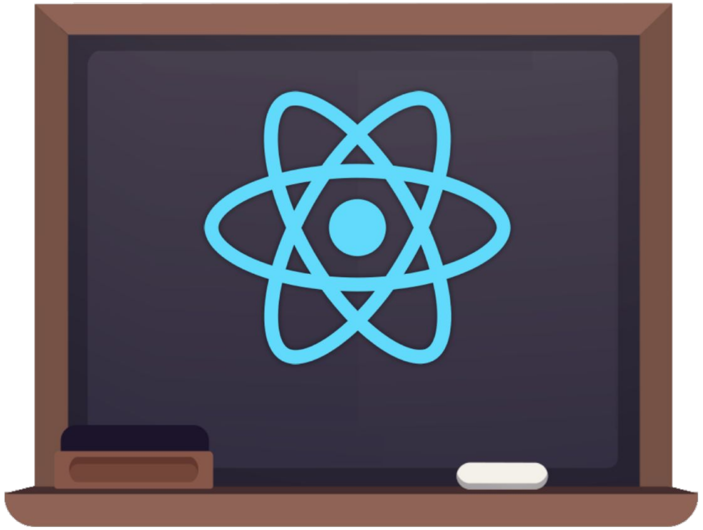


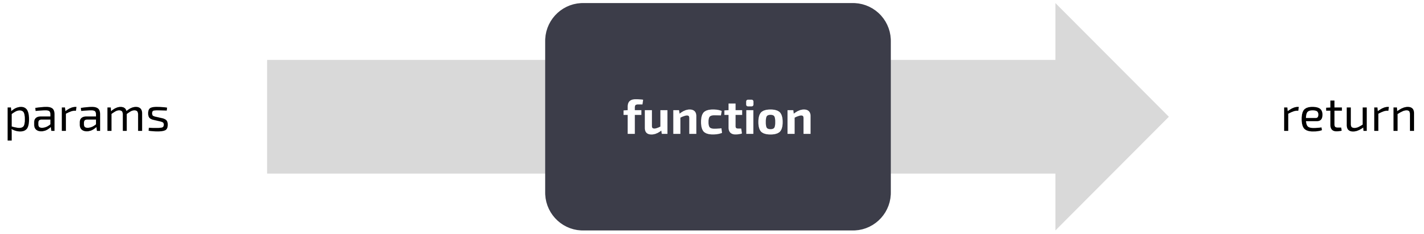
4. Props & state

Introduction to React

Index

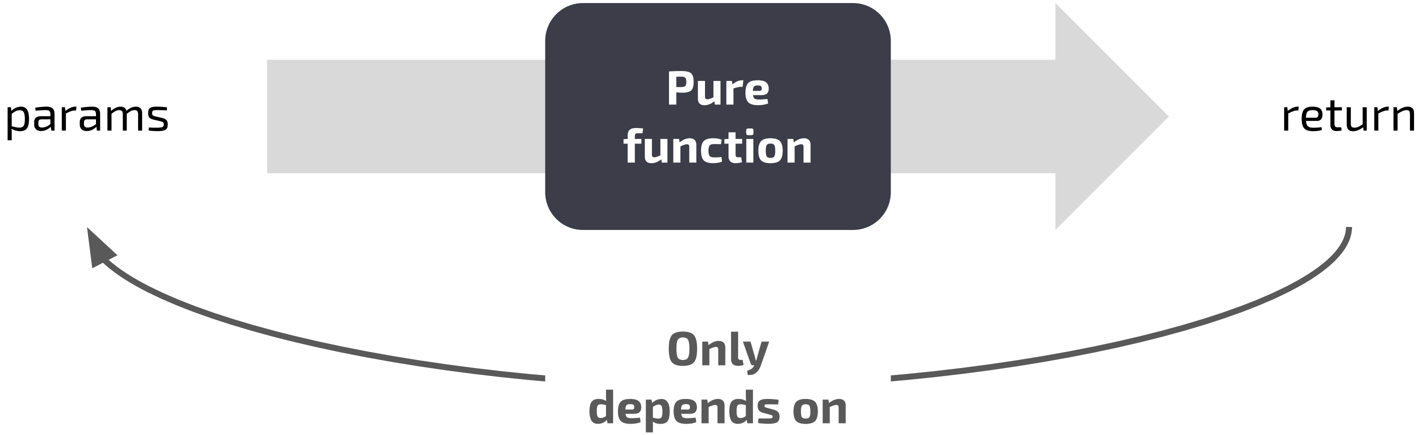
Index

Props - Functions flow

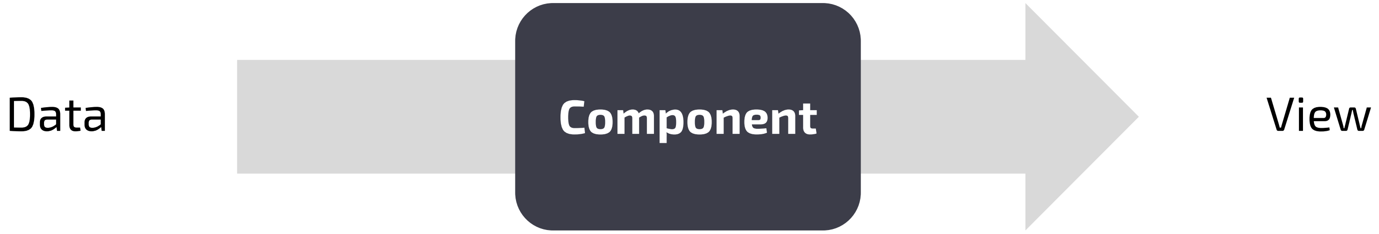


Props - Pure functions

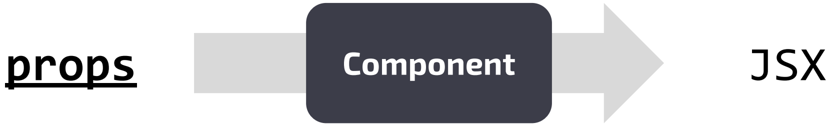
- Deterministic
- Easy to test
- Easy to optimize



Props - Components flow



Props - Components flow



```
const Header = props => <h1> {props.label} </h1>;
```

Props - Component example

Declaration

```
const Header = props =>
 <h1> {props.label} </h1>;
```

Usage

```
<Header label="My wishlist"/>
```

Props - Functional vs Classy

Functional component

```
const Header = props =>
  <h1> {props.label} </h1>;
```

~

Class component

```
class Header extends Component {
  render() {
    return
      <h1>{this.props.label}</h1>;
  }
}
```

Props - Functional vs Classy (destructuring)

Functional component

```
const Header = ({ label }) =>
  <h1> {label} </h1>;
```

~

Class component

```
class Header extends Component {
  render() {
    const { label } = this.props;
    return <h1>{label}</h1>;
  }
}
```

Props - Functional vs Classy (destructuring)

