**1. ES6 là gì?**

ES6 (ECMAScript 2015) là phiên bản thứ 6 của ECMAScript, tiêu chuẩn chính cho JavaScript. Nó giới thiệu nhiều tính năng mới giúp mã JavaScript dễ viết và bảo trì hơn.

Đây là một trong những bản cập nhật lớn nhất, giới thiệu nhiều tính năng giúp viết mã JavaScript ngắn gọn, dễ đọc và bảo trì hơn. ES6 đặt nền tảng cho các phiên bản sau này như ES7, ES8 và hỗ trợ phát triển các framework, thư viện như React, Angular.

Ngoài ra, ES6 được hỗ trợ bởi hầu hết các trình duyệt hiện đại và nếu cần tương thích với các trình duyệt cũ, ta có thể sử dụng công cụ như **Babel** để chuyển đổi mã ES6 về ES5.

**2. Tính năng mới của ES6?**

* **Arrow functions:** Viết hàm ngắn gọn hơn (()=>{}).
* **Template literals:** Sử dụng `` thay vì + để nối chuỗi.
* **Destructuring:** Trích xuất giá trị từ mảng/đối tượng.
* **Default parameters:** Gán giá trị mặc định cho tham số hàm.
* **Let & const:** Khai báo biến có phạm vi block.
* **Modules (import/export):** Quản lý và tái sử dụng mã.
* **Classes:** Định nghĩa lớp dễ dàng.
* **Promises:** Hỗ trợ bất đồng bộ.
* **Spread & Rest operators:** Dễ dàng xử lý mảng và đối tượng.

**3. Phân biệt map và forEach?**

* **map:** Trả về mảng mới sau khi áp dụng hàm callback cho từng phần tử.
*  Trả về một **mảng mới**, giữ nguyên mảng gốc.
*  Thích hợp khi cần tạo ra mảng sau khi xử lý các phần tử.
* **forEach:** Thực thi hàm callback trên từng phần tử, nhưng **không trả về** mảng mới.
* Không trả về kết quả, chỉ thực thi hàm callback trên từng phần tử.
* Thường dùng khi cần thao tác mà không cần mảng mới.

**4. Reducer, Filter, Find**

* **reduce:** Gộp giá trị từ mảng thành một giá trị duy nhất.
* **filter:** Lọc mảng theo điều kiện.
* **find:** Tìm phần tử đầu tiên khớp điều kiện.

**5. React là gì? Ưu và nhược điểm?**

* **React:** Thư viện JavaScript để xây dựng giao diện người dùng.
* **Ưu điểm:** Hiệu suất cao (DOM ảo), tái sử dụng component, cộng đồng lớn.
* **Nhược điểm:** Cần học thêm, phụ thuộc thư viện bên ngoài.

**6. SEO là gì?**

SEO (Search Engine Optimization) là quá trình tối ưu hóa website để tăng khả năng xuất hiện trên các công cụ tìm kiếm như Google..

**7. SSR và CSR? Ưu nhược điểm?**

* **SSR (Server-Side Rendering):** Render HTML từ server.
  + Ưu: SEO tốt hơn, tải trang đầu nhanh.
  + Nhược: Tải trang chậm khi chuyển trang.
* **CSR (Client-Side Rendering):** Render HTML từ trình duyệt.
  + Ưu: UI mượt mà, tải trang nhanh sau lần đầu.
  + Nhược: SEO kém, tải trang đầu chậm.

**8. DOM ảo?**

DOM ảo là một bản sao DOM trong bộ nhớ, giúp React cập nhật UI nhanh hơn bằng cách so sánh và thay đổi chỉ những phần cần thiết. DOM ảo là một **bản sao của DOM thật** được lưu trữ trong bộ nhớ. Nó không thực sự thay đổi giao diện ngay lập tức mà hoạt động như một lớp trung gian giữa logic ứng dụng và giao diện người dùng.

**9. JSX là gì? Vì sao nên dùng?**

JSX là cú pháp mở rộng của JavaScript, cho phép viết HTML trong React.

