Reusable React Component

Learn to reuse a react component.

We'll cover the following

- Exercises:

Have a closer look at the Search component. The label element has the text "Search: "; the `id/htmlFor` attributes have the `search` identifier; the value is called `search`; and the callback handler is called `onSearch`. The component is very much tied to the search feature, which makes it less reusable for the rest of the application and non search-related tasks. It also risks introducing bugs if two of these Search components are rendered side by side, because the htmlFor/id combination is duplicated, breaking the focus when one of the labels is clicked by the user.

Since the Search component doesn't have any actual "search" functionality, it takes little effort to generalize other search domain properties to make the component reusable for the rest of the application. Let's pass an additional `id` and `label` prop to the Search component, rename the actual value and callback handler to something more abstract, and rename the component accordingly:

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

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


     <InputWithLabel
       id="search"
       label="Search"

       value={searchTerm}

       onInputChange={handleSearch}
     />

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

```
};


const InputWithLabel = ({ id, label, value, onInputChange }) => (

 <>

   <label htmlFor={id}>{label}</label>
    

   <input

     id={id}

     type="text"
     value={value}

     onChange={onInputChange}

   />
 </>
);
```

src/App.js

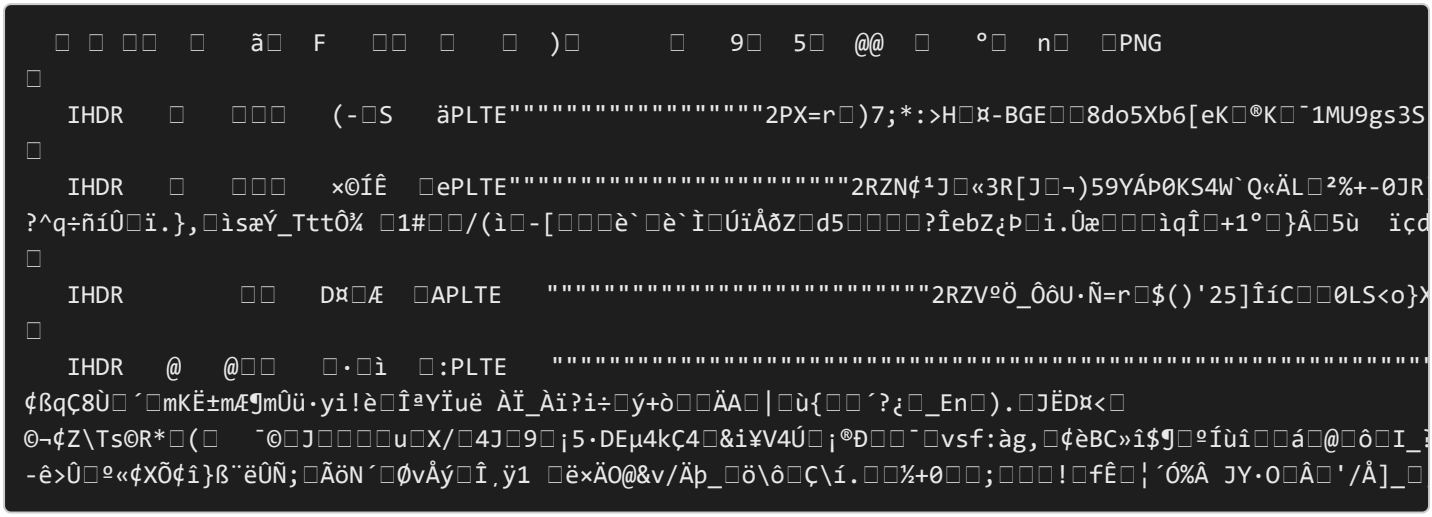
It's not fully reusable yet. If we want an input field for data like a number (`number`) or phone number (`tel`), the `type` attribute of the input field needs to be accessible from the outside too:

```
const InputWithLabel = ({
 id,
 label,
 value,

 type = 'text',

 onInputChange,
}) => (
 <>
   <label htmlFor={id}>{label}</label>
    
   <input
     id={id}

     type={type}
     value={value}
     onChange={onInputChange}
   />
 </>
);
```

src/App.js

From the App component, no `type` prop is passed to the InputWithLabel component, so it is not specified from the outside. The default parameter from the function signature takes over for the input field.

function signature takes over for the input field.



With just a few changes we turned a specialized Search component into a more reusable component. We generalized the naming of the internal implementation details and gave the new component a larger API surface to provide all the necessary information from the outside. We aren't using the component elsewhere, but we increased its ability to handle the task if we do.

Exercises:

- Confirm the changes from the last section.
- Read more about Reusable React Components.
- Before we used the text "Search:" with a ":". How would you deal with it now? Would you pass it with `label="Search:"` as prop to the InputWithLabel component or hardcode it after the `<label htmlFor={id}>{label}:</label>` usage in the InputWithLabel component? We will see how to cope with this later.