中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION
信息安全研究实验室

Web应用开发基础

授课教师：游伟 副教授

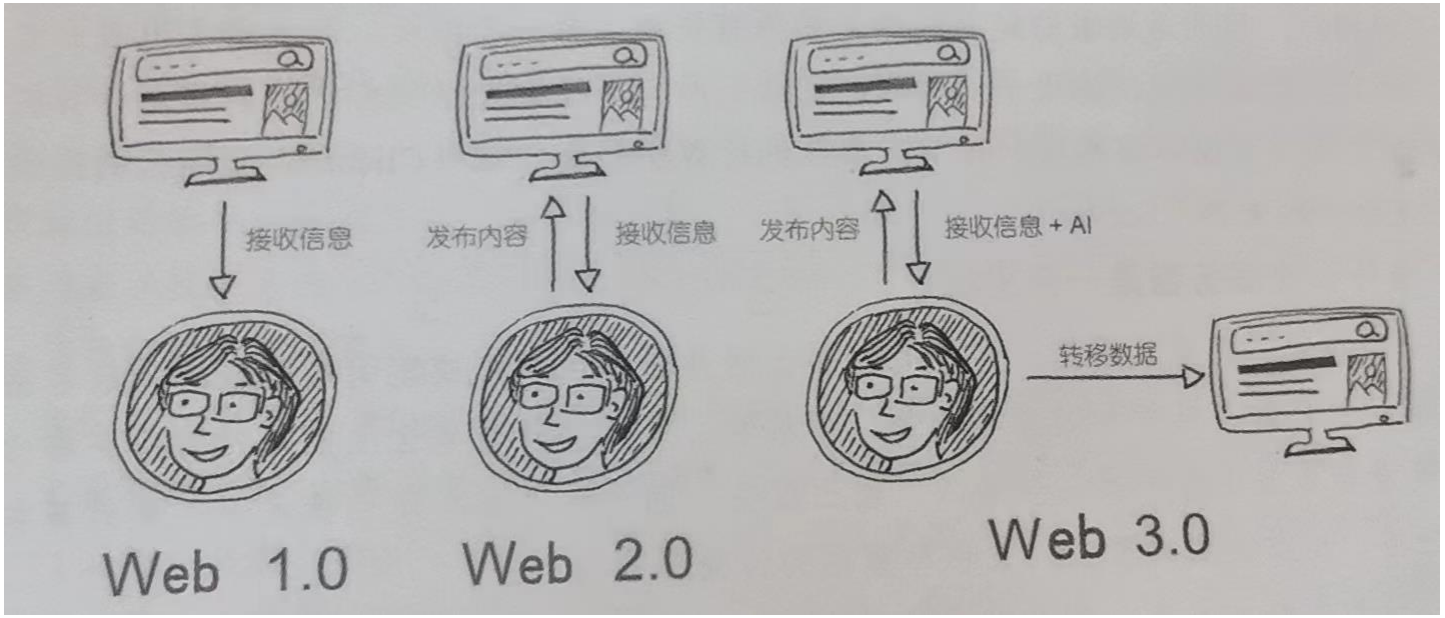
课程主页：https://www.youwei.site/training

本次讲座部分内容素材来源于曹巍老师《网页设计》课程资料

目录

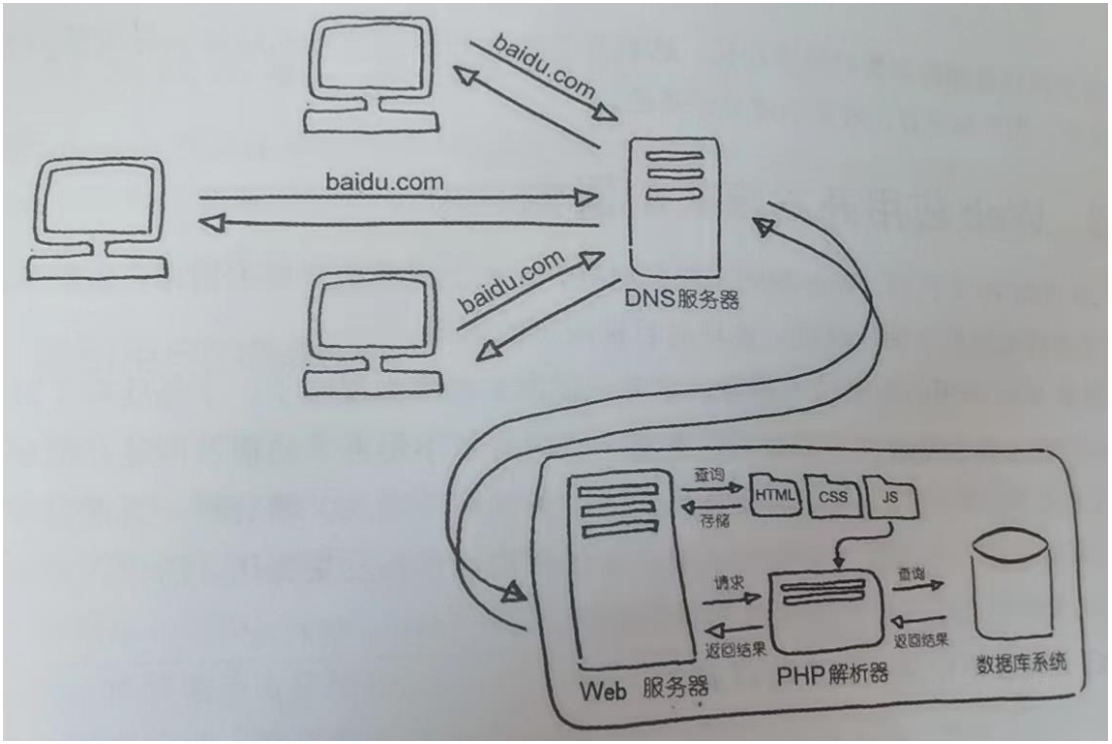
1.1 Web发展简史

- 被动的Web 1.0：打开一个电子图书应用，只能看书架上已有的书
- 交互的Web 2.0：不仅可以看已有的书，还可以进行评价
- 智能的Web 3.0：除了浏览和评价，还会根据浏览历史自动推荐



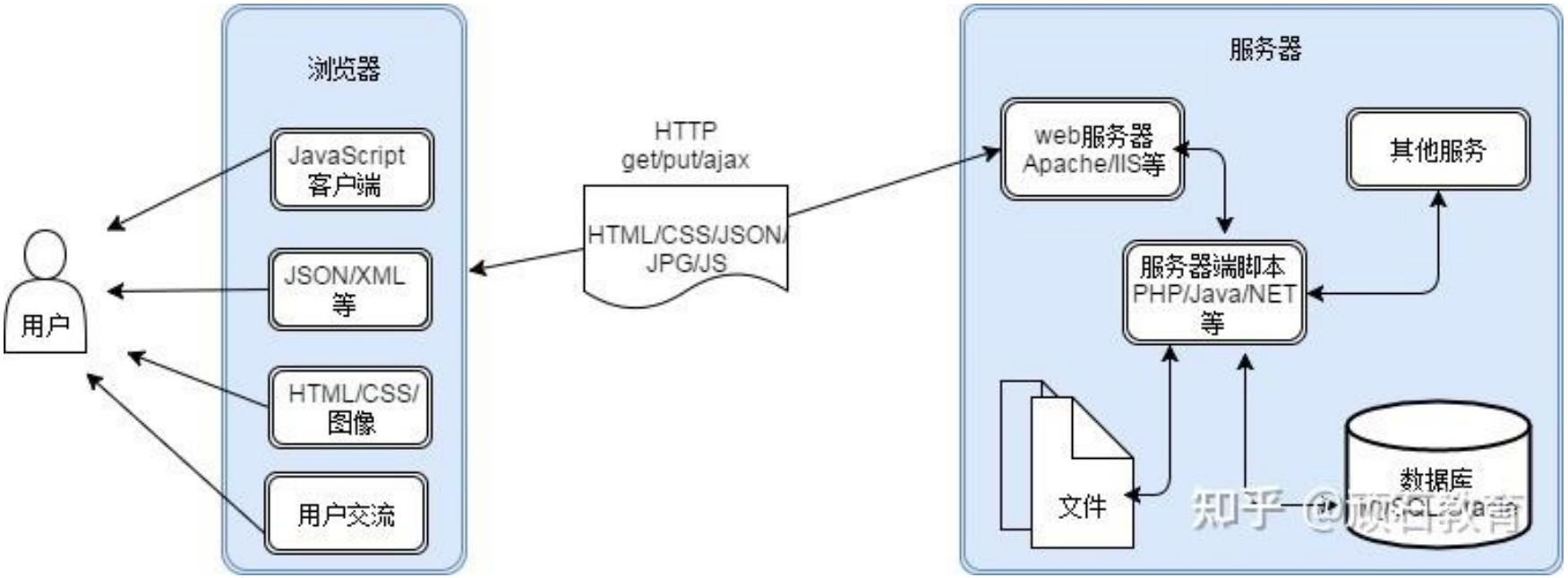
1.2 每一次浏览网页都发生了什么

- 客户端发送请求
- 服务端返回响应
- 浏览器解析代码并渲染网页内容



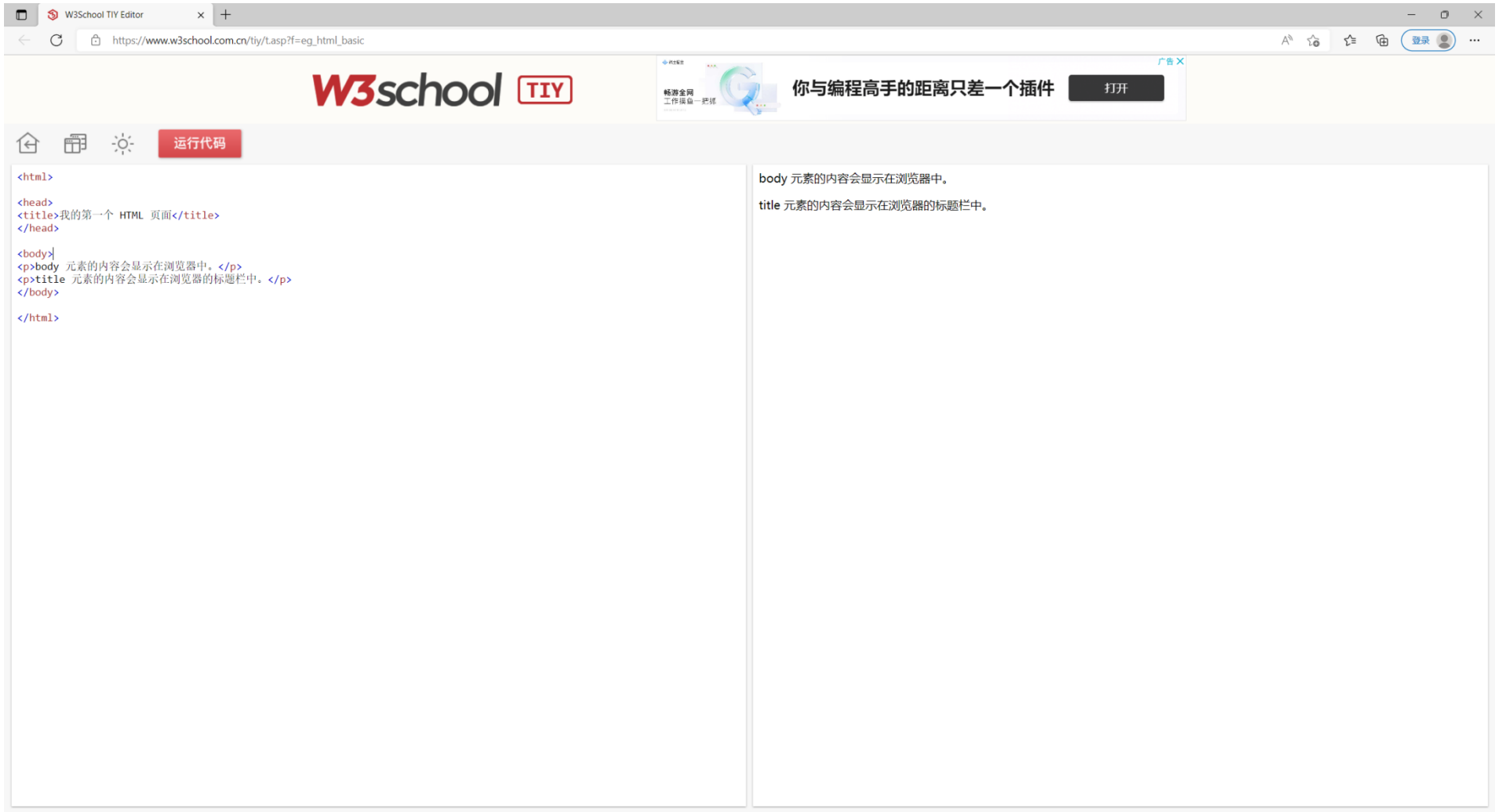
1.3 Web应用开发模式

- 前端技术：HTML + CSS + JavaScript
- 后端技术：PHP/ASP/JSP + 数据库 + 数据处理
- 前后端通信：HTTP + AJAX



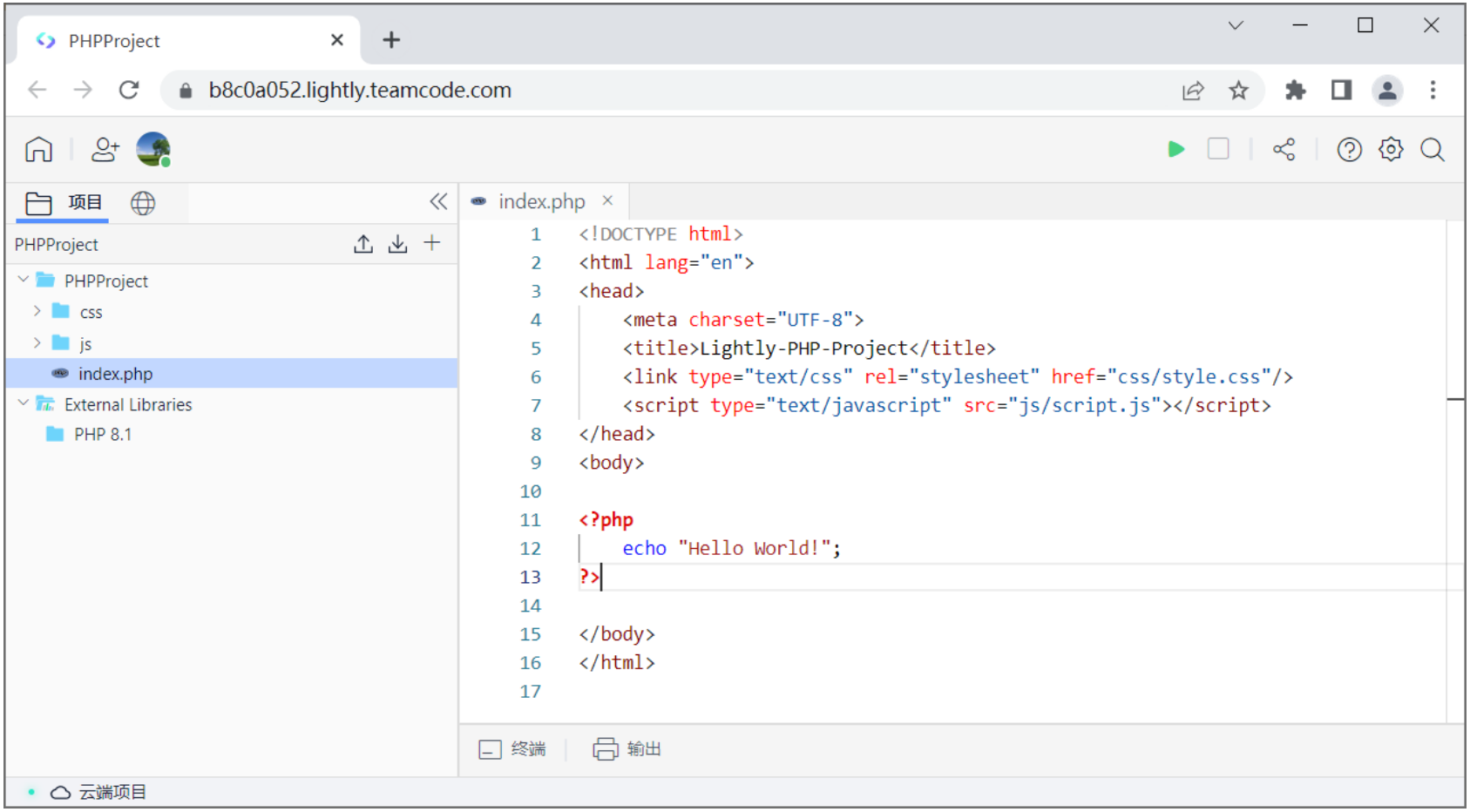
前端在线编辑平台

- https://www.w3school.com.cn/tiy/t.asp?f=eg_html_basic



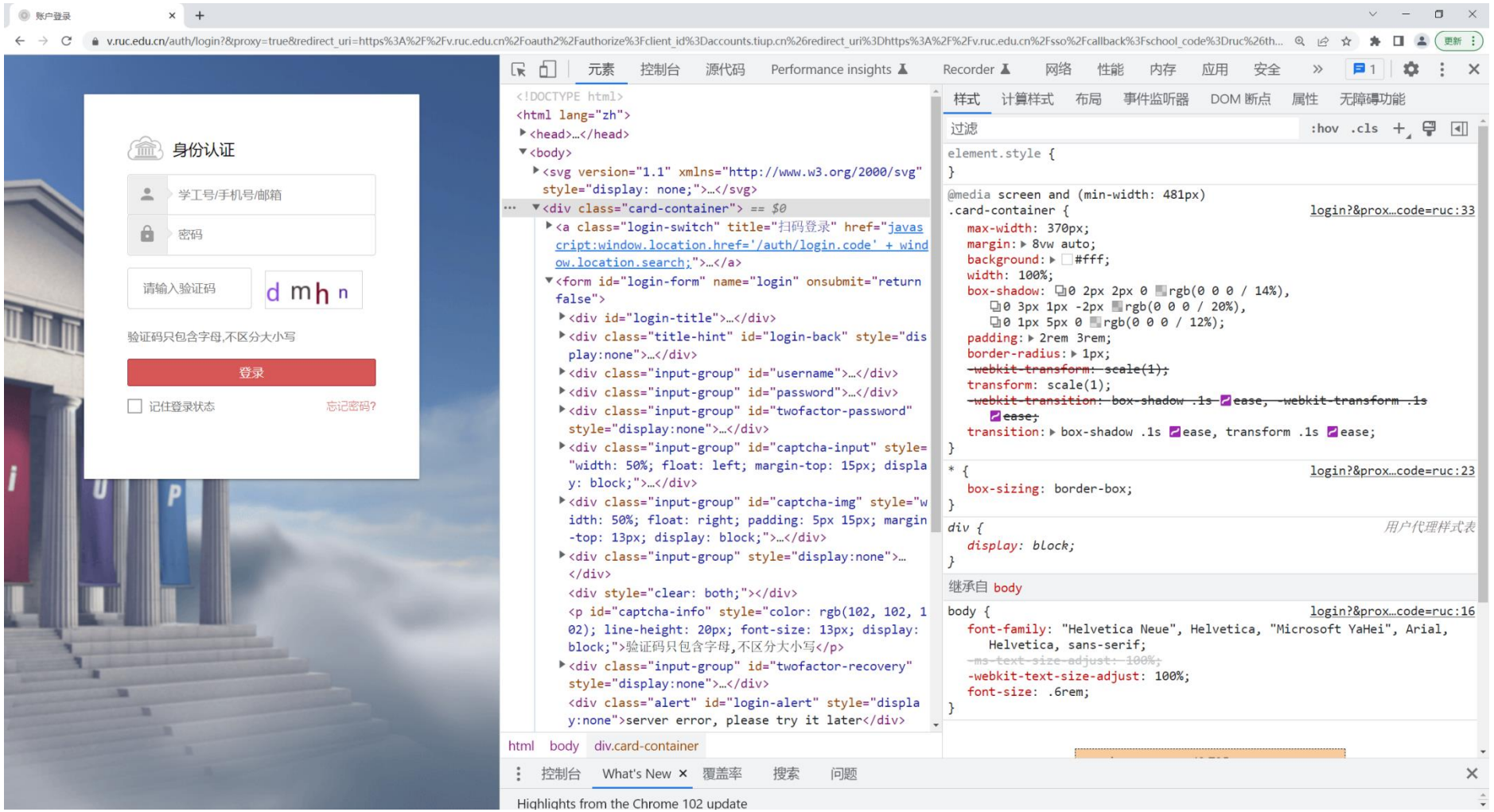
后端PHP在线编辑平台

- https://lightly.teamcode.com/php



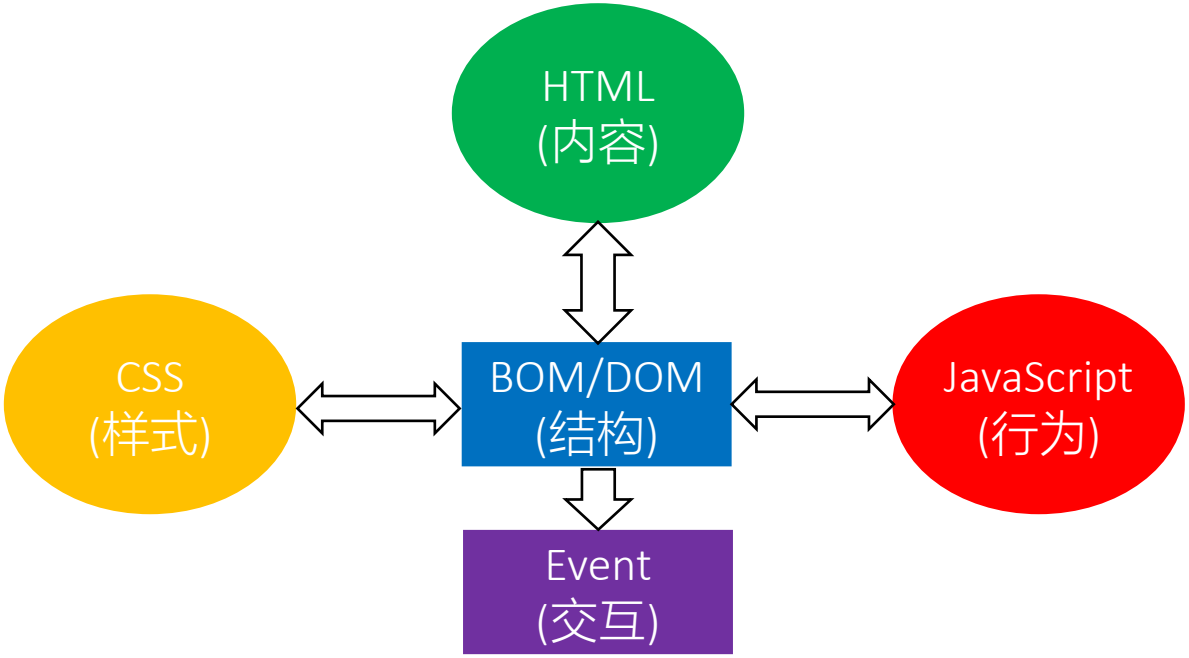
目录

引子


账户登录
v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%3A%2F%2Fv.ruc.edu.cn%2Fsso%2Fcallback%3Fschool_code%3Druc%26th...
身份认证
学工号/手机号/邮箱
密码
请输入验证码
d mh n
验证码只包含字母,不区分大小写
登录
记住登录状态
忘记密码?
元素 控制台 源代码 Performance insights Recorder 网络 性能 内存 应用 安全
样式 计算样式 布局 事件监听器 DOM 断点 属性 无障碍功能
过滤 :hov .cls
element.style {
}
@media screen and (min-width: 481px)
.card-container { login?&prox…code=ruc:33
max-width: 370px;
margin: 8vw auto;
background: #fff;
width: 100%;
box-shadow: 0 2px 2px 0 rgb(0 0 0 / 14%), 0 3px 1px -2px rgb(0 0 0 / 20%), 0 1px 5px 0 rgb(0 0 0 / 12%);
padding: 2rem 3rem;
border-radius: 1px;
-webkit-transform: scale(1);
transform: scale(1);
-webkit-transition: box-shadow .1s ease, -webkit-transform .1s ease;
transition: box-shadow .1s ease, transform .1s ease;
}
* { login?&prox…code=ruc:23
box-sizing: border-box;
}
div { 用户代理样式表
display: block;
}
继承自 body
body { login?&prox…code=ruc:16
font-family: "Helvetica Neue", Helvetica, "Microsoft YaHei", Arial, Helvetica, sans-serif;
-ms-text-size-adjust: 100%;
-webkit-text-size-adjust: 100%;
font-size: .6rem;
}
html body div.card-container
控制台 What's New 覆盖率 搜索 问题
Highlights from the Chrome 102 update

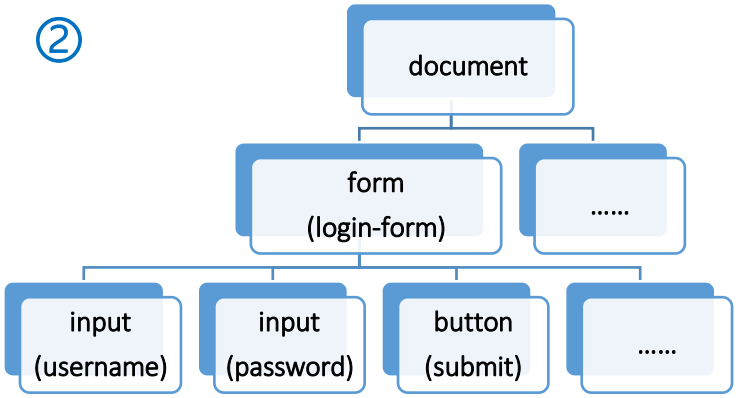
Web前端架构

- 三种语言：HTML + CSS + JavaScript
- 两个机制：BOM/DOM (浏览器文档对象模型) + Event (事件)



Web前端示例

①

② 

③

对象	事件	响应
button (submit)	onclick	login()
.....	.....	.....

④
```
<form id="login-form">
  <input id="username" ... />
  <input id="password" ... />
  <button id="submit" ... />
  .....
</form>
```

⑤
```
<style>
  .input { background-color: white;
           height: 45px; .....
         }
  .....
</style>
```

⑥
```
<script>
 function login() {
   result = check_verification();
   .....
 }
 button=getElementById("submit");
 button.onclick = login;
</script>
```

① 页面显示
② BOM/DOM
③ Event
④ HTML
⑤ CSS
⑥ JavaScript

2.1 HTML

- HTML是HyperText Markup Language（超文本标记语言）的缩写，是一种为普通文件中某些字句加上标签的语言，其目的是运用标签（tag）对文件达到预期的效果

HTML 代码

```
<html>
  <head>
    <title>学习 HTML</title>
  </head>
  <body>
    <h1>欢迎来到HTML的世界</h1>
  </body>
</html>
```

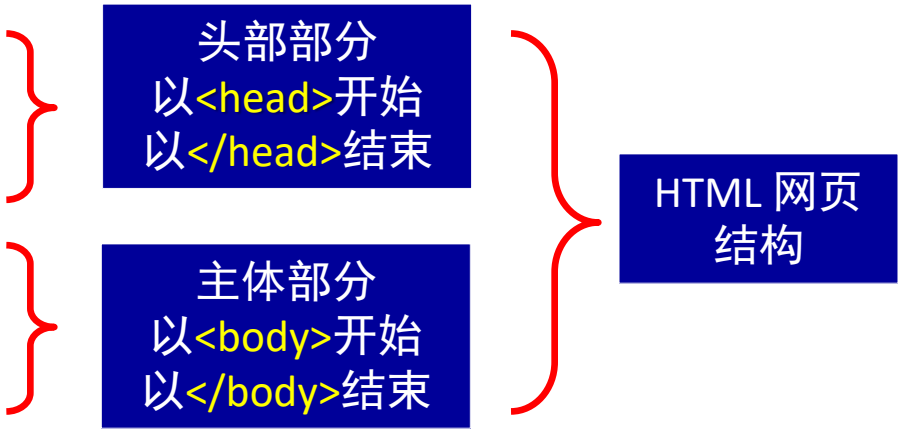
浏览器处理代码并显示

学习更多：https://www.w3school.com.cn/html/index.asp

2.1.1 HTML文档结构

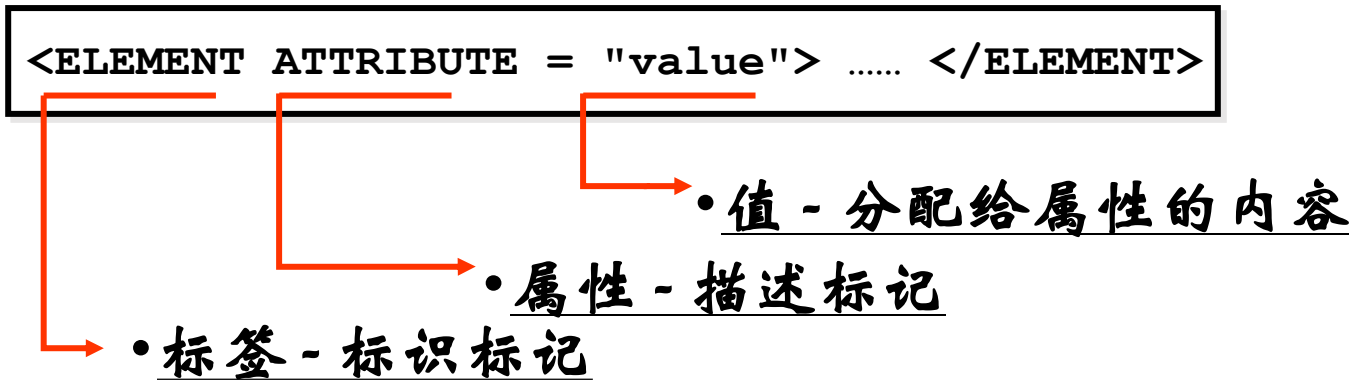
- HTML 文档以<html>...</html>标签标记开始和结束
  - 头部部分：<head>...</head>，包括标题和其他说明信息
  - 主体部分：<body>...</body>，包括文本、图像、链接、表单等

```
<html>
  <head>
    <title>学习 HTML</title>
  </head>
  <body>
    <h1>欢迎来到HTML的世界</h1>
  </body>
</html>
```



2.1.2 标签、属性、元素

- 标签是由尖括号包围的关键词，是HTML的基本组成单位
  - 大部分标签是双标记标签，由起始标记和闭合标记组成，如<h1></h1>
  - 一些标签是单标记标签，在起始标记中进行闭合，如
- 标签内部可以添加属性，属性的定义方式为一个键值对
  - 一个标签可以有多个属性
  - 属性值应该被包含在引号中
- 元素指的是从开始标签到结束标签所有的符号

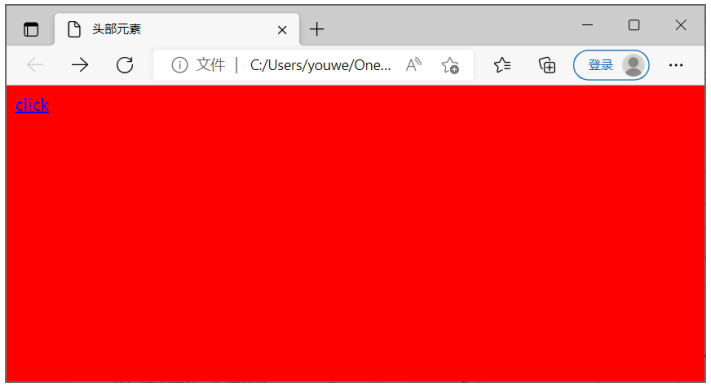


2.1.3 常见的头部元素

```
<html>
  <head>
    <title>头部元素</title>
    <base href="http://www.youwei.site/base/"
          target="_blank"/>
    <link rel="stylesheet" href="mystyle.css"/>
    <meta http-equiv="Content-Type"
          content="text/html; charset=UTF-8"/>
    <script type="text/javascript">
      alert('hi');
    </script>
    <style type="text/css">
      body { background-color: red; }
    </style>
  </head>
  <body>
    <a href="test.html">click</a>
  </body>
</html>
```

由于头部base元素的设置，点击超链接将在空白页面（_blank），打开http://www.youwei.site/base/test.html。

标签	描述
<head>	定义关于文档的信息。
<title>	定义文档标题。
<base>	定义页面上所有链接的默认地址或默认目标。
<link>	定义文档与外部资源之间的关系。
<meta>	定义关于 HTML 文档的元数据。
<script>	定义客户端脚本。
<style>	定义文档的样式信息。



2.1.4 常见的主体元素

- 标头
- 段落
- 换行
- 字体
- 文本格式化
- 预格式化文本
- 水平分割线
- 图片
- 超链接
- 列表
- 表格
- 表单
- 区块
- 框架

标头

- 使用<h1>到<h6>标签，指定不同级别的标头（heading）

```
<html>
  <head>
    <title>动物世界</title>
  </head>
  <body>
    <h1>从大自然获得灵感</h1>
    <h2>从大自然获得灵感</h2>
    <h3>从大自然获得灵感</h3>
    <h4>从大自然获得灵感</h4>
    <h5>从大自然获得灵感</h5>
    <h6>从大自然获得灵感</h6>
  </body>
</html>
```

段落

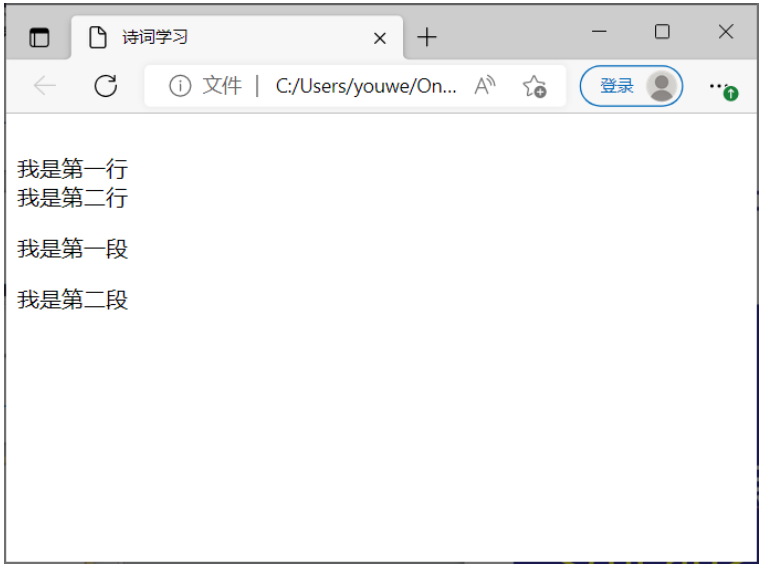
- 使用 ... 标签用于标记段落（paragraph）
- 属性align用于控制段落对齐方式

```
<html>
  <head>
    <title>诗词学习</title>
  </head>
  <body>
    <p>我是第一段</p>
    <p>我是第二段</p>
    <p align="left">左对齐显示</p>
    <p align="center">居中显示</p>
    <p align="right">右对齐显示</p>
  </body>
</html>
```

换行

- 文本中用
标签，会强制换行（break）

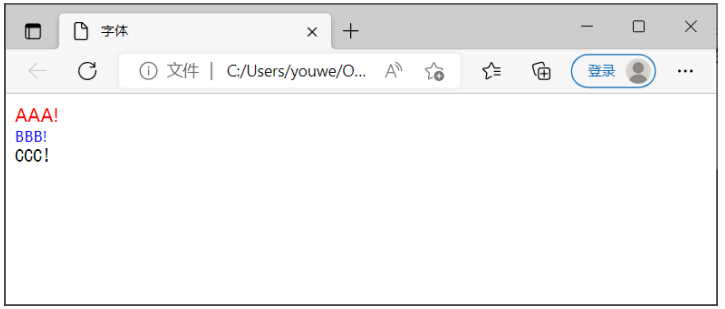
```
<html>
  <head>
    <title>诗词学习</title>
  </head>
  <body>
    <br/>我是第一行<br/>我是第二行
    <p>我是第一段</p>
    <p>我是第二段</p>
  </body>
</html>
```



字体

- 使用<font>标签，规定文本的字体、字体尺寸、字体颜色

```
<html>
  <head>
    <title>字体</title>
  </head>

  <body>
    <font size="3" color="red">AAA!</font>
    <br/>

    <font size="2" color="#0000FF">BBB!</font>
    <br/>

    <font face="SimHei">CCC!</font>
    <br/>
  </body>
</html>
```



