TreeNode

August 17, 2026

TreeNode

Kind: Class

Source: `packages/core/injector/topology-tree/tree-node.ts`

Part of: Core

`TreeNode` models a node in the injector topology tree, linking a value to its parent and child nodes. It provides utilities for maintaining relationships, moving nodes between parents, calculating nesting depth, and preventing circular dependency paths.

Methods

Method	Signature	Returns
`addChild`	`addChild(child: TreeNode<T>)`	`void`
`removeChild`	`removeChild(child: TreeNode<T>)`	`void`
`relink`	`relink(parent: TreeNode<T>)`	`void`
`getDepth`	`getDepth()`	`void`
`hasCycleWith`	`hasCycleWith(target: T)`	`void`

Properties

Property	Type
`value`	`T`
`children`	`any`

Where it refuses work

- `TreeNode` stops the work with an early return when `visited.has(current!)`, in 2 places.

- `TreeNode` stops the work with an early return when `current.value === target`.

Diagram

```
graph LR
  Root[Root TreeNode] --> Feature[Feature TreeNode]
  Feature --> ServiceA[Service A TreeNode]
  Feature --> ServiceB[Service B TreeNode]

  ServiceB -. relink() .-> Root
  ServiceA -. hasCycleWith() .-> Feature
```

Usage

```
import { TreeNode } from './tree-node';

type Provider = {
  name: string;
};

const app = new TreeNode<Provider>({ name: 'AppModule' });
const feature = new TreeNode<Provider>({ name: 'FeatureModule' });
const usersService = new TreeNode<Provider>({ name: 'UsersService' });

app.addChild(feature);
feature.addChild(usersService);

console.log(usersService.getDepth()); // 2

// Prevent creating a relationship that would introduce a cycle.
if (!app.hasCycleWith(usersService)) {
  app.addChild(usersService);
}

// Move the service directly under the application node.
usersService.relink(app);

// Remove a node when it is no longer part of the topology.
app.removeChild(feature);
```

AI Coding Instructions

- Use `addChild()` and `removeChild()` rather than manually changing `parent` or `children` relationships.

- Call `hasCycleWith()` before adding or relinking nodes when constructing dependency graphs dynamically.
- Use `relink()` to move an existing node; it removes the node from its previous parent before attaching it to the new parent.
- Treat `getDepth()` as topology-derived state and avoid caching its result if the tree can be relinked.
- Keep this class focused on tree structure; injector-specific resolution logic belongs in the topology tree integration layer.

How it works

`TreeNode<T>` is a generic, mutable tree-node class. It stores a read-only `value`, a public mutable `Set` of child nodes, and a private parent-node reference that may be `null`. tree-node.ts:1-8

- **Construction:** `new TreeNode({ value, parent })` assigns the supplied value and parent reference, and starts with an empty `children` set. Construction does **not** add the new node to the supplied parent's `children` set. tree-node.ts:3-9
- `addChild(child)`**:** adds that exact `TreeNode<T>` object to `children`. Because `children` is a `Set`, the collection is keyed by node identity rather than by `value`. This method does not change `child`'s private parent reference. tree-node.ts:3,11-13
- `removeChild(child)`**:** removes that node object from `children`; removing an object that is absent has no additional code path. It does not change the removed child's parent reference. tree-node.ts:15-17
- `relink(parent)`**:** if the node currently has a parent, removes itself from that parent's child set; it then replaces its parent reference and adds itself to the new parent's child set. tree-node.ts:19-24 It accepts only a non-null `TreeNode<T>` parent in its type signature. tree-node.ts:19
- `getDepth()`**:** walks parent references beginning with the node itself and returns the number of nodes encountered through the `null` parent. Thus, a root node has depth `1`. tree-node.ts:26-43 It tracks visited node objects; if following parents reaches a previously visited node, it returns `-1` instead of a depth. tree-node.ts:27-43
- `hasCycleWith(target)`**:** walks from this node through its parents and returns `true` when a current node's `value === target`, including this node's own value. tree-node.ts:46-56 If the parent chain ends without a match, or a repeated node is reached before a match, it returns `false`. tree-node.ts:57-64

There is no runtime validation of constructor inputs, child-parent consistency, relinking to a descendant, or value uniqueness, and the class contains no explicit error throwing. tree-node.ts:6-24

Within `TopologyTree`, nodes wrap `Module` instances: the root is created with `parent: null`, newly discovered module nodes are explicitly added to their parent after construction, and an existing node may be relinked after depth and ancestor-value checks. topology-tree.ts:8-15,33-54