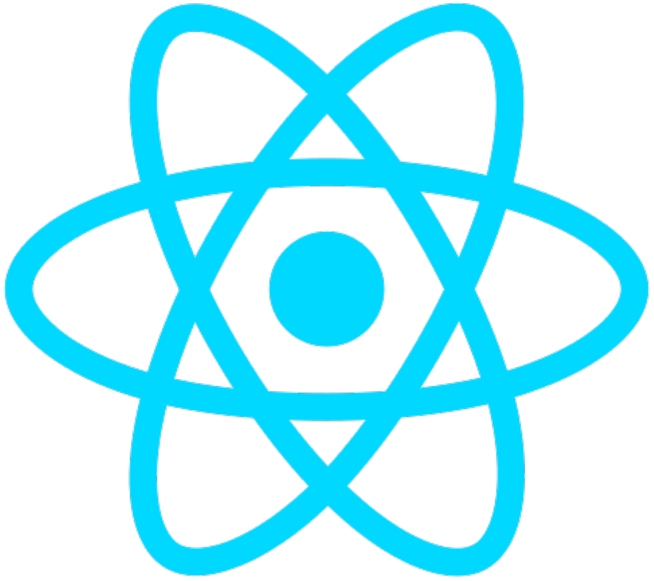


React Native

Lecture 1: Introduction

Shubhang Sharma
Taraksh Goyal
Deepak Soni

Cognition 4.0
2025

UPES MTC

Whoami 👨🏻‍💻

Shubhang Sharma

Technical Associate Head at MTC and intern at Codequantm and Athlyn AI

I Specialize in building full stack applications.

Currently passed First Year at the university.

Github: https://github.com/rustedshader

Speakers 👨‍🏫

Taraksh Goyal

https://github.com/Tarakshgoyal

Deepak Soni

https://github.com/diezo

MTC Social Media Handles 📸 🔗

Instagram: https://www.instagram.com/upesmtc/

Linkedin: https://www.linkedin.com/company/upesmtc/

Facebook: https://www.facebook.com/upesmtc

Microsoft
Technical Community

Materials 📚

All source and material will be available at:

Github: https://github.com/Tarakshgoyal/COGNITION-4.0.git

Reference Documentations:

React Native: https://reactnative.dev/docs/getting-started

Expo: https://docs.expo.dev/

Imagine yourself as a full stack developer …

You want to create application for a client. He asks you to create application that run on IOS , Android and maybe even web.

In traditional way you have to learn switft/Objective C for IOS , Java/Kotlin for Android and React for web and keep the project in sync in all these different codebase.

Here comes React Native,

What if you could write one codebase using the web technologies you already love (JavaScript, React), and have it magically transform into beautiful, native apps for both iOS and Android?

The time learning other technologies can be well spent to make that product better !

What is React Native ? 🤔

- React Native is a framework created by Meta (Facebook) for building mobile apps using JavaScript , Typescript and React principles.
- It enables cross-platform development, allowing you to use a single codebase to build apps for both iOS and Android

Motivation 🎯

- React Native lets you write one codebase and deploy it across iOS, Android, and even other platforms like tvOS, VisionOS, and desktop environments.
- The demand for React Native developers is rapidly increasing, with job listings up 50% in the past year.
- Top companies—including Meta (Facebook), Airbnb, UberEats, Microsoft, Shopify, Tesla, and many more—use React Native to power their mobile apps. Learning React Native gives you skills trusted by global tech giants
- React Native's code reusability (up to 90% between platforms) means you can build and update apps faster, with less maintenance and lower costs.
- React Native is open-source and backed by Meta, ensuring ongoing updates and reliability.

Requirements For Having Smooth Experience 😎

Windows

- **Operating System:** Windows 10 or 11 (x86/64)
- **Processor:** Intel Core i5 8th Gen or newer (or AMD Ryzen 5 equivalent or better)
- **RAM:** 16 GB DDR4 or higher
- **Graphics:** Dedicated GPU (NVIDIA GTX 1050 or better) for faster emulator performance, though integrated Intel UHD Graphics 620 or better is sufficient
- **Storage:** SSD with at least 50 GB free for development tools, emulators, and project files
- **Other Tools:** Latest LTS Node.js, npm, git, and Android Studio (with virtualization enabled in BIOS for emulator performance)
- **Display:** Full HD (1920x1080) or higher recommended

macOS

- **Operating System:** Latest version of macOS (macOS Ventura or newer)
- **Processor:** Apple Silicon (M1/M2) or Intel Core i5 (2018 or newer)
- **RAM:** 16 GB unified memory (Apple Silicon) or DDR4 (Intel)
- **Storage:** SSD with at least 50 GB free for Xcode, Android Studio, emulators, and project files
- **Other Tools:** Latest LTS Node.js, npm, git, Xcode (for iOS Simulator), Android Studio (for Android Emulator), Watchman (recommended)
- **Display:** Retina display or Full HD (1920x1080) or higher

