

珠峰前端架构课

正式课上课时间：周二、周四晚 8:00-10:30 周日下午 2:00 - 5:00

- 本期正式课价格非常优惠
- 下周发布最新大纲

一.pnpm 管理项目

为什么选择它？

- 快：pnpm 是同类工具速度的将近 2 倍
- 高效：node_modules 中的所有文件均链接自单一存储位置
- 支持单体仓库：monorepo，单个源码仓库中包含多个软件包的支持
- 权限严格：pnpm 创建的 node_modules 默认并非扁平结构，因此代码无法对任意软件包进行访问

二.Vite 介绍

- 极速的服务启动，使用原生 ESM 文件，无需打包！（原来整个项目的代码打包在一起，然后才能启动服务）
- 轻量快速的热重载 无论应用程序大小如何，都始终极快的模块热替换（HMR）

- 丰富的功能 对 TypeScript、JSX、CSS 等支持开箱即用。
- 优化的构建 可选“多页应用”或“库”模式的预配置 Rollup 构建
- 通用的插件 在开发和构建之间共享 Rollup-superset 插件接口。
- 完全类型化的 API 灵活的 API 和完整 TypeScript

Vite3 修复了 400+issue,减少了体积, Vite 决定每年发布一个新的版本

三.项目初始化

```
pnpm init # 初始化package.json
pnpm install vite -D # 安装vite
```

1.package.json

增添启动命令

```
"scripts": {
  "dev": "vite",
  "build": "vite build"
}
```

2.index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge" />
    <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
    <title>vite-start</title>
  </head>
  <body>
    <!-- 稍后vue项目挂载到这个元素上 -->
    <div id="app"></div>
    <!-- vite 是基于esModule的 -->
    <script type="module" src="/src/main.ts">
</script>
  </body>
</html>
```

3.main.ts

```
pnpm install vue # 安装vue
```

```
import { createApp } from "vue"
import App from "./App.vue" // 这里会报错, 不支持.vue
createApp(App).mount("#app")
```

4.env.d.ts

```
declare module "*.vue" {
  import type { DefineComponent } from "vue"
  const component: DefineComponent<{}, {}, any>
  export default component
}
```

5.vite.config.ts

我们需要让 vite 支持.vue 文件的解析

```
pnpm install @vitejs/plugin-vue -D
```

```
import { defineConfig } from "vite"
import vue from "@vitejs/plugin-vue"
export default defineConfig({
  plugins: [vue()]
})
```

6.vue-tsc

- Vite 仅执行 `.ts` 文件的转译工作，并不执行任何类型检查。[vue-tsc](#) 可以对 Vue3 进行 Typescript 类型校验

```
pnpm install typescript vue-tsc -D
```

创建 `tsconfig.json`

```
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    "moduleResolution": "node",
    "strict": true,
    "sourceMap": true,
    "jsx": "preserve",
    "esModuleInterop": true,
    "lib": ["esnext", "dom"]
  },
  "include": ["src/**/*.ts", "src/**/*.d.ts",
    "src/**/*.tsx", "src/**/*.vue"]
}
```

```
"scripts": {  
  "dev": "vite",  
  "build": "vue-tsc --noEmit &&vite build"  
},
```

此时再次运行 `pnpm build` 则会对文件内容进行 `ts` 检测

四.EsLint配置

开发项目需要安装 `vscode` 插件 `volar`

```
npx eslint --init
```

1.校验语法并提示错误行数

```
? How would you like to use ESLint? ...  
  To check syntax only  
> To check syntax and find problems  
  To check syntax, find problems, and enforce  
code style
```

2.采用js-module

```
? What type of modules does your project use?  
...  
> JavaScript modules (import/export)  
CommonJS (require/exports)  
None of these
```

3.项目采用vue语法

```
? Which framework does your project use? ...  
React  
> Vue.js  
None of these
```

