8. Distructuring

*var* array = ['javascript', 'css', 'js'];  
*var* [a, ...rest] = array;  
console.log(rest);  
  
*var* course = {  
 name: 'javascript',  
 price: 1000,  
 image: 'image-address',  
 children: {  
 name: 'reactjs'  
 }  
};  
*var* {name, ...rest} = course;  
console.log(rest);  
  
*var* {name: parentName, children: {name}} = course;  
console.log(parentName);  
console.log(name);

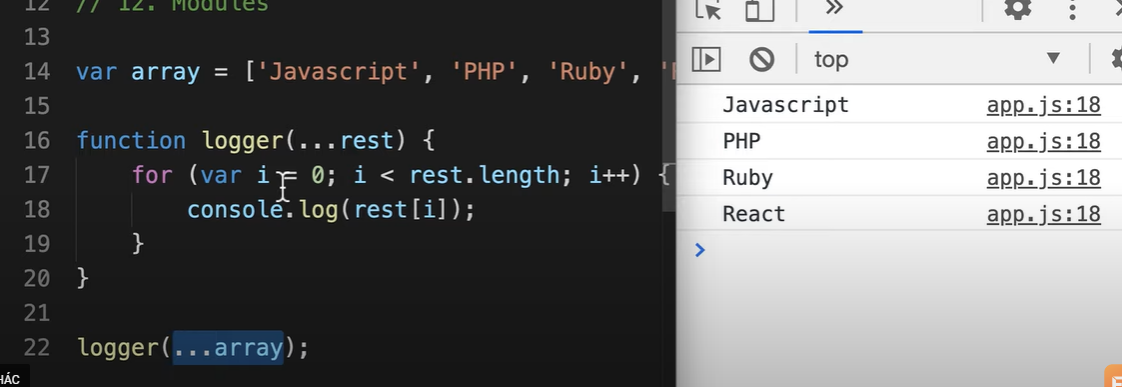
*const* {  
 a = 'default',  
 b,  
 ...rest  
} = {  
 b: 'val1',  
 c: 'val2',  
 d: 'val3'  
};  
  
console.log(a, b, rest); *‘default’,’val1’,{c:’val2’;d:’val3’}*

10. spreat – Nối mãng, object

*var* array1 = ['javascript', 'css', 'js', 'jsx'];  
*var* array2 = ['react', 'react-dom', 'react-dom-dom', 'react-dom-dom'];  
*var* array3 = [...array1, ...array2];  
console.log(array3)

*var* object1 = {  
 name: 'javascript',  
}  
*var* object2 = {  
 price: 1000  
}  
*var* object3 = {

name: ‘abc’,  
 ...object1,  
 ...object2  
}  
console.log(object3)

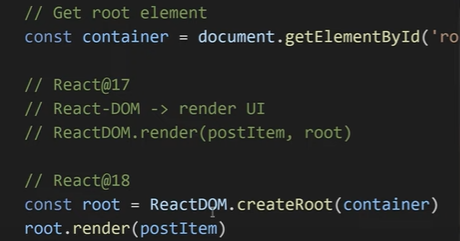


React create element

<script>  
 *const* h1DOM = document.createElement('h1');  
 h1DOM.tite = 'hello';  
 h1DOM.className = 'heading';  
 h1DOM.innerText = 'hello guys!';  
 document.body.appendChild(h1DOM);  
  
 *// react  
 // react.createElement(type,props,children,n)  
 const* h1React = React.createElement('h1',{  
 title: 'hello',  
 className: 'heading'  
 },'Hello guys!')  
  
