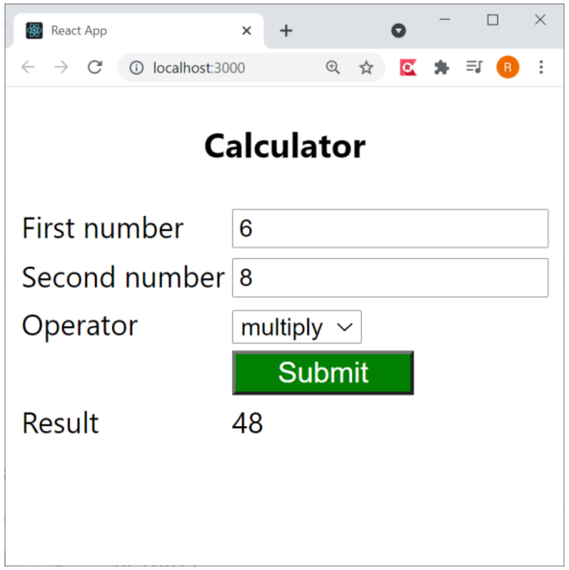
Lesson 12

BACKEND ACCESS REDUX

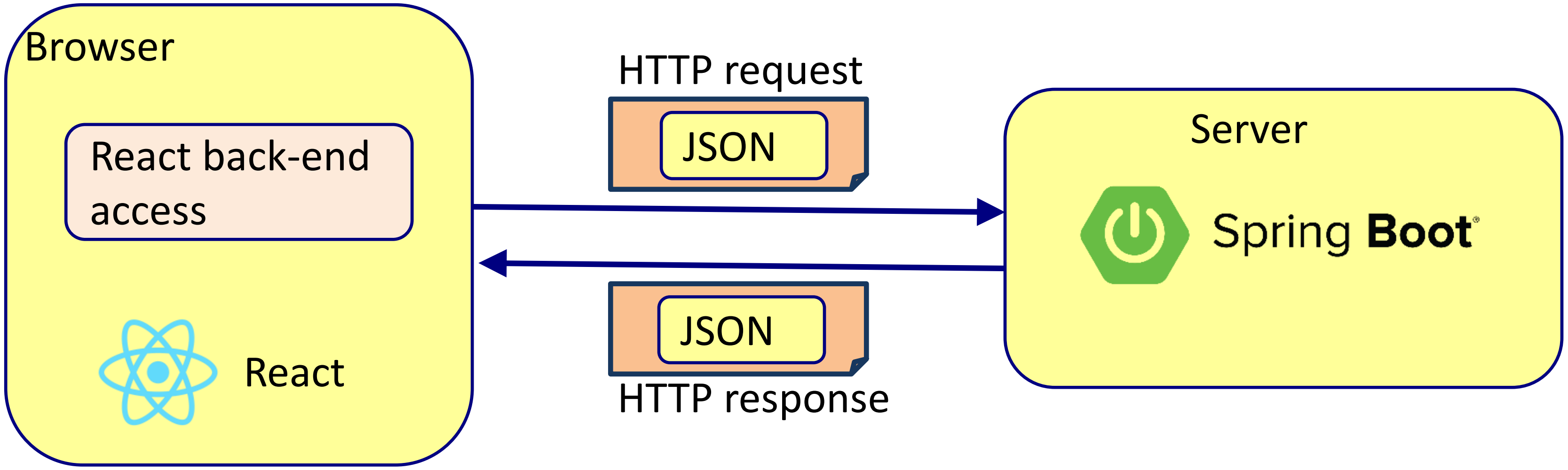
BACKEND ACCESS

Calculator

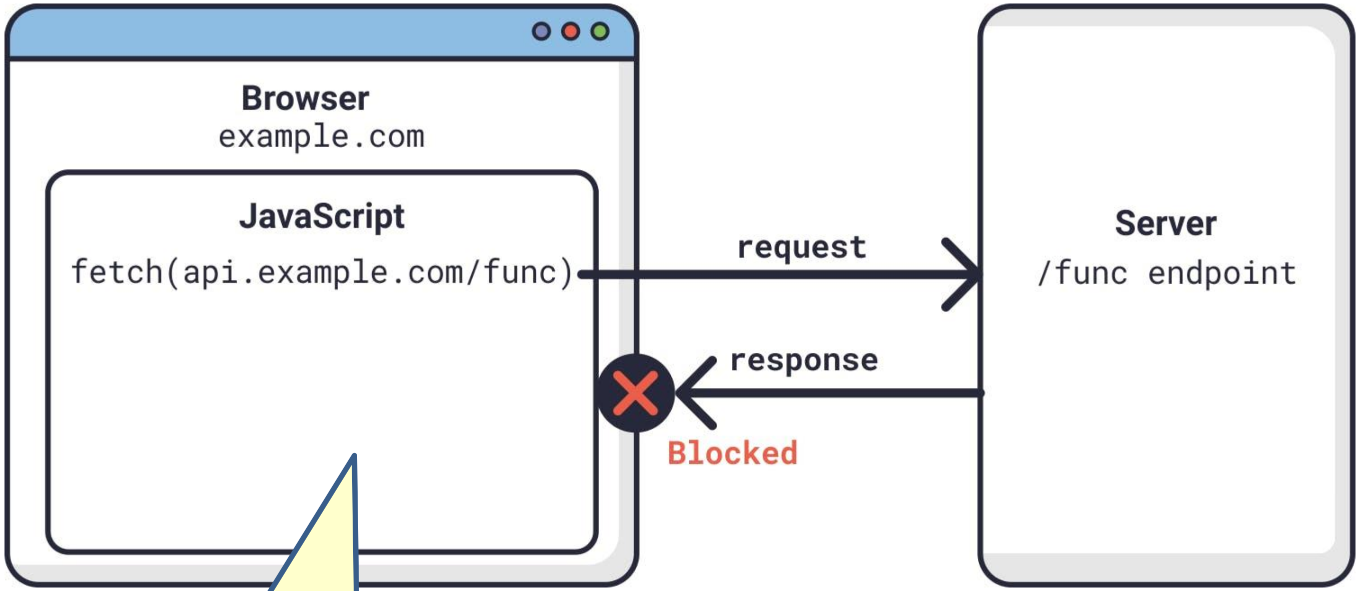

React App
localhost:3000

Calculator

First number 6
Second number 8
Operator multiply
Submit
Result 48



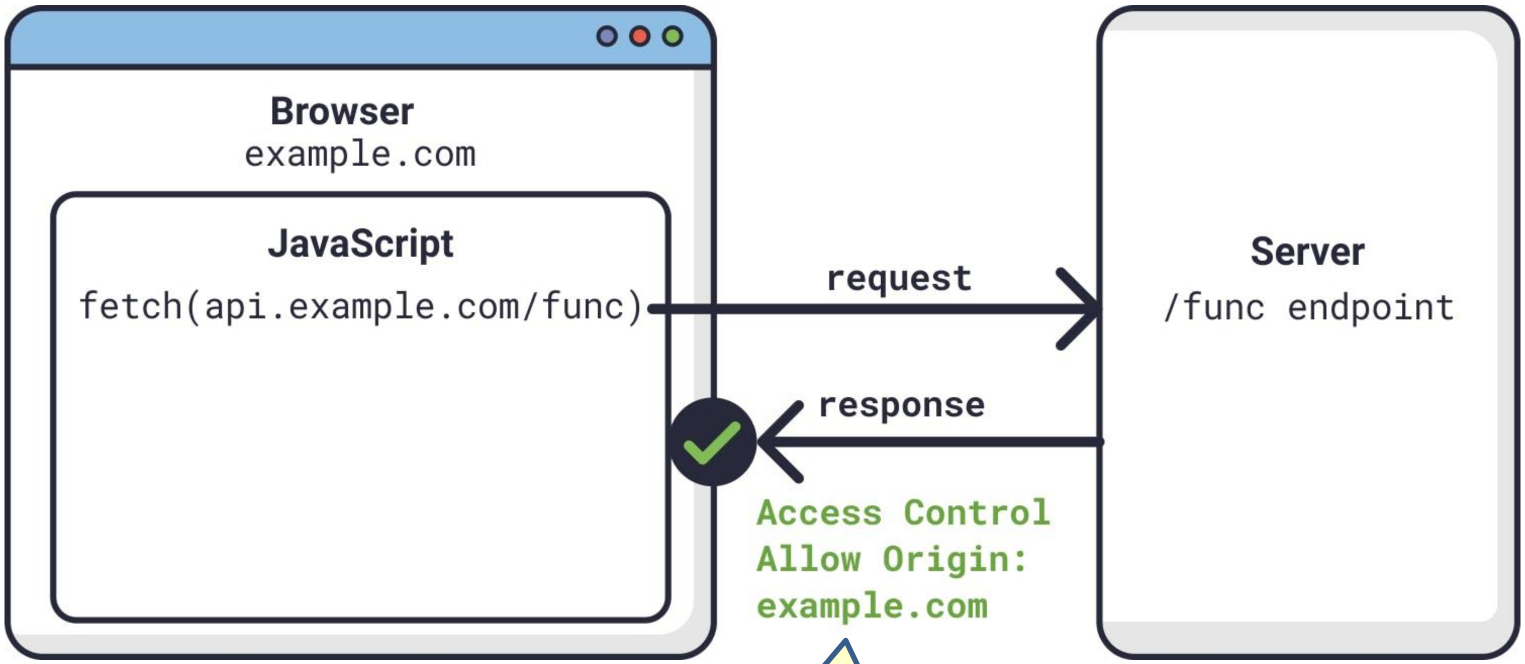
Same origin policy



By default, an application running on domain **example.com** may only request data from example.com, not **api.example.com**

CORS

- Cross-Origin Resource Sharing



Allow websites to access data from other domains

Calculator Rest controller

```
@RestController
public class CalculatorController {

  @CrossOrigin
  @GetMapping("/calc/{first}/{second}/{operator}")
   public ResponseEntity<?> calculate(@PathVariable int first, @PathVariable int second, @PathVariable String
operator) {
      double result;
       switch(operator) {
         case "add":
            result = first + second; break;
         case "subtract":
            result = first - second; break;
         case "multiply":
            result = first * second; break;
         case "divide":
            result = first / second; break;
         case "clear":
            result = 0; break;
         default:
            result = 0;
       }
       return new ResponseEntity<Result>(new Result(result), HttpStatus.OK);
   }
}
```

Enable CORS

Calculator.js

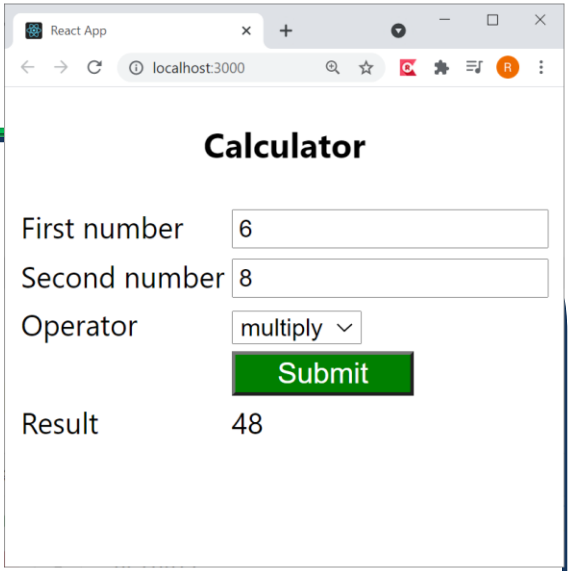
```
export const Calculator = () => {
    const [first, setFirst] = useState(0);
    const [second, setSecond] = useState(0);
    const [operator, setOperator] = useState('add');
    const [result, setResult] = useState(0);

    const fetchBackend = e => {
        const url = 'http://localhost:8080/calc/'+first+'/'+second+'/'+operator;
        const response = fetch(url)
            .then((response) => response.json())
            .then((data) => {
                setResult(data.value);
            });
        e.preventDefault();
    }
```

