O'REILLY®

From Local State to Global Control

September 2023

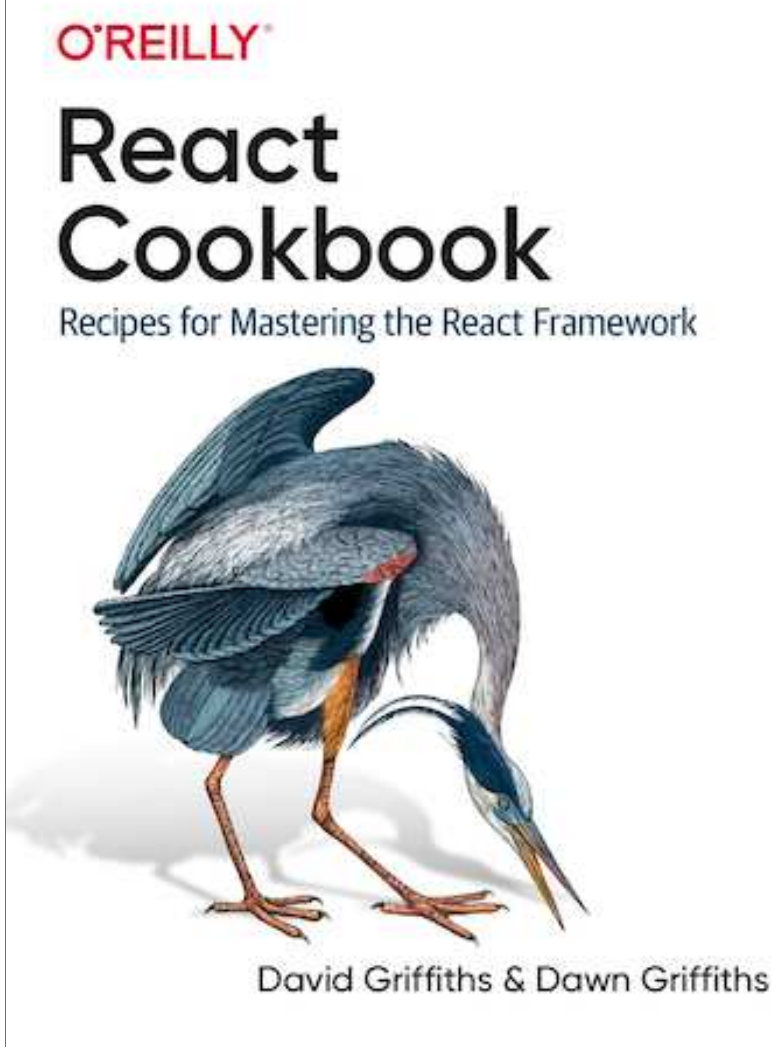
Introduction

About me

- Author

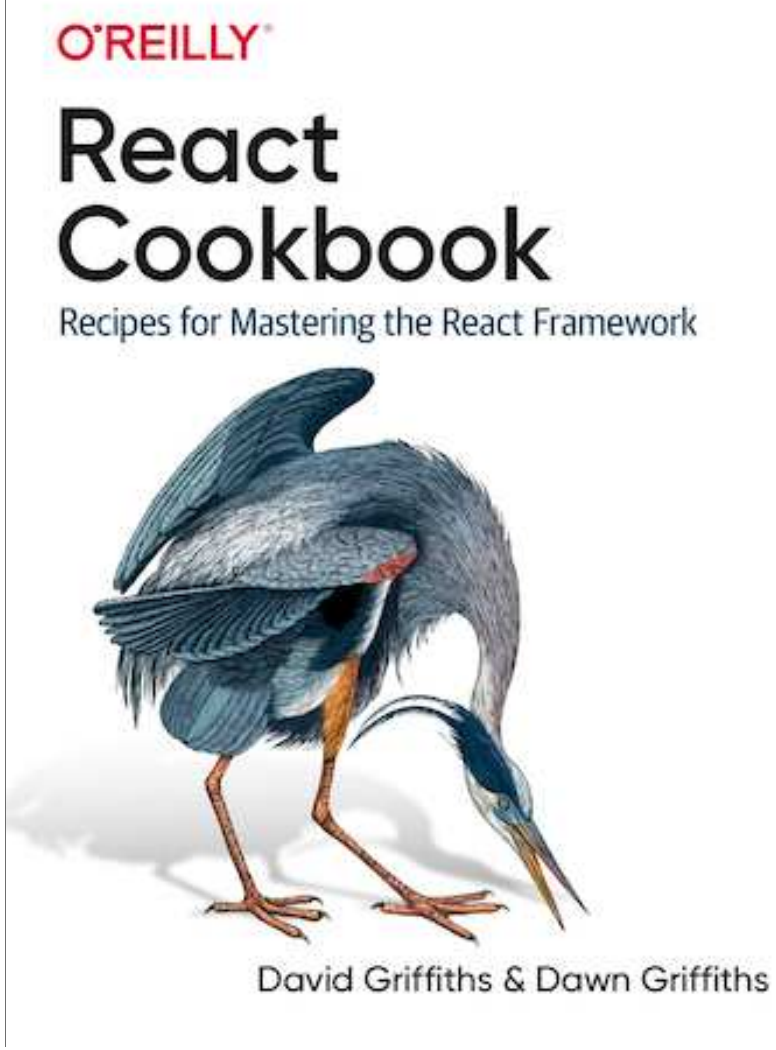
About me

- Author



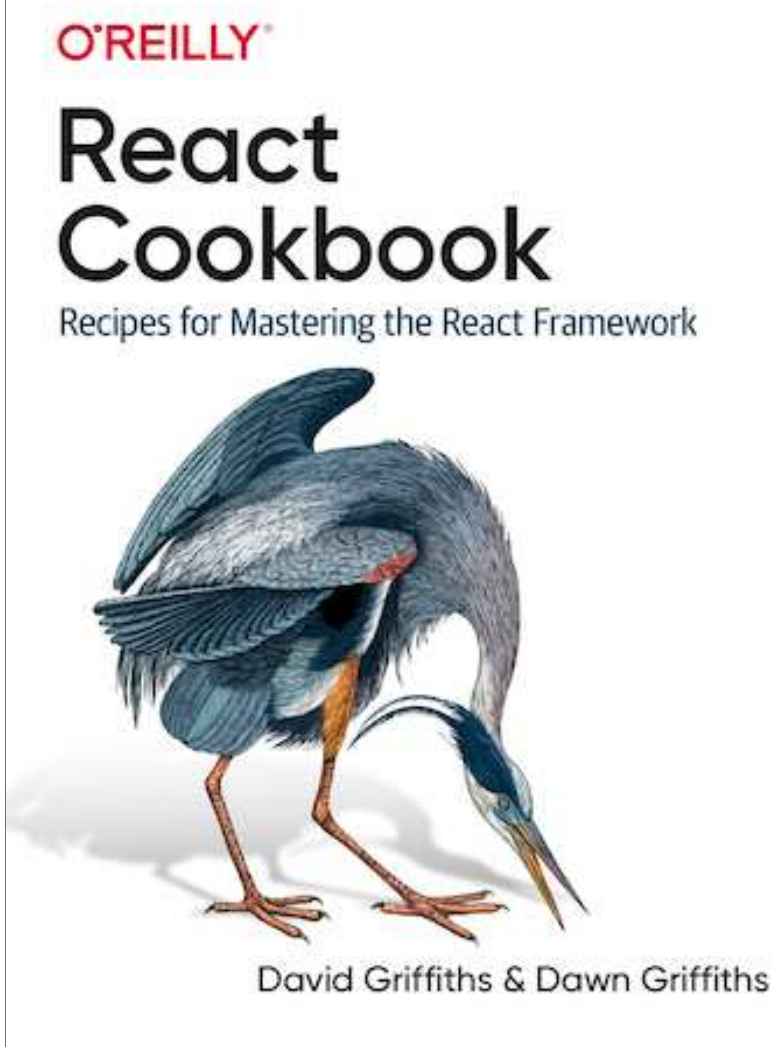
About me

- Author
- https://www.herescreen.com



About me

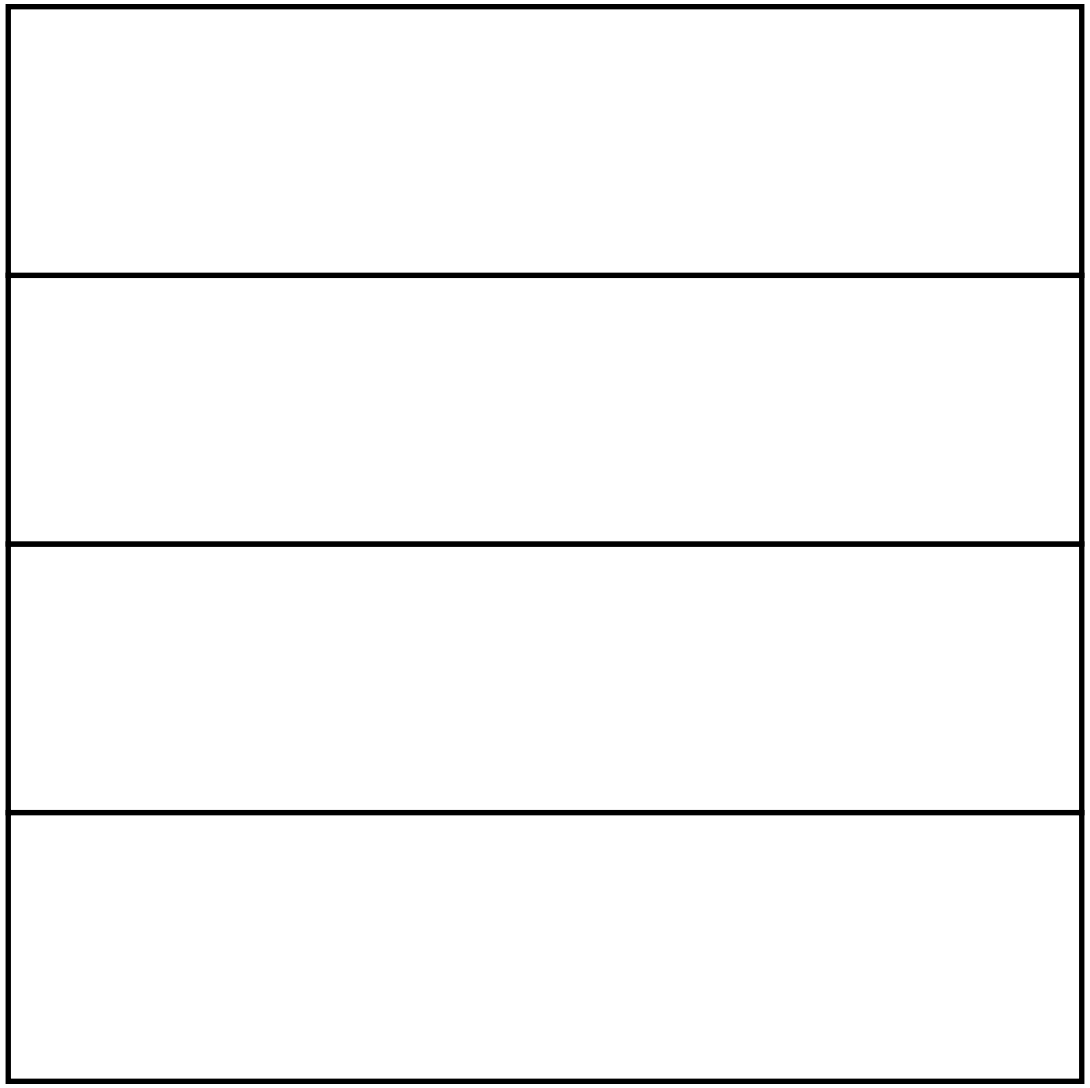
- Author
- https://www.herescreen.com
- https://linktr.ee/dogriffiths



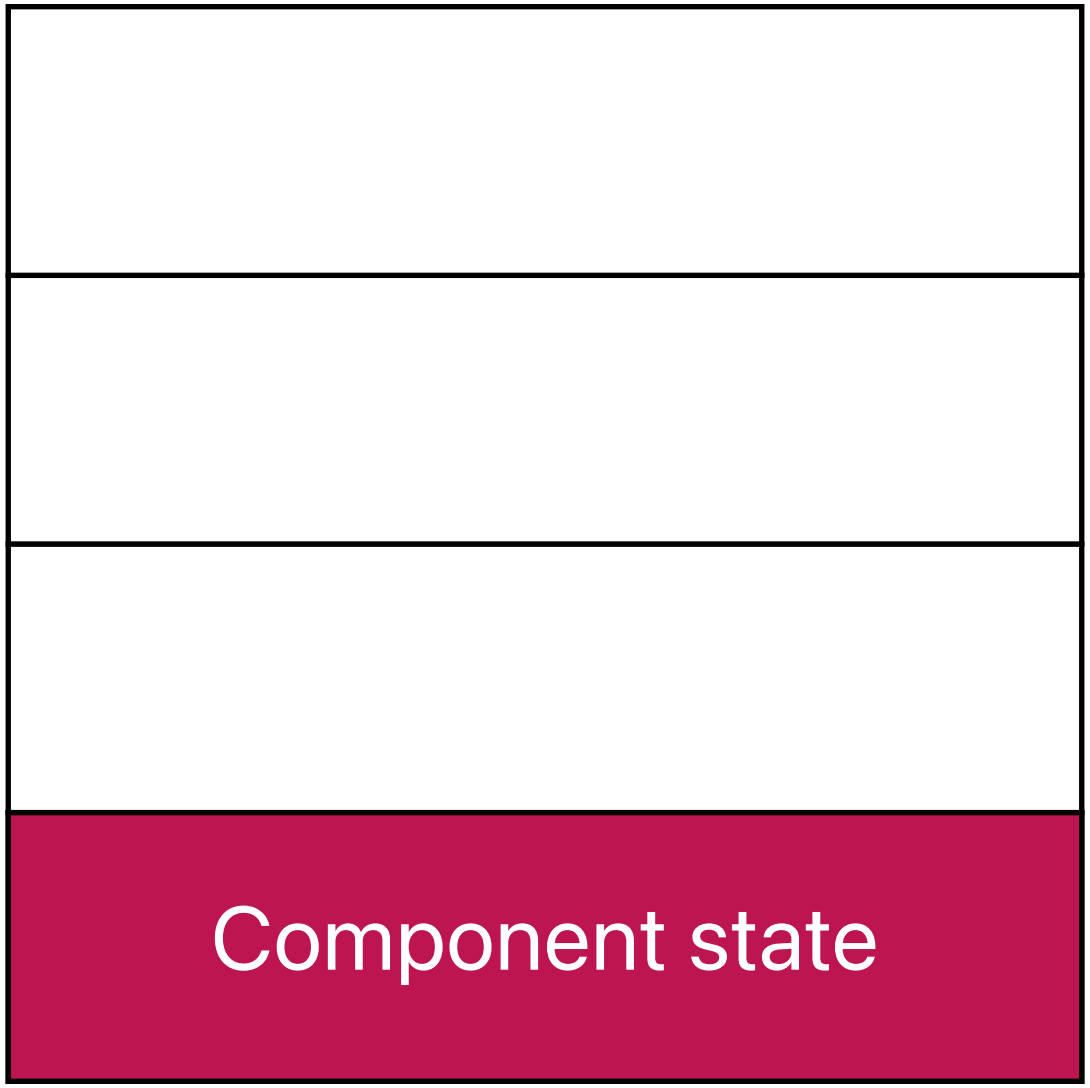
Why write the book?

