《Web前端开发基础》

> 讲师：阿杜

一、前端概要

1. 学习路线

- HTML标记语言：组成网页架构的元素组件
- CSS样式语言：美化网页的样式
- JavaScript程式语言：控制网页的动态效果
- jQuery程式语言：协助及加强JavaScript的实现

2. 网页组成

- 百度图片

- 例如:大家所熟知的百度首页，右击→查 看网页源代码即可看到HTML内容

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>首页</title>
    <meta name="Keywords" content="关键字">
    <meta name="Description" content="简介、描述">
    <link rel="stylesheet" href="./css/main.css">
    <style>
        /* css代码 */
    </style>
    <script type="text/javascript" src="./js/main.js"></script>
</head>
<body>
    <!— 内容 —>
    <h1>我的第一个标题</h1>
    <p>我的第一个段落。</p>
    <script type="text/javascript">
        // js代码
    </script>
```

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

- 页面标签

标签	解释
<!DOCTYPE>	不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。
<html>	定义了页面的开始点和结束点，在它们之间是文档的头部和主体
<head>	定义页面的头部， 中的元素可以引用脚本、指示浏览器在哪里找到样式表、提供元信息等等。
<title>	定义页面的标题。
<meta>	定义页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词。
<link>	定义页面与外部资源的关系，rel="stylesheet"说明href连接的文档是一个样式表。
<style>	用于为 HTML 页面定义样式信息
<script>	用于定义客户端脚本，比如 JavaScript。既可以包含脚本语句，也可以通过 src 属性指向外部脚本文件
<body>	定义文档的主体,比如文本、超链接、图像、表格和列表等等。

二、HTML 标记语言

1. HTML介绍

- HTML是一种用于创建网页的标记语言，可以使用HTML创建网页文档，用浏览器打开会自动解析。
- HTML是由标签和内容构成的。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    文档的内容...
</body>
</html>
```

2. 文本格式化标签

- 标签描述

标签	描述
 	换行
	标题，定义标题字体大小，1最大，6最小
	段落
	斜体
<cite></cite>	引用
	加粗
	强调加粗
<del></del>	删除线
<a>	文本

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <h1>hello world 标题1</h1>
    <h2>hello world 标题2</h2>
    <br/>
    <h3>hello world 标题3</h3>
    <p>hello world 段落</p>
    <i>hello world 斜体</i>
    <br/>
    <cite>hello world 引用</cite>
    <br/>
    <b>hello world 加粗</b>
    <br/>
    <strong>hello world 强调加粗</strong>
    <br/>
    <del>hello world 删除</del>
    <a>hello world 文本</a>
</body>
</html>
```

- 运行结果

hello world 标题1

hello world 标题2

hello world 标题3

hello world 段落

hello world 斜体
hello world 引用
hello world 加粗
hello world 强调加粗
~~hello world 删除~~ hello world 文本

3．列表标签

- 标签描述

标签	描述	参数
	无序列表	- type=disc 默认，实心圆 - square 实心方块 - cicle 空心圆
	有序列表	type=1 默认数字，其他值：A/a/I/i/1
	列表项目	在有序列表和无序列表中使用

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <!— 无序列表 —>
    <ul type="circle">
        <li>Coffee</li>
        <li>Milk</li>
    </ul>
    <!— 有序列表 —>
    <ol type="A">
        <li>Bread</li>
        <li>Juice</li>
    </ol>
</body>
</html>
```

- 运行结果

  - Coffee
  - Milk

  A. Bread
  B. Juice

4. 超链接标签

- 超链接标签: `<a href="网址"></a>`
- 属性描述

属性	描述
href	指定链接跳转地址
target	链接打开方式，常用值: _blank 打开新窗口
title	文字提示属性
name	定义锚点

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <a href="http://www.baidu.com/" target="_blank">访问百度!</a>
    <p>如果你将 target 属性设置为 "_blank", 链接将在新窗口打开。</p>
    <a name="tips">基本的注意事项 - 有用的提示</a>
    <a href="#tips">有用的提示</a>
</body>
</html>
```

- 运行结果

访问百度!

如果你将 target 属性设置为 "_blank", 链接将在新窗口打开。

基本的注意事项 – 有用的提示 有用的提示

5. 图片标签

- 图片标签: `<img src="图片文件路径" alt="图片提示" />`
- 属性描述

属性	描述
alt	图片加载失败时的提示信息
title	文字提示属性
width	设置宽度, 如100
height	设置高度, 如100

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <img src="boat.gif" alt="Big Boat"/>
    <img
src="https://sdn.youdianzhishi.com/covers/2020/12/2/88b4978df01b4773867104837ceaebbd.
png" width="500px" height="200px"/>
</body>
</html>
```

- 运行结果

Big Boat

6．表格标签

- 标签描述

标签	描述
<table>	表格标签
<thead>	表格头部
<tr>	行标签
<th>	列名
<tbody>	表格内容
<tr>	列标签
<td>	列内容

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <!-- border标识边框宽度 -->
    <table border="1">
        <thead>
            <tr>
                <th>主机名</th>
                <th>IP</th>
                <th>操作系统</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>www.baidu.com</td>
                <td>192.168.1.10</td>
                <td>CentOS7</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

- 运行结果

主机名	IP	操作系统
www.baidu.com	192.168.1.10	CentOS7

7. 表单标签

- 表单标签: `<form></form>`
- 属性描述

属性	描述
action	提交的目标地址(URL)
method	提交方式:get(默认)和post
enctype	编码类型 - application/x-www-form-urlencoded 默认值，编码字符 - multipart/form-data 传输数据为二进制类型，如提交文件 - text/plain 纯文本的传输

- 表单项标签: `<input></input>`
- 属性描述

属性	描述
type	- text:单行文本框 - password:密码输入框 • checkbox:多选框 - radio:单选框 - file:文件上传选择框 - button:普通按钮 - submit:提交按钮 - reset:重置按钮
name	表单项名，用于存储内容值
value	表单项的默认值
disabled	禁用该元素
checked	默认被选中，用于选项框

