■ React Native/Expo Project Code Export

■ TypeScript/JavaScript Files & Configuration:

- [APP] app.json
- [TSX] app\(tabs)_layout.tsx
- [TSX] app\(tabs)\index.tsx
- [TSX] app\(tabs)\two.tsx
- [TSX] app\+html.tsx
- [TSX] app\+not-found.tsx
- [TSX] app_layout.tsx
- [TSX] app\modal.tsx
- [JS] babel.config.js
- [TSX] components\EditScreenInfo.tsx
- [TSX] components\ExternalLink.tsx
- [TSX] components\StyledText.tsx
- [TSX] components\Themed.tsx
- [TS] components\useClientOnlyValue.ts
- [TS] components\useClientOnlyValue.web.ts
- [TS] components\useColorScheme.ts
- [TS] components\useColorScheme.web.ts
- [TS] constants\Colors.ts
- [DTS] expo-env.d.ts
- [JS] metro.config.js
- [DTS] nativewind-env.d.ts
- [PKG] package.json
- [JS] tailwind.config.js
- [TSC] tsconfig.json

■ File: app.json

```
================================================================================
  1: {
  2:   "expo": {
  3:     "name": "Car_pooling_wizard",
  4:     "slug": "car-pooling-wizard",
  5:     "version": "1.0.0",
  6:     "orientation": "portrait",
  7:     "icon": "./assets/images/icon.png",
  8:     "scheme": "carpoolingwizard",
  9:     "userInterfaceStyle": "automatic",
 10:     "newArchEnabled": false,
 11:     "splash": {
 12:       "image": "./assets/images/splash-icon.png",
 13:       "resizeMode": "contain",
 14:       "backgroundColor": "#ffffff"
 15:     },
 16:     "ios": {
 17:       "supportsTablet": true,
 18:       "bundleIdentifier": "com.carpoolingwizard.app"
 19:     },
 20:     "android": {
 21:       "adaptiveIcon": {
 22:         "foregroundImage": "./assets/images/icon.png",
 23:         "backgroundColor": "#ffffff"
 24:       },
 25:       "package": "com.carpoolingwizard.app"
 26:     },
 27:     "web": {
 28:       "bundler": "metro",
 29:       "output": "static",
 30:       "favicon": "./assets/images/favicon.png"
 31:     },
 32:     "plugins": [
 33:       "expo-router"
 34:     ],
 35:     "experiments": {
 36:       "typedRoutes": true
 37:     }
 38:   }
 39: }
```

■ File: app\(tabs)_layout.tsx

```
================================================================================
  1: import React from 'react';
  2: import FontAwesome from '@expo/vector-icons/FontAwesome';
  3: import { Link, Tabs } from 'expo-router';
  4: import { Pressable, View } from 'react-native';
  5:
  6: import Colors from '@/constants/Colors';
  7: import { useColorScheme } from '@/components/useColorScheme';
  8: import { useClientOnlyValue } from '@/components/useClientOnlyValue';
  9:
 10: function TabBarIcon(props: {
 11:   name: React.ComponentProps<typeof FontAwesome>['name'];
 12:   color: string;
 13: }) {
 14:   return <FontAwesome size={28} style={{ marginBottom: -3 }} {...props} />;
 15: }
 16:
 17: export default function TabLayout() {
 18:   const colorScheme = useColorScheme();
 19:
 20:   return (
 21:     <Tabs
 22:       screenOptions={{
 23:         tabBarActiveTintColor: Colors[colorScheme ?? 'light'].tint,
 24:         headerShown: useClientOnlyValue(false, true),
 25:       }}>
 26:       <Tabs.Screen
 27:         name="index"
```

```
        options={{
          title: 'Tab One',
          tabBarIcon: ({ color }) => <TabBarIcon name="code" color={color} />,
          headerRight: () => (
            <Link href="/modal" asChild>
              <Pressable>
                {({ pressed }) => (
                  <View className="mr-4">
                    <FontAwesome
                      name="info-circle"
                      size={25}
                      color={Colors[colorScheme ?? 'light'].text}
                      style={{ opacity: pressed ? 0.5 : 1 }}
                    />
                  </View>
                )}
              </Pressable>
            </Link>
          ),
        }}
      />
      <Tabs.Screen
        name="two"
        options={{
          title: 'Tab Two',
          tabBarIcon: ({ color }) => <TabBarIcon name="code" color={color} />,
        }}
      />
    </Tabs>
  );
}
```

