

# Web 与 Web 应用基础

## 酒井科协暑培 2023

李轶凡

清华大学计算机系学生科协

2023 年 7 月 11 日

① 前言

② Web 页面

③ Web 通信

④ Web 应用基础

⑤ 作业说明

1 前言

## Browser/Server

网页是文件

## ② Web 页面

### ③ Web 通信

## 4 Web 应用基础

## 教学目标

- 理解浏览器与网站服务器的工作方式
  - 能上手实践，搭建简单的静态网站
  - 对 Web 应用架构有初步了解，明白各个模块的存在意义
  - 对于选择了 Rust 小学期的同学，可以选取合适合自己的功能点、较为轻松地上手 OJ 大作业

注意事项

- 不会涉及具体的语言细节，如 Rust 中该使用什么框架
  - 对于标注 [进阶] 部分的知识点与内容，仅供学有余力同学参考，不必在当前阶段掌握

当我们访问网站的时候，发生了些什么？

1 前言

## Browser/Server

网页是文件

## ② Web 页面

### ③ Web 通信

## 浏览器与服务器

Browser 浏览器代表了使用者  
访问网站时，一定有 Server 服务器为你提供服务

1 前言

## Browser/Server

网页是文件

## ② Web 页面

### ③ Web 通信

# 网页是文件...

- 浏览器通知服务器，自己需要某一路径的文件（某一型号的包裹）...
- 一些文件（包裹）从访问的服务器寄了回来...
- 浏览器收到这些包裹并拆开，将各个元件按照规则组装好...
- “包裹”里一般有哪些种类的文件呢？

## ① 前言

## ② Web 页面

HTML

CSS, JavaScript

页面调试

## ③ Web 通信

## ④ Web 应用基础

## ⑤ 作业说明

## ① 前言

## ② Web 页面

HTML

CSS, JavaScript

页面调试

## ③ Web 通信

## ④ Web 应用基础

## ⑤ 作业说明

# HTML 简介

*HTML*, 灵动的编码之舞 / 将网页世界织成绚烂的画

她是结构的诗人 / 用标签勾勒

<div> 和 <p> / 行间的节奏

标题 <h1> 至 <h6> 层次分明 / 呼应着内容的主次

图片 <img> 在浏览中翩然起舞 / 与文字相映, 共舞奇妙的韵律

*CSS* 是她的伙伴, 外观的艺术家 / 搭配颜色和布局, 赋予网页鲜活的气息

*JavaScript* 则是她的魔法师 / 注入交互的魔力, 让网页焕发生机与活力

*HTML*, 她是网页世界的建筑师 / 用标签构筑梦想, 让创意自由展现

愿她的舞 / 继续编织

网页的诗篇 / 永不停歇

by ChatGPT

# HTML 简介

- HTML (Hyper Text Markup Language, 超文本标记语言) 是一种用于描述网页文档的语言
  - HTML 定义了许多不同种类的标签，它们被尖括号包围，起始标签和结束标签通常成对出现
  - 标签自身则表示了内容的类型与格式，标签内含有具体的内容。例如，下面的代码表示加粗的“你好”

1

<b> 你好 </b>

# HTML 简介

- Word 中，我们手动在 Tab 栏里调整格式，文本的写作、结构和格式的调整相分离
- HTML 中，内容的格式信息也以标签的形式，作为纯文本存储在文件中，便于生成、编辑、传输、渲染

# 常用 HTML 标签

有一些 HTML 标签用来表征文本：

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

表 1: 常用的 HTML 标签

# 常用 HTML 标签

作为“超文本标记语言”，HTML 标签还可以表征文本以外的内容：

- 图片、视频、超链接等同样会显示在网页上的内容
- 标题、引入代码等不会直接显示，但是与网页有关的内容

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

表 2: 另外一些常用的 HTML 标签

# HTML 实例

```
1 <!DOCTYPE html>
2 <!--声明文档类型-->
3 <html>
4 <head>
5   <title>Tab 标题 </title>
6 </head>
7 <body>
8   <h1> 大标题 </h1>
9   <p> 一个段落 <b> 加粗了一些内容
10  <-- </b></p>
11  <HR>
12  
13  <!--一张图片-->
14 </body>
</html>
```