Functional component

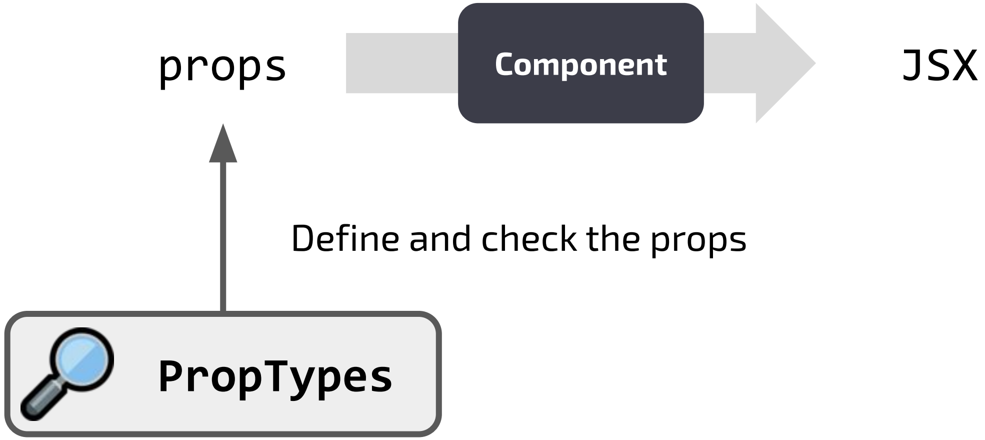
```
const Header = ({ label }) =>
  <h1> {label} </h1>;
```

~

Class component

```
class Header extends Component {
  render() {
    const { label } = this.props;
    return <h1>{label}</h1>;
  }
}
```

Props – Prop Types



Props – Prop Types

```
import PropTypes from 'prop-types';

const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
    label: PropTypes.string.isRequired
}
```

Props – Prop Types

```
import PropTypes from 'prop-types';

const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
    label: PropTypes.string.isRequired
}
```

!

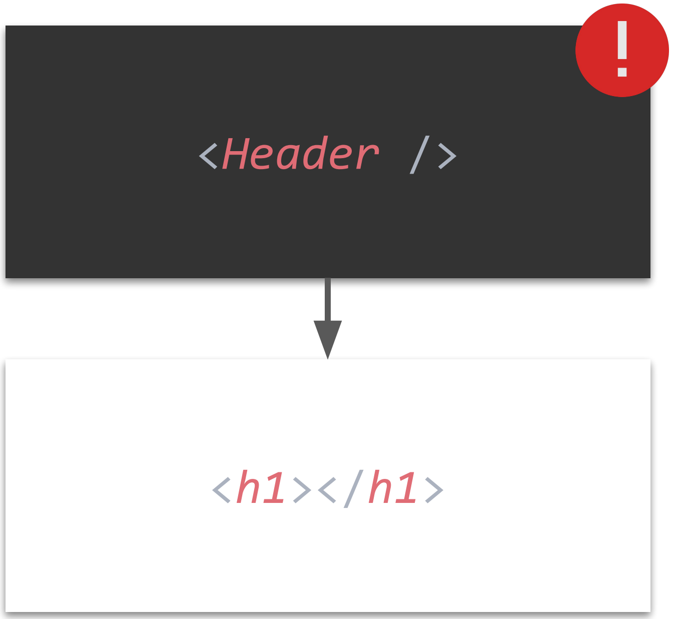
```
<Header label={3} />
```

```
<h1>3</h1>
```

OpenWebinars

Props – Prop Types

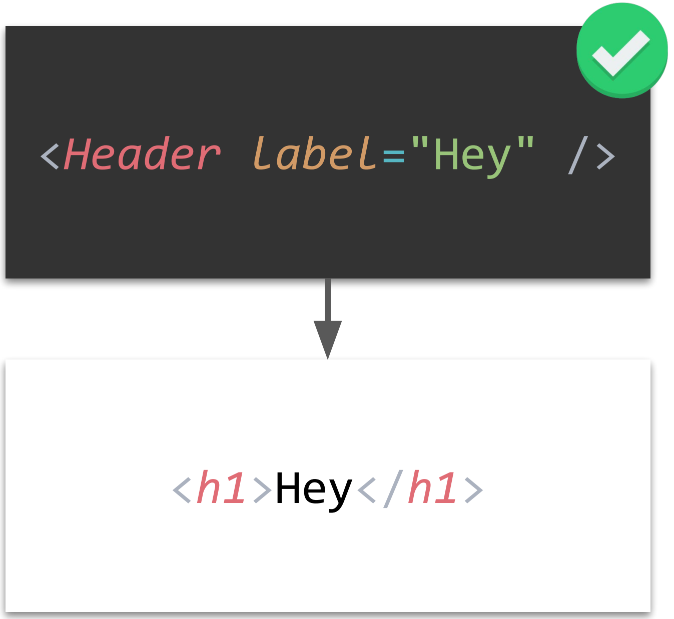
```
import PropTypes from 'prop-types';

const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
    label: PropTypes.string.isRequired
}
```



OpenWebinars

Props - Prop Types

```
import PropTypes from 'prop-types';

const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
   label: PropTypes.string.isRequired
}
```



```
<Header label="Hey" />
```

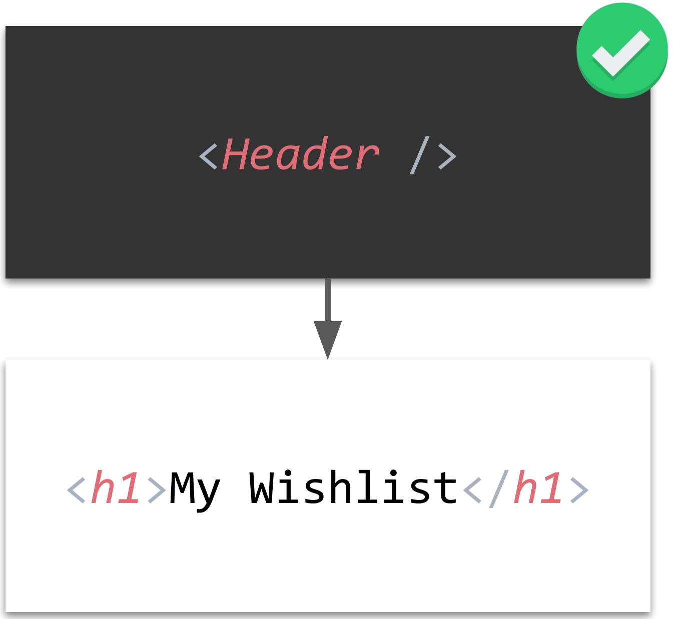
```
<h1>Hey</h1>
```

Props – Prop Types

```
import PropTypes from 'prop-types';
const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
    label: PropTypes.string
}
Header.defaultProps = {
    label: 'My Wishlist'
}
```

Props - Prop Types

```
import PropTypes from 'prop-types';
const Header = ({label}) => <h1>{label}</h1>

Header.propTypes = {
   label: PropTypes.string.isRequired
}
Header.defaultProps = {
   label: 'My Wishlist'
}
```