Mobile Simulators 📱

- You can either install Expo Go App on Your Mobile Devices from play store or App Store and scan the QR code displayed on your CLI's.
- Note: Your mobile and Computer has to be on same network.

For Android

Install Android Studio

https://developer.android.com/studio

For IOS (Only on Mac)

Install Xcode.

Install IOS Simulator from Xcode > Settings > Components > IOS xyz

Let’s Gets Started 🚀🚀🚀

Code Editors

Choose any code editor you're comfortable with! In this workshop, we'll be using Visual Studio Code (VSCode).

JavaScript Runtimes

To manage dependencies and run your React Native project, you’ll need a JavaScript runtime. We recommend Bun for its speed and simplicity, but you can also use npm, Yarn, or pnpm.

- Bun (My preference): Faster and more efficient. (Written in Zig)
- Node.js based options:
  - npm (comes with Node.js)
  - Yarn
  - pnpm

For People Having Old Machines 😥

We Got You ! 😁

For People Who don’t have as powerful machines that could run emulators

or react native code they can use.

https://snack.expo.dev/

Your First Hello World ! 🤩

https://github.com/Tarakshgoyal/COGNATION-4.0/blob/main/lecture-1/installation_commands.md

React Native Installation and Setup Commands

React Native Blank Template Setup

Note

Before Starting Make Sure You have already done the pre workshop steps.

Step 1

Open CMD on Windows or Terminal on Macos

Step 2

```
cd Desktop
```

Step 3

If using Bun

```
bunx create-expo-app@latest hello-world --template blank
```

If using Node

```
npx create-expo-app@latest hello-world --template blank
```

Step 4

```
cd hello-world
```

Step 5

```
code .
```

Step 6

If using Bun

```
bun run start
```

If using Node

```
npm run start
```

Project Structure

```
shubhang in ~/dev/COGNATION-4.0/lecture-1/hello-world on main ● λ tree -L 1
.
├── App.js
├── app.json
├── assets
├── index.js
├── node_modules
├── package-lock.json
└── package.json

3 directories, 5 files
```

Structure Explanation

App.js

- This is the main entry point for your React Native application. It contains the root component that defines the UI and logic of your app.

app.json

- The configuration file for your Expo project, defining metadata, platform-specific settings, and Expo SDK options.

assets/

- A directory for storing static assets like images, fonts, sounds, or other media files used in the app.

index.js

- The bootstrap file that registers the root component of your app with Expo.

node_modules/

- A directory containing all the dependencies and their sub-dependencies installed via bun , npm or yarn.

package-lock.json

- A lock file generated by npm to ensure consistent dependency versions across installations.

package.json

- The project’s metadata file, listing dependencies, scripts, and other configurations.

.gitignore

- Purpose: Ignore Enviornment files

Some Quick Components Overview

View

- It is a basic container used to structure and style layouts. It works like a   in web development, supports flexbox for layout, styling, touch handling, and can nest other components inside it.

Text

- It is used to display text in an app. It supports nesting, styling (like font size, color, and alignment), and touch handling.

Status Bar

- It lets you control the appearance and behavior of the device's status bar—the area at the top of the screen showing time, battery, and network info.

Stylesheet

- It is a utility for defining and organizing styles in your app, similar to CSS but written in JavaScript.

Lets Jump to The Working of React Native !