- 请大家照着上述例子，编写简单的 HTML 文件，存储为 xxx.html，并用浏览器打开
- Any questions?
- 更多内容，可参考 菜鸟教程

# HTML 标签属性

HTML 标签有很多属性，用来提供与标签有关的附加信息。属性总是以名称/值对的形式出现，比如：`name="value"`。

- `id`: 指定唯一标识。
- `class`: 指定类名，可以与其他元素共享。
- `src`: 源，用于链接外部资源，例如图片。
- `href`: 超链接，用于链接到外部网页。
- `style`: 指定样式。

```
  
<div id="main" class="cont" style="color: red;">
```

# DIV 标签

我希望元素横向排列该怎么办呢...?

<div> 标签可以在网页中创建一个无语义的块级容器，可以用来分组、包含、布局和样式化其他 HTML 元素。

<div> 标签没有特定的含义或目的，它只是一个容器。

思路：新建一个 div，将其内部元素的排列方向设为水平；将需要左右排列的元素置于该 div 内部。

## 1 前言

## 2 Web 页面

HTML

CSS, JavaScript

页面调试

## 3 Web 通信

## 4 Web 应用基础

## 5 作业说明

# CSS 简介

- 现在的网站似乎有些朴素...连 info 都比你好看
- 我们可以在标签内添加 style="..." 属性，进行美化
- 也可以在文档头部添加 <style> 标签，在其中定义不同类型的标签的格式，进行批量调整

```
1  <!DOCTYPE html>
2  <!-- 声明文档类型 --&gt;
3  &lt;html&gt;
4  &lt;head&gt;
5      &lt;title&gt;Tab 标题 &lt;/title&gt;
6      &lt;style&gt;
7          h1 {
8              color: red;
9              font-weight: bold;
10         }
11     &lt;/style&gt;
12     <!-- 定义了 h1 标签的颜色和字体 --&gt;
13 &lt;/head&gt;
14 &lt;body&gt;
15     &lt;!-- 此部分内容同上 --&gt;
16 &lt;/body&gt;
17 &lt;/html&gt;</pre>
```



# CSS 简介

- 我们可以在文档头部添加 `<style>` 标签，在其中定义不同格式的标签，他们控制着网页的外观、颜色、布局、动画效果
- 同一网站不同页面的格式往往是统一的，因此这些样式可以被抽离出来，成为 CSS(Cascading Style Sheets) 文件
- 除了元素类型，CSS 还可以根据元素的类、`id` 进行选择 [拓展]
- 之所以叫做“层叠样式表”，是因为当有多个规则作用于同一个元素时，规则会按照一定顺序叠加 [拓展]

# JavaScript 简介

- 在点击一个图标后，我希望网页的内容发生一些变化...
- 可以通过一种“简易”的脚本语言来达到这一目的：

```
1  <!--重复内容略-->
2  <body>
3      
```



# JavaScript 简介

- JavaScript 是一种脚本语言，它不需要编译，嵌入 HTML 后随时可以解释运行
- 它是图灵完备的，可以实现任何逻辑；  
它的优势在于可以方便地访问网络、操纵 **HTML** 文档内的元素（划重点！）
- 其语法在很多地方与 C++ 等语言类似，例如循环的写法  
因此同学们可以参考网上的教程进行自学
- 和 CSS 文件一样，JavaScript 代码也可以被抽离至单独的文件中

# JavaScript [拓展] 知识点

- JavaScript 和 Java 没有半毛钱关系  
~~纯粹是发明者想要蹭热点~~
- JavaScript 有很多特别神奇的特性
- JavaScript 还被广泛运用在浏览器以外的场景中  
敬请期待前端 Track 等相关课程

# 综合实例

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title> 综合实例 </title>
5 <link rel="stylesheet"
6   href="style.css">
7
8 </head>
9 <body>
10 <h1> 按钮示例 </h1>
11 <button id="myButton"
12   class="my-button"> 点击我
13 </button>
14 <script src="script.js"></script>
15 </body>
</html>
```

page.html

```
1 let button =
2   document.getElementById("myButton");
3
4 button.addEventListener("click", function() {
5   button.style.backgroundColor = "red";
6 });
```

script.js