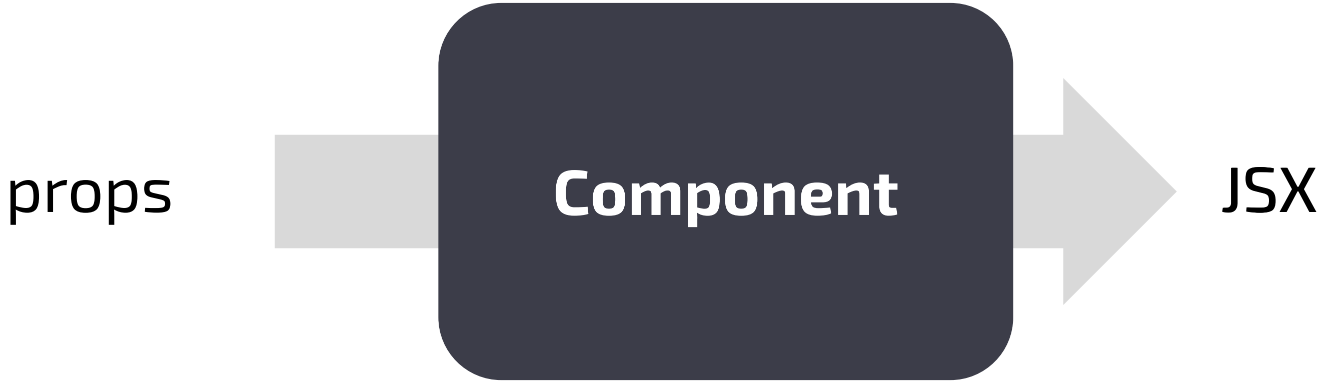
```
<Header />
```

```
<h1>My Wishlist</h1>
```

