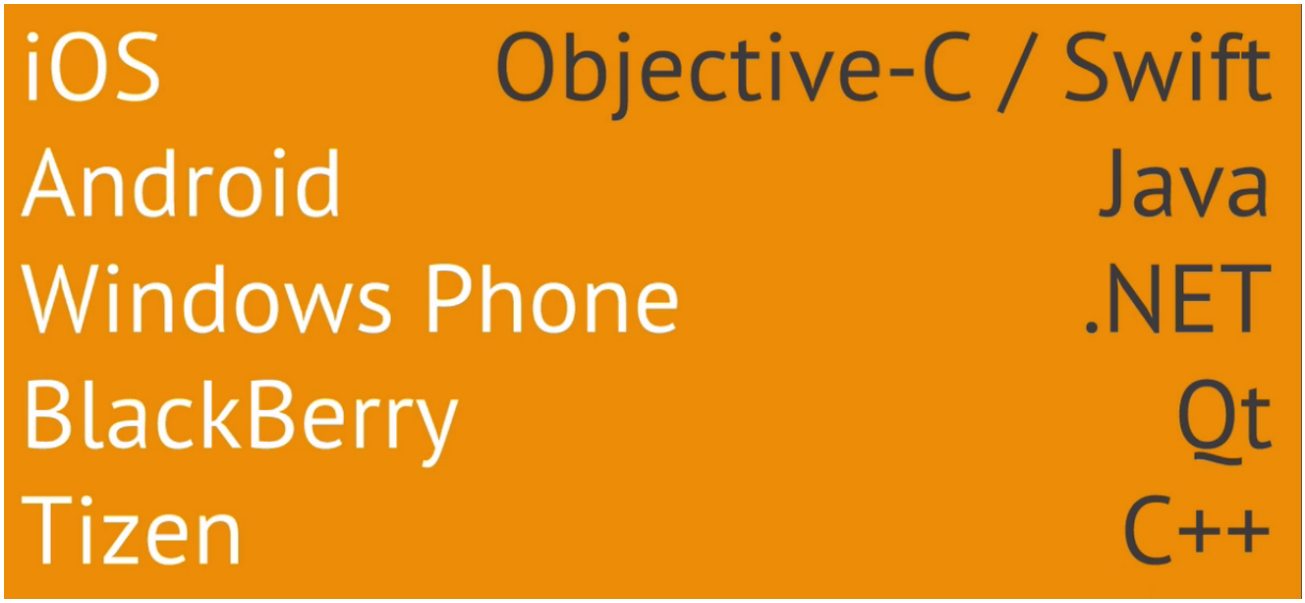
01

Porque Cordova

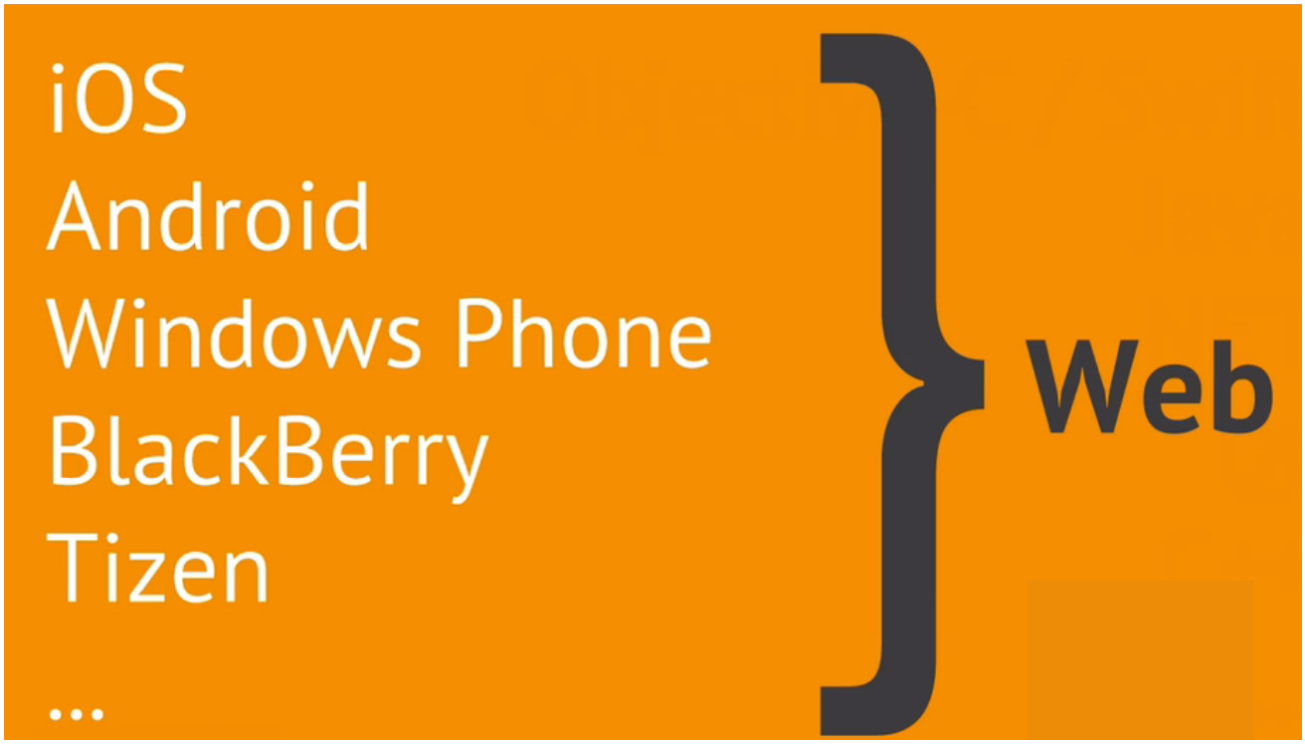
Transcrição

Iremos falar neste curso sobre o Cordova. O nosso objetivo é desenvolver aplicativos *mobile* novos. Quando pensamos no cenário atual de aplicativos, temos uma imagem um pouco desoladora. As plataformas mais famosas são `iOS` e `Android`, mas existem outras:

Se quisermos desenvolver para cada uma destas plataformas, teremos que aprender linguagens e *frameworks*, totalmente diferentes. Por exemplo, se você está em uma plataforma `iOS` terá que trabalhar com `Objetive-C` ou `Swift`, no `Android`, você trabalhará com `Java`. Trabalhamos de forma diferente em cada plataforma.



Para cada uma destas plataformas, temos um custo para desenvolver uma app, porque precisaremos de uma equipe diferente, com conhecimentos diversos, além da manutenção e outros fatores. Várias questões que não tornam viável. Porém, todas estas plataformas, além de suportarem aplicativos nativos nas diversas linguagens, elas têm algo em comum: todas suportam **Web**.



As plataformas suportam o *web browser*, o que na verdade significa que todas conseguem executar `HTML`, `CSS` e `Java Script`. Desde o princípio, a Web tem esta característica multiplataforma já.

