Thinking in React Exercises

Baby steps with React

Writing Components

Understanding how to write your own components and then rendering on the DOM.

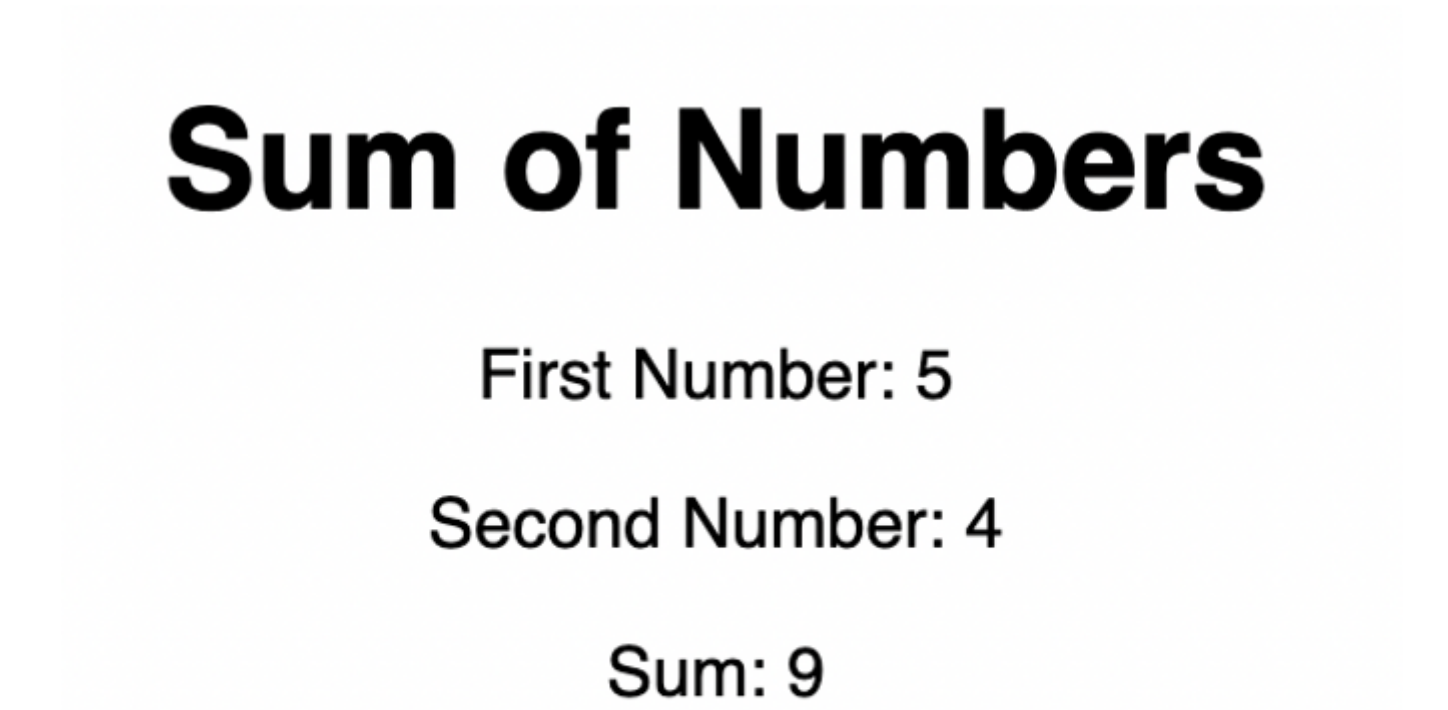
```
Components let you split the UI into independent, reusable pieces,
and think about each piece in isolation.

Always start component names with a capital letter. React treats
components starting with lowercase letters as DOM tags. For example,
<div /> represents an HTML div tag, but <Sum /> represents a component.
```

COPY

Live Example:

1. Write a React component to display two numbers and the sum of those two numbers.



Solution:

```
// App.js
import Sum from "./Sum";
export default function App() {
  return (
    <div className="App">
      <Sum />
    </div>
  );
}

// Sum.js
export default function Sum() {
  return (
    <div>
      <h1>Sum of Numbers</h1>
      <p>First Number: 5</p>
      <p>Second Number: 4</p>
      <p>Sum: 9</p>
    </div>
  );
}
```

COPY

Challenge for Students:

1. Create an EmployeeCard component to display name, designation and work experience. Import the file in App.js file and render data on DOM.