■ File: app\(tabs)\index.tsx

```
import { View, Text, TouchableOpacity } from 'react-native';
import { useState } from 'react';

export default function TabOneScreen() {
  const [count, setCount] = useState(0);

  return (
    <View className="flex-1 items-center justify-center bg-blue-500">
      <View className="bg-white p-8 rounded-2xl shadow-lg">
        <Text className="text-4xl font-bold text-blue-600 mb-4 text-center">
          Tailwind Test ?
        </Text>

        <Text className="text-xl text-gray-700 mb-6 text-center">
          Count: {count}
        </Text>

        <TouchableOpacity
          onPress={() => setCount(count + 1)}
          className="bg-blue-500 px-6 py-4 rounded-lg active:bg-blue-600">
          <Text className="text-white font-semibold text-center text-lg">
            Click Me!
          </Text>
        </TouchableOpacity>

        <TouchableOpacity
          onPress={() => setCount(0)}
          className="bg-red-500 px-6 py-4 rounded-lg mt-3 active:bg-red-600">
          <Text className="text-white font-semibold text-center text-lg">
            Reset
          </Text>
        </TouchableOpacity>
      </View>

      <View className="mt-8">
        <Text className="text-white text-lg font-bold mb-2">
          ? Colors Working
```

```
        </Text>
        <Text className="text-yellow-300 text-lg font-bold mb-2">
          ? Spacing Working
        </Text>
        <Text className="text-green-300 text-lg font-bold">
          ? Styles Working
        </Text>
      </View>
    </View>
  );
}
```

■ File: app\(tabs)\two.tsx

```
import EditScreenInfo from '@/components/EditScreenInfo';
import { Text, View } from '@/components/Themed';

export default function TabTwoScreen() {
  return (
    <View className="flex-1 items-center justify-center">
      <Text className="text-xl font-bold">Tab Two</Text>
      <View
        className="my-7 h-[1px] w-[80%]"
        lightColor="#eee"
        darkColor="rgba(255,255,255,0.1)"
      />
      <EditScreenInfo path="app/(tabs)/two.tsx" />
    </View>
  );
}
```

■ File: app\+html.tsx

```
import { ScrollViewStyleReset } from 'expo-router/html';

// This file is web-only and used to configure the root HTML for every
// web page during static rendering.
// The contents of this function only run in Node.js environments and
// do not have access to the DOM or browser APIs.
export default function Root({ children }: { children: React.ReactNode }) {
  return (
    <html lang="en">
      <head>
        <meta charSet="utf-8" />
        <meta httpEquiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

        {/*
          Disable body scrolling on web. This makes ScrollView components work closer to how they do on nat
          However, body scrolling is often nice to have for mobile web. If you want to enable it, remove th
        */}
        <ScrollViewStyleReset />

        {/* Using raw CSS styles as an escape-hatch to ensure the background color never flickers in dark-m
        <style dangerouslySetInnerHTML={{ __html: responsiveBackground }} />
        {/* Add any additional <head> elements that you want globally available on web... */}
      </head>
      <body>{children}</body>
    </html>
  );
}

const responsiveBackground = `
body {
  background-color: #fff;
}
@media (prefers-color-scheme: dark) {
  body {
    background-color: #000;
```

```
 37:   }
 38: }`;
```

■ File: app\+not-found.tsx

```
  1: import { Link, Stack } from 'expo-router';
  2:
  3: import { Text, View } from '@/components/Themed';
  4:
  5: export default function NotFoundScreen() {
  6:   return (
  7:     <>
  8:       <Stack.Screen options={{ title: 'Oops!' }} />
  9:       <View className="flex-1 items-center justify-center p-5">
 10:         <Text className="text-xl font-bold">This screen doesn't exist.</Text>
 11:
 12:         <Link href="/" className="mt-4 py-4">
 13:           <Text className="text-sm text-[#2e78b7]">Go to home screen!</Text>
 14:         </Link>
 15:       </View>
 16:     </>
 17:   );
 18: }
```

