[typescript项目架构 1](#_Toc23478)

[typescript 1](#_Toc18160)

[Import vs require 1](#_Toc12948)

[Class-based vs Prototype-base 1](#_Toc14238)

Typescript项目webstorm的环境配置

File--> settings -->language & frameworks --> Typescript --> enable typescript compiler --> use tsconfig.json

# typescript项目架构

<http://stackoverflow.com/questions/31173738/typescript-getting-error-ts2304-cannot-find-name-require>

对于单独的typescript文件，用于demo的用途，可以使用declare var require:any;来进行node的关键词require的使用。

便于使用let fs=require(“fs”)

Node中使用npm install依赖时，部分包存在.d.ts文件，如moment，当不存在时可以使用上述方法引入包依赖。;

使用Number进行字符转数字

# typescript

使用tsc，系统读取文件tsconfig.ts文件进行系统配置，

Tsc file.ts必要时需要指定参数，来设置其运行环境

如: tsc --lib es6 ctrip-abroad-hotel.ts

参考: <https://zhongsp.gitbooks.io/typescript-handbook/content/doc/handbook/Compiler%20Options.html>

<https://www.typescriptlang.org/docs/handbook/compiler-options.html>

# Import vs require

ts中的module resolution有两种策略: Classic 和node

前者根据文件所在目录依次向上寻找、为向后兼容

后者模仿nodejs的风格。

Node使用require来导入包依赖，对于参数为相对路径，进入包依赖后，根据文件中的package.json来进行查找main主要入口文件；若路径为绝对路径，则查找本目录的node\_modules，若不存在，则往上寻找。

<https://www.typescriptlang.org/docs/handbook/module-resolution.html>

# Class-based vs Prototype-base

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model>

# 编译选项文件tsconfig

Include、exclude、files三个指令用于包含或移除至编译文件范畴。

但是files作用域更强，即使exclude指定了，若出现在files，那么仍然需要编译。

# 编译生成的文件

.js.map: 使用js文件与源代码.ts文件的映射文件。

细节误区

对于函数参数使用？标记，获取后，若调用时未传递，则为undefined，

但是if() 和？：两者不同

Promise的reject，在调用时需要添加try catch

Try{} catch{}

异常抛出