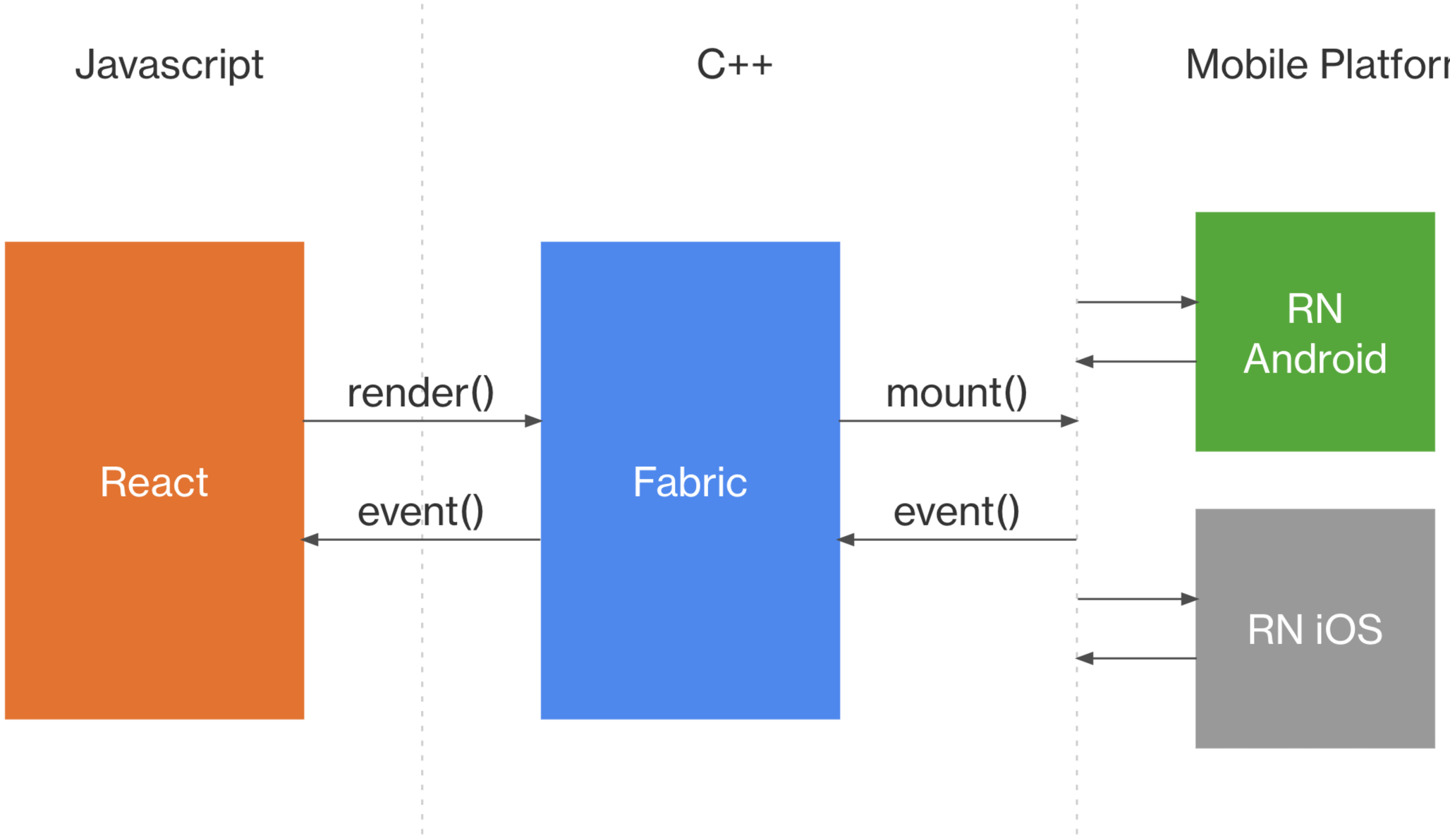
Spoiler: Little Boring 😔

Fabric

Fabric is the modern rendering system for React Native.

Key Points:

- Unify rendering logic in C++ to reduce duplicated code.
- Improve performance by minimizing JavaScript-Native communication.
- Enable better integration with native platforms (e.g., easier to add native modules).
- Support new React features like concurrent rendering.