Solution

```
// App.js
import EmployeeCard from "./EmployeeCard";

export default function App() {
  return (
    <div className="App">
      <EmployeeCard />
    </div>
  );
}
```

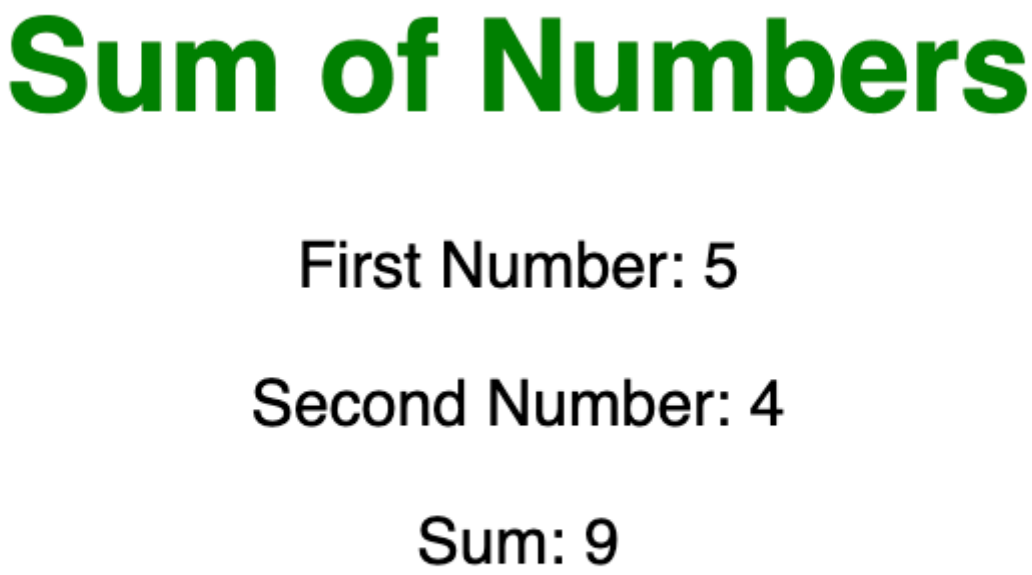
COPY

Adding Inline Style:

The syntax for adding styles in React is slightly different from basic HTML/CSS.

Live Example:

1. Change the colour of heading ‘Sum of Numbers’ to green.



Solution

```
export default function Sum() {
  return (
    <div>
      <h1 style={{ color: "green" }}>Sum of Numbers</h1>
      <p>First Number: 5</p>
      <p>Second Number: 4</p>
      <p>Sum: 9</p>
    </div>
  );
}
```

COPY

Challenge for Students:

1. Change the fontSize of “First Number”, “Second Number” and “Sum” to 20px.
2. In EmployeeCard,
    1. change the color of “Designation:” to red.
    2. change the color of “Experience:” to blue.
    3. make the font size of “10 years” to 18px

`hint: use span tag to change colors in middle of a paragraph`

return - The return method in React is a way to return data from a component.

Homework:

Learn to write `className` in React.

