REACT JS

CHEATSHEET

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

COMPONENTS

```
import React from 'react';
import ReactDOM from 'react-dom'
```

```
class Hello extends React.component {
  render() {
    return <div className="message-box">Hello {this.props.name}</div>
  }
}
```

```
const el = document.body;
ReactDOM.render(<Hello name="John" />,el);
```

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

Import Multiple Exports

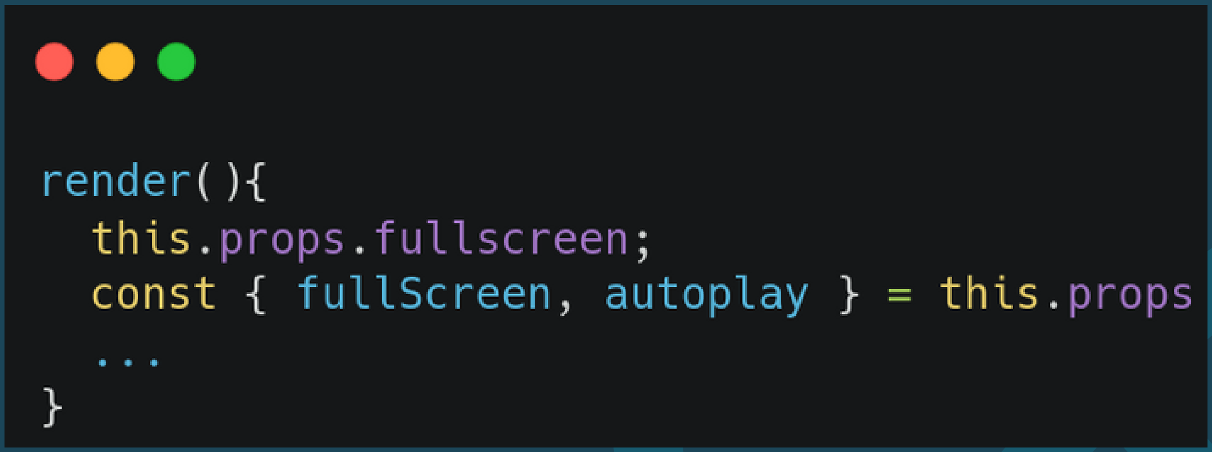
```
import React, {Component} from 'react';
import ReactDOM from 'react-dom';
```

```
class Hello extends Component{
  ...
}
```

Properties

```
<Video fullscreen={true} autoplay={false} />
```

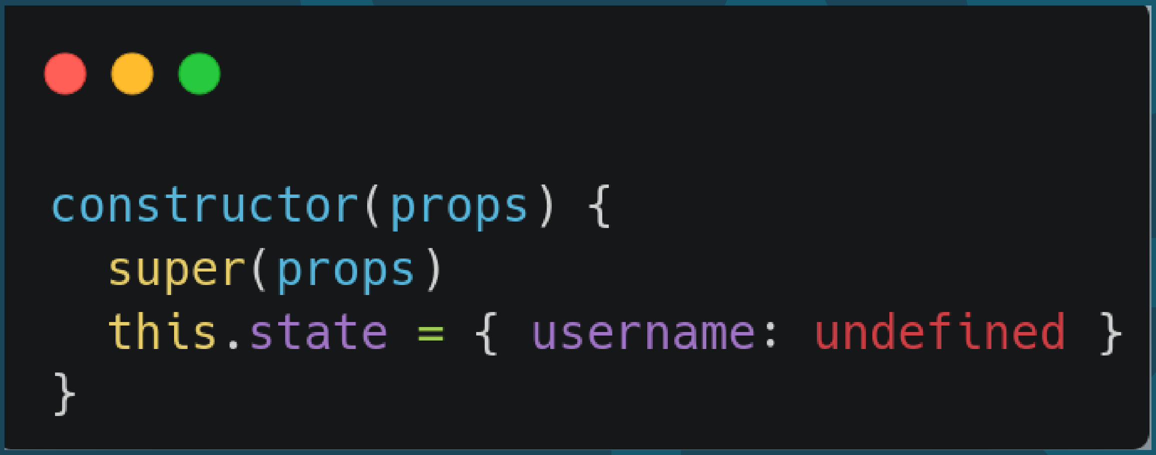
+92 315 980 7707
https://ismail.vercel.app
ismaeel.kheshgi@gmail.com



