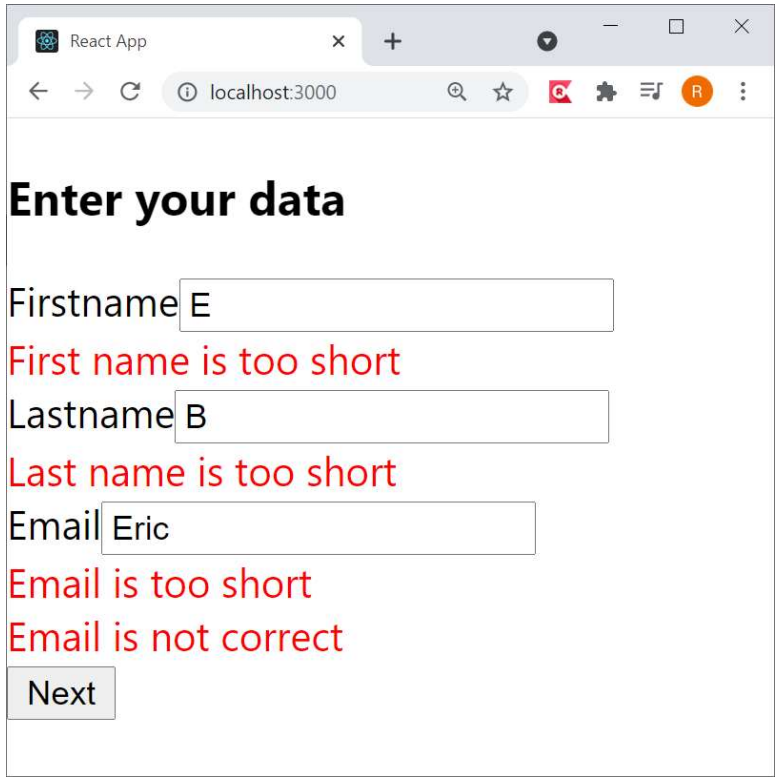
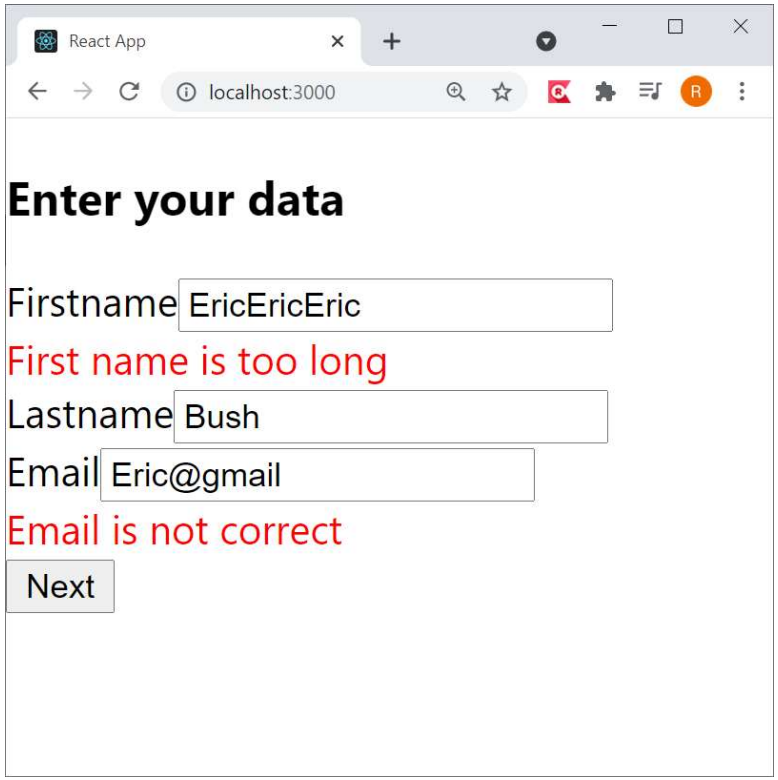
Lesson 12