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
```

```
<body>
    <form name="input" action="index.html" method="post">
        First name: <input type="text" name="firstname"><br>
        Last name: <input type="text" name="lastname">
        <br/>
        <input type="radio" name="sex" value="male" checked>男<br>
        <input type="radio" name="sex" value="female">女
        <br/>
        <input type="checkbox" name="vehicle" value="Bike">我喜欢自行车<br>
        <input type="checkbox" name="vehicle" value="Car">我喜欢小汽车
        <br/>
        Username: <input type="text" name="user">
        <input type="submit" value="提交了">
        <input type="button" value="验证按钮" onclick="alert('密码输入正确')">
        <input type="reset">
    </form>
</body>
</html>
```

- 运行结果

8. 下拉选择框标签

- 下拉选择框标签：`<select></select>`
- 属性描述

属性	描述
name	下拉列表的名称，用于存储下拉值
disabled	禁用该元素
multiple	设置可以选择多个项目
size	指定下拉列表中的可见行数

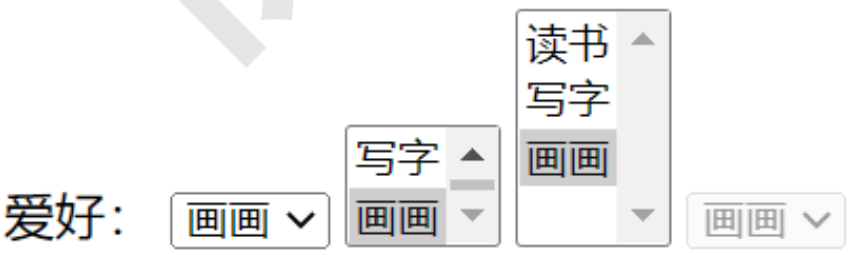
- 下拉选择框选项标签：`<option></option>`
- 属性详情

属性	描述
value	选项值
selected	默认下拉项

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <a>爱好: </a>
    <select>
        <option value="read">读书</option>
        <option value="write">写字</option>
        <option value="draw" selected>画画</option>
    </select>
    <select size="2">
        <option value="read">读书</option>
        <option value="write">写字</option>
        <option value="draw" selected>画画</option>
    </select>
    <select multiple>
        <option value="read">读书</option>
        <option value="write">写字</option>
        <option value="draw" selected>画画</option>
    </select>
    <select disabled>
        <option value="read">读书</option>
        <option value="write">写字</option>
        <option value="draw" selected>画画</option>
    </select>
</body>
</html>
```

- 运行结果



9. 按钮标签

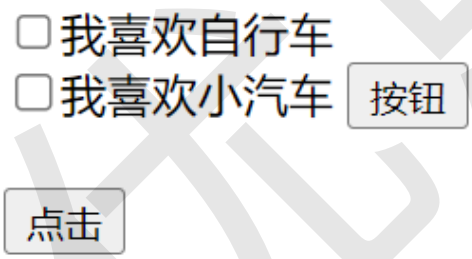
- 按钮标签：`<input></input>`
- 属性描述

属性	描述
type	- button：普通 - submit：提交 - reset：重置

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <form action="http://www.baidu.com" method="post">
        <input type="checkbox" name="vehicle" value="Bike">我喜欢自行车<br>
        <input type="checkbox" name="vehicle" value="Car">我喜欢小汽车
        <button type="reset">按钮</button>
    </form>
    <br/>
    <button onclick="alert(123)">点击</button>
</body>
</html>
```

- 运行结果



10. 块标签

- 块标签：`<div></div>`
- 标签用于在HTML文档中定义一个区块，常用语标签集中起来，然后用样式对他们进行统一排版。
- 属性描述

属性	描述
align	- left 居左，默认 - right 居右 - center 居中 - justify 两端对齐

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
</head>
<body>
    <div style="background-color:#a4a4a6;" align="right">
        <h3>这是一个在 div 元素中的标题。</h3>
        <p>这是一个在 div 元素中的文本。</p>
      </div>
</body>
</html>
```

- 运行结果

三、CSS 样式语言

1. CSS介绍

- 层叠样式表，是一种样式表语言，用来描述HTML和XML文档的呈现。
- 随着HTML的发展，为了满足页面设计者的要求，HTML添加了很多显示功能，但是随着这些功能的增加，使得HTML越来越杂乱
  HTML 页面也越来越臃肿，CSS便随之诞生。
- CSS 用于简化HTML标签，把关于样式部分的内容提取出来，进行单独的控制，使结构与样式分离开发。
- CSS 是以HTML为基础，设置网页的外观显示样式，如字体、颜色、背景的控制及整体的布局等，和HTML 类似
- CSS也不是真正的编程语言，甚至不是标记语言。它是一门样式表语言。当浏览器读到一个样式表，它就会按照这个样式表来对文档进行格式化（渲染）。

2. CSS使用方法

- 内联方式（行内样式）

  ```
  <p style="color:red">在HTML中如何使用css样式</p>
  ```

- 内部方式（内嵌样式），在head标签中使用

```
<style type="text/css">
  p{
      color:red;
  }
</style>
```

- 外部导入方式(推荐)，在head标签中使用

  `<link href="main.css" type="text/css" rel="stylesheet"/>`

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <link href="main.css" type="text/css" rel="stylesheet"/>
</head>
<body>
    <div>
        <p style="color:red">在HTML中如何使用css样式</p>
        <a>我是一串文字</a>
    </div>
    <style type="text/css">
        a {
            color: blue;
        }
    </style>
</body>
</html>
```

2.1 选择器

- **选择器:**需要改变样式的HTML元素
- **格式:** 选择器{属性:值;属性:值;属性:值;....}
- **常见选择器:**标签选择器、类选择器、ID选择器、派生选择器

1）元素选择器

- 使用html标签作为选择器，为指定标签设置样式。
- 示例1: h1元素设置样式

```
h1 {
    color: red;
    font-size: 14;
}
```

- 示例2:多个元素设置样式

```
h1,h2,h3,h4,h5,h6 {
    color: green;
}
```

- 示例3:子元素会继承最高级元素所有属性

```
body {
    color: #000;
    font-family: Verdana, serif; /*字体*/
}
```

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <p>在HTML中如何使用css样式</p>
        <a>我是一串文字</a>
    </div>
    <style type="text/css">
        a {
          color: blue;
        }
        body {
            color: rgb(33, 150, 189);
            font-family: Verdana, serif; /*字体*/
        }
    </style>
</body>
</html>
```

- 运行结果

在HTML中如何使用css样式

我是一串文字

2）ID选择器

- 使用“id”作为选择器，为指定id设置样式。
- 使用格式：#id名{样式...}
- 特点：每个id名称只能在HTML文档中出现一次;在实际开发中，id一般预留JavaScript使用
- 使用方式：

```
/*第一步：给标签指定id*/
<p id="t">...</p>
/*第二步：针对id设置样式*/
#t {
    color: red;
}
```