■ File: app_layout.tsx

```
  1: import "../global.css";
  2: import FontAwesome from '@expo/vector-icons/FontAwesome';
  3: import { DarkTheme, DefaultTheme, ThemeProvider } from '@react-navigation/native';
  4: import { useFonts } from 'expo-font';
  5: import { Stack } from 'expo-router';
  6: import * as SplashScreen from 'expo-splash-screen';
  7: import { useEffect } from 'react';
  8: import 'react-native-reanimated';
  9:
 10: import { useColorScheme } from '@/components/useColorScheme';
 11:
 12: export {
 13:   ErrorBoundary,
 14: } from 'expo-router';
 15:
 16: export const unstable_settings = {
 17:   initialRouteName: '(tabs)',
 18: };
 19:
 20: SplashScreen.preventAutoHideAsync();
 21:
 22: export default function RootLayout() {
 23:   const [loaded, error] = useFonts({
 24:     SpaceMono: require('../assets/fonts/SpaceMono-Regular.ttf'),
 25:     ...FontAwesome.font,
 26:   });
 27:
 28:   useEffect(() => {
 29:     if (error) throw error;
 30:   }, [error]);
 31:
 32:   useEffect(() => {
 33:     if (loaded) {
 34:       SplashScreen.hideAsync();
 35:     }
 36:   }, [loaded]);
 37:
 38:   if (!loaded) {
 39:     return null;
 40:   }
 41:
 42:   return <RootLayoutNav />;
 43: }
```

```
 44:
 45: function RootLayoutNav() {
 46:   const colorScheme = useColorScheme();
 47:
 48:   return (
 49:     <ThemeProvider value={colorScheme === 'dark' ? DarkTheme : DefaultTheme}>
 50:       <Stack>
 51:         <Stack.Screen name="(tabs)" options={{ headerShown: false }} />
 52:         <Stack.Screen name="modal" options={{ presentation: 'modal' }} />
 53:       </Stack>
 54:     </ThemeProvider>
 55:   );
 56: }
```

■ File: app\modal.tsx

```
  1: import { StatusBar } from 'expo-status-bar';
  2: import { Platform } from 'react-native';
  3:
  4: import EditScreenInfo from '@/components/EditScreenInfo';
  5: import { Text, View } from '@/components/Themed';
  6:
  7: export default function ModalScreen() {
  8:   return (
  9:     <View className="flex-1 items-center justify-center">
 10:       <Text className="text-xl font-bold">Modal</Text>
 11:       <View
 12:         className="my-7 h-[1px] w-[80%]"
 13:         lightColor="#eee"
 14:         darkColor="rgba(255,255,255,0.1)"
 15:       />
 16:       <EditScreenInfo path="app/modal.tsx" />
 17:
 18:       <StatusBar style={Platform.OS === 'ios' ? 'light' : 'auto'} />
 19:     </View>
 20:   );
 21: }
```

■ File: babel.config.js

```
  1: module.exports = function(api) {
  2:   api.cache(true);
  3:   return {
  4:     presets: ['babel-preset-expo'],
  5:     plugins: ['nativewind/babel', 'react-native-reanimated/plugin']
  6:   };
  7: };
```

■ File: components\EditScreenInfo.tsx

```
  1: import React from 'react';
  2:
  3: import { ExternalLink } from './ExternalLink';
  4: import { MonoText } from './StyledText';
  5: import { Text, View } from './Themed';
  6:
  7: import Colors from '@/constants/Colors';
  8:
  9: export default function EditScreenInfo({ path }: { path: string }) {
 10:   return (
 11:     <View>
 12:       <View className="items-center mx-12">
 13:         <Text
 14:           className="text-[17px] leading-6 text-center"
 15:           lightColor="rgba(0,0,0,0.8)"
 16:           darkColor="rgba(255,255,255,0.8)">
 17:           Open up the code for this screen:
```

```
 18:         </Text>
 19:
 20:         <View
 21:           className="rounded-[3px] px-1 my-2"
 22:           darkColor="rgba(255,255,255,0.05)"
 23:           lightColor="rgba(0,0,0,0.05)">
 24:           <MonoText>{path}</MonoText>
 25:         </View>
 26:
 27:         <Text
 28:           className="text-[17px] leading-6 text-center"
 29:           lightColor="rgba(0,0,0,0.8)"
 30:           darkColor="rgba(255,255,255,0.8)">
 31:           Change any of the text, save the file, and your app will automatically update.
 32:         </Text>
 33:       </View>
 34:
 35:       <View className="mt-4 mx-5 items-center">
 36:         <ExternalLink
 37:           className="py-4"
 38:           href="https://docs.expo.io/get-started/create-a-new-app/#opening-the-app-on-your-phonetablet">
 39:           <Text className="text-center" lightColor={Colors.light.tint}>
 40:             Tap here if your app doesn't automatically update after making changes
 41:           </Text>
 42:         </ExternalLink>
 43:       </View>
 44:     </View>
 45:   );
 46: }
```