Call the backend

Set the result

Calculator.js

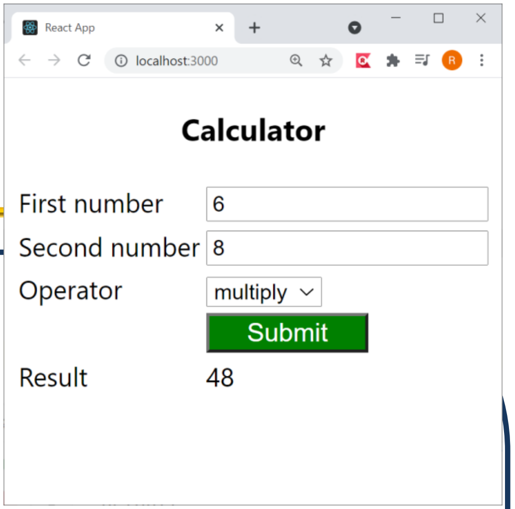
```
let calcpage = (
     <form>
         <h3>Calculator</h3>
         <table class="center">
             <tr>
                 <td>First number</td>
                 <td><input
                     type="text"
                     name="first"
                     value={first}
                     onChange={e => setFirst(e.target.value)} /></td>
             </tr>
             <tr>
                 <td>Second number</td>
                 <td><input
                     type="text"
                     name="second"
                     value={second}
                     onChange={e => setSecond(e.target.value)} /></td>
             </tr>
             <tr>
```



Calculator

First number	6
Second number	8
Operator	multiply
	Submit
Result	48

Calculator.js


React App

localhost:3000

Calculator

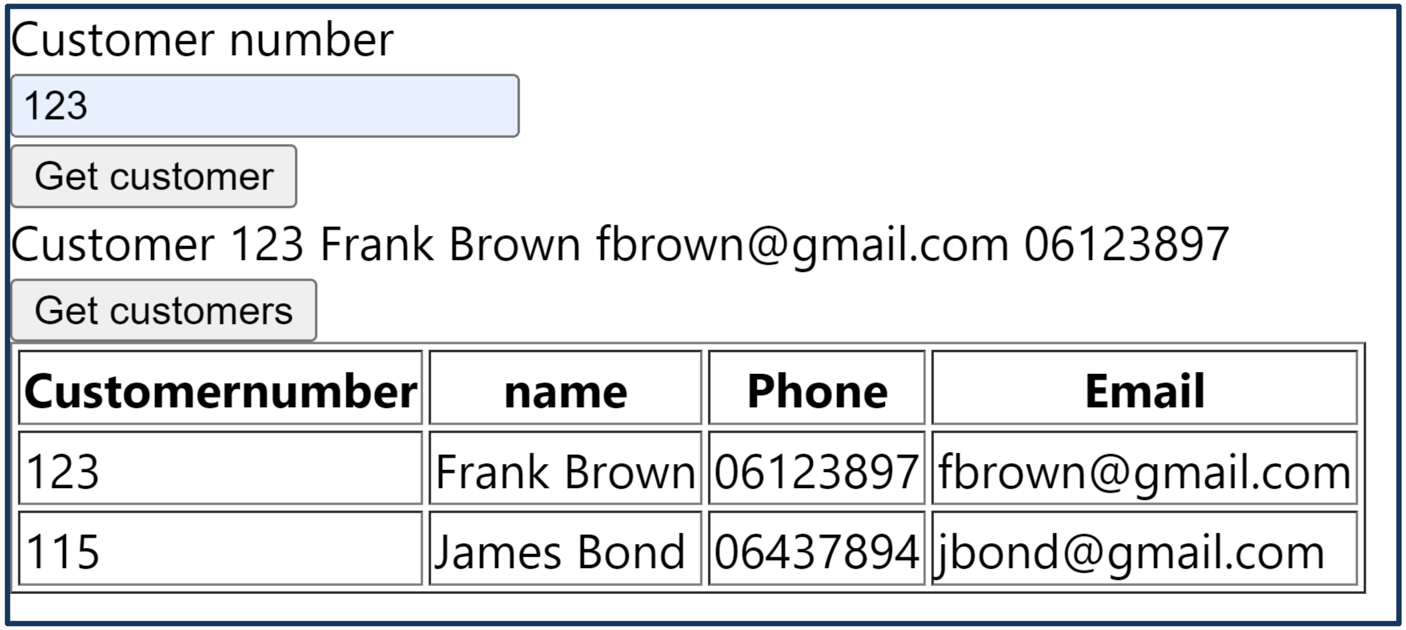
First number	6
Second number	8
Operator	multiply
	Submit
Result	48

```
                <td>Operator</td>
                <td><select
                    type="text"
                    name="operator"
                    value={operator}
                    onChange={e => setOperator(e.target.value)} >
                    <option>add</option>
                    <option>subtract</option>
                    <option>multiply</option>
                    <option>divide</option>
                    <option>clear</option>
                </select></td>
            </tr>
            <tr>
                <td></td>
                <td><button onClick={fetchBackend}>Submit</button></td>
            </tr>
            <tr>
                <td>Result</td>
                <td>{result}</td>
            </tr>
        </table>
    </form>
);
return calcpage;
```

REST back-end

```
@CrossOrigin(origins = "*")
@RestController
public class CustomerController {

   @GetMapping("/customers/{customernumber}")
   public ResponseEntity<?> getCustomer(@PathVariable String customernumber) {
      Customer customer = new Customer("123", "Frank Brown", "06123897","fbrown@gmail.com");
      return new ResponseEntity<Customer> (customer, HttpStatus.OK);
   }

   @GetMapping("/customers")
   public ResponseEntity<?> getCustomers() {
      Customers customers  = new Customers();
      customers.addCustomer(new Customer("123", "Frank Brown", "06123897","fbrown@gmail.com"));
      customers.addCustomer(new Customer("115", "James Bond", "06437894","jbond@gmail.com"));

      return new ResponseEntity<Customers> (customers, HttpStatus.OK);
   }
}
```

Customers front-end


Customer number

Customer 123 Frank Brown fbrown@gmail.com 06123897

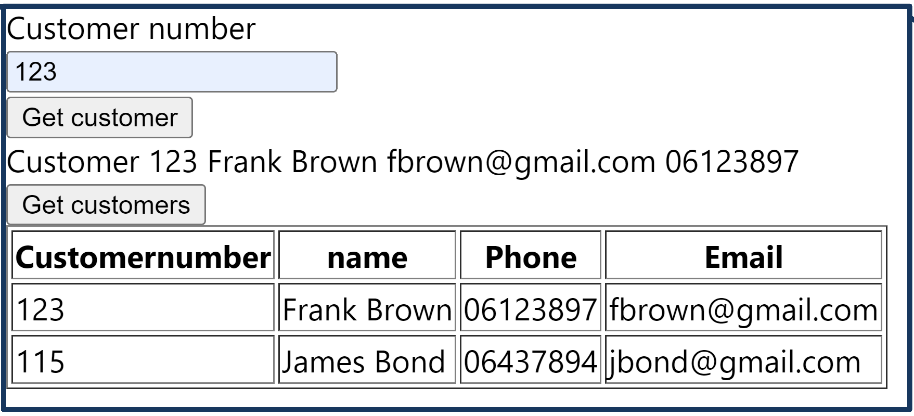
Customernumber	name	Phone	Email
123	Frank Brown	06123897	fbrown@gmail.com
115	James Bond	06437894	jbond@gmail.com

Customers front-end

```
export const Customers = () => {
  const cleanuser = { customernumber: "", name: "", phone: "", email: "" };
  const [customer, setCustomer] = useState(cleanuser);
  const [customers, setCustomers] = useState([]);
  const [customernumber, setCustomernumber] = useState("");

  const fetchCustomer = (e) => {
    const url = 'http://localhost:8080/customers/' + customernumber;
    const response = fetch(url)
      .then((response) => response.json())
      .then((data) => {
        setCustomer(data);
      })
  }

  const fetchCustomers = (e) => {
    const url = 'http://localhost:8080/customers';
    const response = fetch(url)
      .then((response) => response.json())
      .then((data) => {
        setCustomers(data.customers);
      })
  }
```

Customers front-end

```
return (
  <div>
    Customer number
    <div>
      <input
        type="text"
        name="customernumber"
        value={customernumber}
        onChange={e => setCustomernumber(e.target.value)} /></div>

    <div><button onClick={e => fetchCustomer()} >Get customer</button></div>

    <div>Customer
{customer.customernumber}  {customer.name}  {customer.email}  {customer.phone}</div>
```


Customer number

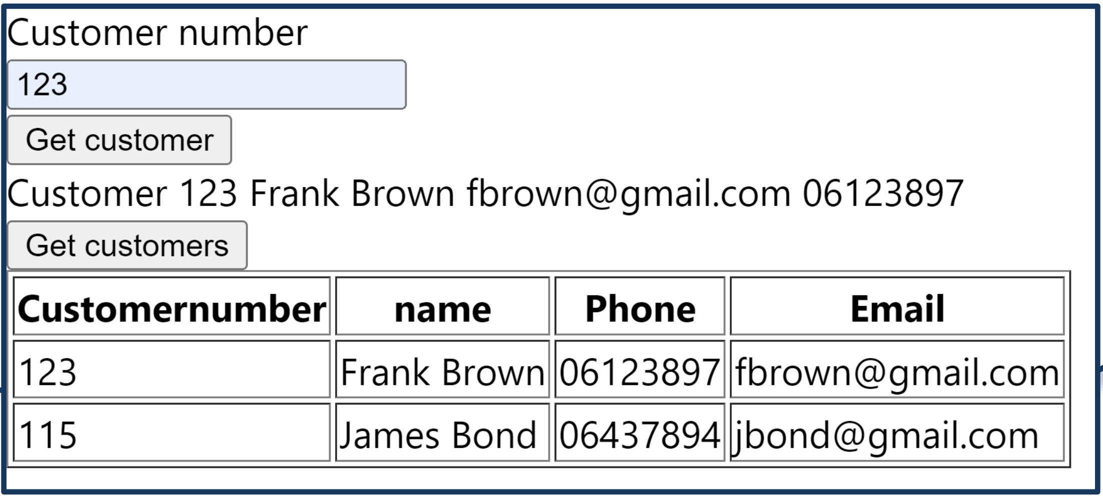
123

Customer 123 Frank Brown fbrown@gmail.com 06123897

Customernumber	name	Phone	Email
123	Frank Brown	06123897	fbrown@gmail.com
115	James Bond	06437894	jbond@gmail.com

Customers front-end

```
      <div><button onClick={e => fetchCustomers()} >Get customers</button></div>
      <div>
        <table border = {1} >
          <thead>
            <tr><th>Customernumber</th><th>name</th><th>Phone</th><th>Email</th></tr>
          </thead>
          <tbody>
            {customers.map(customer => (
              <tr key={customer.customernumber}>
                <td>{customer.customernumber}</td>
                <td>{customer.name}</td>
                <td>{customer.phone}</td>
                <td>{customer.email}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
};
```



Customer number

123

Customer 123 Frank Brown fbrown@gmail.com 06123897

Customernumber	name	Phone	Email
123	Frank Brown	06123897	fbrown@gmail.com
115	James Bond	06437894	jbond@gmail.com

Async-await

```
export const Customers = () => {
  const cleanuser = { customernumber: "", name: "", phone: "", email: "" };
  const [customer, setCustomer] = useState(cleanuser);
  const [customers, setCustomers] = useState([]);
  const [customernumber, setCustomernumber] = useState("");

  const  fetchCustomer = async() => {
    const url = 'http://localhost:8080/customers/' + customernumber;
    const response = await fetch(url);
    const data = await response.json();
    setCustomer(data);
  }

  const  fetchCustomers = async() => {
    const url = 'http://localhost:8080/customers';
    const response = await fetch(url);
    const data = await response.json();
    setCustomers(data.customers);
  }
```

BACKEND ACCESS WITH AXIOS

Calling the backend with axios

```
const fetchBackend = e => {
     const url = 'http://localhost:8080/calc/'+first+'/'+second+'/'+operator;
     const response = axios.get(url)
         .then((response) => {
             setResult(response.data.value);
         });
     e.preventDefault();
 }
```

Call the backend

Set the result

Axios vs. Fetch

- Advantages of Axios
  - Request and response interception
  - Streamlined error handling
  - Protection against XSRF
  - Support for upload progress
  - Response timeout
  - The ability to cancel requests
  - Support for older browsers
  - Automatic JSON data transformation