- 代码实例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <p id="ap">在HTML中如何使用css样式</p>
    </div>
    <style type="text/css">
        #ap{
            color: blue;
        }
    </style>
</body>
</html>
```

- 运行结果

3）类选择器

- 使用“类名”作为选择器，为指定类设置样式。
- 使用格式：.类名{样式...}
- 使用方式

```
/*第一步：给标签指定类*/
<p class="c">...</p>
/*第二步：针对类设置样式*/
.c {
    color: red;
}
```

- 代码示例

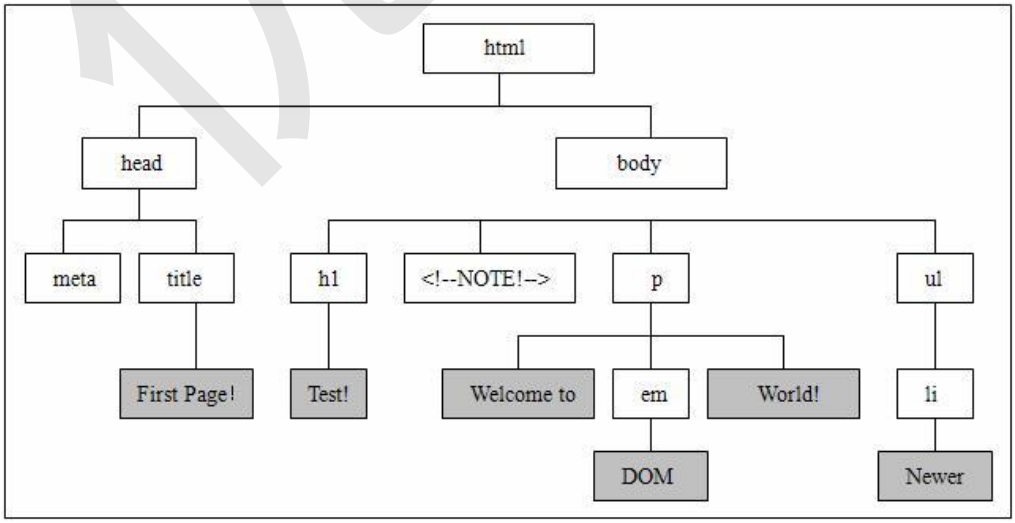
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <p class="ap">在HTML中如何使用css样式</p>
    </div>
    <style type="text/css">
        .ap{
            color: blue;
        }
    </style>
</body>
</html>
```

- 运行结果

在HTML中如何使用css样式

4)派生选择器

- 依据元素在其位置的上下文关系来定义样式。



- 使用方式

```
/*第一步：定义标签内容*/
<div class="c">
    <h1>一号标题</h1>
    <p>这是一个段落</p>
</div>
/*第二步：设置局部样式*/
<style type="text/css">
.c p {
    color: red;
}
</style>
```

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div class="c">
        <h1>一号标题</h1>
        <p>这是一个段落</p>
    </div>
    <style type="text/css">
        .c p {
            color: red;
        }
    </style>
</body>
</html>
```

- 运行结果

一号标题

这是一个段落

2.2 CSS常用属性

- **padding（内边距）**:钻戒到盒子内边框的距离
- **margin（外边距）**:钻戒盒子距离桌子边缘的距离
- **border(边框)**:钻戒盒子边框宽度

1）内边距和外边距

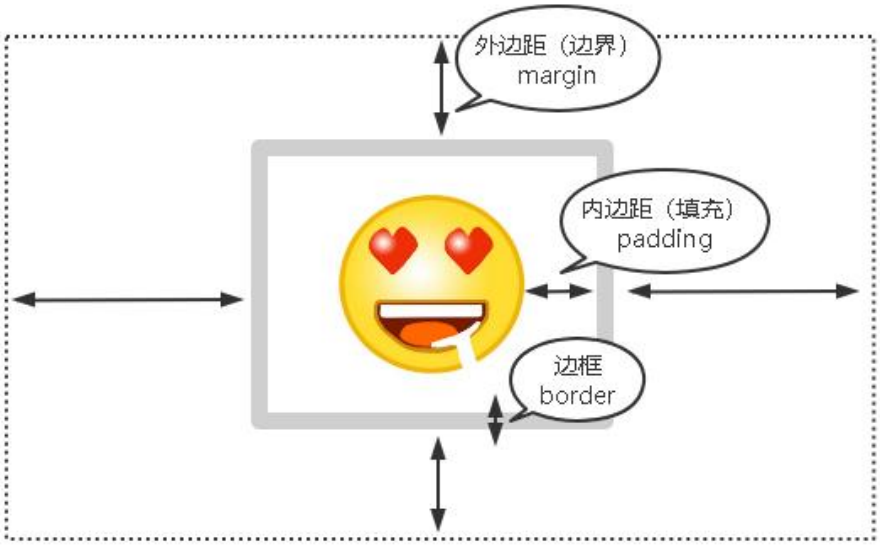
- 内边距

属性	描述
padding	设置四边的内边距
padding-top	上内边距
padding-right	右内边距
padding-left	左内边距
padding-bottom	下内边距

- 外边距

属性	描述
margin	设置四边的外边距，使用方法同padding
margin-top	上外边距
margin-right	右外边距
margin-bottom	下外边距
margin-left	左外边距

- 使用方式:




```
.a {
  padding: 10px 5px 15px 20px; /*上右下左*/
    padding: 10px 5px 15px; /*上右下，实际左内边距也是5px*/
    padding: 10px 5px; /*上右，实际下左也是10px和5px*/
  padding: 10px; /*四边都是10px*/
    padding-top: 10px; /*上*/
}
```

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!— <link href="main.css" type="text/css" rel="stylesheet"/> —>
</head>
<body>
    <div>
        <div class="d">在HTML中如何使用css样式</div>
    </div>
    <style type="text/css">
        .d {
            background-color: rgb(152, 152, 208);
            padding: 10px;
            margin: 10px;
        }
    </style>
</body>
</html>
```

