目录

第一章：BS 模式开发之 Web 编程

1.1 概述

学习网站：http://www.w3school.com.cn/

1.1.1 常见的 Web 技术

1、web 前端开发技术

(1) HTML、CSS、XML、Javascript、AJAX

HTML 简介:

超文本标记语言(英文全称:HyperText Markup Language)

“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素

CSS 简介：

CSS: 层叠样式表(英文全称：Cascading Style Sheets)

CSS 是一种定义样式结构如字体、颜色、位置等的语言，被用于描述网页上的信息格式化和现实的方式

XML 简介：

XML 指可扩展标记语言（EXtensible Markup Language）

XML 是一种标记语言，很类似 HTML

XML 的设计宗旨是传输数据，而非显示数据

XML 是各种应用程序之间进行数据传输的最常用的工具

Javascript 简介：

JavaScript 是一种属于网络的脚本语言,已经被广泛用于 Web 应用开发,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。通常 JavaScript 脚本是通过嵌入在 HTML 中来实现自身的功能的。

AJAX 简介：

Ajax 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术

2、Web 服务器端开发技术

(1) CGI、ASP、PHP

CGI 简介：

CGI（英文全称：Common GatewayInterface）通用网关接口

CGI 是 Web 服务器运行时外部程序的规范,按 CGI 编写的程序可以扩展服务器功能。CGI 应用程序能与浏览器进行交互,还可通过数据库 API 与数据库服务器等外部数据源进行通信,从数据库服务器中获取数据。

你可以简单的认为 CGI 程序是服务器端的一个可执行程序。

ASP 简介：

ASP（英文全称：Active Server Pages）动态服务器页面

是 MicroSoft 公司开发的服务器端脚本环境,可用来创建动态交互式网页并建立强大的 web 应用程序。

ASP 提供了一些内置对象，使用这些对象可以使服务器端脚本功能更强

PHP 简介：

PHP（外文名:PHP: Hypertext Preprocessor，中文名：“超文本预处理器”）是一种通用开源脚本语言。语法吸收了 C 语言、Java 和 Perl 的特点，利于学习，使用广泛，主要适用于 Web 开发领域

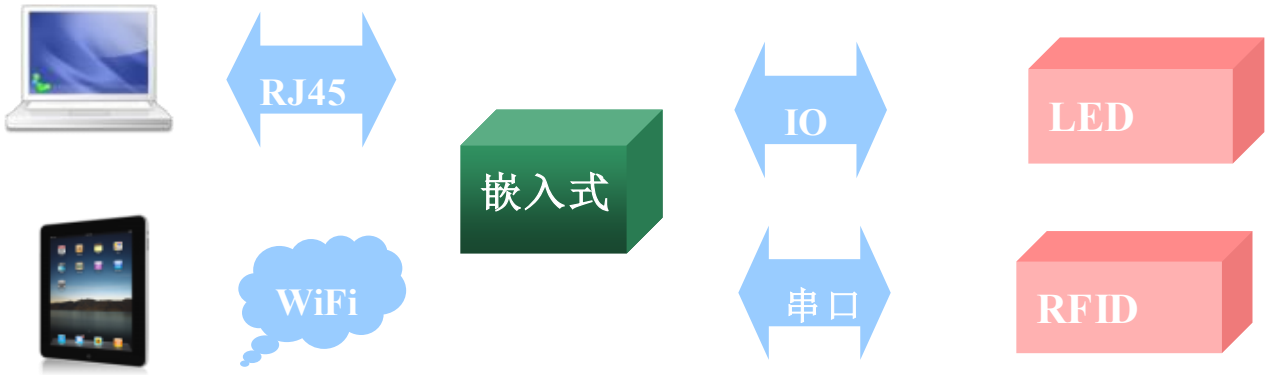
3、数据管理

(1) Oracle、MySQL、SQLServer、SQLite

1.1.2 嵌入式 web 开发模型

功能展示：

模型：



功能：通过网页实现对 LED(GPIO)和 RFID 模组（UART）监控

架构：B/S 架构

1.1.3 B/S 架构

B/S 架构

Browser/Server（浏览器/服务器结构），是随着 Internet 技术的兴起，是对 C/S 结构的一种变化或者改进的结构。

用户界面完全通过 www 浏览器实现，一部分事物逻辑在前端实现，但是主要事务逻辑在服务器端实现。

随着 Windows98/Windows2000 将浏览器技术植入操作系统内部，这种结构更成为当今应用软件的首选体系结构。

B/S 架构 与 C/S 架构对比

C/S 模式

客户/服务器模式

胖客户/瘦服务器

QQ、微信、飞信

B/S 模式

浏览器/服务器模式

瘦客户/胖服务器

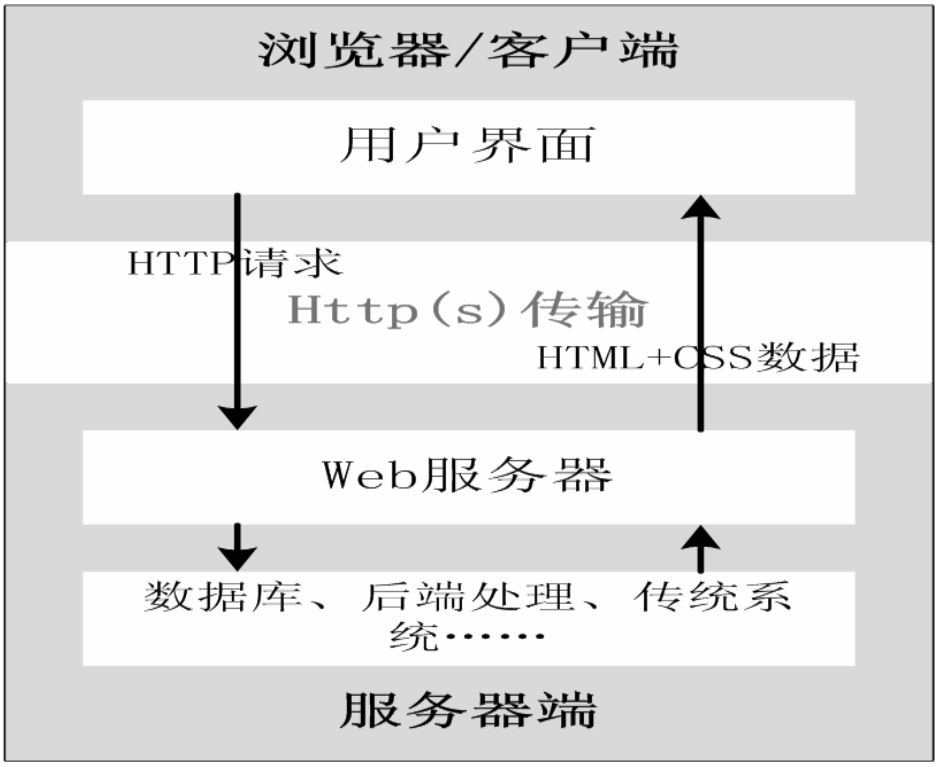
浏览器就可完成操作

WebQQ、七雄争霸、洛克王国

角度	C/S	B/S
硬件环境	专用网络	广域网
安全要求	面向相对固定的用户群 信息安全的控制能力很强	面向是不可知的用户群 对安全的控制能力相对弱

程序架构	更加注重流程 系统运行速度可较少考虑	对安全以及访问速度要多重的考虑 B/S 结构的程序架构是发展的趋势
软件重用	差	好
系统维护	升级难	开销小、方便升级
处理问题	集中	分散
用户接口	与操作系统关系密切	跨平台，与浏览器相关
信息流	交互性低	交互密集

1.1.4 Web 原理



1.1.5 boa web 服务器移植

详情参见文档

第二章：HTML 语言

2.1 什么是 HTML?

