link: null
title: 珠峰架构师成长计划
description: tsconfig.json
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=181 sentences=428, words=2459

课程大纲

- 单元测试和E2E测试
- SSR服务器端渲染
- 静态化
- HMR 热更新

1.测试

```
mkdir 1.test
cd 1.test
npm init -y

pnpm install vue
pnpm i vite @vitejs/plugin-vue -D
pnpm install @babel/core @babel/preset-env  typescript @babel/preset-typescript  -D
pnpm install jest ts-jest ts-node @types/node @types/jest babel-jest @vue/vue3-jest -D
pnpm install  @vue/test-utils@next jest-transform-stub -D
echo done
```

```
{
  "compilerOptions": {
    "composite": true,
    "module": "esnext",
    "moduleResolution": "node",
    "skipLibCheck": true,
  },
  "include": ["vite.config.ts"]
}
```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "esnext",
    "useDefineForClassFields": true,
    "module": "esnext",
    "moduleResolution": "node",
    "strict": true,
    "jsx": "preserve",
    "sourceMap": true,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "lib": ["esnext", "dom"],
    "skipLibCheck": true,
    "strictNullChecks":true,
    "noEmit":true,
    "noEmitOnError:true,
    "baseUrl": ".",
    "paths": {
     "@/*": ["src/*"]
    }
  },
  "include": ["src/**/*.ts", "src/**/*.d.ts", "src/**/*.tsx", "src/**/*.vue"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

vite.config.ts

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { resolve } from "path"
export default defineConfig({
  plugins: [vue()],
  resolve: {
   alias: {
     "@": resolve("src")
   }
 },
})
```

package.json

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

index.html

```

<html lang="en">

<head>
  <meta charset="UTF-8" />
  <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.ts">script>
body>

html>
```

.gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

```

src\main.ts

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')

```

src\env.d.ts

```

declare module '*.vue' {
  import type { DefineComponent } from 'vue'

  const component: DefineComponent
  export default component
}
```

src\App.vue

```

import TodoApp from './components/TodoApp.vue'

```

src\components\TodoApp.vue

```

  addTodo

import logoUrl from "@/assets/logo.png"

img{
  width:50px;
}

```

package.json

```
{
    "scripts": {
    "dev": "vite",
    "build": "vite build",
+   "test": "jest"
  },
}
```

src\sum.ts

```
export default function (a:number,b:number) {
  return a + b;
}
```

src\sum.spec.ts

```
import sum from './sum';
it('test sum', () => {
  expect(sum(1, 2)).toBe(3);
});
```

babel.config.ts

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

jest.config.ts

```
import type { Config } from '@jest/types';
const config: Config.InitialOptions = {
   transform: {
    "^.+\\.js{{content}}quot;: "babel-jest",
    "^.+\\.ts{{content}}quot;: "ts-jest",
  },
  testMatch: ["/src/**/*.spec.(t|j)s"],
  testEnvironment: "jsdom",
  transformIgnorePatterns: ["/node_modules/"]
};
export default config;
```

src\components\TodoApp.spec.ts

```
import { mount } from '@vue/test-utils'
import TodoApp from './TodoApp.vue'

test('render todoApp', () => {
   const wrapper = mount(TodoApp)
  const addTodo = wrapper.get('[data-test="addTodo"]')
  expect(addTodo.text()).toBe('addTodo')
})
```

jest.config.ts

```
import type { Config } from '@jest/types';
const config: Config.InitialOptions = {
   transform: {
    "^.+\\.js{{content}}quot;: "babel-jest",
    "^.+\\.ts{{content}}quot;: "ts-jest",
+   "^.+\\.vue{{content}}quot;: "@vue/vue3-jest",
+   ".+\\.(css|scss|png|jpg|svg){{content}}quot;: "jest-transform-stub"
  },
+ moduleNameMapper: {
+   "^@/(.*){{content}}quot;: "/src/$1"
+ },
  testMatch: ["/src/**/*.spec.(t|j)s"],
  testEnvironment: "jsdom",
  transformIgnorePatterns: ["/node_modules/"]
};
export default config;
```