JSX expression and props

Understanding how to write JSX and pass data as props.

```
JSX is a syntax extension to JavaScript.
It is helpful as a visual aid when working with UI inside the JavaScript code.

Props are a way of passing data from parent to child component.
```

COPY

Single Data

Live Example:

1. Write a React component to display two numbers and the sum of those two numbers.

1. Calculate the sum dynamically.

2. Take the numbers as props

Solution:

```
// App.js
<SumCard firstNumber={4} secondNumber={5} />

// SumCard.js
export function SumCard({ firstNumber, secondNumber }) {
  return (
    <div>
      <h1 className="app-header" style={headerStyle}>
        sum of two numbers
      </h1>
      <div className="app-body">
        <p> First Number : {firstNumber} </p>
        <p> Second Number: {secondNumber} </p>
        <p> Sum: {firstNumber + secondNumber} </p>
      </div>
    </div>
  );
}
```

COPY

Object Data

Challenge for Students:

1. Write a React component to display all the properties of product object on the DOM.

Data:

```
const product = {
    id: "AC1023",
    name: "Air Conditioner",
    price: 29600,
    specification: "1 Ton, 4 Star Rating",
    warranty: "5 Years Compressor Warranty"
};
```

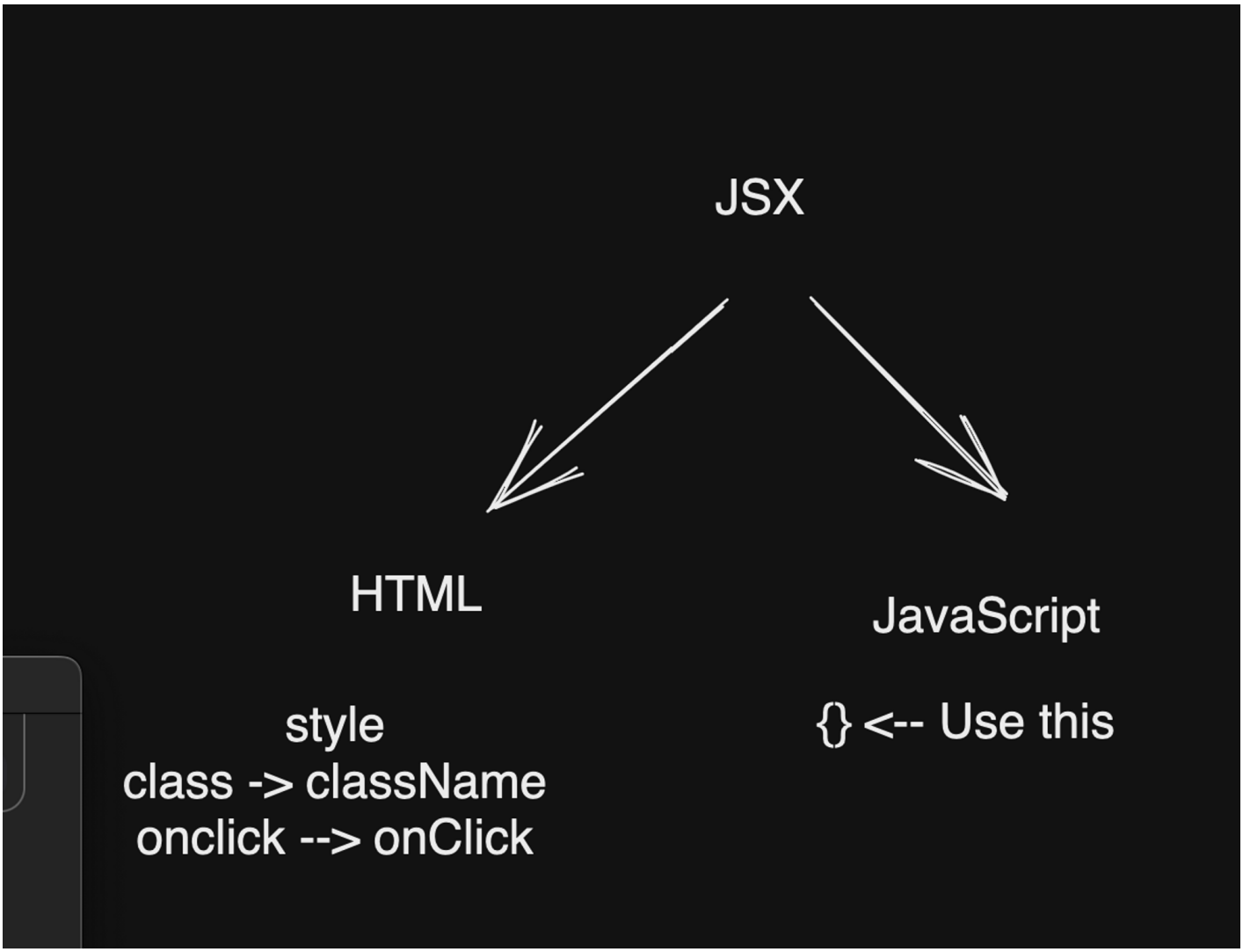
COPY

Solution

```
export default function Product() {
  const product = {
    id: "AC1023",
    name: "Air Conditioner",
    price: 29600,
    specification: "1 Ton, 4 Star Rating",
    warranty: "5 Years Compressor Warranty"
  };

  return (
    <div>
      <h1>{product.name}</h1>
      <p>Price: INR {product.price}</p>
      <p>Specification: {product.specification}</p>
      <p>Warranty: {product.warranty}</p>
    </div>
  );
}
```

COPY



Array Data

Live Example:

1. Write a React component to list out all fruits name and their price from the fruits array in the format `{name} : INR {price}`.

   Data:

```
const fruits = [
    { id: 1, name: "Apple", price: 20 },
    { id: 2, name: "Orange", price: 54 },
    { id: 3, name: "Grapes", price: 30 },
    { id: 4, name: "Pineapple", price: 70 }
  ];
```

COPY

Solution

```
// Fruits.js
export default function Fruits() {
  const fruits = [
```

```
    { id: 1, name: "Apple", price: 20 },
    { id: 2, name: "Orange", price: 50 },
    { id: 3, name: "Grapes", price: 30 },
    { id: 4, name: "Pineapple", price: 70 }
  ];
  return (
    <div>
        <ul>
          {fruits.map((fruit) => (
              <li key={fruit.id}>
              {fruit.name} : INR {fruit.price}
            </li>
          ))}
        </ul>
  </div>
  );
}
```

COPY

https://codesandbox.io/s/wizardly-surf-oyfdb2?file=/src/App.jsx

Live Challenge

```
export default function App() {
  return (
    <div className="App">
      <h1 className="app-header">tanaypratap's box</
      h1>
      <div className="app-body">
        <ProductList productList={phones}
        productHeader={"phones"} />
      </div>
    </div>
  );
}
```

Conditional Styling

Live Example:

1. Display the fruits in red colour, for which the price is less than INR 50.

Solution

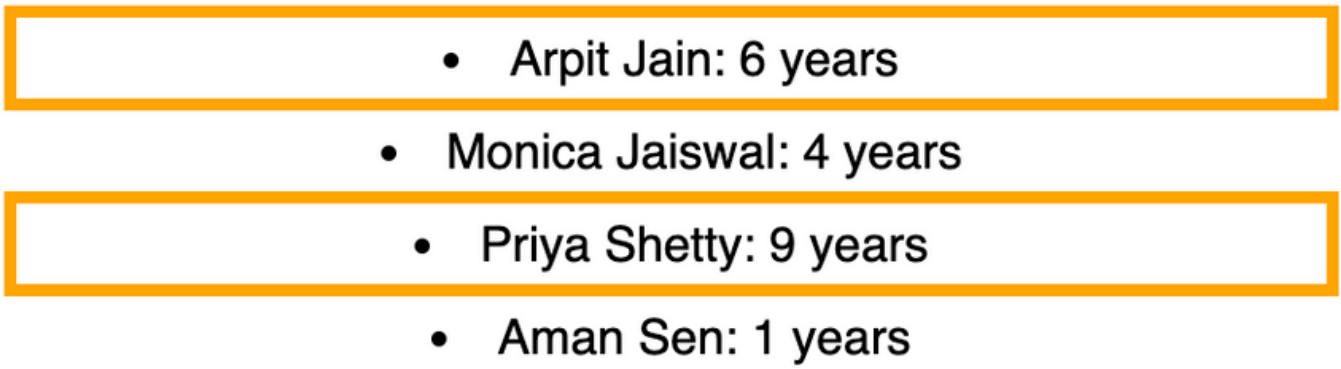
```
// Fruits.js
export default function Fruits() {
  const fruits = [
    { id: 1, name: "Apple", price: 20 },
    { id: 2, name: "Orange", price: 50 },
    { id: 3, name: "Grapes", price: 30 },
    { id: 4, name: "Pineapple", price: 70 }
  ];
  return (
    <div>
      <h1>Fruits</h1>
      {fruits.map((fruit) => (
        <li
          key={fruit.id}
          style={{ color: fruit.price < 50 ? "red" : "black" }}
        >
          {fruit.name} : INR {fruit.price}
        </li>
      ))}
    </div>
  );
}
```