Component state

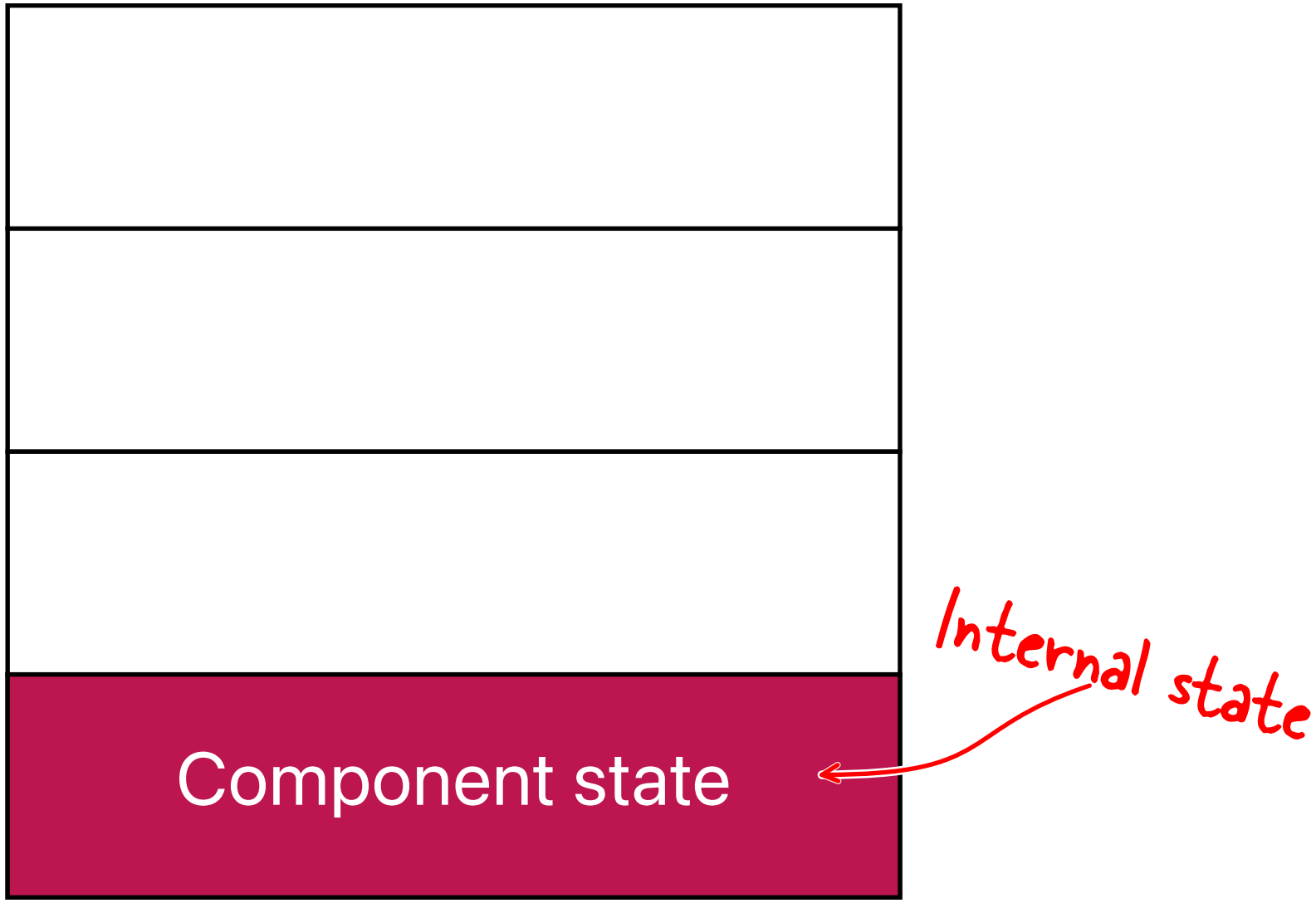
Different levels of state



Different levels of state



Different levels of state



useState

```
const [name, setName] = useState("")
return <>
</>
```

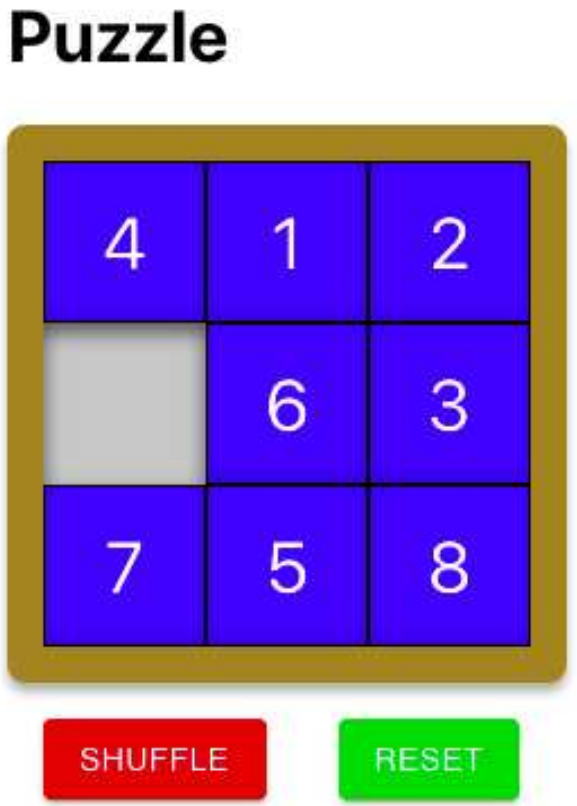
useState

```
const [name, setName] = useState("")
return <>
  Name: {name}
</>
```

useState

```
const [name, setName] = useState("")
return <>
  Name: {name}
  <input value={name}
         onChange={(evt) => setName(evt.target.value)}
      />
</>
```

Complex components



Developer response

Managing complex component state (recipe 3.1)

Managing complex component state (recipe 3.1)

Managing complex component state (recipe 3.1)

- Some components might have many `useState()` calls

Managing complex component state (recipe 3.1)

- Some components might have many `useState()` calls
- There might be complex code that then uses those values

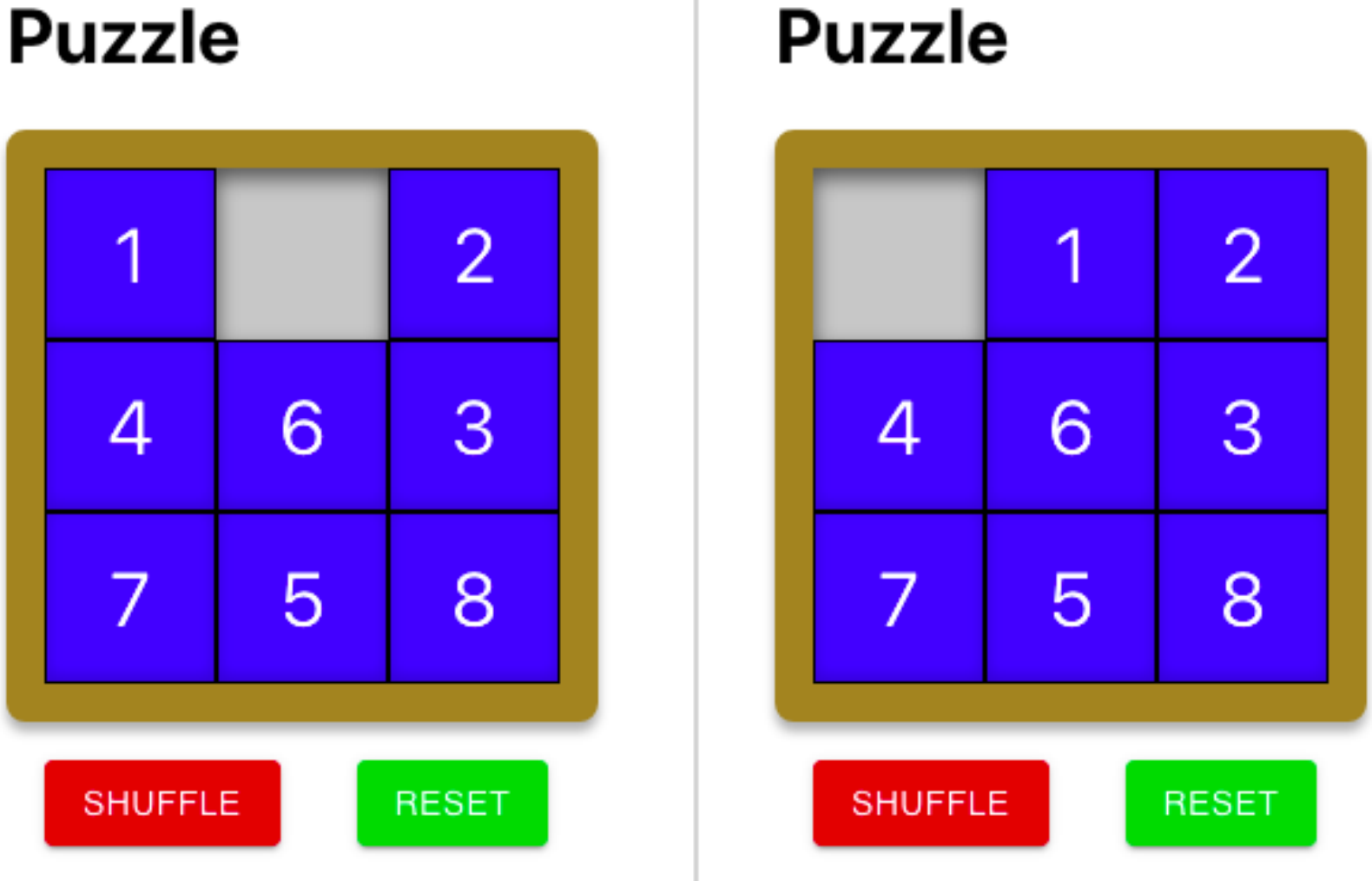
Managing complex component state (recipe 3.1)

- Some components might have many `useState()` calls
- There might be complex code that then uses those values
- You can extract from the complex state code into a **reducer**

Create a reducer: reducer.js

```
function reducer(state, action) {
}

export default reducer
```

Moving a tile



Create a reducer: reducer.js

```
function reducer(state, action) {
}

export default reducer
```

Example state and action

```
state = {
  items: ['1', null, '2', '4', '6', '3', '7', '5', '8'],
}

action = {
    type: 'move',
    payload: 0
}
```

New state object returns

```
return {
  items: [null, '1', '2', '4', '6', '3', '7', '5', '8'],
}
```

Create a reducer: reducer.js

```
function reducer(state, action) {
}

export default reducer
```

Create a reducer: reducer.js

```
function reducer(state, action) {
  switch (action.type) {
  }
}

export default reducer
```

Create a reducer: reducer.js

```
function reducer(state, action) {
  switch (action.type) {
    ....
    case 'shuffle': {
      let newState = { ...state }
      do {
        for (let i = 0; i < 300; i++) {
          newState = reducer(
            { ...newState },
            {
              type: 'move',
              payload: Math.floor(Math.random() * 9),
            }
          )
        }
      } while (newState.complete)
      return newState
    }
    default: {
      throw new Error('Unknown action: ' + action.type)
    }
  }
}

export default reducer
```

Using a reducer

```
import reducer from './reducer'
```

Using a reducer

```
import reducer from './reducer'
...
const [state, dispatch] = useReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
```

Using a reducer

```
import reducer from './reducer'
...
const [state, dispatch] = useReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
...
return <>
  <p>There are {state.items.length} items</p>
</>
```

Using a reducer

```
import reducer from './reducer'
...
const [state, dispatch] = useReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
...
return <>
  <p>There are {state.items.length} items</p>
  <button onClick={() => dispatch({ type: 'shuffle' })}>
     Shuffle
  </button>
</>
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
        expect(state.complete).toBe(true)
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
        expect(state.complete).toBe(true)
        state = reducer(state, { type: 'move', payload: 6 })
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
        expect(state.complete).toBe(true)
        state = reducer(state, { type: 'move', payload: 6 })
        expect(state.complete).toBe(false)
    })
})
```

Testing a reducer

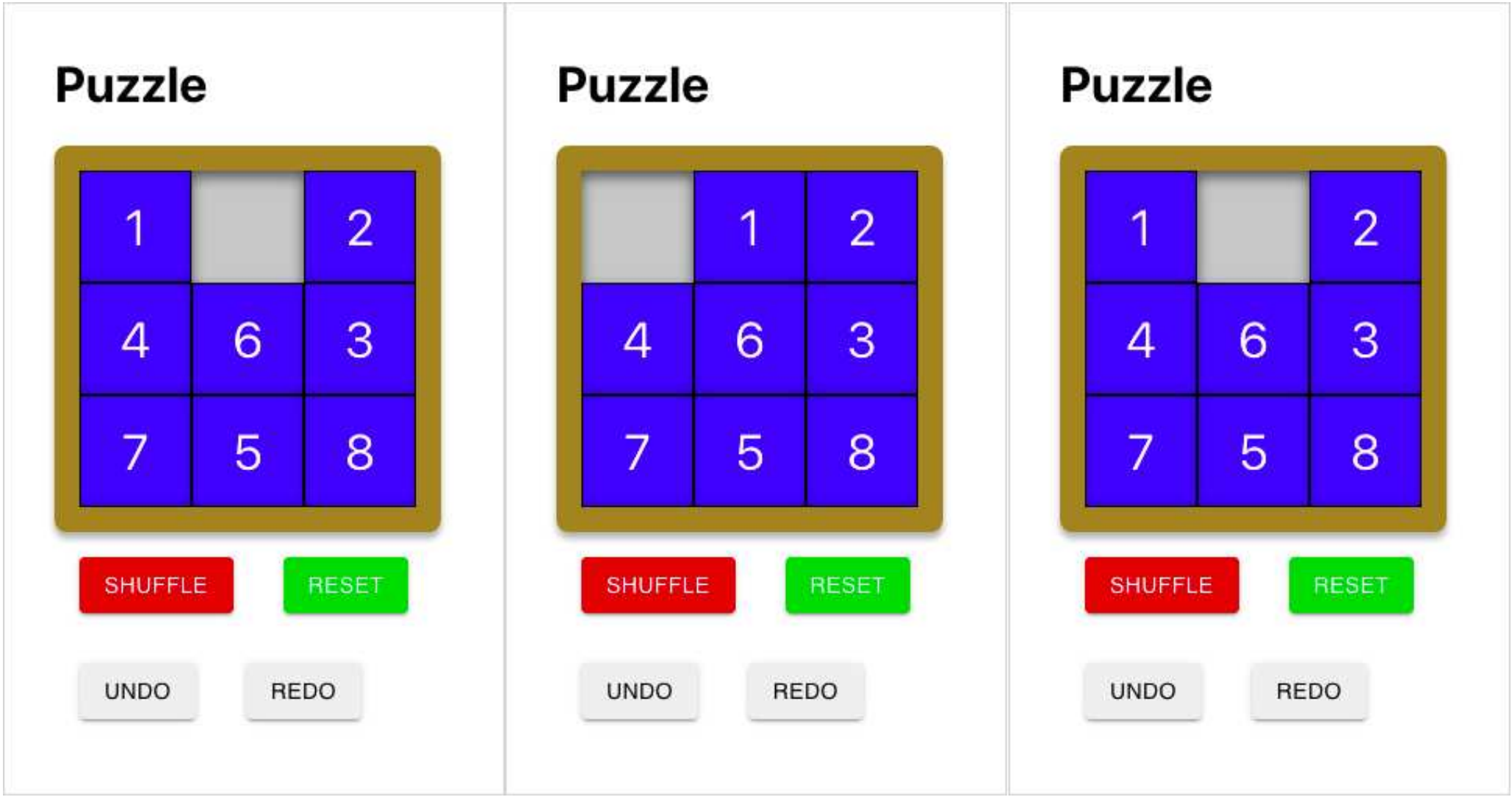
```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
        expect(state.complete).toBe(true)
        state = reducer(state, { type: 'move', payload: 6 })
        expect(state.complete).toBe(false)
    })
    it('should be able to move 1 down if gap below', () => {
    })
})
```

Testing a reducer

```
import reducer from './reducer'

describe('reducer', () => {
    it('should say when it is complete', () => {
        let state = {
            items: ['1', '2', '3', '4', '5', '6', '7', null, '8'],
        }
        state = reducer(state, { type: 'move', payload: 8 })
        expect(state.complete).toBe(true)
        state = reducer(state, { type: 'move', payload: 6 })
        expect(state.complete).toBe(false)
    })
    it('should be able to move 1 down if gap below', () => {
        let state = {
            items: ['1', '2', '3', null, '5', '6', '7', '8', '4'],
        }
        state = reducer(state, { type: 'move', payload: 0 })
        expect(state.items).toEqual([null, '2', '3', '1', '5',
            '6', '7', '8', '4',
        ])
    })
})
```

Undo-ing a reducer (recipe 3.2)

Undo-ing a reducer (recipe 3.2)



Undo-ing a reducer (recipe 3.2)

```
import reducer from './reducer'
import { useReducer } from 'react'

import './Puzzle.css'

const Puzzle = () => {
  const [state, dispatch] = useReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
```