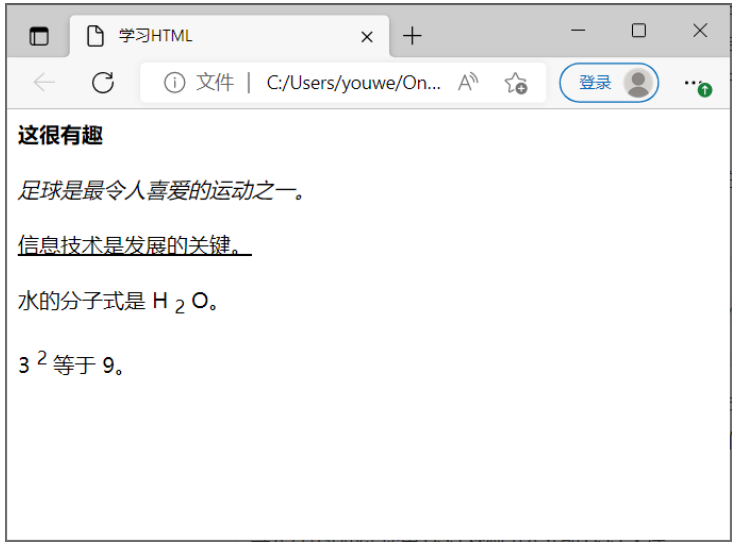
属性	值	描述
color	*rgb(x,x,x)* *#xxxxxx* *colorname*	不赞成使用。请使用样式取代它。 规定文本的颜色。
face	*font_family*	不赞成使用。请使用样式取代它。 规定文本的字体。
size	*number*	不赞成使用。请使用样式取代它。 规定文本的大小。

颜色	3位十六进制颜色值	6位十六进制颜色值	RGB
	#000	#000000	rgb(0,0,0)
	#F00	#FF0000	rgb(255,0,0)
	#0F0	#00FF00	rgb(0,255,0)
	#00F	#0000FF	rgb(0,0,255)
	#FF0	#FFFF00	rgb(255,255,0)
	#0FF	#00FFFF	rgb(0,255,255)
	#F0F	#FF00FF	rgb(255,0,255)
	#888	#888888	rgb(136,136,136)
	#FFF	#FFFFFF	rgb(255,255,255)

文本格式化

- HTML使用标签对输出的文本进行格式

```
<html>
  <head>
    <title>学习HTML</title>
  </head>

  <body>
    <p>
      <b>这很有趣</b>
      <br/><br/>

      <i>足球是最令人喜爱的运动之一。</i>
      <br/><br/>

      <u>信息技术是发展的关键。</u>
      <br/><br/>

      水的分子式是 H <sub>2</sub> O。
      <br/><br/>

      3 <sup>2</sup> 等于 9。
      <br/><br/>
    </p>
  </body>
</html>
```

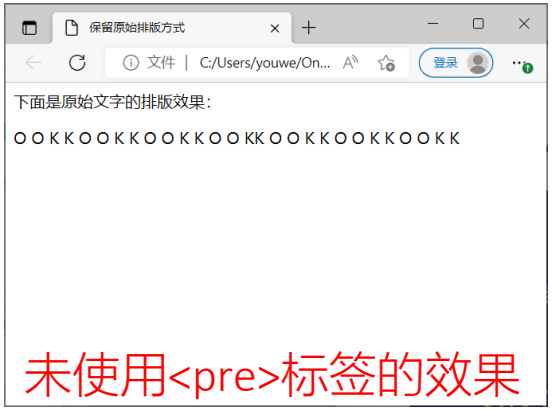


名称	HTML代码
黑体	
斜体	
下划线	<u></u>
中划线	<s></s>
闪烁	<blink></blink>
上标	
下标	

预格式化文本

- <pre> 标签用于显示具有预先定义格式的文本（Preformatted）

```
<html>
  <head>
    <title>保留原始排版方式</title>
  </head>

  <body>
    <p>下面是原始文字的排版效果：</p>

    <pre>
            O  O                  K      K
         O        O               K    K
       O            O             K  K
      O               O            KK
       O            O             K  K
         O        O               K    K
            O  O                  K      K
    </pre>
  </body>
</html>
```

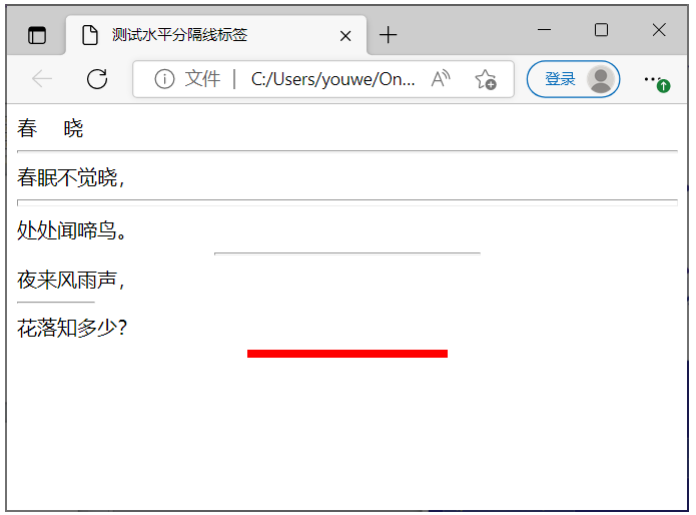


水平分割线

属性	功能	单位	默认值
size	设置水平分隔线的粗细	pixel	2
width	设置水平分隔线的宽度	pixel，%	100%
align	设置水平分隔线的对齐方式		center
color	设置水平分隔线的颜色		black

- <hr/>用于段落与段落之间的分隔（Horizontal Rule）
- 通过设置<hr/>标签的属性值，可以控制水平分隔线的样式

```
<html>
  <head>
    <title>测试水平分隔线标签</title>
  </head>
  <body>
    春　晓
    <hr/>
    春眠不觉晓，
    <hr size="6"/>
    处处闻啼鸟。
    <hr width="40%"/>
    夜来风雨声，
    <hr width="60" align="left"/>
    花落知多少？
    <hr size="6" width="30%" align="center" color="red"/>
  </body>
</html>
```



图片

属性	功能	单位	默认值
src	指定图像的URL地址		
width	设置图像宽度	pixel, %	
height	设置图像高度	pixel, %	
align	指定图像与文字的对齐关系		bottom

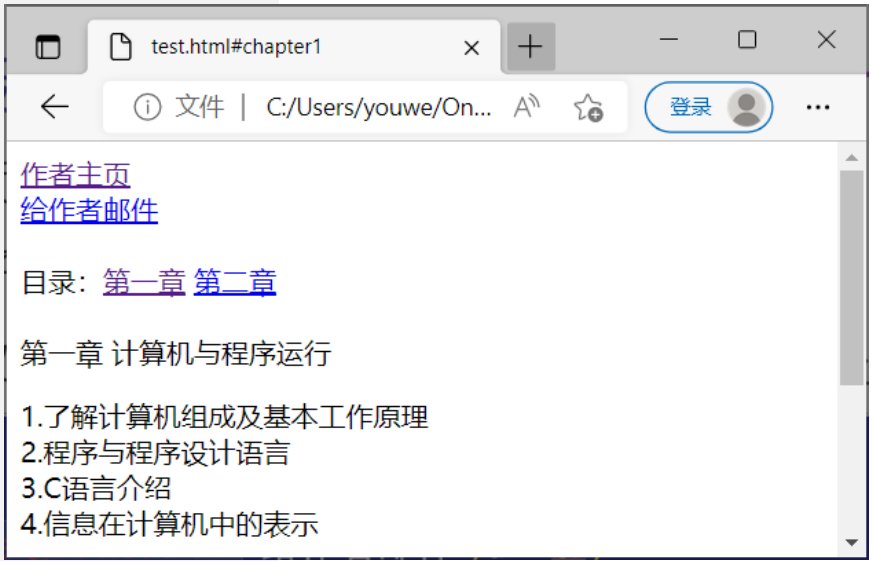
- 图像由<img>标签定义（Image）

```
<html>
  <head>
    <title>仙剑世界</title>
  </head>

  <body>
    <h1>
      <font size="3" color="green">
        <b>李逍遥和赵灵儿</b>
      </font>
    </h1>

    <p><img align="bottom" src="pic.jpg" height=100/>底部对齐</p>
    <p><img align="top" src="pic.jpg" height=100/>顶部对齐</p>
    <p><img align="middle" src="pic.jpg" height=100/>居中对齐</p>
    <p><img src="pic.jpg" height=100>默认对齐</p>
  </body>
</html>
```

超级链接

- <a>标签用于创建超级链接（Anchor）
  - 网页链接：跳转到另一个页面
  - 电子邮件链接：发送电子邮件
  - 长文档内部链接：跳转到文档特定书签处

```
<body>
  <a href="https://www.youwei.site">作者主页</a> <br/>
  <a href="mailto:youwei@ruc.edu.cn">给作者邮件</a> <br/>
  目录：
  <a href="#chapter1">第一章</a>
  <a href="#chapter2">第二章</a>
  <br/><br/>
  <a name="chapter1">第一章 计算机与程序运行</a>
  <p>
    1.了解计算机组成及基本工作原理<br/>
    2.程序与程序设计语言<br/>
    ……
  </p>
  <a name="chapter2">第二章 基本概念</a>
  <p>
    1.数据存储、数据类型、变量、常量<br/>
    2.运算符和表达式<br/>
    ……
  </p>
</body>
```



列表

- 列表用于按逻辑方式对数据分组
  - 标签用于创建无序列表（Unordered List）
  - 标签用于创建有序列表（Ordered List）

```
…
<body>
  <h1>星期中的每一天</h1>
  <ul>
    <li>星期日</li>
    <li>星期一</li>
    <li>星期二</li>
    <li>星期三</li>
    <li>星期四</li>
    <li>星期五</li>
    <li>星期六</li>
  </ul>
</body>
…
```

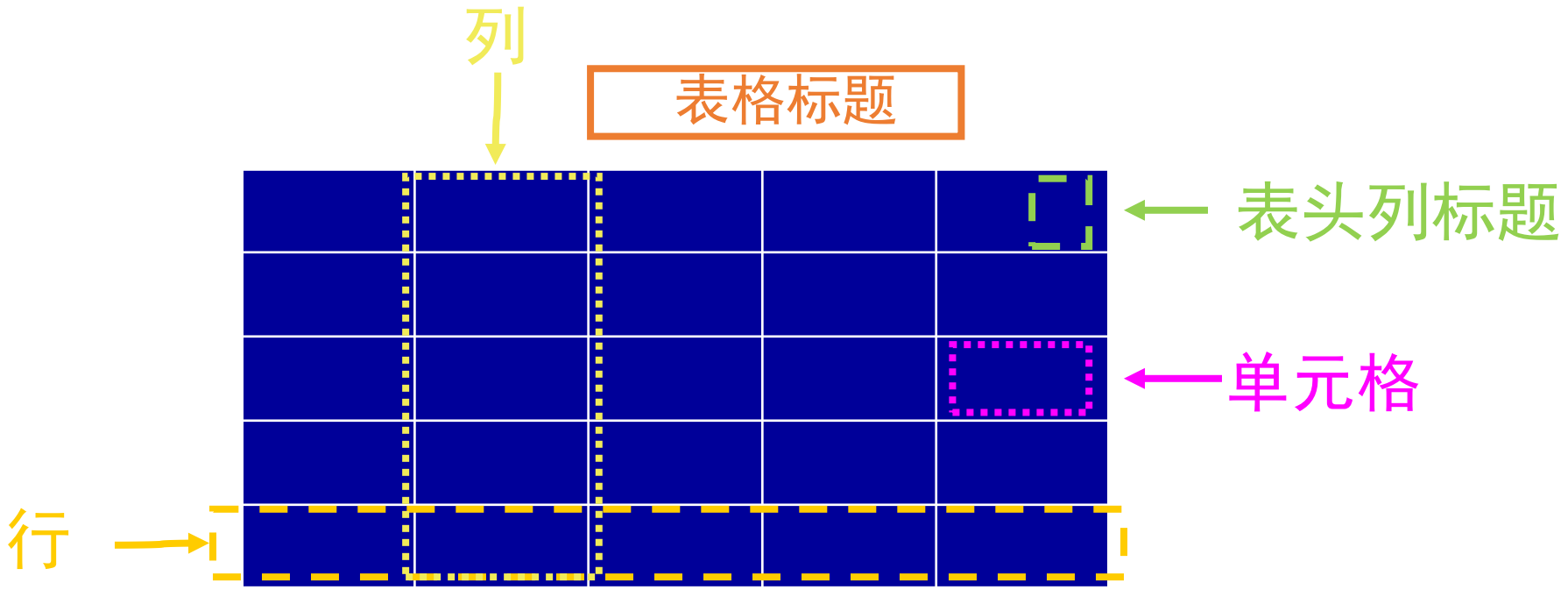
无序列表

```
…
<body>
  <h1>星期中的每一天</h1>
  <ol>
    <li>星期日</li>
    <li>星期一</li>
    <li>星期二</li>
    <li>星期三</li>
    <li>星期四</li>
    <li>星期五</li>
    <li>星期六</li>
  </ol>
</body>
…
```

有序列表

表格

- 表格由 <table> 标签来定义
  - 每个表格均有若干行，由 <tr> 标签定义（Table Row）
  - 每行分割为若干单元格，由 <td> 标签定义（Table Data Cell）
  - 表头可以设置列标题，由<th>标签定义（Table Header Cell）
  - 整个表格的标题，由<caption>标签定义



表格

学员档案信息

姓名	性别	分数
Robert	M	80
Mary	F	18

```
<html>
  <head>
    <title>使用表格</title>
  </head>
  <body>
    <table border="2">
      <caption align="top">学员档案信息</caption>
      <tr>
        <th>姓名</th> <th>性别</th> <th>分数</th>
      </tr>
      <tr>
        <td>Robert</td> <td>M</td> <td>80</td>
      </tr>
      <tr>
        <td align="left">Mary</td>
        <td align="center">F</td>
        <td align="right">18</td>
      </tr>
    </table>
  </body>
</html>
```

<table>标签代表表格的开始，border=2表示边框尺寸为2

<caption>表示表格标题

<tr>表示行，这是表格的第一行，有三列数据，<th>代表标题单元格

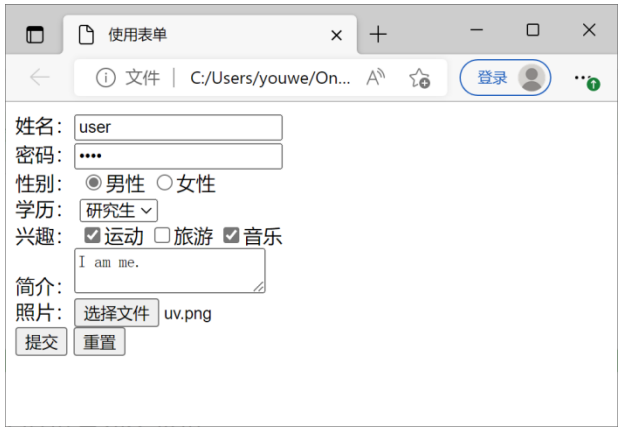
<td>代表数据单元格

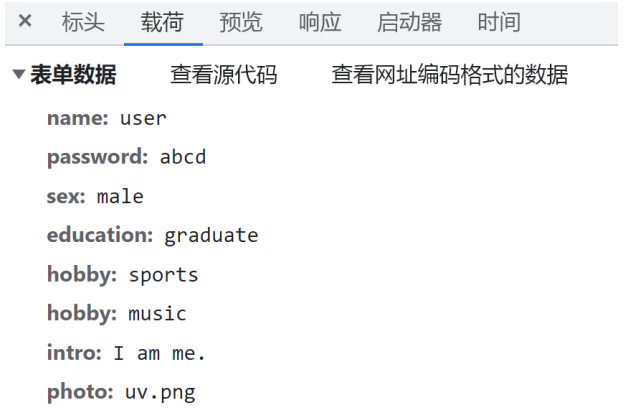
不同的对齐方式

表单

- 表单由<form>标签定义，用于收集用户的输入信息

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  姓名：<input type="text" name="name"/> <br/>
  密码：<input type="password" name="password"/> <br/>
  性别：
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历：
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣：
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介：<textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片：<input type="file" name="photo" id="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```



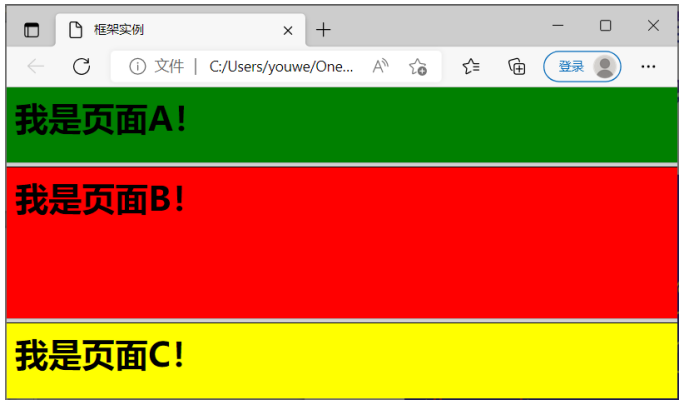


框架

- 将浏览器的页面显示区域进行划分，分成多个独立可控制区域，这些区域称为框架，每个框架中都可以打开网页

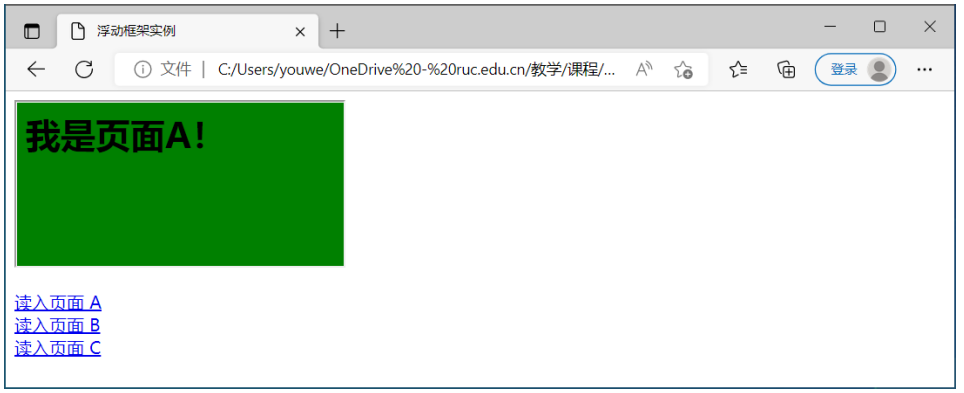
普通框架（框架集）

```
<html>
  <head><title>框架实例</title></head>
  <frameset rows="25%,50%,25%">
    <frame src="frame_a.htm"/>
    <frame src="frame_b.htm"/>
    <frame src="frame_c.htm"/>
  </frameset>
</html>
```



浮动框架（窗口嵌套）

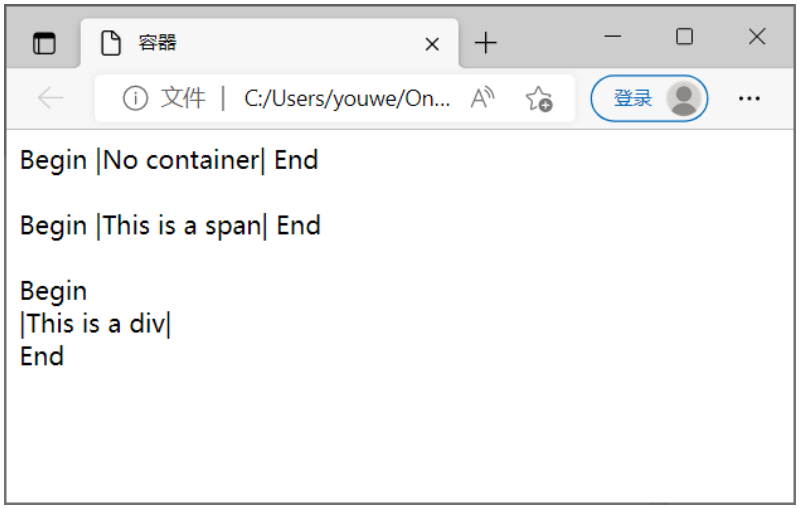
```
<html>
  <head><title>浮动框架实例</title></head>
  <body>
    <iframe src="A.html" name="window"></iframe><br/>
    <a href="A.html" target="window">读入页面 A</a><br/>
    <a href="B.html" target="window">读入页面 B</a><br/>
    <a href="C.html" target="window">读入页面 C</a><br/>
  </body>
</html>
```



容器

- 容器本身无意义，其存在是配合CSS为其内部内容定义显示规则
  - 是行内元素，在它的前后不会换行
  - 是块级元素，它包含的元素会自动换行

```
<html>
<head>
  <title>容器</title>
</head>
<body>
  Begin |No container| End
  <br/><br/>


  Begin <span class="myclass">|This is a span|</span>
  End
  <br/><br/>


  Begin <div id="myid"> |This is a div| </div> End
  <br/><br/>
</body>
</html>
```



实例：微人大认证页面HTML代码

```
<html>
  <head>
    <meta charset="utf-8"/>
    <title>账户登录</title>
    <link rel="stylesheet" href="style.css"/>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body onload="body_onload_handler()">
    <div class="card-container">
      <form id="login-form" onsubmit="return false">
        <div id="login-title"><h3>身份认证</h3></div>
        <div class="input-group" id="username">
          <input type="text" name="username" placeholder="学工号/手机号/邮箱" required>
        </div>
        <div class="input-group" id="password">
          <input type="password" name="password" placeholder="密码" required>
        </div>
        <div class="input-group" id="captcha-input">
          <input type="text" name="code" placeholder="请输入验证码">
        </div>
        <div class="input-group" id="captcha-img"><img src="img/code_0.png"/></div>
        <p id="captcha-info">验证码只包含字母,不区分大小写</p>
        <div class="alert" id="login-alert"></div>
        <div class="form-group">
          <button id="login-submit" type="submit"><span>登录</span></button>
        </div>
      </form>
    </div>
  </body>
</html>
```

2.2 CSS

- CSS是Cascading Style Sheets（层叠样式表）的缩写，是一种样式语言，用于修饰HTML元素，从而告诉浏览器如何将这些元素精美地显示在网页中

```
<html>
  <head>
    <style>
    body { background-color: lightblue; }
    #myid { color: white; text-align: center; }
    .myclass { font-family: verdana;
               font-size: 20px; }
    </style>
  </head>
  <body>
    <h1 id="myid">我的第一个 CSS 实例</h1>
    <p class="myclass">这是一个段落。</p>
  </body>
</html>
```

学习更多：https://www.w3school.com.cn/css/index.asp

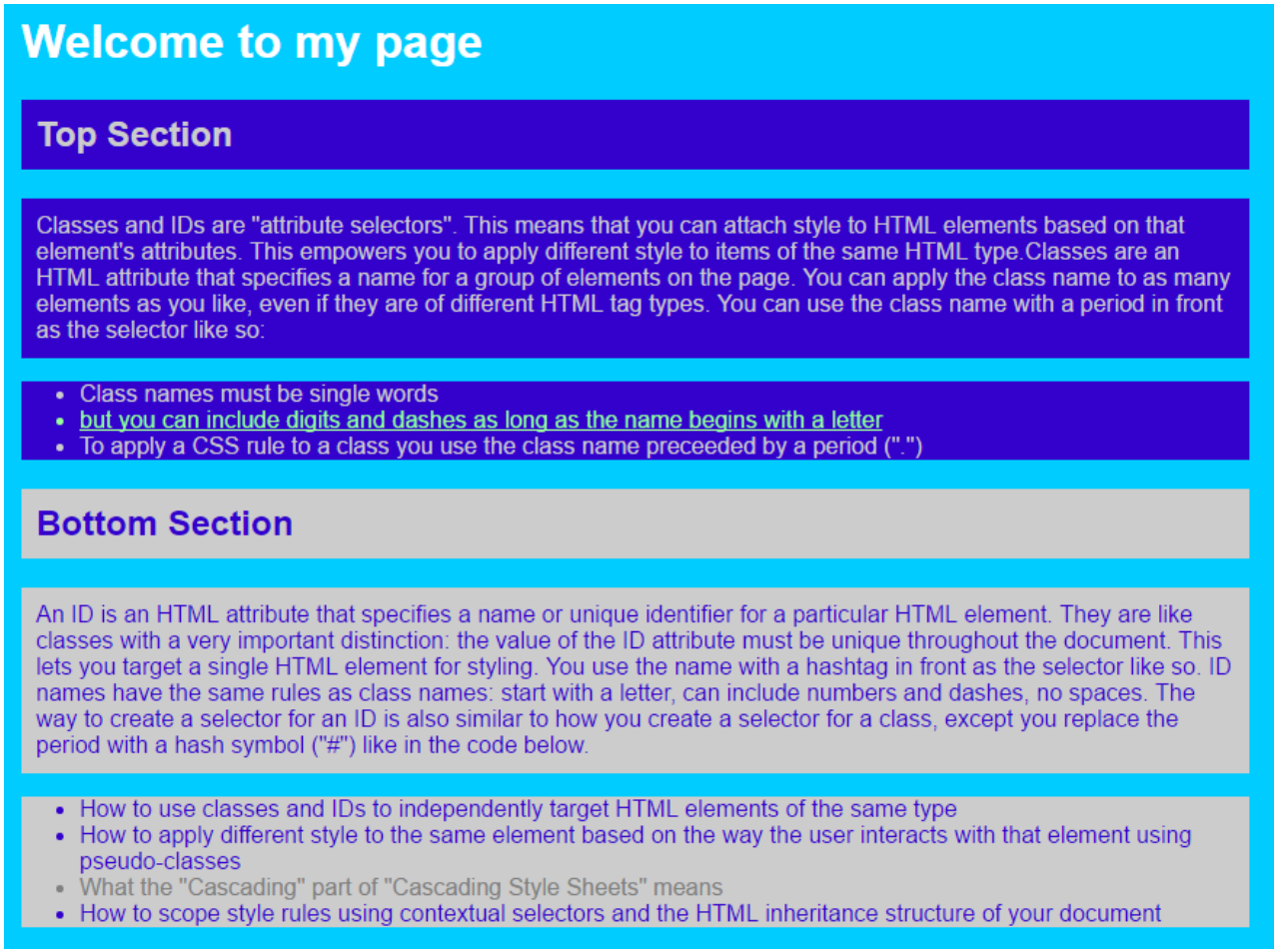
2.2.1 选择器

- 指示浏览器该条规则应用于HTML文档中的哪一个（些）标签
  - class选择器：为同一个class的多个元素设置样式
  - id选择器：为指定id的唯一元素设置样式
  - 元素选择器：为同一标签的全部元素设置样式
  - 伪类和伪元素：选择处于特殊状态的元素或者元素中特别的内容

	选择器	例子	例子描述
	*	*	选取所有元素。
属性选择器	.class	.intro	选取所有 class="intro" 的元素。
属性选择器	#id	#firstname	选取 id="firstname" 的那个元素。
元素选择器	element	p	选取所有 <p> 元素。
元素选择器	element1,element2,..	div, p	选取所有 元素和所有 <p> 元素。
上下文选择器	element1 element2	div p	选取 元素的所有后代 <p> 元素。
上下文选择器	element1>element2	div>p	选取 元素子代的所有 <p> 元素。

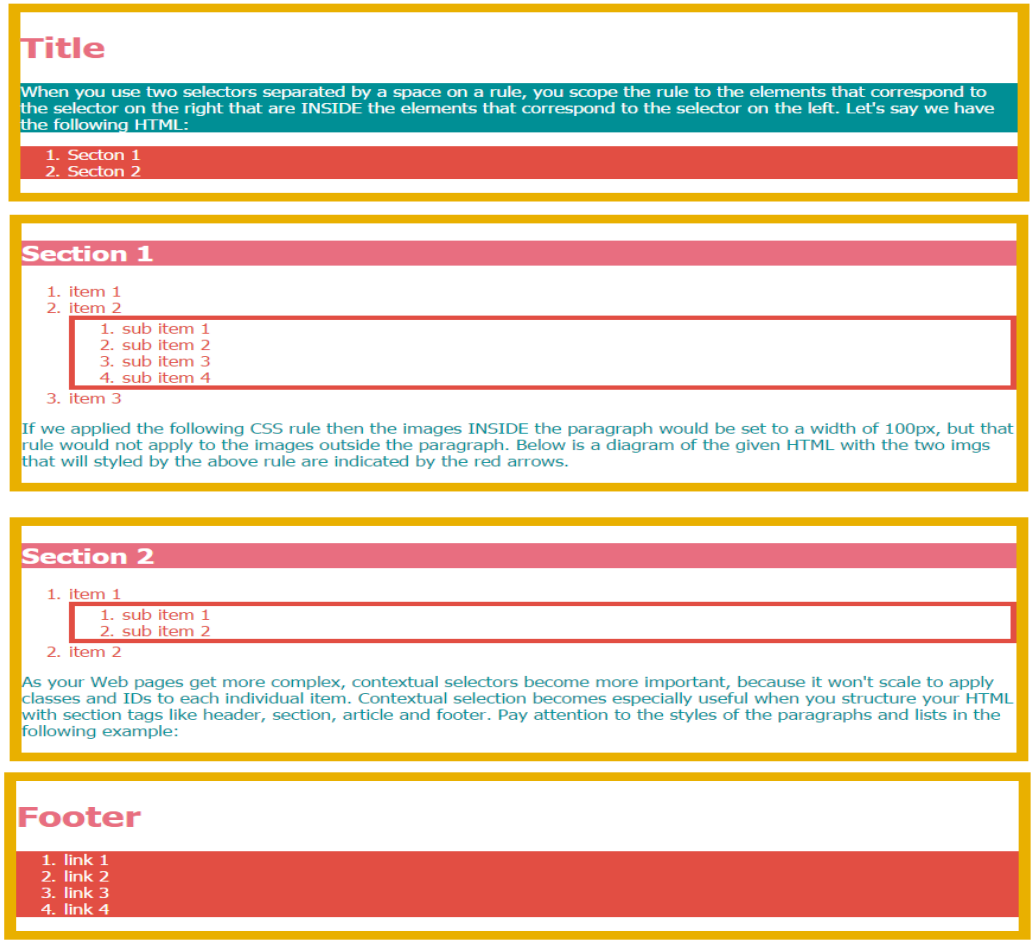
属性选择器

- https://codepen.io/techie4good/pen/VKkJYd



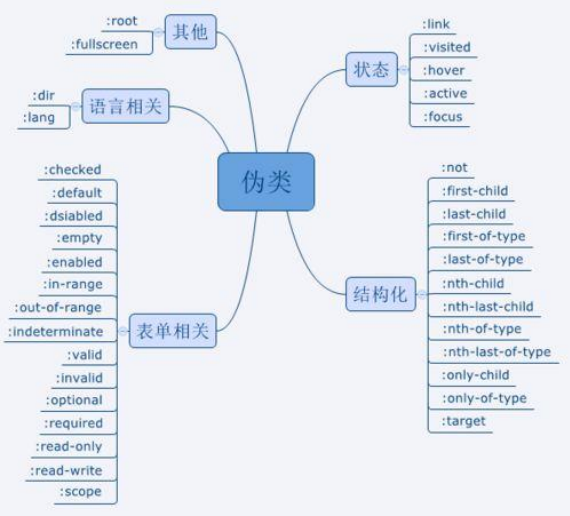
上下文选择器

- https://codepen.io/techie4good/pen/qaERNj



伪类和伪元素：实现基于文档树外信息的格式化

- 伪类：为处于某个状态的已有元素添加对应的样式，这个状态是根据用户行为而动态改变的
  - 状态伪类：基于元素当前状态进行选择的
  - 结构性伪类：通过文档结构的互相关系来匹配元素



:link 应用于未被访问过的链接；

:hover 应用于鼠标悬停到的元素；

:active 应用于被激活的元素；

:visited 应用于被访问过的链接，与:link互斥。

:focus 应用于拥有键盘输入焦点的元素。

状态伪类

结构性伪类

1. :first-child 选择某个元素的第一个子元素；
2. :last-child 选择某个元素的最后一个子元素；
3. :nth-child(n) 匹配属于其**父元素**的第 n个子元素，**不论元素的类型**；
4. :nth-last-child() 从这个元素的最后一个子元素开始算,选匹配属于其父元素的第 n个子元素，不论元素的类型；
5. :nth-of-type() 规定属于其父元素的第n个 指定 元素；
6. :nth-last-of-type() 从元素的最后一个开始计算,规定属于其父元素的指定 元素；
7. :first-of-type 选择一个上级元素下的第一个同类子元素；
8. :last-of-type 选择一个上级元素的最后一个同类子元素；
9. :only-child 选择它的父元素的唯一一个子元素；
10. :only-of-type 选择一个元素是它的上级元素的唯一一个相同类型的子元素；
11. :checked匹配被选中的input元素，这个input元素包括radio和checkbox。
12. :empty 选择的元素里面没有任何内容。
13. :disabled匹配禁用的表单元素。
14. :enabled匹配没有设置disabled属性的表单元素。
15. :valid匹配条件验证正确的表单元素。
16. :in-range选择具有指定范围内的值的 <input> 元素。
17. :optional选择不带 "required" 属性的 <input> 元素。
18. :focus选择获得焦点的 <input> 元素。


```
<html>
  <head>
    <style>
      body { background-color: black; }
      .nav-class {
        margin: 15px 20px; padding: 0 15px; font-size: 14px;
        display: -webkit-box; display: -ms-flexbox; display: flex;
        background: hsla(0,0%,100%,.9); border-radius: 6px;
      }
      .nav-item {
        padding: 8px 0; color: #000; text-align: center;
        -webkit-box-flex: 1; -ms-flex: 1; flex: 1;
      }
      .nav-item :first-child        { border-top: 2px solid #cf4a65; }
      .nav-item :first-child :hover { background: #cf4a65; color: #f2f2f2; }
      .nav-item :nth-child(2)       { border-top: 2px solid #e68654; }
      .nav-item :nth-child(2) :hover { background: #e68654; color: #f2f2f2; }
      .nav-item :nth-child(3)       { border-top: 2px solid #2db0d2; }
      .nav-item :nth-child(3) :hover { background: #2db0d2; color: #f2f2f2; }
      .nav-item :nth-last-child(2)       { border-top: 2px solid #dcbf55; }
      .nav-item :nth-last-child(2) :hover { background: #dcbf55; color: #f2f2f2; }
      .nav-item :last-child        { border-top: 2px solid #64b36a; }
      .nav-item :last-child :hover { background: #64b36a; color: #f2f2f2; }
    </style>
  </head>
  <body>
    <div class="nav-class">
      <div class="nav-item class-border-0"><span>全部</span></div>
      <div class="nav-item class-border-1"><span>投稿</span></div>
      <div class="nav-item class-border-2"><span>表白</span></div>
      <div class="nav-item class-border-3"><span>吐槽</span></div>
      <div class="nav-item class-border-4"><span>闲置</span></div>
    </div>
  </body>
</html>
```

注意：伪类与前置选择器之间不能有空格，此处为了排版增加空格

状态伪类:hover应用于鼠标悬停的元素

结构性伪类:first-child, :nth-child, :nth-last-child, :last-child应用于.nav-item类的第几个元素

伪类和伪元素

选择器	例子	例子描述
::after	p::after	在每个 <p> 元素之后插入内容。
::before	p::before	在每个 <p> 元素之前插入内容。
::first-letter	p::first-letter	选择每个 <p> 元素的首字母。
::first-line	p::first-line	选择每个 <p> 元素的首行。
::selection	p::selection	选择用户选择的元素部分。

- 伪元素：创建一些不在文档树中的元素，并为其添加样式

```
<html>
  <head>
    <style>
      h1::before {
        content: url("http://www.youwei.site/ruc.png");
      }
      h2::after {
        content: url("http://www.youwei.site/vruc.png ");
      }
      h3::selection {
        color: red; background: yellow;
      }
    </style>
  </head>

  <body>
    <h1>这个标题前面会插入中国人民大学的logo</h1>
    <br/>

    <h2>这个标题后面会插入微人大的logo</h2>
    <br/>

    <h3>这个标题选中的文本会被高亮显示</h3>
    <br/>
  </body>
</html>
```

伪元素::before, ::after, ::selection

2.2.2 样式定义与冲突处理

- 样式定义方式及优先级（优先级越高，对最终样式影响越大）

```
/*所有span元素的默认字体颜色样式*/

span { color: black; }
```

浏览器默认样式（优先级=1）

```
<head>
  <link rel="stylesheet" href="mystyle.css">
</head>
```

外部样式mystyle.css（优先级=2）

```
<head>
  span { color: limegreen; }
</head>
```

内部样式（优先级=3）

```
<body>
  <span style="color:limegreen">周冲</span>
</head>
```

内联样式（优先级=4）

样式名称	定义
浏览器默认样式	如果显式没有给出样式声明，HTML元素本身也有样式的默认设置
外部样式	写在独立的css文件中，例如mystyle.css
内部样式	在html文档的<head>结构内定义
内联样式	在标签内部直接定义

