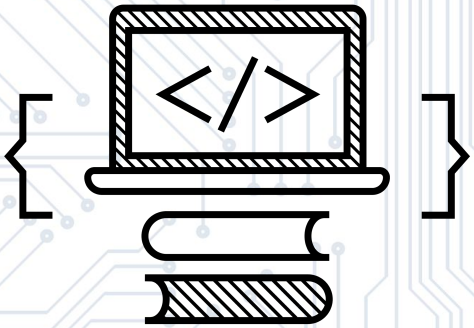


The React Context API

Web Development

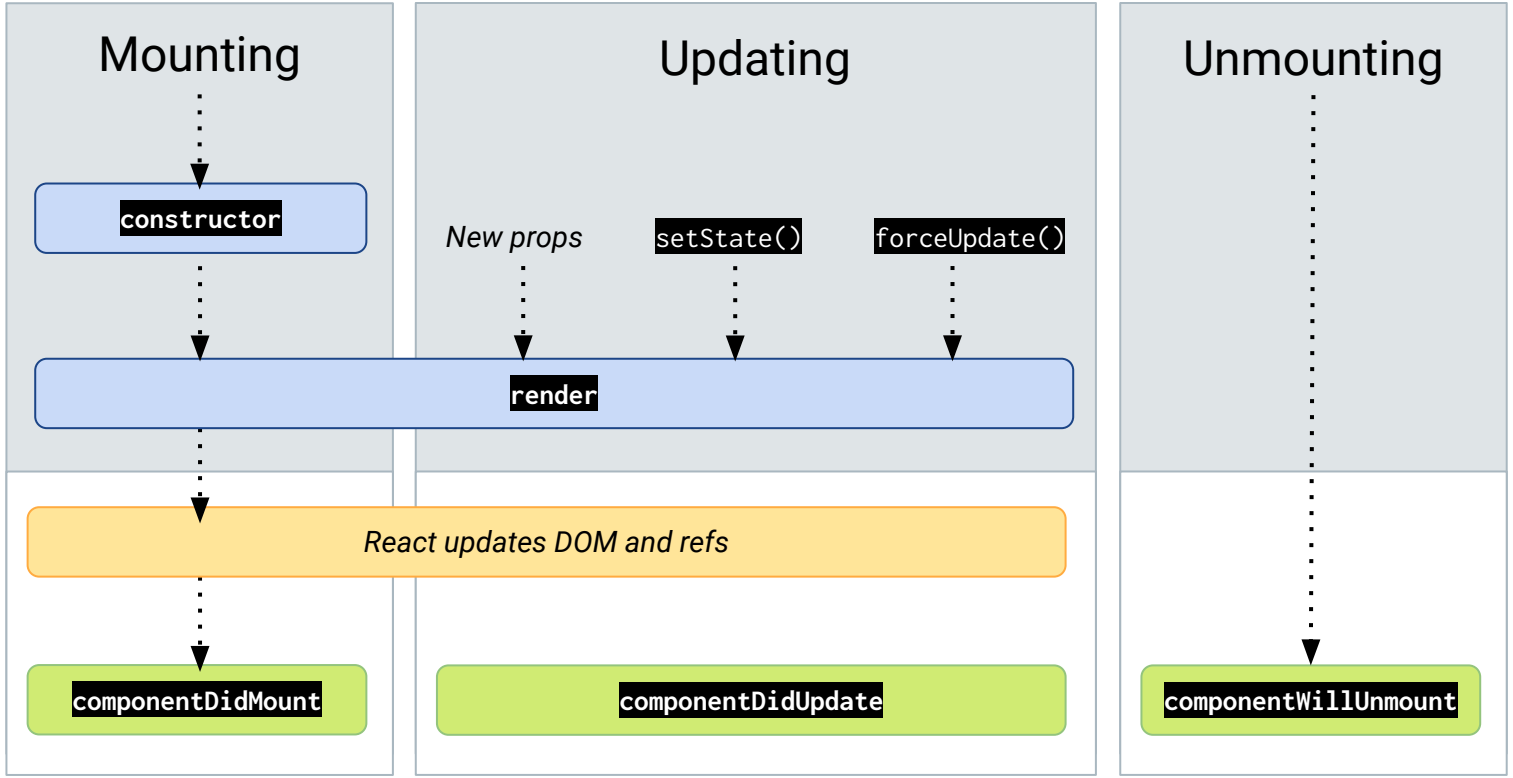
Lesson 20.2

Giving Context

Component Lifecycle

Render Phase
Pure and has no side effects. May be paused, aborted, or restarted by React.

