# 1.Reactjs là gì?

**React** là thư viện **JavaScript** front-end được phát triển bởi **Facebook** vào năm 2011.  
Nó được dùng để xây dựng giao diện người dùng (**User Interface – UI**) dạng Single Page.

ReactJS phổ biến nhất trong lĩnh vực phát triển web­­. React được tạo ra để xây dựng giao diện người dùng nhanh và tương tác cho các ứng dụng web và di động. Nhờ vào một số tính năng nổi bật – có thể được sử dụng để phát triển cả ứng dụng web và ứng dụng di động, luồng dữ liệu một chiều, các components có thể tái sử dụng, tạo các ứng dụng động, v.v.

Các tính năng quan trọng của React bao gồm:

* Hỗ trợ kết xuất phía máy chủ.
* Tuân theo ràng buộc dữ liệu một chiều hoặc luồng dữ liệu.
* Sử dụng các component UI có thể tái sử dụng hoặc kết hợp để phát triển giao diện người dùng.
* Tăng hiệu suất của ứng dụng với Virtual DOM.
* JSX làm cho mã dễ đọc và viết.
* Dễ dàng tích hợp với các frameworks khác (Angular, BackboneJS) vì nó chỉ là một thư viện giao diện.
* Dễ dàng viết các trường hợp kiểm thử UI và tích hợp với các công cụ như JEST.

# 2. JEST là gì?

Jest là một framework để unit test JavaScript do Facebook thực hiện dựa trên Jasmine và cung cấp mô hình giả tự động tạo và môi trường Jsdom. Nó thường được sử dụng để kiểm thử các component trong React.

# 3.JSX là gì? Các trình duyệt có đọc được JSX không?

JSX (JavaScript XML) là một loại cú pháp mở rộng dành cho ngôn ngữ JavaScript viết theo kiểu XML.

Mã lệnh viết bằng JSX sẽ được chuyển sang JavaScript để trình duyệt có thể hiểu được.

Các trình duyệt không thể đọc được các thẻ JSX bởi vì JSX không phải một đối tượng Javascript thông thường.  
Về cơ bản, nó là một phương ngữ mới phổ biến mà chỉ đơn giản tích hợp các mẫu HTML vào mã   
JavaScript. Để đọc được cần có sự hỗ trợ của WebPack hoặc Babel.

# 4. DOM ảo (Virtual DOM) là gì?

* DOM là viết tắt của Document Object Model. DOM đại diện cho một tài liệu HTML có cấu trúc cây logic. Mỗi nhánh của cây kết thúc bằng một nút và mỗi nút chứa các đối tượng.
* React giữ một bản “đại diện” nhưng nhẹ hơn của DOM “thực” trong bộ nhớ, gọi là DOM ảo (Virtual DOM). Virtual DOM so sánh trạng thái trước đó của nó và chỉ cập nhật các thành phần có trạng thái thay đổi trong DOM thực thay vì cập nhật tất cả các thành phần, như các ứng dụng web thông thường.

# 5.  Tại sao bạn thích dùng Reactjs?

1. **Dễ dàng tạo các ứng dụng động: ít code hơn và nhiều tính năng.**
2. **Hiệu suất cải thiện: sử dungj DOM ảo.** React sử dụng DOM ảo, giúp các ứng dụng web hoạt động nhanh hơn. Virtual DOM so sánh trạng thái trước đó của nó và chỉ cập nhật các thành phần có trạng thái thay đổi trong DOM thực thay vì cập nhật tất cả các thành phần, như các ứng dụng web thông thường.
3. **Các thành phần có thể tái sử dụng (component): l**àm giảm đáng kể thời gian phát triển một ứng dụng.
4. **Luồng dữ liệu một chiều:** vì dữ liệu chảy theo một hướng, nên việc debug và biết vấn đề xảy ra ở đâu trong ứng dụng sẽ trở nên dễ dàng hơn.
5. **Các công cụ chuyên dụng để debug dễ dàng:** Facebook đã phát hành một tiện ích mở rộng Chrome mà chúng ta có thể sử dụng để debug các ứng dụng React. Điều này làm cho quá trình gỡ lỗi React cho các ứng dụng web nhanh hơn và dễ dàng hơn.

# 6.Làm thế nào để tạo một ứng dụng React?

* [Cài đặt NodeJS](https://nodejs.org/en/download/) trên máy tính vì chúng ta cần npm để cài đặt thư viện React.
* Cài đặt gói create-react-app bằng command prompt hoặc terminal.

# 7. Context trong React?

Context cung cấp một cách thức để truyền dữ liệu qua cây thành phần (component tree) mà không phải chuyển các Props xuống theo cách thủ công ở mọi cấp độ.

# 8. Liệt kê một vài giới hạn của React?

* React chỉ là một thư viện, không phải là 1 framework toàn diện.
* React là một thư viện lớn mất nhiều thời gian để học.
* Nó có thể hơi khó để làm quen đối với các lập trình mới.
* Việc code sẽ phức tạp hơn vì React dùng inline-templating và JSX

# 9. Chức năng của hàm render() trong React là gì?

Hàm **render** chính là mô tả cụ thể của UI tại bất cứ thời điểm nào. Vì thế nếu dữ liệu thay đổi, React sẽ thực hiện việc **update UI** với dữ liệu tương ứng. Khi dữ liệu thay đổi, React sẽ tự động gọi hàm **render** để **update lại UI**.

# 10. Props?

Là một cơ chế quan trọng để truyền dữ liệu từ một component cha đến một component con. Props là các thông số đầu vào mà bạn có thể truyền từ một thành phần gốc (cha) tới các thành phần con bên trong cây thành phần.

Props giúp bạn tái sử dụng và tạo các thành phần linh hoạt, có khả năng hiển thị dữ liệu động.

# 11.Form trong React là gì?

React sử dụng form để cho phép người dùng tương tác với các ứng dụng web.

* Sử dụng form, người dùng có thể tương tác với ứng dụng và nhập thông tin cần thiết bất cứ khi nào cần. Form chứa các phần tử nhất định, chẳng hạn như trường văn bản, nút, hộp kiểm, nút radio, v.v.
* Form được sử dụng cho nhiều tác vụ khác nhau như xác thực người dùng, tìm kiếm, lọc, lập chỉ mục, v.v.

# 12. Refs được dùng như thế nào?

Refs thường được dùng để trả về một tham chiếu tới 1 phần tử.

# 13.State?

Được sử dụng để quản lý dữ liệu có thể thay đổi trong một component. State cho phép bạn theo dõi và cập nhật các giá trị dữ liệu mà có thể thay đổi trong quá trình tương tác với ứng dụng.

**Props không thể thay đổi, state có thể thay đổi trong component**

**setState() để cập nhật state**: Để cập nhật state, bạn cần sử dụng phương thức setState(). Việc gọi setState() sẽ gây ra việc cập nhật lại state và kích hoạt việc render lại giao diện.

# 14. Hàm setState trong Reactjs là đồng bộ hay bất đồng bộ? Tại sao?

Hàm **setState** là bất đồng bộ (asynchronous). Điều này có nghĩa là khi bạn gọi **setState**, React không thực hiện cập nhật state ngay lập tức, mà nó sẽ xếp hàm cập nhật vào trong hàng đợi và tiếp tục thực hiện các công việc khác. Sau khi React hoàn thành các công việc ưu tiên, nó mới thực hiện cập nhật state và kích hoạt việc render lại giao diện.

Lý do hàm setState được thiết kế là bất đồng bộ là để tối ưu hóa hiệu suất. Nếu React thực hiện cập nhật state ngay lập tức và kích hoạt render ngay khi bạn gọi setState, có thể sẽ có nhiều lần render không cần thiết khi bạn thay đổi nhiều lần state trong một khoảng thời gian ngắn. Bằng cách làm cho setState bất đồng bộ, React có thể tự quản lý và tổng hợp các cập nhật state để tránh việc render không cần thiết.

# 15. Vì sao phải sử dụng key trong danh sách?

* Key rất quan trọng trong danh sách bởi vì:
* Key là một định danh độc nhất và nó được sử dụng để xác định những mục nào đã thay đổi, được cập nhật hoặc bị xóa khỏi danh sách.
* Key cũng giúp xác định thành phần nào cần được render lại thay vì lần nào cũng render lại tất cả các thành phần. Do đó, key giúp làm tăng hiệu suất, vì chỉ các thành phần được cập nhật mới được render lại.

# 16.  Mục đích của tham số thứ 2 (là hàm callback) của setState là gì?

Vì **setState** là **bất đồng bộ** nên hàm callback được sử dụng để xử lý 1 logic nào đó sau khi setState kết thúc và component được hiển thị.

# 17. Khái niệm hook trong reactjs?

Hook là một tính năng quan trọng trong React từ phiên bản 16.8 trở đi, giúp bạn sử dụng các tính năng của React như state và lifecycle trong các functional components mà không cần sử dụng các class component.

Hook giúp bạn tái sử dụng logic, trạng thái (state), và hành vi (behavior) trong các functional component một cách dễ dàng hơn và giảm thiểu sự phức tạp của việc quản lý state và lifecycle trong class component. Các hook cơ bản bao gồm:

* useState: Giúp bạn quản lý state trong functional component.
* useEffect: Cho phép bạn thực hiện các tác vụ sau mỗi lần render hoặc khi state thay đổi.
* useContext: Giúp bạn truy cập vào context trong functional component.
* useReducer: Giúp bạn quản lý state phức tạp thông qua reducer.
* useCallback: Giúp bạn tạo callback được memoized để tránh việc render không cần thiết.
* useMemo: Giúp bạn memoize giá trị tính toán để tránh tính toán không cần thiết.
* useRef: Cho phép bạn tạo một tham chiếu liên quan đến DOM hoặc bất kỳ giá trị nào mà không gây ra việc render lại.