* **Lợi ích:** Trực quan, dễ bảo trì, phát hiện lỗi sớm.

**10. Trình duyệt có đọc được JSX không?**

Không. Trình duyệt cần Babel để chuyển JSX thành JavaScript.

**11. Babel là gì?**

Babel là trình biên dịch mã giúp chuyển đổi ES6/JSX thành mã ES5 tương thích trình duyệt. Toolchain được sử dụng để chuyển đổi mã JavaScript chuẩn mới (ECMAScript 2015+) về phiên bản cũ hơn để đảm bảo chương trình có thể hoạt động bình thường trong các trình duyệt cũ

**12. Không có JSX có code React không?**

Có. JSX là tùy chọn, bạn có thể dùng React.createElement.

**13. Lưu ý khi sử dụng JSX?**

* Đóng tất cả thẻ.
* Chỉ có **một** phần tử cha.
* Sử dụng className thay vì class.

**14. Làm sao viết mã JavaScript trong JSX?**

Dùng {} để nhúng JavaScript trong JSX:

jsx

Copy code

<h1>{`Xin chào ${name}`}</h1>

**15. Component là gì?**

* Component là một khối mã độc lập, có thể tái sử dụng, quản lý giao diện trong React. Component giống như các hàm trả về Html. Các component là các thành phần có thể sử dụng lại.

**16. Các loại component?**

* **Function Component:** Đơn giản, sử dụng hàm.
* **Class Component:** Sử dụng lớp, hỗ trợ lifecycle.

**17. Phân biệt Function và Class Component?**

* Function: Ngắn gọn, hỗ trợ Hook.
* Class: Phức tạp hơn, có lifecycle methods.

**18. State? Cách cập nhật?**

* **State:** Trạng thái của component, lưu dữ liệu động. State là nơi lưu trữ các giá trị thuộc tính thuộc về component. State là nơi lưu trữ các giá trị thuộc tính thuộc về component
* **Cập nhật:** Dùng setState hoặc useState.
* Hai cách xử lý dữ liệu trong React là gì state va props
* Trong vòng đời của component, Mounting là giai đoạn gì? 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.
* Phương thức componentDidMount được gọi khi Components được tạo ra lần đầu tiên
* Chỉ class components mới có vòng đờ

**19. State thay đổi thì điều gì xảy ra?**

React re-render UI với giá trị state mới. React sẽ thực hiện Render lại, Cập nhật DOM, Gọi các lifecycle methods. sẽ được gọi (nếu có) sau khi state thay đổi và component được render lại.

**20. Props là gì?**

Props là dữ liệu truyền từ component cha sang con. để truyền data đến component bên ngoài

**21. Phân biệt state và props?**

* **State:** Nội bộ, có thể thay đổi.
* **Props:** Truyền từ ngoài, không thể thay đổi.

**22. Lifecycle methods?**

Các giai đoạn của component: Mounting, Updating, Unmounting.

 **Mounting** (Giai đoạn "Mount"):

* Xảy ra khi component được tạo ra và chèn vào DOM.
* Các phương thức liên quan: constructor(), render(), componentDidMount().

 **Updating** (Giai đoạn "Update"):

* Xảy ra khi component nhận props mới hoặc state thay đổi, dẫn đến việc render lại.
* Các phương thức liên quan: shouldComponentUpdate(), render(), componentDidUpdate().

 **Unmounting** (Giai đoạn "Unmount"):

* Xảy ra khi component bị loại bỏ khỏi DOM.
* Phương thức liên quan: componentWillUnmount().

**23. Hook là gì?**

Hooks là những hàm cho phép bạn "kết nối" React state và lifecycle vào các components sử dụng hàm. Hooks là vòng đời của ứng dụng ReactJS

**24. Một số Hook đã học?**

* useState: Gọi useState từ bên trong một function component. Gọi useContext từ bên trong useState.
* useEffect: Khi một giá trị đang được sử dụng, mảng phụ thuộc của Effect thay đổi, Khi thành phần ngắt kết nối
* useContext
* useReducer

**25. useState?**

* **Cập nhật:** setState(newValue).
* Gọi useState từ bên trong một function component
* Gọi useContext từ bên trong useState
* Bạn ko thể cập nhật trạng thái (với một hàm được trả về từ useState) trong một hook useEffect

**26. useEffect?**

* **Khi bạn trả về một hàm từ bên trong cuộc gọi useEffect , ham se duoc thuc thi Khi một giá trị đang được sử dụng, mảng phụ thuộc của Effect thay đổi**

**27. Router dùng để làm gì?**

Quản lý điều hướng trang (SPA) với React Router.

