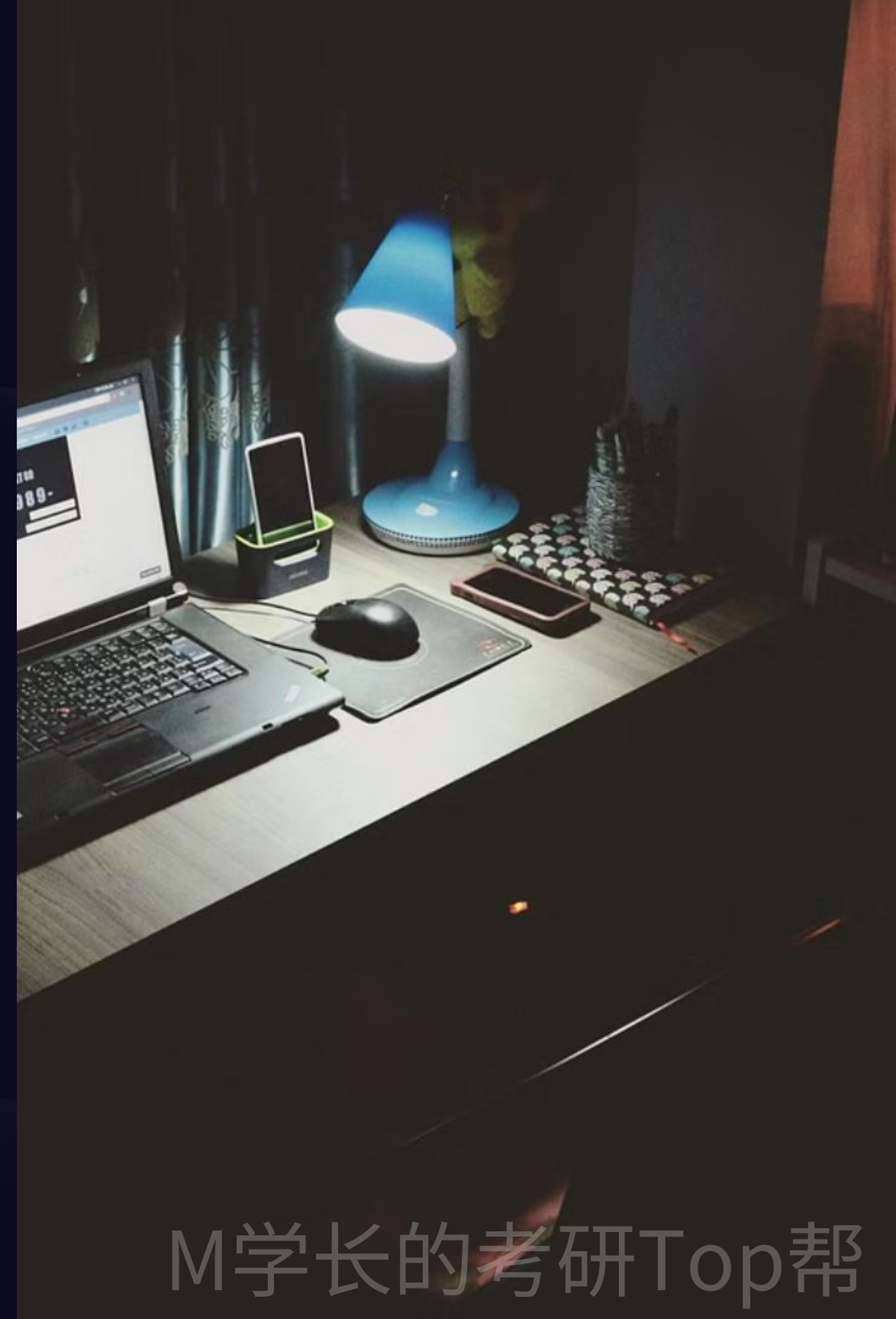


前端精讲

前后端联动

在现代的Web开发实践中，前后端联动是一个关键的构成部分。



M学长的考研Top帮

实现前后端联动

1

前端设计

首先，创建具有良好用户界面的登录和注册页面，让用户可以输入自己的信息。

2

后端处理

服务器将处理前端发送的请求，并返回相应的数据。

3

数据交互

确保前端和后端可以进行高效的数据交互。

前端开发概念

前端开发是Web开发的一个重要分支，主要关注网站或应用程序的用户界面（UI）和用户体验（UX）的设计与实现。前端开发者通过编写HTML、CSS和JavaScript代码来构建网页和应用的客户端部分，确保它们在用户的浏览器上正确显示并具有良好的交互性。