- 运行结果

在HTML中如何使用css样式

2）字体

- 属性

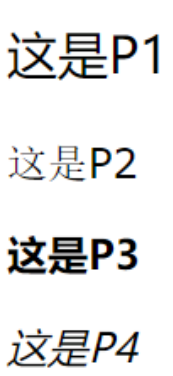
属性	描述	值
font-size	设置字体大小	- xx-small 、 x-small 、 small 、 medium 、 large 、 x- large、xx-large，从小到大，默认值 medium - length 固定长度，例如12px - 浏览器解析字体大小最小是12px
font-family	字体系列。可以写多个 如果第一个不支持，使 用下一个	Microsoft YaHei
font-weight	设置字体的粗细	- normal 默认值 - bold 粗体 - bolder 更粗 - lighter 更细
font-style	字体样式	- normal 正常 - italic 斜体 - oblique 倾斜的字体

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <p class="p1">这是P1</p>
        <p class="p2">这是P2</p>
        <p class="p3">这是P3</p>
        <p class="p4">这是P4</p>
    </div>
    <style type="text/css">
        .p1 {
            font-size: 20px;
        }
        .p2 {
          font-family: Verdana, serif;
        }
        .p3 {
            font-weight: bold;
        }
        .p4 {
            font-style: italic;
        }
    </style>
```

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

- 运行结果



3）文本

- 属性

属性	描述	值
color	字体颜色	- 颜色名称，例如red - 十六进制值，例如#ff0000 - rgb 代码，例如rgb(255,0,0)
text-align	文本对齐方式	- left左边 - right 右边 - center 中间 - justify 两端对齐文本效果
text-decoration	文本修饰	- none 默认，定义标准的文本，例如去掉超链接下划线 - line-through 删除线 - underline 文本下加一条线
text-overflow	文本溢出后显示效果	- clip 修剪文本 - ellipsis 显示省略号来代表被修剪的文本 - string 使用给定的字符串来代表被修剪的文本
letter-spacing	字符间的距离	- normal 默认 - length 自定义间距
line-height	行间的距离(行高)	- normal 默认 - length 设置固定值

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!— <link href="main.css" type="text/css" rel="stylesheet"/> —>
```

```
</head>
<body>
    <div>
        <p class="p1">这是P1</p>
        <p class="p2">这是P2</p>
        <p class="p3">这是P3</p>
        <p class="p4">This is some long text that will not fit in the box</p>
        <p class="p5">这是p5</p>
        <p class="p6">这是P6</p>
    </div>
    <style type="text/css">
        .p1 {
            color: rgb(85, 160, 100);
        }
        .p2 {
            text-align: center;
        }
        .p3 {
            text-decoration: underline;
        }
        .p4 {
            width: 50px;
            height: 50px;
            text-overflow: ellipsis;
            overflow: hidden;    /*隐藏文本溢出的部分*/
            white-space: nowrap; /*强制在一行显示文本*/
        }
        .p5 {
            letter-spacing: 10px;
        }
        .p6 {
            line-height: 100px;
        }
    </style>
</body>
</html>
```

- 运行结果

这是P1

这是P2

这是P3

This ...

这 是 p 5

这是P6

4）边框

- 属性

属性	描述	值
border	所有边框样式的缩写	示例:border: 1px solid blue; 宽度 样式 颜色
border-radius	圆角边框	示例:border-radius:1px;
box-shadow	给元素添加阴影	格式:box-shadow: h-shadow v-shadow blur spread color inset; - h-shadow 必选，水平阴影的位置 - v-shadow 必选，垂直阴影的位置 - blur 可选，模糊程度 - spread 可选，阴影的大小 - color 可选，阴影的颜色 - inset 可选，从外层的阴影(开始时)改变阴影内侧阴影 示例1:box-shadow: 1px 2px 3px 1px #c2c2c2; 示例2:box-shadow: 0 5px 20px 0 #e8e8e8;

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <div class="d1"></div>
        <div class="d2"></div>
        <div class="d3"></div>
```

```
    </div>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
            margin-bottom: 10px;
        }
        .d1 {
            border: 1px solid blue;
        }
        .d2 {
            border-radius: 15px;
            border: 3px solid rgb(117, 117, 159);
        }
        .d3 {
            box-shadow: 1px 2px 3px 1px #c2c2c2;
        }
    </style>
</body>
</html>
```

- 运行结果

5）背景

- 属性

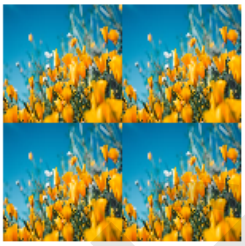
属性	描述	值
background-color	背景颜色	- 颜色名称，例如red - 十六进制值，例如#ff0000 - rgb 代码，例如rgb(255,0,0)
background-image	背景图片	- url('URL') 图片路径 - none 不显示背景图片
background-repeat	设置是否及如何重复背景图像	- repeat 默认。背景图像将在垂直方向和水平方 向重复 - repeat-x 背景图像将在水平方向重复 - repeat-y 背景图像将在垂直方向重复 - no-repeat 背景图像将仅显示一次
background-position	背景图片的位置	- left、top、top right、center left、center center、center right、bottom left、 bottom center、bottom right - x% y% 水平位置和垂直位置
background-size	背景图片的尺寸	- length 背景的高度和宽度，例如80px 60px - percentage 以父元素的百分比设置背景图像的高度和宽度，例如50% 50%

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div>
        <div class="d1"></div>
        <div class="d2"></div>
        <div class="d3"></div>
        <div class="d4"></div>
        <div class="d5"></div>
    </div>
    <style type="text/css">
        div {
            height: 100px;
            width: 100px;
            margin-bottom: 10px;
        }
        .d1 {
            background-color: #aa2f2f;
        }
        .d2 {
```

