Code Nation

React

State - Part two

{codenation}®

Learning Objectives

- To understand what state is
- To include methods in class components

Starterfiles

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4]
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
      </div>
    );
  }
}

export default App;
```

>src/App.js

Props and states

Props

Props

Getting information from **outside** the components.

*could either be a class or function, however function is preferred.

State

State

Getting information from **inside** the components. The data is managed from inside the component.

*strictly class only

State is an object!

*just like props

State is also special property.

Every time React detects the state has been changed, this triggers a re-render of the components which have changed.

Stateless and Stateful components

Functional components are known as stateless components.

Functional components are known as stateless components.

Class components are known as stateful components.

Changing state using methods

this.setState()

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [5,6,7,8]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>
          {eachNumber}
        </ul>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [5,6,7,8]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

A button is added here. Listen to the click event, and reference to the function specified above.

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [5,6,7,8]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

When the function 'addNumberHandler' is called, this would set the state to the new values stated here using 'this.setState'.

Passing an argument using anonymous function

```jsx
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = (number) => {
    this.setState({
      numbers: [number]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick ={() => this.addNumberHandler(10)}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = (number) => {
    this.setState({
      numbers: [number]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick ={() => this.addNumberHandler(10)}>Ad
      </div>
    );
  }
}
export default App;
```

This anonymous function is called and taking in a number.

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = (number) => {
    this.setState({
      numbers: [number]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick ={() => this.addNumberHandler(10)}>
      </div>
    );
  }
}
export default App;
```

Brackets are used here when passing an argument. We are passing an anonymous function returns our method call.

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = (number) => {
    this.setState({
      numbers: [number]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick ={() => this.addNumberHandler(10)}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

This is passed to our method, which we can then use as a variable anywhere in our function, including `setState()`

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {
  state = {
    numbers : [1,2,3,4]
  }
  addNumberHandler = (number) => {
    this.setState({
      numbers: [number]
    })
  }
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })
    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <button onClick ={() => this.addNumberHandler(10)}>Add Number</button>
      </div>
    );
  }
}
export default App;
```

Use of event.target and spread operator

The idea...

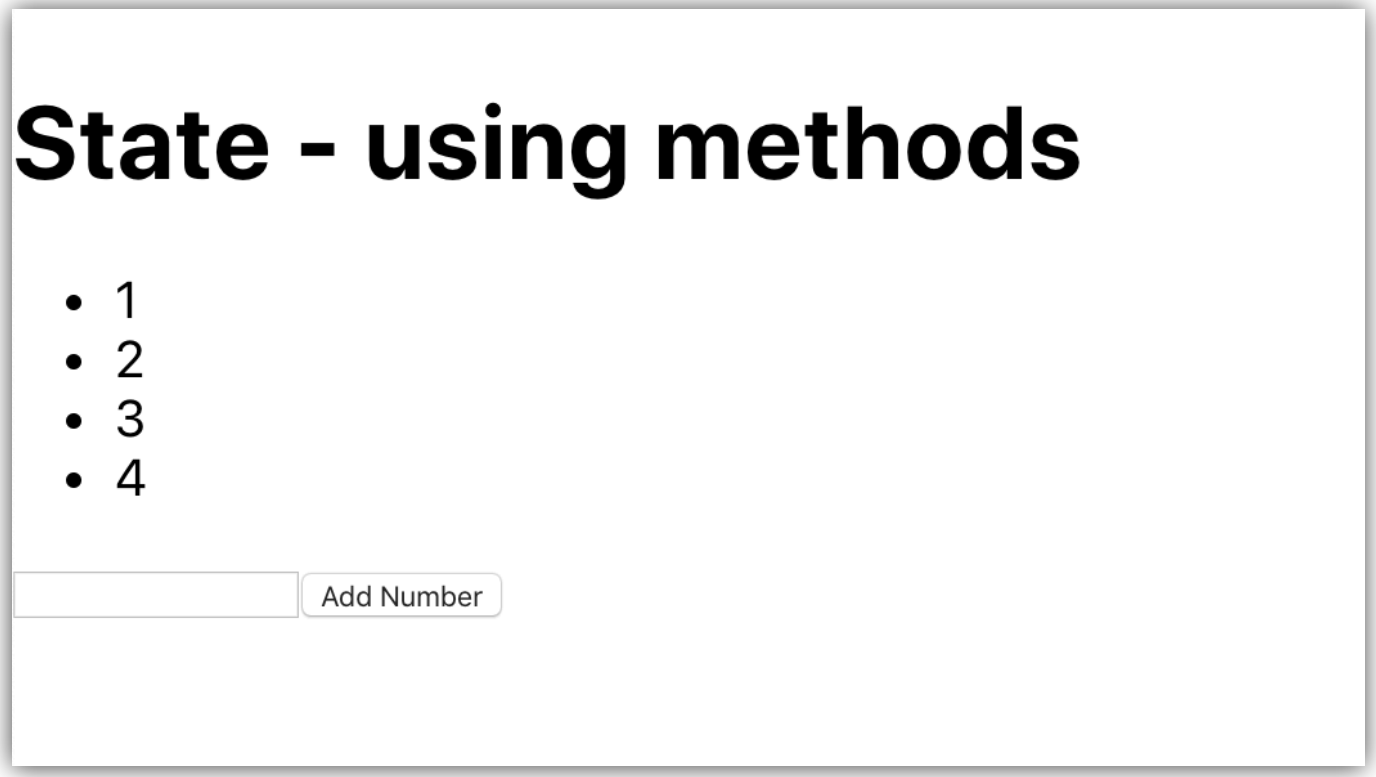
- User enters a value
- This gets stored somewhere
- Then added to the end of the list of values

Requirement?

- User enters a value: **variable**
- This gets stored somewhere: **this.setState**
- Then added to the end of the list of values: **function**

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
```