Source: https://reactnative.dev/architecture/xplat-implementation

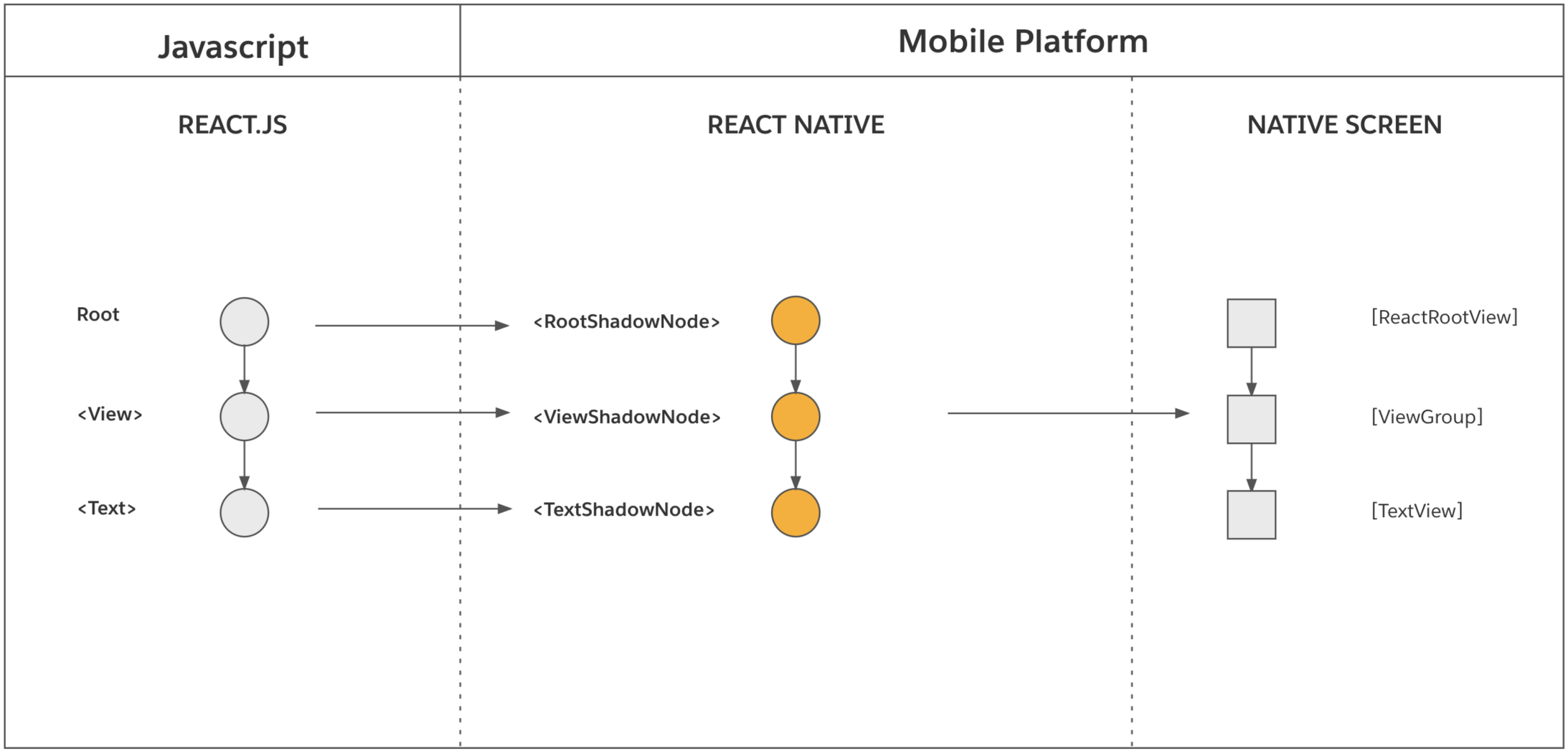
How does React Native Works ?

Render Pipeline (Render, Commit, Mount)

Render (Phase 1)

- React runs your JavaScript code (your components) and creates a React Element Tree, a lightweight description of your UI (like a blueprint).
- For each React Host Component (like or), Fabric creates a corresponding React Shadow Node in C++. These nodes form the React Shadow Tree, which mirrors the structure of the Element Tree but lives in C++ for faster processing.
- Composite components (like your custom) don’t get Shadow Nodes; only native components do. This keeps the Shadow Tree lean.

