第13课 前端精讲

（本节课了解即可，面试里问到的概率不大，故而本节课无面试Q/A）

课程目标

本课程旨在详细介绍如何在 C++ 编写的 HTTP 服务器中实现前后端联动。我们将通过创建简单的登录和注册页面，展示前端 HTML 与后端 C++ 代码如何相互作用。

HTML 基础教程

什么是 HTML?

- **HTML**（HyperText Markup Language，超文本标记语言）是用于创建网页的标准标记语言。它描述了一个网页的结构和内容。
- HTML，即超文本标记语言，是构建网页的基础。它通过标记符号来结构化文本，使得文本具有网络表示的形式。
- **HTML定义了网页的内容与结构，使得文本可以包含链接、图片以及其他多媒体内容，为用户提供交互性体验。**

HTML 基本结构

- 一个典型的HTML文档主要包含以下部分：
  a. `<!DOCTYPE html>`：这是一个文档类型声明，通常位于HTML文档的开头。它告诉浏览器文档的类型和版本，这有助于浏览器正确解释文档的结构和内容。`<!DOCTYPE html>` 是HTML5的文档类型声明。
  b. `<html>`：`<html>` 元素是整个HTML文档的根元素。它包含了整个HTML文档的内容，定义了文档的开始和结束。
  c. `<head>`：`<head>` 元素位于 `<html>` 内，用于包含文档的元数据，这些元数据不会直接显示在网页上，但对网页的显示和行为有重要影响。常见的元数据包括：
     - `<title>`：`<title>` 元素用于定义文档的标题，将显示在浏览器的标题栏或选项卡上，以及搜索引擎结果中的标题。
     - `<meta>`：`<meta>` 元素用于设置字符集、关键词、描述等元数据。
     - `<link>`：`<link>` 元素通常用于引入外部样式表，以定义文档的样式和布局。

    - `<script>`：`<script>` 元素用于引入JavaScript代码，以添加交互性和动态功能。

d. `<title>`：`<title>` 元素位于 `<head>` 内，用于定义文档的标题。文档的标题将显示在浏览器的标题栏或选项卡上，帮助用户识别页面内容。

e. `<body>`：`<body>` 元素是HTML文档中可见内容的容器。它包含了文本、图像、链接、表单、段落等可视元素，这些元素将在用户的浏览器中呈现为可交互和可见的页面内容。用户看到的大部分页面内容都位于 `<body>` 元素内。

示例代码

```
<!DOCTYPE html>
<html>
<head>
    <title>我的第一个 HTML 页面</title>
</head>
<body>
    <h1>欢迎学习 HTML</h1>
    <p>HTML 是构建网页的基础。</p>
</body>
</html>
```

常用标签

- `<h1>到<h6>`：标题标签，`<h1>` 表示最大的标题。
- `<p>`：段落标签，用于定义文本的段落。
- `<a href="url">`：链接标签，用于创建指向其他页面的链接。
- `<img src="image.jpg">`：图像标签，用于嵌入图片。
- `<ul>`、`<ol>`、`<li>`：无序列表、有序列表和列表项标签。
- `<div>`：用于定义文档中的分区或节。
- `<span>`：用于对文档中的行内元素进行分组。

Html 在线运行 - 在线工具这里可以预览HTML的效果

```
<!DOCTYPE html>
<html>
<head>
    <title>我的网页</title>
```

```
</head>
<body>
    <h1>欢迎来到我的网站</h1>
    <p>这是一个段落，介绍一些基本的 HTML 标签。</p>

    <h2>链接和图片</h2>
    <p>这是一个链接到 <a href="https://www.example.com">example.com</a> 的例子。</p>
    <p>下面是一个嵌入的图片：</p>
    <img src="image.jpg" alt="描述性文字">

    <h3>列表</h3>
    <p>以下是一个无序列表的示例：</p>
    <ul>
        <li>列表项一</li>
        <li>列表项二</li>
        <li>列表项三</li>
    </ul>

    <p>以下是一个有序列表的示例：</p>
    <ol>
        <li>第一项</li>
        <li>第二项</li>
        <li>第三项</li>
    </ol>

    <h4>使用 Div 和 Span</h4>
    <div>
        <p><span>Span</span> 用于行内元素的分组。</p>
    </div>

    <h5>次级标题</h5>
    <p>更小的标题标签 <h5>。</p>

    <h6>最小的标题</h6>
    <p>这是最小的标题标签 <h6>。</p>
</body>
</html>

