РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

КРОССПЛАТФОРМЕННЫЕ ПРИЛОЖЕНИЯ

Термины

Фреймфорки определяются как сложные среды разработки программного обеспечения.
Фреймворк (framework), который в какой-то мере подменяет термин «технология» и более привычный — software development kit.
Экосистема объединяет сообщество разработчиков. Система предусматривает возможность использования открытых и общих ресурсов каждым членом сообщества (фактически каждым желающим), а с другой стороны, предусматривает механизмы пожертвования своих разработок сообществу.

Современные тенденции развития технологий разработки таковы, что можно явно выделить так называемые **нативные** — «родные», можно сказать, официальные технологии разработки, предлагаемые «держателями» мобильных платформ, и иные технологии от сторонних фирм. Официальные, или **нативные (от англ. native — родной**), технологии предлагаются непосредственно разработчиками мобильных операционных систем (или мобильных платформ).
Разнообразие всевозможных проектов, стартапов и уже устоявшихся и признанных фреймворков и экосистем от сторонних фирм поражает.
Неофициальные технологии могут привлечь разработчика, например, языками программирования, которыми владеет разработчик, и, как следствие, более низким порогом вхождения в разработку, более короткой кривой обучения, а также универсальностью и простотой

В настоящее время большинство фреймворков, ориентированных под **web-разработчиков**, используют одну из двух фундаментальных технологий, позволяющих разрабатывать или гибридные приложения, или приложения, аналогичные нативным (или и то и другое).

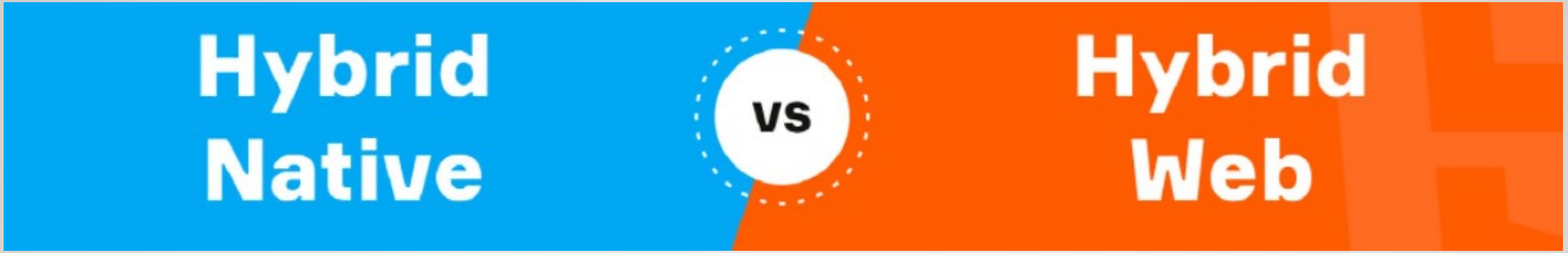
Подходы к кроссплатформенности

https://blog.sibirix.ru/crossplatform-frameworks/

Кроссплатформенность в проекте нужна не ради пользовательского удобства, а для оптимизации стоимости разработки и поддержки приложения. Это решение позволяет написать исходный код для нескольких мобильных платформ, но результатом каждой отдельной сборки станут отдельные исполняемые файлы.

Кроссплатформенные фреймворки можно разделить на две группы:

- ***Hybrid-Native*** (первый подход, гибридно-нативная разработка) и
- ***Hybrid-Web*** (второй подход, гибридная веб-разработка).



Гибридно-нативный подход объединяет фреймворки с нативным пользовательским интерфейсом и общим кодом; и фреймворки с общей кодовой базой и нативным кодом. Среди них: кроссплатформенные платформы разработки приложений ***React Native, Xamarin, NativeScript и Flutter***. О каждом из этих фреймворков мы поговорим позже.

Гибридная веб-разработка кроссплатформенных приложений осуществляется на базе фреймворков с веб-интерфейсом и общими компонентами и с единой кодовой базой, работающей, где угодно. Примером такого кроссплатформенного фреймворка является ***Ionic***

5 основных преимуществ гибридного подхода:

1. Кроссплатформенность. Создав одно приложение, вы можете экспортировать его для любой ОС.
2. Исходя из первого пункта — скорость разработки и масштабирования.
3. Низкая цена. В несколько раз меньше, чем при нативном подходе, особенно с учетом дальнейшей поддержки.
4. Использование одного языка — JavaScript (хотя вот Dart (язык для Flutter) позиционируется в качестве его альтернативы).
5. Одинаковый интерфейс и UX. Одна команда разработчиков и один код максимально приблизят приложение к единому формату UI и UX на всех платформах (Но будут ли они соответствовать гайдам OS — другой вопрос).

5 основных недостатков гибридного подхода:

1. Зачастую более низкая производительность.

2. Меньше возможностей интеграции с «начинкой» устройства.

3. Работа с унифицированным стеком технологий не даст той гибкости настройки и оптимизации, какая есть у каждой ОС со своим собственным стеком технологий.