>src/App.js

Second part of code

```
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>
          {eachNumber}
        </ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}

export default App;
```

>src/App.js

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
```

currentNumber is created in the 'state', this is for storing a number temporary.

>src/App.js

Second part of code

```
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>
          {eachNumber}
        </ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}

export default App;
```

Details required when user is entering a value, call the function 'recordNumberHandler' and takes the value 'currentNumber' stored in state.

>src/App.js

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
```

When this function is called, the console log displays the value the user has provided. 'this.setState' updates the 'currentNumber' to be the 'event.target.value', i.e. the value the user provided.

>src/App.js

Second part of code

```
  render(){
    const eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>
          {eachNumber}
        </ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
      </div>
    );
  }
}

export default App;
```

When the button is pressed, the function 'addNumberHandler' will be called.

>src/App.js

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber:
    })
  }
```

This is a 'spread' operator, it takes the values already stored in state.

>src/App.js

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
```

And then adds the new number at the end of this array.

>src/App.js

First part of code...

```
import React, {Component} from 'react';
import './App.css';
class App extends Component {
  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
```

Then reset the value of 'currentNumber'

>src/App.js

Second part of code

```
render(){
  const eachNumber = this.state.numbers.map((number, index)=>{
    return <li key = {index}>{number}</li>
  })

  return (
    <div className="App">
      <h1>State - using methods</h1>
      <ul>
        {eachNumber}
      </ul>
      <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
      <button onClick={this.addNumberHandler}>Add Number</button>
    </div>
  );
}
}

export default App;
```

As the state needs to be updated, the browser re-renders.

>src/App.js

Rendering dynamic content

We can render dynamic content when a certain condition is met.

First part of code...

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4],
    currentNumber: "",
    showingNumbers: true
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
  showNumbersHandler = () => {
    let show = this.state.showingNumbers;
    this.setState({showingNumbers: !show})
  }
```

Second part of code

```
  render(){
    let eachNumber = null;
    if(this.state.showingNumbers === true){
      eachNumber = this.state.numbers.map((number, index)=>{
        return <li key = {index}>{number}</li>
      })
    }

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

First part of code...

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4],
    currentNumber: ""
    showingNumbers: true
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
  showNumbersHandler = () => {
    let show = this.state.showingNumbers;
    this.setState({showingNumbers: !show})
  }
```

Add a new property called **showingNumbers** and set it to **true**

First part of code...

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4],
    currentNumber: "",
    showingNumbers: true
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.c
      currentNumber: ""
    })
  }
  showNumbersHandler = () => {
    let show = this.state.showingNumbers;
    this.setState({showingNumbers: !show})
  }
```

This method stores the current state of the showingNumbers in a variable.

When the function runs, it sets the state to the opposite.