2.2.2 样式定义与冲突处理

- 同一级别下，重复定义样式：谁后定义听谁的

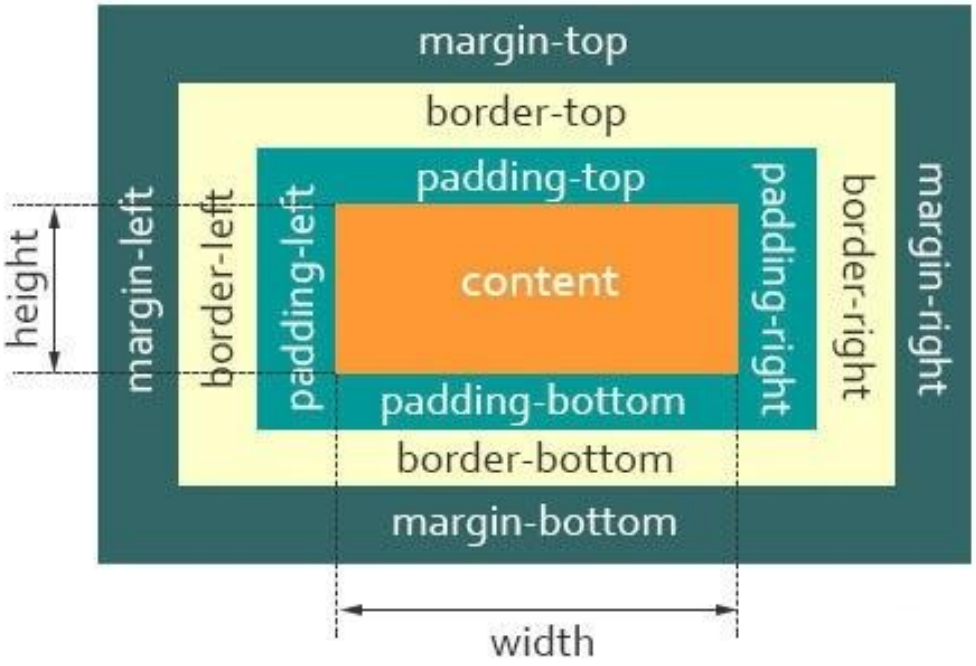
```
<head>
  <style>
    h2 { color: yellow; }
    h2 { color: blue; }
  </style>
</head>
<body> <h2>最终是蓝色</h2> </body>
<!-- 类比：相同变量的多次赋值，以最后一次赋值为准 -->
```

- 不同级别下，重复定义样式：谁优先级高听谁的

```
<head>
  <link rel="stylesheet" href="my.css">    <!--第一种：外部样式 -->
  <style> h2 { color: yellow; } </style>   <!--第二种：内部样式 -->
</head>
<body>
  <h2 style="color:green;"> 最终是绿色</h2> <!--第三种：内联样式 -->
</body>
<!-- 类比：局部变量与全局变量重名；子类对父类方法的重写 -->
```

2.2.3 盒子模型

- 每一个可以包裹内容的元素都可以定义盒子模型
  - 该模型分为内外两层盒子：外层盒子边框不可见，内层盒子边框可见
  - margin：定义两个盒子之间的距离（外边距）
  - padding：定义盒子内边框与内容之间的距离（内边距）

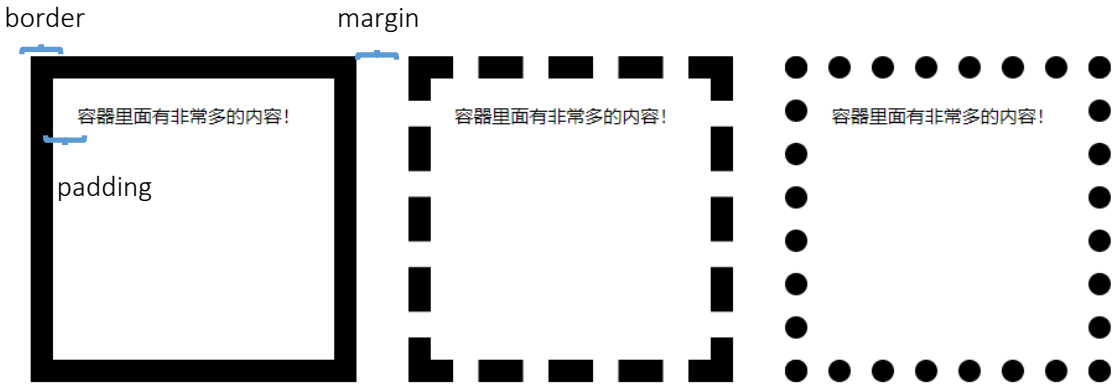


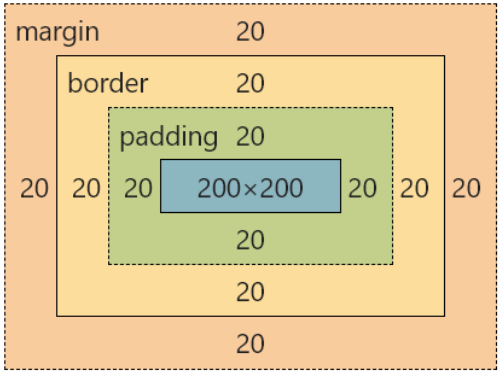
盒子宽度 = margin_left + margin_right
+ padding_left + padding_right
+ border_left + border_right
+ width

盒子高度 = margin_top + margin_bottom
+ padding_top + padding_bottom
+ border_top + border_bottom
+ height

2.2.3 盒子模型

```
<html>
  <head>
    <title>盒子模型</title>
    <style>
      .demo { display: inline-block; width:200px; height:200px;
              border:20px; margin:20px; padding: 20px; }
      #demo1 { border-style: solid; }
      #demo2 { border-style: dashed; }
      #demo3 { border-style: dotted; }
    </style>
  </head>
  <body>
    <div class="demo" id="demo1">容器里面的内容</div>
    <div class="demo" id="demo2">容器里面的内容</div>
    <div class="demo" id="demo3">容器里面的内容</div>
  </body>
</html>
```

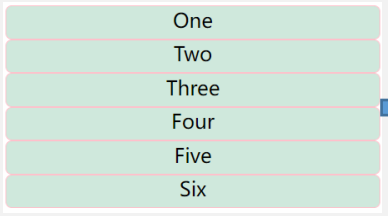




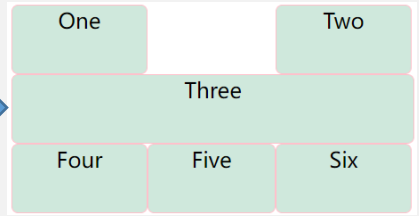
2.2.4 布局

```html
<html>
  <head>
    <style>
      .wrapper { width: 300px; }
      .wrapper > div {
        border-radius: 5px; border: 1px solid pink;
        background-color: rgb(207,232,220);
        padding: 0; margin: 0; text-align: center;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box1">One</div>
      <div class="box2">Two</div>
      <div class="box3">Three</div>
      <div class="box4">Four</div>
      <div class="box5">Five</div>
      <div class="box6">Six</div>
    </div>
  </body>
</html>
```

<!-- 默认布局 -->



<!-- 目标布局 -->




```css
<!-- Float布局 -->
.wrapper > div { height: 50px; }
.box1 { float: left; width: 100px; }
.box2 { float: right; width: 100px; }
.box3 { clear: both; width: 300px; }
.box4 { float: left; width: 100px; }
.box5 { float: left; width: calc(100% - 206px); }
.box6 { float: left; width: 100px; }
```

```css
<!-- Position布局: -->
.wrapper {position: relative;}
.wrapper > div { height: 50px; }
.box1 { position: absolute; left: 0px; top: 0px; width: 100px; }
.box2 { position: absolute; right: 0px; top: 0px; width: 100px; }
.box3 { position: absolute; left: 0px; top: 50px; width: 100%; }
.box4 { position: absolute; left: 0px; top: 100px; width: 100px; }
.box5 { position: absolute; left: 100px; top: 100px; width: 100px; }
.box6 { position: absolute; right: 0px; top: 100px; width: 100px; }
```

```css
<!- Flex布局: -->
.wrapper {
  width: 300px; display:flex;
  flex-wrap: wrap;
  flex-direction: row;
  justify-content: space-between;
}
.wrapper > div { height: 50px; }
.box1 { width: 100px; }
.box2 { width: 100px; }
.box3 { width: 100%; }
.box4 { width: 100px; }
.box5 { width: calc(100% - 206px); }
.box6 { width: 100px; }
```

```css
<!-- Grid布局: -->
.box2 { grid-column: 3; }
.box3 { grid-column: 1 / span 3; }
.wrapper {
  display: grid;
  grid-template-rows: 50px 50px 50px;
  grid-template-columns: 100px 100px 100px;
}
```

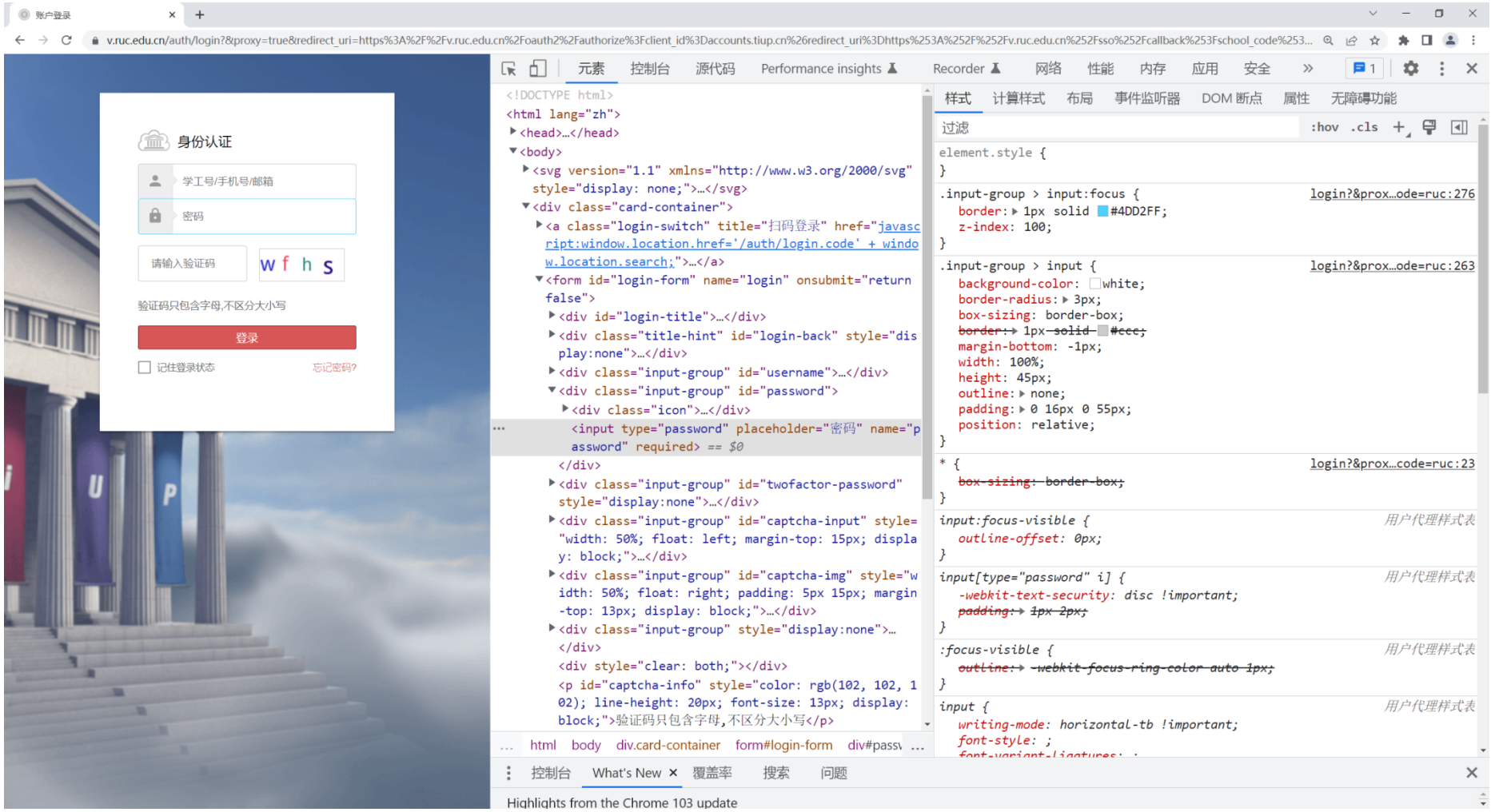
2.2.5 常用的样式


文字相关的样式
文本颜色
color:#333333;十六进制
color:red;关键字
color:rgb(255,255,0)；
文字大小
font-size:16px;
浏览器默认的字号16px 最小字号12px
文字背景色
background: red;
文字加粗
font-weight:bold;
文字字体
font-family: '微软雅黑'
文字下划线
text-decoration:none;去掉下划线
text-decoration:underline;加下划线
边框线
border: 1px solid #333333; (常用)
solid实线
dashed虚线
dottded点状
默认加粗文字样式去掉,如h1
font-weight:normal;
文字水平居中
text-align:center;作用是让块元素里的行内元素居中
文字垂直居中
height = line-height时
段落首行缩进实际（应用场景文章详情页面）
text-indent:2em;
首行缩进了2个文字的宽度
省略号
overflow:hidden;----溢出隐藏
overflow: auto;----自适应（内容过多会自动出滚动条）
text-overflow: ellipsis;---省略号
自动出省略号：四点
overflow:hidden;
text-overflow: ellipsis;
容器的宽度width；
white-space: nowrap;不换行
无序里li标签的点去掉
list-style:none;
背景的相关样式
背景图片的写法
background:url(*img/a.jpg*);
背景不平铺
background-repeat: no-repeat;
背景图片的位置 x,y
background-position: x,y;
x：水平方向 单位可以是px、% left居左/right居右/center居中
y：垂直方向 单位可以是px、% top居上/bottom居下/center居中
综合
background: url(*img/a.jpg*) no-repeat left center;
边框的相关样式
上边框
border-top:1px solid #333;
下边框
border-bottom:1px solid #333;
左边框
border-left:1px solid #333;
右边框
border-right:1px solid #333;
图片如何实现居中
法一：因为img行内块
将img转成块 display: block;
一定要给图片宽度width: npx;
居中margin: 0 auto;
（margin: auto auto;是错误的，auto不能上下居中）
法二：img外层套块级标签
将text-align: center;这个属性给在套img的块级标签上
img标签会有留白如何解决？
将img行内块标签转化为块就可以 ，用disply:block;
但是注意转成块以后的居中就不能用行居中的方式text-align:center;了，得用块居中的方式magin:auto;
行标签和块标签的转换化
特点
行标签
宽度和高度设置无效
span a
块标签
宽度和高度设置有效
div、p、h1-h6、ul、li、dl、dt、dd
行转块
display:block;
块转行
display:inline;
块元素转成行内块
display:inline-block;
本身存在bug（默认有空隙）
解决办法：把用了的标签与上一标签写在编译器同一行即可
清空默认
* {padding: 0;margin: 0;}
浏览器默认的外边距：8px
ul, li { list-style-type: none; }
a{text-decoration: none;}

2.2.5 常用的样式

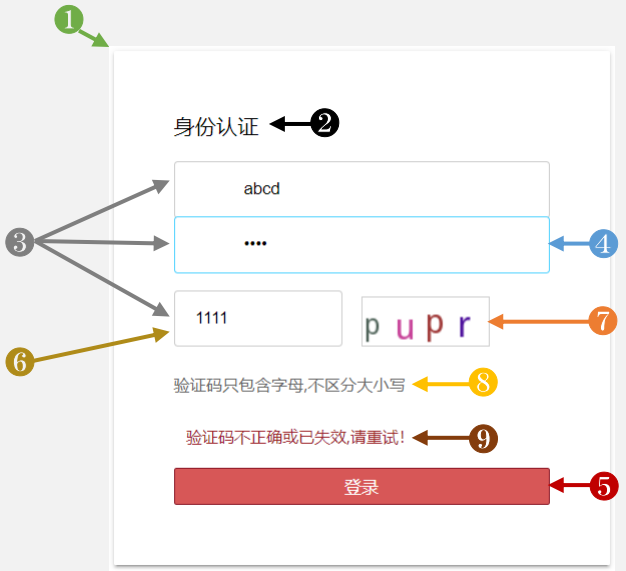

盒模型
padding内边距
应用场景：嵌套的
margin外边距
应用场景：两个独立的块与块之间的距离
值的意义
一个值
padding-top:20px;上边距
padding-bottom:20px;下边距
padding-left: 20px;左内边距
padding-right:20px;右内边距
两个值
margin: 10px 20px;
第一个值指的是上下
第二个值指的是左右
三个值
margin: 5px 8px 15px;
第一个值指的是上
第二个值指的是左右
第三个值指的是下
四个值
margin: 5px 10px 8px 20px;
第一个值是上
第二个值是右
第三个值是下
第四个值是左
padding和margin的区别
外边距能够调盒子的位置
内边距能够改变盒子的大小
margin自身存在外边距合并问题
取最大值进行合并
如何解决
一般在开发页面中，给其中一个盒子下边距，那么给其他盒子也一直下边距
注意事项
块元素-------支持上下左右 padding margin
行标签-------只支持左右padding margin上下padding margin不支持（跟转成块无关）
想让上下内边距生效----------把行标签转成块标签 display：block；
防止父代随子代中的margin塌陷
给父代加一个透明border
border: 1px solid transparent;
浮动
应用场景
横向导航
左浮动
float:left;
右浮动
float:right;
不浮动
float:none;
清除浮动
理由
只要元素浮动，那么就会出现问题（外层容器无法包裹住里层元素）
方法
.clearfix:after{display: block;clear: both;content:''; } 给在浮动元素父级
圆形
先做正方形
（宽高一样）
加border-radius:50%;宽度和高度一致
如何让一个盒子在当前的浏览器中：水平居中、垂直居中
绝对定位加负margin就可以实现： 负的盒子宽度的一半、负的盒子高度的一半
position: absolute;left:50%;top:50%;
使左上角的点定到浏览器正中
margin-left: -100px;margin-top: -100px;
定位
与浮动一样，都脱离了文档流，即加了定位后，转成行内块
绝对定位
position:absolute;
给了绝对定位，一定要给父代相对定位
（因为有时我们只给了绝对没有相对，在ie浏览器上会出现问题）
如果没有给相对，那么就会逐层向上查找，直到找到body标签为止
不占原来位置，盒子有父级时使用，定位指相对于body
相对定位
position:relative;
占原来位置，定位指相对于自己原来位置
固定定位
position:fixed;
相对于页面，但不滚动
只要有position就用top/bottom/right/left
top:0;距离上方bottom:0;距离下方left:0;距离左方right:0;距离右方
z-index：数字;
值越大层级越高
但是需要注意的是一定要与position连用
子绝父相
父亲是相对position: relative;
孩子是绝对position: absolute;

实例：使用CSS美化微人大认证页面



实例：使用CSS美化微人大认证页面

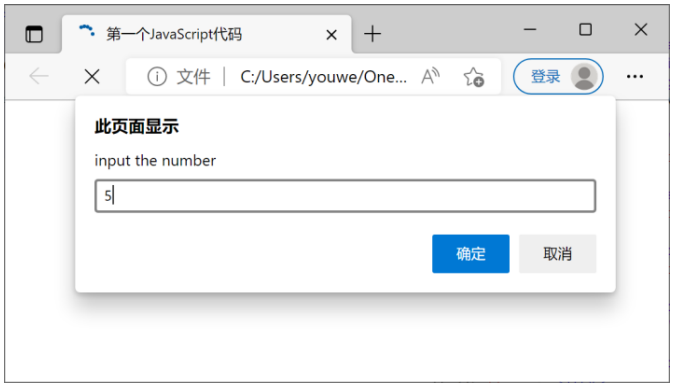
```css
/* style.css */

.card-container { 1
  max-width: 300px; margin: 8vw auto; background: #fff; width: 100%; padding: 2rem 3rem; border-radius: 1px;
  box-shadow: 0 2px 2px 0 rgb(0 0 0 / 14%), 0 3px 1px -2px rgb(0 0 0 / 20%), 0 1px 5px 0 rgb(0 0 0 / 12%);
}

h3 { font-size: 17px; font-weight: 400; } 2

.input-group > input { 3
  background-color: white; border-radius: 3px; box-sizing: border-box;
  border: 1px solid #ccc; margin-bottom: -1px; width: 100%; height: 45px;
  outline: none; padding: 0 16px 0 55px; position: relative;
}

.input-group > input:focus { border: 1px solid #4DD2FF; z-index: 100; } 4

.button {
  position: relative; font-size: 0.85rem; background: #d75757;
  border: 1px solid #891524; padding: .3rem 2rem;
  border-radius: 2px; width: 100%; cursor: pointer; overflow: hidden; 5
}
.button span { color: #fff; position: relative; z-index: 1;  }

#captcha-input { width: 45%; float: left; margin-top: 15px; display: block; } 6
#captcha-input input { padding-left: 16px; }

#captcha-img { width: 45%; float: right; padding: 5px 15px; margin-top: 15px; display: block; border: 1px solid #ccc;} 7
#captcha-img img { padding-left: 16px; border: 1px solid #ccc; }

#captcha-info { clear: both; color: rgb(102, 102, 102); line-height: 20px; font-size: 13px; display: block; margin-top: 80px; } 8

.alert { position: relative; margin: 0.5em,0,-1em,0; padding: 10px; border-radius: 2px; font-size: .8rem; color: #9c232a; } 9
```

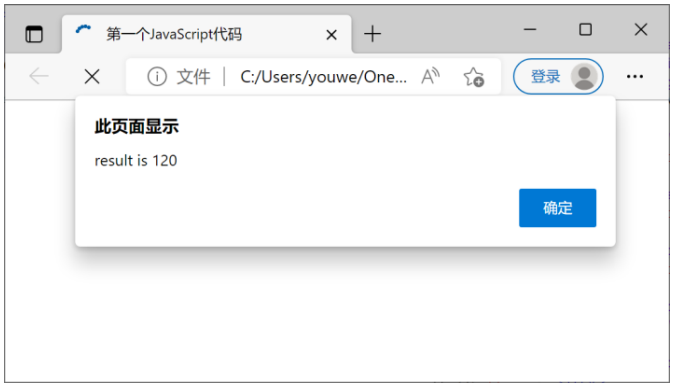


2.3 JavaScript

- JavaScript是一门基于对象的、弱类型、解释性语言，其运行于客户端浏览器，主要用于与页面元素进行动态交互

```
<html>
  <head>
    <title>第一个JavaScript代码</title>
  </head>
  <body>
    <script type="text/javascript">
      function factorial(n) {          //递归函数计算阶乘
        if (n == 1) return n;
        else return n * factorial(n-1);
      }
      n = prompt("input the number"); //类似于scanf
      var res = factorial(n);          //函数调用
      res = "result is " + res;        //变量类型自动转变
      alert(res);                      //类似于printf
      console.log(res);                //在控制台显示(F12)
    </script>
  </body>
</html>
```





学习更多：https://www.w3school.com.cn/js/index.asp

2.3.1 在网页中引入JavaScript代码

- 在页面中直接嵌入JavaScript

```
<html>
  <head>
    <script type="text/javascript"> alert("JavaScript in head"); </script>
  </head>
  <body>
    <script type="text/javascript"> alert("JavaScript in body"); </script>
  </body>
</html>
```

- 链接外部JavaScript文件

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

```
/* javascript_head.js */
alert("JavaScript in head");
```

```
/* javascript_body.js */
alert("JavaScript in body");
```

注意：在<head>...</head>中的脚本运行时机先于在<body>...</body>中的脚本，此时页面元素尚未完成装载。

2.3.1 在网页中引入JavaScript代码

- 作为标签的属性值使用
  - 通过“javascript:”调用JavaScript函数

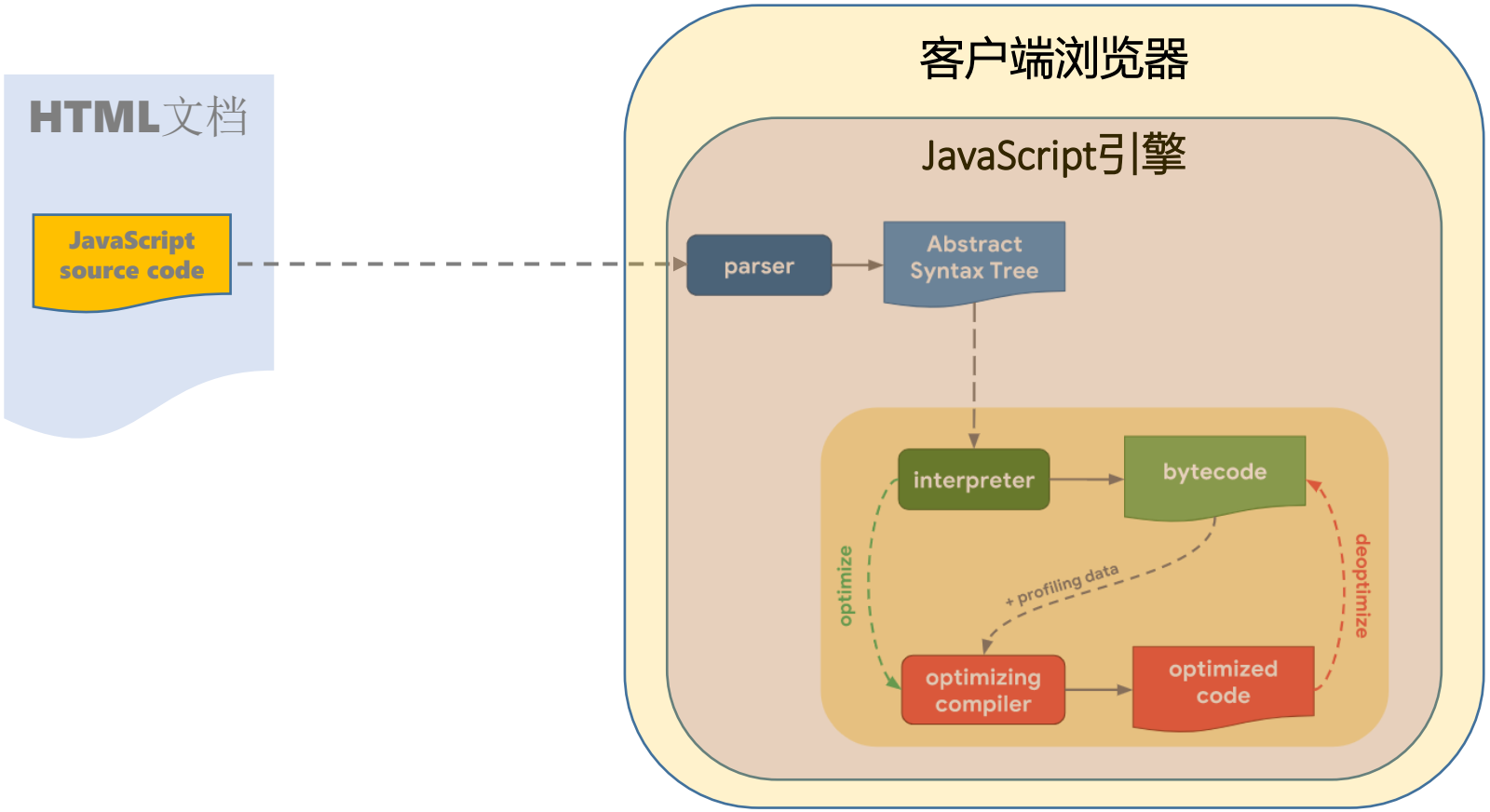
```
<html>
  <body>
    <a href="javascript:alert('你好JavaScript')">测试</a>
  </body>
</html>
```

  - 在事件中调用JavaScript函数

```
<html>
  <body>
    <input type="button" value="测试" onclick="alert('你好JavaScript')" />
  </body>
</html>
```

2.3.2 JavaScript代码执行流程

- 浏览器将嵌入HTML文档的JavaScript代码交由JavaScript引擎
- 解析->解释执行（非热点代码）->JIT编译执行（热点代码）



2.3.3 变量

- 定义方式
  - 使用var、let关键字，后跟变量名进行声明
  - 不使用var、let关键字声明，而是直接赋值
- 作用域：变量起作用的地方（空间）
  - 全局作用域：最外层函数定义的变量
    未声明而直接赋值的变量
  - 局部作用域：函数作用域（函数内部）
    块级作用域（代码块内部）
- 生命周期：变量生存的时段（时间）
  - 全局变量：从定义位置开始，一直到整个代码结束才会销毁
  - 局部变量：在函数中定义，在函数中使用，函数结束时自动销毁

```
var global_a = 1; //最外层函数定义的变量

function fun(x) {
  global_b = 2; //未声明而直接赋值的变量
  var local_a = 3; //函数作用域

  if (x > 0) {
    //块作用域，使用var声明提升至函数作用域
    var local_b = 4;
    //块作用域，使用let声明不发生变量提升
    let local_c = 5;
  }

  alert(local_a); //3
  alert(local_b); //4

  //ReferenceError: local_c is not defined
  try { alert(local_c); }
  catch(e) { alert(e); }
}

fun(1);
alert(global_a); //1
alert(global_b); //2
```

2.3.4 数据类型

- JavaScript是弱类型脚本语言，无需声明变量类型，运行时根据需要，自动确定变量的数据类型并进行数据类型转换
  - 基本类型存储在栈内存，把一个基本类型的变量赋值给另一个变量实际上进行的是值拷贝
  - 引用类型指针（地址）存储在栈内存，这个指针指向的对象存在堆内存，如果进行拷贝，拷贝的是栈内存中存储的指针，而不是对象真正的值

大类	数据类型	说明
基本类型	数值类型（Number）	包含整数和浮点数，特殊值：NaN, ±Infinity
	布尔类型（Boolean）	只有true和false两个值 【两个值小写】
	字符串类型（String）	表示一个字符序列，必须使用单引号（''）或双引号（""）括起来
	Undefined类型（Undefined）	用来确定一个已经创建但还没有赋初值的变量，只有一个值为：undefined 【小写】
	Null类型（Null）	表名某个变量的值为空，只有一个值为：null【小写】
引用类型	Object类型（Object）	对象由一些列属性（变量）和方法（函数）构成的集合，访问它们时用. (英文点号)
	数组（Array）	一系列的变量组成，数量元素的类型无需一致
	函数（Function）	包含一段可执行的代码

2.3.4 数据类型

```
n = 5.5;
alert(typeof(n)); //number

n = 5;
alert(typeof(n)); //number

n = true;
alert(typeof(n)); //boolean

n = "hello javascript";
alert(typeof(n)); //string

var m; //变量定义但未初始化
alert(typeof(m)); //undefined

try {alert(x);} //引用一个未定义的变量
catch(e) {alert(e);} //ReferenceError: x is not defined
```

```
n = null; //特殊值null被认为是一个空的对象引用
alert(typeof(n)); //object
alert(n instanceof Object); //null不是Object类实例

n = new Object(); //实例化一个对象
alert(typeof(n)); //object
alert(n instanceof Object); //Object类实例

n = [1, 2, 3];
alert(typeof(n)); // object
alert(n instanceof Array); //Array类实例
alert(n instanceof Object); //Object类实例

n = function() {};
alert(typeof(n)); // function
alert(n instanceof Function); //Function类实例
alert(n instanceof Object); //Object类实例
```

- 在对变量赋值时会自动改变变量的类型
- 引用未定义的变量，会引发异常
- typeof：判断某个变量/表达式的数据类型
- instancof：判断某个变量是否是指定类的实例，相当于C++的dynamic_cast

2.3.5 类型转换

- 类型转换分为：隐式类型转换和显式类型转换
  - 显式类型转换：开发人员在编写代码时，强制指定对值的类型进行转换
  - 隐式类型转换：对不同类型值使用运算符时，值可在类型之间自动转换

```
//ToNumber
str1 = "123", str2 = "123.456", str3 = "12xyz";
num11 = Number(str1), num12 = Number(str2), num13 = Number(str3);              //显式转换
num21 = parseInt(str1), num22 = parseInt(str2), num23 = parseInt(str3);        //显式转换
num31 = parseFloat(str1), num32 = parseFloat(str2), num33 = parseFloat(str3);  //显式转换
num41 = str1 - 3, num42 = str1 - str2, num43 = str1 - str3;                    //隐式转换
alert(num11); /* 123 */          alert(num12); /* 123.456 */       alert(num13); /* NaN */
alert(num21); /* 123 */          alert(num22); /* 123 */           alert(num23); /* 12 */
alert(num31); /* 123 */          alert(num32); /* 123.456 */       alert(num33); /* 12 */
alert(num41); /* 120 */          alert(num42); /* -0.4560... */    alert(num43); /* NaN */

//ToString
boola = true, numb = 123;
stra = String(boola); strb = numb.toString();                                  //显式转换
strc = numb + "456";                                                           //隐式转换
alert(stra); /* true */          alert(strb); /* 123 */           alert(strc); /* 123456 */

//ToBoolean
strx = "123", numy = 0, strz = "";
boolx = Boolean(strx), booly = Boolean(numy);                                  //显式转换
boolz = !!strz;                                                                //隐式转换
alert(boolx); /* true */          alert(booly); /* false */         alert(boolz); /* false */
```

ToNumber Conversions

Argument Type	Result
Undefined	Return **NaN**.
Null	Return **+0**$_{\mathbb{F}}$.
Boolean	If *argument* is **true**, return **1**$_{\mathbb{F}}$. If *argument* is **false**, return **+0**$_{\mathbb{F}}$.
Number	Return *argument* (no conversion).
String	Return ! StringToNumber(*argument*).
Object	Apply the following steps: 1. Let *primValue* be ? ToPrimitive(*argument*, number). 2. Return ? ToNumber(*primValue*).

ToString Conversions

Argument Type	Result
Undefined	Return **"undefined"**.
Null	Return **"null"**.
Boolean	If *argument* is **true**, return **"true"**. If *argument* is **false**, return **"false"**.
Number	Return Number::toString(*argument*).
String	Return *argument*.
Object	Apply the following steps: 1. Let *primValue* be ? ToPrimitive(*argument*, string). 2. Return ? ToString(*primValue*).

ToBoolean Conversions

Argument Type	Result
Undefined	Return **false**.
Null	Return **false**.
Boolean	Return *argument*.
Number	If *argument* is **+0**$_{\mathbb{F}}$, **-0**$_{\mathbb{F}}$, or **NaN**, return **false;** otherwise return **true**.
String	If *argument* is the empty String (its length is 0), return **false**; otherwise return **true**.
Object	Return **true**.

