

Web 基础

2024.7.18

课程目标

- 理解 Web 通信的过程以及其中的重要概念
- 了解 Web 页面的构成
- 初步了解 Web 应用架构，了解各个模块存在的意义

网页是文件

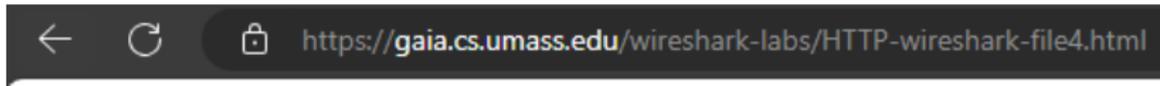
客户（浏览器）通知服务器，自己想要文件（**请求**）

被通知的服务器向客户发送一些文件（**响应**）

客户（浏览器）收到文件，按照一定的规则组装好

我们看到浏览器显示的网页

准确地定位请求



向谁请求？ 请求什么文件？

用网址 ⇒ URL(Uniform Resource Locator) 定位

协议://主机名/路径

协议://主机名 [: 端口号]/路径 [? 查询]

Web 的核心: HTTP

HTTP(HyperText Transfer Protocol): 超文本传输协议

定义请求的方式和响应的方式

首先由客户发起请求 (**Http request**), 正文可携带信息
服务器进行响应 (**Http response**), 正文可携带文件

- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

HTTP request: 请求方法

用不同请求方法完成不同任务

- GET: 获取资源
- POST: 提交数据
- PUT: 更新/替换服务器资源
- DELETE: 删除服务器资源

可以想一想完成下面的任务应该用什么请求方法

- 访问一个网页
- 提交自己的用户名和密码
- 更新自己的头像
- 注销自己的账户

理论上大部分请求方法是可以混用的，但是这并不好

HTTP request: 参数传递

请求如何携带信息，比如说用户名和密码？

- 路径传参：

http://www.site.com/info/aaa/head.png

- 查询字符串传参：

http://www.site.com/info?username = aaa&require = head

- 请求体传参：

http://www.site.com/info

```
1   {
2       "username": "aaa",
3       "telephone": "12345678901"
4   }
```

JSON 是一种常用的数据交换格式

[拓展]HTTP request: 身份验证

用户无关 ⇒ 用户相关: 需要身份验证
当然... 可以直接发验证码!

一种验证方法: Cookies
请求中自动携带 Cookie 信息

Cookies 带来的安全性问题



HTTP response: 状态码与正文信息

服务器收到请求后（进行一些操作，然后）返回响应

响应没有分类，但是有状态码表示响应的结果

常见状态码：

- 200 OK: 请求成功
- 400 Bad Request: 请求错误
- 403 Forbidden: 没有权限访问
- 404 Not Found: 请求的资源未找到
- 500 Internal Server Error: 服务器内部错误

响应也和请求一样有正文部分。响应的正文中可以携带用户请求的文件

一个简化流程：网上查询选课结果

现在请求和响应都已经介绍过了，我们用一个例子来再体会一遍整个的流程。这是一个网上选课的例子。

1. 用户访问选课系统，客户请求 `http://www.course.com`，请求方法为 GET
2. 服务器返回对应文件，状态码为 200 OK，浏览器显示页面
3. 用户输入用户名和密码登录，客户请求 `http://www.course.com/login`，请求方法为 POST，请求体中携带用户名和密码
4. 服务器收到请求，验证发现密码正确，服务器去数据库中找到了用户的课表，携带在响应的正文中返回