Render (Phase 1)



Source: https://reactnative.dev/architecture/render-pipeline

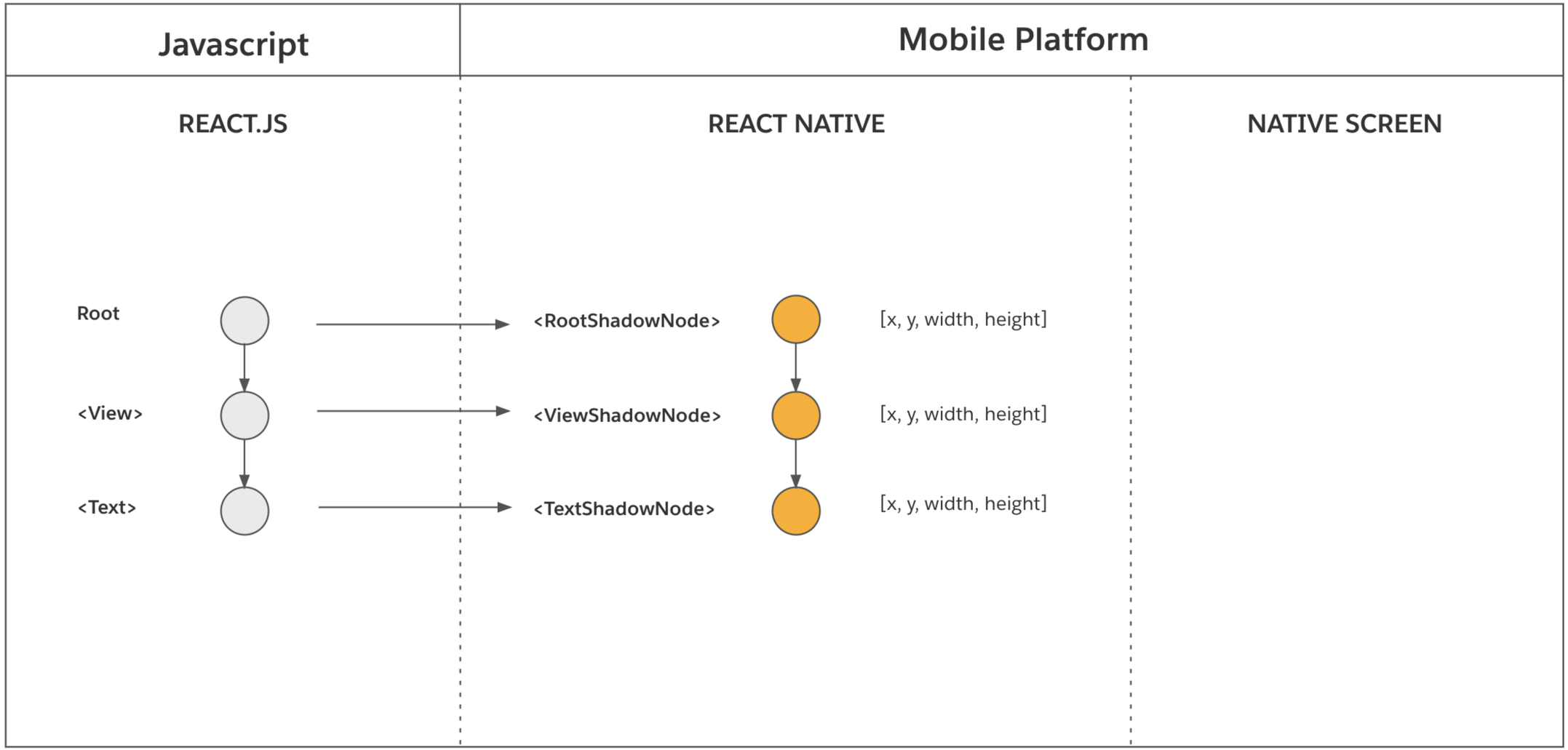
Commit (Phase 2)

Once the Shadow Tree is built, Fabric “commits” it, meaning it’s ready to be processed for display. This phase has two parts:

- **Layout Calculation**: Fabric uses Yoga (a cross-platform layout engine) to calculate the size and position (x, y, width, height) of each Shadow Node based on styles (like flexbox) and screen constraints.
- **Tree Promotion**: The Shadow Tree is marked as the “next tree” to be rendered, scheduling it for the final step.

Layout calculation ensures your UI fits the screen perfectly, and doing it in C++ (via Yoga) is faster than JavaScript.

Commit (Phase 2)



Source: https://reactnative.dev/architecture/render-pipeline

Mount (Phase 3)

The Shadow Tree, now with layout data, is transformed into a Host View Tree the actual native UI elements displayed on the screen (e.g., UIView on iOS, android.widget.TextView on Android).