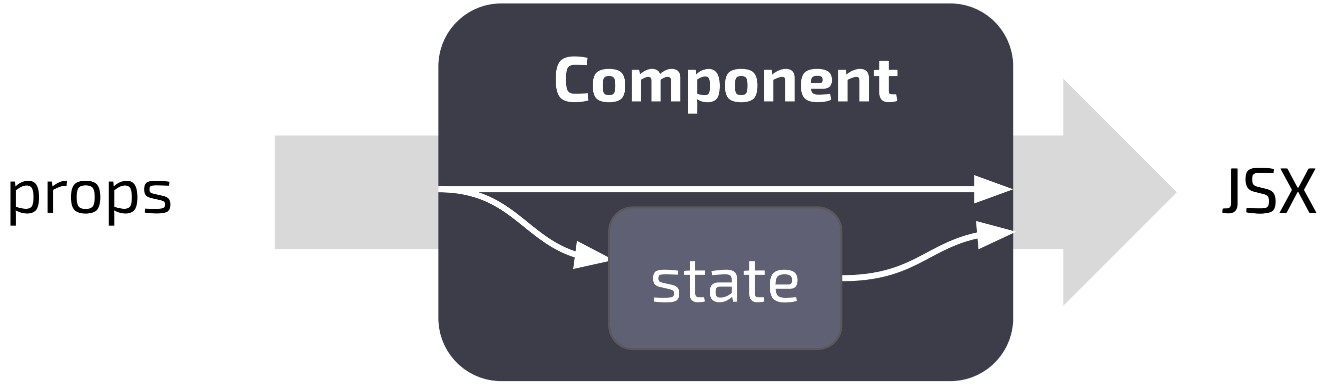
OpenWebinars

Index

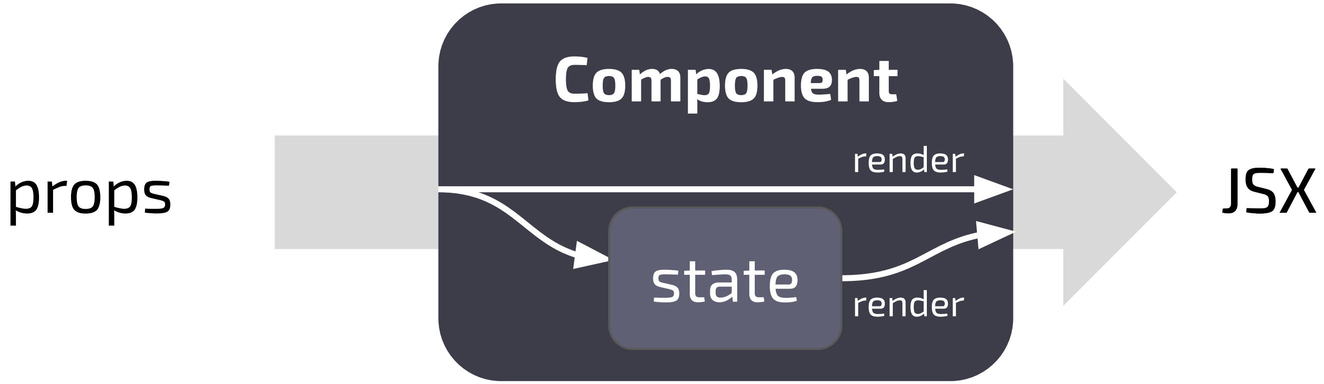
State – Components flow



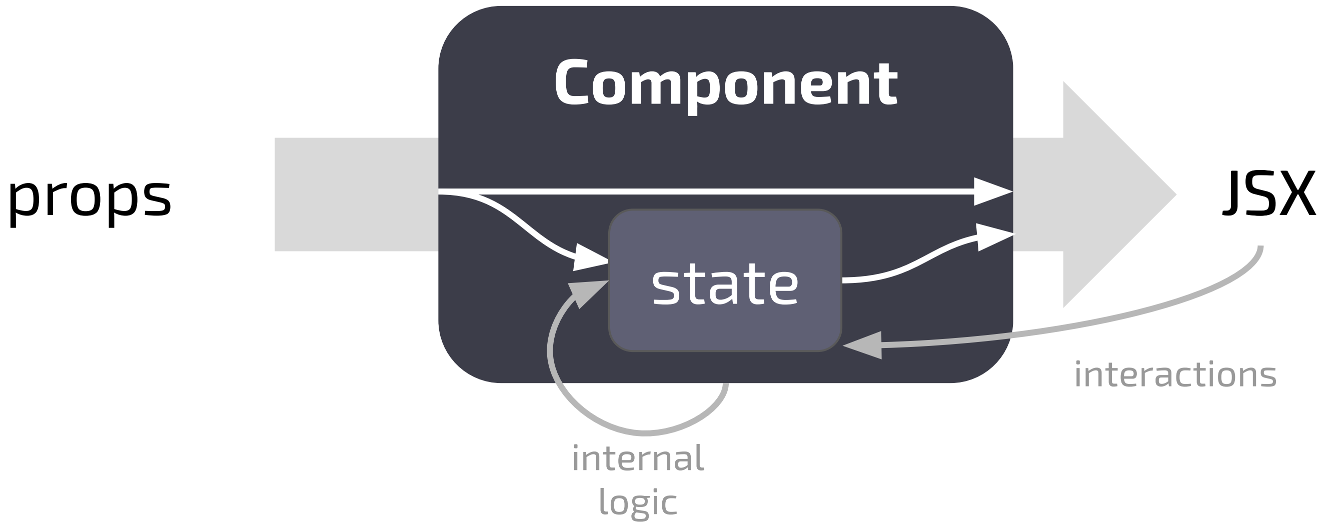
State - Components flow



State – Components flow



State – Components flow

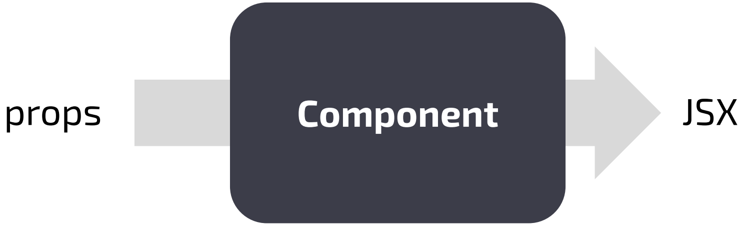


State – useState Hook

```
[value, setValue] = useState(defaultValue)
```

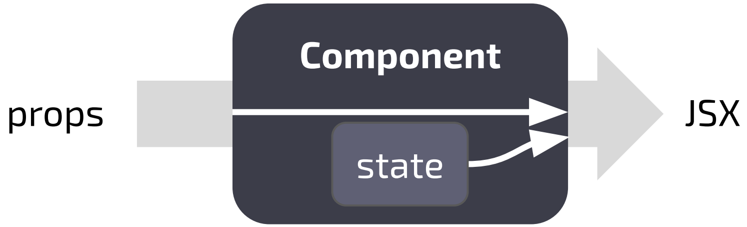
State – Components flow

```
const TodoItem = ({ label }) => {
  return (
    <p>
      {label}
    </p>
  );
};
```



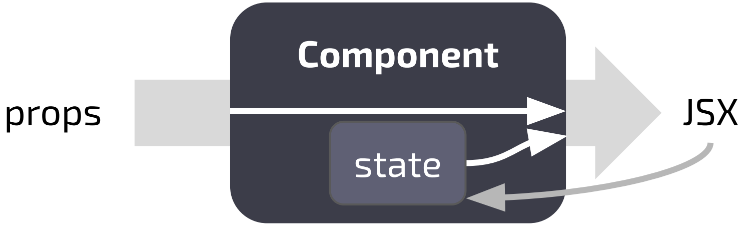
🌟 State – Components flow

```
const TodoItem = ({ label }) => {
  const [checked] = useState(false);
  return (
    <p>
      {checked ? '✓' : 'X'} {label}
    </p>
  );
};
```



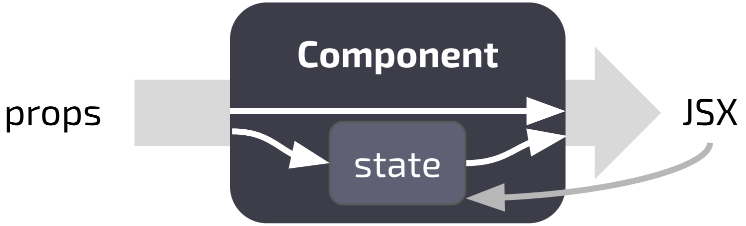
State – useState Hook

```
const TodoItem = ({ label }) => {
  const [checked, setChecked] = useState(false);
  return (
    <p onClick={() => setChecked(!checked)}>
      {checked ? '✓' : 'X'} {label}
    </p>
  );
};
```



State – useState Hook

```
const TodoItem = ({ label, defChk}) => {
  const [checked, setChecked] = useState(defChk);
  return (
    <p onClick={() => setChecked(!checked)}>
      {checked ? '✓' : 'X'} {label}
    </p>
  );
};
```



OpenWebinars

State – Functional vs Classy

Functional component

```
const TodoItem = ({ label, defChk }) => {
 const [checked, setChecked] = useState(defChk);
 return (
   <p onClick={() => setChecked(!checked)}>
     {checked ? '✓' : 'X'} {label}
   </p>
 );
};
```

Class component

```
class TodoItem extends Component {
 state = { checked: this.props.defChk };

 render() {
   const { checked } = this.state;
   const { label } = this.props;
   return (
     <p onClick={() =>
       this.setState({ checked: !checked })
     }>
       {checked ? '✓' : 'X'} {label}
     </p>
   );
 }
}
```

State – Functional vs Classy

Functional component

```
const TodoItem = ({ label, defChk }) => {
 const [checked, setChecked] = useState(defChk);
 return (
   <p onCLick={() => setChecked(!checked)}>
     {checked ? '✓' : 'X'} {label}
   </p>
 );
};
```

Class component

```
class TodoItem extends Component {
 state = { checked: this.props.defChk };

 render() {
   const { checked } = this.state;
   const { label } = this.props;
   return (
     <p onCLick={() =>
       this.setState({ checked: !checked })
     }>
       {checked ? '✓' : 'X'} {label}
     </p>
   );
 }
}
```

State – Functional vs Classy

Functional component

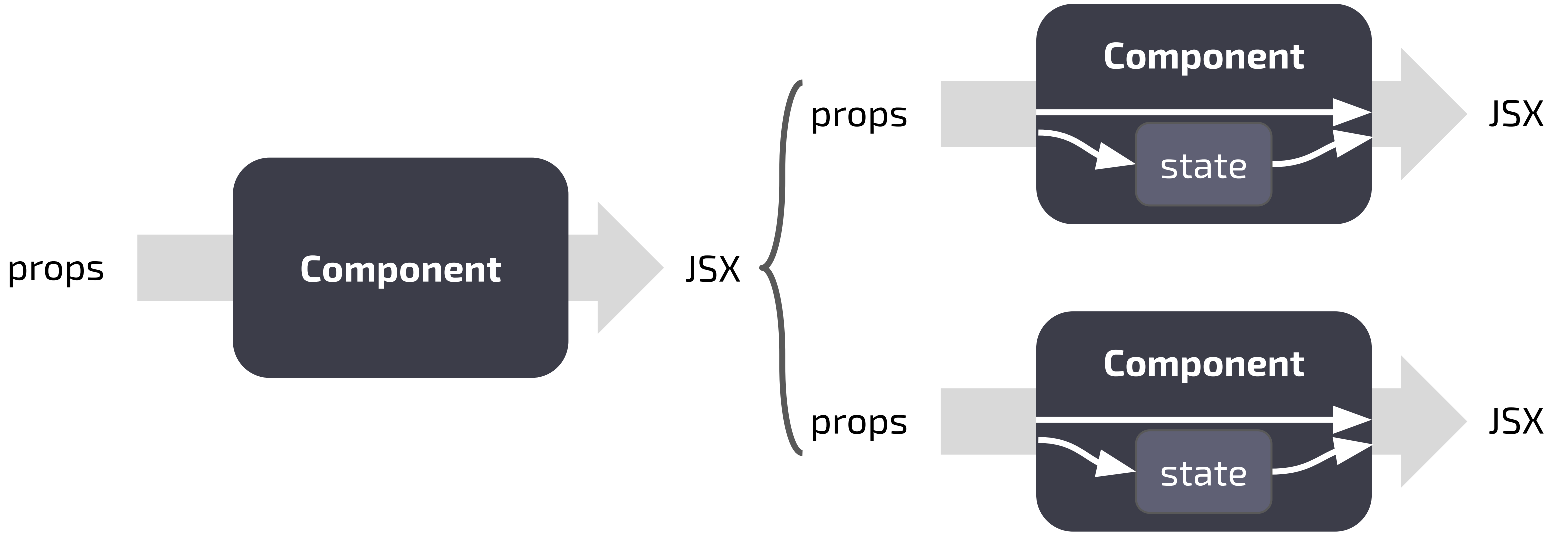
```
const TodoItem = ({ label, defChk }) => {
 const [checked, setChecked] = useState(defChk);
 return (
   <p onCLick={() => setChecked(!checked)}>
     {checked ? '✓' : 'X'} {label}
   </p>
 );
};
```