小结



两个实体：客户和服务端

两种消息：HTTP 请求和 HTTP 响应

请求方法：GET、POST、PUT、DELETE 等

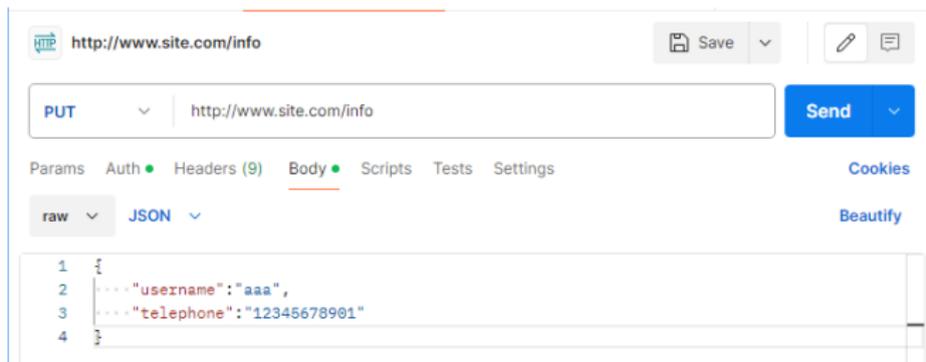
响应状态码

请求与响应的正文

HTTP 调试

如果想手动发送请求呢？

可以使用 Postman/Apifox



[拓展]HTTPS：“安全”的 HTTP

HTTP Secure：超文本传输**安全**协议

HTTP 的不安全之处：信息明文传递、无法证明信息完整、无法验证身份

HTTPS 的针对性改进：HTTP+SSL/TLS

HTTPS 相比 HTTP 的不足

常用 HTML 标签

在 HTML 标签中，一部分标签用于对文本进行修饰

标签	描述
< <i>h1</i> > – < <i>h6</i> >	定义标题，从最高级别到最低级别
< <i>p</i> >	定义段落
< <i>ul</i> >	定义无序列表
< <i>ol</i> >	定义有序列表
< <i>li</i> >	定义列表项
< <i>table</i> >	定义表格
< <i>tr</i> >	定义表格的行
< <i>td</i> >	定义表格的单元格
< <i>th</i> >	定义表格的表头单元格
< <i>strong</i> >	定义加粗的文本
< <i>em</i> >	定义斜体的文本
< <i>br</i> >	插入换行符

常用 HTML 标签

HTML 标签还可以用于表示文本之外的内容：

- 图片、超链接等网页上文本之外的内容
- 网页标题等其他内容

标签	描述
<code>< html ></code>	定义 HTML 文档的根元素
<code>< head ></code>	定义文档的头部区域
<code>< body ></code>	定义文档的主体区域
<code>< img ></code>	插入图像
<code>< a ></code>	创建链接
<code>< div ></code>	定义一个区块或容器
<code>< span ></code>	标记或包装文本片段
<code>< header ></code>	定义页眉部分
<code>< style ></code>	定义内部样式表

HTML 实例

```
1 <!DOCTYPE html>
2 <!-- 文档类型声明 -->
3 <html>
4 <head>
5     <title>Basic Example</
      title>
6 </head>
7 <body>
8     <h1>最高级别标题</h1>
9     <h2>第二级别标题</h2>
10    <p>这是一段文本，<strong>
      >里面有一些加粗的文
      字</strong></p>
11    <hr>
12    
13    <!-- 展示一张图片 -->
14 </body>
15 </html>
```



大家可以照着刚才的例子自己尝试写一个简单的 HTML 文件，保存为 xxx.html，然后尝试用浏览器打开

更详细的介绍可以参考菜鸟教程：

<https://www.runoob.com/html/html-tutorial.html>

HTML 标签的属性

HTML 标签可以有属性，属性用于提供附加信息

属性一般以名称-值对（比如 name="value"）的形式出现，放在开始标签中

- class: 为元素定义类名
- id: 为元素定义唯一标识符
- style: 为元素定义样式
- src: 用于链接外部资源
- href: 超链接，用于链接外部网页

比如:

```
1   
2 <div id="main" class="container" style="color: red;">
```

div 标签

如何更好对页面进行“排版”？

<div> 标签定义 HTML 文档中的一个分隔区块或一个区域
自身没有语义，仅作为容器用于对内容进行分区 (division)
“提供一个可以定义 style 的开始标签”

- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

CSS 简介

- 我们需要美化网页
- 比如... 把一些文字变成蓝色?
- 在某一段文字的开始标签中添加 style 属性
- 在头部添加 style 标签

CSS: 加入 style 属性

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Basic Example</
   title>
5 </head>
6 <body>
7   <h1 style="color:
   royalblue;">最高级别
   标题</h1>
8   <h2>第二级别标题</h2>
9   <p>这是一段文本, <strong>
   >里面有一些加粗的文
   字</strong></p>
10  <hr>
11  
12 </body>
13 </html>

```



CSS: 在头部加入 style 标签

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Basic Example</
   title>
5   <style>
6     h1 {
7       color:blue;
8     }
9   </style>
10 </head>
11 <body>
12   <h1>最高级别标题</h1>
13   <h1>第二个最高级别标题</
    h1>
14 </body>
15 </html>
```



