标准的绑定事件监听函数的方法。其语法如下：

element.addEventListener(type,listener,useCapture);

type, 去除事件前的 “on”, 比如 “click”;

useCapture, 事件处理的时期或阶段, 默认冒泡事件 ( false )  
还是事件捕获 ( true )。

其对应的删除绑定事件的方法是 removeEventListener()。



# 绑定事件 III

Listing 83: 事件捕获与事件冒泡

```
1  </head>
2
3 <body>
4   <div id="xk1">1
5     <div id="xk2">2
6       <div id="xk3">3</div>
7     </div>
8   </div>
9   <script>
10    var e1 = document.getElementById("xk1"),
11        e2 = document.getElementById("xk2"),
12        e3 = document.getElementById("xk3");
13    e1.addEventListener("click", function () { console.log("1") }, true);
14    e2.addEventListener("click", function () { console.log("2") }, true);
15    e3.addEventListener("click", function () { console.log("3") }, true);
16    e1.addEventListener("click", function () { console.log("111") }, false)
17      ;
18    e2.addEventListener("click", function () { console.log("222") }, false)
19      ;
20    e3.addEventListener("click", function () { console.log("333") }, false)
21      ;
```

# 绑定事件 IV

```
1  
2  
3  
333  
222  
111
```

Listing 84: 绑定事件兼容性写法

```
1 function on(e, type, callback){  
2     if(e.addEventListener){  
3         e.addEventListener(type, callback);  
4     } else {  
5         type = "on" + type;  
6         e.attachEvent(type, callback);  
7     }  
8 }
```