link: null
title: 珠峰架构师成长计划
description: vite.config.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=159 sentences=314, words=1946

1.Vite

- Vite (法语意为 "快速的"，发音 /vit/) (https://cn.vitejs.dev/)是下一代前端开发与构建工具
- 💡 极速的服务启动 使用原生 ESM 文件，无需打包!
- ⚡ 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重载（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 TypeS

2.配置开发环境

```
npm install vue  --save
npm install  @vitejs/plugin-vue @vue/compiler-sfc vite --save-dev
```

vite.config.js

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'

export default defineConfig({
  plugins: [vue()]
})
```

package.json

```
{
  "name": "vite2-prepare",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite preview"
  },
  "dependencies": {
    "vue": "^3.0.5"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^1.2.4",
    "@vue/compiler-sfc": "^3.0.5",
    "vite": "^2.4.0"
  }
}
```

index.html

```

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite Apptitle>
  head>
  <body>
    <div id="app">div>
    <script type="module" src="/src/main.js">script>
  body>
html>
```

src\main.js

```
import { createApp } from 'vue'
import App from './App.vue'
createApp(App).mount('#app')
```

src\App.vue

```

//https://github.com/vuejs/rfcs/blob/master/active-rfcs/0040-script-setup.md
import HelloWorld from './components/HelloWorld.vue'
```

src\components\HelloWorld.vue

```

  {{ msg }}
```

3.静态资源处理

src\App.vue

```

+
+

//https://github.com/vuejs/rfcs/blob/master/active-rfcs/0040-script-setup.md
import HelloWorld from './components/HelloWorld.vue'
<span class="hljs-addition">+import imgUrl from './assets/avatar.jpg'</span>
```

```
+

//https://github.com/vuejs/rfcs/blob/master/active-rfcs/0040-script-setup.md
import HelloWorld from './components/HelloWorld.vue'
import imgUrl from './assets/avatar.jpg'

+</span>
<span class="hljs-addition">+.avatar{</span>
<span class="hljs-addition">+  width:200px;</span>
<span class="hljs-addition">+  height:200px;</span>
<span class="hljs-addition">+  background-image: url(./assets/avatar.jpg);</span>
<span class="hljs-addition">+  background-size: contain;</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+
```

- public目录 (https://cn.vitejs.dev/guide/assets.html#the-public-directory)
- 如果有以下需求
    - 这些资源不会被源码引用（例如 robots.txt）
    - 这些资源必须保持原有文件名（没有经过 hash）
- 那么你可以将该资源放在指定的 `public` 目录中，它应位于你的项目根目录
- 该目录中的资源在开发时能直接通过 `/` 根路径访问到，并且打包时会被完整复制到目标目录的根目录下

`public\avatar.jpg`

4.配置别名

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
+import {resolve} from 'path';

// https://vitejs.dev/config/
export default defineConfig({
+ resolve:{
+   alias:{
+    '@':resolve('src')
+   }
+ },
  plugins: [vue()]
})
```

src\App.vue

```
+

//https://github.com/vuejs/rfcs/blob/master/active-rfcs/0040-script-setup.md
<span class="hljs-addition">+import HelloWorld from "@/components/HelloWorld.vue";</span>
<span class="hljs-addition">+import avatarUrl from "@/assets/avatar.jpg";</span>

.avatar {
  width: 200px;
  height: 200px;
<span class="hljs-addition">+ background-image: url(@/assets/avatar.jpg);</span>
  background-size: contain;
}
```

5.样式处理

src\main.js

```
import { createApp } from 'vue'
import App from './App.vue'
+import './global.css'
createApp(App).mount('#app')
```

src\global.css

```
#app {
    background-color: lightgrey;
}
```

- 当 `<style></code>` 标签有 `<code>scoped</code>` 属性时，它的 `CSS` 只作用于当前组件中的元素</li><li>它使用了<code>data-v-hash</code>的方式来使css有了它对应模块的标识</li></ul><p>src\components\HelloWorld.vue</p><pre><code class="lang-diff"><template> <h1>{{ msg }}</h1> <span class="hljs-addition">+ <a>超链接</a></span> </template> <span class="hljs-addition">+<style scoped></span> <span class="hljs-addition">+a {</span> <span class="hljs-addition">+ color: #42b983;</span> <span class="hljs-addition">+}</span> <span class="hljs-addition">+</style>`src\components\HelloWorld.vue

```

  {{ msg }}
+ 超链接

+</span>
<span class="hljs-addition">+.link {</span>
<span class="hljs-addition">+  color: #42b983;</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+
```

