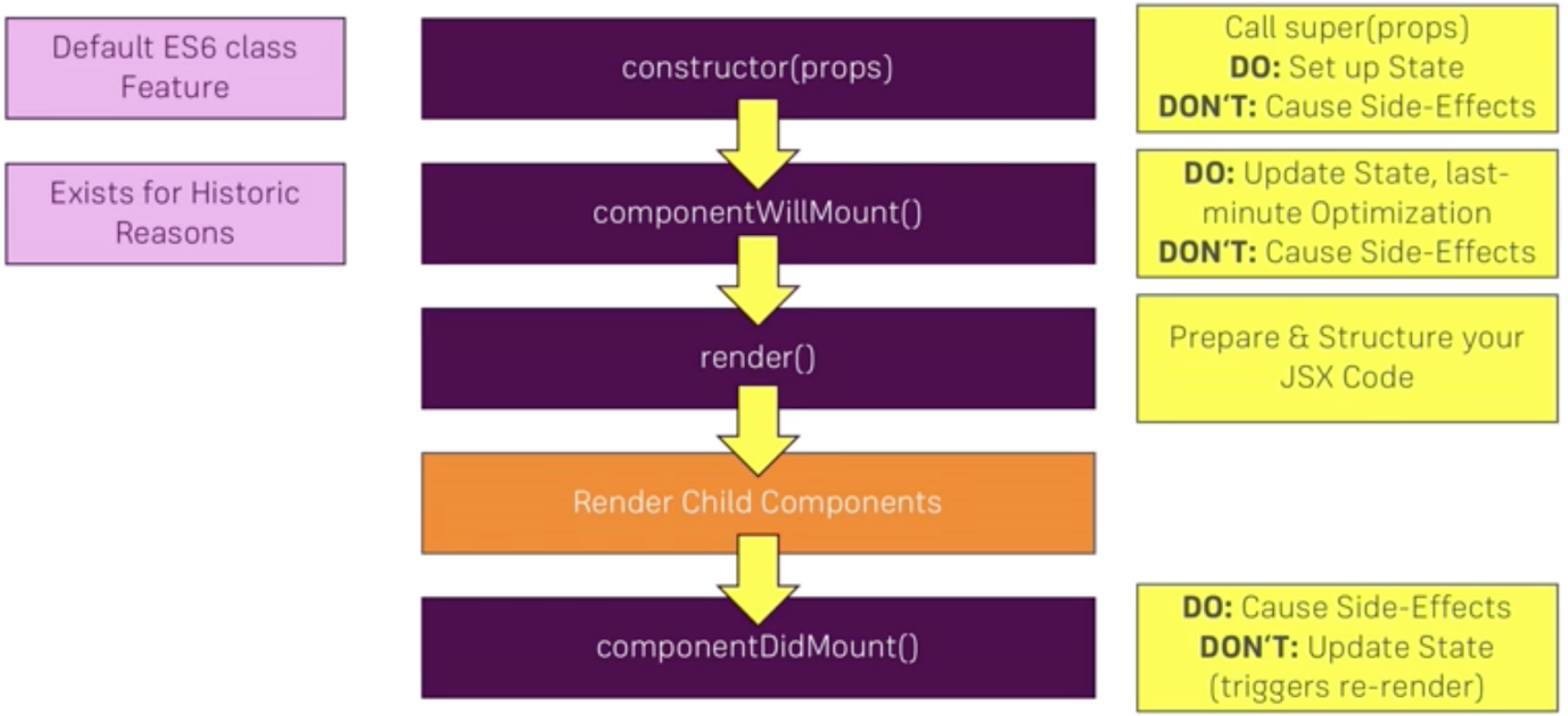
Advanced concepts

ROHAN RAJORE

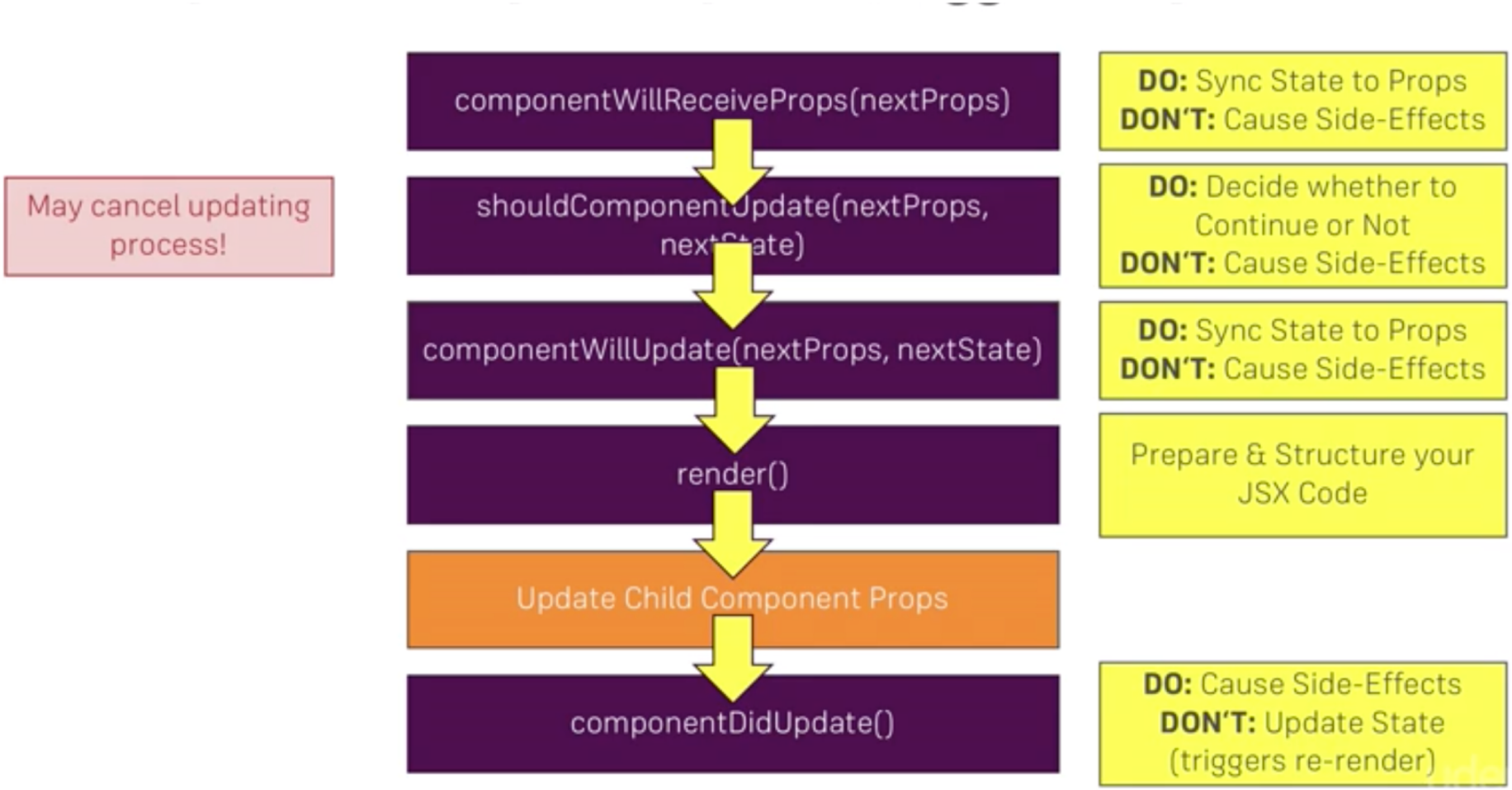
Component Lifecycle

ROHAN RAJORE

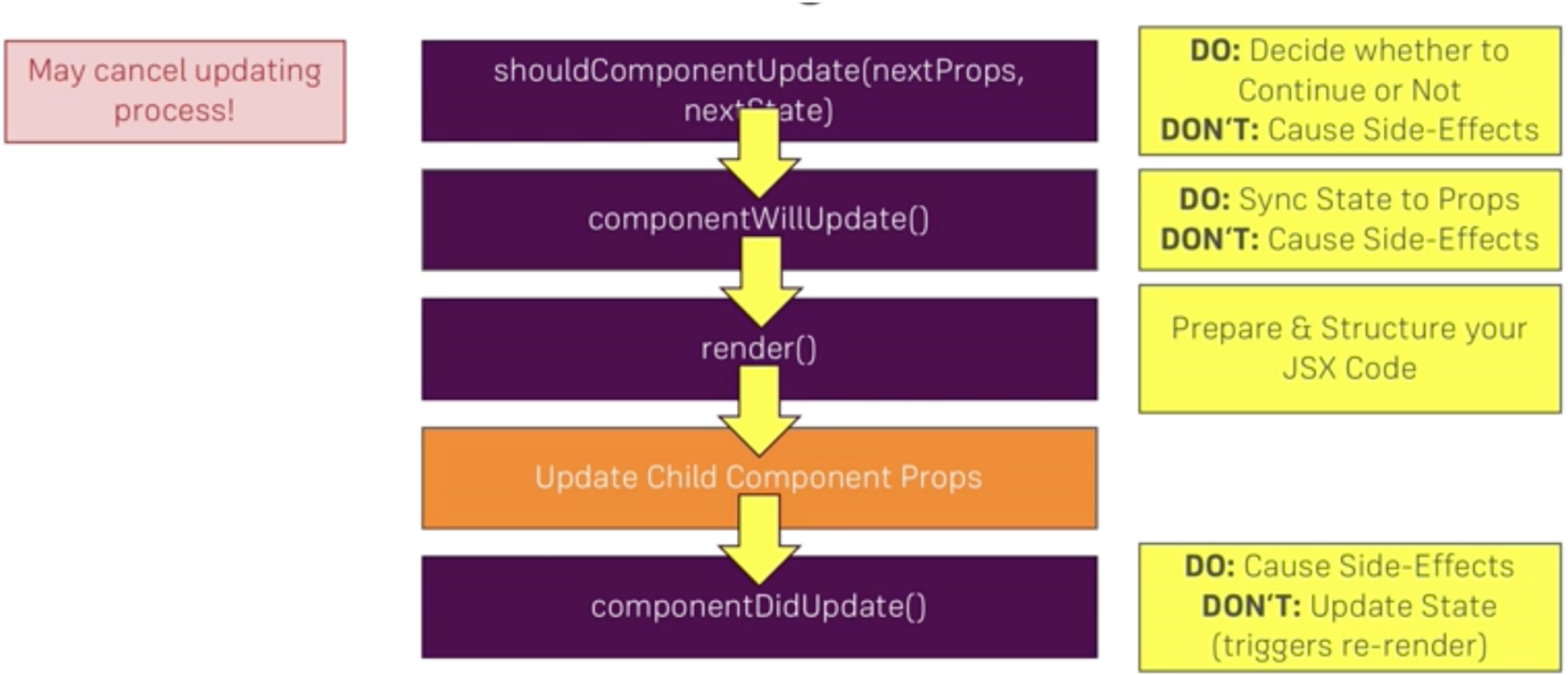
Component Lifecycle - creation



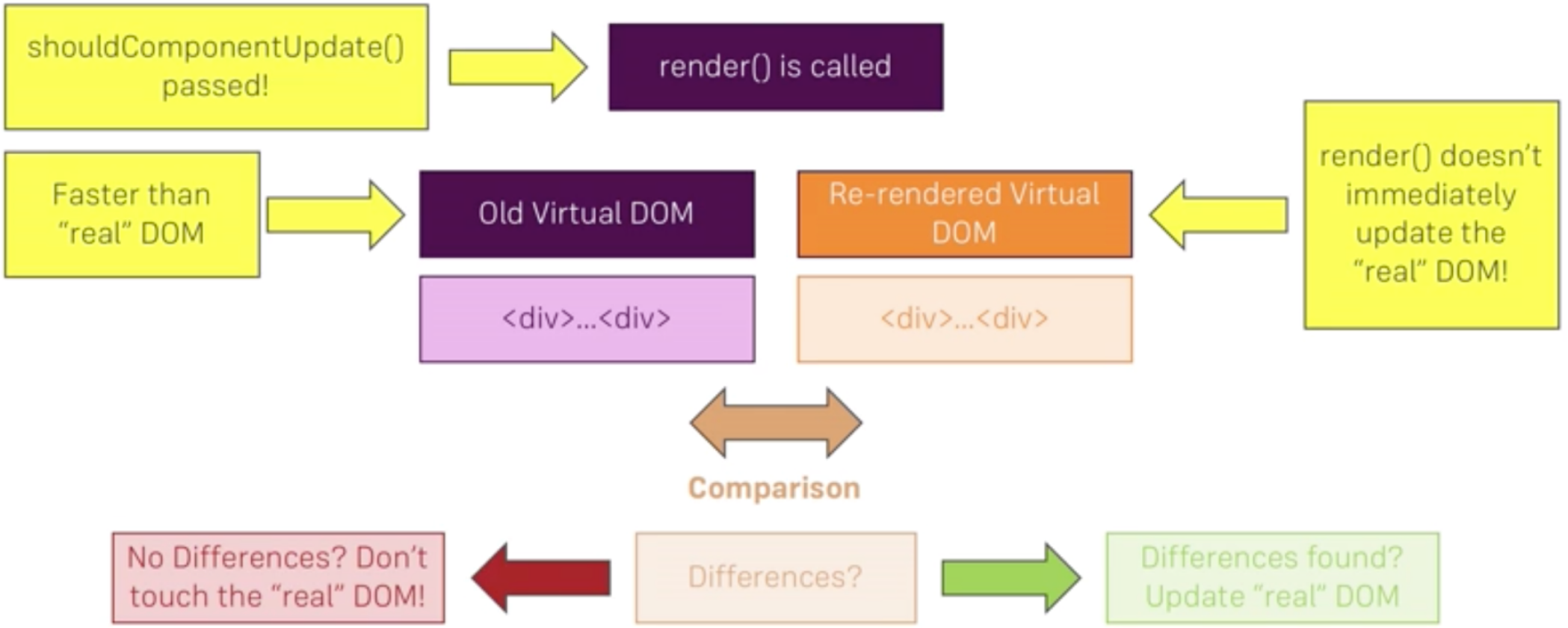
Component Lifecycle – Update (caused by parent)



Component Lifecycle – Update (triggered by internal change in state)



Reacts DOM update strategy



Refs & DOM

- Refs provide a way to access DOM nodes or React elements created in the render method.
- In the typical React dataflow, props are the only way that parent components interact with their children.
- To modify a child, you re-render it with new props.
- However, there are a few cases where you need to imperatively modify a child outside of the typical dataflow.