COPY

Challenge for Students:

1. Given an employee data. List out the employee details and add a border to the employee details who have more than 5 years of experience.

Employee List



Data:

```
const employees = [
    { id: "E1", name: "Arpit Jain", workExperience: 6 },
    { id: "E2", name: "Monica Jaiswal", workExperience: 4 },
    { id: "E3", name: "Priya Shetty", workExperience: 9 },
    { id: "E4", name: "Aman Sen", workExperience: 1 }
  ];
```

COPY

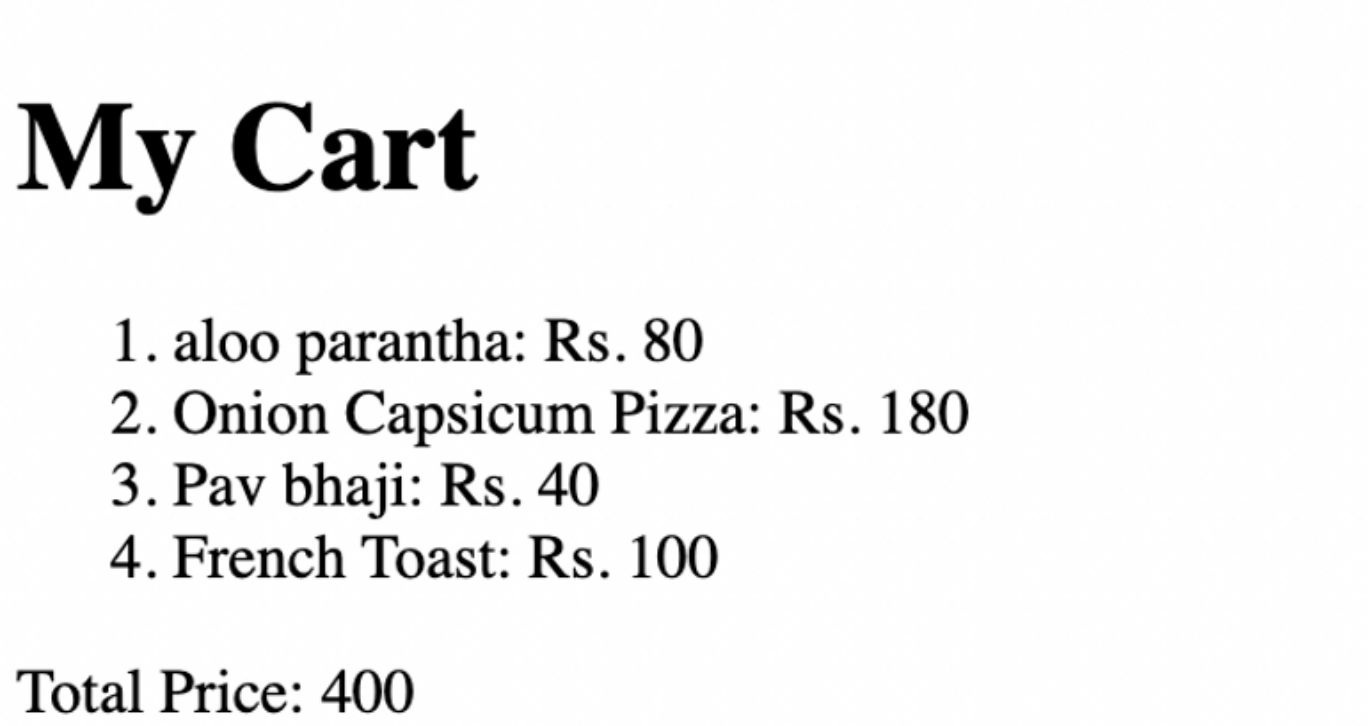
Solution

```
export default function Employee() {
  const employees = [
    { id: "E1", name: "Arpit Jain", workExperience: 6 },
    { id: "E2", name: "Monica Jaiswal", workExperience: 4 },
    { id: "E3", name: "Priya Shetty", workExperience: 9 },
    { id: "E4", name: "Aman Sen", workExperience: 1 }
  ];
  return (
    <div>
      <h1>Employee List</h1>
      {employees.map((employee) => (
        <li
          key={employee.id}
          style={{
            border: employee.workExperience > 5 ? "4px solid orange" : "",
            padding: "4px"
          }}
        >
          {employee.name}: {employee.workExperience} years
        </li>
      ))}
    </div>
  );
```

```
}
```

