Roots are derived, not added

Every tree has **roots** — the skills you must already hold *before* you can enter the tree. And here's the elegant part: you never place roots by hand. The tree grows its own.

The derivation is mechanical. Look at the lowest skills in the tree — the ones whose prerequisites are *not* themselves in the tree. Those external prerequisites are the tree's roots, surfaced one level deep beneath it. In one line:

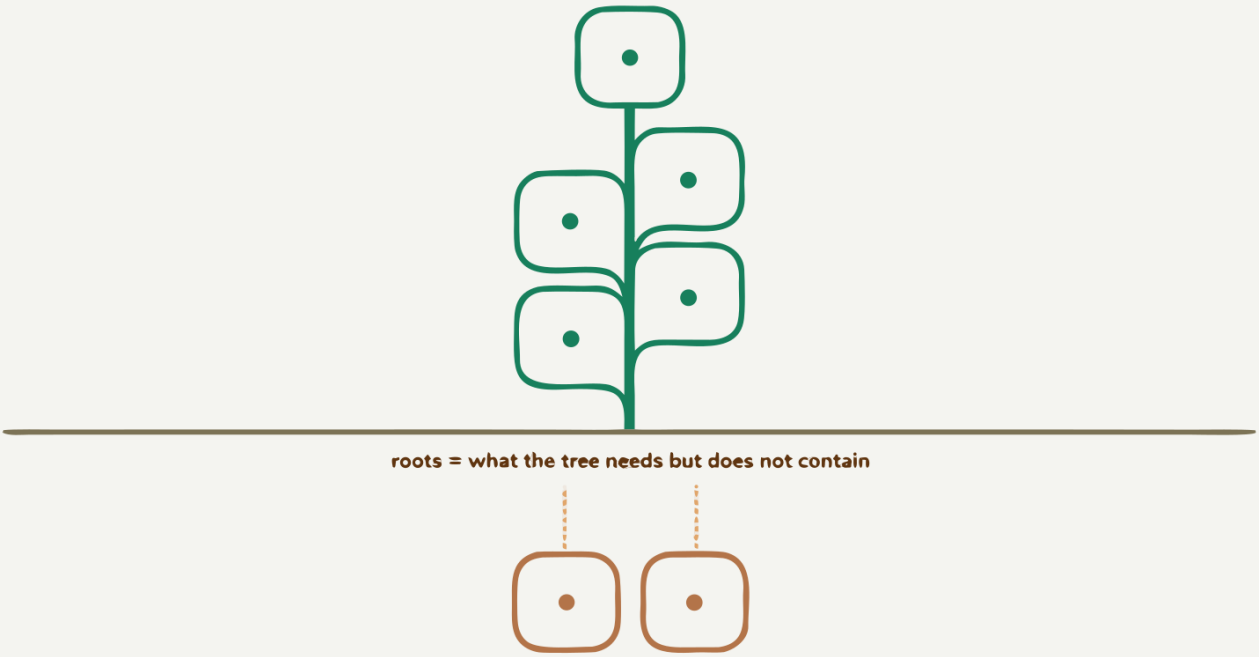
Roots = prerequisites of the tree's skills that aren't in the tree.

Say a "Frameworks" tree starts at "React I," and React I requires "JavaScript fundamentals" — a skill that lives in the JavaScript tree, not this one. Then JavaScript fundamentals appears *as a root* under the Frameworks tree. It isn't defined there (skills are defined only once); it's shown there, below the ground line, telling every reader: *hold this before you climb here.*

Why derive instead of declare?

- **Roots can't go stale.** Change a prerequisite on a skill, and every tree's roots update themselves. Hand-placed entry requirements drift; derived ones can't.
- **Roots can't be wishful.** Nobody can decorate a tree with aspirational entry bars that no skill inside actually demands. If it's a root, some skill in the tree genuinely requires it.
- **It keeps the mental model whole.** There is exactly one kind of connection in the method — the prerequisite. Roots aren't a second system; they're the same connections viewed from the other side of the tree's boundary.

Roots are the tree's honest answer to "am I ready to start this?" — and later, at the forest level, the same idea becomes the honest answer to "am I ready for this role?"



Revision #2
Created 2026-07-08 16:08:49 UTC by Admin
Updated 2026-07-10 09:30:15 UTC by Admin