link: null
title: 珠峰架构师成长计划
description: index.html
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=74 sentences=296, words=2433

1.初始化项目

```
pnpm init -y
pnpm install vite @vitejs/plugin-vue @rollup/pluginutils vue/compiler-sfc hash-sum --save-dev
```

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

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>vuetitle>
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

```

  App

export default {
  name: 'App'
}

h1 {
  color: red;
}

```

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

2.实现vue插件

vite.config.js

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

plugins\plugin-vue.js

```
import { createFilter, normalizePath } from '@rollup/pluginutils';
import { parse, compileScript, rewriteDefault, compileTemplate, compileStyleAsync } from 'vue/compiler-sfc';
import hash from 'hash-sum';
import path from 'path';
import fs from 'fs';
const root = process.cwd();
const descriptorCache = new Map();
function vue(pluginOptions) {
  const { include = /\.vue$/, exclude } = pluginOptions;
  const filter = createFilter(include, exclude);
  return {
    name: 'vue',
    async load(id) {
      //.log('id', id);//C:\aproject\zhufengwebpack202202\16.viteplugin\src\App.vue
      const { filename, query } = parseVueRequest(id);
      if (!filter(filename)) {
        return null;
      }
      if (query.has('vue')) {
        const descriptor = await getDescriptor(filename);
        if (query.get('type')
          let block = descriptor.styles[Number(query.get('index'))];
          if (block) {
            return { code: block.content };
          }
        }
      }
```

```
    },
    async transform(code, id) {
      const { filename, query } = parseVueRequest(id);
      if (!filter(filename)) {
        return null;
      }
      if (query.get('type')
        const descriptor = await getDescriptor(filename);
        let result = await transformStyle(code, descriptor, query.get('index'));
        return result;
      } else {
        let result = await transformMain(code, filename);
        return result;
      }

    }
  }
}
async function transformStyle(code, descriptor, index) {
  const block = descriptor.styles[index];
  //如果是CSS，其实翻译之后和翻译之前内容是一样的，最终返回的JS靠packages\vite\src\node\plugins\css.ts
  const result = await compileStyleAsync({
    filename: descriptor.filename,
    source: code,
    id: `data-v-${descriptor.id}`,//必须传递，不然报错
    scoped: block.scoped
  });
  let styleCode = result.code;
  return {
    code: styleCode
  };
  /*  let styleScript = `
   let style = document.createElement('style');
   style.innerText = ${JSON.stringify(styleCode)};
   document.head.appendChild(style);
   `;
   return {
     code: styleScript
   }; */
}
async function transformMain(source, filename) {
  const descriptor = await getDescriptor(filename, source);
  const scriptCode = genScriptCode(descriptor, filename);
  const templateCode = genTemplateCode(descriptor, filename);
  const stylesCode = genStyleCode(descriptor, filename);
  let resolveCode = [
    stylesCode,
    templateCode,
    scriptCode,
    `_sfc_main.render=render`,
    `export default _sfc_main`
  ].join('\n');
  return {
    code: resolveCode
  }
}
function genStyleCode(descriptor, filename) {
  let styleCode = '';
  if (descriptor.styles.length) {
    descriptor.styles.forEach((style, index) => {
      const query = `?vue&type=style&index=${index}&lang=css`;
      const styleRequest = normalizePath(filename + query);// /
      styleCode += `\nimport ${JSON.stringify(styleRequest)}`;
    });
    return styleCode;
  }
}
function genTemplateCode(descriptor, filename) {
  let result = compileTemplate({ source: descriptor.template.content, id: filename });
  return result.code;
}
/**
 * 获取此.vue文件编译 出来的js代码
 * @param {*} descriptor
 * @param {*} filename
 */
function genScriptCode(descriptor, filename) {
  let scriptCode = '';
  let script = compileScript(descriptor, { id: filename });
  scriptCode = rewriteDefault(script.content, '_sfc_main');//export default => const _sfc_main
  return scriptCode;
}

async function getDescriptor(filename, source) {
  let descriptor = descriptorCache.get(filename);
  if (descriptor) return descriptor;
  const content = await fs.promises.readFile(filename, 'utf8');
  const result = parse(content, { filename });
  descriptor = result.descriptor;
  descriptor.id = hash(path.relative(root, filename));
  descriptorCache.set(filename, descriptor);
  return descriptor;
}
function parseVueRequest(id) {
  const [filename, querystring = ''] = id.split('?');
  let query = new URLSearchParams(querystring);
  return {
    filename, query
  };
}
export default vue;
```