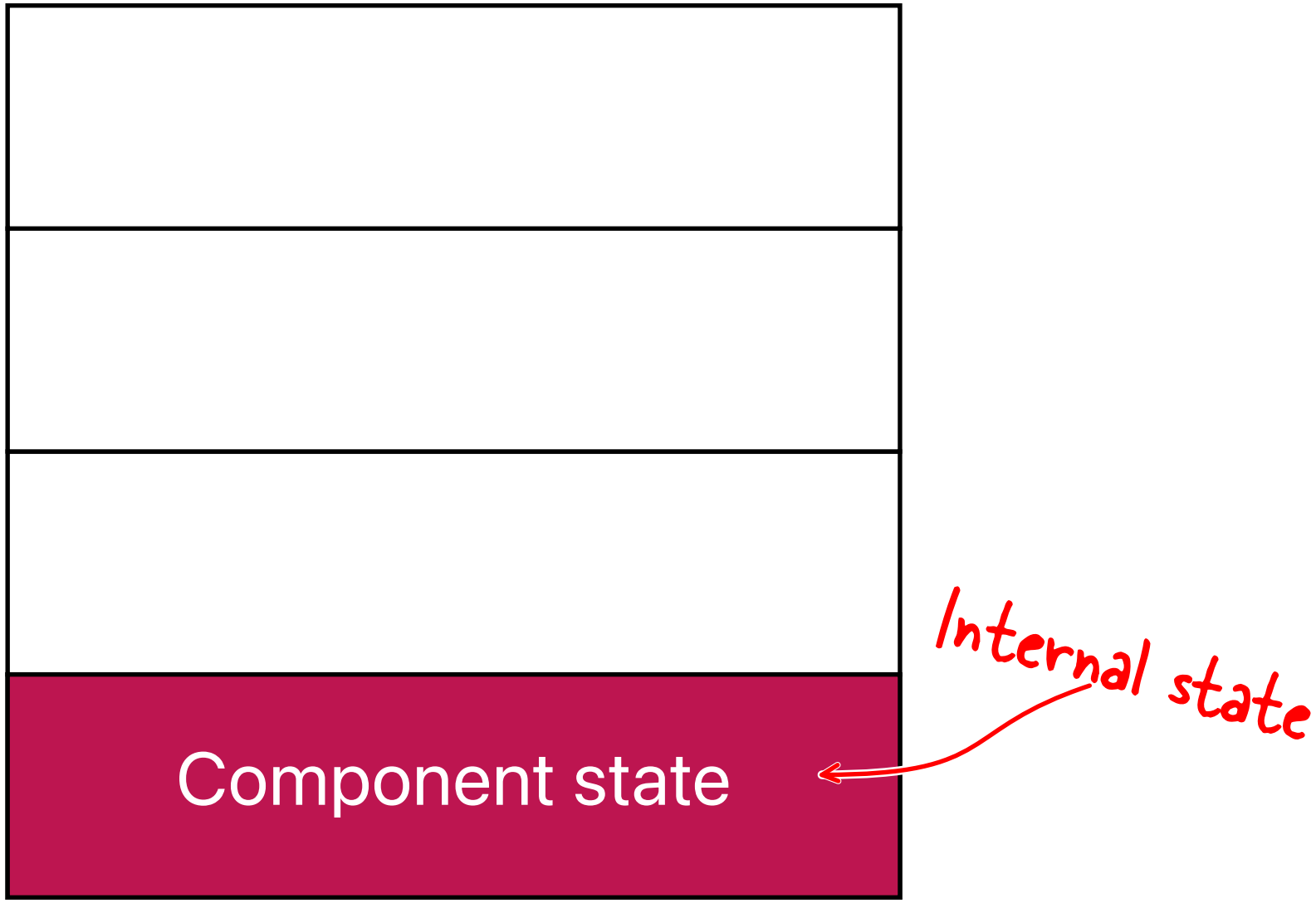
Undo-ing a reducer (recipe 3.2)

```
import reducer from './reducer'
import useUndoReducer from './useUndoReducer'

const Puzzle = () => {
  const [state, dispatch] = useUndoReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
....
```

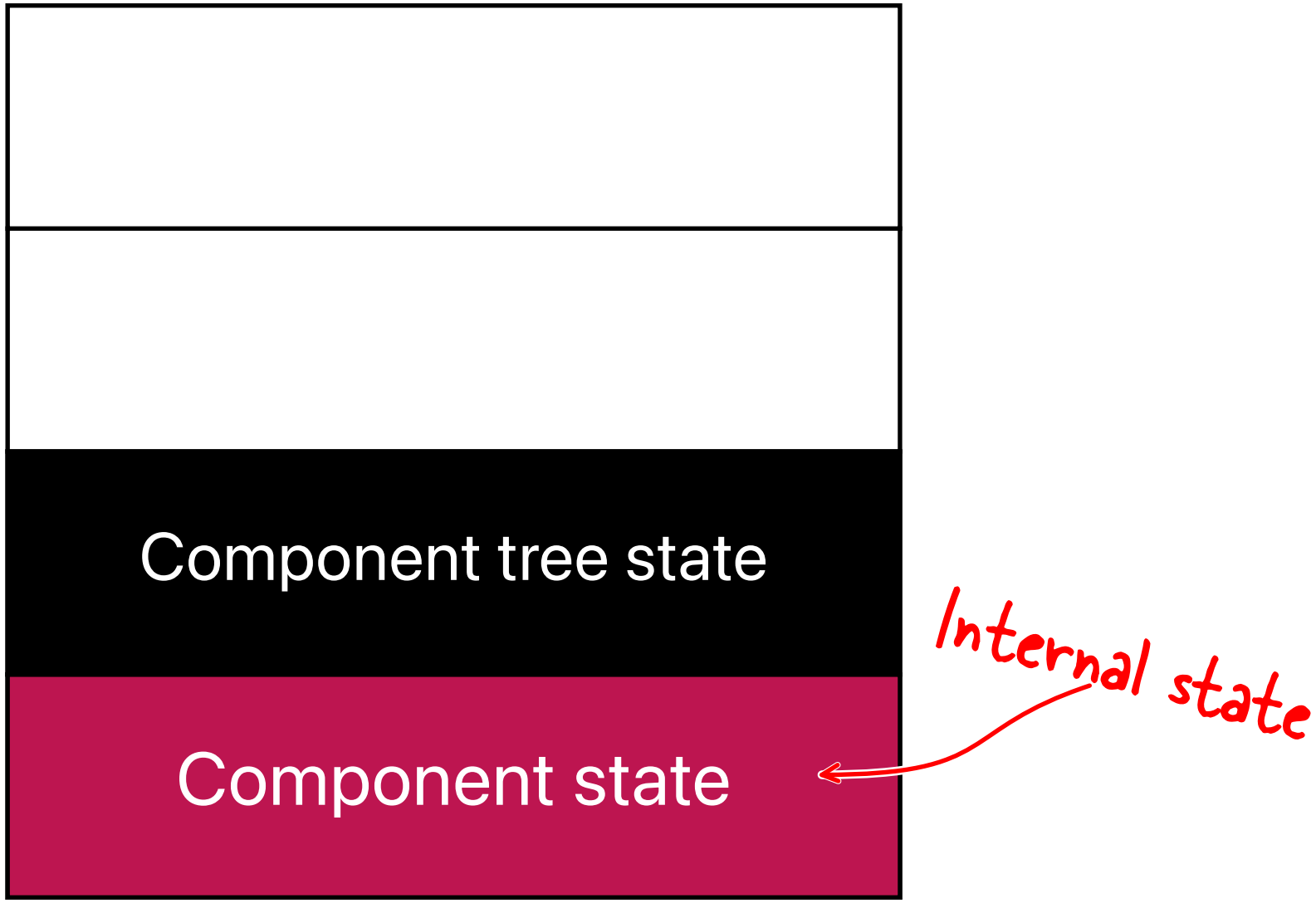
Undo-ing a reducer (recipe 3.2)

```
import reducer from './reducer'
import useUndoReducer from './useUndoReducer'

const Puzzle = () => {
  const [state, dispatch] = useUndoReducer(reducer, {
    items: ['4', '1', '2', '7', '6', '3', null, '5', '8'],
  })
....
      <button
        onClick={() => dispatch({ type: 'undo' })}
      >
        Undo
      </button>
```

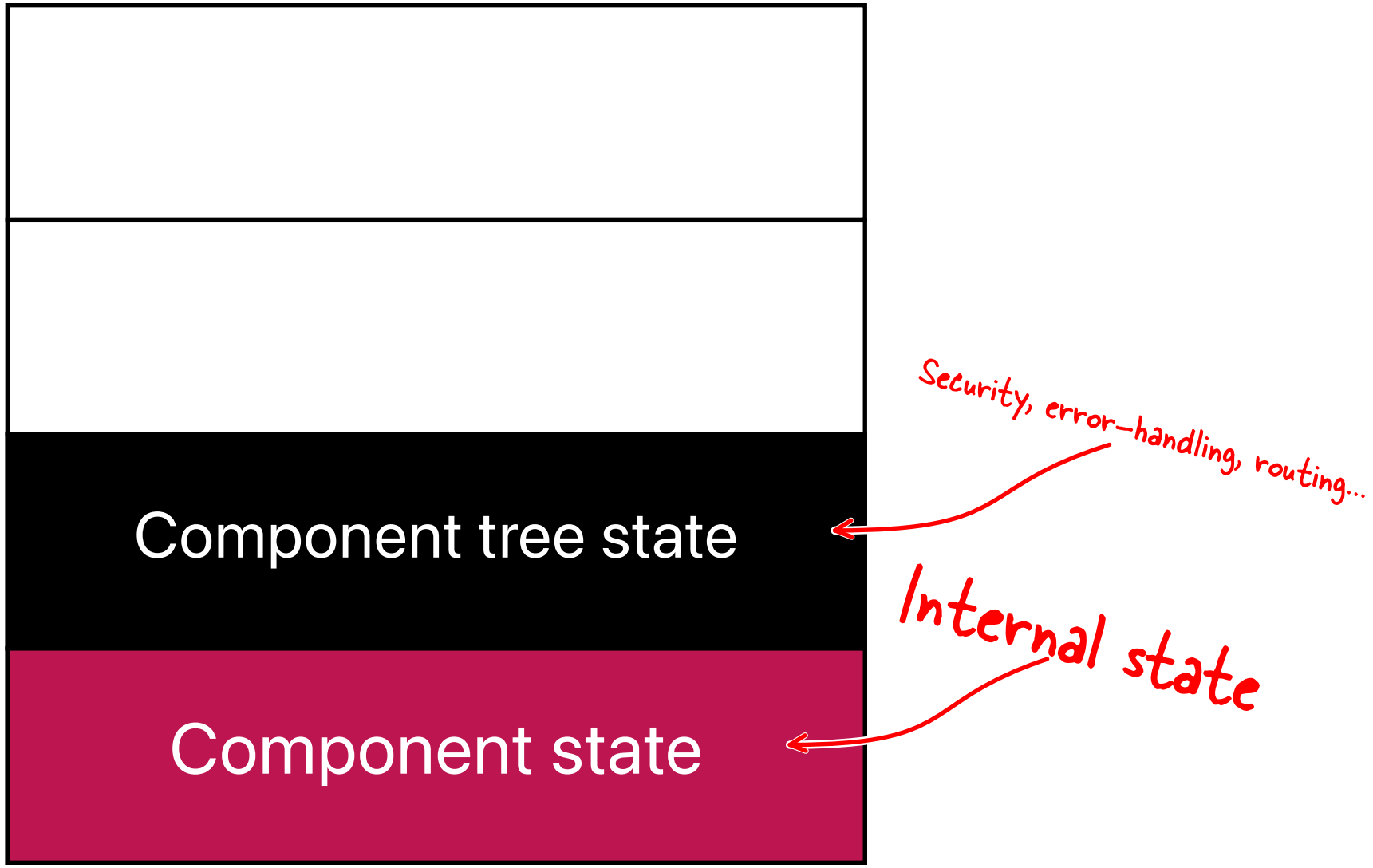
Different levels of state



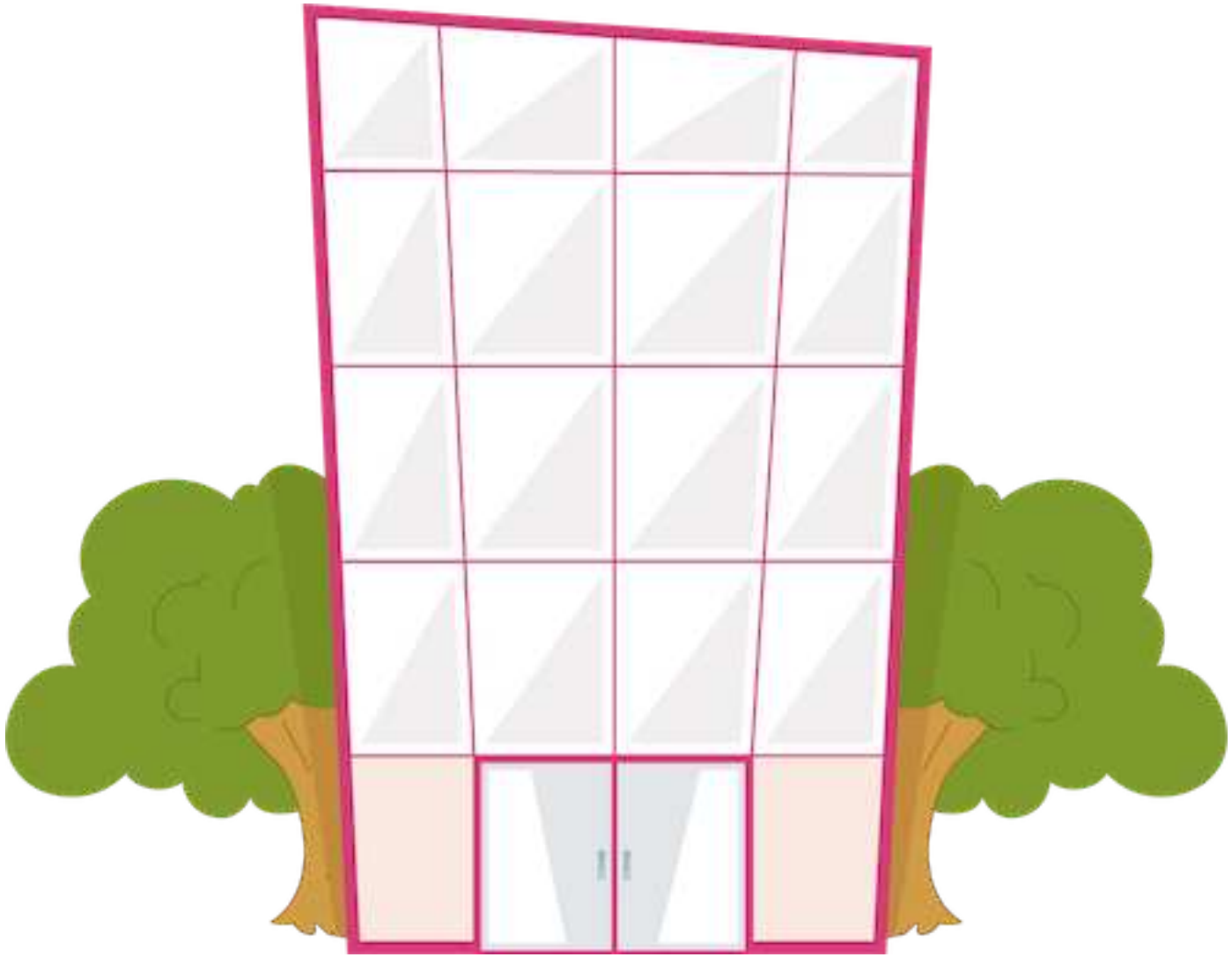
Different levels of state



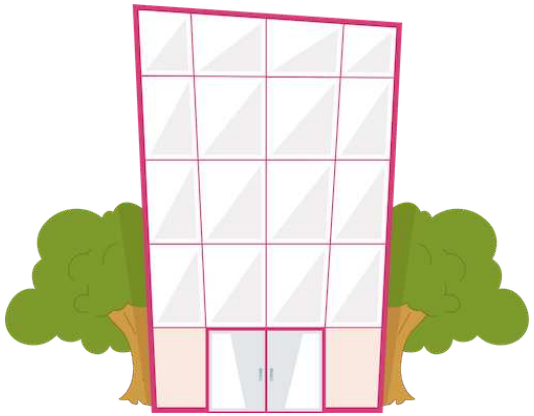
Different levels of state

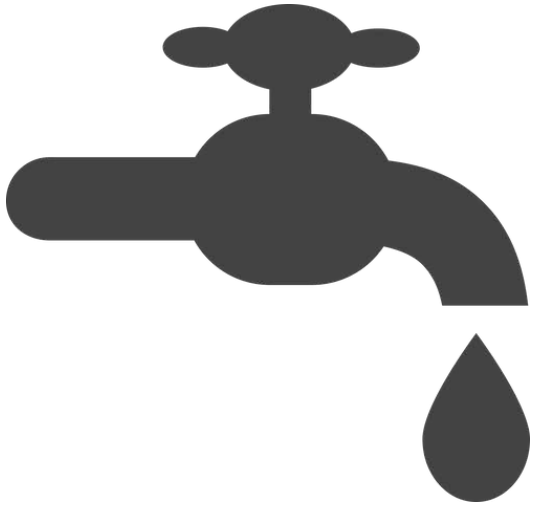


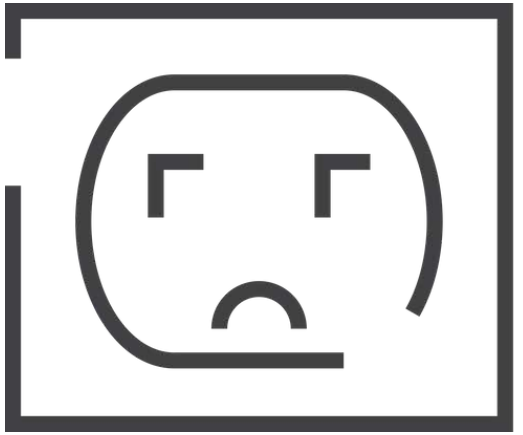
Utilities



Utilities







Use context to secure a component tree (recipe 7.5)

