

Lesson 11

REACT OBJECTS LISTS VALIDATION

```
let user = { name: "John Doe", age: 10 }; let copiedUser = { ...user };
```

Spread operator

```
let studentNames = ["Daniel", "Jane", "Joe"];  
let names = [...studentNames];
```

Copy an array

```
let user = { name: "John Doe", age: 10 };  
let copiedUser = { ...user };
```

Copy an object

Spread operator

```
let maleNames = ["Daniel", "Peter", "Joe"];  
let femaleNames = ["Sandra", "Lucy", "Jane"];  
let otherNames = ["Bill", "Jill"];
```

Array
concatenation

```
let moreNames = [...otherNames, ...femaleNames, ...maleNames];  
let names = [...moreNames, "Ben", "Fred"];
```

```
let userName = { name: "John Doe" };  
let userSex = { sex: "Male" };
```

```
let user = { ...userName, ...userSex };
```

Merge objects

WORKING WITH STATE OBJECTS

Routing with state

React App

localhost:3000

Page 1

Name

Next

React App

localhost:3000/paget...

Username: John Doe

Page 2

Address

Next

React App

localhost:3000/paget...

Username: John Doe

Address: Mainstreet 1 Chicago

Page 3

Creditcard number

Next

React App

localhost:3000/pagef...

Thank you for your order!

Username: John Doe

Address: Mainstreet 1 Chicago

Creditcard: 97865435321

State values: pageone.js

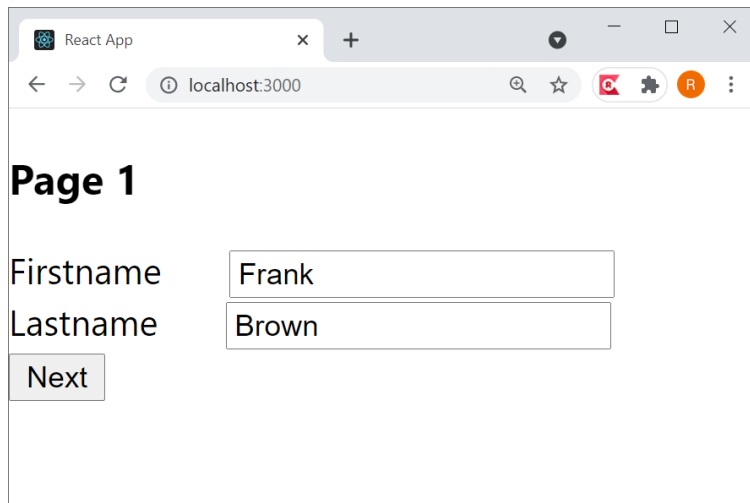
```
import React, { useState } from 'react';

export const Pageone = (props) => {
  const [firstname, setFirstname] = useState("");
  const [lastname, setLastname] = useState("");

  const handleOnSubmit = () => {
    props.history.push("/pagetwo", {firstname:firstname, lastname : lastname});
  }
}
```

What if we have many form fields?

Pass all state values



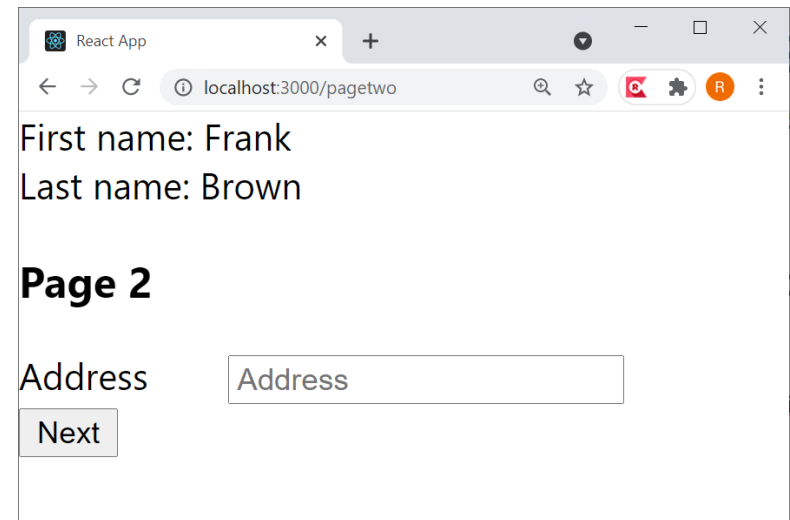
React App

localhost:3000

Page 1

Firstname

Lastname



React App

localhost:3000/pagetwo

First name: Frank

Last name: Brown

Page 2

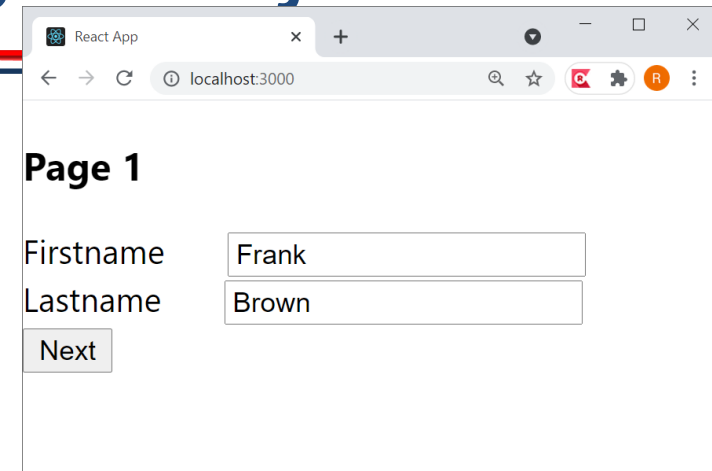
Address

State values: pageone.js

```
let page1 = (  
  <>  
    <h3>Page 1</h3>  
    <div>  
      First name  
      <input  
        type="text"  
        placeholder="First name"  
        value={firstname}  
        onChange={e => setFirstname(e.target.value)} />  
    </div>  
    <div>  
      Last name  
      <input  
        type="text"  
        placeholder="Last name"  
        value={lastname}  
        onChange={e => setLastname(e.target.value)} />  
    </div>  
    <button onClick={handleOnSubmit}>Next</button>  
  </>  
>);  
return page1;  
>}
```

value

Set the value



The screenshot shows a web browser window titled 'React App' at 'localhost:3000'. The page content is titled 'Page 1'. It contains two text input fields: 'Firstname' with the value 'Frank' and 'Lastname' with the value 'Brown'. Below these fields is a 'Next' button.