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。

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

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。

HTML 基础教程

HTML定义

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。

功能与应用

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。

学习重要性

掌握HTML是进行网页设计和开发的基础，对于任何希望涉足WEB开发的程序员而言，都是必备的技能。

HTML 基本结构

DOCTYPE声明

DOCTYPE是HTML文档的必要前缀，用于声明文档的类型和版本。

HTML根元素

HTML标签定义了文档的开始和结束。

头部元信息

HEAD元素包含了文档的元数据，比如字符编码、样式表、脚本链接等。不会直接显示在网页上

正文内容

BODY元素包含了网页可见的所有内容，包括文本、图片、表格等。

代码层面

`<!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

`<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。

头部元信息

- 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。

正文

`<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。

`<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。

常用标签

1

标题标签

H1至H6标签定义标题，其中H1表示最高级别标题。

3

链接标签

A标签用于创建超链接，可以连接到不同的网页或网页内部的一个位置。

2

段落标签

P标签定义文本的段落，是HTML中非常基本的元素之一。

4

图像标签

IMG标签用来插入图片，是网页中不可或缺的元素，增加了页面的丰富性与吸引力。

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。
- ``：链接标签，用于创建指向其他页面的链接。
- ``：图像标签，用于嵌入图片。
- ``、``、``：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。
- ``：用于对文档中的行内元素进行分组。

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>
```



www.toolhelper.cn



Html 在线运行 - 在线工具

HTML/CSS/JavaScript在线代码运行的工具，用于在线进行代码测试，你可以在将你的HTML/CSS/JavaScript代码复制到代码框内，实时显示运行结果

表单标签

1 表单创建

FORM标签允许用户输入数据，并将数据提交到服务器进行处理。

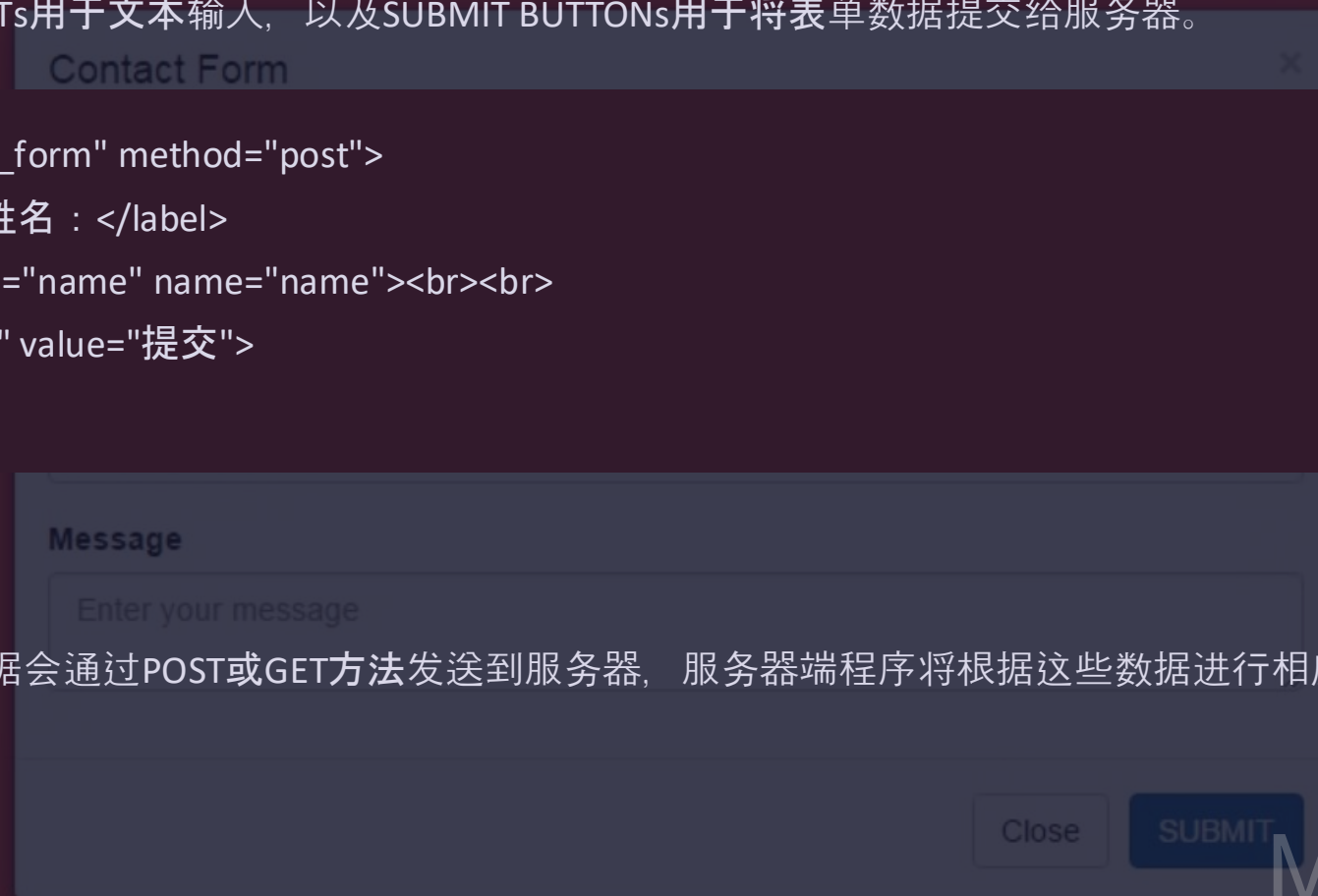
2 标签样例

表单通常包括TEXT INPUTs用于文本输入，以及SUBMIT BUTTONs用于将表单数据提交给服务器。

