山东大学 计算机科学与技术 学院

云计算技术 课程实验报告

学号：202100130103	姓名： 陈一苇	班级： 计机 21.1
实验题目：利用主流云平台搭建个人博客或网站		
实验学时：2	实验日期： 2024.3.22	

实验目的：熟悉使用主流云平台并搭建个人博客或者网站。
具体包括：
参考方案：基于主流云平台，设计、实现个人博客或者网站的搭建，撰写实验报告（附带网站链接并可以访问），并在网站上呈现此次实验报告。

硬件环境：
联网的计算机一台

软件环境：
Windows

实验步骤与内容：

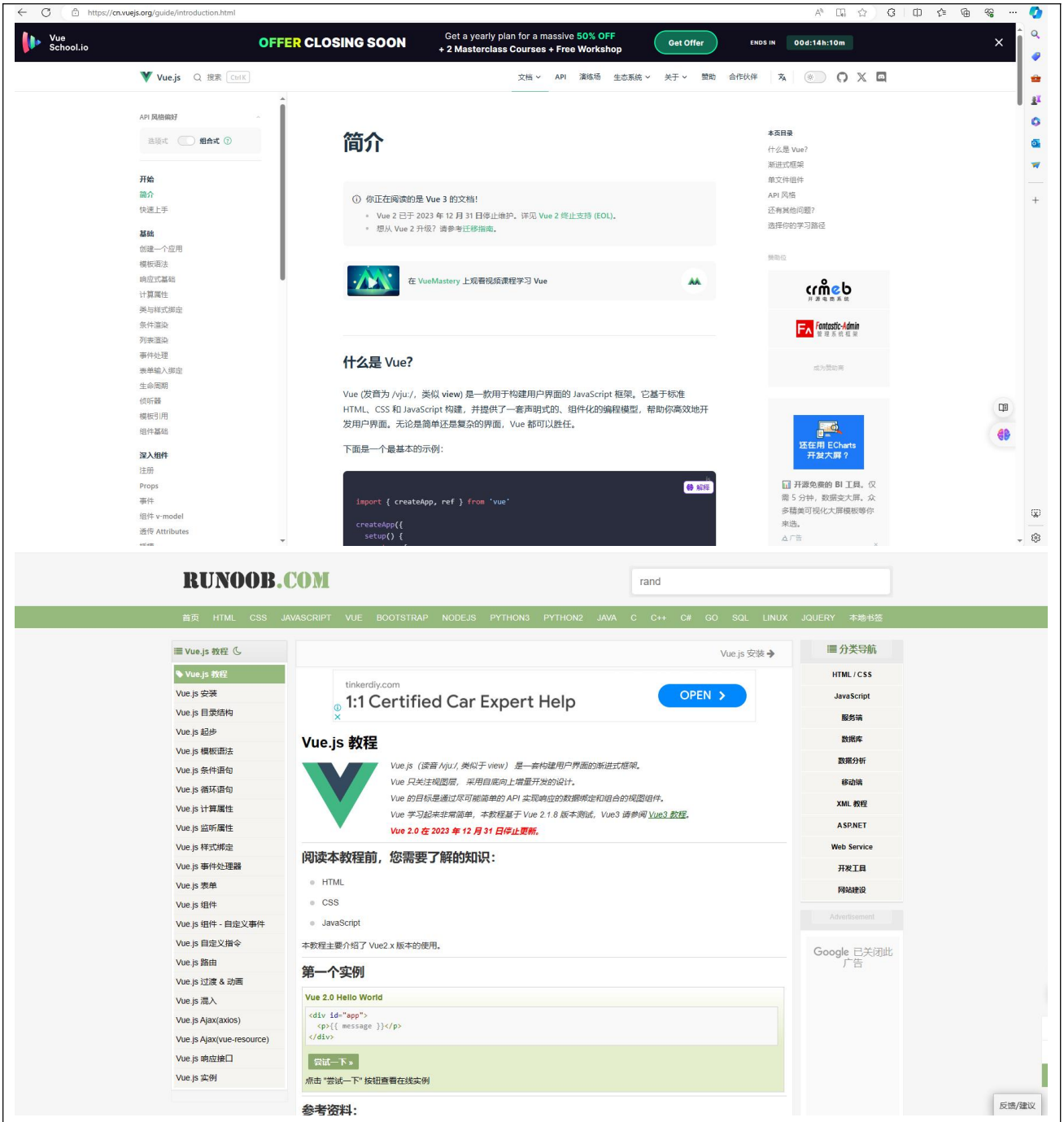
一、了解书写个人博客或网站的方法

1、静态网站生成器（如 Jekyll、Hugo、Hexo）：
优点：简单易用，快速部署，通常不需要服务器端编程。
缺点：功能有限，不如 React+TypeScript 灵活，难以实现复杂的交互式功能。
2、内容管理系统（如 WordPress）：
优点：丰富的插件和主题，适合非技术用户。
缺点：性能可能不如定制的 React+TypeScript 网站，且可能需要更多的维护。
3、网站构建平台（如 Wix、Squarespace）：
优点：拖放界面，无需编程知识。
缺点：定制性和控制力不如 React+TypeScript，可能有月费。
与基于 React+TypeScript 的方法相比，上述方法可能在性能、灵活性和可扩展性方面存在缺点。React+TypeScript 提供了强大的组件化和状态管理能力，适合开发复杂的动态网站和应用。且对于长期项目，虽然 TypeScript 可以减少维护成本。
当然它也有一些潜在的缺点，例如：对于新手来说，React 和 TypeScript 的学习曲线可能比较陡峭；初始开发成本可能较高，因为需要编写更多的类型定义和接口等。

