Trainer guide
Examples and exercises

React-2 – Props

Contents

Trainer demo | React props are like HTML attributes

Link to environment

This example relates props to HTML attributes, as they are passed in a very similar way.

src/App.jsx

- In this example, we have a simple React app that displays a single <img> element on the page.
- As we've seen, we can pass all the usual HTML attributes like `src` and `alt`.
- But remember that this is not HTML - it's JSX.
- `src` and `alt` are pieces of information that we've passed to this JSX element.
- In React, the pieces of information that you pass to a JSX element are called "**props**", which is short for "properties". Props are how you pass data from one component to another (i.e. from a parent component to a child component).
- Props can be strings, written within quotation marks, as you can see here
- But you can pass **any** JavaScript values as props, including numbers, booleans, objects, arrays, and functions.
    - If the value you're passing is not a string, then you place it in curly braces, as shown with the "width" and "height" props.
- The props can pass to an <img> tag are defined by the HTML standard. But when you make your own components, you can pass whatever props you'd like.
- We'll see this in the next example.

Trainer demo | Passing props to custom components

Link to environment

In this demo, you will show learners how to access props passed to a child component.

src/App.jsx

- In this app, we have a little shop that sells fruits.
- *Focus on the contents of the App component.*
    - We have an ItemCard component that we reuse 4 times.
    - We're passing in different props each time like the symbol, name and price of the item.
    - However, we can see on the page that all we see is lemons.
- *Focus on the ItemCard component.*
    - Highlight that the data (symbol, name, price) is hard-coded into the component, so the component will display a lemon every time.
    - What do we want instead?
        - We want the ItemCard component *to* **know what props it receives**, so that we can display that information.
- *Add a `props` parameter to the ItemCard function*
    - Every component in React accepts a parameter called `props`.
    - `props` is a JavaScript object that contains all of the props that were passed to this component.
    - Question - how do you access properties of an object in JavaScript?
        - Answer: dot notation
    - For example, instead of hard-coding the fruit symbol, we can instead access it from props:
        - *Write `const symbol = props.symbol`*
    - Likewise, we can access the name and price:
        - *`const name = props.name`*
        - *`const price = props.price`*
- The different items are now displayed correctly on the page.

- *Demo destructuring props on a single line within the component body*
    - *`const { symbol, name, price } = props`*
- *Then, demo destructuring props within the function parameters:*

```
function ItemCard({ symbol, name, price }) {
  return (
    <div className="item-card">
      <div className="symbol">{symbol}</div>
      <h3>{name}</h3>
      <p>£{price.toFixed(2)}</p>
    </div>
  )}
```

React-2a | Exercise: Create a reusable component with props

Link to environment

This exercise gives learners practice on creating their own reusable components using props.

src/App.jsx

- Here is a web app that provides information about different animals.
- The code is starting to look repetitive. We could improve this by creating a reusable component instead of repeating the same code.

Exercise

- Your task is to refactor this code by creating a reusable component called AnimalCard that accepts the following props:
    - symbol
    - name
    - classification

Solve the exercise live before moving on.

Solution reference

React-2b | Example: The children prop

Link to environment

This example demonstrates the special `children` prop in React.

src/App.jsx

- Up until now, we've written our React components using self-closing tags.
- However, it's also possible to write React components where there is content between opening and closing tags, like HTML elements.
- For example, when we use this Card component, there is an opening tag <Card> and a closing tag </Card>.
- The elements between the opening and closing tags are called "children".
- Let's look at how to access these children from the Card's perspective.

src/components/Card.jsx

- There is a special prop that is automatically available to every component in React – that is, the `children` prop. The `children` prop represents the content that is nested inside the opening and closing tags of this component when it is used.
- To render a component's children, simply write reference the `children` prop within curly braces, like we've done here.
- In this case, we've rendered the children within a with a class name of "card". This means the same styling is applied every time we use the Card component.

src/App.jsx

- Looking at it from the other side once again, now when you see elements being passed within opening and closing tags, remember that those elements are passed via the `children` prop.
- The children prop can be used to pass any type of content, including HTML elements, strings, numbers, or other React components. It's a powerful feature of React that helps us to build reusable and flexible components.