State objects: pageone.js

```
export const Pageone = () => {  
  const navigate = useNavigate();  
  const [user, setUser] = useState({  
    firstname: "",  
    lastname: "",  
    address: "",  
    creditcard: ""  
  })  
  
  const handleOnSubmit = () => {  
    console.log("p1 user="+user.firstname);  
    navigate("/pagetwo", {state: user});  
  }  
}
```

user is an object

Pass the whole
object to the
next page

React App

localhost:3000

Page 1

Firstname

Lastname

React App

localhost:3000/pagetwo

First name: Frank
Last name: Brown

Page 2

Address

State objects: pageone.js

```
let page1 = (  
  <>  
    <h3>Page 1</h3>  
    <div>  
      Firstname  
      <input  
        type="text"  
        placeholder="First name"  
        name="firstname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value })} />  
    </div>  
    <div>  
      Lastname  
      <input  
        type="text"  
        placeholder="Last name"  
        name="lastname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value,})} />  
    </div>  
    <button onClick={handleOnSubmit}>Next</button>  
  </>  
);  
return page1;  
}
```

name

React App

localhost:3000

Page 1

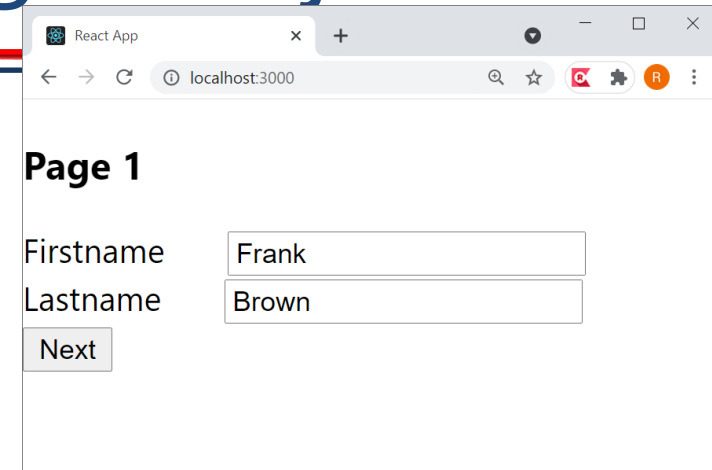
Firstname

Lastname

Update user object, leave all fields as is but update the field firstname with the value "Frank"

State objects: pageone.js

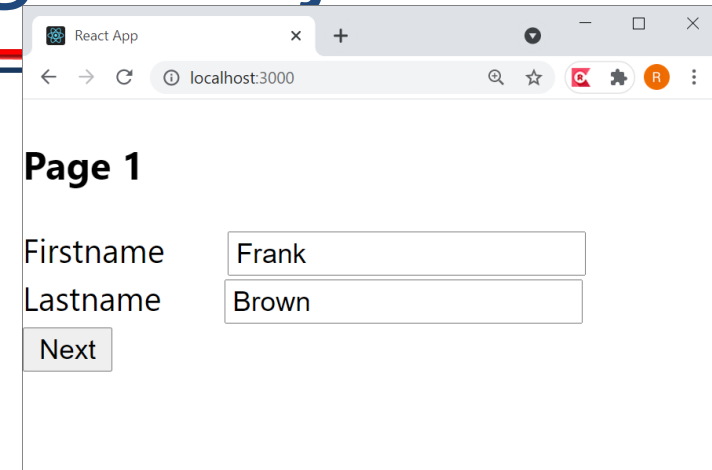
```
let page1 = (  
  <>  
    <h3>Page 1</h3>  
    <div>  
      Firstname  
      <input  
        type="text"  
        placeholder="First name"  
        name="firstname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value })} />  
    </div>  
    <div>  
      Lastname  
      <input  
        type="text"  
        placeholder="Last name"  
        name="lastname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value,})} />  
    </div>  
    <button onClick={handleOnSubmit}>Next</button>  
  </>  
>);  
return page1;  
>
```



Same method

State objects: pageone.js

```
let page1 = (  
  <>  
    <h3>Page 1</h3>  
    <div>  
      Firstname  
      <input  
        type="text"  
        placeholder="First name"  
        name="firstname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value })} />  
    </div>  
    <div>  
      Lastname  
      <input  
        type="text"  
        placeholder="Last name"  
        name="lastname"  
        onChange={e => setUser({...user,[e.target.name]: e.target.value,})} />  
    </div>  
    <button onClick={handleOnSubmit}>Next</button>  
  </>  
);  
return page1;  
}
```



React App

localhost:3000

Page 1

Firstname

Lastname

Same method

State objects: pageone.js

```
const [user, setUser] = useState({
  firstname: "",
  lastname: "",
  address: "",
  creditcard: ""
})
...
const handleFieldChange = (e) => {
  setUser({...user,[e.target.name]: e.target.value });
}
```

Define function

```
let page1 = (
  ...
  <div>
    Firstname
    <input
      ...
      name="firstname"
      onChange={handleFieldChange} />
    </div>
  <div>
    Lastname
    <input
      ...
      name="lastname"
      onChange={handleFieldChange} />
    </div>
  ...
)
```