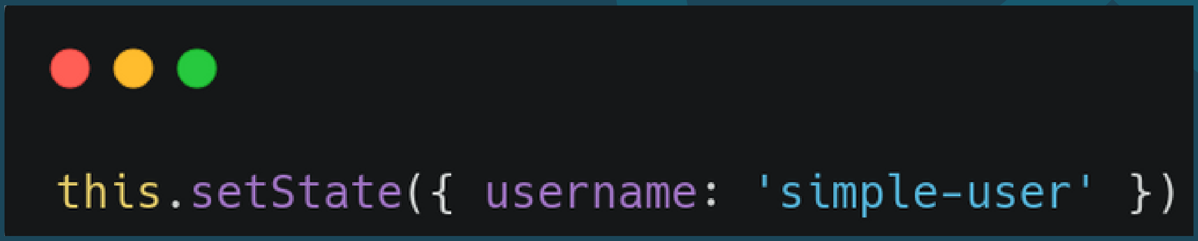
```
render(){
  this.props.fullscreen;
  const { fullScreen, autoplay } = this.props
  ...
}
```

Use this.props to access properties passed to the component.

States



```
constructor(props) {
  super(props)
  this.state = { username: undefined }
}
```

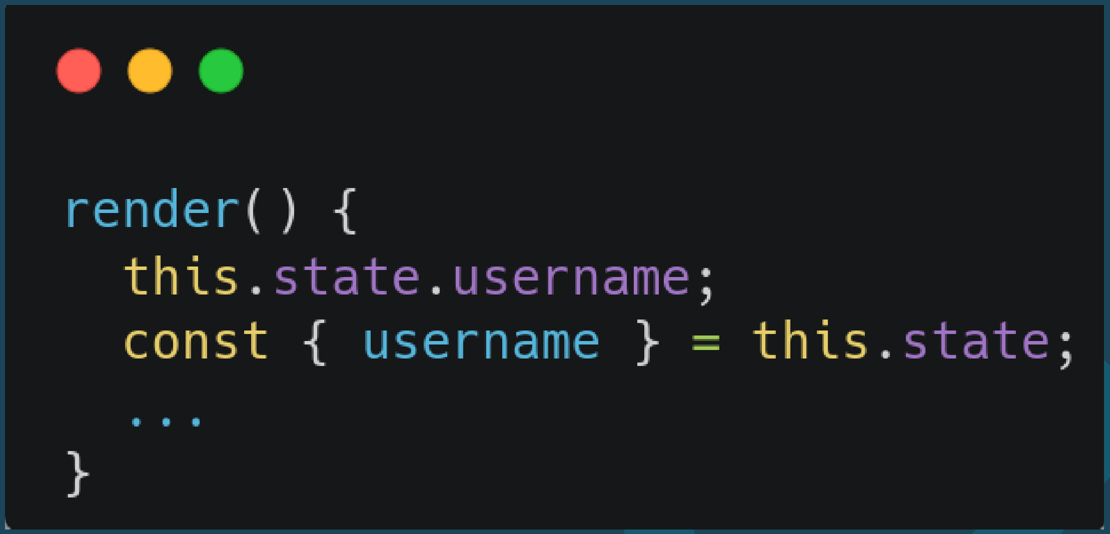


```
this.setState({ username: 'simple-user' })
```