 *const* ulReact = React.createElement(  
 'ul',  
 {  
 id: 'ul-id',  
 style: 'color: red;'  
 },  
 React.createElement('li',*null*,'Javascript'),  
 React.createElement('li',*null*,'ReactJS')  
 )  
 console.dir(h1DOM);  
 console.log(h1React);  
 console.log(ulReact);  
 document.body.appendChild(ulReact)  
</script>

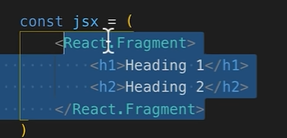
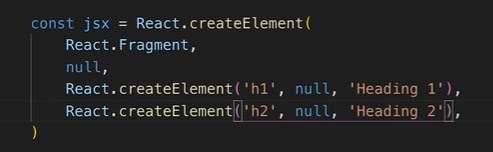
ReactDOM

<script>  
 *const* postItem = React.createElement(  
 'div',  
 {  
 className: 'post-item'  
 },  
 React.createElement(  
 'h2',  
 {  
 title: 'hoc react'  
 },  
 'hoc react'  
 ),  
 React.createElement('p',{},'hoc reactjs')  
 )  
 console.log(postItem)  
 ReactDOM.render(postItem, root)  
</script



JSX

<script *type*="text/babel">  
 *const* course = [  
 {  
 name: 'html,css'  
 },  
 {  
 name: 'web responsive'  
 },  
 {  
 name: 'reactjs'  
 }  
 ]  
  
 *const* jsx = (  
 <ul>  
 {course.map(course=>  
 <li>{course.name}</li>  
 )}  
 </ul>  
 );  
 console.log(jsx);  
 ReactDOM.render(jsx, document.getElementById('root'))  
</script>

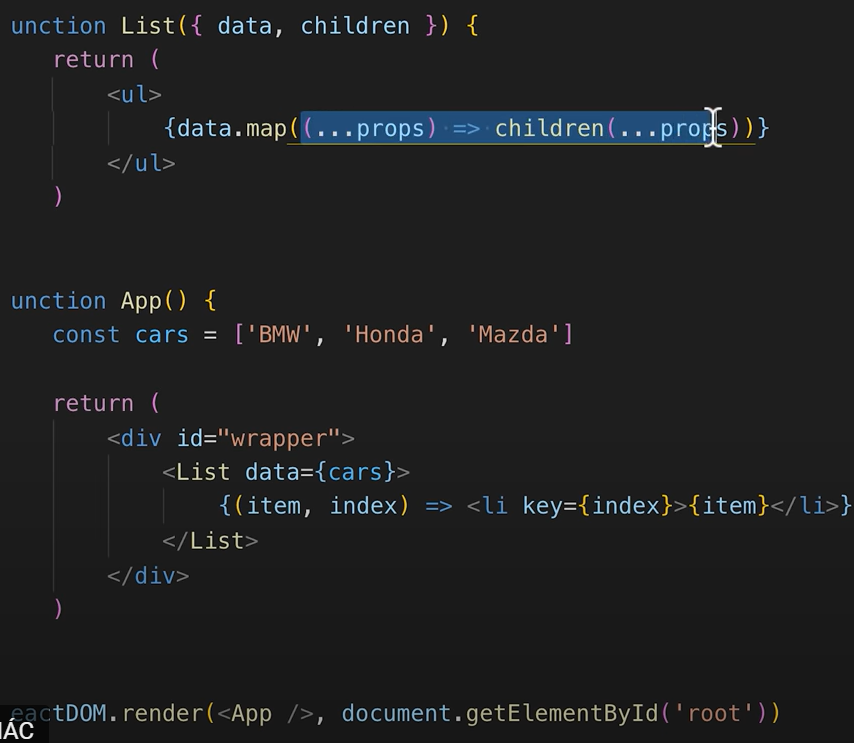
= 

React Element Type

<script *type*="text/babel">  
 *function* Header(){  
 *return* (  
 <div *className*="header">New Header</div>  
 )  
 }  
  
 *class* Content *extends* React.Component {  
 render(){  
 *return* (  
 <div *className*="content">New Content</div>  
 )  
 }  
 }  
  
 *const* app = (  
 <div *className*="wrapper">  
 <Header></Header>  
 <Content></Content>  
 <div *className*="footer">Footer</div>  
 </div>  
 )  
  
