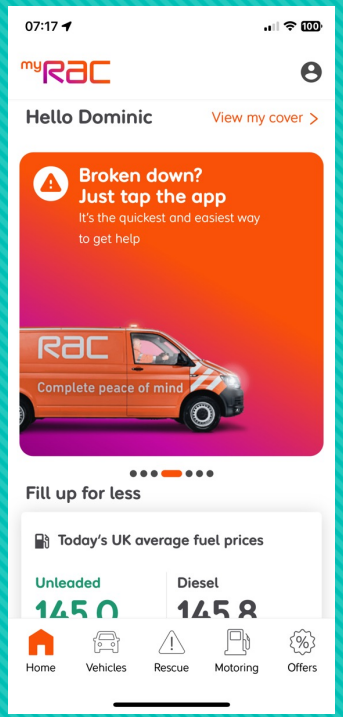
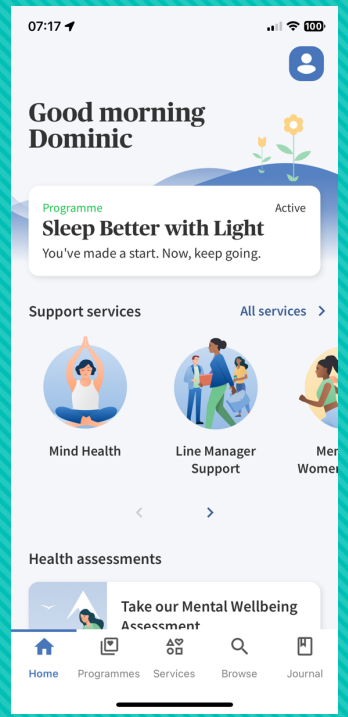
Enhance your React Native apps debugging using Flipper

- By Dominic Garbett

Who am I?

- Senior Software Engineer @ Rac
- Building mobile apps in React Native mainly in the insurance industry

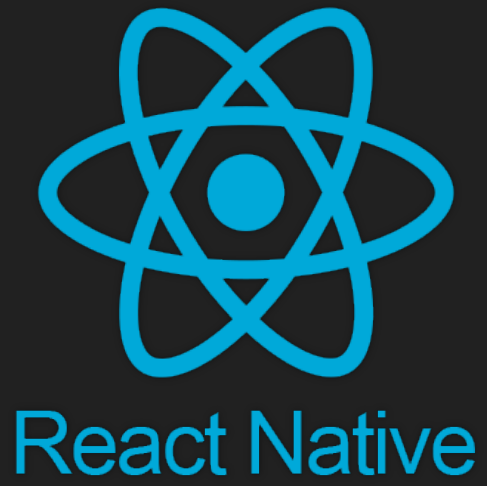


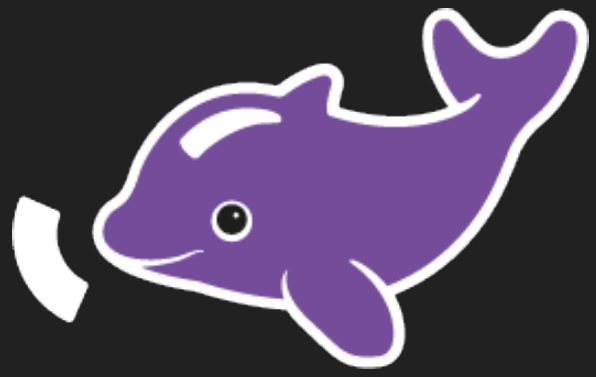


Things I enjoy talking about

- Developer Experience and building tooling to speed up delivery
- Automated testing
- How to build High performing agile squads
- Cycling
- Coffee

What is this talk about?





React Native is great, but debugging with a vanilla setup can be tricky

- Too much time lost using console.log()
- chrome Dev menu can be clunky and limited on what it offers
- Tools such as Chrome dev tools uses remote debugging so experience isn't the same as running JavaScript on a real device

What are the options ?

