概述

如果一个目录下存在一个tsconfig.json文件，那么它意味着这个目录是TypeScript项目的根目录。
tsconfig.json文件中指定了用来编译这个项目的根文件和编译选项。
一个项目可以通过以下方式之一来编译：

使用tsconfig.json

- 不带任何输入文件的情况下调用tsc，编译器会从当前目录开始去查找tsconfig.json文件，逐级向上搜索父目录。
- 不带任何输入文件的情况下调用tsc，且使用命令行参数--project（或-p）指定一个包含tsconfig.json文件的目录。

当命令行上指定了输入文件时，tsconfig.json文件会被忽略。

示例

tsconfig.json示例文件:

- 使用"files"属性

```
{
    "compilerOptions": {
        "module": "system",
        "noImplicitAny": true,
        "removeComments": true,
        "preserveConstEnums": true,
        "outFile": "../../built/local/tsc.js",
        "sourceMap": true
    },
    "files": [
        "core.ts",
        "sys.ts",
        "types.ts",
        "scanner.ts",
        "parser.ts",
        "utilities.ts",
        "binder.ts",
        "checker.ts",
        "emitter.ts",
        "program.ts",
        "commandLineParser.ts",
        "tsc.ts",
        "diagnosticInformationMap.generated.ts"
    ]
}
```

- 使用"include"和"exclude"属性

```
{
    "compilerOptions": {
        "module": "system",
```

```
            "noImplicitAny": true,
            "removeComments": true,
            "preserveConstEnums": true,
            "outFile": "../../built/local/tsc.js",
            "sourceMap": true
        },
        "include": [
            "src/**/*"
        ],
        "exclude": [
            "node_modules",
            "**/*.spec.ts"
        ]
    }
```

细节

"compilerOptions"可以被忽略，这时编译器会使用默认值。在这里查看完整的[编译器选项](./Compiler Options.md)列表。

"files"指定一个包含相对或绝对文件路径的列表。
"include"和"exclude"属性指定一个文件glob匹配模式列表。
支持的glob通配符有：

- * 匹配0或多个字符（不包括目录分隔符）
- ? 匹配一个任意字符（不包括目录分隔符）
- **/ 递归匹配任意子目录

如果一个glob模式里的某部分只包含*或.*，那么仅有支持的文件扩展名类型被包含在内（比如默认.ts，.tsx，和.d.ts， 如果allowJs设置能true还包含.js和.jsx）。

如果"files"和"include"都没有被指定，编译器默认包含当前目录和子目录下所有的TypeScript文件（.ts, .d.ts 和 .tsx），排除在"exclude"里指定的文件。JS文件（.js和.jsx）也被包含进来如果allowJs被设置成true。
如果指定了"files"或"include"，编译器会将它们结合一并包含进来。
使用"outDir"指定的目录下的文件永远会被编译器排除，除非你明确地使用"files"将其包含进来（这时就算用exclude指定也没用）。

使用"include"引入的文件可以使用"exclude"属性过滤。
然而，通过"files"属性明确指定的文件却总是会被包含在内，不管"exclude"如何设置。
如果没有特殊指定，"exclude"默认情况下会排
除node_modules，bower_components，jspm_packages和<outDir>目录。

任何被"files"或"include"指定的文件所引用的文件也会被包含进来。A.ts引用了B.ts，因此B.ts不能被排除，除非引用它的A.ts在"exclude"列表中。

需要注意编译器不会去引入那些可能做为输出的文件；比如，假设我们包含了index.ts，那么index.d.ts和index.js会被排除在外。
通常来讲，不推荐只有扩展名的不同来区分同目录下的文件。

tsconfig.json文件可以是个空文件，那么所有默认的文件（如上面所述）都会以默认配置选项编译。

在命令行上指定的编译选项会覆盖在tsconfig.json文件里的相应选项。

@types， typeRoots和types

默认所有*可见的*"@types"包会在编译过程中被包含进来。
node_modules/@types文件夹下以及它们子文件夹下的所有包都是*可见的*;
也就是说，./node_modules/@types/， ../node_modules/@types/
和../../node_modules/@types/等等。

如果指定了typeRoots，*只有*typeRoots下面的包才会被包含进来。
比如：

```
{
   "compilerOptions": {
       "typeRoots" : ["./typings"]
   }
}
```

这个配置文件会包含*所有*./typings下面的包，而不包含./node_modules/@types里面的包。

如果指定了types，只有被列出来的包才会被包含进来。
比如：

```
{
   "compilerOptions": {
        "types" : ["node", "lodash", "express"]
   }
}
```

这个tsconfig.json文件将*仅会*包含
./node_modules/@types/node，./node_modules/@types/lodash
和./node_modules/@types/express。/@types/。
node_modules/@types/*里面的其它包不会被引入进来。

指定"types": []来禁用自动引入@types包。

注意，自动引入只在你使用了全局的声明（相反于模块）时是重要的。
如果你使用import "foo"语句，TypeScript仍然会查找node_modules和
node_modules/@types文件夹来获取foo包。

使用extends继承配置

tsconfig.json文件可以利用extends属性从另一个配置文件里继承配置。

extends是tsconfig.json文件里的顶级属性（与compilerOptions，files，include，和exclude一样）。
extends的值是一个字符串，包含指向另一个要继承文件的路径。

在原文件里的配置先被加载，然后被来至继承文件里的配置重写。
如果发现循环引用，则会报错。

来至所继承配置文件的`files`，`include`和`exclude`*覆盖*源配置文件的属性。

配置文件里的相对路径在解析时相对于它所在的文件。

比如：

`configs/base.json`:

```
{
  "compilerOptions": {
    "noImplicitAny": true,
    "strictNullChecks": true
  }
}
```

`tsconfig.json`:

```
{
  "extends": "./configs/base",
  "files": [
    "main.ts",
    "supplemental.ts"
  ]
}
```

`tsconfig.nostrictnull.json`:

```
{
  "extends": "./tsconfig",
  "compilerOptions": {
    "strictNullChecks": false
  }
}
```

compileOnSave

在最顶层设置`compileOnSave`标记，可以让IDE在保存文件的时候根据`tsconfig.json`重新生成文件。

```
{
    "compileOnSave": true,
    "compilerOptions": {
        "noImplicitAny" : true
    }
}
```

要想支持这个特性需要Visual Studio 2015， TypeScript1.8.4以上并且安装atom-typescript插件。

模式

到这里查看模式: http://json.schemastore.org/tsconfig.