原始值	转换为数值	转换为字符串	转换为布尔值
false	0	"false"	false
true	1	"true"	true
0	0	"0"	false
1	1	"1"	true
"0"	0	"0"	**true**
"000"	0	"000"	**true**
"1"	1	"1"	true
NaN	NaN	"NaN"	false
Infinity	Infinity	"Infinity"	true
-Infinity	-Infinity	"-Infinity"	true
""	**0**	""	**false**
"20"	20	"20"	true

原始值	转换为数值	转换为字符串	转换为布尔值
"Runoob"	NaN	"Runoob"	true
[ ]	**0**	""	true
[20]	**20**	"20"	true
[10,20]	NaN	"10,20"	true
["Runoob"]	NaN	"Runoob"	true
["Runoob","Google"]	NaN	"Runoob,Google"	true
function(){}	NaN	"function(){}"	true
{ }	NaN	"[object Object]"	true
null	**0**	"null"	false
undefined	NaN	"undefined"	false

学习更多：
https://tc39.es/ecma262/#sec-type-conversion

2.3.6 运算符

- 与C/C++的运算符类似
  - 优先级：决定了表达式中运算执行的先后顺序
  - 结合性：多个具有同样优先级的运算符表达式中的运算顺序

优先级	运算符	说明	结合性
1	[]、.、()	字段访问、数组索引、函数调用和表达式分组	从左向右
2	++ -- -~!delete new typeof void	一元运算符、返回数据类型、对象创建、未定义的值	从右向左
3	*、/、%	相乘、相除、求余数	从左向右
4	+、-	相加、相减、字符串串联	从左向右
5	<<、>>、>>>	左位移、右位移、无符号右移	从左向右
6	<、<=、>、>=、instanceof	小于、小于或等于、大于、大于或等于、是否为特定类的实例	从左向右
7	==、!=、===、!==	相等、不相等、全等，不全等	从左向右

优先级	运算符	说明	结合性
8	&	按位“与”	从左向右
9	^	按位“异或”	从左向右
10	|	按位“或”	从左向右
11	&&	短路与（逻辑“与”）	从左向右
12	||	短路或（逻辑“或”）	从左向右
13	?:	条件运算符	从右向左
14	=、+=、-=、*=、/=、%=、&=、|=、^=、<、<=、>、>=、>>=	混合赋值运算符	从右向左
15	,	多个计算	按优先级计算，然后从右向左

使用（）改变默认的优先级和结合性

2.3.6 运算符

- +：既是字符串运算符又是加法运算符（类比C++操作符重载）
  - 左值/右值中只要有一个是string类型，则进行字符串拼接
  - 否则尝试把左值/右值均转化内成number类型，进行算术加法

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

The abstract operation ApplyStringOrNumericBinaryOperator takes arguments *lval* (an ECMAScript language value), *opText* (**, *, /, %, +, -, <<, >>, >>>, &, ^, or |), and *rval* (an ECMAScript language value) and returns either a normal completion containing either a String, a BigInt, or a Number, or a throw completion. It performs the following steps when called:

1. If *opText* is +, then
   a. Let *lprim* be ? ToPrimitive(*lval*).
   b. Let *rprim* be ? ToPrimitive(*rval*).
   c. If Type(*lprim*) is String or Type(*rprim*) is String, then
      i. Let *lstr* be ? ToString(*lprim*).
      ii. Let *rstr* be ? ToString(*rprim*).
      iii. Return the string-concatenation of *lstr* and *rstr*.
   d. Set *lval* to *lprim*.
   e. Set *rval* to *rprim*.
2. NOTE: At this point, it must be a numeric operation.
3. Let *lnum* be ? ToNumeric(*lval*).
4. Let *rnum* be ? ToNumeric(*rval*).
5. If Type(*lnum*) is different from Type(*rnum*), throw a **TypeError** exception.
6. If Type(*lnum*) is BigInt, then
   a. If *opText* is **, return ? BigInt::exponentiate(*lnum*, *rnum*).
   b. If *opText* is /, return ? BigInt::divide(*lnum*, *rnum*).
   c. If *opText* is %, return ? BigInt::remainder(*lnum*, *rnum*).
   d. If *opText* is >>>, return ? BigInt::unsignedRightShift(*lnum*, *rnum*).
7. Let *operation* be the abstract operation associated with *opText* and Type(*lnum*) in the following table:
8. Return *operation*(*lnum*, *rnum*).

```
a = "3.145";          //字符串
b = 3.145;            //数值

c = a+2;              //字符串拼接
d = b+2;              //算数加法

//使用显式类型转换改变默认的隐式类型转换规则
e = String(Number(a) + 2);

alert(c); //3.1452
alert(d); //5.145
alert(e); //5.145
```

学习更多：
https://tc39.es/ecma262/#sec-additive-operators

2.3.6 运算符

- ==/!=和===/!==区别
  - ==/!=：先进行类型转换，再比较值（值相等，则为true） <= LooselyEqual
  - ===/!==：比较类型和值（值相等且类型相同，则为true）<= StrictlyEqual

IsLooselyEqual (*x*, *y*)

The abstract operation IsLooselyEqual takes arguments *x* (an ECMAScript language value) and *y* (an ECMAScript language value) and returns either a normal completion containing a Boolean or a throw completion. It provides the semantics for the comparison *x* == *y*. It performs the following steps when called:

1. If Type(*x*) is the same as Type(*y*), then
   a. Return IsStrictlyEqual(*x*, *y*).
2. If *x* is **null** and *y* is **undefined**, return **true**.
3. If *x* is **undefined** and *y* is **null**, return **true**.
4. NOTE: This step is replaced in section B.3.6.2.
5. If Type(*x*) is Number and Type(*y*) is String, return ! IsLooselyEqual(*x*, ! ToNumber(*y*)).
6. If Type(*x*) is String and Type(*y*) is Number, return ! IsLooselyEqual(! ToNumber(*x*), *y*).
7. If Type(*x*) is BigInt and Type(*y*) is String, then
   a. Let *n* be StringToBigInt(*y*).
   b. If *n* is **undefined**, return **false**.
   c. Return ! IsLooselyEqual(*x*, *n*).
8. If Type(*x*) is String and Type(*y*) is BigInt, return ! IsLooselyEqual(*y*, *x*).
9. If Type(*x*) is Boolean, return ! IsLooselyEqual(! ToNumber(*x*), *y*).
10. If Type(*y*) is Boolean, return ! IsLooselyEqual(*x*, ! ToNumber(*y*)).
11. If Type(*x*) is either String, Number, BigInt, or Symbol and Type(*y*) is Object, return ! IsLooselyEqual(*x*, ? ToPrimitive(*y*)).
12. If Type(*x*) is Object and Type(*y*) is either String, Number, BigInt, or Symbol, return ! IsLooselyEqual(? ToPrimitive(*x*), *y*).
13. If Type(*x*) is BigInt and Type(*y*) is Number, or if Type(*x*) is Number and Type(*y*) is BigInt, then
    a. If *x* or *y* are any of **NaN**, $+\infty_{\mathbb{F}}$, or $-\infty_{\mathbb{F}}$, return **false**.
    b. If $\mathbb{R}(x) = \mathbb{R}(y)$, return **true**; otherwise return **false**.
14. Return **false**.

IsStrictlyEqual (*x*, *y*)

The abstract operation IsStrictlyEqual takes arguments *x* (an ECMAScript language value) and *y* (an ECMAScript language value) and returns a Boolean. It provides the semantics for the comparison *x* === *y*. It performs the following steps when called:

1. If Type(*x*) is different from Type(*y*), return **false**.
2. If Type(*x*) is Number, then
   a. Return Number::equal(*x*, *y*).
3. If Type(*x*) is BigInt, then
   a. Return BigInt::equal(*x*, *y*).
4. Return SameValueNonNumeric(*x*, *y*).

```
alert(5=="5");          //true
alert(5==="5");         //false

alert(1==true);         //true
alert(1===true);        //false

alert(false=="");       //true
alert(false==="");      //false
```

学习更多：
https://tc39.es/ecma262/#sec-equality-operators

2.3.7 函数

- 函数的主要功能是将代码组织为可复用的单位，可以完成特定的任务并返回数据
- 函数是JavaScript组织代码的基本单位

```
function 函数名( [ 参数1, [ 参数2, [ 参数N ] ] ] )
{
        [ 语句组 ];
        [ return [表达式] ];
}
```

- **function**：必选项，定义函数用的关键字。
- 函数名：必选项，合的**JavaScript**标识符。
- 参数：可选项，合法的**JavaScript**标识符，外部的数据可以通过参数传送到函数内部。
- 语句组：可选项，**JavaScript**程序语句，当为空时函数没有任何动作
- **return**：可选项，遇到此指令函数执行结束并返回，当省略该项时函数将在右花括号处结束。
- 表达式：可选项，其值作为函数返回值。

函数表达式

- 将声明的函数赋值给一个变量，通过变量完成函数调用和参数的传递，它也是JavaScript中另一种实现自定义函数的方式

① 函数的定义方式不同

② 函数的调用方式不同

③ 函数定义与调用顺序不同

```
var fn = function sum(num1, num2) {
  return num1 + num2;
};
fn();
```

函数表达式

```
sum();
function sum(num1, num2) {
  return num1 + num2;
};
```

函数声明方式

匿名函数

- 定义：没有函数名称的函数，既是函数表达式的另一种表示形式，又可以通过函数声明的方式实现调用
- 作用：可以有效避免全局变量污染以及函数名的冲突问题

① 函数表达式中省略函数名

```
var fn = function (num1, num2) {
  return  num1 + num2;
};
fn(1, 2);
```

② 自调用方式

```
(function (num1, num2) {
  return  num1 + num2;
})(2, 3);
```

③ 处理事件

```
document.body.onclick  = function () {
  alert('Hi, everybody!');
};
```

回调函数

- 定义：一个函数A作为参数传递给一个函数B，然后在B函数体内调用函数A，此时，我们称函数A为回调函数
- 作用：函数体中某部分功能由调用者决定，此时可用回调函数

方法名称	功能描述
find()	返回数组中满足回调函数的第一个元素的值，否则返回undefined
every()	测试数组的所有元素是否都通过了回调函数的测试
some()	测试数组中的某些元素是否通过由回调函数实现的测试
forEach()	对数组的每个元素执行一次提供的函数
map()	创建一个新数组，其结果是该数组中的每个元素都调用一次提供的回调函数后返回的结果
reduce()	对累加器和数组中的每个元素（从左到右）应用一个函数，将其减少为单个值
reduceRight()	接收一个函数作为累加器（accumulator）和数组的每个值（从右到左）将其减少为单个值

在JavaScript中为数组提供了很多利用回调函数实现具体功能的方法

回调函数

- 以map()方法为例进行演示，对arr数组中的每个元素都按顺序调用一次回调函数。

```
var arr = ['a', 'b', 'c'];
arr.map(function(value, index) {
  console.log(value, index);
});
```

- 参数：

map()的参数是一个回调函数fn。

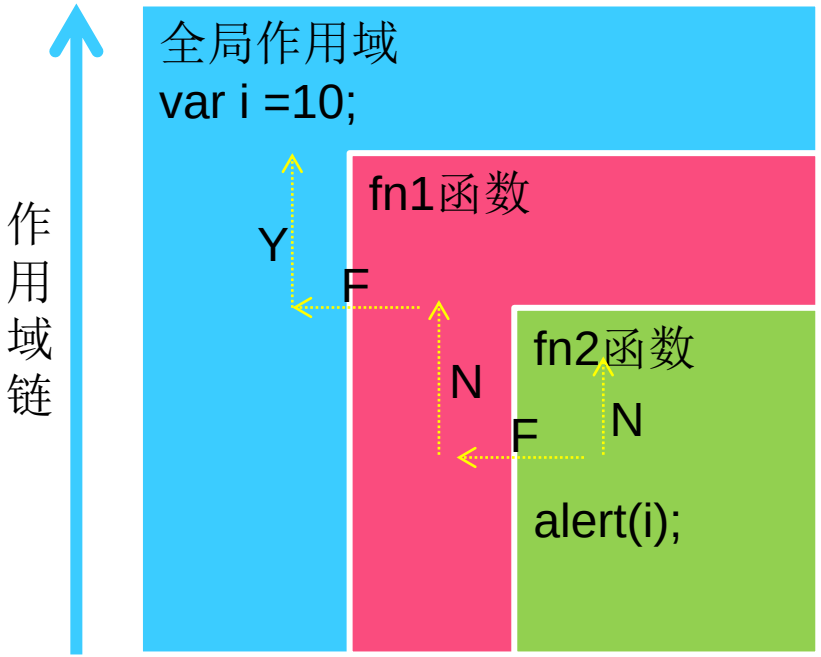
fn的第1个参数表示当前数组的元素。

fn的第2个参数表示对应元素的索引下标。

- 返回值：回调函数每次执行后的返回值会组合起来形成一个新数组。
- 示例结果：在控制台依次可查看到，“a 0”、“b 1”和“c 2”。

函数嵌套与作用域链

- 函数嵌套：在一个函数内部存在另一个函数的声明
- 作用域链：各层嵌套函数局部作用域由内到外构成的链
  - 内层函数只能在外层函数作用域内执行
  - 在内层函数执行过程中，若需要引入某个变量，会在当前作用域中寻找
  - 若未找到，则继续向上一层级的作用域中寻找，直到全局作用域链



符号含义参考

F：表示向上查找；N：没有找到；Y：找到

```
<script type="text/javascript">
  var i = 10; //全局变量i

  function fn1() { //在全局作用域定义函数fn1
    function fn2() { //在fn1局部作用域定义函数fn2
      alert(i);
    }
    fn2(); //在fn1局部作用域调用函数fn2
  }
  fn1();
</script>
```

2.3.8 对象

- JavaScript是基于对象的语言，JavaScript中的对象本质是由名-值对构成的集合
- 分类：
  - 内置对象
  - 浏览器对象
  - 自定义对象

内置对象

- Math数学对象：所有基本数学计算的功能和常量的属性和方法
- Date日期对象：获取、设置日期和时间的属性和方法
- String字符串对象：对字符串进行处理的属性和方法
- Array数组对象：数组操作相关的属性和方法

方法	作用
abs	返回某数的绝对数
ceil	返回大于或等于指定数的最小整数
floor	返回小于等于其数值参数的最大整数
random	返回[0-1)之间的随机数
round	返回四舍五入后的整数

Math对象

String对象

属性/方法	说明	用法
length	字符串长度	strObj.length
charAt	返回指定索引位置的字符	strObj.charAt(index)
concat	进行字符串连接	str1.concat(str2,str3)
indexOf	String 对象内第一次出现子字符串的字符位置	str1.indexOf(str2[, startIndex])
lastIndexOf	返回 String 对象中子字符串最后出现的位置	str1.lastIndexOf(str2[, startindex])
substring	返回 String 对象中指定位置的子字符串	**str1.substring(start, end)**
toLowerCase toUpperCase	大小写转换	str1.toLowerCase( )

方法	作用	返回值
getYear	返回年度	
getMonth	返回月份	0-11
getDate	返回日期	1-31
getDay	返回星期几	0-6
getHours	返回小时数	0-23
getMinutes	返回分钟数	0-59
getSeconds	返回秒数	0-59

Data对象

Array对象

属性/方法	作用	格式
length	获得数组元素个数	数组.length
toString	数组转换为字符串,各个数组元素用逗号连接	数组.toString()
join	将数组组合为字符串,由分隔符隔开	数组.join(分隔符)
push	添加若干个元素到数组末端	数组.push(元素1,元素2)
pop	删除最后一个元素	数组.pop()
reverse	将数组内元素的索引次序翻转	数组.reverse()

内置对象

```
<!-- Math对象使用实例：随机生成一个0-9之间的整数 -->

<script type="text/javascript">
  x = Math.random(); //返回一个[0,1)之间的随机浮点数x
  y = x * 10;        //y是一个[0, 10)之间的随机浮点数
  z = floor(y);      //对y下取整，得一个[0, 10)之间的整数
  alert(z);
</script>
```

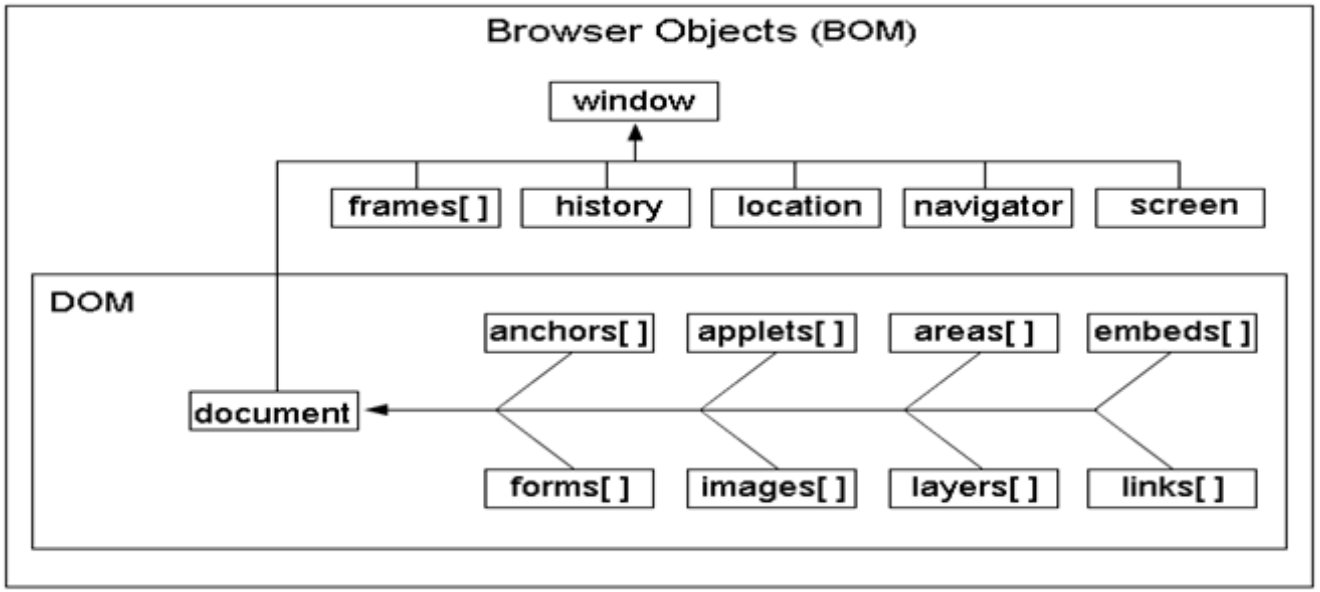
```
<!-- String对象使用实例：获得URL字符串中的主机名 -->

<script type="text/javascript">
  url = "http://www.youwei.site/ruc.png";
  start = url.indexOf("//");       //查找“//”所在位置
  end = url.indexOf("/", start+2); //查找“/”所在位置
  host = url.substring(start+2,end); //[start+2,end)子串
</script>
```

```
<!-- Date对象使用实例：获取当前日期的年/月/日和时/分/秒 -->

<script type="text/javascript">
  date = new Date();
  year = date.getYear()+1900; //getYear()+1900为当前年份
  month = date.getMonth()+1;  //getMonth()+1为当前月份
  day = date.getDay();        //getDay()为当前日期
  hour = date.getHours();     //getDay()为当前时点
  minute = date.getMinutes(); //getDay()为当前分点
  second = date.getSeconds(); //getDay()为当前秒点
</script>
```

```
<!-- Array对象使用实例：用数组实现栈/队列操作 -->

<script type="text/javascript">
  stack = [];      //模拟栈，后进先出
  stack.push(1); stack.push(2);
  alert(stack.pop()); alert(stack.pop()); //2 1

  queue = Array(); //模拟队列，先进先出
  queue.unshift(1); queue.unshift(2);
  alert(queue.pop()); alert(pop.shift()); //1 2
</script>
```

2.4 BOM/DOM

- BOM（Browser Object Model）：浏览器对象模型，定义了操作浏览器的标准方法，根对象是window
- DOM（Document Object Model）：文档对象模型，定义了操作HTML文档的标准方法，根对象是document



2.4.1 BOM

- BOM：浏览器对象模型，用于访问和网页无关的浏览器功能。

☑ **window**：处于对象层次的最顶端,提供处理窗口的方法和属性；

☑ **document**：装入到当前窗口的文档，包含一个文档所有的标签元素，将其封装供使用；

☑ **location**：用于创建、访问或修改URL对象；

☑ **history**：包含客户机先前已经请求过的URL；

☑ **screen**：提供关于用户屏幕的信息，如大小颜色等；

☑ **navigator**：提供关于浏览器对象的信息,包括浏览器版本、型号等信息

学习更多：https://www.w3school.com.cn/js/js_window.asp

window

- 功能：表示浏览器的一个实例，同时也是一个Global对象

属性	描述
closed	返回窗口是否已被关闭。
defaultStatus	设置或返回窗口状态栏中的默认文本。
innerHeight	返回窗口的文档显示区的高度。
innerWidth	返回窗口的文档显示区的宽度。
length	设置或返回窗口中的框架数量。
name	设置或返回窗口的名称。
opener	返回对创建此窗口的窗口的引用。
outerHeight	返回窗口的外部高度，包含工具条与滚动条。
outerWidth	返回窗口的外部宽度，包含工具条与滚动条。
pageXOffset	设置或返回当前页面相对于窗口显示区左上角的 X 位置。

属性	描述
pageYOffset	设置或返回当前页面相对于窗口显示区左上角的 Y 位置。
parent	返回父窗口。
screenLeft	返回相对于屏幕窗口的x坐标
screenTop	返回相对于屏幕窗口的y坐标
screenX	返回相对于屏幕窗口的x坐标
screenY	返回相对于屏幕窗口的y坐标
self	返回对当前窗口的引用。等价于 Window 属性。
status	设置窗口状态栏的文本。
top	返回最顶层的父窗口。

方法	描述
alert()	显示带有一段消息和一个确认按钮的警告框。
atob()	解码一个 base-64 编码的字符串。
btoa()	创建一个 base-64 编码的字符串。
blur()	把键盘焦点从顶层窗口移开。
clearInterval()	取消由 setInterval() 设置的 timeout。
clearTimeout()	取消由 setTimeout() 方法设置的 timeout。
close()	关闭浏览器窗口。
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框。
createPopup()	创建一个 pop-up 窗口。
focus()	把键盘焦点给予一个窗口。
getSelection()	返回一个 Selection 对象，表示用户选择的文本范围或光标的当前位置。
getComputedStyle()	获取指定元素的 CSS 样式。
matchMedia()	该方法用来检查 media query 语句，它返回一个 MediaQueryList对象。
moveBy()	可相对窗口的当前坐标把它移动指定的像素。

方法	描述
moveTo()	把窗口的左上角移动到一个指定的坐标。
open()	打开一个新的浏览器窗口或查找一个已命名的窗口。
print()	打印当前窗口的内容。
prompt()	显示可提示用户输入的对话框。
resizeBy()	按照指定的像素调整窗口的大小。
resizeTo()	把窗口的大小调整到指定的宽度和高度。
scroll()	已废弃。 该方法已经使用了 scrollTo() 方法来替代。
scrollBy()	按照指定的像素值来滚动内容。
scrollTo()	把内容滚动到指定的坐标。
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式。
setTimeout()	在指定的毫秒数后调用函数或计算表达式。
stop()	停止页面载入。
postMessage()	安全地实现跨源通信。

窗口操作

- 打开新窗口：win = window.open(url, name, features, replace)
- 关闭窗口：win.close()

参数	说明
URL	可选。打开指定的页面的URL。如果没有指定URL，打开一个新的空白窗口
name	可选。指定target属性或窗口的名称。支持以下值： • _blank - URL加载到一个新的窗口。这是默认 • _parent - URL加载到父框架 • _self - URL替换当前页面 • _top - URL替换任何可加载的框架集 • *name* - 窗口名称
replace	Optional.Specifies规定了装载到窗口的 URL 是在窗口的浏览历史中创建一个新条目，还是替换浏览历史中的当前条目。支持下面的值： • true - URL 替换浏览历史中的当前条目。 • false - URL 在浏览历史中创建新的条目。

```
<html>
<body>
  <input type="submit" value="打开"
         onclick="window.win = open('http://www.ruc.edu.cn',
                                    '_blank','width=200,height=200')">
  <input type="submit" value="关闭"
         onclick="win.close();")>
</body>
</html>
```

参数	说明
specs	可选。一个逗号分隔的项目列表。支持以下值：

channelmode=yes|no|1|0	是否要在影院模式显示 window。默认是没有的。仅限IE浏览器
directories=yes|no|1|0	是否添加目录按钮。默认是肯定的。仅限IE浏览器
fullscreen=yes|no|1|0	浏览器是否显示全屏模式。默认是没有的。在全屏模式下的 window，还必须在影院模式。仅限IE浏览器
height=pixels	窗口的高度。最小.值为100
left=pixels	该窗口的左侧位置
location=yes|no|1|0	是否显示地址字段.默认值是yes
menubar=yes|no|1|0	是否显示菜单栏.默认值是yes
resizable=yes|no|1|0	是否可调整窗口大小.默认值是yes
scrollbars=yes|no|1|0	是否显示滚动条.默认值是yes
status=yes|no|1|0	是否要添加一个状态栏.默认值是yes
titlebar=yes|no|1|0	是否显示标题栏.被忽略，除非调用HTML应用程序或一个值得信赖的对话框.默认值是yes
toolbar=yes|no|1|0	是否显示浏览器工具栏.默认值是yes
top=pixels	窗口顶部的位置.仅限IE浏览器
width=pixels	窗口的宽度.最小.值为100

延时执行与周期性执行

- 延迟代码执行：tid = setTimeout(code, delay)
- 取消延迟执行：clearTimeout(tid)
- 周期性执行：tid = setInterval(code, interval)
- 取消周期性执行：clearInterval(tid)

```
<html>
<head>
  <script>
    function showClock() //自定义函数
    {
      d = new Date(); //创建一个时间对象
      document.title = d.toLocaleString(); //标题上显示当前时间
      tid = window.setTimeout("showClock()",1000); //1秒更新一次
    }
  </script>
</head>
<body>
  <input type="submit" value="开始"
         onclick="showClock()">
  <input type="submit" value="取消"
         onclick="window.clearTimeout(tid);")>
</body>
</html>
```

```
<html>
<head>
  <script>
    function showClock() //自定义函数
    {
      d = new Date(); //创建一个时间对象
      document.title = d.toLocaleString(); //标题上显示当前时间
    }
  </script>
</head>

<body>
  <input type="submit" value="开始"
  onclick="window.tid = window.setInterval('showClock()',1000")>
  <input type="submit" value="取消"
         onclick="window.clearTimeout(tid);")>
</body>
</html>
```

location

- 功能：表示窗口中当前显示文档的URL地址，并提供对URL进行解析和操作的属性和方法

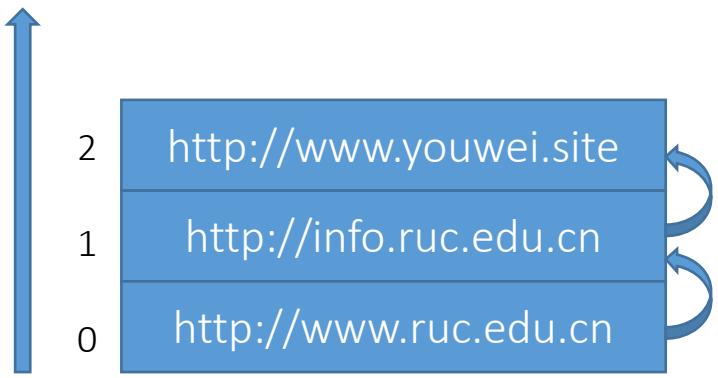
属性和方法	说明
href	获取或设置当前页面的URL,是location的默认属性
host	服务器名字
pathname	URL中主机后面的名字
port	URL请求中的端口号
protocol	URL中使用的协议
search	执行get请求的URL中的？后面部分
reload()	重新载入当前页面

```
//当前窗口URL：https://www.youwei.site:8080/uv/game/rank.php?action=show_rank

alert(location.href); /* https://www.youwei.site:8080/uv/game/rank.php?action=show_rank */
alert(location.host); /* www.youwei.site */
alert(location.pathname); /* /uv/game/rank.php */
alert(location.port); /* 8080 */
alert(location.protocol); /* https */
alert(location.search); /* ?action=show_rank */
location.href = "http://www.ruc.edu.cn"; /* 页面跳转 */
```

history

- 功能：表示用户浏览过的URL历史，并提供对浏览历史URL记录的访问操作

属性和方法	说明
go(i)	加载 history 列表中的某个具体页面，i代表相对位置：-1上一个页面，1前进一个页面
back()	加载 history 列表中的前一个 URL（等价于点击后退按钮或调history.go(-1)）
forward()	加载 history 列表中的下一个 URL（等价于点击前进按钮或 history.go(1)）



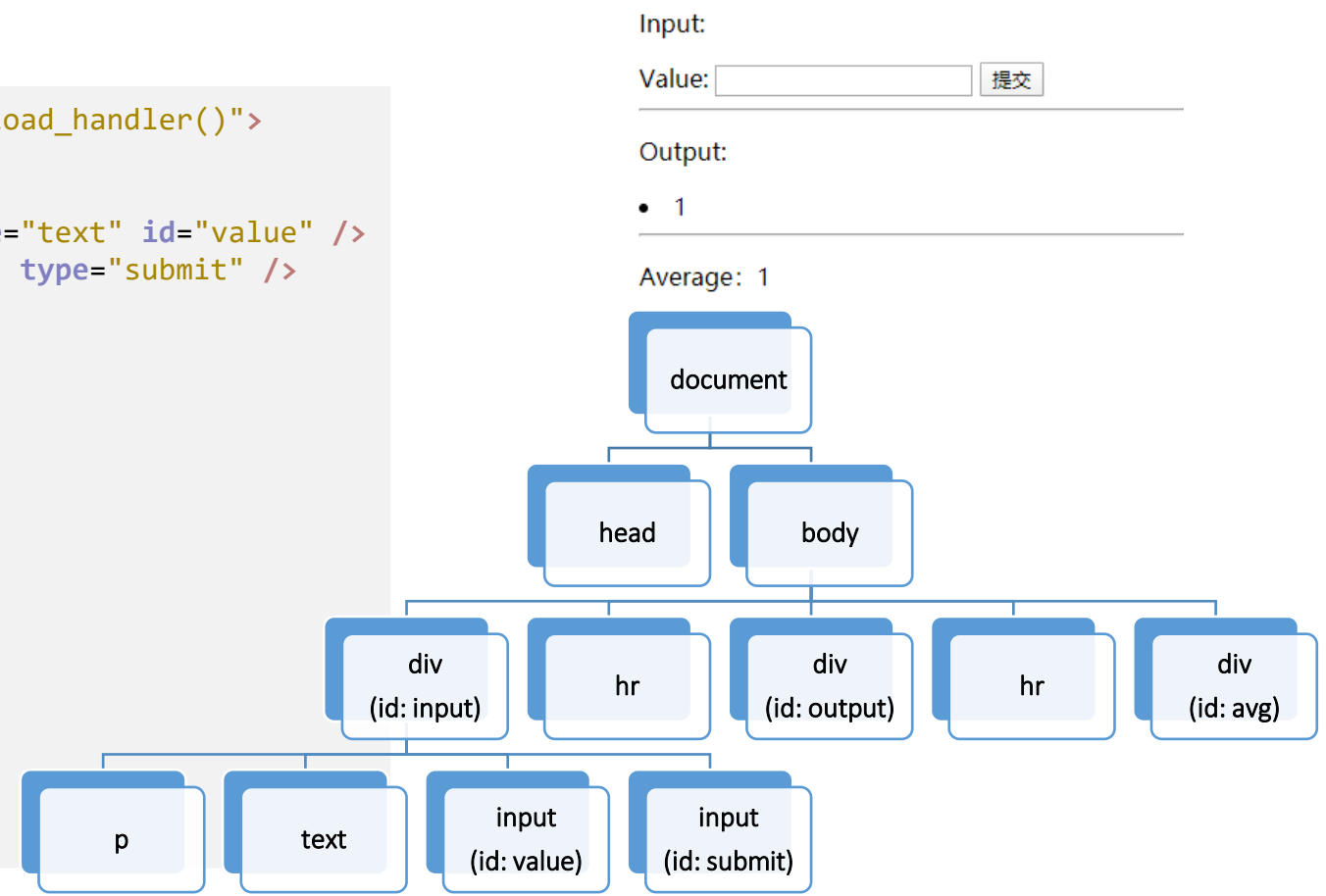
浏览历史栈

```
//当前窗口URL：http://info.ruc.edu.cn

/*
欲访问http://www.ruc.edu.cn：history.back(); 或者 history.go(-1);
欲访问http://www.youwei.site：history.forward(); 或者 history.go(1);
*/
```

2.4.2 DOM

- 将整个HTML文档描述成一棵树，文档中的每个标签对应DOM树的一个节点

```
<body onload="body_onload_handler()">
  <div id="input">
    <p>Input:</p>
    Value: <input type="text" id="value" />
    <input id="submit" type="submit" />
  </div>

  <hr />

  <div id="output">
    <p>Output:</p>
    <li>1</li>
  </div>

  <hr />

  <div id="avg">
    <p>Average: 1</p>
  </div>
</body>
```



学习更多：https://www.w3school.com.cn/js/js_htmldom.asp

DOM操作

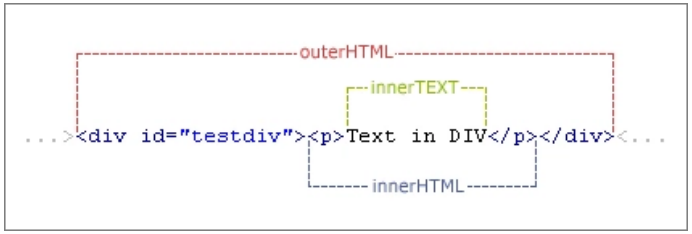
定位输入框：

- document.getElementById("value")
- document.body.childNodes[1].firstChild.nextSibling.nextSibling.nextSibling

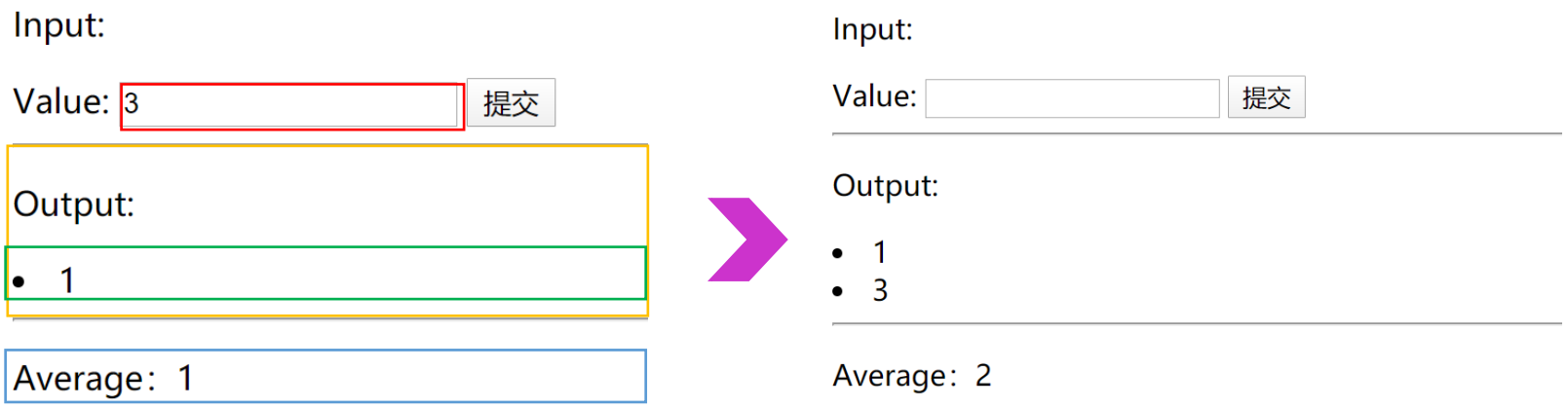
- 获取节点
  - 按id查询：node = document.getElementById(id)
  - 按类名查询：node = document.getElementsByClass(class)
  - 按标签名查询：node = document.getElementsByTagName(tag)
- 节点指针
  - 返回目标节点的所有子节点的列表：node.childNodes
  - 指向在childNodes列表中的第一个节点：node.firstChild
  - 指向在childNodes列表中的最后一个节点：node.lastChild
  - 指向父节点：node.parentNode
  - 指向前一个兄弟节点：node.previousSibling
  - 指向后一个兄弟节点：node.nextSibling
  - 指向这个节点所属的文档：node.ownerDocument

DOM操作

- 节点操作
  - 创建节点：document.createElement(tag)
  - 添加到指定位置：parentNode.appendChild(newNode);
parentNode.insertBefore(newNode, targetNode);
  - 删除节点：parentNode.removeChild(childNode)
  - 替换节点：parentNode.replaceChild(newNode, oldNode)
- 属性操作
  - 获取属性：node.getAttribute(attrName)
  - 设置属性：node.setAttribute(attrName, value)
  - 删除属性：node.removeAttribute(attrName)
- 文本操作
  - innerText：节点内部内容(包括目标元素标签)
  - innerHTML：节点内部内容(不含目标元素标签)
  - outerHTML：含目标元素标签的节点完整内容