Call function

Call function

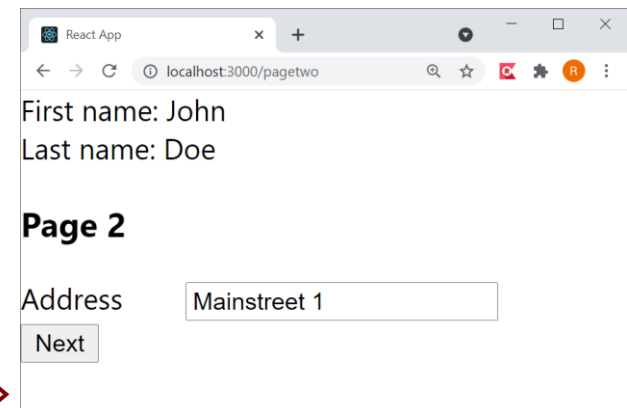
State objects: pagetwo.js

```
const location = useLocation();
const navigate = useNavigate();
const [user, setUser] = useState(location.state);

const handleOnSubmit = () => {
  navigate("/pagethree", { state: user });
}
const handleFieldChange = (e) => {
  setUser({ ...user, [e.target.name]: e.target.value });
}
```

Create a local user object based on the state of the passed user object

```
let page2 = (
  <div>
    <div>First name: {user.firstname}</div>
    <div>Last name: {user.lastname}</div>
    <>
      <h3>Page 2</h3>
      <div>
        Address
        <input
          type="text"
          placeholder="Address"
          name="address"
          onChange={handleFieldChange} />
      </div>
      <button onClick={handleOnSubmit}>Next</button>
    </>
  </div>
)
```



NEW ROUTER (REACT 6.4)

App.js

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import { Pageone, Pagetwo, Pagethree, Pagefour } from './pages';
import './App.css';

function App() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Pageone />} />
          <Route path="/pagetwo" element={<Pagetwo />} />
          <Route path="/pagethree" element={<Pagethree />} />
          <Route path="/pagefour" element={<Pagefour />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

App.js (React V6.4)

```
import { createBrowserRouter, RouterProvider } from 'react-router-dom';
import { Pageone, Pagetwo, Pagethree, Pagefour } from './pages';
import './App.css';

const router = createBrowserRouter([
  { path: "/", element: <Pageone /> },
  { path: "/pageone", element: <Pageone /> },
  { path: "/pagetwo", element: <Pagetwo /> },
  { path: "/pagethree", element: <Pagethree /> },
  { path: "/pagefour", element: <Pagefour /> }
]);

function App() {
  return (
    <div>
      <RouterProvider router={router} />
    </div>
  );
}

export default App;
```


WORKING WITH LISTS

List of strings

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

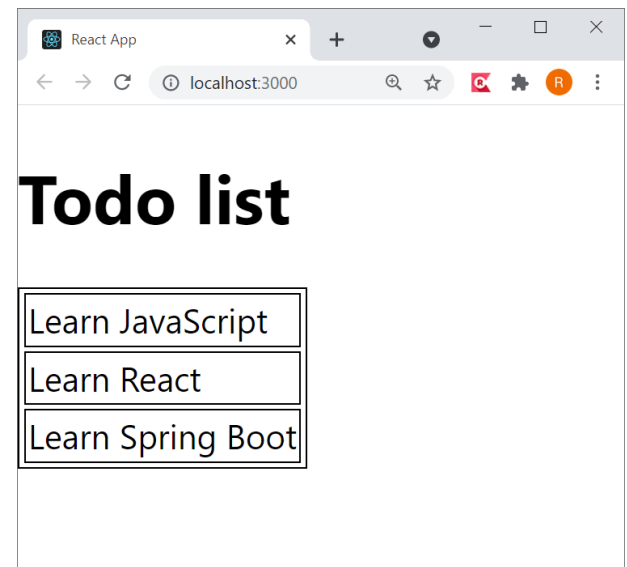
export const ToDoList = () => {

  const todoList = [
    'Learn JavaScript',
    'Learn React',
    'Learn Spring Boot',
  ];

  return (
    <div>
      <h1>Todo list</h1>
      <table>
        <tbody>
          {todoList.map(item => (
            <tr key={item}>
              <td>{item}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
```

Map expects a
unique key

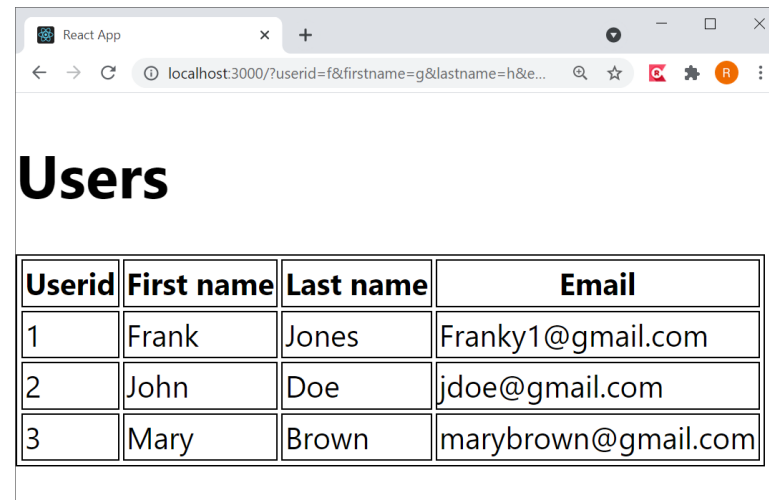
item is the
key



List of users

