从零搭建一个 Webpack5 + Vue3 项目

掘金 webpack5 搭建 Vue3 项目——开发环境

https://juejin.cn/post/7129038787589439495

Webpack5 搭建 Vue3 + TS 项目

https://juejin.cn/post/6955430382485553166

参考项目

https://github.com/Jamie-Yang/vue3-boilerplate

基础配置

创建项目

```bash
mkdir admin-webpack
cd admin-webpack
pnpm init
```

配置 webpack

```bash
pnpm i webpack webpack-cli -D
```

main.js

```JavaScript
console.log("Hello Webpack");
```

支持页面入口

安装插件 clean-webpack-plugin

安装插件 html-webpack-plugin

```Bash
pnpm i clean-webpack-plugin html-webpack-plugin -D
```

webpack.config.js

```JavaScript
const path = require("path");
const { VueLoaderPlugin } = require("vue-loader");
const HtmlWebpackPlugin = require("html-webpack-plugin");
const { CleanWebpackPlugin } = require("clean-webpack-plugin");
module.exports = {
  mode: "development", // 环境模式
  entry: path.resolve(__dirname, "./src/main.js"), // 打包入口
  output: {
    path: path.resolve(__dirname, "dist"), // 打包出口
    filename: "js/[name].js", // 打包完的静态资源文件名
  },
  module: {
    rules: [
      {
        test: /\.vue$/,
        use: ["vue-loader"],
      },
    ],
  },
  plugins: [
    new VueLoaderPlugin(),
```

```
    new CleanWebpackPlugin(),
    new HtmlWebpackPlugin({
      templateContent: `
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Webpack App</title>
  </head>
  <body>
    <div id="app" />
  </body>
</html>
    `,
    }),
  ],
};
```

添加调试服务器

Bash

```
pnpm i webpack-dev-server -D
```

webpack.config.js

JavaScript

```
devServer: {
  contentBase: path.resolve(__dirname, './dist'),
  port: 8080,
  static: {
    directory: path.join(__dirname, 'public'),
  },
}
```

package.json

Bash

```
"scripts": {
```

```
  "dev": "webpack serve --progress --config ./webpack.config.js"
}
```

resolve

webpack.config.js

JavaScript

```
  resolve: {
    alias: {
      '@': path.resolve('src'),
    },


    extensions: ['.ts', '.tsx', '.js', '.jsx', '.vue', '.json'],
```

Vue3 基础环境

Vue SFC 组件

Bash

```
pnpm i vue-loader -D
pnpm i vue
```

App.vue

HTML

```
<template lang="">
    <div>
        <h1>{{ msg }}</h1>
    </div>
</template>
<script>
import { ref } from "vue"
export default {
    setup() {
        const msg = ref(1)
```

```
        setInterval(() => {
            msg.value++
        }, 1000)
        return { msg }
    }
}
</script>
```

main.js

```JavaScript
import { createApp } from "vue";
import App from "./App.vue";

createApp(App).mount("#app");
```

vue-loader：它是基于 `webpack` 的一个的 `loader` 插件，解析和转换 `.vue` 文件，提取出其中的逻辑代码 `script`、样式代码 `style`、以及 `HTML` 模版 `template`，再分别把它们交给对应的 `loader` 去处理如 `style-loader`、`less-loader` 等等，核心的作用，就是 `提取` 。

webapck.config.js

```JavaScript
const { VueLoaderPlugin } = require("vue-loader");

module.exports = {
  module: {
    rules: [
      {
        test: /\.vue$/,
        use: ["vue-loader"],
      },
    ],
  },
  plugins: [new VueLoaderPlugin()],
};
```

CSS Loader

Bash

```
pnpm i style-loader css-loader
```

JavaScript

```
module.exports = {
  module: {
    rules: [
      { test: /\.vue$/, use: ["vue-loader"] },
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"],
      },
    ],
  },
  plugins: [new VueLoaderPlugin()],
};
```