Class component

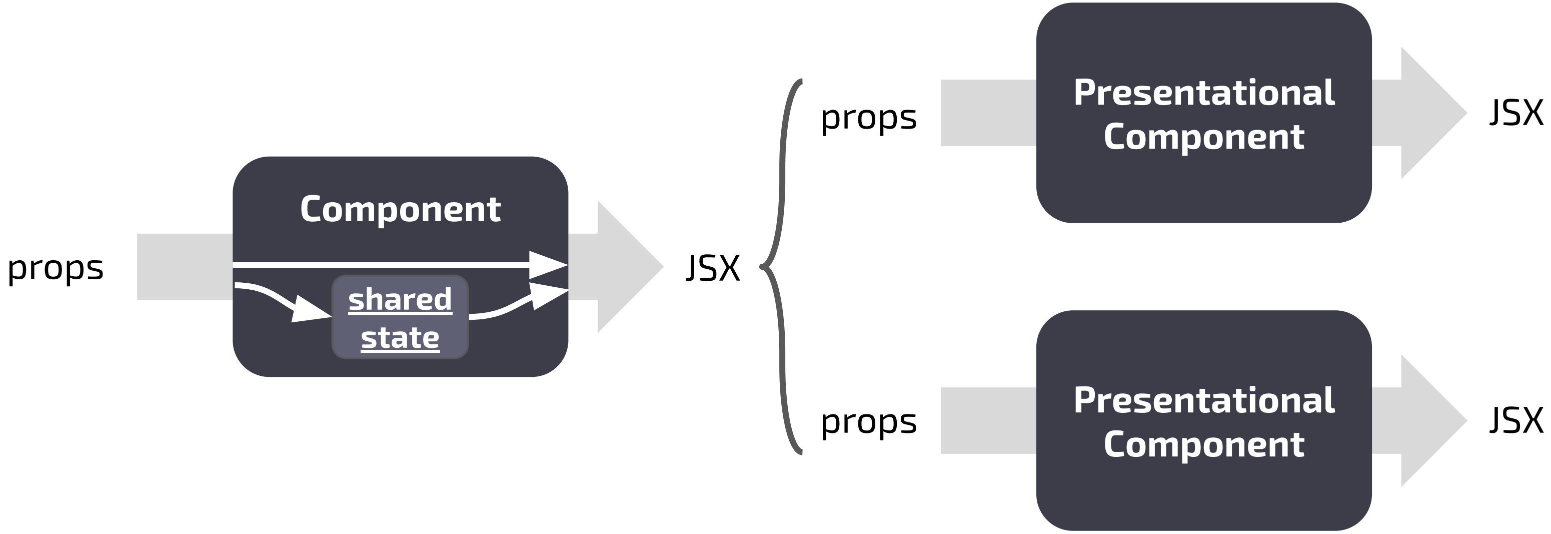
```
class TodoItem extends Component {
 state = { checked: this.props.defChk };

 render() {
   const { checked } = this.state;
   const { label } = this.props;
   return (
     <p onCLick={() =>
       this.setState({ checked: !checked })
     }>
       {checked ? '✓' : 'X'} {label}
     </p>
   );
 }
}
```

OpenWebinars

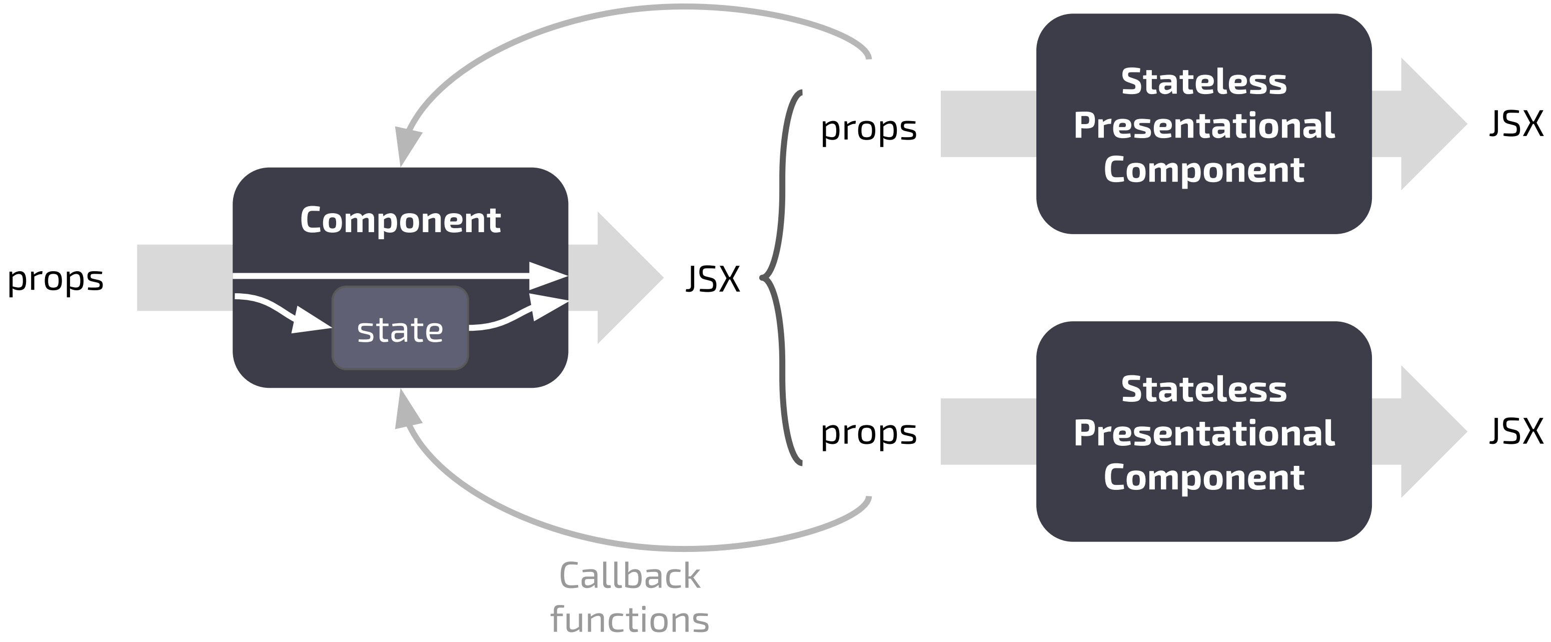
State - Delegated state



State - Delegated state



State - Delegated state


props
Component
state
JSX
props
Stateless Presentational Component
JSX
props
Stateless Presentational Component
JSX
Callback functions