Commit Phase
Can work with DOM, run side effects, and schedule updates.



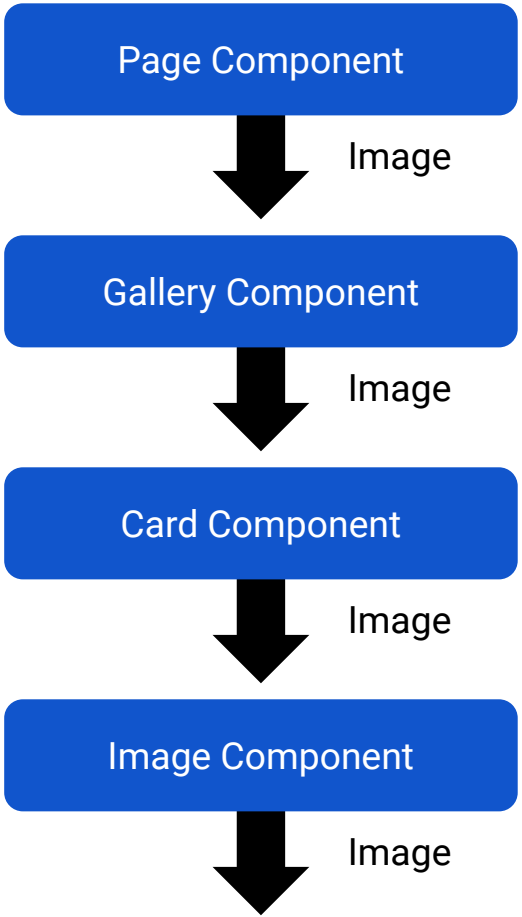
Prop Drilling

Prop drilling is the process you go through to get data to parts of the React component tree.

Although it seems tedious, prop drilling is often necessary to avoid complicating the global state of your application

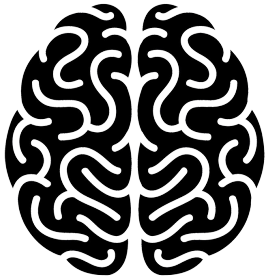
Remember, it's often best to keep our state as close as possible to where it's relevant.

Why can't we just add state to the lowest level component?



Container vs. Presentational Components

Separate the logic from the looks



Container components include the logic of the application, which is often stored in the component's state, requiring us to use class components instead of functional components.

The Gallery Page component from the previous activity is a container component.

Presentational components are primarily used for UI elements like layout, and often use stateless functional components (often referred to as “dumb components”) to render their content. All of the components in the `components` folder from the previous activity are presentational components.

Not **all** presentational components have to be stateless. Some may contain UI elements in their state.

Context API

The Context API allows you to pass data through the component tree without having to pass down props through many component levels.

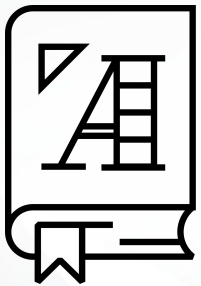
The Context API consists of two key parts:

01 **Provider**

The **provider** wraps our presentational component and stores data in its state.

02 **Consumer**

The **consumer** goes inside the presentational component and allows us to grab values from the provider.



The technique of passing data directly into a component is known as **dependency injection**.

Providers & Consumers

Provider

A **context provider** is used to wrap a component that has a child component that will need access to the Context object.

Consumer

A **context consumer** is used to access properties of a Context object.

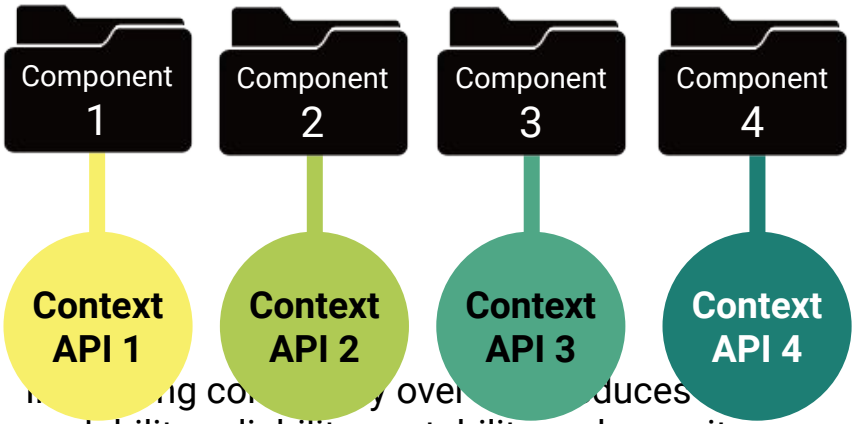
All consumers must be descendants of their respective providers.

