# Introduction to States

## States in React

* States are used to store a components dynamic data.
* It allows the component to keep track of changing information in between renders. When the state is updated render() is called to update the component.
* React states are attached to each component.
* Unlike props, states are:
  + **Mutable** – they can be changed.
    - *While this.state is immutable, you modify a copy of it with setState(). This lets us keep track of changes***.**
  + Controlled solely by the component it is initialised in, not the component passing it down.

States should be used to store component data that changes over a component’s lifecycle.

## Initialising a State

States can be initialised per component, so you can have multiple states per project. States aren initialised in the constructor, so you can only use them in class components. So this:

const myComp = () => (<p>Example</p>);

Would become this:

export class myComp extends Component {

render() {

return (<p>Example</p>);

}

}

You can initialise the state by giving it an initial value in your components constructor. The state is an object, with properties and values. When overriding the component constructor don’t forget to call **super()** before anything else. If you’re using props in your component make sure to pass that in super too.

constructor(props) {

super(props);

this.state = {

clicked: false,

clicks: 0

};

}

And with that our components state is initialised with a set of default properties and values.

## Interacting with States

Thing of the state as an object; you can read values from properties just like any other.

const hasClicked = this.props.clicked;

Updating the state however is different. **Don’t update this.state directly**, it’s immutable. Instead use **this.setState()** to schedule an update. If we updated this.state directly the component wouldn’t re-render. That’s because the state wouldn’t be able to detect the change.

This.setState() takes an object which is the partial section of the state you wish to update. This means it is possible to update more than one property at a time.

this.setState({

clicked: true,

clicks: this.state.clicks + 1

});

When a method needs to interact with the state it is important to bind **this** to the method. This can be done in the constructor like so:

constructor(props) {

super(props)

this.state = {

clicked: false,

clicks: 0

};

// addClick is a function that uses state

this.addClick = this.addClick.bind(this)

}

Alternatively, you can define your function in the constructor. This is useful for smaller operations but can clutter the constructor.

// Declaring functions in the constructor

// avoids the need for bind()

this.resetClicks = () => this.setState(

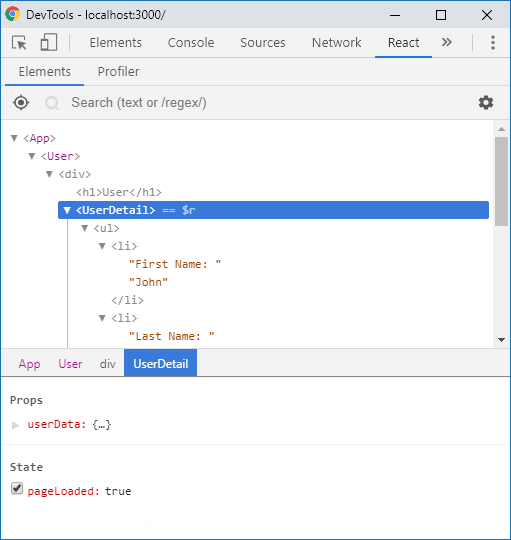
{ clicks: 0 }

);

## Debugging

**React Developer Tools** is a Chrome extension that allows you to debug React apps more efficiently.

It allows you to monitor and manipulate state data on a component by component basis, updating the component in real time.



Extension: <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>