```
export const UsersList = () => {
  const initialList = [
    { userid: "1", firstname: "Frank", lastname: "Jones", email: "Franky1@gmail.com" },
    { userid: "2", firstname: "John", lastname: "Doe", email: "jdoe@gmail.com" },
    { userid: "3", firstname: "Mary", lastname: "Brown", email: "marybrown@gmail.com" }
  ];
  const [userlist, setUserlist] = useState(initialList);
  return (
    <div>
      <h1>Users</h1>
      <table>
        <thead>
          <tr><th>Userid</th><th>First name</th><th>Last name</th><th>Email</th></tr>
        </thead>
        <tbody>
          {userlist.map(user => (
            <tr key={user.userid}>
              <td>{user.userid}</td>
              <td>{user.firstname}</td>
              <td>{user.lastname}</td>
              <td>{user.email}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
```

Userid is
the key

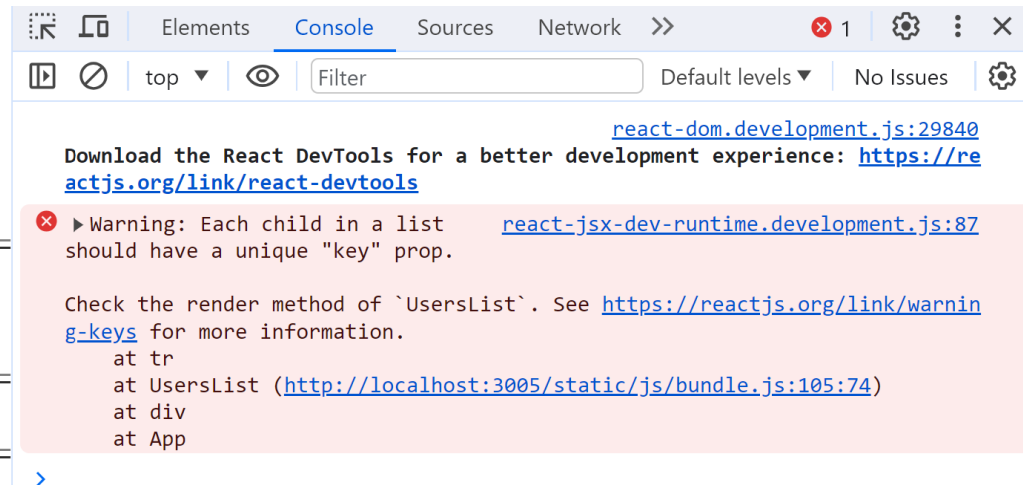


Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com
3	Mary	Brown	marybrown@gmail.com

If you don't use a key

Users

Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com
3	Mary	Brown	marybrown@gmail.com



Keys helps react to find which items in an array have changed or added or removed

MODIFY A LIST IN THE SAME PAGE

Add to a list of objects

React App

localhost:3000

Users

Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com

Add a new user

Userid

Firstname

Lastname

Email

Add User

UsersList.js(1/4)

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

```
export const UsersList = () => {
```

```
  const cleanuser = { userid: "", firstname: "", lastname: "", email: "" };
```

```
  const [user, setUser] = useState(cleanuser);
```

```
  const initialList = [
```

```
    { userid: "1", firstname: "Frank", lastname: "Jones", email: "Franky1@gmail.com" },
```

```
    { userid: "2", firstname: "John", lastname: "Doe", email: "jdoe@gmail.com" },
```

```
  ];
```

```
  const [userlist, setUserlist] = useState(initialList);
```

```
  const handleSubmit = (e) => {
```

```
    if (user) {
```

```
      setUserlist(userlist.concat(user)); //add user to the list
```

```
    }
```

```
    //clear user
```

```
    setUser(cleanuser);
```

```
    //prevent POST request
```

```
    e.preventDefault();
```

```
  }
```

```
  const handleFieldChange = (e) => {
```

```
    setUser({ ...user, [e.target.name]: e.target.value });
```

```
  }
```

One user

List of users

Called when you click the "Add user" button

Do not send a
POST request

Called when you
change one of the
form fields

Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com

Add a new user

Userid

Firstname

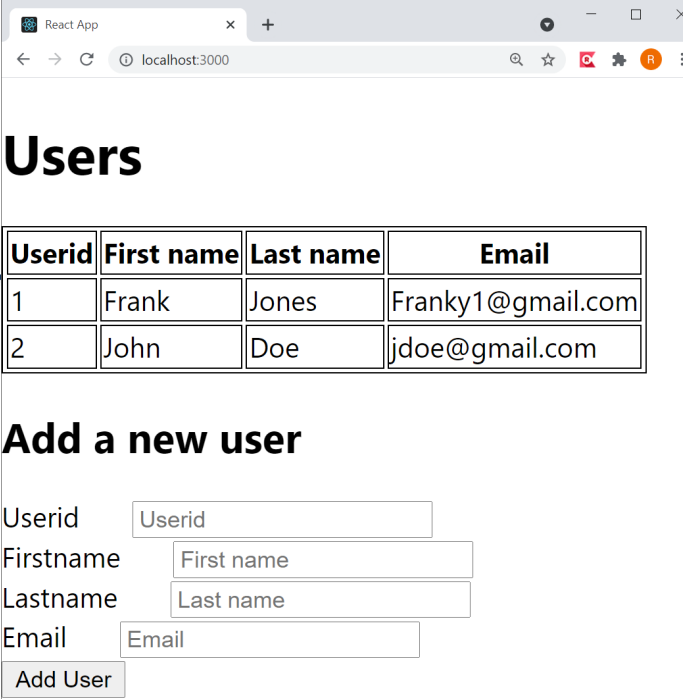
Lastname

Email

UsersList.js(2/4)

