TEST YOUR KNOWLEDGE #1

In this exercise, you will create a page containing five React components. The HTML that your page must eventually render has been provided in the file lab11a-test01-markup-only.html. The CSS has been provided, though it's possible (but not necessary) you may want to change it based on the HTML that your components render. The starting code provides an array of painting objects (later in the chapter, you will learn how to fetch data from an API in React instead). As shown in Figure 11.11, there are three user interactions: clicking on a painting in the list will display the painting's data in the form; changing form data will change it in the list as well; clicking undo will revert the painting data to its original state.

With React, you may prefer to start working first on the most "outer" component (in this case `App`), or you may decide to start working first from the most "inner" component (which in this case is `PaintingListItem`), or start with the simplest (in this case that would be the `Header` component). For this exercise, we will focus initially just on rendering data and markup, and then add behaviors later.

1. Create the `Header` functional component. The button won't do anything yet. Remember that you can look at lab11a-test01-markup-only.html to see what markup your component should render.
2. Create the `PaintingList` functional component. Initially just have this component render the root `<section>` element with some temporary text.
3. Create the `EditPaintingForm` functional component. Initially just have this component render the root `<section>` element with the `<form>` and `<div>` elements. Assume a single painting object is passed via props (as shown in Figure 11.12) whose data will be displayed by the component.
4. Modify the `App` functional component so that it uses these three components (there is boilerplate text in the start file which indicates where they are to be located).
5. Create the `PaintingListItem` functional component. Assume a single painting object is passed via props (as shown in Figure 11.12) whose data will be displayed by the component.
6. Your `PaintingList` component is going to display multiple `PaintingListItem` components (one for each painting). Assume the entire array of paintings is passed (as shown in Figure 11.12). Use the `map()` function to render each painting object as a `PaintingListItem` (see Listing 11.2 for reminder how to do this). Verify this works.
7. Now start adding in the state data. As shown in Figures 11.11 and 11.12, state will be implemented in the parent `App` component. Since `App` was initially created as a functional component, you can use the Hooks approach; if you wish to use the traditional React state approach, you will have to convert `App` into a class component. What state data will you need? As can be seen in Figure 11.12, you will need a list of paintings and the current painting (whose data will be editable in the form and which will be displayed with the different background color in the list). This state data will be passed via props to your other components. Verify this works.

8. Now start adding in the behaviors. Add an `onClick` event handler to the `<div>` element in the `PaintingListItem` component (we are doing this instead of having an explicit Edit link as in Figure 11.10). This will call the change handler that is passed into `PaintingList` and `PaintingListItem`. As shown in Figures 11.11 and 11.12, this will be implemented in the parent `App` component. The change handler will change the current painting state variable. Verify this works.
9. Add in the editing behavior. Use the controlled form components approach. The change handler in `EditPaintingForm` should create a new painting object which is a copy of the current painting, except for what has changed in the form, and then pass that object to the update method in the `App` parent. When this is completed, changing a value in the form will also change the value in the painting list display as well. Verify this works.
10. Finally, add in some conditional rendering to `PaintingListItem` so that it displays the current painting differently and implement the Undo changes button (simply set the paintings state variable equal to the initial data array).

TOOLS INSIGHT

React Component Libraries

At the time of writing (spring 2020), React has become extremely popular with developers. According to a yearly survey of over 21,000 developers (https://2019.stateofjs.com/), React continues to have the highest numbers in terms of satisfaction and usage for JavaScript frameworks amongst the respondents to this survey.

One of the advantages of using a development technology that has broad usage is that a flourishing ecosystem of related tools and libraries inevitably emerge, which in turn makes that technology even more attractive. For many years, this was a key attraction of jQuery. At present, React has displaced jQuery in this regard (for instance, in the above mentioned survey, 72% of respondents claimed to be actively using React, while only 3% claimed to be still actively using jQuery).

React's component approach is especially well suited to creating libraries of elements that can be used by others or to make use of third-party component libraries. The sheer number of available React component libraries makes it impossible to mention more than a small handful here.

Back in Chapter 4 you briefly learned about CSS frameworks, some of which provided comprehensive libraries of styles (for instance, Bootstrap or Bulma), while others were more minimal or utility-oriented. Similarly, there are comprehensive React component libraries, such as Ant Design, Material-UI, or Fabric React, which encapsulate a wide-range of behaviors and appearances using a consistent visual design language. Other React component libraries focus on particular tasks, such as form validation (React Hook Form, Formik), animation (react-spring, react-animations), charting (Vis, Rechart, Nivo), data retrieval (axios, Apollo), state management (Redux), or testing (Jest, React Testing Library).