빌드 도구들

- Babel
- Browserify
- Duo
- Grunt
- Gulp
- Jspm
- Webpack
- MSBuild
- NuGet

Babel

설치

```
npm install @babel/cli @babel/core @babel/preset-typescript --save-dev
```

.babelrc

```
{
  "presets": ["@babel/preset-typescript"]
}
```

커맨드 라인 인터페이스 사용

```
./node_modules/.bin/babel --out-file bundle.js src/index.ts
```

package.json

```
{
  "scripts": {
    "build": "babel --out-file bundle.js main.ts"
  },
}
```

커맨드 라인 인터페이스 사용

```
npm run build
```

Browserify

설치

```
npm install tsify
```

커맨드 라인 인터페이스 사용

```
browserify main.ts -p [ tsify --noImplicitAny ] > bundle.js
```

API 사용

```
var browserify = require("browserify");
var tsify = require("tsify");

browserify()
    .add("main.ts")
    .plugin("tsify", { noImplicitAny: true })
    .bundle()
    .pipe(process.stdout);
```

자세한 내용: smrq/tsify

Duo

설치

```
npm install duo-typescript
```

커맨드 라인 인터페이스 사용

```
duo --use duo-typescript entry.ts
```

API 사용

```
var Duo = require("duo");
var fs = require("fs")
var path = require("path")
var typescript = require("duo-typescript");

var out = path.join(__dirname, "output.js")

Duo(__dirname)
    .entry("entry.ts")
    .use(typescript())
    .run(function (err, results) {
        if (err) throw err;
        // 컴파일된 결과를 출력 파일에 작성합니다
        fs.writeFileSync(out, results.code);
    });
```

자세한 내용: frankwallis/duo-typescript

Grunt

설치

```
npm install grunt-ts
```

기본 Gruntfile.js

```
module.exports = function(grunt) {
    grunt.initConfig({
        ts: {
            default : {
                src: ["**/*.ts", "!node_modules/**/*.ts"]
            }
        }
    });
    grunt.loadNpmTasks("grunt-ts");
    grunt.registerTask("default", ["ts"]);
};
```

자세한 내용: TypeStrong/grunt-ts

Gulp

설치

```
npm install gulp-typescript
```

기본 gulpfile.js

```
var gulp = require("gulp");
var ts = require("gulp-typescript");

gulp.task("default", function () {
    var tsResult = gulp.src("src/*.ts")
        .pipe(ts({
              noImplicitAny: true,
              out: "output.js"
        }));
    return tsResult.js.pipe(gulp.dest("built/local"));
});
```

자세한 내용: ivogabe/gulp-typescript

Jspm

설치

```
npm install -g jspm@beta
```

주의사항: 현재 jspm의 TypeScript 지원은 0.16beta 입니다.

자세한 내용: TypeScriptSamples/jspm

Webpack

설치

```
npm install ts-loader --save-dev
```

Webpack 2 사용 시 기본 webpack.config.js

```
module.exports = {
    entry: "./src/index.tsx",
    output: {
        path: '/',
        filename: "bundle.js"
    },
    resolve: {
        extensions: [".tsx", ".ts", ".js", ".json"]
    },
    module: {
        rules: [
            // '.ts' 또는 '.tsx' 확장자를 가진 모든 파일은 'ts-loader'에 의해 처리됩니다.
            { test: /\.tsx?$/, use: ["ts-loader"], exclude: /node_modules/ }
        ]
    }
}
```

Webpack 1 사용 시 기본 webpack.config.js

```
module.exports = {
    entry: "./src/index.tsx",
    output: {
        filename: "bundle.js"
    },
    resolve: {
        // '.ts'와 '.tsx'를 해석 가능한 확장자로 추가합니다.
        extensions: ["", ".webpack.js", ".web.js", ".ts", ".tsx", ".js"]
    },
    module: {
        loaders: [
            // '.ts' 또는 '.tsx' 확장자를 가진 모든 파일은 'ts-loader'에 의해 처리됩니다.
            { test: /\.tsx?$/, loader: "ts-loader" }
        ]
    }
}
```

ts-loader에 대한 자세한 내용은 여기를 참조하세요.

대안:

- awesome-typescript-loader

MSBuild

로컬에 설치된 `Microsoft.TypeScript.Default.props` (맨 위)와 `Microsoft.TypeScript.targets` (맨 아래) 파일을 포함하도록 프로젝트 파일을 업데이트하세요:

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- 하단에 default props 포함 -->
  <Import
      Project="$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)\TypeScript\Microsoft.Typ
      Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)\TypeScript\Mic

  <!-- TypeScript 환경 설정 -->
  <PropertyGroup Condition="'$(Configuration)' == 'Debug'">
    <TypeScriptRemoveComments>false</TypeScriptRemoveComments>
    <TypeScriptSourceMap>true</TypeScriptSourceMap>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)' == 'Release'">
    <TypeScriptRemoveComments>true</TypeScriptRemoveComments>
    <TypeScriptSourceMap>false</TypeScriptSourceMap>
  </PropertyGroup>

  <!-- 하단에 default targets 포함 -->
  <Import
      Project="$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)\TypeScript\Microsoft.Typ
      Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)\TypeScript\Mic
</Project>
```

MSBuild 컴파일러 옵션 정의에 대한 자세한 내용: MSBuild 프로젝트의 컴파일러 옵션 설정

NuGet

- 우-클릭 -> Manage NuGet Packages
- `Microsoft.TypeScript.MSBuild`를 검색하세요
- `Install` 클릭
- 설치가 완료되면 다시 빌드 하세요!

자세한 내용은 패키지 매니저 다이얼로그와 NuGet과 nightly builds 사용을 참고하세요