Create a context: SecurityContext.js

```
import React from 'react'

export default React.createContext({})
```

Create a provider: SecurityProvider.js

```
import { useRef, useState } from 'react'
import SecurityContext from './SecurityContext'
import LoginForm from './LoginForm'

export default (props) => {
  return (
    <SecurityContext.Provider
      value={{
      }}
    >
      {props.children}
    </SecurityContext.Provider>
  )
}
```

Create a provider: SecurityProvider.js

```
import { useRef, useState } from 'react'
import SecurityContext from './SecurityContext'
import LoginForm from './LoginForm'

export default (props) => {
  const [showLogin, setShowLogin] = useState(false)
  ....
  return (
    <SecurityContext.Provider
      value={{
        login: () => setShowLogin(true)
        ...
      }}
    >
      {props.children}
    </SecurityContext.Provider>
  )
}
```

Create a provider: SecurityProvider.js

```
import { useRef, useState } from 'react'
import SecurityContext from './SecurityContext'
import LoginForm from './LoginForm'

export default (props) => {
  const [showLogin, setShowLogin] = useState(false)
  ....
  return (
    <SecurityContext.Provider
      value={{
        login: () => setShowLogin(true)
        ...
      }}
    >
      {showLogin ? (
        <LoginForm
          onLogin={async (username, password) => {
            // Clever code to do login goes here....
          }}
        />
      ) : null}
      {props.children}
    </SecurityContext.Provider>
  )
}
```

Wrap the tree in the SecurityProvider

```
function App() {
  return (
    <div className="App">
      <BrowserRouter>
          <ComponentOne/>
          <ComponentTwo/>
          <ComponentThree/>
          ...
      </BrowserRouter>
    </div>
  )
}

export default App
```

Wrap the tree in the SecurityProvider

```
import SecurityProvider from './SecurityProvider'

function App() {
  return (
    <div className="App">
      <BrowserRouter>
          <ComponentOne/>
          <ComponentTwo/>
          <ComponentThree/>
          ...
      </BrowserRouter>
    </div>
  )
}

export default App
```

Wrap the tree in the SecurityProvider

```
import SecurityProvider from './SecurityProvider'

function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <SecurityProvider>
          <ComponentOne/>
          <ComponentTwo/>
          <ComponentThree/>
          ...
        </SecurityProvider>
      </BrowserRouter>
    </div>
  )
}

export default App
```

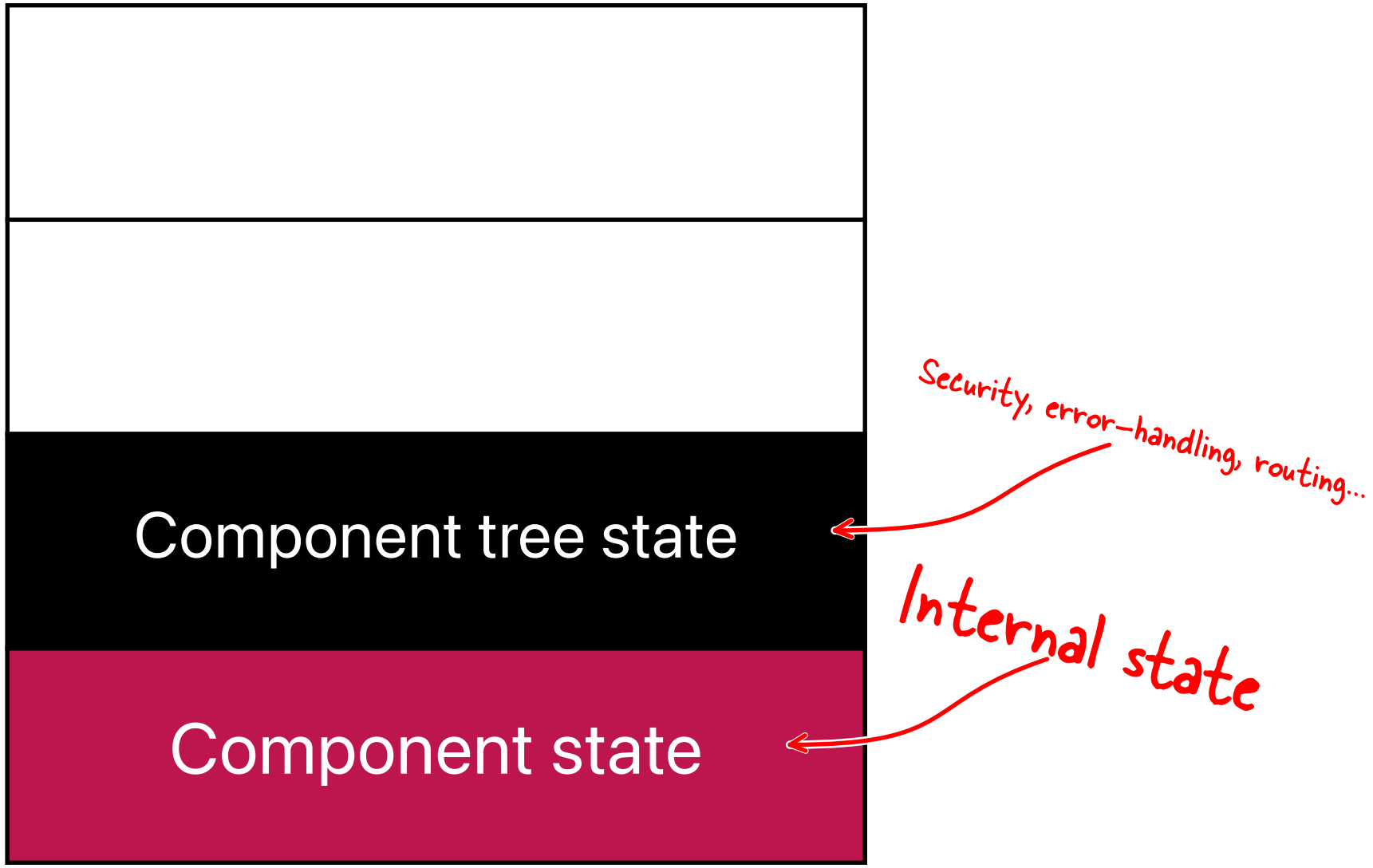
Access security from ComponentOne.js

```
const security = useContext(SecurityContext)
```

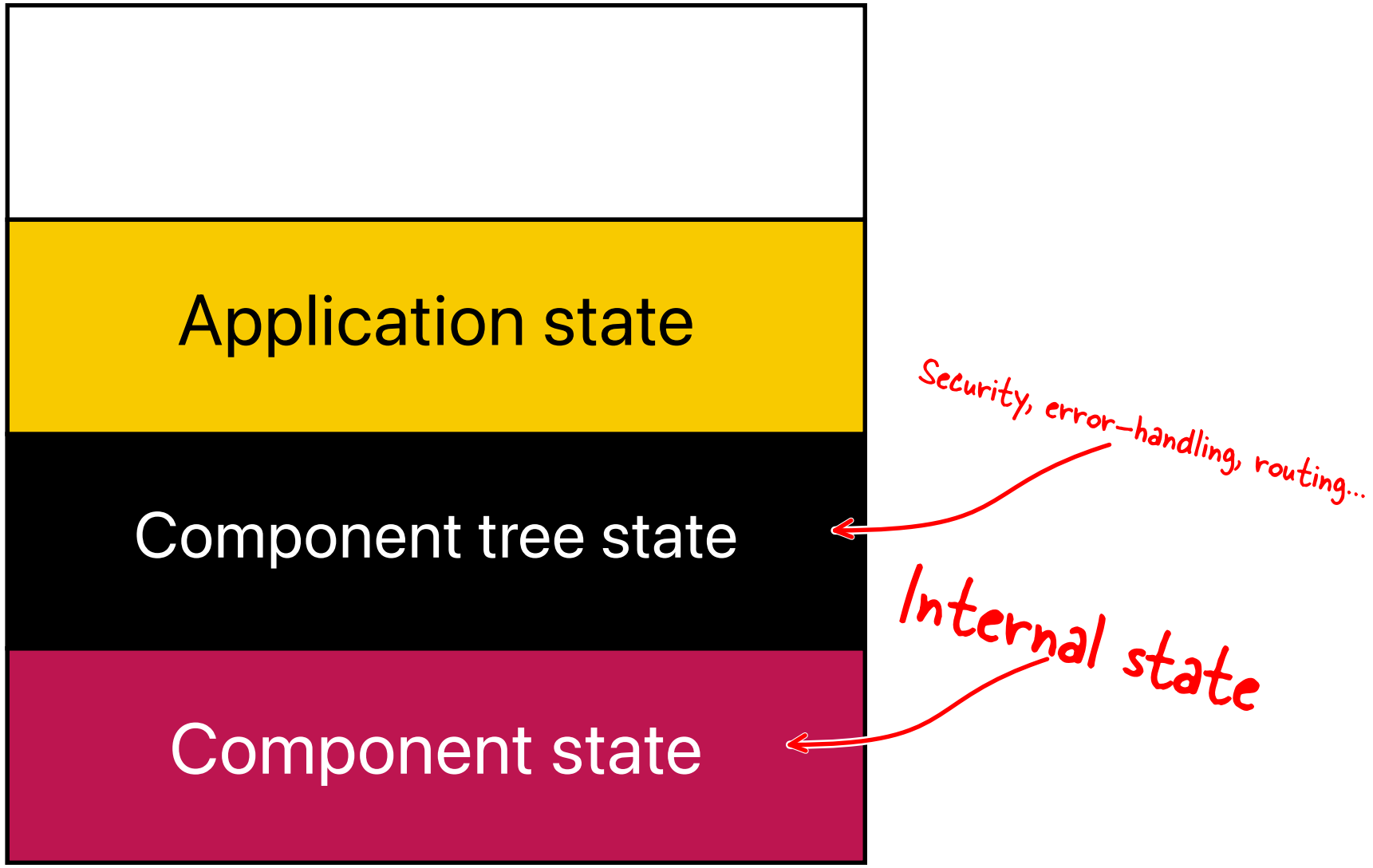
Access security from ComponentOne.js

```
const security = useContext(SecurityContext)
<button
  onClick={() => security.login())}
>
  Open login form
</button>
```

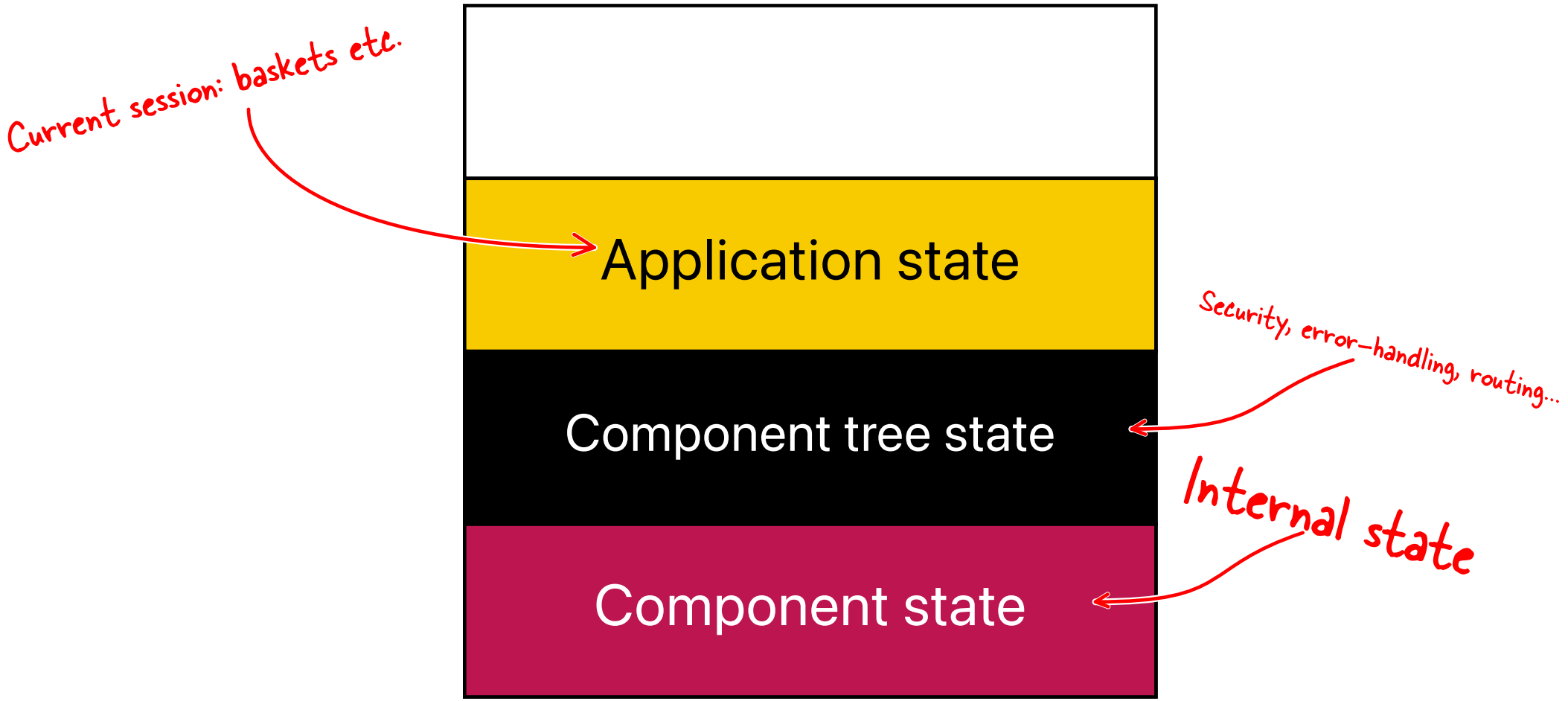
Different levels of state



Different levels of state



Different levels of state



Managing central application state (recipe 3.6)

Managing central application state (recipe 3.6)

Managing central application state (recipe 3.6)

- Some state needs to available to entire app

Managing central application state (recipe 3.6)

- Some state needs to available to entire app
- Needs be managed by separate code...

Managing central application state (recipe 3.6)

- Some state needs to available to entire app
- Needs be managed by separate code...
- ...so could use a reducer

Managing central application state (recipe 3.6)

- Some state needs to available to entire app
- Needs be managed by separate code...
- ...so could use a reducer
- And we need it to be available to entire component tree...

