ส่วนที่ 1

เริ่มต้นกับ React Native

บทที่ 1 จะช่วยให้คุณเริ่มต้นใช้งานได้โดยดูว่า React Native คืออะไร มันทำงานอย่างไร ความสัมพันธ์กับ React คืออะไร และเมื่อใดที่คุณอาจต้องการใช้ React Native (และเมื่อคุณอาจไม่ทำ) บทนี้ให้ภาพรวมของส่วนประกอบของ React Native ซึ่งเป็นแกนหลักของ React Native สรุปว่าด้วยการสร้างโปรเจ็กต์ React Native ขนาดเล็ก

บทที่ 2 ครอบคลุมถึงสถานะและคุณสมบัติ: คืออะไร ทำงานอย่างไร และทำไมมีความสำคัญในการพัฒนาแอปพลิเคชัน React Native นอกจากนี้ยังครอบคลุมถึงข้อมูลจำเพาะของ React Component และวิธีการ React lifecycle

ในบทที่ 3 คุณจะสร้างแอป React Native แรกของคุณ—แอป Todo—จากพื้นดินขึ้น คุณจะได้เรียนรู้เกี่ยวกับการใช้เมนูนักพัฒนาซอฟต์แวร์ใน iOS และ Android สำหรับการดีบักแอป

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

3

เริ่มต้นกับ React Native

บทนี้ครอบคลุม

- แนะนำ React Native

- จุดแข็งของ React Native
- การสร้างส่วนประกอบ
- การสร้างโครงการเริ่มต้น

การพัฒนาแอปพลิเคชันมือถือแบบเนทีฟอาจมีความซับซ้อน ด้วยความซับซ้อน
สภาพแวดล้อม เฟรมเวิร์กที่ละเอียด และการคอมไพล์ที่ใช้เวลานานที่นักพัฒนาต้องเผชิญ
การพัฒนาแอปพลิเคชั่นมือถือที่มีคุณภาพนั้นไม่ใช่เรื่องง่าย ไม่แปลกใจเลยที่
ตลาดได้เห็นส่วนแบ่งของการแก้ปัญหามาสู่ฉากที่พยายามจะแก้ปัญหา
ปัญหาที่ควบคู่ไปกับการพัฒนา Native Mobile Application และพยายามทำ
ให้
มันง่ายกว่า
แก่นของความซับซ้อนนี้คืออุปสรรคของการพัฒนาข้ามแพลตฟอร์ม NS
แพลตฟอร์มต่าง ๆ นั้นแตกต่างกันโดยพื้นฐานและไม่ได้แบ่งปันการพัฒนามากนัก-
สภาพแวดล้อมตัวเลือก API หรือรหัส ด้วยเหตุนี้เราจึงต้องแยกทีม
ทำงานในแต่ละแพลตฟอร์มซึ่งทั้งแพงและไม่มีประสิทธิภาพ
แต่นี่เป็นช่วงเวลาที่น่าตื่นเต้นในการพัฒนาแอปพลิเคชั่นมือถือ เรากำลังเป็นพยาน
กระบวนทัศน์ใหม่ในแนวการพัฒนามือถือและ React Native อยู่บน
แนวหน้าของการเปลี่ยนแปลงนี้ในวิธีที่เราสร้างและออกแบบแอปพลิเคชันมือถือ ตอนนี้
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 4

สามารถสร้างแอพข้ามแพลตฟอร์มที่มีประสิทธิภาพเช่นเดียวกับเว็บแอปพลิเคชันด้วย
ภาษาเดียวและทีมเดียว ด้วยการเพิ่มขึ้นของอุปกรณ์พกพาและส่วนย่อย
ความต้องการพรสวรรค์ที่เพิ่มขึ้นอย่างต่อเนื่องทำให้เงินเดือนนักพัฒนาสูงขึ้นและสูงขึ้น
React
Native นำเสนอความสามารถในการส่งมอบแอปพลิเคชั่นที่มีคุณภาพในทุก
แพลตฟอร์มที่
เวลาและค่าใช้จ่ายเพียงเล็กน้อย ในขณะที่ยังคงมอบประสบการณ์ผู้ใช้ที่มีคุณภาพสูงและ
ประสบการณ์นักพัฒนาที่น่ายินดี

1.1 แนะนำ React และ React Native

React Native เป็นเฟรมเวิร์กสำหรับการสร้างแอพมือถือดั้งเดิมใน JavaScript
โดยใช้
โต้ตอบไลบรารี JavaScript; React Native Code จะคอมไพล์ไปยัง
องค์ประกอบดั้งเดิมของจริง ถ้า
คุณไม่แน่ใจว่า React คืออะไร มันเป็นไลบรารี JavaScript ที่โอเพ่นซอร์สและใช้
งานภายใน
เฟสบุ๊ค. เดิมใช้เพื่อสร้างส่วนต่อประสานผู้ใช้สำหรับเว็บแอปพลิเคชัน มันมี
ตั้งแต่มีการพัฒนาและตอนนี้ยังใช้สร้างแอปพลิเคชันฝั่งเซิร์ฟเวอร์และมือถือได้อีกด้วย
(โดยใช้ React Native)
React Native มีอะไรให้ทำมากมาย นอกจากจะได้รับการสนับสนุนและโอเพ่น
ซอร์สโดย
Facebook ก็ยังมีชุมชนผู้สร้างแรงบันดาลใจมากมายอยู่เบื้องหลัง ใบหน้า-
กลุ่มหนังสือที่มีผู้ใช้หลายล้านคนขับเคลื่อนโดย React Native เช่นเดียวกับ
Face-