```
pnpm i eslint-plugin-vue@latest @typescript-  
eslint/eslint-plugin@latest @typescript-  
eslint/parser@latest eslint@latest -D
```

支持 vue 中 ts eslint 配置

```
pnpm i @vue/eslint-config-typescript -D
```

4..eslintrc.js

```
module.exports = {  
  env: {  
    browser: true,
```

```
    es2021: true,
    node: true
  },
  extends: [
    "eslint:recommended",
    "plugin:vue/vue3-essential", // vue3解析
https://eslint.vuejs.org/
    "plugin:@typescript-eslint/recommended"
  ],
  parser: "vue-eslint-parser", // 解析 .vue文件
  parserOptions: {
    parser: "@typescript-eslint/parser", // 解析
    .ts 文件
    ecmaVersion: "latest",
    sourceType: "module"
  },
  plugins: ["vue", "@typescript-eslint"],
  rules: {}
}
```

5. .eslintignore 配置


```
node_modules
dist
*.css
*.jpg
*.jpeg
*.png
*.gif
*.d.ts
```

最终安装 `vscode` 中 `eslint` 插件: `eslint` 只是检测代码规范

```
"lint": "eslint --fix --ext .ts,.tsx,.vue src -  
-quiet"
```

五.Prettier 配置

1.eslint 中进行配置

在 eslint 中集成 prettier 配置

```
pnpm install prettier eslint-plugin-prettier  
@vue/eslint-config-prettier -D
```

```
module.exports = {  
  "env": {  
    "browser": true,
```

```

    "es2021": true,
    "node": true
  },
  "extends": [
    "eslint:recommended",
    "plugin:vue/vue3-essential", // vue3解析
https://eslint.vuejs.org/
    "plugin:@typescript-eslint/recommended",
+    "@vue/prettier"
  ],
  "parser": "vue-eslint-parser", // 解析 .vue文件
  "parserOptions": {
    "parser": '@typescript-eslint/parser',
    // 解析 .ts 文件
    "ecmaVersion": "latest",
    "sourceType": "module"
  },
  "plugins": [
    "vue",
    "@typescript-eslint"
  ],
+  rules: {
+    "prettier/prettier": [
+      "error",
+      {
+        singleQuote: false, //使用单引号

```

```

+         semi: false, /////末尾添加分号
+         tabWidth: 2,
+         trailingComma: "none",
+         useTabs: false,
+         endOfLine: "auto"
+     }
+ ]
}
}

```

2. .prettierrc.js

```

module.exports = {
  singleQuote: false, //使用单引号
  semi: false, /////末尾添加分号
  tabWidth: 2,
  trailingComma: "none",
  useTabs: false,
  endOfLine: "auto"
}

```

.prettierignore

```

node_modules
dist

```

最终安装vscode中Prettier插件：prettier只是用来格式化代码

这里需要配置Format On Save 为启用，保存时自动格式化 Default Formatter选择Prettier - Code formatter

3.editorconfig

`.editorconfig`

```
root = true

[*]
charset = utf-8
indent_style = space
indent_size = 2
end_of_line = lf
```

最终安装vscode中EditorConfig for VS Code插件

六.husky

```
git init
pnpm install husky -D
npm set-script prepare "husky install"
npx husky add .husky/pre-commit "pnpm lint"
```

七.commitlint

类型	描述
build	主要目的是修改项目构建系统(例如 glup, webpack, rollup 的配置等)的提交
chore	不属于以上类型的其他类型
ci	主要目的是修改项目继续集成流程(例如 Travis, Jenkins, GitLab CI, Circle 等)的提交
docs	文档更新
feat	新功能、新特性；
fix	修改 bug；
perf	更改代码，以提高性能；
refactor	代码重构（重构，在不影响代码内部行为、功能下的代码修改）；
revert	恢复上一次提交；
style	不影响程序逻辑的代码修改(修改空白字符，格式缩进，补全缺失的分号等，没有改变代码逻辑)
test	测试用例新增、修改；

代码提交检测