1、HTML 是超文本标记语言(Hyper Text Markup Language)

2、HTML 由各种各样的标签(tag)组成，如<html></html>、<body></body>

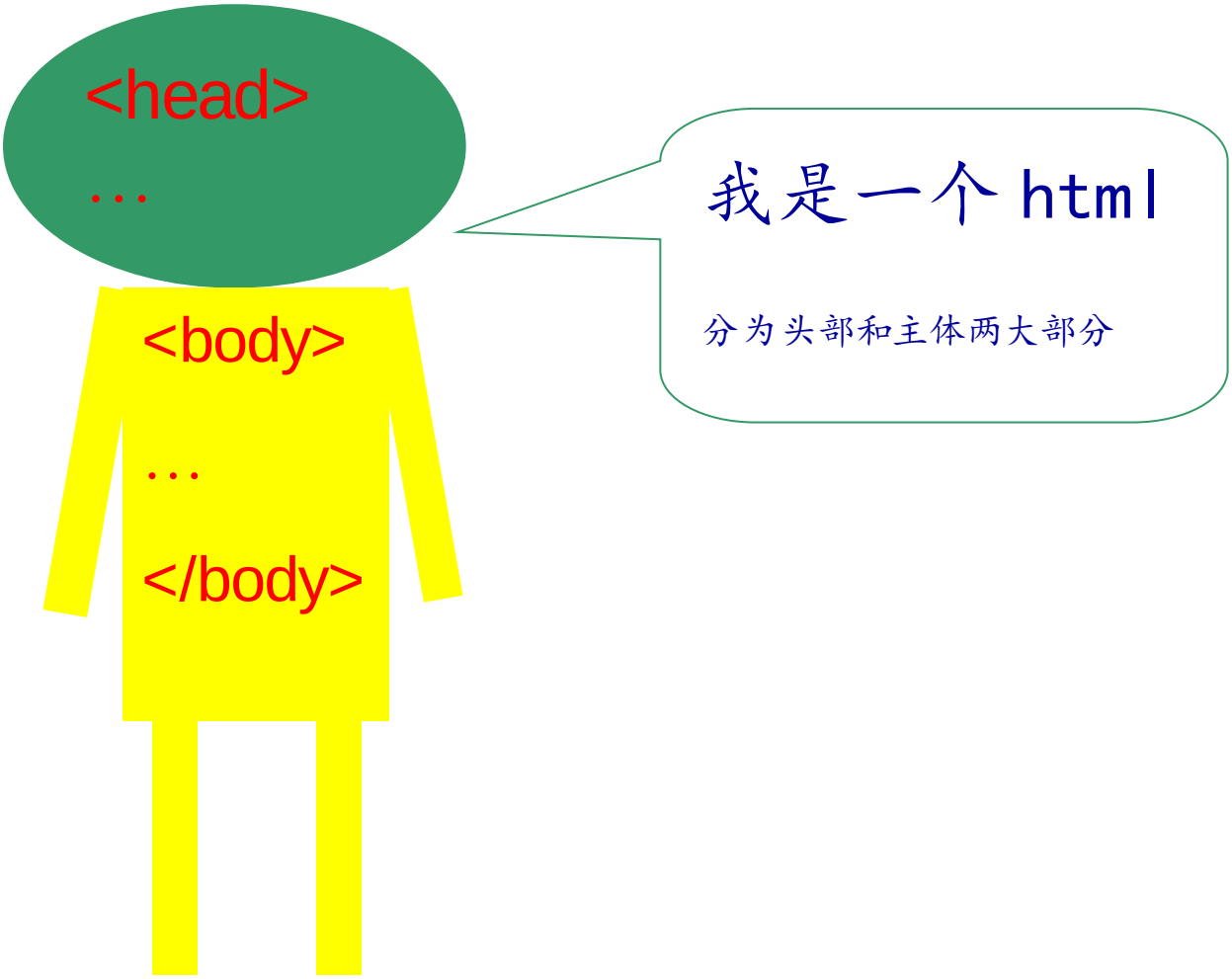
3、HTML 文档 = 网页
（1）一种纯文本文件，扩展名为.htm 或.html
（2）最终显示结果取决于 Web 浏览器的显示风格及其对标记的解释能力
（3）编辑工具：记事本，写字板、Sublime、FrontPage、Dreamweaver 等

2.2 HTML 语法

1、HTML 标签
（1）由尖括号包围的关键词，比如<html>
（2）通常是成对（开始标签，结束标签）出现的比如,<label></lable>例外：
<img>
（3）注释标签：<!--注释-->、<comment>注释</comment>
2、HTML 元素
（1）开始标签(start tag)到结束标签(end tag)的所有代码
3、HTML 属性
（1）在 HTML 元素的开始标签中规定
（2）以名称/值对的形式出现:<img src=“1.png">

开始我的 HTML 之旅

2.3 HTML 标签



2.3.1 HTML 头部<head></head>

主要放置标题标签、元信息标签等

2.3.2 HTML 主体<body></body>

放置页面中所有的内容，如文字、标题、链接、图片、表格、表单等

例子(01_first_html.html)：

```
<!-- this is my first html(the content cannot display in html,because it's just a note)-->
<html>
    <head>
        <title>01_first_html</title>
    </head>
    <body>
        <label>hello, HTML!!!</label>
        <br>
        <img src="./image/01_first_html.png">
    <body>
</html>
```

2.3.3 标题标记<title></title>

声明网页标题，指示网页作用。
默认为网页另存为后的网页文件名字
默认为收藏此网页时的收藏夹中的名字

2.3.3 元信息标记<meta>

1、提供有关页面的元信息，比如针对搜索引擎和更新频度的描述和关键词
2、<meta> 标签位于文档的头部，不包含任何内容。
3、<meta> 标签的属性定义了与文档相关联的名称/值对。
4、在 HTML 中，<meta> 标签没有结束标签。
5、<meta> 标签永远位于 head 元素内部。
6、元数据总是以名称/值的形式被成对传递的。

meta 元素必须的属性：

属性	值	描述
content	some_text	定义与 http-equiv 或 name 属性相关的元信息

meta 元素可选的属性：

属性	值	描述
http-equiv	content-type expires refresh set-cookie	把 content 属性关联到 HTTP 头部。
name	author description keywords generator revised others	把 content 属性关联到一个名称。
scheme	some_text	定义用于翻译 content 属性值的格式。

用法：

<meta 属性=值 content=值>

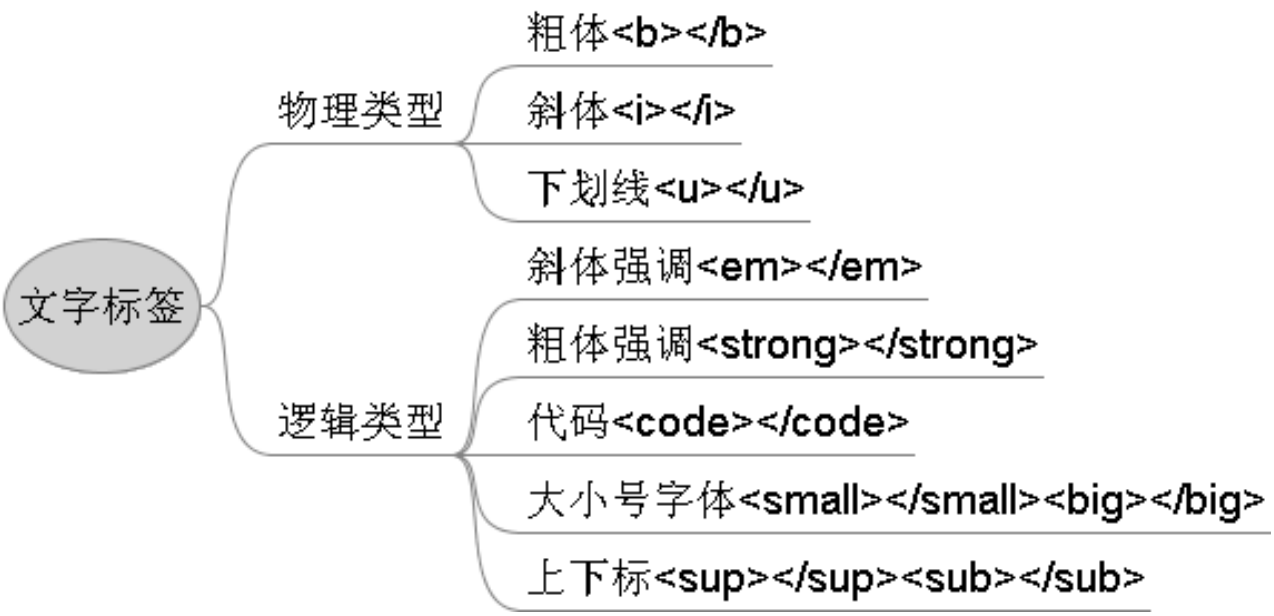
例如：<meta name=“keywords”
content=“W3school”>