Second part of code

```
  render(){
    let eachNumber = null;
    if(this.state.showingNumbers === true){
      eachNumber = this.state.numbers.map((number, index)=>{
        return <li key = {index}>{number}</li>
      })
    }

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

We can put JavaScript functionality inside the render method.

Second part of code

```
  render(){
    let eachNumber = null;
    if(this.state.showingNumbers === true){
      eachNumber = this.state.numbers.map((number, index)=>{
        return <li key = {index}>{number}</li>
      })
    }

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

A new button to call the function

First part of code...

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4],
    currentNumber: "",
    showingNumbers: true
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
  showNumbersHandler = () => {
    let show = this.state.showingNumbers;
    this.setState({showingNumbers: !show})
  }
```

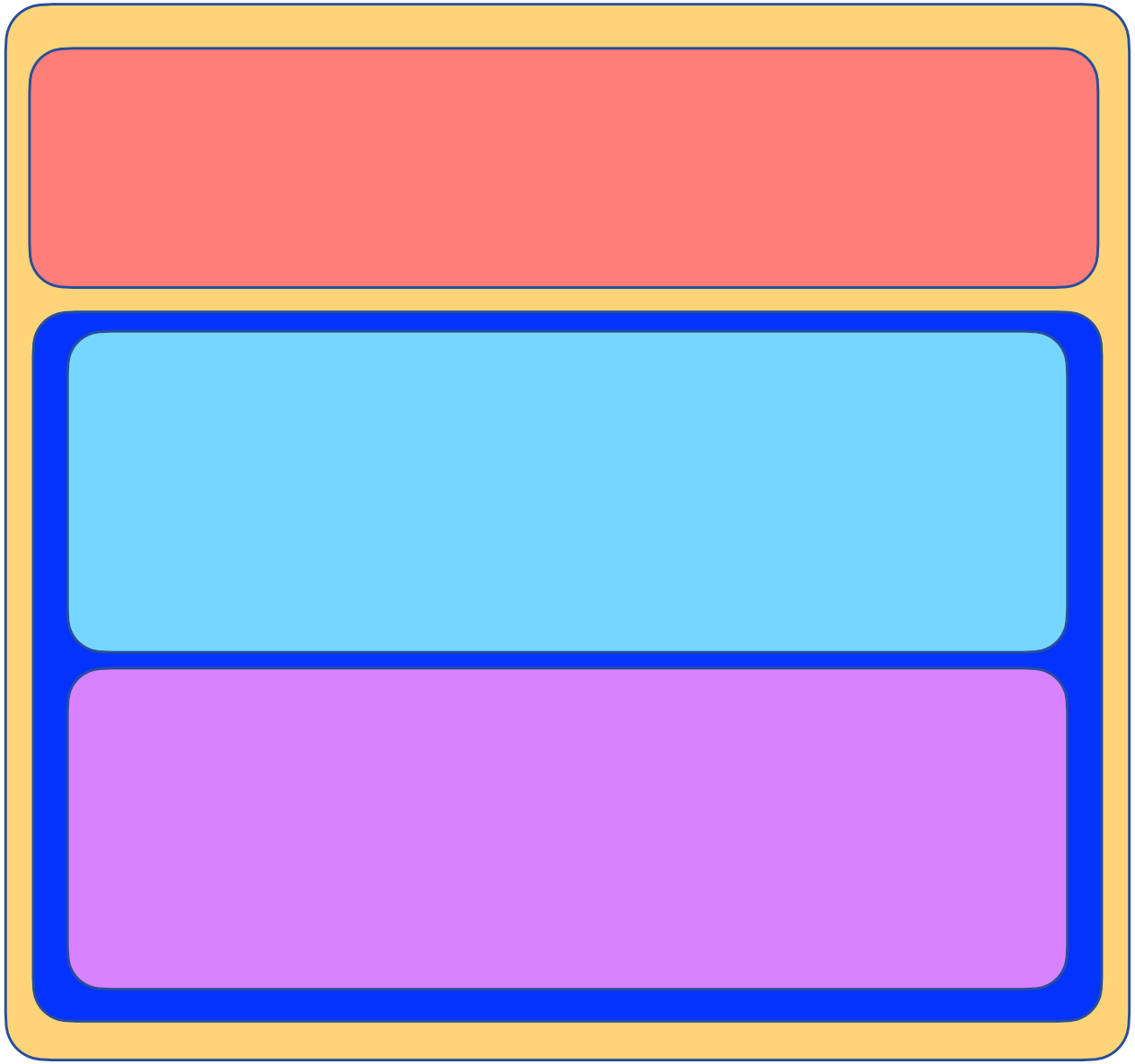
Second part of code

