# React Reconciliation

## How ReactJS renders your application

Presented by Jim Sproch

April 11, 2016 @ PHILLY ETE

# What is React?

A Javascript library for building user interfaces

The “**V**” in **MVC**

# Today’s Topics

## React's Design

- Re-render the whole app on every update

## Virtual DOM

- An implementation detail / performance hack

## Reconciliation

- The "magic" that makes React work

# React's Design

## React Components

```
function UserBoxComponent(props) {
  return (
    <div>
      <img src={props.user.image} />
      <span>{props.user.name}</span>
    </div>
  );
}
```

Idempotent functions that take in the current state of your application and return the UI of your application.

# React's Design

## React Components

```
function UserBoxComponent(props) {
  return (
    React.createElement('div', {},
      [React.createElement('img',{
        src: props.user.image}),
      React.createElement('span',
        {},props.user.name)]
  ));
}
```

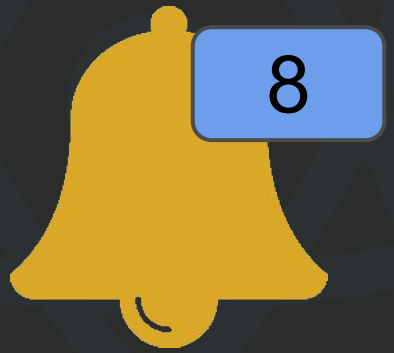
# React's Design

## React Components

```
function UserBoxComponent(props) {
  return (
    <div>
      <img src={props.user.image} />
      <span>{props.user.name}</span>
    </div>
  );
}
```