```
return (  
  <div>  
    <h1>Users</h1>  
  
    <table>  
      <tr><th>Userid</th><th>First name</th><th>Last name</th><th>Email</th></tr>  
      {userlist.map(user => (  
        <tr key={user.userid}>  
          <td>{user.userid}</td>  
          <td>{user.firstname}</td>  
          <td>{user.lastname}</td>  
          <td>{user.email}</td></tr>  
      )})  
    </table>  
  </div>  
)
```

Show table of users



Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com

Add a new user

Userid

Firstname

Lastname

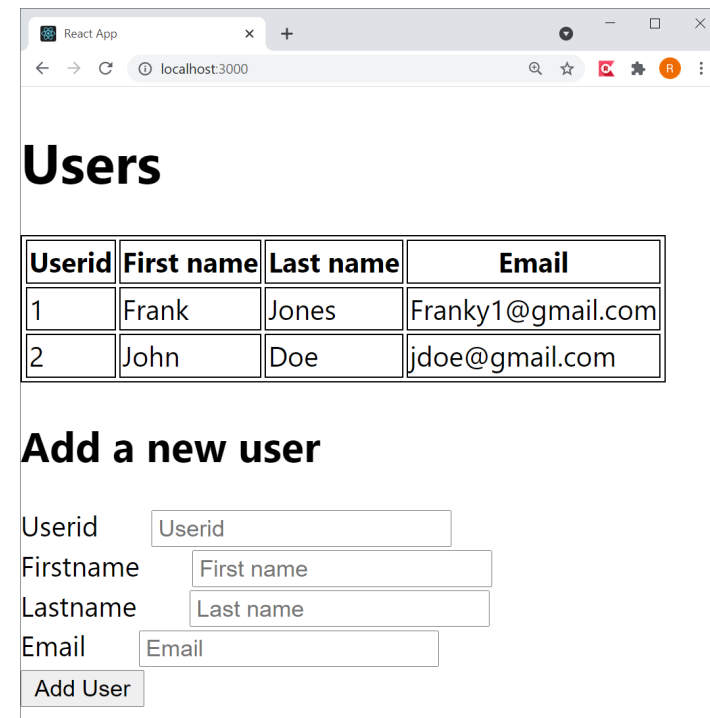
Email

UsersList.js(3/4)

```
<p><h2>Add a new user</h2></p>
<form onSubmit={handleSubmit}>
  <div>
    Userid
    <input
      type="text"
      placeholder="Userid"
      name="userid"
      value={user.userid}
      onChange={handleFieldChange} />
  </div>
  <div>
    Firstname
    <input
      type="text"
      placeholder="First name"
      name="firstname"
      value={user.firstname}
      onChange={handleFieldChange} />
  </div>
```

<form>

Call handleSubmit when you click the "Add user" button



Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com

Add a new user

Userid

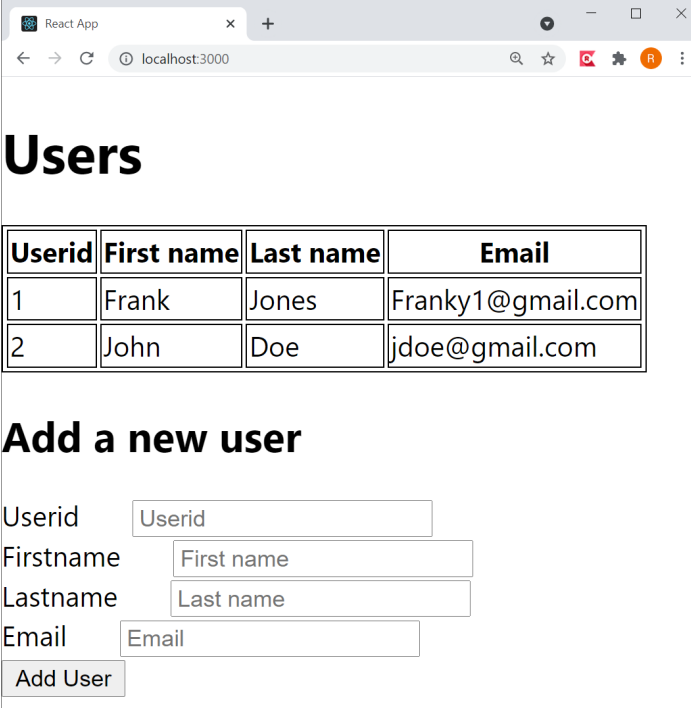
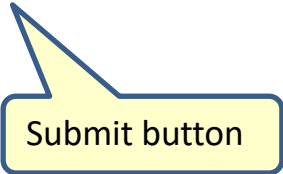
Firstname

Lastname

Email

UsersList.js(4/4)

```
<div>
  Lastname
  <input
    type="text"
    placeholder="Last name"
    name="lastname"
    value={user.lastname}
    onChange={handleFieldChange} />
</div>
<div>
  Email
  <input
    type="text"
    placeholder="Email"
    name="email"
    value={user.email}
    onChange={handleFieldChange} />
</div>
<button type="submit">Add User</button>
</form>
</div>
);
};
```



Userid	First name	Last name	Email
1	Frank	Jones	Franky1@gmail.com
2	John	Doe	jdoe@gmail.com

Add a new user

Userid

Firstname

Lastname

Email

Remove from a list of objects

React App

localhost:3000

Users

Userid	First name	Last name	Email	
1	Frank	Jones	Franky1@gmail.com	Remove
2	John	Doe	jdoe@gmail.com	Remove

Add a new user

Userid

Userid

Firstname

First name

Lastname

Last name

Email

Email

Add User

UsersList.js

```
const removeUser = (e) => {  
  const newUserlist = userlist.filter((user) => user.userid !== e.target.value);  
  setUserlist(newUserlist);  
}
```

Create a new list and filter out the user to be removed

