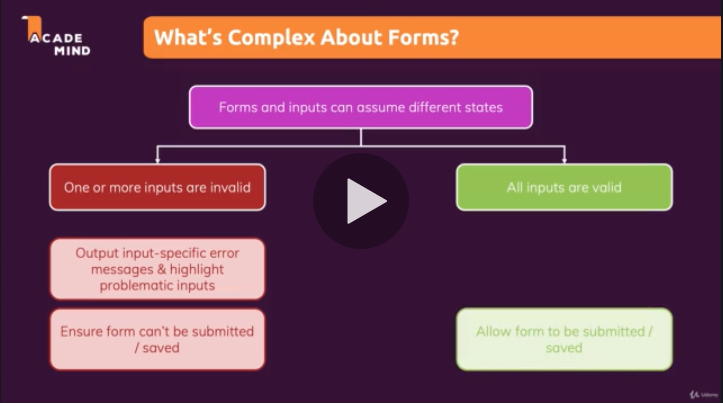
Working with forms and user input

* When to valid :
  + Two states are required to every input valid and invalid





* Read value from input using state:

import { useState } from "react";

const SimpleInput = (props) => {

  const [enteredName, setEnteredName] = useState('');

  const nameInputChangeHandler = event =>{

    setEnteredName(event.target.value);

  };

  const formSubmissionHandler = event =>{

    event.preventDefault();

    console.log(enteredName);

  }

  return (

    <form onSubmit={formSubmissionHandler}>

      <div className='form-control'>

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

        <input type='text' id='name' onChange={nameInputChangeHandler}/>

      </div>

      <div className="form-actions">

        <button>Submit</button>

      </div>

    </form>

  );

};

export default SimpleInput;

* Read input using useRef( ) hook:

import { useRef, useState } from "react";

const SimpleInput = (props) => {

  const nameInputRef = useRef();

  const formSubmissionHandler = event =>{

    event.preventDefault();

    const enteredValue = nameInputRef.current.value;

    console.log(enteredValue)

  }

  return (

    <form onSubmit={formSubmissionHandler}>

      <div className='form-control'>

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

        <input type='text' id='name' ref={nameInputRef}/>

      </div>

      <div className="form-actions">

        <button>Submit</button>

      </div>

    </form>

  );

};

export default SimpleInput;

* useInput custome react hook:

import { useState } from 'react';

const useInput = (validateValue) => {

  const [enteredValue, setEnteredValue] = useState('');

  const [isTouched, setIsTouched] = useState(false);

  const valueIsValid = validateValue(enteredValue);

  const hasError = !valueIsValid && isTouched;

  const valueChangeHandler = (event) => {

    setEnteredValue(event.target.value);

  };

  const inputBlurHandler = (event) => {

    setIsTouched(true);

  };

  const reset = () => {

    setEnteredValue('');

    setIsTouched(false);

  };

  return {

    value: enteredValue,

    isValid: valueIsValid,

    hasError,

    valueChangeHandler,

    inputBlurHandler,

    reset

  };

};

export default useInput;

* Use of useInput hook:

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() !== "");

  let formIsValid = false;

  if (enteredNameIsValid) {

    formIsValid = true;

  }

  const formSubmissionHandler = (event) => {

    event.preventDefault();

    if (!enteredNameIsValid) {

      return;

    }

    console.log(enteredName);

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

    resetNameInput();

  };

  const nameInputClasses = nameInputHasError

    ? "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="form-actions">

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

      </div>

    </form>

  );

};

export default SimpleInput;