CSS: 抽离成文件

同一个网站不同页面的格式往往有相同的地方
我们实现了在一个文件内只定义一次，那如果是多个文件呢？

将之前定义的内容提取出来成为 **CSS(Cascading Style Sheets)**
文件

HTML+CSS

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Basic Example</
   title>
5 </head>
6   <link rel="
   stylesheet" href
   ="style.css">
7 </head>
8 </head>
9 <body>
10  <h1>最高级别标题</h1>
11  <h1>第二个最高级别标题</
   h1>
12  <hr>
13  
14 </body>
15 </html>
```

```
1 h1 {
2   color:blue;
3 }
```

Listing 2: style.css



[拓展]CSS 的 Cascading

如果有两个 CSS 文件对 h1 进行了不同的规定，以哪一个为准？

如果 CSS 文件和 style 标签/style 属性进行了不同的规定，以哪一个为准？

如果外层的 style 属性和内层的 style 属性进行了不同的规定，以哪一个为准？

- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

能否引入动态?

选课系统：点击按钮后提示选课成功

网页没有动态：重新请求

JavaScript(JS) 可以帮我们实现这样的效果

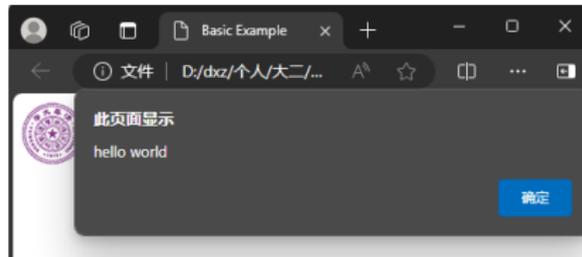
JavaScript 简介

- JavaScript 是一种脚本语言, 不用编译, 可直接嵌入 HTML 运行
- JavaScript 可以操纵 HTML 网页内部的元素
- JavaScript 在很多地方都和 C++ 很像
- JavaScript 也可以抽取为单独的文件
- JavaScript 和 Java 并没有什么关系

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Basic Example</
      title>
5 </head>
6 <body>
7     
10    <!-- 展示一张图片 -->
11 </body>
12 </html>

```



综合实例：HTML+CSS+JavaScript

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <<I<title>Advanced Example</
  title>
5 <link rel="stylesheet"
  href="style.css">
6 </head>
7 <body>
8 <button id="myButton"
  class="my_Button">
  Click me</button>
9
10 <script src="script.js
  "></script>
11 </body>
12 </html>

```

Listing 3: page.html

```

1 .my_Button {
2   padding: 10px, 20px;
3   background-color: blue;
4   color:white;
5   border:none;}

```

Listing 4: style.css

```

1 let button=document.
  getElementById("myButton
  ");
2 button.addEventListener("
  click",function(){
3   button.style.
4     backgroundColor="red
      ";
  });

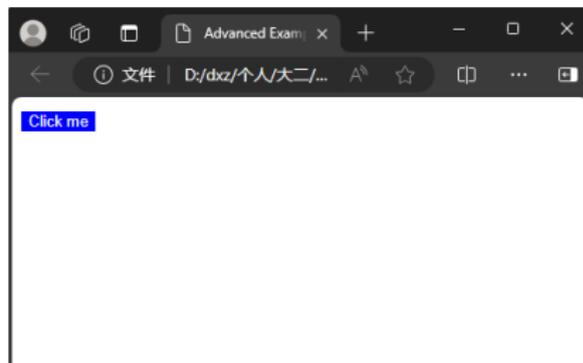
```

Listing 5: script.js

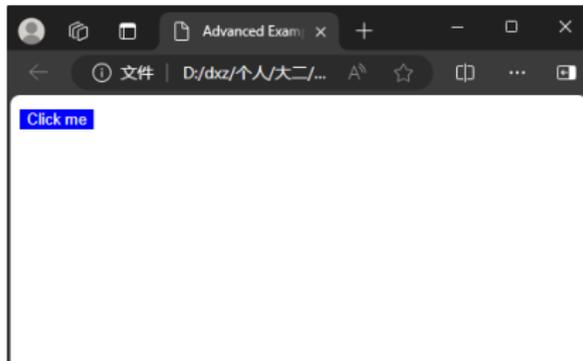
综合实例：HTML+CSS+JavaScript

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Advanced Example</
   title>
5   <link rel="stylesheet"
   href="style.css">
6 </head>
7 <body>
8   <button id="myButton"
   class="my_Button">
   Click me</button>
9
10  <script src="script.js
   "></script>
11 </body>
12 </html>
```