```
<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>
```

3 数据提交

填写并提交表单后，数据会通过POST或GET方法发送到服务器，服务器端程序将根据这些数据进行相应的处理。



Contact Form

Message

Enter your message

Close SUBMIT

SERVER SIDE FILTERING

using

服务器端代码实现

服务器端的代码实现是完成前后端联动的关键。服务器要高效地响应来自客户端的请求，并提供必要的服务。

By Name or Email

Search

Sort By

Name	Email	Phone	Created	Status
Mark Zuckerberg	mark@gmail.com	7777777777	2016-06-28 08:23:23	Inactive
Jeff Bezos	jeff@gmail.com	9898899889	2016-07-09 08:23:23	Active
Larry Page	larry@gmail.com	9999999999	2016-07-10 08:23:23	Inactive
Narendra Modi	narendra@gmail.com	1234321564	2016-07-14 08:23:23	Active
Bill Gates	bill@gmail.com	8888888888	2016-07-12 08:23:23	Active
Barack Obama	barack@gmail.com	6666666666	2016-07-13 08:23:23	Inactive

静态页面服务

服务器负责提供静态页面，如登录或注册页面。通过编写C++中的Router类，服务器能够对于特定的HTTP GET请求，返回相应的HTML文档。

请求类型	路径	功能描述
GET	/login	显示登录页面
GET	/register	显示注册页面

Router类的处理

```
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;  
});
```

```
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...  
});
```

表单数据处理

接收**POST**请求

当用户提交表单时，服务器通过POST请求接收数据。

1

2

解析表单数据

服务器利用HttpRequest类解析请求体中的数据，例如用户名和密码。

3

验证登录信息

服务器将解析出的数据与数据库中存储的用户信息进行对比，以验证用户的登录请求。

readFile实现

ReadFile函数用于读取给定路径的HTML文件，将内容转换为字符串形式，准备发送给客户端。这个函数在处理静态页面显示请求时非常重要。

```
ator {  
t-icon-wr  
header#top  
ent_page  
ent-menu  
r > .sf-sub  
lover span,  
ent-menu-it  
:art,.ascend  
ffffff!impo  
ul>li.but  
ggle a i  
isparent
```

代码实现

```
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

目的

readFile函数的目的是以字符串形式获取和返回文件内容，使其可以在WEB服务器响应中使用。

方法

该函数通过标准输入输出流（ifstream）打开文件，读取内容，并在操作成功时返回字符串。

错误处理

如果文件无法打开，函数将返回一条错误信息，指出无法打开指定的文件路径。

前后端联动机制

用户操作

用户在客户端进行操作，如填写和提交表单，发起与服务器的交互。

HTTP请求

前端发出HTTP请求，并将包含用户数据的表单发送到服务器。

服务器处理

服务器接收请求并处理，如验证用户凭证或数据库查询。

反馈响应

服务器根据处理结果生成响应，通过HTTP响应发送回前端，前端显示相应信息。

代码解析

1

HttpRequest类

此类负责解析客户端请求，提取出方法、路径以及数据等信息。

2

HttpResponse类

用于构建服务器的响应，包括状态码、响应头以及响应体等信息。

3

Router类

根据请求的路径和方法，决定由哪个处理函数来执行请求的具体逻辑。

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 */
}
```

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。

类选择器：

```
/* 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：

```
/* 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：

```
/* 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。

声明与声明块

```
/* 选择器 */  
div.example-block {  
    /* 声明块 */  
    background-color: #f0f0f0; /* 声明 */  
    padding: 20px; /* 声明 */  
    border: 1px solid #ccc; /* 声明 */  
    box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1); /* 声明 */  
    transition: background-color 0.3s ease-in-out; /* 声明 */  
}
```

/* 这段CSS代码的作用是：为所有类名为 "example-block" 的 div 元素设置背景颜色、内边距、边框样式、阴影效果和过渡动画。 */

继承和层叠

继承示例：

```
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。

课后练习

1

拓展页面

尝试添加更多的前端页面，增加网站的功能性和互动性。

2

增加后端逻辑

在后端添加处理逻辑，增加如用户权限管理等高级功能。

3

美化用户界面

应用CSS样式，提高页面美观，增强用户体验。



谢谢大家

M学长的考研Top帮