

Codesandbox

- <https://codesandbox.io/>
- letcure 13: <https://codesandbox.io/s/lecture-13-rmvgk6>

Topics

- List and Keys
- Multiple Components
- Props vs State
- Refs
- Lifecycle Methods

Lists in React

`<p class="mark">list.jsx</p>`

- Use `map` to create a list of elements
- Each element needs a **unique** `key` prop among its siblings
 - “ `<p style="color:rgba(255,0,0,0.8);">Warning: Each child in a list should have a unique "key" prop.</p>` ”
- `key` is used by React to identify which items have changed, added, or removed
- Data from the model or locally generated based on the data

Multiple components

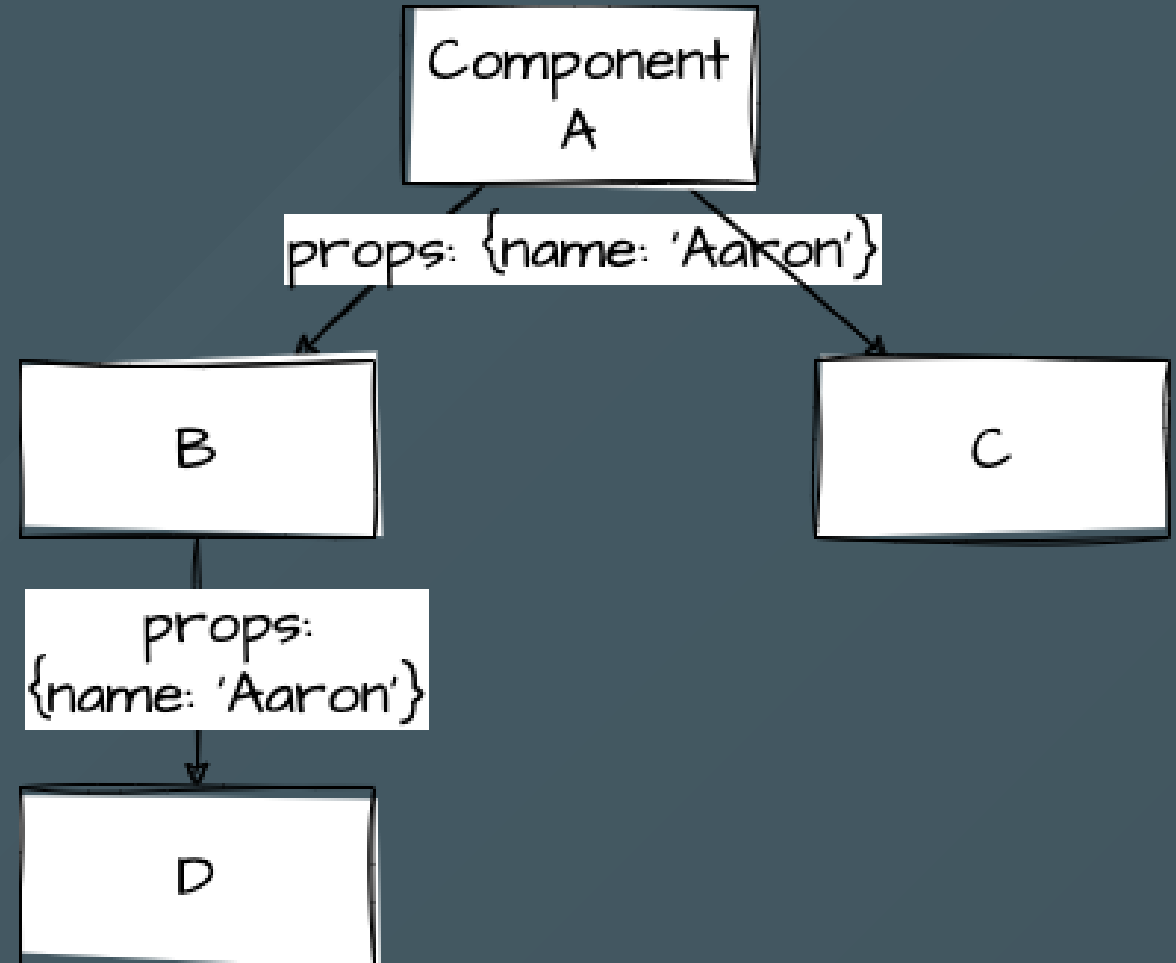
```
<p class="mark">multiple.jsx</p>
```

- presentational components vs. container components
- presentational components: only render UI
- container components: manage state

Props

`<p
class="mark">props.jsx</p>`

- `props` is **immutable**
- `props` is passed from parent to child, **uni-directional**
- `props` change would trigger re-render
- callback functions can be passed as `props`



Prop Types

- `prop-types` [package](#)

