2. Best Practice

- The **state** should be **immutable**. We shouldn’t update the array/object directly.
- The best practice is to create a new array/object from the array/object in the previous state using the **spread** operator.

JAVASCRIPT

```javascript
this.state.contactsList = initialContactsList
```

JAVASCRIPT

```javascript
this.state.contactsList.push(...)
```

JAVASCRIPT

```javascript
this.setState(prevState => ({
  contactsList: [...prevState.contactsList, newContact ],
}))
```

2.1 Updating a Property of an Item inside List

We should not update the property of a list item directly. To update the property of a list item, we should create a new object and return it to the list.

Syntax:

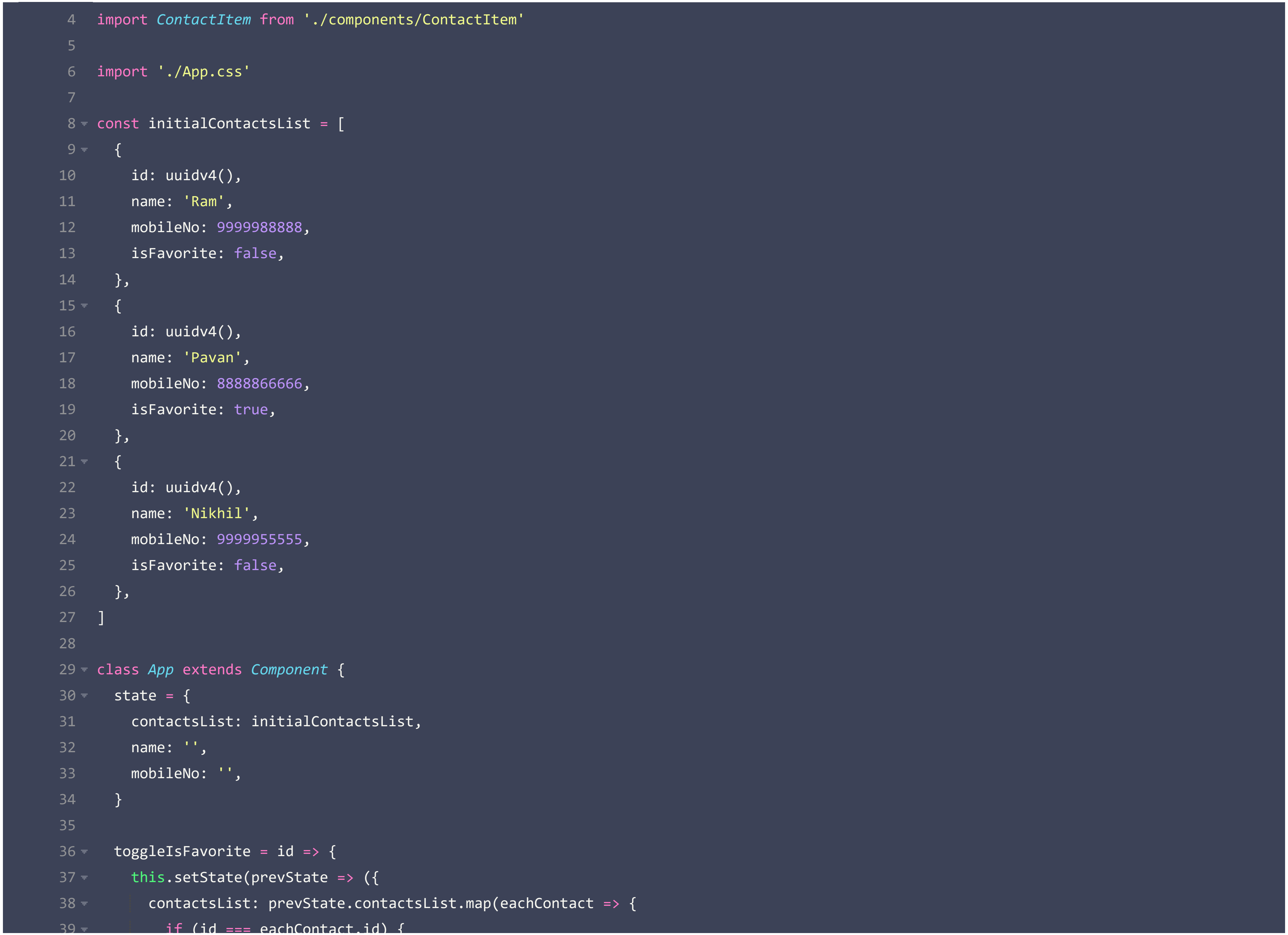
```
{...object, newItem}
```

Exmaple:

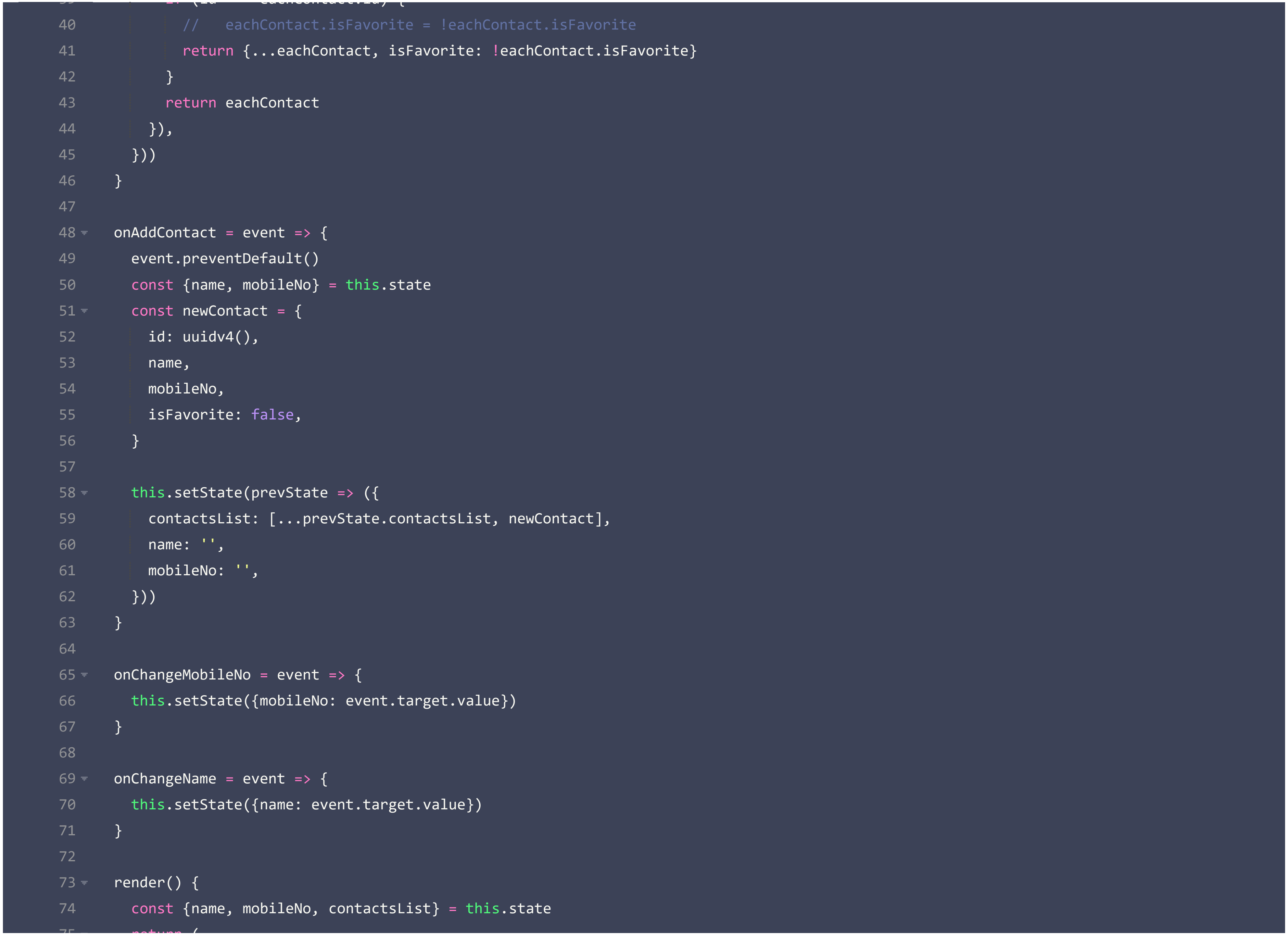
File: src/App.js

JSX

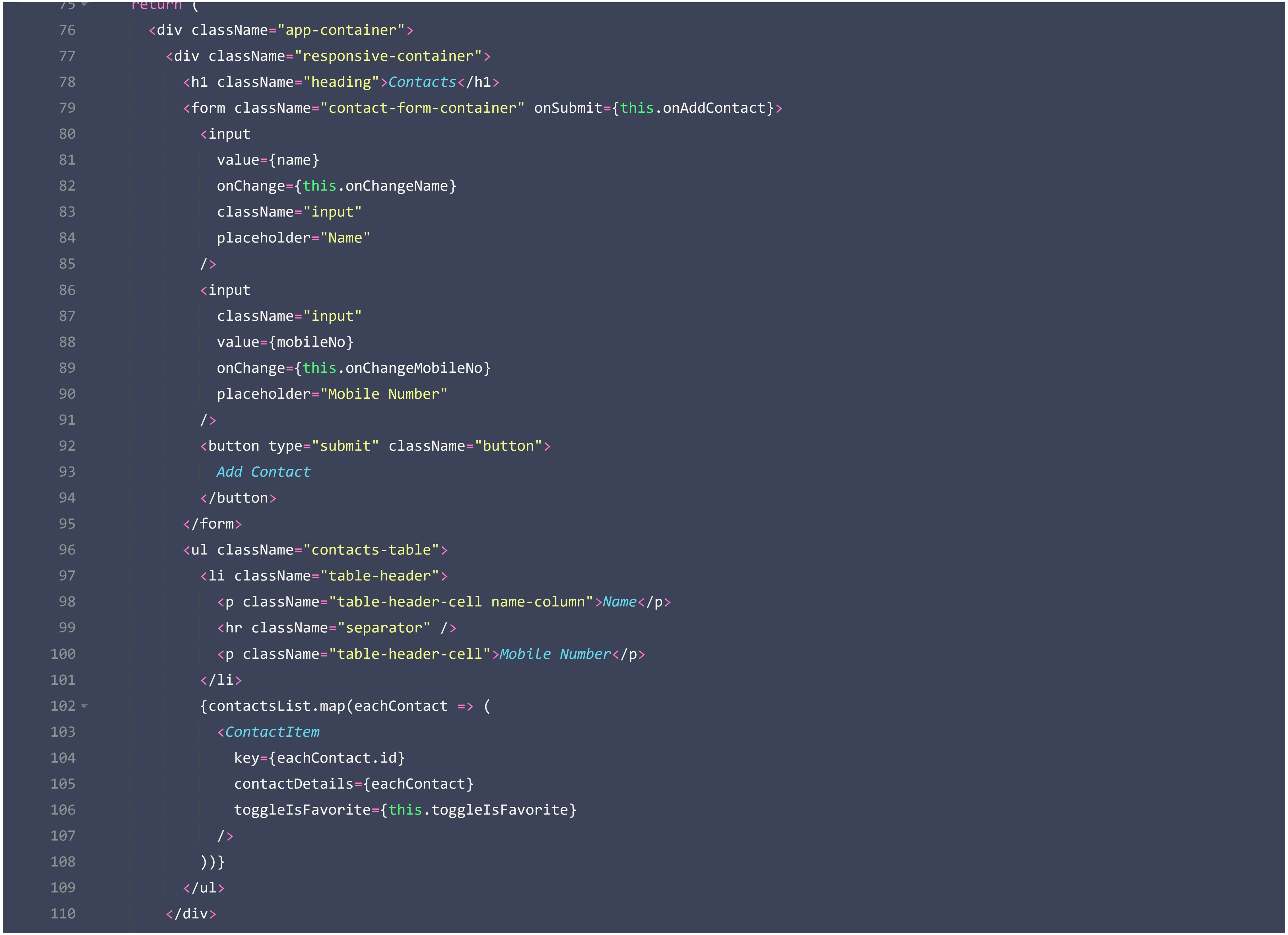
```jsx
import {Component} from 'react'
import {v4 as uuidv4} from 'uuid'

```



```
import ContactItem from './components/ContactItem'

import './App.css'

const initialContactsList = [
  {
    id: uuidv4(),
    name: 'Ram',
    mobileNo: 9999988888,
    isFavorite: false,
  },
  {
    id: uuidv4(),
    name: 'Pavan',
    mobileNo: 8888866666,
    isFavorite: true,
  },
  {
    id: uuidv4(),
    name: 'Nikhil',
    mobileNo: 9999955555,
    isFavorite: false,
  },
]

class App extends Component {
  state = {
    contactsList: initialContactsList,
    name: '',
    mobileNo: '',
  }

  toggleIsFavorite = id => {
    this.setState(prevState => ({
      contactsList: prevState.contactsList.map(eachContact => {
        if (id === eachContact.id) {
```



```
        if (id === eachContact.id) {
          //   eachContact.isFavorite = !eachContact.isFavorite
          return {...eachContact, isFavorite: !eachContact.isFavorite}
        }
        return eachContact
      }),
    }))
  }

  onAddContact = event => {
    event.preventDefault()
    const {name, mobileNo} = this.state
    const newContact = {
      id: uuidv4(),
      name,
      mobileNo,
      isFavorite: false,
    }

    this.setState(prevState => ({
      contactsList: [...prevState.contactsList, newContact],
      name: '',
      mobileNo: '',
    }))
  }

  onChangeMobileNo = event => {
    this.setState({mobileNo: event.target.value})
  }

  onChangeName = event => {
    this.setState({name: event.target.value})
  }

  render() {
    const {name, mobileNo, contactsList} = this.state
    return (
```



```
    return (
      <div className="app-container">
        <div className="responsive-container">
          <h1 className="heading">Contacts</h1>
          <form className="contact-form-container" onSubmit={this.onAddContact}>
            <input
              value={name}
              onChange={this.onChangeName}
              className="input"
              placeholder="Name"
            />
            <input
              className="input"
              value={mobileNo}
              onChange={this.onChangeMobileNo}
              placeholder="Mobile Number"
            />
            <button type="submit" className="button">
              Add Contact
            </button>
          </form>
          <ul className="contacts-table">
            <li className="table-header">
              <p className="table-header-cell name-column">Name</p>
              <hr className="separator" />
              <p className="table-header-cell">Mobile Number</p>
            </li>
            {contactsList.map(eachContact => (
              <ContactItem
                key={eachContact.id}
                contactDetails={eachContact}
                toggleIsFavorite={this.toggleIsFavorite}
              />
            ))}
          </ul>
        </div>
      </div>
    )
  }
}

export default App
```

Collapse