```
{
    "scripts": {
    "dev": "vite",
    "build": "vite build",
    "test": "jest",
+   "test:coverage": "jest --coverage"
  },
}
```

jest.config.ts

```
import type { Config } from '@jest/types';
const config: Config.InitialOptions = {
  transform: {
    "^.+\\.js{{content}}quot;: "babel-jest",
    "^.+\\.ts{{content}}quot;: "ts-jest",
    "^.+\\.vue{{content}}quot;: "@vue/vue3-jest",
    ".+\\.(css|scss|png|jpg|svg){{content}}quot;: "jest-transform-stub"
  },
  moduleNameMapper: {
    "^@/(.*){{content}}quot;: "/src/$1"
  },
  testMatch: ["/src/**/*.spec.(t|j)s"],
  testEnvironment: "jsdom",
+ transformIgnorePatterns: ["/node_modules/"],
+ coverageDirectory: "coverage",
+ coverageProvider: "v8",
+ collectCoverageFrom: ["src/**/*.{js,vue}", "!src/main.ts", "!src/App.vue"],
+ coverageThreshold: {
+   global: {
+     branches: 40,
+     functions: 80,
+     lines: 90,
+     statements: 80
+   }
+ }
};
export default config;
```

```
pnpm install cypress -D
pnpm install @cypress/vue@next @cypress/vite-dev-server -D
echo done
```

src\components\TodoApp.vue

```
+
+ addTodo
+
+   {{ todo.text }}
+

<span class="hljs-addition">+import { ref, reactive } from 'vue';</span>
<span class="hljs-addition">+interface Todo {</span>
<span class="hljs-addition">+  text: string;</span>
<span class="hljs-addition">+}</span>
<span class="hljs-addition">+const newTodo = ref('');</span>
<span class="hljs-addition">+const todos = reactive<Array<Todo>>([]);</span>
<span class="hljs-addition">+const addTodo = () => {</span>
<span class="hljs-addition">+  todos.push({ text: newTodo.value });</span>
<span class="hljs-addition">+  newTodo.value = '';</span>
<span class="hljs-addition">+}</span>
```

src\components\TodoApp.cy.ts

```
import { mount } from '@cypress/vue'
import TodoApp from './TodoApp.vue'
describe('TodoApp', () => {
  it('render TodoApp', () => {
    mount(TodoApp)
    const text = 'play';
    cy.get('[data-test="newTodo"]').type(text)
    cy.get('[data-test="addTodo"]').click();
    cy.get('.todo-list li')
      .should('have.length', 1)
      .last()
      .should('have.text', text)
  })
});
```

cypress\integration\1-getting-started\todoApp.spec.js

```

describe('TodoApp', () => {
  beforeEach(() => {
    cy.visit('http://127.0.0.1:3000')
  })
  it('can add new todo items', () => {
    const text = 'play';
    cy.get('[data-test="newTodo"]').type(text)
    cy.get('[data-test="addTodo"]').click();
    cy.get('.todo-list li')
      .should('have.length', 1)
      .last()
      .should('have.text', text)
  })
})
```

cypress\plugins\index.js

```
const { startDevServer } = require('@cypress/vite-dev-server');
module.exports = (on, config) => {
  on('dev-server:start', async (options) => startDevServer({ options }));
}
```

cypress.json

```
{
  "component": {
    "testFiles": "**/*.cy.{js,ts,jsx,tsx}",
    "componentFolder": "src"
  }
}
```

package.json

```
{
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "test": "jest",
    "test:coverage": "jest --coverage",
+   "test:e2e": "cypress open",
+   "test:ct": "cypress open-ct"
  },
}
```

node_modules\cypress\types\index.d.ts

node_modules\cypress\types\cypress-expect.d.ts

2. SSR服务器端渲染

- [project-references (https://www.typescriptlang.org/docs/handbook/project-references.html)](https://www.typescriptlang.org/docs/handbook/project-references.html)

```
pnpm install react react-dom  react-router-dom
pnpm install @types/react @types/react-dom @vitejs/plugin-react typescript vite @types/node fs-extra @types/fs-extra -D
```

tsconfig.node.json