 ReactDOM.render(app, document.getElementById('app'))  
</script>

<script *type*="text/babel">  
  
 *const* Form = {  
 Input(){  
 *return* <input />  
 },  
 Checkbox(){  
 *return* <input *type*="checkbox"/>  
 }  
 }  
  
 *function* Button({title,href,onClick}){  
 *let* Component = 'button';  
 *const* props = {}  
  
 *if*(href){  
 Component = 'a';  
 props.href = href;  
 }  
 *if*(onClick){  
 props.onClick = onClick;  
 }  
 *return* (  
 <Component {...props}>{title}</Component>  
 )  
 }  
  
 *function* App(){  
 *const* type = 'Input';  
 *const* Component = Form[type];  
 *return* (  
 <React.Fragment>  
 <div *id*="wrapper">  
 <Component/>  
 </div>  
 <Button  
 *title*="Click me!"  
 *// href = "google.com"  
 onClick*={()=>console.log(Math.random())}  
 />  
 </React.Fragment>  
 )  
 }  
  
 ReactDOM.render(<App/>, document.getElementById('root'))

JSX #3



Hook

1. UseState

Two wide binding

import "./styles.css";

import React, { useState } from "react";

const gifts = ["iPhone 12", "iPhone 13", "iPhone 14"];

export default function App() {

const [gift, setGift] = useState();

const randomGift = () => {

const index = Math.floor(Math.random() \* gifts.length);

setGift(gifts[index]);

};

return (

<React.Fragment>

<p>{gift || "chưa có phần thưởng"}</p>

<button onClick={randomGift}>Chọn phần thưởng</button>

</React.Fragment>

);

}

Form

import "./styles.css";

import React, { useState } from "react";

const gifts = ["iPhone 12", "iPhone 13", "iPhone 14"];

export default function App() {

const [name, setName] = useState("");

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

const handleSubmit = () => {

console.log({

name,

email

});

};

return (

<React.Fragment>

<input

placeholder="name"

value={name}

onChange={(e) => setName(e.target.value)}

/>

<input

placeholder="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

/>

<button onClick={handleSubmit}>Register</button>

</React.Fragment>

);

}

Form radio

import "./styles.css";

import React, { useState } from "react";

const courses = [

{ id: 1, name: "PHP" },

{ id: 2, name: "JAVA" },

{ id: 3, name: "JAVASCRIPT" }

];

export default function App() {

const [checked, setChecked] = useState();

const handleSubmit = () => {

*// call api*

console.log({

id: checked

});

};

return (

<React.Fragment>

{courses.map((course) => (

<div key={course.id}>

<input

type="radio"

name="course"

checked={course.id === checked}

onChange={() => setChecked(course.id)}

/>

{course.name}

</div>

))}

<button onClick={handleSubmit}>Submit</button>

</React.Fragment>

);

}

Form checkbox

import "./styles.css";

import React, { useState } from "react";

const courses = [

{ id: 1, name: "PHP" },

{ id: 2, name: "JAVA" },

{ id: 3, name: "JAVASCRIPT" }

];

export default function App() {

const [checked, setChecked] = useState([]);

console.log(checked);

const handleCheck = (id) => {

setChecked((prev) => {

const isChecked = checked.includes(id);

if (isChecked) {

return checked.filter((item) => item !== id);

} else {

return [...prev, id];

}

});

};

const handleSubmit = () => {

*// call api*

console.log({ ids: checked });

};

return (

<React.Fragment>

{courses.map((course) => (

<div key={course.id}>

<input

type="checkbox"

name="course"

checked={checked.includes(course.id)}

onChange={() => handleCheck(course.id)}

/>

{course.name}

</div>

))}

<button onClick={handleSubmit}>Submit</button>

</React.Fragment>

);

}

Toto list

import "./styles.css";

import React, { useState } from "react";

export default function App() {

const storageJobs = JSON.parse(localStorage.getItem("jobs"));

const [job, setJob] = useState("");

const [jobs, setJobs] = useState(storageJobs ?? []);

const handleSubmit = () => {

setJobs((prev) => {

const newJobs = [...prev, job];

const jsonJob = JSON.stringify(newJobs);

localStorage.setItem("jobs", jsonJob);

return newJobs;