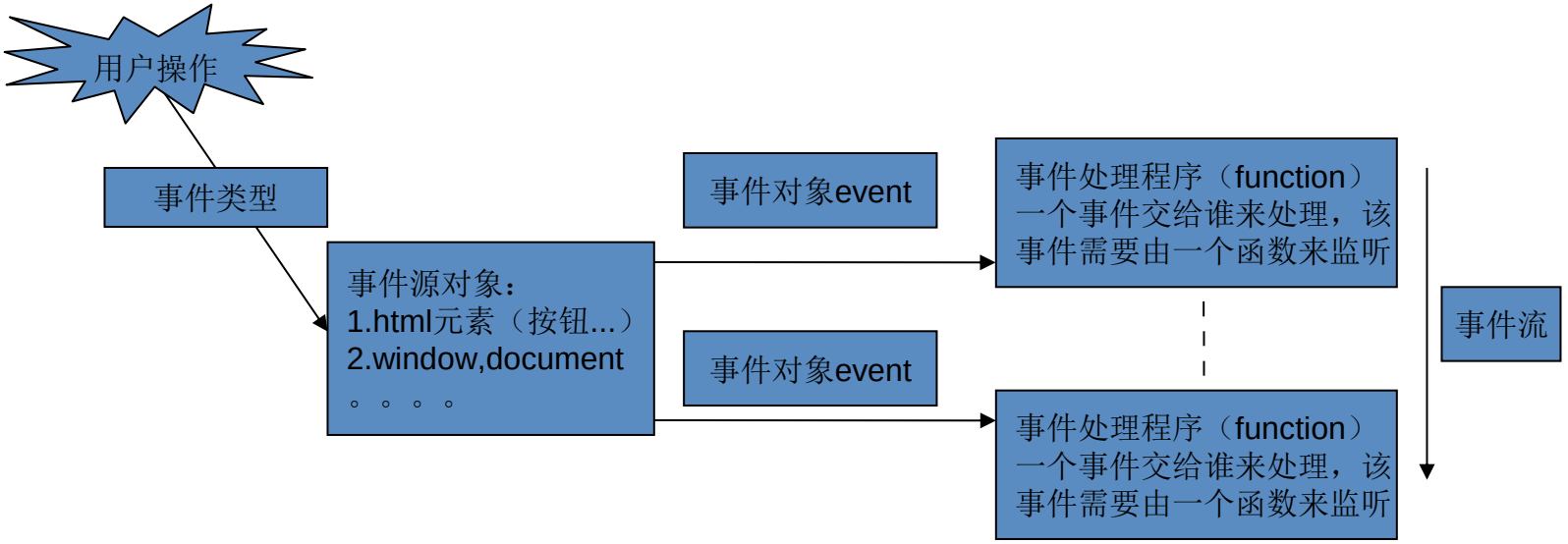
DOM操作示例




```
/* 当提交按钮被点击时触发执行onclick_handler()函数 */
function onclick_handler() {
  value = document.getElementById("value");          //找到value输入框对象
  output = document.getElementsByTagName("div")[1];  //找到output区块对象
  li = document.createElement("li");                 //创建li列表对象
  li.innerHTML = value.value;                        //设置li列表对象的内容为value输入框对象的值
  output.appendChild(li);                            //将新创建的li列表对象添加为output区块对象的儿子节点
  value.value = "";                                  //清空value输入框对象的值
}
/* 当output区块插入DOM节点时触发执onDOMNodeInserted_handler函数 */
function onDOMNodeInserted_handler() {
  avg = document.getElementById("avg");              //找到avg区块对象
  lis = document.getElementsByTagName("li");         //找到所有的li列表对象
  sum = 0;                                           //设置变量sum的初始值为0
  for (i = 0; i < lis.length; i++) {                 //遍历每一个li列表对象
    sum += Number(lis[i].innerText);                 //将li列表对象的内容显示转化为数值内容，累加到sum变量
  }
  avg.innerText = "Average: " + sum/lis.length;      //设置avg区块对象的内容为sum与li列表对象个数的商（平均值）
}
```

2.5 Event

- 事件驱动编程机制：事件是一种异步编程的实现方式，是一种传递信息的机制，为Web程序提供了互动性
- 用户在某个页面元素（事件源对象）上进行某种类型的操作（事件类型）时产生事件对象，浏览器会按照特定顺序（事件流）将事件分派给目标页面元素及其祖先元素上绑定的事件处理程序

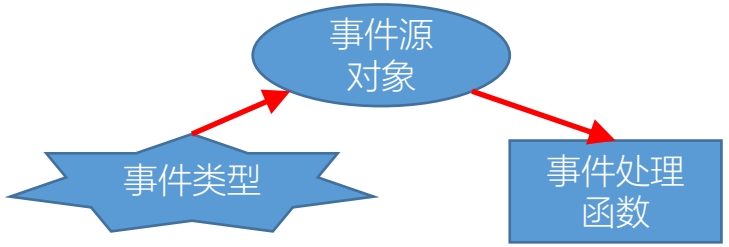


学习更多：https://www.w3school.com.cn/js/js_htmldom_events.asp

2.5.1 基本概念

- 事件：可以被JavaScript侦测到的行为，如用户与页面交互时产生的操作（单击鼠标、按下键盘上的某个键）或页面自身的一些行为（如页面加载）
- 事件类型：不同事件类型对应不同的用户交互行为
- 事件源对象：引发事件的元素（事件发生在谁身上）
- 事件处理程序：对事件进行处理的程序或响应某个事件的函数
- 事件对象：在触发某个事件时，会产生一个事件对象event，其中包含所有与事件有关的信息（包括导致事件的元素、事件类型及其他与特定事件相关的信息）
- 事件流：事件流描述的是从页面中接收事件的顺序。事件流开发者认为，当有页面元素触发事件时，该元素的容器控件以及整个页面都按照特定顺序响应该元素触发的事件，事件发生顺序就是事件流

2.5.2 绑定事件处理函数



- 行内绑定：<标签 属性列表 事件类型=“事件处理函数的调用” >

```
<body onload="body_onload_handler()">
```

- 动态绑定：事件源对象.事件类型 = 事件处理函数的名字

```
submit = document.getElementById("submit");
submit.onclick = submit_onclick_handler;
```

- 事件监听：事件源对象.addEventListener(“事件类型”,事件处理函数的名字)

```
output = document.getElementById("output");
output.addEventListener(
    "DOMNodeInserted",
    output_onDOMNodeInserted_handler,
    false);
```

Input:

Value: 3 提交

Output:

- 1

Average：1

2.5.2 绑定事件处理函数

```
<html>
  <head>
    <script>
      function body_onload_handler(e) {
        submit = document.getElementById("submit");
        submit.onclick = submit_onclick_handler;      //动态绑定
        output = document.getElementById("output");
        output.addEventListener("DOMNodeInserted",    //事件监听
                                output_onDOMNodeInserted_handler, false);
      }
      function button_onclick_handler(e) {
        ……
      }
      function output_onDOMNodeInserted_handler(e) {
        ……
      }
    </script>
  </head>
  <body onload="body_onload_handler()">  <!--行内绑定，浏览器加载事件 -->
    ……
  </body>
<html>
```

2.5.3 Web浏览器常见事件

事件类型	事件名称	事件说明
浏览器窗口事件	load	页面加载完成时触发
	unload	窗口关闭时触发
鼠标事件	click	在元素上单击鼠标左键时触发
	mousedown	在元素上按下鼠标时触发
	mouseup	在元素上释放鼠标时触发
键盘事件	keydown	当键盘按键按下时触发
	keyup	当键盘按键释放时触发
表单事件	focus	当表单元素获得焦点时触发
	submit	当表单提交时触发
剪贴板事件	oncopy	在用户拷贝元素内容时触发
	onpaste	在用户粘贴元素内容时触发
DOM事件	DOMNodeInserted	有DOM元素被添加时触发
	DOMNodeRemoved	有DOM元素被删除时触发
……		

2.5.3 Web浏览器常见事件

```
<html>
  <body>
    <button id="button">点击</button>
    <script type="text/javascript">
      button = document.getElementById("button");
      button.onclick = function(e) { console.log("click"); }          //单击鼠标左键
      button.onmouseup = function(e) { console.log("mouseup"); }      //释放鼠标
      button.onmousedown = function(e) { console.log("mousedown"); }  //按下鼠标
    </script>
  </body>
</html>
```

```
mousedown
mouseup
click
```

```
<html>
  <body>
    <button id="button">点击</button>
    <script type="text/javascript">
      document.onkeydown = function(e) { console.log("keydown: " + e.code); } //键盘按键按下
      document.onkeyup = function(e) { console.log("keyup" + e.code);         //键盘按键释放
      document.oncopy = function(e) { console.log("copy"); }                  //复制行为发生
    </script>
  </body>
</html>
```

```
keydown: ControlLeft
keydown: KeyC
copy
```

2.5.4 事件对象

- 事件对象只有事件发生时才会产生，且只能在事件处理函数内访问，在所有事件处理函数运行结束后，事件对象就被销毁

属性和方法	描述
bubbles	返回布尔值，指示事件是否是起泡事件类型。
cancelable	返回布尔值，指示事件是否可拥可取消的默认动作。
currentTarget	返回其事件监听器触发该事件的元素。
eventPhase	返回事件传播的当前阶段。
target	返回触发此事件的元素（事件的目标节点）。
timeStamp	返回事件生成的日期和时间。
type	返回当前 Event 对象表示的事件的名称。
initEvent()	初始化新创建的 Event 对象的属性。
preventDefault()	通知浏览器不要执行与事件关联的默认动作。
stopPropagation()	不再派发事件。

通用事件对象的常见属性和方法

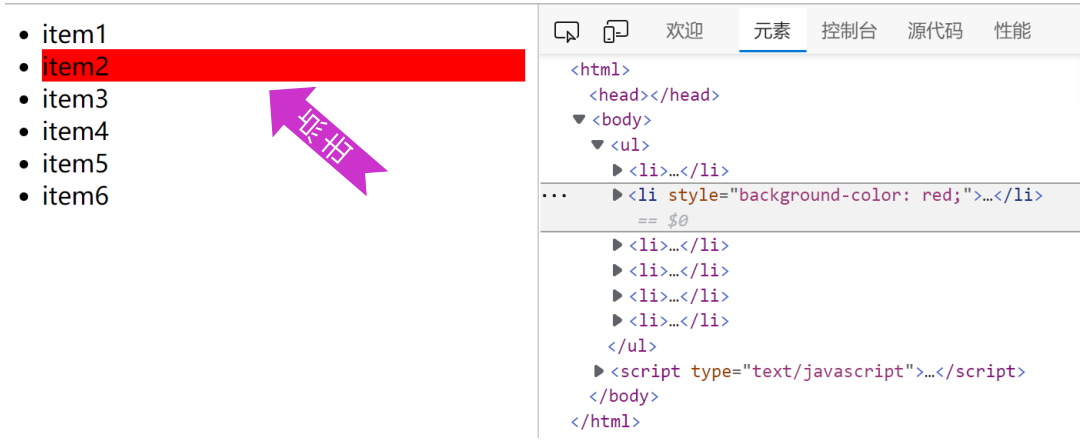
事件类型	属性和方法	描述
键盘事件	keyCode	键盘中对应键位的键值
	charCode	键盘中对应键位的 Unicode 编码
	shiftKey	是否按下 Shift 键
	ctrlKey	是否按下 Ctrl 键
	altKey	是否按下 Alt 键
鼠标事件	button	数值代表按下了鼠标左/中/右键
	clientX	相对于浏览器窗口左上角的水平坐标
	clientY	相对于浏览器窗口左上角的垂直坐标
	offsetX	鼠标位置距离事件源对象左边缘的水平距离
	offsetY	鼠标位置距离事件源对象左边缘的垂直距离
剪贴板事件	clickBoard.getData()	获得剪贴板中的内容
	clickBoard.setData	设置剪贴板中的内容
	clickBoard.clearData()	从剪贴板中删除数据

特定事件对象的常见属性和方法

2.5.4 事件对象

```html
<html>
  <body>
    <ul>
      <li>item1</li>
      <li>item2</li>
      <li>item3</li>
      <li>item4</li>
      <li>item5</li>
      <li>item6</li>
    </ul>

    <script type="text/javascript">
      ul = document.getElementsByTagName("ul")[0];
      ul.onclick = function(e) {
        alert(e.target); //[object HTMLLIElement]
        alert(e.currentTarget); //[object HTMLUListElement]
        e.target.style.backgroundColor = "red";
      }
    </script>
  </body>
</html>
```

```html
<html>
  <body>
    <ul>
      <li>item1</li>
      <li>item2</li>
      <li>item3</li>
      <li>item4</li>
      <li>item5</li>
      <li>item6</li>
    </ul>

    <script type="text/javascript">
      lis = document.getElementsByTagName("li");
      for (li of lis) {
        li.onclick = function() {
          this.style.backgroundColor = "red";
        }
      }
    </script>
  </body>
</html>
```



左边代码：只在<ul>元素上绑定一个事件处理函数

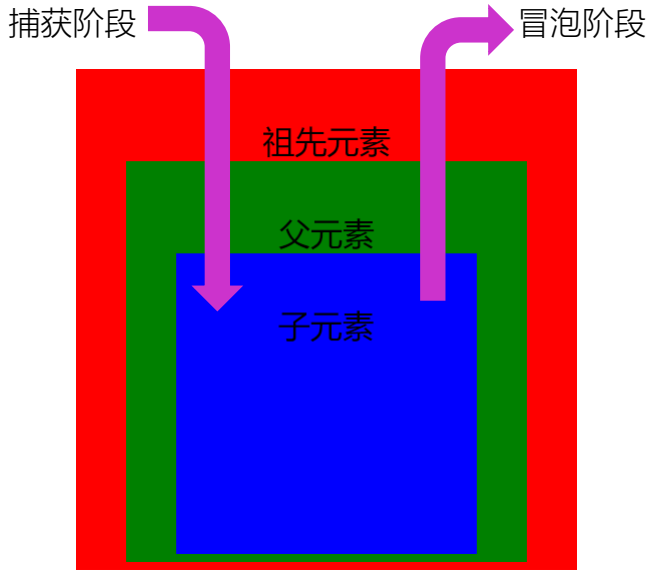
- target是引起触发事件的元素（鼠标点击在item2这个<li>元素上）
- currentTarget是事件绑定的元素（事件处理函数绑定在<ul>上）
- 事件委托：当有多个类似元素（<li>）需要绑定事件时，给其共同父级元素（<ul>）添加一个事件函数，处理所有元素的事件

右边代码：在每一个<li>元素上都绑定一个事件处理函数

- target和currentTarget都是每一个被点击的<li>元素
- 相比于事件委托，对多个类似元素逐一绑定事件处理函数，既浪费时间，又不利于性能

2.5.5 事件流

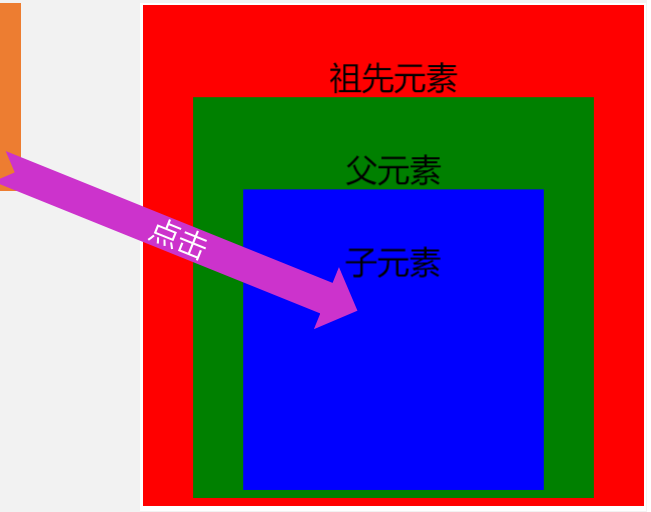
- 当事件发生在某个DOM节点时，事件在DOM结构中进行一级一级的传递，事件流便描述了从页面中接收事件的顺序
- 事件流处理的三个阶段：
  - 捕获过程：事件从Document节点自上而下向目标节点传播的阶段
  - 触发过程：真正的目标节点正在处理事件的阶段
  - 冒泡过程：事件从目标节点自下而上向Document节点传播的阶段



2.5.5 事件流

```
<html>
  <head>
    <style>
      div { padding:25px; border:2px; text-align:center; }
      #ancestor { width:200px; height:200px; background-color:red; }
      #parent { width:150px; height:150px; background-color:green; }
      #child { width:100px; height:100px; background-color:blue; }
    </style>
  </head>
  <body>
    <div id="ancestor">
      <span>祖先元素</span>
      <div id="parent">
        <span>父元素</span>
        <div id="child">
          <span>子元素</span>
        </div>
      </div>
    </div>
    <script type="text/javascript">
      ancestor = document.getElementById("ancestor");
      parent = document.getElementById("parent");
      child = document.getElementById("child");
      ancestor.addEventListener("click", function(e){alert("click-ancestor");}, false);
      parent.addEventListener("click", function(e){alert("click-parent");}, false);
      child.addEventListener("click", function(e){alert("click-child");}, false);
    </script>
  </body>
</html>
```

点击子元素时的输出：
click-child
click-parent
click-ancestor

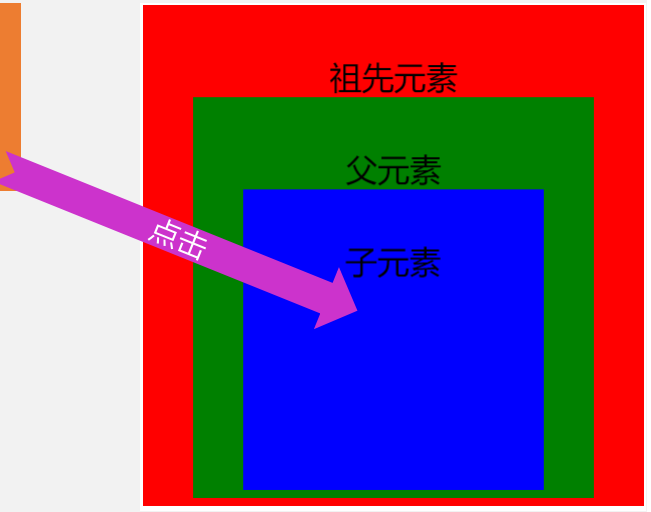


addEventListener的第三个参数指定在哪个阶段调用事件处理函数：
- false: 在冒泡阶段调用事件处理函数，默认值
- true: 在捕获阶段调用事件处理函数

2.5.5 事件流

```
<html>
  <head>
    <style>
      div { padding:25px; border:2px; text-align:center; }
      #ancestor { width:200px; height:200px; background-color:red; }
      #parent { width:150px; height:150px; background-color:green; }
      #child { width:100px; height:100px; background-color:blue; }
    </style>
  </head>
  <body>
    <div id="ancestor">
      <span>祖先元素</span>
      <div id="parent">
        <span>父元素</span>
        <div id="child">
          <span>子元素</span>
        </div>
      </div>
    </div>
    <script type="text/javascript">
      ancestor = document.getElementById("ancestor");
      parent = document.getElementById("parent");
      child = document.getElementById("child");
      ancestor.addEventListener("click", function(e){alert("click-ancestor");}, true);
      parent.addEventListener("click", function(e){alert("click-parent");}, true);
      child.addEventListener("click", function(e){alert("click-child");}, true);
    </script>
  </body>
</html>
```

点击子元素时的输出：
click-ancestor
click-parent
click-child



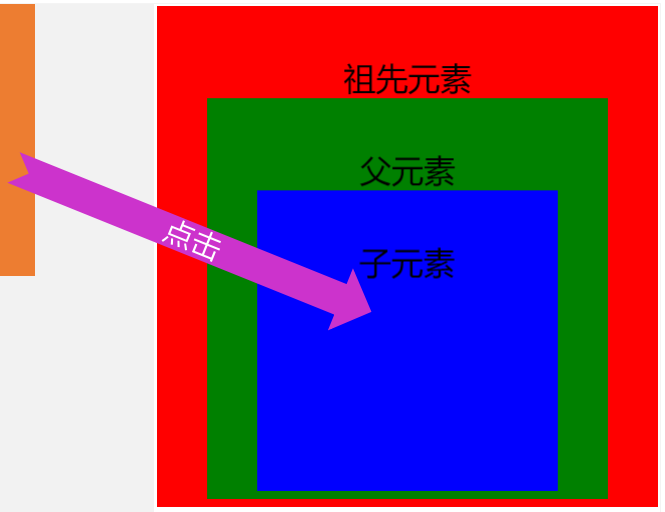
addEventListener的第三个参数指定在哪个阶段调用事件处理函数：
- false: 在冒泡阶段调用事件处理函数，默认值
- true: 在捕获阶段调用事件处理函数

2.5.5 事件流

```
<html>
  <head>
    <style> …… </style>
  </head>
  <body>
    <div id="ancestor">
      <span>祖先元素</span>
      <div id="parent">
        <span>父元素</span>
        <div id="child">
          <span>子元素</span>
        </div>
      </div>
    </div>
    <script type="text/javascript">
      ancestor = document.getElementById("ancestor");
      parent = document.getElementById("parent");
      child = document.getElementById("child");
      ancestor.addEventListener("click", function(e){alert("click-ancestor-cap");}, true);
      parent.addEventListener("click", function(e){alert("click-parent-cap");}, true);
      child.addEventListener("click", function(e){alert("click-child-cap");}, true);
      ancestor.addEventListener("click", function(e){alert("click-ancestor-bub");}, false);
      parent.addEventListener("click", function(e){alert("click-parent-bub");
                                                   e.stopPropagation(); }, false);
      child.addEventListener("click", function(e){alert("click-child-bub");}, false);
    </script>
  </body>
</html>
```

点击子元素时的输出：

click-ancestor-cap
click-parent-cap
click-child-cap
click-child-bub
click-parent-bub

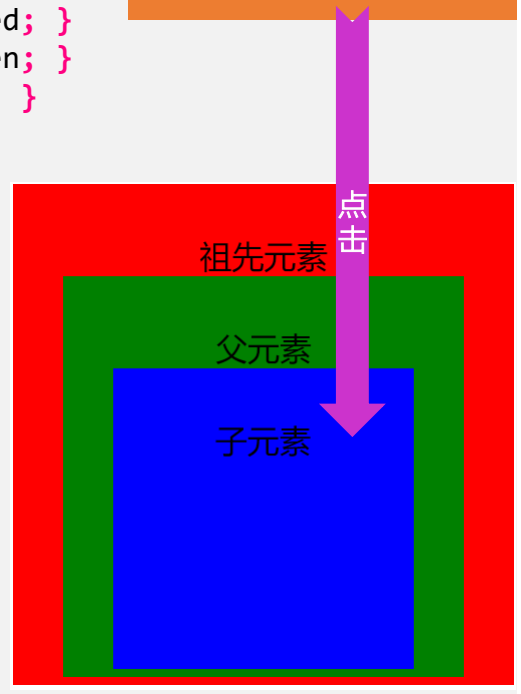


同一个对象可以在不同阶段注册不同的事件处理函数
使用stopPropagatoin可以终止事件的传播

2.5.5 事件流

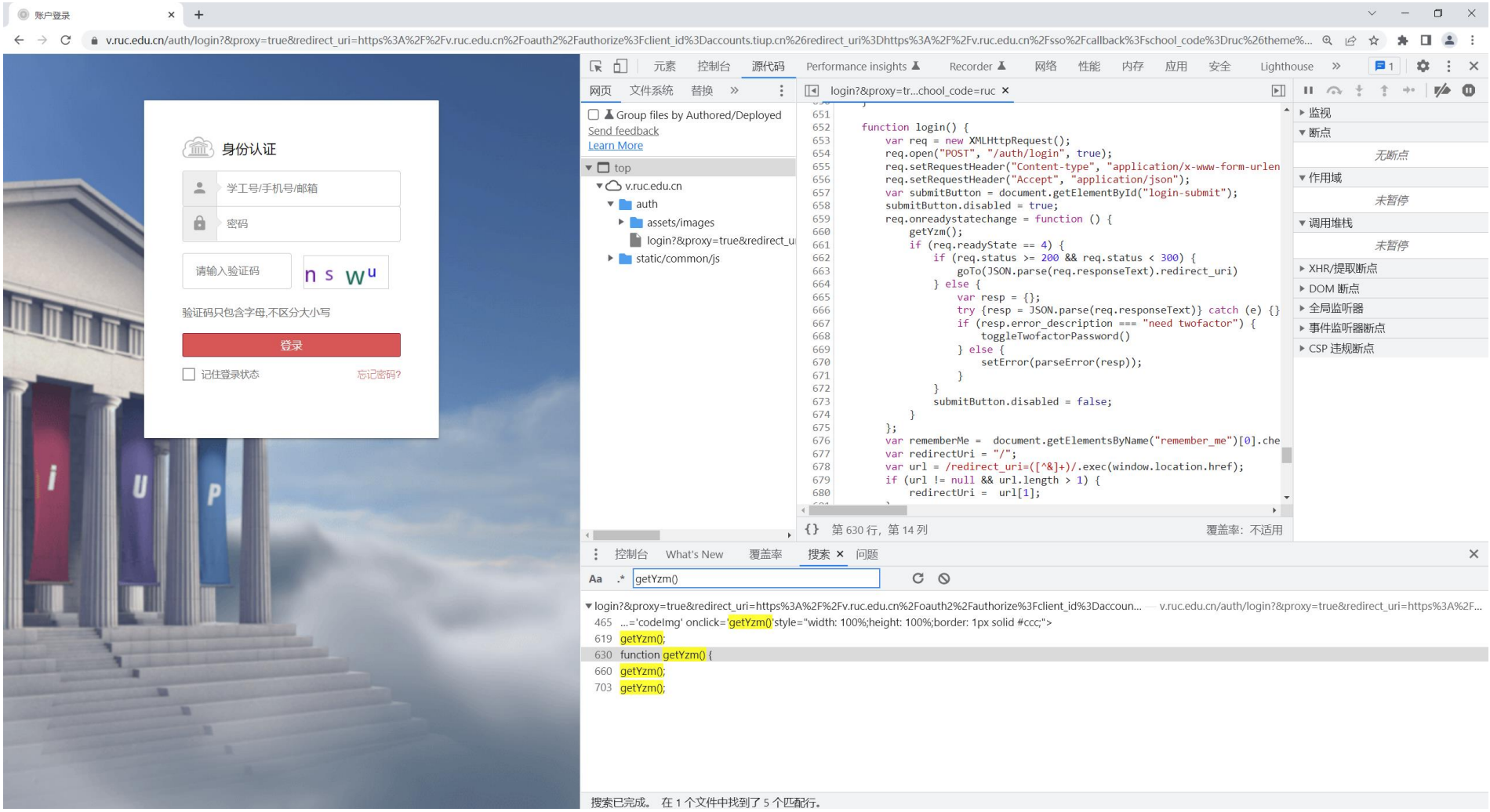
```
<html>
  <head>
    <style>
      div { padding:25px; border:2px; text-align:center; }
      #ancestor { width:200px; height:200px; background-color:red; }
      #parent { width:150px; height:150px; background-color:green; }
      #child { width:100px; height:100px; background-color:blue; }
    </style>
  </head>
  <body>
    <div id="ancestor"  onclick="alert('click-ancestor1')">
      <span>祖先元素</span>
      <div id="parent"  onclick="alert('click-parent1')">
        <span>父元素</span>
        <div id="child" onclick="alert('click-child1')">
          <span>子元素</span>
        </div>
      </div>
    </div>
    <script type="text/javascript">
      ancestor = document.getElementById("ancestor");
      parent = document.getElementById("parent");
      child = document.getElementById("child");
      ancestor.onclick = function(e){ alert("click-ancestor2");};
      parent.onclick = function(e){alert("click-parent2");};
      child.onclick = function(e){alert("click-child2");};
    </script>
  </body>
</html>
```

点击子元素时的输出：
click-child2
click-parent2
click-ancestor2



以行内绑定和动态绑定方式注册的事件处理函数，将在冒泡阶段被调用
同一个对象在同一个阶段注册多个事件处理函数，以最后一次注册为准

实例：为微人大认证页面添加前端交互



实例：为微人大认证页面添加前端交互

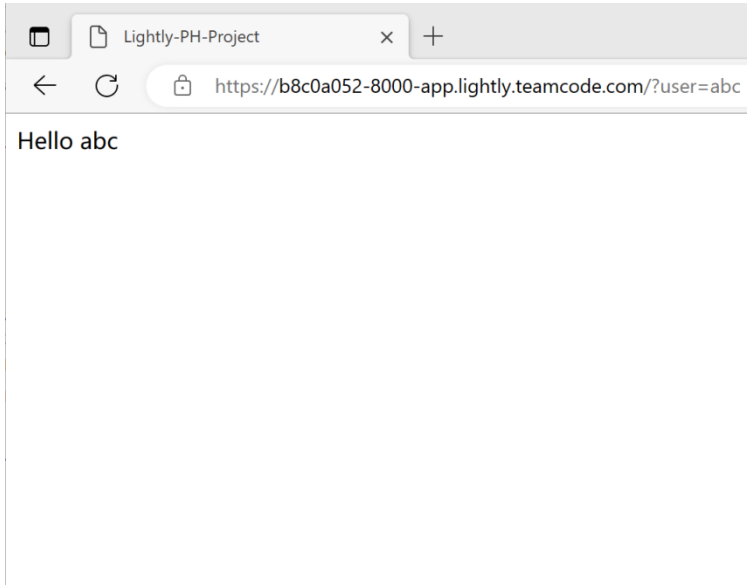
```
/* script.js */

function body_onload_handler() { //页面加载时触发
  button = document.getElementById("login-submit");
  button.onclick = login; //注册登录按钮的鼠标点击事件处理函数

  img = document.getElementById("captcha-img").childNodes[1];
  img.onclick = getYzm; //注册验证码图片的鼠标点击事件处理函数
  img.click();
}

function getYzm(e) { //随机生成一个0-9的整数x，使用图片code_x.png作为验证码图片
  x = Math.floor(Math.random()*10);
  img = e.target;
  img.src = "img/code_" + x + ".png";
}

function login() { //登录时检验验证码/用户名/密码
  img = document.getElementById("captcha-img").childNodes[1];
  form = document.getElementById("login-form");
  info = document.getElementById("login-alert");

  //维护一个验证码正确值和验证码图片下标的映射
  checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
  correct_code = checks[img.src.charAt(img.src.length-5)];

  username = form.username.value;
  password = form.password.value;
  code = form.code.value;

  if (code != correct_code) { info.innerText = "验证码不正确或已失效,请重试！"; }
  else if (username != "abcd" || password != "1234") { info.innerText = "用户名或密码不正确,请重试！"; }
  else { info.innerText = "验证成功！"; }
}
```

目录

3.1 PHP

- PHP是PHP：Hypertext Preprocessor（超文本预处理器）的缩写，是一种服务器端执行的脚本语言，用于生成动态页面内容

```
<html>
  <head>
    <title>第一个PHP代码</title>
  </head>

  <body>
    <span>Hello </span>
    <?php
      $user = $_GET["user"];
      echo $user;
    ?>
  </body>
</html>
```



学习更多：https://www.w3school.com.cn/php/index.asp

3.1.1 在网页中引入PHP代码

- XML标记风格（推荐）

```
<?php
  echo "this is xml style";
?>
```

- 简短标记风格

```
<?
  echo "this is xml style";
?>
```

- 脚本标记风格

```
<script language="php">
  echo "this is xml style";
</script>
```

- ASP标记风格

```
<%
  echo "this is xml style";
%>
```

3.1.3 常量

- 预定义常量

常 量 名	功 能
__FILE__	默认常量，PHP程序文件名
__LINE__	默认常量，PHP程序行数
PHP_VERSION	内建常量，PHP程序的版本，如“3.0.8_dev”
PHP_OS	内建常量，执行PHP解析器的操作系统名称，如“Windows”
TRUE	这个常量是一个真值（True）
FALSE	这个常量是一个假值（False）
NULL	一个null值
E_ERROR	这个常量指到最近的错误处
E_WARNING	这个常量指到最近的警告处
E_PARSE	这个常量指解析语法有潜在问题处
E_NOTICE	这个常量为发生不寻常，但不一定是错误处

```php
<?php
  echo "本文件路径和文件名为：".__FILE__."<br/>";
  echo "当前行数为：".__LINE__."<br/>";
  echo "当前的操作系统为：".PHP_OS."<br/>";
  echo "当前的PHP版本为：".PHP_VERSION."<br/>";
?>
```

```
本文件路径和文件名为：/workspace/PHPProject/index.php
当前行数为：13
当前的操作系统为：Linux
当前的PHP版本为：8.1.9
```

```php
<?php
  define("DEFAULT_PATH", "/var/www/");
  define("NORMAL_USER", "0");
  echo DEFAULT_PATH."<br/>";
  echo constant("NORMAL_USER")."<br/>";
  echo defined("DEFAULT_PATH")."<br/>";
  echo defined("HELLO_WORLD")."<br/>";
?>
```

```
/var/www/
0
1

END
```

- 自定义常量
  - 使用define()函数声明常量
  - 使用constant()函数或常量名获取常量的值
  - 使用defined()函数判断常量是否已经被定义

3.1.4 变量

- 变量采用美元符号（$）加一个变量名的方式来表示
- 变量在使用前不需要预先定义，可以直接在使用时定义变量
- 变量赋值：给变量一个具体的数据值
  - 传值赋值：等号两边的变量值互不影响，任何一个变量值的改变都不会影响另一个变量
  - 引用赋值：等号两边的变量指向内存中同一存储空间，任何一个变量值的改变都会引起另一个变量的变化

```php
<?php
  echo "使用传值方式赋值";
  $a = "hello";
  $b = $a;                          //将变量$a的值赋值给变量$b
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
  $a = "hello world";               //改变变量$a的值
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
?>
```

```
使用传值方式赋值变量a的值为hello
变量b的值为hello
变量a的值为hello world
变量b的值为hello
```

```php
<?php
  echo "使用传值方式赋值";
  $a = "world";
  $b = &$a;                         //将变量$a的引用赋值给变量$b
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
  $a = "hello world";               //改变变量$a的值
  echo "变量a的值为".$a."<br/>";
  echo "变量b的值为".$b."<br/>";
?>
```

```
使用传值方式赋值变量a的值为world
变量b的值为world
变量a的值为hello world
变量b的值为hello world
```

3.1.4 变量

- 变量检测与销毁
  - 使用isset()函数判断给定变量是否存在
  - 使用unset()函数销毁给定变量
- 可变变量：变量名称不是预先定义好的，而是动态地设置和使用
  - 指使用一个变量的值作为另一个变量的名称，又称为变量的变量
  - 可变变量通过在一个变量名称前使用两个美元符号（$$）符号实现

```php
<?php
  $a = 123;
  if (isset($a)) echo "\$a exists<br/>";

  $a = null;
  if (!isset($a)) echo "\$a not exist<br/>";
  echo "value of \$a: $a<br/>";

  unset($a);
  if (!isset($a)) echo "\$a not exist<br/>";
  echo "value of \$a: $a<br/>";
?>
```

```
$a exists
$a not exist
value of $a:
$a not exist

Warning: Undefined variable $a in /workspace/PHPProject/index.php on line 21
value of $a:
```

```php
<?php
  $a = "hello"; // 声明变量 $a
  $$a = "world"; // 声明变量 $hello
  echo $a."<br/>";
  echo $hello."<br/>";
  echo $$a."<br/>";
?>
```

```
hello
world
world
```

3.1.5 数据类型

基本数据类型

类　型	说　明	检测函数
boolean（布尔型）	这是最简单的类型。只有两个值，真值（true）和假值（false）	is_bool
string（字符串型）	字符串就是连续的字符序列，可以是计算机能表示的一切字符的集合	is_string
integer（整型）	整型数据类型只能包含整数。可以是正整数或负整数	is_integer/is_int
float/double（浮点型）	浮点数据类型wygd用来存储数字，和整型不同的是它有小数位	is_float/is_double

is_numeric

复合数据类型

类　型	说　明	检测函数
array（数组）	将多个相互关联的数据组织在一起形成一个整体，作为一个单元使用	is_array
object（对象）		is_object

特殊数据类型

类　型	说　明	检测函数
resource（资源）	资源是一种特殊的数据类型，保存着对外部资源的引用，如打开的文件、数据库连接等	--
null（空值）	空值表示没有为变量设置任何值	is_null

3.1.6 类型转换

- PHP是弱类型脚本语言，无需声明变量类型，运行时根据需要，自动确定变量类型并进行数据类型转换
- 类型转换分为：隐式类型转换和显式类型转换
  - 隐式类型转换：对不同类型值使用运算符时，值可在类型之间自动转换
  - 显式类型转换：开发人员在编写代码时，强制指定对值的类型进行转换

3.1.6 类型转换

Converting to boolean

When converting to bool, the following values are considered **false**:

- the boolean **false** itself
- the integer 0 (zero)
- the floats 0.0 and -0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements
- the special type NULL (including unset variables)
- SimpleXML objects created from attributeless empty elements, i.e. elements which have neither children nor attributes.

Every other value is considered **true** (including any resource and **NAN**).

Converting to string

A bool **true** value is converted to the string "1". bool **false** is converted to "" (the empty string). This allows conversion back and forth between bool and string values.

An int or float is converted to a string representing the number textually (including the exponent part for floats). Floating point numbers can be converted using exponential notation (4.1E+6).

Arrays are always converted to the string "Array"; because of this, echo and print can not by themselves show the contents of an array. To view a single element, use a construction such as `echo $arr['foo']`. See below for tips on viewing the entire contents.

In order to convert objects to string, the magic method __toString must be used.

Resources are always converted to strings with the structure "Resource id #1", where 1 is the resource number assigned to the resource by PHP at runtime. While the exact structure of this string should not be relied on and is subject to change, it will always be unique for a given resource within the lifetime of a script being executed (ie a Web request or CLI process) and won't be reused. To get a resource's type, use the get_resource_type() function.

null is always converted to an empty string.