例子（02_meta.html）：

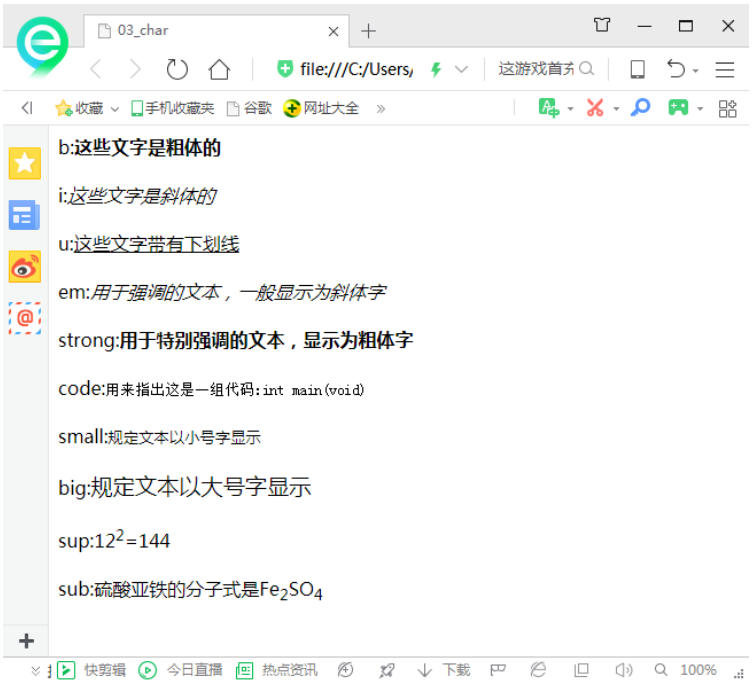
```
<html>
    <head>
        <title>02_meta</title>
        <meta http-equiv="content-type" content="text/html; charset=gb2312" />
        <meta http-equiv="refresh" content="5;url=http://www.baidu.com/">
        <meta name="keywords" content="jump">
    </head>
    <body>
        5 秒后我们将去一个神奇的地方！(这个网页的关键字是：jump)
    </body>
</html>
```

注：content ： 内容

2.3.5 文字标签（<b>、<i>、<u>、<big>、<small>……）



例：（03_char.html）：



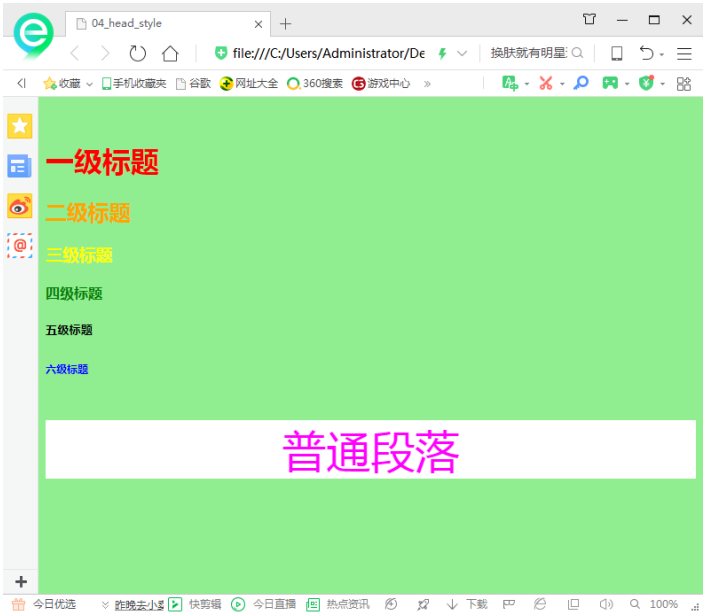
例子（03_char）：

```
<html>
    <head>
    <title>03_char</title>
    </head>
```

```
<body>
    b:<b>这些文字是粗体的</b>
    <br>
    <br>
    i:<i>这些文字是斜体的</i>
    <br>
    <br>
    u:<u>这些文字带有下划线</u>
    <br>
    <br>
    em:<em>用于强调的文本，一般显示为斜体字</em>
    <br>
    <br>
    strong:<strong>用于特别强调的文本，显示为粗体字</strong>
    <br>
    <br>
    code:<code>用来指出这是一组代码:int main(void)</code>
    <br>
    <br>
    small:<small>规定文本以小号字显示</small>
    <br>
    <br>
    big:<big>规定文本以大号字显示</big>
    <br>
    <br>
    sup:12<sup>2</sup>=144
    <br>
    <br>
    sub:硫酸亚铁的分子式是 Fe<sub>2</sub>SO<sub>4</sub>
    <br>
    <br>
</body>
</html>
```

2.3.6 标题标签：（<h1>-<h6>）

例:04_head_style.html



代码：

```
<html>
    <head>
        <title>04_head_style</title>
    </head>
    <body style="background-color:#90EE90">
        <br>
        <h1 style="color:red">一级标题</h1>
        <h2 style="color:orange">二级标题</h2>
        <h3 style="color:yellow">三级标题</h3>
        <h4 style="color:green">四级标题</h4>
        <h5 style="color:black">五级标题</h5>
        <h6 style="color:blue">六级标题</h6>
        <p style="background-color: white; font-family: arial; color: #ff00ff; font-size:50px;
text-align: center">普通段落</p>
        <br>
    </body>
</html>
```

style：样式

2.3.7 HTML 样式

HTML 的 style 属性提供了一种改变所有 HTML 元素的样式的通用方法。

可以通过 style 来设置背景颜色、字体、字体颜色、字体尺寸、对齐方式。

格式：

<TAG style="background-color: white; font-family: arial; color: #ff00ff; font-size:50px; text-align: center"></TAG>

说明：

HTML 不推荐使用 、<font>、<align>、<color>、<bgcolor>等来设置 HTML 样式，style、CSS 成为首选

2.3.8 超链接标签

超链接<a>

（1）电子邮件超链接

注：subject ---主题

（2）页面内的超链接

回到顶部：

回到某一位置：

（3）页面外的超链接

外网：

内网：

（4）图片超链接

<img src="image/a.jpg">

例子(05_a_href）：

```
<html>
    <head>
        <title>05_a_href</title>
    </head>
    <body>
        <br>
        <a href="mailto:kitty_zjy@126.com?subject=Hi">email to others</a>
        <br>
        <br>
        <a href="#mike">去中间看看</a>
```

```
<br>
<br>
<a href="http://http://www.1000phone.com//">千锋，用良心做教育</a>
<br>
<br>
<a href="a.html" target="_blank">打开本地一个新的网页</a>
<br>
<br>
<a href="a.html"><img src="image/a.jpg"></a>
<br>
<br>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p><a name="mike">中间</a></p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
<p>some information</p>
```

```
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <p>some information</p>
        <a href="#top">回到顶部</a>
        <br>
    </body>
</html>
```

注：

打开本地一个新的网页

_blank

浏览器总在一个新打开、未命名的窗口中载入目标文档

2.3.9 表格标签<table></table>

表格标记

标 签	描 述
<table>...</table>	用于定义一个表格开始和结束
<caption> ...</caption>	定义表格的标题。在表格中也可以不用此标签。
<th>...</th>	定义表头单元格。表格中的文字将以粗体显示，在表格中也可以不用此标签，<th>标签必须放在<tr>标签内
<tr>...</tr>	定义一行标签，一组行标签内可以建立多组由<td>或<th>标签所定义的单元格
<td>...</td>	定义单元格标签，一组<td>标签将将建立一个单元格，<td>标签必须放在<tr>标签内

例：06_table_web_resume.html

代码：

```
<html>
    <head>
        <title>06_table</title>
    </head>
    <body>
```

```
        <table border=10>
            <caption>table
            </caption>
            <tr align=center>
                <th colspan=3>学生信息
                </th>
                <th colspan=2>成绩
                </th>
            </tr>
            <tr align=center>
                <th>姓名
                </th>
                <th>性别
                </th>
                <th>专业
                </th>
                <th>课程
                </th>
                <th>分数
                </th>
            </tr>
            <tr align=center>
                <td>kitty
                </td>
                <td>女
                </td>
                <td>电子信息
                </td>
                <td>web 应用开发
                </td>
                <td>80
                </td>
            </tr>
        </table>
    </body>
</html>
```

