Building a Livestream Shopping App with React Native

MAY 2023

Overview

What is NTWRK?

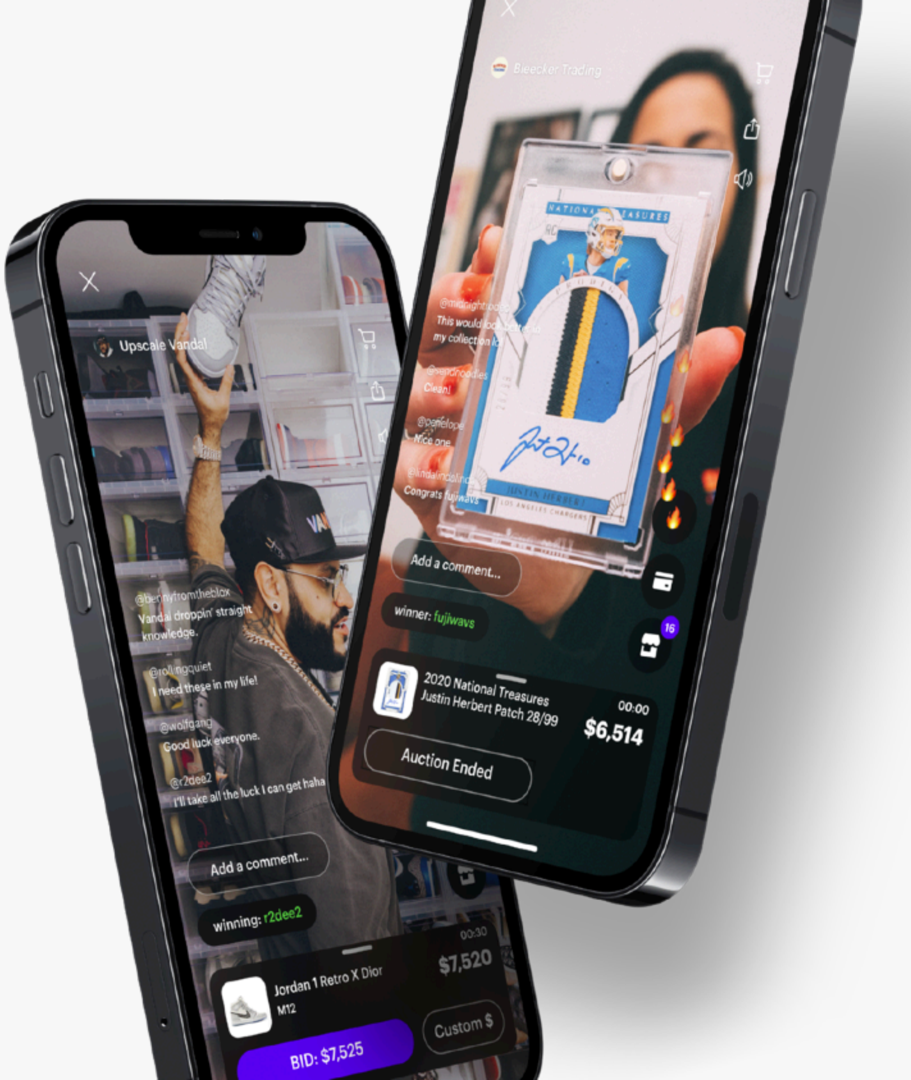
How are we using React Native?

What were some learnings?

What is NTWRK?

NTWRK is an online marketplace where businesses build shopping-oriented communities around product categories such as Sneakers, Fashion, Designer Toys, Art, Music, and Trading Cards.

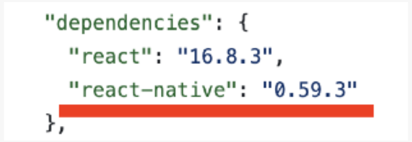
Through the NTWRK app, we're providing sellers the tools to grow their brands with a unique set of social commerce functionality.