```
  render(){
    let eachNumber = null;
    if(this.state.showingNumbers === true){
      eachNumber = this.state.numbers.map((number, index)=>{
        return <li key = {index}>{number}</li>
      })
    }

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{eachNumber}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

Class Components



Properties and methods go here

Render( ) method

JS logic can go here.

return statement

Using ternary operator

```
Condition ? exprIfTrue : exprIfFalse
```

Second part of code

```
  render(){
    let eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{this.state.showingNumbers? eachNumber: null}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

Second part of code

```
  render(){
    let eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{this.state.showingNumbers? eachNumber: null}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

No more **if** statement here

Second part of code

```
  render(){
    let eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{this.state.showingNumbers? eachNumber: null}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

Ternary operator

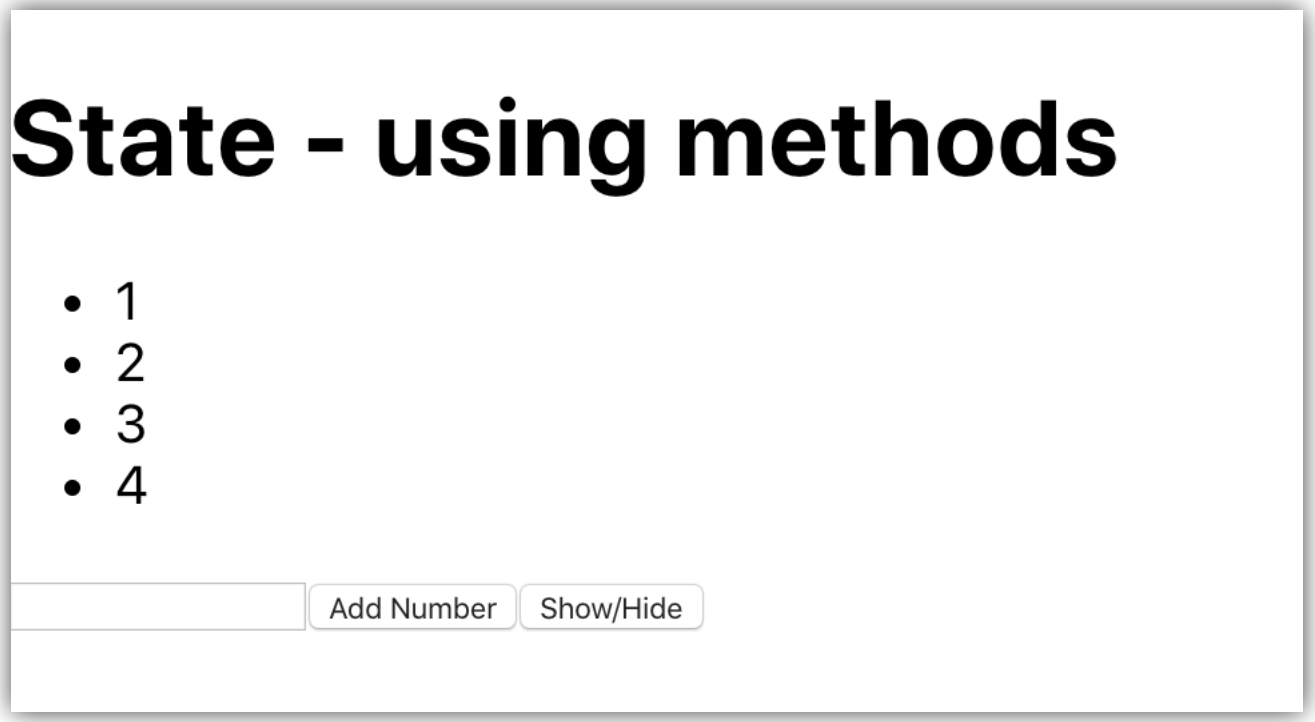
Endfiles

First part of code...