VALIDATION
BACKEND ACCESS

VALIDATION

Form validation





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

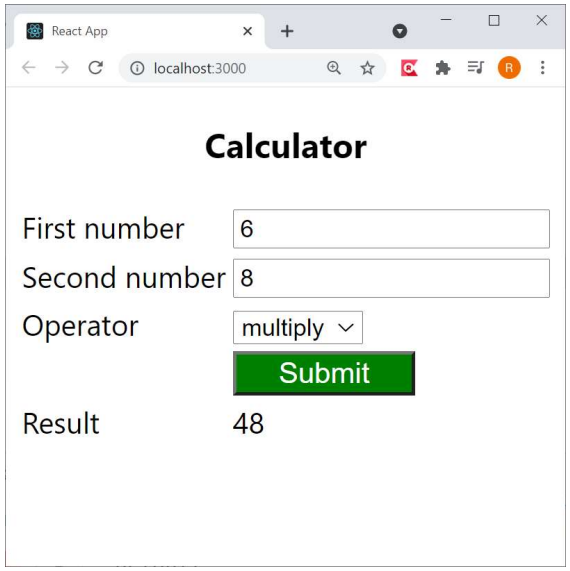
Validation: pageone.js

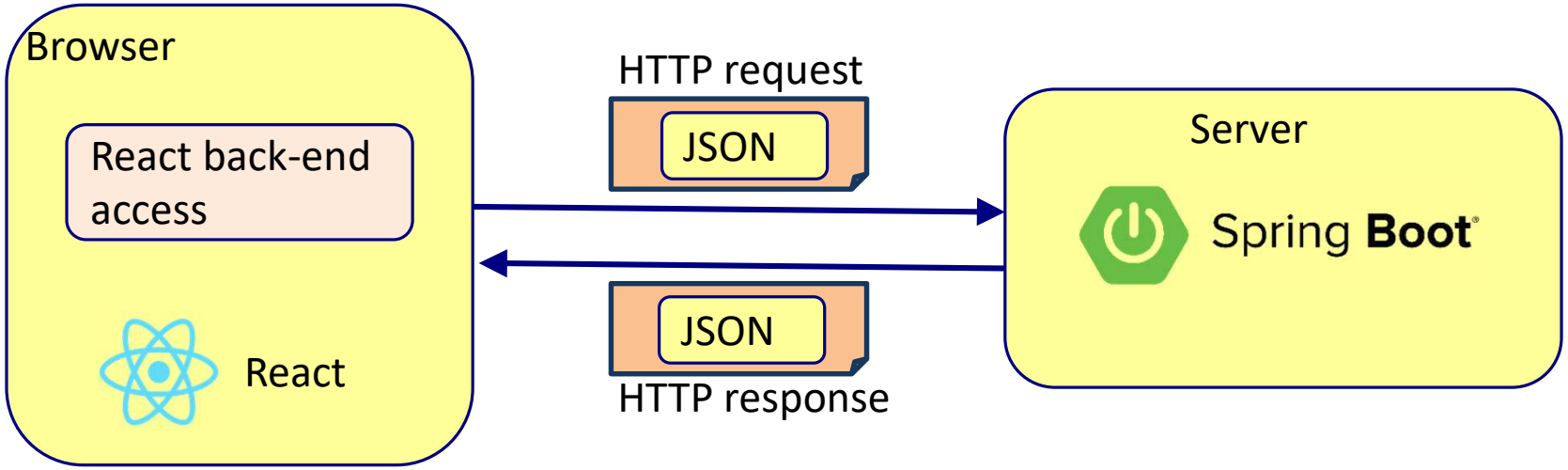
```
const formValidation = () => {
  const firstNameErr = {};
  const lastNameErr = {};