4. Публикация в сторы может быть еще тем челенджем. Прохождение проверки приложения на соответствие правилам в Apple App Store и в Google Play Store будет отличаться, т.к. требования к приложениям у них отличаются.

5. Ограниченная реализация визуальных и графических элементов в приложениях, особенно анимации.

Нативное будущее кроссплатформенной разработки приложений

Итак, гибридные кроссплатформенные платформы приложений можно использовать, когда:

- стоит задача быстро разработать приложение более чем для 2-х мобильных платформ;
- нужно сэкономить на бюджете;
- требуется относительно простое приложение без сложной анимации;
- допускается исключение из разработки многих параметров нативного функционала.

С развитием мобильных проектов в мире растет спрос на мобильную разработку, растет скорость создания приложений, растет интерес к простым решениям с доступной стоимостью разработки. Все это открывает огромные возможности для улучшения гибридных приложений, которые зачастую могут оказаться не хуже нативных.

React Native

Применяется для перевода кода приложения в машинный язык и обеспечивает нативный внешний вид мобильных приложений.

При работе с React Native вам необходимо сделать сборку своих собственных элементов управления, построить иерархию с учетом интеграции и разработать пользовательский интерфейс на языке React Native.

Основные причины для выбора этой платформы при разработке мобильного приложения.

- Кросс-платформенность. Приложение будет работать на всех платформах: почти весь код написан на JavaScript, общем языке для всех платформ, и этот код взаимодействует с нативными компонентами ОС. Но учтите, что, например, приведение приложения в соответствие с формальными требованиями разных ОС надо будет делать отдельно.
- Простота и легкость создания. Кросс-платформенный фреймворк React Native достаточно прост и удобен (конечно, если разработчик понимает, что делает и что хочет получить в итоге).
- Экономия времени. Кроссплатформенность, плагины с открытым исходным кодом и простота разработки — все это упрощает задачу и сокращает время.
- Сходство с нативными приложениями. Приложения, созданные с помощью React Native, по поведению и внешнему виду близки к нативным. React Native идеально подходит, если вам нужна скорость нативного приложения, но не нужна сложность.

Из существенных минусов:

- это молодой фреймворк и некоторые компоненты отсутствуют, да и обновлений слишком много, а это грозит тем, что вам постоянно придется следить за последними версиями фреймворка и его библиотек,

- не подойдет для проект с тяжелой и сложной графикой и анимацией и, если очень надо, возможно, придется поковыряться с нативным кодом,

- при верстке элементов разработчики iOS пригрустнут — визуального редактора интерфейса нет и все делается в коде с помощью JSX-разметки (не то, чтобы она обязательна, но так можно хотя бы увидеть иерархию компонентов). Для тех, кто по Android'у заметят сходство с XML и выдохнут.

Flutter

Позволяет создавать приложения на языке Dart. Flutter предлагает множество элементов интерфейса, которые выглядят нативными для ОС, но не являются таковыми. Этот фреймворк может подойти для создания унифицированного UX,
и он придерживается иного подхода, чем React Native.

Flutter не превращает исходный код в нативный, который выполняется платформой. Фактически он рисует окно на экране телефона и выводит все элементы сам

А теперь подробнее о плюсах:

1. Язык ***Dart***
   Изначально он задумывался как инструмент для разработки клиентских приложений, был оптимизирован и создан для разработки пользовательского интерфейса. Этот язык предоставляет отличные возможности для разработки кроссплатформенных мобильных приложений. Кроме того, ***Dart*** был создан ***Google*** как расширенная версия ***JavaScript***.
2. Массовый переход на ***Flutter***
   Flutter потребовался всего год, чтобы превзойти ***React Native*** по популярности (хотя ***React Native*** была самой востребованной платформой в свое время). Сегодня с ***Flutter*** вы получаете мощный источник ресурсов для обучения и более квалифицированных разработчиков для создания приложения.
3. Легкость обучения
   Кажется, что не существует такого языка или набора инструментов разработки, которые просты в освоении. Однако есть большая разница между тем, когда есть четкая документация и стандартные шаблоны проектирования, и когда можно найти только разнородную документацию и противоречивые способы решения, казалось бы, простых проблем. ***Flutter*** — это первый случай!
4. Высокая скорость
   Приложения ***Flutter*** компилируются в машинный код с использованием графики и движков рендеринга, встроенных в **C/C++.** Вот почему такие приложения быстрые и надежные. В сфере кроссплатформенных технологий это дает очевидное преимущество Flutter перед конкурентами.
5. Крутая разработка. ***Flutter*** создан с отличным языком и быстрым движком рендеринга (***Skia***)

В целом — довольны, получилось очень неплохо. Но есть минусы:

1. Сырой API. Могут поменять спецификацию «на лету».

2. Не работают некоторые нативные вещи, которые должны работать из коробки. Например, до сих пор есть глюки в текстовых полях на некоторых моделях телефонов (задваивается текст). Решается костылями. А очень неприятно решать костылями то, что ожидаемо должно правильно работать из коробки.

