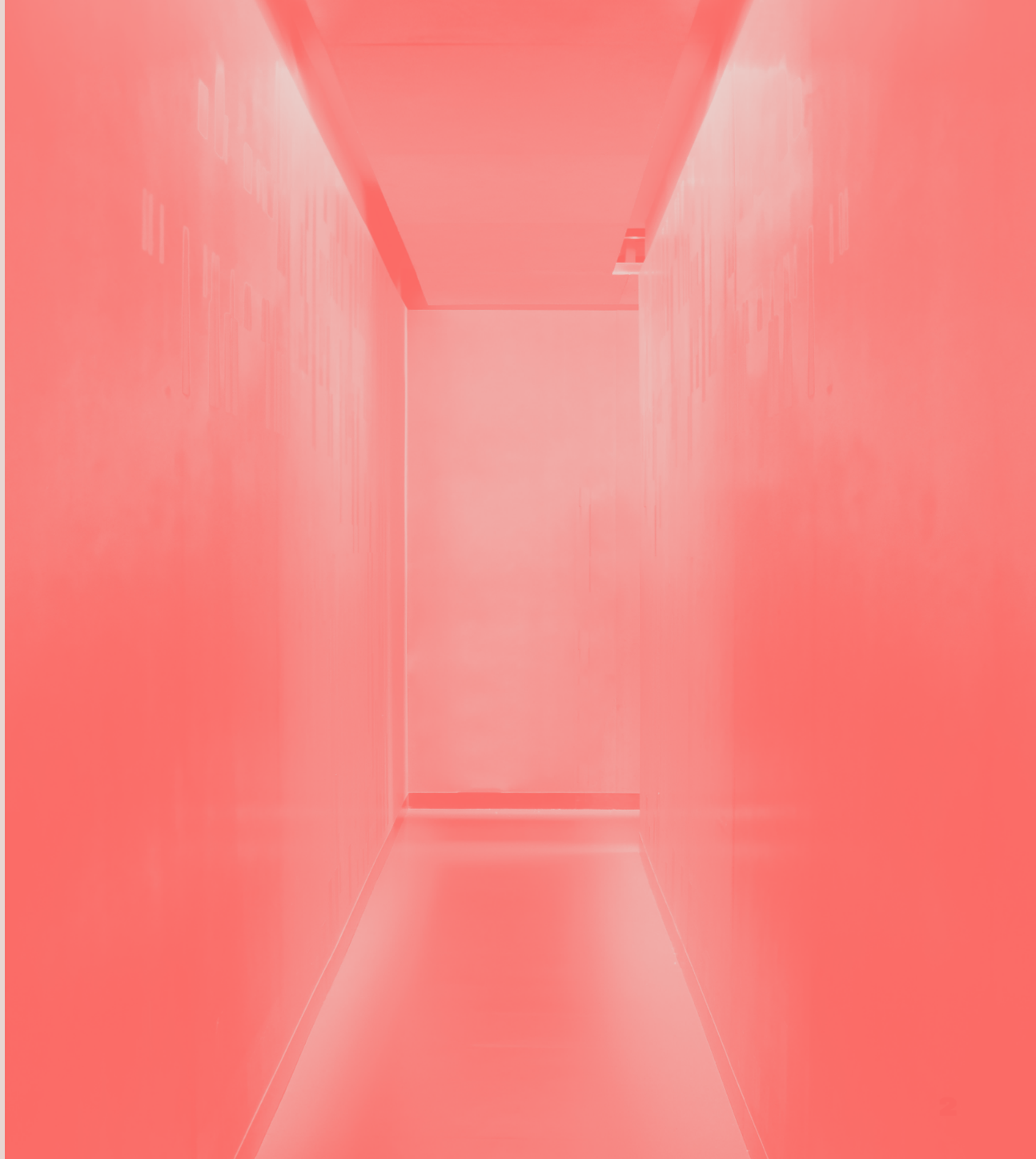
REACT FOR BE DEVS

REACT

REACT

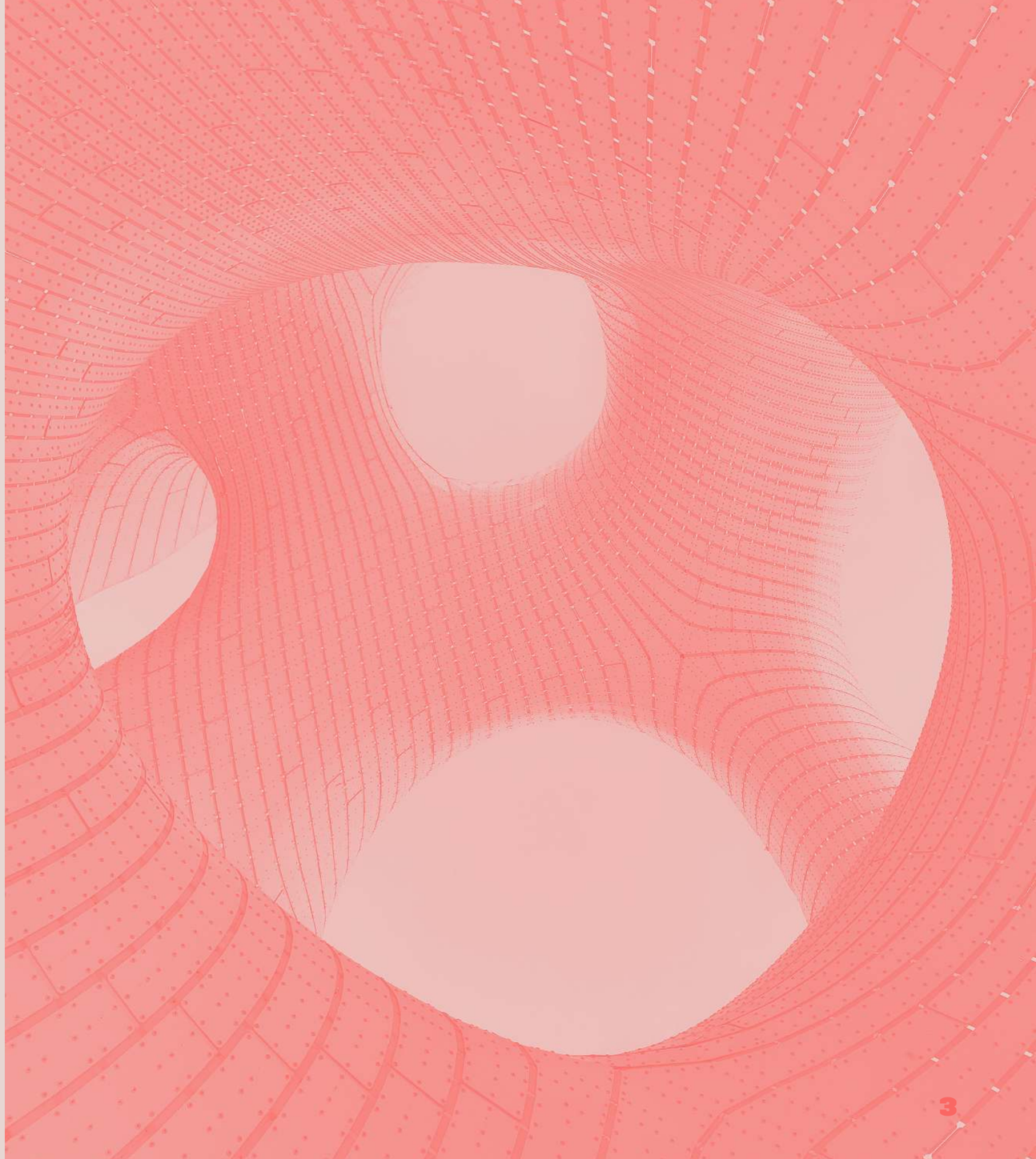
- Component based library to build composable UIs
- OpenSourced in 2013
- Implemented and Maintained by Facebook
- Learn once, write anywhere
  - React-Native
  - React-Native-Desktop
  - React-Native-Windows



COMPONENTS

“Components let you split the UI into independent, reusable pieces.[^1]”

- Main Building block of a React App
  - Describe the look and feel of one section in the UI



REACT COMPONENTS

```
const Button = () => {
  return (
    <button type='button'>
      A button
    </button>
  )
}

// Usage
React.renderComponent(<Button />, document.body)
```

REACT CLASS COMPONENTS

» Alternative syntax for components

```
class Button extends React.Component {
  render() {
    return (
      <button type='button'>
        A button
      </button>
    )
  }
}

// Usage
React.renderComponent(<Button />, document.body)
```

JSX

» JavaScript XML

  » extension to write XML in JS

» Allows to combine data preparation with render logic

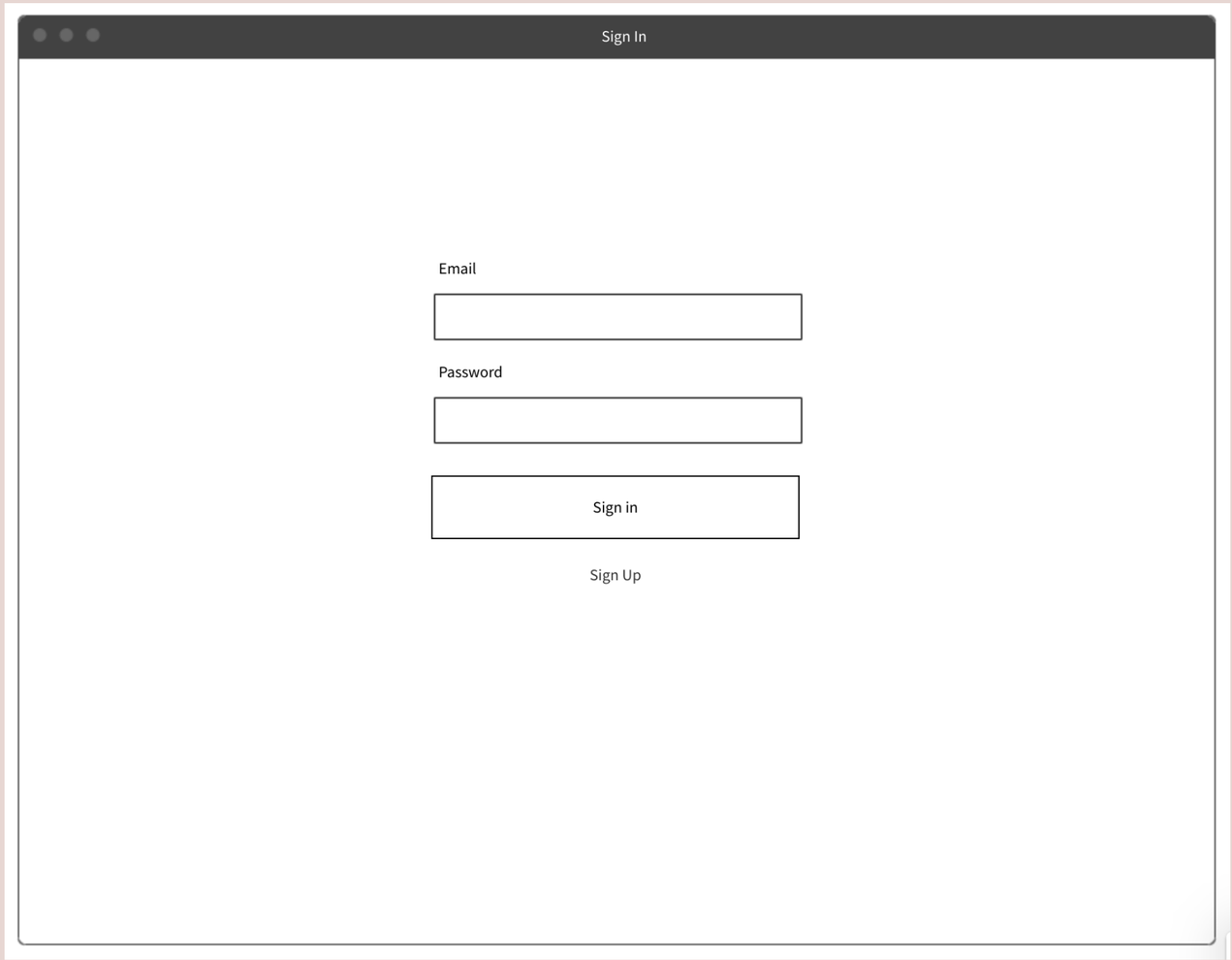
```
const Button = () => {
  return (
    <button type='button'>
      A button
    </button>
  )
}
```