**28. Formik và Yup?**

* **Formik:** Quản lý form.
* **Yup:** Validation.

**29. Bất đồng bộ là gì?**

Thực thi không đồng thời.

* **Cách xử lý:** Callback, Promise, async/await.

**30. Axios là gì?**

Thư viện HTTP client dùng để gọi API.

**31. Redux là gì?**

Quản lý state toàn cục.

**32. Khi nào dùng Redux?**

Ứng dụng phức tạp, nhiều state chia sẻ.

**33. Thành phần Redux?**

* Store
* Reducer
* Action

**34. Middleware là gì?**

Lớp trung gian xử lý logic giữa Action và Reducer.

**35. SSR, CSR, SSG?**

* **SSR:** Server-Side Rendering.
* **CSR:** Client-Side Rendering.
* **SSG:** Static-Site Generation.

**36. Tính năng của Next.js?**

* SSR, SSG, Routing, API Routes, tối ưu SEO.

**ChatGPT said:**

ChatGPT

**Lifecycle (Chu kỳ sống)** trong React là quá trình mà một component trải qua từ khi được tạo ra cho đến khi bị hủy bỏ. Mỗi component trong React sẽ có một **lifecycle** nhất định, bao gồm nhiều giai đoạn khác nhau: **mounting**, **updating**, và **unmounting**.

**Các giai đoạn của Lifecycle:**

1. **Mounting** (Giai đoạn "Mount"): Đây là giai đoạn khi component được tạo ra và chèn vào trong DOM.
   * **constructor()**: Được gọi khi một instance của component được tạo ra. Đây là nơi ta có thể khởi tạo state hoặc binding các phương thức.
   * **static getDerivedStateFromProps()**: Được gọi mỗi khi props thay đổi. Dùng để cập nhật state dựa trên thay đổi của props. Phương thức này là **static**, không có quyền truy cập vào this.
   * **render()**: Phương thức này trả về JSX để render component vào DOM.
   * **componentDidMount()**: Được gọi ngay sau khi component đã được render lần đầu tiên vào DOM. Đây là nơi tốt nhất để thực hiện các thao tác như gọi API, thiết lập subscription, v.v.
2. **Updating** (Giai đoạn "Update"): Đây là giai đoạn khi một component đã được mount và sau đó có sự thay đổi về state hoặc props, khiến component cần phải render lại.
   * **static getDerivedStateFromProps()**: Như đã đề cập, phương thức này được gọi mỗi khi props thay đổi.
   * **shouldComponentUpdate()**: Phương thức này được gọi trước khi component render lại. Dùng để tối ưu hiệu suất bằng cách quyết định xem có cần render lại hay không (mặc định là true).
   * **render()**: Cũng giống như khi mounting, phương thức này trả về JSX để render component.
   * **getSnapshotBeforeUpdate()**: Được gọi ngay trước khi thay đổi được áp dụng vào DOM (trước khi render).
   * **componentDidUpdate()**: Được gọi ngay sau khi component đã được render lại và DOM đã được cập nhật. Đây là nơi thích hợp để xử lý các tác vụ sau khi render, ví dụ như gọi API hoặc cập nhật DOM.
3. **Unmounting** (Giai đoạn "Unmount"): Đây là giai đoạn khi component sẽ bị loại bỏ khỏi DOM.
   * **componentWillUnmount()**: Được gọi trước khi component bị unmount và bị hủy bỏ khỏi DOM. Đây là nơi thích hợp để dọn dẹp, như xóa subscription hoặc hủy bỏ các tác vụ bất đồng bộ.