@name NgModule.id Set to module.id anti-pattern @category compiler @shortDescription Setting NgModule.id to module.id is a common anti-pattern

@description Using `module.id` as an NgModule `id` is a common anti-pattern and is likely not serving a useful purpose in your code.

NgModules can be declared with an `id`:

```
@NgModule({
  id: 'my_module'
})
export class MyModule {}
```

Declaring an `id` makes the NgModule available for lookup via the `getNgModuleById()` operation. This functionality is rarely used, mainly in very specific bundling scenarios when lazily loading NgModules without obtaining direct references to them. In most Angular code, ES dynamic `import()` (`import('./path/to/module')`) should be used instead, as this provides a direct reference to the NgModule being loaded without the need for a global registration side-effect.

If you are not using `getNgModuleById`, you do not need to provide `id`s for your NgModules. Providing one has a significant drawback: it makes the NgModule non-tree-shakable, which can have an impact on your bundle size.

In particular, the pattern of specifying `id: module.id` results from a misunderstanding of `@NgModule.id`. In earlier versions of Angular, it was sometimes necessary to include the property `moduleId: module.id` in `@Component` metadata.

Using `module.id` for `@NgModule.id` likely results from confusion between `@Component.moduleId` and `@NgModule.id`. `module.id` would not typically be useful for `getNgModuleById()` operations as the `id` needs to be a well-known string, and `module.id` is usually opaque to consumers.

@debugging You can remove the `id: module.id` declaration from your NgModules. The compiler ignores this declaration and issues this warning instead.