2.3.10 表单<form></form>

HTML 页面与服务器交互的手段

(1) 属性

name：表单的名称

method：表单数据从浏览器传输到服务器的方法

get：将表单数据附加在 URL 地址后面，长度不超过 8192 个字符,不具有保密性，默认为 get

post：将表单数据包含在表单的主体中，一起传输到服务器上。没有长度限制,密文传输

action：用来定义表单处理程序

(2) <form></form>内的标签

a、<input> 表单输入标签

常用的文本域、按钮都是使用这个标签

属性：

name 域名称

type 域类型

value 元素值

type 属性值:

text 文字域 password密码域

file 文件域 checkbox 复选框

radio 单选框 button 普通按钮

submit 提交按钮 reset 重置按钮

hidden 隐藏域 image 图像域

b、选择列表 <select><option></option></select>

菜单和列表是为了节省网页的空间而产生的

属性

name 菜单和列表的名称

size 显示的选项数目

multiple 列表中的选项为多项

selected 默认被选中的选项(option 中的属性)

c、文本域 <textarea></textarea>

用来制作多行文本输入域

属性

name 文字域的名称

rows 文字域的行数

cols 文字域的列数

例:07_form.html

代码：

```
<html>
    <head>
        <title>07_form</title>
    </head>
    <body>
        <label>==============表单：&ltform&gt==============</label>
        <form name="form1" method="get" action="/cgi-bin/a.cgi">
            <br>
            用户名:<input type="text" name="user">
            <br>
            <br>
            密码 :<input type="password" name="pwd">
            <br>
            <br>
            文件 :<input type="file" name="file">
            <br>
            <br>
            性别 :
```

```
                <br>
                男:<input type="radio" name="sex" value="female"> 女:<input type="radio" name="sex" value="male">
                <br>
                <br>
                我喜欢自行车：
                <input type="checkbox" name="bike">
                <br>
                我喜欢汽车：
                <input type="checkbox" name="vehicle">
                <br>
                <br>
                <select name="cars">
                    <option value="volvo">Volvo</option>
                    <option value="BMW" selected>BMW</option>
                    <option value="Infiniti">Infiniti</option>
                    <option value="Audi">Audi</option>
                </select>
                <br>
                <br>
                说点什么吧？
                <textarea name="comment" rows="3" cols="10"></textarea>
                <br>
                <br>
                <input type="button" value="普通按钮">
                <br>
                <br>
                <input type="hidden" name="fortest" value="testinfo">
                <br>
                <br>
                <input type="image" src="./image/globe_yellow.png">
                <br>
                <br>
                <input type="submit" value="提交">
                <input type="reset" value="复位">
            </form>
        </body>
    </html>
```

第三章： Javascript

3.1 什么是 Javascript?

Javascript 是一种基于对象并具有安全性能的脚本语言，是由浏览器内解释器翻译成可执行格式后执行，在概念和设计方面，Java 和 Jvascript 是两种完全不同的语言。

Javascript 的四个特点：基于对象的语言、简单性、动态性、跨平台性

3.2 网页使用 js 脚本的三种方式

3.2.1 直接添加脚本

<input type=”button” onclick=”alert(‘欢迎’);” value=”点击”>

例子(12_js_method1)：

```
<html>
    <head>
        <title>12_js_method1</title>
    </head>
    <body>
        <input type="button" value="click me" onclick="alert('hello, everyone! My name is JS');">
    </body>
</html>
```

3.2.2 使用 script 标记插入脚本

```
<script type=“text/javascript”>
    //在这里编写 JavaScript 代码
</script>
```

例子(12_js_method2)：

```
<html>
    <head>
        <title>12_js_method2</title>
    </head>
    <body>
        <script type="text/javascript">
            alert("oh! see you again!");
```

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

3.2.3 链接脚本文件

```
<script type=“text/javascript”
  src=“文件名.js”> </script>
```

例子(12_js_method3)：

```
<html>
    <head>
        <title>12_js_method3</title>
    </head>
    <script type="text/javascript" src="12_js_method3.js"></script>
    <body>
    </body>
</html>
```

12_js_method3.js

```
alert("oh, see you the third time!");

alert("Bye Bye!!");
```

3.3 js 编程

3.3.1 js 编程概述

1、JS 支持的两种类型的注释

（1）行注释——在行末

（//注释）

（2）块注释——可以跨越多行

（/*注释*/）

2、分号是语句的结束符号

3、大小写

（1）JavaScript 是大小写敏感的，这意味着大写字母同相应的小写字母是不同的。

（2）JavaScript 的保留关键字是小写的

3.3.2 js 保留关键字（全部用小写）

break	case	catch	continue	debugger
default	delete	do	else	false
finally	for	function	if	in
Instanceof	new	null	return	switch
this	throw	true	try	typeof
var	void	while	with	

3.3.3 js 变量

（1）变量的声明

使用 var 关键字进行变量的声明：var x;

在声明的时候可以同时对变量进行初始化：var y=4;

使用逗号将多个变量隔开：var x,y=5,z='hello';

（2）变量的命名

变量必须由字母、数字和下划线组成

字母或者是下划线开头，不可以是数字

变量不可以是保留字

（3）变量区别大小写

3.3.5 js 数据类型

（1）字符串、数字、布尔、数组、对象、Nul、Undefined
（2）拥有动态类型
相同的变量可用作不同的类型

```
var x                   // x 为 undefined
var x = 6;              // x 为数字
var x = "Bill";         // x 为字符串
```

typeof（x）可以得到变量的类型
4、运算符
（1）算数运算符
+、－、*、/、%、++、－－
（2）逻辑运算符
&&、||、！
（3）比较运算符
==、>、<、>=、<=、!=、===
（4）位运算符
~、&、|、^、<<、>>、>>>(无符号右移)
（5）字符串运算符
+(合并运算符)
（6）赋值运算符
=、+=、-=、 *=、/=、%=
（7）条件运算符
?:（条件 ？结果 1 ： 结果 2）

3.3.6 js 控制语句

1、if 语句

```
if(条件表达式){
    执行语句或语句群;
}else{
    执行语句或语句群;
}
```

2、switch 语句

```
switch (表达式){
    case 值 1 :执行语句 1; break;
    case 值 2 :执行语句 2; break;
    case 值 n :执行语句 n; break;
}
```

3、for 语句
for(初始值；判断条件；调整值){执行语句或语句群;}

4、while 语句

while(条件表达式){执行语句;}

5、do while 语句

```
do{
    执行语句;
}while(条件表达式);
```

3.3.7 js 函数

函数的语法结构:

```
function 函数名(参数 1,参数 2,…){
函数体
}
```

函数参数不是函数的必选内容

在调用一个需要参数的函数时没有传递参数，JavaScript 就会将参数表示为未定义(undefined)

3.3.8 js 对象

JS 是面向对象的编程语言（OOP）

对象是一种特殊的数据类型，拥有属性和方法，其中属性是指与对象有关的值;方法是指对象可以执行的行为。

其中 JavaScript 中的常用对象：浏览器对象 Window、文本对象 Document（HTML DOM）、

内部对象 Date、 Math、 String。

3.3.8.1 浏览器对象（window 对象）

Window 对象表示浏览器中打开的窗口，打开一个 HTML 网页会创建一个 window 对象

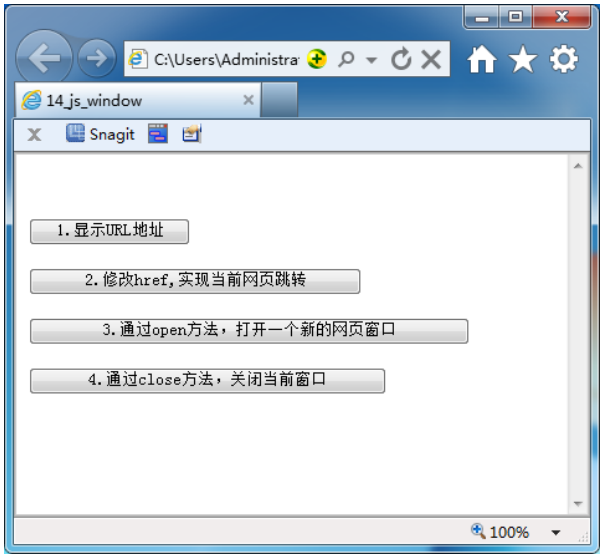
Window 对象是全局对象

window.open()打开一个新的窗口

window.close()关闭当前窗口

window.location.href：返回完整的 URL;对其进行赋值，则能够跳转到相应的网页

例:14_js_window.html



网页代码：

```
<html>
    <head>
        <title>14_js_window</title>
    </head>

    <script type="text/javascript" src="14_js_window.js"></script>

    <body>
        <br>
        <br>
        <input type="button" value="1.显示 URL 地址" onclick="window_show_href()">
        <br>
        <br>
        <input type="button" value="2. 修改 href, 实现当前网页跳转" onclick="window_set_href()">
        <br>
        <br>
        <input type="button" value="通过 open 方法，打开一个新的网页窗口" onclick="window_open()">
        <br>
        <br>
        <input type="button" value="通过 close 方法，关闭当前窗口" onclick="window_close()">
    </body>

</html>
```