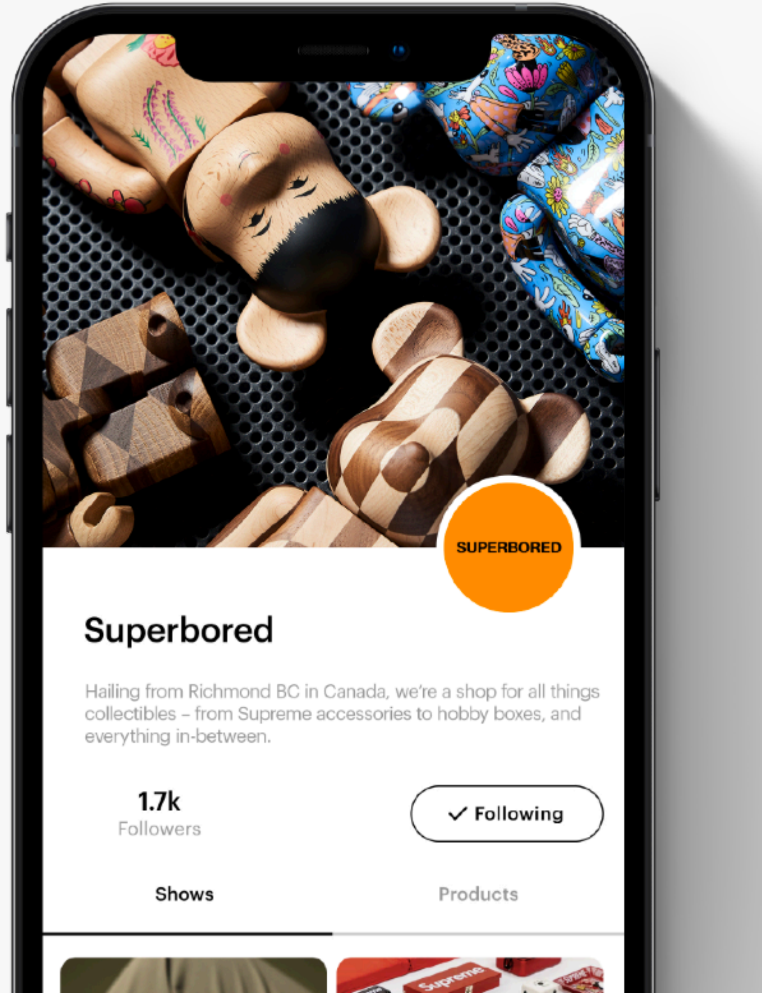
Native Mobile First

Started as a shopping-only Swift app in 2018 then pivoted to React Native in mid-2019 to target both iOS and Android.

Expanded to include entire feature set for businesses to set up and sell products live in 2020.



```
"dependencies": {
  "react": "16.8.3",
  "react-native": "0.59.3"
},
```



Superbored

Hailing from Richmond BC in Canada, we're a shop for all things collectibles – from Supreme accessories to hobby boxes, and everything in-between.

Why React Native?

We're a start-up. We need to move fast.

Time To Market
- ~4-8 devs work on mobile.
- Feature set + release cadence is identical for iOS, Android.

Leveraging Existing Experience + Ecosystem
- Devs primarily come from a TypeScript, React background.
- Robust and growing ecosystem of open source in RN community.

Live Shopping + React Native

Live Show UI

Features

Notifications

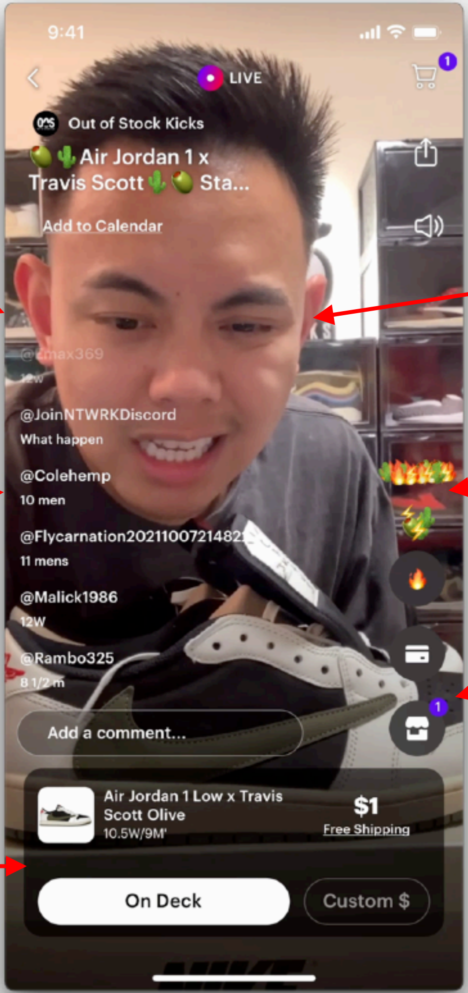
Chat

Current Auction

Video

Reactions

Product Catalog



Out of Stock Kicks

🍏🌵Air Jordan 1 x Travis Scott🌵🍏 Sta...

Add to Calendar

@max369
12W

@JoinNTWRKDiscord
What happen

@Colehemp
10 men

@Flycarnation20211007214821
11 mens

@Malick1986
12W

@Rambo325
8 1/2 m

Add a comment...

Air Jordan 1 Low x Travis Scott Olive
10.5W/9M'

$1
Free Shipping

Live Show UI

Implementation

Pusher (3rd party)
`pusher-js`

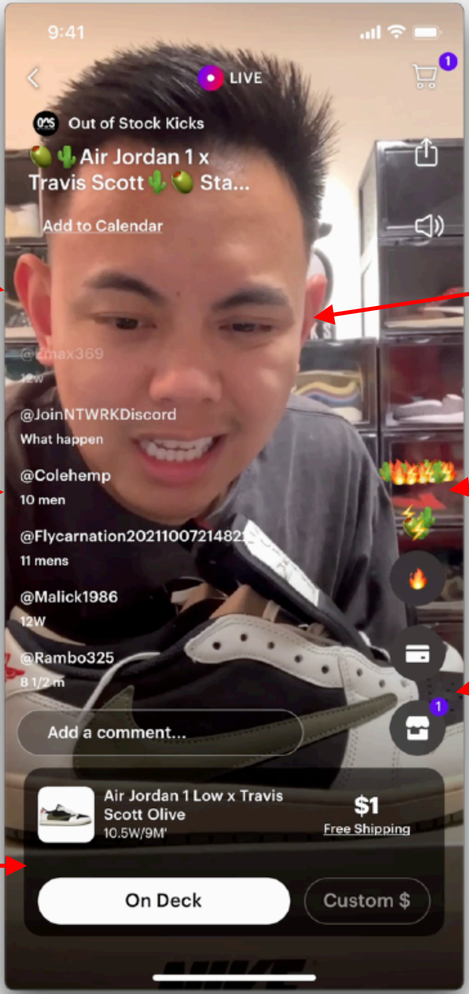
Stream (3rd party)
`stream-chat-react-native`

