Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.

Period-3 An introduction to Cross-Platform App Development with React Native

Note: This description is too big for a single exam-question. It will be divided up into separate questions for the exam

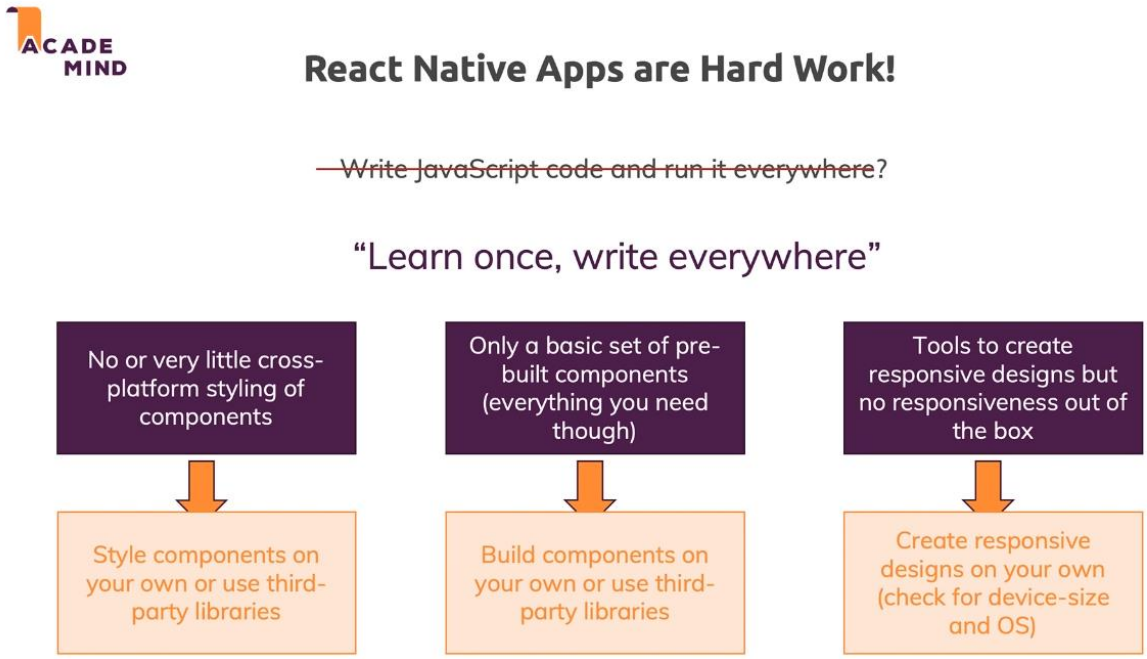
Explain Pros & Cons with React Native + Expo used to implement a Mobile App for Android and IOS, compared to using the Native Tools/languages for the two platforms.

- Pros:
  - Man skal kun lave koden en gang.
  - Man programmerer i JS/React
  - *"Make any app. Run it everywhere. - Build one project that runs natively on all your users' devices"*
  - Det er nemt at komme i gang
  - Kan køres på både Mac og Windows(apple begrænser ikke i samme grad)
  - Det er gratis og open source
  - Kan køres på app, ved scanning af QR-kode
  - Det går hurtigt at bygge videre
  - Expo har features/moduler liggende "ved hånden"
- Cons:
  - Der kan være forskellige standard funktioner alt efter hvilket styresystemer man udvikler på. Fx en <Button> på IOS er bare tekst, hvorimod på Android er det en knap med border i blå. Så man kan komme ud for at ens apps ser lidt anderledes ud på de forskellige telefoner.
  - Expo kan kun bruges i Expo miljøet, som det eneste.
  - React native features/moduler skal importeres/"bygges" selv

