wawiwa

Exercises

Exercise 1

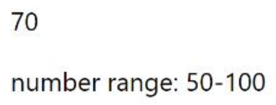
In App.js create a function called NumberRangeChecker. The function should accept a number.

The function will return the number range based on this conditions:

- Under 50
- 50-100
- Over 100

Inside the component's JSX, use embedded expressions to show the number and the determined range of the number.

For example:



```
70

number range: 50-100
```

Exercise 2

Create a functional component that will be responsible for a passenger ticket.

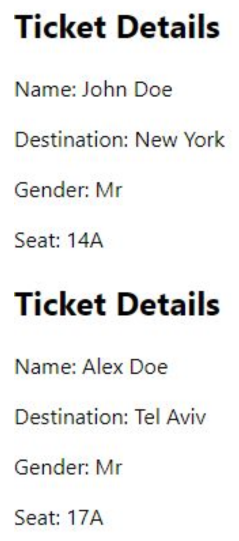
The components gets those parameters:

- Name
- Destination
- Gender (Mr/Mrs)
- Seat

The component will return a div with this details.

Render this component twice with different information.

For example:



Ticket Details

Name: John Doe

Destination: New York

Gender: Mr

Seat: 14A

Ticket Details

Name: Alex Doe

Destination: Tel Aviv

Gender: Mr

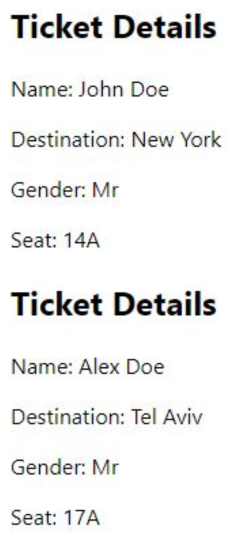
Seat: 17A

Exercise 3

Move the functional component from ex.2 into a new file.

Render the tickets again on the screen.

For example:



Ticket Details

Name: John Doe

Destination: New York

Gender: Mr

Seat: 14A

Ticket Details

Name: Alex Doe

Destination: Tel Aviv

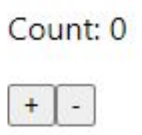
Gender: Mr

Seat: 17A

Exercise 4

Create a Counter component that displays a count value and allows the user to increment or decrement the count using buttons.

1. Create a new React component called Counter.
2. Inside the Counter component, import the useState hook.
3. Declare a state variable called count sing the useState hook. Initialize it to 0.
4. Render the current value of count inside a <p> element.
5. Render two buttons: one for incrementing the count and another for decrementing the count.



wawiwa

Exercise 5

1. Assign an onClick event handler to the increment button that increases the count value by 1.
2. Assign an onClick event handler to the decrement button that decreases the count value by 1.

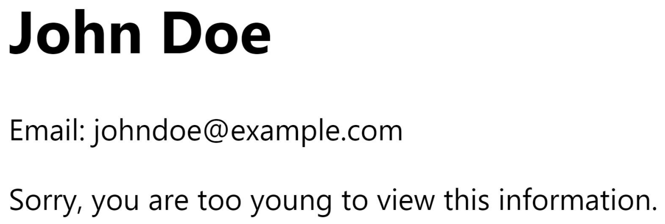
Exercise 6

Display a user's name, email, and age.

Add a condition to the component so that it only displays the user's age if their age is greater than 18. If the user's age is less than 18, display a message saying "Sorry, you are too young to view this information."

Example of user object:

```
const user = {
  name: 'John Doe',
  email: 'johndoe@example.com',
  age: 14,
};
```



John Doe

Email: johndoe@example.com

Sorry, you are too young to view this information.

wawiwa

Exercise 8

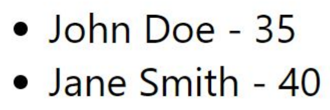
Create a new component called Menu.

Inside this component, create an array called users that contains multiple user objects. Each object should have a properties for name and age.

Example: [{ name: 'John Doe', age: 35 }, { name: 'Jane Smith', age: 40 }]

Use the “map” method to render the list.

*Assign a unique key to each list item using the index parameter.



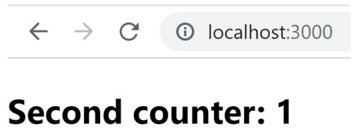
- John Doe - 35
- Jane Smith - 40

wawiwa

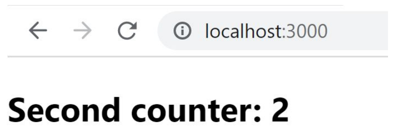
Exercise 9

Create a seconds counter.