# React's Design

# React's Design



# React's Design

# React's Design

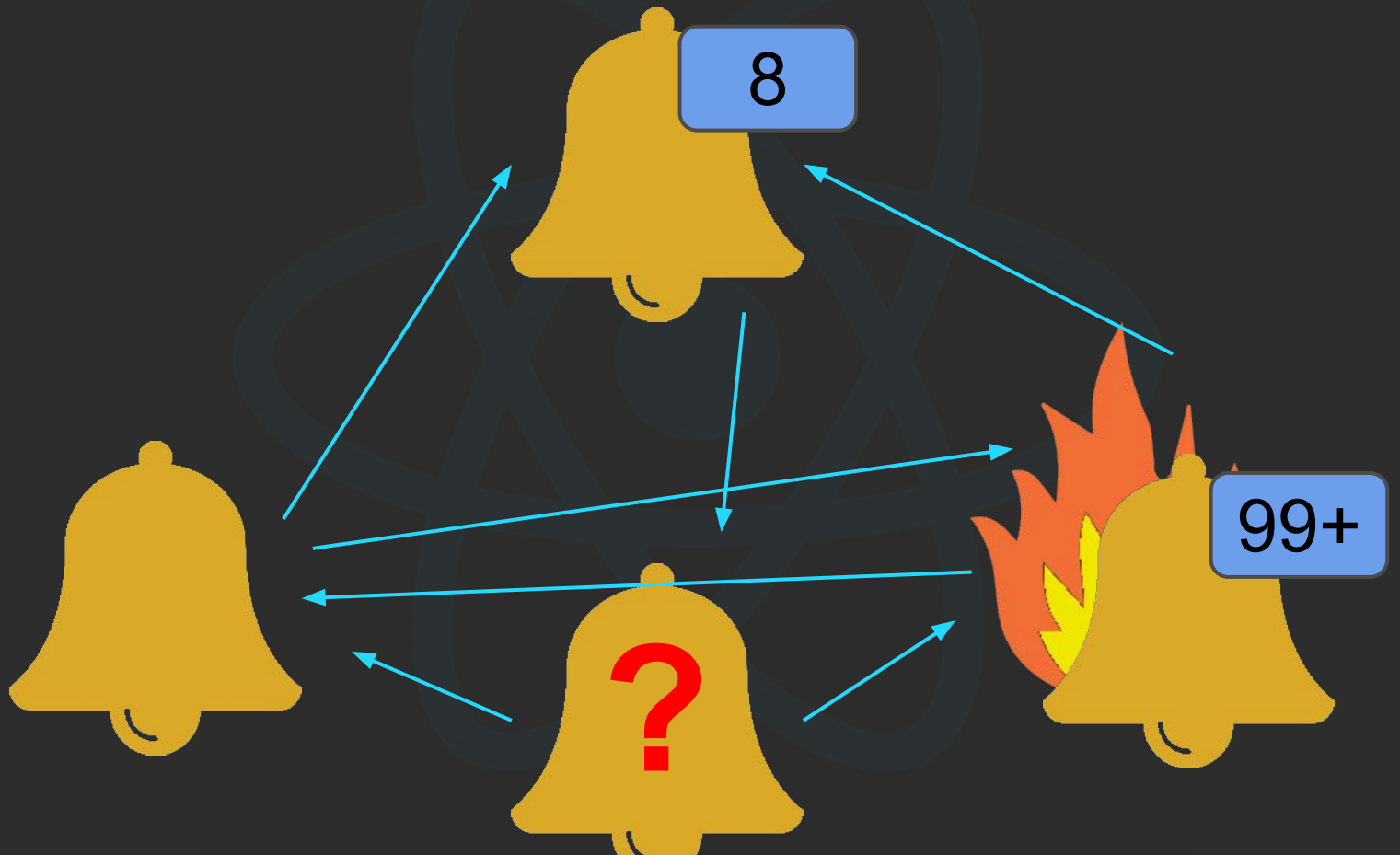
# React's Design

# React's Design

# React's Design

```
if (count > 99) {
  if (!hasFire()) { addFire(); }
} else {
  if (hasFire()) { removeFire(); }}
if (count === 0) {
  if (hasBadge()) { removeBadge(); }
  return;}
if (!hasBadge()) { addBadge(); }
text = count > 99 ? '99+' : count.toString();
getBadge().setText(text);
```