3. Нет НОРМАЛЬНОГО WYSIWYG-редактора (проект зефир скорее мертв, чем жив).

4. Виджеты. Пришлось писать на нативе. В итоге довольно сложная архитектура, т.к. виджет должен работать в паре с основным приложением, а поднимать дартовское приложение в фоне для виджета — это сразу большой расход памяти (напомню, у виджетов например на айфонах есть ограничение в 24 мегабайта). Выкрутиться можно, но архитектура получается довольно дремучая.

Ionic

Это фреймворк типа ***Hybrid-Web***, где приложение на телефоне выполняются в специальной оболочке (***UIWebView*** для ***iOS*** и ***WebView*** для ***Android***), а она уже позволяет показывать ***HTML*** и выполнять ***JavaScript***. Получается, приложение работает как бы в веб-браузере :)

Нет необходимости в глубоких познаниях в каком-либо из фреймворков. У ***Ionic*** огромная встроенная библиотека общих инструментов и большое количество плагинов и модулей, которые обеспечивают доступ к нативным функциям.

И еще немного о преимуществах этого фреймворка:

1. Используется инструментарий, хорошо известный всем разработчикам.
2. Разработка на фреймворке Ionic происходит намного быстрее, чем на остальных. Он основан на Angular, а это значит, что во время разработки приложение можно запускать в браузере и смотреть, как оно выглядит.
3. Есть готовые шаблоны приложений и встроенные пресеты, так что начать разработку — проще простого.
4. Концепция: «Создай один раз, используй всегда и везде».
5. Доступны несколько компонентов пользовательского интерфейса.

Из минусов:

- по отзывам разработчиков тестирование приложения может быть довольно сложным. Протестировать встроенные функции устройства (например, камера, вибрация, мигание), которые есть в документации Ionic, нельзя, ведь эти функции просто не могут быть выполнены в браузере;

- различные нативные «фишки» сложно совместить друг с другом;

- работать приложения на Ionic будут работать плохо на старых устройствах Android: чтобы запуститься приложение на Ionic будет использовать браузер, установленный по умолчанию на устройстве. Устройства со старыми версиями Android (4.0−4.3) по дефолту используют браузер Android, а не Chrome;
- если на проекте сложная графика, то Ionic не подойдет.

Xamarin

Сравнительно новый кроссплатформенный фреймворк Hybrid-Native, основанный на принципах основ Microsoft. Совместим абсолютно с любой ОС. Xamarin дает возможность создавать приложения, которые почти невозможно отличить от их нативных аналогов.

Фреймворк состоит из нескольких основных частей:

- Xamarin. IOS — библиотека классов для C# с доступом к iOS SDK);
- Xamarin. Android — библиотека классов для C# с доступом к Android SDK;
- компиляторы для iOS и Android;
- среды разработки: IDE Xamarin Studio (родная) или Visual Studio.

NativeScript

Этот фреймворк использует нативные средства рендеринга каждой платформы — то есть фактически предоставляет нативный пользовательский интерфейс.

Сравнительная таблица

	React Native	Flutter	Ionic	Xamarin	NativeScript
Язык	JavaScript + React	Dart	JavaScript, HTML, CSS + Angular, React, Vue	C# + .NET	JavaScript, TypeScript
Разрабатываемые приложения	Кроссплатформенные Гибридно-нативные	Кроссплатформенные Гибридно-нативные	Кроссплатформенные Гибридная веб-разработка	Кроссплатформенные Гибридно-нативные	Кроссплатформенные Гибридно-нативные
Первый релиз	2015	2017	2013	2011	2015
Разработчик	Facebook	Google	Drifty Co.	Microsoft	Telerik
Платформы	Android, iOS, UWP	Android, iOS, Google, Fuchsia, Web, Desktop	Android, iOS, Web	Android, iOS, UWP	Android, iOS
Открытый ресурс	да	да	да + платные пакеты	да + платные пакеты	да
Инструменты фронтенда	Native + Declarative UI components	Built-in widgets	HTML, CSS + widgets	Xamarin. iOS/Android or Xamarin.Forms	полностью собственный интерфейс
Производительность	Высокая, близкая к нативной	Очень высокая	Средняя из-за веб технологий	iOS / Android: высокий, близкая к нативной	Высокая; уменьшается при запуске приложений

	React Native	Flutter	Ionic	Xamarin	PhoneGap
Язык	JavaScript + React	Dart	JavaScript, HTML, CSS + Angular, React, Vue	C# + .NET	JavaScript, HTML, CSS
Приложения	Кроссплатформенные	Кроссплатформенные	Кроссплатформенные гибридные	Кроссплатформенные	Кроссплатформенные гибридные
Первый релиз	2015	2017	2013	2011	2009
Кто разрабатывает	Facebook + сообщество	Google + сообщество	Drifty Co.	Microsoft	Adobe
Сообщество	Очень большое	Небольшое, но активно развивается	Большое	Большое	Большое
Платформы	Android, iOS, UWP	Android, iOS, Google Fuchsia, Web, Desktop	Android, iOS, Web	Android, iOS, UWP	Android, iOS, Windows Phone 8
Open source	Да	Да	Да + платные пакеты	Да + платные пакеты	Да
Инструменты фронтенда	Компоненты Native + Declarative UI	Встроенные виджеты	HTML, CSS + виджеты	Xamarin.iOS/Android или Xamarin.Forms	HTML, CSS
Производительность	Высокая, близка к нативной	Очень высокая	Средняя из-за веб-технологий	iOS/Android: высокая, близка к нативной Forms: средняя	Средняя из-за веб-технологий

https://tproger.ru/translations/cross-platform-frameworks-for-mobile-development/#6

