Forms in React

Earlier we introduced a new button to fetch data explicitly with a button click. We'll advance its use with a proper HTML form, which encapsulates the button and input field for the search term with its label.

Forms aren't much different in React's JSX than in HTML. We'll implement it in two refactoring steps with some HTML/JavaScript. First, wrap the input field and button into an HTML form element:

```
const App = () => {
  ...

  return (
    <div>
      <h1>My Hacker Stories</h1>

      <form onSubmit={handleSearchSubmit}>
        <InputWithLabel
          id="search"
          value={searchTerm}
          isFocused
          onInputChange={handleSearchInput}
        >
          <strong>Search:</strong>
        </InputWithLabel>

        <button type="submit" disabled={!searchTerm}>
          Submit
        </button>

      </form>
      <hr />

      ...
    </div>
  );
};
```

src/App.js

Instead of passing the `handleSearchSubmit` handler to the button, it's used in the new form element. The button receives a new `type` attribute called

`submit`, which indicates that the form element handles the click and not the button.

Since the handler is used for the form event, it executes `preventDefault` in React's synthetic event. This prevents the HTML form's native behavior, which leads to a browser reload.

```
const App = () => {
  ...
  const handleSearchSubmit = event => {

    setUrl(`${API_ENDPOINT}${searchTerm}`);

    event.preventDefault();
  };

  ...
};
```

src/App.js

Now we can execute the search feature with the keyboard's `Enter` key. In the next two steps, we will only separate the component into its standalone SearchForm component:

```
const SearchForm = ({
  searchTerm,
  onSearchInput,
  onSearchSubmit,
}) => (
  <form onSubmit={onSearchSubmit}>
    <InputWithLabel
      id="search"
      value={searchTerm}
      isFocused
      onInputChange={onSearchInput}
    >
      <strong>Search:</strong>
    </InputWithLabel>

    <button type="submit" disabled={!searchTerm}>
      Submit
    </button>
  </form>
);
```

src/App.js

The new component is used by the App component. The App component still manages the state for the form, because the state is used in the App component to fetch data passed as props (`stories.data`) to the List

component:

```
const App = () => {
  ...

  return (
    <div>
      <h1>My Hacker Stories</h1>
      <SearchForm
        searchTerm={searchTerm}
        onSearchInput={handleSearchInput}
        onSearchSubmit={handleSearchSubmit}
      />

      <hr />

      {stories.isError && <p>Something went wrong ...</p>}

      {stories.isLoading ? (
        <p>Loading ...</p>
      ) : (
        <List list={stories.data} onRemoveItem={handleRemoveStory} />
      )}
    </div>
  );
};
```

src/App.js

Forms aren't much different in React than HTML. When we have input fields and a button to submit data from them, we can give our HTML more structure by wrapping it into a form element with a `onSubmit` handler. The button that executes the submission needs only the "submit" `type`.

```
  □ □ □□  □   ã□  F   □□  □   □  )□     □   9□  5□  @@  □   °□  n□  □PNG
□
   IHDR   □   □□□   (-□S   äPLTE"""""""""""""""""""2PX=r□)7;*:>H□¤-BGE□□8do5Xb6[eK□®K□¯1MU
□
   IHDR   □   □□□   ×©ÍÊ  □ePLTE"""""""""""""""""""""""""2RZN¢¹J□«3R[J□¬)59YÁÞ0KS4W`Q«ÄL□²%
?^q÷ñíÛ□ï.},□ìsæÝ_TttÔ¾ □1#□□/(ì□-[□□□è`□è`Ì□ÚïÅðZ□d5□□□□?ÎebZ¿Þ□i.Ûæ□□□ìqÎ□+1°□}Â□5
□
   IHDR       □□   D¤□Æ  □APLTE   """""""""""""""""""""""""""2RZVºÖ_ÔôU·Ñ=r□$()'25]ÎíC□□0
□
   IHDR   @   @□□   □·□ì  □:PLTE   """""""""""""""""""""""""""""""""""""""""""""""""""""""""
¢ßqÇ8Ù□´□mKË±mÆ¶mÛü·yi!è□ÎªYÏuë ÀÏ_Àï?i÷□ý+ò□□ÄA□|□ù{□□´?¿□_En□).□JËD¤<□
©¬¢Z\Ts©R*□(□   ¯©□J□□□□u□X/□4J□9□¡5·DEµ4kÇ4□&i¥V4Ú□¡®Ð□□¯□vsf:àg,□¢èBC»î$¶□ºÍùî□□á□@□
-ê>Û□º«¢XÕ¢î}ß¨ëÛÑ;□ÃöN´□ØvÅý□Î¸ÿ1 □ë×ÄO@&v/Äþ_□ö\ô□Ç\í.□□½+0□□;□□□!□fÊ□¦´Ó%Â JY·O□Â□'
```

Exercises:

- Confirm the changes from the last section.
- Try the code without `preventDefault`.

- Read more about preventDefault for Events in React.

- Read more about React Component Composition.