1. Create a Hoc that add random color to background of 2 component.

//colorChanger

const HOC = (Prospect) => {

console.log(Prospect); //a component

const color = ["pink", "orange", "gold", "grey", "blue"];

const randomColour = color[Math.floor(Math.random() \* 5)];

return () => {

return (

<div style={{ backgroundColor: randomColour }}>

<Prospect />

</div>

);

};

};

export default HOC;

1. Advance task (use lodash.js)
2. symmetric difference of 2 arrays

const arrOne = [{

    id: 20,

    name: 'alex'

}, {

    id: 30,

    name: 'alina'

}]

const arrTwo = [{

    id: 40,

    name: 'hello'

}, {

    id: 30,

    name: 'world'

}]

result = [{

    id: 20,

    name: 'alex'

}, {

    id: 40,

    name: 'hello'

}]

&&

res= [{

     id: 30,

     name: alina

},{

        id: 30,

        name: ‘world’

    }]

1. const str = ['u', 'ec']

const arr = [{

    storageVal: 'u',

     table: ['E', 'F']

},{

    storageVal: 'data',

    id: 3

}, {

    storageVal: 'ec',

    table: ['E']

}]

get all the tables ['E', 'F'], ['E']

1. Flatten an array const a = [['E'], ['F']]

Output: ['E', 'F']

1. const t = ['E', 'F'], [['F'], ['G']]

Output: ['E', 'G', 'F']

1. Create a search box (Optional)

import \_ from "lodash";

const ArrayEx = () => {

const \_ = require("lodash");

//Question 2A

const arrOne = [

{

id: 20,

name: "alex",

},

{

id: 30,

name: "alina",

},

];

const arrTwo = [

{

id: 40,

name: "hello",

},

{

id: 30,

name: "world",

},

];

const symDiff = \_.xorBy(arrOne, arrTwo, "id");

console.log("Question 2A: ", symDiff);

const sameId = (a, b) => {

return a.id == b.id;

};

const similar = \_.intersectionWith(

\_.unionWith(arrOne, arrTwo, sameId),

arrTwo,

sameId

);

console.log("Question 2A: ", similar);

//Question 2B

const str = ["u", "ec"];

const arr = [

{

storageVal: "u",

table: ["E", "F"],

},

{

storageVal: "data",

id: 3,

},

{

storageVal: "ec",

table: ["E"],

},

];

const result = \_.filter(arr, ({ storageVal }) => \_.includes(str, storageVal));

console.log("Question 2B: ", result);

//Question 2C

const a = [["E"], ["F"]];

const newArray = \_.flatten(a);

console.log("Question 2C: ", newArray);

//Question 2D

const t = [

["E", "F"],

[["F"], ["G"]],

];

const flatArray = \_.uniq(\_.flattenDeep(t));

console.log("Question 2D: ", flatArray);

return null;

};

export default ArrayEx;

1. Complete below tasks using Hooks

* Create an array of object and update its second index
* Create an object, update it's 2 property values
* Practice all datatypes for state variable

import React, { useState } from "react";

import HOC from "./HOC";

const Hooks = () => {

const [array, setArray] = useState([67, 12, 23]);

const [customerData, setCustomerData] = useState([

{

name: "Jason",

customerId: 1,

gender: "male",

},

{ name: "Alice", customerId: 2, gender: "female" },

]);

const handleArray = () => {

const updateArray = [...array];

updateArray[1] = 45;

setArray(updateArray);

};

const handleUpdate = () => {

const updateCustomer = [

...customerData,

{ name: "Eunice", customerId: 3, gender: "female" },

];

updateCustomer[1].customerId = 4;

setCustomerData(updateCustomer);

};

const customerList = customerData.map((info) => (

<>

<p>{info.name}</p>

<p>{info.customerId}</p>

<p>{info.gender}</p>

</>

));

return (

<>

<p>{array.map((e) => e)}</p>

<p>{customerList}</p>

<button onClick={handleArray}>Update Array</button>

<button onClick={handleUpdate}>Update Customer</button>

</>

);

};

export default HOC(Hooks);

1. Create a form using Functional Component. Add validation. (Controlled Input)

import React, { useState } from "react";

const ControlledForm = () => {

const [firstName, setFirstName] = useState("");

const [lastName, setLastName] = useState("");

const [email, setEmail] = useState("");

const [firstNameError, setFirstNameError] = useState({});

const [lastNameError, setLastNameError] = useState({});

const [emailError, setEmailError] = useState({});

const handleSubmit = (e) => {

e.preventDefault();

const isValid = formValidation();

if (isValid) {

setFirstNameError = "";

setLastNameError = "";

setEmailError = "";

}

};

const formValidation = () => {

const firstNameError = {};

const lastNameError = {};

const emailError = {};

let isValid = true;

if (firstName.trim().length === 0) {

firstNameError.firstNameEmpty = "This field is empty.";

isValid = false;

}

if (lastName.trim().length === 0) {

lastNameError.lastNameEmpty = "This field is empty.";

isValid = false;

}

if (email.trim().length === 0) {

emailError.emailEmpty = "This field is empty.";

isValid = false;

}

if (!/\S+@\S+\.\S+/.test(email)) {

emailError.emailWrongInput = "This email format is wrong.";

isValid = false;

}

setFirstNameError(firstNameError);

setLastNameError(lastNameError);

setEmailError(emailError);

return isValid;

};

return (

<form onSubmit={handleSubmit}>

<label>First Name</label>

<input

type="text"

value={firstName}

onChange={(e) => {

setFirstName(e.target.value);

}}

/>

<br></br>

{Object.keys(firstNameError).map((key) => {

return <p style={{ color: "black" }}>{firstNameError[key]}</p>;

})}

<label>Last Name</label>

<input

type="text"

value={lastName}

onChange={(e) => {

setLastName(e.target.value);

}}

/>

<br></br>

{Object.keys(lastNameError).map((key) => {

return <p style={{ color: "black" }}>{lastNameError[key]}</p>;

})}

<label>Email</label>

<input

type="text"

value={email}

onChange={(e) => {

setEmail(e.target.value);

}}

/>

{Object.keys(emailError).map((key) => {

return <p style={{ color: "black" }}>{emailError[key]}</p>;

})}

<button type="submit">Submit</button>

</form>

);

};

export default ControlledForm;

1. Practice Refs for Function & Class Component