- The “seconds” state initialized to 0
- Every one second (1000 milliseconds) the “seconds” state increments by 1
- Use useState and UseEffect hooks
- Use setInterval
- Use the clearInterval function to clean up the interval when the component is unmounted.



Second counter: 1



Second counter: 2

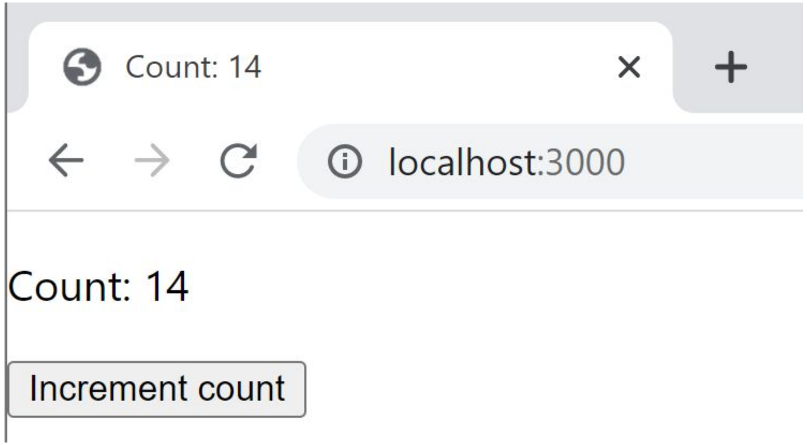
wawiwa

Exercise 10

Create an app that renders a counter and a button to increment it.

Update the document title with the current count.

The count state initialized to 0.



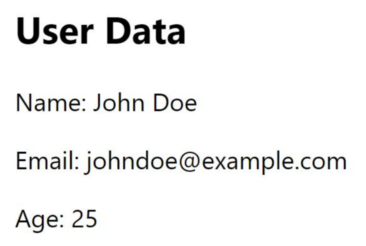
Exercise 11

Create a new React context using the createContext function. This will be the User Context.

In the User Context, define an initial user object with the properties name, email, and age.

Create a UserProvider component that wraps a UserDisplay component.

Inside it show user's data.



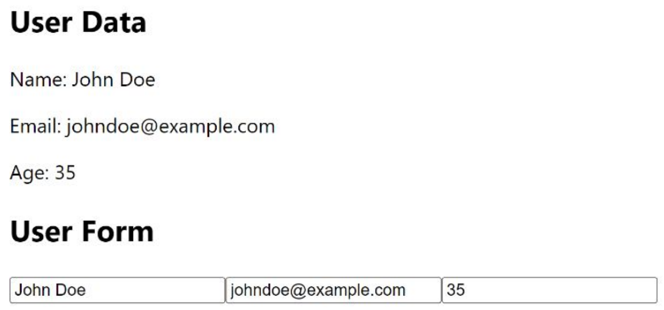
```
User Data

Name: John Doe

Email: johndoe@example.com

Age: 25
```

wawiwa

Exercise 12

Create a new component called EditUser.

The component allows the user to update their information with input fields for name, email, and age.



User Data

Name: John Doe

Email: johndoe@example.com

Age: 35

User Form

John Doe	johndoe@example.com	35

wawiwa

Exercise 13

Create a calculator app:

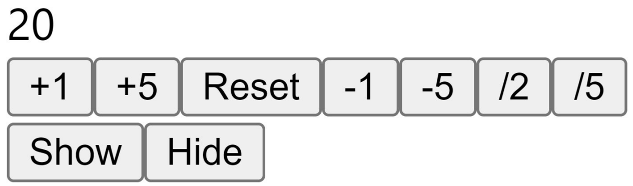
- The app has the next calculation buttons:

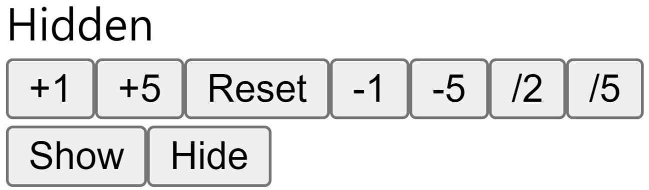
  +1, +5, -1, -5, /2, /5, Reset(0)

- The app has “Show” and “Hide” buttons for the counter label

Introduction:

- The beginning state contains:
  a. Counter (0)
  b. Show (true)
- There are 4 types of actions



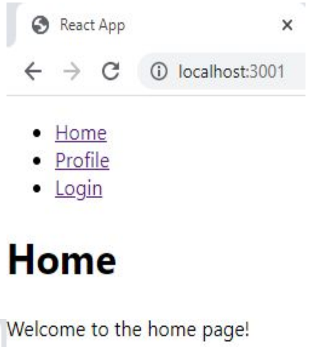


wawiwa

Exercise 16

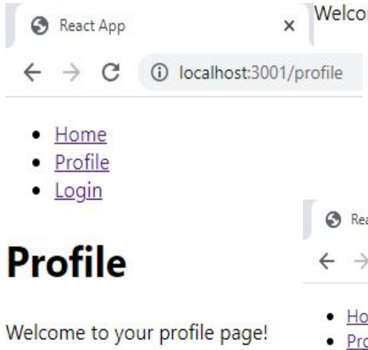
In this exercise, you'll build a simple website using React Router.

1. Create three components: Home, Profile, and Login. Each component should render some simple content, such as a heading and a paragraph.
2. Create a Navbar component that displays links to the Home, Login and Profile components.
3. Use React Router to create routes for each of the components. The Home component should be the default route.
4. The Login component will contain inputs of username and password.



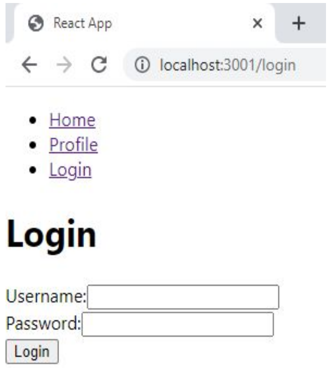
Home

Welcome to the home page!



Profile

Welcome to your profile page!



Login

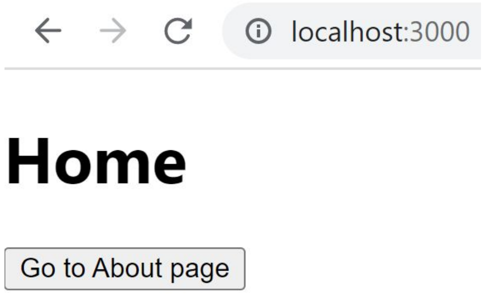
Username:
Password:

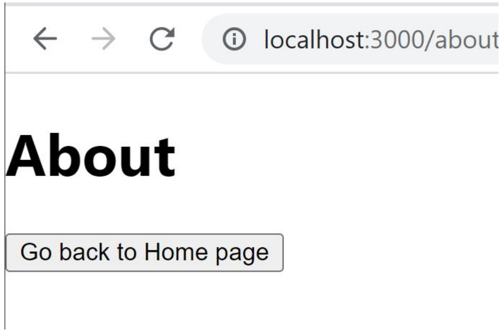
wawiwa

Exercise 17

Create a simple application with two pages: a home page and an about page.

The home page should display a button that navigates to the about page when clicked. The about page should display a button that navigates back to the home page when clicked.

