# **Why Ionic Should Be Preferred Over Cordova**

## **Introduction**

In the ever-evolving world of mobile app development, the need for frameworks that allow developers to build high-quality mobile applications for both iOS and Android has never been more important. Two of the most popular frameworks for hybrid mobile app development are **Ionic** and **Apache Cordova**. While both offer ways to develop mobile apps using web technologies like HTML, CSS, and JavaScript, there are significant differences that make **Ionic** the preferred choice for many developers today.

This document compares **Ionic** and **Cordova**, focusing on key aspects such as ease of use, performance, community support, UI/UX, and development experience, ultimately explaining why Ionic is a more favorable solution for modern mobile app development.

## **What Is Ionic?**

Ionic is an open-source framework for building cross-platform mobile applications using web technologies such as HTML, CSS, and JavaScript. It leverages Apache Cordova for accessing native device functionalities but extends its capabilities by providing a library of pre-designed components and tools to build high-quality user interfaces (UI) that resemble native mobile apps.

Ionic focuses on providing a consistent user experience across all devices and platforms. It allows developers to build once and deploy anywhere, including web, Android, and iOS platforms.

## **What Is Apache Cordova?**

Apache Cordova, originally known as PhoneGap, is an open-source platform that enables developers to build mobile applications using HTML, CSS, and JavaScript. It provides a wrapper that enables web-based apps to run in native containers, allowing developers to access native device features such as the camera, GPS, and file system.

While Cordova itself is a powerful tool for building cross-platform apps, it lacks the modern UI components and development tools that are available with Ionic.

## **Comparison of Ionic and Cordova**

### **1. User Interface and Experience**

One of the key areas where Ionic outshines Cordova is in the **user interface (UI)**. While Cordova allows you to access native features, it does not come with a library of pre-built UI components, meaning developers have to create custom interfaces from scratch or rely on third-party libraries. This often leads to inconsistent designs and an overall suboptimal user experience.

**Ionic**, on the other hand, comes with a rich set of pre-designed, mobile-optimized UI components that mimic native behaviors and animations. The framework’s components are responsive and can be customised to match the look and feel of the platform (iOS or Android). Ionic’s components, like buttons, cards, inputs, modals, and navigation bars, are all designed to offer a seamless, high-quality user experience, making it easier to create polished and professional applications.

### **2. Development Tools and Ecosystem**

Ionic provides a comprehensive suite of tools to streamline the development process, including the **Ionic CLI (Command Line Interface)** and **Ionic Studio**. These tools help automate tasks such as creating, building, running, and testing apps. Ionic also offers integration with popular development tools like **Visual Studio Code**, making it easy to get started with building applications.

In contrast, **Cordova** lacks such a rich ecosystem. It relies on third-party tools and lacks the out-of-the-box support provided by Ionic for UI, navigation, and native device integrations. Developers using Cordova often need to write a lot of custom code to implement features that Ionic already offers.

### **3. Cross-Platform Development**

Both **Ionic** and **Cordova** allow you to create cross-platform mobile applications. However, **Ionic** takes cross-platform development a step further by providing **native-like experiences** with minimal effort. Ionic’s **Angular-based architecture** (though it also supports React and Vue) makes it easy to build apps that run seamlessly across multiple platforms while maintaining high performance and responsiveness.

Cordova does allow for cross-platform development, but because it lacks the built-in UI components and tools provided by Ionic, it often requires a greater degree of custom development and testing to ensure that apps perform well across different devices.

### **4. Performance**

Both **Ionic** and **Cordova** rely on web views to render apps, but **Ionic** optimizes performance with features like lazy loading, change detection, and pre-compiled native code for faster startup times. Ionic’s integration with Angular and tools like **Ionic Capacitor** provides better performance by compiling apps as native binaries, which makes them run smoother on mobile devices.

In contrast, **Cordova** apps can suffer from slower performance due to the additional layer of abstraction created by the web view. Performance optimizations often require manual intervention and careful coding practices to ensure that apps are responsive and efficient on mobile devices.

### **5. Capacitor vs. Cordova Plugins**

**Capacitor** is a modern solution introduced by the Ionic team to address the limitations of Cordova plugins. While Cordova relies on plugins for accessing native device features, these plugins often suffer from compatibility issues and may not be as up-to-date as needed. **Capacitor** is an evolution of Cordova that provides a more consistent and easy-to-use API for accessing native features. Additionally, Capacitor ensures that plugins are maintained by the Ionic team, ensuring a higher level of reliability and compatibility.

Cordova plugins, while widely available, may require additional configuration and can be harder to maintain, especially when working with the latest iOS and Android versions.

### **6. Community Support and Documentation**

Both **Ionic** and **Cordova** have large, active communities and well-documented resources. However, **Ionic** has a more modern approach, with official documentation that is regularly updated and more accessible to developers. The Ionic community is also much larger due to its rich ecosystem and extensive use in enterprise environments.

Cordova, while still widely used, has seen a decrease in its popularity with the rise of more modern frameworks. As a result, its community is not as vibrant as Ionic’s, and some Cordova plugins may be poorly maintained or outdated.

## **Why Ionic Should Be Preferred Over Cordova**

### **1. Better UI/UX Design**

Ionic’s pre-built UI components offer a superior user experience that is highly customizable and closely mimics native mobile apps, making it the go-to choice for creating polished, professional apps.

### **2. Comprehensive Development Ecosystem**

Ionic provides developers with a full suite of tools, including a command-line interface (CLI) and integration with popular frameworks like Angular, React, and Vue. This comprehensive development ecosystem streamlines the process of building, testing, and deploying apps, saving valuable development time.

### **3. Improved Performance**

Ionic’s integration with **Capacitor** and performance optimisations such as lazy loading and Angular’s change detection mechanisms lead to faster load times and better performance, making it a better option for performance-critical apps.

### **4. Seamless Native Integration with Capacitor**

Capacitor, the successor to Cordova, offers a modern, simplified API for accessing native device features, making it a far more reliable and easier-to-use solution compared to Cordova plugins.

### **5. Modern Development Practices**

Ionic leverages the power of Angular (or React or Vue) to implement modern web development practices such as two-way data binding, component-based architecture, and dependency injection, which simplifies app development and maintenance.

### **6. Strong Community and Corporate Support**

With Ionic’s strong backing from both the open-source community and enterprises, you’ll have access to better resources, more frequent updates, and official support.

## **Conclusion**

While **Cordova** is a solid framework for building cross-platform mobile applications, **Ionic** provides a modern, feature-rich alternative with a focus on UI/UX design, performance, and native integration. Ionic’s vast ecosystem, superior development tools, and integration with **Capacitor** make it a more complete solution for building hybrid mobile applications. Developers seeking a streamlined, high-quality development experience with robust community support should consider **Ionic** over **Cordova**.

By using Ionic, developers can create apps that feel and perform like native applications, enabling them to deliver better experiences to their users while saving time and effort during the development process.

**References:**

* [Ionic Official Website](https://ionicframework.com/)
* [Apache Cordova Official Website](https://cordova.apache.org/)
* [Capacitor Official Website](https://capacitorjs.com/)