```
{
  "compilerOptions": {
    "composite": true,
    "module": "esnext",
    "moduleResolution": "node"
  },
  "include": ["vite.config.ts"]
}
```

vite.config.ts

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

export default defineConfig({
  plugins: [react()]
})

```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "ESNext",
    "useDefineForClassFields": true,
    "lib": ["DOM", "DOM.Iterable", "ESNext"],
    "allowJs": false,
    "skipLibCheck": false,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "ESNext",
    "moduleResolution": "Node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "noEmitOnError": true,
    "jsx": "react-jsx",
    "types": ["vite/client"]
  },
  "include": ["src"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

src\vite-env.d.ts

index.html

```

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite Apptitle>
  head>
  <body>
    <div id="root">div>
    <script type="module" src="/src/main.tsx">script>
  body>
html>
```

src\main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom'

ReactDOM.render(
  <h1>helloh1>,
  document.getElementById('root')
)
```

.gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

```

package.json

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

src\main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom'
+import { BrowserRouter, Routes, Route,Link } from 'react-router-dom';
+import Home from './routes/Home';
+import User from './routes/User';
+import Profile from './routes/Profile';

ReactDOM.render(
+
+    首页
+    用户管理
+    个人中心
+
+      } />
+      } />
+      } />
+
+  ,
  document.getElementById('root')
)
```

src\routes\Home.tsx

```
export default function () {
  return (
    <div>Homediv>
  )
}
```

src\routes\User.tsx

```
export default function () {
  return (
    <div>Userdiv>
  )
}
```

src\routes\Profile.tsx

```
export default function () {
  return (
    <div>Profilediv>
  )
}
```

src\main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom'
import { BrowserRouter} from 'react-router-dom';
+import App from './App';
ReactDOM.render(

+
  ,
  document.getElementById('root')
)
```

src\App.tsx

```
import {Link, useRoutes } from 'react-router-dom';
import routesConfig from './routeConfig';

export default function () {
  return (
    <>
      <Link to="/">首页Link>
      <Link to="/user">用户管理Link>
      <Link to="/profile">个人中心Link>
      {useRoutes(routesConfig)}
    </>
  )
}
```

src\routeConfig.tsx

```
const routesModules = import.meta.globEager("./routes/*.tsx");
const routesConfig = Object.keys(routesModules).map(url => {
  const name = url.match(/\.\/routes\/(.+)\.tsx$/)![1].toLowerCase();
  const Component = routesModules[url].default;
  return {
    path: name === 'home' ? `/` : `/` + name,
    element: <Component />
  }
});
export default routesConfig;
```

```
pnpm install express  morgan
pnpm install @types/node @types/express @types/morgan cross-env -D
pnpm install ts-node-dev -g
```

tsconfig.json

```
{
  "compilerOptions": {
    "target": "ESNext",
    "useDefineForClassFields": true,
    "lib": ["DOM", "DOM.Iterable", "ESNext"],
    "allowJs": false,
    "skipLibCheck": false,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "ESNext",
    "moduleResolution": "Node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "noEmitOnError": true,
    "jsx": "react-jsx",
    "types": ["vite/client"]
  },
+  "ts-node": {
+    "compilerOptions": {
+      "module": "commonjs",
+      "esModuleInterop":true
+    }
+ },
  "include": ["src"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

tsconfig.node.json

```
{
  "compilerOptions": {
    "composite": true,
    "module": "esnext",
    "moduleResolution": "node",
    "esModuleInterop": true
  },
+  "include": ["vite.config.ts","ssr-entry.ts"]
}
```

package.json

```
{
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
+   "ssr": "ts-node-dev --respawn ssr.ts"
  }
}
```

ssr-entry.ts

```
import express, { Express } from "express";
import { createServer } from 'vite'
const app: Express = express();

; (async function () {
  const vite = await createServer({
    server: {
      middlewareMode:'html'
   }
  })
  app.use(vite.middlewares);
  app.listen(8000,()=>console.log('ssr server started on 8000'))
})();
```

ssr-entry.ts

