这个快速上手指南会告诉你如何结合使用TypeScript和Knockout.js。

这里我们假设你已经会使用Node.js和npm

新建工程

首先，我们新建一个目录。
暂时命名为`proj`，当然了你可以使用任何喜欢的名字。

```
mkdir proj
cd proj
```

接下来，我们按如下方式来组织这个工程：

```
proj/
   ├ src/
   └ built/
```

TypeScript源码放在`src`目录下，结过TypeScript编译器编译后，生成的文件放在`built`目录里。

下面创建目录：

```
mkdir src
mkdir built
```

初始化工程

现在将这个文件夹转换为npm包。

```
npm init
```

你会看到一系列提示。
除了入口点外其它设置都可以使用默认值。
你可以随时到生成的`package.json`文件里修改这些设置。

安装构建依赖

首先确保TypeScript已经全局安装。

```
npm install -g typescript
```

我们还要获取Knockout的声明文件，它描述了这个库的结构供TypeScript使用。

```
npm install --save @types/knockout
```

获取运行时依赖

我们需要Knockout和RequireJS。
RequireJS是一个库，它可以让我们在运行时异步地加载模块。

有以下几种获取方式：

1. 手动下载文件并维护它们。
2. 通过像Bower这样的包管理下载并维护它们。
3. 使用内容分发网络（CDN）来维护这两个文件。

我们使用第一种方法，它会简单一些，但是Knockout的官方文档上有讲解如何使用CDN，更多像RequireJS一样的代码库可以在cdnjs上查找。

下面让我们在工程根目录下创建`externals`目录。

```
mkdir externals
```

然后下载Knockout和下载RequireJS到这个目录里。
最新的压缩后版本就可以。

添加TypeScript配置文件

下面我们想把所有的TypeScript文件整合到一起 - 包括自己写的和必须的声明文件。

我们需要创建一个`tsconfig.json`文件，包含了输入文件列表和编译选项。
在工程根目录下创建一个新文件`tsconfig.json`，内容如下：

```
{
    "compilerOptions": {
        "outDir": "./built/",
        "sourceMap": true,
        "noImplicitAny": true,
        "module": "amd",
        "target": "es5"
    },
    "files": [
        "./src/require-config.ts",
        "./src/hello.ts"
    ]
}
```

这里引用了`typings/index.d.ts`，它是Typings帮我们创建的。
这个文件会自动地包含所有安装的依赖。

你可能会对`typings`目录下的`browser.d.ts`文件感到好奇，尤其因为我们将在浏览器里运行代码。
其实原因是这样的，当目标为浏览器的时候，一些包会生成不同的版本。
通常来讲，这些情况很少发生并且在这里我们不会遇到这种情况，所以我们可以忽略`browser.d.ts`。

你可以在这里查看更多关于`tsconfig.json`文件的信息

写些代码

下面我们使用Knockout写一段TypeScript代码。
首先，在src目录里新建一个hello.ts文件。

```
import * as ko from "knockout";

class HelloViewModel {
    language: KnockoutObservable<string>
    framework: KnockoutObservable<string>

    constructor(language: string, framework: string) {
        this.language = ko.observable(language);
        this.framework = ko.observable(framework);
    }
}

ko.applyBindings(new HelloViewModel("TypeScript", "Knockout"));
```

接下来，在src目录下再新建一个require-config.ts文件。

```
declare var require: any;
require.config({
    paths: {
        "knockout": "externals/knockout-3.4.0",
    }
});
```

这个文件会告诉RequireJS从哪里导入Knockout，好比我们在hello.ts里做的一样。
你创建的所有页面都应该在RequireJS之后和导入任何东西之前引入它。
为了更好地理解这个文件和如何配置RequireJS，可以查看文档。

我们还需要一个视图来显示HelloViewModel。
在proj目录的根上创建一个文件index.html，内容如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Hello Knockout!</title>
    </head>
    <body>
        <p>
            Hello from
            <strong data-bind="text: language">todo</strong>
            and
            <strong data-bind="text: framework">todo</strong>!
        </p>

        <p>Language: <input data-bind="value: language" /></p>
        <p>Framework: <input data-bind="value: framework" /></p>

        <script src="./externals/require.js"></script>
        <script src="./built/require-config.js"></script>
        <script>
            require(["built/hello"]);
```

```
        </script>
    </body>
</html>
```

注意，有两个script标签。
首先，我们引入RequireJS。
然后我们再在`require-config.js`里映射外部依赖，这样RequireJS就能知道到哪里去查找它们。
最后，使用我们要去加载的模块去调用`require`。

将所有部分整合在一起

运行

```
tsc
```

现在，在你喜欢的浏览器打开`index.html`，所有都应该好用了。
你应该可以看到页面上显示“Hello from TypeScript and Knockout!”。
在它下面，你还会看到两个输入框。
当你改变输入和切换焦点时，就会看到原先显示的信息改变了。