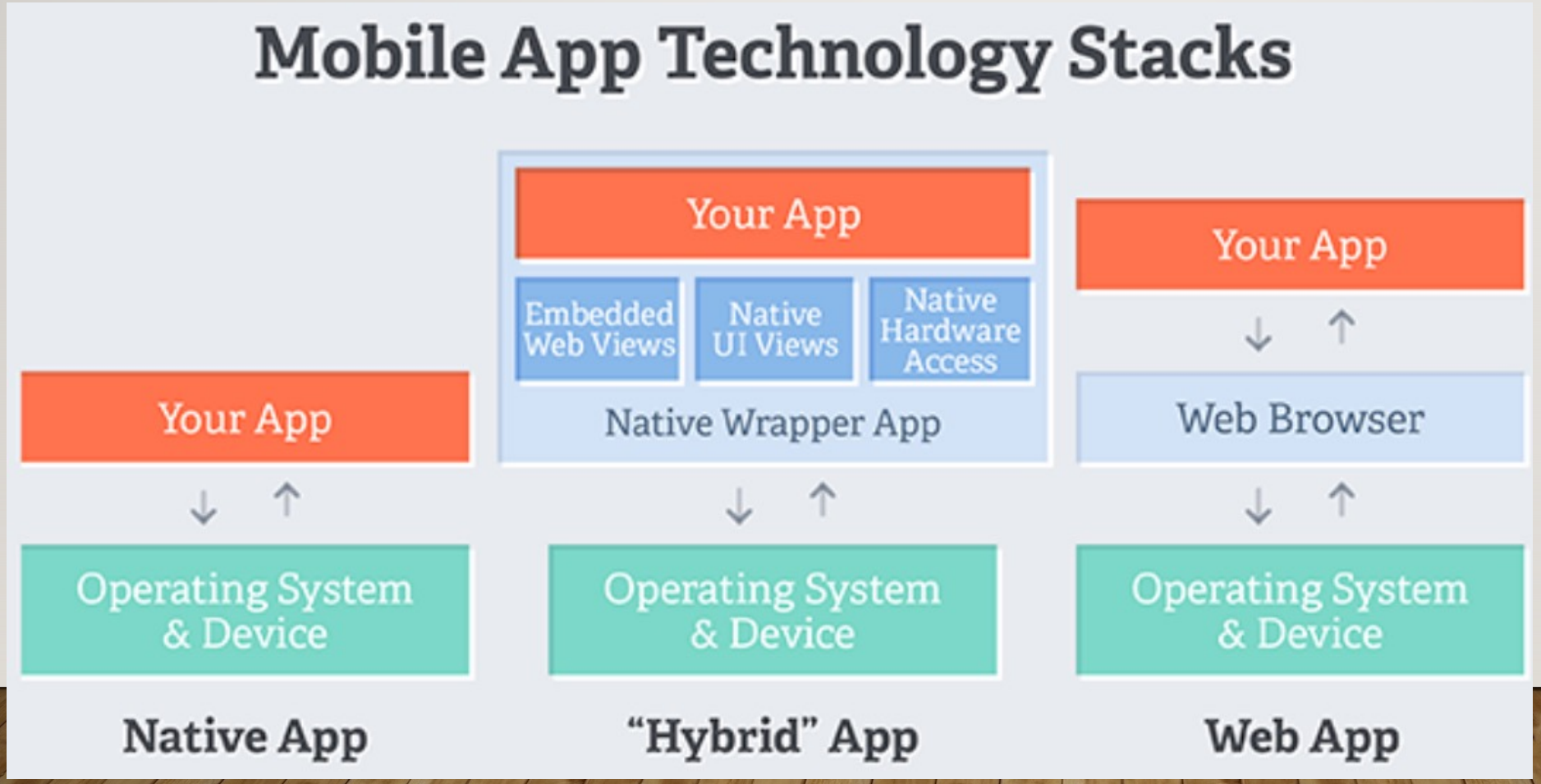
Гибридные приложения

Термин «***гибридная разработка***» отражает суть этого подхода — он сочетает в себе ***native-*** и ***web-разработку***. Гибридное мобильное приложение представляет собой ***web-код*** (то есть код ***JavaScript, HTML и CSS***), который выполняется компонентом ***webview*** (дословно **web-просмотр**), интегрированным в ***native-приложение***. Гибридные приложения имеют ***web-интерфейс*** и производительность браузера. Однако, в отличие от ***web-приложений***, гибридные приложения могут получить доступ к функциональности устройства, к таким его частям, как датчики акселерометра и компаса, камера, геолокация и т. д.