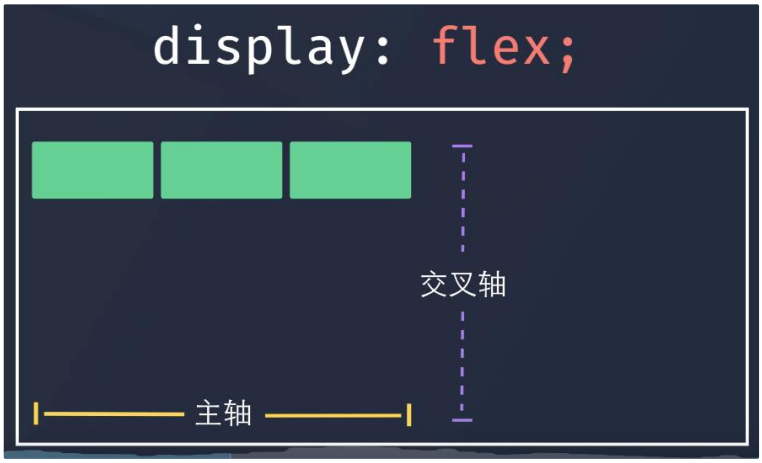
```
            background-image: url(https://images.unsplash.com/photo-1490750967868-88aa4486c946?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8Zmxvd2VyfGVufDB8fDB8fA%3D%3D&auto=format&fit=crop&w=500&q=60);
        }
        .d3 {
            background-image: url(https://images.unsplash.com/photo-1490750967868-88aa4486c946?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8Zmxvd2VyfGVufDB8fDB8fA%3D%3D&auto=format&fit=crop&w=500&q=60);
            background-repeat: no-repeat;
        }
        .d4 {
            background-image: url(https://images.unsplash.com/photo-1490750967868-88aa4486c946?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8Zmxvd2VyfGVufDB8fDB8fA%3D%3D&auto=format&fit=crop&w=500&q=60);
            background-position: top right;
        }
        .d5 {
            background-image: url(https://images.unsplash.com/photo-1490750967868-88aa4486c946?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8Zmxvd2VyfGVufDB8fDB8fA%3D%3D&auto=format&fit=crop&w=500&q=60);
            background-size: 50% 50%;
        }
    </style>
</body>
</html>
```

- 运行结果



6）flex弹性布局

- 在之前要控制HTML元素的布局，会用到padding、margin、postion、float等方 法，经过反反复复调试才能实现效果。 自从Flex弹性布局出现，一切似乎豁然开朗！
- 启用Flex布局，只需要在外部元素设置display: flex属性。
- Flex布局有一个隐式的坐标空间，水平方向有一条主轴，垂直方向有一条交叉轴：



- 属性

属性	描述	值
justify-content	改变主轴（横向）的布局	- flex-start:靠左对齐 - flex-end:靠右对齐 - center:居中对齐 - space-evenly:平分空间 - space-between:两端对齐
align-items	改变交叉轴（竖向）的布局	- flex-start:靠上对齐 - flex-end:靠下对齐 - center:居中对齐
flex-direction	设置主轴的方向	- row;（默认值）主轴水平方向，从左往右 如图: - row-reverse; 主轴水平方向的逆方向，从右往左 - column; 主轴为垂直方向，从上往下 - column-reverse; 主轴为垂直方向的逆方向，从下往上
flex-wrap	此属性定义，如果一条轴线上排列不下，换行的方式	- nowrap; （默认）不换行 如图: - wrap; 正常换行 从上到下 如图: - wrap-reverse; 逆方向换行 从下到上

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <div class="d0">
        <div class="d1">1</div>
        <div class="d2">2</div>
        <div class="d3">3</div>
        <div class="d4">4</div>
```

```
        <div class="d5">5</div>
    </div>
    <style type="text/css">
        .d0 {
          display: flex;
          flex-direction: row-reverse;
          justify-content: center;
        }
        .d1, .d2, .d3, .d4, .d5 {
            height: 100px;
            width: 100px;
            margin-bottom: 10px;
            background-color: aquamarine;
            text-align: center;
            line-height: 100px;
            margin-right: 10px;
        }
    </style>
</body>
</html>
```

- 运行结果

四、JavaScript 客户端脚本语言

1. JavaScript发展史

2. JavaScript介绍

- **JavaScript**(简称JS):是一种轻量级客户端脚本语言，通常被直接嵌入 HTML 页面，在浏览器上执行。
- **JavaScript**的主要用途:
  - 使网页具有交互性，例如响应用户点击，给用户提供更好的体验
  - 处理表单，检验用户输入，并及时反馈提醒
  - 浏览器与服务端进行数据通信，主要使用Ajax异步传输

- 在网页中添加标签，添加样式，改变标签属性等--对Dom节点进行操作

3. 基本使用

- 内部方式(内嵌样式)，在body标签中使用

```
<script type="text/javascript">
    <!--
    javaScript语言
    -->
</script>
```

- 外部导入方式(推荐)，在head标签中使用

```
<script type="text/javascript" src="my.js"></script>
```

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <p>在HTML中如何使用javascript</p>
    <script>
        var name = "hello"; // 定义变量
        alert(name); // 警告框方法，浏览器提示消息
        /* alert(“你好”) */ // 单行与多行注释
    </script>
</body>
</html>
```

4. 事件

- **事件**指的是当HTML中发生某些事件时所调用的方法(处理程序)。 例如点击按钮，点击后做相应操作，例如弹出一句话
- 用法：<*some-HTML-element* *some-event* ='JavaScript 代码'>
- 常用事件

事件	描述
onclick	用户点击 HTML 元素时触发
onmouseover	鼠标指针移动到指定的元素上时触发
onkeydown	用户按下键盘按键
onchange	HTML元素改变时触发
onmouseout	用户从一个 HTML 元素上移开鼠标时发生
onload	浏览器已完成页面的加载

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body onkeydown="alert('onkeydown')">
    <button onclick="alert('onclick')">onclick</button>
    <button onmouseover="alert('onmouseover')">onmouseover</button>
    <button onmouseout="alert('onmouseout')">onmouseout</button>
    <br><br>
    输入你的名字: <input type="text" id="fname" onchange="myFunction()">
    <p>当你离开输入框后，将小写字母转为大写字母。</p>

    <script type="text/javascript">
        window.onload = f;
        function f() {
            alert("页面加载完毕");
        }
        function myFunction(){
            var x=document.getElementById("fname");
            x.value=x.value.toUpperCase();
        }
    </script>
</body>
</html>
```

5. 选择器

- JS选择器主要用来获取HTML页面中的元素,将页面中的元素保存到一个对象中,然后就可以对这些对象的属性值进行相应操作,以实现一些动态效果,以达到页面的生动,易用。
- 需要注意的一点是操作的一定是对象,直接将元素当做对象使用是不行的。
- **常见选择器:**标签选择器、类选择器、ID选择器
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <button type="button" id="btn">点我-id</button>
    <div id="main">
        <p>Hello world!1</p>
        <p>Hello world!2</p>
        <p>Hello world!3</p>
    </div>

    <script type="text/javascript">
        var z = document.getElementById('btn'); //获取id为btn的元素
        z.onclick = function () {
            alert('亲，有什么可以帮助你的-id?')
        }

        var x = document.getElementById('main'); //获取id为main的元素
        var y = x.getElementsByTagName("p"); // 返回的是一个数组，下标获取
        console.log(z, y)
        document.write('div中的第二段文本是:' + y[1].innerHTML); //向当前文档写入内容
    </script>
</body>
</html>
```