GraphQL, Pusher

Agora WebRTC
(3rd party)
`react-native-agora`

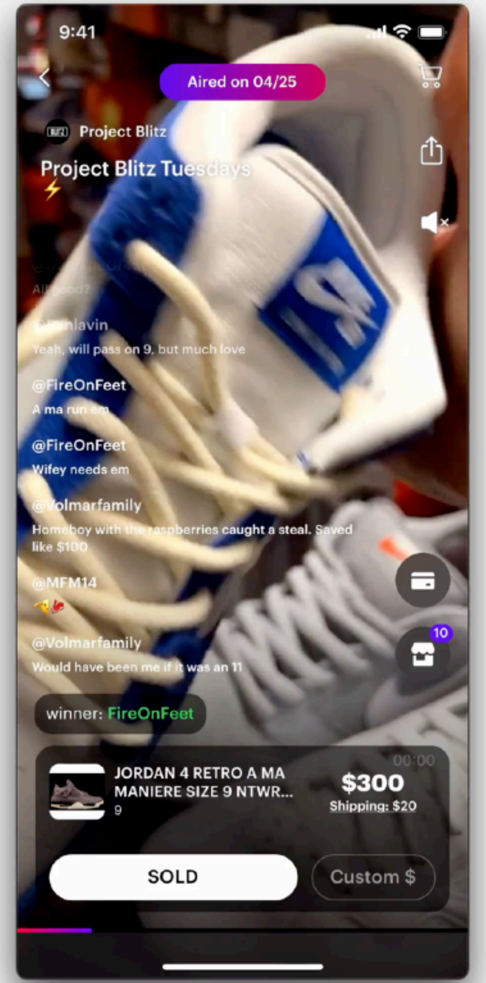
WebSocket

GraphQL, Pusher
`@apollo/client`



Deep Dive: Product Catalog

Packages + Routing



Project Blitz Tuesdays ⚡

@...lavin
Yeah, will pass on 9, but much love

@FireOnFeet
A ma run em

@FireOnFeet
Wifey needs em

@Volmarfamily
Homeboy with the raspberries caught a steal. Saved like $100

@MFM14

@Volmarfamily
Would have been me if it was an 11

winner: FireOnFeet

JORDAN 4 RETRO A MA MANIERE SIZE 9 NTWR...
9
$300
Shipping: $20

Packages

@react-navigation/stack

– app-wide routing + screen components

– used for nested navigation inside ‘product catalog’ bottom sheet.

@gorhom/bottom-sheet

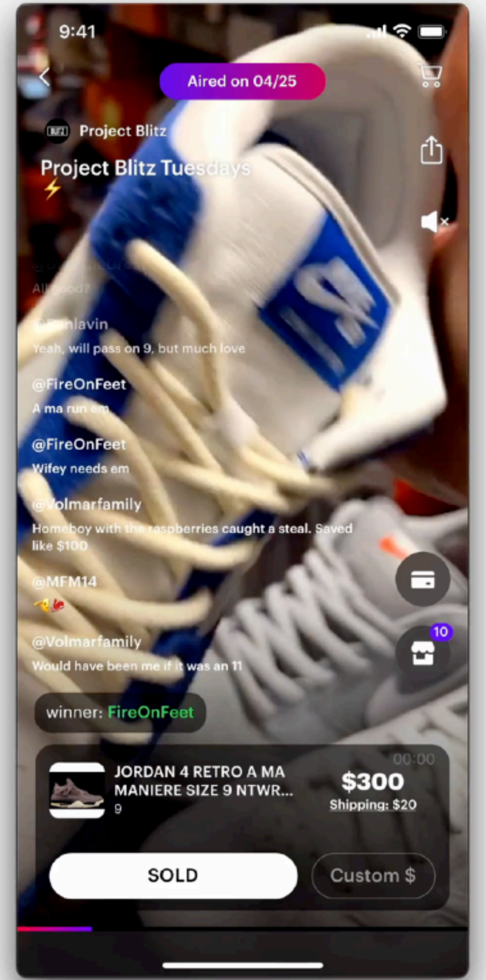
– renders the ‘bottom sheet’ with excellent interaction handling.

react-native-tab-view

– manages the ‘buy now’ + ‘auction’ tabbed list views.

jotai

– state management (passing values, refs, etc. around)



All good?

@…nlavin
Yeah, will pass on 9, but much love

@FireOnFeet
A ma run em

@FireOnFeet
Wifey needs em

@Volmarfamily
Homeboy with the raspberries caught a steal. Saved like $100

@MFM14
👈🌶

@Volmarfamily
Would have been me if it was an 11

winner: FireOnFeet

JORDAN 4 RETRO A MA MANIERE SIZE 9 NTWR...
9
00:00
$300
Shipping: $20

Package Nesting in Component Tree

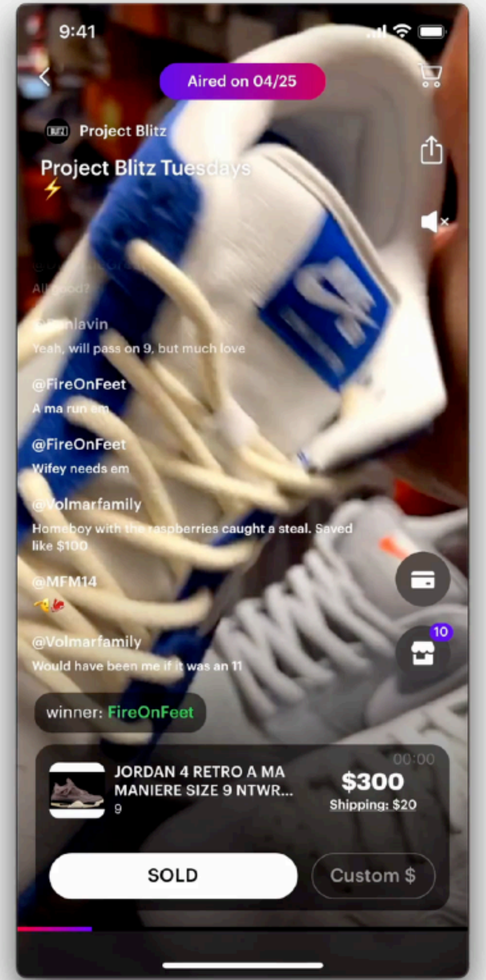
react-navigation > bottom-sheet > react-navigation > react-native-tab-view

```tsx
// App.tsx
<DeepLinkSubscriber> {/* legacy – captures external deeplinks and calls `navigate`
  ...
  <NavigationContainer> {/* app root nav container */}
    ...
    <AppStack.Navigator>
      ...
      <AppStack.Screen name="show" component={ShowScreen} />
    </AppStack.Navigator>
    <GlobalVideoPlayer />
  </NavigationContainer>
</DeepLinkSubscriber>
```

This screen component just sets some state, captures route params, etc.

The entire show is rendered in a component that can go into an in-app PiP mode. This makes it visible above all other screens.

```tsx
<>
  ...
  {displayCatalog && <Catalog onClose={closeCatalog} />}
</>
```



All good?

@...nlavin
Yeah, will pass on 9, but much love

@FireOnFeet
A ma run em

@FireOnFeet
Wifey needs em

@Volmarfamily
Homeboy with the raspberries caught a steal. Saved like $100

@MFM14

@Volmarfamily
Would have been me if it was an 11

JORDAN 4 RETRO A MA MANIERE SIZE 9 NTWR...
9
00:00
$300
Shipping: $20

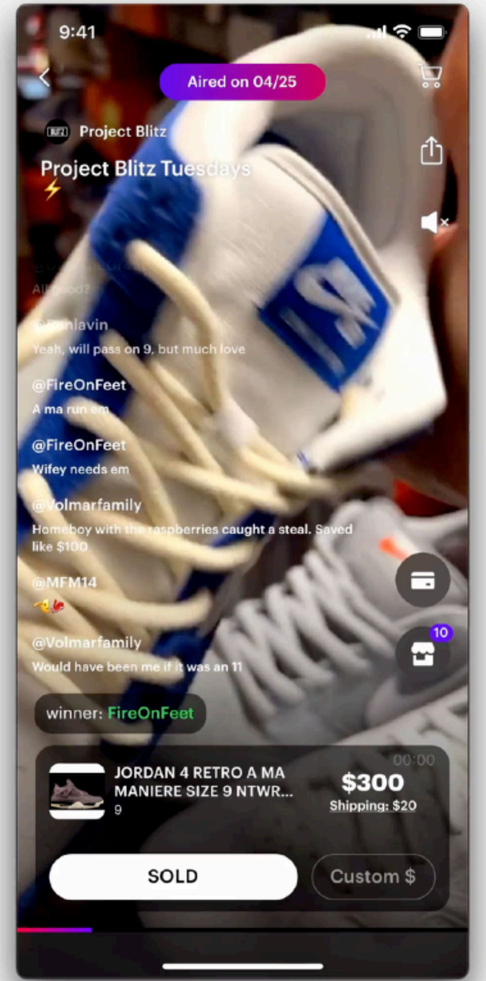
Package Nesting in Component Tree

react-navigation > **bottom-sheet > react-navigation > react-native-tab-view**

```
// Catalog.tsx
<BottomSheet>
  <NavigationContainer independent={true}>
    <Stack.Navigator>
      <Stack.Screen name="productList" component={ProductListScreen} />
      ...
    </Stack.Navigator>
  </NavigationContainer>
</BottomSheet>
```

bottom-sheet supports react-navigation
but only with an independent nav container!

```
// ProductList.tsx
<BottomSheetView>
  <Text>Products</Text>
  <TabView renderScene={tabLists} />
</BottomSheetView>

const tabLists = SceneMap({
  auction: AuctionList,
  buyNow: BuyNowList,
});
```



@...lavin
Yeah, will pass on 9, but much love
@FireOnFeet
A ma run em
@FireOnFeet
Wifey needs em
@Volmarfamily
Homeboy with the raspberries caught a steal. Saved like $100
@MFM14
@Volmarfamily
Would have been me if it was an 11

Everything works great! But…

How do we deep link into a screen nested in the bottom-sheet?