As stated above, directly converting an array, object, or resource to a string does not provide any useful information about the value beyond its type. See the functions print_r() and var_dump() for more effective means of inspecting the contents of these types.

Most PHP values can also be converted to strings for permanent storage. This method is called serialization, and is performed by the serialize() function.

Converting to integer

From booleans

false will yield 0 (zero), and **true** will yield 1 (one).

From floating point numbers

When converting from float to int, the number will be rounded *towards zero*. As of PHP 8.1.0, a deprecation notice is emitted when implicitly converting a non-integral float to int which loses precision.

If the float is beyond the boundaries of int (usually +/- 2.15e+9 = 2^31 on 32-bit platforms and +/- 9.22e+18 = 2^63 on 64-bit platforms), the result is undefined, since the float doesn't have enough precision to give an exact int result. No warning, not even a notice will be issued when this happens!

From strings

If the string is numeric or leading numeric then it will resolve to the corresponding integer value, otherwise it is converted to zero (0).

From NULL

null is always converted to zero (0).

From other types

The behaviour of converting to int is undefined for other types. Do *not* rely on any observed behaviour, as it can change without notice.

学习更多：
https://www.php.net/manual/en/language.types.type-juggling.php

3.1.7 字符串

- 字符串的定义
  - 使用单引号定义字符串：单引号串中的内容被认为是普通字符直接输出
  - 使用双引号定义字符串：双引号串中的内容可以被解释而且替换
  - 使用定界符定义字符串：原样输出定界符中的内容，变量用其值来替换

```php
<?php
  $ruc = "中国人民大学";
  $str1 = "I Like \"$ruc\"<br/>";
  $str2 = 'I Like \"$ruc\"<br/>';

  $html = <<<str
    <font color="#AE0B2A"> $ruc 是中国共产党创办的第一所新型正规大学，<br/>
                           是一所以人文社会科学为主的综合性研究型全国重点大学，<br/>
                           直属于教育部，由教育部与北京市共建。</font>
  str;

  echo $str1;
  echo $str2;
  echo $html;
?>
```

I Like "中国人民大学"
I Like \"$ruc\"
中国人民大学 是中国共产党创办的第一所新型正规大学，
是一所以人文社会科学为主的综合性研究型全国重点大学，
直属于教育部，由教育部与北京市共建。

字符串常用操作

数组与字符串的转换

Syntax

```
explode(separator,string,limit)
```

Parameter Values

Parameter	Description
separator	Required. Specifies where to break the string
string	Required. The string to split
limit	Optional. Specifies the number of array elements to return. Possible values: • Greater than 0 - Returns an array with a maximum of *limit* element(s) • Less than 0 - Returns an array except for the last -*limit* elements() • 0 - Returns an array with one element

Syntax

```
implode(separator,array)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies what to put between the array elements. Default is "" (an empty string)
array	Required. The array to join to a string

学习更多：
https://www.w3schools.com/php/php_ref_string.asp

子串操作

Syntax

```
substr(string,start,length)
```

Parameter Values

Parameter	Description
string	Required. Specifies the string to return a part of
start	Required. Specifies where to start in the string • A positive number - Start at a specified position in the string • A negative number - Start at a specified position from the end of the string • 0 - Start at the first character in string
length	Optional. Specifies the length of the returned string. Default is to the end of the string. • A positive number - The length to be returned from the start parameter • Negative number - The length to be returned from the end of the string • If the *length* parameter is 0, NULL, or FALSE - it return an empty string

```php
<?php
  $str = "Hello world. It's a beautiful day.";
  $arr = explode(" ", $str);
  print_r($arr);
?>
```

Array ([0] => Hello [1] => world. [2] => It's [3] => a [4] => beautiful [5] => day.)

```php
<?php
  $arr = array('Hello','World!','Beautiful','Day!');
  $str = implode(" ", $arr);
  echo $str;
?>
```

Hello World! Beautiful Day!

```php
<?php
  $str = "Hello World!";
  $substr = substr($str, 6, 5);
  echo $substr;
?>
```

World

3.1.8 数组

- 数组的类型
  - 索引数组：键名（下标）由数字组成，默认从0开始
  - 关联数组：键名是字符串，或者数字和字符串混合的形式
- 创建数组
  - 通过数组标识符“[]”创建数组
  - 使用array()函数创建数组

```php
<?php
  $arr1 = array("中国人民大学", "北京大学", "清华大学");
  $arr2[1]="中国人民大学"; $arr2[2]="北京大学"; $arr2[3]="清华大学";

  echo "打印数组键和值如下 :<br/>";
  var_dump($arr1); echo "<br/>";

  echo "打印单独某一数组元素 :<br/>";
  echo "$arr2[0]:".$arr2[0]."<br/>"; // 打印数组$arr2第0个元素
  echo "$arr2[1]:".$arr2[1]."<br/>"; // 打印数组$arr2第1个元素
  echo "$arr2[2]:".$arr2[2]."<br/>"; // 打印数组$arr2第2个元素
  echo "$arr2[3]:".$arr2[3]."<br/>"; // 打印数组$arr2第3个元素
?>
```

```
打印数组键和值如下 :
array(3) { [0]=> string(18) "中国人民大学" [1]=> string(12) "北京大学" [2]=> string(12) "清华大学" }
打印单独某一数组元素 :

Warning: Undefined array key 0 in /workspace/PHPProject/index.php on line 21
$arr2[0] :
$arr2[1] :中国人民大学
$arr2[2] :北京大学
$arr2[3] :清华大学
```

```php
<?php
  $arr3 = array("ruc"=>"中国人民大学","pku"=>"北京大学","thu"=>"清华大学");
  $arr4 = ["ruc"=>"中国人民大学", "pku"=>"北京大学", "thu"=>"清华大学"];

  echo "打印数组键和值如下 :<br/>";
  var_dump($arr3); echo "<br/>";

  echo "打印单独某一数组元素 :<br/>";
  echo '$arr4["ruc"]:'.$arr4["ruc"]."<br/>"; // 打印键值为“ruc”的元素
  echo '$arr4["pku"]:'.$arr4["pku"]."<br/>"; // 打印键值为“pku”的元素
  echo '$arr4["thu"]:'.$arr4["thu"]."<br/>"; // 打印键值为“thu”的元素
?>
```

```
打印数组键和值如下 :
array(3) { ["ruc"]=> string(18) "中国人民大学" ["pku"]=> string(12) "北京大学" ["thu"]=> string(12) "清华大学" }
打印单独某一数组元素 :
$arr4["ruc"] :中国人民大学
$arr4["pku"] :北京大学
$arr4["thu"] :清华大学
```

3.1.8 数组

- 遍历数组
  - 使用list()函数遍历数组

```php
<?php
  $arr = array("中国人民大学", "北京大学", "清华大学");
  list($ruc, $pku, $thu) = $arr;        // 将数组$arr中三个元素分别赋值给$ruc, $pku, $thu
  echo "ruc: ".$ruc."<br/>";
  echo "pku: ".$pku."<br/>";
  echo "thu: ".$thu."<br/>";
?>
```

```
ruc: 中国人民大学
pku: 北京大学
thu: 清华大学
```

  - 使用foreach循环遍历数组

```php
<?php
  $arr1 = array("中国人民大学", "北京大学", "清华大学");
  $arr3 = array("ruc"=>"中国人民大学","pku"=>"北京大学","thu"=>"清华大学");

  foreach($arr1 as $value) {              // 遍历索引数组
    echo $value."<br/>";
  }

  foreach($arr3 as $key=>$value) {        // 遍历关联数组
    echo $key.":".$value."<br/>";
  }
?>
```

```
中国人民大学
北京大学
清华大学

ruc:中国人民大学
pku:北京大学
thu:清华大学
```

3.1.8 数组

- 多维数组：包含一个或多个数组的数组
  - 对于n维数组，需要n个索引来选取元素
  - 可以用n重循环遍历n维数组的每一个元素

```php
<?php
  $sites = array
  (
    "ruc" => array("中国人民大学", "https://www.ruc.edu.cn"),
    "pku" => array("北京大学", "https://www.pku.edu.cn"),
    "thu" => array("清华大学", "https://www.tsinghua.edu.cn")
  );

  print("<pre>");
  print_r($sites);
  print("</pre>");

  echo $sites["ruc"][0].": ".$sites["ruc"][1]."<br/>";

  foreach($sites as $key=>$values) {
    echo $key.": ";
    foreach($values as $value) {
      echo $value." ";
    }
    echo "<br/>";
  }
?>
```

```
Array
(
    [ruc] => Array
        (
            [0] => 中国人民大学
            [1] => https://www.ruc.edu.cn
        )

    [pku] => Array
        (
            [0] => 北京大学
            [1] => https://www.pku.edu.cn
        )

    [thu] => Array
        (
            [0] => 清华大学
            [1] => https://www.tsinghua.edu.cn
        )

)
```

中国人民大学: https://www.ruc.edu.cn
ruc: 中国人民大学 https://www.ruc.edu.cn
pku: 北京大学 https://www.pku.edu.cn
thu: 清华大学 https://www.tsinghua.edu.cn

数组常用操作

统计数组元素个数

Syntax

```
count(array, mode)
```

Parameter Values

Parameter	Description
array	Required. Specifies the array
mode	Optional. Specifies the mode. Possible values: • 0 - Default. Does not count all elements of multidimensional arrays • 1 - Counts the array recursively (counts all the elements of multidimensional arrays)

```php
<?php
  $sites = array
  (
    "ruc" => array("中国人民大学", "https://www.ruc.edu.cn"),
    "pku" => array("北京大学", "https://www.pku.edu.cn"),
    "thu" => array("清华大学", "https://www.tsinghua.edu.cn")
  );

  echo "一维数组\$sites[\"ruc\"]元素个数: ".count($sites["ruc"])."<br/>";
  echo "二维数组\$sites中的一维数组个数: ".count($sites)."<br/>";
  echo "二维数组\$sites递归所有元素个数: ".count($sites, 1)."<br/>";
?>
```

```
一维数组$sites["ruc"]元素个数: 2
二维数组$sites中的一维数组个数: 3
二维数组$sites递归所有元素个数: 9
```

检查数组中是否存在某个值

Syntax

```
in_array(search, array, type)
```

Parameter Values

Parameter	Description
search	Required. Specifies the what to search for
array	Required. Specifies the array to search
type	Optional. If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

```php
<?php
  $postcodes = array
  (
    "ruc"=>"100872",
    "pku"=>100871,
    "thu"=> array("100084")
  );

  if (!in_array("pku", $postcodes)) echo "pku does not exist <br/>";
  if (in_array(100872, $postcodes)) echo "100872 exists (loose) <br/>";
  if (!in_array(100872, $postcodes, true))
    echo "100872 does not exist (strict) <br/>";
  if (in_array([100084], $postcodes)) echo "[100084] exists <br/>";
?>
```

```
pku does not exist
100872 exists (loose)
100872 does not exist (strict)
[100084] exists
```

学习更多:
https://www.w3schools.com/php/php_ref_array.asp

3.1.9 函数

- 参数传递方式
  - 按值传递
  - 按引用传递
  - 默认参数值
  - 变长参数
- 返回值传递方式
  - 按值传递
  - 按引用传递

/* 参数按值传递 */

```php
<?php
  function example($m) {
    $m = $m*5+10;
    echo "<p>在函数内：\$m = $m</p>";
  }

  $m = 1;
  example($m) ;
  echo "<p>在函数外：\$m = $m</p>" ;
?>
```

在函数内：$m = 15

在函数外：$m = 1

/* 参数按引用传递 */

```php
<?php
  function example(&$m) {
    $m = $m*5+10;
    echo "<p>在函数内：\$m = $m</p>";
  }

  $m = 1;
  example($m) ;
  echo "<p>在函数外：\$m = $m</p>" ;
?>
```

在函数内：$m = 15

在函数外：$m = 15

/* 默认参数值 */

```php
<?php
  function values($price, $tax=0) {
    $price = $price+($price*$tax);
    echo "<p>价格：$price</p>";
  }

  values(100, 0.25);
  values(100);
?>
```

价格：125

价格：100

/* 变长参数 */

```php
<?php
  function sum(){
    $total = 0;
    $vars = func_get_args();//参数数组
    foreach($vars as $var)
      $total += $var;
    return $total;
  }

  echo sum(1, 2)."<br/>";
  echo sum(1, 2, 3)."<br/>";
?>
```

3
6

/* 返回值传递方式 */

```php
<?php
  function &test() {
    static $b = 0;
    $b++;
    return $b;
  }
?>
```

```php
<?php
  $a = test();
  echo "$a<br/>";
  $a = 5;
  $a = test();
  echo "$a<br/>";
?>
```

1
2

```php
<?php
  $a = &test();
  echo "$a<br/>";
  $a = 15;
  $a = test();
  echo "$a<br/>";
?>
```

1
16

3.1.9 函数

- 变量作用域与生命周期
  - 局部变量：在函数内部定义的变量，作用域和生命周期是其所在函数
  - 全局变量：定义在所有函数以外的变量，作用域和生命周期是整个PHP文件，需使用global关键字声明，才可在用户自定义函数内部使用全局变量
  - 静态变量：作用域是其所在函数，生命周期是整个PHP文件

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    echo "函数内部输出变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

Warning: Undefined variable $str in **/workspace/PHPProject/index.php** on line **15**
函数内部输出变量值：
在函数外部输出变量值：函数外部定义的变量

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    $str = "...函数内部定义的变量...";
    echo "函数内部输出变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

函数内部输出变量值：...在函数内部定义的变量...
在函数外部输出变量值：函数外部定义的变量

```php
<?php
  $str = "函数外部定义的变量";
  function fun() {
    global $str;
    echo "函数内部输出变量值：".$str."<br/>";
    $str = "...在函数内部改变变量的值 ...";
    echo "改变后的变量值：".$str."<br/>";
  }
  fun();
  echo "在函数外部输出变量值：".$str."<br/>";
?>
```

函数内部输出变量值：函数外部定义的变量
改变后的变量值：...在函数内部改变变量的值 ...
在函数外部输出变量值：...在函数内部改变变量的值 ...

```php
<?php
  function fun1() {
    static $num = 0;  //初始化静态变量
    $num += 1;
    echo $num."  ";
  }
?>
```

```php
<?php
  function fun2() {
    $num = 0;  //函数内部局部变量
    $num += 1;
    echo $num."  ";
  }
?>
```

```php
<?php
  echo "fun1()中num的值：";
  for ($i = 0; $i < 9; $i++) fun1();
  echo "fun2()中num的值：";
  for ($i = 0; $i < 9; $i++) fun2();
?>
```

fun1()中num的值：1 2 3 4 5 6 7 8 9 fun2()中num的值：1 1 1 1 1 1 1 1 1

3.1.9 函数

- 文件的引用
  - include：使用include引用外部文件时，只有代码执行到include语句时，调用的外部文件才会被引用并读取；当引用的文件发生错误时，系统只给出个警告错误，而整个php文件会继续执行
  - require：在php文件被执行之前，php解析器会用被引用的文件的全部内容替换require语句形成新的php文件，最后按新的php文件执行程序；如果调用的文件没有找到，require语句会输出错误信息，立即终止执行
  - include_once/require_once：与include/require相同，但能确保一个被包含的文件只能被包含一次

```php
<?php
  include "nonexist.php";
  echo "continue execution";
?>
```

Warning: include(nonexist.php): Failed to open stream: No such file or directory in **/workspace/PHPProject/index.php** on line **12**

Warning: include(): Failed opening 'nonexist.php' for inclusion (include_path='.:/usr/local/lib/php') in **/workspace/PHPProject/index.php** on line **12**
continue execution

```php
<?php
  require "nonexist.php";
  echo "continue execution";
?>
```

Warning: require(nonexist.php): Failed to open stream: No such file or directory in **/workspace/PHPProject/index.php** on line **12**

Fatal error: Uncaught Error: Failed opening required 'nonexist.php' (include_path='.:/usr/local/lib/php') in /workspace/PHPProject/index.php:12 Stack trace: #0 {main} thrown in **/workspace/PHPProject/index.php** on line **12**

3.2 与Web页面交互

- 提交表单
- 上传文件
- 会话控制
- 其它信息
- 数据库操作

请求网址: https://www.youwei.site/training/form/submit.php?name=user&password=abcd&sex=male&education=graduate&hobby=sports&hobby=music&intro=I+am+me.&photo=head.jpg

请求方法: GET

3.2.1 提交表单

- GET方法
  - 请求数据以地址的形式表现在请求行，对传输数据的大小有限制
  - 从全局数组变量$_GET[]获得通过GET方法传递的数据

```
<form action="submit.php" method="GET" enctype="multipart/form-data">
  姓名: <input type="text" name="name"/> <br/>
  密码: <input type="password" name="password"/> <br/>
  性别:
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历:
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣:
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介: <textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片: <input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  /* 逐个定义变量 */
  $name = $_GET["name"];
  $password = $_GET["password"];
  $sex = $_GET["sex"];
  $education = $_GET["education"];
  $hobby = $_GET["hobby"];
  $intro = $_GET["intro"];
  $photo = $_GET["photo"];

/* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
?>
```

请求网址: https://0c660a4e-8000-app.lightly.teamcode.com/submit.php

请求方法: POST

3.2.1 提交表单

- POST方法
  - 将请求参数封装在HTTP请求数据中，对传输数据的大小无限制
  - 从全局数组变量$_POST[]获得通过POST方法传递的数据

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  姓名: <input type="text" name="name"/> <br/>
  密码: <input type="password" name="password"/> <br/>
  性别:
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历:
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣:
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介: <textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片: <input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  /* 使用可变变量+foreach循环 */
  foreach($_POST as $key=>$value) {
    $$key = $value;
  }

  /* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
?>
```

3.2.2 上传文件

- 全局数组变量$_FILES[]：存储上传的文件信息
  - 是一个二维数组（上传单个文件）或三维数组（上传多个文件）
  - 其一维数组的键值是<input>标签的name属性值

元 素 名	说　明
$_FILES["filename"]["name"]	存储上传文件的文件名。如text.txt、title.jpg等
$_FILES["filename"]["size"]	存储文件大小，单位为字节
$_FILES["filename"]["tmp_name"]	存储文件在临时目录中使用的文件名。因为文件在上传时，首先要将其以临时文件的身份保存在临时目录中
$_FILES["filename"]["type"]	存储上传文件的MIME类型，MIME类型规定各种文件格式的类型。每种MIME类型都是由"/"分隔的主类型和子类型组成的。例如："image/gif"，主类型为"图像"，子类型为GIF格式的文件，"text/html"代表HTML格式的文本文件
$_FILES["filename"]["error"]	存储了上传文件的结果。如果返回0，则说明文件上传成功

- 关键函数
  - is_uploaded_file(string filename)：检查是否上传过指定文件
  - move_uploaded_file(string src, string dest)：移动文件位置

上传单个文件

```
<form action="submit.php" method="POST" enctype="multipart/form-data">
  ……
  照片：<input type="file" name="photo"/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

```
<?php
  if (is_uploaded_file($_FILES["photo"]["tmp_name"]) && $_FILES["photo"]["error"] == 0) {
    echo "Upload: ".$_FILES["photo"]["name"]."<br/>";
    echo "Type: ".$_FILES["photo"]["type"]."<br/>";
    echo "Size: " .($_FILES["photo"]["size"] / 1024)." Kb<br/>";
    echo "Stored in: ".$_FILES["photo"]["tmp_name"]."<br/>";
    echo "Path: ".$_SERVER["DOCUMENT_ROOT"]."<br/>";
    $temporary_path = $_FILES["photo"]["tmp_name"];                    // 上传文件的临时存放路径
    $relative_path = "file/".$_FILES["photo"]["name"];                 // 上传文件的目标存放路径（相对路径）
    $absolute_path = $_SERVER["DOCUMENT_ROOT"]."/".$relative_path;     // 上传文件的目标存放路径（绝对路径）
    move_uploaded_file($temporary_path, $absolute_path);               // 从临时存放路径移动到目标存放路径
    echo "<img src=\"$relative_path\"/>";
  }
?>
```

上传多个文件

```html
<form action="submit.php" method="POST" enctype="multipart/form-data">
  姓名: <input type="text" name="name"/> <br/>
  密码: <input type="password" name="password"/> <br/>
  性别:
      <input type="radio" name="sex" value="male" checked/>男性
      <input type="radio" name="sex" value="female"/>女性
      <br/>
  学历:
      <select name="education">
        <option value="undergraduate"/>本科</option>
        <option value="graduate" selected/>研究生</option>
      </select>
      <br/>
  兴趣:
      <input type="checkbox" name="hobby" value="sports"/>运动
      <input type="checkbox" name="hobby" value="travel"/>旅游
      <input type="checkbox" name="hobby" value="music"/>音乐
      <br/>
  简介: <textarea name="intro" placeholder="不超过100字"></textarea> <br/>
  照片: <input type="file" name="photo" multiple/> <br/>

  <input type="submit"/>
  <input type="reset"/>
</form>
```

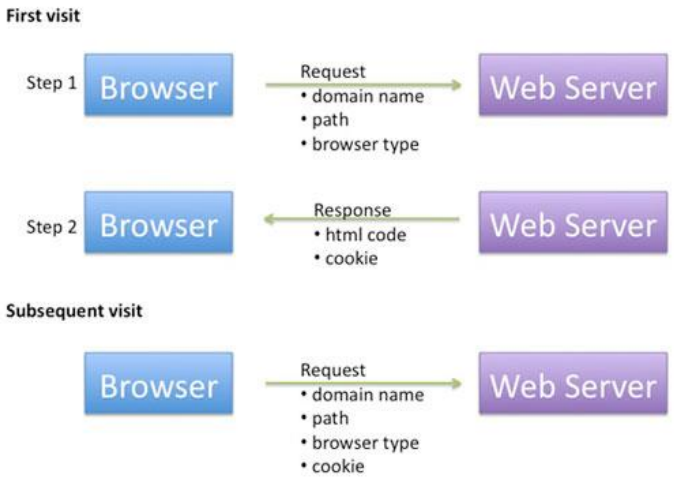
```php
<?php
  /* 使用可变变量+foreach循环 */
  foreach($_POST as $key=>$value) {
    $$key = $value;
  }

  /* 输出变量值 */
  echo "name: ".$name."<br/>";
  echo "password: ".$password."<br/>";
  echo "sex: ".$sex."<br/>";
  echo "education: ".$education."<br/>";
  echo "hobby: ".$hobby."<br/>";
  echo "intro: ".$intro."<br/>";
  echo "photo: ".$photo."<br/>";
?>
```

3.2.3 会话控制

- **会话**的含义是指某个用户在网站上的有始有终的一系列动作的集合。例如，用户在访问网站时， 会话就是指从用户登入站点到到关闭浏览器所经过的这段过程
- Session是将信息保存在服务器上，并通过一个Session ID来传递客户端的信息，服务器在接收到Session ID后根据这个ID来提供相关的Session信息资源
- Cookie是将所有的信息以文本文件的形式保存在客户端，并由浏览器进行管理和维护

Cookie

- Cookie是一段存放在客户端的文本数据，由服务器端生成，发送给客户端浏览器。客户端浏览器如启用cookie，则会将这个小文本数据保存到其某个目录下的文本文件内（**永久Cookie**）或内存中（**临时Cookie**）
- 客户端下次登录同一域名下的网页，浏览器则会自动将Cookie读入之后，传给服务器端。服务器端可以对该Cookie进行读取并验证。一般情况下，Cookie中的值是以key-value的形式进行表达的
- HTTP是一种无状态协议。Cookie可认为是一种**跨页面**的数据共享机制，常被用来保存用户认证相关的信息
- 问题：什么时候需要跨页面数据共享？

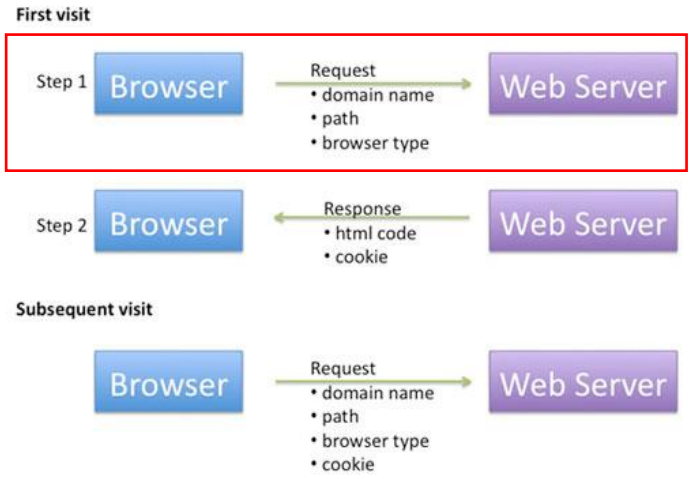


Cookie

- 常见属性

cookie属性	基本描述	举例	备注
name=value	cookie键值对	id=a3fWa	
expires	cookie过期时间	expires=Tue, 10-Jul-2013 08:30:18 GMT	
domain	指定哪些主机可以接收cookie	Domain=mozilla.org; 不设置则等于当前页面domian	定义cookie将被种植在哪些地方
path	指示哪些路径的请求会发送cookie	Path=/docs	定义cookie将会种植在哪些地方
secure	指定通过https请求发送cookie		限制后续请求访问该cookie的姿势
httponly	指示是否允许通过JavaScript Document.cookie API访问cookie		限制后续请求访问该cookie的姿势
hostonly	标记该cookie与创建cookie的页面主机名的相关性		如果host-only-flag为true时，只有当前域名与该Cookie的Domain属性完全相等，才检查path、secure、httponly等属性的匹配性
samesite	让服务器指定是否允许跨站请求发送cookie	SameSite=Lax	限制后续的跨站访问是否允许携带cookie

微人大：登录请求



▼ 常规

请求网址: https://v.ruc.edu.cn/auth/login

请求方法: POST

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▶ 响应标头 (13)

▼ 请求标头 查看源代码

Accept: application/json

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Connection: keep-alive

Content-Length: 505

Content-type: application/x-www-form-urlencoded

Host: v.ruc.edu.cn

Origin: https://v.ruc.edu.cn

Referer: https://v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%253A%252F%252Fv.ruc.edu.cn%252Fsso%252Fcallback%253Fschool_code%253Druc%2526theme%253Dschools%26response_type%3Dcode%26school_code%3Druc%26scope%3Dall%26state%3DfwQ6exD-ERNF_Ep0%26theme%3Dschools&school_code=ruc

sec-ch-ua: "Google Chrome";v="105", "Not)A;Brand";v="8", "Chromium";v="105"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

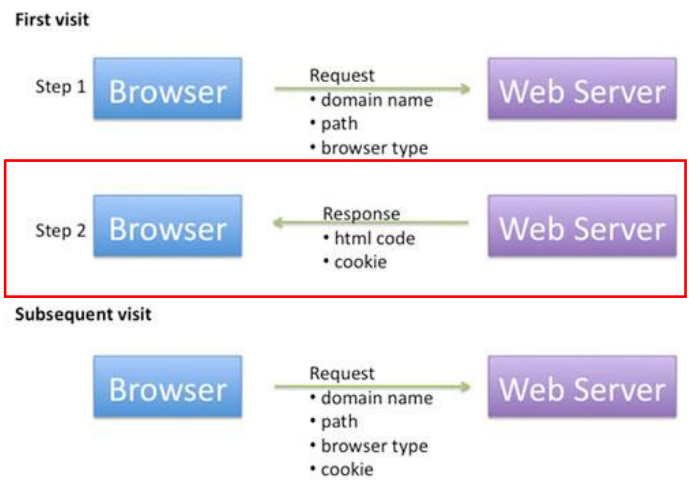
Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36

▼ 表单数据 查看源代码 查看网址编码格式的数据

{"username":"ruc[redacted]","password":"[redacted]","code":"qeew","remember_me":"false","redirect_uri":"https://v.ruc.edu.cn/oauth2/authorize?client_id=accounts.tiup.cn&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Fsso%2Fcallback%3Fschool_code%3Druc%26theme%3Dschools&response_type=code&school_code=ruc&scope=all&state=fwQ6exD-ERNF_Ep0&theme=schools","twofactor_password":"","twofactor_recovery":"","token":"kgo4glwg1r","captcha_id":"1uKgeOyhNjYEjsOyMYYY"}:

微人大：登录响应



▼ 常规

请求网址: https://v.ruc.edu.cn/auth/login

请求方法: POST

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▼ 响应标头　　查看源代码

Access-Control-Allow-Origin: *

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

Cache-Control: no-store

Connection: keep-alive

Content-Length: 290

Content-Type: application/json; charset=utf-8

Date: Mon, 26 Sep 2022 07:11:56 GMT

Expires: Fri, 01 Jan 1990 00:00:00 GMT

Pragma: no-cache

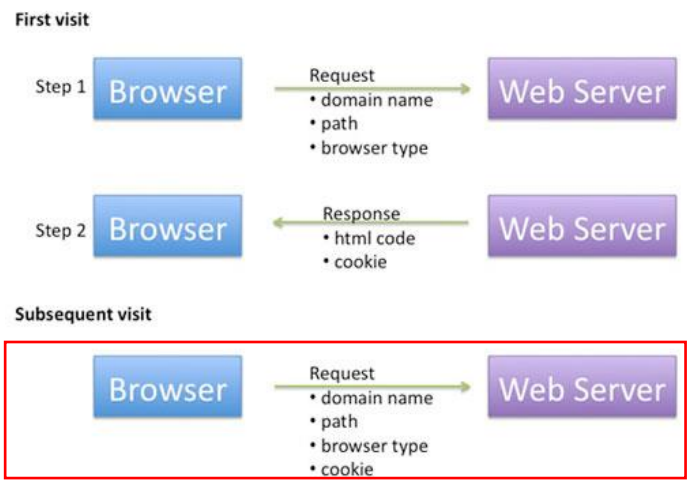
Server: none

Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly

Set-Cookie: session=6fd37[redacted]792fa5; Path=/; HttpOnly

X-Content-Type-Options: nosniff

微人大：后续请求



▼ 常规

请求网址: https://v.ruc.edu.cn/me

请求方法: GET

状态代码: 200 OK

远程地址: 10.21.1.142:443

引荐来源网址政策: strict-origin-when-cross-origin

▶ 响应标头 (12)

▼ 请求标头 查看源代码

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Connection: keep-alive

Cookie: is_simple=1; session=6fd37[redacted]792fa5; access_token=jeAlhChnRhOVkJl7J6xJlg; tiup_uid=[redacted]652

Host: v.ruc.edu.cn

Referer: https://v.ruc.edu.cn/auth/login?&proxy=true&redirect_uri=https%3A%2F%2Fv.ruc.edu.cn%2Foauth2%2Fauthorize%3Fclient_id%3Daccounts.tiup.cn%26redirect_uri%3Dhttps%253A%252F%252Fv.ruc.edu.cn%252Fsso%252Fcallback%253Fschool_code%253Druc%2526theme%253Dschools%26response_type%3Dcode%26school_code%3Druc%26scope%3Dall%26state%3DfwQ6exD-ERNF_Ep0%26theme%3Dschools&school_code=ruc

sec-ch-ua: "Google Chrome";v="105", "Not)A;Brand";v="8", "Chromium";v="105"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

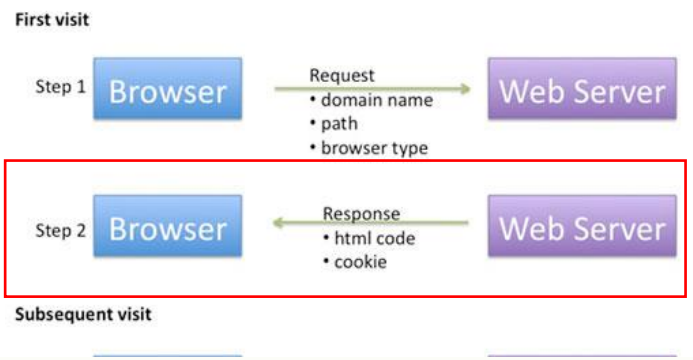
Sec-Fetch-Site: same-origin

Sec-Fetch-User: ?1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36

服务器端操作Cookie



创建Cookie

语法

```
setcookie(name,value,expire,path,domain,secure,httponly)
```

参数	描述
name	必需。规定 cookie 的名称。
value	必需。规定 cookie 的值。
expire	可选。规定 cookie 的有效期。
path	可选。规定 cookie 的服务器路径。
domain	可选。规定 cookie 的域名。
secure	可选。规定是否通过安全的 HTTPS 连接来传输 cookie。
httponly	可选。规定是否设置http-only选项。

语法

```
header(string,replace,http_response_code)
```

参数	描述
string	必需。规定要发送的报头字符串。
replace	可选。指示该报头是否替换之前的报头，或添加第二个报头。 默认是 true（替换）。false（允许相同类型的多个报头）。
http_response_code	可选。把 HTTP 响应代码强制为指定的值。（PHP 4 以及更高版本可用）

```php
<?php
  header("Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly");
  setcookie("session","6fd37[redacted]792fa5", 0, "/", "", false, true);
?>
<html>
</html>
```

Set-Cookie: is_simple=1; Path=/; Max-Age=3600; HttpOnly

Set-Cookie: session=6fd373[redacted]792fa5; path=/; HttpOnly

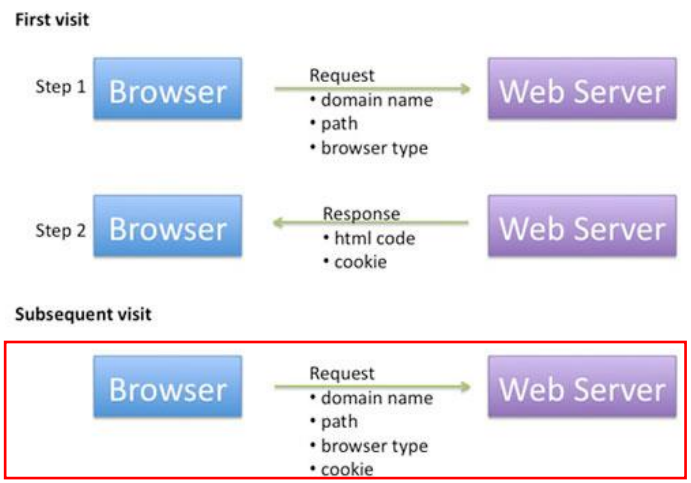
注意：setcookie()和header()是通过HTTP请求响应的Header来完成的，需要在请求响应内容输出之前执行。

删除Cookie

```php
<?php
  setcookie("session", "", 0);  // 设置value为空或者expire为0，即可清空Cookie
?>
```

服务器端操作Cookie

- 获取Cookie：全局数组变量$_COOKIE[]