- The child to be modified could be an instance of a React component, or it could be a DOM element. For both of these cases, React provides an escape hatch.

When to use Refs?

- Managing focus, text selection, or media playback.
- Triggering imperative animations.
- Integrating with third-party DOM libraries.
- Avoid using refs for anything that can be done declaratively

Creating Ref

- Refs are created using React.createRef() and attached to React elements via the refattribute. Refs are commonly assigned to an instance property when a component is constructed so they can be referenced throughout the component.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }
  render() {
    return <div ref={this.myRef} />;
  }
}
```

Accessing Refs

- When a ref is passed to an element in render, a reference to the node becomes accessible at the `current` attribute of the ref.

```
const node = this.myRef.current;
```

Value of the Ref

- When the ref attribute is used on an HTML element, the ref created in the constructor with React.createRef() receives the underlying DOM element as its current property.
- When the ref attribute is used on a custom class component, the ref object receives the mounted instance of the component as its current.
- **You may not use the ref attribute on function components** because they don't have instances.

About Ref update

- An important thing to understand here is that refs manipulate **actual** DOM as opposed to **virtual** DOM which basically contradicts the main principle of how React works.

Context API

- Context provides a way to pass data through the component tree without having to pass props down manually at every level.
- In a typical React application, data is passed top-down (parent to child) via props, but this can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application.
- Context provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.

Context API

- sometimes the same data needs to be accessible by many components in the tree, and at different nesting levels.
- Context lets you “broadcast” such data, and changes to it, to all components below.
- Common examples where using context might be simpler than the alternatives include managing the current locale, theme, or a data cache.