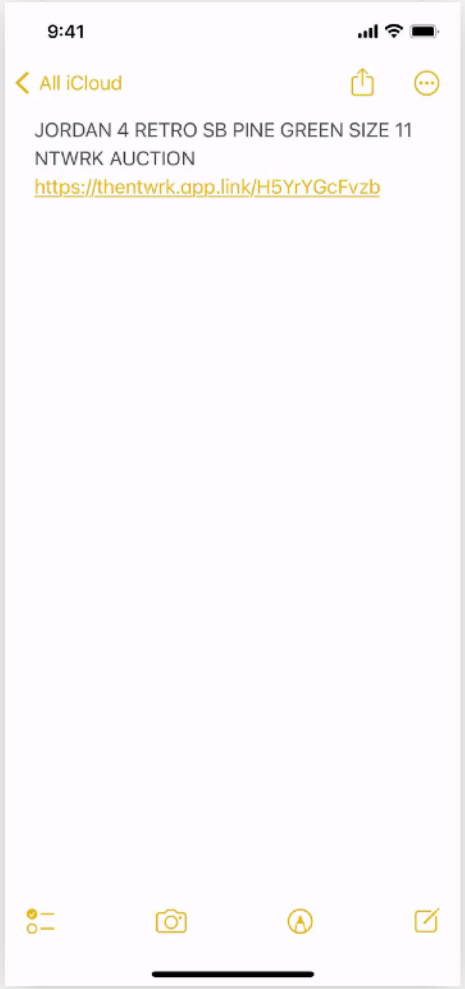
How do we route back out of one of these nested screens?

How to deeplink from the root?

Requires a small trick because of the ‘independent’ navigation container.
Independent = fully disconnected from parent containers

Step 1:
Capture the route params from the show screen once it’s mounted.

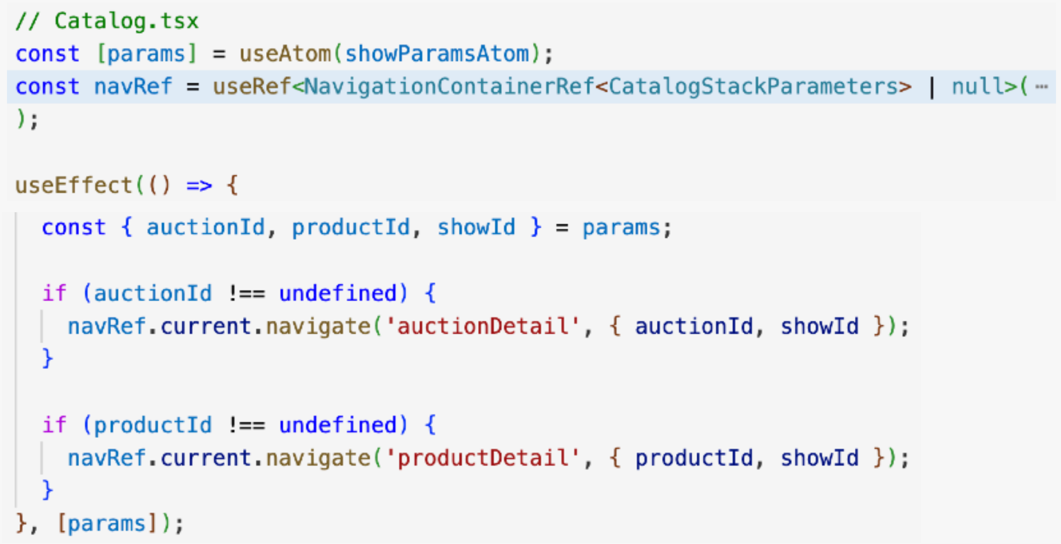
```
// Show.tsx
const route =
  useRoute<RouteProp<ApplicationStackParameters, ApplicationRoutes.SHOW>>();

useEffect(() => {
  setShowParams({ ...route.params });
  return () => setShowParams(null);
}, [route.params]);
```


9:41

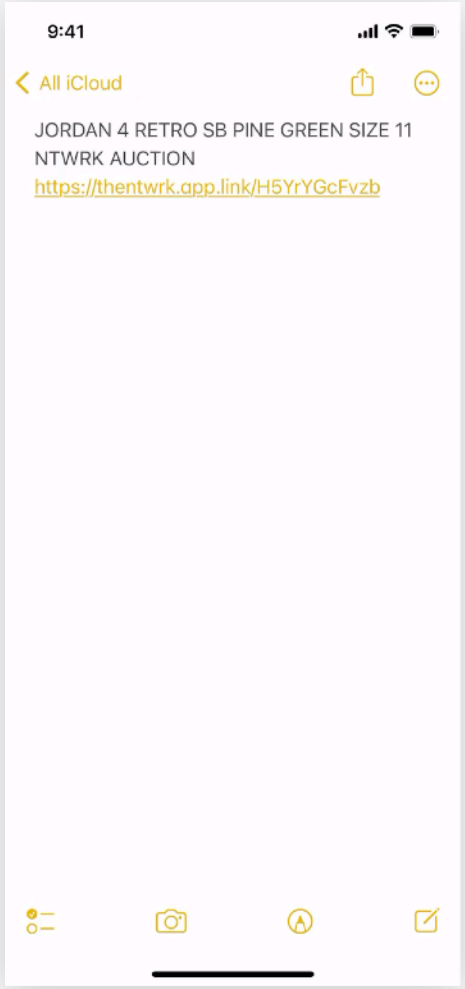
JORDAN 4 RETRO SB PINE GREEN SIZE 11
NTWRK AUCTION
https://thentwrk.app.link/H5YrYGcFvzb

How to deeplink from the root?