■ File: components\ExternalLink.tsx

```
  1: import { Link } from 'expo-router';
  2: import * as WebBrowser from 'expo-web-browser';
  3: import React from 'react';
  4: import { Platform } from 'react-native';
  5:
  6: export function ExternalLink(
  7:   props: Omit<React.ComponentProps<typeof Link>, 'href'> & { href: string }
  8: ) {
  9:   return (
 10:     <Link
 11:       target="_blank"
 12:       {...props}
 13:       // @ts-expect-error: External URLs are not typed.
 14:       href={props.href}
 15:       onPress={(e) => {
 16:         if (Platform.OS !== 'web') {
 17:           // Prevent the default behavior of linking to the default browser on native.
 18:           e.preventDefault();
 19:           // Open the link in an in-app browser.
 20:           WebBrowser.openBrowserAsync(props.href as string);
 21:         }
 22:       }}
 23:     />
 24:   );
 25: }
```

■ File: components\StyledText.tsx

```
  1: import { Text, TextProps } from './Themed';
  2:
  3: export function MonoText(props: TextProps) {
  4:   return <Text {...props} style={[props.style, { fontFamily: 'SpaceMono' }]} />;
  5: }
```

■ File: components\Themed.tsx

```
  1: import { Text as DefaultText, View as DefaultView } from 'react-native';
  2:
  3: import Colors from '@/constants/Colors';
  4: import { useColorScheme } from './useColorScheme';
  5:
  6: type ThemeProps = {
  7:   lightColor?: string;
  8:   darkColor?: string;
  9: };
 10:
 11: export type TextProps = ThemeProps & DefaultText['props'];
 12: export type ViewProps = ThemeProps & DefaultView['props'];
 13:
 14: export function useThemeColor(
 15:   props: { light?: string; dark?: string },
 16:   colorName: keyof typeof Colors.light & keyof typeof Colors.dark
 17: ) {
 18:   const theme = useColorScheme() ?? 'light';
 19:   const colorFromProps = props[theme];
 20:
 21:   if (colorFromProps) {
 22:     return colorFromProps;
 23:   } else {
 24:     return Colors[theme][colorName];
 25:   }
 26: }
 27:
 28: export function Text(props: TextProps) {
 29:   const { style, lightColor, darkColor, ...otherProps } = props;
 30:   const color = useThemeColor({ light: lightColor, dark: darkColor }, 'text');
 31:
 32:   return <DefaultText style={[{ color }, style]} {...otherProps} />;
 33: }
 34:
 35: export function View(props: ViewProps) {
 36:   const { style, lightColor, darkColor, ...otherProps } = props;
 37:   const backgroundColor = useThemeColor({ light: lightColor, dark: darkColor }, 'background');
 38:
 39:   return <DefaultView style={[{ backgroundColor }, style]} {...otherProps} />;
 40: }
```

■ File: components\useClientOnlyValue.ts

```
  1: // This function is web-only as native doesn't currently support server (or build-time) rendering.
  2: export function useClientOnlyValue<S, C>(server: S, client: C): S | C {
  3:   return client;
  4: }
```

■ File: components\useClientOnlyValue.web.ts

```
  1: import React from 'react';
  2:
  3: // `useEffect` is not invoked during server rendering, meaning
  4: // we can use this to determine if we're on the server or not.
  5: export function useClientOnlyValue<S, C>(server: S, client: C): S | C {
  6:   const [value, setValue] = React.useState<S | C>(server);
  7:   React.useEffect(() => {
  8:     setValue(client);
  9:   }, [client]);
 10:
 11:   return value;
 12: }
```

■ File: components\useColorScheme.ts

```
  1: export { useColorScheme } from 'react-native';
```