State - Delegated state

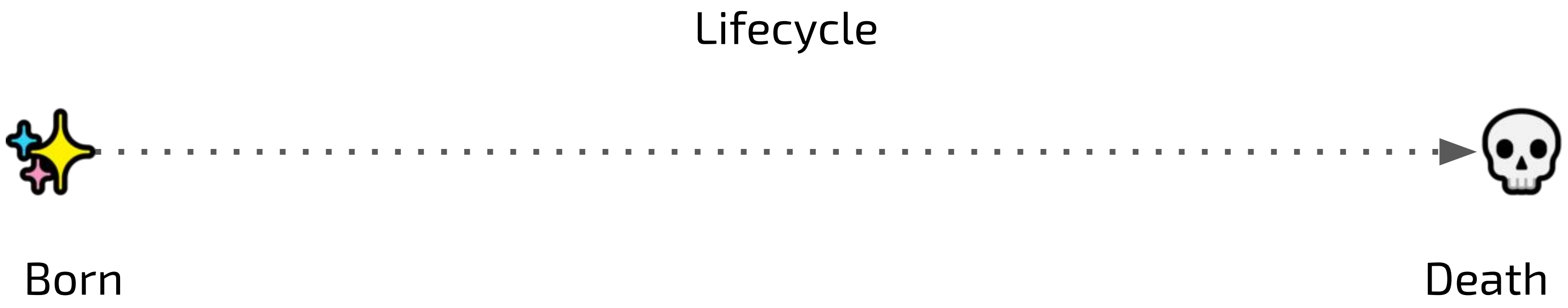
```
const Search = () => {
 const [search, setSearch] = useState('');
 return (
   <div>
     <SearchInput
       search={search}
       onSearchChange={setSearch}
     />
     <SearchDisplay
       search={search}
       onClear={() => setSearch('')}
     />
   </div>
 );
};
```

```
const SearchInput = ({search, onSearchChange}) => (
 <input
   value={search}
   onChange={e => onSearchChange(e.target.value)}
 />
);
```

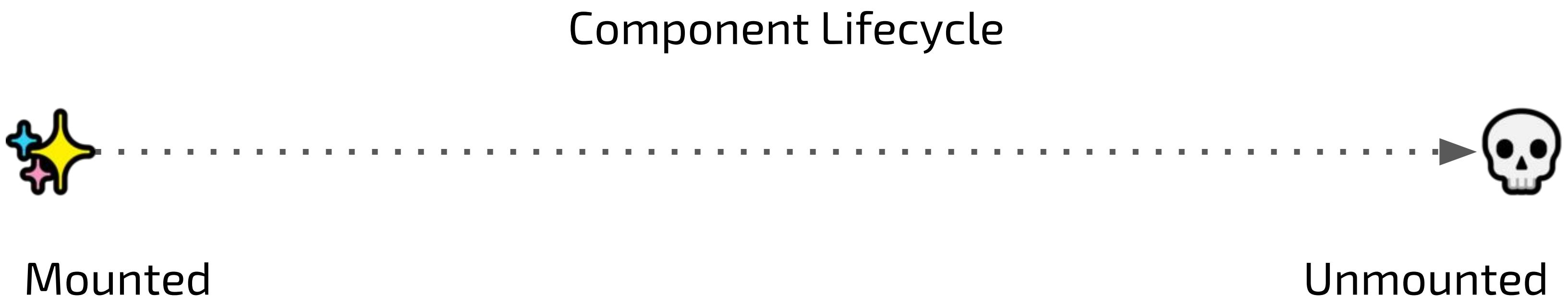
```
const SearchDisplay = ({ search, onClear }) => (
 <div>
   <p>Current search: {search}</p>
   <button onClick={onClear}>Clear</button>
 </div>
);
```

Index

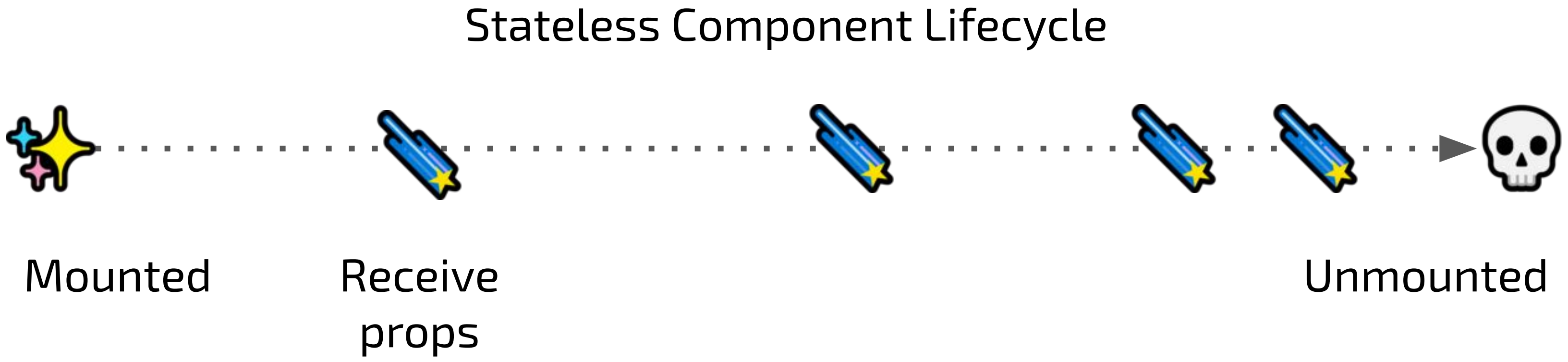
Lifecycle



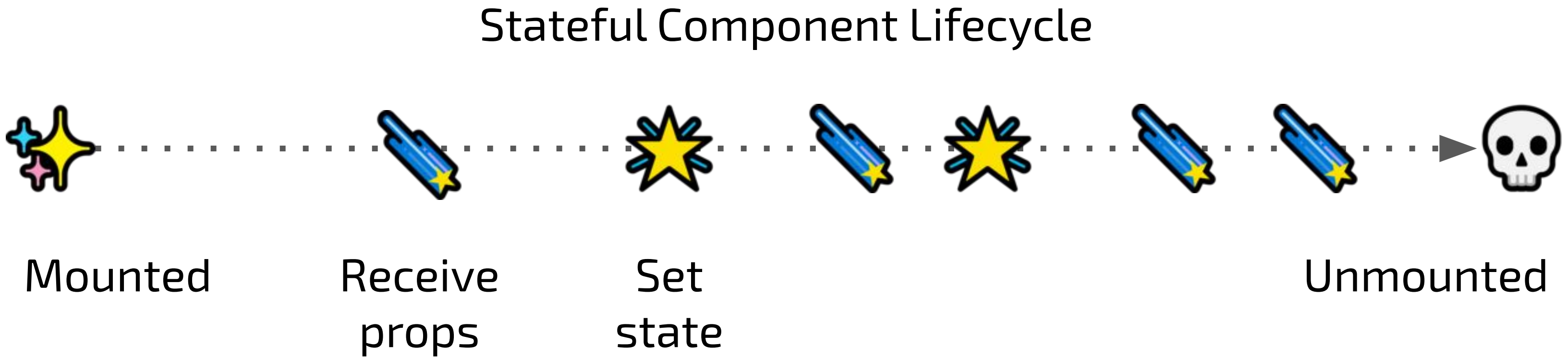
Lifecycle - Component lifecycle



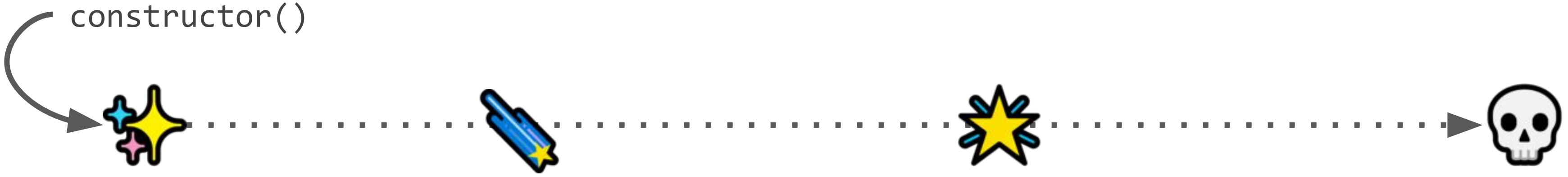
Lifecycle - Stateless Component Lifecycle