# React's Design

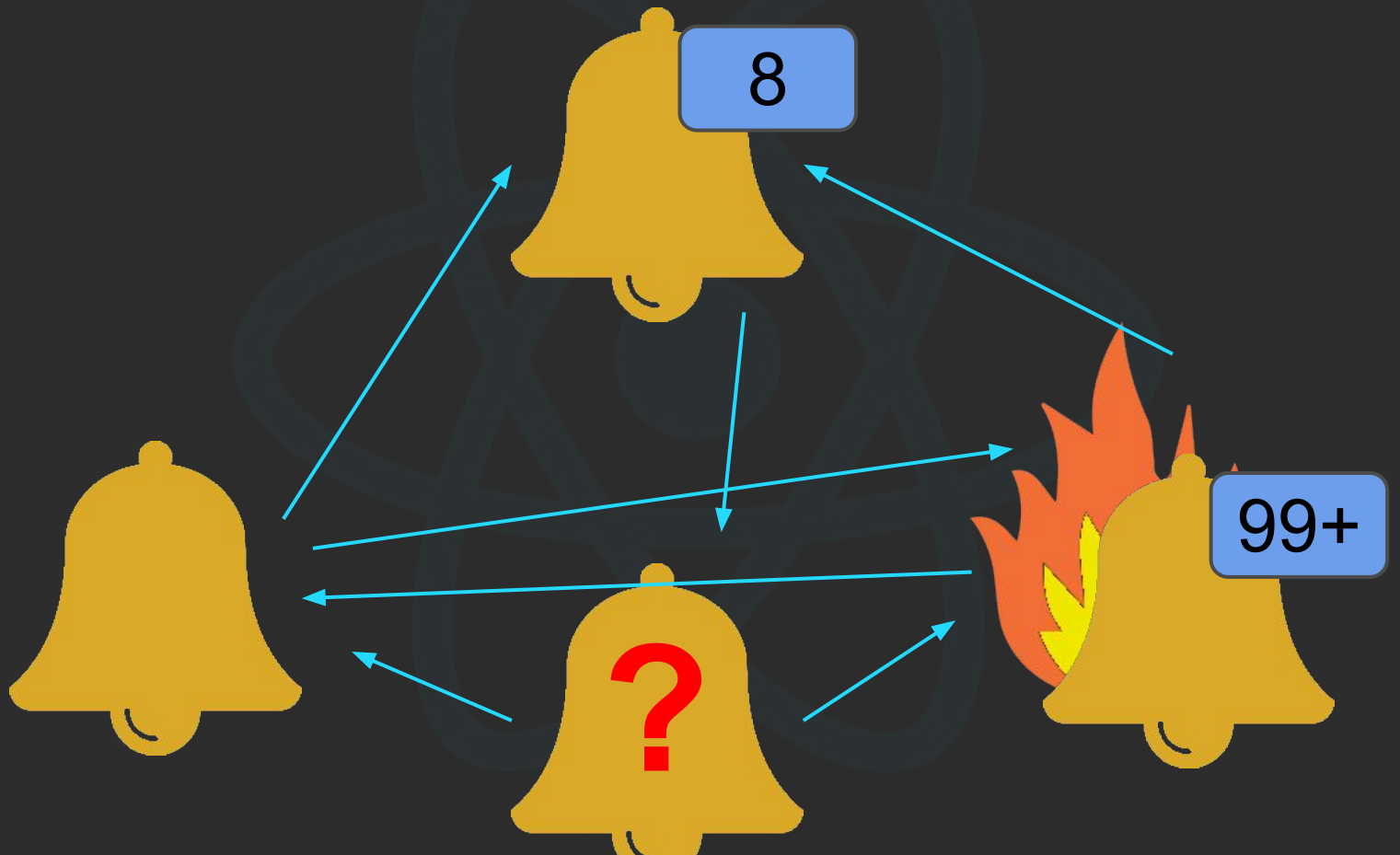


# React's Design

"Controlling complexity is the essence of computer programming."

— Brian Kernighan

# React's Design



# React's Design

State Transition Complexity

$O(N(N-1)) \Rightarrow O(N^2-N)$

# React's Design

Mutation is hard

**Let's not do mutation!**

# React's Design

```
text = count > 99 ? '99+' : count.toString();

<bell>
  <if test={count > 99}>
    <img src="fire.png" class="background" />
  </if>
  <if test={count > 0}>
    <badge count={text} class="foreground" />
  </if>
</bell>
```

# React's Design

Rebuild the whole dom, for every change

```
function SimpleComponent(props) {
   return <button onClick={props.bar} />;
}
```

Sounds expensive

# Virtual DOM

## Virtual DOM ≠ Shadow DOM

# Virtual DOM

## Fast / Light-Weight Nodes

- Created and thrown away with every render

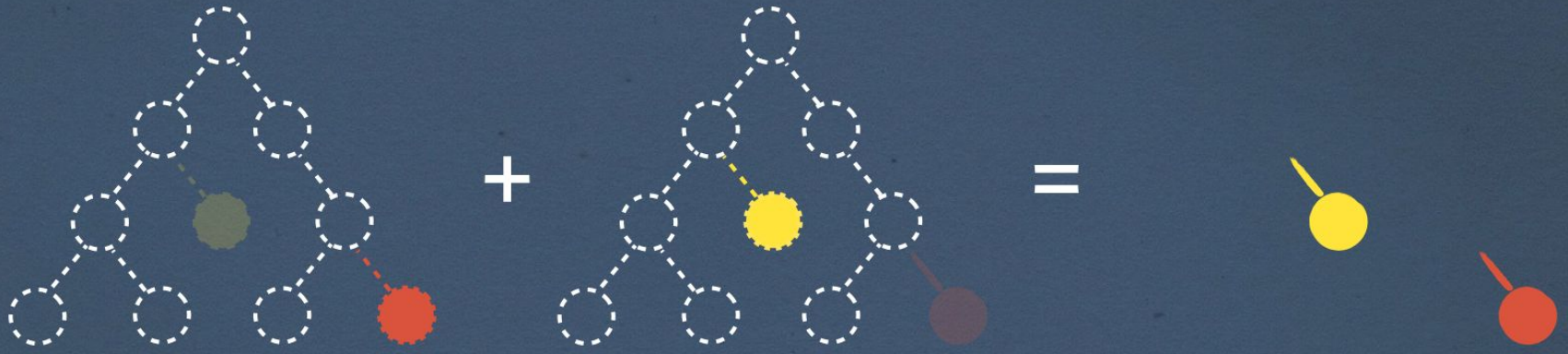
## Avoid Layout Thrash

- Reading from the DOM can force a reflow

## Queue Updates

- Executed after reconciliation

# Virtual DOM




Virtual DOM t=1 Virtual DOM t=2 Mutations

The diff algorithm generates a list of DOM mutations, the same way version controls output text mutations