Managing central application state (recipe 3.6)

- Some state needs to available to entire app
- Needs be managed by separate code...
- ...so could use a reducer
- And we need it to be available to entire component tree...
- ...so accessible through a context

Managing central application state (recipe 3.6)

- Some state needs to available to entire app
- Needs be managed by separate code...
- ...so could use a reducer
- And we need it to be available to entire component tree...
- ...so accessible through a context
- The solution: Redux

Reducer to manage a shopping basket

```
const reducer = (state = {}, action = {}) => {
  switch (action.type) {
    case 'buy': {
    }
    case 'clearBasket': {
    }
    default:
      return { ...state }
  }
}

export default reducer
```

Reducer to manage a shopping basket

```
const reducer = (state = {}, action = {}) => {
  switch (action.type) {
    case 'buy': {
      const basket = state.basket ? [...state.basket] : []
      // Code to add to basket goes here...
      return {
        ...state,
        basket,
      }
    }
    case 'clearBasket': {
    }
    default:
      return { ...state }
  }
}

export default reducer
```

Reducer to manage a shopping basket

```
const reducer = (state = {}, action = {}) => {
  switch (action.type) {
    case 'buy': {
      const basket = state.basket ? [...state.basket] : []
      // Code to add to basket goes here...
      return {
        ...state,
        basket,
      }
    }
    case 'clearBasket': {
      return {
        ...state,
        basket: [],
      }
    }
    default:
      return { ...state }
  }
}

export default reducer
```

Create a store using the basket reducer App.js

```
function App() {
  return (
    <div className="App">
          <MyFirstComponent/>
          <MySecondComponent/>
          <Basket/>
    </div>
  )
}

export default App
```

Create a store using the basket reducer App.js

```
import { Provider } from 'react-redux'
import { createStore } from 'redux'
import reducer from './reducer'
const store = createStore(reducer)
function App() {
  return (
    <div className="App">
      <Provider store={store}>
          <MyFirstComponent/>
          <MySecondComponent/>
          <Basket/>
      </Provider>
    </div>
  )
}

export default App
```

Use the store from Basket.js

```
import { useDispatch, useSelector } from 'react-redux'
const Basket = () => {
}

export default Basket
```

Use the store from Basket.js

```
import { useDispatch, useSelector } from 'react-redux'
const Basket = () => {
  const basket = useSelector((state) => state.basket)

  return (
    <div className="Basket">
    </div>
  )
}

export default Basket
```

Use the store from Basket.js

```
import { useDispatch, useSelector } from 'react-redux'
const Basket = () => {
  const basket = useSelector((state) => state.basket)

  return (
    <div className="Basket">
      <h2>Basket</h2>
      {basket.map((item) => (
        <div className="Basket-item">
          <div className="Basket-itemName">{item.name}</div>
          ...
        </div>
      ))}
    </div>
  )
}

export default Basket
```

Use the store from Basket.js

```
import { useDispatch, useSelector } from 'react-redux'
const Basket = () => {
  const basket = useSelector((state) => state.basket)
  const dispatch = useDispatch()

  return (
    <div className="Basket">
      <h2>Basket</h2>
      {basket.map((item) => (
        <div className="Basket-item">
          <div className="Basket-itemName">{item.name}</div>
          ...
        </div>
      ))}
      <button onClick={() => dispatch({ type: 'clearBasket' })}>
        Clear
      </button>
    </div>
  )
}

export default Basket
```

Using Redux libraries

Using Redux libraries

- Can use middleware to sync Redux store with a server (recipe 3.6)

Using Redux libraries

- Can use middleware to sync Redux store with a server (recipe 3.6)
- Can use `redux-persist` to save the store in LocalStorage

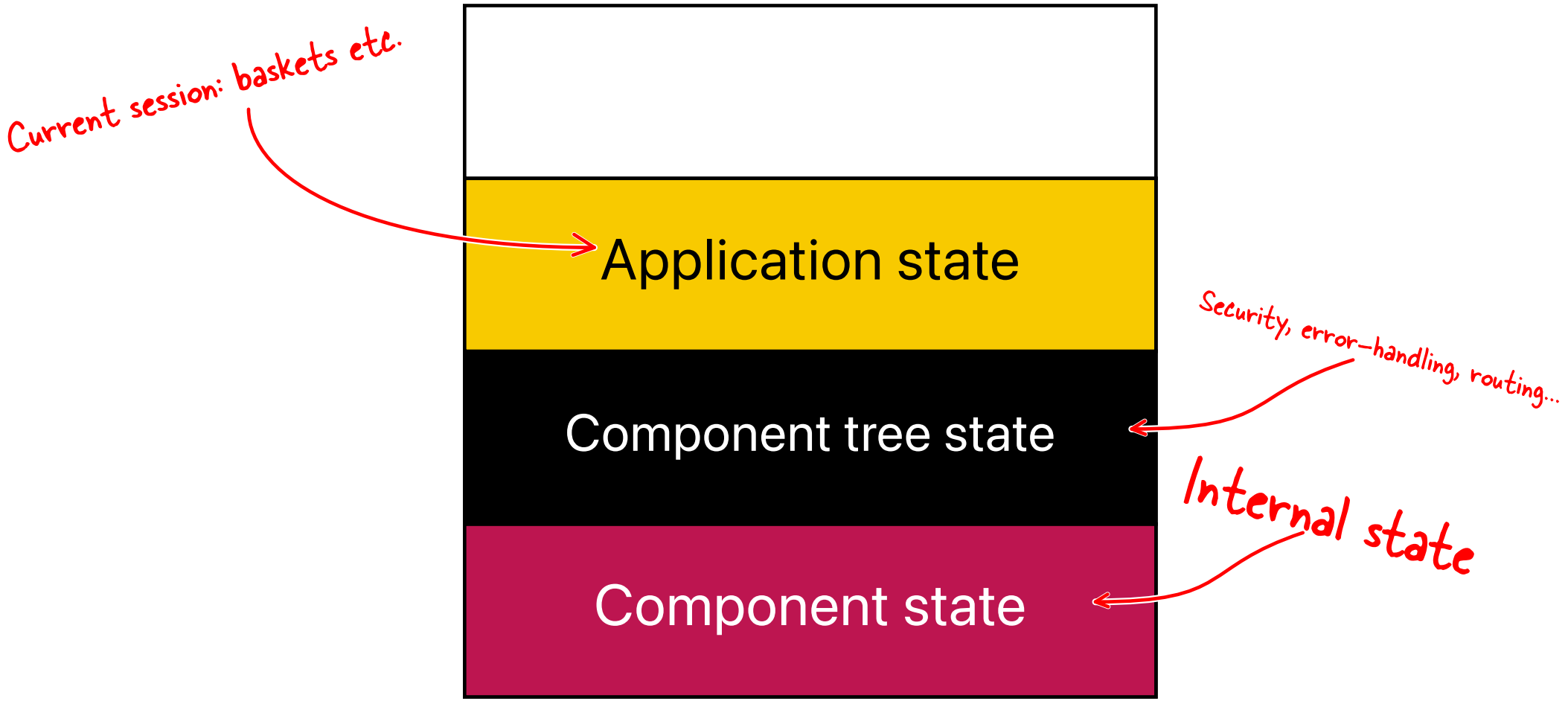
Using Redux libraries

- Can use middleware to sync Redux store with a server (recipe 3.6)
- Can use `redux-persist` to save the store in LocalStorage
- ...means you can refresh the page and the data is safe (recipe 3.7)

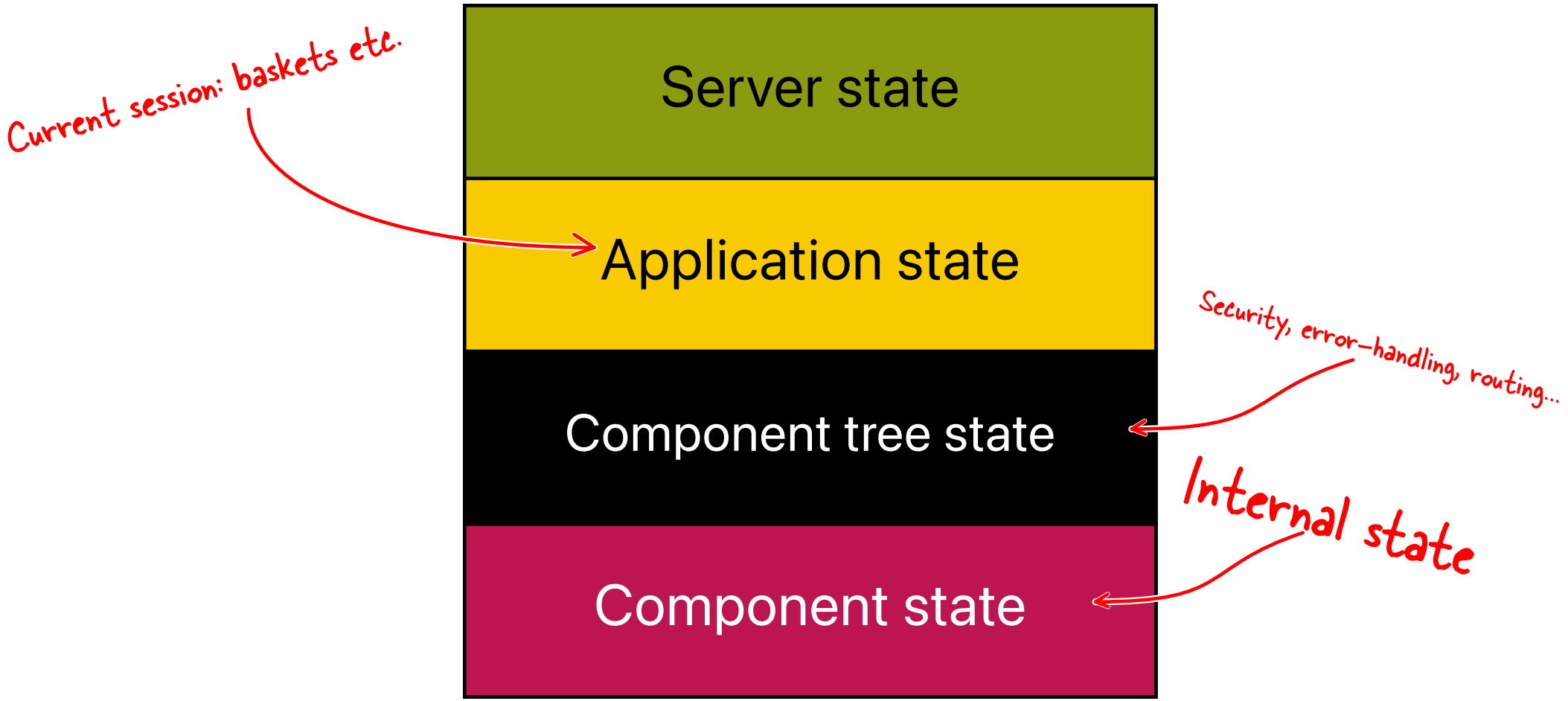
Using Redux libraries

- Can use middleware to sync Redux store with a server (recipe 3.6)
- Can use `redux-persist` to save the store in LocalStorage
- ...means you can refresh the page and the data is safe (recipe 3.7)
- Can use with libraries like `reselect` to derive states (recipe 3.8)

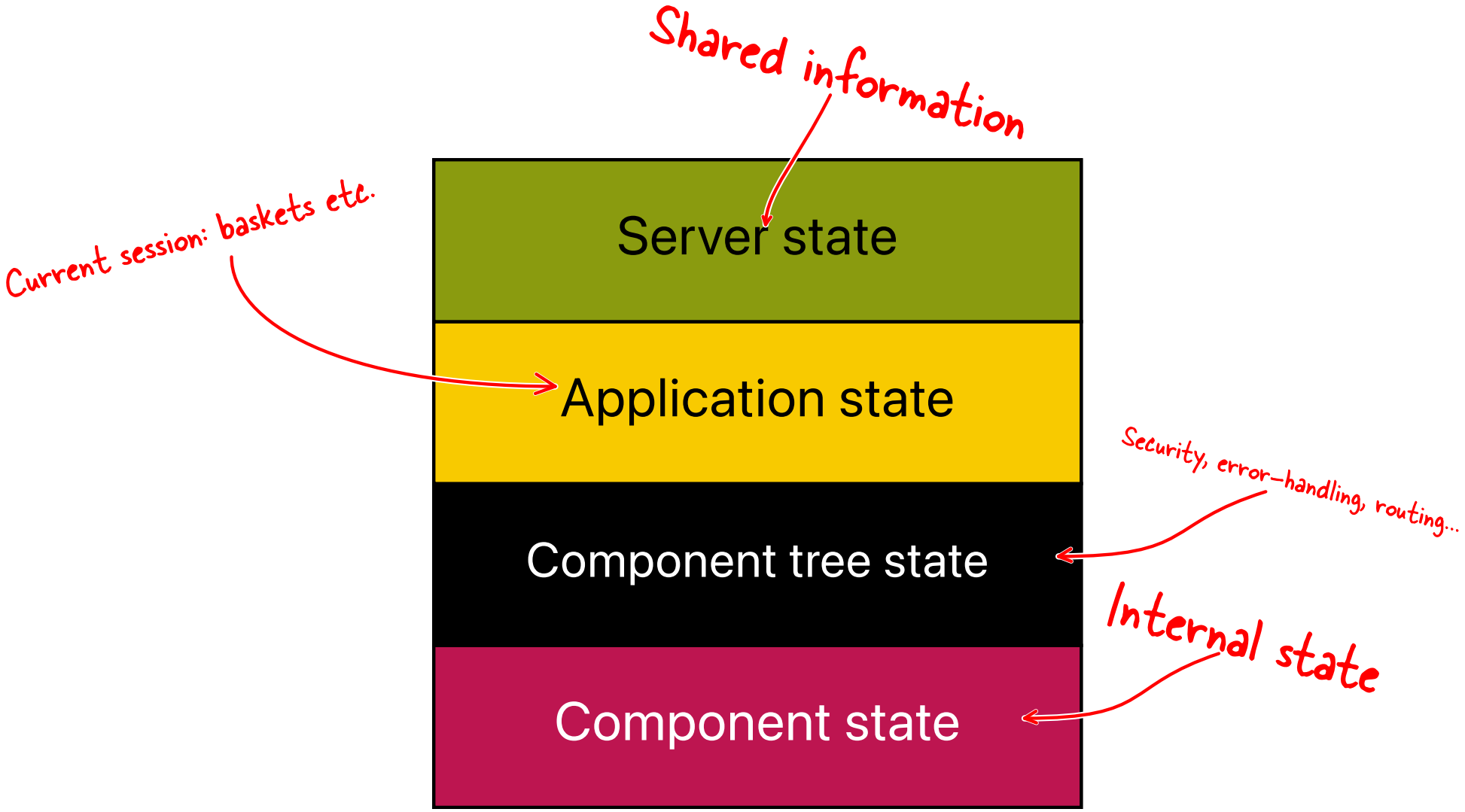
Different levels of state



Different levels of state



Different levels of state



Reading data when offline? (recipe 11.3)

Reading data when offline? (recipe 11.3)

Reading data when offline? (recipe 11.3)

- Many users are on mobile devices

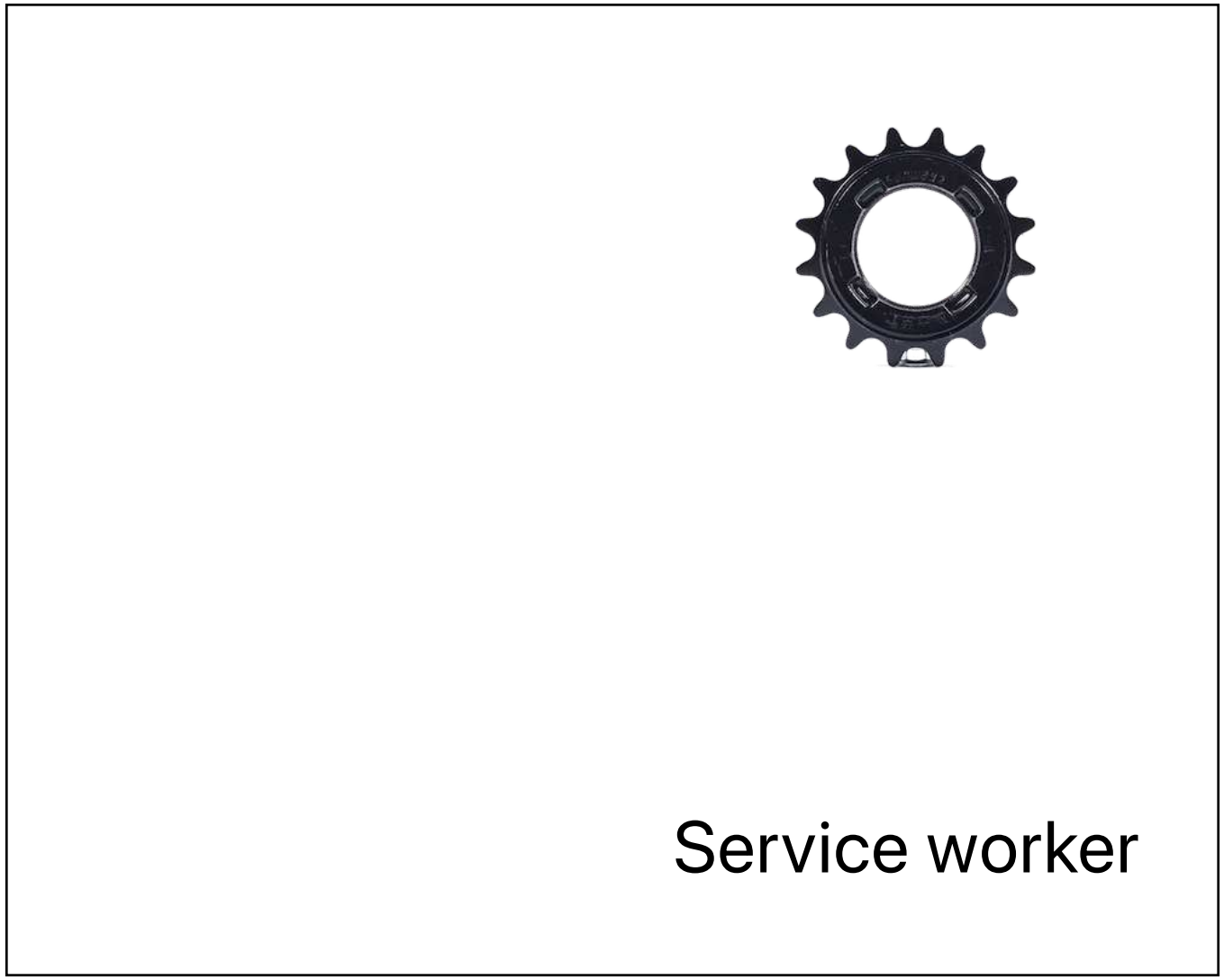
Reading data when offline? (recipe 11.3)

- Many users are on mobile devices
- Mobile devices have poor network connections

Reading data when offline? (recipe 11.3)

- Many users are on mobile devices
- Mobile devices have poor network connections
- Can you cache data locally so it can work offline?

