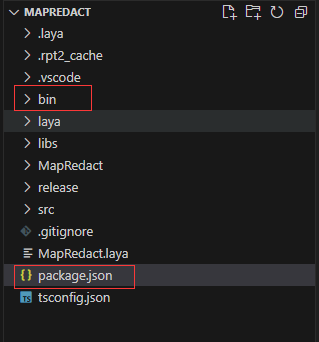
Laya 自动编译

# 前置：

## --node 版本 18.13.0

## --package.json

将package.json文件添加到与bin同级的目录下



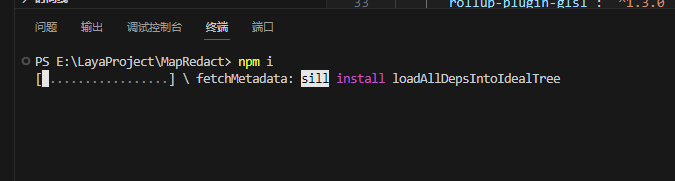
### package.json文件说明

#### devDependencies

(1)内容：是一个对象，配置模块依赖的模块列表，key是模块名称，value是版本范围

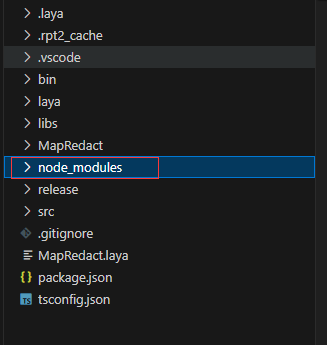
(2)作用：该模块中所列举的插件属于开发环境的依赖

(3)使用：在终端执行 npm i将下载devDependencies中所有的模块到package.json同级目录下



注：可能需要挂梯子

模块下载完成



#### Scripts说明

(1)内容：是一个对象，配置后续编译命令

(2)作用：后续 npm run中介绍

#### Scripts build说明

参数说明

node --max\_old\_space\_size=8192 .\\node\_modules\\gulp\\bin\\gulp.js --gulpfile=.laya\\compile.js compile

",

使用node和gulp执行.laya\\compile.js中的compile任务

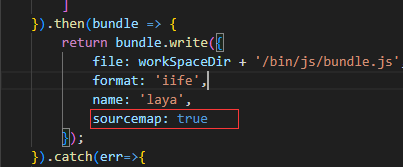
至此，使用node编译typeScript就实现了

问题1：如何在vscode中断点

问题2：如何在typeScripts改变时自动编译

# 如何在vscode中断点

1. 修改tsconfig.json 中 sourceMap为true
2. 修改compile.js



1. 修改launch.json 中 sourceMap为true
2. 使用f5运行项目，即可在vscode中添加断点

# 如何在typeScripts文件改变时自动编译

在compile.js文件中新增watch任务，代码如下：

var mainBuildConfig = {

    output: {

        sourcemap: true,

        file: '../bin/js/bundle.js',

        format: 'iife',

        name: 'laya',

    },

    cache: false,

    input: workSpaceDir + '/src/Main.ts',

    onwarn: (waring, warn) => {

        if (waring.code == "CIRCULAR\_DEPENDENCY") {

            console.log("warnning Circular dependency:");

            console.log(waring);

        }

    },

    treeshake: false, //建议忽略

    plugins: [

        typescript({

            clean: true,

            tsconfig: workSpaceDir + "/tsconfig.json",

            check: false, //Set to false to avoid doing any diagnostic checks on the code

            tsconfigOverride: { compilerOptions: { removeComments: false } },

            include: /.\*.ts/,

        }),

        glsl({

            include: /.\*(.glsl|.vs|.fs)$/,

            sourceMap: true,

            compress: false

        })

    ]

};

gulp.task("watch", prevTasks, function () {

    // 发布时调用编译功能，判断是否点击了编译选项

    if (global.publish && !global.config.compile) {

        return;

    } else if (global.publish && global.config.compile) {

        // 发布时调用编译，workSpaceDir使用publish.js里的变量

        workSpaceDir = global.workSpaceDir;

    }

    console.log('Rollup Version:', rollup.VERSION);

    const rollupConfig = [mainBuildConfig]

     console.log( rollupConfig);

    // return;

    const watcher = rollup.watch(rollupConfig);

    var t0 = Date.now();

    watcher.on('event', event => {

        // console.log(event);

        switch (event.code) {

            case 'START':

                t0 = Date.now()

                console.info('\n\nRebuilding ' + new Date().toString());

                break;

            case 'BUNDLE\_START':

                console.info('Bundling:' + (event && event.input && path.resolve(event.input)));

                break;

            case 'BUNDLE\_END':

                console.info('Bundled!' + (event && event.input && path.resolve(event.input)));

                break;

            case 'END':

                console.info('Done!Cost:' + (Date.now() / 1000 - t0 / 1000).toFixed(1) + 's');

                t0 = Date.now();

                break;

            case 'ERROR':

            case 'FATAL':

                console.error("Rollup error: ", event);

        }

    });

    console.log('Rollup is watching for changes...');

    return watcher;

});

在终端中执行npm run watch 即可在ctrl+s时自动编译