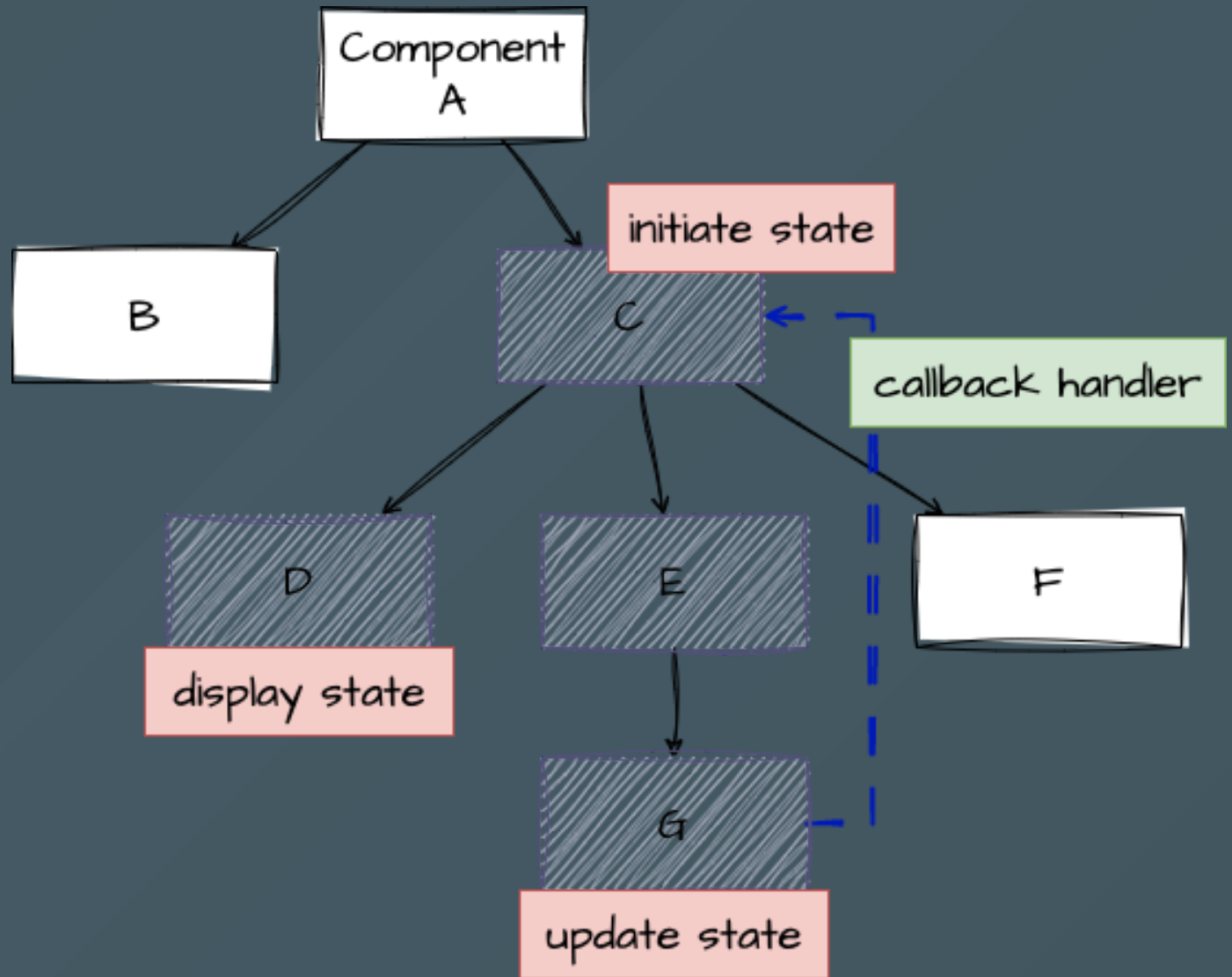
Props vs State

- `props` is immutable
- `state` is mutable with `setState`
- `props` is passed from parent to child, while `state` is local to the component
- `props` and `state` change would trigger re-render

Callbacks in Props

`<p class="mark">callback s.jsx</p>`

- Callback functions can be passed as props
- Callback functions can be used to pass data from child to parent



Controlled Components

- <https://reactjs.org/docs/forms.html#controlled-components>

Refs

```
<p class="mark">refs.jsx</p>
```

- `ref` is used to access DOM elements, or React components
- focus, text selection, media playback, triggering imperative animations, etc.
- how to use
 - `React.createRef()` create a ref
 - `ref={this.myRef}` assign the ref to a DOM element
 - `this.myRef.current` access the DOM element
- should not overuse `ref`, use `state` instead

forwardRef

- <https://reactjs.org/docs/forwarding-refs.html>

Component Lifecycle

- What is the lifecycle of a React component?
- What are the lifecycle methods?
- What are the use cases of lifecycle methods?

Lifecycle Methods

<p class="mark">lifecycles.jsx</p> <div class="small">

- Mounting: when a component is being inserted into the DOM
 - `constructor()`
 - `componentWillMount()`
 - `render()`
 - `componentDidMount()`
 - `render()`
- Updating: when a component is being re-rendered by changes in props or state, or `forceUpdate()`
 - `componentDidUpdate()` / `shouldComponentUpdate()`

Lifecycle in Components Tree

`<p class="mark">lifecycleChild.jsx</p>`

- `componentDidMount()` is called after all children are mounted
- children's updating lifecycle methods are called before parent's
- execution is recursive in depth-first order

Example: Clock

```
<p class="mark">clock.jsx</p>
```

How to start with React?

1. According to the design and requirements, write down the HTML structure
2. Break down the HTML structure into React components
3. Confirm the data flow (props and state) between components
4. Display state - connect the components with data flow
5. Modify state - add event handlers to specific components / lifecycle methods
6. Beautify the UI with CSS
7. Refactor the code from step 2 to 6