- 任何以 .module.css 为后缀名的 CSS 文件都被认为是一个 CSS modules 文件
- 导入这样的文件会返回一个相应的模块对象 src\components\HelloWorld.vue

```

  {{ msg }}
+ 超链接

<span class="hljs-addition">+import style from './HelloWorld.module.css';</span>

```

src\components\HelloWorld.module.css

```
.link {
    color: #42b983;
}
```

- Vite 也同时提供了对 .scss, .sass, .less, .styl 和 .stylus 文件的内置支持。没有必要为它们安装特定的 Vite 插件，但必须安装相应的预处理器依赖
- 如果是用的是单文件组件，可以通过 `style lang="sass"`（或其他预处理器）自动开启

```
npm i less sass -S
```

src\components\HelloWorld.vue

```

  {{ msg }}
  超链接
+ less
+ sass

import { reactive } from 'vue'
import style from './HelloWorld.module.css';

+</span>
<span class="hljs-addition">+@color:red;</span>
<span class="hljs-addition">+h2{</span>
<span class="hljs-addition">+  color:@color;</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+
+</span>
<span class="hljs-addition">+$color:green;</span>
<span class="hljs-addition">+h3{</span>
<span class="hljs-addition">+  color:$color;</span>
<span class="hljs-addition">+}</span>

```

- postcss (https://cn.vitejs.dev/guide/features.html#postcss)
- 如果项目包含有效的 PostCSS 配置 (任何受 postcss-load-config 支持的格式，例如 postcss.config.js)，它将会自动应用于所有已导入的 CSS

```
npm install autoprefixer --save
```

```
module.exports = {
  plugins: [
      require('autoprefixer')
  ]
}
```

```
>0.2%
not dead
not op_mini all
```

src\components\HelloWorld.vue

```

  {{ msg }}
  超链接
  less
  sass
+

import { reactive } from 'vue'
import style from './HelloWorld.module.css';

@color:red;
h2{
  color:@color;
}

$color:green;
h3{
  color:$color;
}

+</span>
<span class="hljs-addition">+.postcss{</span>
<span class="hljs-addition">+    height:50px;</span>
<span class="hljs-addition">+    width:200px;</span>
<span class="hljs-addition">+    background-color: orange;</span>
<span class="hljs-addition">+    transform: rotate(90deg);</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+
```

6.typescript

```
cnpm install typescript @babel/core @babel/preset-env  @babel/preset-typescript --save-dev
```

.babelrc

```
{
    "presets": [
        ["@babel/preset-env"],
        "@babel/preset-typescript"
    ]
}
```

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

src\components\HelloWorld.vue

```
  {{ msg }}
  less
  sass

+ {{state.count}}

import { reactive,defineProps } from 'vue'
<span class="hljs-addition">+defineProps({</span>
<span class="hljs-addition">+  msg:String</span>
<span class="hljs-addition">+})</span>
<span class="hljs-addition">+interface State {</span>
<span class="hljs-addition">+  count:number;</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+let state = reactive<State>({count:0});</span>
<span class="hljs-addition">+const handleClick = ()=>{</span>
<span class="hljs-addition">+  console.log(state.count);</span>
<span class="hljs-addition">+  state.count++;</span>
<span class="hljs-addition">+}</span>

@color:red;
h2{
  color:@color;
}

$color:green;
h3{
  color:$color;
}

.postcss{
    height:50px;
    width:200px;
    background-color: orange;
    transform: rotate(90deg);
}
```

- 让typescript识别支持 `.vue`文件 src\shims-vue.d.ts

```
declare module '*.vue' {
  import { DefineComponent } from 'vue'
  const component: DefineComponent
  export default component
}
```

- 如果你的库依赖于某个全局库，使用/// 指令
- 三斜线指令仅可放在包含它的文件的最顶端
- 三斜线引用告诉编译器在编译过程中要引入的额外的文件 src\vite-env.d.ts **7.配置代理**
- server-proxy (https://cn.vitejs.dev/config/#server-proxy)
- 为开发服务器配置自定义代理规则
- 期望接收一个 { key: options } 对象。如果 key 值以 ^ 开头，将会被解释为 RegExp。configure 可用于访问 proxy 实例。

