`$localize` Global Import Migration

What does this schematic do?

If you're using i18n, this schematic adds an import statement for `@angular/localize` to `polyfills.ts` that will look something like this:

```
/***************************************************************
 * Load `$localize` - used if i18n tags appear in Angular templates.
 */
import '@angular/localize/init';
```

It also lists `@angular/localize` as a dependency in your app's `package.json` to ensure the import is found.

```
"devDependencies": {
  ...
  "@angular/localize": "...",
  ...
}
```

`@angular/localize` is a new package that supports i18n of messages in Ivy applications. This package requires a global `$localize` symbol to exist. The symbol is loaded by importing the `@angular/localize/init` module, which has the side-effect of attaching it to the global scope.

Why is this migration necessary?

Prior to Angular version 9, Angular's internationalization (i18n) system inlined translated messages into the compiled output as part of this template compilation. This approach required running the template compiler once per target locale, often leading to slow production build times.

In the new i18n system, the Angular compiler tags i18n messages in the compiled code with a global `$localize` handler. The inlining of translations then occurs as a post-compilation step for each locale. Because the application does not need to be built again for each locale, this makes the process much faster.

The post-compilation inlining step is optional—for example during development or if the translations will be inlined at runtime. Therefore this global `$localize` must be available on the global scope at runtime. To make `$localize` available on the global scope, each application must now import the `@angular/localize/init` module. This has the side-effect of attaching a minimal implementation of `$localize` to the global scope.

If this import is missing, you will see an error message like this:

```
Error: It looks like your application or one of its dependencies is using i18n.
Angular 9 introduced a global `$localize()` function that needs to be loaded.
Please run `ng add @angular/localize` from the Angular CLI.
(For non-CLI projects, add `import '@angular/localize/init';` to your polyfills.ts
file)
```

This schematic automatically adds the `@angular/localize/init` import for you if your app uses Angular's i18n APIs.

Why is my tslint failing?

The import of `@angular/localize/init` may cause a tslint error for `no-import-side-effect` because it adds to the global context (that is, a side effect). To fix this error, add the following to your `tslint.config`:

```
"no-import-side-effect": [
  true,
  {
    "ignore-module": "(core-js/.*|zone\\.js/.*|@angular/localize/init)$"
  }
]
```

Do I need to change how I write i18n in my Angular templates?

The template syntax for i18n has not changed, so you will still want to use the `i18n` attribute as you did before.