由于我想实现一个高度定制化且功能丰富的网站，所以选用了 React+TypeScript 方法。

二、了解 React 相关知识

参考简介 | Vue.js (vuejs.org)
参考 Vue.js 教程 | 菜鸟教程 (runoob.com)



1、React 基础

组件：理解函数组件和类组件的区别，以及如何使用它们。

JSX：学习 JSX 语法，它允许在 JavaScript 中编写类似 HTML 的代码。

状态管理：理解 state 和 props 的概念，以及如何在组件间传递数据。

2、TypeScript 基础

类型注解：学习如何为变量、函数参数和返回值添加类型注解。

接口和类：使用接口和类来定义对象的结构和类型。

泛型：理解泛型的概念，以及如何在 React 组件中使用它们。

3、React 高级概念

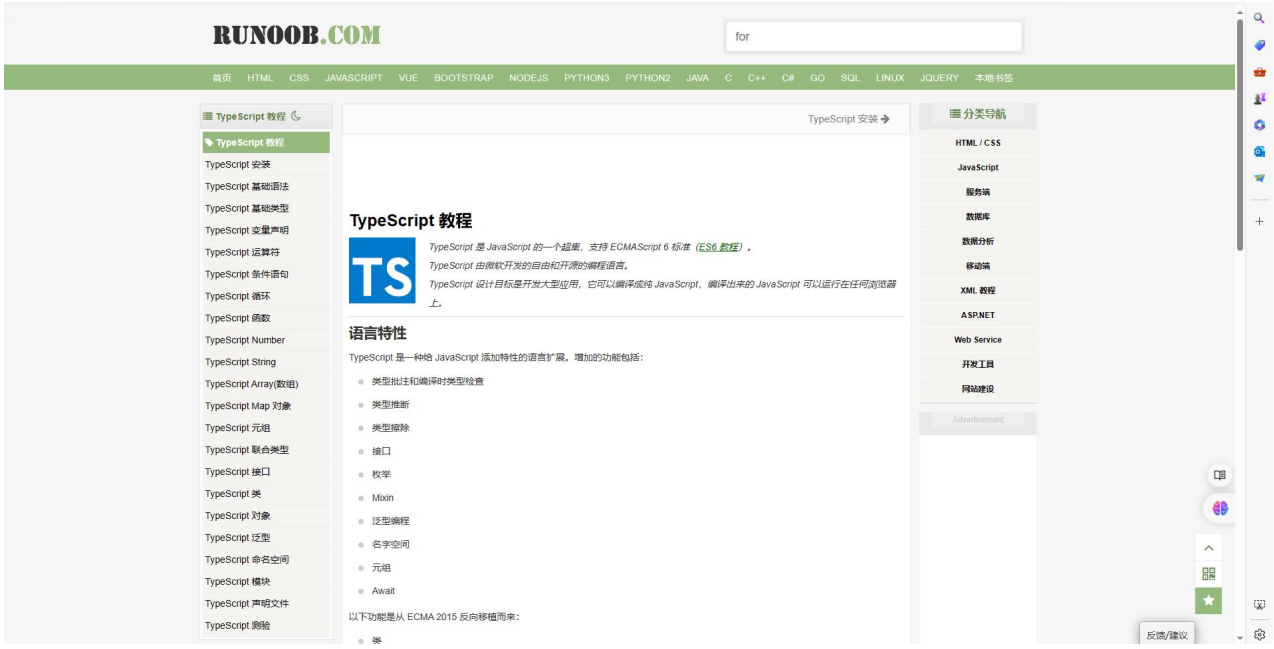
Hooks：学习 useState，useEffect 等常用的 React Hooks，以及自定义 Hooks 的创建。

上下文：使用 React Context API 来跨组件共享状态。

路由：使用 React Router 来处理应用内的导航。
4、构建和部署
Webpack：理解 Webpack 的基本配置和作用。
Babel：学习 Babel 如何将 JSX 和 ES6+ 代码转换为兼容老版本浏览器的 JavaScript。
部署：了解如何将 React 应用部署到服务器或静态网站托管服务。
5、测试
单元测试：学习如何使用 Jest 等测试框架进行单元测试。
端到端测试：了解如何使用 Cypress 或 Selenium 进行端到端测试。
6、性能优化：
代码分割：实现懒加载和代码分割，以提高应用的加载速度。
使用纯组件：通过 PureComponent 或 React.memo 来避免不必要的渲染。
7、样式化
CSS-in-JS：探索如 styled-components 或 emotion 等 CSS-in-JS 库的使用。
CSS 模块：使用 CSS 模块来避免样式冲突。

三、了解 TypeScript 相关知识

参考 TypeScript 教程 | 菜鸟教程 (runoob.com)