```php
<?php
  setcookie("session","6fd37[redacted]792fa5", 0, "/", "", false, true);
  if (isset($_COOKIE["session"])) echo "名为session的Cookie项目的值为".$_COOKIE["session"];
  else echo "名为session的Cookie项目不存在";
?>
```

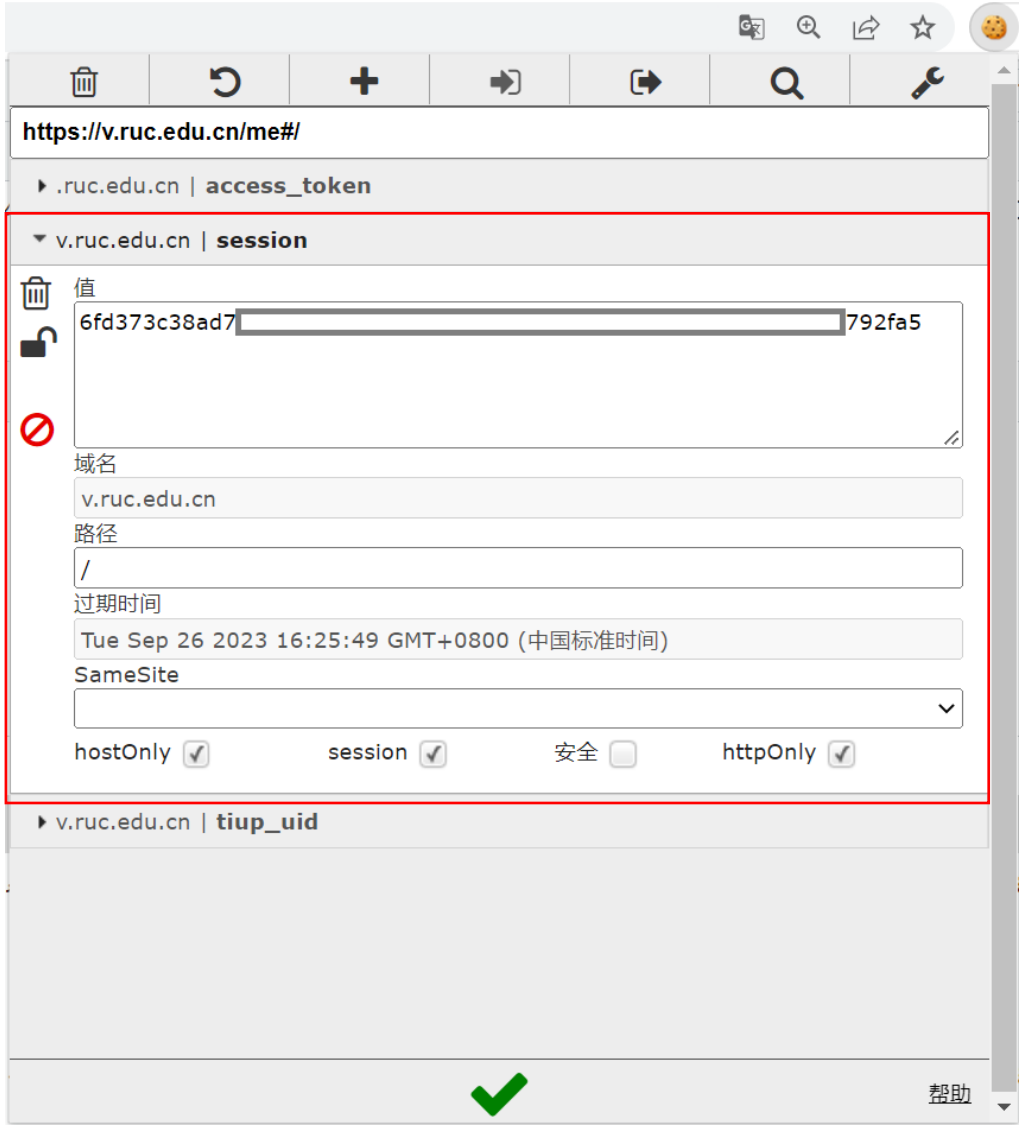
第一次访问： 名为session的Cookie项目不存在

第二次访问： 名为session的Cookie项目的值为6fd37[redacted]792fa5

注意：

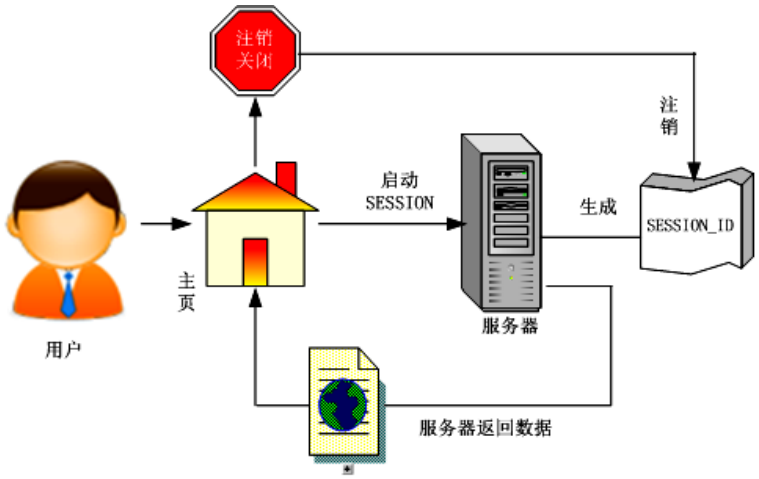
- 通过setcookie()函数创建Cookie后，在当前页应用echo $_COOKIE["name"]不会有任何输出
- 在刷新后或者到达下一个页面时才可以看到Cookie值
- 因为setcookie()函数执行后，会向客户端发送一个Cookie，若不刷新或者浏览下一个页面，客户端就不能将Cookie送回

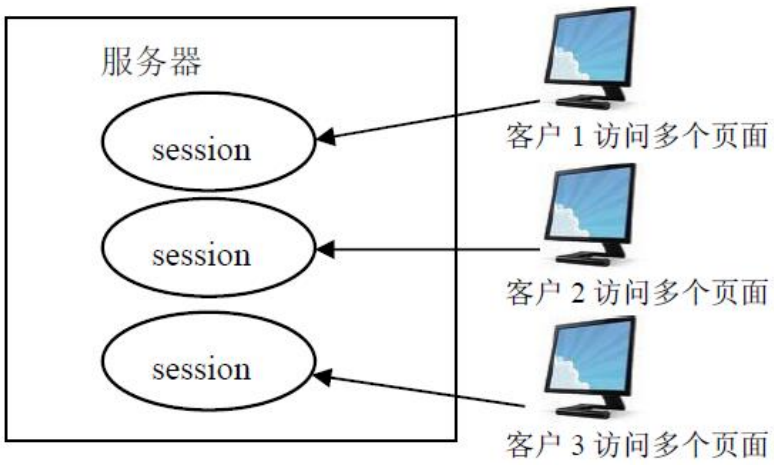
客户端操作Cookie：EditThisCookie扩展



Session

- Session中的数据可以被同一个客户在网站的一次会话过程共享。但是对于不同客户来说，每个人的Session是不同的
- Session可认为是一种保存在服务器端的跨页面（跨请求）的数据共享机制
- 当用户结束会话时（如关闭浏览器），服务端的Session并不会被立即删除（若非主动告知，服务端并不会知道用户结束了会话）。除非程序通知服务器删除一个Session，否则服务器会一直保留这个Session对象，直到其超时失效，被垃圾收集机制收集掉



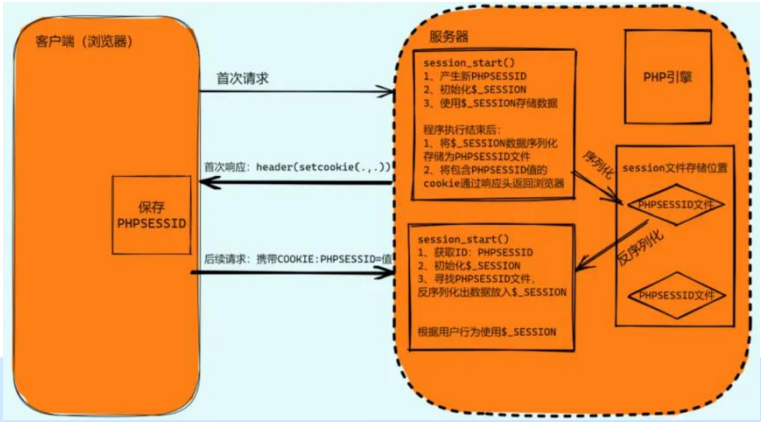


Session

- 当程序需要为某个客户端的请求创建一个Session的时候，服务器可以首先检查这个客户端的请求里是否已包含了一个Session标识：称为Session ID（通常为一个随机的长字符串）
  - 如果已包含一个Session ID则说明以前已经为此客户端创建过Session，服务器就按照Session ID把这个 Session检索出来使用
  - 如果客户端请求不包含Session ID，则为此客户端创建一个Session并且生成 一个与此Session相关联的Session ID， Session ID的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串，这个Session ID可在响应中返回给客户端保存
- 一般情况下， **Session ID被保存在客户端的Cookie中**（用户登录成功后，将用户认证成功的信息存放在一个Session中，并将此Session ID设置存放在客户端Cookie中）

Session

- 开启Session：session_start()
- 终结Session：session_destroy()
- 获得Session ID：session_id()
- 存储/获取Session变量：全局数组变量$_SESSION[]



注意：

- 如果客户端发送给服务器的Cookie中包含名为“PHPSESSID”的项，session_start()会使用$_COOKIE["PHPSESSID"]对应的session
- 客户端发送给服务器的Cookie中不包含名为“PHPSESSID”的项，session_start()会创建一个名为“PHPSESSION”的Cookie项发送给客户端
- session_destroy()只销毁服务器端session，不会销毁客户端名为“PHPSESSID”的Cookie项

```php
<?php
  session_start();                    // 开启Session

  $username = $_POST["username"];     // 获取表单中的用户名
  $password = $_POST["password"];     // 获取表单中的密码

  if ($username == "admin001" && $password == "admin001")
    $_SESSION["role"] = "admin";                          // 管理员身份
  else if ($username == "teacher001" && $password == "teacher001")
    $_SESSION["role"] = "teacher";                        // 存储Session变量
  else if ($username == "student001" && $password == "student001")
    $_SESSION["role"] = "student";                        // 教师身份
  else session_destroy();                                 // 存储Session变量
                                                          // 学生身份
  echo "Session ID: ".session_id()."<br/>";     // 获得Session ID   // 存储Session变量
  echo "Role: ".$_SESSION["role"]."<br/>";      // 获取Session变量  // 终结Session
?>
```

Session ID: 028c70308f8682092859519569ad4787
Role: admin

实例：微人大认证页面会话控制


```php
<?php
  /* pannel.php: 面板页面 */
  session_start();
  if (isset($_SESSION["username"])) {
    $role = $_COOKIE[“role”]; //安全问题?
    echo "你好，".$role."<br/>";
    echo "<a href='logout.php'>退出</a>";
  } else echo "无权访问！";
?>
```

```php
<?php
  /* logout.php: 退出页面 */
  session_start();    // 启用Session
  session_unset();    // 销毁$_SESSION数组
  session_destroy(); // 销毁Session
  header("Location: index.php");
?>
```

```php
<?php
  /* index.php: 首页 */
  session_start(); // 启用Session
  if (isset($_SESSION["username"])) { // 判断是否处于成功登录状态
    header("Location: pannel.php"); // 跳转到面板页面
    exit(); // 终止后续代码的执行
  }
?>
<form action="login.php" method="POST" id="login-form">
<?php $_SESSION["code"] = rand()%10; ?>
  <img src="img/code_<?=$_SESSION["code"]?>.png">
</form>
```

```php
<?php
  /* login.php: 登录页面 */
  session_start();                          // 启用Session
  $username = $_POST["username"];           // 获取表单中的用户名
  $password = $_POST["password"];           // 获取表单中的密码
  $code = $_POST["code"];                   // 获取表单中的验证码

  //维护一个验证码正确值和验证码图片下标的映射
  $checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
  $correct_code = $checks[$_SESSION["code"]];

  if ($code == $correct_code) {
    if ($username == "admin001" && $password == "admin001")                  // 管理员
      $_SESSION["role"] = "admin";                                           // 在Session中记录角色
    else if ($username == "teacher001" && $password == "teacher001")         // 教师
      $_SESSION["role"] = "teacher";                                         // 在Session中记录角色
    else if ($username == "student001" && $password == "student001")         // 学生
      $_SESSION["role"] = "student";                                         // 在Session中记录角色

    if (isset($_SESSION["role"])) {                                                  // 登录成功
      $_SESSION["username"] = $username;                                     // 在Session中记录用户名
      setcookie("role",$_SESSION["role"], 0, "/", "", false, true);          // 设置名为“role”的Cookie项
      header("Location: pannel.php");                                        // 跳转到用户面板
    } else echo "用户名或密码不正确！";                                      // 错误提示
  } else echo "验证码不正确！";                                              // 错误提示
?>
```

3.2.4 其它信息

- 全局数组变量$_ENV[]：PHP的环境变量信息
- 全局数组变量$_GLOBAL[]：引用全局作用域中可用的全部变量
- 全局数组变量$_SERVER[]：与PHP服务器相关的信息
- 全局数组变量$_REQUEST[]：获取GET方法、POST方法和通过Cookie传到服务器的信息，是$_GET[]、$_POST[]和$_COOKIE[]的汇总

3.2.4 其它信息

- 全局数组变量$_SERVER[]：与PHP服务器相关的信息

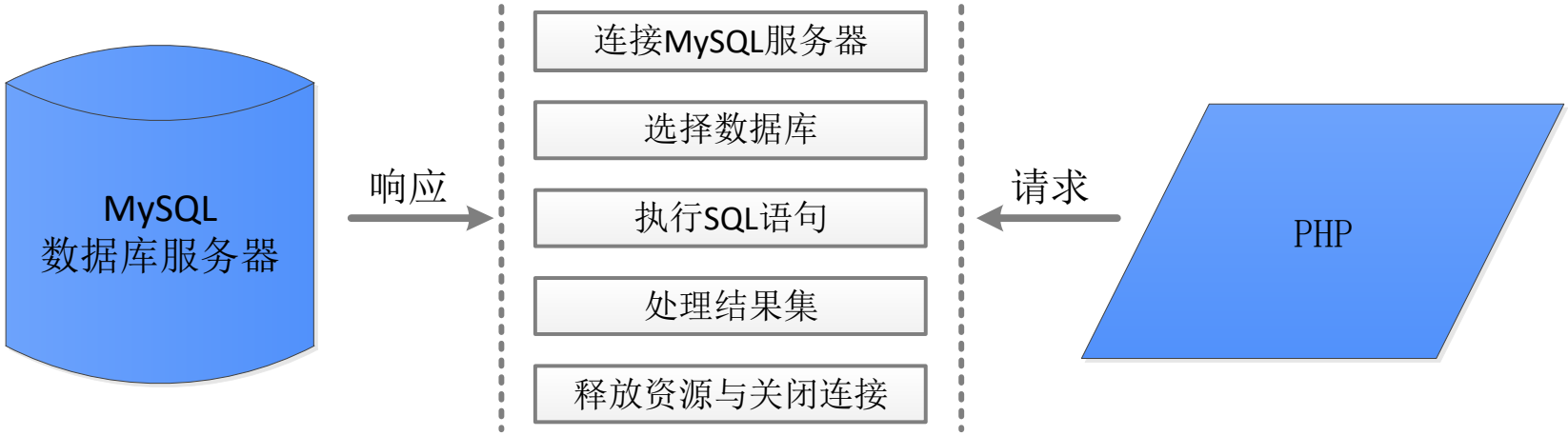
数组元素	说明
$_SERVER['PHP_SELF']	当前执行脚本的文件名，与document root有关。例如，在地址为http://c.biancheng.net/test.php/foo.bar的脚本中使用 $_SERVER['PHP_SELF'] 将得到 /test.php/foo.bar
$_SERVER['SERVER_ADDR']	当前运行脚本所在服务器的 IP 地址
$_SERVER['SERVER_NAME']	当前运行脚本所在服务器的主机名。如果脚本运行于虚拟主机中，该名称就由那个虚拟主机所设置的值决定
$_SERVER['SERVER_PROTOCOL']	请求页面时通信协议的名称和版本。例如，“HTTP/1.0”
$_SERVER['REQUEST_METHOD']	访问页面使用的请求方法。例如“GET”“HEAD”“POST”“PUT”
$_SERVER['DOCUMENT_ROOT']	当前运行脚本所在的文档根目录。在服务器配置文件中定义
$_SERVER['HTTP_ACCEPT_LANGUAGE']	当前请求头中 Accept-Language: 项的内容（如果存在）。例如，“en”
$_SERVER['REMOVE_ADDR']	浏览当前页面的用户 IP 地址，注意与 $_SERVER['SERVER_ADDR'] 的区别
$_SERVER['SCRIPT_FILENAME']	当前执行脚本的绝对路径
$_SERVER['SCRIPT_NAME']	包含当前脚本的路径
$_SERVER['REQUEST_URI']	URI 用来指定要访问的页面。例如，“index.html”
$_SERVER['PATH_INFO']	包含由客户端提供的、跟在真实脚本名称之后并且在查询语句（query string）之前的路径信息（如果存在）。例如，当前脚本是通过 URL http://c.biancheng.net/php/path_info.php/some/stuff?foo=bar 被访问的，那么 $_SERVER['PATH_INFO'] 将包含 /some/stuff

```php
<?php
  echo $_SERVER["SERVER_NAME"]."<br/>";
  echo $_SERVER["SERVER_PROTOCOL"]."<br/>";
  echo $_SERVER["REQUEST_METHOD"]."<br/>";
  echo $_SERVER["REQUEST_URI"]."<br/>";
  echo $_SERVER["DOCUMENT_ROOT"]."<br/>";
  echo $_SERVER["SCRIPT_FILENAME"]."<br/>";
?>
```

```
0.0.0.0
HTTP/1.1
GET
/?port=8000&dcsId=b8c0a052&token=Q_OuBnzfQaa4Hb_odoc3yQ
/workspace/PHPProject
/workspace/PHPProject/index.php
```

3.2.5 数据库操作

- PHP提供了大量的MySQL数据库操作函数，可以方便地实现访问MySQL数据库的各种需要，从而轻松完成Web应用程序开发



3.2.5 数据库操作

- **连接MySQL服务器**：使用mysql_connect()函数建立与MySQL服务器的连接。
- **选择数据库**：使用mysql_select_db()函数选择MySQL数据库服务器上的数据库，并与该数据库建立连接。
- **执行SQL语句**：在所选择的数据库中使用mysql_query()函数执行SQL语句。
- **处理结果集**：执行完SQL语句后，使用mysql_fetch_array()、mysql_fetch_row()、mysql_fetch_object()等函数对结果集进行相关操作。
- **释放资源**：处理完结果集后，需要使用mysql_free_result()函数关闭结果集，以释放系统资源。
- **关闭连接**：为了避免多用户连接造成系统性能下降甚至死机，在完成数据库的操作后，应使用mysql_close()函数断开与MySQL服务器的连接。

mysql扩展的使用

连接MySQL服务器

在操作MySQL数据库之前，需先与MySQL数据库服务器建立连接。在PHP的mysql扩展中通常使用mysql_connect()函数与其建立连接，其声明方式如下：

```
resource mysql_connect ([ string $server [, string $username [, string $password [,bool $new_link [, int $client_flags ]]]]] )
```

- $server参数的默认值是“localhost:3306”，其中localhost表示本地服务器，3306表示默认端口号（可以省略）。
- $username参数表示登陆MySQL服务器的用户名。
- $password参数表示MySQL服务器的用户密码。
- $new_link参数表示该函数每次被调用时总是打开新的连接。
- $client_flags参数的值是MySQL客户端常量，在实际使用中较少，具体参考PHP手册。

连接MySQL服务器

为了让读者更好地掌握mysql_connect()函数的用法，接下来通过一个案例来演示如何进行数据库的连接，如例1-1所示。 [点击查看例1-1]

1.2.1 连接MySQL服务器

注意：在连接数据库发生错误时，会出现错误信息，但在上线项目中建议对错误信息进行屏蔽，并可以自定义错误提示，通常有如下两种方式：

1. 在mysql_connect()函数前面添加符号“@”，可以用于屏蔽这个函数出错信息的显示。
2. 当需要自定义错误提示时，可以写成如下形式：

```
mysql_connect('localhost:3306','root','123456') or die('数据库服务器连接失败！')
```

- 如果调用函数出错，将执行or后面的语句，其中die()函数用于停止脚本执行并向用户输出错误信息。
- 建议在程序开发阶段不要屏蔽错误信息，避免出错后难以找到问题。

选择数据库

连接MySQL数据库成功之后，接下来使用mysql_select_db()函数选择数据库，其声明方式如下：

```
bool mysql_select_db ( string $database_name [, resource $link_identifier ] )
```

- 参数$database_name表示要选择的数据库名称。
- 可选参数$link_identifier表示MySQL连接，默认使用最近打开的连接；如果没有找到该连接，则尝试不带参数调用mysql_connect()来创建；如果没有找到并无法建立该连接，则会生成E_WARNING级别的错误。

为了让读者更好地掌握mysql_select_db()函数的用法，接下来通过一个案例来演示如何使用此函数选择数据库，如例1-2所示。

[点击查看例1-2]

执行SQL语句

完成数据库的选择后，就是对SQL语句的执行操作了，在PHP中，通常使用mysql_query()函数执行SQL语句，其声明方式如下：

```
resource mysql_query ( string $query [, resource $link_identifier = NULL] )
```

- 参数$query表示SQL查询语句，但是查询字符串不应以“;”结束。
- $link_idenifier是可选项，表示MySQL连接标识，若省略，则使用最近打开的连接。
- 该函数仅对SELECT、SHOW、EXPLAIN或DESCRIBE语句执行成功时返回一个资源标识符，如果查询执行失败则返回FALSE。
- 如果SQL语句是INSERT、DELETE、UPDATE等操作指令，成功则返回TRUE，否则返回FALSE。

处理结果集

执行完SQL语句后，需要使用函数从结果集中获取信息，在PHP中常用的处理结果集的函数有mysql_fetch_row()函数、mysql_fetch_assoc()函数、mysql_fetch_array()函数以及mysql_fetch_object()函数。下面分别介绍这几个函数的具体使用方法。

处理结果集

1. mysql_fetch_row ()函数

首先了解一下mysql_fetch_row()函数，其声明方式如下：

```
array  mysql_fetch_row ( resource $result )
```

- 该函数的返回值是数组类型。
- 参数$result表示资源型结果集。每执行一次该函数都将从结果集资源中取出一条记录放入到一维数组中，下标从0开始，并且内部数据指针自动指向下一条数据，直到没有更多行时返回FALSE。

处理结果集

2. mysql_fetch_assoc ()函数

mysql_fetch_assoc()函数也可以获取结果集，与mysql_fetch_row()函数的唯一区别是通过字段名称来获取数据，其声明方式如下：

```
array mysql_fetch_assoc ( resource $result )
```

- array表示该函数的返回值类型是数组。
- 参数$result表示资源型结果集。每执行一次该函数都将从结果集资源中取出一条记录放入到一维数组中，并且内部数据指针自动指向下一条数据，直到没有更多行时返回FALSE。

处理结果集

2. mysql_fetch_assoc ()函数

注意：由于mysql扩展自 PHP 5.5.0 起已废弃，并在将来会被移除。所以它可以由以下两个函数替换：

- 使用 PDO_MySQL 扩展中的PDOStatement::fetch(PDO::FETCH_ASSOC) 。
- 使用MySQLi扩展中的mysqli_fetch_assoc() 。

处理结果集

3. mysql_fetch_array()函数

使用mysql_fetch_array()函数同样可以获取结果集中的数据，其声明如下所示：

```
array mysql_fetch_array ( resource $result [, int $result_type ] )
```

- $result是资源类型的参数，传入的是由mysql_query()函数返回的数据指针。
- $result_type是可选的常量，其值可以是MYSQL_BOTH（默认参数），MYSQL_ASSOC或MYSQL_NUM中的一种，其中MYSQL_ASSOC只得到关联索引形如mysql_fetch_assoc()，MYSQL_NUM只得到数字索引形如mysql_fetch_row()。

处理结果集

3. mysql_fetch_array()函数

注意：mysql_fetch_array()函数返回的字段名区分大小写，这是初学者最容易忽略的问题。在获取结果集时，用 mysql_fetch_array() 并不明显，比用 mysql_fetch_row() 慢，而且还提供了明显更多的值。同时，如果结果中的两个或以上的列具有相同的字段名，最后一列优先级最高，要访问其他列，必须用该列的数字索引或给该列起别名。

处理结果集

4. mysql_fetch_object()函数

函数mysql_fetch_object与mysql_fetch_array()类似，只有一点区别，即前者返回的是一个对象而不是数组，其声明方式如下所示：

```
object mysql_fetch_object ( resource $result )
```

- 参数$result是调用mysql_query()函数返回的结果集。
- 由于该函数的返回值类型是object类型，所以只能通过字段名来访问数据，并且此函数返回的字段名大小写敏感。

mysql扩展的使用

多学一招：myql_result()

函数mysql_result()可从结果集中获取一个单元的内容，其声明方式如下所示：

```
mixed mysql_result ( resource $result , int $row [, mixed $field ] )
```

- 获取内容的字段参数可以是字段的偏移量$row或者是字段名$field，其中$field是可选参数。

处理结果集

除了以上这些函数以外还有一些其他经常用到的函数，下面对其分别介绍。

1. mysql_num_rows()

在执行SELECT查询语句时，使用mysql_num_rows()函数可以返回查询的记录数，其声明方式如下：

```
int mysql_num_rows ( resource $result )
```

- $result表示查询时返回的结果集资源。
- 此函数仅仅对SELECT查询语句有效。

2. mysql_affected_rows()

当需要取得INSERT，UPDATE 或者 DELETE语句执行后影响的行数时，可以使用mysql_affected_rows()函数，其声明方式如下：

处理结果集

```
int mysql_affected_rows ([ resource $link_identifier = NULL ] )
```

- int表示执行成功则返回受影响行的数目，如果最近一次查询失败，则返回 -1。
- 参数$link_identifier表示MySQL连接。如果不指定，则使用最近打开的连接。
- mysql_affected_rows()函数在PHP5.5.0起将废弃，它可由mysqli_affected_rows()函数或PDOStatement::rowCount()方法来代替。

3. mysql_insert_id()

在项目中，经常要取得上一次插入操作时产生的的ID号，其声明方式如下：

```
int mysql_insert_id ([ resource $link_identifier ] )
```

- 此函数只有一个可选参数$link_identifier即返回的资源结果集。
- 如果上一查询没有产生AUTO_INCREMENT的ID值，则该函数的返回值为0。

释放资源与关闭连接

1. mysql_free_result()

处理完结果集后，若考虑到返回很大的结果集会占用多少内存时，需调用mysql_free_result()函数以释放系统资源，其声明方式如下：

```
bool mysql_free_result ( resource $result )
```

- 函数的返回值类型是布尔类型，执行成功返回TRUE，执行失败返回FALSE。

释放资源与关闭连接

2. mysql_close()

当一次性返回的结果集比较大，或网站访问量比较多时，最好用mysql_close()函数手动进行释放，其声明方式如下：

```
bool mysql_close ([ resource $link_identifier = NULL ] )
```

- 函数的返回值类型是布尔型，成功时返回TRUE，失败时返回FALSE。
- $link_identifer代表要关闭的MySQL连接资源。如果没有指定$link_identifer，则关闭上一个打开的连接。

释放资源与关闭连接

2. mysql_close()

注意：

通常不需要使用mysql_close()，因为已打开的非持久连接会在脚本执行完毕后自动关闭。且自PHP5.5.0起该函数已废弃，并在将来会被移除，本函数可由mysqli_close()函数或在PDO中为PDO对象设置一个NULL值来代替。

实例：微人大认证页面数据库查询


```php
<?php
  /* pannel.php: 面板页面 */
  session_start();
  if (isset($_SESSION["username"])) {
    $role = $_COOKIE[“role”]; //安全问题?
    echo "你好，".$role."<br/>";
    echo "<a href='logout.php'>退出</a>";
  } else echo "无权访问！";
?>
```

```php
<?php
  /* logout.php: 退出页面 */
  session_start();    // 启用Session
  session_unset();    // 销毁$_SESSION数组
  session_destroy(); // 销毁Session
  header("Location: index.php");
?>
```

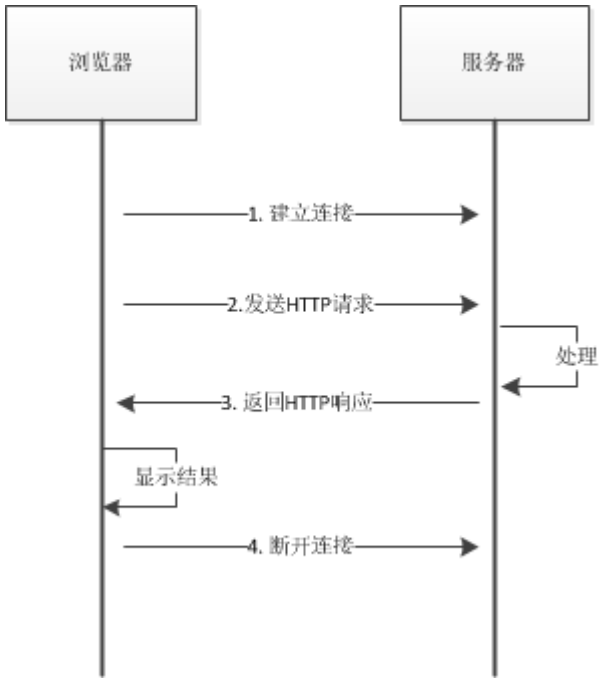
```php
<?php
  /* index.php: 首页 */
  session_start(); // 启用Session
  if (isset($_SESSION["username"])) { // 判断是否处于成功登录状态
    header("Location: pannel.php"); // 跳转到面板页面
    exit(); // 终止后续代码的执行
  }
?>
<form action="login.php" method="POST" id="login-form">
<?php $_SESSION["code"] = rand()%10; ?>
  <img src="img/code_<?=$_SESSION["code"]?>.png">
</form>
```

```php
<?php
  /* login.php: 登录页面 */
  session_start();                          // 启用Session
  $username = $_POST["username"];           // 获取表单中的用户名
  $password = $_POST["password"];           // 获取表单中的密码
  $code = $_POST["code"];                   // 获取表单中的验证码

  //维护一个验证码正确值和验证码图片下标的映射
  $checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
  $correct_code = $checks[$_SESSION["code"]];

  if ($code == $correct_code) {
    if ($username == "admin001" && $password == "admin001")                  // 管理员
      $_SESSION["role"] = "admin";                                           // 在Session中记录角色
    else if ($username == "teacher001" && $password == "teacher001")         // 教师
      $_SESSION["role"] = "teacher";                                         // 在Session中记录角色
    else if ($username == "student001" && $password == "student001")         // 学生
      $_SESSION["role"] = "student";                                         // 在Session中记录角色

    if (isset($_SESSION["role"])) {                                                  // 登录成功
      $_SESSION["username"] = $username;                                     // 在Session中记录用户名
      setcookie("role",$_SESSION["role"], 0, "/", "", false, true);          // 设置名为“role”的Cookie项
      header("Location: pannel.php");                                        // 跳转到用户面板
    } else echo "用户名或密码不正确！";                                      // 错误提示
  } else echo "验证码不正确！";                                              // 错误提示
?>
```

目录

4.1 HTTP协议与消息

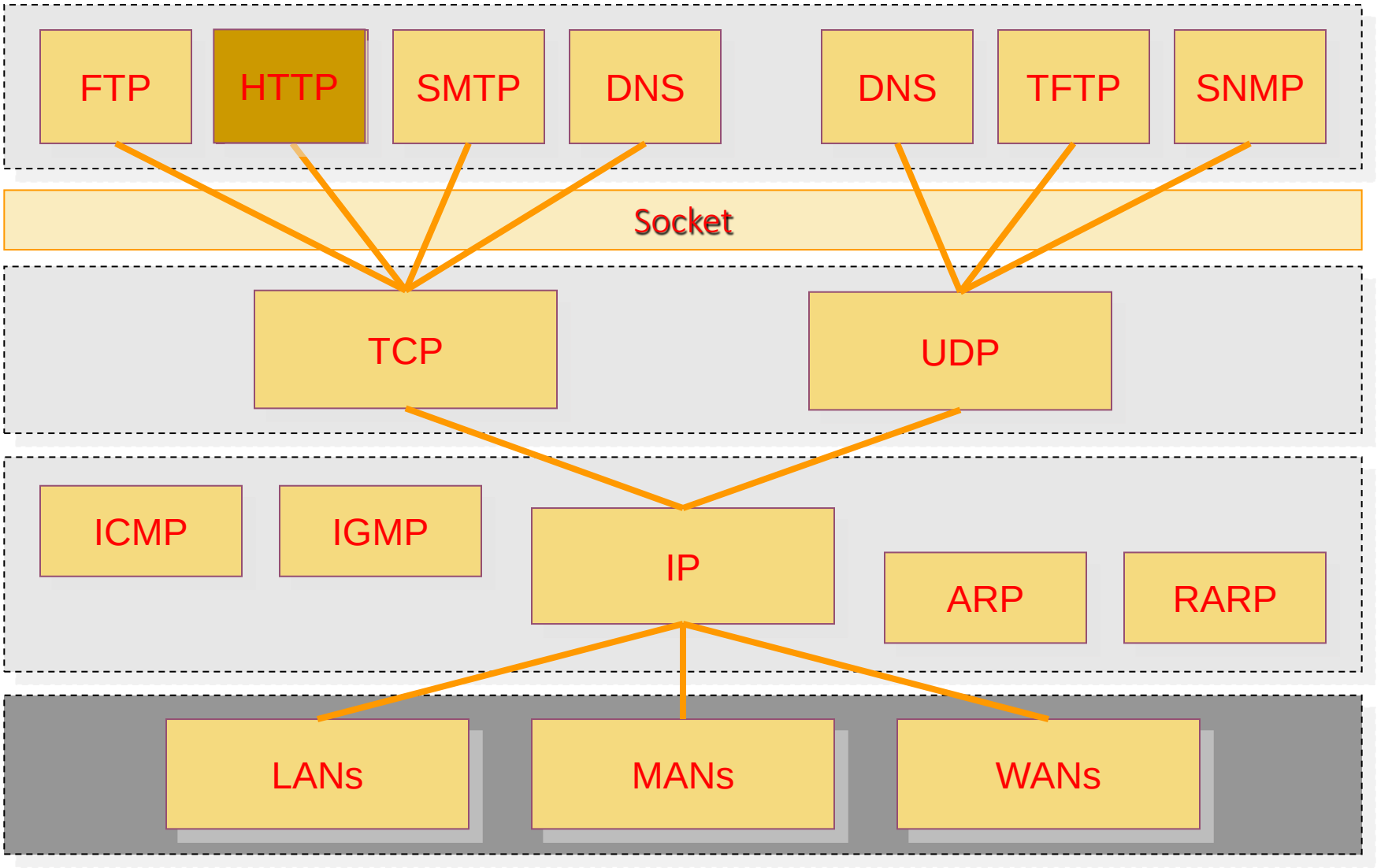
- HTTP：超文本传输协议（HyperText Transfer Protocol），基于TCP/IP之上的应用协议



HTTP协议通讯过程

1. 客户机与服务器建立连接。只要单击某个超级链接，HTTP 的工作开始。
2. 客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可能的内容。
3. 服务器接到请求后，处理并返回响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME 信息包括服务器信息、实体信息和可能的内容。
4. 客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

4.1.1 HTTP协议


FTP
HTTP
SMTP
DNS
DNS
TFTP
SNMP
Socket
TCP
UDP
ICMP
IGMP
IP
ARP
RARP
LANs
MANs
WANs

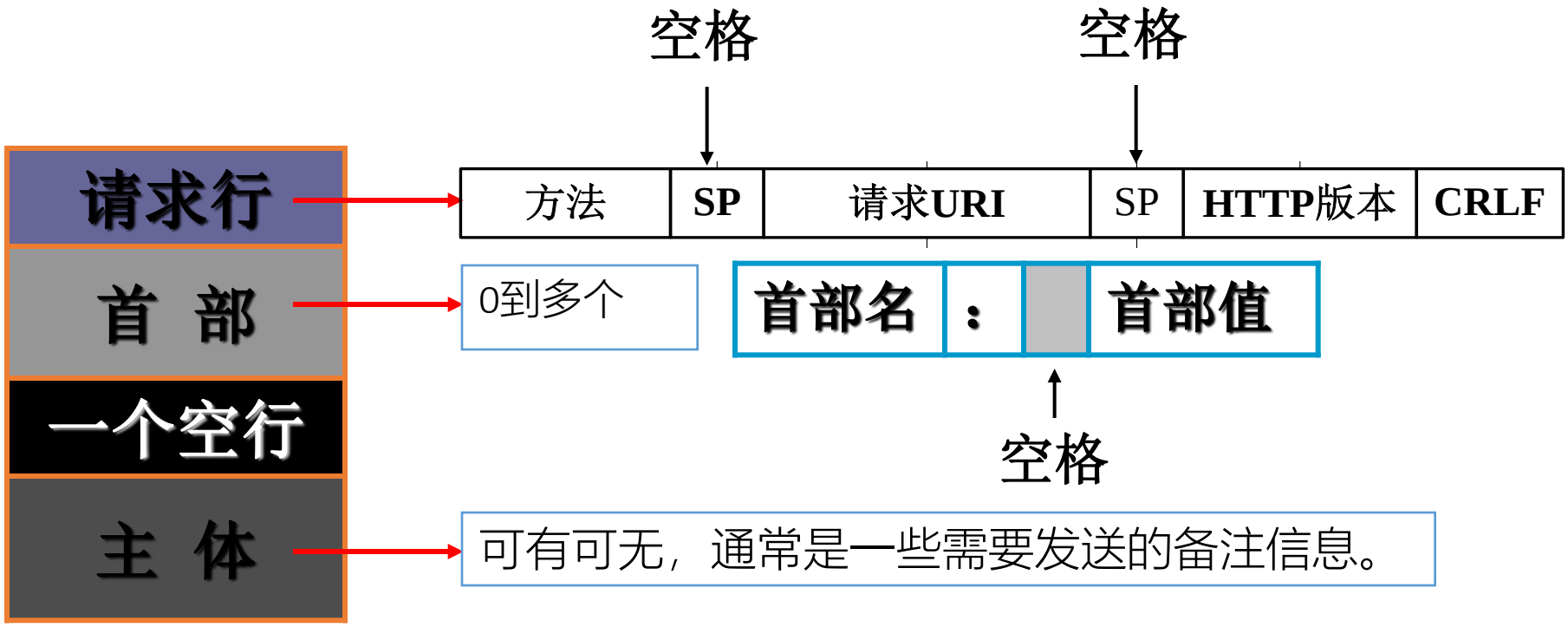
4.1.1 HTTP协议

- 无连接
  - 每次连接只处理一个请求
  - 服务器处理完客户的请求，并收到客户的应答后，即断开连接
- 无状态
  - 协议对于事务处理没有记忆能力
  - 如果后续处理需要前面的信息，则它必须重传
- 媒体独立的
  - 只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送
  - 客户端以及服务器指定使用适合的内容类型

4.1.2 HTTP消息

- 消息类型：
  - 请求：客户端->服务器端
  - 响应：服务器端->客户端
- 消息构成：
  - 请求/响应行
  - 消息头
  - 消息体

HTTP请求


空格
空格
请求行
方法
SP
请求URI
SP
HTTP版本
CRLF
首 部
0到多个
首部名
:
首部值
空格
一个空行
主 体
可有可无，通常是一些需要发送的备注信息。

HTTP请求

① 是请求方法，HTTP/1.1 定义的请求方法有8种。一般常用的是GET和POST。
② 为请求对应的URL地址，它和报文头的Host属性组成完整的请求URL
③ 是协议名称及版本号。
④ 是HTTP的报文头，包含若干个属性，格式为"属性名:属性值"，服务端据此获取客户端信息。
⑤ 是报文体，承载请求参数的数据等内容

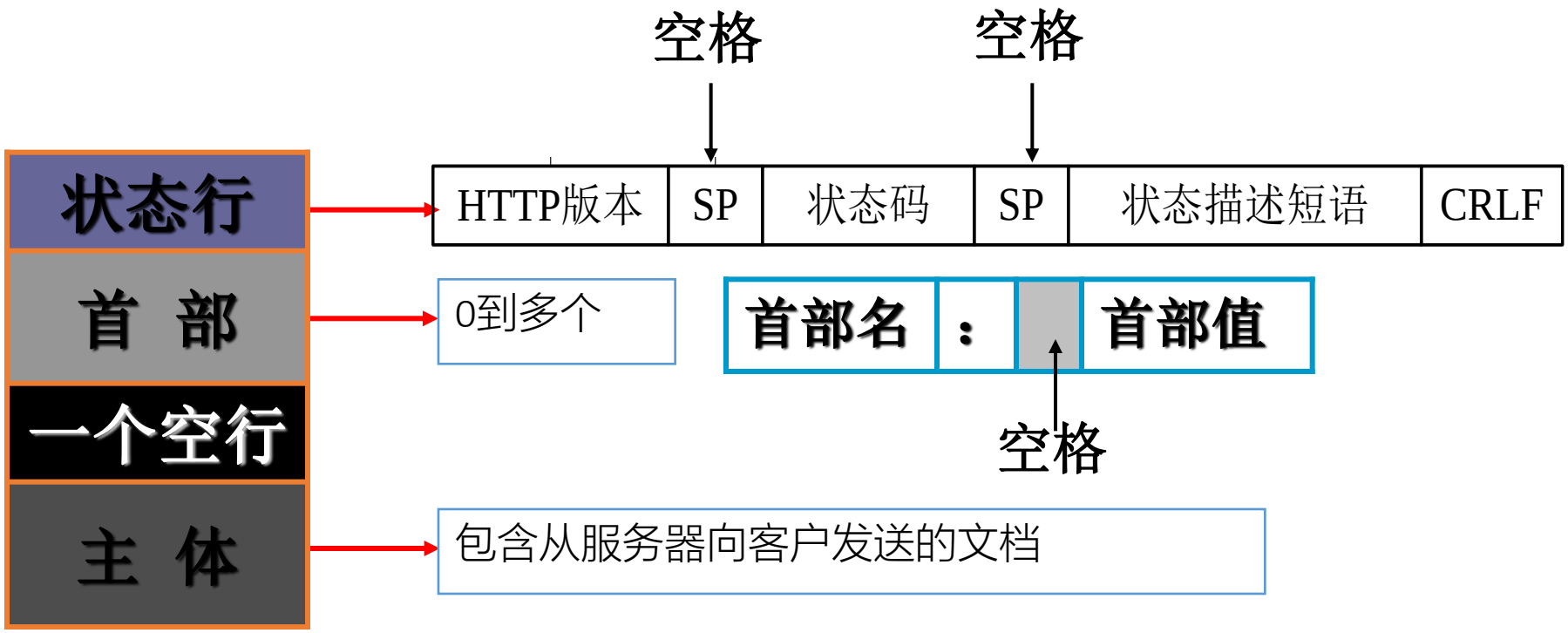
HTTP请求

方法名	备注
GET	获取一个URL指定的资源,即资源实体
POST	从客户端向服务器端发送一些信息（需要主体部分）
HEAD	只请求文档的首部信息，而不包含文档的内容
PUT	从服务器向客户端发送一些信息
DELETE	请求服务器删除指定的页面
TRACE	网络跟踪,允许客户端查看消息回收过程（用于测试）
CONNECT	与PROXY之间的连接管理
OPTIONS	查询能力,允许客户端查看服务器的性能

HTTP请求

首部字段	含义
User-agent	标志客户程序
Accept	客户端能够接受的媒体格式
Accept-charset	客户端能够处理的字符集
Accept-encoding	客户端能够处理的编码方案
Accept-language	客户端能够接受的语言
Authorization	客户端具有何种准许
From	用户的电子邮件地址
Host	客户端的主机和端口号
If-modified-since	只当比指明日期更加新时才发送这个文档
If-match	只当与给定标记匹配时才发送这个文档
If-not-match	只当与给定标记不匹配时才发送这个文档
If-range	只发送缺少的那部分文档
If-unmodified-since	若在指明日期之后未改变，则发送文档
Referrer	指明被链接的文档的URL
User-went	标志客户程序

HTTP响应

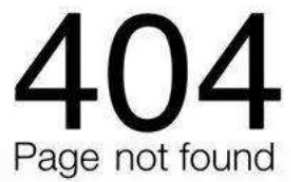


响应首部：指明服务器的配置或主体信息类型、长度、压缩方法、

HTTP响应

① 报文协议及版本；
② 状态码及状态描述；
③ 响应报文头，也是由多个属性组成；
④ 响应报文体，即我们真正要的“干货”。

HTTP响应



HTTP的响应状态码由5段组成：