Наиболее популярной платформой для разработки гибридных мобильных приложений в настоящее время является ***Apache Cordova.*** История появления этой технологии такова: в 2008–2009 гг. в рамках канадского стартапа ***Nitobi*** был создан проект ***PhoneGap*** как среда с открытым исходным кодом, которая позволяла получить доступ к нативным функциям мобильного устройства из встроенного ***webview***.

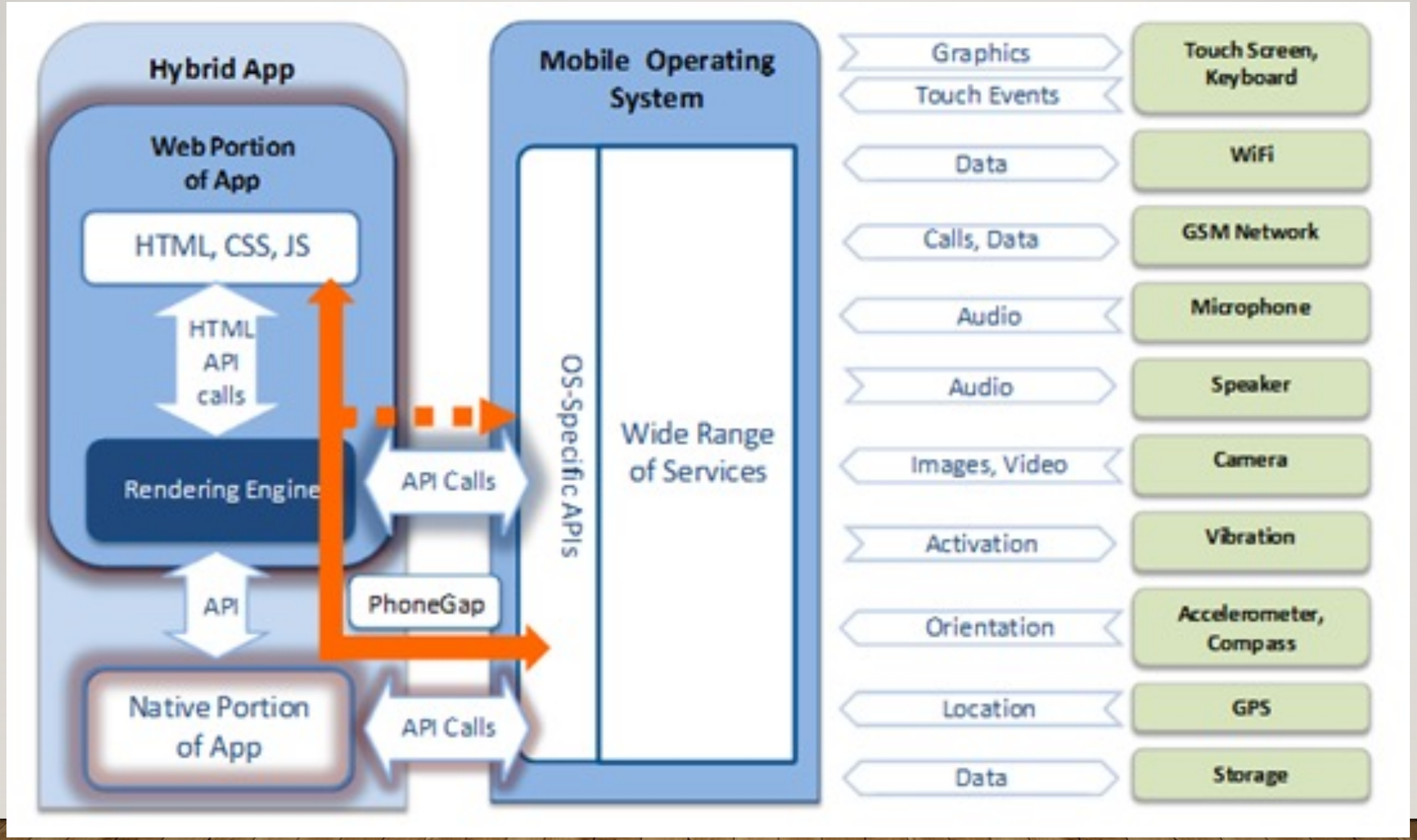
Целью проекта было обеспечить возможность построения мобильных приложений исключительно на ***web-технологиях*** (***HTML/CSS и JavaScript***), но с возможностью вызова нативного кода.

Cordova внедряет web-код в ***webview*** и предусматривает интерфейс (***Application Programm Interface***) для доступа к собственным ресурсам устройства (к файловой системе и другой аппаратуре) из кода ***JavaScript*** через базовые плагины.