  const emailErr = {};
  let isValid = true;

  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

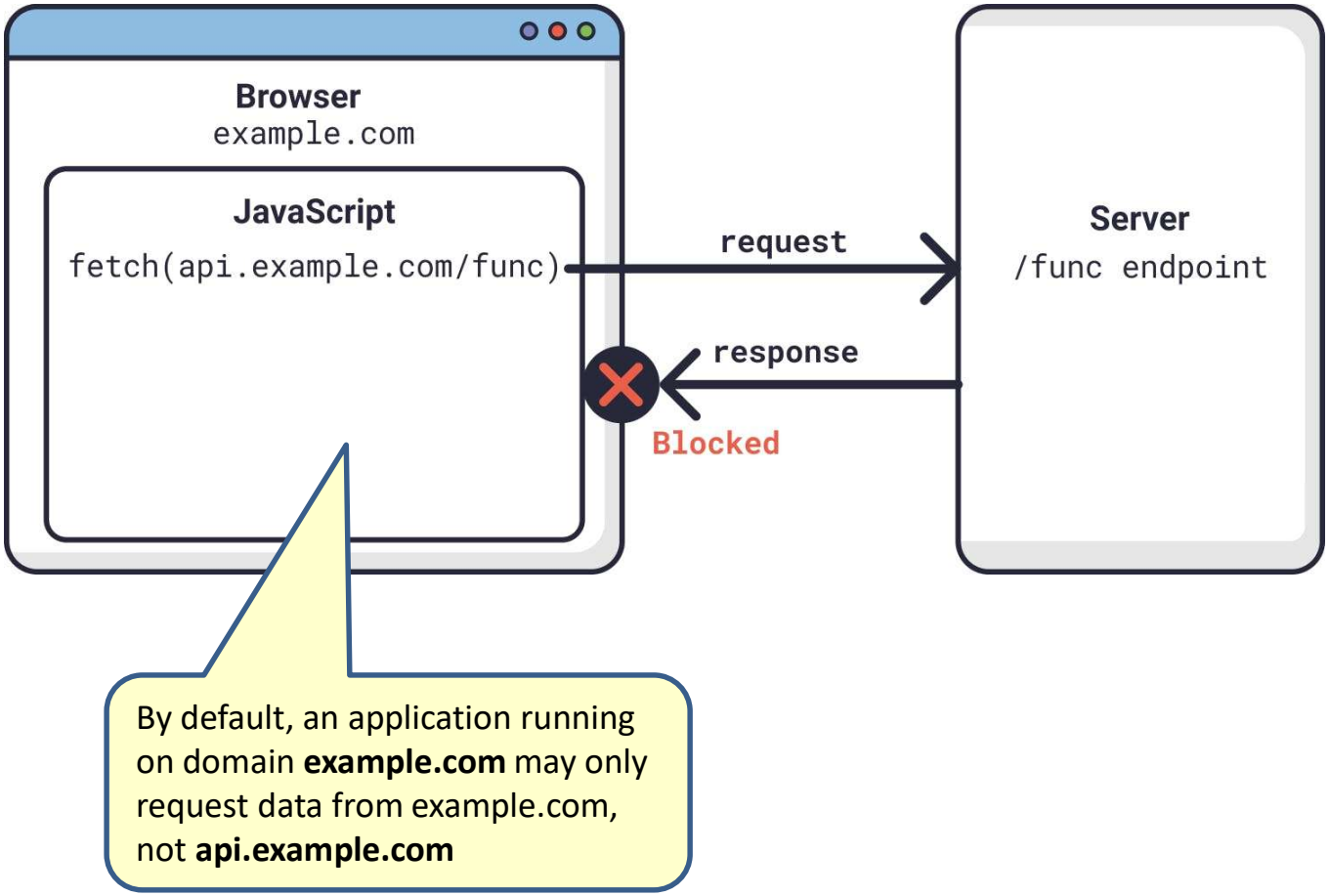
BACKEND ACCESS

Calculator


React App
localhost:3000
Calculator
First number
6
Second number
8
Operator
multiply
Submit
Result
48

