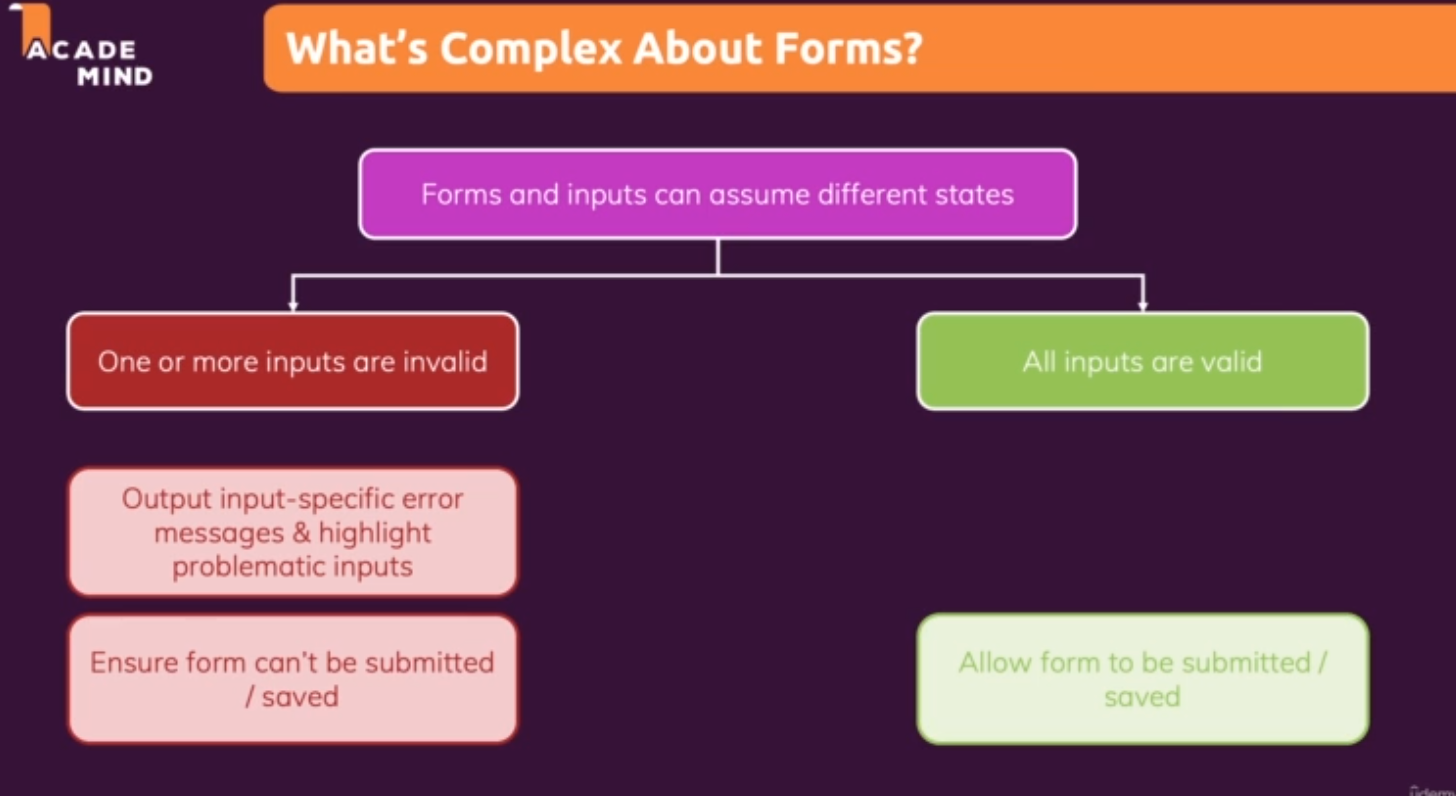
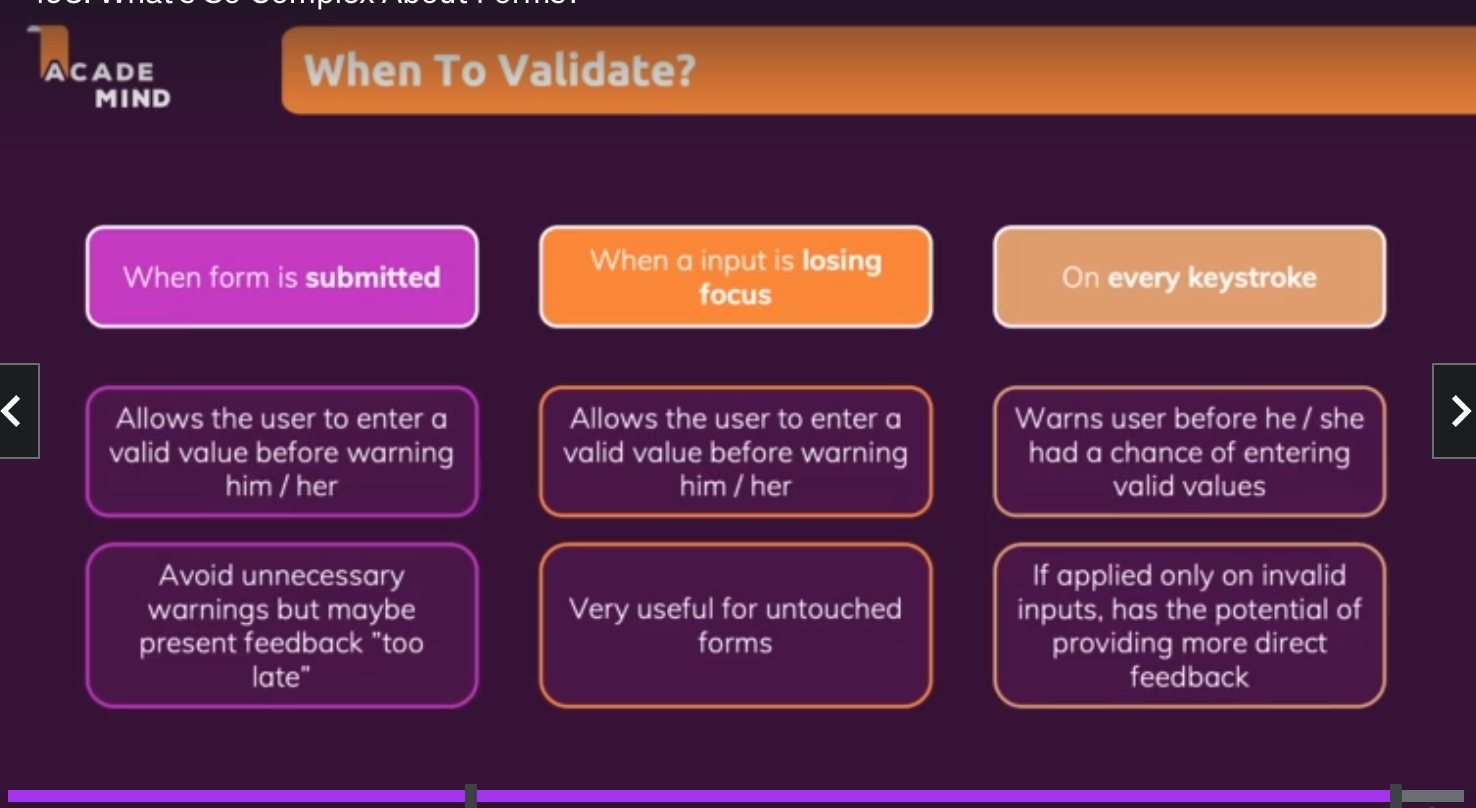
Lets focus on forms with react. Forms can have different inputs like valid or invalid.