Fetch vs. Axios

```
const fetchBackend = e => {
      const url = 'http://localhost:8080/calc/'+first+'/'+second+'/'+operator;
      const response = fetch(url)
          .then((response) => response.json())
          .then((data) => {
              setResult(data.value);
          });
      e.preventDefault();
  }
```

Explicit JSON transformation

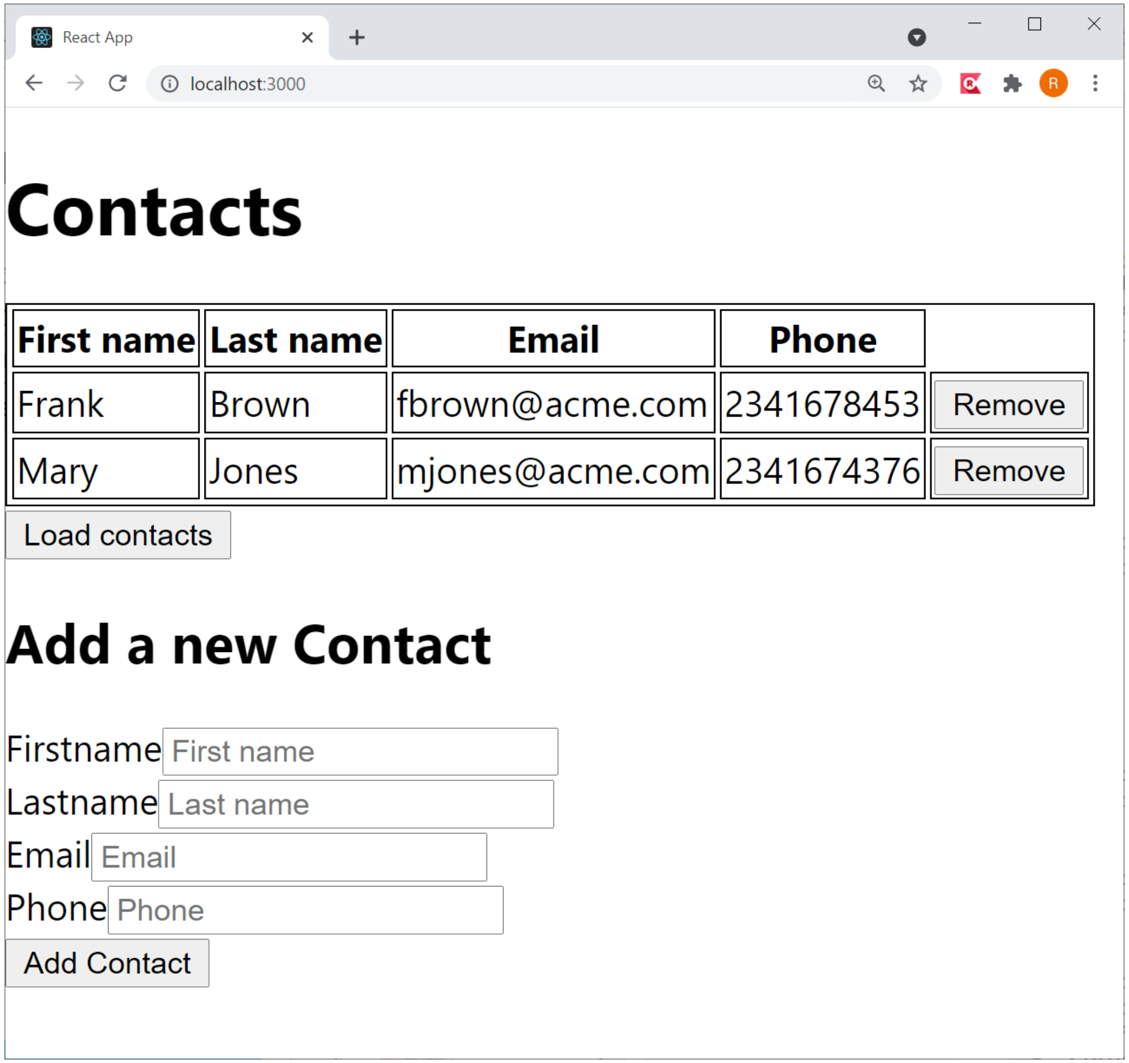
```
const fetchBackend = e => {
      const url = 'http://localhost:8080/calc/'+first+'/'+second+'/'+operator;
      const response = axios.get(url)
          .then((response) => {
              setResult(response.data.value);
          });
      e.preventDefault();
  }
```

Automatic JSON transformation

Contacts back-end

```
@RestController
@CrossOrigin
public class ContactController {
   private Map<String, Contact> contacts = new HashMap<String, Contact>();
   public ContactController() {
      contacts.put("Frank", new Contact("Frank", "Brown", "fbrown@acme.com", "2341678453"));
      contacts.put("Mary", new Contact("Mary", "Jones", "mjones@acme.com", "2341674376"));
   }

   @GetMapping("/contacts/{firstName}")
   public ResponseEntity<?> getContact(@PathVariable String firstName) {}

   @PostMapping("/contacts")
   public ResponseEntity<?> addContact(@RequestBody Contact contact) {}

   @DeleteMapping("/contacts/{firstName}")
   public ResponseEntity<?> deleteContact(@PathVariable String firstName) {

   }

   @PutMapping("/contacts/{firstName}")
   public ResponseEntity<?> updateContact(@PathVariable String firstName, @RequestBody Contact contact) {}

   @GetMapping("/contacts")
   public ResponseEntity<?> getAllContacts() {}
}
```

Contacts front-end



Contacts

First name	Last name	Email	Phone	
Frank	Brown	fbrown@acme.com	2341678453	Remove
Mary	Jones	mjones@acme.com	2341674376	Remove

Load contacts

Add a new Contact

Firstname	First name
Lastname	Last name
Email	Email
Phone	Phone

Add Contact

Contacts front-end

```
import React, { useState } from 'react';
import axios from 'axios';
export const Contacts = () => {
  const cleancontact = { firstName: "", lastName: "", email: "", phone: "" };
  const [contact, setContact] = useState(cleancontact);
  const [contactlist, setContactlist] = useState([]);

    const loadContacts = () => {
    const contacts = axios.get("http://localhost:8080/contacts")
      .then((response) => {
        console.log(response.data.contacts);
        setContactlist(response.data.contacts);
      });
  }
  const addContact = (contact) => {
    axios.post("http://localhost:8080/contacts", contact)
        .then((response) => {
          console.log("added contact "+response.data.firstname);
          loadContacts();
        }); //add user to the list
  }
  const removeContact = (e) => {
    let url = "http://localhost:8080/contacts/"+e.target.value;
    console.log("removing contact with url="+url);
    axios.delete(url)
        .then((response) => {
          console.log("removed contact "+response.headers);
          loadContacts();
        }); //remove user to the list
  }
```

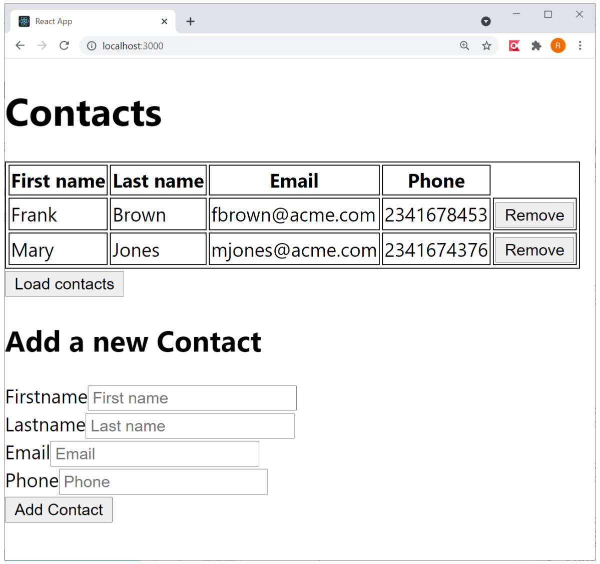
get

post

Reload contacts

delete

Reload contacts



Contacts

First name	Last name	Email	Phone	
Frank	Brown	fbrown@acme.com	2341678453	Remove
Mary	Jones	mjones@acme.com	2341674376	Remove

Load contacts

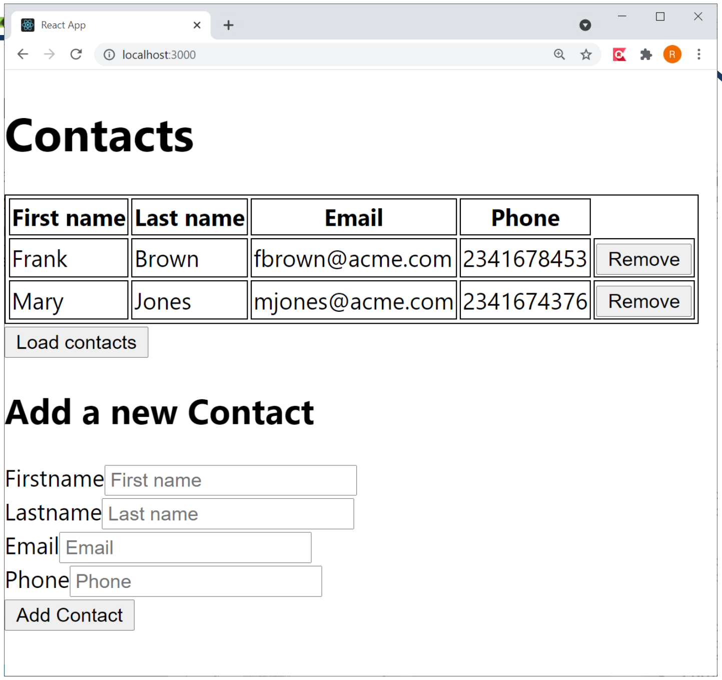
Add a new Contact

Firstname First name
Lastname Last name
Email Email
Phone Phone
Add Contact

Contacts front-end

```
const handleSubmit = (e) => {
  //prevent POST request
  e.preventDefault();
  console.log("handleSubmit");
  if (contact) {
    console.log("call the server");
    addContact(contact);
  }
  //clear user
  setContact(cleancontact);
  console.log("load contacts");
  loadContacts();
  console.log("load contacts done");
}

const handleFieldChange = (e) => {
  setContact({ ...contact, [e.target.name]: e.target.value });
}
```

Called when we click the **Add Contact** button



Contacts

First name	Last name	Email	Phone	
Frank	Brown	fbrown@acme.com	2341678453	Remove
Mary	Jones	mjones@acme.com	2341674376	Remove

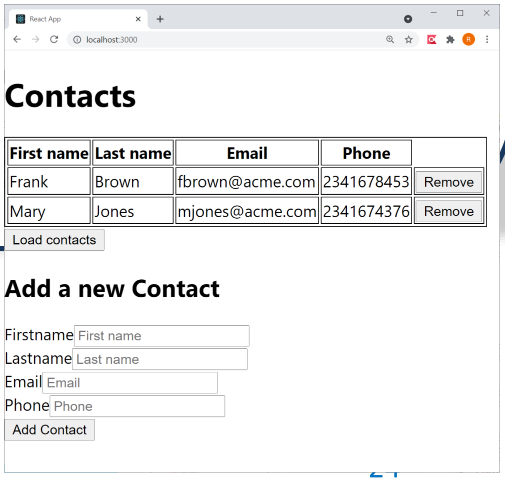
Load contacts

Add a new Contact

Firstname First name
Lastname Last name
Email Email
Phone Phone
Add Contact

Contacts front-end

```
return (
  <div>
    <h1>Contacts</h1>

    <table>
      <thead>
        <tr><th>First name</th><th>Last name</th><th>Email</th><th>Phone</th></tr>
      </thead>
      <tbody>
        {contactlist.map(contact => (
          <tr key={contact.firstName}>
            <td>{contact.firstName}</td>
            <td>{contact.lastName}</td>
            <td>{contact.email}</td>
            <td>{contact.phone}</td>
            <td><button onClick={removeContact} value={contact.firstName} >Remove</button></td>
          </tr>
        ))}
      </tbody>
    </table>
    <button onClick={loadContacts}>Load contacts</button>
```