```
import express, { Express,Request,Response } from "express";
+import logger from 'morgan';
import { createServer } from 'vite'
+import fs from 'fs-extra';
+import path from 'path';
const app: Express = express();

; (async function () {
  const vite = await createServer({
    server: {
+       middlewareMode: 'ssr'
    }
  })
  app.use(vite.middlewares);
+ app.use(logger('dev'));
+ app.get('*', async (req:Request, res:Response) => {
+   const indexPath = path.join(__dirname, 'index.html');
+   let originHtml = await fs.readFile(indexPath, 'utf8');
+   originHtml = await vite.transformIndexHtml(req.url, originHtml);
+   const { render } = await vite.ssrLoadModule('/src/server-entry.tsx');
+   const renderHtml = await render(req.url);
+   originHtml = originHtml.replace('', renderHtml);
+   res.set('Content-Type', "text/html;charset=utf-8");
+   res.send(originHtml);
+ });
  app.listen(8000, () => console.log('ssr server started on 8000'))
})();
```

src\server-entry.tsx

```
import React from 'react'
import ReactDOMServer from 'react-dom/server'
import { StaticRouter } from 'react-router-dom/server';
import App from './App';
export function render(url: string, context: string) {
  return ReactDOMServer.renderToString(
    <StaticRouter location={url}>
      <App />
    StaticRouter>
  );
}
```

src\client-entry.tsx

```
import React from 'react'
import ReactDOM from 'react-dom'
import { BrowserRouter } from 'react-router-dom';
import App from './App';
+ReactDOM.hydrate(

  ,
  document.getElementById('root')
)
```

index.html

```

  Vite App

+


+
```

package.json

```
{
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "dev:ssr": "ts-node-dev --respawn ssr-entry.ts",
+    "build:ssr": "cross-env NODE_ENV=production && npm run build:client && npm run build:server && npm run build:ssr-entry",
+   "build:client": "tsc && vite build  --outDir dist/client",
+   "build:server": "tsc && vite build --outDir dist/server --ssr src/server-entry.tsx",
+   "build:ssr-entry": "tsc --esModuleInterop true --outDir dist ssr-entry.ts",
+   "start:ssr": "pm2 start ./dist/ssr-entry.js"
  },
}
```

ssr-entry.ts

```
import express, { Express,Request,Response } from "express";
import logger from 'morgan';
import { createServer } from 'vite'
import fs from 'fs-extra';
import path from 'path';
const app: Express = express();
+const NODE_ENV = process.env.NODE_ENV;
; (async function () {
  const indexPath = path.join(__dirname, 'client/index.html');
  let originHtml = await fs.readFile(indexPath, 'utf8');
+  if (NODE_ENV === 'development') {
    const vite = await createServer({
      server: {
        middlewareMode: 'ssr'
      }
    })
    app.use(vite.middlewares);
    app.use(logger('dev'));
    app.get('*', async (req:Request, res:Response) => {
      originHtml = await vite.transformIndexHtml(req.url, originHtml);
      const { render } = await vite.ssrLoadModule('/src/server-entry.tsx');
      const renderHtml = await render(req.url);
      originHtml = originHtml.replace('', renderHtml);
      res.set('Content-Type', "text/html;charset=utf-8");
      res.send(originHtml);
    });
+ } else {
+   app.get('*.html', async (req: Request, res: Response) => {
+     serve(req, res)
+   })
+   app.use(express.static(path.resolve(__dirname, 'client')));
+   app.get('*', async (req: Request, res: Response) => {
+     serve(req, res)
+   })
+ }
+ async function serve(req: Request, res: Response) {
+     const { render } = require('./server/server.js');
+     const renderHtml = await render(req.url);
+     originHtml = originHtml.replace('', renderHtml);
+     res.set('Content-Type', "text/html;charset=utf-8");
+     res.send(originHtml);
+   }
  app.listen(8000, () => console.log('ssr server started on 8000'))
})();
```

package.json

```
{
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "dev:ssr": "ts-node-dev --respawn ssr-entry.ts",
    "build:ssr": "cross-env NODE_ENV=production && npm run build:client && npm run build:server && npm run build:ssr-entry",
    "build:client": "tsc && vite build  --outDir dist/client",
    "build:server": "tsc && vite build --outDir dist/server --ssr src/server-entry.tsx",
    "build:ssr-entry": "tsc --esModuleInterop true --outDir dist ssr-entry.ts",
    "start:ssr": "pm2 start ./dist/ssr-entry.js",
+   "static": "ts-node  static.ts"
  }
}
```

static.ts

```
import path from 'path';
import fs from 'fs-extra';
(async function () {
  const indexPath = path.join(__dirname, 'dist/client/index.html');
  let originHtml = await fs.readFile(indexPath, 'utf8');
  let routes = await fs.readdir('./src/routes');
  routes = routes.map((route:string) => {
    route = path.basename(route, '.tsx').toLowerCase();
    return route;
  });
  const { render } = require('./dist/server/server-entry.js');
  const staticDir = path.join(__dirname, 'dist/static');
  await fs.ensureDir(staticDir);
  for (let route of routes) {
    const renderHtml = await render('/'+(route==='home'?'':route))
    let routeHtml = originHtml.replace('', renderHtml);
    if (route === 'home') route = 'index';
    await fs.writeFile(path.join(__dirname, 'dist/static', `${route}.html`), routeHtml);
  }
  await fs.copy(path.join(__dirname, 'dist/client/assets'),path.join(__dirname, 'dist/static/assets'));
})();
```

tsconfig.node.json

```
{
  "compilerOptions": {
    "composite": true,
    "module": "esnext",
    "moduleResolution": "node",
    "esModuleInterop": true
  },
+  "include": ["vite.config.ts","ssr-entry.ts","static.ts"]
}
```

3.HMR

```
pnpm install vite -D
```

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

src\main.js

```
export function render() {
  app.innerHTML = 'title';
}
render();
if (import.meta.hot) {
  import.meta.hot.accept((newModule) => {
    newModule.render();
  });
}
```

index.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>hmrtitle>
head>

<body>
  <div id="app">div>
  <script type="module" src="/src/main.js">script>
body>

html>
```

src\render.js

```
export function render() {
  app.innerHTML = 'title5';
}
```

src\main.js

```
import { render } from './render';
render();
if (import.meta.hot) {
  import.meta.hot.accept(['./render'], ([renderMod]) => {
    renderMod.render();
  });
}
```

src\render.js

```
+let counter = { number: 0 };
+let timer = setInterval(() => {
+  console.log(counter.number++);
+}, 1000);

export function render() {
  app.innerHTML = 'title';
}

+if (import.meta.hot) {
+  import.meta.hot.dispose(() => {
+    console.log('dispose render.js');
+    clearInterval(timer);
+  });
+}
```

src\render.js

```
+let counter = import.meta.hot.data.counter || { number: 0 };
let timer;

export function render() {
  timer = setInterval(() => {
    app.innerHTML = counter.number++;
  }, 1000);
}

if (import.meta.hot) {
//每个模块有一个data属性,保存热更新前的状态
+ import.meta.hot.data.counter = counter;
  import.meta.hot.dispose(() => {
    console.log('dispose render.js');
    clearInterval(timer);
  });
}
```

src\render.js

```
+export let counter = import.meta.hot.data.counter || { number: 0 };
let timer;

export function render() {
  timer = setInterval(() => {
    app.innerHTML = counter.number++;
  }, 1000);
}

if (import.meta.hot) {
  import.meta.hot.data.counter = counter;
  import.meta.hot.dispose(() => {
    console.log('dispose render.js');
    clearInterval(timer);
  });
}
```

src\main.js

```
import { render } from './render';
render();
if (import.meta.hot) {
  import.meta.hot.accept(['./render'], ([renderMod]) => {
+    if (renderMod.counter.number < 10) {
+      renderMod.render();
+    } else {
+      //强制刷新
+      import.meta.hot.invalidate();
    }
  });
+  //import.meta.hot.accept();
+  import.meta.hot.decline();
}
```