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { resolve } from 'path';

// https://vitejs.dev/config/
export default defineConfig({
  resolve: {
    alias: {
      '@': resolve('src')
    }
  },
+ server: {
+   proxy: {
+     '/api': {
+       target: 'http://jsonplaceholder.typicode.com',
+       changeOrigin: true,
+       rewrite: (path) => path.replace(/^\/api/, '')
+     }
+   }
+ },
  plugins: [vue()]
})
```

src\App.vue

```
//https://github.com/vuejs/rfcs/blob/master/active-rfcs/0040-script-setup.md
import HelloWorld from "@/components/HelloWorld.vue";
import avatarUrl from "@/assets/avatar.jpg";
<span class="hljs-addition">+fetch('/api/todos/1')</span>
<span class="hljs-addition">+  .then(response => response.json())</span>
<span class="hljs-addition">+  .then(json => console.log(json))</span>

.avatar {
  width: 200px;
  height: 200px;
  background-image: url(@/assets/avatar.jpg);
  background-size: contain;
}
```

8.mock

```
npm i mockjs vite-plugin-mock -D
node ./node_modules/vite-plugin-mock/node_modules/esbuild/install.js
```

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { resolve } from 'path';
+import { viteMockServe } from "vite-plugin-mock";
// https://vitejs.dev/config/
export default defineConfig({
  resolve: {
    alias: {
      '@': resolve('src')
    }
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://jsonplaceholder.typicode.com',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, '')
      }
    }
  },
+  plugins: [vue(),viteMockServe({})]
})
```

mock\test.ts

```
import { MockMethod } from 'vite-plugin-mock';
export default [
    {
        url: '/api/get',
        method: 'get',
        response: ({ query }) => {
            return {
                code: 0,
                data: {
                    name: 'vben',
                },
            };
        },
    },
] as MockMethod[];
```

9.ESLint

- ESLint是一个开源的 JavaScript 的 linting 工具
    - 代码质量问题：使用方式有可能有问题
    - 代码风格问题：风格不符合一定规则

```
npm install eslint eslint-plugin-vue  @vue/eslint-config-typescript @typescript-eslint/parser @typescript-eslint/eslint-plugin --save-dev
```

src\components\HelloWorld.vue

```
  {{ msg }}
  less
  sass

    {{ state.count }}

import { reactive,defineProps } from 'vue'
defineProps({
<span class="hljs-addition">+ msg:{</span>
<span class="hljs-addition">+   type:String,</span>
<span class="hljs-addition">+   default:''</span>
<span class="hljs-addition">+ }</span>
})
interface State {
  count:number;
}
let state = reactive<State>({count:0});
const handleClick = ()=>{
  console.log(state.count);
  state.count++;
}

@color:red;
h2{
  color:@color;
}

$color:green;
h3{
  color:$color;
}

.postcss{
    height:50px;
    width:200px;
    background-color: orange;
    transform: rotate(90deg);
}
```

src\main.ts

```
import { createApp } from 'vue'
import App from './App.vue'
import './global.css'
createApp(App).mount('#app')
```

.eslintrc.js

```
module.exports = {
    root: true,
    env: {
        browser: true,
        es2021: true,
        node: true
    },
    extends: [
        "plugin:vue/vue3-recommended",
        "eslint:recommended",
        "@vue/typescript/recommended"
    ],
    parserOptions: {
        ecmaVersion: 2021
    },
    rules: {
       "no-unused-vars": "off",
       "@typescript-eslint/no-unused-vars": "off",
    }
}
```

.eslintignore

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

package.json

```
{
  "name": "zhufeng-vite2-prepare",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite preview",
+   "lint":"eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --quiet",
+   "lint:fix":"eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --fix"
  },
  "dependencies": {
    "less": "^4.1.1",
    "sass": "^1.35.2",
    "vue": "^3.0.5"
  },
  "devDependencies": {
+   "@typescript-eslint/eslint-plugin": "^4.28.2",
+   "@typescript-eslint/parser": "^4.28.2",
    "@vitejs/plugin-vue": "^1.2.4",
    "@vue/compiler-sfc": "^3.0.5",
+   "@vue/eslint-config-typescript": "^7.0.0",
    "autoprefixer": "^10.2.6",
+   "eslint": "^7.30.0",
+   "eslint-plugin-vue": "^7.13.0",
    "mockjs": "^1.1.0",
    "vite": "^2.4.0",
    "vite-plugin-mock": "^2.9.1"
  }
}
```