Because we'll be using **hooks,** our hook will use the consumer behind the scenes.

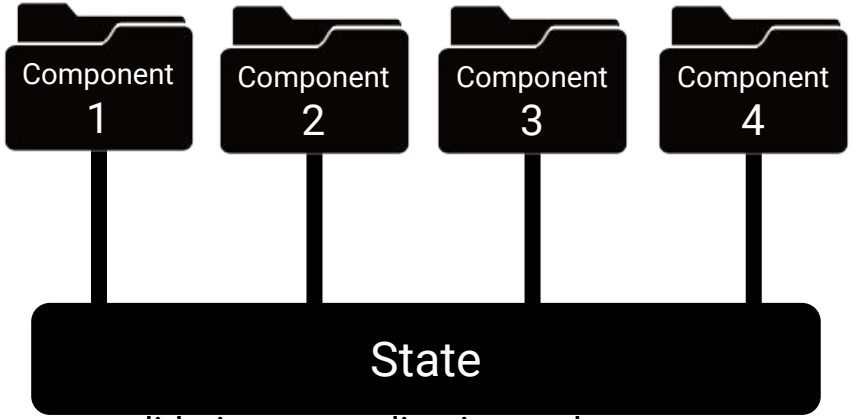
Tightly Coupled vs. Loosely Coupled

Using the Context API can be very helpful, but it comes at a cost. Adding stateful logic to a component creates a dependent relationship to another component. This is known as coupling.

When writing applications, it is best to keep components loosely coupled so that you can change one without affecting the other. This also helps ensure reusability.



scalability, reliability, portability and security.



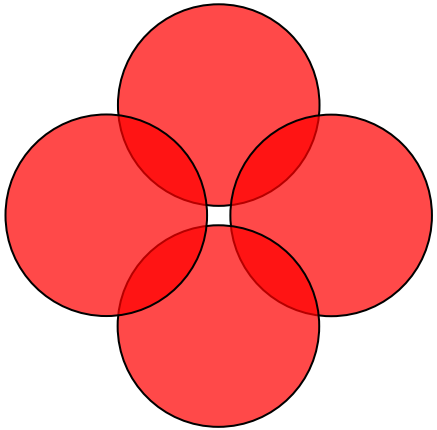
consolidation, normalization and governance.

Loosely Coupled Components

Using the Context API can be very helpful, but it comes at a cost. Adding stateful logic to a component creates a dependent relationship to another component. This is known as coupling.

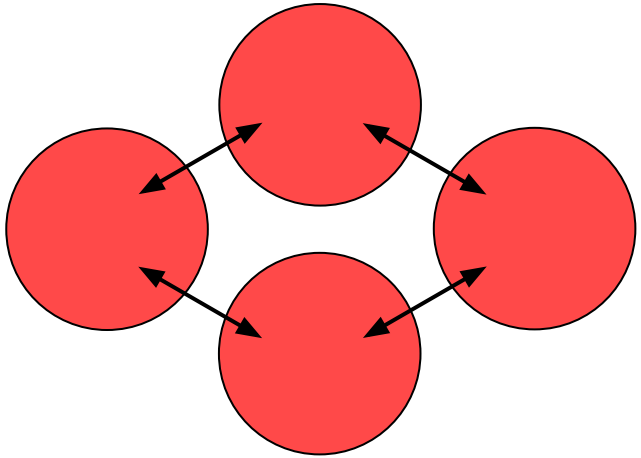
Tight Coupling

More Interdependency, coordination and information flow



Loose Coupling

When writing applications, it is best to keep components loosely coupled so that you can change one without affecting the other. This also helps ensure reusability.



Loosely Coupled Components

01

Tightly coupled components should be avoided when possible.

02

Coupling increases the dependencies necessary for a component. This means that one single change could have unintended consequences throughout the application. This also makes it harder to test and debug individual pieces.

03

Coupling reduces the overall reusability of components.

State Management

Try to be mindful of how often you're using the Context API.

If you find yourself using the Context API throughout several components, it may be time to consider using Redux or another state management library.

REMEMBER: There’s no perfect way to structure your application. It isn’t uncommon to be far along into development before you realize that you need to use the Context API. This is completely normal!