搭建Monorepo环境

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)

全局安装pnpm

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

```
pnpm init -y # 初始化配置文件
```

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

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

环境搭建

> 打包项目Vue3采用rollup进行打包代码，安装打包所需要的依赖

依赖	
typescript	在项目中支持Typescript
rollup	打包工具
rollup-plugin-typescript2	rollup 和 ts的 桥梁
@rollup/plugin-json	支持引入json
@rollup/plugin-node-resolve	解析node第三方模块
@rollup/plugin-commonjs	将CommonJS转化为ES6Module
minimist	命令行参数解析
execa@4	开启子进程

```
pnpm install typescript rollup rollup-plugin-typescript2
@rollup/plugin-json @rollup/plugin-node-resolve @rollup/plugin-
commonjs minimist execa@4   -D -w
```

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": true, // 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
  }
}
```

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

- 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-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"
        ]
    }
}
```

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

> 配置`ts`引用关系

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

开发环境

创建开发时执行脚本， 参数为要打包的模块和打包的格式并且开启`sourcemap`

解析用户参数

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

```
const execa = require('execa');
const args = require('minimist')(process.argv.slice(2));
const target = args._.length ? args._[0] : 'reactivity';
const sourcemap = args.s;
const formats = args.f;
execa('rollup', [
    '-wc',
    '--environment',
    [
        `TARGET:${target}`,
        `FORMATS:${formats || 'global'}`,
        sourcemap ? `SOURCE_MAP:true` : ``
    ].filter(Boolean).join(',')
], {
    stdio: 'inherit'
})
```

rollup配置

```
import path from 'path';
// 获取packages目录
const packagesDir = path.resolve(__dirname, 'packages');
// 获取对应的模块
const packageDir = path.resolve(packagesDir, process.env.TARGET);
// 全部以打包目录来解析文件
const resolve = p => path.resolve(packageDir, p);
const pkg = require(resolve('package.json'));
const name = path.basename(packageDir); // 获取包的名字

// 配置打包信息
const outputConfigs = {
    'esm-bundler': {
        file: resolve(`dist/${name}.esm-bundler.js`),
        format: 'es'
    },
    cjs: {
        file: resolve(`dist/${name}.cjs.js`),
        format: 'cjs'
    },
    global: {
        file: resolve(`dist/${name}.global.js`),
        format: 'iife'
```

```
    }
}
// 获取formats
const packageFormats = process.env.FORMATS &&
 process.env.FORMATS.split(',');
const packageConfigs =  packageFormats || pkg.buildOptions.formats;

import json from '@rollup/plugin-json'
import commonjs from '@rollup/plugin-commonjs';
import {nodeResolve} from '@rollup/plugin-node-resolve'
import tsPlugin from 'rollup-plugin-typescript2'

function createConfig(format,output){
    output.sourcemap = process.env.SOURCE_MAP;
    output.exports = 'named';
    let external = []
    if(format === 'global'){
        output.name = pkg.buildOptions.name
    }else{ // cjs/esm 不需要打包依赖文件
        external = [...Object.keys(pkg.dependencies || {})]
    }
    return {
        input:resolve('src/index.ts'),
        output,
        external,
        plugins:[
            json(),
            tsPlugin(),
            commonjs(),
            nodeResolve()
        ]
    }
}
// 开始打包把
export default packageConfigs.map(format=>
createConfig(format,outputConfigs[format]));
```

生产环境

```
const fs = require('fs');
const execa = require('execa')
const targets = fs.readdirSync('packages').filter(f => {
    if (!fs.statSync(`packages/${f}`).isDirectory()) {
        return false;
    }
    return true;
});
async function runParallel(source, iteratorFn) {
    const ret = [];
```

```
    for (const item of source) {
        const p = Promise.resolve().then(() => iteratorFn(item))
        ret.push(p);
    }
    return Promise.all(ret)
}
async function build(target) {
    await execa(
        'rollup',
        [
            '-c',
            '--environment',
            `TARGET:${target}`
        ],
        { stdio: 'inherit' }
    )
}
runParallel(targets, build)
```