Step 2:
– If there are show params, display the catalog.
– On first render of the catalog, use those params to manually navigate to a specific screen.



```
// Catalog.tsx
const [params] = useAtom(showParamsAtom);
const navRef = useRef<NavigationContainerRef<CatalogStackParameters> | null>(⋯
);

useEffect(() => {
  const { auctionId, productId, showId } = params;

  if (auctionId !== undefined) {
    navRef.current.navigate('auctionDetail', { auctionId, showId });
  }

  if (productId !== undefined) {
    navRef.current.navigate('productDetail', { productId, showId });
  }
}, [params]);
```



JORDAN 4 RETRO SB PINE GREEN SIZE 11
NTWRK AUCTION
https://thentwrk.app.link/H5YrYGcFvzb

What about routing back out?

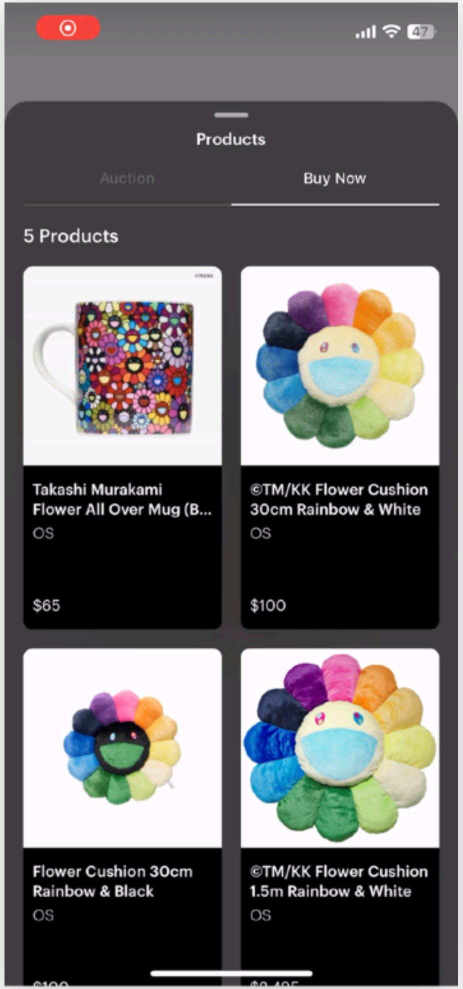
Independent navigator makes this tricky. Have to capture a reference to the parent nav container + bottom sheet to route back out.

```
// Catalog.tsx
const catalogRef = useRef<BottomSheet>(null);
useSetNestedNavigation(catalogRef);

// ...

<BottomSheet ref={catalogRef}>
  <NavigationContainer independent={true}>
```

Capture the sheet so that it can be closed when navigating outside of nested nav container.

```
// BuyNowButton.tsx
const navigate = useNestedNavigation();

const handleBuyNow = () => {
    navigate(ApplicationRoutes.BUY_NOW_CHECKOUT, {
      productID: productId,
      variantID: variantId,
    });
};
```

Rendered in a lot of places! Needs to handle both dismissing the sheet (if embedded inside one) or just routing directly to the 'buy now' screen.



5 Products

Takashi Murakami Flower All Over Mug (B...
OS
$65

©TM/KK Flower Cushion 30cm Rainbow & White
OS
$100

Flower Cushion 30cm Rainbow & Black
OS

©TM/KK Flower Cushion 1.5m Rainbow & White
OS

What about routing back out?

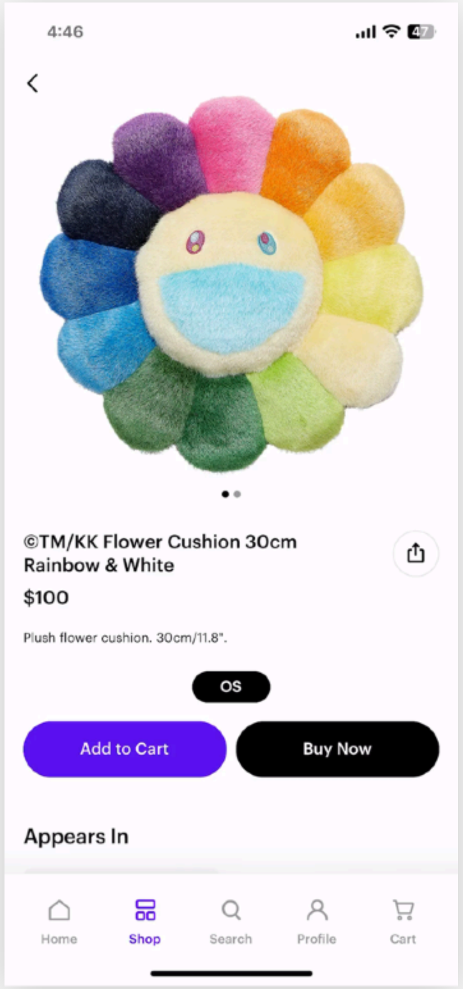
```
export const useSetNestedNavigation = (
  catalogRef: React.RefObject<BottomSheetMethods>,
) => {
  const setNavigate = useSetAtom(navigateAtom);
  const { navigate } = useNavigation();

  useEffect(() => {
    setNavigate({
      navigate: (...args: any) => {
        catalogRef.current?.close({
          duration: 250,
        });
        setTimeout(() => {
          navigate(...args);
        }, 250);
      },
    });
```

Capture the parent scope's navigate function.