});

setJob("");

};

return (

<React.Fragment>

<div style={{ padding: 32 }}>

<input value={job} onChange={(e) => setJob(e.target.value)} />

<button onClick={handleSubmit}>Add</button>

<ul>

{jobs.map((job, index) => (

<li key={index}>{job}</li>

))}

</ul>

</div>

</React.Fragment>

);

}

1. useEffect

* UseEffect(callback): gọi callback mỗi khi re-render, 1 element đc thêm vào DOM
* UseEffect(callback, []): chỉ gọi callback 1 lần khi component mounted
* UseEffect(callback, [depes]): gọi lại mỗi khi depensentive thay đổi
* Update DOM

import { useEffect, useState } from 'react'

*//1. update DOM*

function Content() {

const [title, setTitle] = useState('')

useEffect(() => {

document.title = title

})

return (

<div>

<input

value = {title}

onChange = {e => setTitle(e.target.value)}

/>

</div>

)

}

export default Content

* Call API

import { useEffect, useState } from 'react'

*//2. Call API*

function Content() {

const [title, setTitle] = useState('')

const [posts, setPosts] = useState([])

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/posts')

.then(res => res.json())

.then(posts => {

setPosts(posts)

})

},[]) // thêm mãng rỗng depensentive

return (

<div>

<input

value = {title}

onChange = {e => setTitle(e.target.value)}

/>

<ul>

{posts.map(post => (

<li key={post.id}>{post.title}</li>

))}

</ul>

</div>

)

}

export default Content

CALL API

import { useEffect, useState } from "react";

const tabs = ["posts", "comments", "albums", "todos", "photos", "users"];

function Content() {

const [title, setTitle] = useState();

const [posts, setPosts] = useState([]);

const [type, setType] = useState("posts");

console.log(type);

useEffect(() => {

fetch(`https://jsonplaceholder.typicode.com/${type}`)

.then((res) => res.json())

.then((posts) => {

setPosts(posts);

console.log(posts);

});

}, [type]);

return (

<div>

{tabs.map((tab) => (

<button

key={tab}

style={type === tab ? { color: "#fff", backgroundColor: "#333" } : {}}

onClick={() => setType(tab)}

>

{tab}

</button>

))}

<ul>

{posts.map((post) => (

<li key={post.id}>{post.title || post.name}</li>

))}

</ul>

</div>

);

}

export default Content;

SCROLL

import "./styles.css";

import { useState } from "react";

import Content from "./Content";

export default function App() {

const [show, setShow] = useState(false);

return (

<div style={{ margin: "2rem" }}>

<button onClick={() => setShow(!show)}>{show ? "Hide" : "Show"}</button>

{show && <Content />}

</div>

);

}

import { useEffect, useState } from "react";

const tabs = ["posts", "comments", "albums", "photos", "todos", "users"];

function Content() {

const [title, setTitle] = useState("");

const [contents, setContents] = useState([]);

const [type, setType] = useState("posts");

const [buttonToTop, setButtonToTop] = useState(false);

const [width, setWidth] = useState(window.innerWidth);

const [countdown, setCountdown] = useState(5);

useEffect(() => {

fetch(`https://jsonplaceholder.typicode.com/${type}`)

.then((res) => res.json())

.then((contents) => {

setContents(contents);

});

}, [type]);

useEffect(() => {

const handleScroll = () => {

if (window.scrollY >= 300) {

setButtonToTop(true);

} else {

setButtonToTop(false);

}

*// setButtonToTop(window.scrollY >= 300)*

};

window.addEventListener("scroll", handleScroll);

*//Cleanup function*

return () => {

window.removeEventListener("scroll", handleScroll);

};

}, []);

useEffect(() => {

const handleResize = () => {

setWidth(window.innerWidth);

};

window.addEventListener("resize", handleResize);

return () => {

window.removeEventListener("resize", handleResize);

};