COPY

2. Given an array of objects representing your cart containing some food items ordered online. Each object consists of: id, name and price. Write a React component that shows these items in an ordered list and in the end calculate and show the “total price” using reduce.



Data:

```
const mycart = [
  {
    id: 1,
    name: "aloo parantha",
    price: 80
  },
  {
    id: 2,
    name: "Onion Capsicum Pizza",
    price: 180
  },
  {
    id: 3,
    name: "Pav bhaji",
    price: 40
  },
  {
    id: 4,
    name: "French Toast",
    price: 100
```

```
  }
];
```

COPY

Solution

```
const mycart = [
  {
    id: 1,
    name: "aloo parantha",
    price: 80
  },
  {
    id: 2,
    name: "Onion Capsicum Pizza",
    price: 180
  },
  {
    id: 3,
    name: "Pav bhaji",
    price: 40
  },
  {
    id: 4,
    name: "French Toast",
    price: 100
  }
];

export default function FoodCart() {
  return (
    <div>
      <h1>My Cart</h1>
      <ol>
        {mycart.map(({ id, name, price }) => (
          <li key={id}>
            {name}: Rs. {price}
          </li>
        ))}
      </ol>
      <p>Total Price: {mycart.reduce((totalPrice, item) => (totalPrice += item.price), 0)
    </div>
  );
}
```

COPY

useState Hook

When we declare a state variable with useState, it returns a pair — an array with two items. The first item is the current value, and the second is a function that lets us update it.

COPY

Changing the state upon an event

Live Example:

1. Write a Counter Component which has two buttons `+` and `-` and an initial count of 0 on the screen. Both the buttons will increase/decrease the count by 1 respectively. Display the increasing/decreasing count on click of the buttons.

   https://codesandbox.io/s/cool-marco-pcg7dy?file=/src/App.jsx

2. Write a React component to list out all fruits name and their price from the fruits array in the format `{name} : INR {price}`. On click of a button 'Highlight Budget Fruits', display the fruits in red colour, for which the price is less than INR 50.

   https://codesandbox.io/s/jovial-cookies-fjg66h?file=/src/App.jsx

Challenge for students

1. Given an employee data. List out the employee details on DOM. Then upon click of a button "Highlight Employees", add a border to the employee details who have more than 5 years of experience.

   Data:

```
const employees = [
    { id: "E1", name: "Arpit Jain", workExperience: 6 },
    { id: "E2", name: "Monica Jaiswal", workExperience: 4 },
    { id: "E3", name: "Priya Shetty", workExperience: 9 },
    { id: "E4", name: "Aman Sen", workExperience: 1 }
  ];
```