```
<table>  
  <tr><th>Userid</th><th>First name</th><th>Last name</th><th>Email</th></tr>  
  {userlist.map(user => (  
    <tr key={user.userid}>  
      <td>{user.userid}</td>  
      <td>{user.firstname}</td>  
      <td>{user.lastname}</td>  
      <td>{user.email}</td>  
      <td><button onClick={removeUser} value={user.userid}>Remove</button></td>  
    </tr>  
  )  
  )  
</table>
```

Add button to the table

Call removeUser
function when
clicked

Pass userid as
value in event

Userid	First name	Last name	Email	
1	Frank	Jones	Franky1@gmail.com	Remove
2	John	Doe	jdoe@gmail.com	Remove

**MODIFY THE LIST IN SEPARATE
PAGES**

Add to a list of objects



React App

localhost:3000

Users

Userid	First name	Last name	Email	
22	John	Johnson	jjohnson@gmail.com	<button>Remove</button>
44	Mary	Jones	mjones@hotmail.com	<button>Remove</button>

Add User

React App

localhost:3000/adduser

Add a new user

Userid

Firstname

Lastname

Email

Add User

App.js(1/2)

```
import './App.css';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import { UsersList } from './UsersList';
import { AddUser } from './AddUser';
import React, { useState } from 'react';

function App() {

  const initialList = [
    { userid: "1", firstname: "Frank", lastname: "Jones", email: "Franky1@gmail.com" },
    { userid: "2", firstname: "John", lastname: "Doe", email: "jdoe@gmail.com" },
    { userid: "3", firstname: "Mary", lastname: "Brown", email: "marybrown@gmail.com" },
  ];
  const [userlist, setUserlist] = useState(initialList);

  const onAddUser = (user) => {
    setUserlist(userlist.concat(user)) ;
  }

  const onRemoveUser = (userid) => {
    setUserlist(userlist.filter((user) => user.userid !== userid)) ;
  }
}
```

List of users is in the top component

Methods to modify the list of users in the top component

App.js

```
return (  
  <div >  
    <BrowserRouter>  
      <Routes>  
        <Route exact path="/" element={<UsersList userList={userlist}  
removeUserFunction={onRemoveUser}/>} />  
        <Route path="/adduser" element={<AddUser addUserFunction={onAddUser}/>} />  
      </Routes>  
    </BrowserRouter>  
  </div>  
);  
}  
  
export default App;
```

Pass **userlist** and
onRemoveUser function to
the child component

Pass **onAddUser** function to
the child component

React App

localhost:3000

Users

Userid	First name	Last name	Email	
22	John	Johnson	jjohnson@gmail.com	Remove
44	Mary	Jones	mjones@hotmail.com	Remove

Add User

React App

localhost:3000/adduser

Add a new user

Userid

Firstname

Lastname

Email

Add User

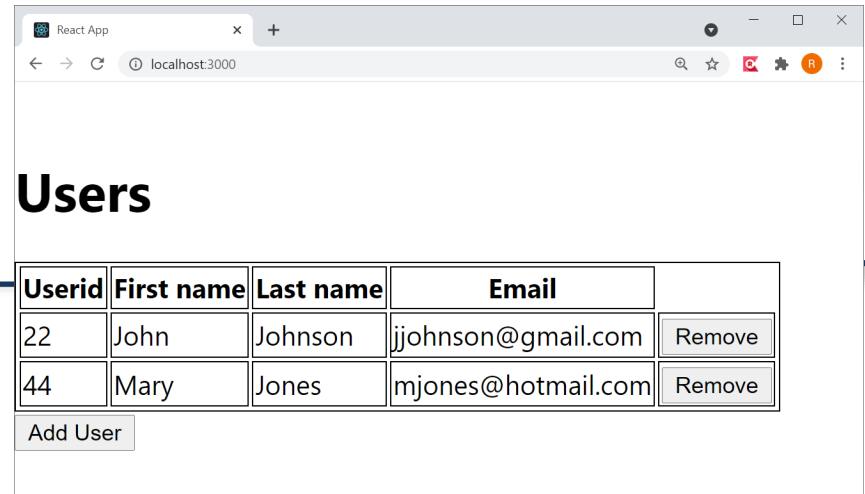
UsersList.js(1/2)

```
import React from 'react';
import { useNavigate } from 'react-router-dom';

export const UsersList = ({userlist, removeUserFunction}) => {
  const navigate = useNavigate();

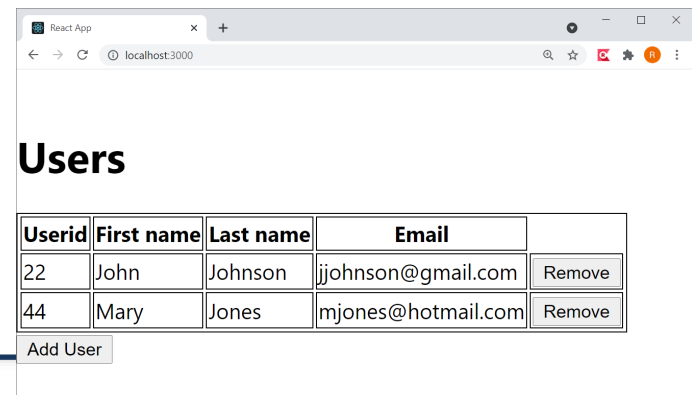
  const handleAddUser = () => {
    navigate('/adduser');
  }

  const removeUser = (e) => {
    removeUserFunction(e.target.value);
  }
}
```



UsersList.js(1/2)