1、类型注解
学习如何为变量、函数参数和返回值添加类型注解，以确保类型的正确性。
2、接口（Interfaces）
使用接口定义对象的形状，包括属性和方法的类型。
3、类（Classes）
理解 TypeScript 中的类和继承，以及如何使用它们来创建复杂的类型。
4、泛型（Generics）
学习泛型的概念，以及如何使用泛型来创建可重用的组件和函数。
5、枚举（Enums）
使用枚举来定义一组命名常量。
6、类型推断（Type Inference）
理解 TypeScript 的类型推断机制，以及如何利用它简化代码。

7、类型兼容性 (Type Compatibility)
学习 TypeScript 的结构类型系统，以及类型兼容性的规则。
8、高级类型 (Advanced Types)
探索联合类型、交叉类型、类型别名等高级类型的使用。
9、模块 (Modules)
理解如何使用模块来组织和封装代码。
10、命名空间 (Namespaces)
学习命名空间的概念，以及如何使用它们来避免全局作用域污染。
11、装饰器 (Decorators)
了解装饰器的概念，以及如何使用它们来扩展类的功能。
12、配置 TypeScript 编译器 (tsconfig.json):
学习如何配置 TypeScript 编译器选项，以适应不同的项目需求。
13、工具链
熟悉 TypeScript 的编译过程，以及如何使用工具链（如 TSC、Webpack）。
14、集成开发环境 (IDE) 支持
利用 IDE（如 Visual Studio Code）提供的 TypeScript 支持来提高开发效率。

四、基本步骤
1、环境准备
安装 Node.js 和 npm（Node.js 的包管理器）。
安装代码编辑器，如 Visual Studio Code。
2、项目初始化
使用 create-react-app 脚手架工具创建一个新的 React 项目，并选择 TypeScript 模板。
3、项目结构设置:
配置项目文件夹结构，例如分离组件、样式、工具函数等。
4、编写组件
使用 TypeScript 编写 React 组件，为组件的 props 和 state 定义类型。
5、状态管理
根据需要，使用 Redux 或 Context API 等状态管理工具来管理应用状态。
6、样式处理
使用 CSS 或 CSS-in-JS 库（如 styled-components）来添加样式。
7、路由配置:
使用 React Router 等路由库来设置页面导航。
8、后端服务
如果需要，搭建后端服务，如使用 Express 或 Koa，并与前端进行连接。
9、数据获取
使用 fetch 或 axios 等 HTTP 客户端进行 API 调用，获取数据。
10、测试
编写单元测试和集成测试，确保组件和功能的正确性。
11、构建与部署
使用 Webpack、Babel 等工具优化项目构建。
将构建好的项目部署到服务器或静态网站托管服务。
12、维护与更新
定期更新依赖，修复 bug，添加新功能。

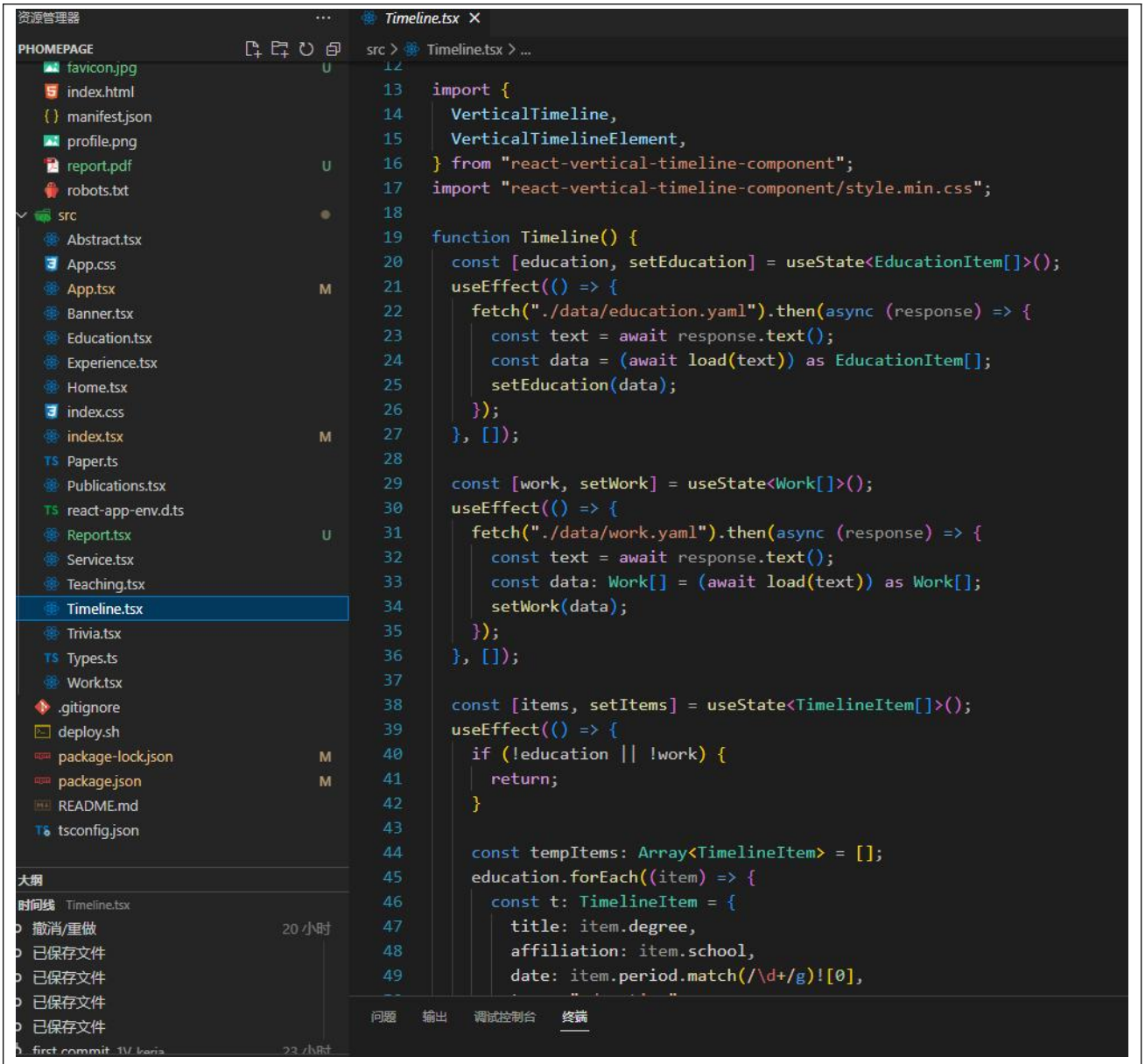
这个项目是用 facebook/create-react-app: Set up a modern web app by running one command. (github.com)引导的