We can use useRef or useState for form validation as below:-



The above code is an example with useRef as well as state. It is recommended to clear the forn input value with useState rather than ref because with ref it would be DOM manipulation.

We should perform validation on browser as well as on the server.

To add basic validation we can add some more state . We can also use react hookform library for form input validation.

Final code can be refactored as below ➖

<https://github.com/academind/react-complete-guide-code/blob/16-working-with-forms/code/06-refactoring-and-deriving-states/src/components/SimpleInput.js>

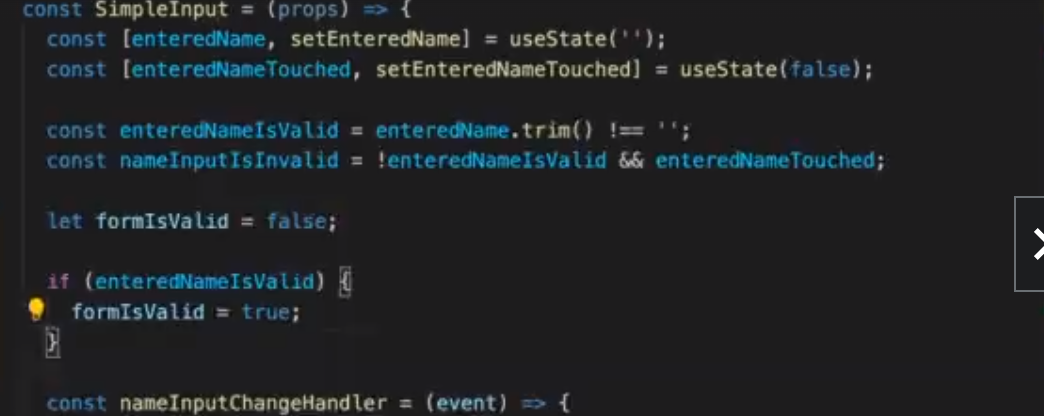




Overall form validity can be handled using an additional state like isFormValid and setFormIsValid. However in most of the cases we do not need state for re-evaluation of the forms because that is not going to add any extra advantage to our application.

We can simply use variables and assign values to it to get the latest form validity .

For example below code can be used as reference.



For complex forms we can use custom hooks to handle validation for handling all the validation logic and we can reuse the same functions.

All the state logic can be handled using custom hook as below.

Custom form hook:-



We can re-use the custom hook components as below:-

import useInput from '../hooks/use-input';

const SimpleInput = (props) => {

const {

value: enteredName,

isValid: enteredNameIsValid,

hasError: nameInputHasError,

valueChangeHandler: nameChangedHandler,

inputBlurHandler: nameBlurHandler,

reset: resetNameInput,

} = useInput((value) => value.trim() !== '');

const {

value: enteredEmail,

isValid: enteredEmailIsValid,

hasError: emailInputHasError,

valueChangeHandler: emailChangeHandler,

inputBlurHandler: emailBlurHandler,

reset: resetEmailInput,

} = useInput((value) => value.includes('@'));

let formIsValid = false;

if (enteredNameIsValid && enteredEmailIsValid) {

formIsValid = true;

}

const formSubmissionHandler = (event) => {

event.preventDefault();

if (!enteredNameIsValid) {

return;

}

console.log(enteredName);

// nameInputRef.current.value = ''; => NOT IDEAL, DON'T MANIPULATE THE DOM

resetNameInput();

resetEmailInput();

};

const nameInputClasses = nameInputHasError

? 'form-control invalid'

: 'form-control';

const emailInputClasses = emailInputHasError

? 'form-control invalid'

: 'form-control';

return (

<form onSubmit={formSubmissionHandler}>

<div className={nameInputClasses}>

<label htmlFor='name'>Your Name</label>

<input

type='text'

id='name'

onChange={nameChangedHandler}

onBlur={nameBlurHandler}

value={enteredName}

/>

{nameInputHasError && (

<p className='error-text'>Name must not be empty.</p>

)}

</div>

<div className={emailInputClasses}>

<label htmlFor='email'>Your E-Mail</label>

<input

type='email'

id='email'

onChange={emailChangeHandler}

onBlur={emailBlurHandler}

value={enteredEmail}

/>

{emailInputHasError && (

<p className='error-text'>Please enter a valid email.</p>

)}

</div>

<div className='form-actions'>

<button disabled={!formIsValid}>Submit</button>

</div>

</form>

);

};

export default SimpleInput;