ByReferenceModuleOpaqueKeyFactory

August 18, 2026

ByReferenceModuleOpaqueKeyFactory

Kind: Class

Source: `packages/core/injector/opaque-key-factory/by-reference-module-opaque-key-factory.ts`

Part of: Core

`ByReferenceModuleOpaqueKeyFactory` creates opaque string keys for module definitions using their object reference identity. It provides separate key generation paths for static modules and dynamic modules, allowing the injector to distinguish module registrations without relying on module names alone.

Implements: `ModuleOpaqueKeyFactory`

Methods

Method	Signature	Returns
`createForStatic`	`createForStatic(moduleCls: Type, originalRef: Type	ForwardReference)`
`createForDynamic`	`createForDynamic(moduleCls: Type, dynamicMetadata: Omit<DynamicModule, 'module'>, originalRef: DynamicModule	ForwardReference)`

Where it refuses work

- `ByReferenceModuleOpaqueKeyFactory` stops the work with an early return when `originalRef[K_MODULE_ID]`.

Diagram

```
graph LR
  A[Module Definition] --> B[ByReferenceModuleOpaqueKeyFactory]
  B --> C{Module Type}
```

```
  C -->|Static| D[createForStatic]
  C -->|Dynamic| E[createForDynamic]
  D --> F[Static Opaque Key]
  E --> G[Dynamic Opaque Key]
  F --> H[Injector Module Registry]
  G --> H
```

Usage

```
import { ByReferenceModuleOpaqueKeyFactory } from './by-reference-module-opaque-key-factory';

const opaqueKeyFactory = new ByReferenceModuleOpaqueKeyFactory();

const staticModuleKey = opaqueKeyFactory.createForStatic();
const dynamicModuleKey = opaqueKeyFactory.createForDynamic();

console.log(staticModuleKey);
console.log(dynamicModuleKey);

// Store the generated keys when registering modules in an injector registry.
const moduleRegistry = new Map<string, unknown>();
moduleRegistry.set(staticModuleKey, StaticModule);
moduleRegistry.set(dynamicModuleKey, dynamicModuleDefinition);
```

AI Coding Instructions

- Use `createForStatic()` when registering a standard module definition and `createForDynamic()` for dynamically configured module instances.
- Treat returned keys as opaque identifiers; do not parse, manually construct, or depend on their string format.
- Preserve reference-based identity semantics when changing this factory or its callers, since injector module deduplication depends on unique keys.
- Ensure static and dynamic module registration paths continue to use their respective factory methods to avoid key collisions.

Relationships

- IMPORTS → `DynamicModule`
- IMPORTS → `ForwardReference`
- IMPORTS → `Type`
- IMPORTS → `randomStringGenerator`