14_js_window.js

```
function window_show_href()
{
    alert(window.location.href);//显示当前 URL
}

function window_set_href()
{
    window.location.href = "http://www.hao123.com/"; //修改 href，实现网页跳转
}


function window_open()
{
    window.open("http://www.baidu.com");//打开一个新的窗口
}

function window_close()
{
    window.close();//关闭当前窗口
}
```

3.3.8.2 文本对象（Document 对象）

每个载入浏览器的 HTML 文档都会成为 Document 对象。
Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问

1、提供了从 JS 脚本中对 HTML 页面中的所有元素进行访问
2、可以通过 getElementById()方法,来根据对应的 ID 号去访问、控制 HTML 页面中的标签元素
3、可以通过 title，URL 属性获取当前文档的标题，URL 信息等
4、可以通过 write 方法在 HTML 页面中写入 HTML 表达式

例:15_js_document.html

```
<html>
```

```
<head>
<title>15_js_document</title>
</head>
<script type="text/javascript" src="15_js_document.js"></script>
<body>
<label id="label2">label_text</label>
<input type="button" value="change_label" onclick="change_label()">
<br>
<br>

<input id ="input2" type="text" value="input_text">
<input type="button" value="change_input" onclick="change_input()">
<br>
<br>

<img id="img2" src="./img/document.png"   width=200 height=200 >
<input type="button" value="change_src" onclick="change_img()">
</body>
</html>
```

15_js_document.js

```
alert("title:"+document.title);
alert("URL:"+document.URL);
document.write("<h1>Hello World!</h1>");

function change_label()
{
    document.getElementById("label2").innerHTML="name";
}

function change_input()
{
    document.getElementById("input2").value="kitty";
}

function change_img()
{
```

```
    document.getElementById("img2").src="./img/change.png";
}
```

3.3.8.3 内部对象（Date 对象）

提供了操作时间和日期的方法

拥有一系列属性和方法，可以用来获取系统当前时间或者设置 Date 对象中的时间

通过 getTime()方法可返回距 1970 年 1 月 1 日 00:00:00.000（GMT 时间）到现在的毫秒数。GMT 是格林威治标准时间

（1）Date 对象方法

getFullYear()：返回当前年份

getMonth()：返回当前月份，0～11

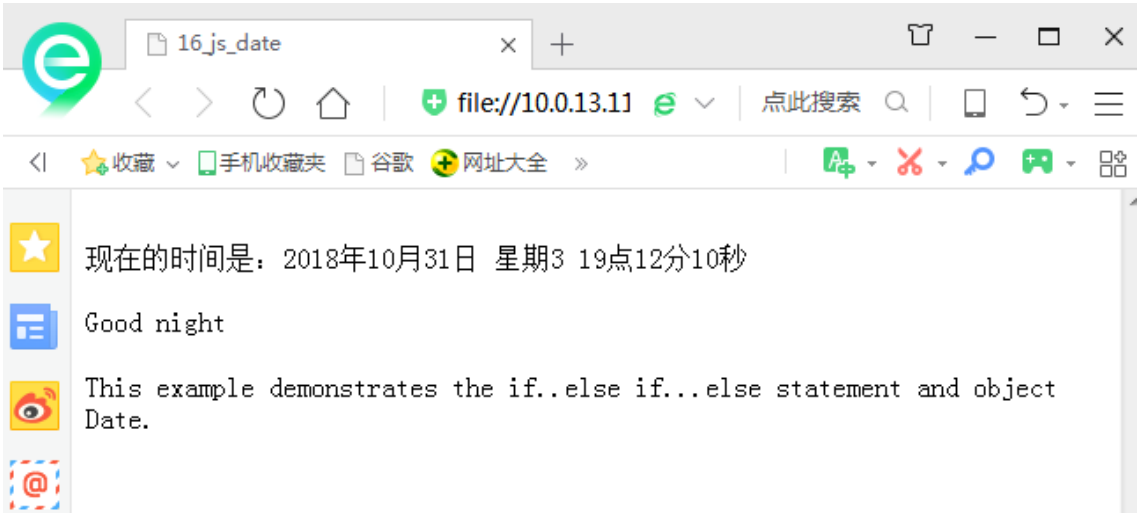
getDay()：返回星期中的某一天，0~6,0 表示周日

getDate()：返回一月中的某一天

getHours()：返回当前时间的小时，0～23

getMinutes()：返回当前时间的分钟，0～59

getSeconds()：返回当前时间的秒，0～59



例:16_js_date.html

```
<html>
    <head>
    <title>16_js_date</title>
    <script type="text/javascript" src="16_js_date.js"></script>
    </head>
    <body>
    <p>
    This example demonstrates the if..else if...else statement and object Date.
    </p>
    </body>
```

```
</html>
```

16_js_date.js

```
var d = new Date();
var hour = d.getHours();
document.write("<br>");
document.write("现在的时间是：");
document.write(d.getFullYear());
document.write("年");
document.write(d.getMonth()+1);
document.write("月");
document.write(d.getDate());
document.write("日");
document.write(" 星期");
document.write(d.getDay()+" ");
document.write(d.getHours());
document.write("点");
document.write(d.getMinutes());
document.write("分");
document.write(d.getSeconds());
document.write("秒");
document.write("<br>");
document.write("<br>");
if (hour < 12)
{
    document.write("Good morning");
}
else if (hour < 18)
{
    document.write("Good afternoon");
}
else if(hour < 24)
{
    document.write("Good night");
}
```

例: 17_js_setTimeout.html


```
<html>
<head>
    <title>_time_by_setTimeout</title>
</head>

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

<body onload="start_onload()">

<div align="center">

    <h1>Qfedu Timer</h1>
    <input type="text" id="time">
    <br>
    <input type="button" value="时间暂停" onclick="stop()">
    <input type="button" value="时间开始" onclick="start()">

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

time_by_setTimeout.js

```
var stop_flag = 0;

function timeout()
{
    var time = new Date();
    var h = time.getHours();
```

```
        var m = time.getMinutes();
        var s = time.getSeconds();

        document.getElementById("time").value = h+":"+m+":"+s;

        stop_flag = setTimeout("timeout()",1000); //指定的毫秒数后调用函数 timeout
    }

    function start_onload()
    {
        timeout();
    }

    function stop()
    {
        clearTimeout(stop_flag); //通过 setTimeout 返回值，停止定时
    }

    function start()
    {
        timeout();
    }
```

例子（17_js_setInterval）：


```
<html>
```

```
<head>
    <title>_time_by_setInterval</title>
</head>

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

<body onload="start_onload()">
<div align="center">

    <h1>Qfedu Timer</h1>
    <input type="text" id="time">
    <br>
    <input type="button" value="时间暂停" onclick="stop_1()">

</div>
</body>

</html>
```

time_by_setInterval.js

```
var stop_flag = 0;

function time()
{
    var time = new Date();
    var h = time.getHours();
    var m = time.getMinutes();
    var s = time.getSeconds();

    document.getElementById("time").value = h+":"+m+":"+s;
}


function start_onload()
{
    stop_flag = setInterval("time()",1000);//在指定的毫秒数后调用函数 time()函数
}
```