HTML

```
<style scoped>
h1 {
    color: blue;
}
</style>
```

Typescript

- 使用 `babel-loader`、`ts-loader` 等处理 SFC 的 `<script>` 模块；

Bash

```
pnpm i -D typescript ts-loader
```

JavaScript

```
const { VueLoaderPlugin } = require("vue-loader");

module.exports = {
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        exclude: /node_modules/,
        use: [
          {
            loader: "ts-loader",
            options: {
              // transpileOnly: true, // 关闭项目运行时的类型检查
              appendTsSuffixTo: ["\\.vue$"], // 给 .vue文件添加个 .ts后缀用于编
              // happyPackMode: true,
            },
          },
        ],
      },
    ],
  },
};
```

tsconfig.json

JSON

```
{
    "compilerOptions": {
        "target": "es5", // 指定编译后的ECMAScript目标版本： 'ES3' (default),
        "module": "esnext", // 用来指定要使用的模块标准: 'none', 'commonjs', 'a
        "strict": true, // 启用所有严格类型检查选项。
        "jsx": "preserve", // 指定jsx代码用于的开发环境: 'preserve', 'react-na
        "importHelpers": true, // 从 tslib 导入辅助工具函数（比如 __extends， _
        "moduleResolution": "node", // 用于选择模块解析策略，有'node'和'classic
        "experimentalDecorators": true, // 模块名到基于 baseUrl的路径映射的列表
        "skipLibCheck": true, // 忽略所有的声明文件（ *.d.ts）的类型检查。
```

```
        "esModuleInterop": true, // 支持在 CommonJs 模块下使用 import d from
        "allowSyntheticDefaultImports": true, // 允许从没有设置默认导出的模块中
        "sourceMap": true, // 生成相应的 .map文件。
        "baseUrl": ".", // 解析非相对模块名的基准目录，相对模块不会受baseUrl的影响
        "paths": {
            // 用于设置模块名称到基于baseUrl的路径映射
            "@/*": [
                "src/*"
            ]
        },
        "lib": [
            "esnext",
            "dom",
            "dom.iterable",
            "scripthost"
        ] // lib用于指定要包含在编译中的库文件
    },
    "include": [
        "src/**/*.ts",
        "src/**/*.tsx",
        "src/**/*.vue",
        "tests/**/*.ts",
        "tests/**/*.tsx",
        "types/**/*.d.ts",
        "types/*.d.ts"

    ], // 指定要编译的路径列表，但是和files的区别在于，这里的路径可以是文件夹，也可以是
    "exclude": [
        "node_modules"
    ] // exclude表示要排除的、不编译的文件，它也可以指定一个列表，规则和include一样，
}
```

src/shims-vue.d.ts

TypeScript

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

App.vue

```JavaScript
<script lang="typescript">
export default {
  setup() {
      const msg: Ref<number> = ref(1)
  }
};
</script>
```

Setup 语法糖

无需配置

```TypeScript
<script lang="ts" setup>
import { ref, Ref } from "vue"
// export default {
//     setup() {
const msg: Ref<number> = ref(1)
setInterval(() => {
    msg.value++
}, 1000)
// return { msg }
    // }
// }
</script>
```

支持 Less

- 使用 `less-loader`、`sass-loader` 等处理 `<style>` 模块；

支持 Pug 模版

- 使用 `pug-plain-loader` 等处理 `<template>` 模块。

vue-router

```Bash
pnpm i vue-router
```

main.ts

```TypeScript
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
    history: createWebHistory(),
    routes: [
        { path: '/', component: () => import('~/pages/index.vue') }
    ]
})
app.use(router)
```

App.vue

```HTML
<template>
  ...
  <router-view></router-view>
  ...
</template>
```

/pages/index.vue

```HTML
<template>
    index page ....
</template>
```

webpack.config.js

history 模式支持

```JavaScript
  devServer: {
    historyApiFallback: true, // 支持history 模式
  },
```

Pinia 全局状态管理

```Bash
pnpm i pinia
```

stores/user.ts

```TypeScript
import { acceptHMRUpdate, defineStore } from "pinia";
import {ref} from 'vue'
export const useUserStore = defineStore("user", () => {
  const count = ref(0);

  function add(num: number = 1) {
    count.value = count.value + num;
  }

  return {
    add,
    count,
  };
});
```

main.ts

```TypeScript
import {createPinia} from 'pinia'
app.use(createPinia())
app.mount("#app");
```

App.vue

```HTML
<button @click="user.add()">count: {{ user.count }}</button>


import { useUserStore } from "./stores/user";
const user = useUserStore()
```

Tailwind 支持

React tailwindcss 环境搭建

https://juejin.cn/post/7035803312188293133

```Bash
pnpm i style-loader css-loader
pnpm i postcss postcss-loader postcss-preset-env
pnpm i tailwindcss
```

webpack.config.js

```JavaScript
    module: {
      rules: [
        {

          test: /\.css$/i,
          include: path.resolve(__dirname, 'src'),
          use: ['style-loader', 'css-loader', 'postcss-loader'],
        },
      ]
    }
```

src/style.css

src/style.css

```CSS
@tailwind base;
@tailwind components;
@tailwind utilities;
```