Три различных варианта взаимодействия с операционной системой и устройством: нативное приложение, гибридное и web-приложение

Web-часть гибридного приложения (***Web portion of App***) взаимодействует с операционной системой (***OS-Specific APIs***) опосредованно — через движок ***webview (Rendering Engine)***. Нативная часть приложения, которая обычно представлена плагинами, работает с сервисами операционной системы непосредственно. Стрелки от ***HTML, CSS, JS*** к сервисам ***OS-Specific APIs*** показывают запрещённое взаимодействие.



Архитектура гибридного приложения

Достоинства и недостатки гибридных приложений

Достоинства:

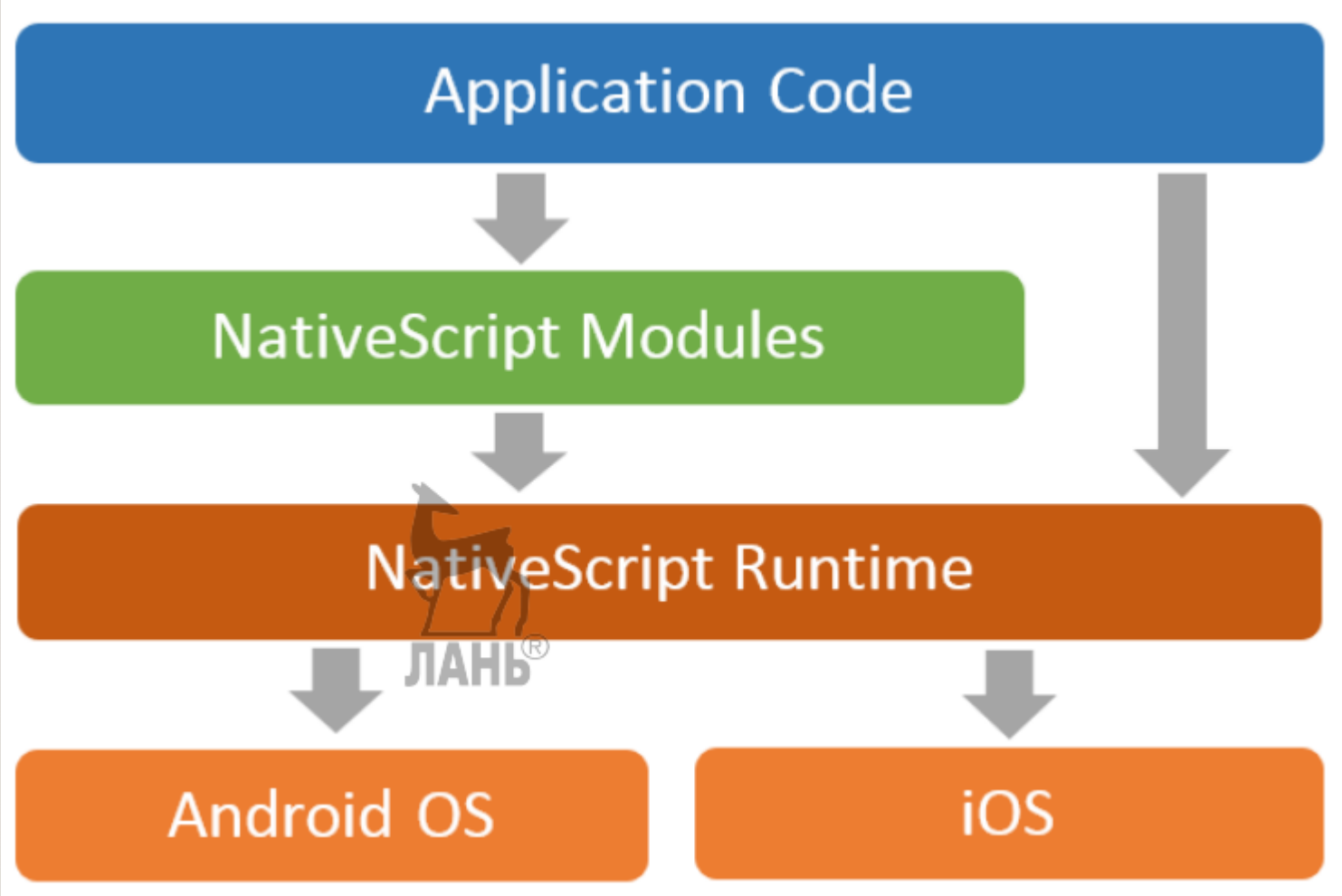
- Популярные и универсальные для всех платформ языки программирования: ***JavaScript, HTML, CSS***.
- Кроссплатформенный подход: один и тот же код можно использовать для ***iOS, Android, Windows Phone*** и других мобильных платформ.
- Сокращение времени и стоимости разработки за счёт универсальности кода и сокращения сроков обучения.
- Мощная экосистема разработки с множеством ресурсов. ***Apache Cordova*** — это платформа с открытым исходным кодом. Ядро платформы обеспечивает доступ к основным возможностям мобильной операционной системы и функциям устройства, а сообщество разработало широкий спектр подключаемых модулей, позволяющих задействовать дополнительные функциональные возможности устройств.