```
function stop_1()
{
    clearInterval(stop_flag); //通过 setInterval()返回值，停止调用
}
```

setTimeout 与 setInterval 函数区别：

setTimeout 函数设置超时调用函数，超时后自动调用所设置的函数。超时后自动调用了回调函数，如果还想调用回调函数，必须重新调用 setTimeout 进行超时设定。

而 setInterval 函数只需要设置一次，就可以多次调用回调函数，直到调用 clearInterval

3.3.8.4 内部对象（Math 对象）

执行常见的算数任务
使用 Math 的属性和方法的语法：
var pi_value=Math.PI; var sqrt_value=Math.sqrt(15);

Math 对象属性

属性	描述
E	返回算术常量 e，即自然对数的底数（约等于2.718）。
LN2	返回 2 的自然对数（约等于0.693）。
LN10	返回 10 的自然对数（约等于2.302）。
LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。
LOG10E	返回以 10 为底的 e 的对数（约等于0.434）。
PI	返回圆周率（约等于3.14159）。
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。
SQRT2	返回 2 的平方根（约等于 1.414）。

Math 对象方法

方法	描述
abs(x)	返回数的绝对值。
acos(x)	返回数的反余弦值。
asin(x)	返回数的反正弦值。
atan(x)	以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。
atan2(y,x)	返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。
ceil(x)	对数进行上舍入。
cos(x)	返回数的余弦。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。
log(x)	返回数的自然对数（底为e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0 ~ 1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。
toSource()	返回该对象的源代码。
valueOf()	返回 Math 对象的原始值。

例：js_math.html

```
<html>
    <head>
        <title>19_js_string</title>
    </head>
    <script type="text/javascript" src="js_math.js"></script>

    <body>
        数字 1：<input type="text" id="num1" size=50>
        <br>
        <br>
        数字 2：<input type="text" id="num2" size=50>
        <br>
        <br>
        结果：<input type="text" id="result" size=50>
        <br>
        <br>

        <input type="button" value="求两个数的较大值" onclick="fun(1)">
```

```
        <input type="button" value="求两个数的和" onclick="fun(2)">
        <br>

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

js_math.js

```
function fun(deal_num)
{
    var num1 = document.getElementById("num1").value;//
    var num2 = document.getElementById("num2").value;
    if(isNaN(num1) || isNaN(num2))
    {
        alert("请输入有效的数字！");
        document.getElementById("num1").value='';
        document.getElementById("num2").value='';
        document.getElementById("result").value='';
        return ;
    }

    var dest;//存储处理结果
    switch(deal_num)
    {

        case 1:
            alert("求较大值！！！");
            alert(Number(num1));
            alert(Number(num2));
            dest =Math.max(Number(num1),Number(num2));
            break;
        case 2:
            dest =Number(num1)+Number(num2);
            break;
    }

    document.getElementById("result").value = dest;//显示结果
}
```

3.3.8.5 内部对象（String 类 对象）

String 对象用于处理文本（字符串）

字符串是 JavaScript 的一种基本的数据类型。

String 对象定义了大量操作字符串的方法，例如从字符串中提取字符或子串，或者检索字符等

常用方法

charAt()返回在指定位置的字符

indexOf()检索字符串

substr()从起始索引号提取字符串中指定数目的字符串

substring()提取字符串中两个指定的索引号之间的字符串

例: 19_js_string.html

```
<html>
    <head>
        <title>19_js_string</title>
    </head>
    <script type="text/javascript" src="19_js_string.js"></script>

    <body>
        原字符串：<input type="text" id="src" size=50>
        <br>
        <br>
        处理方式：<input type="text" id="way" size=50>
        <br>
        <br>
        得到结果：<input type="text" id="dest" size=50>
        <br>
        <br>

        <input type="button" value="1.返回指定位置字符" onclick="deal_string(1)">
        <input type="button" value="2.检索字符串" onclick="deal_string(2)">
        <br>
        <input type="button" value="3.从起始索引号提取字符串中指定数目的字符" onclick="deal_string(3)">
        <br>
        <input type="button" value="4.提取字符串中介于两个指定下标之间的字符" onclick="deal_string(4)">
        <br>
        <br>
```

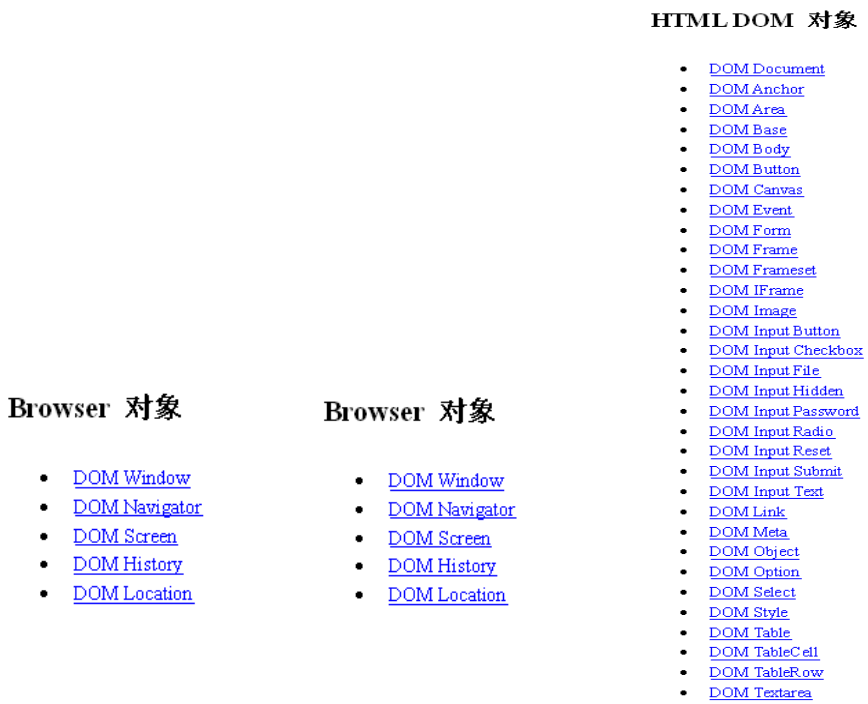
```
            <br>

            假设原字符串为：hello world
            <br>
            1.处理方式输入：0
            <br>
            按下《1.返回指定位置字符》则查找第二个字符，得到结果为：h
            <br>
            <br>
            2.处理方式输入：w
            <br>
            按下《2.检索字符串》则查找 w 在字符“hello world”中的位置，得到结果为：6
            <br>
            <br>
            3.处理方式输入：1,3
            <br>
            按下《3.从起始索引号提取字符串中指定数目的字符》获取从下标 1 开始后的 3 个字符，得到结果为：ell
            <br>
            <br>
            4.处理方式输入：2,4
            <br>
            按下《4.提取字符串中介于两个指定下标之间的字符》获取下标从 2 到 4 之间的（4-2=2）个字符，得到结果为：ll
        </body>
    </html>
```

19_js_string.js

```
    function deal_string(deal_num)
    {
        var src = document.getElementById("src").value;//
        var way = document.getElementById("way").value;
        var dest;//存储处理结果
        switch(deal_num)
        {
            case 1:
                dest = src.charAt(Number(way));//返回在指定位置的字符。
                break;
```

```
            case 2:
                dest = src.indexOf(way);//lastIndexOf();从后向前搜索字符串。
                break;

            case 3:
                /****src.substr(start,length)****/
                var start = way.substr(0,1);
                var length = way.substr(2,1);
                dest = src.substr(Number(start),Number(length));//字符串中抽取从第一个参数下
标开始的指定数目（第二个参数）的字符。
                break;

            case 4:
                /****src.substring(start,stop)***/
                var start = way.substring(0,1);
                var stop = way.substring(2);
                dest = src.substring(Number(start),Number(stop));//提取字符串中介于两个指定
下标之间的字符。
                //dest = src.substring(2);//提取字符串从 2 开始以后的字符
                break;

        }

        document.getElementById("dest").value = dest;//显示结果
    }
```

3.3.8.6 其他对象



例子（js_print）：

```
<html>
    <head>
        <title>JS 打印机
        </title>
        <script type="text/javascript" src="js_printer.js">
        </script>
    </head>
    <body onload="start_printer()">
        <div id="content" style="text-align:center;font-size:20px;">
        </div>
    </body>
</html>
```