main.ts

```TypeScript
import './style.css';
```

postcss.config.js

```JavaScript
const tailwindcss = require('tailwindcss');
module.exports = {
  plugins: [
    'postcss-preset-env',
    tailwindcss
  ],
};
```

tailwind.config.js

```JavaScript
module.exports = {
  content: [
    './index.html',
    './src/**/*.{vue,js,ts,jsx,tsx}'
  ],
  darkMode: 'class', // or 'media' or 'class'
  theme: {
    extend: {
      zIndex: {
        '-1': '-1'
      },
      flexGrow: {
        5: '5'
```

```
      },
      maxHeight: {
        'screen-menu': 'calc(100vh - 3.5rem)',
        modal: 'calc(100vh - 160px)'
      },
      transitionProperty: {
        position: 'right, left, top, bottom, margin, padding',
        textColor: 'color'
      },
      keyframes: {
        fadeOut: {
          from: { opacity: 1 },
          to: { opacity: 0 }
        },
        fadeIn: {
          from: { opacity: 0 },
          to: { opacity: 1 }
        }
      },
      animation: {
        fadeOut: 'fadeOut 250ms ease-in-out',
        fadeIn: 'fadeIn 250ms ease-in-out'
      }
    }
  },
  plugins: [
    require('@tailwindcss/forms')
  ]
}
```

postcss.config.js

JavaScript

```
const tailwindcss = require("tailwindcss");
module.exports = {
  plugins: ["postcss-preset-env", tailwindcss],
};
```

index.vue

```HTML
  <button
  class="
    py-2
    px-4
    font-semibold
    rounded-lg
    shadow-md
    text-white
    bg-green-500
    hover:bg-green-700
    border-none
    cursor-pointer
  "
>
  TailWindCSS Ready!!
</button>
```

静态资源文件

type 配置

- asset/resource 发送一个单独的文件并导出 URL
- asset/inline 导出一个资源的 data URI
- asset/source 导出资源的源代码
- asset 在导出一个 data URI 和发送一个单独的文件之间自动选择。之前通过使用 url-loader，并且配置资源体积限制实现

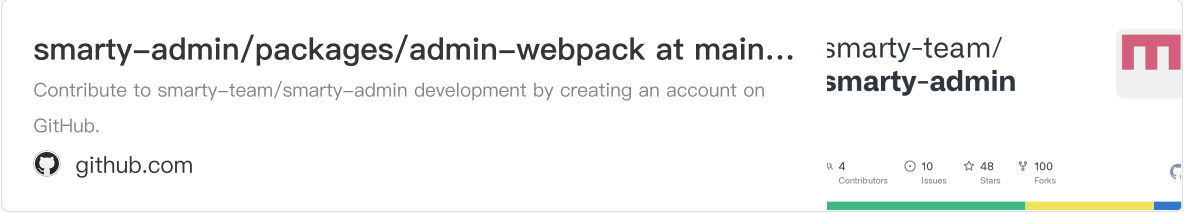
```JavaScript
    {
      test: /\.(png|jpe?g|gif|webp)(\?.*)?$/,
      type: 'asset',
      generator: { filename: 'img/[contenthash:8][ext][query]' },
    },
```

Admin 项目

目录说明

```Bash
src
+ assets # 静态资源
+ components # 组件
+ composables  # 自定义CompositionAPI
+ layouts  # 布局管理器
+ pages  # 视图
+ router  # 路由
+ stores  # 状态管理库
```

参考代码位置



安装类库

```Bash
pnpm i vue-i18n  # 国际化
pnpm i @vueuse/core @vueuse/head  # hooks
```

main.ts

```Bash
import { createI18n } from "vue-i18n";
const i18n = createI18n({
    legacy: false,
    locale: "en",
  });
```

- assets

- composables
- components
- layouts
- pages
- router

src/router/index.ts

```Bash
// import Index from "@/pages/index.vue"

import Index from "@/pages/dashboard/index.vue"
import { createRouter, createWebHistory } from 'vue-router'
import Dashboard from '@/layouts/default.vue'
const router = createRouter({
    history: createWebHistory(

    ),
    routes: [
        // { path: '/', component: Index },
        // { path: '/a', component: () => import('@/pages/index.vue') },
        { path: '/login', component: () => import('@/pages/login.vue') },
        { path: '/dashboard', component: Dashboard,
            children: [
                {
                    path: '/',
                    component: () => import('@/pages/dashboard/index.vue')
                },
                {
                    path: '/vue',
                    component: () => import('@/pages/vue.vue')
                },
                {
                    path: '/react',
                    component: () => import('@/pages/react.vue')
                }


            ] },
    ]
})
```

```
})
export default router
```