五、关键代码

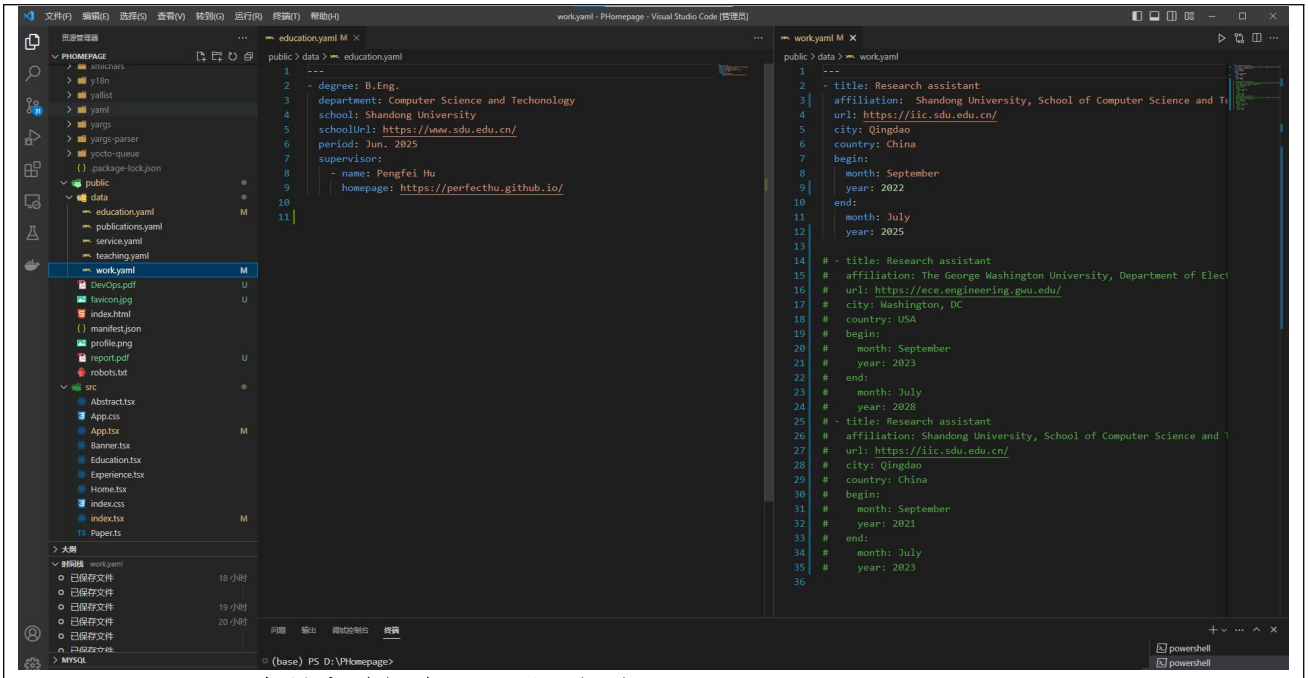
src/Abstract.tsx 中是主页第一页个人介绍部分

```
function Abstract() {
  return (
    <Container style={{ marginTop: "2rem", marginBottom: "1rem" }}>
      <Row className="align-items-center">
        <Col md="auto">
          <img src="./profile.png" width={200} alt="Yiwei Chen" />
        </Col>
        <Col>
          <h2>
            Yiwei Chen
            {/* <sup style={{ fontSize: "60%" }}>
              <FontAwesomeIcon icon={faVolumeUp} onClick={playFirstName} />
            </sup>{" "} */}
          </h2>
          <h5>Undergraduate</h5>
          <h5>School of Computer Science and Technology</h5>
          <h5>Shandong University</h5>
          I am currently an undergraduate student in the School of Computer Science and Technology at Shandong University, in the laboratory of the Insti
          I'm fortunate and grateful to be advised by{" "}
          <a href="https://perfecthu.github.io/">
            Prof. Pengfei Hu
          </a>{" "}.
          I am interested in the field of Security & Privacy, Mobile Computing.
          I am currently involved in research in the field of acoustic eavesdropping in the laboratory.
        </Col>
      </Row>
    </Container>
  );
}
```