Listing 6: page.html



综合实例：HTML+CSS+JavaScript



```
1 .my_Button {
2   padding: 10px, 20px;
3   background-color: blue;
4   color:white;
5   border:none;}
```

Listing 7: style.css

```
1 let button=document.
   getElementById("myButton
   ");
2 button.addEventListener("
   click",function(){
3   button.style.
       backgroundColor="red
       ";
4 });
```

Listing 8: script.js

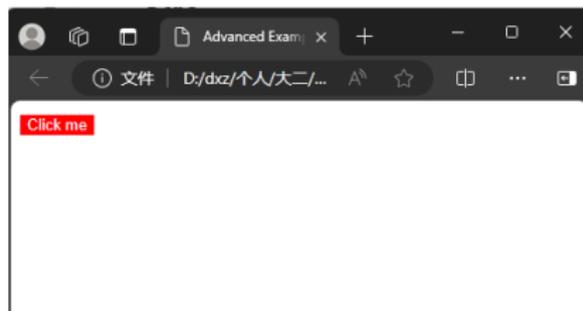
综合实例：HTML+CSS+JavaScript

```
1 .my_Button {
2   padding: 10px, 20px;
3   background-color: blue;
4   color:white;
5   border:none;}
```

Listing 9: style.css

```
1 let button=document.
   getElementById("myButton
   ");
2 button.addEventListener("
   click",function(){
3   button.style.
       backgroundColor="red
       ";
4 });
```

Listing 10: script.js



引入 CSS 在 head 中，而引入
JS 在 body 中

[拓展] 组件库

- element UI: <https://element.eleme.io/>
- bootstrap: <https://getbootstrap.com/>

- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

Web 开发工具

- Google Chrome 或其他使用 Chromium 内核的浏览器 (比如 MS Edge)
- 可用 F12/fn+F12/右键检查
- 可用 Ctrl+S 保存网页



- 百度热搜 > ○ 换一换
- 1 总书记的一本笔记
 - 2 八所海事局：北部湾海域有四艘沉船
 - 3 我国大潮钻探探寻青藏高原自然密码
 - 4 刘德华演唱会嗓子哑 多次挥手示意
 - 5 湖南一厅官被二次举报论文抄袭

静态网页

- 回想 URL：服务器只需要查找？
- 静态网页：网页的内容已经确定且不再变化
- 动态网页：可能有确定的框架，但具体内容根据用户动态生成
- 如果是静态网站：服务器确实只需要查找并把请求的文件发回去
- 所以如果是静态网站，“智慧”主要在客户端体现

静态网页的服务器端

- 由于不需要进行逻辑上的处理，一个通用的服务端软件就可以满足需求
- 把文件都准备好放在特定路径下，有客户请求就发

确实非常省事，但这样就能满足所有需求了吗？

- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

动态网页



- 服务器可能有一个模板文件
- 服务器收到请求后根据请求的内容读取对应的数据
- 服务器将内容填充到模板中形成文件
- 服务器把文件放在响应的正文中返回给客户

服务器拥有了“智慧”！

看起来不错...

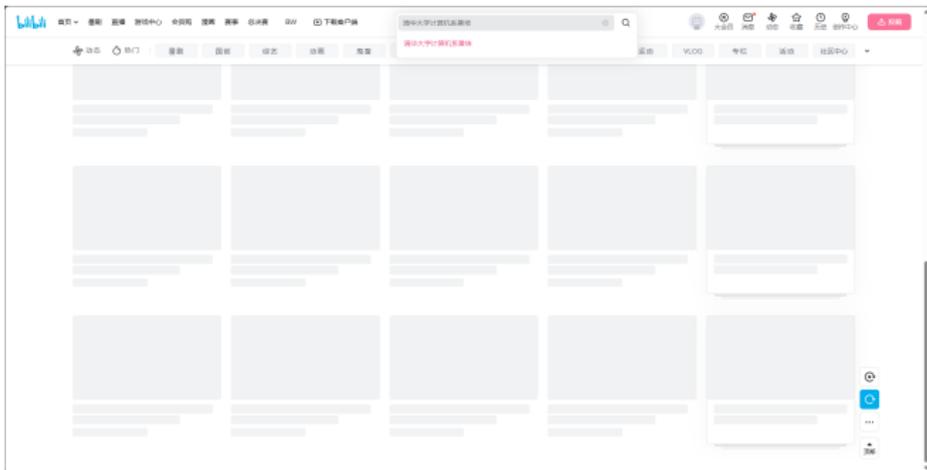
- 1 简介
- 2 Web 通信
 - HTTP request
 - HTTP response
- 3 Web 页面
 - HTML
 - CSS
 - JavaScript
 - 页面调试
- 4 Web 应用基础
 - 静态网页
 - 动态网页
 - 前后端分离的开发模式
 - 持久化存储

前后端分离

- 能不能“平衡一下工作量”？
- 前后端分离：把填充的任务交给客户端
- 前端：网站的前台，即展现给用户的网页
- 后端：用户可能看不见，实现具体逻辑、功能
- 将模板和内容（如图片）分开存储，实现前后端分离

简化前后端分离例子：B 站搜暑假培

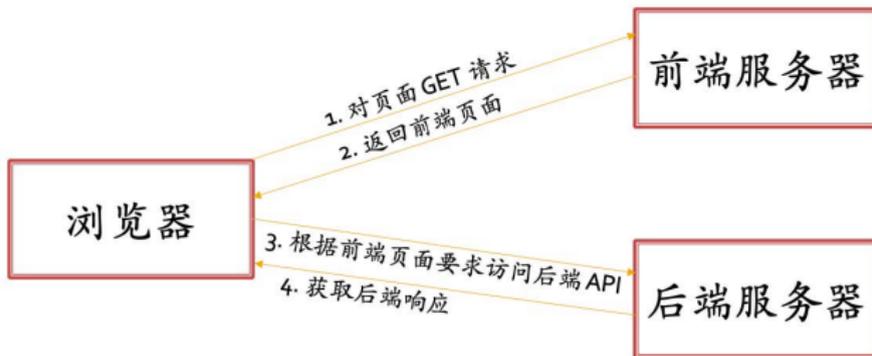
1. 用户输入关键词进行搜索，向前端服务器请求
2. 前端服务器返回模板，我们看到框架



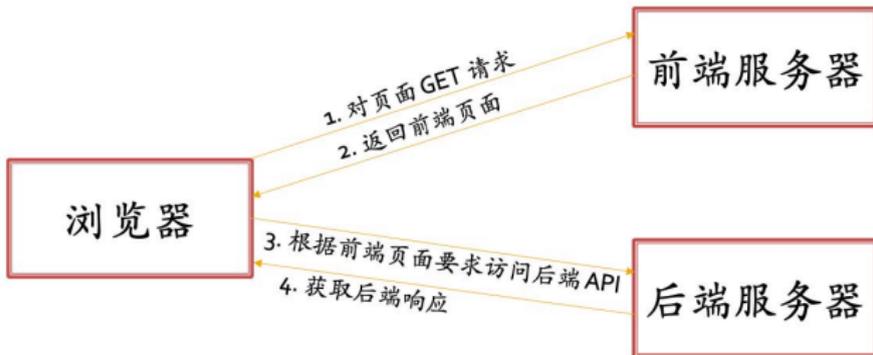
简化前后端分离例子：B 站搜暑假培

1. 用户输入关键词进行搜索，向前端服务器请求
2. 前端服务器返回模板，我们看到框架
3. 网页上 JS 脚本向后端服务器请求数据
4. 客户端收到响应后进行填充，我们看到具体内容
 - 前端服务器和后端服务器可以不是一台
 - 传输数据的格式由开发者规定，称为 API
 - 另一个好处：用户体验

前后端分离小结



[拓展] 前后端分离带来的安全问题



前端服务器返回内容中的 JS 决定向后端服务器请求的内容
如果攻击者修改了 JS...

数据如何保存？

- 比如：希望查看历史记录
- ——可以把数据保存在**服务器端**
- 但如果是希望记录“上次看到”呢？
- ——有些数据需要保存在**客户端**

感谢大家的倾听!

希望大家上完课后有所收获!

祝大家学业、生活一切顺利!

参考资料

- 《计算机网络：自顶向下方法》第八版
- 计算机系学生科协技能引导文档：<https://docs.net9.org/>
- 2023 年暑培 Web 基础课程
- 其他一些知乎上的文章