Contacts

First name	Last name	Email	Phone	
Frank	Brown	fbrown@acme.com	2341678453	Remove
Mary	Jones	mjones@acme.com	2341674376	Remove

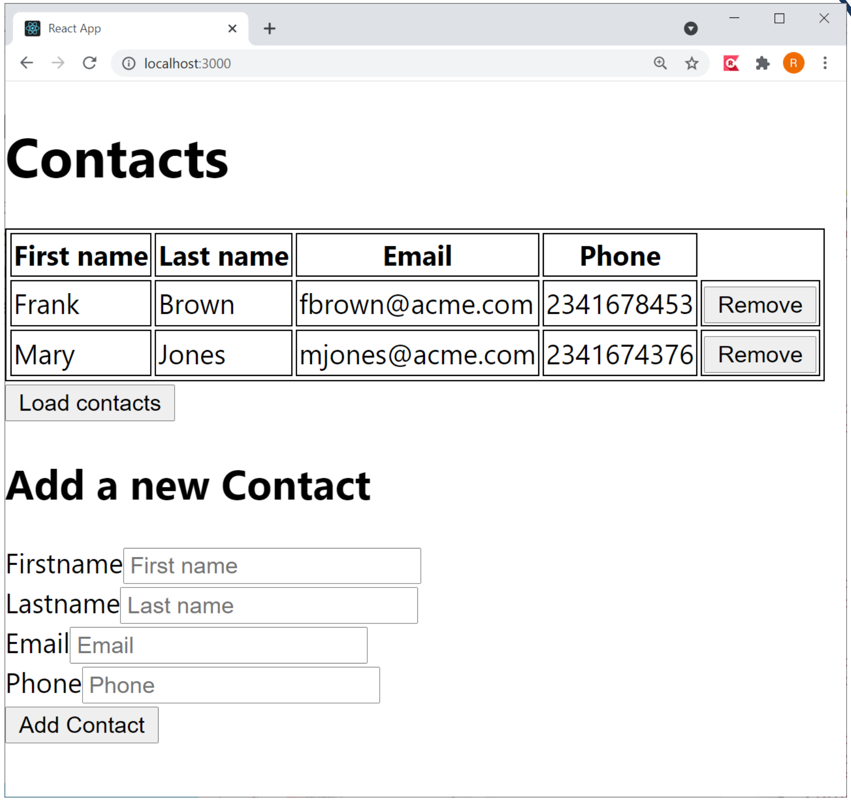
Load contacts

Add a new Contact

Firstname First name
Lastname Last name
Email Email
Phone Phone
Add Contact

Contacts front-end

```
<h2>Add a new Contact</h2>
   <form onSubmit={handleSubmit}>
     <div>
       Firstname
       <input
         type="text"
         placeholder="First name"
         name="firstName"
         value={contact.firstName}
         onChange={handleFieldChange} />
     </div>
     <div>
       Lastname
       <input
         type="text"
         placeholder="Last name"
         name="lastName"
         value={contact.lastName}
         onChange={handleFieldChange} />
     </div>
     <div>
       Email
       ...
     </div>
     <div>
       Phone
      ...
     </div>
     <button type="submit">Add Contact</button>
   </form>
 </div>
```



Contacts

First name	Last name	Email	Phone	
Frank	Brown	fbrown@acme.com	2341678453	Remove
Mary	Jones	mjones@acme.com	2341674376	Remove

Load contacts

Add a new Contact

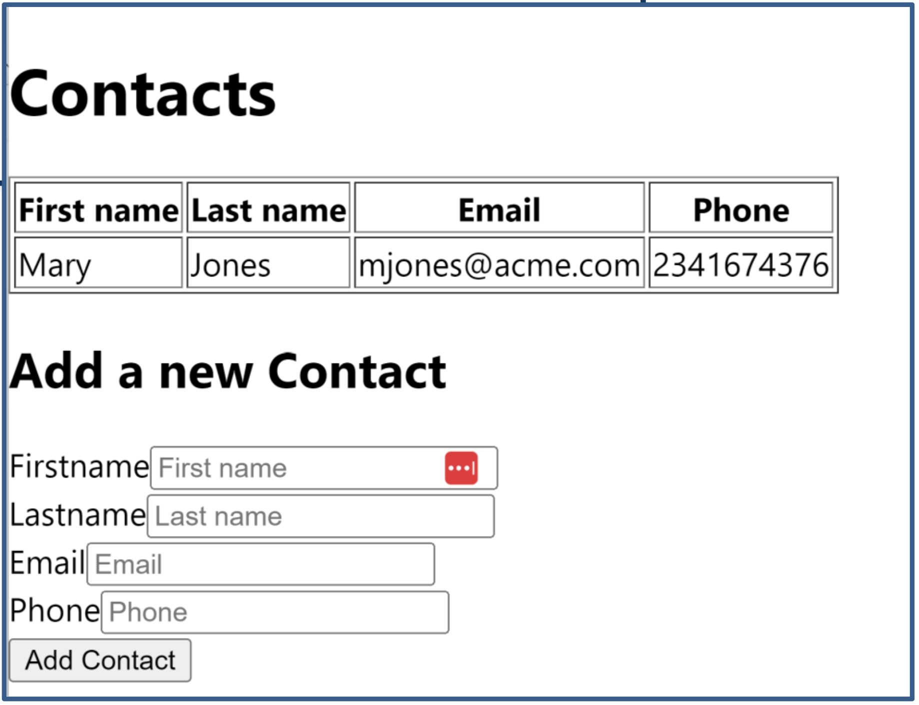
Firstname First name
Lastname Last name
Email Email
Phone Phone
Add Contact

Load contacts

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

export const Contacts = () => {
  const cleancontact = { firstName: "", lastName: "", email: "", phone: "" };
  const [contact, setContact] = useState(cleancontact);
  const [contactlist, setContactlist] = useState([]);

  React.useEffect(() => {
    loadContacts();
  }, []);
```

Call loadContacts() after the page is rendered

Load contacts when we load the page



Contacts

First name	Last name	Email	Phone
Mary	Jones	mjones@acme.com	2341674376

Add a new Contact

Firstname	First name
Lastname	Last name
Email	Email
Phone	Phone

Add Contact

Load contacts

```
const client = axios.create({
  baseURL: "http://localhost:8080/contacts"
});

const loadContacts = () => {
  const contacts = client.get()
    .then((response) => {
      setContactlist(response.data.contacts);
    });
}

const addContact = (contact) => {
  client.post("http://localhost:8080/contacts",contact)
      .then((response) => {
        loadContacts();
      }); //add user to the list
}

const removeContact = (e) => {
  client.delete("/"+e.target.value)
      .then((response) => {
        loadContacts();
      }); //remove user from the list
}
```

Create a react client

Use the client

POST still needs the URL

Use the client

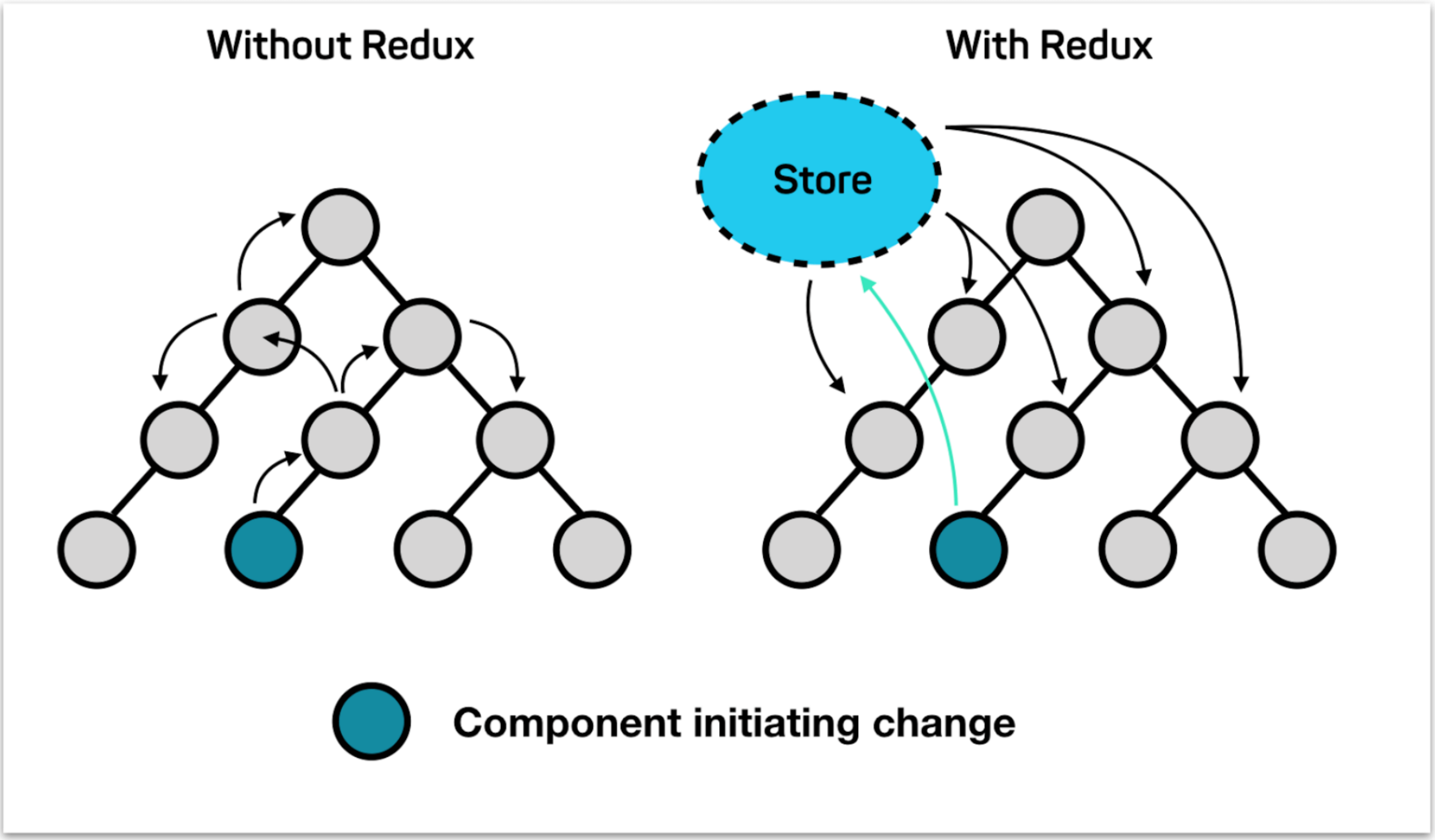
REDUX

What is redux?

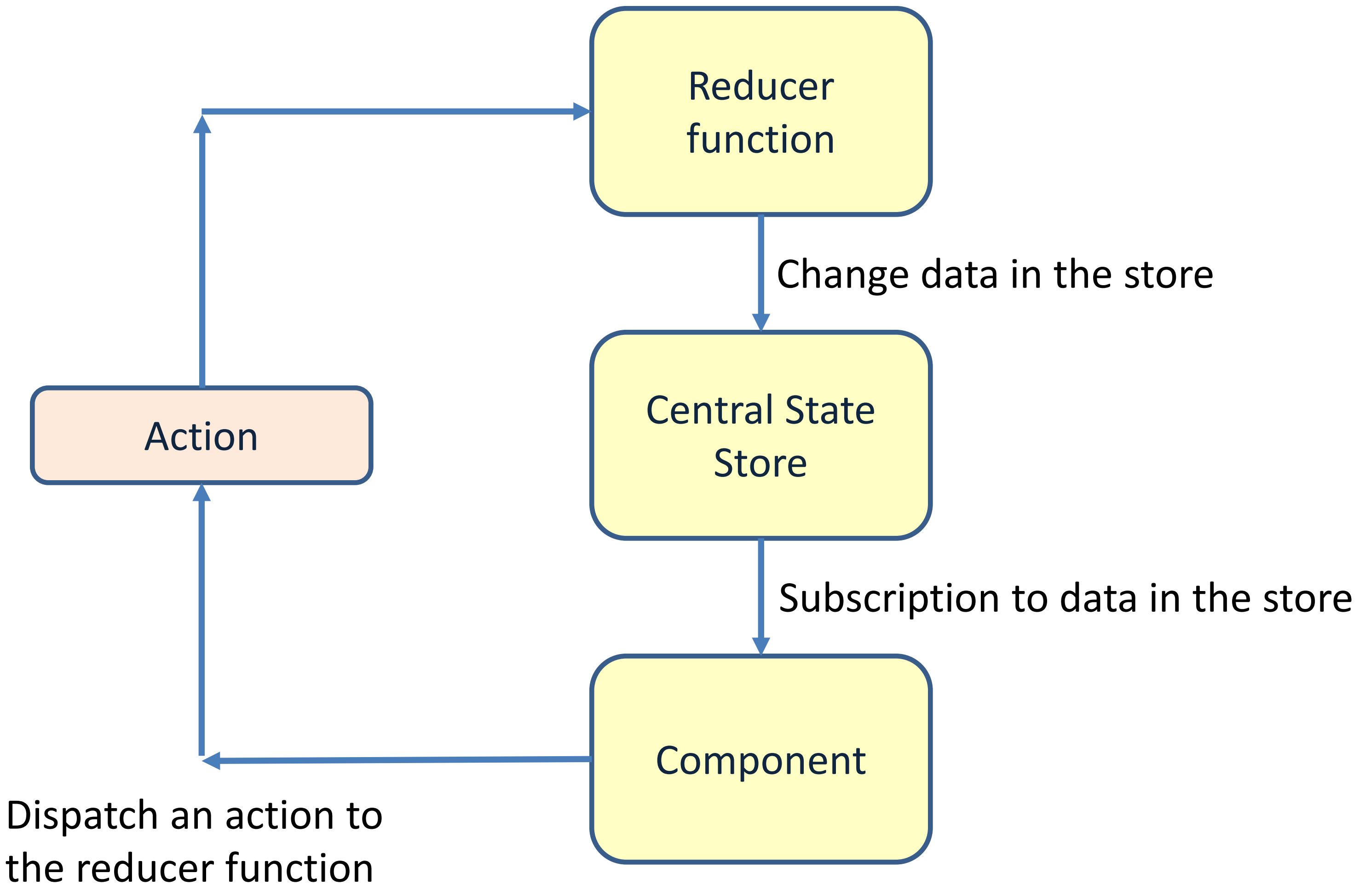
- State management system for app-wide state.

- Local state within 1 component
  - useState()
- State shared between components
  - Placing state in parent component
  - Passing state to next component
  - Redux

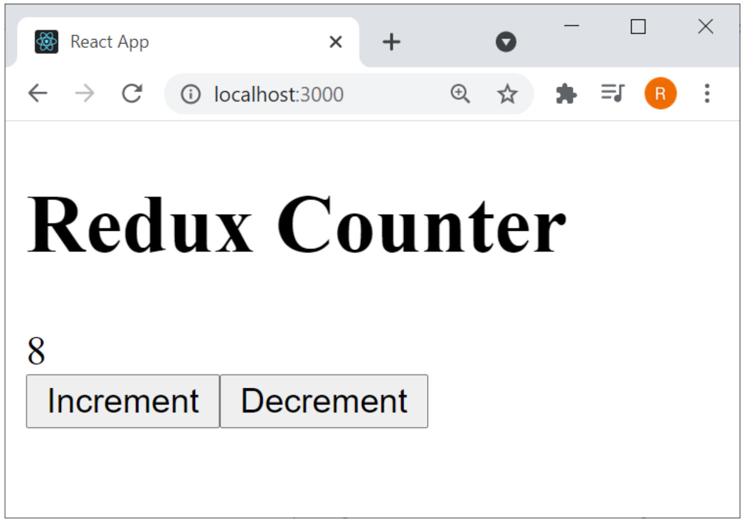
Why Redux?


Without Redux
With Redux
Store
Component initiating change

How does redux work?


Reducer function
Change data in the store
Central State Store
Subscription to data in the store
Component
Action
Dispatch an action to the reducer function

Simple counter



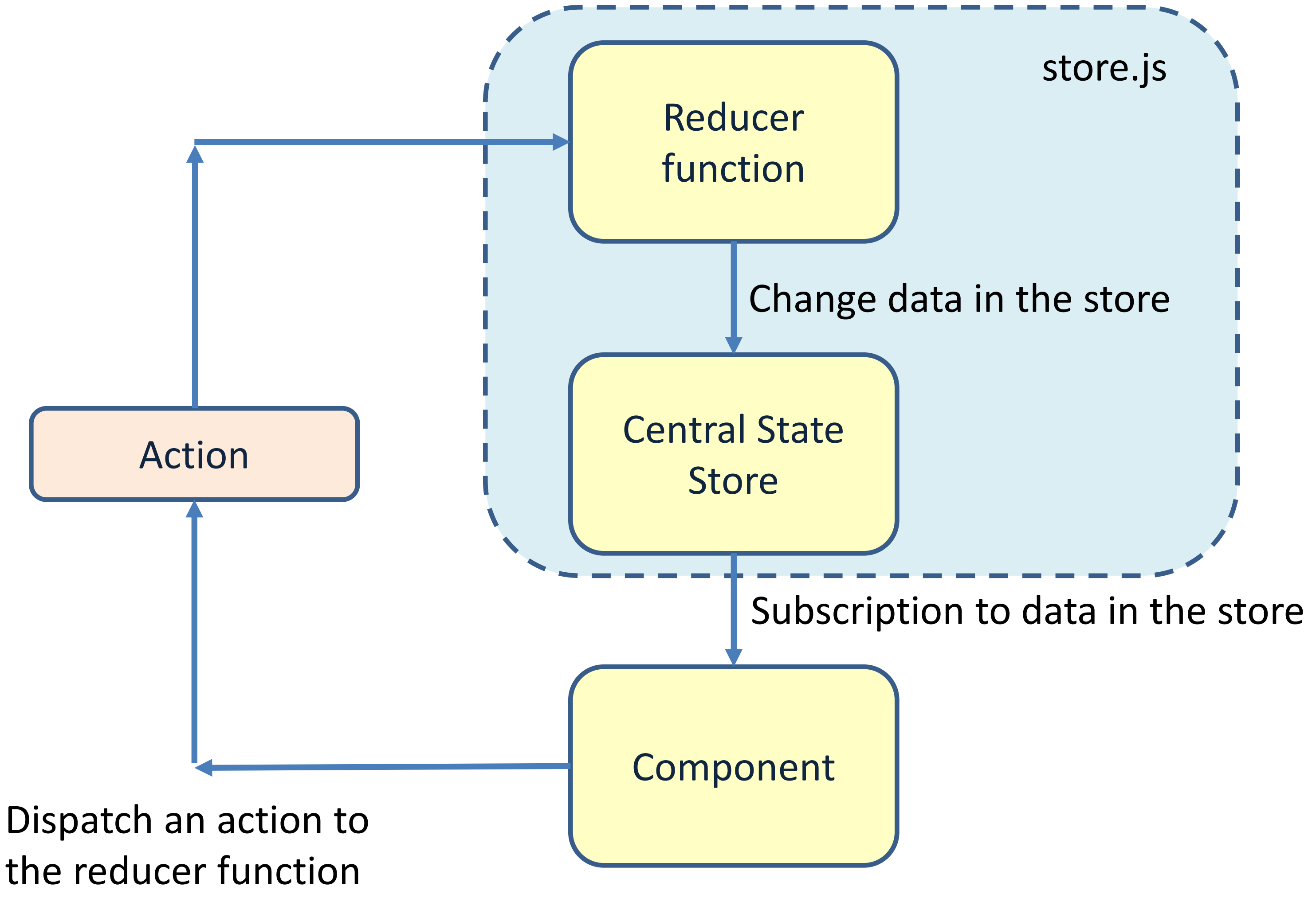
Redux Counter

8

Create a react redux app

1. Create a store
2. Make the store available to the whole app
3. Let the component use the store

1. Create the store



1. Create the store

store/store.js

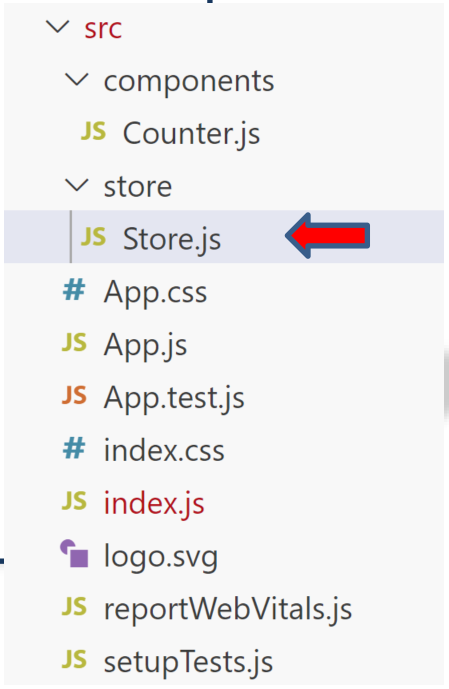
```
import { configureStore } from "@reduxjs/toolkit";

const counterReducer = (state = {counter : 0 }, action ) => {
  if (action.type === 'increment'){
      return { counter : state.counter + 1};
  }
  if (action.type === 'decrement'){
    return { counter : state.counter - 1};
  }
  return state;
}


const store = configureStore({
  reducer: counterReducer
});


export default store;
```

Initial state

Reducer function for the store

Create the store

Export the store



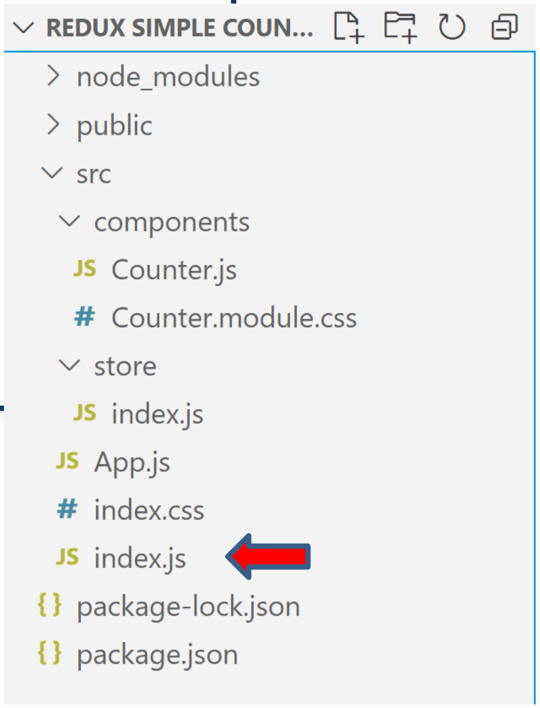
2. Make the store available to the whole app

index.js

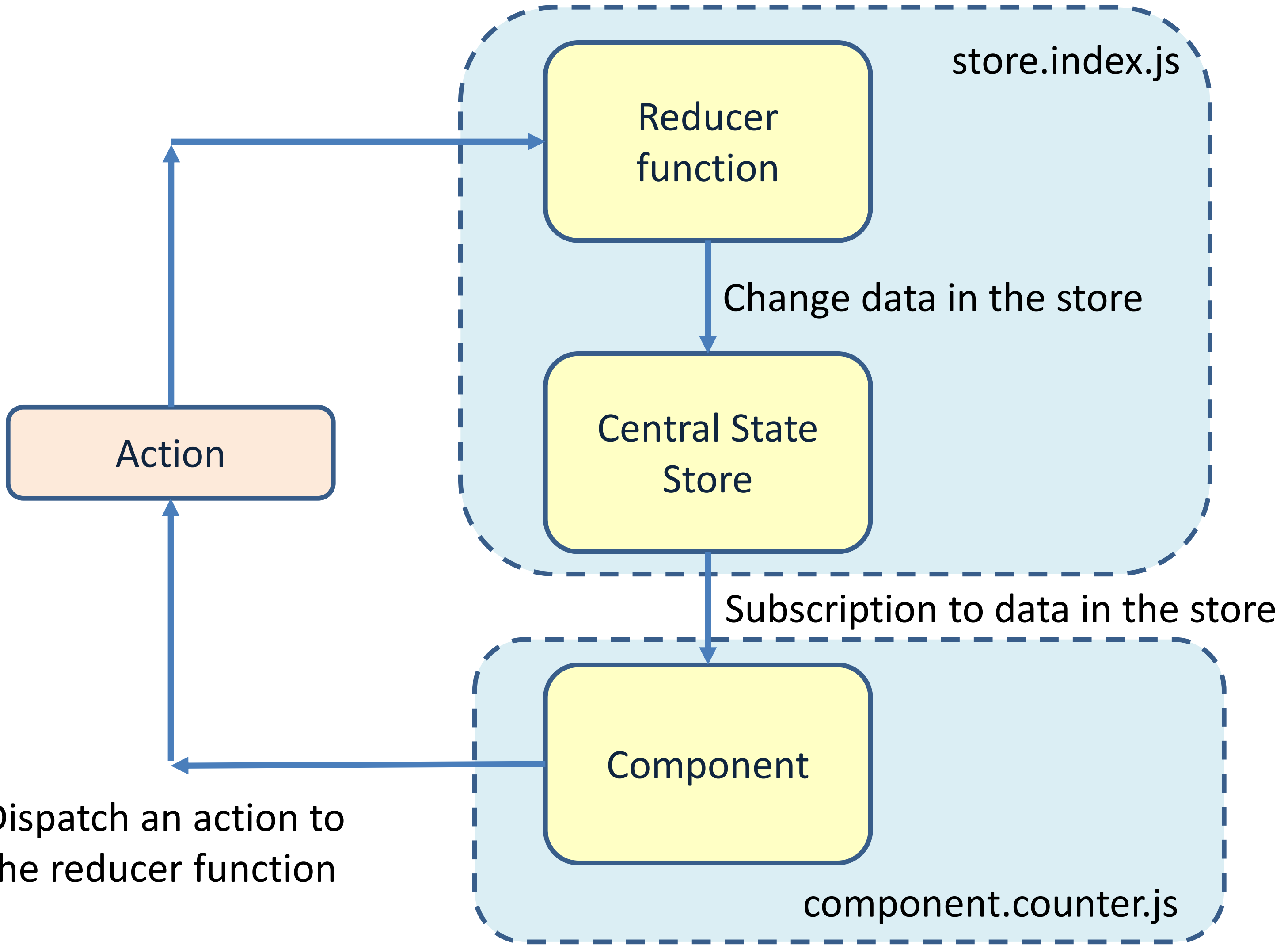
```
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux'