```

表单标签 `<form>`

- `<form>` 标签用于创建一个HTML表单，用于收集用户输入。
- 示例：

```html
<form action="/submit_form" method="post">
    <label for="name">姓名：</label>
    <input type="text" id="name" name="name"><br><br>
    <input type="submit" value="提交">
</form>
```

总结

- HTML是网页设计和开发的基础，了解其基本结构和常用标签是学习网页设计的第一步。
- 通过实践和编写更多的HTML代码，你将更好地理解如何构建和设计网页。

HTML CSS Javascript比较

1. HTML (HyperText Markup Language)
    - 作用：用于构建网页内容结构和提供语义化信息。
    - 特点：HTML是一种标记语言，通过标签来描述文档的结构和内容，如标题、段落、表格、图像等。它不涉及页面的样式或交互行为。
2. CSS (Cascading Style Sheets)
    - 作用：负责网页的样式呈现，包括颜色、字体、布局、尺寸等视觉效果。
    - 特点：CSS是一种样式表语言，可以独立于HTML进行样式设计，并能够控制任何XML（包括HTML）文档的展示方式。它允许开发者将内容与表现分离，实现代码复用及维护性提升。
3. JavaScript
    - 作用：为网页添加动态功能和交互性，如响应用户事件、操作DOM元素、执行异步请求、处理数据等。
    - 特点：JavaScript是一种解释型编程语言，它可以嵌入到HTML中并在客户端浏览器上运行，提供了丰富的API以操纵网页内容和行为。随着Node.js的出现，JavaScript还可以用于服务器端开发。

比较异同：

- 相似点：
    - 都是前端开发的核心组成部分，共同协作构建现代Web应用程序。
    - HTML、CSS、JavaScript在浏览器环境中协同工作，实现从静态内容展示到动态交互体验的转变。
- 不同点：

- HTML关注的是内容的组织和结构化，CSS关注的是内容的展现样式，而JavaScript关注的是页面的动态逻辑和交互。
- HTML和CSS不具备程序执行能力，而JavaScript拥有完整的编程特性，可以处理复杂的业务逻辑和数据处理任务。
- HTML和CSS相对静态，而JavaScript可以根据用户的交互和数据变化实时更新页面内容和样式。

代码实践

创建前端界面

- **登录页面（login.html）**：

```
<form action="/login" method="post">
    用户名: <input type="text" name="username">
    密码: <input type="password" name="password">
    <input type="submit" value="登录">
</form>
```

- **注册页面（register.html）**：

```
<form action="/register" method="post">
    用户名: <input type="text" name="username">
    密码: <input type="password" name="password">
    <input type="submit" value="注册">
</form>
```

服务器端代码实现

- 使用 `Router` 类处理前端页面请求和表单提交。
- **提供静态页面**:

```
router.addRoute("GET", "/login", [this](const HttpRequest& req) {
    HttpResponse response;
    response.setStatusCode(200);
    response.setHeader("Content-Type", "text/html");
    response.setBody(readFile("path/to/login.html")); // 读取 HTML 文件
```

```
    return response;
});
```

- **处理表单提交**:

```
router.addRoute("POST", "/login", [&db](const HttpRequest& req) {
    auto params = req.parseFormBody();
    std::string username = params["username"];
    std::string password = params["password"];
    // 进行登录验证...
});
```

readFile实现

```
std::string readFile(const std::string& filePath) {
    // 使用标准库中的ifstream打开文件
    std::ifstream file(filePath);

    // 判断文件是否成功打开
    if (!file.is_open()) {
        // 若未能成功打开文件，返回错误信息
        return "Error: Unable to open file " + filePath;
    }

    // 使用stringstream来读取文件内容
    std::stringstream buffer;
    // 将文件内容读入到stringstream中
    buffer << file.rdbuf();

    // 将读取的内容转换为字符串并返回
    return buffer.str();
}