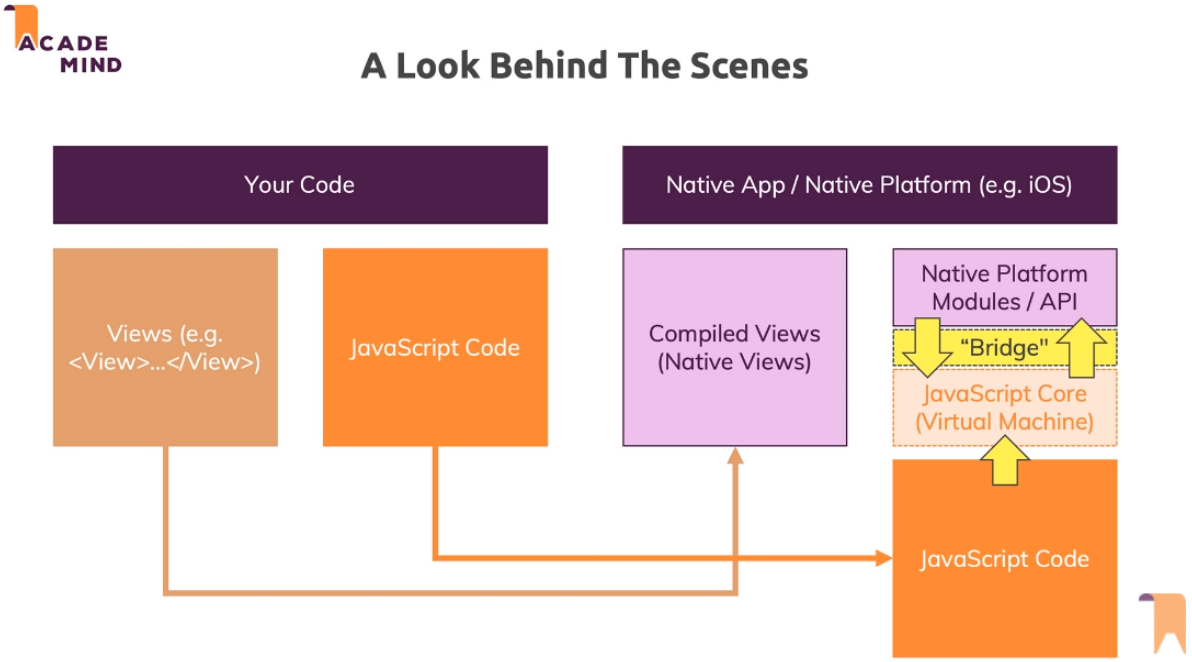
What is meant by the React Native Paradigm "Learn once, write anywhere" compared to for example the original (now dead) idea with Java "Write Once, run everywhere".

- Se evt. video fra 20/11 38:00-40:00
- Når man har lært JS, så kan man med ganske lidt ekstra viden lærer at skrive React, React Native, Vue.js og Angular, Node.js osv.
- De har alle tilfælles at de bygger på JS, og derfor bruger man meget af den samme viden til at udvikle/programmere i det framework man nu har valgt. Man skal altså kun lave lidt små justeringer, da det bygger videre på noget viden man allerede har og derfor kun "forlænger" dette.

Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.



In React Native, which parts of your code gets compiled to Native Code (Widgets) and which parts do NOT?



- Frontend/GUI bliver kompileret til telefonens OS. Enten Android eller iOS.
- GUI'en er noget af det som er tungt at køre og derfor er det smart at det netop bliver kompileret. Dette er blandt andet en af de stærke ting ved React Native.
- JavaScript/Backend koden bliver kørt på en bestemt tråd, som bliver startet af React Native selv. Det er en form for app inde i vores app og kan lidt sammenlignes med JVM(Java Virtual Machine)

Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.

Explain the basic building block in a React Native Application and the difference(s) between a React Application and a React Native App.

- React Applikation:
  - Et library som understøtter både frontend og server.
  - Kan bruges til både mobile apps og hjemmesider
  - Ved brug af DOM bliver nye features renderet med det samme.
  - Component-baseret arkitektur, er godt til genbrug af kode
  - Bruger web components i stedet for native components
- React Native Applikation:
  - Har ikke nogle HTML elementer, så man kan blandt andet ikke bare lave tags osv, men skal bruge et <Text></Text> felt til at displaye skrift på telefonskærmen
  - Er cross-platform mobilt framework som virker på både iOS, Android og Microsoft
  - React Native bruger Reactjs
  - Bruger native components i stedet for web components

Explain and demonstrate ways to handle User Input in a React Native Application

- Ved at bruge hooks (useState) så kan man opdatere værdier, med f.eks. isLoggedIn og setIsLoggedIn, samt i komponenten GetLoginData.js (teamFinderApp/team-finder-app) i toppen
- Man har to "værdier" i en hook, hvor den ene værdi holder den satte værdi og den anden bruger man til at "overskrive"/opdatere den holde værdi med. Lidt alla getter og setter.