# Reconciliation

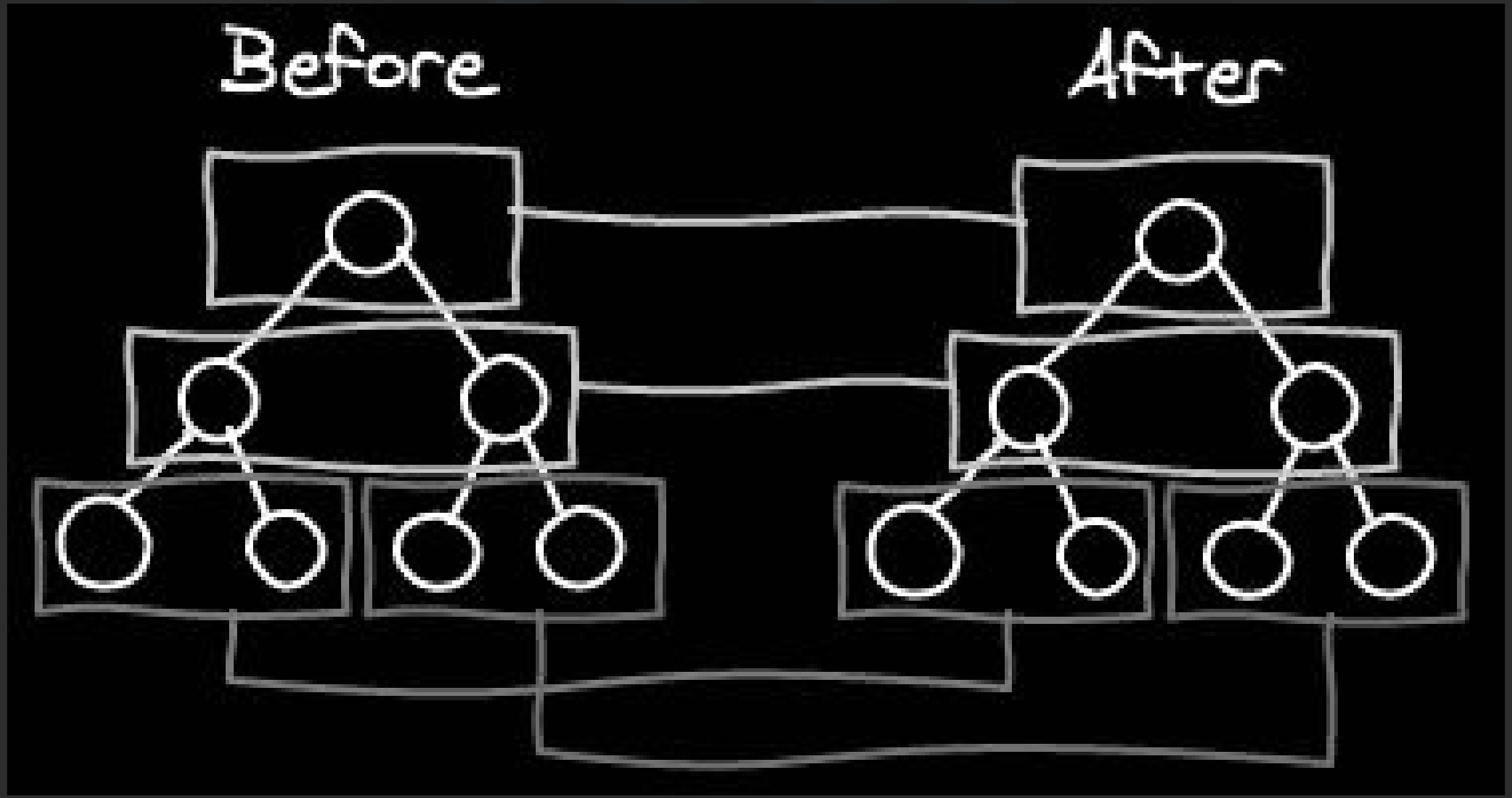
The process of updating your UI to match your application state

