JavaScriptからの移行ガイド

前提事項：

- あなたはJavaScriptを知っている
- プロジェクトで使用されているパターンや構築ツール(webpackなど)を知っている

そのような前提をもとに、一般にプロセスは以下のステップからなる：

- `tsconfig.json` を追加する
- ソースコードの拡張子を `.js` から `.ts` に変更する。any型を使ってエラーを抑制する
- 新しいコードはTypeScriptで記述し、できるだけ `any` を使わないようにする
- 古いコードに戻り、型アノテーションを追加し、バグを修正する
- サードパーティ製JavaScriptコードの型定義を使用する

これらの点のいくつかをさらに議論しましょう。

すべてのJavaScriptは有効なTypeScriptであることに注意してください。つまり、TypeScriptコンパイラにJavaScriptをいくつか与えると、TypeScriptコンパイラによって生成されたJavaScriptは元のJavaScriptとまったく同じように動作します。つまり、拡張子を `.js` から `.ts` に変更しても、コードベースに悪影響はありません。

エラーを抑制する

TypeScriptはすぐに型チェックを開始しますが、元のJavaScriptコードはあなたが思っていたほど整っていない可能性があります。したがって、エラーが発生します。これらのエラーの多くは、 `any` を使用して抑制することができます。例：

```
var foo = 123;
var bar = 'hey';

bar = foo; // ERROR: cannot assign a number to a string
```

エラーが正しいものであったとしても(ほとんどの場合、推測された情報は、開発者の考えよりも優れています)、あなたはおそらくTypeScriptで新しいコードを書きつつ、古いコードを徐々に更新することに集中することになるでしょう。型エラーを抑制する方法は以下の通りです:

```
var foo = 123;
var bar = 'hey';

bar = foo as any; // Okay!
```

他の場所で何かを `any` とアノテーションしたいことがあるかもしれません。例:

```
function foo() {
    return 1;
}
var bar = 'hey';
bar = foo(); // ERROR: cannot assign a number to a string
```

エラーを抑制した結果：

```
function foo(): any { // Added `any`
    return 1;
}
var bar = 'hey';
bar = foo(); // Okay!
```

> 注意：エラーを抑止することは危険ですが、新しく書いたTypeScriptコードでエラー通知を受けることができます。 `// TODO:` コメントを残したほうが良いかもしれません。

サードパーティ製JavaScript

JavaScriptをTypeScriptに変更することはできますが、世界全体を変更してTypeScriptを使用することはできません。これは、TypeScriptのアンビエント定義のサポートが必要な部分です。最初は `vendor.d.ts` (`.d.ts` 拡張子が、これは宣言ファイルであることを指定しています)を作成し、色々と雑多に詰め込むことをオススメします。あるいは、ライブラリに固有のファイルを作成します(例えば、jqueryのための `jquery.d.ts`)。

> 注意： DefinitelyTyped と呼ばれるOSSリポジトリには、上位90%のJavaScriptライブラリには、よくメンテナンスされた、厳密な型定義が存在します。ここにあるように独自の定義を作成する前に、そこを見ることをお勧めします。それにもかかわらず、この素早くもダーティな方法は、初期のTypeScriptにおける障壁を減らすために不可欠な知識です。

`jquery` の場合を考えてください。jqueryのために些細な定義を素早く簡単に作ることができます:

```
declare var $: any;
```

場合によっては、明示的なアノテーションを何か(例えば `JQuery`)に追加したいかもしれないし、型宣言スペースに何かが必要な場合もあります。 `type` キーワードを使って簡単に行うことができます：

```
declare type JQuery = any;
declare var $: JQuery;
```

これにより、将来の更新が簡単になります。

ここでも、 DefinitelyTyped に高品質の「jquery.d.ts」が存在します。しかし、サードパーティ製のJavaScriptを使用している場合、JavaScript -> TypeScriptの摩擦をすばやく克服する方法を、あなたは既に知っています。次に、アンビエント宣言について詳しく説明します。

サードパーティ製NPMモジュール

グローバル変数宣言と同様に、グローバルモジュールを簡単に宣言できます。例えば `jquery` をモジュール(https://www.npmjs.com/package/jquery)として使用したい場合は、あなた自身で以下のように書くことができます：

```
declare module "jquery";
```

必要に応じてファイルにインポートすることができます：

```
import * as $ from "jquery";
```

> さらに、より高品質のjqueryモジュール宣言を提供する DefinitelyTyped に高品質の `jquery.d.ts` が存在します。しかし、あなたのライブラリには存在しないかもしれません。なので、既に説明したように、少ない摩擦で移行を継続するための素早い方法があります🚀

外部の非jsリソース

JS以外のファイルであっても、単純な `*` スタイルの宣言でインポートできます。例えば `.css` です(webpackスタイルのローダーやCSSモジュールのようなものを使用している場合)。理想的には、 `global.d.ts` ファイルの中でインポートします。

```
declare module "*.css";
```

これで、 `import * as foo from "./some/file.css"` をインポートすることができます。

同様に、htmlテンプレート(angularなど)を使用している場合は、次のことを行えます:

```
declare module "*.html";
```

その他

もしチームがTypeScriptへの移行に積極的でなくて、もっと静かにアップグレードしたいときは、チームを説得せずに気づかれずにアップグレードする方法について、TypeScriptチームがブログ記事を書いています。