}, []);

return (

<div>

<div style={{ textAlign: "center" }}>Window width: {width}px</div>

<div style={{ textAlign: "center" }}>Countdown: {countdown}</div>

<input

value={title}

placeholder="Type something..."

onChange={(e) => setTitle(e.target.value)}

/>

<div style={{ textAlign: "center" }}>

{tabs.map((tab) => (

<button

key={tab}

style={

type === tab

? {

color: "#fff",

backgroundColor: "#333",

margin: "0 2px"

}

: { margin: "5px 2px" }

}

onClick={() => setType(tab)}

>

{tab.toUpperCase()}

</button>

))}

</div>

<ul>

Input value: {title || "..."}

{contents.map((content) => (

<li key={content.id}>{content.title || content.name}</li>

))}

</ul>

{buttonToTop && (

<button

onClick={() =>

window.scrollTo({

top: 0,

behavior: "smooth"

})

}

style={{

position: "fixed",

right: 20,

bottom: 20

}}

>

Go to top

</button>

)}

</div>

);

}

export default Content;

Countdown timer

import { useState, useEffect } from "react";

function Content() {

const [countdown, setCountdown] = useState(180);

console.log(countdown);

useEffect(() => {

let timerId = setInterval(() => {

setCountdown((prev) => prev - 1);

console.log("count...");

}, 1000);

*// cleanup function*

return () => clearInterval(timerId);

}, []);

return <h1>{countdown}</h1>;

}

export default Content;

Preview avatar

import { useEffect, useState } from "react";

function Content() {

const [avatar, setAvatar] = useState();

useEffect(() => {

*// cleanup*

return () => {

avatar && URL.revokeObjectURL(avatar.preview);

};

}, [avatar]);

const handlePreviewAvatar = (e) => {

const file = e.target.files[0];

file.preview = URL.createObjectURL(file);

setAvatar(file);

};

return (

<div>

<input type="file" onChange={handlePreviewAvatar} />

{avatar && <img src={avatar.preview} width="80%"></img>}

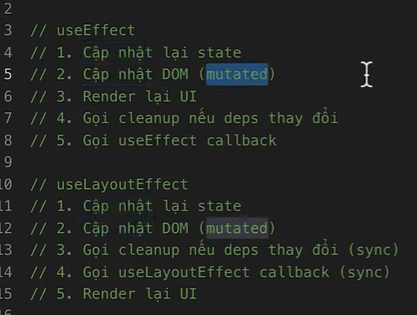
</div>

);

}

export default Content;

KHÁC NHAU CỦA USEEFECT VÀ USELAYOUT EFFECT



USE REF HOOCK: LƯU 1 GIÁ TRỊ QUA THAM CHIẾU BÊN NGOÀI

React.memo HOC: CHỉ định cho conponent chỉ rerender lại khi props thay đổi

import "./styles.css";

import { useState } from "react";

import Content from "./Content";

export default function App() {

const [count, setCount] = useState(0);

const incrase = () => {

setCount(count + 1);

};

return (

<div style={{ margin: "2rem" }}>

<Content />

<h1>{count}</h1>

<button onClick={incrase}>Click me</button>

</div>

);

}

import { memo } from "react";

function Content() {

console.log("rerender");

return <p>Hello anh em</p>;

}

export default memo(Content);

USE CALLBACK: dùng để tránh render lại cpn khi sữ dụng callback, bằng cách chỉ lấy giá trị tham chiếu của hàm đó

import "./styles.css";

import { useState, useCallback } from "react";

import Content from "./Content";

export default function App() {

const [count, setCount] = useState(0);

const handleIncrease = useCallback(() => {

setCount((prev) => prev + 1);

}, []);

return (

<div style={{ margin: "2rem" }}>

<Content onIncrease={handleIncrease} />

<h1>{count}</h1>

</div>

);

}

import { memo } from "react";

function Content({ onIncrease }) {

console.log("rerender");

return (

<>

<p>Hello anh em</p>

<button onClick={onIncrease}>Click me!</button>

</>

);

}

export default memo(Content);

