Exercise 9

1. Creating a Product Card

Create the product card and render it to the DOM. The product card needs:

1. The product image (`img` element).
    - `src` attribute must be equal to the `imageUrl` variable.
    - `alt` attribute must be equal to the string `Computer`.
    - `width` attribute must be equal to the `256`.
2. The product title (`h3` element).
    - Its text content must be `Computer`.
3. The product price (`span` element).
    - Its text content must be `Price: $129.99`.
4. The product description (`p` element).
    - Its text content must be `New Model`.

Hint:

1. The `src` attribute must contain the `imageUrl` variable as a value. Use curly braces `{}`.

2. The `alt` attribute must contain the string `"Computer"` as a value. Enclose it in double quotes.

3. The `width` attribute must contain the number `256` as a value. Use curly braces `{}`.

3. Ensure that the `h3`, `span`, and `p` elements have correct values.

2. Functional Components

Creating an Encyclopedia Source

Create an encyclopedia source of two cards. Each card will display an image and a description. The goal is to practice creating functional components and passing data to child components using props.

Instructions:

1. Create a `Card` component as a reusable card template.
2. In the `Card` component, accept the following props:
    - `link`: The URL of the image for the card.
    - `alt`: The alternative text for the image.
    - `description`: The description text for the card.
3. Use the `Card` component twice in the `App` component to create two cards with the following content:

- **Card 1** (About the Sun):
    - Image URL: `https://codefinity-content-media.s3.eu-west-1.amazonaws.com/code-1/react/introduction-to-react/sun.png`
    - Alternative text: Provide an appropriate description.
    - Description: "The sun shone brightly, casting a warm glow across the tranquil beach."
- **Card 2** (About the Mountain):
    - Image URL: `https://codefinity-content-media.s3.eu-west-1.amazonaws.com/code-1/react/introduction-to-react/mountain.png`
    - Alternative text: Provide an appropriate description.
    - Description: "The hiker gazed in awe at the majestic mountain towering before them."

Ensure that both the `Card` and `App` components are appropriately modified to display the cards correctly.

Hint:

1. You can use the same `Card` component for both cards and control their content using props.

2. To pass the URL as a prop, create a prop called `link` and provide the URL as its value. Enclose the value in double quotes since it's a string.
3. Similarly, create props for `alt` and `description` and provide the appropriate values.
4. To use these props within the child component `Card`, access them using dot notation since `props` is an object.

3. Conditional Rendering with the && Operator

We want to notify students who have passed an exam. If a student's score exceeds 60 points, we'll display a success message with their name and score.

```
const Notification = (props) =>
  props.mark > 60 && (
    <p>
      {props.name} has passed the test with {props.mark} points.
    </p>
  );
```

The `Notification` component conditionally renders a paragraph `<p>` element based on the `mark` prop value.

4. Conditional Rendering with the Ternary Operator

Consider a scenario where we want to greet users differently based on their logged-in. The `Greeting` component demonstrates conditional rendering with the ternary operator.

```
const Greeting = (props) =>
  props.loggedIn ? (
    <p>Welcome to the home page, {props.name}.</p>
  ) : (
    <p>Hello {props.name}, could you please log in.</p>
  );
```

In this example, the `Greeting` component greets users differently based on the `loggedIn` prop value.

5. Render a Data Collection

We will have an `App` component that will pass the prop `toys`, an array of objects. The `ToyCard` component will utilize the `map()` method to render each toy in the array.

```jsx
// Data from backend
const toysData = [
  { id: "id-1", name: "Rainbow Sparkle Unicorn Plush" },
  { id: "id-2", name: "Jungle Adventure Playset" },
  { id: "id-3", name: "Magical Princess Castle Dollhouse" },
  { id: "id-4", name: "RoboBot Transformer Robot" },
  { id: "id-5", name: "SuperBlast Action Figure" },
];

// Child component
const ToyCard = (props) => (
  <ul>
    {props.toys.map((toy) => (
      <li>{toy.name}</li>
    ))}
  </ul>
);

// Parent component
const App = () => (
  <>
    <ToyCard toys={toysData} />
  </>
```

Let's fix the app using the `key` props for the items. The `key` prop is set on the element that will be rendered multiple times within an array of data.