# 18.  Higher-Order component là gì?

Higher-Order component (HOC) là một hàm nhận một component và trả về một component mới.

Về cơ bản, đó là một pattern có nguồn gốc từ bản chất component của React.

Chúng có thể chấp nhận bất kỳ component con nào được cung cấp động nhưng chúng sẽ không sửa đổi hoặc sao chép bất kỳ hành vi nào từ các component đầu vào của chúng.

HOC được sử dụng cho nhiều trường hợp bên dưới:

* Tái sử dụng mã, logic và trừu tượng.
* Tính trừu tượng và thao tác trên state.
* Các thao tác trên props.

# 19. Redux là gì?

Redux là một thư viện JavaScript quản lý trạng thái (state management) được sử dụng chủ yếu trong các ứng dụng React để quản lý và chia sẻ trạng thái của ứng dụng trên toàn bộ cây thành phần. Redux giúp bạn duy trì trạng thái ứng dụng trong một nguồn duy nhất (single source of truth) và quản lý các thay đổi của trạng thái một cách dễ dàng và dự đoán.

Redux có một số khái niệm chính:

* Store: Là nơi chứa toàn bộ trạng thái của ứng dụng. Trạng thái này được quản lý dưới dạng một đối tượng. Store là nguồn dữ liệu duy nhất cho toàn bộ ứng dụng.
* State: Là trạng thái của ứng dụng, được lưu trữ trong store. State không thể thay đổi trực tiếp, mà chỉ có thể thay đổi thông qua các hành động (actions) được gửi đến reducer.
* Action: Là một đối tượng mô tả một hành động cần thực hiện. Mỗi action bao gồm một loại (type) và dữ liệu liên quan (payload).
* Reducer: Là một hàm xử lý hành động và cập nhật trạng thái mới dựa trên action và trạng thái hiện tại. Reducer làm cho việc quản lý trạng thái trở nên dễ dàng và dự đoán.
* Dispatch: Là hàm được cung cấp bởi store để gửi các action tới reducer để cập nhật trạng thái. Khi bạn dispatch một action, Redux sẽ tự động điều hướng nó tới reducer tương ứng.

# 20. React Router là gì?

React Router là một thư viện được sử dụng trong ứng dụng React để quản lý việc điều hướng (routing) giữa các trang và thành phần của ứng dụng. Nó cho phép bạn tạo ra các đường dẫn URL và ánh xạ chúng tới các thành phần React tương ứng, giúp tạo ra trải nghiệm điều hướng trong các ứng dụng đơn trang (Single Page Application - SPA).

React Router cung cấp một loạt các thành phần và API để giúp bạn quản lý việc điều hướng, bao gồm:

* BrowserRouter: Một component chứa tất cả các Route và giúp xác định các đường dẫn URL.
* Route: Component xác định cách ánh xạ giữa một đường dẫn URL cụ thể và một thành phần React.
* Link: Component tạo ra các liên kết điều hướng bằng cách tạo ra các thẻ <a> để chuyển trang, nhưng không gây tải lại trang.
* NavLink: Tương tự như Link, nhưng cho phép bạn thêm các lớp CSS khi liên kết được chọn.
* Switch: Cho phép bạn chỉ ánh xạ đến một Route duy nhất trong các Route con của nó.
* React Router giúp bạn xây dựng các ứng dụng đơn trang có nhiều trang ảo một cách dễ dàng và hiệu quả. Thay vì tải lại toàn bộ trang mỗi khi điều hướng, React Router cho phép bạn thay đổi nội dung mà không cần tải lại trang, giúp cải thiện hiệu suất và trải nghiệm người dùng.

# 21. Định tuyến React (React routing) khác như thế nào so với định tuyến quy ước (conventional routing)?

|  |  |  |
| --- | --- | --- |
|  | **Định tuyến React** | **Định tuyến quy ước** |
| 1. | Trang HTML đơn | Mỗi view là một tệp HTML mới |
| 2. | Người dùng điều hướng nhiều chế độ xem trong cùng một tệp | Người dùng điều hướng nhiều tệp cho mỗi chế độ xem |
| 3. | Trang không làm mới vì nó là một tệp duy nhất | Trang làm mới mỗi khi người dùng điều hướng |
| 4. | Cải thiện hiệu suất | Hiệu suất chậm hơn |

# 22. Điểm khác nhau giữa State và Props?

|  |  |  |
| --- | --- | --- |
|  | **State** | **Props** |
| **Cách dùng** | Chứa dữ liệu về components | Cho phép truyền dữ liệu từ component sang những components khác như một argument |
| **Khả năng thay đổi (Mutability)** | Có thể thay đổi | Không thể thay đổi |
| **Chỉ đọc (Read-Only)** | Có thể thay đổi | Là read-only |
| **Component con** | Component con không thể truy cập | Component con có thể truy cập |
| **Stateless components** | Không thể có state | Có thể có props |