# Reconciliation

Truly Minimal Diff: $O(N^3)$

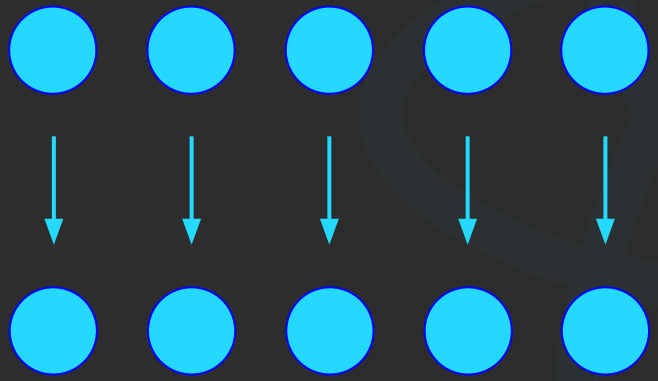
React Diff: $O(N)$

# Reconciliation

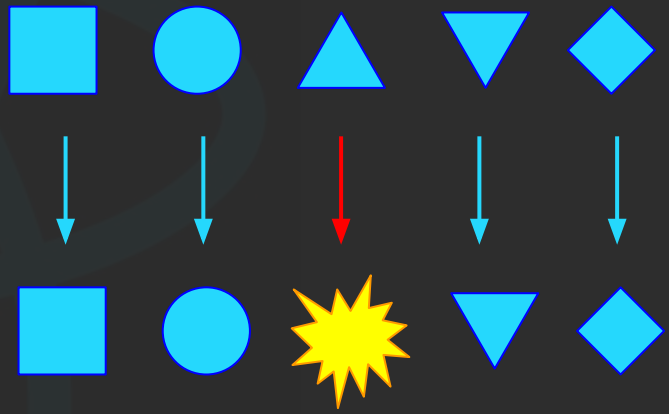


# Reconciliation

## Same



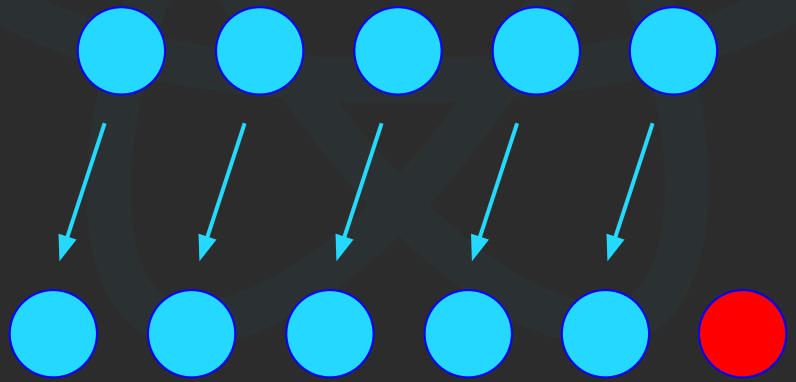
## Different



# Reconciliation

```
<Circle />
<Circle />
<Circle />
<Circle />
<Circle />
```

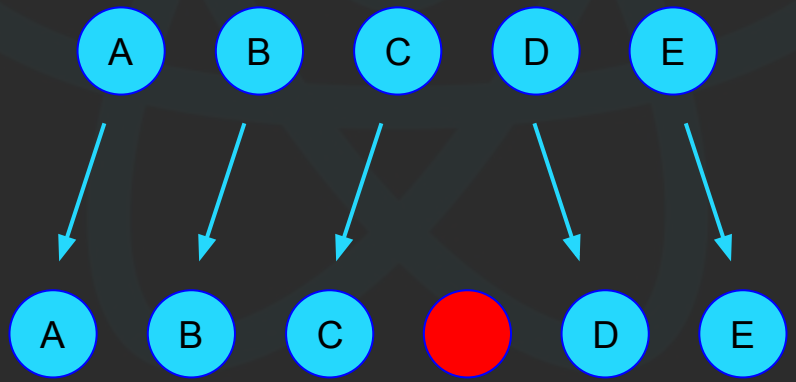
```
<Circle />
<Circle />
<Circle />
<Circle />
<Circle />
<Circle />
```