```
1xx 消息，一般是告诉客户端，请求已经收到了，正在处理，别急...
2xx 处理成功，一般表示：请求收悉、我明白你要的、请求已受理、已经处理完成等信息.
3xx 重定向到其它地方。它让客户端再发起一个请求以完成整个处理。
4xx 处理发生错误，责任在客户端，如客户端的请求一个不存在的资源，客户端未被授权，禁止访问等。
5xx 处理发生错误，责任在服务端，如服务端抛出异常，路由出错，HTTP版本不支持等。
```

状态码	原因短语
200	正确 ok
201	创建 created
202	接收 accepted
204	无内容 no content
300	多种选择
301	永久移动
302	暂时移动
304	未被修改

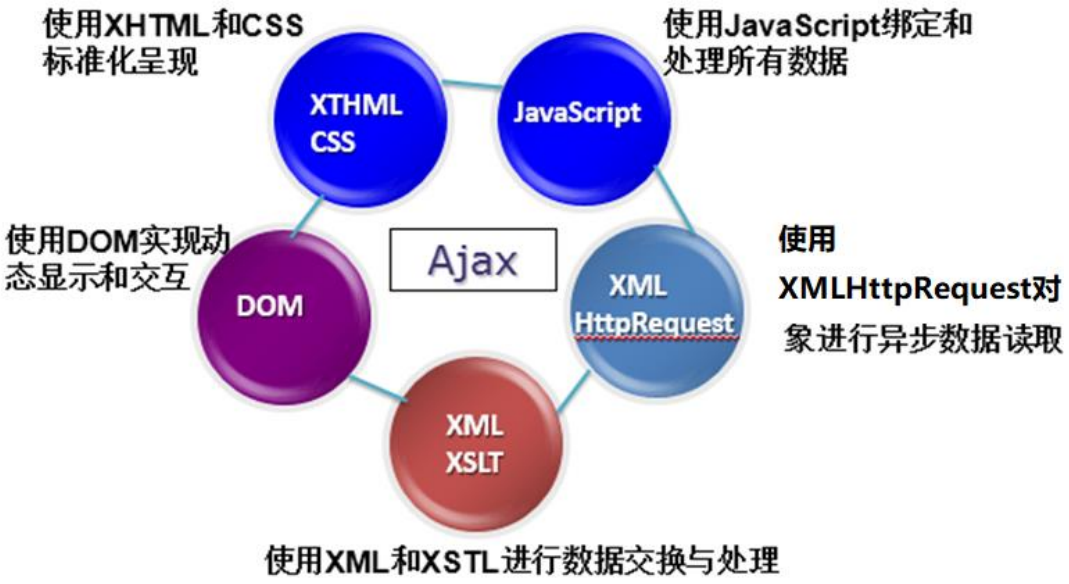
状态码	原因短语
400	错误请求
401	未授权
403	禁止
404	未发现
500	内部服务错误
501	未实现
502	错误网关
503	服务未提供

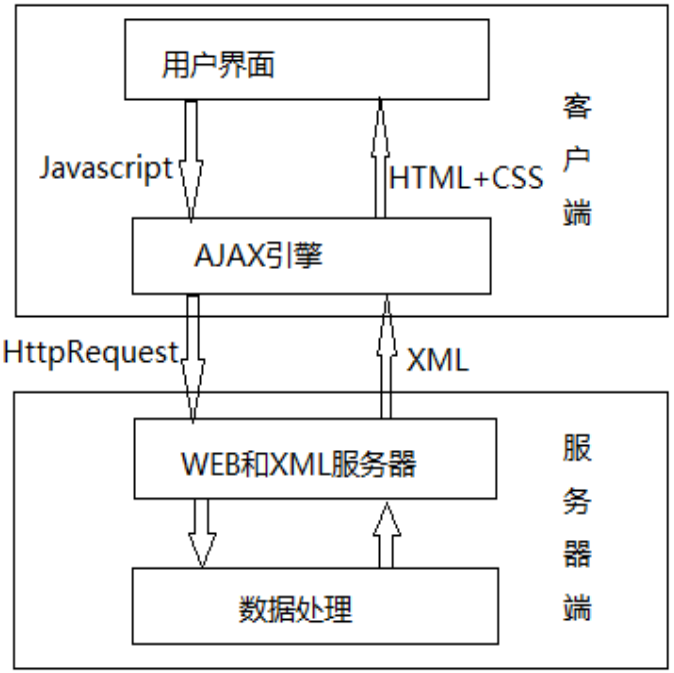
HTTP响应

首部字段	说明
Date	给出当前日期
Last-Modified	请求文档最近修改时间
Content-encoding	主体所用的编码
Content-length	数据长度（字节）
Content-type	数据类型
Accept-Range	服务器指明它能接受请求的字节范围
Age	表示发送者对响应（或重验证）在源服务器上产生以来的时间估计
Location	用于为完成请求或识别一个新资源，使接收者能重定向于由它指示的URI而不是请求URI
Proxy-Authenticate	必须包含在407响应中，由一个challenge和parameters组成，challenge指明认证方案，而parameters应用于此请求URI的代理
Retry-After	用于一个503响应，向请求端指明服务不可得的时长；用于3xx（重定向）响应，指明用户代理再次提交已重定向请求之前的最小等待时间
Server	包含源服务器用于处理请求的软件信息
WWW-Authenticate	必须包含在401响应中，字段值至少应该包含一个指明认证方案的callenge和适用于请求URI的参数，用于通知客户方需要的认证信息（如用户名、口令等）

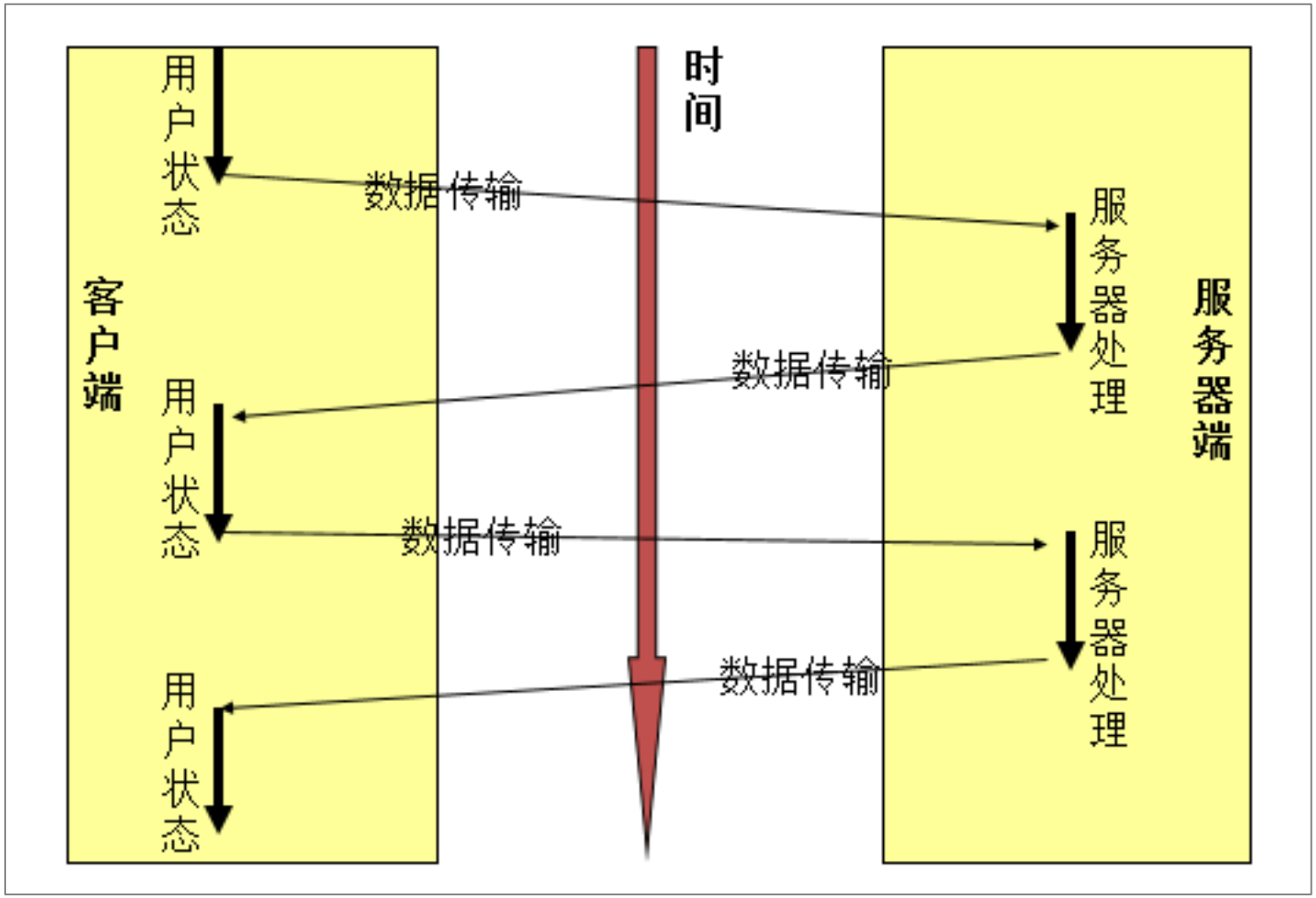
4.3 AJAX

- Ajax(Asynchronous JavaScript and XML)：异步的JavaScript和XML技术，是一种支持异步请求的技术

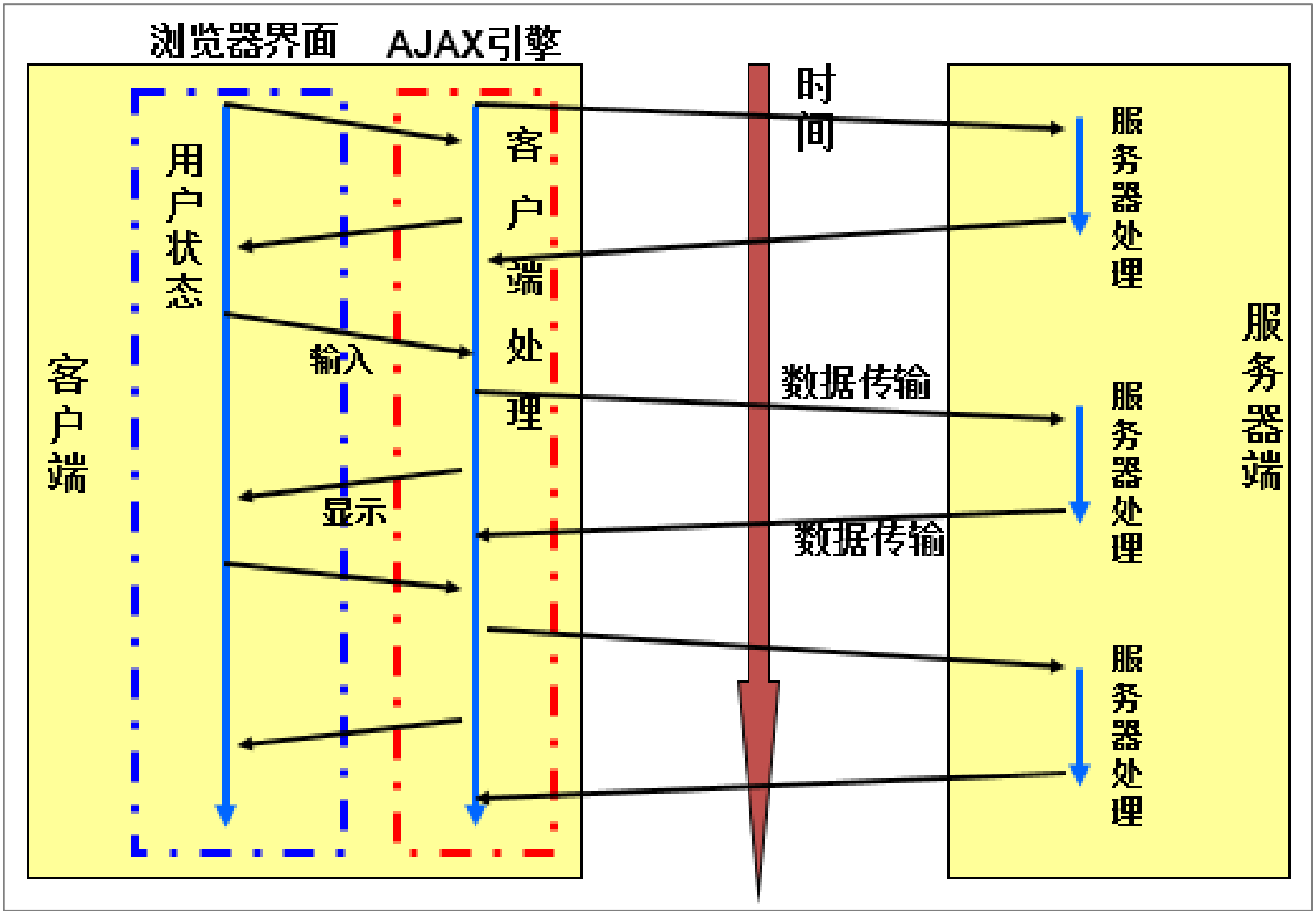




传统Web应用的工作模式

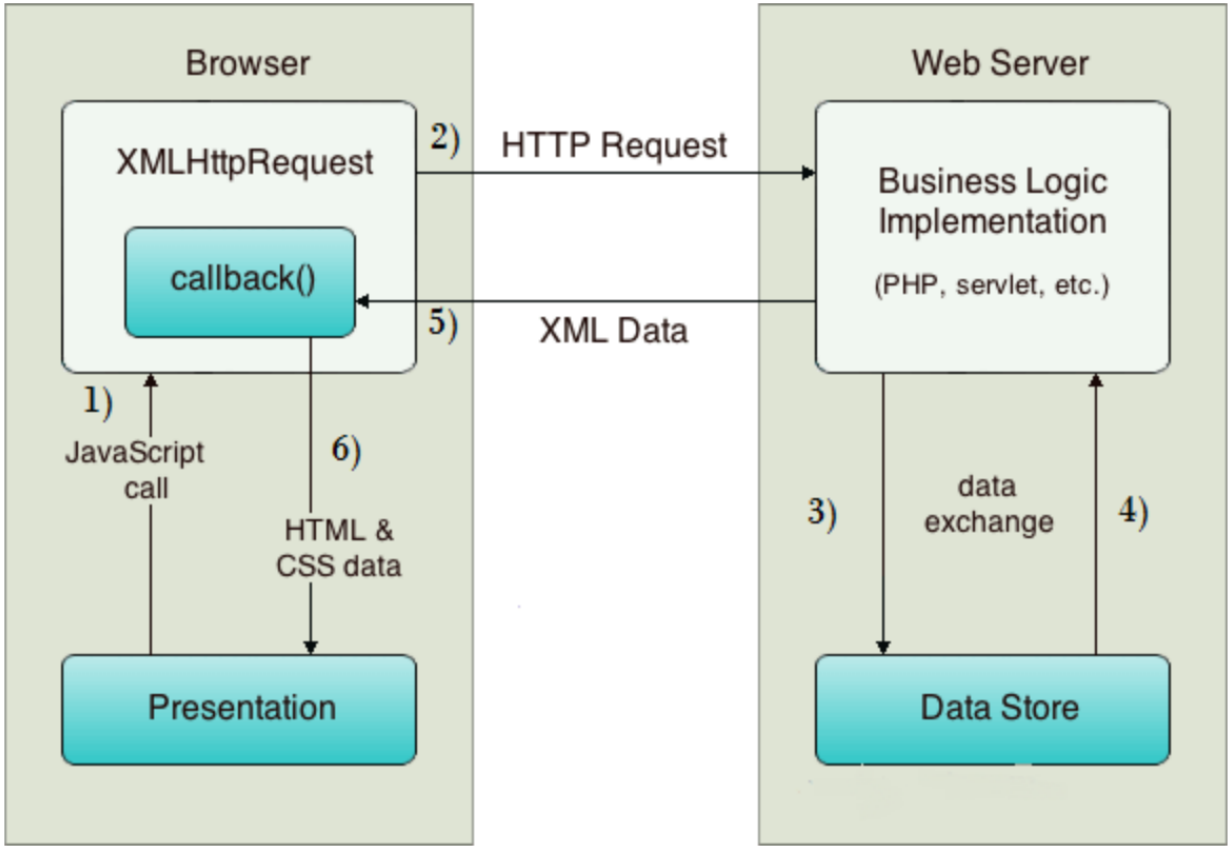


Ajax Web应用的工作模式



Ajax如何工作

1. 用户从 `UI` 发送请求， `JavaScript` 中调用 `XMLHttpRequest` 对象。
2. `HTTP` 请求由 `XMLHttpRequest` 对象发送到服务器。
3. 服务器使用 `JSP` ， `PHP` ， `Servlet` ，ASP.net 等与数据库交互。
4. 检索数据。
5. 服务器将 `XML` 数据或 `JSON` 数据发送到 `XMLHttpRequest` 回调函数。
6. `HTML` 和 `CSS` 数据显示在浏览器上。



XMLHttpRequest 对象

- XMLHttpRequest 是 AJAX 的基础。XMLHttpRequest 术语缩写为XHR，中文可以解释为可扩展超文本传输请求。
- XMLHttpRequest 对象可以在不向服务器提交整个页面的情况下，实现局部更新网页。
- XMLHttpRequest的对象用于客户端和服务器之间的异步通信。

属性	描述
onreadystatechange	存储函数（或函数名），每当 readyState 的属性改变时，就会调用该函数。
readyState	存有的 XMLHttpRequest 的状态从 0 到 4 发生变化。 0：请求未初始化 1：服务器连接已建立 2：请求已接收 3：请求处理中 4：请求已完成，且响应已就绪
responseText	以文本形式返回响应。
responseXML	以 XML 格式返回响应
Status	将状态返回为数字（例如，“Not Found”为 404，“OK”为 200）
statusText	以字符串形式返回状态（例如，“Not Found”或“OK”）

方法	描述
abort()	取消当前请求。
getAllResponseHeaders()	以字符串形式返回完整的 HTTP 标头集。
getResponseHeader(headerName)	返回指定 HTTP 标头的值。
void open（method，URL）	打开指定获取或交的方法和 URL 的请求。
void open（method，URL，async）	与上面相同，但指定异步或不。
void open (method, URL, async, userName, password)	与上面相同，但指定用户名和密码。
void send（content）	发送获取请求。
setRequestHeader（label，value）	将标签/值对添加到要发送的 HTTP 标头。

AJAX XHR-请求

`XMLHttpRequest` 对象用于和服务器交换数据。

当你的页面全部加载完毕后，客户端会通过 `XMLHttpRequest` 对象向服务器请求数据，服务器端接受数据并处理后，向客户端反馈数据。

如需将请求发送到服务器，我们使用 `XMLHttpRequest` 对象的 `open()` 和 `send()` 方法：

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

方法	描述
open(*method*,*url*,*async*)	规定请求的类型、URL 以及是否异步处理请求。 • *method*：请求的类型；GET 或 POST • *url*：文件在服务器上的位置 • *async*：true（异步）或 false（同步）
send(*string*)	将请求发送到服务器。 • *string*：仅用于 POST 请求

GET请求与POST请求

- GET请求的特点：

```
xmlhttp.open("GET","demo_get2.html?fname=Henry&lname=Ford",true);
xmlhttp.send();
```

  - GET请求可被缓存
  - GET请求保留在浏览器历史记录中
  - GET请求可被收藏为书签
  - GET请求不应在处理敏感数据时使用
  - GET请求有长度限制
  - GET请求只应当用于取回数据

- POST请求的特点：

```
xmlhttp.open("POST","ajax_test.html",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

  - POST请求不会被缓存
  - POST请求不会保留在浏览器历史记录中
  - POST请求不能被收藏为书签
  - POST请求对数据长度没有要求

同步请求与异步请求

Async=true

当使用 `async=true` 时，请规定在响应处于 `onreadystatechange` 事件中的就绪状态时执行的函数：

实例

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

同步请求与异步请求

Async = false

如需使用 `async=false` ，请将 `open()` 方法中的第三个参数改为 false：

```
xmlhttp.open(&amp;quot;GET&amp;quot;,&amp;quot;test1.txt&amp;quot;,false);
```

我们不推荐使用 `async=false` ，但是对于一些小型的请求，也是可以的。

记住，JavaScript 会等到服务器响应就绪才继续执行。若服务器繁忙或缓慢，应用程序会挂起或停止。

注意：当使用 `async=false` 时，请不要编写 `onreadystatechange` 函数 - 把代码放到 `send()` 语句后面即可：

实例

```
xmlhttp.open("GET","ajax_info.txt",false);
xmlhttp.send();
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

AJAX XHR-响应

由于 HTTP 响应是由服务端发出的，并且服务器做出响应需要时间（比如网速慢等原因），所以我们需要监听服务器响应的状态，然后才能进行处理。

- **状态行**

  `xhr.status` 状态码，如200，304，404等；

- **响应主体**

  `xhr.responseText` 与 `xhr.responseXML` 都表示响应主体。

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 `responseText` 或 `responseXML` 属性。

属性	描述
responseText	获得字符串形式的响应数据。
responseXML	获得 XML 形式的响应数据。

AJAX XHR-响应

responseText 属性

如果来自服务器的响应并非 XML，请使用 `responseText` 属性。

`responseText` 属性返回字符串形式的响应，因此您可以这样使用：

实例

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

提示：对于 `responseText` 属性，只有当 `readyState` 属性值变为4时，`responseText` 属性才可用，因为这表明AJAX请求已经结束！

AJAX XHR-响应

responseXML 属性

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 `responseXML` 属性：

实例

请求 cd_catalog.xml 文件，并解析响应：

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
  {
  txt=txt + x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("myDiv").innerHTML=txt;
```

AJAX XHR-readyState

当发送一个请求后，客户端需要确定这个请求什么时候会完成，因此，XMLHttpRequest对象提供了 `onreadystatechange` 事件机制来捕获请求的状态，继而实现响应。

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 `readyState` 改变时，就会触发 `onreadystatechange` 事件。

`readyState` 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

属性	描述
`onreadystatechange`	存储函数（或函数名），每当 `readyState` 属性改变时，就会调用该函数。
`readyState`	存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。 • 0: 请求未初始化 • 1: 服务器连接已建立 • 2: 请求已接收 • 3: 请求处理中 • 4: 请求已完成，且响应已就绪
`status`	200: "OK" 404: 未找到页面

AJAX XHR-readyState

readyState状态说明

0：请求未初始化

此阶段确认XMLHttpRequest对象是否创建，并为调用 `open()` 方法进行未初始化作好准备，值为0表示对象已经存在，否则浏览器会报错：对象不存在。

1：服务器连接已建立

此阶段对XMLHttpRequest对象进行初始化，即调用 `open()` 方法，根据参数 `（method,url,true）` 完成对象状态的设置。并调用 `send()` 方法开始向服务端发送请求。

值为1表示正在向服务端发送请求。

2：请求已接收

此阶段接收服务器端的响应数据。但获得的还只是服务端响应的原始数据，并不能直接在客户端使用。

值为2表示已经接收完全部响应数据，并为下一阶段对数据解析作好准备。

3：请求处理中

此阶段解析接收到的服务器端响应数据即根据服务器端响应头部返回的MIME类型把数据转换成能通过 `responseBody` ， `responseText` 或 `responseXML` 的属性存取的格式，为在客户端调用作好准备。

状态3表示正在解析数据。

4：请求已完成，且响应已就绪

此阶段确认全部数据都已经解析为客户端可用的格式，解析已经完成。值为4表示数据解析完毕，可以通过的XMLHttpRequest对象的属性取得数据。

AJAX XHR-readyState

在 `onreadystatechange` 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 `readyState` 等于 4 且状态为 200 时，表示响应已就绪：

实例

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
```

实例：通过AJAX获取微人大验证码及其验证结果


```php
<?php
  /* pannel.php: 面板页面 */
  session_start();
  if (isset($_SESSION["username"])) {
    $role = $_COOKIE[“role”]; //安全问题?
    echo "你好，".$role."<br/>";
    echo "<a href='logout.php'>退出</a>";
  } else echo "无权访问！";
?>
```

```php
<?php
  /* logout.php: 退出页面 */
  session_start();    // 启用Session
  session_unset();    // 销毁$_SESSION数组
  session_destroy(); // 销毁Session
  header("Location: index.php");
?>
```

```php
<?php
  /* index.php: 首页 */
  session_start(); // 启用Session
  if (isset($_SESSION["username"])) { // 判断是否处于成功登录状态
    header("Location: pannel.php"); // 跳转到面板页面
    exit(); // 终止后续代码的执行
  }
?>
<form action="login.php" method="POST" id="login-form">
<?php $_SESSION["code"] = rand()%10; ?>
  <img src="img/code_<?=$_SESSION["code"]?>.png">
</form>
```

```php
<?php
  /* login.php: 登录页面 */
  session_start();                          // 启用Session
  $username = $_POST["username"];           // 获取表单中的用户名
  $password = $_POST["password"];           // 获取表单中的密码
  $code = $_POST["code"];                   // 获取表单中的验证码

  //维护一个验证码正确值和验证码图片下标的映射
  $checks = ["pupr", "khrb", "huks", "egwb", "ekdv", "khns", "wuxc", "tcyk", "nupf", "brpk"];
  $correct_code = $checks[$_SESSION["code"]];

  if ($code == $correct_code) {
    if ($username == "admin001" && $password == "admin001")                  // 管理员
      $_SESSION["role"] = "admin";                                           // 在Session中记录角色
    else if ($username == "teacher001" && $password == "teacher001")         // 教师
      $_SESSION["role"] = "teacher";                                         // 在Session中记录角色
    else if ($username == "student001" && $password == "student001")         // 学生
      $_SESSION["role"] = "student";                                         // 在Session中记录角色

    if (isset($_SESSION["role"])) {                                                  // 登录成功
      $_SESSION["username"] = $username;                                     // 在Session中记录用户名
      setcookie("role",$_SESSION["role"], 0, "/", "", false, true);          // 设置名为“role”的Cookie项
      header("Location: pannel.php");                                        // 跳转到用户面板
    } else echo "用户名或密码不正确！";                                      // 错误提示
  } else echo "验证码不正确！";                                              // 错误提示
?>
```

目录

实践项目：简易匿名墙开发

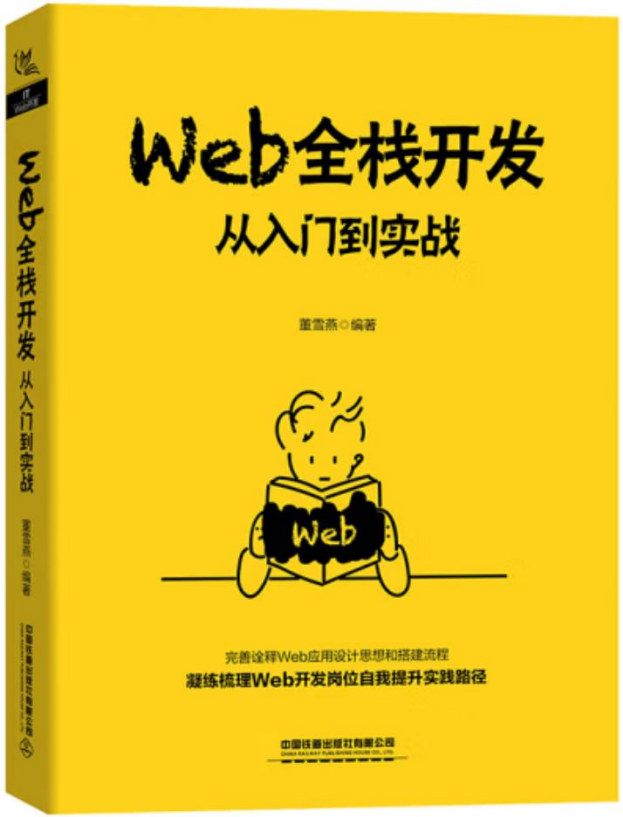
任务1：模仿实现匿名墙的静态页面展示效果（HTML+CSS）

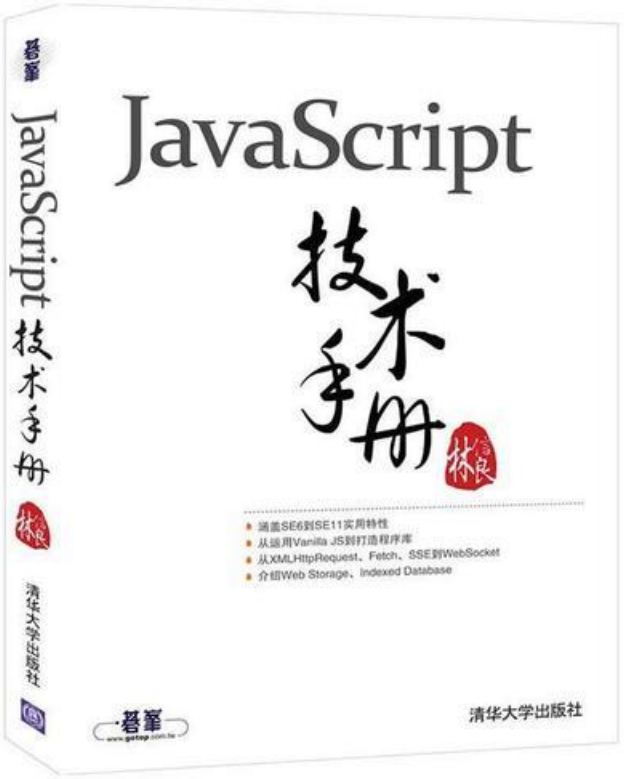
任务2：前端页面动态交互（JavaScript+BOM/DOM+Event）

任务3：后端数据存取（PHP+MySQL）

https://www.xiaoyuantongxing.com/treehole/20220429RUC?code=xxx

推荐图书


Web全栈开发
从入门到实战
董雪燕◎编著
Web
完善诠释Web应用设计思想和搭建流程
凝练梳理Web开发岗位自我提升实践路径
中国铁道出版社有限公司


JavaScript
技术手册
涵盖SE6到SE11实用特性
从运用Vanilla JS到打造程序库
从XMLHttpRequest、Fetch、SSE到WebSocket
介绍Web Storage、Indexed Database
清华大学出版社


第2版
PHP核心技术与最佳实践
PHP Core Technology and Best Practice, Second Edition
PHP核心技术
与最佳实践
机械工业出版社
China Machine Press

推荐课程

- 《网络空间安全引论》秋季学期 => Crypto、Misc
- 《Web安全》春季学期 => Web
- 《程序设计安全》秋季学期 => Pwn
- 《软件安全分析》春季学期 => Reverse

信息安全专业选修课

CTF赛题类型