10.Prettier

- ESLint 主要解决的是代码质量问题
- 代码质量规则 (code-quality rules)
  - no-unused-vars
  - no-extra-bind
  - no-implicit-globals
  - prefer-promise-reject-errors
- 代码风格规则 (code-formatting rules)
  - max-len
  - no-mixed-spaces-and-tabs
  - keyword-spacing
  - comma-style
- 代码风格问题需要使用 `Prettier`
- Prettier 声称自己是一个有主见的代码格式化工具 (opinionated code formatter)

```
npm install prettier eslint-plugin-prettier  @vue/eslint-config-prettier --save-dev
```

```
{
  "name": "zhufeng-vite2-prepare",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite preview",
    "lint": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --quiet",
    "lint:fix": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --fix"
  },
  "dependencies": {
    "less": "^4.1.1",
    "sass": "^1.35.2",
    "vue": "^3.0.5"
  },
  "devDependencies": {
    "@typescript-eslint/eslint-plugin": "^4.28.2",
    "@typescript-eslint/parser": "^4.28.2",
    "@vitejs/plugin-vue": "^1.2.4",
    "@vue/compiler-sfc": "^3.0.5",
+   "@vue/eslint-config-prettier": "^6.0.0",
    "@vue/eslint-config-typescript": "^7.0.0",
    "autoprefixer": "^10.2.6",
    "eslint": "^7.30.0",
+   "eslint-plugin-prettier": "^3.4.0",
    "eslint-plugin-vue": "^7.13.0",
    "mockjs": "^1.1.0",
+   "prettier": "^2.3.2",
    "vite": "^2.4.0",
    "vite-plugin-mock": "^2.9.1"
  }
}
```

.eslintrc.js

```
module.exports = {
  root: true,
  env: {
    browser: true,
    es2021: true,
    node: true,
  },
  extends: [
    "plugin:vue/vue3-recommended",
    "eslint:recommended",
    "@vue/typescript/recommended",
+   "@vue/prettier",
+   "@vue/prettier/@typescript-eslint",
  ],
  parserOptions: {
    ecmaVersion: 2021,
  },
  rules: {
    "no-unused-vars": "off",
    "@typescript-eslint/no-unused-vars": "off",
+   "prettier/prettier": ["error", { endOfLine: "auto" }],
  },
};
```

11.单元测试

```
cnpm i jest@next babel-jest@next @types/jest vue-jest@next ts-jest@next @vue/test-utils@next --save-dev
```

```
{
  "name": "zhufeng-vite2-prepare",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite preview",
    "lint": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --quiet",
    "lint:fix": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --fix"
  },
  "dependencies": {
    "less": "^4.1.1",
    "sass": "^1.35.2",
    "vue": "^3.0.5"
  },
  "devDependencies": {
    "@typescript-eslint/eslint-plugin": "^4.28.2",
    "@typescript-eslint/parser": "^4.28.2",
    "@vitejs/plugin-vue": "^1.2.4",
    "@vue/compiler-sfc": "^3.0.5",
    "@vue/eslint-config-prettier": "^6.0.0",
    "@vue/eslint-config-typescript": "^7.0.0",
    "autoprefixer": "^10.2.6",
    "eslint": "^7.30.0",
    "eslint-plugin-prettier": "^3.4.0",
    "eslint-plugin-vue": "^7.13.0",
    "mockjs": "^1.1.0",
    "prettier": "^2.3.2",
    "vite": "^2.4.0",
    "vite-plugin-mock": "^2.9.1"
  }
}
```