USEMEMO: để tránh thực hiện lại các logic không cần thiết

import "./styles.css";

import { useMemo, useRef, useState } from "react";

export default function App() {

const [name, setName] = useState("");

const [price, setPrice] = useState("");

const [products, setProcucts] = useState([]);

const nameRef = useRef();

const handleSubmit = () => {

setProcucts([...products, { name, price: +price }]);

setName("");

setPrice("");

nameRef.current.focus();

};

const total = useMemo(() => {

const result = products.reduce((result, prod) => {

console.log("tinh laij");

return result + prod.price;

}, 0);

return result;

}, [products]);

return (

<div style={{ margin: "2rem" }}>

<input

ref={nameRef}

value={name}

placeholder="enter name..."

onChange={(e) => setName(e.target.value)}

/>

<br />

<input

value={price}

placeholder="enter price..."

onChange={(e) => setPrice(e.target.value)}

/>

<br />

<button onClick={handleSubmit}>Add</button>

<br />

Total: {total}

<ul>

{products.map((product, index) => (

<li key={index}>

{product.name} - {product.price}

</li>

))}

</ul>

</div>

);

}

useReduce: Dùng tương tự state nhưng cho những trường hợp phức tạp

import { useState, useReducer } from "react";

*// useState*

*// 1. init state = 0*

*// 2. action: up,down*

*// useReducer*

*// 1. init state: 0*

*// 2. action*

*// 3. reduce*

*// 4. dispatch*

*// init state*

const initState = 0;

*// action*

const UP\_ACTION = "up";

const DOWN\_ACTION = "down";

*//reduce*

const reducer = (state, action) => {

switch (action) {

case UP\_ACTION:

return state + 1;

case DOWN\_ACTION:

return state - 1;

default:

throw Error("invalid action");

}

};

export default function App() {

const [count, dispatch] = useReducer(reducer, initState);

return (

<div style={{ margin: "2rem" }}>

<h1>{count}</h1>

<button onClick={() => dispatch(DOWN\_ACTION)}>Down</button>

<button onClick={() => dispatch(UP\_ACTION)}>Up</button>

</div>

);

}

TOTO sữ dụng reducer

import { useState, useReducer, useRef } from "react";

*// 1. init state:*

const initState = {

job: "",

jobs: []

};

*// 2. action*

const SET\_JOB = "set\_job";

const ADD\_JOB = "add\_job";

const DELETE\_JOB = "delete\_job";

const setJob = (payload) => {

return {

type: SET\_JOB,

payload

};

};

const addJob = (payload) => {

return {

type: ADD\_JOB,

payload

};

};

const deleteJob = (id) => {

console.log(id);

return {

type: DELETE\_JOB,

id

};

};

*// 3. reduce*

const reducer = (state, action) => {

console.log("Action: ", action);

console.log("Prev state: ", state);

let newState;

switch (action.type) {

case SET\_JOB:

newState = { ...state, job: action.payload };

break;

case ADD\_JOB:

newState = {

...state,

jobs: [...state.jobs, action.payload]

};

break;

case DELETE\_JOB:

const newJobs = [...state.jobs];

newJobs.splice(action.id, 1);

newState = {

...state,

jobs: newJobs

};

break;

default:

throw new Error("invalid action");

}

console.log("new state: ", newState);

return newState;

};

export default function App() {

const [state, dispatch] = useReducer(reducer, initState);

const { job, jobs } = state;

const inputRef = useRef();

const handleSubmit = () => {

dispatch(addJob(job));

dispatch(setJob(""));

inputRef.current.focus();

};

return (

<div style={{ margin: "2rem" }}>

<h3>Todo</h3>

<input

ref={inputRef}

value={job}

placeholder="enter todo"

onChange={(e) => dispatch(setJob(e.target.value))}

/>

<button onClick={handleSubmit}>Add</button>

<ul>

{jobs.map((job, index) => (

<li key={index}>

{job}

<span onClick={() => dispatch(deleteJob(index))}>&times;</span>

</li>

))}

</ul>

</div>

);

}