6. JS操作HTML

- 插入内容

  document.write(“ 这是JS写入的段落 ”); //向文档写入HTML内容

  x = document.getElementById(‘demo’); //获取id为demo的元素

  x.innerHTML=“<a>Hello” //向元素插入HTML内容

- 改变标签属性:

  document.getElementById(“image”).src=“b.jpg“ //修改img标签src属性值

- 改变标签样式

  x = document.getElementById(“p”) //获取id为p的元素

  x.style.color=“blue” //字体颜色

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
```

```
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <input id="input"></input>
    <p id="demo"></p>
    <script type="text/javascript">
        //直接写入到body中
        document.write('<p>这是JS写入的段落</p>')
        //通过id选择器写入
        x = document.getElementById('demo')
        x.innerHTML = "Hello"
        //修改标签属性
        document.getElementById('input').type = 'button'
        console.log(document.getElementsByClassName('p1'))
        //修改标签样式
        x.style.color = "blue"
    </script>
</body>
</html>
```

7. 数据类型：字符串、数组、对象

- 在JS中，数据类型有:字符串、数字、布尔、数组、对象、Null、Undefined

7.1 字符串

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        var s = "hello world"
        //字符串长度
        console.log("字符串长度:", s.length)
        //根据索引获取值
        console.log("第5个字符:", s[4])
        //替换字符
        console.log(s.replace('h', 'H'))
        //字符串转数组
        console.log(s.split(' '))
        //数组转字符串
        var arr = s.split(' ')
        console.log(arr.join(','))
```

```
        //找到返回匹配的字符，否则返回null
        console.log(s.match('w'))
        //字符串拼接
        console.log(s + "aaa")
    </script>
</body>
</html>
```

7.2 数组

- **数组**是一个序列的数据结构。
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        //数组定义
        var computer = new Array()
        //或者
        var computer1 = ["主机", "显示器", "键盘", "鼠标"]
        console.log(computer, computer1)
        //向数组中添加元素
        computer[0] = "联想"
        computer[1] = "华为"
        computer[2] = "小米"
        //或者
        computer.push("华硕")
        console.log(computer)
        //通过索引查找元素
        console.log(computer[1])
        //数组长度
        console.log(computer.length)
        //数组遍历
        for(var i = 0; i < computer.length; i++){
            console.log(computer[i])
        }
        //数组删除
        console.log(computer.slice(0, computer.length - 1))
    </script>
</body>
</html>
```

7.3 对象

- **对象**是一个具有映射关系的数据结构。用于存储有一定关系的元素。
- **格式** `d = {'key1':value1, 'key2':value2, 'key3':value3}`
- 注意:对象通过key来访问value，因此字典中的key不允许重复。
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        //对象的定义
        var user = {
            name: '张三',
            sex: '男',
            age: '30'
        }
        console.log(user)
        //通过属性名查询值
        console.log(user.name)
        //或者
        console.log(user['sex'])
        //增加或修改
        user.height = "180cm"
        console.log(user)
        user['height'] = '175cm'
        console.log(user)
    </script>
</body>
</html>
```

8. 操作符、流程控制

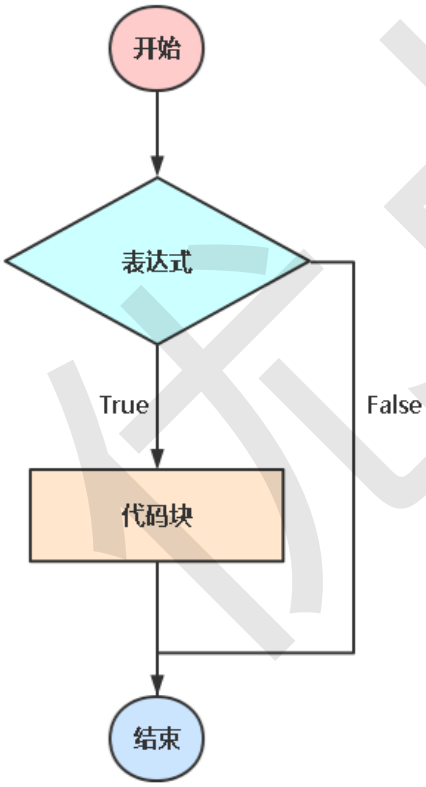
8.1 操作符

- 一个特定的符号，用它与其他数据 类型连接起来组成一个表达式。常用于条件 判断，根据表达式返回True/False采取动作。
- 常用操作符
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        //比较操作符
        console.log(1 == 2)
        //算数操作符
        console.log(1 + 2)
        var num = 0
        num++
        console.log(num)
        //逻辑操作符
        console.log(num > 0 && num > -1)
        //赋值操作符
        num += 1
        console.log(num)
    </script>
</body>
</html>
```

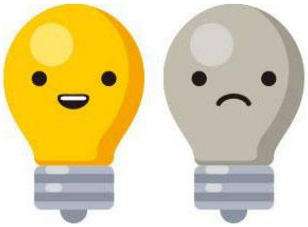
8.2 条件判断



- **if条件判断:**判定给定的条件是否满足(True或False)， 根据判断的结果决定执行的语句。
- 语法:

```
if (表达式) {
    <代码块>
} else if (表达式) {
    <代码块>
} else {
    <代码块>
}
```

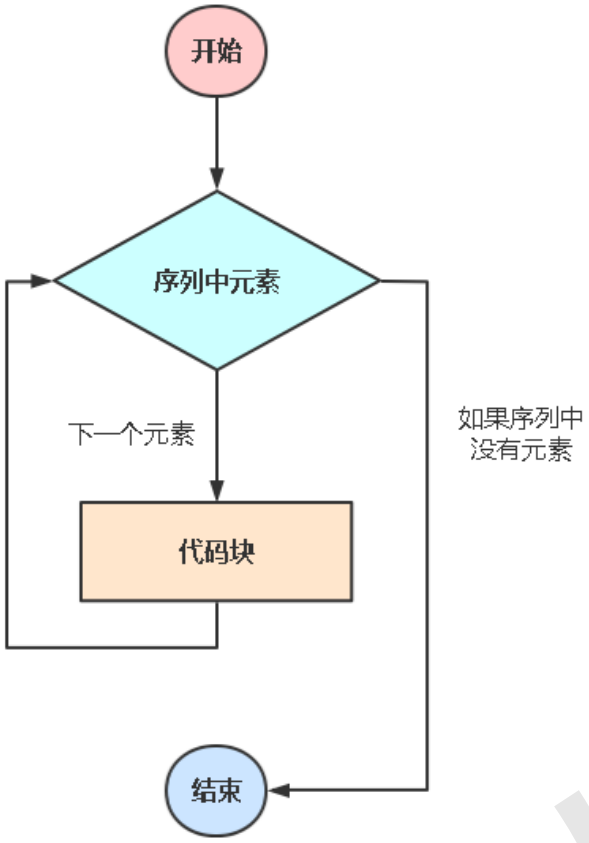
- 代码示例:根据用户点击做不同操作



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <img id="myimage" src="img/off.jpg" alt="">
    <button type="button" onclick="changeImage('on')">开灯</button>
    <button type="button" onclick="changeImage('off')">关灯</button>

    <script type="text/javascript">
        function changeImage(status) {
            x = document.getElementById('myimage');
            if (status == 'on') {
                x.src = "img/on.jpg";
            } else if (status == 'off') {
                x.src = "img/off.jpg";
            }
        }
    </script>
</body>
</html>
```