```

函数解释：

- 函数目的：此函数用于读取指定路径的文件内容，并将其作为字符串返回。
- 参数：`const std::string& filePath` 表示要读取的文件的路径。
- 流程解析：

a. 打开文件：使用 `std::ifstream` 类创建 `file` 对象并尝试打开指定路径的文件。

b. 检查文件是否打开：使用 `file.is_open()` 检查文件是否成功打开。如果无法打开文件，函数返回一条错误信息，表明无法打开指定的文件。

c. 读取文件内容：创建一个 `std::stringstream` 对象 `buffer`。使用 `file.rdbuf()` 读取整个文件的内容，并使用 `<<` 运算符将其内容传输到 `buffer` 中。

d. 返回文件内容：使用 `buffer.str()` 将 `stringstream` 中的内容转换为 `std::string` 类型，并返回这个字符串。

前后端联动机制

1. **用户交互**：
   - 用户在浏览器中填写表单并提交。
2. **前端请求**：
   - 浏览器向服务器发送 HTTP 请求，包含表单数据。
3. **服务器处理**：
   - 服务器接收请求，`HttpRequest` 类解析请求数据。
   - 进行业务逻辑处理，如验证登录凭据。
4. **服务器响应**：
   - 根据处理结果，服务器使用 `HttpResponse` 类创建响应。
   - 将响应数据发送回浏览器。
5. **用户获得反馈**：
   - 浏览器显示来自服务器的响应，如登录成功或失败的消息。

代码解析

- `HttpRequest` 类解析来自客户端的请求，提取出关键信息如请求类型（GET/POST），路径，以及提交的数据。
- `HttpResponse` 类用于构建要发送回客户端的响应，包括状态码（如200，404），响应头，以及响应体。
- `Router` 类根据请求的路径和方法，决定调用哪个处理函数，实现请求的具体逻辑。

测试

- 运行服务器并通过浏览器访问登录和注册页面。
- 尝试填写表单并提交，观察服务器如何响应不同的输入。

CSS教程

CSS（Cascading Style Sheets）是一种样式表语言，用于描述HTML文档或XML（如SVG、MathML等）文档的呈现方式。它为网页设计提供了丰富的视觉和布局控制功能。

定义：

- CSS 是一种样式表技术，允许开发者将内容与表现形式分离，使内容更易于维护和复用。
- 它提供了一套声明式规则来指定网页元素应该如何显示，包括字体、颜色、布局、尺寸、动画效果等等。

作用：

1. **美化界面** - 控制文本样式（如大小、颜色、行高、对齐方式等）、背景、边框、阴影等外观属性。
2. **布局控制** - 通过盒模型、定位、浮动、Flexbox 或 Grid 等布局机制实现页面结构的复杂排版。
3. **响应式设计** - 根据设备视口大小、分辨率和方向调整布局和样式。
4. **交互性增强** - 使用伪类和JavaScript配合实现动态效果和用户交互反馈。
5. **可访问性优化** - 提供替代文字、焦点样式等，帮助残障人士更好地访问网站内容。

语法规则：

```
/* 选择器 */
selector {
    /* 声明块 */
    property: value;
    another-property: another-value;
}

/* 示例 */
body {
    background-color: #f4f4f4; /* 背景颜色 */
    font-family: Arial, sans-serif; /* 字体系列 */
}

h1 {
    color: #333; /* 文本颜色 */
    font-size: 2em; /* 字体大小 */
```

```
}
```

详细语法要素：

选择器 (Selectors)

选择器是用来指定要应用样式的HTML元素或元素组的关键部分：

元素选择器：

```
/* 元素选择器根据HTML标签名称来匹配相应类型的元素 */
p {
    color: blue;
}
```

解释： 这个例子展示了元素选择器，它会选择页面上所有的 `<p>` （段落）元素，并将它们的文本颜色设置为蓝色。

ID选择器：

```
/* ID选择器通过元素的id属性精确匹配单个元素 */
#main-header {
    font-size: 24px;
    text-align: center;
}
```

解释： ID选择器用于定位具有特定ID（例如“main-header”）的元素。在HTML中，ID是唯一的，因此这个规则只会影响标记了 `id="main-header"` 的元素。

类选择器：

```css
/* 类选择器匹配所有包含指定类名的元素 */
.highlight {
    background-color: yellow;
}
```