Недостатки:

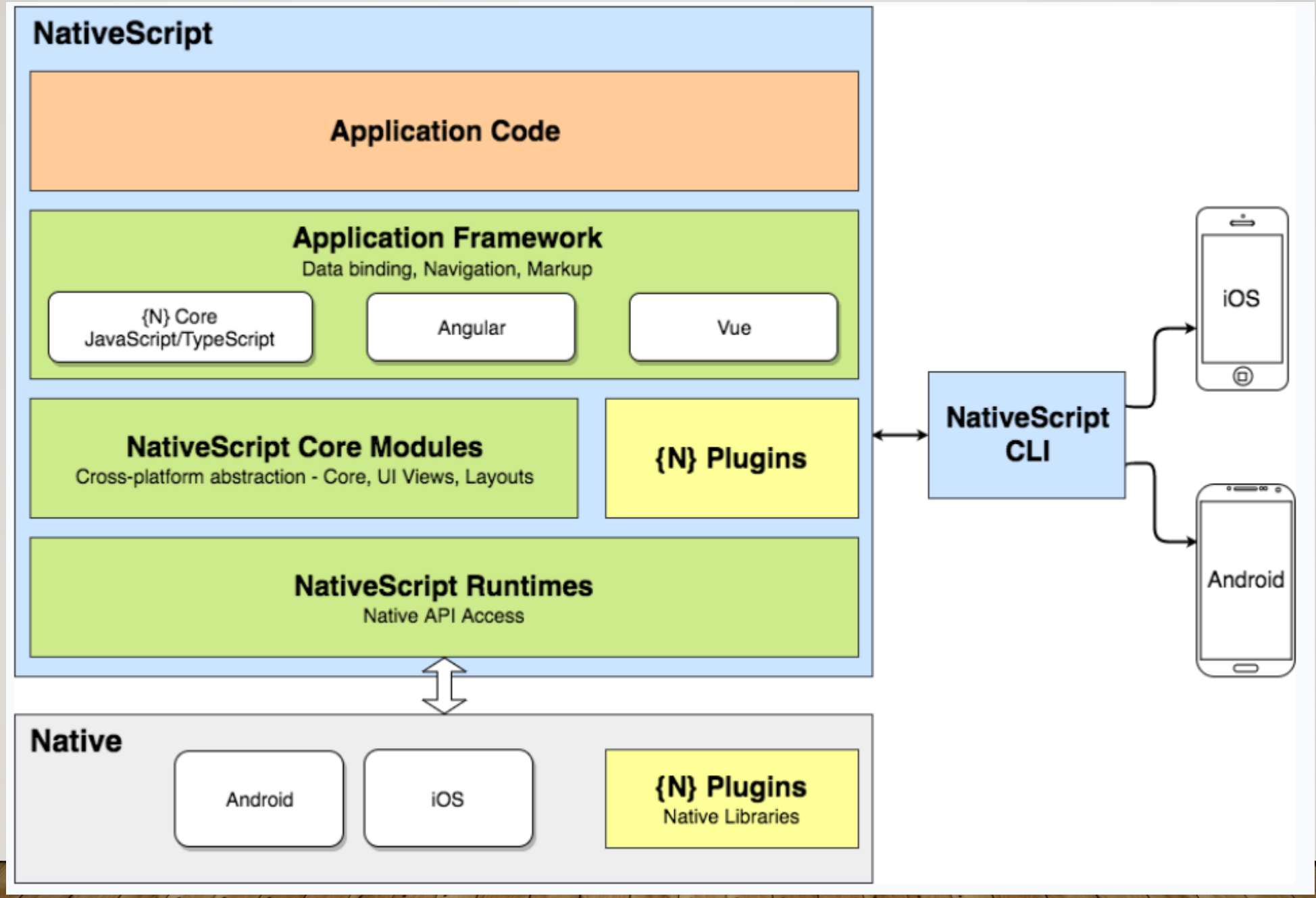
- Поскольку ***webview*** разные для разных платформ (даже для разных версий), то могут потребоваться дополнительные настройки и оптимизация кода, чтобы приложение работало, как ожидается, на всех устройствах.
- Проблемы с производительностью, особенно для задач с быстрой графикой: у ***webview*** есть некоторые проблемы при обработке графики, типичной для игр или приложений с динамичным графическим интерфейсом.
- Для использования уникальных функций платформы потребуется написать дополнительный нативный код (если не удаётся найти готовый подключаемый модуль — плагин).
- Часто плагин, реализующий требуемую функцию, оказывается устаревшим и может потребовать доработки.
- Медленное внедрение новых версий платформ. Можно ждать несколько месяцев, прежде чем ***Apache Cordova*** представит проверенную поддержку новой версии мобильной платформы и ее функций.

