



Sovellusohjelmoinnin jatkokurssi

Oppimispäiväkirja

Antti Venetjoki

SISÄLLYS

1	Viikkotehtävät	3
1.1	Mobiilisovelluskehityksen yleiskuva	3
1.2	React Native	3
1.2.1	Pohdinta	3
1.2.2	Github-linkki.....	3
1.2.3	Todiste ohjelman toiminnasta	4
2	Viikkoharjoitukset 2	5
2.1	Pohdinta	5
2.2	Github-linkki	5
2.3	Kuva ohjelman toiminnasta	5
3	Viikkoharjoitukset 3	6
3.1	SafeAreaView	6
3.2	Responsiivinen layout	6
3.3	StyleSheet-esimerkki	7
	Käytetyt lähteet	8

1 Viikkotehtävät

1.1 Mobiilisovelluskehityksen yleiskuva

Natiivi mobiilikehityksessä kehitetään sovellusta tietyllä alustalle tai käyttöjärjestelmälle. Nykypäivänä yleisimmät ovat iOS ja Android. Natiivikehityksessä myös yleisesti valitaan jokin käyttöjärjestelmä versio, jonka ominaisuuksia käytetään hyväksi kehityksessä. Tämä myös tarkoittaa, että laitteet, joiden käyttöjärjestelmän versio on vanhempi kuin kehityksessä valittu versio, niin sovellus ei toimi kyseisellä laitteella.

Cross-platform kehityksessä käytetään sovelluskehystä, jota käyttäen voidaan kehittää sovellus useammalle alustalle. Muodoltaan sovelluskehukset ovat kirjastoja, joista löytyy hyvin määritelty ohjelmointirajapinta (API), jonka alta löytyy implementaatiot monelle eri alustalle.

Web-kehityksessä kehitetään ohjelmia, jotka toimivat verkkoselaimen päällä. Tällöin käytännössä kaikki laitteet, joista löytyy verkko selain pystyvät käyttämään ohjelmaa.

1.2 React Native

1.2.1 Pohdinta

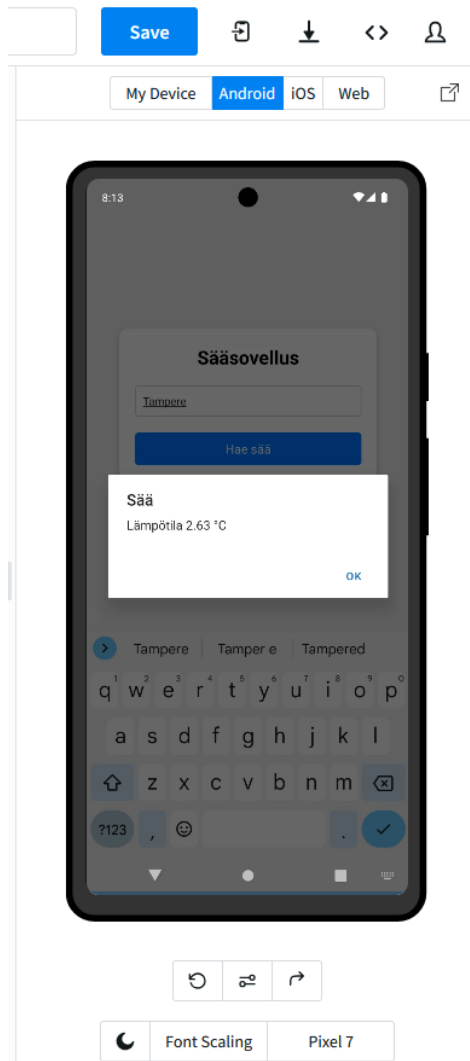
React on suunniteltu web-sovelluksille ja käyttää HTML:ää sekä CSS:ää, kun taas React Native on tarkoitettu mobiilisovelluksille (iOS ja Android), ja se käyttää natiivikomponentteja kuten `<View>` ja `<Text>`. React toimii selaimessa, kun taas React Native kääntää koodin natiiviksi mobiilisovellukseksi. Lisäksi tyylit määritellään eri tavalla: Reactissa CSS:n avulla ja React Nativessa JavaScript-objekteina.

En saanut ympäristöäni toimimaan vielä, joten alla oleva ohjelma on testattu snack.expo.dev verkkoympäristössä.