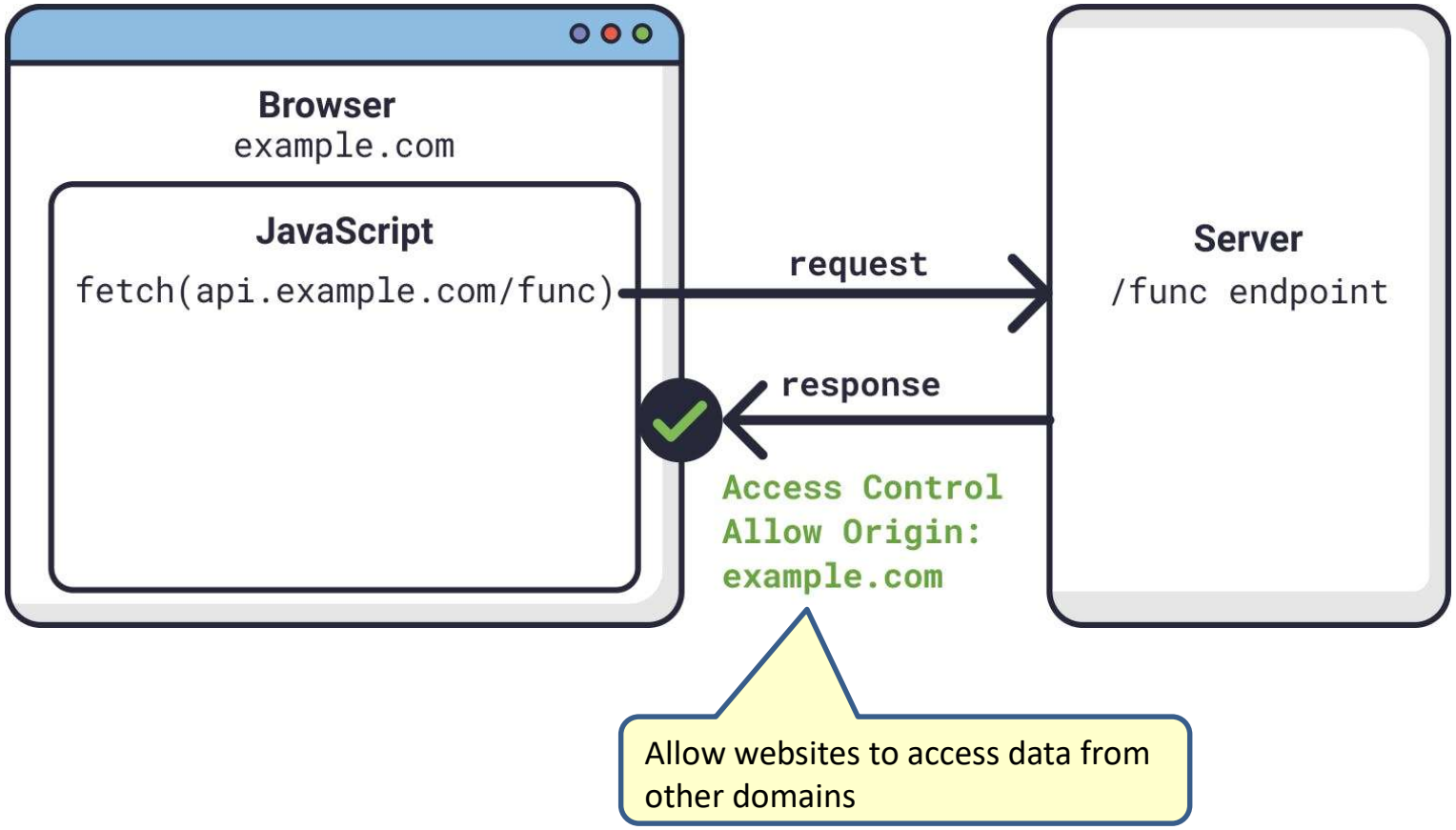

Browser
React back-end access
React
HTTP request
JSON
JSON
HTTP response
Server
Spring Boot

Same origin policy



CORS

- Cross-Origin Resource Sharing



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

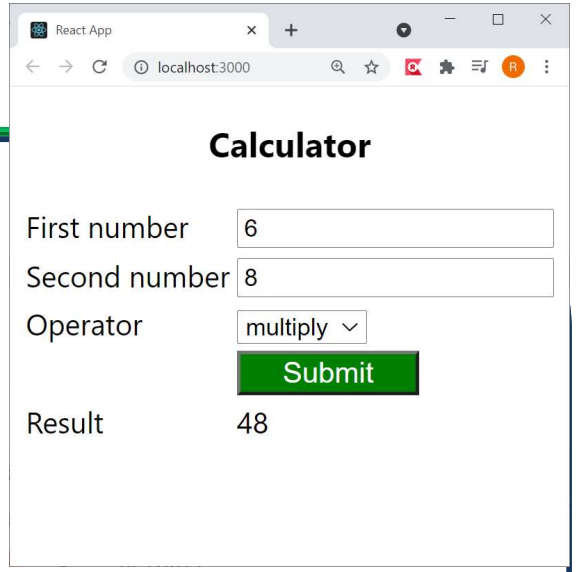
Calculator.js