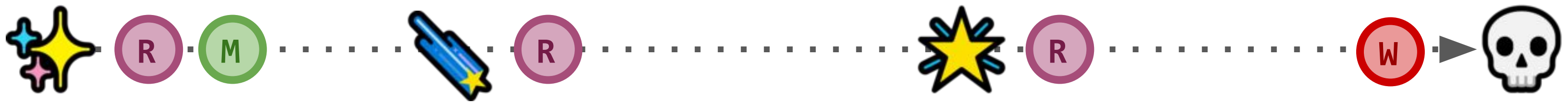
Lifecycle – Stateful Component Lifecycle



Lifecycle - Classy Component Lifecycle Methods

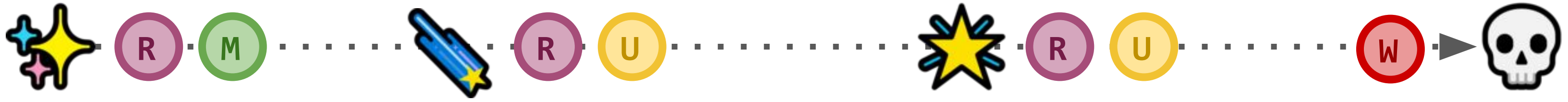


Lifecycle - Classy Component Lifecycle Methods



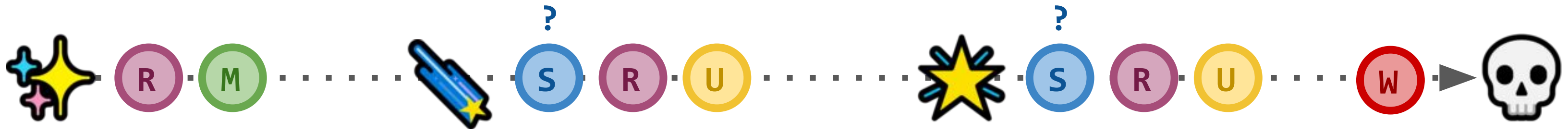
M	componentDidMount()
R	render()
W	componentWillUnmount()

Lifecycle - Classy Component Lifecycle Methods



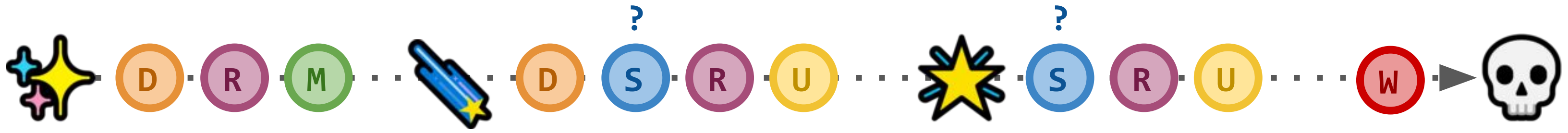
M	`componentDidMount()`
R	`render()`
W	`componentWillUnmount()`
U	**`componentDidUpdate()`**

Lifecycle - Classy Component Lifecycle Methods



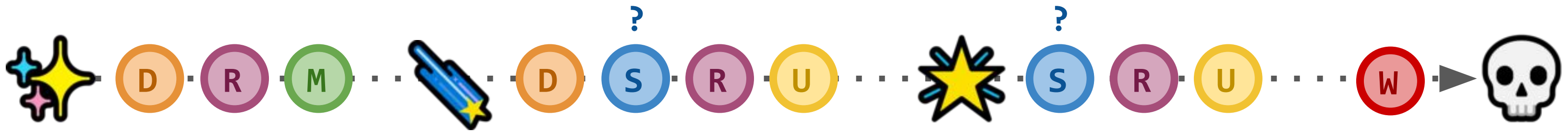
M	componentDidMount()
R	render()
W	componentWillUnmount()
U	componentDidUpdate()
S	**shouldComponentUpdate**()

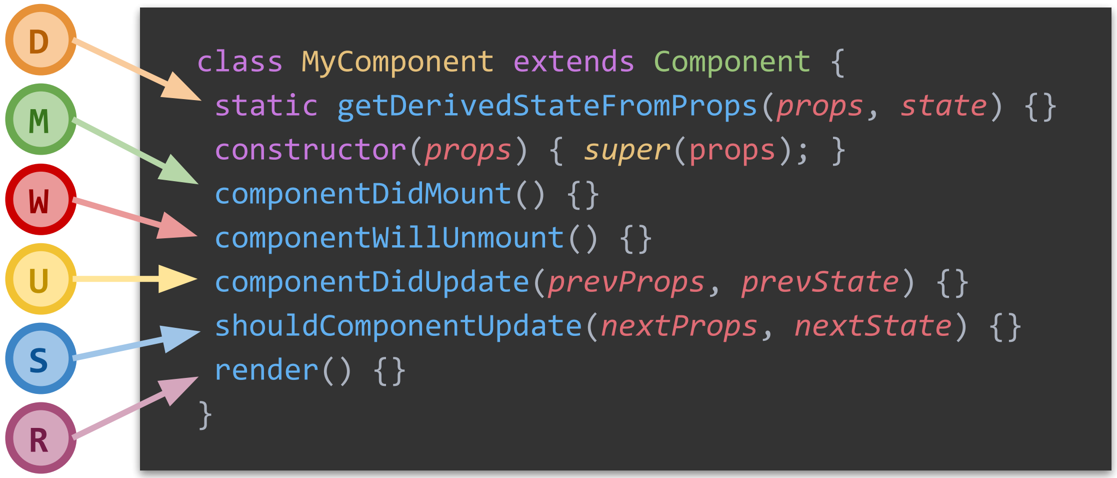
Lifecycle - Classy Component Lifecycle Methods



M	componentDidMount()
R	render()
W	componentWillUnmount()
U	componentDidUpdate()
S	shouldComponentUpdate()
D	**getDerivedStateFromProps**()

Lifecycle - Classy Component Lifecycle Methods

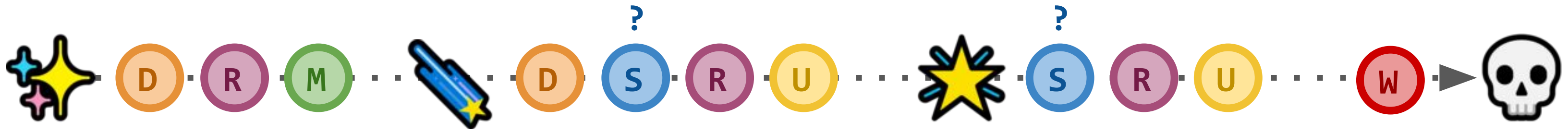




```
class MyComponent extends Component {
  static getDerivedStateFromProps(props, state) {}
  constructor(props) { super(props); }
  componentDidMount() {}
  componentWillUnmount() {}
  componentDidUpdate(prevProps, prevState) {}
  shouldComponentUpdate(nextProps, nextState) {}
  render() {}
}
```