import './index.css';
import App from './App';
import store from './store/store'

ReactDOM.render(
    <Provider store={store}>
        <App />
    </Provider>,
    document.getElementById('root')
);
```

Provide the store at the highest level of the application structure

Provide the store to all child components of the App



3. Let the component use the store



3. Let the component use the store

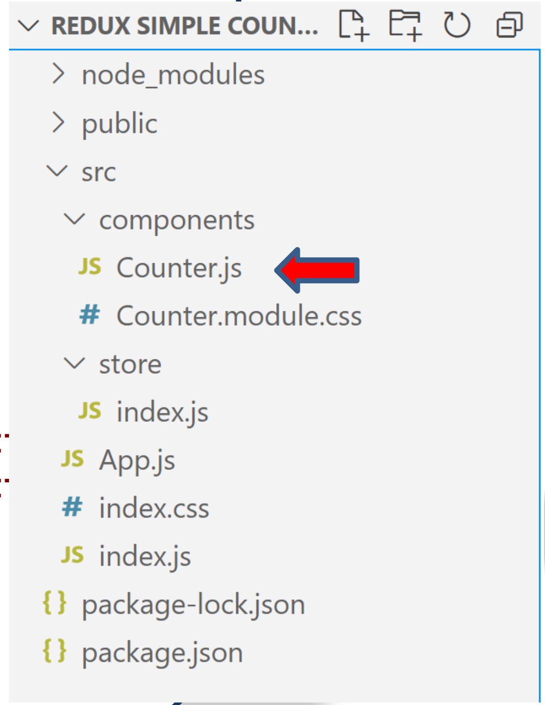
components/counter.js

```
import { useSelector, useDispatch } from 'react-redux';

const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);

  const incrementHandler = () => {
    dispatch({ type : 'increment'});
  }
  const decrementHandler = () => {
    dispatch({ type : 'decrement'});
  }
  return (
    <div>
      <h1>Redux Counter</h1>
      <div>{counter}</div>
      <div>
        <button onClick={incrementHandler}>Increment</but
        <button onClick={decrementHandler}>Decrement</but
      </div>
    </div>
  );
};
export default Counter;
```

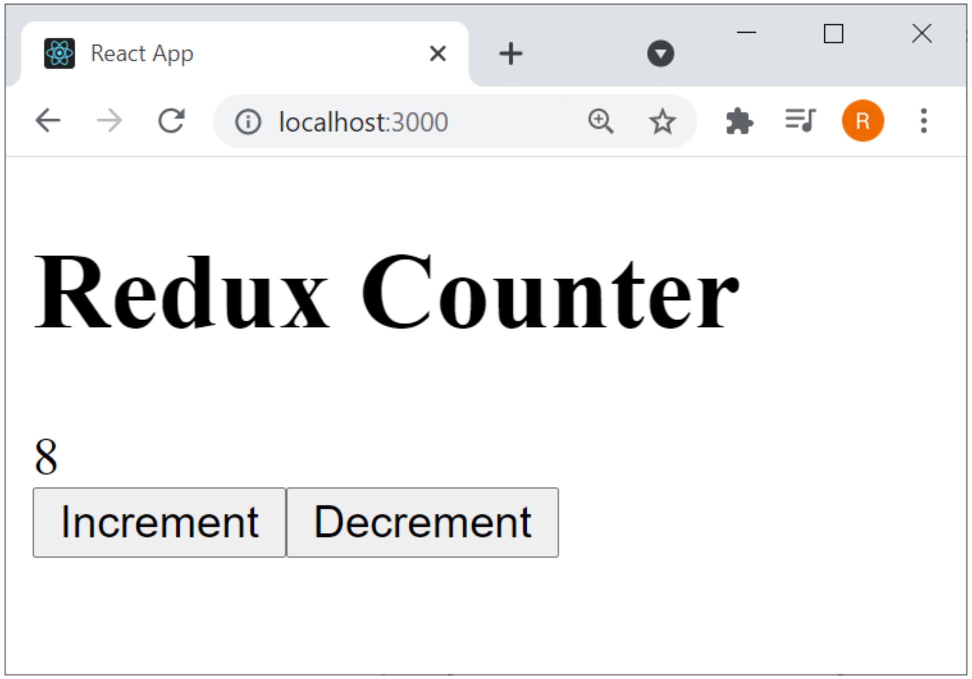
Select state.counter from the store

The counter value gets a subscription to the provided selector

Dispatch the 'decrement' action to the reducer function in the store


```
REDUX SIMPLE COUN...
  > node_modules
  > public
  v src
    v components
      JS Counter.js
      # Counter.module.css
    v store
      JS index.js
    JS App.js
    # index.css
    JS index.js
  {} package-lock.json
  {} package.json
```

Simple Counter app

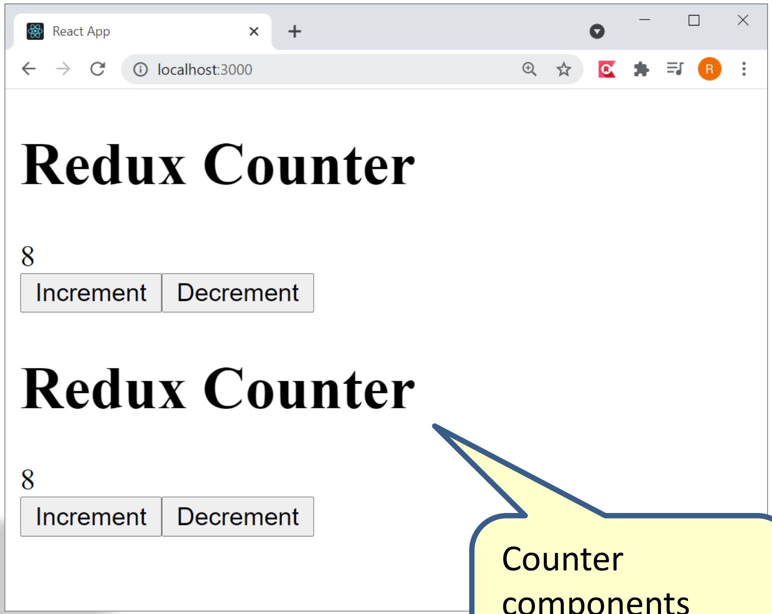


Redux Counter

8

2 Counter components

```
import Counter from './components/Counter';


function App() {
  return (
    <div>
      <Counter />
      <Counter />
    </div>
  );
}

export default App;
```



Redux Counter

8

Increment Decrement

Redux Counter

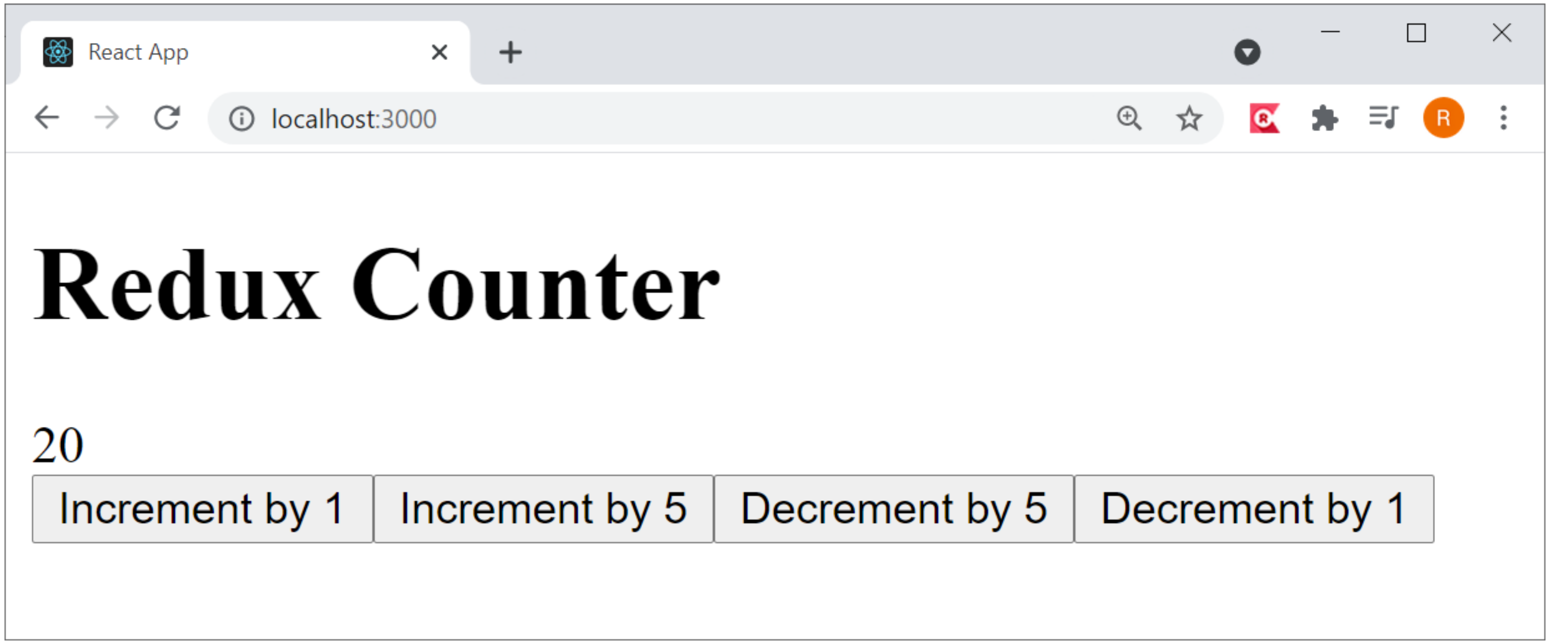
8

Increment Decrement

Counter components share the same state

ADD A PAYLOAD TO THE ACTION

Counter with a value



Redux Counter

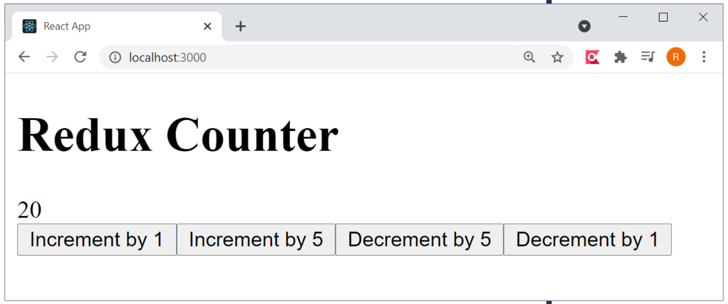
20

Counter.js

components/counter.js