3.3.9 全局函数

全局函数可用于整个 JavaScript 程序中

函数	描述
decodeURI()	解码某个编码的 URI。
decodeURIComponent()	解码一个编码的 URI 组件。
encodeURI()	把字符串编码为 URI。
encodeURIComponent()	把字符串编码为 URI 组件。
escape()	对字符串进行编码。
eval()	计算 JavaScript 字符串，并把它作为脚本代码来执行。
getClass()	返回一个 Java Object 的 Java Class。
isFinite()	检查某个值是否为有穷大的数。
isNaN()	检查某个值是否是数字。
Number()	把对象的值转换为数字。
parseFloat()	解析一个字符串并返回一个浮点数。
parseInt()	解析一个字符串并返回一个整数。
String()	把对象的值转换为字符串。
unescape()	对由 escape() 编码的字符串进行解码。

例子（18_js_function）：

```
<html>
    <head>
        <title>18_js_function</title>
    </head>
    <script type="text/javascript" src="18_js_function.js"></script>
    <body>
        data1:<input type="text" id="data1"/>
        <br>
        data2:<input type="text" id="data2"/>
        <br>
        <br>
        result:<label id="result"></label>
        <br>
        <br>
        <input type="button" id="add" value="相加" onclick="calc(0)"/>
        <input type="button" id="minus" value="相减" onclick="calc(1)"/>
    </body>
</html>
```

18_js_function.js

```
function calc(action)
{

    if(isNaN(document.getElementById("data1").value)                    ||
isNaN(document.getElementById("data2").value))
    {
```

```
            alert("请输入有效的数字！");
            document.getElementById("data1").value='';
            document.getElementById("data2").value='';
        }
        else
        {
            if(0 == action)

    document.getElementById("result").innerHTML=Number(document.getElementById("data1")
.value)+Number(document.getElementById("data2").value);
            else if(1 == action)

    document.getElementById("result").innerHTML=Number(document.getElementById("data1")
.value)-Number(document.getElementById("data2").value);
        }
    }
```

第三章：AJAX

3.1 AJAX 概述

概述：

1、AJAX 是 Asynchronous JavaScript And XML 的缩写
2、AJAX 是一种用于创建快速动态网页的技术
3、AJAX 不是新的编程语言，而是一种使用现有标准的新方法
4、AJAX 最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。
传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面
5、AJAX 不需要任何浏览器插件，但需要用户允许 JavaScript 在浏览器上执行。
6、有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。

AJAX 是基于现有的 Internet 标准

AJAX 是基于现有的 Internet 标准，并且联合使用它们：

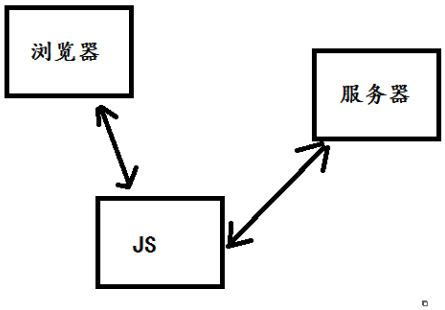
1、XMLHttpRequest 对象 (异步的与服务器交换数据)

2、JavaScript/DOM (信息显示/交互)

3、CSS (给数据定义样式)

4、XML (作为转换数据的格式)

浏览器是借助 JS 与服务器通信的，js 可以从浏览器获取数据，也可以更新浏览器的数据。



浏览器与服务器通信采用的就是 AJAX 技术，AJAX 核心是 XMLHttpRequest 对象;

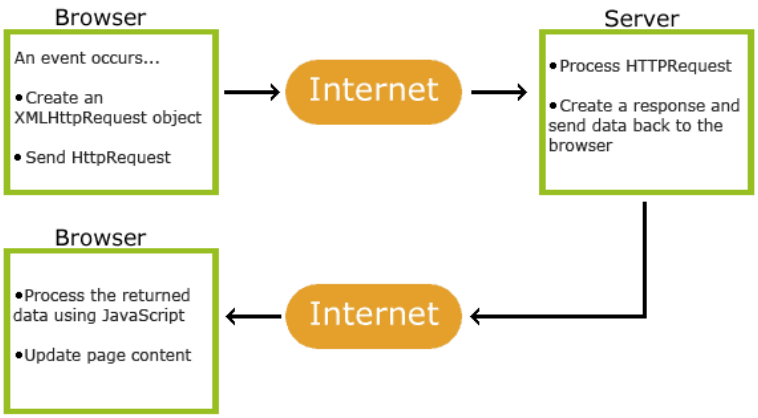
也就是说，咱们学习 AJAX 掌握 XMLHttpRequest 对象是非常重要的。这个对象有很多种方法可以实现浏览器与服务器之间的传递信息。

通过 JavaScript 的 XMLHttpRequest 对象完成发送请求到服务器并返回结果的任务，然后使用 JavaScript 更新局部的网页;

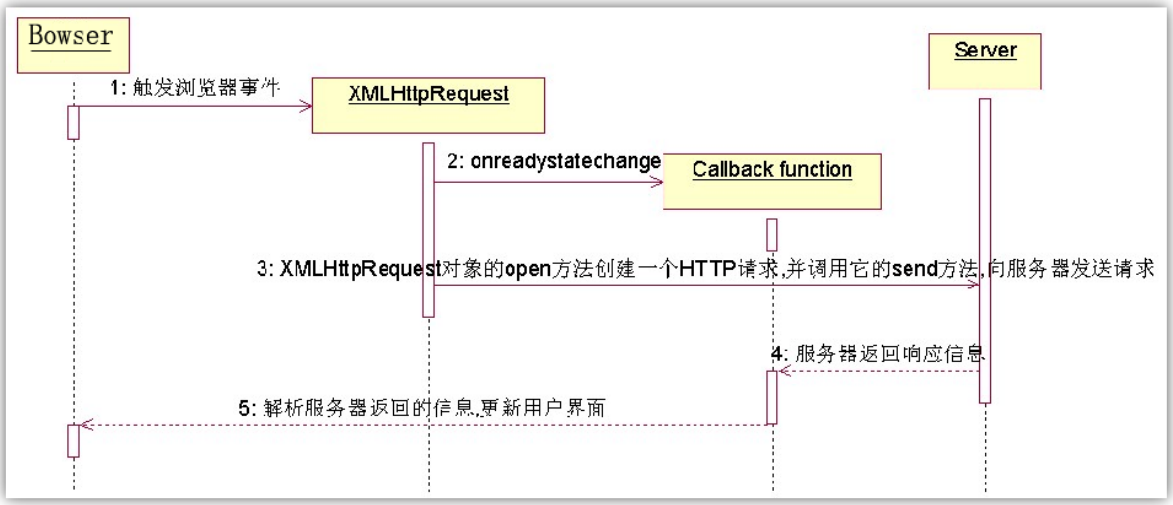
具有异步特性;

什么是异步特性呢，消息的出现是随机的，通信双方不需要建立同步时钟

3.2 AJAX 原理



AJAX 局部更新网页流程图：



异步流程：

1、创建对象
2、设置回调函数 ， fun 函数
3、open 创建服务器请求
4、send 向服务器发送请求，
5、服务器有结果会自动调用 fun 回调函数
在回调函数里面根据服务器返回的响应信息 更新用户界面

3.3 XMLHttpRequest

3.3.1 根据不同的浏览器创建异步请求对象

1、创建一个 xmlRequest

```
function getXMLHttpRequest()
{
    var xmlhttp = null;
    if (window.XMLHttpRequest)//自动检测当前浏览器的版本，如果是 IE5.0 以上的高版本的浏览器
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();//创建请求对象
    }
    else////如果浏览器是底版本的
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");//创建请求对象
    }
```

```
    return xmlhttp;//返回请求对象
}
```