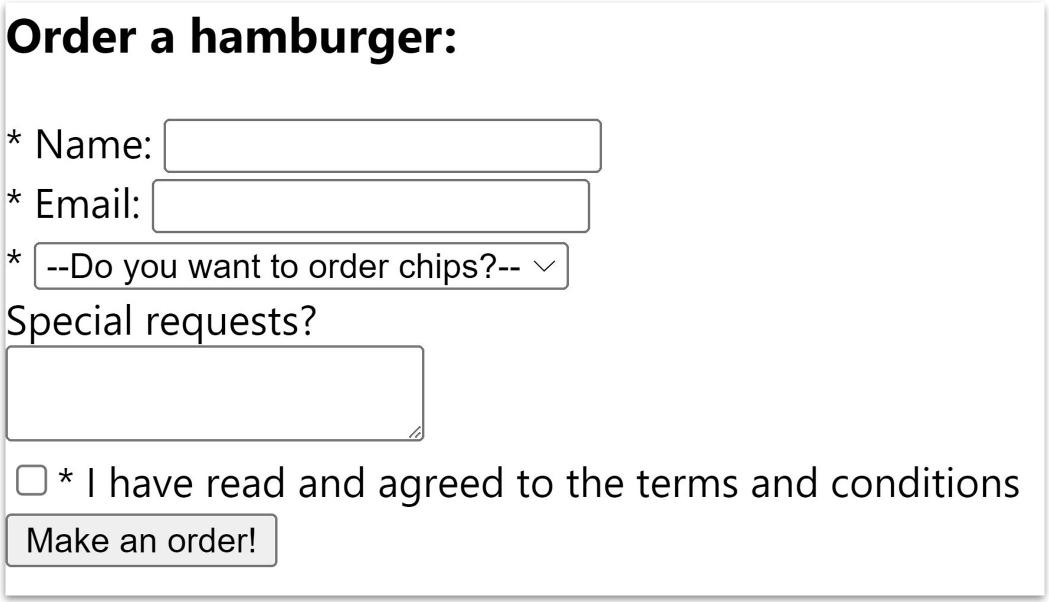


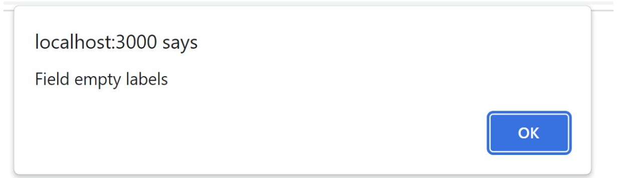


wawiwa

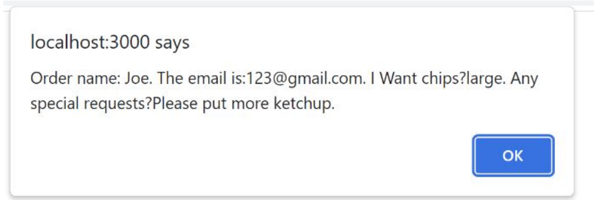
Exercise 18

Create the following form:


Order a hamburger:
* Name:
* Email:
* --Do you want to order chips?--
Special requests?
* I have read and agreed to the terms and conditions
Make an order!


localhost:3000 says

Field empty labels


localhost:3000 says

Order name: Joe. The email is:123@gmail.com. I Want chips?large. Any special requests?Please put more ketchup.

- Clicking on the submit button, check if there are empty labels that must be field.
- If there is no missed data, alert the order, else alert a message for the user.

wawiwa

Exercise 19

Copy the following code:

```
const products = [
  {
    name: 'Product 1',
    description: 'Description of product 1',
    price: '$10.99'
  },
  {
    name: 'Product 2',
    description: 'Description of product 2',
    price: '$24.99'
  },
  {
    name: 'Product 3',
    description: 'Description of product 3',
    price: '$15.49'
  },
];
```

```
return (
    <div>
      {products.map((product, index) => (
        <div key={index}>
          <div>{product.name}</div>
          <div>{product.description}</div>
          <div>{product.price}</div>
          </div>
      ))}
    </div>
  );
```

Exercise 19A

1. Style the product name to be bold with 20px font size.
2. Style the price to be a red color.

wawiwa

Exercise 19B

Create a style object variable “ProductContainer” that will style each product with:

```
display: 'flex',
justifyContent: 'space-between',
alignItems: 'center',
padding: '10px'
```

Exercise 19C

1. Convert “ProductContainer” into a styled component.
2. Create a styled component for the div that contains {product.name}, call it “ProductName”.
3.
    a. Create a styled component to the div that contains {product.price}, call it “ProductPrice”.
    b. On hover ProductPrice component color the font to green.
4. Move the styled components to to a separate file. Export the components there and then import them.

wawiwa

MATERIAL-UI

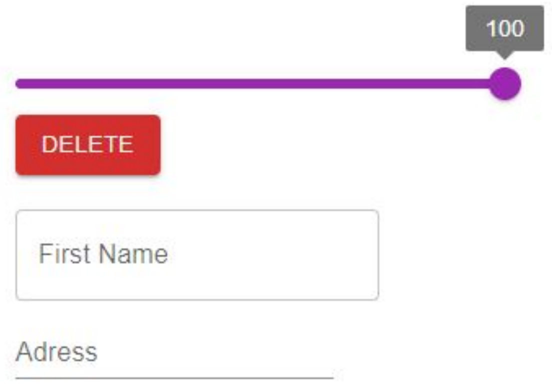
Exercise 21

Create the next components:

- Slider
- Delete button
- 2 TextFields

Explore the components:

- Progress
- Skeleton
- Snackbar
- Accordion
- App Bar
- Drawer


100
DELETE
First Name
Adress