Explain and demonstrate how to handle state in a React Native Application

- https://reactnative.dev/docs/state
- Props er værdier som bliver sat af deres "parent" og de er fixeret igennem et komponents livscyklus. Hvor ved data der skal forandres undervejs, bruger vi State.
- En props er en værdi man selv laver og definere, som man så kan læse fra ved det modtagende sted, hvor hvis man skal ændre den værdi man sender afsted, så skal man bruge en state.

Explain and demonstrate how to communicate with external servers, in a React Native Application

- https://reactnative.dev/docs/network
- Det gør vi i vores TeamFinderApp/serverFacade
- Her anvender vi et fetch kald til at "logge ind" og hente andre players data. Ved at modtage fetchdata i form af et request, med en .then() for at få vores promise, som vi så awaiter, for at kunne få vores .json() data.
- Vi anvender et fetch kald til at hente information ned vedrørende vores players. Disse gemmer vi i en variable, som vi sætter ind i en useState, som vi derefter kan bruge. Fra useState'en kan man så trække de værdier ud, som man har behov for. Det kunne for eksempel være et navn eller en position.

Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.

Explain and Demonstrate ways to debug a React Native Application

- console.log()/debug.log()
- Se Max's video inden eksamen. - fra 03:10:14 til 03:41:29.
- https://www.youtube.com/watch?v=qSRrxpdMpVc&feature=youtu.be&t=11416&ab_channel=Academind

Explain and demonstrate how to use Native Device Features in a React Native/Expo app.

- https://reactnative.dev/docs/touchablenativefeedback
- Et eksempel ville være hvordan Android telefoner har en native feature, som f.eks. en "ripple effekt" når man trykker på en knap. Demo i eksemplet for oven.
- Max's video og opgaven - rn-complete-guide demo på ripple effekt.

Explain and demonstrate a React Native Client that uses geo-components (Location, MapView, etc.)

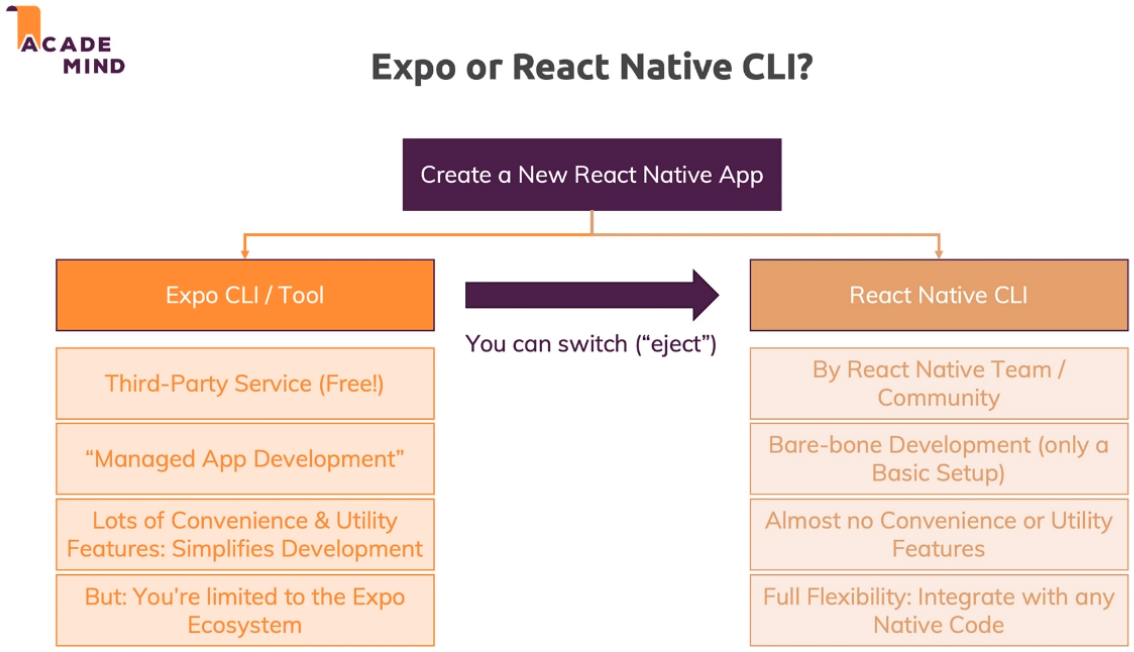
- TeamFinderApp/app.js/getLocationAsync
    - Først beder vi om lov til at anvende geolocation på personens telefon. Hvis vi får lov, sætter vi koordinaterne ind i en useState(setPosition). Herefter har vi nu fået fat i longitude og latitude. Vi sætter også disse koordinater ind i vores useState region. Den bruger vi i næste punkt.
- Hvis setRegion er "null", printes der en text besked med "....fetching data" på skærmen, er den er der nogle data i region, hentes disse og plottes ind i vores MapView. Ud fra vores koordinater bliver der vist et kort i det område der plottes ind.
    - Vi har også en MapView Marker som viser vores position. Der sættes en markør i det område vi befinder os i, indenfor nogle få meter. Den har en title("You") og en description med ens koordinater i.
- Vi har også en variable der hedder otherPlayers, som henter sin information fra et fetch kald op mod serveren. Den modtagere et array og laver dette om til et map, hvor den herefter løber mappet igennem og trækker koordinater og navn osv ud, og plotter de andre spiller ind på kortet i form af en pink farvet marker.

Demonstrate both server and client-side, of the geo-related parts of your implementation of the ongoing semester case.

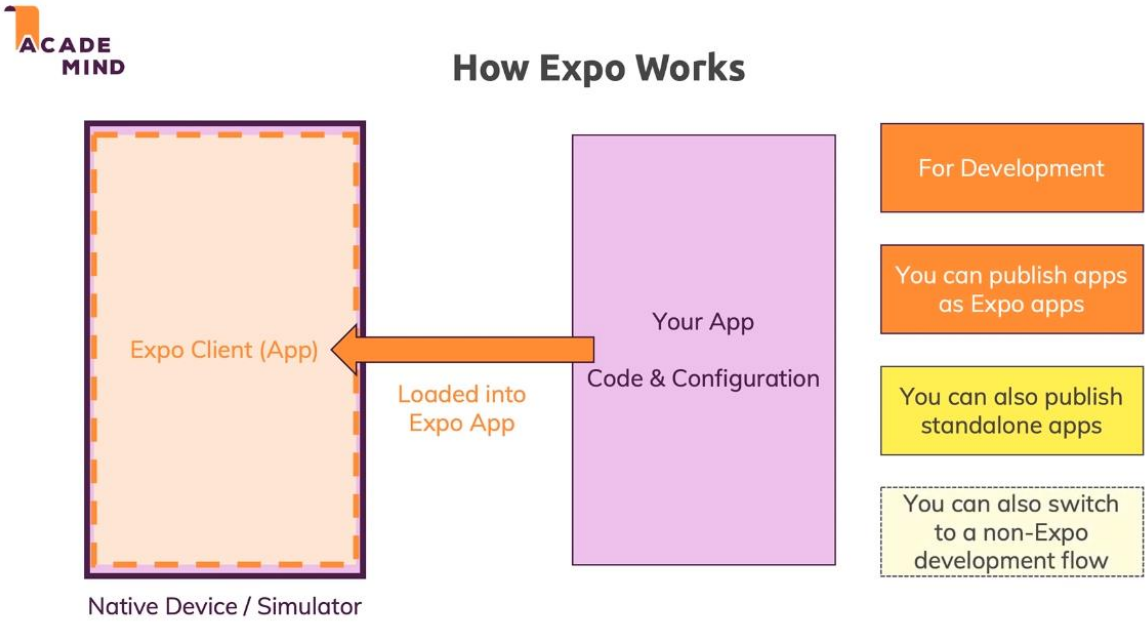
- Se vores Github side fra de sidste dage i Periode 3 og det private repo, som er deployed fra Periode 2 på vores server.

Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.

Forskelle på Expo og “normal” react native:



Hvordan virker Expo så:



Skrevet af Tobias Anker Boldt-Jørgensen, Nina Lisakowski og Andreas Petersen.

Core components:

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrollling `<div>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text