```
import React, {Component} from 'react';
import './App.css';

class App extends Component {

  state = {
    numbers : [1,2,3,4],
    currentNumber: "",
    showingNumbers: true
  }
  recordNumberHandler = (event) => {
    console.log(event.target.value)
    let num = parseInt(event.target.value)
    this.setState({
      currentNumber: num
    })
  }
  addNumberHandler = () => {
    this.setState({
      numbers: [...this.state.numbers, this.state.currentNumber],
      currentNumber: ""
    })
  }
  showNumbersHandler = () => {
    let show = this.state.showingNumbers;
    this.setState({showingNumbers: !show})
  }
```



Second part of code

```
  render(){
    let eachNumber = this.state.numbers.map((number, index)=>{
      return <li key = {index}>{number}</li>
    })

    return (
      <div className="App">
        <h1>State - using methods</h1>
        <ul>{this.state.showingNumbers? eachNumber: null}</ul>
        <input type="number" onChange={this.recordNumberHandler} value={this.state.currentNumber}/>
        <button onClick={this.addNumberHandler}>Add Number</button>
        <button onClick={this.showNumbersHandler}>Show/Hide</button>
      </div>
    );
  }
}

export default App;
```

Learning Objectives

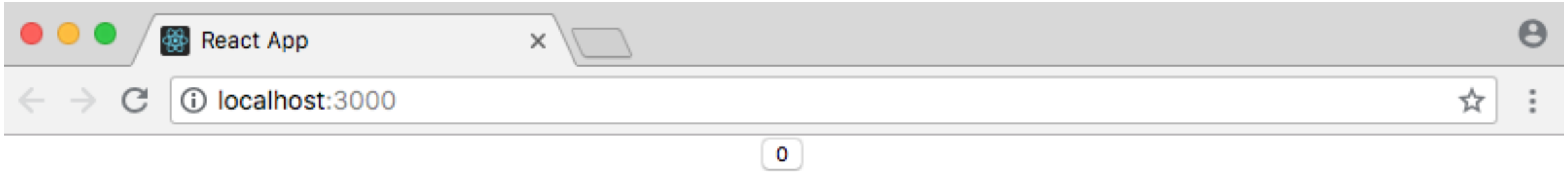
- To understand what state is
- To include methods in class components

Activity(1)

- Update the previous code-along project by:
  - Adding input boxes for the user to enter details
  - Update these by adding to the state
  - Display updated information
  - Show/hide everything when pressing a button

Activity(2)

- Create a simple counter button that will increment every time it is clicked.



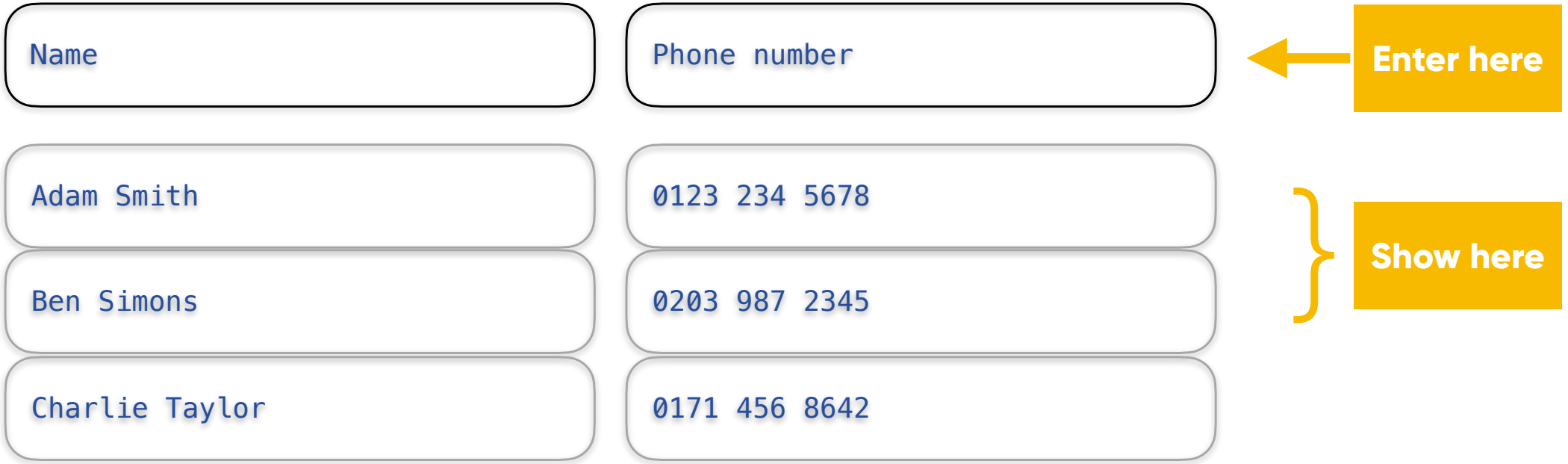
Activity(2) – cont'd

- Create a button, decrement the number when it is clicked.

- And then the increment button
- (You may need to split the value being changed from the increment button)

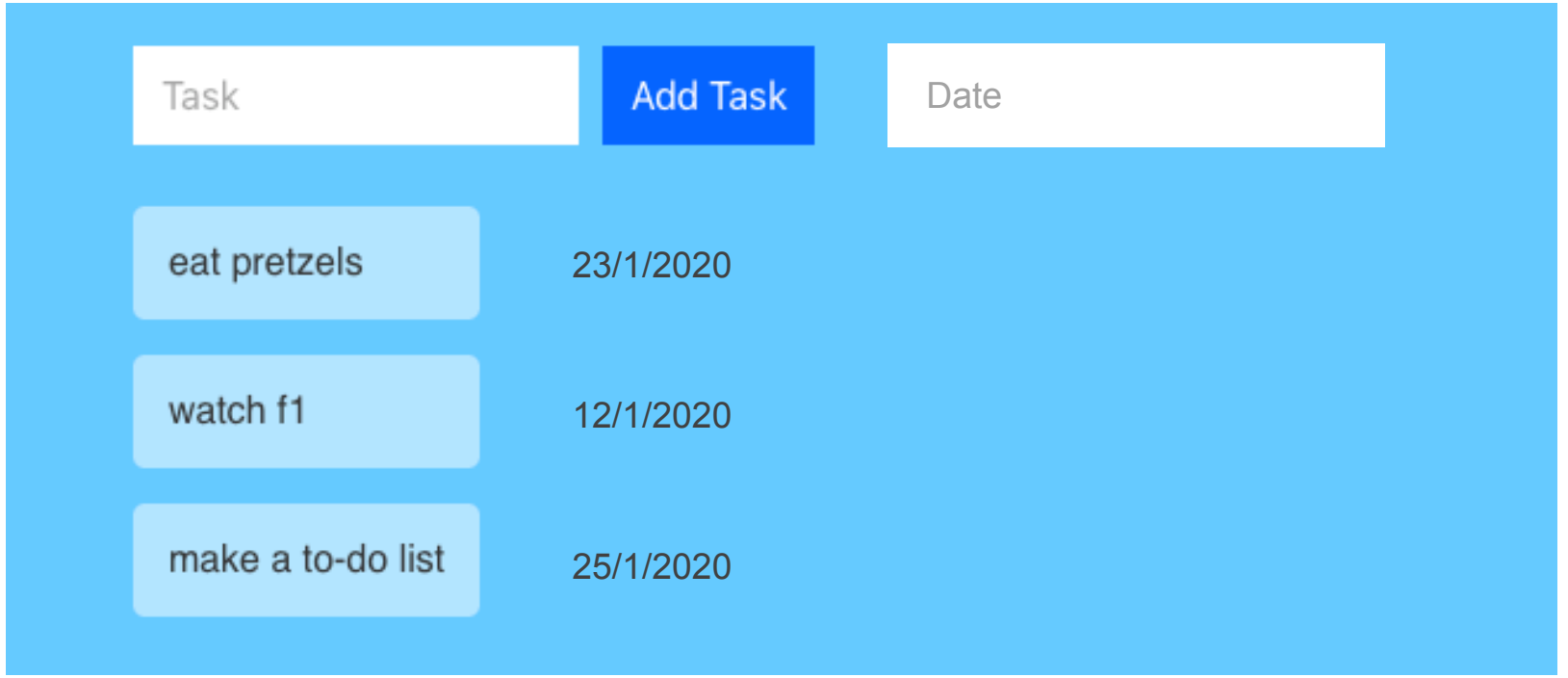
Activity(3) - Address Book

- Create an address book, you can:
  - Add phone number and name
  - Then show these in a list when entered



Activity(4) - To-do list

- Make a to-do list
- You should be able to add and remove items
- Extension: enable user to add date of completion



Activity(5)

- Create a simple calculator (interface)
- Challenge: make it work!