Será que nós não conseguiremos usar o conceito de **multiplataforma** da Web para desenvolver app? Web não é app... Mas se precisarmos desenvolver um app, teremos que procurar soluções nativas e ficaremos perdidos com a variedade de plataformas? Por isso, nasceu o *Cordova* - que será o foco do nosso curso. Ele irá nos possibilitar criar aplicativos multiplataformas com a base do `HTML` e `JS`. Ou seja, escrevemos apenas um código e o nosso aplicativo irá rodar em todas as plataformas. Não será Web, mas efetivamente um aplicativo.

Vamos fixar estas ideias... O *Cordova* irá permitir que você escreva o código `HTML`, `CSS` ou `JS`, da maneira como estiver acostumado a fazer na Web. Existirão algumas diferenças que trataremos mais adiante, mas será o mesmo código criado no `Front-End`. No entanto, o objetivo do *Cordova* é "empacotar" tudo isto e criar um app.

O *Cordova* é feito das linguagens `HTML` , `CSS` e `JS` empacotado em um app instalável nos sistemas operacionais móvel, em geral. Ele seria como uma "casca" - quase nativa da plataforma - que envolve estas linguagens. Isto permitirá que o aplicativo funcione em tecnologias que, geralmente, usávamos na Web.

Na verdade, não teremos um navegador, em vez disso, teremos o que chamamos de *WebView* - uma forma que todas as plataformas (`iOS` , `Android` e outras) de executar `HTML` , `CSS` e `JS` , dentro de si mesmas, porém fora de um navegador. Isto significa que não teremos barra de endereço, botões de voltar, menu... Não precisamos mais da interface do navegador, sobrando apenas uma espécie de "renderizador" de `HTML` . Esta é a definição de *WebView*.