Create a 'wrapped' navigate function that dismisses the sheet then calls the original navigate function.

Return wrapped navigate function if it exists. Otherwise return the current scope's navigate function.

```
export const useNestedNavigation = () => {
  const [externalNavigate] = useAtom(navigateAtom);
  const { navigate: internalNavigate } = useNavigation();
  return externalNavigate?.navigate ?? internalNavigate;
};
```

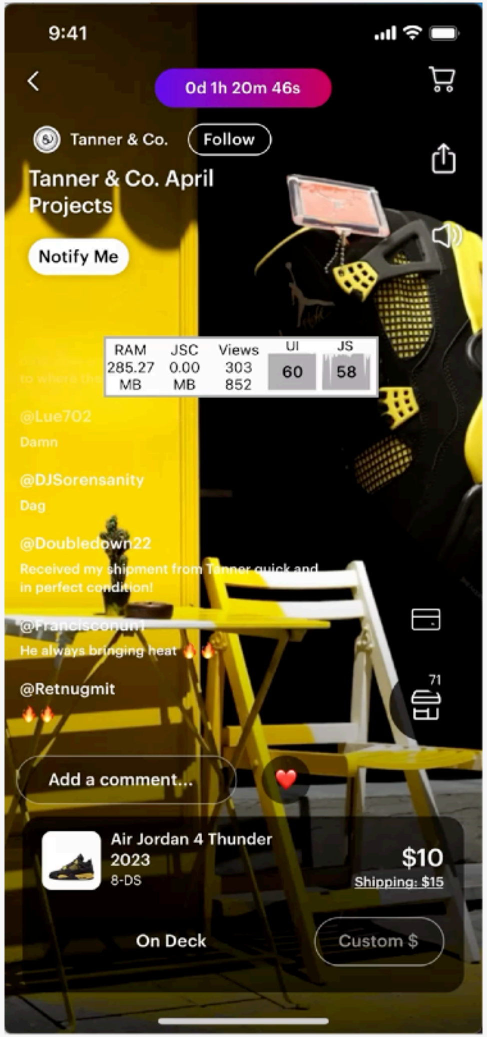

4:46
©TM/KK Flower Cushion 30cm Rainbow & White
$100
Plush flower cushion. 30cm/11.8".
OS
Add to Cart
Buy Now
Appears In
Home
Shop
Search
Profile
Cart

Key Takeaways

Always Monitor Performance

– Third-parties have rarely been the sole cause of perf issues.
– Self-inflicted through excessive re-rendering, poor usage of state / contexts.

Keep the FPS monitor on!
Use Flipper + Profiler to debug.



Tanner & Co. April Projects

RAM	JSC	Views	UI	JS
285.27 MB	0.00 MB	303 852	60	58

@Lue702
Damn

@DJSorensanity
Dag

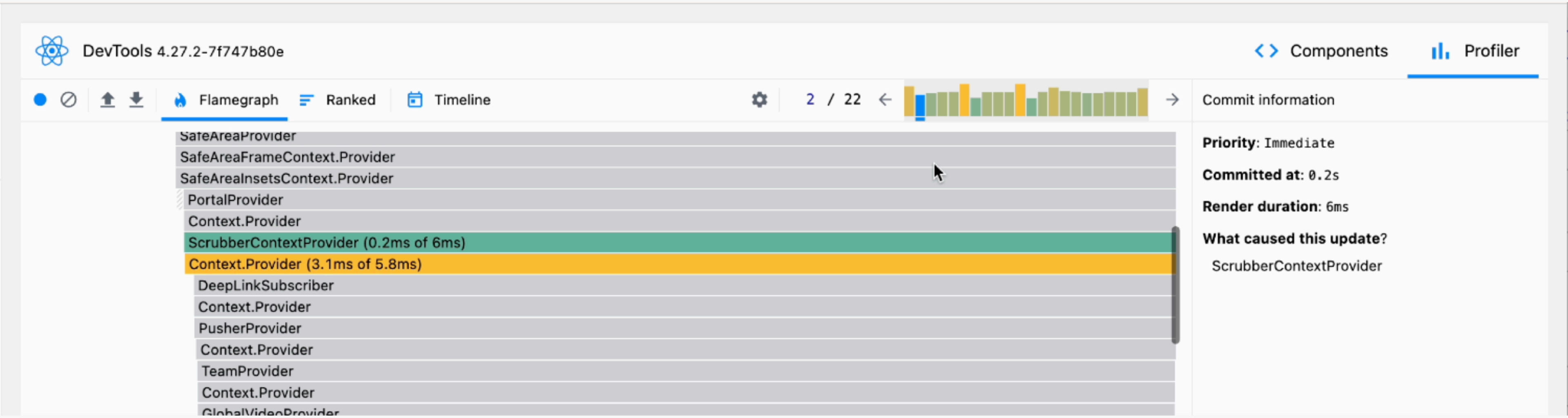
@Doubledown22
Received my shipment from Tanner quick and in perfect condition!