REACT WITHOUT JSX

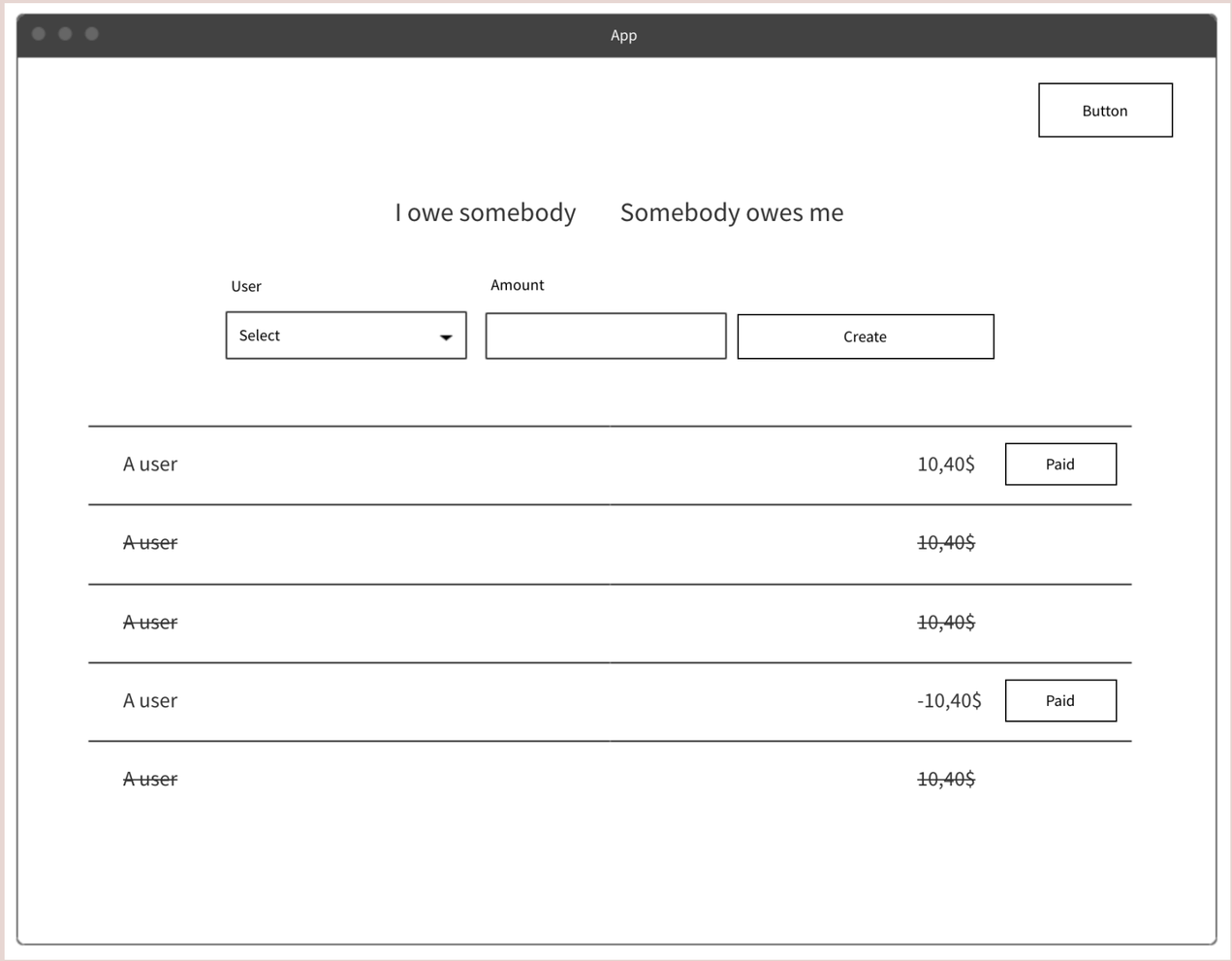
» React can be used without JSX

```
const Button = () => {
  return React.createElement(
    'button',
    { type: 'button' },
    'A button'
  )
}
```

WHICH COMPONENTS DO YOU SEE



WHICH COMPONENTS DO YOU SEE



BUILDING THE FIRST REACT COMPONENT

EMBEDDING EXPRESSIONS

```
const CurrentTime = () => {
  return (
    <h1>
      {(new Date()).toLocaleDateString()}
    </h1>
  )
}
```

EMBEDDING EXPRESSIONS

```
const FagoMenu = () => {
  return (
    <a href={`/menu/${(new Date()).toLocaleDateString()}`}>
      Go to todays menu
    </a>
  )
}
```

CONDITIONAL RENDERING

```
const CurrentTime = () => {
  // ...
  return (
    <h1>
      {isToday
        ? 'Today'
        : 'Not Today'}
    </h1>
  )
}
```

CONDITIONAL RENDERING

```
const CurrentTime = () => {
  // ...
  return (
    <h1>
      {isToday && 'Today'}
      {!isToday && 'Not today'}
    </h1>
  )
}
```

LOOP OVER ARRAYS

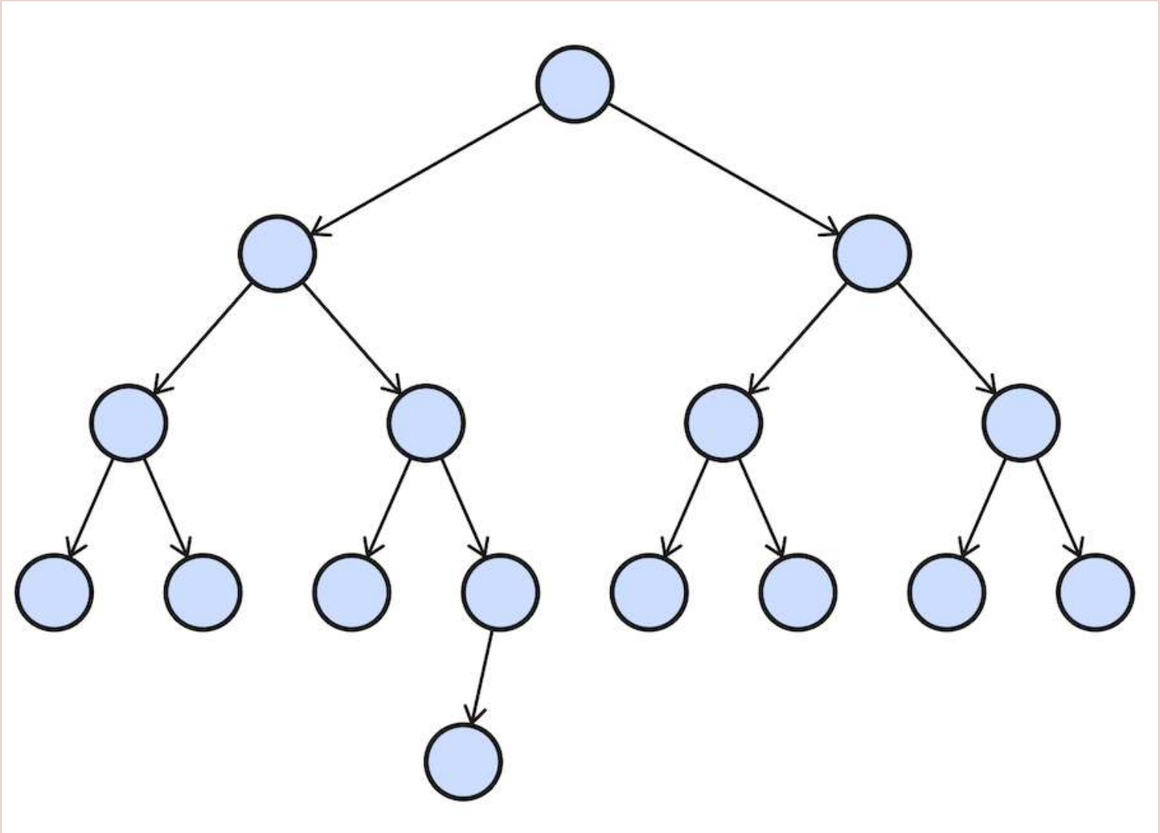
```
const UserList = ({ users }) => {
  return (
    <ul>
      {users.map((user) => {
        return (<li key={user.id}>{user.name}</li>)
      })}
    </ul>
  )
}
```

KEY PROPERTY IN LOOPS

- Is required when interating over lists
- Helps react to decide if an element needs to be rerendered
- Video explanation
- Detailed explanation

COMPONENT COMPOSITION

» Components can be nested and composed together



REACT PROPS

- Possibility to customize components
  - Can be seen as component configuration
- Props are passed to the component
  - A component at a lower level of the tree can't modify given props directly

REACT PROPS

```
const Button = ({ children, disabled = false }) => {
  //               ^^^^^^^^^^^^^^^^^^^^^^^^^^
  // props which are passed to the component

  return (
    <button disabled={disabled} className='button'>
      {children}
    </button>
  )
}

const usage = <Button disabled>A button</Button>
// 1)                 ^^^^^^^^
// 2)                          ^^^^^^^^^^
// 1) shortcut for disabled={true}
// 2) child components/nodes passed to a component
```

STATE IN REACT

- What we've seen so far:
  - Components can render chunks of UI
  - Components can be nested

STATE IN REACT

“How can we interact with components?”