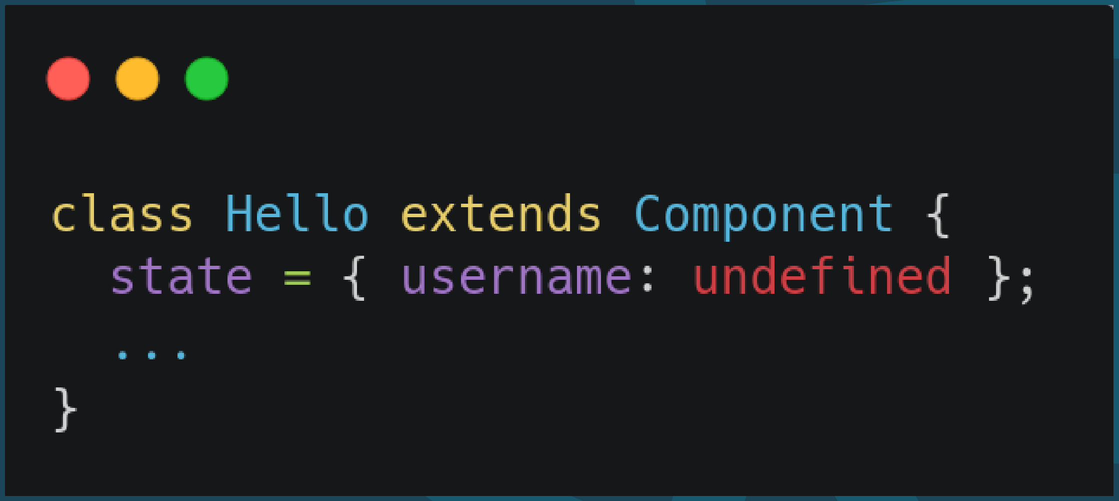
+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com



```
render() {
  this.state.username;
  const { username } = this.state;
  ...
}
```



```
class Hello extends Component {
  state = { username: undefined };
  ...
}
```

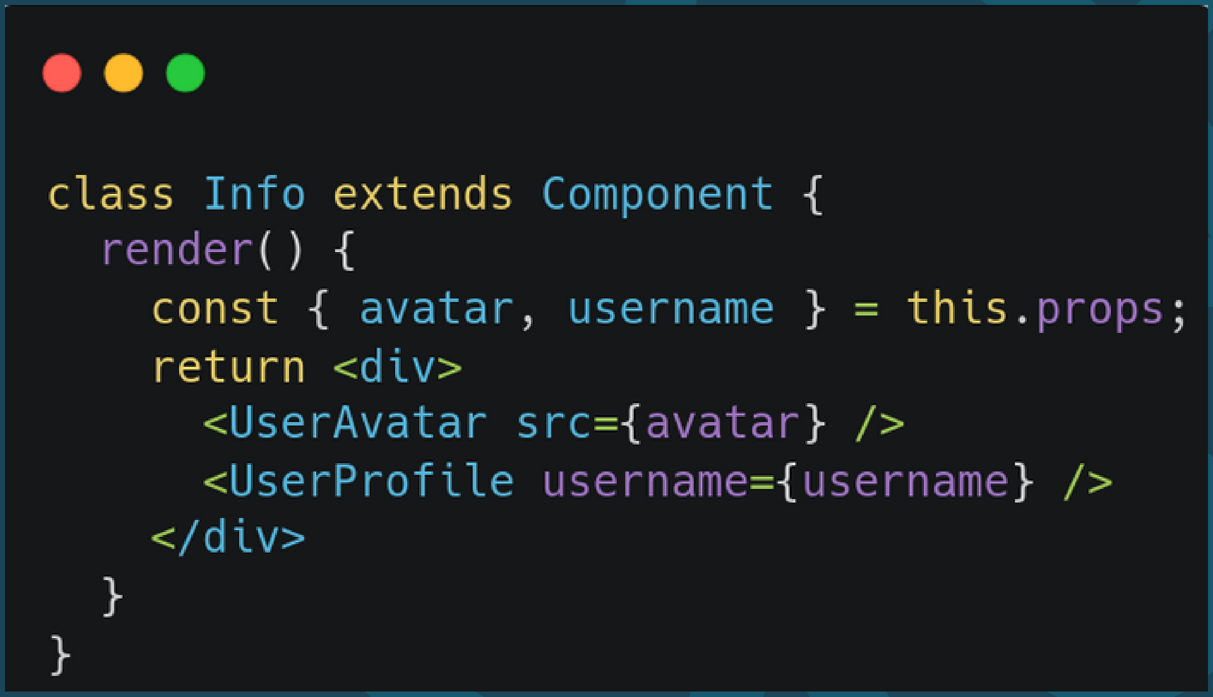
Use states {this.state} to manage dynamic data.