```
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

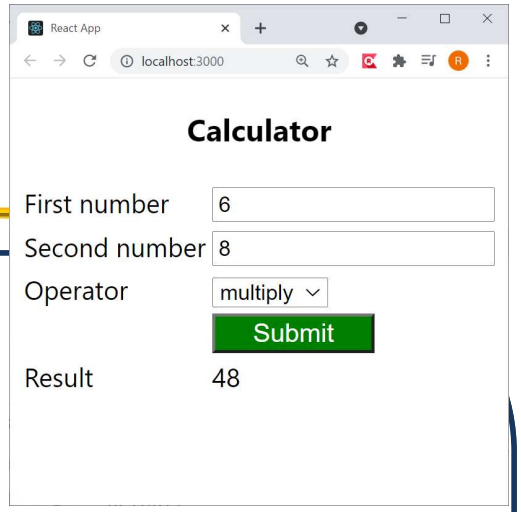
Set the result

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.js




```
                <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;
```

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

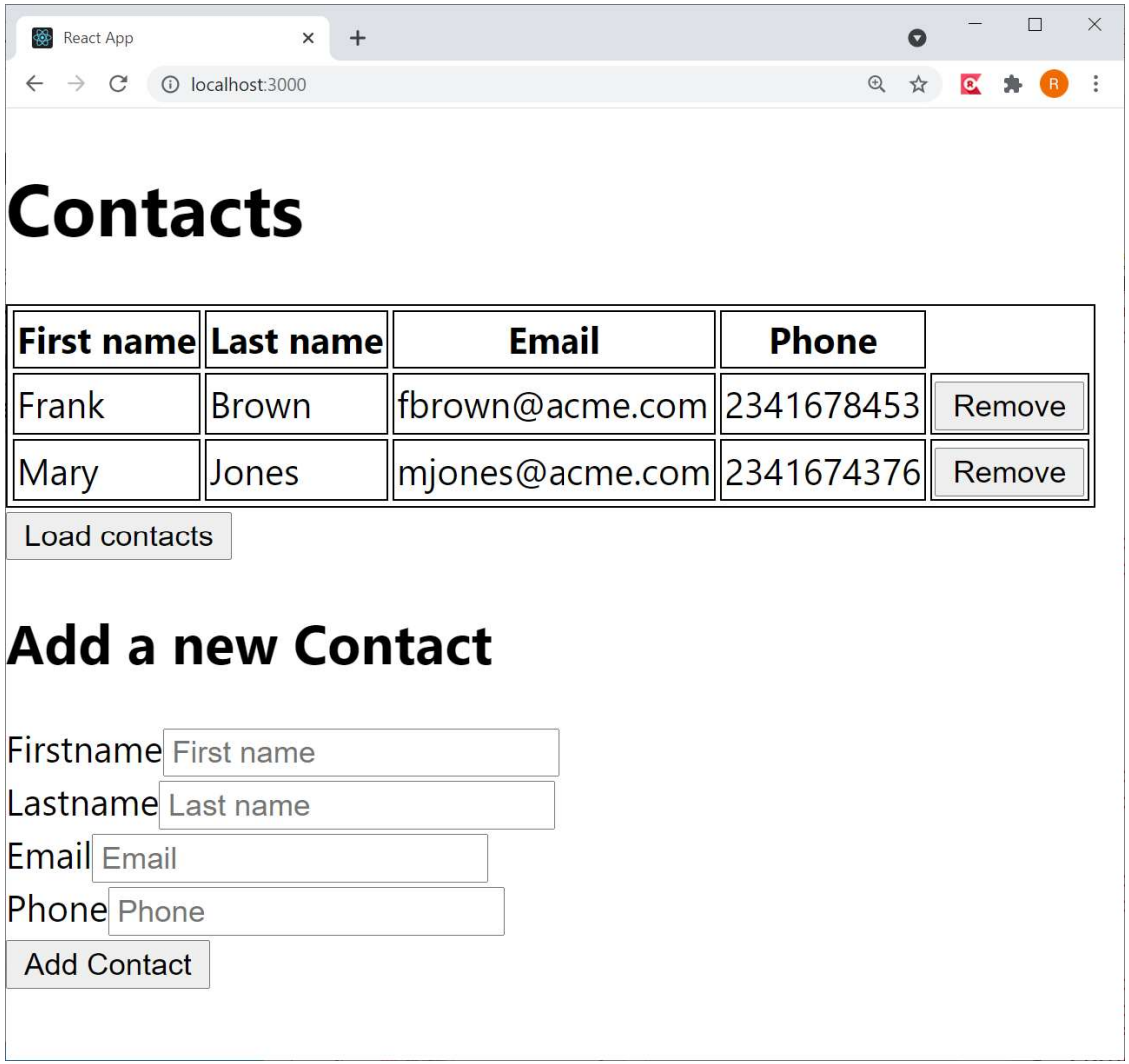
```
const fetchBackend = e => {
      const url = 'http://localhost:8080/calc/'+first+'/'+second+'/'+operator;
      const response = axios.get(url)
          .then((response) => {
              setResult(response.data.value);
          });
      e.preventDefault();
  }
```

