|  |  |
| --- | --- |
| 1. Khái niệm ReactJs | - Là thư viện Javascript, tạo ra các thành phần giao diện người dùng có tính tái sử dụng cao  - Được phát triển bởi Fb ,thường dc sd để phát triển ứng dụng single-page  - ReactJS trừu tượng hóa DOM, cung cấp một mô hình lập trình đơn giản hơn và hiệu suất tốt hơn |
| 2. Ưu nhược điểm của ReactJS | \* Ưu điểm  - Có thể sử dụng ReactJS để viết ứng dụng trực tiếp bằng JavaScript  - Biến các thành phần UI phức tạp trở thành những component độc lập.  - Chuyển các dữ liệu đã được tùy biến đến một UI component cụ thể  - Có thể thay đổi trạng thái cho nhiều component con và không gây ảnh hưởng đến component gốc  - Khả năng xác định chính xác khi nào cần render lại cũng như khi nào bỏ đi phần tử DOM  \* Nhược điểm  - Reactjs chỉ là thư viện giúp render ra phần view. React không phải là một MVC framework, vì thế nó không có phần Model và Controller, mà phải kết hợp với các thư viện khác. React cũng sẽ không có 2-way binding hay là Ajax  - Khi muốn tích hợp Reactjs vào các framework MVC truyền thống yêu cầu cần phải cấu hình lại  - React khá nặng nếu so với các framework khác React có kích thước tương tương với Angular. Trong khi đó Angular là một framework hoàn chỉnh  - Khó tiếp cận cho người mới học Web. |
| 3.SSR (Server side render) | Nội dung HTML đc xử lý trên server và trả lại cho browser nội dung HTML đã hoàn chỉnh và đc hiển thị ngay khi nó trả về client  \* Ưu điểm  - Thời gian tải trang ban đầu nhanh hơn so với CSR – Điều này mang lại trải nghiệm tốt cho người dùng.  - SEO tốt hơn – Do thời gian tải ban đầu nhanh hơn, các bot của công cụ tìm kiếm có thể thu thập thông tin và lập chỉ mục các trang tốt hơn.  - Tối ưu cho người dùng có kết nối internet chậm – Họ có thể xem trước HTML trong khi JavaScript đang xử lý.  \* Nhược điểm  - Trang web phải xử lý lại hoàn toàn và load lại từ đầu nếu chỉ có một thay đổi nhỏ trong nội dung. (Ví dụ tiêu đề thay đổi …)  - Việc xử lý nội dung HTML khiến hao tốn tài nguyên server, gây chậm trễ khi xử lý các request khác.  - Lượng request lên server rất nhiều, do mọi tác vụ đều phải xử lý lại trên server và render lại HTML. |
| 4.CSR(Client Side Rendering) | Server chỉ có nhiện vụ trả về mã nguồn JS+data thô cho client.Việc sinh nội dung đều thực hiện ngay trên client.(Code js có nv render giao diện HTML)  \* Ưu điểm  - Thời gian phản hồi nhanh.  - Ít yêu cầu HTTP hơn đối với máy chủ.  - Mã máy chủ và máy khách hoạt động độc lập với nhau giúp người dùng có thể tạo nhiều ứng dụng trên nhiều nền tảng.  \* Nhược điểm  - Lượng dữ liệu lần đầu load về khá nặng  - Nội dung hiển thị ra lần đầu chậm hơn so với SSR, tuy nhiên những lần sau thì quá nhanh do việc xử lý ngay trên client.  - SEO bị ảnh hưởng, do nội dung web được sinh trên client, khiến crawler của của Search engine không tiếp xúc được nội dung. Gần đây mới có Google giải quyết được vấn đề này, tuy nhiên dù Google nắm phần lớn thị phần tìm kiếm thì tại những quốc gia khác nhau, sẽ có ảnh hưởng nhất định. |
| 5.React element | - React element chỉ là một đối tượng JavaScript cũ đơn thuần mà không có các phương thức riêng, là những gì được trả về từ các components.  - Nó mô tả những gì chúng ta muốn thấy trên màn hình, nó là thành phần đại diện của DOM  - Là 1 phần tử giúp xây dựng UI trong React  \* React.createElement có cú pháp sau:  **React.createElement(type, [props], [...children])**  + type có thể là một thẻ HTML như h1, div hoặc nó có thể là React component  + prop là các thuộc tính bạn muốn phần tử có children chứa các thẻ HTML khác hoặc có thể là component |