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

Nesting



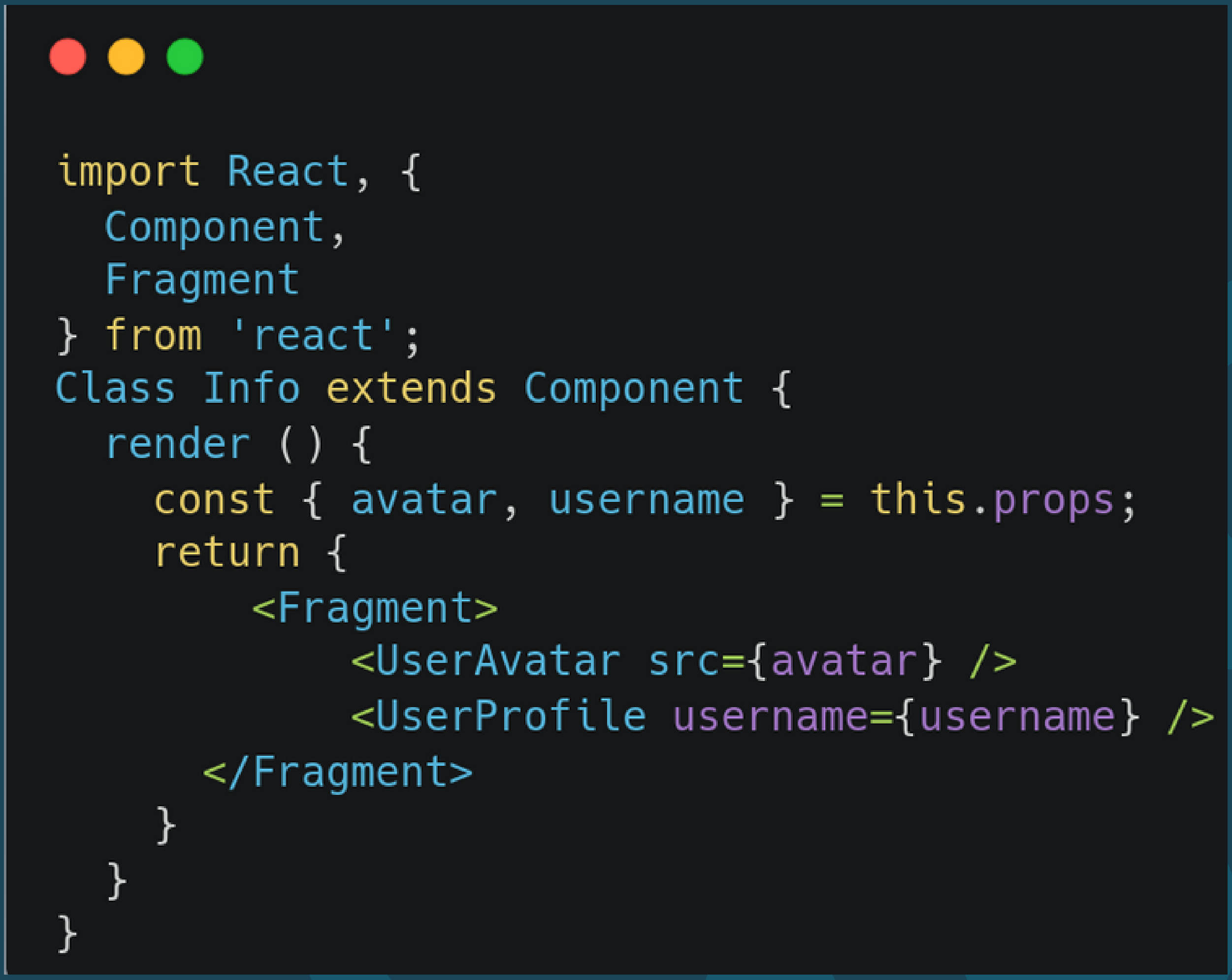
```
class Info extends Component {
  render() {
    const { avatar, username } = this.props;
    return <div>
      <UserAvatar src={avatar} />
      <UserProfile username={username} />
    </div>
  }
}
```

As of React v 16.2.0, fragments can be used to return multiple children without adding extra wrapping nodes to the DOM.