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 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
  }
```

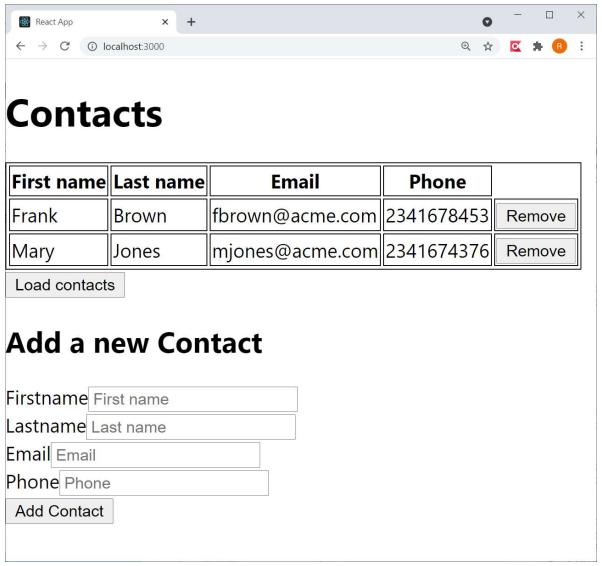
get

post

Reload contacts

delete

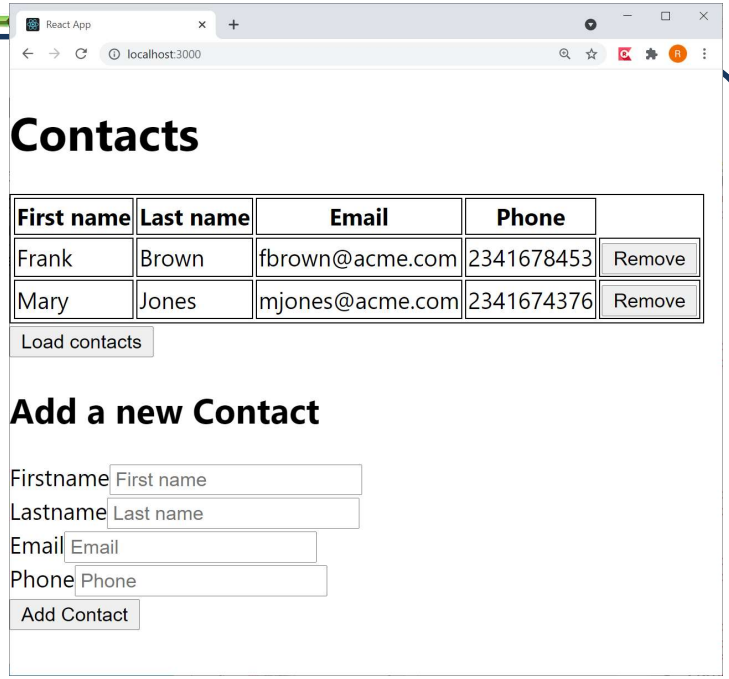
Reload contacts



Contacts front-end

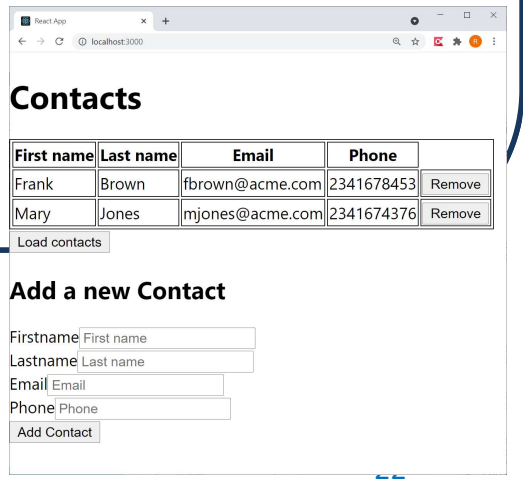
Called when we click the **Add Contact** button

```
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 });
}
```



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 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>
```