Use a Service Worker

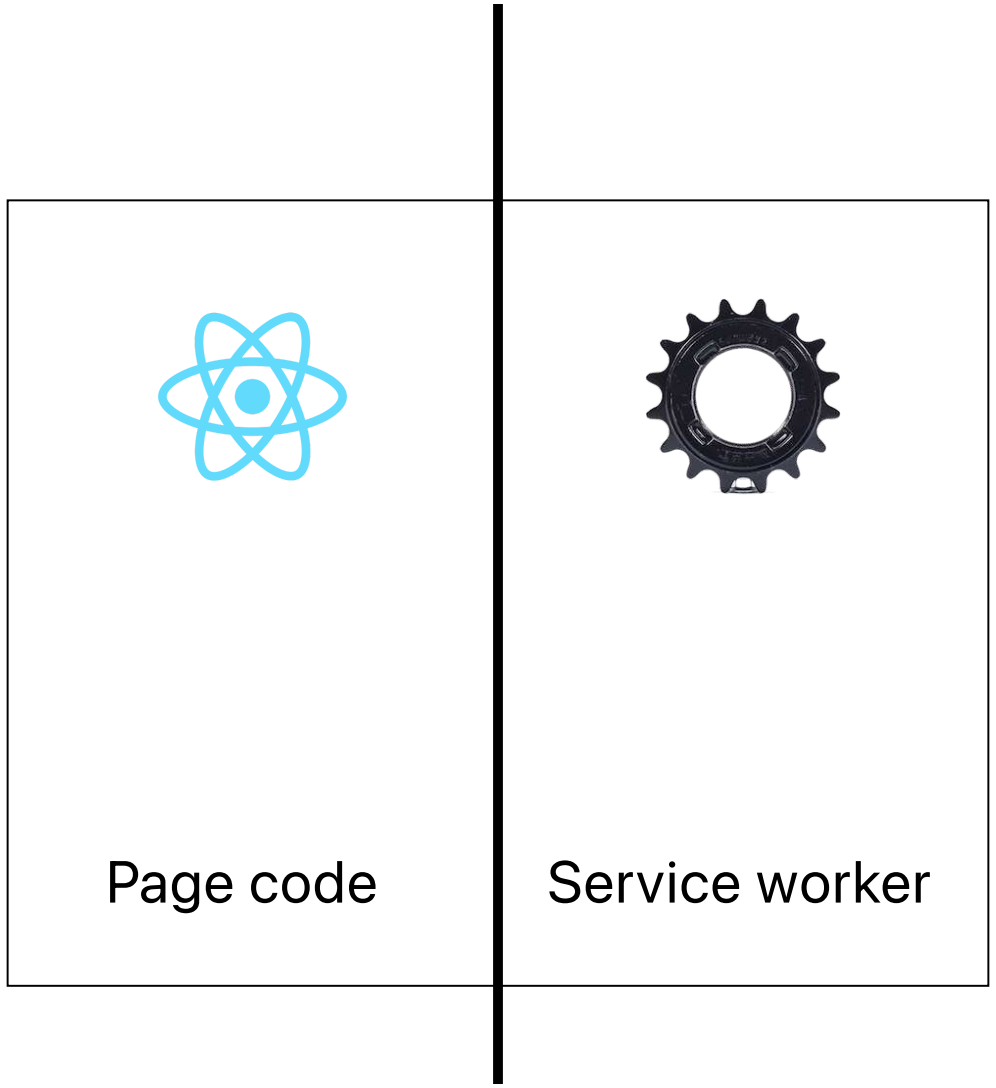


Use a Service Worker

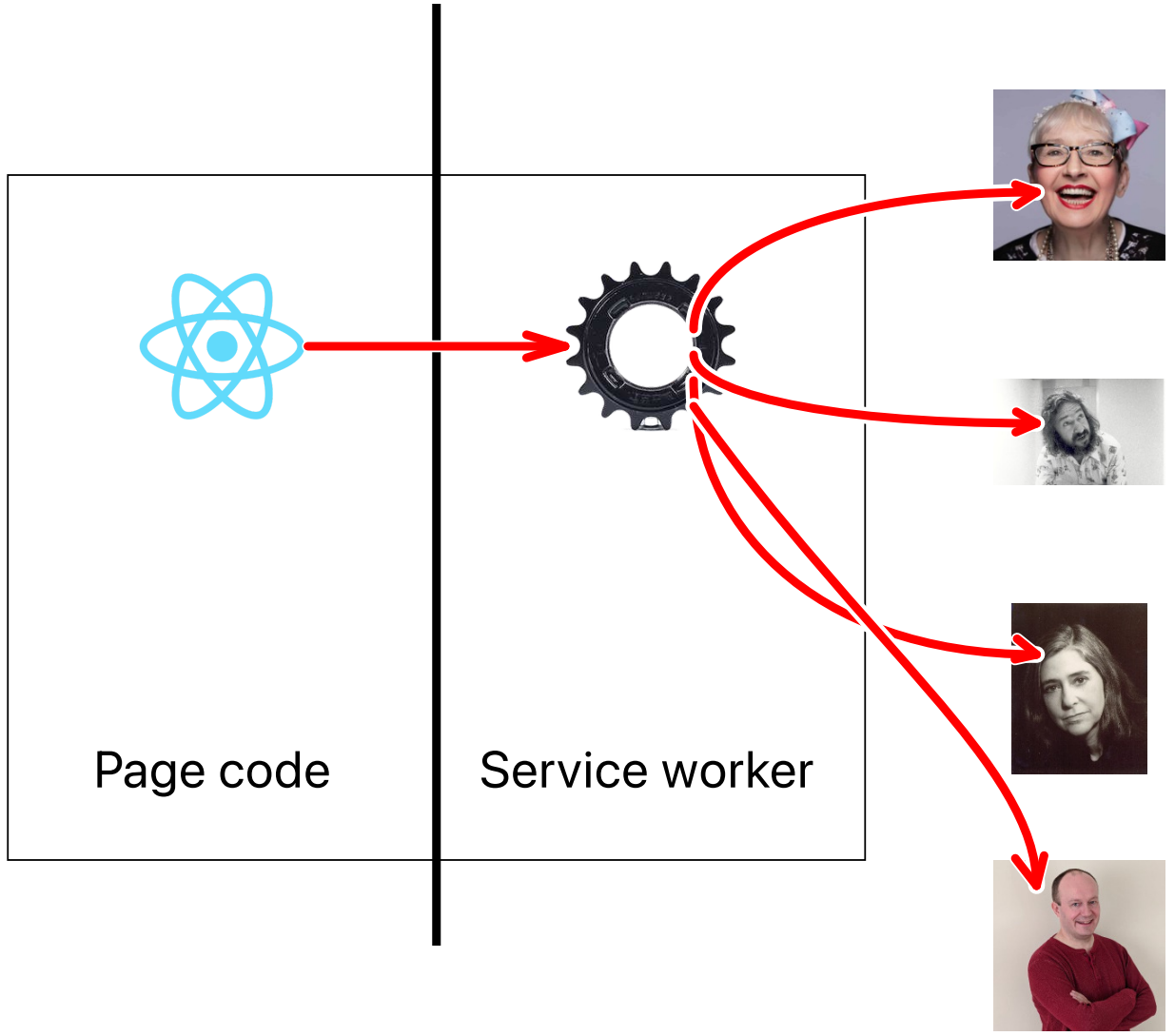
Web worker

Service worker

Use a Service Worker



Use a Service Worker



Service workers

Service workers

- JavaScript that runs in the background (not the page)

Service workers

- JavaScript that runs in the background (not the page)
- Can intercept all networks connections

Service workers

- JavaScript that runs in the background (not the page)
- Can intercept all networks connections
- Can use cache storage for images, fonts, network responses

Service workers (part 2)

Service workers (part 2)

- Not normally used when in development mode

Service workers (part 2)

- Not normally used when in development mode
- Only available with HTTPS or with localhost

Service workers (part 2)

- Not normally used when in development mode
- Only available with HTTPS or with localhost
- Might need to unregister when developing

Register a service worker index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorkerRegistration from './serviceWorkerRegistration';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Register a service worker index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorkerRegistration from './serviceWorkerRegistration';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

serviceWorkerRegistration.register();
```

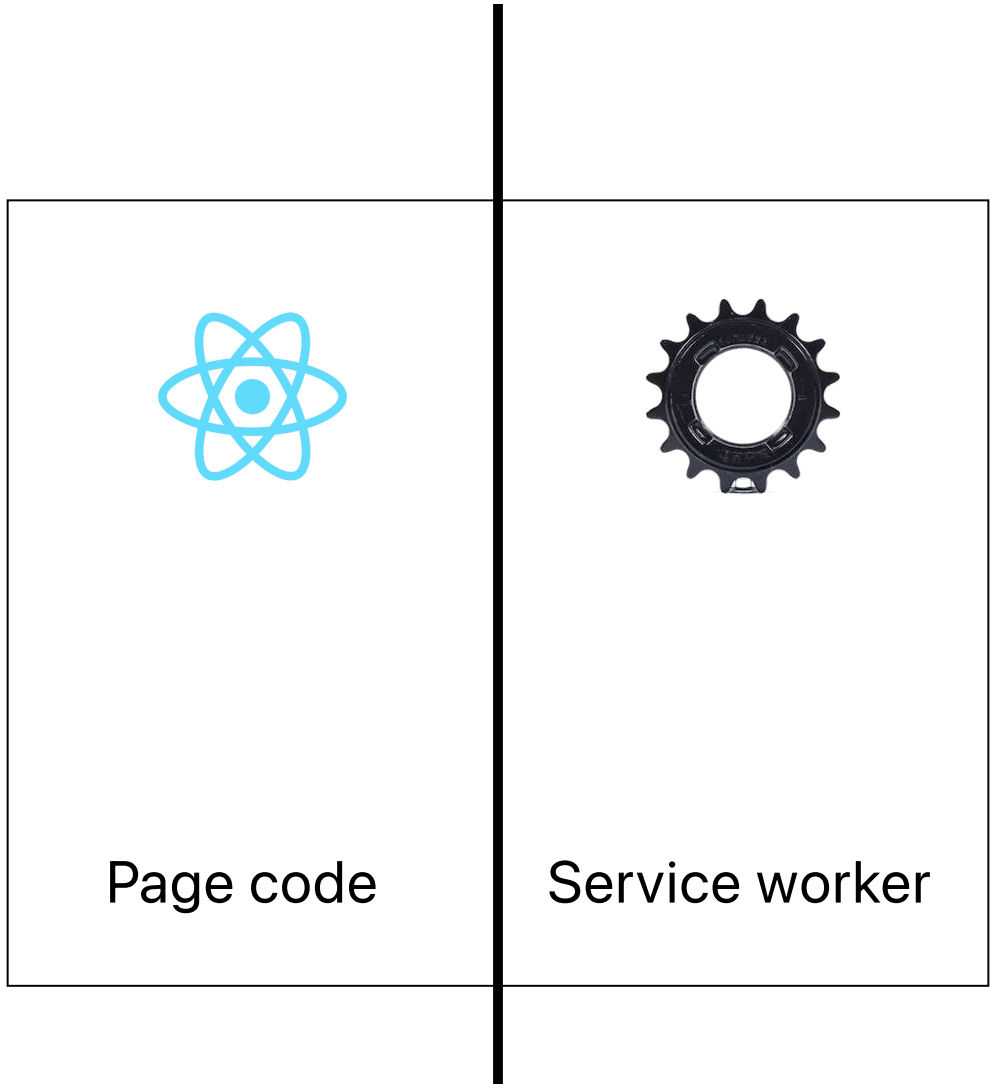
The service worker: service-worker.js

```
import { registerRoute } from 'workbox-routing';
import { StaleWhileRevalidate } from 'workbox-strategies';

// All the other service-worker code...
```

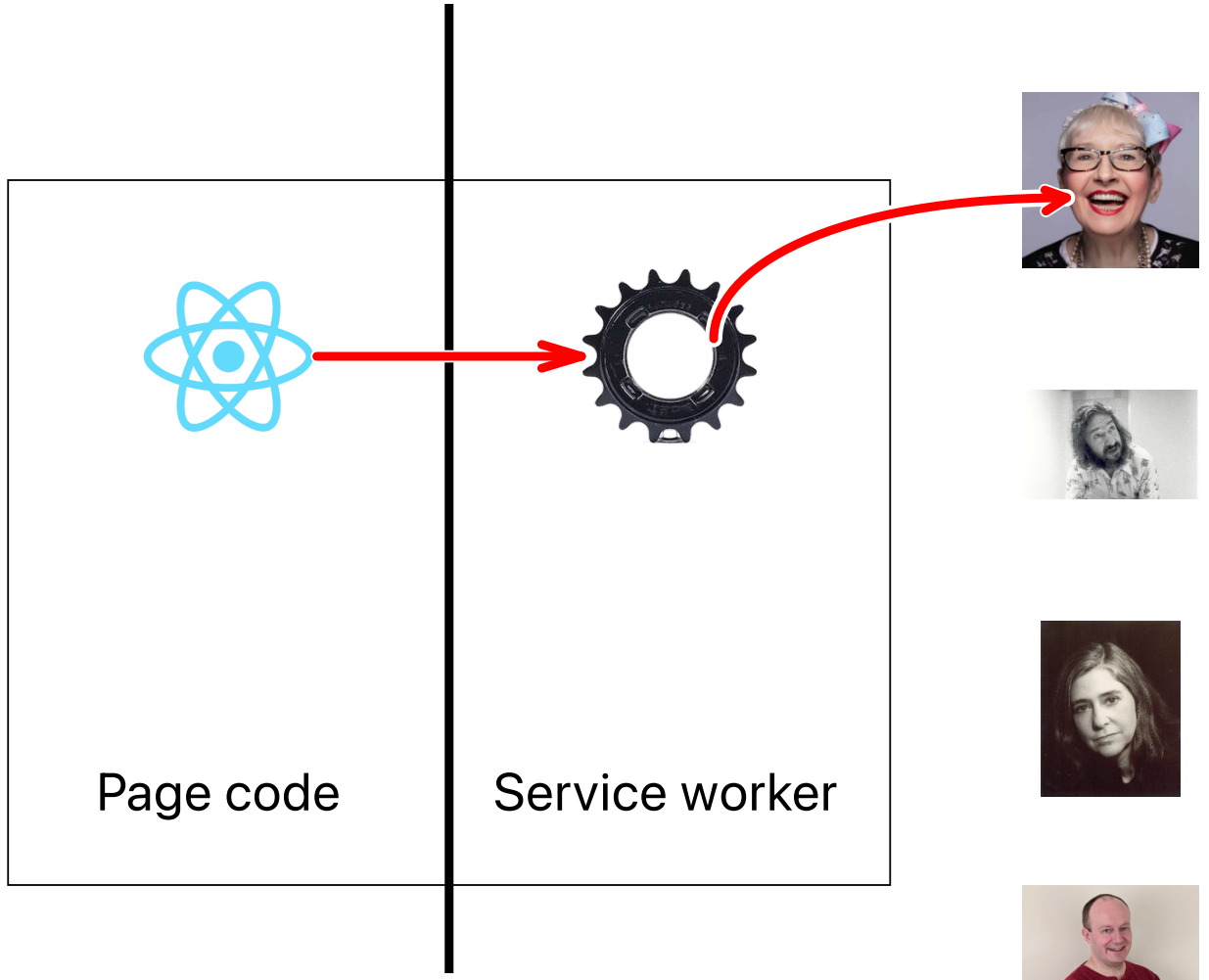
The service worker: service-worker.js