■ File: components\useColorScheme.web.ts

```
  1: // NOTE: The default React Native styling doesn't support server rendering.
  2: // Server rendered styles should not change between the first render of the HTML
  3: // and the first render on the client. Typically, web developers will use CSS media queries
  4: // to render different styles on the client and server, these aren't directly supported in React Native
  5: // but can be achieved using a styling library like Nativewind.
  6: export function useColorScheme() {
  7:   return 'light';
  8: }
```

■ File: constants\Colors.ts

```
  1: const tintColorLight = '#2f95dc';
  2: const tintColorDark = '#fff';
  3:
  4: export default {
  5:   light: {
  6:     text: '#000',
  7:     background: '#fff',
  8:     tint: tintColorLight,
  9:     tabIconDefault: '#ccc',
 10:     tabIconSelected: tintColorLight,
 11:   },
 12:   dark: {
 13:     text: '#fff',
 14:     background: '#000',
 15:     tint: tintColorDark,
 16:     tabIconDefault: '#ccc',
 17:     tabIconSelected: tintColorDark,
 18:   },
 19: };
```

■ File: expo-env.d.ts

```
  1: /// <reference types="expo/types" />
  2:
  3: // NOTE: This file should not be edited and should be in your git ignore
```

■ File: metro.config.js

```
  1: const { withNativeWind } = require("nativewind/metro");
  2:
  3: module.exports = withNativeWind({
  4:   resolver: {},
  5: });
```

■ File: nativewind-env.d.ts

```
  1: /// <reference types="nativewind/types" />
```

■ File: package.json

```
  1: {
  2:   "name": "car_pooling_wizard",
  3:   "main": "expo-router/entry",
  4:   "version": "1.0.0",
```

```
  5:   "scripts": {
  6:     "start": "expo start",
  7:     "android": "expo start --android",
  8:     "ios": "expo start --ios",
  9:     "web": "expo start --web",
 10:     "test": "jest --watchAll"
 11:   },
 12:   "jest": {
 13:     "preset": "jest-expo"
 14:   },
 15:   "dependencies": {
 16:     "@expo/vector-icons": "~14.0.4",
 17:     "@react-navigation/native": "^7.0.14",
 18:     "expo": "~52.0.47",
 19:     "expo-font": "~13.0.4",
 20:     "expo-linking": "~7.0.5",
 21:     "expo-router": "~4.0.21",
 22:     "expo-splash-screen": "~0.29.24",
 23:     "expo-status-bar": "~2.0.1",
 24:     "expo-system-ui": "~4.0.9",
 25:     "expo-web-browser": "~14.0.2",
 26:     "nativewind": "^2.0.11",
 27:     "react": "18.3.1",
 28:     "react-dom": "18.3.1",
 29:     "react-native": "0.76.9",
 30:     "react-native-reanimated": "~3.16.1",
 31:     "react-native-safe-area-context": "4.12.0",
 32:     "react-native-screens": "~4.4.0",
 33:     "react-native-web": "~0.19.13",
 34:     "tailwindcss": "^3.3.2"
 35:   },
 36:   "devDependencies": {
 37:     "@babel/core": "^7.25.2",
 38:     "@types/react": "~18.3.12",
 39:     "jest": "^29.2.1",
 40:     "jest-expo": "~52.0.6",
 41:     "react-test-renderer": "18.3.1",
 42:     "typescript": "~5.3.3"
 43:   },
 44:   "private": true
 45: }
```

■ File: tailwind.config.js

```
  1: /** @type {import('tailwindcss').Config} */
  2: module.exports = {
  3:   content: [
  4:     "./app/**/*.{js,jsx,ts,tsx}",
  5:     "./components/**/*.{js,jsx,ts,tsx}",
  6:   ],
  7:   presets: [require("nativewind/preset")],
  8:   theme: {
  9:     extend: {},
 10:   },
 11:   plugins: [],
 12: };
```

■ File: tsconfig.json

```
  1: {
  2:   "extends": "expo/tsconfig.base",
  3:   "compilerOptions": {
  4:     "strict": true,
  5:     "paths": {
  6:       "@/*": ["./*"]
  7:     }
  8:   },
  9:   "include": [
 10:     "**/*.ts",
```

```
    "**/*.tsx",
    ".expo/types/**/*.ts",
    "expo-env.d.ts",
    "nativewind-env.d.ts"
  ]
}
```