```
1 .my-button {
2   padding: 10px 20px;
3   background-color: blue;
4   color: white;
5   border: none;
6   cursor: pointer;
7 }
8 h1 {
9   color: #39c5bb;
10 }
```

style.css

# 综合实例

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title> 综合实例 </title>
5 <link rel="stylesheet" href="style.css">
6
7 </head>
8 <body>
9 <h1> 按钮示例 </h1>
10
11 <button id="myButton" class="my-button"> 点击
12   我 </button>
13 <script src="script.js"></script>
14
15 </body>
</html>
```

page.html



# 综合实例

```
1 .my-button {  
2     padding: 10px 20px;  
3     background-color: blue;  
4     color: white;  
5     border: none;  
6     cursor: pointer;  
7 }  
8 h1 {  
9     color: #39c5bb;  
10 }
```

style.css

```
1 let button =  
2 →   document.getElementById("myButton");  
3  
4 button.addEventListener("click", function() {  
5     button.style.backgroundColor = "red";  
6 });
```

script.js



# 综合实例

```
1 .my-button {  
2     padding: 10px 20px;  
3     background-color: blue;  
4     color: white;  
5     border: none;  
6     cursor: pointer;  
7 }  
8 h1 {  
9     color: #39c5bb;  
10 }
```

style.css

```
1 let button =  
2 →  document.getElementById("myButton");  
3  
4 button.addEventListener("click", function() {  
5     button.style.backgroundColor = "red";  
6 });
```

script.js



- 引入 CSS / JS 文件时的相对路径与绝对路径
- <script> 标签的位置

## ① 前言

## ② Web 页面

HTML

CSS, JavaScript

页面调试

## ③ Web 通信

## ④ Web 应用基础

## ⑤ 作业说明

# Web 开发工具

- Google Chrome (大部分现代浏览器都采用了 Chromium 内核)
- F12 ( $\text{Fn} + \text{F12}$ )
- $\text{Ctrl} + \text{S}$  保存网页

# 总结：网页是文件...



HTML, CSS, JavaScript  
统称为「前端三件套」。

---

我们已经知道了包裹里有哪些文件。

---

那么，“快递单”怎么填？

## ① 前言

## ② Web 页面

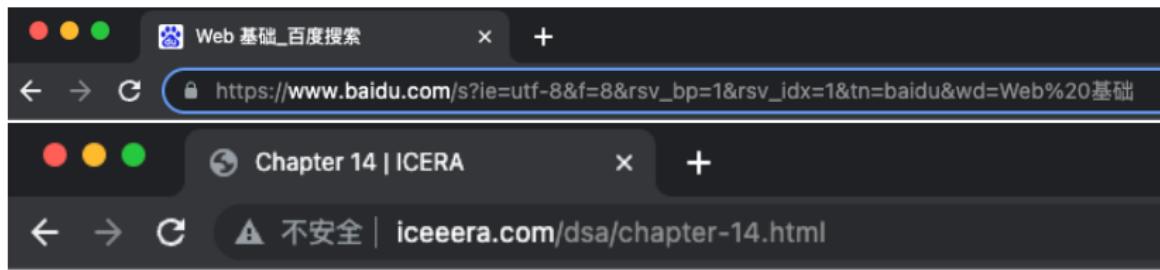
## ③ Web 通信

HTTP

## ④ Web 应用基础

## ⑤ 作业说明

# 地址怎么填？



- 主机（域名？）——服务提供者
- 协议://主机名 [: 端口]/路径 [? 查询]
- 「网址」其实就是 URL (Uniform Resource Locator)

## ② Web 页面

### ③ Web 通信

## HTTP

4 Web 应用基础

# 其他信息怎么填？

- 有时候，我们不仅希望请求一些文件；我们还需要主动发送一些信息，例如用户名、密码…
  - HTTP (Hyper Text Transfer Protocol)** 规定了沟通的格式
- 
- 每一次沟通都由客户端发起，发送一个 `request`，请求正文 中可以携带各种信息
  - 服务器收到请求后，会返回一个 `response`，正文中也可以带 有 `HTML` 等（任意格式的）文件

## HTTP request

请求从客户端发送至服务端。不同的请求方法具有不同的语义：

- GET: 请求指定的资源
- POST: 提交数据
- PUT: 用提交的数据更新或替换指定资源
- DELETE: 删除服务器端指定的资源

思考一下，下面这些需求应该用哪些请求方法？

- 访问一个网页
- 发送自己的用户名和密码
- 更新邱老师的职位信息

~~事实上，大部分请求方法都可以混用，但是这样不符合开发规范~~

# HTTP request 中的参数传递

我们知道了有不同的请求方法，那么该怎么携带具体信息呢？

- 路径传参，例如用 GET 方法请求

`http://info.com/score/2022010897/`

- Query String，例如用 GET 方法请求

`http://info.com/score?id=2022010897&semester=23fall`

- 请求体传参，我们可以在请求的正文中传输任意格式的信息作为参数，例如 POST 请求 `http://info.com/score` 并携带以下 JSON 格式的信息

```
{  
    "id": 2022010897,  
    "semester": "23fall",  
    "GPA": 4.0  
}
```

JSON 是一种很常见的数据交换格式，请同学自行查找资料

# HTTP response

服务器收到客户端发来的请求后，会进行处理，并返回一个 `response`。`response` 没有不同的分类，但是有不同的状态码，表示不同的响应结果：

- 200 OK: 请求成功
- 400 Bad Request: 请求有误
- 404 Not Found: 请求的资源不存在
- 500 Internal Server Error: 服务器内部错误

# HTTP response



418  
I'm a teapot

## HTTP response

服务器收到客户端发来的请求后，会进行处理，并返回一个 response。response 没有不同的分类，但是有不同的状态码，表示不同的相应结果：

- 200 OK: 请求成功
- 400 Bad Request: 请求有误
- 404 Not Found: 请求的资源不存在
- 500 Internal Server Error: 服务器内部错误

HTTP response 的正文中可以携带 HTML 文档、JSON 数据、图片等任何内容

# HTTP 实例

一个极其简化版本的查分流程：

- ① 客户访问 info.com, 浏览器 GET 请求 `http://info.com/[request]`
- ② 服务器端收到请求, 返回一个 HTML 文件 [response], 浏览器显示该页面
- ③ 客户输入用户名密码后点击登陆, 内嵌 JS 代码致使浏览器 POST 请求 `http://info.com/login`, 且请求 body 中含有 JSON 数据 [request]
- ④ 服务器验证收到的用户名和密码正确, 去数据库里查找该用户的成绩, 返回一个 JSON 数据 [response]

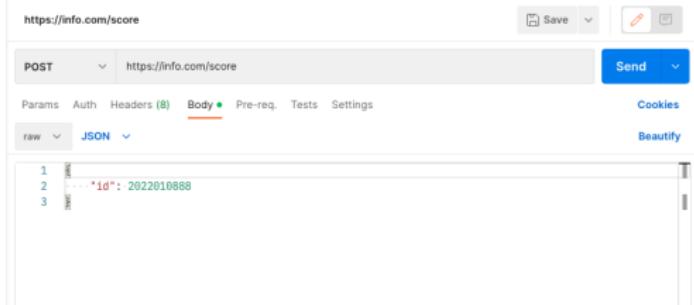
# 小结



# HTTP 调试

Debug 时，我们希望能向任意地址发送任意方法的信息，携带任意正文...

推荐使用 Apifox / Postman：



The screenshot shows the Apifox interface for sending a POST request to the URL `https://info.com/score`. The request method is set to `POST`. The `Body` tab is selected, showing the following JSON payload:

```
1 {  
2   "id": 2022010888  
3 }
```

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

静态网页

动态网页

前后端分离开发模式

持久化存储

## ⑤ 作业说明

现在，我们已经知道了包裹里有哪些类型的文件，快递单怎么填。

---

该怎么处理包裹的内容、进行收发呢？

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

静态网页

动态网页

前后端分离开发模式

持久化存储

## ⑤ 作业说明

# 静态网页

- 有些网站的内容是提前创建好、固定不变的，这些称为静态网页。例如 Hexo 等生成的个人博客。
- 与之相对的是动态网页，网站内容会根据使用者的身份、后台数据库等因素动态生成，例如淘宝、京东。
- 对于静态网站，每一个地址对应的“包裹”里的内容是固定的
- 服务端不需要对内容做逻辑上的处理，只需要将已经打包好地包裹不断根据快递单寄出即可

# 静态网页

- 由于没有具体业务逻辑，一个通用的服务端软件即可满足需求，例如 Nginx
- 准备好 HTML, CSS, JS 文件，放在服务器指定路径上，让 Nginx 做一个无情的发包机器
- 网站就搭建好了……？

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

静态网页

动态网页

前后端分离开发模式

持久化存储

## ⑤ 作业说明

# 动态网页

- 有些网页的内容不得不动态地变化...
- 内容需要实时更新，或者我们希望根据用户的身份显示不同的内容

京东首页 | 海外

你好，请登录 [免费注册](#) 我的订单 | 我的京东 | 企业采购 | 客户服务 | 网站导航 | 手机京东 | 网站无障碍

**京东**

计算机网络  搜索

计算机网络 英文 | 计算机网络谢希仁 | 物联网导论 | 数据结构 | python | tcp ip详解 | python程序设计基础 | pytorch |

全部商品分类 大牌奥莱 美妆馆 超市 生鲜 京东国际 拍卖 金融 京东五金城

全部结果 > “计算机网络”

品牌:	电子工业出版社	人民邮电出版社	清华大学	机械工业	高等教育出版社 (HIG...)	中国人民大学出版社	文轩	北京大学出版社	更多	+多选				
步步高书	科学出版社 (SCIENCE...)	iTuring	清华大学出版社 (TSL)	海尚 (DANGDANG.CO...)	机械工业出版社 (CHP...)	博文视点	中国水利水电出版社							
计算机与互联网:	计算机工具书	操作系统	编程语言与程序设计	计算机安全	计算机理论、基础知识	数据库	硬件与维护	更多						
高中专教材教辅:	大学教材	高职高专教材	中职中专教材	研究生教材										
其他图书:	Computers & Internet...	法律法规	理论法学	法律普及与律师	法律实务	刑法	司法案例与司法解释	行政法	更多					
出版社:	人民邮电出版社	清华大学出版社	电子工业出版社	机械工业出版社	高等教育出版社	科学出版社	中国铁道出版社	中国水利水电出版社	更多	+多选				
高级选项:	<input checked="" type="checkbox"/> 包装	<input type="checkbox"/> 是否盗版	<input type="checkbox"/> 客户评分	<input type="checkbox"/> 折扣	<input type="checkbox"/> 科目	<input type="checkbox"/> 题型类型	<input type="checkbox"/> 产品尺寸	<input type="checkbox"/> 线长	<input type="checkbox"/> 题型类型	<input type="checkbox"/> 连接线	<input type="checkbox"/> 安装位置	<input type="checkbox"/> 通用性别	<input type="checkbox"/> 封面类型	<input type="checkbox"/> 特色功能
<input type="button" value="搜索"/> <input type="button" value="销量"/> <input type="button" value="评论数"/> <input type="button" value="出版时间"/> <input type="button" value="价格"/>														
共87万+件商品 1/100 < >														
<input type="checkbox"/> 海外日本   <input type="checkbox"/> 京东物流   <input type="checkbox"/> 货到付款   <input type="checkbox"/> 仅显示有货   <input type="checkbox"/> 新品   <input type="checkbox"/> 拍拍二手   <input type="checkbox"/> 五金城														


¥110.60


¥38.80


¥50.90


¥69.80


¥138.00

清华大学计算机系学生科协

李轶凡

Web 与 Web 应用基础

53 / 77

# 传统模式

- 服务器在收到请求后，根据请求内容，从数据库中读取数据
- 将数据处理后填入预先准备好的模板当中，构成完整的 HTML 文件
- 将文件打包，作为包裹内容寄回

这样做有什么问题呢？

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

静态网页

动态网页

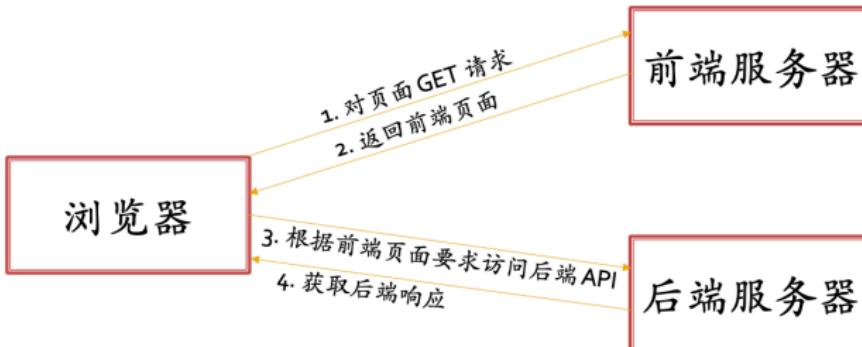
前后端分离开发模式

持久化存储

## ⑤ 作业说明

# 前后端分离模式

- 对于绝大部分动态网页而言，“动态”的是具体的数据，网页的框架、呈现给用户的界面是“静态”的
- 把“动态”数据填入“静态”模板这一过程其实可以放到客户端进行
- “前端”指的就是用户界面，“后端”指的就是数据与业务逻辑，两者可以完全分离

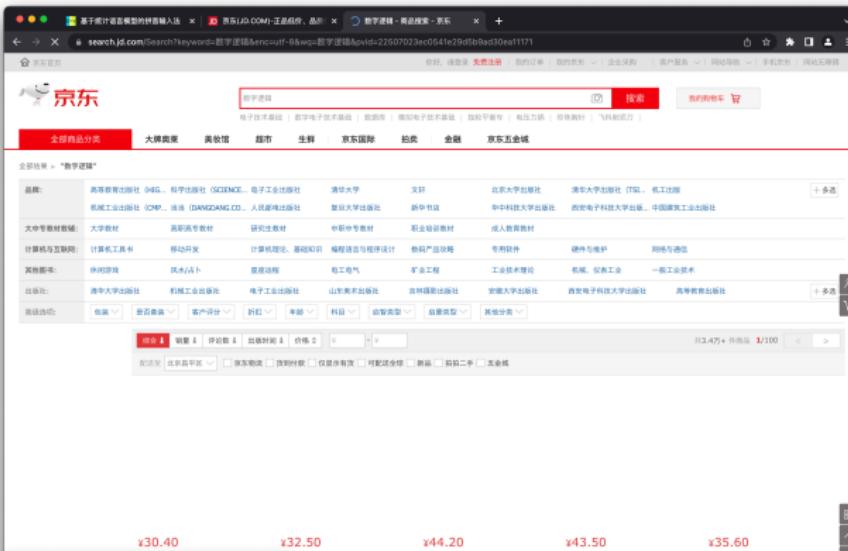


# 前后端分离开发模式

仍然以访问京东为例：

- ① 客户端请求 jd.com，返回了由 HTML, CSS, JS 文件构成静态网页

此时我们可能看得到网页的大体框架，却看不到具体的商品



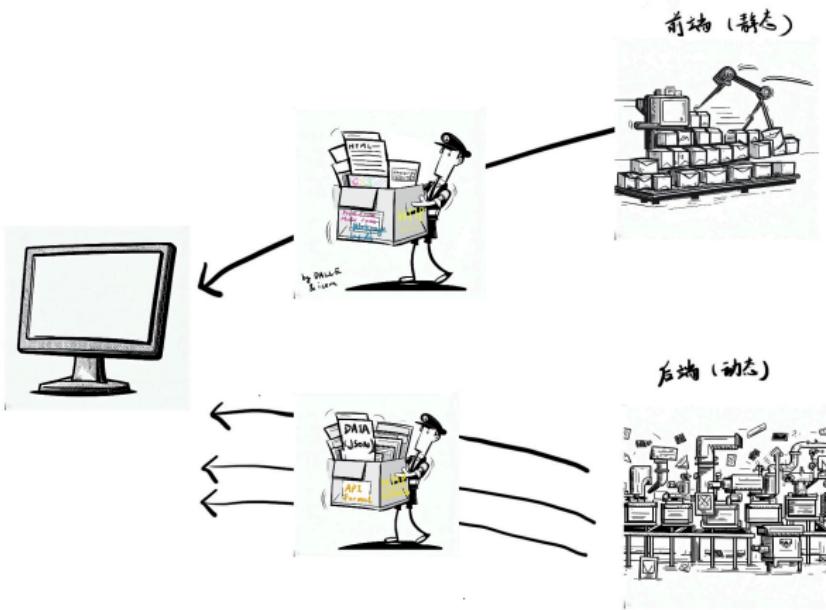
# 前后端分离开发模式

仍然以访问京东为例：

- ① 客户端请求 jd.com，返回了由 HTML, CSS, JS 文件构成静态网页  
此时我们可能看得到网页的大体框架，却看不到具体的商品
- ② 网页上的 JS 脚本会向指定的后端服务器发起请求，获取商品信息
- ③ JS 脚本将获取到的信息填入网页，呈现出我们看到的内容

前后端之间的通信也往往使用 HTTP 协议；传输数据的具体格式由开发者规定，称为 API

# 小结



# Rust 小学期 - Online Judge

<https://lab.cs.tsinghua.edu.cn/rust/projects/oj/requirements/>

## 基础要求 (40')

此部分功能按点给分。并且必须按顺序实现（也就是说，某个功能点得分的必要条件是它与之前的所有功能均正确实现）。除特殊说明外，最终结果以程序在助教环境中测试的结果为准。

- (15) 基础评测：运行 Web 服务器，实现 POST /jobs API。
  - 从命令行参数获得配置文件路径，解析配置文件内容
  - 接受到 POST /jobs 请求时，从请求中获取 source\_code 字段，根据配置文件中 Rust 语言的配置，使用 rustc 命令编译为可执行文件。
  - 根据题目 ID 找到题目的各个数据点，对题目的各个数据点进行评测，衡量测量程序运行的真实时间 (wall time)，在超过设定的时间限制一段时间 (如 500ms) 后强制终止程序；
  - 比对程序输出的结果与答案，需要支持两种模式：standard (忽略文末空行和行末空格) 和 strict (严格比较)
  - 实现阻塞评测，在评测后才返回响应，因此只会出现 Finished 状态，数据点结果仅会出现 Waiting (前序数据尚未评测完成或编译错误)、Accepted (答案正确)、Wrong Answer (答案错误)、Time Limit Exceeded (超时) 和 Runtime Error (运行时错误，程序异常退出)
  - 不需要保存评测历史，所有评测 ID 都可返回 0
- (5) 多语言支持：支持配置文件中提供的其他语言（保证使用的编译命令存在于系统中）
  - 请保证你的系统内安装了 gcc 和 g++
  - 如果你在 Windows 中开发（但没有使用 WSL），可以本地修改测例的配置为其他编译器（如 MSVC, MinGW 等），但不要提交到 Tsinghua GitLab 代码仓库上
- (10) 评测列表：保存启动以来的评测列表，实现下列 /jobs 开头的 API，实现对任务的搜索，详情查询和重新评测。
  - GET /jobs：筛选并获取评测任务列表；
  - GET /jobs/{jobId}：获取指定评测任务信息；
  - PUT /jobs/{jobId}：重新评测指定评测任务。
- (5) 用户支持：实现所有 /users 开头的 API，支持简单的用户功能。此时创建任务时应该进一步检查用户 ID，并且支持通过 ID 和名称搜索相应用户的评测任务。

在前后端分离的开发范式下，遵循给定的 API，完成一个（动态的）后端服务器

API 是应用程序编程接口，定义了软件组件之间交互的规范，如标准数据格式、通信协议等

# Rust 小学期 - Online Judge

<https://lab.cs.tsinghua.edu.cn/rust/projects/oj/requirements/>

## 提高要求 (40')

此部分功能没有具体的要求，助教将视完成情况酌情给分。其中有些项目存在依赖关系，无法单独完成。所有项目得到的分数综合不超过 40 分。标记有“”的功能是助教认为复杂度较高的功能，请谨慎上手。

实现提高要求部分不应该影响已有的自动化测试。如要实现任何未提及的功能，请先与助教确认可行性，并评估应得分数，如未经确认直接实现则不得分。

OJ 前端    用户管理    评测技术    评测方式

只可选择一个方向完成，可以选择子项目完成。

- CLI 前端 (10')：使用 Rust 编写 CLI 客户端，实现以下功能：
  - (3') 提交单个程序（提交的程序支持使用不同编程语言编写，如 Rust, C++），并等待结果
  - (4') 根据一定的配置批量提交不同用户的程序，并以机器可读格式（如 JSON）输出评测结果
  - (3') 查询排行榜并以表格形式渲染到终端
- Web 前端 (10')：不要求使用 Rust，建议使用现代前端框架（如 React、Vue、Angular 等），实现以下功能（效果参考 TUOJ）：
  - (5') 提交单个程序，并渲染评测状态和每个数据点结果
  - (5') 渲染比赛排行榜

使用现代的前端框架，完成一个前端网页，前后端通过约定的 API 通信；这个前端页面应该可以编译成静态的文件，挂载在服务器上

Any questions?

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

静态网页

动态网页

前后端分离开发模式

持久化存储

## ⑤ 作业说明

# 持久化存储

等等，我们的数据该怎么保存呢？

总不至于，服务器重启，评测数据全部丢失了吧？

# 持久化存储

- 朴素想法：写到文件里去，下次启动的时候还在（这也需要一定的规范，例如之前提到的 JSON）
- 格式规整的信息被存储到了“Word 文档”里，当你想要查找的时候...为什么不存到“Excel”里面去呢？
- 数据库以表单形式持久化存储数据，可以方便地进行查询、修改、删除等操作
- 列：对应不同属性
- 行：对应不同项

# 敬请期待...

- 数据库：@Kaiming 的暑期培训第二讲
- 后端：Django 或 Rust 后端课程
- 前端：@Ashimetaru 的 React 课程

## ① 前言

## ② Web 页面

## ③ Web 通信

## ④ Web 应用基础

## ⑤ 作业说明

# 课程作业（普通版、提高版）PART I

尝试使用 Postman / Apifox 等工具发送 Http 请求

入口：GET [iceeera.com:230/task/start/](http://iceeera.com:230/task/start/)

按照得到的响应中的提示操作，最终可以获得一个 Code

# 课程作业（普通版、提高版）PART II

生成类似的静态网页（请大家发挥想象力，随意修改内容）

# 课程作业（普通版、提高版）PART II

- 先保存现有的网页（网页，全部文件）
- 修改主 html 文件，要求对 HTML, CSS / 样式, JavaScript 至少各进行一处修改或添加
- JavaScript 修改部分普通要求为点击某一按钮后弹出提示框
- 提高要求为点击某一按钮后改变网页中的某一元素的内容（课件中有一个类似的例子）
- 将 PART I 中得到的 Code 放到提示框或任意显著元素当中

## 课程作业（普通版）PART III

完成后，将网页文件与相应的资源文件打包上传到清华云盘

# 课程作业（提高版）PART III

- ① 将得到的网页文件上传到服务器上（推荐使用 `scp` 指令）
- ② 对服务器上提供的 `nginx` 进行配置，使其作为一个简易的 Web 服务器运行

完成后，任何人都可以通过 `IP:< 分配给你的网页服务端口 >` 访问到你的网页

# 课程作业（提高版）PART III NGINX

## NGINX 部分指南：

- ① 确保你的账号有 Root 权限
- ② 使用 scp 指令或者 Vscode 将文件上传到服务器的任意路径
- ③ 修改 nginx 的配置文件：将 http 块最后两行 include 语句用 # 注释掉，并在 http 块里面添加以下内容

```
server {  
    listen 80;  
    root #< 你存放网页的文件夹的路径 >;  
  
    location / {  
        index #< 你的主 html 文件名 >;  
        try_files $uri $uri/ =404;  
    }  
}
```

# 课程作业（提高版）PART III NGINX

## NGINX 部分指南：

- ① 确保你的账号有 Root 权限
- ② 使用 scp 指令或者 Vscode 将文件上传到服务器的任意路径
- ③ 修改 nginx 的配置文件
- ④ 使用 sudo nginx -t 检查配置文件是否有语法错误
- ⑤ 输入 sudo service nginx restart 重启 nginx 服务

如欲了解更多，可以参考

<https://blog.xial.moe/index.php/2022/09/21/nginx-peizhiwenjian/>

# 课程作业（高级版）

对于选了 Rust 小学期的同学...  
搭建一个功能完善的 OJ (有前后端)  
可以替代本次作业

# 作业说明

- 如果遇到问题，欢迎在课程群或者私聊讲师提问
- 普通作业的两个 PART 可以单独完成
- DDL 为一周后 (OJ 除外)
- 通过提 issue 的方式在  
<https://github.com/sast-summer-training-2023/sast2023-web>  
仓库中上交自己的部署网址或链接，并说明自己完成的作业内容

谢谢！

提问时间