+92 315 980 7707

https://ismail.vercel.app

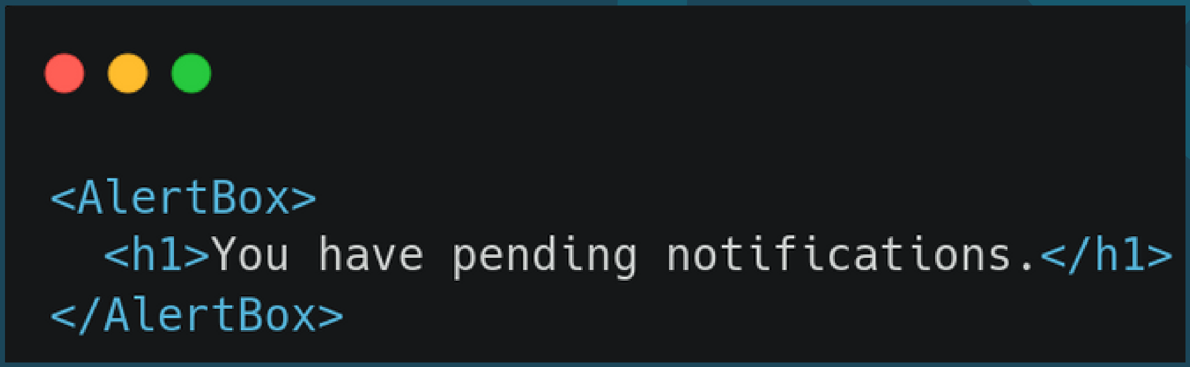
ismaeel.kheshgi@gmail.com



```
import React, {
  Component,
  Fragment
} from 'react';
Class Info extends Component {
  render () {
    const { avatar, username } = this.props;
    return {
        <Fragment>
            <UserAvatar src={avatar} />
            <UserProfile username={username} />
      </Fragment>
    }
  }
}
```

Nest components to separate concerns

Children

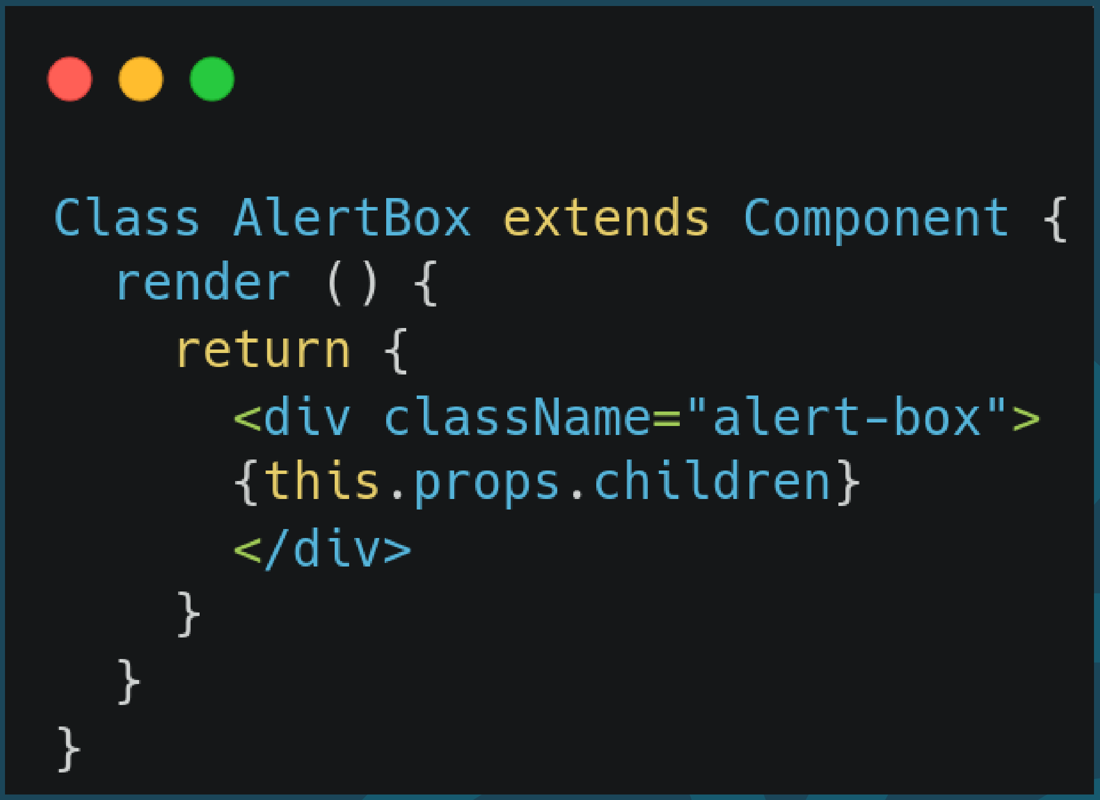


```
<AlertBox>
  <h1>You have pending notifications.</h1>
</AlertBox>
```

