// in line comment

/\* this is a multi

Line comment \*/

Data types

undefined, null, boolean, string, symbol, number and object

Variables

var myName = “vansh”

let ourName = “hey”

const pi = 3.184

var a; //declaring

a= 8; //assigning

var b=8;

console.log(“hey”)

variables are case sensitive

String can be in ‘ “ `

Escape characters

+ to add string

.length;

firstname[0];

strings are immutable

var newarray = [“hey”, 20];

nested or multidimensional array

arrays are mutable

var a = array.pop();

var b = array.shift();

array.unshift(“hello”);

without var keyword variable automatically becomes global

local variable takes precedence over global variable

==

===

!=

!==

Type script is a superset of java script that adds static typing to the language to help catch errors early in the development process

React is a javascript library

Props

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

Composing components

Extracting components

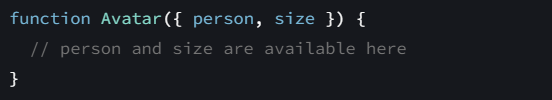
Proptypes (also default, isRequired)

**Pass props to the child component**



Note: Double curly braces to pass an object

**Read props inside the child component**





import \* as React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-navigation/native-stack';

// Define the stack navigator

const Stack = createNativeStackNavigator();

// Define the screens/components

function HomeScreen() {

return (

<View>

<Text>Home Screen</Text>

</View>

);

}

function ProfileScreen() {

return (

<View>

<Text>Profile Screen</Text>

</View>

);

}

// Export the main app function as the default export

export default function App() {

return (

<NavigationContainer>

<Stack.Navigator>

<Stack.Screen

name="Home"

component={HomeScreen}

options={{ title: 'Welcome' }}

/>

<Stack.Screen name="Profile" component={ProfileScreen} />

</Stack.Navigator>

</NavigationContainer>

);

}

**VS**

import \* as React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-navigation/native-stack';

// Define the stack navigator

const Stack = createNativeStackNavigator();

// Define the screens/components

function HomeScreen() {

return (

<View>

<Text>Home Screen</Text>

</View>

);

}

function ProfileScreen() {

return (

<View>

<Text>Profile Screen</Text>

</View>

);

}

// Define the stack navigator inside MyStack

const MyStack = () => {

return (

<NavigationContainer>

<Stack.Navigator>

<Stack.Screen

name="Home"

component={HomeScreen}

options={{ title: 'Welcome' }}

/>

<Stack.Screen name="Profile" component={ProfileScreen} />

</Stack.Navigator>

</NavigationContainer>

);

};

// Export the MyStack component (either as default or named export)

export\ default MyStack;

what is the difference between these and when to use which

also since stack stores the screen or pages in a stack format i.e. on top of each other what if I want to jump to screens

why not always use web sockets?

What kind of data can web sockets handle?

**Documentation**

**Introduction**

This React Native application provides a structured solution for user authentication and navigation between different functionalities, such as viewing graphs, uploading files, using the camera, and logging out. It utilizes a drawer navigator for better user experience and integrates authentication checks.

**Technologies Used**

* React Native
* Expo
* @react-navigation/native
* expo-camera
* expo-media-library

**Features**

* **Login Authentication**: Users can log in with a username and password.
* **Drawer Navigation**: Navigate seamlessly between different screens like Home, Graphs, Upload Files, and Camera.
* **Logout Functionality**: Users can securely log out.
* **Camera Integration**: Capture photos/videos using the device camera.
* **Graph Viewer**: Navigate to the graph viewing screen.

**Prerequisites**

Before running the project, ensure you have the following installed:

* Node.js (v16 or higher)
* Expo CLI

**Components Overview**

**App.jsx**

* Sets up the main navigation for the app, including:
  + **Drawer Navigator**: Includes Home, Graphs, Upload Files, and Camera screens.
  + **Stack Navigator**: Handles transitions between login and main app screens.
* Checks user login state during initialization using SessionManagement.js.

**SessionManagement.js**

* Provides utility functions checkLogin and logout to manage user sessions.

**Screens Description**

**LoginScreen.jsx**

* Allows users to enter credentials and log in.
* Validates credentials using the loginn function from client.js.
* Displays an activity indicator during login.

**HomeScreen.jsx**

* Main screen after login.
* Provides buttons to:
  + Navigate to Graphs
  + Open Settings (placeholder alert)
  + Log out

**GraphScreen.jsx**

* Displays graphs (implementation pending or placeholder).
* Accessed via the drawer navigation.

**UploadFileScreen.jsx**

* Handles file uploads (implementation pending or placeholder).
* Accessed via the drawer navigation.

**CameraScreen.jsx**

* Integrates with expo-camera for capturing photos/videos.
* Accessed via the drawer navigation.

**Navigation Flow**

**Stack Navigator**

* **Login**: First screen if the user is not logged in.
* **Main**: Container for the Drawer Navigator.

**Drawer Navigator (Main)**

* **Home**: Default screen after login.
* **Graphs**: Displays the graphs screen.
* **Upload Files**: Opens the file upload screen.
* **Camera**: Opens the camera screen.

Any scripts that needs to be run to be mentioned not basic stuff like npm install

1. **What are the most important qualities of a successful client manager in your experience?**

Ans:

1. **What are some common challenges you face in managing client relationships?**

Ans:

1. **How do you build and maintain strong client relationships?**

Ans:

1. **What are some key strategies for effectively communicating with clients?**

Ans:

1. **How do you prioritize and manage multiple client projects simultaneously?**

Ans:

1. **How do you effectively manage client expectations throughout the project lifecycle?**

Ans:

1. **How do you handle difficult client conversations (e.g., budget , missed deadlines, client dissatisfaction)?**

Ans:

1. **How do you ensure that client feedback is effectively collected and incorporated into the project?**

Ans: