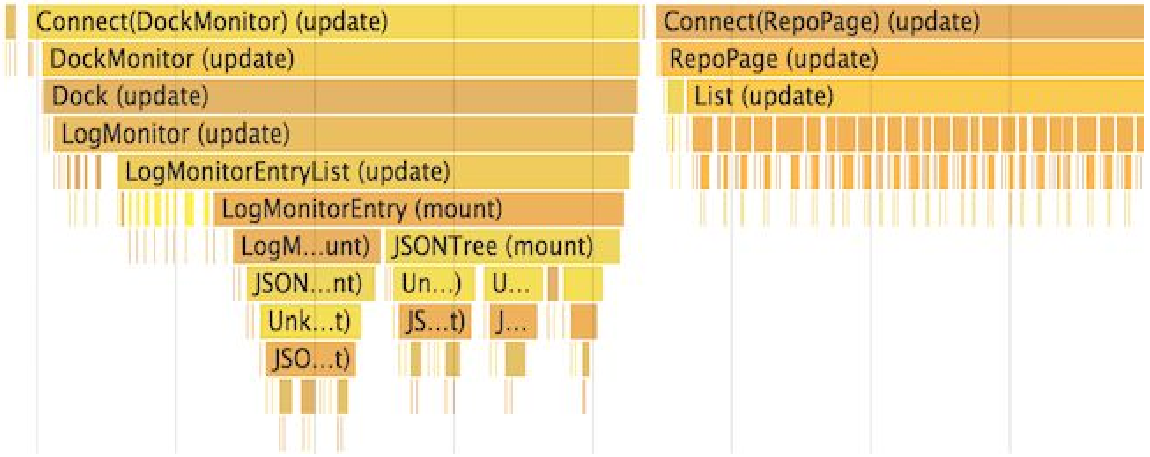
Improving Performance of Your React Components

Mallory Bulkley

@mallorybulkley

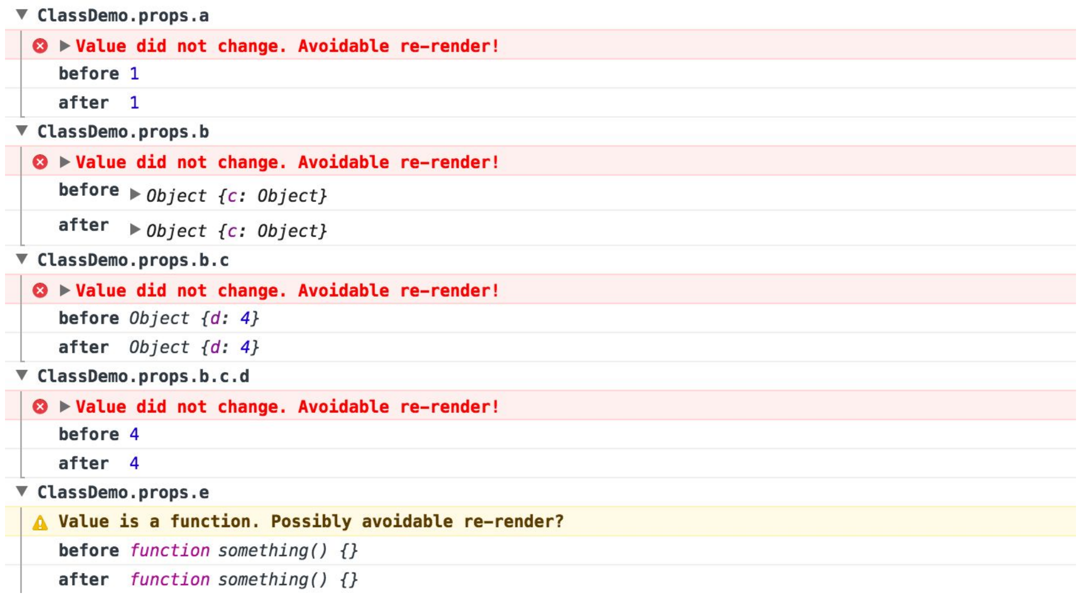
Finding the culprits

- Dev tools Performance tab



Finding the culprits

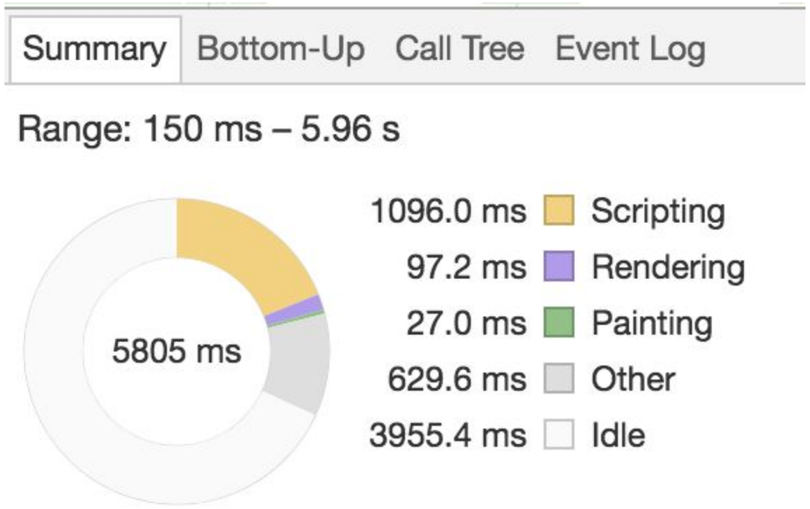
- `why-did-you-update`



```
▼ ClassDemo.props.a
  ⊗ ▶ Value did not change. Avoidable re-render!
    before 1
    after  1
▼ ClassDemo.props.b
  ⊗ ▶ Value did not change. Avoidable re-render!
    before ▶ Object {c: Object}
    after  ▶ Object {c: Object}
▼ ClassDemo.props.b.c
  ⊗ ▶ Value did not change. Avoidable re-render!
    before Object {d: 4}
    after  Object {d: 4}
▼ ClassDemo.props.b.c.d
  ⊗ ▶ Value did not change. Avoidable re-render!
    before 4
    after  4
▼ ClassDemo.props.e
  ⚠ Value is a function. Possibly avoidable re-render?
    before function something() {}
    after  function something() {}
```

Why are they slow?

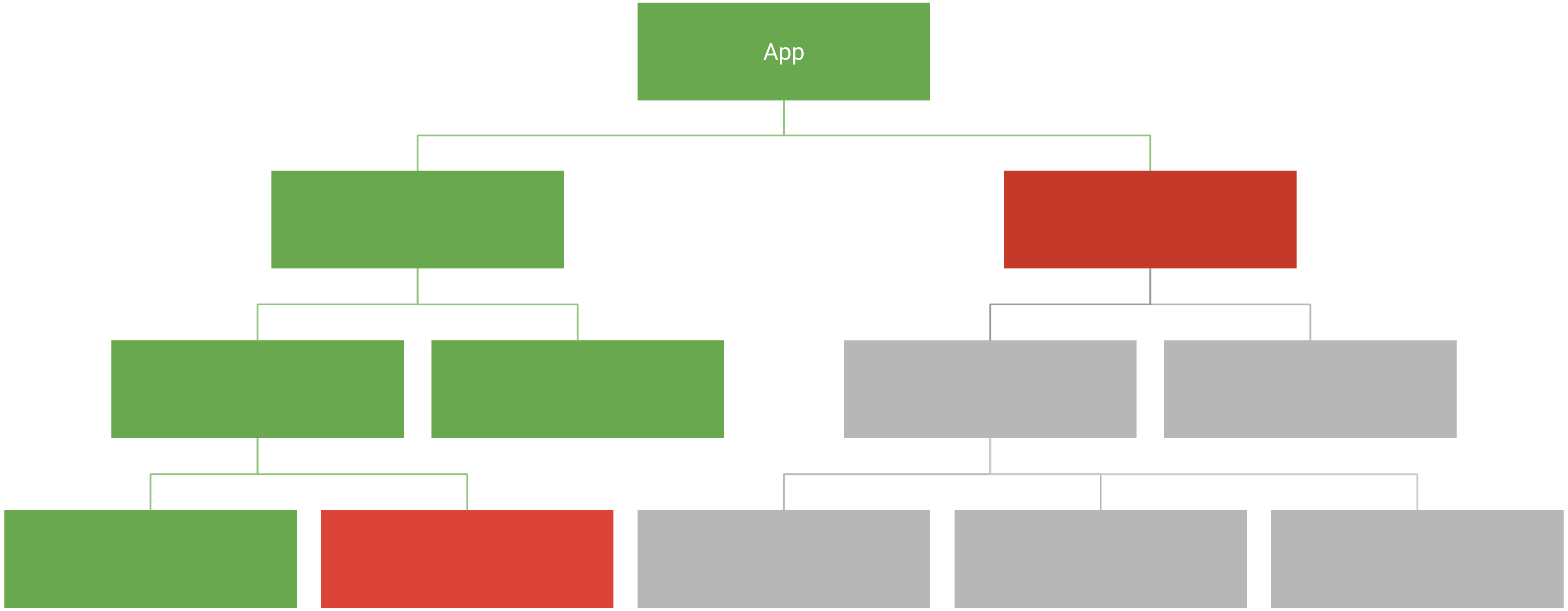
- Excess renders
- Excess reconciliations