```
const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);
  const incrementHandler = () => {
    dispatch({ type : 'increment'});
  }
  const increaseHandler = () => {
    dispatch({ type : 'increase', amount : 5});
  }
  const decrementHandler = () => {
    dispatch({ type : 'decrement'});
  }
  const decreaseHandler = () => {
    dispatch({ type : 'decrease', amount : 5});
  }
  return (
    <div>
      <h1>Redux Counter</h1>
      <div>{counter}</div>
      <div>
        <button onClick={incrementHandler}>Increment by 1</button>
        <button onClick={increaseHandler}>Increment by 5</button>
        <button onClick={decreaseHandler}>Decrement by 5</button>
        <button onClick={decrementHandler}>Decrement by 1</button>
      </div>
    </div>
  );
```

Add a payload to the action



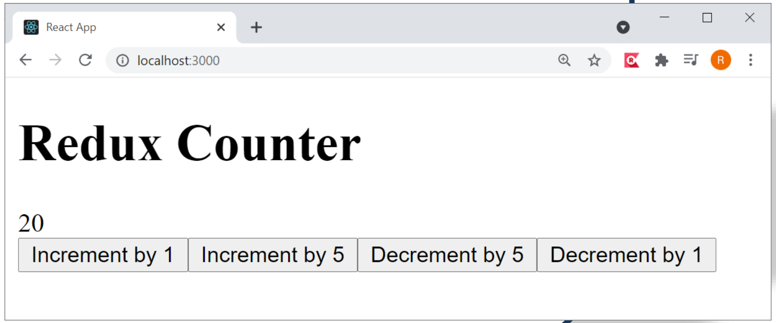
Redux Counter

20

store.js

```
import { configureStore } from "@reduxjs/toolkit";

const counterReducer = (state = { counter: 0 }, action) => {
  if (action.type === 'increment') {
    return { counter: state.counter + 1 };
  }
  if (action.type === 'increase') {
    return { counter: state.counter + action.amount };
  }
  if (action.type === 'decrement') {
    return { counter: state.counter - 1 };
  }
  if (action.type === 'decrease') {
    return { counter: state.counter - action.amount };
  }
  return state;
}

const store = configureStore({
  reducer: counterReducer
});

export default store;
```

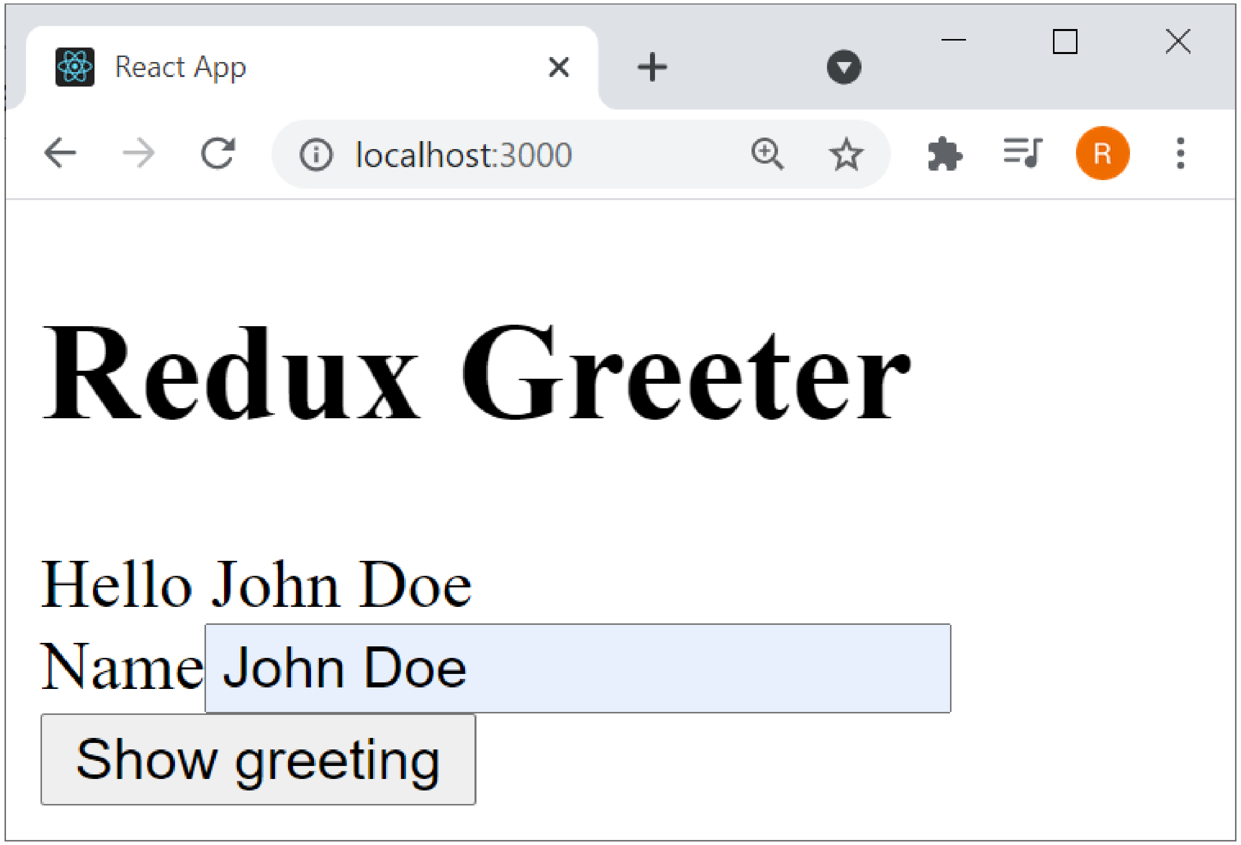
Get the payload from the action



Redux Counter

20

Simple Greeter App



Redux Greeter

Hello John Doe

Name John Doe

Greeter.js

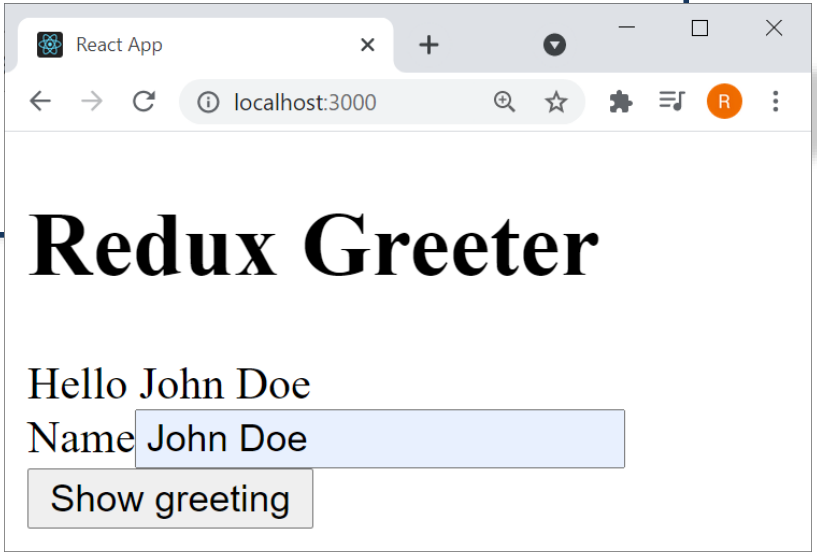
components/Greeter.js

```
import { useSelector, useDispatch } from 'react-redux';
import { useState } from 'react';
export const Greeter = () => {
  const [name, setName] = useState('');
  const dispatch = useDispatch();
  const greetingMessage = useSelector(state => state.greeting);
  const greetingHandler = () => {
    dispatch({ type : 'getgreeting', name : name });
  }
  return (
    <div>
      <h1>Redux Greeter</h1>
      <div>{greetingMessage}</div>
      <div>
      <div>
          Name
          <input
            type="text"
            placeholder="Your name"
            name="name"
            value={name}
            onChange={e => setName(e.target.value)} />
        </div>
        <button onClick={greetingHandler}>Show greeting</button>
      </div>
    </div>
  );
```

store.js

Initial state

```
import { configureStore } from "@reduxjs/toolkit";

const reducer = (state = {greeting : 'Hello' }, action ) => {
  if (action.type === 'getgreeting'){
      return { greeting : "Hello "+action.name};
  }
  return state;
}

const store = configureStore({
  reducer: reducer
});

export default store;
```



Redux Greeter

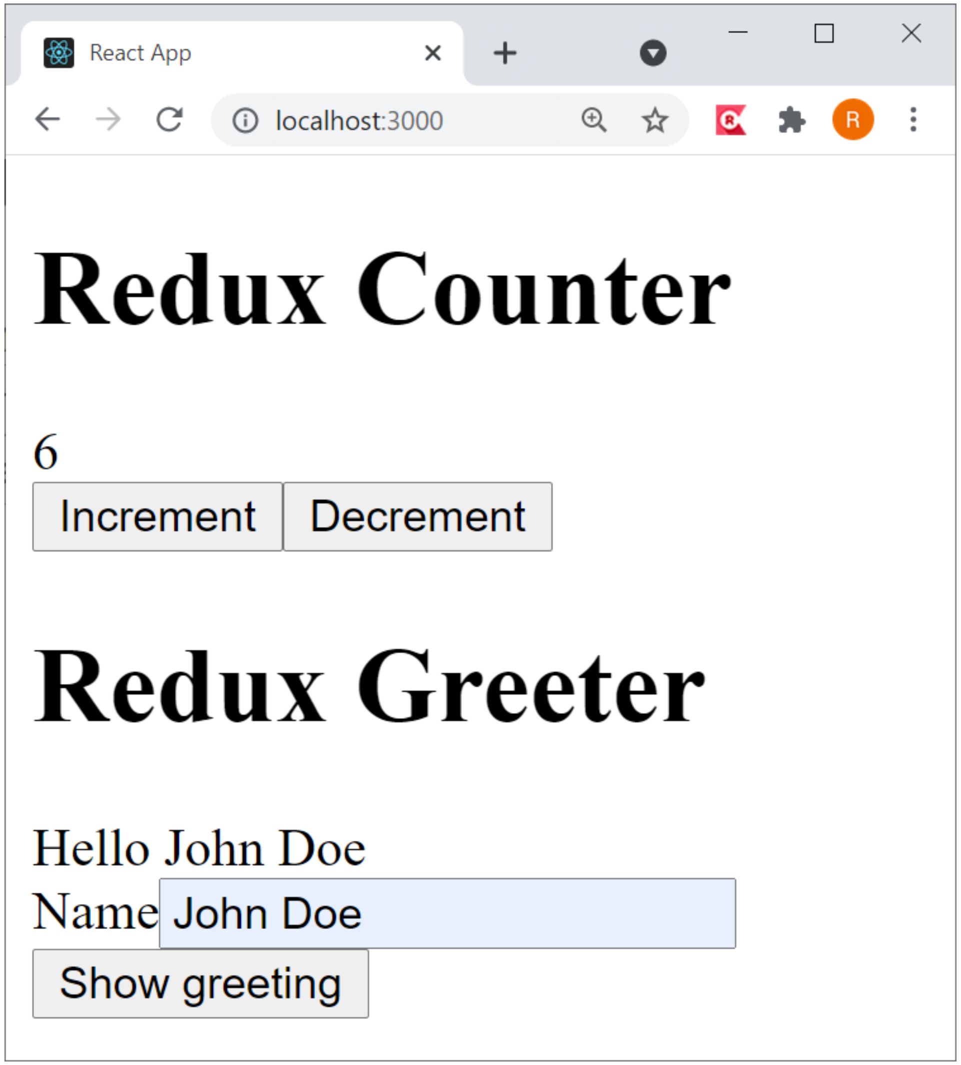
Hello John Doe

Name John Doe

Show greeting

MULTIPLE STATE PROPERTIES

Multiple state properties



React App

localhost:3000

Redux Counter

6

Redux Greeter

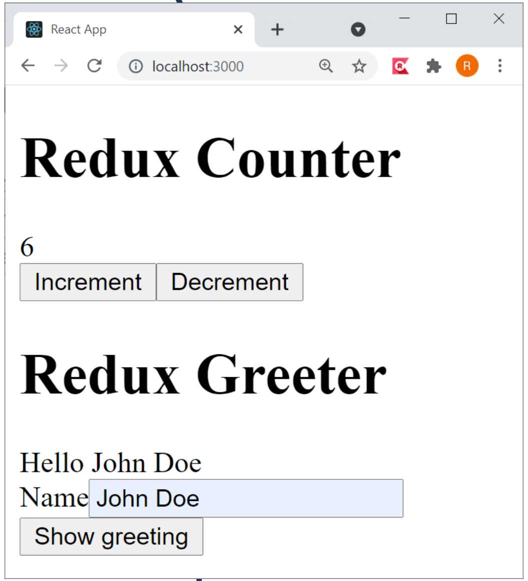
Hello John Doe

Name John Doe

Counter.js

components/Counter.js