Tool	Component View	View application state	View Network requests	Profiler	Bidirectional Communication	Third party plugin system
Chrome Dev tools	✓		✓	✓		
React devtools	✓			✓		
Reactatron	✓	✓	✓	✓	✓	
Flipper	✓	✓	✓	✓	✓	✓

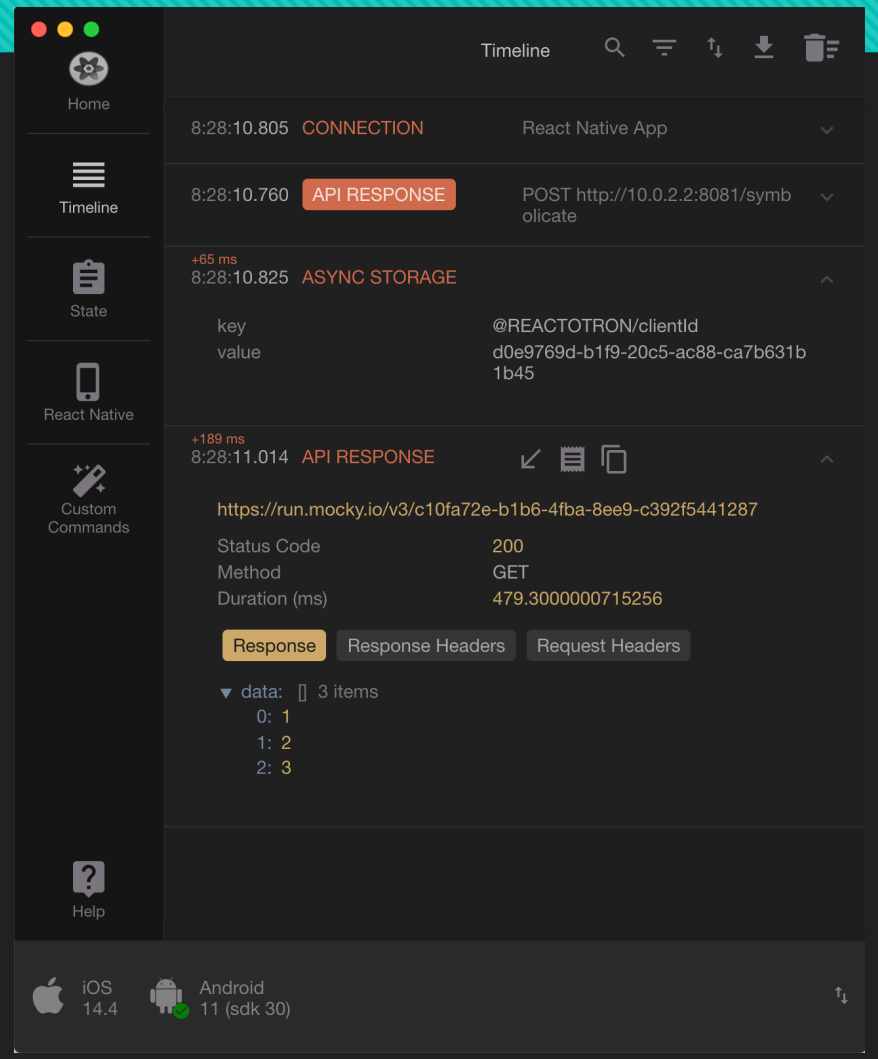
Reactatron

+ Really good at debugging app state

+ Network logs

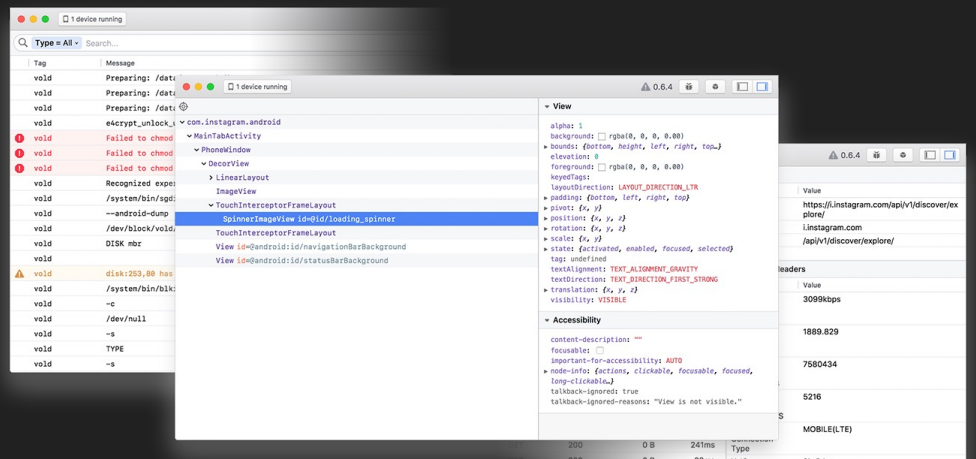
+ Bi-directional communication

- Not so good with extensibility
- No third party plugins or the ability to create your own
- Requires manual setup in your code



