|  |  |
| --- | --- |
|  | [1. Tổng quan ES6](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| Ý chính | **Nội dung** |
| 1. ECMAScript6 (ES6) | - Là phiên bản tiếp theo của ngôn ngữ Javascript, phát hành vào năm 2015 và cung cấp 1 số |
| tính năng mới, bao gồm cải tiến cú pháp, khai báo biến, hàm, classes … và toán tử mới  -Tính năng mới:  + khai báo biến let với const  + Cung cấp thêm phương thức khai báo arrow function, giúp rút gọn cú pháp  + Cho phép sử dụng Template literals để tạo ra chuỗi có chứa các biểu thức javaScript và các kí tự đặc biệt khác  + Cung cấp cú pháp Destructuring assignment để giúp trích xuất các giá trị từ các đối tượng hay mảng.  + Hỗ trợ tách chương trình của mình thành các thành phần độc lập và dễ bảo trì hơn. |
| 2. Arrow Function | - Arrow Function là cú pháp viết hàm trong Javascript Sử djng dấu mũi tên => |
| - Lưu ý: Arrow Function không thay thế hoàn toàn Function bình thường |
| 3. Sử dụng forEach, map, filter, reduce để thao tác với mảng | - forEach: duyệt mảng không có trả về |
| - map: duyệt mảng có trả về 1 mảng mới |
| - filter: duyệt mảng có trả về 1 mảng mới thỏa mãn điều kiện cho trước |
| - reduce: dùng để tính toán với các phần tử của mảng |
| 4. Rest Parameter | - Rest Parameter cho phép biểu diễn số lượng vô hạn đối số dưới dạng một mảng |
| - Tham số biểu diễn bằng dấu … và phải nằm cuối so với các tham số khác |
| 5. Spread Operator | - Toán tử cho phép thao tác với mảng /object như thêm phần tử, kết hợp nhiều mảng/object |
| - Ứng dụng: |
| + Sao chép thành mảng mới |
| + Gộp nhiều mảng |
| + Chuyển chuỗi thành mảng |
| + … |
| 6.Destructuring Assignment | - Là cú pháp cho phép tách rời dữ liệu được lưu trữ trong mảng hoặc object và gán chúng cho các biến riêng biệt một cách dễ dàng và nhanh chóng |

|  |  |
| --- | --- |
|  | [**2. Tổng quan về ReactJS & JSX**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. Tổng quan về ReactJS** | - ReactJS là thư viện của Javascript được sử dụng để phát triển view (Front end), nó có tính |
| tái sử dụng cao |
| **2. Ưu/nhược điểm của ReactJS** | - Tại sao nên dùng ReactJS? : |
| + Trend |
| + Có khả năng tối ưu SEO |
| + Khả năng tái sử dụng và mở rộng cao |
| + Hiệu suất cao hỗ trợ DOM ảo |
| - Ưu điểm: |
| + Viết ứng dụng bằng Javascript |
| + Biến các thành phần UI phức tạp trở nên đơn giản bằng cách khai báo các component đối |
| lập |
| + Khả năng xác định chính xác khi nào cần render cũng như khi nào bỏ đi phần tử trong DOM |
| - Nhược điểm: |
| + Chỉ phục vụ tầng view |
| + Tích hợp ReactJS vào các Framwork MVC truyền thống yêu cầu cần phải tinh chỉnh lại |
| + ReactJS có kích thước khá nặng so với các Framework khác |
| **3. Phân biệt CSR và SSR** | |  | | --- | |  | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **4. React Element** | - là đơn vị cơ bản nhất để tạo thành các componenttrong ứng dụng react  - Mỗi React Element đại diện cho 1 phần tử trong cây DOM của trang web |
| - React Element không phải là những gì chúng ta nhìn thấy trên DOM |
| - Cú pháp: React.createElement(type, [props], […children]) |
| **5. JSX** | - JSX = Javascript + SML |
| - Là 1 ngôn ngữ cho phép kết hợp Javascript vào các thẻ giống HTML để xác định các phần tử |
| giao diện người dùng và chức năng của chúng |
| - Cú pháp XML là được xác định dựa vào cấu trúc NODE lồng nhau, mỗi NODE sẽ có 1 cặp thẻ |
| mở đóng |

|  |  |
| --- | --- |
|  | [**3. React Components, State & Lifecycle method**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. React Component là gì? Các cách tạo** | - Component là block code độc lập, có thể tái sử dụng |
| - Mỗi Componenet có code Javascript, HTML, CSS riêng, chúng có thể tái sử dụng, dễ đọc, |
| dễ viết và dễ test  + Tên component được đặt theo quy tắc PascalCase. |
| - Có 2 loại: |
| + functionComponent  + classCompoent |
| -Các nhóm trong vòng đời của component:  + Mounting: Các phương thức trong nhóm này được gọi khi component được tạo ra và được gắn vào DOM {  Các phương thức bao gồm :  1/ constructor()  2/ static getDerivedStateFormProps()  3/ render()  4/ componentDismount().  }  +Updating: Các phương thức trong nhóm này được gọi khi component được cập nhật thông qua việc truyền các props mới or thông qua việc setState(),{   * static getDerivedStateFromProps() * shouldComponentUpdate() * render() * getSnapshotBeforeUpdate() * componentDidUpdate()   }  + Unmounting: Các phương thức trong nhóm này được gọi khi component bị xoá khỏi DOM  { componentWillUnmount} |
| **2. Props** | - Props là 1 từ viết tắt của properties |
| - Props trả về object, được sử dụng trong React để truyền data qua lại giữa các Component |
| **3. State** | - State là nơi lưu trữ các giá trị thuộc tính của Component |
| - Khi State thay đổi thì Component sẽ được render lại |
| - React cung cấp phương thức setState() để cập nhật State (đối với classComponent) |
| **4. Pure Function** | - Là hàm thuần tuý |
| - Hàm luôn trả về cùng 1 kết quả nếu được truyền vào các tham số không đổi, hàm không phụ thuộc vào trong trạng thái hoặc dữ liệu nào. Hàm chỉ phụ thộc vào tham số đầu vào |
| - Hàm không gây hay tạo ra bất kỳ ảnh hưởng nào đến các đối tượng khác |
| - Lý do: |
| + Dễ kiểm tra |
| + Refactor code tiện lợi hơn |
| + Không gây side effect |

|  |  |
| --- | --- |
|  | [**4. Hook**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. Hooks** | - Hooks là tính năng mới trong React v16.8, đây là những hàm đặc biệt, được React tạo sẵn cho phép kết nối với React state và lifecycle vào function component |
|  |
| - Lợi ích: |
| + Làm các component trở nên gọn nhẹ hơn |
| + Cho phép sử dụng state ngay trong function component  + Giảm đáng kể số lượng code và dễ tiếp cận  + Có thể tái sử dụng logic |
| - Một số hook hay dùng: |
| + useState |
| + useEffect |
| + useLayoutEffect |
| + useContext |
| + useReducer |
| + useMemo |
| + useCallback  + useRef |
| **2. useState** | - Cú pháp: const [useName, setUseName] = useState('initial value') |
| - Cho phép thêm/lưu trữ state tới component react function |
| **3. useEffect** | - Cho phép xử lý các logic trong các vòng đời của component và các side effect |
|  | - các trường hợp: |
|  | + useEffect(callBack) |
|  | + useEffect(callBack, []) |
|  | + useEffect(callBack, [deps] |

|  |  |
| --- | --- |
|  | [**5. React Form & Router**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. React Form** | - Trong ReactJs để tạo form thì có sử dụng lại các thành phần của form HTML như |
| <input/>, <form/>, … kết hợp với sự kiện của form như onChange, onClick, … để xử lý data trên from |
|  |
| - Validate sử dụng các hàm của Javascript |