| 6. Rendering Elements trong DOM | Để render một phần tử React vào root DOM, thì ta dùng phương thức **ReactDOM.render()**  **Cú pháp :**  **ReactDOM.render(element, container[, callback])**  - Phương thức sẽ render phần tử React vào DOM trong vùng chứa được cung cấp và trả về một tham chiếu đến thành phần (hoặc trả về null cho các thành phần không trạng thái).  - Nếu phần tử React trước đó đã được hiển thị vào vùng chứa, thì phần tử này sẽ thực hiện cập nhật nó và chỉ thay đổi DOM khi cần thiết để phản ánh phần tử React mới nhất.  - Nếu lệnh gọi lại tùy chọn được cung cấp, nó sẽ được thực thi sau khi thành phần được hiển thị hoặc cập nhật. |
| 7.JSX(JavaScript XML) | - Là ngôn ngữ kết hợp giữa JS và các thẻ giống HTML để xác định các phần tử giao diện người dùng và chức năng của chúng  - Nó là một phần mở rộng cho JavaScript cho phép viết những gì trông giống như HTML và XML trong JavaScript.  - JSX trông giống như HTML và cung cấp một cú pháp thân thiện để tạo các phần tử DOM.  - JSX chấp nhận bất kỳ biểu thức JavaScript hợp lệ nào bên trong dấu ngoặc nhọn. |
| 8. Vì sao lại sử dụng JSX? Browsers có thể đọc được JSX file không? Tại sao? | - Thứ nhất, JSX với cú pháp gần giống XML, cấu trúc cây khi biểu thị các attributes, điều đó giúp ta dễ dàng định nghĩa, quản lý được các component phức tạp, thay vì việc phải định nghĩa và gọi ra nhiều hàm hoặc object trong javascript. Khi nhìn vào cấu trúc đó cũng dễ dàng đọc hiểu được ý nghĩa của các component. Code JSX ngắn hơn, dễ hiểu hơn code JS.  - Thứ 2, JSX không làm thay đổi ngữ nghĩa của Javascript  - Thứ 3, với cách viết gần với các thẻ HTML, nó giúp những developers thông thường (ví dụ như các designer) có thể hiểu được một cách dễ dàng, từ đó có thể viết hoặc sửa code mà không gặp nhiều khó khăn.  \* Browsers có thể đọc được JSX file không? Tại sao?  - Browers không thể trực tiếp đọc JSX mà phải đọc thông qua trình biên dịch vì chúng chỉ có thể hiểu các đối tượng JavaScript và JSX không phải là một đối tượng JavaScript thông thường. Do đó, chúng ta cần chuyển đổi tập tin JSX thành một đối tượng JavaScript bằng cách sử dụng các bộ chuyển đổi (transpilers) như Babel và sau đó chuyển nó vào trình duyệt |
| 9. React Component là gì | - Trong React, các component hoạt động giống như các hàm trả về các thành phần HTML  - Các component là các thành phần độc lập và có thể sử dụng lại.  - Mỗi component có code JS,HTML và CSS riêng, chúng có thể tái sử dụng, dễ đọc, dễ viết, dễ test.  - Có 2 loại component :  + Function Component (Angular k có)   * Một function Javascript / ES6 function * Phải trả về 1 React element * Nhận props làm tham số nếu cần   + Class Component.   * Là một class ES6, nó sẽ là một component khi nó "kế thừa" React component. * Có thể nhận props (trong hàm khởi tạo) nếu cần. * Có thể maintain data của nó với state * Phải có 1 method render() trả về 1 React element (JSX), hoặc null |
| 10.Props | - Props là 1 từ viết ngắn gọn của properties, và là 1 khái niệm trong ReactJS. Nó trả về 1 đối tượng. Nó lưu trữ các giá trị của các attribute (thuộc tính) của một thẻ (Tag). Props cung cấp cách thức để chúng ta truyền dữ liệu từ component này tới component khác. |
| 11.Pure Function  (Hàm thuần khiết) | - Hàm luôn trả về cùng một kết quả nếu được truyền vào các tham số không đổi. Nó không hề phụ thuộc vào bất kỳ trạng thái hoặc dữ liệu nào.  - Hàm này không tạo ra bất kỳ ảnh hưởng nào đến các đối tượng khác |
| 12. Vì sao nên sd Pure Function | - cải thiện chất lượng code của bạn  - giúp code clean hơn  - hạn chế bug đáng kể.  - dễ kiểm tra  - k gây side effect |