在 js 文件中开始定义这个函数，其他 js 函数直接调用就能创建一个异步请求对象

3.3.2 标准的 XMLHttpRequest 属性

Onreadystatechange 每个状态改变时都会触发这个事件处理器，通常会调用一个 JavaScript 函数。

状态：

readyState：请求的状态。0 = 未初始化，1 = 正在加载，2 = 已加载，3 = 交互中，4 = 完成

status：服务器的 HTTP 状态码（200 对应 OK，404 对应 Not Found（未找到））

状态的改变会触发异步函数，调用回调函数。

不是每一种状态改变都要处理，一般在 readyState 状态值为 4，status 状态值为 200 的时候，处理服务器应答，所以在回调函数里写一个判断，判断 readyState 为 4，status 为 200，再做处理。

responseText 和 responseXML 就是服务器的反馈的结果。

属　　性	描　　述
onreadystatechange	每个状态改变时都会触发这个事件处理器 通常会调用一个 JavaScript 函数
readyState	请求的状态。0 = 未初始化，1 = 正在加载，2 = 已加载， 3 = 交互中，4 = 完成
responseText	服务器的响应，表示为一个字符串
responseXML	服务器的响应，表示为 XML。这个对象可以解析为一个 DOM 对

	象
status	服务器的 HTTP 状态码（200 对应 OK，404 对应 Not Found（未找到））
statusText	HTTP 状态码的相应文本（OK 或 Not Found（未找到）等等）

3.3.3 标准的 XMLHttpRequest 方法

方　　法	描　　述
abort()	停止当前请求
getAllResponseHeaders()	把 HTTP 请求的所有响应首部作为键/值对返回
getResponseHeader("header")	返回指定首部的串值
open(“method”, “url”, true)	建立对服务器的请求。method 参数可以是 GET、POST。url 参数可以是相对 URL 或绝对 URL。 true:异步；false:同步
send(content)	向服务器发送请求
setRequestHeader("header", "value")	把指定首部设置为所提供的值。在设置任何首部之前必须先调用 open()

异步流程：

1、创建对象

2、设置回调函数 ， fun 函数
3、open 创建服务器请求
4、Send 向服务器发送请求，
服务器有结果会自动调用 fun 回调函数

同步流程：

1、创建对象
2、open 建立对服务器的请求
3、send 向服务器发送请求
4、fun 函数处理，服务器反馈结果。

第四章：CGI 编程

4.1 什么是 CGI?

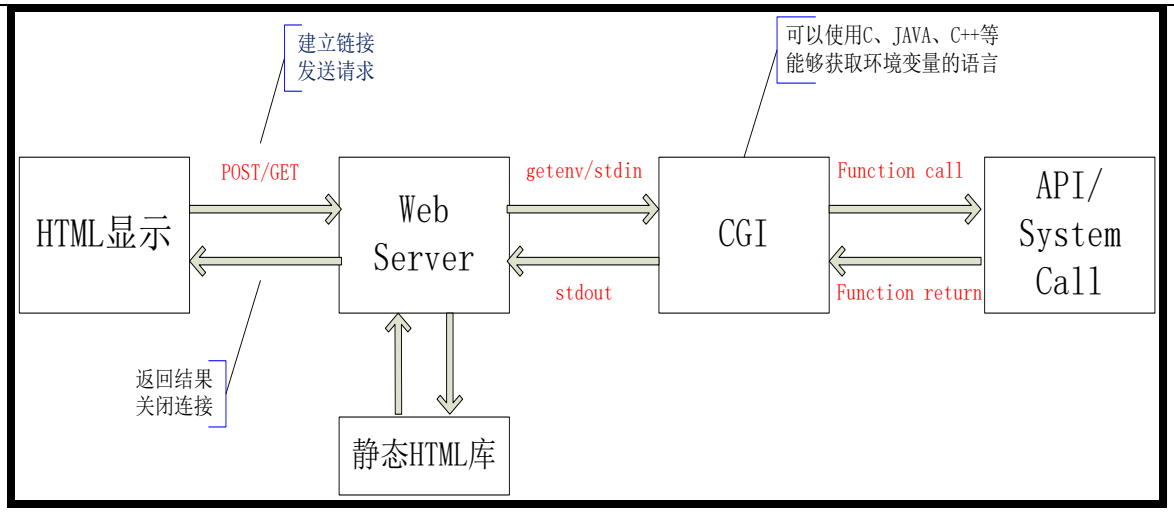
CGI 是通用网关接口(Common Gateway Interface);是 HTTP 服务器与其它程序进行“交谈”的工具
通过 CGI 接口就能在服务器端运行其他的程序

4.2 CGI 处理步骤

1、通过浏览器将用户请求送到服务器
2、服务器接收用户请求并交给 CGI 程序处理
3、CGI 程序把处理结果传送给服务器
4、服务器把结果送回到浏览器

4.3 CGI 编程

1、CGI 可以用任何一种语言编写，只要这种语言具有标准输入、标准输出、和获取环境变量
（1）CGI 程序通过标准输入(stdin)、标准输出(stdout)实现与 web 服务器间信息的传递
（2）环境变量为 Web 服务器和 CGI 接口之间约定的,用来向 CGI 程序传递一些重要的参数



2、CGI 传送给 Web 服务器的信息可以用各种格式,通常是以 HTML 文本或者 XML 文本的形式

（1）传输 HTML 文本第一行输出的内容必须是"Content-Type:text/html"

（2）传输 XML 文本第一行输出的内容必须是"Content-Type:text/xml"

（3）还有其他的一些格式：JIF(image/gif)、JPEG(image/jpeg)、AVI(video/avi)

例子：

```
#include <stdio.h>

int main(void)
{
    printf("content-type:text/html\n\n");
    printf("<html>\n<TITLE>CGI1:CGI hello!</TITLE>\n");
    printf("<center><H1>hello, this is frist CGI demo!</H1></center>\n</html>");
    return 0;
}
```

3、两个重要的 CGI 环境变量

（1）QUERY_STRING：在浏览器端以 GET 的方法输入的数据，数据的内容就是 url 问号后的内容

例子：

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    char *data = NULL;
    float a = 0.0, b = 0.0;
    char  c = 0;
    printf("content-type:text/html\n\n");
    data = getenv("QUERY_STRING");
    if(NULL == data)
    {
```

```
        printf("error");
        return 0;
    }
    //printf("data is:%s\n", data);
    sscanf(data, "%f%c%f",&a, &c, &b);
    if('+' == c)
    {
        printf("%f", a+b);
    }
    else if('-' == c)
    {
        printf("%f", a-b);
    }
    else
    {
        printf("error");
    }

    return 0;
}
```

（2）CONTENT_LENGTH：在浏览器端以 POST 方法输入的数据的字节数，数据的内容通过标准输入获取。

例子：

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    char *dataLen = NULL,buff[100] = {0};
    float a = 0.0, b = 0.0;
    char   c = 0;
    int len = 0;

    printf("content-type:text/html\n\n");
    dataLen = getenv("CONTENT_LENGTH");
    if(NULL == dataLen)
    {
        printf("error1");
        return 0;
```

```
    }
    else
    {
        len = atoi(dataLen);
        if(len > 0)
        {
            if(NULL == fgets(buff, len+1, stdin))
            {
                printf("error2");
                return 0;
            }
            else
            {
                sscanf(buff, "%f%c%f",&a, &c, &b);
                if('+' == c)
                {
                    printf("%f", a+b);
                }
                else if('-' == c)
                {
                    printf("%f", a-b);
                }
                else
                {
                    printf("error3");
                }
            }
        }
    }

    return 0;
}
```

（3）常用的 CGI 环境变量

CGI 环境变量名称	说明
REQUEST_METHOD	请求类型，如“GET ”或“POST ”
CONTENT_TYPE	被发送数据的类型
CONTENT_LENGTH	客户端向标准输入设备发送的数据长度，单位为字节
QUERY_STRING	查询参数，如“id=10010&name=kitty ”
SCRIPT_NAME	CGI 脚本程序名称
PATH_INFO	CGI 脚本程序附加路径
PATH_TRANSLATED	PATH_INFO 对应的绝对路径
REMOTE_ADDR	发送此次请求的主机 IP
REMOTE_HOST	发送此次请求的主机名
REMOTE_USER	已被验证合法的用户名
REMOTE_IDENT	WEB 服务器的登录用户名
AUTH_TYPE	验证类型
GATEWAY_INTERFACE	服务器遵守的 CGI 版本，如：CGI/1.1
SERVER_NAME	服务器主机名、域名或 IP
SERVER_PORT	服务器端口号
SERVER_PROTOCOL	服务器协议，如：HTTP/1.1