@Francisconun1
He always bringing heat 🔥🔥

@Retnugmit
🔥🔥

Air Jordan 4 Thunder 2023
8-DS
$10
Shipping: $15

Performance Monitoring With Flipper



SafeAreaProvider
SafeAreaFrameContext.Provider
SafeAreaInsetsContext.Provider
PortalProvider
Context.Provider
ScrubberContextProvider (0.2ms of 6ms)
Context.Provider (3.1ms of 5.8ms)
DeepLinkSubscriber
Context.Provider
PusherProvider
Context.Provider
TeamProvider
Context.Provider
GlobalVideoProvider

Priority: Immediate

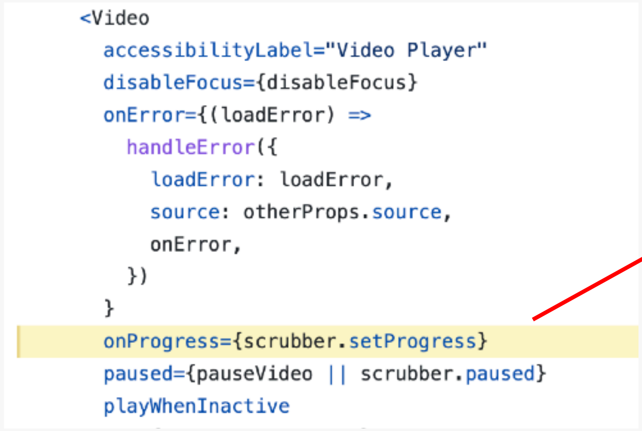
Committed at: 0.2s

Render duration: 6ms

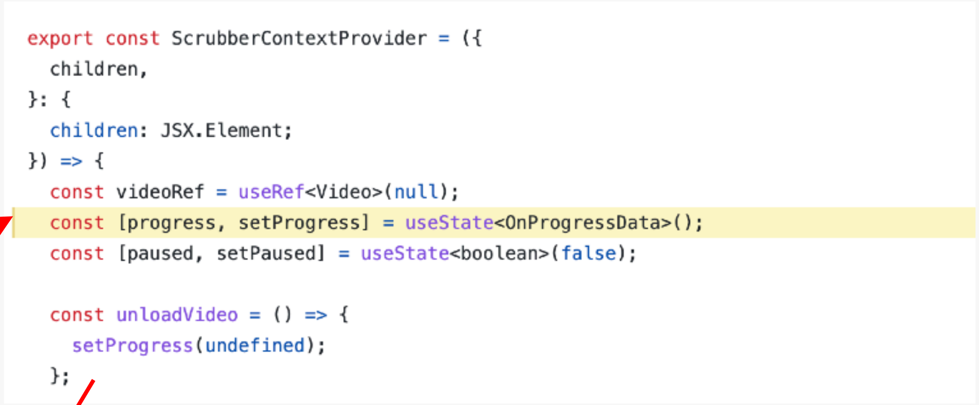
What caused this update?

ScrubberContextProvider

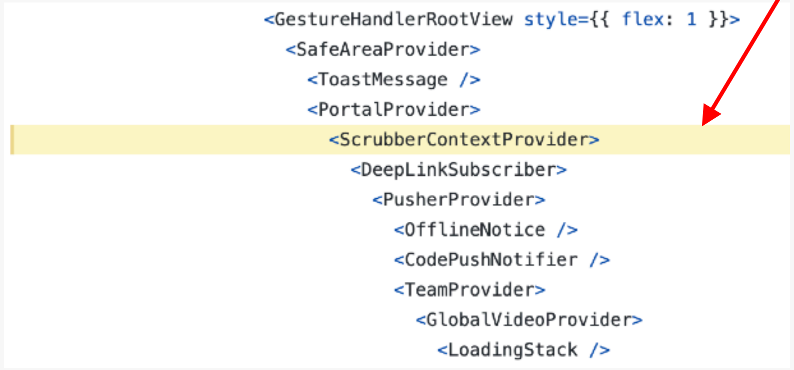
Performance Debugging



```
  <Video
    accessibilityLabel="Video Player"
    disableFocus={disableFocus}
    onError={(loadError) =>
      handleError({
        loadError: loadError,
        source: otherProps.source,
        onError,
      })
    }
    onProgress={scrubber.setProgress}
    paused={pauseVideo || scrubber.paused}
    playWhenInactive
```



```
export const ScrubberContextProvider = ({
  children,
}: {
  children: JSX.Element;
}) => {
  const videoRef = useRef<Video>(null);
  const [progress, setProgress] = useState<OnProgressData>();
  const [paused, setPaused] = useState<boolean>(false);

  const unloadVideo = () => {
    setProgress(undefined);
  };
```



```
<GestureHandlerRootView style={{ flex: 1 }}>
  <SafeAreaProvider>
    <ToastMessage />
    <PortalProvider>
      <ScrubberContextProvider>
        <DeepLinkSubscriber>
          <PusherProvider>
            <OfflineNotice />
            <CodePushNotifier />
            <TeamProvider>
              <GlobalVideoProvider>
                <LoadingStack />
```

One callback triggering a state update all the way at the top of the component tree

= 3-5 JS FPS dip!

The Ecosystem Is Comprehensive & Rapidly Evolves

Every product feature we've needed to build has at least one package (if not multiple packages) addressing one of our problems.

– Almost never have to go outside the JS codebase!
– Cross-library compatibility is quite good!
  (think reanimated, rn gesture handler, navigation, tab-view, bottom sheet)
– Teams *need* to keep up with rapid pace of open source development.
  (i.e. going from react-navigation v3 => v5 => v6; keeping up with every RN release)

Significant improvement in native look and feel since 0.59!

Thank you!