解释： 类选择器应用样式于拥有特定类名的所有元素。任何HTML元素只要设置了类名 `highlight`，其背景色就会被设置为黄色。

组合选择器：

```css
/* 组合选择器结合两个或更多基础选择器来更精准地选择元素 */
.content h2 {
    margin-bottom: 10px;
}
```

解释： 组合选择器将多个选择器链接起来，以同时满足多个条件。在这个例子中，CSS规则应用于属于 `.content` 类的后代 `<h2>` 标题元素。

属性选择器：

```css
/* 属性选择器根据HTML元素的属性及属性值来匹配元素 */
input[type="text"] {
    width: 100%;
    padding: 8px;
}
```

解释： 属性选择器根据元素的属性是否存在及其具体值来选取元素。上述示例中，该规则作用于所有 `type` 属性为 `"text"` 的 `<input>` 元素，为它们设定宽度和内边距样式。

伪类选择器：

```
/* 伪类选择器为元素的不同状态定义样式 */
a:hover {
    color: purple;
}
```

解释： 伪类选择器描述的是元素的一种特殊状态而非永久性特征。这里，当用户鼠标悬停在链接上时，链接的颜色会变为紫色。

声明与声明块

声明与声明块实例：

```
/* 在一个声明块中可以包含多个声明，这些声明共同定义了一个元素的样式 */
div.example-block {
    background-color: #f0f0f0; /* 背景颜色 */
    padding: 20px; /* 内边距 */
    border: 1px solid #ccc; /* 边框 */
    box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1); /* 阴影效果 */
    transition: background-color 0.3s ease-in-out; /* 过渡动画 */
}
```

解释： 此段代码展示了如何在一个声明块中定义一系列样式声明，本例中针对 `div.example-block` 元素设置了多种样式属性。

继承和层叠

继承示例：

```css
body {
    font-family: Arial, sans-serif;
}

/* 子元素默认继承父元素的font-family属性 */
p, h1, h2 {
    /* 其他不相关的样式 */
}
```

解释： CSS中的某些属性如 `font-family` 是可以继承的，这意味着子元素如果没有明确设置字体系列，将会使用其祖先元素（这里是 `body` ）所设定的字体。

高级特性实例

媒体查询：

```css
@media screen and (max-width: 600px) {
    body {
        background-color: lightgreen;
    }
    .sidebar {
        display: none;
    }
}
```

解释： 媒体查询允许根据设备视口大小或其他媒体特性动态调整样式。此例中，当屏幕宽度小于或等于600px时，整个文档背景色将变更为浅绿色，并隐藏带有 `.sidebar` 类的侧边栏元素。

动画与过渡：

```css
/* 使用@keyframes创建动画 */
@keyframes fadeIn {
    0% { opacity: 0; }
    100% { opacity: 1; }
}

```

```css
/* 将动画应用到元素上 */
.box {
    animation: fadeIn 1s ease-in forwards;
}

/* 使用transition进行简单的过渡效果 */
.button {
    transition: background-color 0.5s, transform 0.3s;
}

.button:hover {
    background-color: tomato;
    transform: scale(1.1);
}
```

解释： `@keyframes` 规则用来定义一个动画，此处名为 `fadeIn`，表示从完全透明过渡到完全不透明。然后通过 `animation` 属性将此动画应用到 `.box` 元素上。同时，`.button` 元素利用 `transition` 属性实现了背景色和缩放效果的平滑过渡，在鼠标悬停时触发。

自定义属性（CSS变量）：

```css
:root {
    --primary-color: #007bff;
    --secondary-color: #6c757d;
}

.title {
    color: var(--primary-color);
}

.secondary-text {
    color: var(--secondary-color);
}
```

解释： CSS变量允许开发者定义可以在整个文档范围内复用的样式值。这里，在 `:root` 伪类选择器下定义了两个变量 `--primary-color` 和 `--secondary-color`，并在其他选择器中引用这些变量为元素设置颜色。

结论

通过本课程，学生应能够理解并实现基于 HTTP 的前后端联动机制，了解如何处理和响应前端表单提交，并能够将这些知识应用于实际的服务器项目中。

课后练习

- 尝试添加更多的前端页面和对应的后端处理逻辑。
- 为前端页面添加 CSS 样式，提高用户界面的美观性。