

useField() Custom Hook

The useField() is a React hook created for forms with *many* inputs. Normally an event handler would be needed for each input, useField funnels them into one.

What: A custom hook

When: When you have a form with multiple inputs with each input being saved to state.

Where: Can be imported from ../hooks/usefield, all hook rules apply.

Why: To reduce the amount of event handlers needed for input fields.

How: Import, set a variable to hook, use variable in <input tag />, call variable using .value or .onResetHandler().

This:

XXXXXXXXXXXXXXXXXXXX

```
const App = () => {  
  const [name, setName] = useState("");  
  
  return (  
    <div>  
      <form>  
        name:  
        <input  
          type="text"  
          value={name}  
          onChange={(event) => setName(event.target.value)}  
        />  
      </form>  
    </div>  
  );  
};  
  
export default App;
```

Becomes:

✓✓✓✓✓✓✓✓✓✓✓✓✓✓


```
Complexity is 5 Everything is cool!  
const App = () => {  
  const name = useField("text")  
  
  return (  
    <div>  
      <form>  
        name:  
        <input  
          {...name}  
        />  
      </form>  
    </div>  
  );  
};
```

Syntax of the hook itself:

It needs to be passed a string and will return an array. The first index of that array is an object that holds the value. The second index is a function that will reset the value when called.

```
import { useState } from "react"; 4.1k (gzipped: 1.8k)
```

Complexity is 4 Everything is cool!

```
const useField = (type) => {   
  const [value, setValue] = useState("");  
  
  const onChange = (event) => {  
    | setValue(event.target.value);  
  };  
  
  const onresethandler = () => {  
    | setValue("");  
  };  
  
  return [{ type, value, onChange }, onresethandler];  
};  
  
export default useField;
```

To use it:

```
import { useField } from "../../hooks/useField";
```

```
useField("HTML input type as string")
```

Examples:

```
const username = useField();
const password = useField("password");
const email = useField("email");
const url = useField("url");
const birthdate = useField("date");
```

Once the variable is set to the hook, that variable can be used in an input tag. The parameter *is not required*, but if used should always be a string. Passing a parameter is beneficial for validation and easier user entry.

By passing “password” we are able to get an input field that shows dots.



By passing “email” we are able to get an input field that requires “@”



❗ Please include an '@' in the email address. 'invalid input' is missir '@'.

By passing “date” we are able to get an input field like so:




For more examples on different input values that can be passed into useField()

go to: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#input_types

Form/input Example:

As the user enters text into the field it will update state. That state can be accessed at `username.value`.

<div>Your Name: <input type="text" value="Asht"/></div> <div>Password: <input type="password"/></div> <div>Email Address: <input type="text"/></div>		Field values in state: state of username: Asht state of password:
--	---	---

Any variables set to `useField()` will have the property: `“.onResetHandler()”`. This property is useful to clear the form once the data has been sent and stored elsewhere.

```
const clearForm = (event) => {  
  event.preventDefault();  
  usernameResetHandler();  
  passwordResetHandler();  
  emailResetHandler();  
  urlResetHandler();  
  birthDateHandler();  
};
```

I have shared a practice app, to see `useField` in action, that can be found on Github at:

https://github.com/Ashton-Bennett/test_apps/tree/usefield