+92 315 980 7707

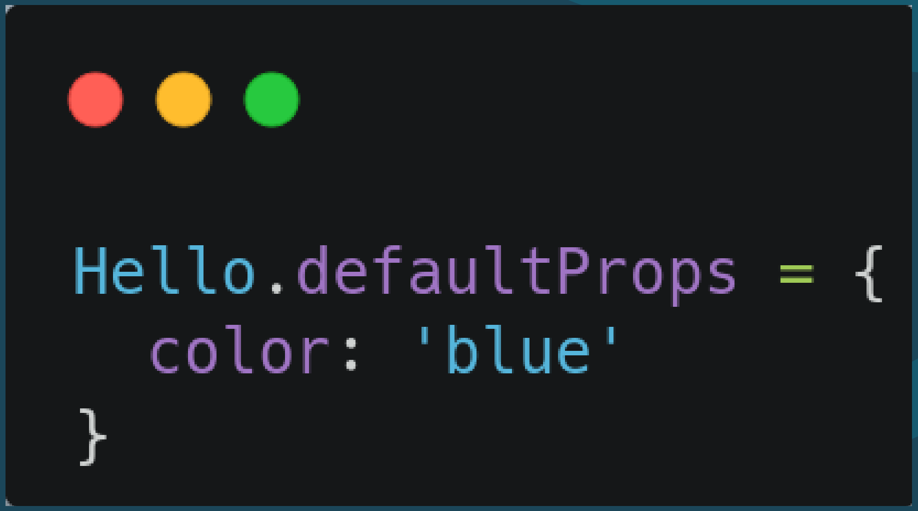
https://ismail.vercel.app

ismaeel.kheshgi@gmail.com



```
Class AlertBox extends Component {
  render () {
    return {
      <div className="alert-box">
      {this.props.children}
      </div>
    }
  }
}
```

Setting Default Props



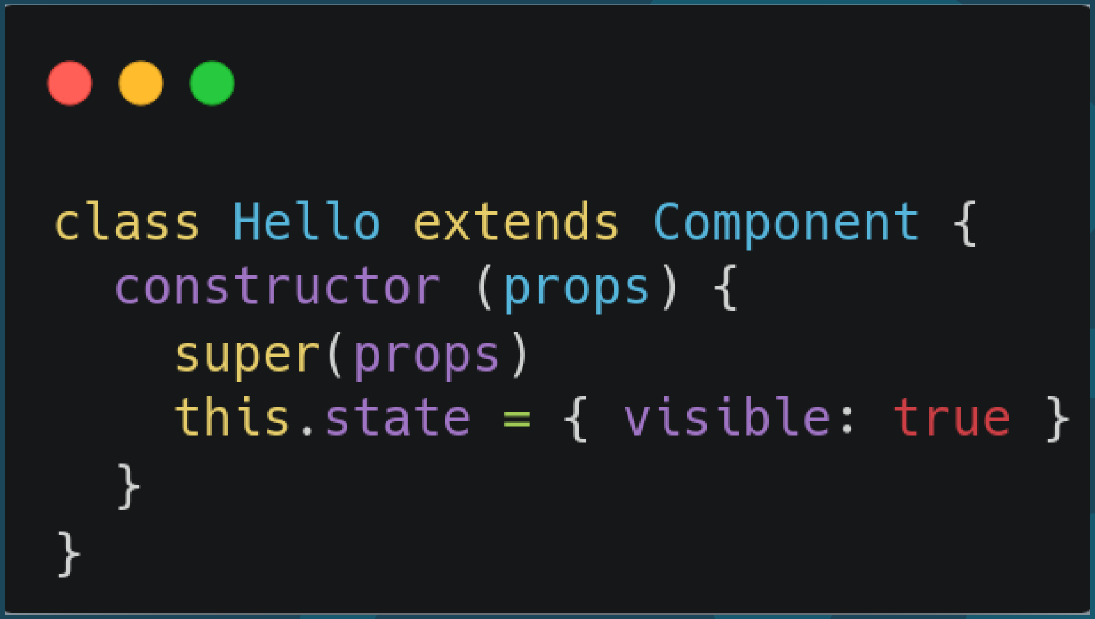
```
Hello.defaultProps = {
  color: 'blue'
}
```