3.实现jsx插件

```
pnpm install @vitejs/plugin-vue-jsx --save-dev
pnpm install @vue/babel-plugin-jsx @babel/plugin-syntax-import-meta @rollup/pluginutils @babel/plugin-transform-typescript hash-sum morgan fs-extra --save-dev
```

vite.config.js

```
import { defineConfig } from "vite";
-import vue from "@vitejs/plugin-vue";
-import vue from "./plugins/plugin-vue";
+import vueJsx from "./plugins/plugin-vue-jsx.js";
export default defineConfig({
+ plugins: [vueJsx({})]
});
```

plugins\plugin-vue-jsx.js

```
import { transformSync } from '@babel/core'
import jsx from '@vue/babel-plugin-jsx'
import importMeta from '@babel/plugin-syntax-import-meta'
import { createFilter } from '@rollup/pluginutils'
import typescript from '@babel/plugin-transform-typescript';
function vueJsxPlugin(options = {}) {
  let root;
  return {
    name: 'vite:vue-jsx',
    config() {
      return {
        esbuild: {

          include: /\.ts$/
        },
        define: {
          __VUE_OPTIONS_API__: true,
          __VUE_PROD_DEVTOOLS__: false
        }
      }
    },
    configResolved(config) {
      root = config.root
    },
    transform(code, id) {
      const {
        include,
        exclude,
        babelPlugins = [],
        ...babelPluginOptions
      } = options
      const filter = createFilter(include || /\.[jt]sx$/, exclude)
      const [filepath] = id.split('?')
      if (filter(id) || filter(filepath)) {
        const plugins = [importMeta, [jsx, babelPluginOptions], ...babelPlugins]
        if (id.endsWith('.tsx') || filepath.endsWith('.tsx')) {
          plugins.push([
            typescript,
            { isTSX: true, allowExtensions: true }
          ])
        }
        const result = transformSync(code, {
          babelrc: false,
          configFile: false,
          ast: true,
          plugins
        })
        return {
          code: result.code,
          map: result.map
        }
      }
    }
  }
}
export default vueJsxPlugin;
```

src\main.js

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

src\App.jsx

```
import { defineComponent } from 'vue';
export default defineComponent({
  setup() {
    return () => (
      <h1>Apph1>
    )
  }
})
```

4.HMR

```
{
  "type":"update",
  "updates":[
    {"type":"js-update","timestamp":1647485594371,"path":"/src/App.jsx","acceptedPath":"/src/App.jsx"}
  ]}
```

```
import { transformSync } from '@babel/core'
import jsx from '@vue/babel-plugin-jsx'
import importMeta from '@babel/plugin-syntax-import-meta'
import { createFilter } from '@rollup/pluginutils'
import typescript from '@babel/plugin-transform-typescript';
+import hash from 'hash-sum'
+import path from 'path'
```

```
function vueJsxPlugin(options = {}) {
+ let needHmr = false
  return {
    name: 'vite:vue-jsx',
    config() {
      return {
        esbuild: {
          include: /\.ts$/
        },
        define: {
          __VUE_OPTIONS_API__: true,
          __VUE_PROD_DEVTOOLS__: false
        }
      }
    },
    configResolved(config) {
      root = config.root
+     needHmr = config.command === 'serve' && !config.isProduction
    },
    transform(code, id) {
      const {
        include,
        exclude,
        babelPlugins = [],
        ...babelPluginOptions
      } = options
      const filter = createFilter(include || /\.[jt]sx$/, exclude)
      const [filepath] = id.split('?')
      if (filter(id) || filter(filepath)) {
        const plugins = [importMeta, [jsx, babelPluginOptions], ...babelPlugins]
        if (id.endsWith('.tsx') || filepath.endsWith('.tsx')) {
          plugins.push([
            typescript,
            { isTSX: true, allowExtensions: true }
          ])
        }
        const result = transformSync(code, {
          babelrc: false,
          configFile: false,
          ast: true,
          plugins
        })
+       if (!needHmr) {
          return { code: result.code, map: result.map }
+       }
+       const hotComponents = []
+       let hasDefault = false
+       for (const node of result.ast.program.body) {
+         if (node.type === 'ExportDefaultDeclaration') {
+           if (isDefineComponentCall(node.declaration)) {
+             hasDefault = true
+             hotComponents.push({
+               local: '__default__',
+               exported: 'default',
+               id: hash(id + 'default')
+             })
+           }
+         }
+       }
+       if (hotComponents.length) {
+         if (hasDefault && (needHmr)) {
+           result.code =
+             result.code.replace(
+               /export default defineComponent/g,
+               `const __default__ = defineComponent`
+             ) + `\nexport default __default__`
+         }

+         if (needHmr && !/\?vue&type=script/.test(id)) {
+           let code = result.code
+           let callbackCode = ``
+           for (const { local, exported, id } of hotComponents) {
+             code +=
+               `\n${local}.__hmrId = "${id}"` +
+               `\n__VUE_HMR_RUNTIME__.createRecord("${id}", ${local})`
+             callbackCode += `\n__VUE_HMR_RUNTIME__.reload("${id}", __${exported})`
+           }
+           code += `\nimport.meta.hot.accept(({${hotComponents
+             .map((c) => `${c.exported}: __${c.exported}`)
+             .join(',')}}) => {${callbackCode}\n})`
+           result.code = code
+         }
+       }
+       return {
+         code: result.code,
+         map: result.map
+       }
      }
    }
  }
}
function isDefineComponentCall(node) {
  return (
    node &&
    node.type
    node.callee.type
    node.callee.name
  )
}
export default vueJsxPlugin;
```