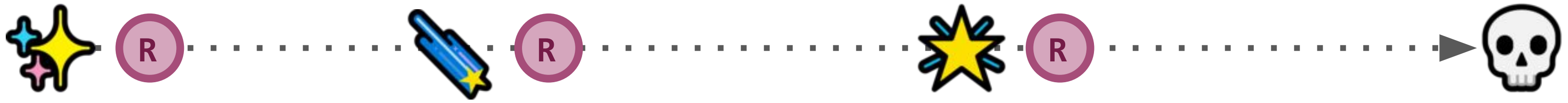
Lifecycle - Classy vs Functional Lifecycle Methods

Classy

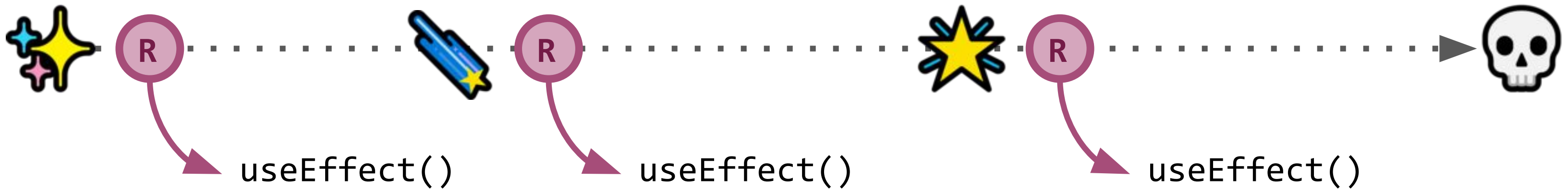


VS

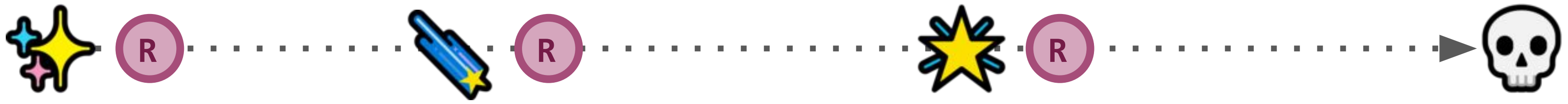
Functional

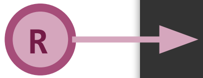


Lifecycle - Functional Lifecycle Methods



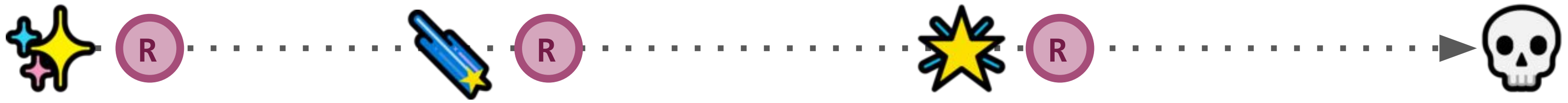
Lifecycle - Functional Lifecycle Methods





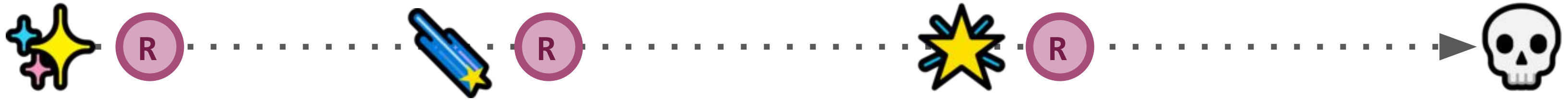
```
const MyComponent = () => {
 useEffect(() => {
   // Do stuff
 });
 return <div>Hello world</div>;
};
```

Lifecycle - Functional Lifecycle Methods



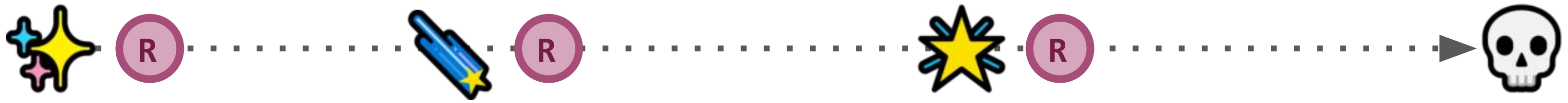
```
const FullName = ({ name, surname }) => {
 const [fullName, setFullName] = useState();
 useEffect(() => {
   setFullName(`${name} ${surname}`);
 }, [name, surname]);
 return <div>Hello {fullName}</div>;
};
```

Lifecycle - Functional Lifecycle Methods



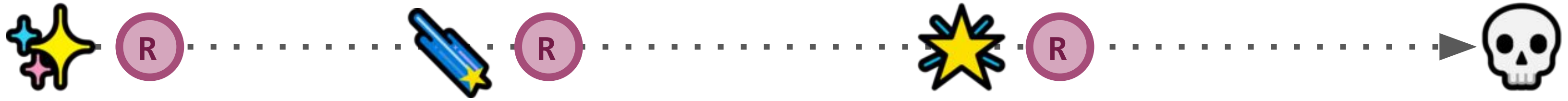
```
const FullName = ({ name, surname }) => {
 const [fullName, setFullName] = useState();
 useEffect(() => {
   setFullName(`${name} ${surname}`);
 }, [name, surname]);
 return <p>Hello {fullName}</p>;
};
```

Lifecycle - Functional Lifecycle Methods



```
const Counter = () => {
 const [counter, setCounter] = useState(0);
 useEffect(() => {
   const interval = setInterval(
     () => setCounter(counter + 1), 1000
   );
   return () => clearInterval(interval);
 }, [counter]);
 return <p>{counter}</p>;
};
```

Lifecycle - Functional Lifecycle Methods



```
const Counter = () => {
 const [counter, setCounter] = useState(0);
 useEffect(() => {
   const interval = setInterval(
     () => setCounter(c => c + 1), 1000
   );
   return () => clearInterval(interval);
 }, []);
 return <p>{counter}</p>;
};
```

Lifecycle - Functional Lifecycle Methods

```
useEffect(
    ()=> {
        // Fn body
        return cleanFn;
    },
    [memo, deps]
)
```

```
componentDidMount()
componentDidUpdate()
shouldComponentUpdate()
componentWillUnmount()
```

Index

Exercises!

1. Separate our Wish List application in several components. Use props to pass data down.
    a. `WishlistInput`
    b. `WishlistItem`

2. Add functionality to our application
    a. The input should create new wishes to add to the list
    b. The wishes checkbox should mark the wish as done
    c. Buttons to archive completed wishes should make the be removed from the list
3. Every wish should be coloured depending on the time that has remained undone: orange (>10s), red (>20s).

4. Props & State

Introduction to React