8.3 for循环



- **for循环：**一般用于遍历数据类型的元素进行处理，例如字符串、数组、对象。
- 语法：

```
for (<变量> in <序列>) {
    <代码块>
}
```

- 代码示例：遍历数组和对象

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        //遍历数组
        var computer = ["主机","显示器","键盘","鼠标"];
        //方式1
        for(i in computer) {
            console.log(computer[i]) // 使用索引获取值
        }
        //方式2
        computer.forEach(function (e) {
```

```
            console.log(e)
        })
        //遍历对象
        var user = {name:"李四",sex:"男",age:"30"};
        //方式1
        for(let k in user) {
            console.log(k + ":" + user[k])
        }
        //方式2
        Object.keys(user).forEach(function (k) {
            console.log(k + ":" + user[k])
        })
    </script>
</body>
</html>
```

9. 函数

- **函数:**是指一段可以直接被另一段程序或代码引用的程序或代码。 在编写代码时，常将一些常用的功能模块编写成函数，放在函数库中供公共使用，可 减少重复编写程序段和简化代码结构。

9.1 普通函数

- 语法:

```
function 函数名称(参数1, 参数2, ...) {
    <代码块>
    return <表达式>
}
```

- 示例代码

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <button type="button" onclick="hello()">你好</button>
    <button type="button" onclick="myFunc('张三', '30')">点我</button>
    <script type="text/javascript">
        //定义方法，无参
        function hello() {
            alert("hello world")
        }
        //定义方法，接收参数
        function myFunc(name, age) {
            alert("欢迎" + name + ", 今年" + age);
```

```
        }
    </script>
</body>
</html>
```

9.2 匿名函数与箭头函数

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <script type="text/javascript">
        //普通函数
        function sum1(x,y) {
            return x+y
        }
        //匿名函数
        sum2 = function(x,y) {
            return x+y
        }
        //箭头函数
        sum3 = (x,y) => {
            return x+y
        }
        console.log(sum1(1,2))
        console.log(sum2(3,4))
        console.log(sum3(5,6))
    </script>
</body>
</html>
```

10. window对象

- 在JavaScript中，一个浏览器窗口就是一个window对象。
- 一个窗口就是一个window对象，这个窗口里面的HTML文档就是一个document对象，document对象是window对象的子对象。
- location也是windows的子对象，用于操作URL地址
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
</head>
<body>
    <button type="button" onclick="location.reload()">刷新当前页面</button>
    <button type="button" onclick="location.href=location.href">重新请求当前页面</button>
    <button type="button" onclick="location.href='http://www.baidu.com'">请求别的页面</button>
</body>
</html>
```

五、JavaScript库：jQuery,简化编程

1. jQuery介绍

> 官方网站：https://jquery.com

- **jQuery 是一个 JavaScript 库**。极大地简化了 JavaScript 编程，例如JS原生代码几十行 实现的功能，jQuery可能一两行就可以实现，因此得到前端程序猿广泛应用。
- 发展至今，主要有三个大版本：
  - 1.x:常用版本
  - 2.x，3.x:除非特殊要求，一般用的少

2. 基本使用

- cdn导入方式

```
<head>
    <script type="text/javascript" src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <script type="text/javascript">
        // jquery代码
    </script>
</body>
```

- **基础语法是:$(selector).action()**
  - $:代表jQuery本身
  - (selector):选择器，查找HTML元素

    - action():对元素的操作
- 代码示例：按钮实现

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <button type="button" id="btn1">点我1</button>
    <button type="button" id="btn2">点我2</button>

    <script type="text/javascript">
        // js实现
        var x = document.getElementById("btn1")
        x.onclick = function () {
            alert('亲，有什么可以帮助你的?')
        }
        // jquery实现
        $("#btn2").click(function () {
            alert('亲，有什么可以帮助你的?')
        })
    </script>
</body>
</html>
```

3. 选择器

- 选择器介绍

名称	语法	示例
标签选择器	element	$(“h2”) 选取所有h2元素
类选择器	.class	$(“.title“) 选取所有class为title的元素
ID选择器	#id	$(“#title”) 选取id为title的元素
并集选择器	selector1,selector2,...	$(“div,p,.title”) 选取所有div、p和拥有class为title的元素
属性选择器		- $(“input[name]=‘username’”) 选取input标签名为username的元素 - $(“[href=‘#’]”) 选取href值等于“#”的元素

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet" /> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <h2>在HTML中如何使用jquery</h2>
    <button class="btn">点击</button>
    <button id="btn">点击1</button>
    <script type="text/javascript">
        // 标签选择器
        $("h2").click(function() {
            alert("你好, 标签")
        })
        //类选择器
        $(".btn").click(function() {
            alert("你好, 类")
        })
        //id选择器
        $("#btn").click(function() {
            alert("你好, id")
        })
    </script>
</body>
</html>
```

4. 操作HTML

- **隐藏和显示元素:**
  - hide() :隐藏某个元素
  - show() :显示某个元素
  - toggle() :hide()和show()方法之间切换
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet" /> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <p id="demo">这是一个段落。</p>
```

```
    <button id="hide" type="button">隐藏</button>
    <button id="show" type="button">显示</button>
    <button id="toggle" type="button">切换</button>

    <script type="text/javascript">
        $("#hide").click(function () {
            $("p").hide();
        })
        $("#show").click(function () {
            $("p").show();
        })
        $("#toggle").click(function () {
            $("p").toggle();
        })
    </script>
</body>
</html>
```

- **获取与设置内容：**
  - text() 设置或返回所选元素的文本内容
  - html() 设置或返回所选元素的HTML内容
  - val() 设置或返回表单字段的值
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <p id="txt">这是一个<b>段落</b>。</p>
    <button type="button" id="btn1">显示文本</button>
    <button type="button" id="btn2">显示HTML</button>
    <p id="demo"></p>

    <script type="text/javascript">
        $("#btn1").click(function () {
            x = $("#txt").text();
            console.log(x)
            $("#demo").text(x).css("color","red") //不会解析b标签
        })
        $("#btn2").click(function () {
            x = $("#txt").html(); //获取
            console.log(x)
            $("#demo").html(x).css("color","red") //会解析b标签, .html()设置
        })
```

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

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <h1>欢迎访问运维管理系统</h1>
    用户名:<input type="text" id="uname" name="username"><br>
    密码:<input type="text" id="pwd" name="password"><br>
    <button type="button" id="btn">显示输入内容</button>
    <p id="demo"></p>

    <script type="text/javascript">
        $("#btn").click(function () {
            x = $("#uname").val();
            y = $("#pwd").val();
            $("#demo").text(x + ', ' + y).css("color","red")
        })
    </script>
</body>
</html>
```

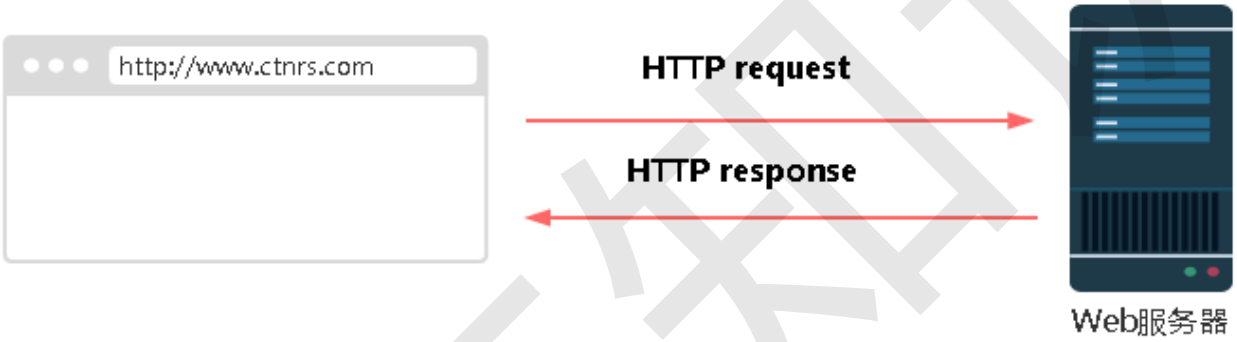
- **设置CSS样式：**
  - css() 设置或返回样式属性(键值)
  - addClass() 向被选元素添加一个或多个类样式
  - removeClass() 从被选元素中删除一个或多个类样式
  - toggleClass() 对被选元素进行添加/删除类样式的切换操作
- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <div id="demo">
        <p>这是一个段落</p>
    </div>
```

```
    <button id="btn">添加样式</button>

    <script type="text/javascript">
        $("#btn").click(function () {
            $("#demo p").css("color", "red")
            // $("#demo p").css({"color":"red","font-size": "30px"})
            // $("#demo").addClass("cls")
            // $("#demo").removeClass("cls")
        })
    </script>
</body>
</html>
```

5. jQuery Ajax前后端数据交互



- 浏览器访问网站一个页面时，Web服务器处理完后会以消息体方式返回浏览器，浏览器自动解析HTML 内容。如果局部有新数据需要更新，需要刷新浏览器重新发起页面请求获取最新数据，如果每次都是通 过刷新解决这个问题，势必会给服务器造成负载加重，页面加载缓慢。
- Ajax(Asynchronous JavaScript And XML，异步JavaScript和XML)，AJAX 是一种在无需重新加载 整个网页的情况下，能够更新部分网页的技术。例如在不刷新页面的情况下查询数据、登录验证等
- **无刷新的好处：**
  - 减少带宽、服务器负载
  - 提高用户体验

5.1 基本使用

- **jQuery Ajax主要使用$.ajax()方法实现，用于向服务端发送HTTP请求。**
- 语法：$.ajax([settings]);
- settings 是$.ajax ( )方法的参数列表，用于配置 Ajax 请求的键值对集合，参数如下：

参数	类型	描述
url	string	发送请求的地址，默认为当前页地址
type	string	请求方式，默认为GET
data	object、 array、string	发送到服务器的数据
dataType	string	预期服务器返回的数据类型，包括JSON、XML、text、HTML等
contentType	string	发送信息至服务器时内容编码类型。默认值："application/x-www-form-urlencoded"。
timeout	number	设置请求超时时间
headers	obejct	设置请求头信息
async	Boolean	默认true，所有请求均为异步请求。设置false发送同步请求

- 代码示例

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>文档的标题</title>
    <!-- <link href="main.css" type="text/css" rel="stylesheet"/> -->
    <script type="text/javascript"
src="https://cdn.bootcdn.net/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <div id='demo'>
        <p id='notice' style="color: red;"></p>
        <h1>用户列表</h1>
        <ul></ul>
    </div>

    <script type="text/javascript">
        $.ajax({
            type: "GET",
            url: "http://127.0.0.1/api/books",
            success: function (result) { // result是API返回的JSON数据
                if(result.code == 200) {
                    for (i in result.data) {
                        $('#demo ul').append("<li>" + result.data[i]['username'] + "
</li>"); // 将li标签追加到ul标签
                    }
                } else {
                    $('#notice').text('数据获取失败!')
                }
            },
            error: function() {
```

```
                $('#notice').text('连接服务器失败，请稍后再试!')
            }
        })
    </script>
</body>
</html>
```

5.2 回调函数

- 回调函数:参数引用一个函数，并将数据作为参数传递给该函数。
- jqXHR:一个XMLHttpRequest对象

参数	函数格式	描述
beforeSend	function(jqXHR,object)	发送请求前调用的函数，例如添加自定义 HTTP 头
success	function(data,textStatus,jqXHR)	请求成功后调用的函数，参数data:可选， 由服务器返回的数据(JSON)
error	function(jqXHR,textStatus,errorThrown)	请求失败时调用的函数
complete	function(jqXHR, textStatus)	请求完成后(无论成功还是失败)调用的 函数