InitializeOnPreviewAllowlist

August 17, 2026

InitializeOnPreviewAllowlist

Kind: Class

Source: `packages/core/inspector/initialize-on-preview.allowlist.ts`

Part of: Core

`InitializeOnPreviewAllowlist` tracks which component or entity types are permitted to run initialization logic while the inspector is in preview mode. It provides a small allowlist API: use `add()` to register allowed entries and `has()` to check whether preview initialization should be enabled for a given entry.

Methods

Method	Signature	Returns
`add`	`add(type: Type)`	`void`
`has`	`has(type: Type)`	`void`

Diagram

```
graph LR
  A[Inspector Preview Flow] --> B[InitializeOnPreviewAllowlist]
  B -->|add(type)| C[Allowed initialization entries]
  A -->|has(type)| B
  B -->|true| D[Run initialization in preview]
  B -->|false| E[Skip initialization]
```

Usage

```
import { InitializeOnPreviewAllowlist } from './initialize-on-preview.allowlist';

const previewAllowlist = new InitializeOnPreviewAllowlist();
```

```ts
// Allow a component type to initialize during inspector preview.
previewAllowlist.add('my-component');

function shouldInitializeInPreview(componentType: string): boolean {
  return previewAllowlist.has(componentType);
}

if (shouldInitializeInPreview('my-component')) {
  // Run preview-safe initialization.
}
```

AI Coding Instructions

- Use `add()` during inspector or component registration to explicitly opt entries into preview initialization.
- Always guard preview-only initialization with `has()` rather than assuming normal runtime initialization is safe in preview.
- Keep allowlisted initialization side-effect-safe; preview code should avoid unexpected persistence, network calls, or global mutations.
- Reuse the same allowlist instance across the relevant inspector preview flow so registrations are visible to checks.

Relationships

- IMPORTS → `Type`