Flipper

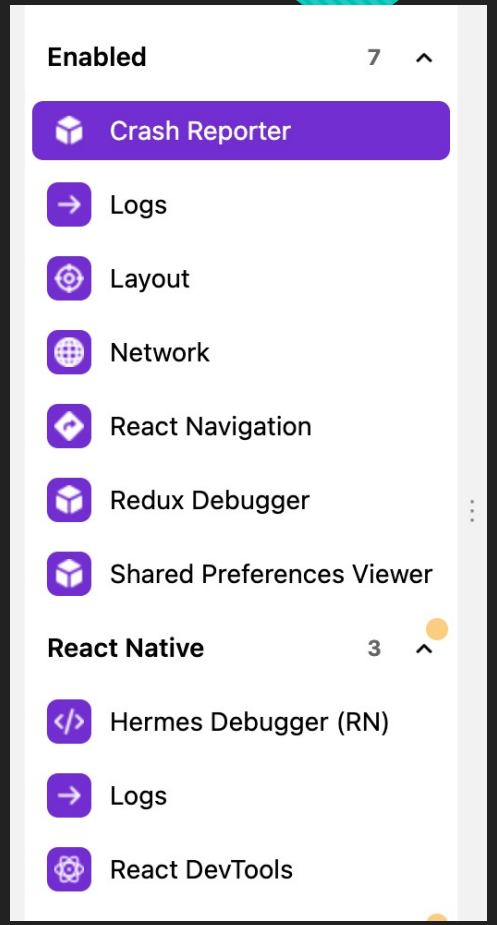
- Open sourced by Meta 2018
- Debugging client for debugging iOS, android and React Native Apps
- Built as a platform to be extensible ,with the ability to create your own plugins and consume others made by third parties.
- Lots of third-party plugins available
- Allow bidirectional communication between React Native and flipper client



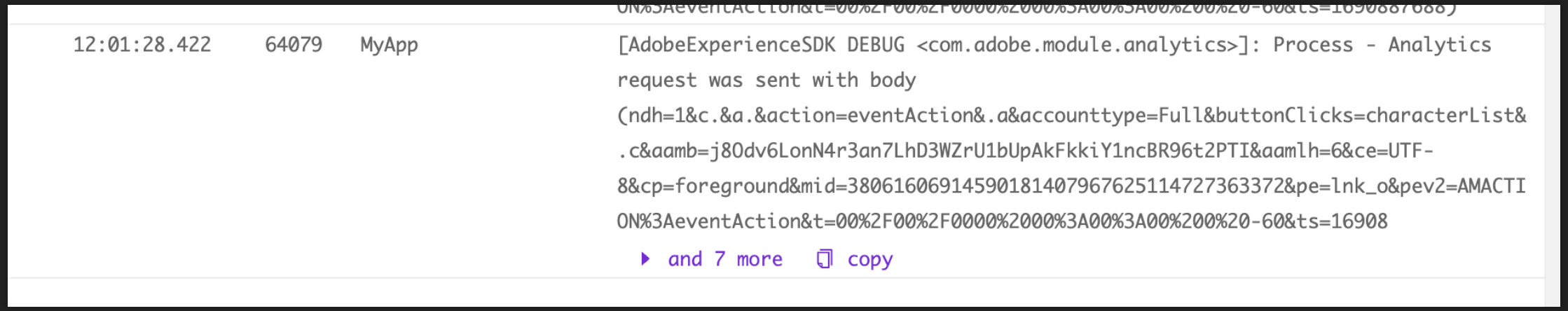
Installing and using flipper

- From **React Native** 0.62 out of the box vanilla installs of React Native – it just works

Plugins that have helped speed up my workflow



What can custom plugins help us with ?



Building you own plugin – Adding to your app

```
const pluginId = 'dom-rn-plugin'; // needs to be same on both client and server

export const useMyFlipperPlugin = () => {
  const [currentConnection, setConnection] =
    useState<Flipper.FlipperConnection | null>();

  useEffect(() => {
    if (__DEV__) {
      if (!currentConnection) {
        addPlugin({
          getId() {
            return pluginId;
          },
          onConnect(conn) {
            setConnection(conn);

            conn.receive('toggleLock', (_data, responder) => {
              toggleDevice();
              // respond with some data
              responder.success({
                ack: true,
              });
            });
          },
          onDisconnect() {
            setConnection(null);
          },
        });
      }
    } //eslint-disable-next-line  react-hooks/exhaustive-deps
  }, []);
```

```
const App = () => {
  useMyFlipperPlugin();

  return (
    <Auth0Provider domain={Config.AUTH_URL} clientId={Config.AUTH_CLIENT_ID}>
      <Provider store={store}>
        <PersistGate loading={null} persistor={persistor}>
          <Router />
        </PersistGate>
      </Provider>
    </Auth0Provider>
  );
};

export default App;
```

```
const sendAnalyticsData = (value: string, eventType: string) => {
  if (currentConnection) {
    currentConnection.send('adobeDataRow', {
      id: new Date(),
      analyticsTag: value,
      eventType: eventType,
      date: new Date(),
    });
  }
};

const sendAppLockedStateData = (state: PinState) => {
  if (currentConnection) {
    currentConnection.send('lockedStateRow', {
      id: new Date(),
      date: new Date(),
      newState: state.toString(),
    });
  }
};
```