src/Timeline.tsx 中是时间线部分



其中引用了\public\data\work.yaml 和\public\data\education.yaml



src/Report.tsx 中是实验报告 PDF 展示部分

Report.tsx U

src > Report.tsx > ...

```tsx
  const [pageNumber, setPageNumber] = useState<number>(1)

  const renderDocumentPage = (num: number, total: number) => {
    if (num <= total) {
      setPageNumber(num)
      requestIdleCallback(() => renderDocumentPage(num + 1, total))
    }
  }

  const onDocumentLoadSuccess = ({ numPages, ...rest }: { numPages: number }) => {
    requestIdleCallback(() => renderDocumentPage(1, numPages))
  }

  return (
    <div
      className={
        'flex h-full w-full justify-center overflow-auto rounded-lg bg-[#525659]'
      }
    >
      <Document
        ref={documentRef}
        file={fileUrl}
        onLoadSuccess={onDocumentLoadSuccess}
        loading="Loading..."
        renderMode="canvas"
      >
        {Array.from({ length: pageNumber }).map((_, i) => (
          <Page pageNumber={i + 1} className="mt-6" loading="Loading..." />
        ))}
      </Document>
    </div>
  )
}

export default Report
```

每一句的解释如下：

```tsx
import { useRef, useState } from 'react'
import { Document, Page, pdfjs } from 'react-pdf'
import 'react-pdf/dist/Page/AnnotationLayer.css'
import 'react-pdf/dist/Page/TextLayer.css'
```

从 react 库中导入 useRef 和 useState 钩子函数。从 react-pdf 库中导入 Document 和 Page 组件以及 pdfjs 对象。导入 react-pdf 库的样式表，用于注释层和文本层的样式。

```tsx
pdfjs.GlobalWorkerOptions.workerSrc = `//unpkg.com/pdfjs-dist@${pdfjs.version}/legacy/build/pdf.worker.min.js`
```

设置 PDF.js 工作线程的源，这是必要的，因为 PDF.js 在浏览器中解析 PDF 文件时使用 Web Workers。

```
interface Props {
  fileUrl: string
}
```

定义一个 TypeScript 接口 Props，它包含一个属性 fileUrl，类型为字符串。

```
const Report = ({ fileUrl }: Props) => {
```

创建一个名为 FilePDF 的 React 函数组件，它接收一个 Props 对象作为参数。

```
const documentRef = useRef<HTMLDivElement>()
```

使用 useRef 钩子创建一个引用，指向包含 PDF 文档的 div 元素。

```
const scale = useRef(1)
```

创建一个引用，用于存储 PDF 页面的缩放比例，默认值为 1。

```
const [pageNumber, setPageNumber] = useState<number>(1)
```

使用 useState 钩子创建一个状态变量 pageNumber 和一个更新它的函数 setPageNumber，初始值为 1。

```
const renderDocumentPage = (num: number, total: number) => {
  if (num <= total) {
    setPageNumber(num)
    requestIdleCallback(() => renderDocumentPage(num + 1, total))
  }
}
```

定义一个函数 renderDocumentPage，用于递归地渲染 PDF 文档的页面，直到达到总页数。

```
const onDocumentLoadSuccess = ({ numPages, ...rest }: { numPages: number }) => {
  requestIdleCallback(() => renderDocumentPage(1, numPages))
}
```

定义一个函数 onDocumentLoadSuccess，当 PDF 文档加载成功时调用，它使用 requestIdleCallback 来开始渲染文档的页面。

```
return (
  <div
    className={
      'flex h-full w-full justify-center overflow-auto rounded-lg bg-[#525659]'
    }
  >
```

返回一个 div 元素，它使用 Tailwind CSS 类来设置样式。

```
<Document
  ref={documentRef}
  file={fileUrl}
  onLoadSuccess={onDocumentLoadSuccess}
  loading="Loading..."
  renderMode="canvas"
>
```

渲染 Document 组件，传入之前创建的引用、文件 URL、加载成功的回调函数等属性。

```
  >
    {Array.from({ length: pageNumber }).map((_, i) => (
      <Page pageNumber={i + 1} className="mt-6" loading="Loading..." />
    ))}
  </Document>
</div>
```

使用 Array.from 和 map 函数来渲染当前页数的 Page 组件。

```
export default Report
```

将 FilePDF 组件作为默认导出。

src/Home.tsx 中是主页邮箱和地址部分

```
function Contact() {
  return (
    <>
      <h3>
        <FontAwesomeIcon icon={faInbox} /> Contact
      </h3>
      <h4>Mail: ywchen at mail dot sdu dot edu dot cn</h4>
      <Button href="https://orcid.org/0009-0004-0445-2335">
        <FontAwesomeIcon icon={faOrcid} /> ORCID
      </Button>{" "}
      <Button href="https://github.com/ywchen789">
        <FontAwesomeIcon icon={faGithub} /> Github
      </Button>{" "}
      {/* <Button href="" rel="me">
        <FontAwesomeIcon icon={faMastodon} /> Mastodon
      </Button>{" "} */}
      <br />
      <br />
      <h4>Address:</h4>
      <span>
        Institute of Intelligent Computing(IIC)<br/>
        The School of Computer Science and Technology<br/>
        Shandong University<br/>
        72 Binhai Road, Jimo, Qingdao, P.R. China
        <br />
      </span>
      <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3213.22773579915431!2d120.68834371195395!3d36.3552622925382!2m3!1f0!2f0!3f0!3m2!1
      title="Map"
      width="100%"
      height="400em"
      style={{ border: 0 }}
      allowFullScreen={false}
      loading="lazy"
      ></iframe>
    </>
  );
}
```

