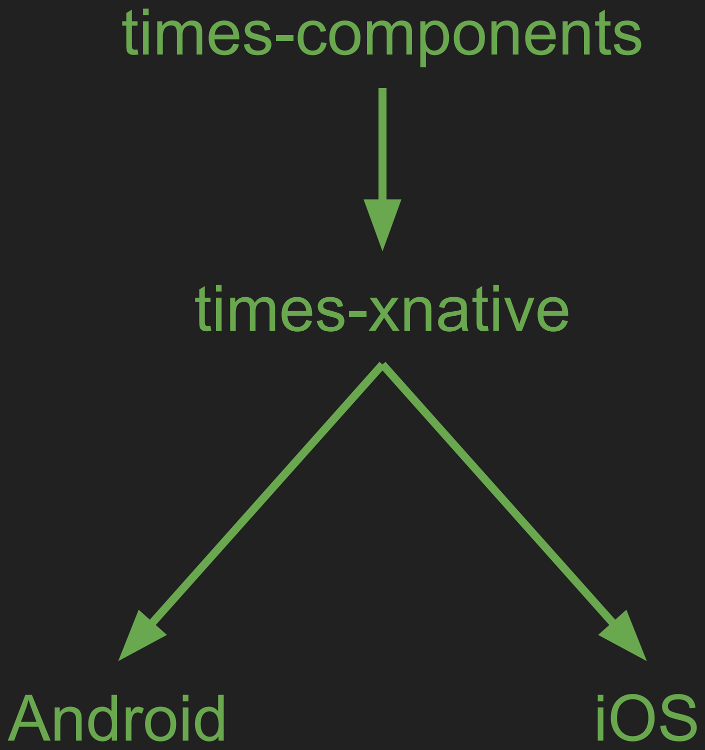
React Native @ The Times


times-components
times-xnative
Android
iOS

times-components

Contains reusable widgets that can be used to compose a page.

It’s used by web+native.

Styling goes here, widgets can be demoed via storybook, a tool to showcase react (native) components.

Most of development is done here.

times-xnative

Cross-native packages, depends on times-component.

Contains the ready to use pages

Pages expose interfaces to bind with the native platforms (endpoint configuration, callbacks, etc).

Targets only Android & iOS

Long term plan is to evaluate if render is a good candidate for it.

Android & iOS

Use times-xnative and bind native components.

React native is used to render individual pages

Navigation is achieved at the native platform level.

Native implementations, i.e. a callback will be implemented here to navigate to another page, or to track a certain event.

Tools of the trade

Yarn

Faster, and stabler package manager for JS packages.

Produces a lockfile (a file where the exact version of a certain package is stored) so that builds are reproducible (package.json can have versions like ">1.2.3").

Yarn monorepo

Used in **times-components** and **times-xnative**

Useful when you have interdependent packages that you need to develop on

All the local packages are symlinked (no need to re-publish)

`/package.json` ← specify **"/packages/*"** containing subpackages
`/node_modules` ← contains dependencies of **all** the packages
`/packages/`**`article-header`**`/package.json`
`/packages/`**`article-header`**`/article-header.js`
`/packages/`**`article`**`/package.json` ← depends on **article-header**
`/packages/`**`article`**`/article.js`

Flow

Static type checker on top of JS.

```
function double(n: number) { return n*2; }
double("1"); // error!
```

Contracts are explicit, mistakes are found at build time.

How does React ~~Native~~ works?

When a component’s **render** method is invoked, it will return a tree representation of the current view (and children).

A driver (React DOM, React Native) knows what a component corresponds to in the relevant platform, and how to render the data tree (create, remove or update views).

In React Native, the driver will bridge from/to JS in order to render in Android/iOS.

Rendering a page on Android (JS)

Register the page via **AppRegistry** in JS

```
AppRegistry.registerComponent("Article", () => ArticleView);
```

Rendering a page on Android

Have a **ReactRootView** somewhere in your layout.

Instantiate a **ReactInstanceManager** (to expose native modules, configuration, etc).

Invoke `ReactRootView#startReactApplication(instanceManager, "Article", bundle)`

The bundle contains a map of options, for example article id.

```
AppRegistry.registerComponent("Article", () => ArticleView);
```

Native Modules

Used to expose constants and/or functions to react.

You can access them in JS from **NativeModules**:

```
import { NativeModules } from "react-native";
const { MyNativeModule } = NativeModules;
```

Must be registered via **ReactPackage#createNativeModules** (returns a list of modules).

The package is added to the react instance manager via **ReactInstanceManager#addPackage**

Native Modules

Module name is returned via **getName**.

Methods exposed via **@ReactMethod** will be accessible in JS.

Methods are **void**, results can be returned via a **Promise** (which are passed as one of the arguments, i.e. our native fetch), with **Promise#resolve** and **Promise#reject** (for result and error).

Constants are exposed via **getConstants** if **hasConstants** returns true (defaults to false), it’s a key-value map, where keys are strings.

Questions?