| 13. Observable side effects là gì? | - Observable side effect là bất kỳ sự tương tác nào với bên ngoài từ bên trong hàm ví dụ như việc thay đổi giá trị biến ở bên ngoài hàm hay gọi hàm khác từ bên trong hàm (Nhưng có thể gọi pure function).  *Chú ý:* Nếu một pure function gọi một pure function thì đây không phải là side effect và hàm gọi vẫn là pure function |
| 14.State | - State là nơi bạn lưu trữ các giá trị thuộc tính thuộc về component từ đó bạn có thể luân chuyển dữ liệu đến các thành phần trong Component hoặc các Component khác.  - Khi state thay đổi, thì components sẽ render lại (Dùng setState())  - Trong các dự án React, state được dùng để phản hồi các yêu cầu từ người dùng, hay lưu trữ một dữ liệu nào đó trong components.  \* React cung cấp phương thức setState() để cập nhật state. Sau khi cập nhật xong, phương thức này ngầm gọi phương thức render() để render lại trang  \* Một số lưu ý khi sử dụng State   * Bạn nên để cấu trúc dữ liệu của state đơn giản nhất có thể, không nên tạo cấu trúc quá lằng nhằng sẽ khó thao tác và ảnh hưởng hiệu năng * Không thay đổi state một cách trực tiếp * State được update không đồng bộ nên bạn cần lưu ý khi sử dụng hàm setState cần giá trị từ state trước |
| 15. So sánh State và Props | - Props là các biến, data được truyền từ component cha và có thể truy cập được ở cả các component con.  - State được khởi tạo và chỉnh sửa và chỉ có thể truy bởi chính bản thân component chứa nó.  - Điểm khác nhau lớn nhất giữa props và state đó là props không thể thay đổi, còn state có thể thay đổi do đó hiệu năng của props tốt hơn state. |
| 16. Component Life Cycle trong ReactJS | - Life cycle của component trong reactjs là quá trình từ khi tạo ra, thay đổi và hủy bỏ component. Gồm 3 giai đoạn: Initialization, Mounting, Updating và UnMounting  + **Initialization**: Đây là giai đoạn mà thành phần được xây dựng với các Props đã cho và trạng thái mặc định. Điều này được thực hiện trong phương thức khởi tạo Component Class  + **Mounting**: Giai đoạn này được thực hiện sau khi quá trình initialization(khởi tạo) được hoàn thành. Nó thực hiện nhiệm vụ chuyển virtual DOM (DOM ảo) trong React thành DOM và hiển thị trên trình duyệt.   * **componentWillMount():** hàm này được gọi ngay trước khi thành phần được gắn trên DOM, tức là hàm này được gọi một lần trước khi hàm render () được thực thi lần đầu tiên. * **componentDidMount():** hàm này được gọi ngay sau khi thành phần được gắn trên DOM, tức là hàm này được gọi một lần sau khi hàm render () được thực thi lần đầu tiên   + **Updating**: là giai đoạn trạng thái của một thành phần được cập nhật và ứng dụng được render lại.   * **componentWillRecieveProps():** Hàm này được gọi trước khi một thành phần bắt đầu truyền props * **shouldComponentUpdate:** trả về giá trị true hoặc false. Điều này sẽ xác định xem thành phần sẽ được cập nhật hay không. Giá trị mặc định là True * **componentWillUpdate :** được gọi ngay trước khi render * **componentDidUpdate** : được gọi sau khi render   + **Unmounting**: là bước cuối cùng của vòng đời thành phần, nơi thành phần được xóa khỏi trang.   * **componentWillUnmount** được gọi sau khi component được ngắt kết nối khỏi dom. |
| 17. Hooks là gì? | - Hooks là 1 tính năng mới trong React v16.8 , đây là những hàm  đặt biệt đc React tạo sẵn cho phép kết nối React state với lifecycle vào function component. Với Hooks bạn có thể sử dụng state và lifecycles mà không cần dùng ES6 Class  - Lợi ích  + Khiến các component trở nên gọn hơn  + Dễ tiếp cận hơn  + Cho phép sd state ngay trong function component.  - Một sô Hooks hay dùng  +useState,useEffect  +useLayoutEffect,useContext,useReduce,useMemo,useCallback,.. |