+92 315 980 7707

https://ismail.vercel.app

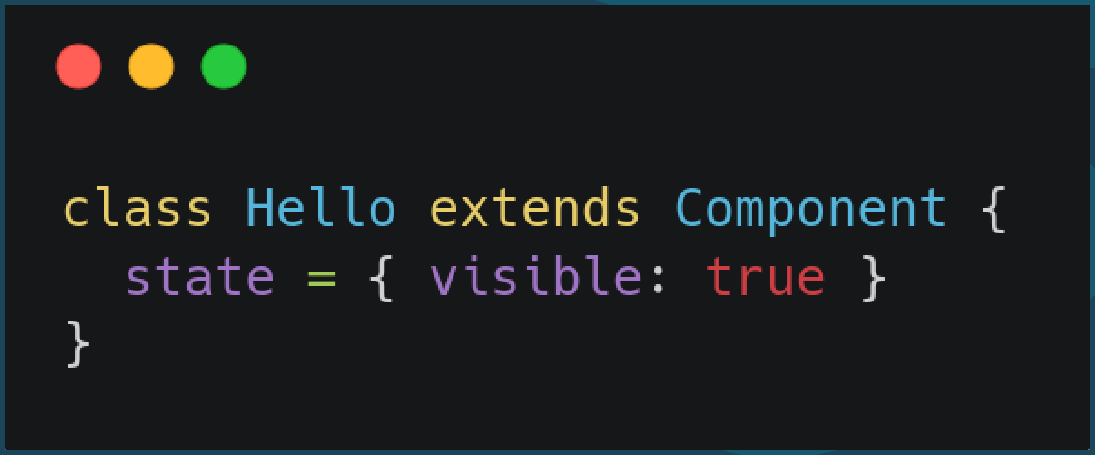
ismaeel.kheshgi@gmail.com

Setting Default State



```
class Hello extends Component {
  constructor (props) {
    super(props)
    this.state = { visible: true }
  }
}
```

Children are passed as the children property



```
class Hello extends Component {
  state = { visible: true }
}
```

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

Functional Components

```
function MyComponent ({ name }) {
  return <div className="message-box">
        Hello {name}
    </div>
}
```

Functional component have no state. Also, their props are passed as the first parameter to a function.

Pure Components

```
import React, { PureComponent } from 'react';
class MessageBox extends PureComponent {
  ...
}
```

Performance-optimized version of React Component
Doesn't re-render if props/state hasn't changed.

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

Component API

```
this.forceUpdate()
```

```
this.setState({ ... })
this.setState(state => { ... })
```

```
this.state
this.props
```

These methods and properties are available for Component instances.

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

LifeCycle

Mounting

constructor (props)	Before rendering
componentWillMount()	Don't use this
render()	Render
componetDidMount()	After rendering (DOM available)
...	...
componentWillUnmount()	Before DOM removal
...	...
componentDidCatch()	Cattch errors (16+)

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com

Set initial the state on constructor(), Add DOM event handlers, timers, (etc) in componentDidMount(), then remove them on componentWillUnmount().

Updatin

componentDidUpdate (prevProps, prevState, snapshot)	Use setState() here, but remember to compare props
shouldComponentUpdate (newProps, newState)	Skips render() if returns false
render()	Render
componentDidUpdate (prevProps, preState)	Operate on the DOM here

Called when parents change properties and setState(). These are not called for initial renders.

+92 315 980 7707

https://ismail.vercel.app

ismaeel.kheshgi@gmail.com