```
pnpm install @commitlint/cli @commitlint/config-conventional -D
npx husky add .husky/commit-msg "npx --no-install commitlint --edit $1"
```

commitlint.config.js配置

```
module.exports = {
  extends: [ "@commitlint/config-conventional" ]
}
```

git commit -m"feat: 初始化工程"

八.路由配置

```
import { createRouter, createWebHistory } from
"vue-router"
const getRoutes = () => {
  // 简单格式化
  const files =
import.meta.glob("../views/*.vue")
  const routes =
Object.entries(files).map(([file, module]) => {
  const name =
file.match(/\.\.\./views\/([^\.]*)\.vue/i)?.[1]
  return {
    path: "/" + name,
```

```
    component: module
  }
})
return routes
}
// 创建路由配置
const router = createRouter({
  history: createWebHistory(),
  routes: getRoutes()
})
export default router
```

```
/// <reference types="vite/client" />
```

需要引入 `vite/client` 得到 ts 支持

```
import router from "./router"
createApp(App).use(router).mount("#app")
```

九.编写 Todo 功能

```
<template>
  <div>
    <input v-model="todo" type="text" />
    <button @click="addTodo">添加内容</button>
    <ul>
```

```
      <li v-for="(item, index) in todos"
:key="index">{{ item }}</li>
    </ul>
  </div>
</template>
<script lang="ts" setup>
const todo = ref("")
const todos = ref<string[]>([])
const addTodo = () => {
  if (!todo.value) return
  todos.value.push(todo.value)
}
</script>
```

1.自动引入插件

```
pnpm install -D unplugin-auto-import
```



```
import AutoImport from "unplugin-auto-import/vite"
export default defineConfig({
  plugins: [
    vue(),
    AutoImport({ imports: [ "vue", "vue-router" ],
    eslintrc: { enabled: false } })
  ]
})
```

.eslintrc

```
extends: [
  "eslint:recommended",
  "plugin:vue/vue3-recommended", // vue3解析
https://eslint.vuejs.org/
  "plugin:@typescript-eslint/recommended",
  "@vue/typescript/recommended",
  "@vue/prettier",
+   "./.eslintrc-auto-import.json"
]
```

tsconfig.json

```
"include": [  
  "src/**/*.*ts",  
  "src/**/*.*d.ts",  
  "src/**/*.*tsx",  
  "src/**/*.*vue",  
  "./auto-imports.d.ts"  
]
```

2. 路径别名

```
export default defineConfig({  
  resolve: {  
    alias: [{ find: "@", replacement:  
path.resolve(__dirname, "src") }]  
  }  
})
```

```
"compilerOptions": {  
  "target": "esnext",  
  "module": "esnext",  
  "moduleResolution": "node",  
  "strict": true,  
  "sourceMap": true,  
  "jsx": "preserve",  
  "esModuleInterop": true,  
  "lib": ["esnext", "dom"],
```

```
    "baseUrl": ".",  
    "paths": {  
      "@/*": ["src/*"]  
    }  
  },  
},
```

3.识别 TSX 文件

```
import { defineComponent, PropType } from "vue"  
export default defineComponent({  
  props: {  
    todos: {  
      type: Array as PropType<string[]>,  
      default: () => []  
    }  
  },  
  render() {  
    return (  
      <ul>  
        {this.todos.map((todo, index) => (  
          <li key={index}>{todo}</li>  
        ))}  
      </ul>  
    )  
  }  
})
```