| 18. Một số lưu ý khi sd Hook? | - Trong cùng một component, bạn có thể sử dụng bao nhiêu useState và useEffect tùy ý nhưng các hook này phải gọi ở trên cùng của function, không được nằm trong vòng lặp, khu vực điều kiện, hay các function con  - Nó chỉ sử dụng trong functional component  - Khi sử dụng useEffect để lấy dữ liệu, cần kiểm tra dữ liệu đã tồn tại hay chưa. Nếu không thì hàm sẽ gửi request liên tục |
| 19. useState | - Là một hook cơ bản của reactjs version > 16.8.  - Cho phép lưu trữ state với React function Component  - Cú pháp: const [state, setState] = useState(initvalue);  + state: định nghĩa tên của state nó có thể là đơn giá trị hoặc object,.. (là tham số của useState)  + setState: định nghĩa tên function dùng cho việc update state (là tham số của useState)  + initialStateValue: là giá trị ban đầu của state. |
| 20.useEffect | - UseEffect cho phép chúng ta xử lý các logic trong liftcycle component và những sideEffect  - useEffect Hook là của 3 phương thức componentDidMount, componentDidUpdate, và componentWillUnmount kết hợp lại với nhau.  - Các TH sử dụng:  + useEffect(callback) Chạy Mounting(sau return) và updating  + useEffect(callback,[])Chạy Mounting 1 lần  useEffect(callback,[deps])Chạy Mounting và updating( chỉ chạy State(deps)) |
| 21. React Form | -Trong ReactJS để tạo form thì có thể tái sd lại các thành phần của form HTML như <input/>,<form>…Kết hợp với các sk như onChange(),onSubmit(),…Để xử lý data trên form  -Validate thì sd các hàm của TS.  -Ngoài ra, có thể sd các thư viện xử lý form và vadate form như:Formik,Yup,react-form-hooks |
| 22.React Router  *Cách triển khai?* | -Là thư viện đc xây dựng trên react, giúp cho LTV xây dụng luồng đi định tuyến cho ứng dụng SPA  -Routers đc xây dựng để định nghĩa nhiều định tuyến khác nhau .Khi người dùng truy cập vào URL nhất định, nếu URL đó dc tìm thấy trong các URL đc định nghĩa bên trong React, thì người dùng sẽ dc dịch chuyển đến page tương ứng |
| 23.Cách sd promise để xử lý bất đồng bộ | - Promise sinh ra để xử lý kết quả của một hành động cụ thể, kết quả của mỗi hành động sẽ là thành công hoặc thất bại và Promise sẽ giúp chúng ta giải quyết câu hỏi “Nếu thành công thì làm gì? Nếu thất bại thì làm gì?”  - Khi một Promise được khởi tạo thì nó có một trong ba trạng thái sau:  + Fulfilled: Hành động xử lý xong và thành công+  + Rejected: Hành động xử lý xong và thất bại  + Pending: Hành động đang chờ xử lý hoặc bị từ chối  Trong đó, hai trạng thái Rejected và Fulfilled ta gọi là Settled, tức là đã xử lý xong. |
| 24. Axios | * Là thư viện HTTP Client dựa trên Promise.Có bản thì nó cung cấp 1 API cho việc hanlde XHR(XMLHttpRequest)   + Tạo XMLHttpRequest từ trình duyệt  + Thực hiện Http Request từ nodejs  + Hỗ trợ promise API  + Chuyển đổi request response  + Hanlde tiền kỳ,hậu kỳ request  + Hủy Request |
| 25. Async/Await  *Cách xử lý bất đồng bộ* | * Là thực hiện các thao tác BDB 1 cách tuần tự * Async/await vẫn sd promise nhưng mã nguồn trong sáng và dễ theo dõi hơn * Async dùng để khai báo 1 hàm BDB   + Tự động biến đổi hàm thông thường thành hàm promise  + Khi gọi tới hàm async nó sẽ handle mọi thứ và trả về kết quả trong hàm của nó   * Await sẽ dc dùng trước thao tác BDB (callAPI),tạm dừng việc thực hiện async   + Khi đặt await trước 1 promise, thì nó sẽ đợi cho đến khi promise kết thúc trả về kết quả  + Await chỉ hoạt động với promise, không hoạt động callback,…  + Await chỉ sd bên trong function async |