src/index.tsx 和 src/App.tsx 控制主页上方的菜单

index.tsx

```tsx
import React from "react";
import ReactDOM from "react-dom";
// import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import { HashRouter as Router, Routes, Route } from "react-router-dom";
import "./index.css";
import Home from "./Home";
// import Experience from "./Experience";
// import Publications from "./Publications";
// import TeachingExperience from "./Teaching";
// import Service from "./Service";
import Timeline from "./Timeline";
import Report from "./Report";
// import Trivia from "./Trivia";
import "bootstrap/dist/css/bootstrap.min.css";
import "academicons/css/academicons.css";

ReactDOM.render(
  <React.StrictMode>
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        {/* <Route path="/experience" element={<Experience />} />
        <Route path="/publications" element={<Publications />} />
        <Route path="/teaching" element={<TeachingExperience />} />
        <Route path="/service" element={<Service />} />
        <Route path="/trivia" element={<Trivia />} />*/}
        <Route path="/timeline" element={<Timeline />} />
        <Route path="/report" element={<Report fileUrl= "./report.pdf" />}/>
      </Routes>
    </Router>
  </React.StrictMode>,
  document.getElementById("root")
);

```

```jsx
<>
  <Navbar expand="lg" style={{ marginBottom: "1rem" }}>
    <Container>
      <Navbar.Brand as={Link} to="/">
        Yiwei Chen
      </Navbar.Brand>
      <Navbar.Toggle aria-controls="basic-navbar-nav" />
      <Navbar.Collapse id="basic-navbar-nav">
        <Nav>
          {/* <Nav.Link as={Link} to="/experience">
            Experience
          </Nav.Link>
          <Nav.Link as={Link} to="/publications">
            Publications
          </Nav.Link>
          <Nav.Link as={Link} to="/teaching">
            Teaching
          </Nav.Link>
          <Nav.Link as={Link} to="/service">
            Service
          </Nav.Link> */}
          <Nav.Link as={Link} to="/timeline">
            Timeline
          </Nav.Link>
          <Nav.Link as={Link} to="/report">
            Report
          </Nav.Link>

          {/* <Nav.Link as={Link} to="/trivia">
            Trivia
          </Nav.Link> */}
        </Nav>
      </Navbar.Collapse>
    </Container>
  </Navbar>
  <Container style={{ minHeight: "1000px" }}>{props.children}</Container>
```

package. json

```json
{
  "name": "react-homepage",
  "version": "0.1.0",
  "private": true,
  "homepage": "https://ywchen789.github.io/Yiwei_Chen/",
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^1.2.36",
```

六、运行方式

1、本地运行

使用 npm ci 来安装所需的依赖项

本地运行 npm start

打开 http://localhost:3000 查看

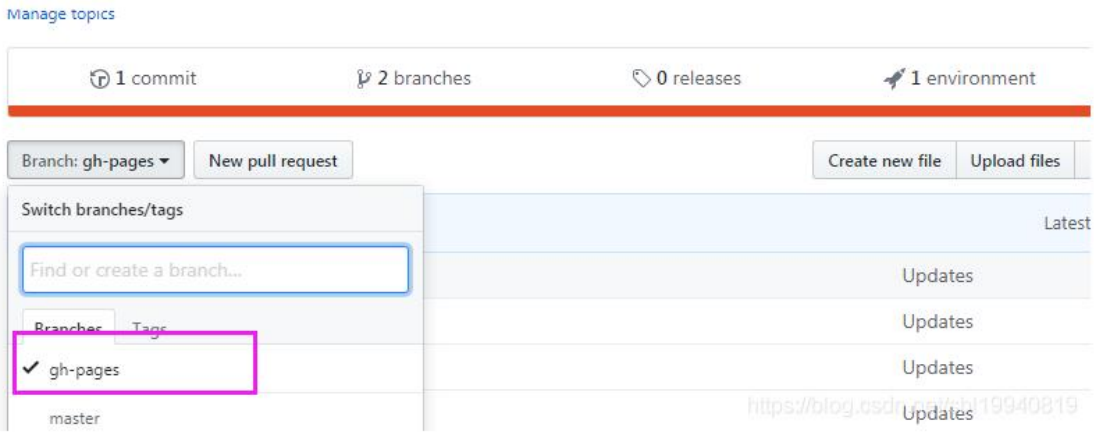
2、deploy 到 github 上

1. 进入到本地代码目录文件
2. 点击git bash

3. 执行如下命令

```
git init
git add .
git commit -m “first commit” （first commit 本次提交的内容）
git remote add origin https://github.com/XXX.git （自己新建的仓库地址）
git push -u origin master （这一句执行的时候，需要根据提示输入你的 github 账号 和密码）
```