```
return (  
  <div>  
    <h1>Users</h1>  
    <table>  
      <thead><tr><th>Userid</th><th>First name</th><th>Last name</th><th>Email</th>  
      </tr></thead>  
      <tbody>  
        {userlist.map(user => (  
          <tr key={user.userid}>  
            <td>{user.userid}</td>  
            <td>{user.firstname}</td>  
            <td>{user.lastname}</td>  
            <td>{user.email}</td>  
            <td><button onClick={removeUser} value={user.userid}>Remove</button></td>  
          </tr>  
        ))}  
      </tbody>  
    </table>  
    <button onClick={handleAddUser}>Add User</button>  
  </div>  
)  
);
```



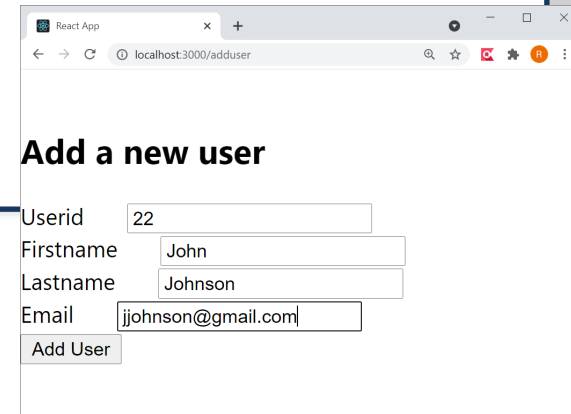
AddUser.js(1/2)

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

export const AddUser = ({addUserFunction}) => {
  const navigate = useNavigate();
  const cleanuser = { userid: "", firstname: "", lastname: "", email: "" };
  const [user, setUser] = useState(cleanuser);

  const handleSubmit = (e) => {
    e.preventDefault();
    addUserFunction(user);
    navigate('/');
  }

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



React App

localhost:3000/adduser

Add a new user

Userid

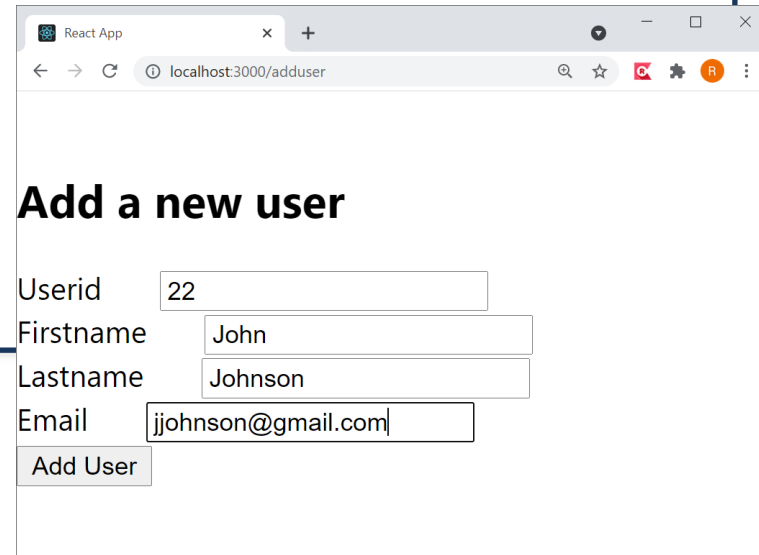
Firstname

Lastname

Email

AddUser.js(2/2)

```
return (  
  <div>  
    <h2>Add a new user</h2>  
    <form onSubmit={handleSubmit}>  
      <div>  
        Userid  
        <input  
          type="text"  
          placeholder="Userid"  
          name="userid"  
          value={user.userid}  
          onChange={handleFieldChange} />  
      </div>  
      ...  
      <button type="submit">Add User</button>  
    </form>  
  </div>  
)  
);
```



A screenshot of a web browser window titled 'React App' with the address bar showing 'localhost:3000/adduser'. The page displays a form titled 'Add a new user'. The form contains four input fields: 'Userid' with the value '22', 'Firstname' with the value 'John', 'Lastname' with the value 'Johnson', and 'Email' with the value 'jjohnson@gmail.com'. Below the input fields is a button labeled 'Add User'.

VALIDATION

Form validation

React App

localhost:3000

Enter your data

Firstname

First name is too short

Lastname

Last name is too short

Email

Email is too short

Email is not correct

Next

React App

localhost:3000

Enter your data

Firstname

First name is too long

Lastname

Email

Email is not correct

Next

Validation: pageone.js

```
export const Pageone = (props) => {  
  const cleanUser = {  
    firstname: "",  
    lastname: "",  
    email: ""  
  }  
  const [user, setUser] = useState(cleanUser);  
  
  const [firstnameError, setFirstnameError] = useState({});  
  const [lastnameError, setLastnameError] = useState({});  
  const [emailError, setEmailError] = useState({});  
  
  const handleOnSubmit = (e) => {  
    e.preventDefault();  
    const isValid = formValidation();  
    if (isValid) {  
      setUser(cleanUser);  
      alert("Form is valid");  
    }  
  }  
}
```

Do validation
after submit

Validation: pageone.js

```
const formValidation = () => {
  const firstNameErr = {};
  const lastNameErr = {};
  const emailErr = {};
  let isValid = true;

  if (user.firstname.trim().length < 2) {
    firstNameErr.firstNameShort = "First name is too short"
    isValid = false;
  }
  if (user.firstname.trim().length > 10) {
    firstNameErr.firstNameShort = "First name is too long"
    isValid = false;
  }
  if (user.lastname.trim().length < 2) {
    lastNameErr.lastNameShort = "Last name is too short"
    isValid = false;
  }
  if (user.email.trim().length < 5) {
    emailErr.emailShort = "Email is too short"
    isValid = false;
  }
}
```

Check the form
fields

Validation: pageone.js

```
const formValidation = () => {
  const firstNameErr = {};
  const lastNameErr = {};
  const emailErr = {};
  let isValid = true;

  if (user.firstname.trim().length < 2) {
    firstNameErr.firstNameShort = "First name is too short"
    isValid = false;
  }
  if (user.firstname.trim().length > 10) {
    firstNameErr.firstNameShort = "First name is too long"
    isValid = false;
  }
  if (user.lastname.trim().length < 2) {
    lastNameErr.lastNameShort = "Last name is too short"
    isValid = false;
  }
  if (user.email.trim().length < 5) {
    emailErr.emailShort = "Email is too short"
    isValid = false;
  }
}
```

Check the form
fields

Validation: pageone.js