| 26. Redux là gì? Tại sao phải dùng Redux?  *Cách triển khai Redux?*  Các nguyên tắc chính của redux? | -Redux là thư viện của JS để quản lý các state trong ứng dụng  -Cung cấp 1 nơi lưu trữ của ứng dụng và đảm bảo state luôn đc cập nhật theo cách có thể dự đoán đc. Nơi lưu trữ gọi là store   * Ưu điểm   + Tăng hiệu suất  + Khả năng bảo trì, kiểm thử  + Có thể dự đoán trạng thái state   * Nhược điểm   + Phức tạp hóa kiến trúc của dự án   * 3 nguyên tắc của Relux:   + Single source of truth: State của toàn bộ ứng dụng được lưu trong trong 1 store duy nhất là 1 Object mô hình cây.  + State is read-only: Chỉ có 1 cách duy nhất để thay đổi state đó là tạo ra một action (là 1 object mô tả những gì xảy ra)  + Changes are made with pure functions: Để chỉ rõ state tree được thay đổi bởi 1 action cần phải viết reducers (pure functions) |
| 27.Redux Actions | - Action là các sự kiện và nó cũng là 1 object tong JS  - Action đc gửi đến store thông qua useDispatch()  - Action thông thường sẽ có 2 thuộc tính  + Type: Chỉ ra hành động sẽ thực hiện  + Payload: Chứa thông tin của data cần gửi đi |
| 28.Redux Reducer | - Là 1 hàm pure function lấy trạng thái của ứng dụng, sau đó hành động và trả về 1 state mới  - Các nguyên tắc  + Chỉ đc tính toán state dựa trên state cũ + action và k dc chỉnh sửa state hiện tại  + K xử lý các logic BDB, tính toán ngẫu nhiên |
| 29.Redux Store | - Là object bất biến trong Redux. Dùng để lưu trữ state của ứng dụng |
| 30.Giới thiệu NestJS | - NextJS là 1 React Framework cung cấp các thành phần cầu thiếu để tạo ra ứng dụng web nhanh chóng, tính năng nổi bật là SSR và tạo các trang web tĩnh SSG.  - Tính năng  + Tối ưu hóa hình ảnh: tối ưu hóa hình ảnh, phông chữ và tập lệnh tự động để cải thiện UX và Core Web Vitals  + Truyền phát HTML động: truyền phát giao diện người dùng ngay lập tức từ máy chủ, được tích hợp với bộ định tuyến ứng dụng react |
| 31. Kn SEO (Search Engine Optimization) | - Là quá trình tăng chất lượng và lưu lượng truy cập website bằng cách tối ưu truy xuất HTML, cấu trúc website thân thiện với các công cụ tìm kiếm dữ liệu như google,bing,yahoo,..  - Các loại hình SEO  + SEO từ khóa  + SEO hình ảnh  + SEO ucleo social |
| 32. ES6 là gì? (ECMAScript6) | - Là phiên bản tiếp theo của nn lập trình JS. Phát hành vào năm 2015, cung cấp 1 số tính năng mới, bao gồm: cải thiện cú pháp khai báo biến, class… và toán tử mới |
| *33. Arrow function và function?* | Đặc điểm của arrow function:   * Không có từ khoá "function" và không có "this" riêng. * Không có khối lệnh nội tại (block scope), chỉ có thể trả về một giá trị duy nhất một cách ngắn gọn. * Tự động ràng buộc ngữ cảnh (context) của "this" với ngữ cảnh của hàm cha gần nhất.   Đặc điểm của function:   * Có từ khoá "function" và có thể có từ khoá "this" riêng. * Có thể có nhiều khối lệnh trong một hàm. * Ngữ cảnh (context) của "this" phụ thuộc vào cách hàm được gọi (global scope, object context, event handler, etc.). |
| *34. Phân biệt let, var và const* |  |
| *35. Trình bày cách ReactJS tạo form, xử lý form và triển khai validation cho form?* | Để tạo, xử lý và triển khai validation cho form trong ReactJS, bạn có thể làm theo các bước sau:   1. **Tạo Form Component**: Đầu tiên, tạo một component để đại diện cho form. Component này sẽ bao gồm các trường nhập liệu và các thành phần khác cần thiết cho form. 2. **Xử lý Thay đổi giá trị**: Sử dụng useState hook để lưu trữ trạng thái của form. Khi giá trị của một trường nhập liệu thay đổi, hàm handleChange sẽ được gọi để cập nhật trạng thái của form. 3. **Xử lý Submit**: Khi người dùng nhấn nút Submit, hàm handleSubmit sẽ được gọi. Trong hàm này, bạn có thể thực hiện các xử lý logic với dữ liệu form, ví dụ: gửi dữ liệu lên máy chủ hoặc thực hiện các thao tác xử lý khác. 4. **Triển khai Validation**: Để triển khai validation cho form, bạn có thể sử dụng các thư viện validation có sẵn như yup, joi, validator, hoặc tự thực hiện validation bằng cách kiểm tra các điều kiện trong hàm handleSubmit. Ví dụ, bạn có thể kiểm tra xem các trường như name, email, password có được điền đầy đủ và đáp ứng các yêu cầu cụ thể hay không. Nếu có lỗi validation, bạn có thể hiển thị thông báo lỗi tương ứng cho người dùng.   