```
import { registerRoute } from 'workbox-routing';
import { StaleWhileRevalidate } from 'workbox-strategies';

// All the other service-worker code...

registerRoute(
    ({url}) => url.origin === 'https://fonts.googleapis.com',
    new StaleWhileRevalidate({
        cacheName: 'stylesheets',
    })
);
```

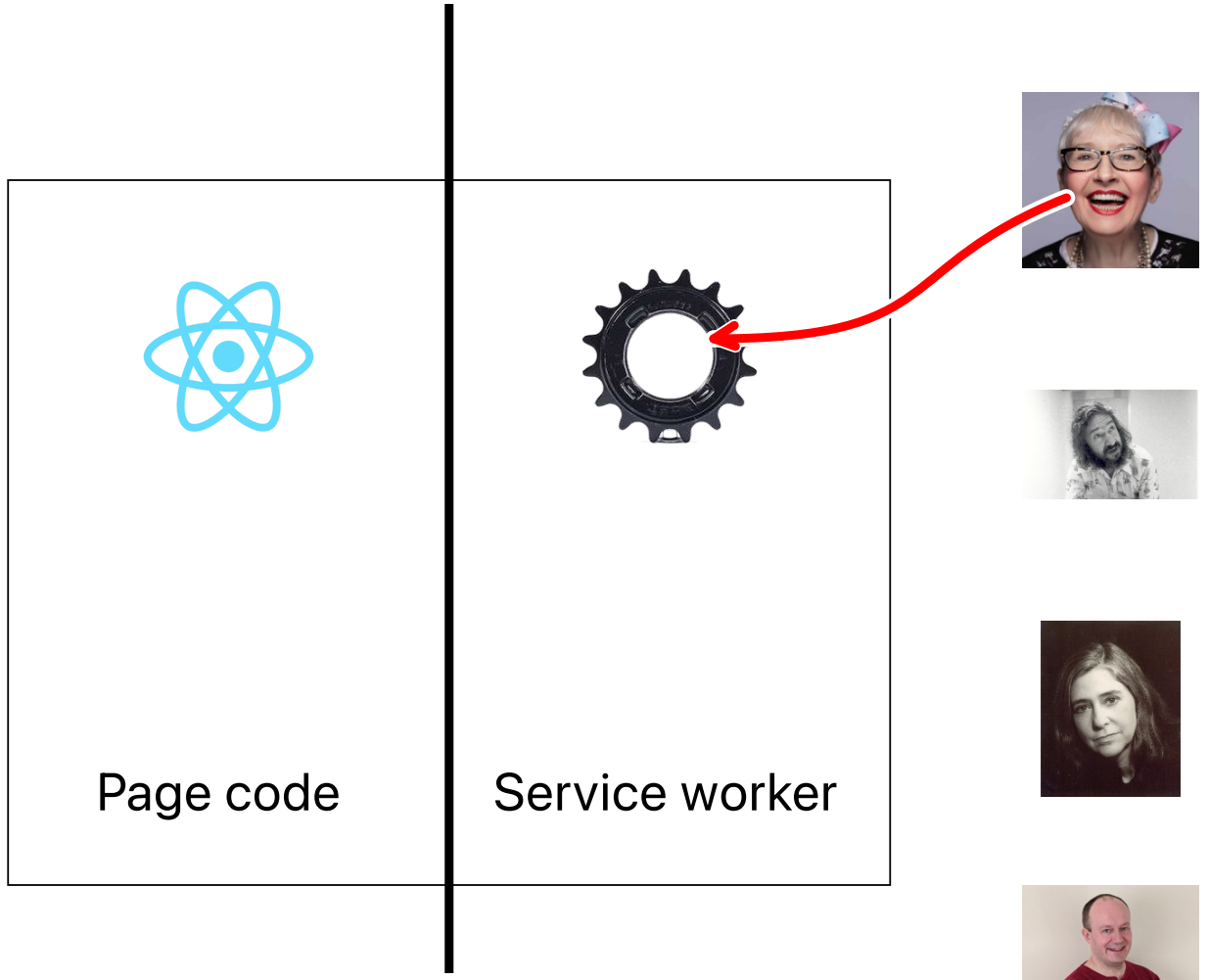
StaleWhileRevalidate



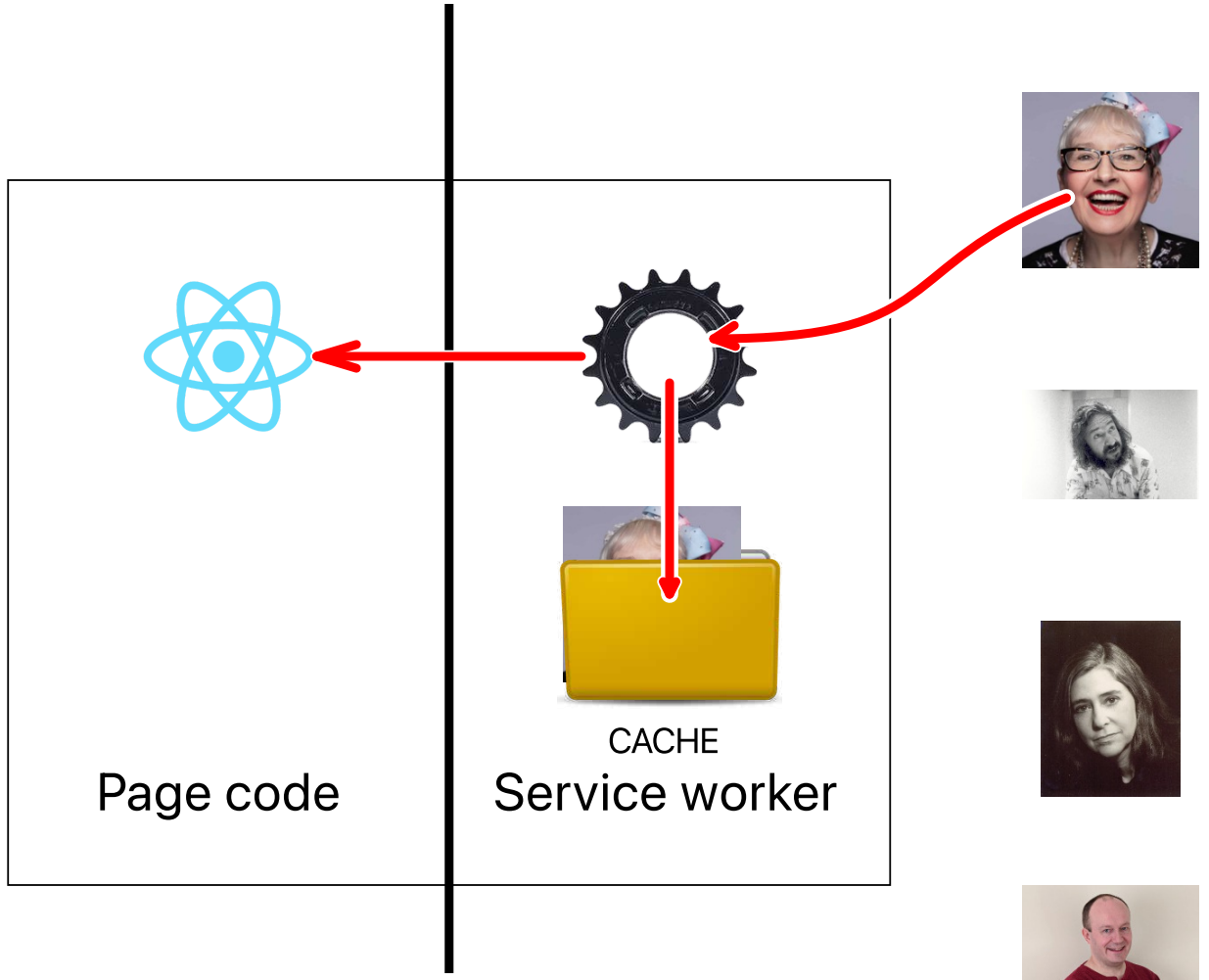
StaleWhileRevalidate



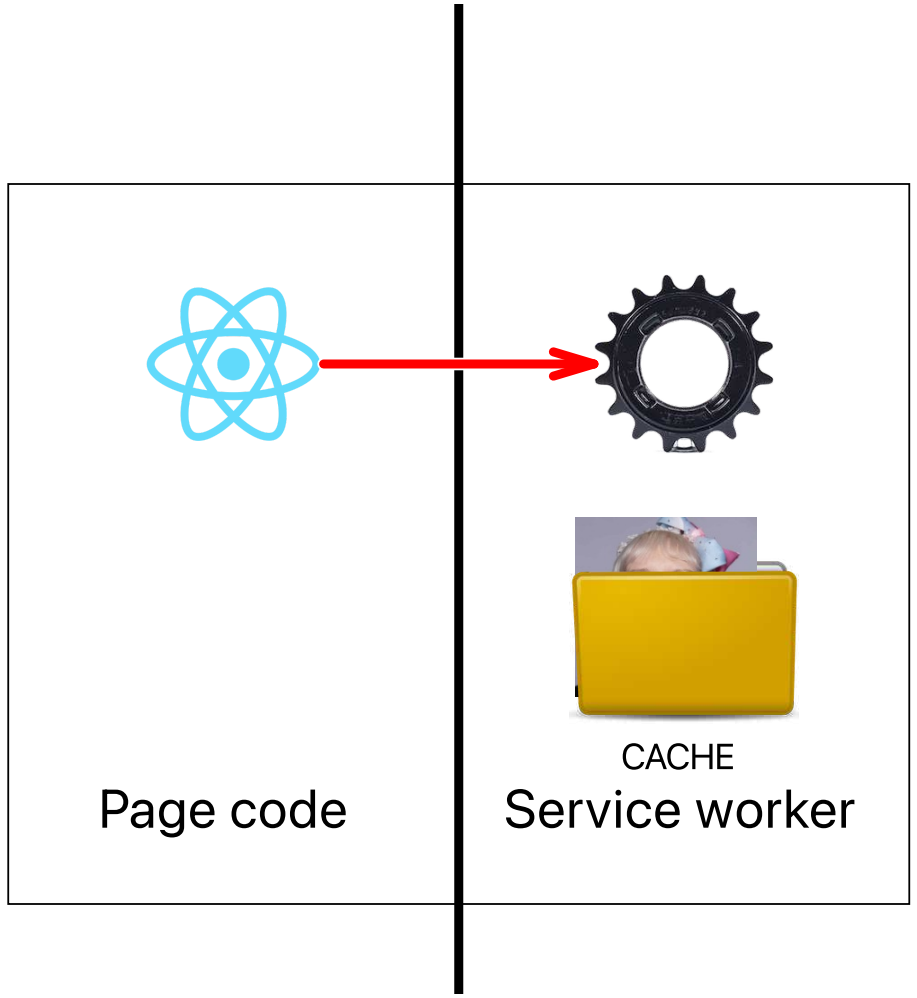
StaleWhileRevalidate



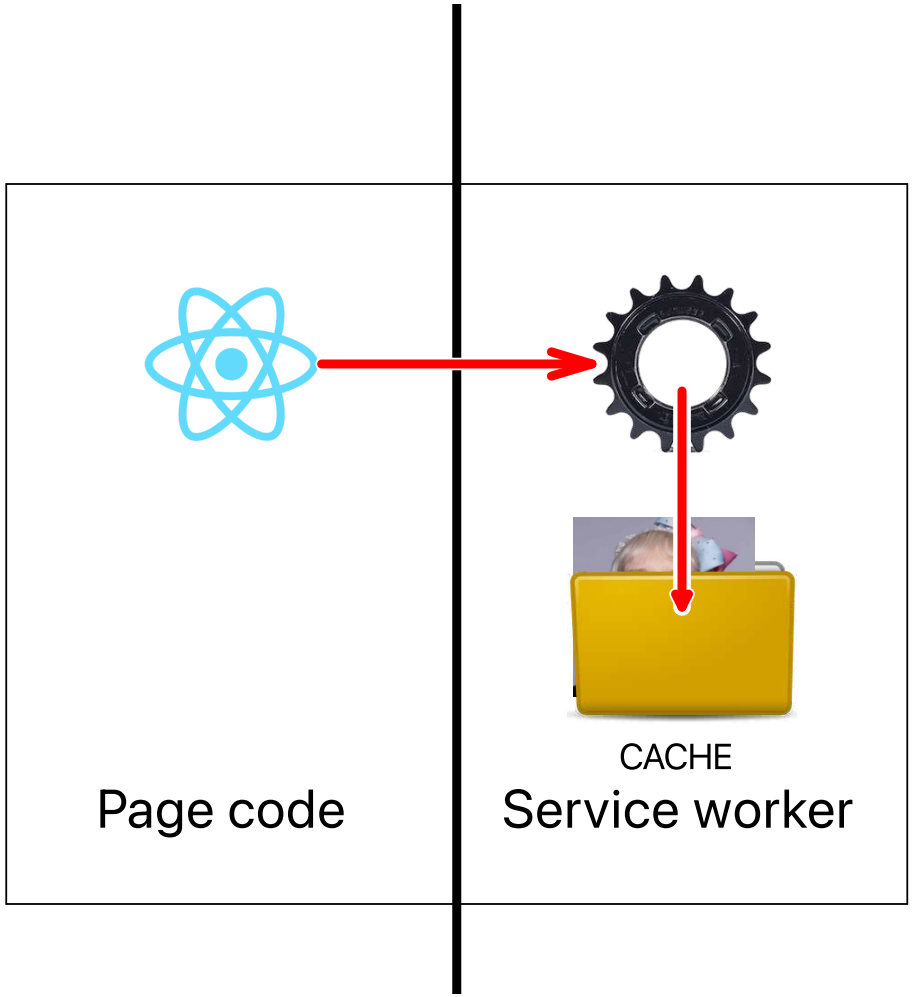
StaleWhileRevalidate



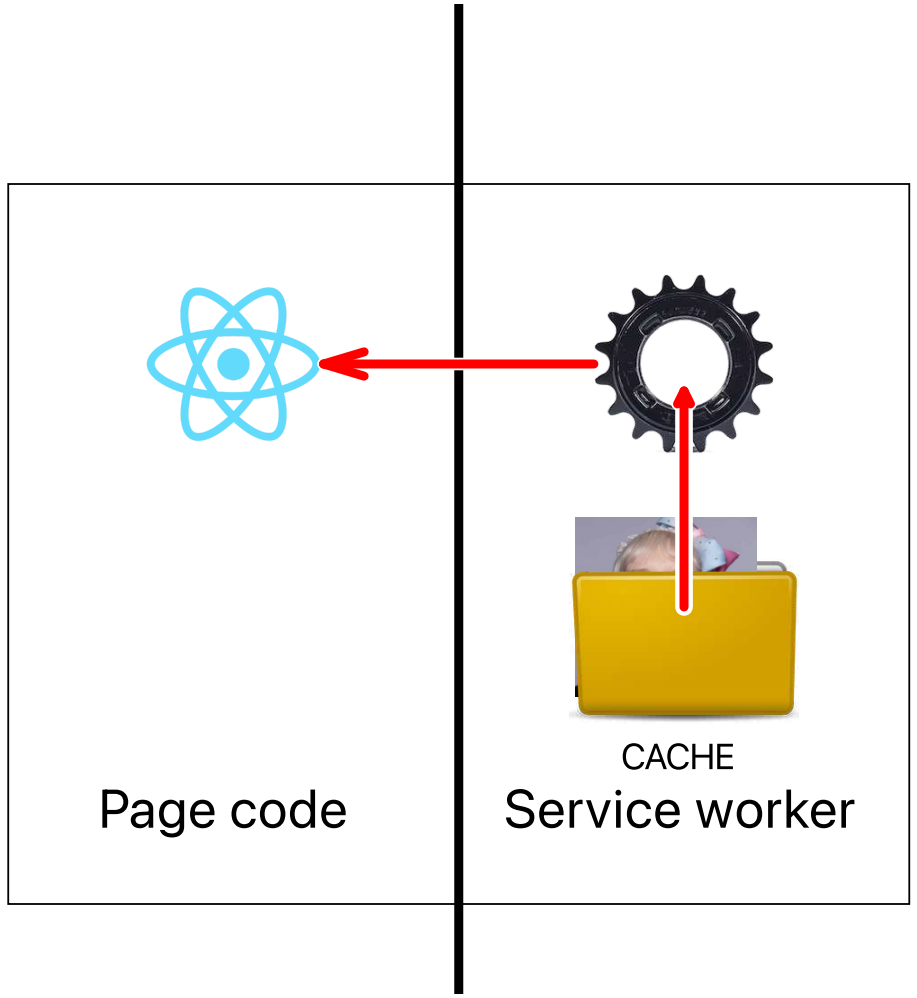
StaleWhileRevalidate



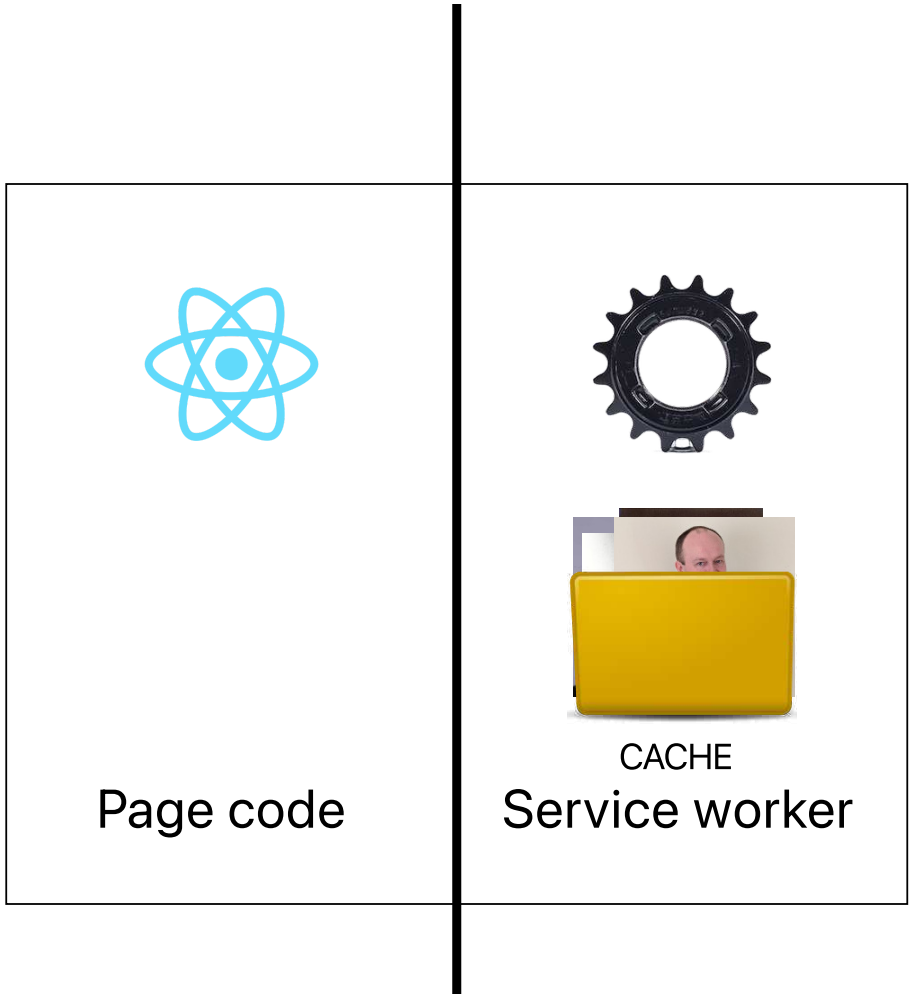
StaleWhileRevalidate



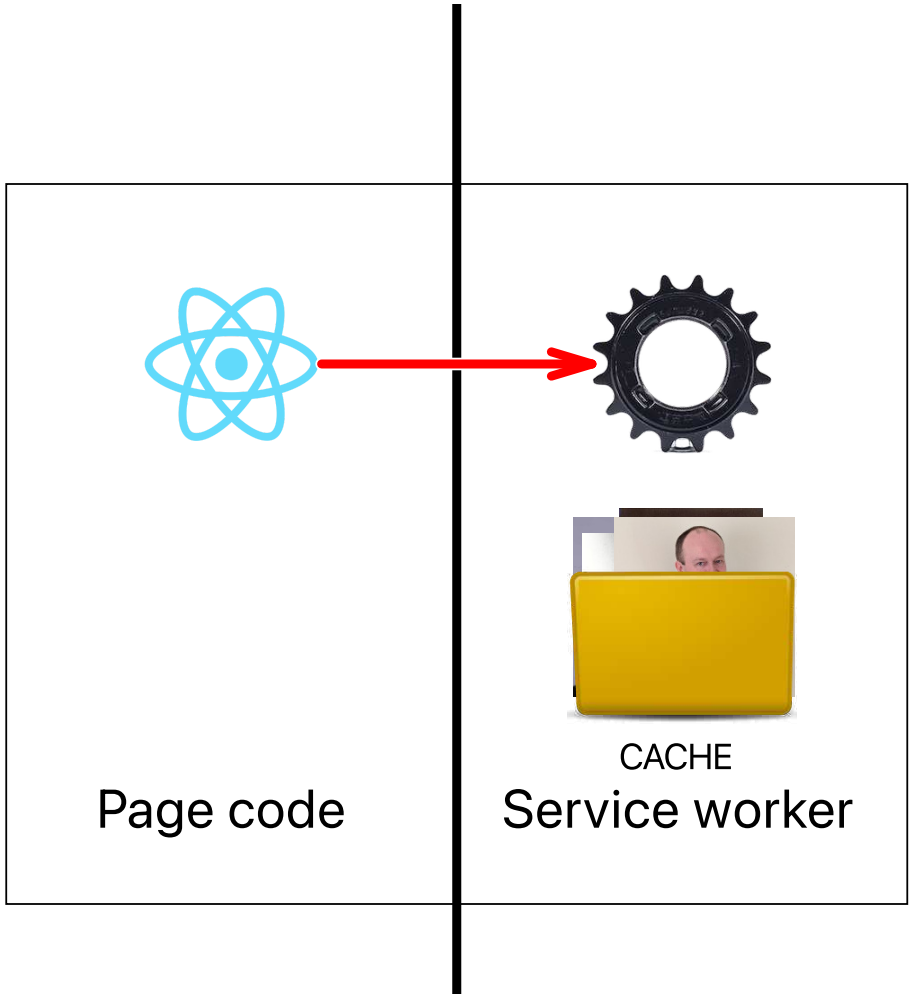
StaleWhileRevalidate



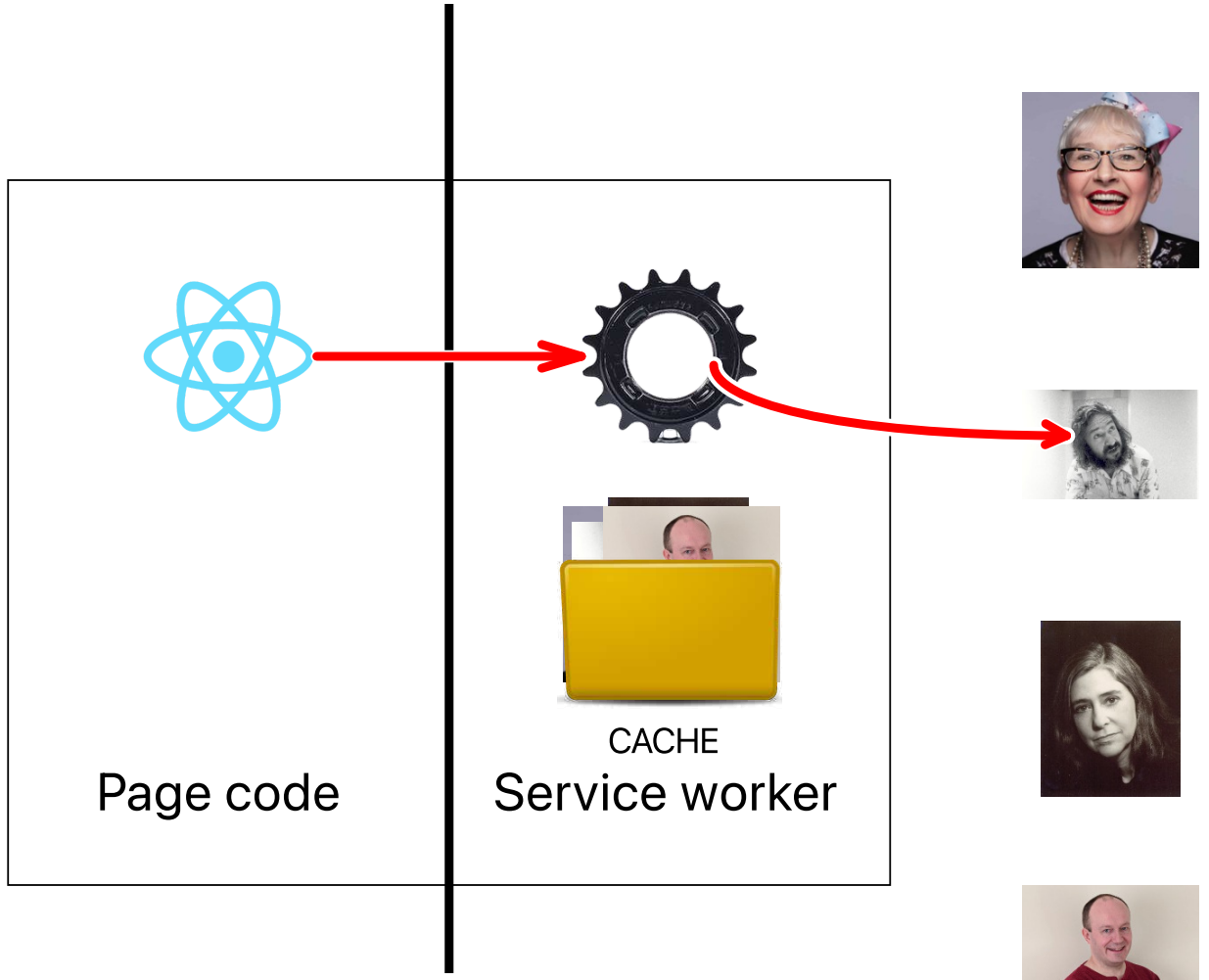
StaleWhileRevalidate



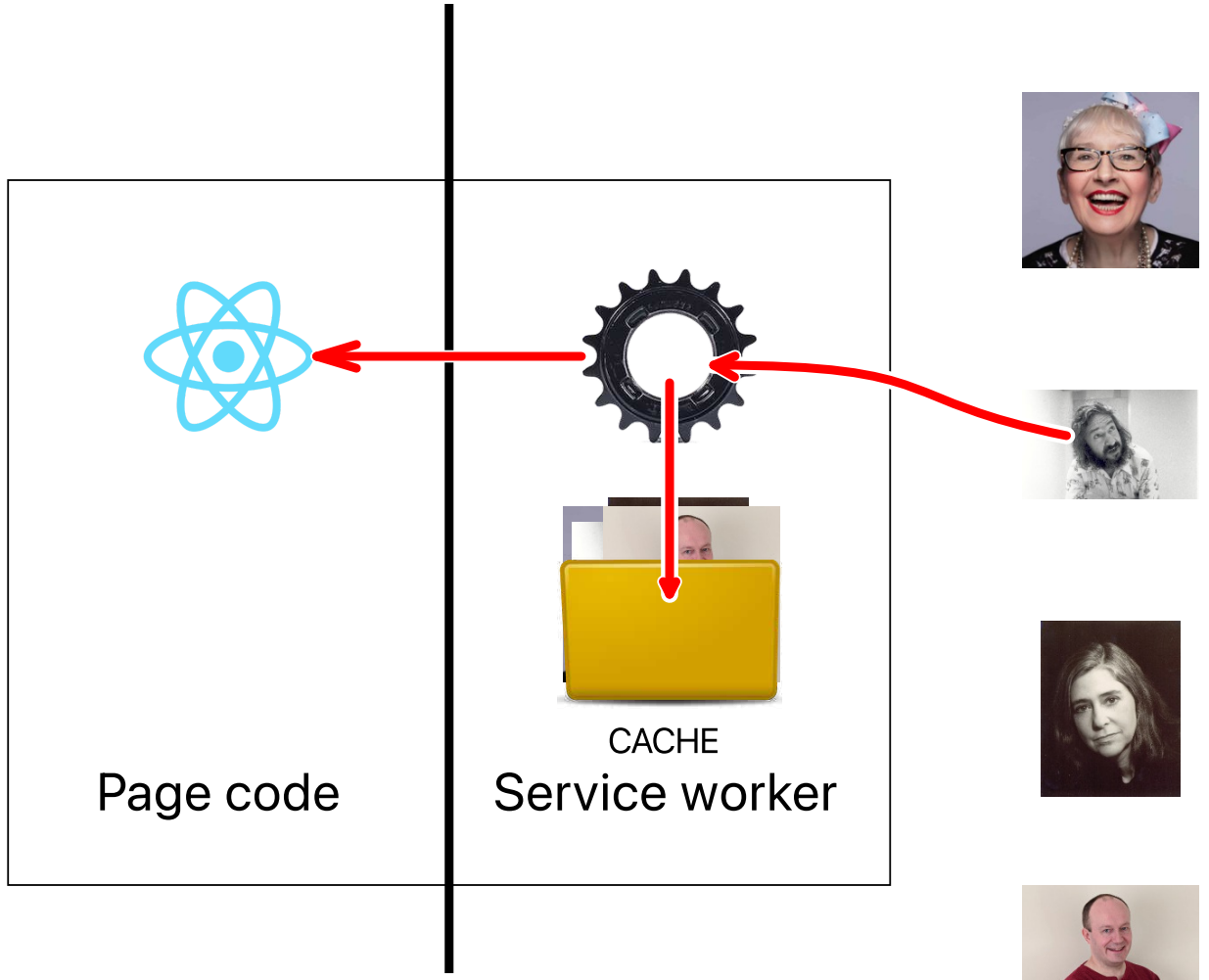
StaleWhileRevalidate



StaleWhileRevalidate



StaleWhileRevalidate



Sending data when offline? (recipe 11.6)

Sending data when offline? (recipe 11.6)

Sending data when offline? (recipe 11.6)

- When people are offline they might want to do more than read

Sending data when offline? (recipe 11.6)

- When people are offline they might want to do more than read
- They might want to send changes to the server

Sending data when offline? (recipe 11.6)

- When people are offline they might want to do more than read
- They might want to send changes to the server
- But how can that work when they have no connection?

Typical code to send data

```
const sendData = () => {
        const options = {
            method: 'POST',
            body: JSON.stringify({timeIs: new Date()}),
            headers: {
                'Content-Type': 'application/json'
            }
        };
        fetch('/endpoint', options)
    };
```

Use background-sync in the server-worker

```
import { registerRoute } from 'workbox-routing';
import {NetworkOnly, StaleWhileRevalidate} from 'workbox-strategies';
import {BackgroundSyncPlugin} from "workbox-background-sync";

// All the other service-worker code...
```

Use background-sync in the server-worker

```
import { registerRoute } from 'workbox-routing';
import {NetworkOnly, StaleWhileRevalidate} from 'workbox-strategies';
import {BackgroundSyncPlugin} from "workbox-background-sync";

// All the other service-worker code...

registerRoute(
    /\/endpoint/,
    new NetworkOnly({
        plugins: [new BackgroundSyncPlugin(
            'endPointQueue1', {
            maxRetentionTime: 24 * 60
        })]
    }),
    'POST'
);
```

Get the code!

Get the code!

- All of the code from today:

Get the code!

- All of the code from today:
- https://tinyurl.com/rcookbook1

Get the code!

- All of the code from today:
- https://tinyurl.com/rcookbook1
- All of the code from the book:

Get the code!

- All of the code from today:
- https://tinyurl.com/rcookbook1
- All of the code from the book:
- https://tinyurl.com/rcookbook2

Get the code!

- All of the code from today:
- https://tinyurl.com/rcookbook1
- All of the code from the book:
- https://tinyurl.com/rcookbook2
- And get the book!

Get the code!

- All of the code from today:
- https://tinyurl.com/rcookbook1
- All of the code from the book:
- https://tinyurl.com/rcookbook2
- And get the book!
- https://tinyurl.com/rcookbook3


O'REILLY®