5.SSR

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

src\entry-client.js

```
import { createApp } from './main'
const { app, router } = createApp()
router.isReady().then(() => {
  app.mount('#app')
})
```

src\entry-server.js

```
import { createApp } from './main'
import { renderToString } from '@vue/server-renderer'
export async function render(url, manifest = {}) {
  const { app, router } = createApp()
  router.push(url)
  await router.isReady()
  const ctx = {}
  const html = await renderToString(app, ctx)
  const preloadLinks = renderPreloadLinks(ctx.modules, manifest)
  return [html, preloadLinks]
}
function renderPreloadLinks(modules, manifest) {
  let links = ''
  const seen = new Set()
  modules.forEach((id) => {
    const files = manifest[id]
    if (files) {
      files.forEach((file) => {
        if (!seen.has(file)) {
          seen.add(file)
          links += renderPreloadLink(file)
        }
      })
    }
  })
  return links
}
function renderPreloadLink(file) {
  console.log('file', file);
  if (file.endsWith('.js') || file.endsWith('.jsx')) {
    return `${file}">`
  } else if (file.endsWith('.css')) {
    return `${file}">`
  } else {
    return ''
  }
}
```

src\main.js

```
import App from './App.jsx'
import { createSSRApp } from 'vue'
import { createRouter } from './router'
export function createApp() {
  const app = createSSRApp(App)
  const router = createRouter()
  app.use(router)
  return { app, router }
}
```

src\router.js

```
import {
  createMemoryHistory,
  createRouter as _createRouter,
  createWebHistory
} from 'vue-router'
const pages = import.meta.glob('./pages/*.jsx')
const routes = Object.keys(pages).map((path) => {
  const name = path.match(/\.\/pages(.*)\.jsx$/)[1].toLowerCase()
  return {
    path: name === '/home' ? '/' : name,
    component: pages[path]
  }
})
export function createRouter() {
  return _createRouter({
    history: import.meta.env.SSR ? createMemoryHistory() : createWebHistory(),
    routes
  })
}
```

src\App.jsx

```
import { defineComponent } from 'vue';

export default defineComponent({
  setup() {
    return () => (
      <div>
        <ul>
          <li><router-link to="/">Homerouter-link>li>
          <li><router-link to="/user">Userrouter-link>li>
        ul>
        <router-view>router-view>
      div>
    )
  }
})
```

src\pages\Home.jsx

```
import { defineComponent } from 'vue';
import { useSSRContext } from "vue"
export default defineComponent({
  setup() {
    const ssrContext = useSSRContext()
    console.log(ssrContext.modules);
    return (props, ctx) => {
      console.log('props', props);
      console.log('ctx', ctx);
      return <h1>Homeh1>;
    }
  }
})
```

src\pages\User.jsx

```
import { defineComponent } from 'vue';

export default defineComponent({
  setup() {
    return () => (
      <h1>Userh1>
    )
  }
})
```

server.js

```
import express from "express";
import logger from 'morgan';
import { createServer } from 'vite'
import fs from 'fs-extra';
import path from 'path';
const app = express();

; (async function () {
  const vite = await createServer({
    server: {
      middlewareMode: 'ssr'
    }
  })
  let manifest = JSON.parse(fs.readFileSync(path.resolve('dist/client/ssr-manifest.json'), 'utf-8'))
  app.use(vite.middlewares);
  app.use(logger('dev'));
  app.use('*', async (req, res) => {
    const url = req.originalUrl
    try {

      let template = fs.readFileSync(path.resolve('index.html'), 'utf-8')

      template = await vite.transformIndexHtml(url, template)

      const { render } = await vite.ssrLoadModule('/src/entry-server.js')

      const [appHtml, preloadLinks] = await render(url, manifest)

      const html = template
        .replace(``, preloadLinks)
        .replace(``, appHtml)

      res.status(200).set({ 'Content-Type': 'text/html' }).end(html)
    } catch (e) {

      vite.ssrFixStacktrace(e)
      console.error(e)
      res.status(500).end(e.message)
    }
  })
  app.listen(8000, () => console.log('ssr server started on 8000'))
})();
```

plugins\plugin-vue-jsx.js

```
import { transformSync } from '@babel/core'
import jsx from '@vue/babel-plugin-jsx'
import importMeta from '@babel/plugin-syntax-import-meta'
import { createFilter, normalizePath } from '@rollup/pluginutils'
import typescript from '@babel/plugin-transform-typescript';
import hash from 'hash-sum'
const path = require('path')
const ssrRegisterHelperId = '/__vue-jsx-ssr-register-helper'
const ssrRegisterHelperCode =
  `import { useSSRContext } from "vue"\n` +
  `export ${ssrRegisterHelper.toString()}`

function ssrRegisterHelper(comp, filename) {
  const setup = comp.setup
```

```
  comp.setup = (props, ctx) => {

    const ssrContext = useSSRContext()
      ; (ssrContext.modules || (ssrContext.modules = new Set())).add(filename)
    if (setup) {
      return setup(props, ctx)
    }
  }
}
function vueJsxPlugin(options = {}) {
  let root;
  let needHmr = false
  return {
    name: 'vite:vue-jsx',
    config() {
      return {
        esbuild: {
          include: /\.ts$/
        },
        define: {
          __VUE_OPTIONS_API__: true,
          __VUE_PROD_DEVTOOLS__: false
        }
      }
    },
    configResolved(config) {
      root = config.root
      needHmr = config.command === 'serve' && !config.isProduction
    },
    transform(code, id, { ssr }) {
      console.log('ssr', ssr);
      const {
        include,
        exclude,
        babelPlugins = [],
        ...babelPluginOptions
      } = options
      const filter = createFilter(include || /\.[jt]sx$/, exclude)
      const [filepath] = id.split('?')
      if (filter(id) || filter(filepath)) {
        const plugins = [importMeta, [jsx, babelPluginOptions], ...babelPlugins]
        if (id.endsWith('.tsx') || filepath.endsWith('.tsx')) {
          plugins.push([
            typescript,
            { isTSX: true, allowExtensions: true }
          ])
        }
        const result = transformSync(code, {
          babelrc: false,
          configFile: false,
          ast: true,
          plugins
        })
        if (!needHmr) {
          return { code: result.code, map: result.map }
        }
        const hotComponents = []
        let hasDefault = false
        for (const node of result.ast.program.body) {
          if (node.type === 'ExportDefaultDeclaration') {
            if (isDefineComponentCall(node.declaration)) {
              hasDefault = true
              hotComponents.push({
                local: '__default__',
                exported: 'default',
                id: hash(id + 'default')
              })
            }
          }
        }
        if (hotComponents.length) {
          if (hasDefault && (needHmr)) {
            result.code =
              result.code.replace(
                /export default defineComponent/g,
                `const __default__ = defineComponent`
              ) + `\nexport default __default__`
          }

          if (needHmr && !/\?vue&type=script/.test(id)) {
            let code = result.code
            let callbackCode = ``
            for (const { local, exported, id } of hotComponents) {
              code +=
                `\n${local}.__hmrId = "${id}"` +
                `\n__VUE_HMR_RUNTIME__.createRecord("${id}", ${local})`
              callbackCode += `\n__VUE_HMR_RUNTIME__.reload("${id}", __${exported})`
            }
            code += `\nimport.meta.hot.accept(({${hotComponents
              .map((c) => `${c.exported}: __${c.exported}`)
              .join(',')}}) => {${callbackCode}\n})`
            result.code = code
          }
        }
        if (ssr) {
          const normalizedId = normalizePath(path.relative(root, id))
          let ssrInjectCode =
            `\nimport { ssrRegisterHelper } from "${ssrRegisterHelperId}"` +
            `\nconst __moduleId = ${JSON.stringify(normalizedId)}`
         git  for (const { local } of hotComponents) {
            ssrInjectCode += `\nssrRegisterHelper(${local}, __moduleId)`
          }
          result.code += ssrInjectCode
        }
```

```
        return {
          code: result.code,
          map: result.map
        }
      }
    }
  }
}
function isDefineComponentCall(node) {
  return (
    node &&
    node.type === 'CallExpression' &&
    node.callee.type === 'Identifier' &&
    node.callee.name === 'defineComponent'
  )
}
export default vueJsxPlugin;
```