# Reconciliation

```
<Circle key="A" />
<Circle key="B" />
<Circle key="C" />
<Circle key="D" />
<Circle key="E" />
```

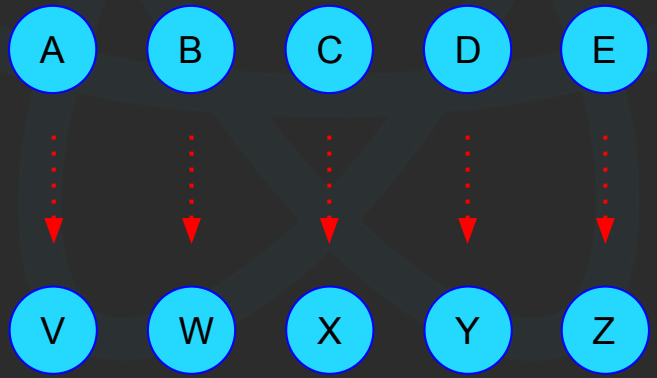
```
<Circle key="A" />
<Circle key="B" />
<Circle key="C" />
<Circle key="XX" />
<Circle key="D" />
<Circle key="E" />
```



# Reconciliation

```
<Circle key="A" />
<Circle key="B" />
<Circle key="C" />
<Circle key="D" />
<Circle key="E" />
```

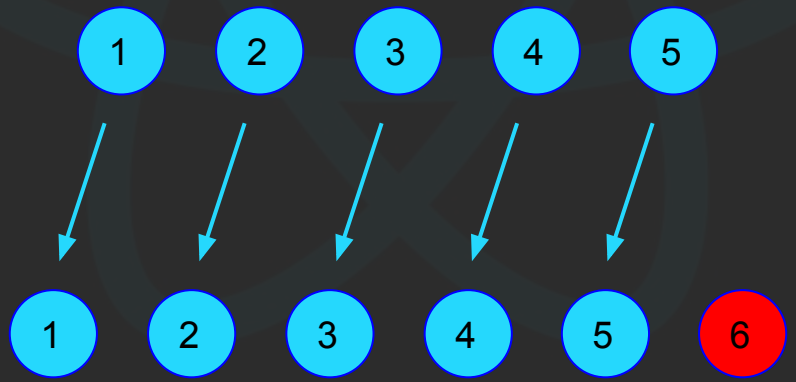
```
<Circle key="V" />
<Circle key="W" />
<Circle key="X" />
<Circle key="Y" />
<Circle key="Z" />
```



# Reconciliation

```
<Circle key="1" />
<Circle key="2" />
<Circle key="3" />
<Circle key="4" />
<Circle key="5" />
```

```
<Circle key="1" />
<Circle key="2" />
<Circle key="3" />
<Circle key="4" />
<Circle key="5" />
<Circle key="6" />
```



# Reconciliation

Keys are about identity
(in addition to performance)

# Reconciliation

## Sub-tree Rendering

- setState

## Selective Sub-tree Rendering

- shouldComponentUpdate

## Batched Updates

- unstable_batchedUpdates

# Today's Topics

## React's Design

- Simple component model for rendering the view layer
- Conceptually re-render the whole app on every update
- Avoid $O(N^2\text{-}N)$ state transition complexity (avoid writing transitions)

## Virtual DOM

- Light-weight descriptors which specify the desired render tree

## Reconciliation

- Calculates a minimal set of changes to apply to the DOM

# Thank you

facebook.github.io/react
Jim Sproch (@jimfb)