```
import { useSelector, useDispatch } from 'react-redux';

const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);

  const incrementHandler = () => {
    dispatch({ type : 'increment'});
  }
  const decrementHandler = () => {
    dispatch({ type : 'decrement'});
  }
  return (
    <div>
      <h1>Redux Counter</h1>
      <div>{counter}</div>
      <div>
        <button onClick={incrementHandler}>Increment</button>
        <button onClick={decrementHandler}>Decrement</button>
      </div>
    </div>
  );
};
export default Counter;
```



Redux Counter

6

Redux Greeter

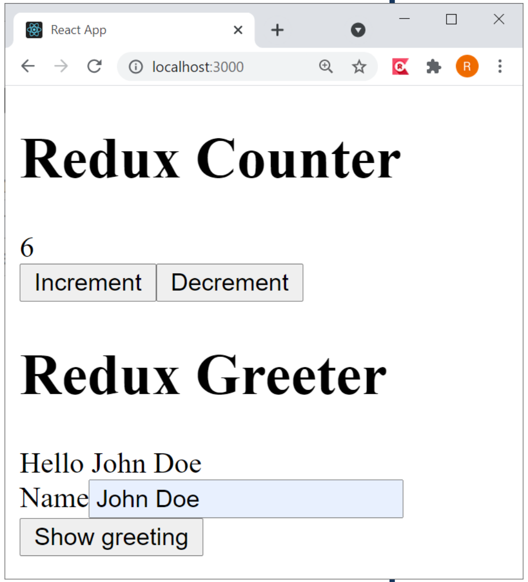
Hello John Doe

Name John Doe

Greeter.js

components/greeter.js

```
import { useSelector, useDispatch } from 'react-redux';
import { useState } from 'react';
export const Greeter = () => {
  const [name, setName] = useState('');
  const dispatch = useDispatch();
  const greetingMessage = useSelector(state => state.greeting);
  const greetingHandler = () => {
    dispatch({ type : 'getgreeting', name : name });
  }
  return (
    <div>
      <h1>Redux Greeter</h1>
      <div>{greetingMessage}</div>
      <div>
      <div>
          Name
          <input
            type="text"
            placeholder="Your name"
            name="name"
            value={name}
            onChange={e => setName(e.target.value)} />
        </div>
        <button onClick={greetingHandler}>Show greeting</button>
      </div>
    </div>
  );
```



Redux Counter

6

Redux Greeter

Hello John Doe

Name John Doe

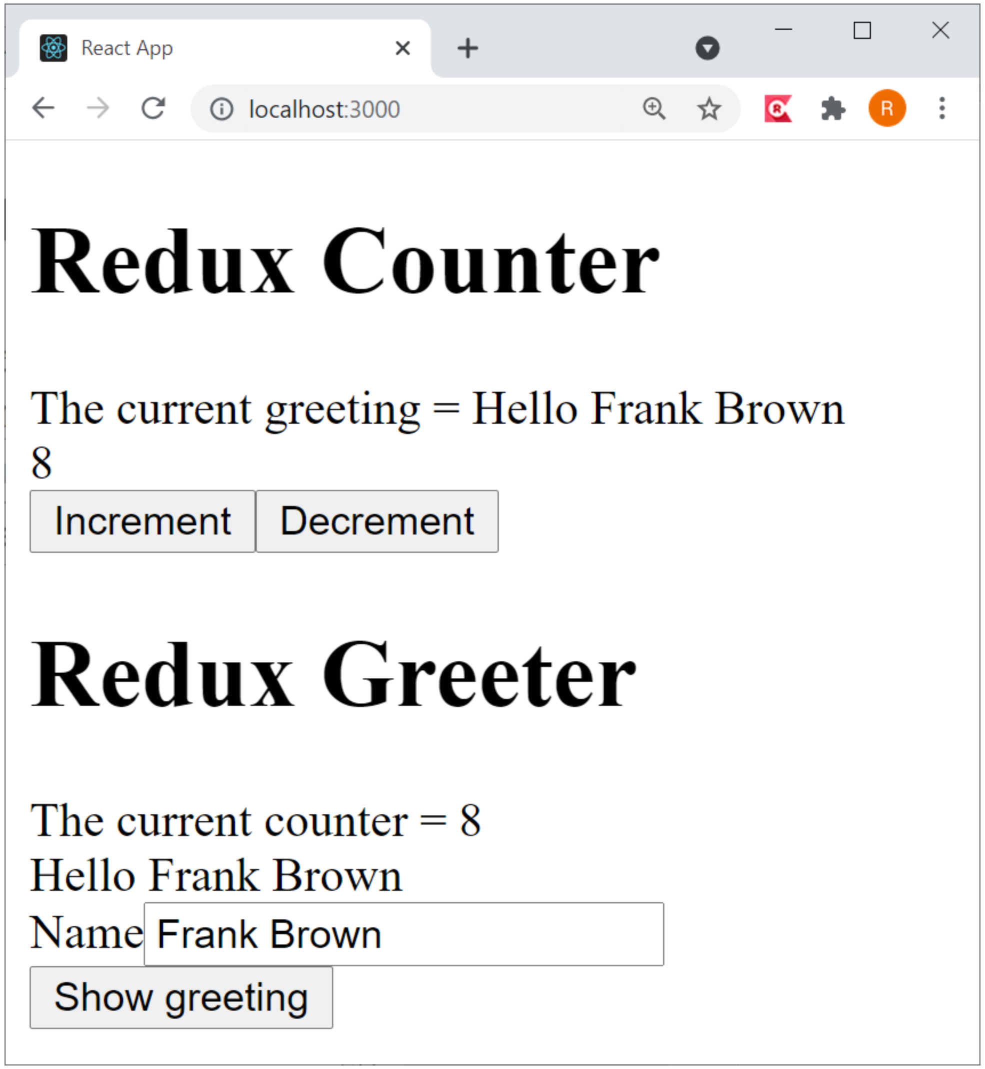
store.js

State contains a counter and a greeting

```
const initalstate = { counter: 0, greeting: 'Hello' };
const reducer = (state = initalstate, action) => {
  if (action.type === 'increment') {
    return {
      counter: state.counter + 1,
      greeting: state.greeting
    };
  }
  if (action.type === 'decrement') {
    return {
      counter: state.counter - 1,
      greeting: state.greeting
    };
  }
  if (action.type === 'getgreeting') {
    return {
      counter: state.counter,
      greeting: "Hello " + action.name
    };
  }
  return state;
}
const store = configureStore({
  reducer: reducer
});
export default store;
```

ALWAYS return a newly created state

Get state from the store



Redux Counter

The current greeting = Hello Frank Brown
8

Redux Greeter

The current counter = 8
Hello Frank Brown
Name Frank Brown

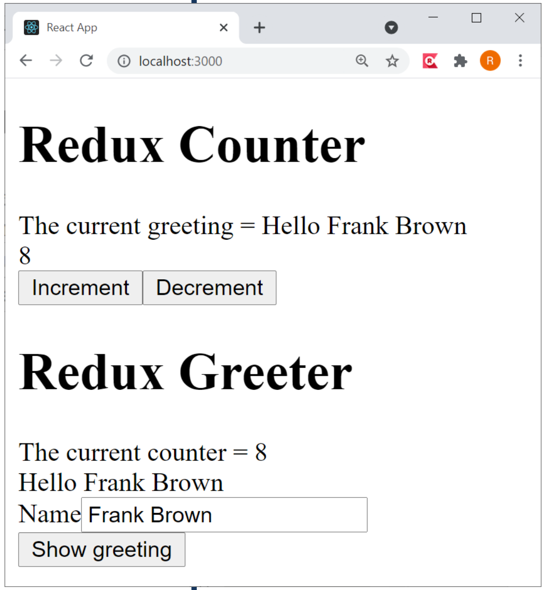
Counter.js

components/Counter.js

```
export const Counter = () => {
const dispatch = useDispatch();
const counter = useSelector(state => state.counter);
const greeting = useSelector(state => state.greeting);

const incrementHandler = () => {
  dispatch({ type : 'increment'});
}

const decrementHandler = () => {
  dispatch({ type : 'decrement'});
}

return (
  <div>
    <h1>Redux Counter</h1>
    <div>The current greeting = {greeting}</div>
    <div>{counter}</div>
    <div>
      <button onClick={incrementHandler}>Increment</button>
      <button onClick={decrementHandler}>Decrement</button>
    </div>
  </div>
);
};
```

Get greeting state

Show greeting state



Redux Counter

The current greeting = Hello Frank Brown
8

Redux Greeter

The current counter = 8
Hello Frank Brown
Name Frank Brown

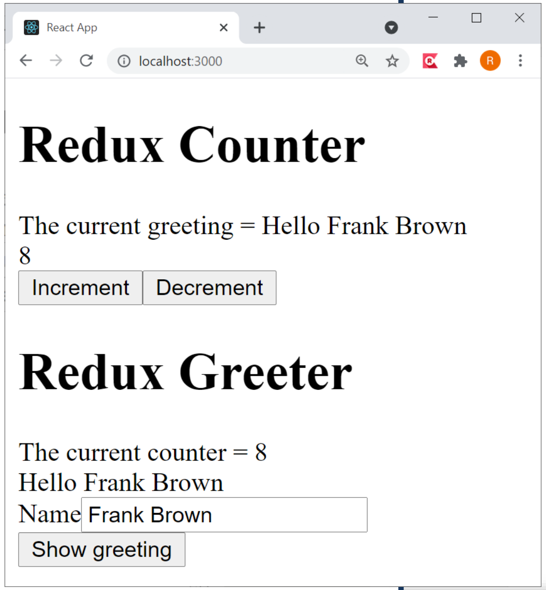
Greeter.js

components/Greeter.js

```
export const Greeter = () => {
const [name, setName] = useState('');
const dispatch = useDispatch();
const greetingMessage = useSelector(state => state.greeting);
const counter = useSelector(state => state.counter);

const greetingHandler = () => {
  dispatch({ type : 'getgreeting', name : name });
}
return (
  <div>
    <h1>Redux Greeter</h1>
    <div>The current counter = {counter}</div>
    <div>{greetingMessage}</div>
    <div>
    <div>
        Name
        <input
          type="text"
          placeholder="Your name"
          name="name"
          value={name}
          onChange={e => setName(e.target.value)} />
      </div>
      <button onClick={greetingHandler}>Show greeting</button>
    </div>
  </div>
```

Get counter state

Show counter state



Redux Counter

The current greeting = Hello Frank Brown
8

Redux Greeter

The current counter = 8
Hello Frank Brown
Name Frank Brown