- vue-jest (https://github.com/vuejs/vue-jest/tree/v3)Jest Vue transformer with source map support
- babel-jest (https://www.npmjs.com/package/babel-jest)Babel jest plugin
- ts-jest (https://www.npmjs.com/package/ts-jest)A Jest transformer with source map support that lets you use Jest to test projects written in TypeScript jest.config.js

```
module.exports = {
  testEnvironment: "jsdom",
  transform: {
    "^.+\\.vue{{content}}quot;: "vue-jest",
    "^.+\\.jsx?{{content}}quot;: "babel-jest",
    "^.+\\.tsx?{{content}}quot;: "ts-jest",
  },
  moduleNameMapper: {
    "^@/(.*){{content}}quot;: "/src/$1",
  },
  testMatch: ["**/tests/**/*.spec.[jt]s"],
};
```

tests\test.ts

```
import { mount } from '@vue/test-utils'
const MessageComponent = {
  template: ' {{ msg }}',
  props: ['msg']
}
test('displays message', () => {
  const wrapper = mount(MessageComponent, {
    props: {
      msg: 'Hello world'
    }
  })
  expect(wrapper.text()).toContain('Hello world')
})
```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    "moduleResolution": "node",
    "strict": true,
    "jsx": "preserve",
    "sourceMap": true,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "lib": ["esnext", "dom"],
+   "types":["vite/client","jest"],
+   "baseUrl": "./",
+   "paths": {
+     "@": ["./src"]
+   }
  },
+ "include": ["src/**/*.ts", "src/**/*.d.ts", "src/**/*.tsx", "src/**/*.vue","tests/**/*.spec.ts", "tests/test.ts"]
}
```

package.json

```
{
   "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite preview",
    "lint": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --quiet",
    "lint:fix": "eslint --ext .ts,vue src/** --no-error-on-unmatched-pattern --fix",
+   "test": "jest  --passWithNoTests"
  }
}
```

12.git hook

- 可以在 `git commit`之前检查代码，保证所有提交到版本库中的代码都是符合规范的
- 可以在 `git push`之前执行单元测试,保证所有的提交的代码经过的单元测试
- husky ([husky](https://www.npmjs.com/package/husky))可以让我们向项目中方便添加git hooks
- lint-staged (https://www.npmjs.com/package/lint-staged)用于实现每次提交只检查本次提交所修改的文件
- lint-staged#configuration (https://github.com/okonet/lint-staged#configuration)
- 可以规范 `git commit -m ""`中的描述信息
- commitlint 推荐我们使用 config-conventional 配置去写 commit
- 提交格式 `git commit -m <type>[optional scope]: <description></description></type>`
  - type ：用于表明我们这次提交的改动类型，是新增了功能？还是修改了测试代码？又或者是更新了文档？
  - optional scope：一个可选的修改范围。用于标识此次提交主要涉及到代码中哪个模块
  - description：一句话描述此次提交的主要内容，做到言简意赅 类型 描述 build 编译相关的修改，例如发布版本、对项目构建或者依赖的改动 chore 其他修改, 比如改变构建流程、或者增加依赖库、工具等 ci 持续集成修改 docs 文档修改 feature 新特性、新功能 fix 修改bug perf 优化相关，比如提升性能、体验 refactor 代码重构 revert 回滚到上一个版本 style 代码格式修改 test 测试用例修改

```
cnpm i husky lint-staged @commitlint/cli @commitlint/config-conventional --save-dev
```

- `prepare`脚本会在 `npm install`(不带参数)之后自动执行
- 当我们执行npm install安装完项目依赖后会执行 `husky install`命令，该命令会创建 `.husky/`目录并指定该目录为 `git hooks`所在的目录

```
npm set-script prepare "husky install"
npm run prepare
```

```
npx husky add .husky/pre-commit "lint-staged"
npx husky add .husky/commit-msg "npx --no-install commitlint --edit $1"
npx husky add .husky/pre-push "npm run test"
```

commitlint.config.js

```
module.exports = {
  extends: ["@commitlint/config-conventional"],
  rules: {
    "type-enum": [
      2,
      "always",
      [
        "feature",
        "update",
        "fixbug",
        "refactor",
        "optimize",
        "style",
        "docs",
        "chore",
      ],
    ],
    "type-case": [0],
    "type-empty": [0],
    "scope-empty": [0],
    "scope-case": [0],
    "subject-full-stop": [0, "never"],
    "subject-case": [0, "never"],
    "header-max-length": [0, "always", 72],
  },
};
```

* *参考*