1.2.2 Github-linkki

github.com

1.2.3 Todiste ohjelman toiminnasta



2 Viikkoharjoitukset 2

2.1 Pohdinta

SafeAreaView on React Nativessa komponentti, joka varmistaa, että sovelluksen sisältö ei jää piiloon laitteiden loven (notch), pyöristettyjen kulmien tai muiden esteiden alle. Se erityisesti hyödyttää moderneja laitteita, joilla on epäsymmetrisiä näyttöjä, kuten iPhone. Käyttämällä *SafeAreaView*:tä varmistat, että käyttöliittymä mukautuu laitteiden turvallisiin alueisiin.

Responsiivinen layout React Nativessa voidaan toteuttaa käyttämällä:

- **Flexboxia** asetteluun.
- **Dimensions API**: avulla saat ruudun koon ja voit mukauttaa tyylejä.
- **Prosenttipohjaiset arvot** leveyteen, korkeuteen, marginaaleihin ja fonttikokoon.
- **Skaalautuvat yksiköt**: Kirjastot kuten *react-native-responsive-screen* tarjoavat prosentuaalisia ratkaisuja.
- **Media query -tyyppinen lähestymistapa** voidaan toteuttaa kirjastoilla kuten *react-native-media-queries*.

2.2 Github-linkki

github.com

2.3 Kuva ohjelman toiminnasta



3 Viikkoharjoitukset 3

3.1 SafeAreaView

SafeAreaView on React Nativen tarjoama komponentti, joka auttaa sovellusta mukautumaan erilaisten mobiililaitteiden näytön muotoihin ja turvallisiin alueisiin. Tämä on erityisen hyödyllistä laitteissa, joissa on lovi ("notch"), pyöristetyt kulmat tai erillinen status bar (kuten iPhone). SafeAreaView luo näytölle "turva-alueen" (safe area), joka estää sisällön näkymisen lovien, status barien tai pyöristettyjen kulmien alueilla. Näin ollen SafeAreaView parantaa sovelluksen käytettävyyttä ja ulkonäköä kaikenkokoisilla ja -muotoisilla laitteilla.

Käyttötarkoitus: SafeAreaView-komponenttia kannattaa käyttää komponenttien ympärillä, joilla on tärkeä visuaalinen tai toiminnallinen rooli, ja jotka tulisi pitää pois näytön reunoilta. Erityisesti ylä- ja alareunan elementit, kuten navigointipalkit, saavat tällä tavoin riittävän tilan eikä niitä estetä.

3.2 Responsiivinen layout

Dimensions API: React Nativessa Dimensions-API mahdollistaa näytön leveyden ja korkeuden hakemisen, jolloin layoutin tai fonttikokojen säädöt voidaan määrittää niiden perusteella.

PixelRatio-moduuli: PixelRatio-moduulin avulla voi säätää tyylimääriä, kuten fonttikokoa ja marginaaleja näytön koon perusteella.

react-native-responsive-screen-kirjasto: Tämä kirjasto sisältää käteviä työkaluja, kuten `widthPercentageToDP` ja `heightPercentageToDP`, joilla saa asetettua prosenttipohjaisia leveyksiä ja korkeuksia.

useWindowDimensions-hook: Tämä Reactin hook mahdollistaa näytön leveyden ja korkeuden dynaamisen hakemisen, jolloin layout mukautuu automaattisesti, jos laitteen suunta muuttuu.

3.3 StyleSheet-esimerkki

```
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 16,
  },
  header: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
  },
  weatherInfo: {
    flex: 3,
    justifyContent: 'center',
    alignItems: 'center',
    fontSize: 18,
    color: '#666',
    padding: 10,
  },
  refreshButton: {
    flex: 1,
    justifyContent: 'flex-end',
    marginBottom: 20,
    padding: 10,
    backgroundColor: '#008CBA',
    borderRadius: 5,
  },
  buttonText: {
    fontSize: 16,
    color: '#fff',
    fontWeight: '600',
    textAlign: 'center',
  },
});

export default styles;
```

Käytetyt lähteet

snack.expo.dev

<https://reactnative.dev/>

<https://home.openweathermap.org/>

<https://github.com/facebook/watchman/tree/v2024.09.30.00>

<https://carbon.now.sh>