Vue3 环境搭建

Vue3 中使用pnpm workspace来实现monorepo (pnpm是快速、节省磁盘空间的包管理器。主要采用符号链接的方式管理模块)

1.全局安装 pnpm

```
npm install pnpm -g # 全局安装pnpm
```

```
pnpm init # 初始化配置文件
```

2.创建.npmrc 文件

```
shamefully-hoist = true
```

> *这里您可以尝试一下安装Vue3, pnpm install vue@next此时默认情况下vue3中依赖的模块不会被提升到node_modules下。 添加**羞耻的提升**可以将 Vue3，所依赖的模块提升到node_modules中*

3.配置 workspace

新建 **pnpm-workspace.yaml**

```
packages:
  - "packages/*"
```

> *将 packages 下所有的目录都作为包进行管理。这样我们的 Monorepo 就搭建好了。确实比lerna + yarn workspace更快捷*

4.安装依赖

> *Vue3 开发环境采用 esbuild。打包 Vue3 项目采用 rollup 进行打包代码，这里我们直接搭建开发环境*

依赖

依赖	
typescript	在项目中支持 Typescript
esbuild	构建工具，默认支持 TS
minimist	命令行参数解析

```
pnpm install typescript minimist  esbuild -D -w
```

5.初始化 TS

```
pnpm tsc --init
```

> *先添加些常用的ts-config配置，后续需要其他的在继续增加*

```
{
  "compilerOptions": {
    "outDir": "dist", // 输出的目录
    "sourceMap": true, // 采用sourcemap
    "target": "es2016", // 目标语法
    "module": "esnext", // 模块格式
    "moduleResolution": "node", // 模块解析方式
    "strict": false, // 严格模式
    "resolveJsonModule": true, // 解析json模块
    "esModuleInterop": true, // 允许通过es6语法引入commonjs模块
    "jsx": "preserve", // jsx 不转义
    "lib": ["esnext", "dom"] // 支持的类库 esnext及dom
  }
}
```

6.创建模块

> *我们现在packages目录下新建两个 package，用于下一章手写响应式原理做准备*

- reactivity 响应式模块
- shared 共享模块

所有包的入口均为`src/index.ts` 这样可以实现统一打包，并在包信息中增添打包格式`formats`属性，用于最终打包后的格式。

- reactivity/package.json

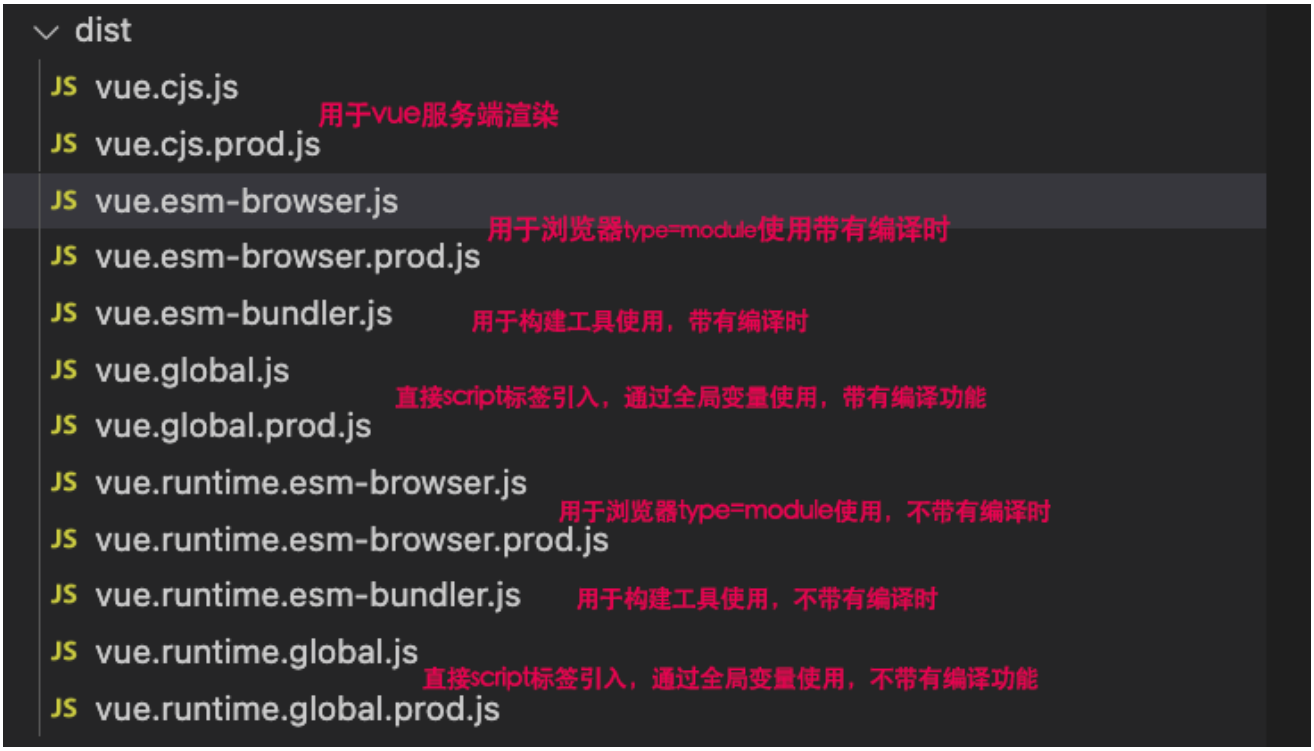
```
{
  "name": "@vue/reactivity",
  "version": "1.0.0",
  "main": "index.js",
  "module": "dist/reactivity.esm-bundler.js",
  "unpkg": "dist/reactivity.global.js",
  "buildOptions": {
    "name": "VueReactivity",
    "formats": ["esm-browser", "esm-bundler", "cjs", "global"]
  }
}
```