O que o *Cordova* irá fazer por nós? Ele irá criar a *WebView* da plataforma, jogará o `HTML` dentro e executará o código. Como não estamos rodando em um navegador, isto significa que o HTML funcione como um app realmente, sem as característica do navegador na plataforma. Quais as vantagens disto?

Teremos um app instalável, publicável na loja (e com todas as vantagens que correspondem), porém implementada com um código plataforma.

Nós falamos até aqui de *Cordova*, mas e *PhoneGap*? Do que se trata?

Nós veremos mais detalhes sobre os dois, durante o curso. Mas para começar, o *PhoneGap* é o precursor do *Cordova*. Quando o projeto foi criado há muitos anos, o nome recebido foi *PhoneGap*. Então, a Adobe comprou a empresa e doou o código do *Phonegap* para o projeto ***Apache***, que passou a se chamar ***Apache Cordova***.

O *Cordova* é o projeto *open source* que se originou a partir do projeto *PhoneGap*. O projeto original ainda é usado hoje em dia? Sim. A Adobe continua usando o nome *Phonegap* para alguns projeto comerciais que ela desenvolve. Na verdade, são serviços e funcionalidades a mais para o próprio *Cordova*. Podemos afirmar que ele é 90% *Cordova*, porém com algumas características proprietárias a mais e os serviços oferecidos são bastante úteis.

A princípio o que veremos é *Cordova* com alguns detalhes de *PhoneGap*.

Vamos esclarecer um detalhe sobre a plataforma: **Cordova não é Web**. Por que falamos isto? É comum encontrarmos pessoas que afirmem que o *Cordova* é Web através de App, um conceito bastante misturado. Na verdade, os dois possuem semelhanças, porque utilizam a mesma linguagem de programação (HTML, CSS e JS). Mas o *Cordova* está mais próximo de um aplicativo do que de uma Web. Por exemplo, ela não se encontra disponível na internet, não possui uma URL, nem tem links que possam me direcionar para algum recurso. Vantagens da Web - como estar indexada no Google - também não possuímos, quando trabalhamos com o aplicativo *Cordova*. No entanto, temos todas as vantagens de trabalhar com um aplicativo: ele será instalados na loja, trabalhará Offline, terá suporte para temas específicos nativos de hardware do aparelho. Todos recursos que temos acesso em um app, teremos no *Cordova*. Logo, o *Cordova* tem mais característica semelhantes a um aplicativo do que a Web, a semelhança. Temos uma app nativa desenvolvida com as linguagens usadas na Web.

Você pode perguntar: por que vamos usar um nativo ou o aplicativo *Cordova*, que é híbrido (mistura HTML com app)? Quais são as vantagens e desvantagens de cada um?

O híbrido irá trazer grandes vantagens no desenvolvimento, porque iremos economizar dinheiro ao criar um único código que rodará em diversas plataformas. Já o nativo, irá oferecer o acesso a detalhes de cada plataforma e dos sistemas operacionais. Do ponto de vista do usuário, não deve existir diferenças. Ele irá instalar e utilizar os apps nativos ou híbridos da mesma forma. E existe diferença de performance? Sim. Se o seu projeto demandar muita performance, que tenha muitos detalhes, realmente o nativo pode ser um melhor desempenho. Porém, na maioria dos casos, a performance do nativo e híbrido são equivalentes. Mas sabemos que existem aplicativos ruins em ambos cenários. Talvez, a única distinção perceptível é que alguns aplicativos nativos têm a opção de usar os componentes padrões da plataforma que eles estão rodando. Com um aplicativo híbrido, você terá mais dificuldade. Como iremos implementar com CSS, HTML, geralmente, termos um design que será aplicado a todas as plataformas. Então, não conseguiremos fazer um design CSS com características de OS, porque no Android teremos um resultado bizarro. Geralmente, iremos fazer um design multiplataforma, que rodará em vários lugares. Esta é uma vantagem do híbrido.

A escolha entre um nativo ou híbrido é mais uma escolha de implementação, do que algo que seja perceptível para o usuário. Para ele, não haverá diferenças e ambos serão usados normalmente.