Steps:

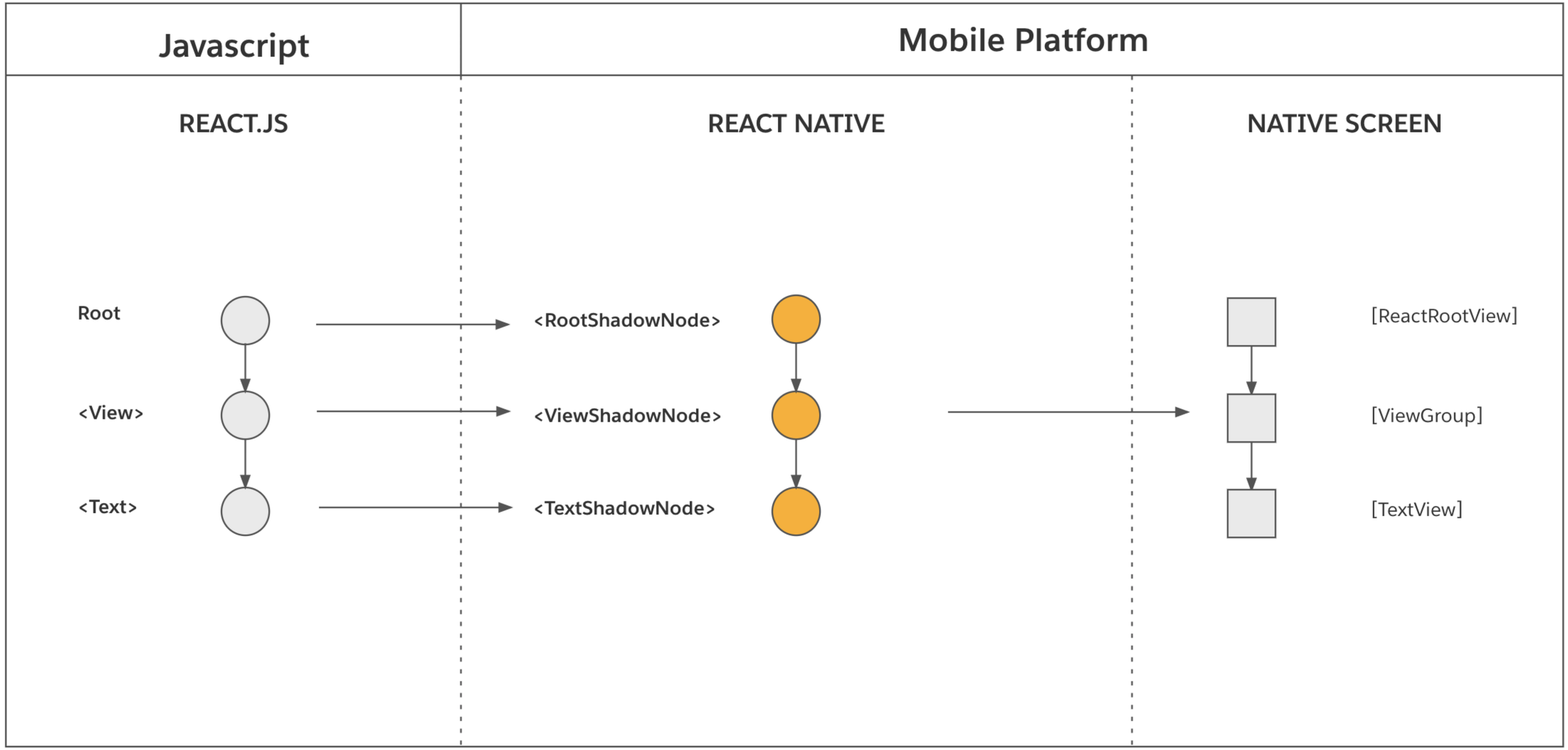
1. Tree Diffing: Fabric compares the new Shadow Tree to the previous one (in C++) to figure out what changed (e.g., “update this TextView’s text, remove that View”). This minimizes unnecessary updates.