STATE IN REACT

“The State of a component is an object that holds some information that may change over the lifetime of the component [5]”

[5] geeksforgeeks.com

REACT STATE (WITHOUT HOOKS)

```
class ToggleButton extends React.Component {
  state = { backgroundColor: 'red' };
  // define a default value for background color

  toggleBackgroundColor = () => {
    const nextBackgroundColor = backgroundColor === 'red' ? 'blue' : 'red'
    this.setState({ backgroundColor: nextBackgroundColor })
    //   ^^^^^^^^^
    // setState calls render method with updated state
  }
  render() {
    return (
       <button
        onClick={() => this.toggleBackgroundColor() }
        style={{ backgroundColor: this.state.backgroundColor }}
      >
        {children}
      </button>
    );
  }
}
```

REACT STATE (WITH HOOKS)

» Alternative syntax with hooks

```
const ToggleButton = () => {
  const [backgroundColor, setBackground] = useState('red')
  // 1)                                    ^^^^^^^^
  // 2) ^^^^^^^^^^^^^^^^
  // 3)                   ^^^^^^^^^^^^^^
  // 1) define a state with a default value "red"
  // 2) the current value of the state
  // 3) function to set the state to something else

  return (
    <button
      onClick={() => setBackground(backgroundColor === 'red' ? 'blue' : 'red')}
      style={{ backgroundColor }}
    >
      {children}
    </button>
  )
}
```

REACT HOOKS

“Hooks allow you to reuse stateful logic without changing your component hierarchy. React Docs”

REACT HOOKS

- Introduced recently to reduce boilerplate
- Makes it possible to use state in functional components
  - Previously one had to convert between functional/class components when state introduced
- hooks are prefixed with use
- Can't be called inside loops, conditions or nested

USESTATE

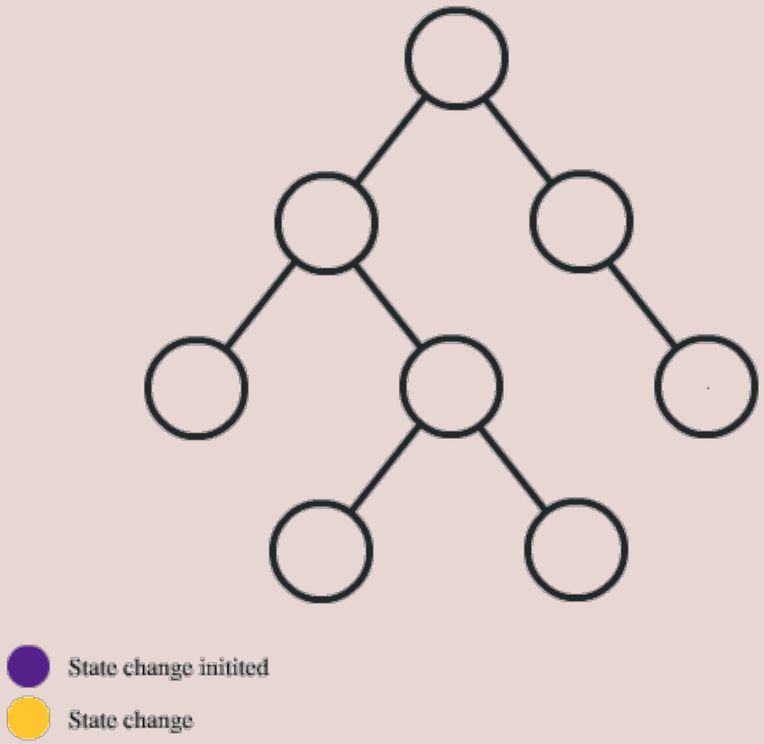
```
const App = () => {
  const [count, setCount] = useState(0)
  const handleIncrement = () => setCount(count + 1)

  return (
    <div>
      <div>{count}</div>
      <button onClick={handleIncrement}>Increment by 1</button>
    </div>
  )
}
```

UNIDIRECTIONAL DATAFLOW

- Props only flow from parent to children
- Parent is responsible to update data
  - might provide callbacks to do so
- set state rerenders all children of component

UNIDIRECTIONAL DATAFLOW



"Source"

KATA

- Build an online integer calculator in react [^1]
- Implement the following arithmetic operations
  - addition
  - subtraction
  - multiplication
- Use JS BigInt datatype
- Data input should be possible via: