- Mô tả dự án đang làm

- Task khó nhất từng làm

- Có tự xử lý task đc kh (hay phải nhờ senior giúp)

- 1 chút về quê quán, trường học, dự án đã làm khi đi học

- Tính các số chẫn từ 0 đến 100

- Tính góc nhỏ nhất tạo bởi kim giờ và phút khi đồng hồ 3h30

- Vue có phải ngôn ngữ (framework) hướng đối tượng không

- Có tự tạo đc cấu trúc thư mục và deploy mọi thứ lên server đc không

- Mô tả những gì làm đc với git (push, pull, tạo nhánh … )

- Team hiện tại có mấy người và chức năng từng người

- Trình độ tiếng anh (t nói đọc viết, không giao tiếp đc)

- Tại sao về cần thơ làm

- Nếu không có dự án thì có làm công nghệ khác được không

- Có câu nào hỏi lại bố mày không, nếu không thì bố mày tắt call

1. **Callback hell là gì?**

* Callback hell là trạng thái nhiều callback lồng vào nhau nhiều cấp khiến cho việc đọc hiểu code trở nên cực kì khó khăn.

Text

Description automatically generated

1. **Làm sao để không gặp phải callback hell**

* Phương pháp chính là phân tách code thành nhiều function nhỏ, mỗi callback được viết thành một function riêng và gọi tới function xử lý tiếp theo.

Text

Description automatically generated

* Ngoài ra, có 2 kĩ thuật để dễ dàng xử lý callback hell:
  + Sử dụng promises:

Text

Description automatically generated

* + Sử dụng async-await:

Text

Description automatically generated

1. **Closure là gì?**

* Là function có thể nhớ nơi nó được tạo ra và có thể truy cập được biến bên ngoài phạm vi của nó

A screenshot of a computer

Description automatically generated with medium confidence

1. **React là gì?**

* Là thư viện JS front-end được phát triển bởi Facebook dùng để xây dựng ứng dụng Single Page

1. **State trong React**

* Là object chứa thông tin có thể thay đổi trong suốt thời gian tồn tại của component

1. **Jsx là gì?**

* Là cú pháp mở rộng của JS viết theo kiểu XML

1. **Prop vs State vs global state(redux)**

* Props được truyền từ component cha sang component con và giá trị của props không thể thay đổi được
* State: có thể thay đổi giá trị, chỉ dùng trong component chứa state đó
* Global state: có thể thay đổi giá trị, được sử dụng bởi nhiều component khác nhau

1. **Redux và useContext vs useReducer?**

* Redux Là thư viện quản lý state có thể dự đoán được dựa trên khái niệm store, để cập nhật store, phải dispatch 1 action tới store thông qua reducer. Reducer nhận vào state hiện tại và action vừa được dispatch để trả ra state mới.
* useContext dùng để state giữa các component => Nhược: khi state trong useContext thay đổi, những component trong context này đều bị re-render.
* userReducer là hook quản lý state giống như useState nhưng cách dùng của giống redux.

1. **Real DOOM vs Virtual DOOM**

* Doom(Document object model) là 1 chuẩn được định nghĩa để thao tác và truy xuất trên code HTML
* DOOM ảo là bản sao của real DOOM, react sử dụng DOOM ảo để tăng hiệu xuất vì khi state thay đổi, react chỉ load lại những phần có dữ liệu bị thay đổi chứ không load lại toàn bộ component lớn.

1. **Render là gì?**

* Là quá trình xảy ra khi state của component thay đổi và component cần cập nhật lại dữ liệu mới.

1. **Var, let, const ?**

* Let tạo ra biến có thể truy cập trong block chứa biến đó
* Var tạo ra biến có thể truy cập trong function chứa biến đó, có thể dẫn tới hoisting
* Const dùng khai báo hằng số

1. **Hoisting?**

* Là cơ chế của JS, dời các khai báo biến lên đầu phạm vi của biến trước khi thực thi code => Bởi vì chỉ có phần khai báo được chuyển lên đầu mà ko có phần gán giá trị nên mặc định biến bị undefined

1. **Null và undefined?**

* Null là object còn undefined là 1 kiểu giá trị
* Phép toán với null trả về 0 còn undefined trả về NaN

1. **Server side render vs Client side render?**

* Server side render: logic thực hiện phần lớn ở server (render, validation, authentication, routing,..)
  + Ưu: load nhanh, chạy được trên phần lớn các trình duyệt
  + Nhược: trang phải load lại nhiều lần, nặng server, tốn băng thông vì server gửi dữ liệu lớn
* Client side render: render HTML, CSS thực hiện ở trình duyệt (client). logic đơn giản thực hiện ở client (routing, validation,..), còn logic phức tạp vẫn nằm ở server(data process, report).
  + Ưu: Page chỉ load khi user chuyển trang hoặc thêm dữ liệu, giảm tải xử lí logic ở server, giảm băng thông do chỉ lấy json và dữ liệu cần thiết
  + Nhược: load lần đầu tiên chậm, project phải chia thành 2 phần front-end và back-end, ko chạy được nếu disable JS

1. **React vs Angular?**

* React chỉ dùng phần View trong mô hình MVC còn angular thì dùng toàn bộ mô hình MVC
* React là server side rendering còn Angular là Client-side rendering
* React dùng DOOM ảo còn Angular dùng DOOM thật
* React dùng one-way binding (dữ liệu truyền 1 chiều từ view sang component hoặc ngược lại), Angular dùng two-wat binding (thay đổi dữ liệu từ component qua view và ngược lại)

1. **Lifecycle trong reactjs?**

- Được tạo ra (Mouting – được thêm vào DOOM) (thực thi 1 lần)

- Qua nhiều thay đổi (Updating) (thực hiện nhiều lần)

- Bị hủy bỏ (Unmouting – bỏ ra khỏi DOOM) (thực thi 1 lần)

1. **Arrow function là gì?**

* Là hàm vô danh, cách khai báo ngắn gọn so với function

1. **Component và element**

* Component nhận vào giá trị đầu vào và trả về element

1. **Reactjs và ReactNative**

* React là thư viện front-end JS còn ReactNative là mobile framework

1. **setTimeOut, setInterval**

* setTimeOut: thực thi block code sau 1 khoảng thời gian nhất định (thực hiện 1 lần duy nhất)
* setInterval: thực thi block code liên tục sao 1 khoảng thời gian nhất định

1. **Custom hook?**

* Là hook tự định nghĩa, tên bắt đầu với “use”.
* Có thể sử dụng những hook khác (react hook – useState, useEffect hoặc custom hook khác)
* Custom hook dùng để tách riêng phần logic khỏi UI, và chia sẻ logic này cho nhiều component khác nhau.

1. **useRef?**

* Luôn luôn giữ state mới nhất nhưng không render giá trị đó lên

1. **Ngăn component hiển thị trong react?**

* Trả về null ở hàm render()

1. **Key là gì khi hiển thị danh sách ?**

* Key giúp xác định item nào đã thay đổi, thêm mới hoặc xóa bỏ.
* Nên sử dụng key để cung cấp định danh cho các phần tử

1. **Tại sao cần viết hoa component?**

* Component ko phải là phần tử DOOM mà là hàm tạo, nên nếu ko được viết hoa có thể dễ gây nhầm lẫn

1. **setState là hàm bất đồng bộ?**

* Hàm setState trong react là bất đồng bộ
* Vì react cố tình chờ khi tất cả component gọi tới setState trước khi re-render để tăng hiệu xuất và tránh render ko cần thiết

1. **Mục đích của tham số thứ 2 (hàm callback) trong setState?**

* Tại vì setState là hàm bất đồng bộ nên hàm callback này dùng để xử lí 1 logic nào đó sau khi setState kết thúc và component được hiển thị

1. **Map vs Filter vs Reduce?**

* Map: trả về từng phần tử trong array => dùng để khi muốn thực hiện 1 chức năng trên toàn bộ các phần tử trong array
* Filter: Trả về các phần tử trong array thỏa mãn đk => dùng để lọc bỏ các phần tử ko cần thiết trong array
* Reduce: tổng hợp các phần tử trong array về 1 giá trị duy nhất

1. **Destructuring?**

* Tách các phần tử của array hoặc object ra thành nhiều biến với 1 câu lệnh
* Text

  Description automatically generated

1. **Rest parameter là gì?**

* Khai báo function với số lượng tham số ko xác định

Text

Description automatically generated

1. **HTTP vs HTTPS?**

* HTTPS là HTTP(hypertext transfer protocol) và sử dụng các chứng chỉ SSL (secure sockets layer) để mã hóa dữ liệu truyền tải nhằm bảo mật giữa server và và trình duyệt.

1. **LocalStorage vs Session Storage vs Cookie?**

* LocalStorage: lưu trữ dữ liệu vĩnh viễn trên trình duyệt nếu user ko xóa hoặc cài lại trình duyệt
* SessionStorage: dữ liệu bị mất khi đóng tab hoặc trình duyệt
* Cookie: sau khi hết hạn, cookie sẽ biến mất khỏi trình duyệt

1. **Template String?**

* Là cú pháp viết biểu thức trong string

1. **Default parameter?**

****

1. **Oauth with google**

* App yêu cầu quyền truy cập tài nguyền từ user
* Nếu user cho phép, app gửi thông tin định danh của app và của user lên server
* Nếu danh tính hợp lệ, server trả về access token
* App yêu cầu tài nguyên và trình access token để xác nhận
* Nếu access token hợp lệ => cung cấp tài nguyên

1. **useMemo, useCallBack**

* useMemo: nhận vào 1 giá trị và danh sách dependencies, Chỉ khi dependencies thay đổi thì tính toán output.
* useCallBack: nhận vào 1 function và danh sách dependencies, trả về function. Chỉ khi dependencies thay đổi thì mới re-reder lại fucntion này.