2. Tree Promotion: The new tree becomes the “previously rendered tree” for the next update.

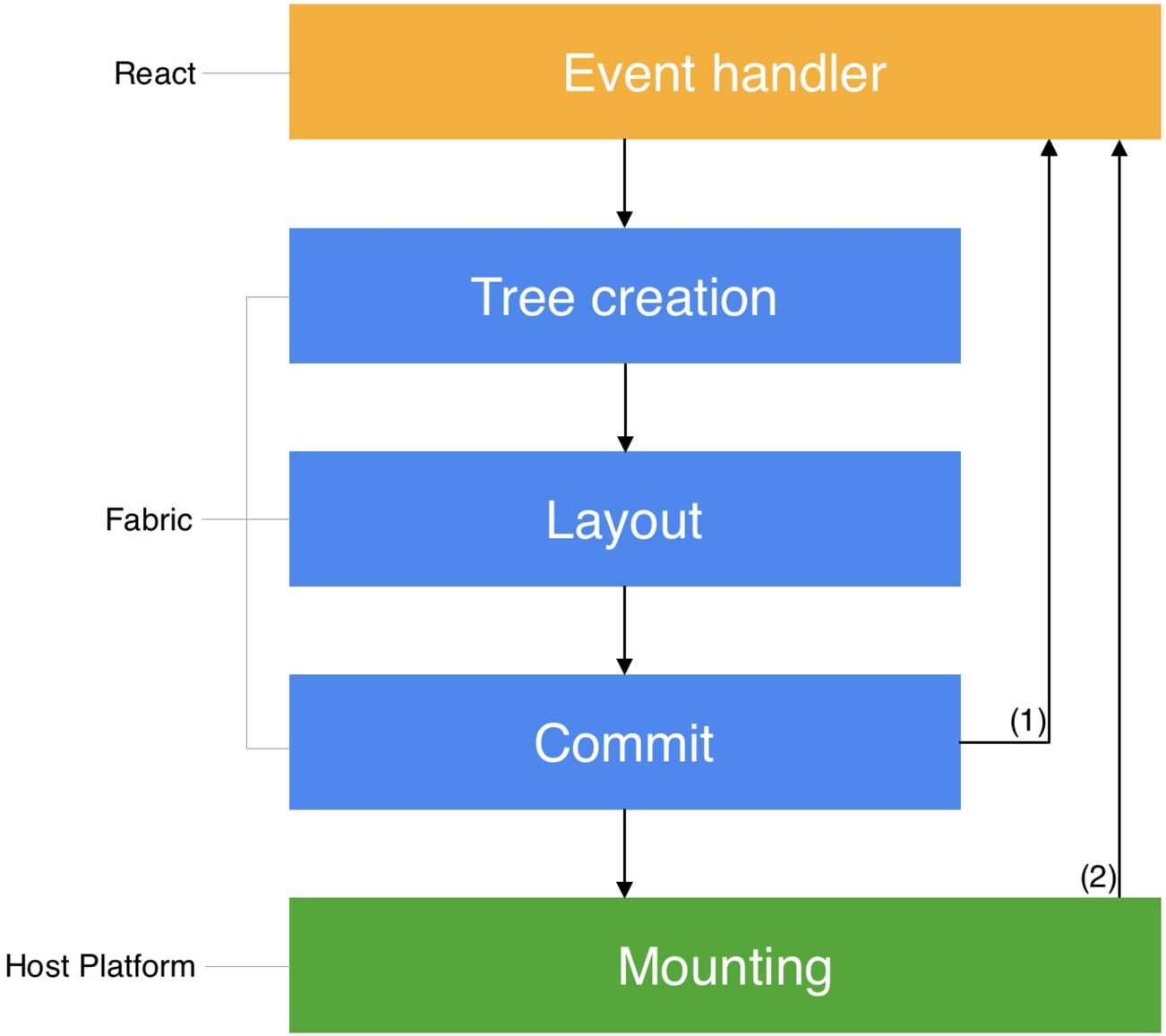
3. View Mounting: Fabric applies the changes to native views on the UI thread, rendering pixels on the screen.

Fabric’s diffing and mounting happen in C++, making them faster and more efficient than the old system, which relied heavily on JavaScript.

Mount (Phase 3)



Source: https://reactnative.dev/architecture/render-pipeline



1. Events from core. For example: onLayout
2. Events from host platform. For example: onChange

Source: https://reactnative.dev/architecture/render-pipeline

Some More Native Terms

- **Hermes**: React Native Javascript Engine.
- **Yoga**: A layout engine that calculates where UI elements go (like CSS flexbox but for native).
- **JSI and JNI**: *JSI* lets JavaScript talk to C++ efficiently, while *JNI* bridges C++ to Android's Java-based views. These are the "pipes" that connect React Native's layers.
- **Some React Native Frameworks:** Expo , Ignite

Thank You