Explain the concept of Hybrid Mobile App Development

Native Apps are specific to a given mobile platform (iOS or Android) using the development tools and language that the respective platform supports. Native apps look and perform the best

HTML5 Apps use standard web technologies, like JavaScript and CSS to create cross platform mobile applications. Limitations from this strategy is, session management, no secure offline storage, no access to native device functionality (camera, calendar etc.)

Hybrid Apps make it possible to embed HTML5 apps inside a thin native container, combining the best (and worst) elements of Native and HTML5 apps.

Hybrid mobile applications are built in a similar manner as websites. Both use a combination of technologies like HTML, CSS, and JavaScript. However, instead of targeting a mobile browser, hybrid applications target a WebView hosted inside a native container. This enables them to do things like access hardware capabilities of the mobile device.

Today, most hybrid mobile applications leverage Apache Cordova, a platform that provides a consistent set of JavaScript APIs to access device capabilities through plug-ins, which are built with native code. As a side note, Apache Cordova originally started as a project named PhoneGap. These days, PhoneGap exists as a distribution of Apache Cordova that includes additional tools.

These plug-ins include APIs for accessing the device's accelerometer, contacts, camera, and more. There is also a number of plug-ins that are built and maintained by the developer community at-large. These can be found in the Apache Cordova Plugins Registry. A curated subset of these plug-ins that have been throughly tested, documented, and extended can also be found at the Telerik Verified Plugins Marketplace.

Explain the Pros & Cons of using Hybrid Mobile App Development compared to Native App Development

Features	Native Apps	HTML5 Apps	Hybrid Apps
Navigation	Fast	Slow	Slow
Look and Feel	Native	Emulated	Emulated
Graphics	Native APIs	HTML5, Canvas, SVG	HTML5, Canvas, SVG
Camera	Yes	No	Yes
Notifications	Yes	No	Yes
Contacts, Calendar	Yes	No	Yes
Geo-location	Yes	Yes	Yes
Swipe	Yes	Yes	Yes
Pinch, Spread	Yes	No	Yes
Connectivity	Online and offline	online	Online and offline
Performance	Fast	Slow	Slow
Development Skills	Objective-C, Java, C#	HTML5, CSS, JavaScript	HTML5, CSS, JavaScript
Distribution	App store	Web	App store

Hybrid / PhoneGap:

- For budgetary limitation, hybrid app may be a better choice.
- If there is a need to quickly develop the app, hybrid app may be a better choice.
- If the app is simple, does not have large animations, does not have lots of clicks and does not require lots of native user interaction, hybrid will be a better choice.

Native:

- If the requirement is to create the best user experience, native development is a better choice.
- For companies with sufficient allocated budget that are planning to build and maintain large app projects and do not want to worry about limitations for implementing new technologies, and support, native is a better choice.

Item	Native has Advantage	Hybrid has Advantage
Cost		✓
Platform Independent		✓
Quicker development		✓
User Experience	✓	
No Limitations	✓	
Support	✓	
Tools and Debugging	✓	

Explain about the "building blocks" involved in an ionic Hybrid Application

Ionic is a complete open-source SDK for hybrid mobile app development
Built on top of AngularJS and Apache Cordova, Ionic provides tools and services for developing hybrid mobile apps using Web technologies like:

- CSS
- HTML5
- Sass

Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova

Explain and demonstrate ways to debug Hybrid Mobile Applications Running on a real device

If the phone is an android device, you will be able to connect the phone, install the app, and then open Chrome and type: *chrome://inspect/#devices*

This will open a page that should display the phone and a *"inspect"* button. Click the button and start debugging.

I don't have an android device, so I can't demonstrate it.

Explain when and why CORS is a problem for Hybrid Mobile Applications

What is CORS?

CORS = Cross origin resource sharing.

The *origin* is the host you are currently viewing.

Say we send an AJAX request to *http://cors.api.com/api*, your host origin will be specified by the Origin header that is automatically included for CORS requests by the browser. Since *ionicframework.com* does not match the host of *api.com*, our request from *ionicframework.com* must ask the server for approval before we can access the resource, in the form of an HTTP OPTIONS request header.

If we get the error above, then we may not access the resource from the server.

Let's take a look what your origin will be when you're running your app via ionic serve, ionic run, or ionic run -l.

Running in the browser

What happens when you run *ionic serve*?

- A local web server is started up.
- Your browser is opened to point at the local server address.
-

This starts you off looking at your app loaded in a browser on your computer with the address *http://localhost:8100* (if you chose localhost).

Your origin will be *localhost:8100*.

Any AJAX request sent out to a host other than *localhost:8100* will have *localhost:8100* as its origin and thus will require a CORS preflight request to see if it can access the resource.

Running on a device

What happens when you run *ionic run*?

- Your files for the app are copied to the device (or simulator).
- The app runs, thus firing a browser on the phone/simulator to run the files that were copied over, something like: *file://some/path/www/index.html*.

Your *origin* will not exist, since you are running off of a *file://* URI; therefore, any request outwards will not require a CORS request.

Running on a device with livereload

What happens when you run *ionic run -l*?

- A local web server is started up.
- The app runs, thus firing a browser on the phone/simulator to run the files from the server *http://192.168.1.1:8100* (or whatever your local IP address is).

Your *origin* will be *192.168.1.1:8100*.

Any AJAX request sent out to a host other than *192.168.1.1:8100* will require a CORS preflight request to see if it can access the resource.

Explain using an example the "fundamentals" of an ionic application.

Ionic uses AngularJS and Cordova to create Hybrid Apps, for the styling of this app Ionic uses CSS and JavaScript like BootStrap does. When creating the HTML for the app, Ionic specific directives is used like this.

```
<ion-side-menu side="left">
    <ion-header-bar class="bar-dark">
      <h1 class="title">Projects</h1>
      <button class="button button-icon ion-plus" ng-click="newProject()">
      </button>
    </ion-header-bar>
    <ion-content scroll="false">
      <ion-list>
        <ion-item ng-repeat="project in projects" ng-click="selectProject(project,
$index)" ng-class="{active: activeProject == project}">
          {{project.title}}
        </ion-item>
      </ion-list>
    </ion-content>
  </ion-side-menu>
```