```
var pattern = new RegExp(/^(("[\\w-\\s]+"|([\\w-]+(?:\\.\\w-)+)*|("[\\w-\\s]+"([\\w-]+(?:\\.\\w-)+)*))(@((?:[\\w-]+\\.)*\\w[\\w-]{0,66})\\.([a-z]{2,6}(?:\\.([a-z]{2})?)?)$)|(@\\[?((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[0-9]{1,2})\\.)(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[0-9]{1,2})\\]?$)/i);

if (!pattern.test(user.email.trim())) {
    emailErr.emailNoEmail = "Email is not correct"
    isValid = false;
}

setFirstnameError(firstNameErr);
setLastnameError(lastNameErr);
setEmailError(emailErr);
return isValid;
}

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

Validation: pageone.js

```
let page1 = (  
  <div>  
    <form onSubmit={handleOnSubmit}>  
      <h3>Enter your data</h3>  
      <div>  
        Firstname  
        <input  
          type="text"  
          placeholder="First name"  
          name="firstname"  
          value={user.firstname}  
          onChange={handleFieldChange} />  
        {Object.keys(firstnameError).map((key) => {  
          return <div style={{ color: "red" }}>{firstnameError[key]}</div>  
        })}  
      </div>  
    </form>  
  </div>  
)
```

Validation: pageone.js

```
<div>
  Lastname
  <input
    type="text"
    placeholder="Last name"
    name="lastname"
    value={user.lastname}
    onChange={handleFieldChange} />

    {Object.keys(lastnameError).map((key) => {
      return <div style={{ color: "red" }}>{lastnameError[key]}</div>
    })}
</div>
```

Validation: pageone.js

```
<div>
  Email
  <input
    type="text"
    placeholder="Email"
    name="email"
    value={user.email}
    onChange={handleFieldChange} />
    {Object.keys(emailError).map((key) => {
      return <div style={{ color: "red" }}>{emailError[key]}</div>
    })}
  </div>
  <button type="submit">Next</button>
</form>
</div>

);
return page1;
}
```

REACT HOOK FORM

React form libraries

- React hook form
- Formik
- React final form

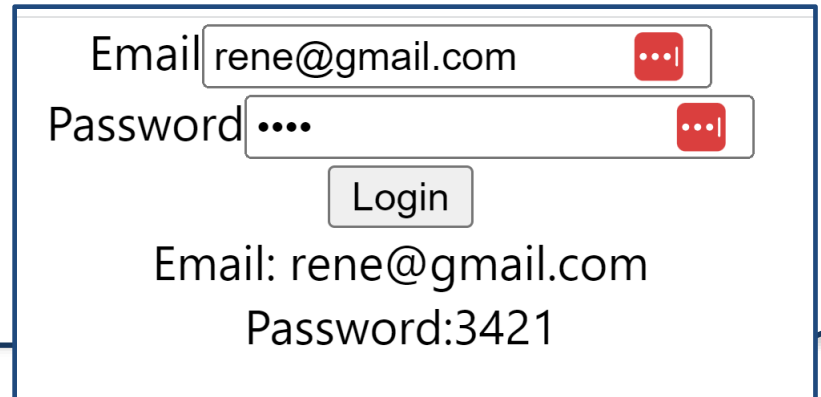
React hook form example

```
import React, { useState } from 'react';
import { useForm } from 'react-hook-form';

export const Entryform = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const {
    register,
    handleSubmit,
    formState: { errors }
  } = useForm();

  const onSubmit = (data) => {
    console.log(data);
    setEmail(data.email);
    setPassword(data.password);
  };
}
```



Email rene@gmail.com

Password

Login

Email: rene@gmail.com
Password:3421

React hook form example

```
let page2 = (  
  <div className="App">  
    <form onSubmit={handleSubmit(onSubmit)}>  
      <div className="form-control">  
        <label>Email</label>  
        <input type="text" name="email" {...register("email")} />  
      </div>  
      <div className="form-control">  
        <label>Password</label>  
        <input type="password" name="password" {...register("password")} />  
      </div>  
      <div className="form-control">  
        <label></label>  
        <button type="submit">Login</button>  
      </div>  
    </form>  
  
    <div>Email: {email}</div>  
    <div>Password: {password}</div>  
  </div>  
);  
return page2;  
}
```

Lets add validation

Email

f

...

Email is not valid.

Password

•

...

Password should be at-least 6 characters.

Login

React hook form example

```
let page2 = (  
  <div className="App">  
    <form onSubmit={handleSubmit(onSubmit)}>  
      <div className="form-control">  
        <label>Email</label>  
        <input  
          type="text"  
          name="email"  
          {...register("email", {  
            required: "Email is required.",  
            pattern: {  
              value: /^[^@ ]+@[^@ ]+\.[^@ .]{2,}$/,  
              message: "Email is not valid."  
            }  
          }  
        )}>  
      </div>  
      {errors.email && <p className="errorMsg">{errors.email.message}</p>}  
    </div>  
  )
```

Validation constraints

Validation error

React hook form example

```
<div className="form-control">
  <label>Password</label>
  <input
    type="password"
    name="password"
    {...register("password", {
      required: "Password is required.",
      minLength: {
        value: 6,
        message: "Password should be at-least 6 characters."
      }
    })}
  />
  {errors.password && (
    <p className="errorMsg">{errors.password.message}</p>
  )}
</div>
<div className="form-control">
  <label></label>
  <button type="submit">Login</button>
</div>
</form>
</div>
);
return page2;
```

Lets add validation

Email f

Email is not valid.

Password •

Password should be at-least 6 characters.

Login

First field with an error gets the focus when you click the submit button

Form is not submitted when there are errors

Form is responsive: error disappears when entered data is valid