Building you own plugin – creating the client side

- Built in react
- Subscribes to the events sent via the application. Outputting them onto a component named plugin

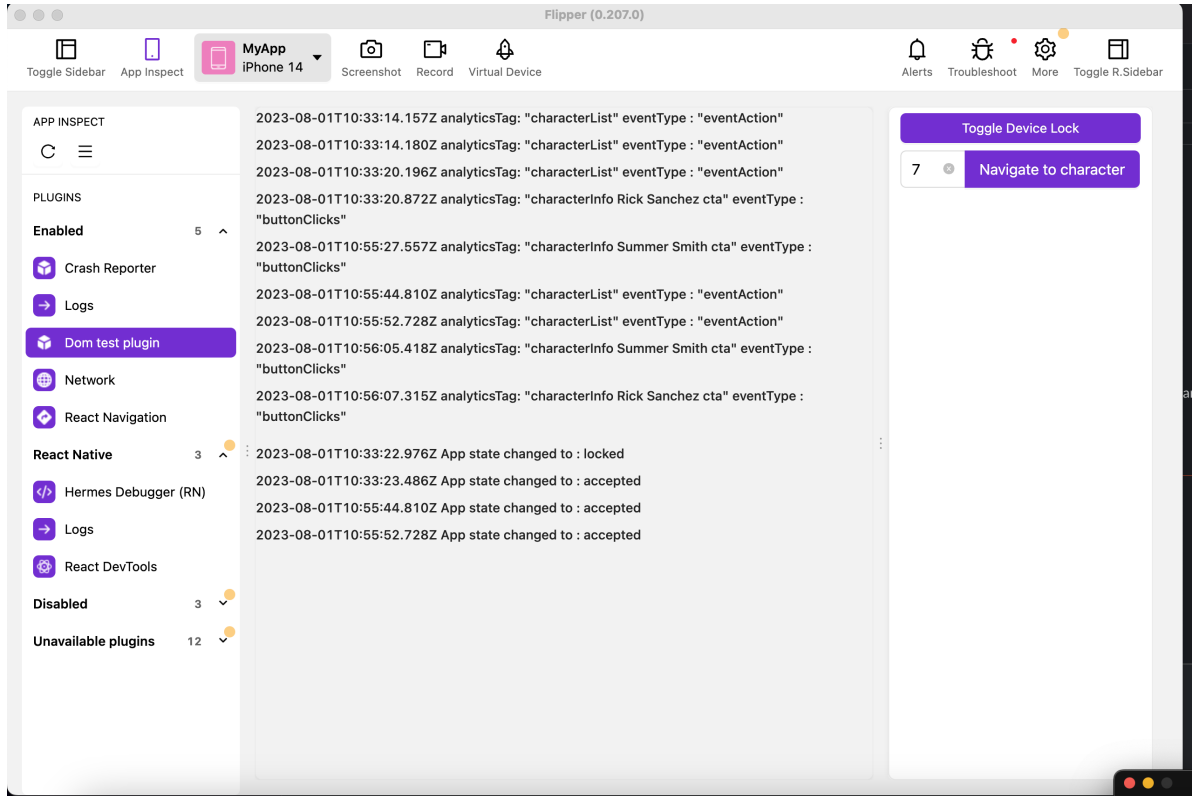
```
// (1)
export function plugin(client: PluginClient<Events, Methods>) {
  // (5)
  const lockedStateRows = createState<Record<string, LockedStateRow>>(
    {},
    { persist: "lockedStateRow" }
  );
  const adobeRows = createState<Record<string, AdobeDataRow>>(
    {},
    { persist: "adobeDatarOWS" }
  );
  const selectedID = createState<string | null>(null, { persist: "selection" });

  // (6)
  client.onMessage("lockedStateRow", (row) => {
    lockedStateRows.update((draft) => {
      draft[row.id] = row;
    });
  });

  client.onMessage("adobeDataRow", (row) => {
    adobeRows.update((draft) => {
      draft[row.id] = row;
    });
  });
```

```
> hello
```

Demo of plugin



Thanks + questions