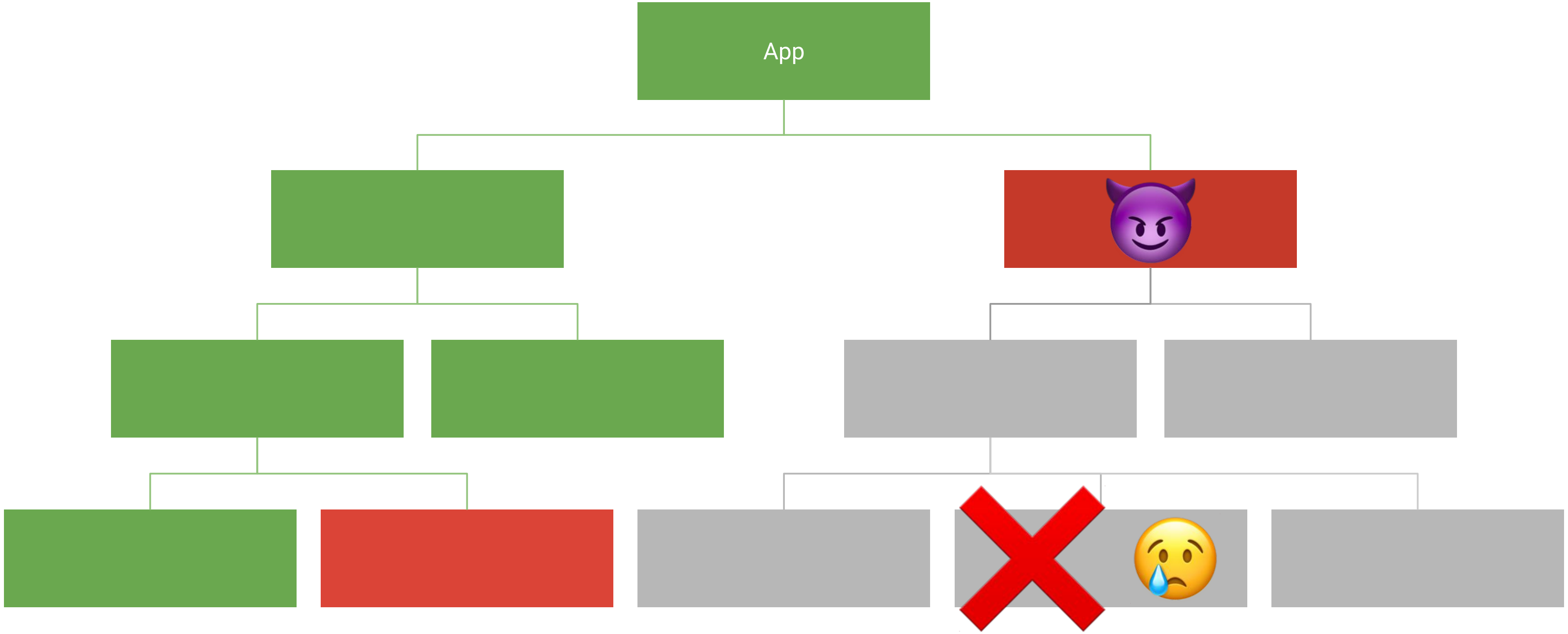


Range: 150 ms – 5.96 s

5805 ms

1096.0 ms	🟨	Scripting
97.2 ms	🟪	Rendering
27.0 ms	🟩	Painting
629.6 ms	▨	Other
3955.4 ms	☐	Idle


App


App

What causes excess reconciliations?