```
pnpm install @vitejs/plugin-vue-jsx -D
```

```
import jsx from "@vitejs/plugin-vue-jsx"
export default defineConfig({
  plugins: [
    vue(),
    jsx(),
    AutoImport({ imports: ["vue", "vue-router"],
    eslintrc: { enabled: false } })
  ],
  resolve: {
    alias: [{ find: "@", replacement:
path.resolve(__dirname, "src") }]
  }
})
```

九.unocss

Atomic CSS原子 CSS 是一种 CSS 架构方法，传统方法使用预处理器编译后生成样式，但是体积大。（类似行内样式，但是行内样式缺点：冗余）

- **Tailwind** 依赖 PostCSS 和 Autoprefixer + **purgeCSS**, 开发环境 css 体积大
- **Windi CSS** 是一种 Tailwind CSS 替代品，不依赖，按需使用。采用预扫描的方式生成样式。但是自定义复杂~~

- `unocss` 是原子 CSS 引擎，规则定义简单易读。支持预设、支持属性、纯 css 图标。

[unocss](#)

```
pnpm install unocss -D
```

```
import { defineConfig } from "vite"
import vue from "@vitejs/plugin-vue"
import jsx from "@vitejs/plugin-vue-jsx"
import Unocss from "unocss/vite"
import { presetUno, presetAttributify,
presetIcons } from "unocss"
export default defineConfig({
  plugins: [
    vue(),
    jsx(),
    Unocss({
      /* options */
      // presetUno 默认预设
      presets: [presetUno(),
presetAttributify(), presetIcons()]
    })
  ]
})
```

```
import "uno.css"
```

```

<template>
  <h1 text-center>TodoList</h1>
  <div flex items-center justify-center>
    <input v-model="todo" type="text" shadow-
inset shadow shadow-green h-25px />
    <button @click="addTodo" ml-10px inline-
block>添加内容</button>
  </div>
  <TodoList :todos="todos"></TodoList>
</template>
<script lang="ts" setup>
import TodoList from "../todo-list"
const todo = ref("")
const todos = ref<string[]>([])
const addTodo = () => {
  todos.value.push(todo.value)
}
</script>

```

```

import { defineComponent, PropType } from "vue"
export default defineComponent({
  props: {
    todos: {
      type: Array as PropType<string[]>,
      default: () => []
    }
  },

```

```

render() {
  return (
    <ul class="bg-gray-200 w-50% m-auto mt-20">
      {this.todos.map((todo, index) => (
        <li key={index} class={"pl-" + index * 5}>
          {todo}
        </li>
      ))}
    </ul>
  )
}
})

```

```
@iconify-json/ep
```

```
import "@iconify-json/ep"
```

<https://icones.js.org/collection/ep>

```
<div class="i-ep-edit"></div>
```

```

import { defineConfig } from "vite"
import vue from "@vitejs/plugin-vue"
import jsx from "@vitejs/plugin-vue-jsx"
import Unocss from "unocss/vite"

```

```

import { presetUno, presetAttributify,
presetIcons } from "unocss"
const paddingLeft = Array.from({ length: 100 },
(_, i) => "pl-" + i * 5)
const safelist = [...paddingLeft]
export default defineConfig({
  plugins: [
    vue(),
    jsx(),
    Unocss({
      safelist,
      rules: [[/^z-(\d+)$/, ([, d]) => ({
margin: `${(d as any) / 4}rem` } )]],
      shortcuts: {
        btn: "py-2 px-4 font-semibold rounded-lg
shadow-md bg-blue-3 text-#fff"
      },
      // presetUno 默认预设
      presets: [
        presetUno(),
        presetAttributify(),
        presetIcons({
          collections: {
            zf: {
              circle: `<svg width="50"
height="50" viewBox="0 0 50 50">

```



```

        <circle cx="25" cy="25" r="20"
/>

        </svg>`
    },
    extraProperties: {
        display: "inline-flex",
        width: "2em",
        height: "2em",
        "vertical-align": "middle"
    },
    customizations: {
        iconCustomizer(collection, icon,
props) {
            // customize all icons in this
collection

            if (collection === "ep") {
                props.width = "4em"
                props.height = "4em"
            }
        }
    }
})
]
})
]
})

```