| - Ngoài ra còn có 1 số thư viện hỗ trợ cho bạn xử lý form và validate: Formik, Yup, |
| react-form-hook |
| **2. React Router** |  |
| - Là thư viện được xây dựng trên React, giúp cho lập trình viên xây dựng luồng đi cho ứng dụng, xây dựng ứng dụng SPA |
| vào 1 URL nhất định, nếu URL đó được tìm thấy trong các URL được định nghĩa trong |
| Router, thì người dùng sẽ được chuyển hướng đến page tương ứng  -Tại sao nên sử dụng Router trong react :  + Giúp quản lý định tuyến .  + Tối ưu trải nghiệm người dung.  + Dễ bảo trì và mở rộng.  + Hỗ trợ SEO: Cho phép tạo ra các đường dẫn tĩnh giúp cho việc tối ưu hoá công cụ tìm kiếm. |
|  |

|  |  |
| --- | --- |
|  | [**6. API Clients**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. Sync và Async** | - Sync (Đồng bộ): |
| + Các công việc thưc hiện 1 cách tuần tự |
| + vd: có 2 công việc A, B thì khi A thực hiện xong thì B mới thực hiện |
| + Điều này ảnh hưởng tới trải nghiệm người dùng |
| - Async (Bất đồng bộ): |
| + Các công việc thực hiện không theo tuần tự |
| + vd: khi A đang thực hiện thì chương trình cũng thực hiện B mà không đợi A hoàn thành |
| - Cơ chế thực hiện bất đồng bộ: callback, promise, async/await |
| **2. Axios** | - Là 1 thư viện HTTP Client dựa trên promise |
| - Cơ bản nó cung cấp 1 API cho việc xử lý XHR (XMLHttpRequest)  - Được viết dựa trên Promse |
| - Công dụng: |
| + Tạo XHR từ trình duyệt |
| + Hỗ trợ promise API |
| + Chuyển đổi dữ liệu request và response |
| + Xử lý tiền kỳ và hậu kỳ request/response |
| + Tự động chuyển đổi JSON |
| + Hủy request |
| + … |
| **3. Sử dụng promise để xử lý bất đồng bộ** | - promise sinh ra để xử lý các kết quả của 1 hành động cụ thể, kết quả của mỗi hành động sẽ là thành công hoặc thất bại và promise giúp giải quyết vấn đề: thành công sẽ làm gì?, thất bại sẽ làm gì? |
| - Khi promise được khởi tạo thì có 1 trong 3 trạng thái: |
| + Fullfilled: 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 2 trạng thái Fullfilled và Rejected được gọi là settled, tức là đã xử lý xong |
| **4. Async và Await** | - Là 1 cơ chế giúp thực hiện các thao tác bất đồng bộ một cách tuần tự hơn |
| - async/await vẫn sử dụng promise nhưng mã ngầm dễ đọc và dễ theo dõi hơn |
| - async: khai báo bất đồng bộ |
| + Tự động biến đổi 1 hàm bình thường thành hàm có kiểu trả về là 1 promise |
| + Khi gọi tới hàm async thì hàm sẽ xử lý mọi thứ và trả về kết quả trong hàm của nó |
| + async cho phép sử dụng await |
| - await: sẽ được sử dụng trước thao tác bất đồng bộ, tạm dừng việc thực hiện các hàm |
| sau nó |
| + Khi đặt await trước 1 promise, nó sẽ đợi cho đến khi promise kết thúc và trả về kết quả |
| + await chỉ hoạt động với promise, không hoạt động với callback |
| + await chỉ sử dụng được bên trong async |

|  |  |
| --- | --- |
|  | [**7. Redux**](file:///C:\Users\admin\Documents\Zalo%20Received%20Files\Luận%20án%20chiến%20sĩ%20module%205.xlsx#Menu!A1) |
|  |  |
| **Ý chính** | **Nội dung** |
| **1. Redux** | - Redux là 1 thư viện để quản lý state của ứng dụng |
| - Redux cung cấp 1 nơi lưu trữ state cho toàn ứng dụng và đảm bảo state chỉ được cập nhật |
| theo cách có thể dự đoán dược |
| - Nơi lưu trữ state được gọi là store và mỗi component đều có thể access bất kỳ state nào |
| mà chúng muốn sử dụng trong store này |
| - Ưu điểm: |
| + Khả năng bảo trì |
| + Có thể dự đoán trạng thái |
| + Hiệu suất |
| + Dễ dàng kiểm thử |
| + Cộng đồng lớn |
| + Dễ dàng tích hợp vào các kiến trúc khác như Angular  -Nguyên tắc của Redux :  +Single source of truth(nguồn đáng tin cậy duy nhất).  + State is read-only (trạng thái chỉ đọc).  +Changes are made with pure functions(Thay đổi được thực hiện bằng các hàm thuần tuý) |
| **2. Redux Actions** | - Action là các sự kiện khi 1 sự kiện được thực hiện |
| - Action là đối tượng trong Javascript |
| - Các action được gửi đến store bằng dispatch |
| - Action thường có 2 thuộc tính: |
| + type: chỉ ra hành động sẽ thực hiện (thường là hằng số) |
| + payload: chứa thông tin dữ liệu được gửi đi |
| **3. Redux Reducer** | - Reducer là các pure function lấy trạng thái hiện tại của ứng dụng, thực hiện hành động và |
| trả về trạng thái mới |
| - Reducer phải tuân thủ theo 1 số quy tắc sau: |
| + Chỉ được tính toán state mới dựa trên các đối số state cũ và action |
| + Không được phép sửa state hiện tại và hãy copy ra giá trị state hiện tại và thay đổi giá trị |
| của state copy đó |
| + Không được xử lý bất đồng bộ, tính toán ngẫu nhiên hoặc là các hàm có side effect |
| - Logic xử lý: |
| + Kiểm tra nếu reducer cần thực hiện truyền vào => action hợp lệ => copy state hiện tại |
| => tạo state mới và return nó |
| + Nếu action không hợp lệ => return state cũ |
| **4. Redux store** | - Store là 1 object tree bất biến trong Redux store là 1 vùng chứa state của ứng dụng, mỗi lần reducer xử lý sẽ trả về 1 state mới. Tập hợp các state này sẽ thành cây state mà store nắm giữ |
| **5. Quá trình xử lý dữ liệu trong redux** | - Người dung thực hiện một hành động trên giao diện người dùng.  - Các thành phần React (component) sẽ gửi 1 action đến Reducer.  - Reducer xử lý action và trả về trạng thái mới của ứng dụng.  - Store cập nhập trạng thái của ứng dụng.  - Các thành phần nhận được thông báo về sự thay đổi trạng thái của ứng dụng, hiển thị lại giao diện người dùng để phản ánh sự thay đổi. |
|  |  |
| **Next.js** | Next.js là một framework React mã nguồn mở phổ biến được sử dụng để xây dựng các ứng dụng web có khả năng rendering trên phía máy chủ (SSR) và tĩnh (SSG). Nó được tạo ra bởi Zeit (hiện đã trở thành Vercel) và hiện được bảo trì bởi cộng đồng các nhà phát triển.  Next.js cung cấp nhiều lợi ích hơn so với phát triển React truyền thống, bao gồm:   1. Rendering trên phía máy chủ: Next.js cho phép bạn render các thành phần React của mình trên máy chủ, điều này có thể cải thiện hiệu suất và SEO. 2. Tự động chia mã: Next.js sẽ tự động chia mã của bạn thành những phần nhỏ hơn, điều này có thể cải thiện thời gian tải trang. 3. Tạo tĩnh trang web: Next.js cho phép bạn tạo ra các tệp HTML tĩnh cho ứng dụng của mình, có thể được phục vụ trực tiếp từ CDN. 4. Hệ thống định tuyến tích hợp: Next.js cung cấp một hệ thống định tuyến tích hợp, cho phép bạn dễ dàng tạo trang và điều hướng giữa chúng. 5. Các đường API tích hợp: Next.js cho phép bạn tạo các đường API để xử lý logic và lấy dữ liệu trên phía máy chủ. 6. Hỗ trợ CSS tích hợp: Next.js cung cấp hỗ trợ tích hợp CSS cho các module, cho phép bạn viết CSS có phạm vi cho các thành phần của mình.   Tổng thể, Next.js là một framework mạnh mẽ có thể giúp bạn xây dựng các ứng dụng web hiệu suất cao một cách dễ dàng. |