launch文件解读

必选属性：

1. type：启动配置的调试器的类型，以下为vs code提供的内置类型，还有一些类似于`pwa-node`，`pwa-chrome`是依靠插件提供的

- node
- php
- go
- chrome

2. request：启动配置的请求类型，目前支持的是`launch`和`attach`

- launch：启动 -- 多用于对没启动的程序的处理
- attach：附加 -- 多用于对已经启动的程序的处理

3. name: 自定义名称，用于区分不同的调试配置（显示在“调试启动配置”下拉列表中）

常见可选属性

1. presentation： 呈现方式（在调试启动配置下的呈现方式）

- order：排序位置
- hidden：是否隐藏
- group：分组

2. preLaunchTask： 在调试会话开始之前启动一个任务
3. postDebugTask： 在调试会话的最后启动的一个任务
4. internalConsoleOptions： 此属性控制调试会话期间调试控制台面板的可见性

- neverOpen：从不打开
- openOnSessionStart：启动会话时打开
- openOnFirstSessionStart：启动第一个会话时打开

部分可选属性

1. program： 启动调试器时要运行的文件
2. args： 传递给程序的参数
3. env： 环境变量
4. envFile： 环境变量文件的路径
5. cwd：用于查找依赖项和其他文件的当前工作目录
6. port： 连接到正在运行的进程时的端口
7. stopOnEntry： 当程序启动时立即中断
8. console： 使用`console`的方式

- internalConsole
- integratedTerminal
- externalTerminal

一些变量 文档

1. `${workspaceFolder}`：工作空间文件夹的根路径
2. `${file}`：编辑器**当前窗口打开的文件**的路径

调试nodejs

注：调试nodejs时不能使用`ES Module`（`import`，`export`）来导入导出模块，需使用`CommonJS`（`require`，`module.exports`）

launch.json文件配置：

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "debug nodejs",
      "cwd": "${workspaceFolder}", // 用于设置当前调试文件的根路径 默认为
${workspaceFolder}
      "skipFiles": [
        "${workspaceFolder}/node_modules/**/*.js", // 调试时不进入node_modules中的程
序
        "<node_internals>/**", // 跳过内部node模块程序
      ],
      "program": "debug.js", // 启动程序入口文件 必须使用绝对路径
      // "runtimeExecutable": "npm", // 执行package.json中定义的script
      // "runtimeArgs": ["run-script", "debug"], // 执行package.json中定义的script
    }
  ]
}
```

完整代码：点击访问

调试ts程序

背景：项目中在`tsconfig.json`文件中配置的路径别名，如果使用常规的`tsc`去编译程序是不能解析出路径别名的，因此引入了`ts-node` + `tsconfig-paths/register`来编译并执行程序

注：项目依赖中需要安装`ts-node`、`typescript`、`tsconfig-paths`

launch.json文件配置：

```
{
    "version": "0.2.0",
    "configurations": [
      {
        "name": "ts-node", // 自定义名称
        "type": "node", // 内置特定执行器
        "request": "launch",
        "env": {
          "NODE_ENV": "test", // 设置node环境变量 process.env.NODE_ENV 可以获取到这个
值
        },
        "runtimeArgs": [
```

```
            "-r",
            "ts-node/register", // 加载模块 ts-node/register
            "-r",
            "tsconfig-paths/register" // 加载模块 tsconfig-paths/register
          ],
          "skipFiles": [
            "${workspaceFolder}/modules/assistant/node_modules/**/*.js", // 调试时跳过node_modules中的程序 必须使用绝对路径
            "<node_internals>/**", // 跳过内部node模块程序
          ],
          "cwd": "${workspaceFolder}/modules/assistant", // 对应runtimeArgs中找的模块的路径
          "protocol": "inspector",
          "program": "./test/photography.spec.ts", // 拼接在cwd的路径后面或者使用绝对路径
          "internalConsoleOptions": "openOnSessionStart"
        }
      ]
}
```

完整代码：点击访问

调试ng&react

1、安装 Debugger for Chrome 插件

launch.json文件配置：

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "debug ng", // 自定义名称
      "url": "http://localhost:4200", // 项目启动地址
      "skipFiles": [
        "${workspaceFolder}/node_modules/**/*.js", // 调试时不进入node_modules中的程序
        "<node_internals>/**", // 跳过内部node模块程序
      ]
    }
  ]
}
```

完整代码：点击访问

调试python

launch.json文件配置：

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "debug python",
      "type": "python",
      "request": "launch",
      "program": "${workspaceFolder}/debug.py", // 调试的程序
      "console": "integratedTerminal"
    }
  ]
}
```

完整代码：点击访问

参考资料

1. Debugging
2. Debugging in VS Code
3. Node.js debugging in VS Code
4. Variables Reference