Đó là một cách cơ bản để tạo, xử lý và triển khai validation cho form trong ReactJS. Tuy nhiên, quá trình xử lý form có thể phức tạp hơn tùy thuộc vào yêu cầu cụ thể của ứng dụng. Bạn có thể tìm hiểu thêm về các thư viện validation và các kỹ thuật xử lý form nâng cao khác để phù hợp với nhu cầu của bạn. |
| *36. Khi nào thì nên sử dụng class component thay vì sử dụng function component?* |  |
| 37. Trình bày cách hoạt động của Virtual DOM | Bất cứ khi nào dữ liệu cơ bản thay đổi, toàn bộ giao diện người dùng sẽ được hiển thị lại (re-rendered) trong DOM ảo.  Sau đó, sự khác biệt giữa bản đại diện DOM trước đó và bản mới được tính toán.Sau khi tính toán xong, DOM thực sẽ chỉ được cập nhật với những thứ đã thực sự thay đổi. |
| *38. Key prop là gì? và lợi ích của việc sử dụng nó trong mảng các DOM elements?* | Trong React, thuộc tính "key" là một thuộc tính đặc biệt được sử dụng để xác định một cách duy nhất các thành phần con trong một mảng các thành phần (DOM elements) được tạo ra bởi một lệnh map hoặc lặp. Mỗi thành phần con trong mảng cần có một key duy nhất để React có thể nhận biết và quản lý chúng một cách hiệu quả.  Lợi ích của việc sử dụng key prop trong mảng các DOM elements là:   1. **Tối ưu hóa hiệu suất**: Key prop giúp React xác định các sự thay đổi trong danh sách các thành phần con một cách hiệu quả. Khi một thành phần có key thay đổi hoặc được thêm hoặc xóa, React chỉ cần cập nhật thành phần đó mà không cần render lại toàn bộ danh sách. Điều này giúp tăng hiệu suất và tối ưu hóa quá trình render. 2. **Đảm bảo sự duy nhất và nhất quán**: Key prop đảm bảo rằng mỗi thành phần con trong danh sách là duy nhất. Nếu không có key prop, React sẽ cảnh báo với một cảnh báo cụ thể để thông báo về việc thiếu key. Key prop cũng giúp duy trì sự nhất quán giữa phần tử trong danh sách và trạng thái ứng dụng. 3. **Giúp định danh và theo dõi các thành phần**: Key prop cung cấp một cách định danh duy nhất cho mỗi thành phần con trong danh sách. Điều này rất hữu ích khi bạn cần theo dõi và xử lý các sự kiện hoặc thay đổi trạng thái của một thành phần cụ thể trong danh sách. 4. **Hỗ trợ trong quá trình hoàn thiện (reconciliation)**: Key prop là một phần quan trọng của quá trình hoàn thiện trong React. Quá trình hoàn thiện là quá trình so sánh các thành phần hiện tại với các thành phần trước đó và xác định xem các thành phần cần được cập nhật, thay đổi hoặc xóa. Key prop giúp React xác định các sự thay đổi một cách chính xác và tự động.   Để sử dụng key prop, bạn chỉ cần gán một giá trị duy nhất cho key cho mỗi thành phần con trong danh sách. Các giá trị key thường được chọn là các giá trị duy nhất như ID hoặc bất kỳ giá trị nào có thể định danh một cách duy nhất các thành phần. |
| 39.Các cách xử lý bất đồng bộ? Ưu nhược điểm | Có 3 cách để xử lý BĐB  - [Callback: Callback là một hàm được truyền như một tham số vào một hàm khác](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1). [Khi hàm chứa callback hoàn thành, callback sẽ được gọi (hay “callback”)](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  + [Ưu điểm: Đây là mô hình khá phổ biến nên rất dễ hiểu và dễ triển khai trong các hàm của chúng ta](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  + Nhược điểm: Khi thao tác bất đồng bộ, các callback phải chờ nhau thực hiện dẫn đến tổng thời gian hoàn thành tác vụ lâu hơn. [Code có thể trở nên dài dòng, khó đọc và khó bảo trì](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  - [Promise: Promise là một đối tượng biểu diễn cho giá trị có thể không được xác định ngay và có thể được xác định trong tương lai](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  + Ưu điểm: Promise giúp viết code bất đồng bộ một cách rõ ràng hơn, giúp tránh “callback hell”. [Nó cũng cho phép bạn xử lý lỗi một cách dễ dàng hơn so với callback](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  + [Nhược điểm: Cú pháp của Promise có thể khá phức tạp và khó hiểu cho những người mới học](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)  - [Async/Await: Async/Await là cú pháp mới được giới thiệu trong ES7 để viết code bất đồng bộ một cách sạch sẽ và dễ hiểu](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)[7](https://codelearn.io/sharing/asyncawait-trong-javascript).  + [Ưu điểm: Async/Await giúp code trở nên ngắn gọn và sạch sẽ hơn](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1). [Nó cũng giúp xử lý lỗi cả trong code đồng bộ lẫn bất đồng bộ theo cùng một cấu trúc](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1).  + Nhược điểm: Do Async/Await hoạt động theo cơ chế chờ (await), nên có thể sinh ra trạng thái chờ khi các câu lệnh phải chờ nhau. [Điều này có thể làm tăng thời gian xử lý và ảnh hưởng đến trải nghiệm người dùng](https://codelearn.io/sharing/bat-dong-bo-trong-javascript-phan-1)[2](https://blog.tomosia.com/tim-hieu-sync-va-async-trong-javascript/). |
| *40. Trình bày nguyên lý hoạt động của ReactJs* | ReactJS hoạt động dựa trên các nguyên lý cốt lõi sau:   1. **Cơ sở của Virtual DOM**: React sử dụng một khái niệm gọi là Virtual DOM. Virtual DOM là một bản sao của DOM thật sẽ được React tạo ra và duy trì. Khi trạng thái của ứng dụng React thay đổi, React sẽ tạo ra một cây Virtual DOM mới và so sánh nó với cây Virtual DOM cũ. Sự khác biệt giữa hai cây này được gọi là diffing. Sau đó, React sẽ chỉ cập nhật các thành phần có sự khác biệt trong DOM thật, thay vì render lại toàn bộ DOM. Điều này giúp tăng hiệu suất và tăng tốc quá trình render. 2. **Component-Based Architecture**: React xây dựng ứng dụng dựa trên mô hình kiến trúc Component-Based. Một component trong React là một đơn vị độc lập, có thể tái sử dụng và có chứa các phần tử UI (user interface) và logic tương ứng. Mỗi component có thể nhận vào các thuộc tính (props) và có thể có trạng thái (state) riêng. Component có khả năng nhúng các component con bên trong nó, tạo thành một cây component phức tạp. 3. **One-Way Data Flow**: React tuân thủ nguyên tắc dòng dữ liệu một chiều (one-way data flow). Dữ liệu trong React chỉ di chuyển một chiều từ component cha đến component con. Một component cha có thể truyền dữ liệu và các hàm xử lý sự kiện xuống cho các component con thông qua props. Khi dữ liệu thay đổi, React sẽ tự động cập nhật lại giao diện người dùng. 4. **Reactive Updates**: React sử dụng cơ chế reactive updates để xác định khi nào cần cập nhật giao diện. Khi một trạng thái (state) của một component thay đổi, React sẽ tự động kích hoạt quá trình cập nhật giao diện tương ứng. React sử dụng một thuật toán hiệu quả gọi là reconciliation để xác định các sự thay đổi cần được áp dụng vào giao diện. 5. **Declarative Syntax**: React sử dụng cú pháp khai báo (declarative syntax) để xác định giao diện người dùng. Thay vì tương tác trực tiếp với DOM, bạn chỉ cần mô tả giao diện dựa trên trạng thái hiện tại của ứng dụng. React sẽ tự động quản lý việc cập nhật DOM để phù hợp với trạng thái mới. |