- shared/package.json

```
{
  "name": "@vue/shared",
  "version": "1.0.0",
  "main": "index.js",
  "module": "dist/shared.esm-bundler.js",
  "buildOptions": {
    "formats": ["esm-bundler", "cjs"]
  }
}
```

> ***formats**为自定义的打包格式*

- `global`立即执行函数的格式，会暴露全局对象
- `esm-browser`在浏览器中使用的格式，内联所有的依赖项。
- `esm-bundler`在构建工具中使用的格式，不提供.prod 格式，在构建应用程序时会被构建工具一起进行打包压缩。
- `cjs`在 node 中使用的格式，服务端渲染。

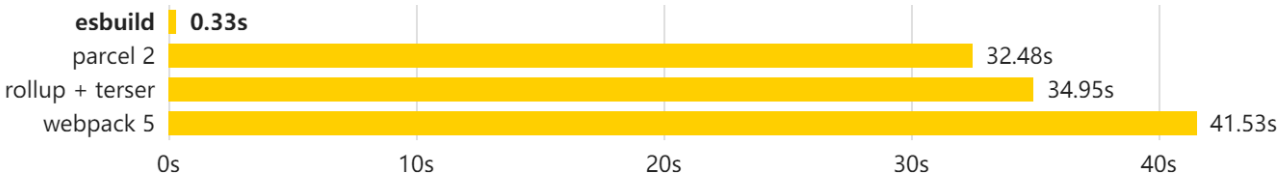



```
pnpm install @vue/shared@workspace --filter @vue/reactivity
```

> *配置ts引用关系*

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

7.开发环境搭建



创建开发时执行脚本， 参数为要打包的模块

解析用户参数

```
"scripts": {
    "dev": "node scripts/dev.js reactivity -f esm"
}
```

```
const { build } = require("esbuild");
const { resolve } = require("path");
const args = require("minimist")(process.argv.slice(2));
```

```
const target = args._[0] || "reactivity";
const format = args.f || "global";
const pkg = require(resolve(__dirname,
`../packages/${target}/package.json`));

const outputFormat = format.startsWith("global") // 输出的格式
  ? "iife"
  : format === "cjs"
  ? "cjs"
  : "esm";

// 这里应该根据 格式 如果是cjs 或者 esm-bundler 应该配置external

const outfile = resolve(
  // 输出的文件
  __dirname,
  `../packages/${target}/dist/${target}.${format}.js`
);

build({
  entryPoints: [resolve(__dirname,
`../packages/${target}/src/index.ts`)],
  outfile,
  bundle: true,
  sourcemap: true,
  format: outputFormat,
  globalName: pkg.buildOptions?.name,
  platform: format === "cjs" ? "node" : "browser",
  watch: {
    // 监控文件变化
    onRebuild(error) {
      if (!error) console.log(`rebuilt~~~~`);
    },
  },
}).then(() => {
  console.log("watching~~~");
});
```

> *这里我们先不关心生产环境和发布相关内容（生产环境和发布，主要是构建工具的配置及 node 操作文件调用命令和Vue3 本身关系不大），我们主要把精力全部放到Vue3 中。直接开始进入下一章节~~。*