NativeScript

Название ***NativeScript*** отражает сущность технологии — создавать приложения с помощью ***JavaScript***, которые выполняются на мобильной платформе с подобающей (нативной) производительностью, используют нативный интерфейс — ***API*** платформы — и получают непосредственный доступ к функциональности устройств (https://docs.nativescript.org). Основой мобильного приложения являются модули ***NativeScript*** и так называемые ***runtimes***. Модули предоставляют доступ к функциям устройства и компонентам пользовательского интерфейса из кода ***JavaScript***, а ***runtimes*** налету (дословно — во время выполнения) интерпретируют, компилируют и выполняют ***JavaScript-код*** и таким образом позволяют использовать уникальные ***API-интерфейсы*** платформы.



Взаимодействие приложения NativeScript с мобильной платформой



Порядок взаимодействия NativeScript-кода с мобильной платформой

Последний рисунок представляет наиболее важные части технологии ***NativeScript***:

- ***Runtimes***, как уже было сказано, являются своеобразными интерпретаторами кода ***TypeScript/JavaScript***, а точнее, базовых модулей — ***NativeScript Core Modules.***
- **Core Modules,** в свою очередь, предназначены для предоставления абстракций, необходимых для доступа к функциям нативных платформ через ***Runtimes***. Они фактически устанавливают определённую дисциплину работы с ***API*** мобильной платформы с помощью языка ***TypeScript/JavaScript***.
- Сборка, установка и запуск приложений на ***iOS*** или ***Android*** через ***CLI NativeScript (Command Line Interface*** — интерфейс командной строки) команды ***NativeScript***.

Достоинства и недостатки *NativeScript*

Достоинства технологии:

- Кроссплатформенный подход. Один код можно использовать для iOS или Android.
- Знания JavaScript, XML и CSS и некоторое понимание нативного программирования в iOS и Android позволяют углубиться в NativeScript.
- Нативный пользовательский интерфейс и производительность на всех платформах и устройствах, поскольку приложения непосредственно используют компоненты iOS или Android.
- Экосистема разработки с открытым исходным кодом.
- Поддержка one day для новых версий платформ: когда новая версия мобильной платформы становится доступной, NativeScript быстро обеспечивает поддержку новой версии и ее функций.

NativeScript — это довольно «молодая» технология, и это её главный недостаток

Гибридная разработка ***Apache Cordova*** и кроссплатформенная разработка ***NativeScript*** имеют много общего.

Оба фреймворка:

- используют один и тот же исходный код приложения для различных мобильных платформ;
- допускают интеграцию с ***JavaScript-библиотеками*** и менеджерами пакетов;
- допускают расширение с помощью плагинов (подключаемых модулей) от разработчиков эко-сообщества;
- имеют открытый код.

Гибридный подход является лучшим выбором в следующих случаях:

- разработка ведётся на ***JavaScript, HTML и CSS***;
- Для получения наибольшей выгоды от мощной экосистемы;
- пользовательский интерфейс разрабатывается с помощью ***Ionic, Kendo UI, jQuery Mobile, Sencha Ext JS*** или других ***JavaScript-библиотек***;
- используются пакеты ***Bower***;
- возможность заимствования кода аналогичного ***web-***приложения;
- кратчайшие сроки разработки и обучения;
- приложение должно выполняться для ***Android, iOS, Windows Phone*** и других платформах;
- приложение должно выполняться одинаково на старых и новых устройствах (то есть на старых и новых версиях мобильных операционных систем);
- приложение не использует активно встроенную функциональность устройства;
- от приложения не требуется максимальная производительность;
- от приложения не требуется специфичный для каждой платформы интерфейс.

NativeScript является лучшим выбором в следующих случаях:

• разработка требует знания ***JavaScript, XML и CSS*** и понимания основ нативного программирования под ***iOS и Android***;

• готовность принять недостатки начальной стадии развития экосистемы сообщества;

• используются модули ***npm***;

• достаточно времени для разработки и обучения;

• приложение должно выполняться на ***Android и iOS***;

• приложение должно выполняться на современных устройствах с последними версиями операционных систем;

• приложение активно использует нативную функциональность устройства;

• требуется пиковая производительность, например для игр;

• требуются специфичные для платформы интерфейс и функциональность.

Cordova и React Native | Таблица Сравнения

	Cordova	React-Native
Когда использовать?	Веб-просмотр, веб-приложение, плагины и т. д.	Команда разработчиков с предварительным знакомством с JavaScript
Материнская организация	Adobe	Facebook
Дата запуска	2009 г.	2015 г.
Кроссплатформенность	да	да
Популярность	Менее популярный	Более популярным
Популярные варианты использования	Spark Chess Sworkit Fan React	Walmart Instagram Uber Eats
Язык программирования	HTML5, CSS3 и JavaScript	JavaScript
Окружающая среда	Гибкая IDE	Гибкая IDE
Перформанс	Почти родной	Почти родной
Компоненты	Фреймворки пользовательского интерфейса, например Dojo Mobile, Sencha Touch или jQuery Mobile.	Кнопки, ввод текста и менее сложные компоненты
Повторное использование кода	да	да
Ценообразование	Свободный	Свободный
Размер сообщества	Маленький	Большой
Поддержка	Сообщество	Сообщество

https://blog.back4app.com/ru/cordova-%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2-react-native-