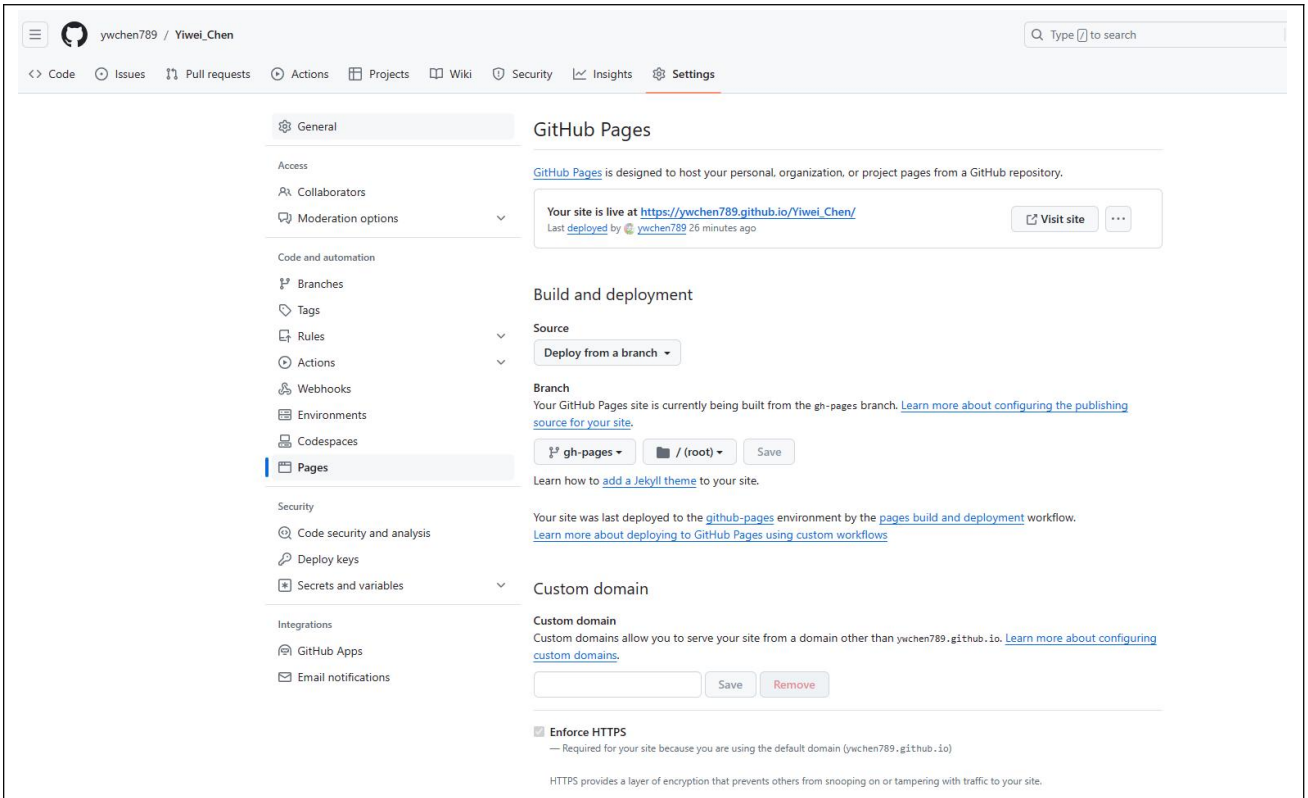
1.配置homepage：
修改你的github仓库地址，例如我的https://github.com/songbl/React-app
“homepage”: “https://songbl.github.io/React-app”, 配置的访问路径
2.配置发布选项
“predeploy”: “npm run build”,
“deploy”: “gh-pages -d build”
predeploy:是将你的项目预编译成静态文件放在build文件夹
deploy:是使用gh-pages 部署你的build文件夹下的内容
####5.安装 gh-pages
npm install gh-pages --save-dev
####6.部署项目到github page上
npm run deploy
####出现下图，就Ok了



由于 package. json 中设置了如下代码

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "predeploy": "npm run build",
  "deploy": "gh-pages -d build"
},
```

运行 npm run deploy



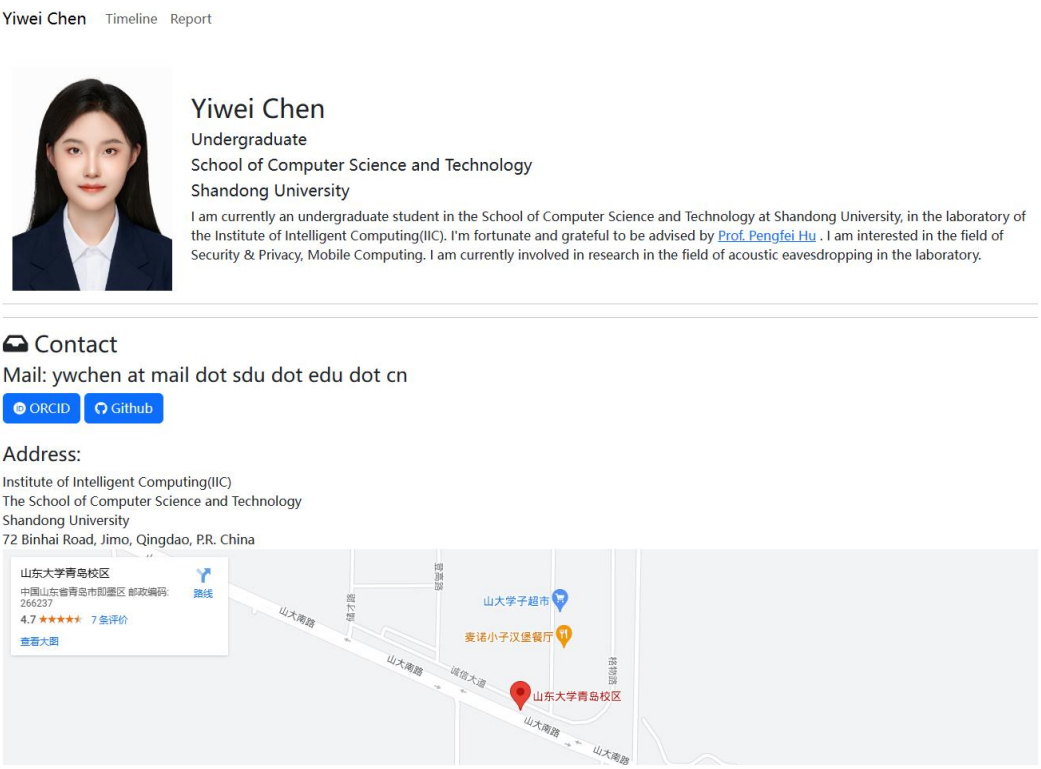
visit site（即 package.json 中的 url）即可访问网页

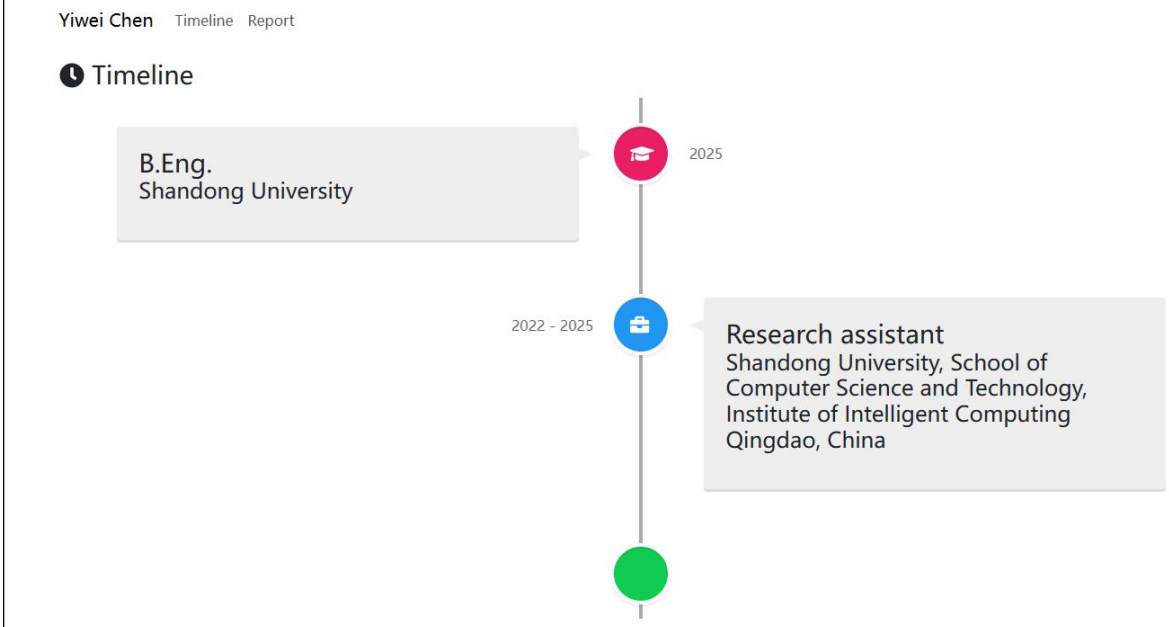
结论分析与体会：

网页网址：Yiwei Chen (ywchen789.github.io)

实验报告请点击“Report”

由于 Address 使用的是谷歌地图，请使用 VPN 查看





一、实验结论

类型安全：TypeScript 的类型系统为项目增加了额外的安全性，减少了运行时错误。
组件化：Vue 的组件化架构使得代码更加模块化，易于管理和维护。
响应式设计：Vue 的响应式系统与 TypeScript 的强类型结合，提供了一个可预测且易于调试的开发环境。
工具生态：Vue CLI 和 TypeScript 的集成提供了强大的开发工具，加速了开发流程。

二、个人体会

学习曲线：初期可能需要一些时间来适应 TypeScript 的类型系统和 Vue 的响应式编程模型。
开发效率：一旦熟悉，你可能会发现开发效率有所提升，特别是在处理大型项目时。
代码质量：类型检查帮助提升了代码质量，使得代码更加健壮和可维护。
社区支持：Vue 和 TypeScript 都有活跃的社区，提供了大量的资源和支持。
总的来说，Vue 和 TypeScript 的结合为开发个人主页提供了一个强大且灵活的框架，使得开发过程更加顺畅，同时也保证了代码的质量和可维护性。这种技术栈的选择，对于追求高效和质量的开发者来说，是一个非常不错的选择。