- Passing down unnecessary props

```
<Component {...rest} />
```

- Object literals defined in render

```
<Component style={{ margin: 0 }} />
```

- Anonymous functions defined in render

```
<Component onChange={(e) => this.handleChange(e)} />
```

How to reduce reconciliations

- Define constants and bind functions outside of render

😢

😊

```
<Component
  onChange={(e) => this.handleChange(e)}
/>
```

```
<Component
  onChange={this.handleChange}
/>
```

```
<Component style={{ margin: 0 }} />
```

```
const myStyles = { margin: 0 };
<Component style={myStyles} />
```

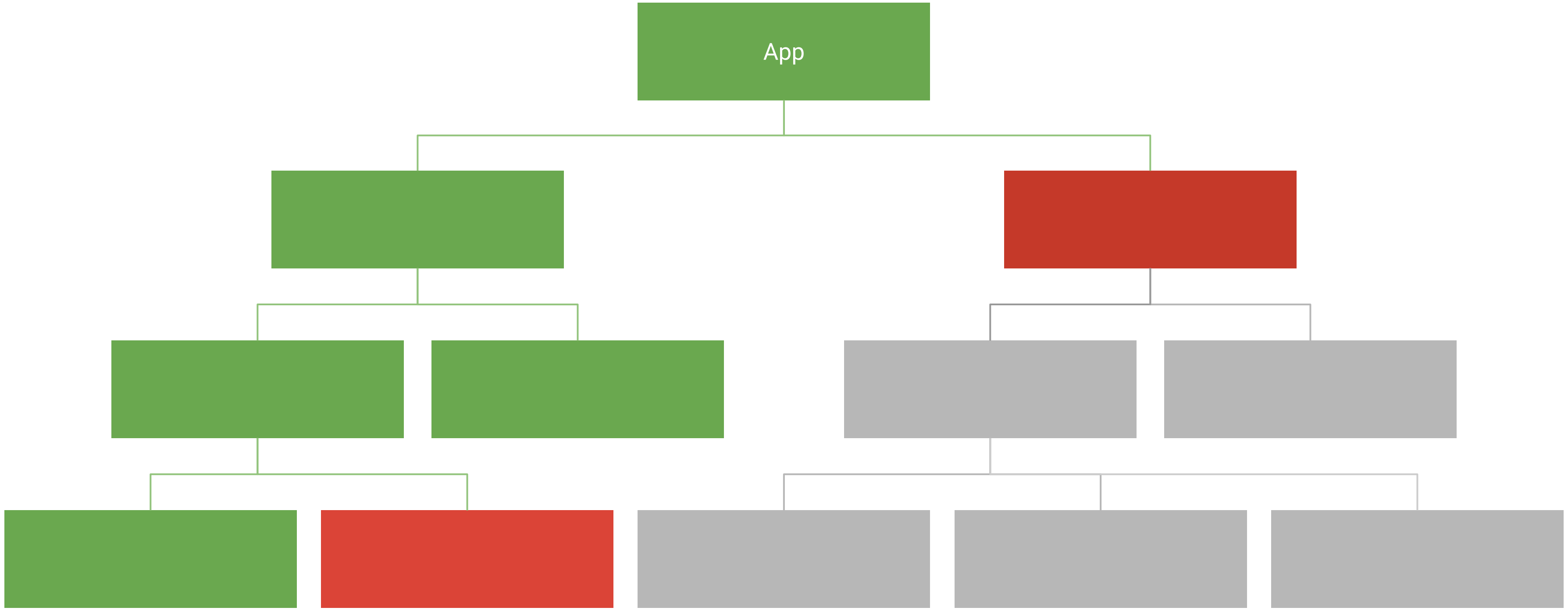
```
<Component {...rest} />
```

```
<Component necessary={props} />
```

How to reduce reconciliations

- shouldComponentUpdate()

```
shouldComponentUpdate(nextProps, nextState) {
    // return true or false
}
```


App

How to reduce reconciliations

- PureComponent

```
MyComponent extends React.PureComponent {

shouldComponentUpdate(nextProps, nextState) {
    return this.props !== nextProps
    && this.state !== nextState
}
```

Alternative for Functional Components

- recompose
  - pure
  - updateOnlyForKeys

```
import { lifecycle, compose, pure } from 'recompose';

…

export default compose(withLifecycle, pure)(Component);
```

In Summary

- Use tools to find problematic components
  - why-did-you-update
  - Performance dev tools
- Reduce unnecessary reconciliations
  - shouldComponentUpdate()
  - PureComponent
  - Define constants and functions outside of render()
- Keep your components rendering based only on (preferably immutable) state/props