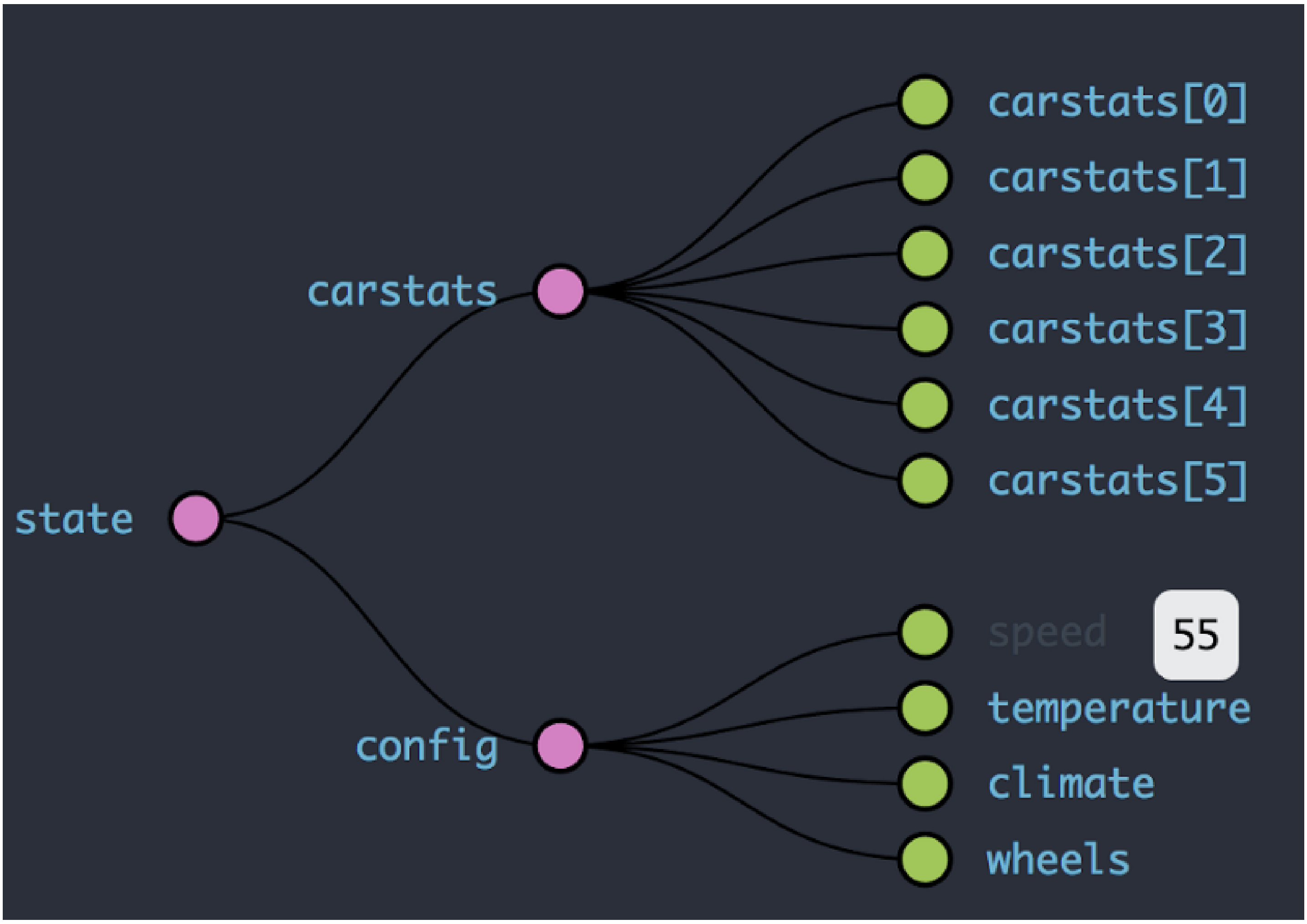
2.5 Redux Three Principles

I've mentioned `Flux` a few times. Flux is a **pattern of state management**, not a downloadable tool like Redux. Redux, on the other hand, is a **practical implementation of the Flux pattern** and has three main principles.

2.5.1 Single source of truth

> *The state of the entire application is stored in an object tree within a **single store**.*



Since all states exist in one place, this is called a `single source of truth`.

This `one-store` approach of Redux is one of the primary differences between it and **Flux's multiple store** approach.

What are the advantages of a single state tree? This makes it easier to debug

What are the advantages of a single state tree? This makes it easier to debug applications or perform internal inspections and to easily implement some

features that were previously difficult to implement (for example, undo / redo).

2.5.2 State is read-only

> *The only way to change the state is to **emit an action** that describes what happened.*

In other words, the application does not directly change the state, but instead expresses the intention to transform the state by passing the action.

In fact, if you look at the Redux API, you can see that it consists of just four methods:

```
store.dispatch(action)
store.subscribe(listener)
store.getState()
replaceReducer(nextReducer)
```

As you can see, there is no **setState()** method. Therefore, **passing an action**is the only channel that can mutate the state of the application.

2.5.3 Changes are made with pure functions

> *You write reducers as **pure functions** to specify the concrete way the state tree is transformed by action.*

Reducers are pure functions that take a previous state and action and return a new state. Keep in mind that you must return a **new state** object instead of changing the old state.

> *"Given the same arguments, it should calculate the next state and return it. No surprises. No side effects. No API calls. No mutations. Just a calculation."— Redux Docs*

The pure function has the following characteristics:

- It does not make outside network or database calls.
- Its return value depends solely on the values of its parameters.
- Its arguments should be considered “immutable”, meaning they should not be changed.
- Calling a pure function with the same set of arguments will always return the same value.