หนังสือตัวจัดการโฆษณา Airbnb, Bloomberg, Tesla, Instagram, Ticketmaster, SoundCloud,
Uber, Walmart, Amazon และ Microsoft เป็นบริษัทอื่นๆ ที่ลงทุน-
หรือใช้ React Native ในการผลิต
ด้วย React Native นักพัฒนาสามารถสร้างมุมมองแบบเนทีฟและเข้าถึงเฉพาะแพลตฟอร์มเฉพาะได้
ส่วนประกอบโดยใช้จาวาสคริปต์ สิ่งนี้ทำให้ React Native แตกต่างจากเฟรมแอพไฮบริดอื่น ๆ
ทำงานเหมือน Cordova และ Ionic ซึ่งเป็นแพ็คเกจการดูเว็บที่สร้างโดยใช้ HTML และ CSS ลงในa
แอปพลิเคชันดั้งเดิม แต่ React Native จะใช้ JavaScript และคอมไพล์เป็นภาษาเนทีฟที่แท้จริง
แอปพลิเคชันที่สามารถใช้ API และส่วนประกอบเฉพาะแพลตฟอร์ม ทางเลือกเช่น Xama-
rin ใช้แนวทางเดียวกัน แต่แอป Xamarin สร้างขึ้นโดยใช้ C # ไม่ใช่ JavaScript หลายเว็บ
นักพัฒนามีประสบการณ์จาวาสคริปต์ ซึ่งช่วยลดความยุ่งยากในการเปลี่ยนจากเว็บเป็นมือถือ
การพัฒนาแอพ
มีประโยชน์มากมายในการเลือก React Native เป็นเฟรมแอปพลิเคชันมือถือ-
งาน. เนื่องจากแอปพลิเคชันแสดงส่วนประกอบดั้งเดิมและ API โดยตรง ความเร็ว และประสิทธิภาพดีกว่าเฟรมเวิร์กไฮบริดเช่น Cordova และ Ionic มาก
ด้วย React Native เรากำลังเขียนแอปพลิเคชันทั้งหมดโดยใช้ lan-

วัด: JavaScript. เราสามารถนำโค้ดกลับมาใช้ใหม่ได้จำนวนมาก ซึ่งจะช่วยลดเวลาในการจัดส่ง
แอปพลิเคชันข้ามแพลตฟอร์ม และการจ้างและค้นหานักพัฒนา JavaScript ที่มีคุณภาพคือ
ง่ายกว่าและถูกกว่าการจ้างนักพัฒนา Java, Objective C หรือ Swift มาก ซึ่งนำไปสู่
กระบวนการโดยรวมที่ราคาไม่แพง
หมายเหตุแอปพลิเคชัน React Native สร้างขึ้นโดยใช้ JavaScript และ JSX เราจะ dis-
พูดถึง JSX ในเชิงลึกในหนังสือเล่มนี้ แต่ตอนนี้คิดว่ามันเป็นไวยากรณ์ JavaScript
ส่วนขยายที่ดูเหมือน HTML หรือ XML
เราจะเจาะลึกลงไปใน React ในบทที่ 2 ก่อนถึงเวลานั้น มาพูดถึงแกนหลักกัน
แนวคิดเพื่อเป็นการแนะนำ
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

ขอแนะนำ React และ React Native

1.1.1 คลาส React พื้นฐาน

ส่วนประกอบเป็นส่วนประกอบสำคัญของแอปพลิเคชัน React หรือ React Native รายการ

จุดสมัครเป็นส่วนประกอบที่ต้องการและประกอบขึ้นจากส่วนประกอบอื่น
เต็นท์ ส่วนประกอบเหล่านี้อาจต้องการส่วนประกอบอื่นๆ เป็นต้น
องค์ประกอบ React Native มีสองประเภทหลัก: stateful และ stateless นี่มัน
ตัวอย่างขององค์ประกอบ stateful โดยใช้คลาส ES6:

```
คลาส HelloWorld ขยาย React.Component {
ตัวสร้าง () {
ซุปเปอร์()
this.state = { ชื่อ: 'คริส' }
}
แสดงผล () {
กลับ (
<SomeComponent />
)
}
}
```

และนี่คือตัวอย่างขององค์ประกอบไร้สัญชาติ:

```
const HelloWorld = () => (
<SomeComponent />
)
```

ความแตกต่างหลัก ๆ คือส่วนประกอบไร้สัญชาติไม่ได้เชื่อมโยงกับกลไกการปรุงยาใดๆ
ods และไม่มีสถานะของตนเอง ดังนั้นข้อมูลใด ๆ ที่จะแสดงผลจะต้องได้รับเป็น
คุณสมบัติ (อุปกรณ์ประกอบฉาก) เราจะพูดถึงวิธีวงจรชีวิตในเชิงลึกในบทที่ 2 แต่
สำหรับตอนนี้เรามาดูพวกเขาและดูชั้นเรียนกันก่อน

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ, สไตล์ชีต } จาก 'react-native'
คลาส HelloWorld ขยาย React.Component {
```

```
ตัวสร้าง () {
ซุปเปอร์()
this.state = {
ชื่อ: 'React Native in Action'
}
}
componentDidMount () {
console.log('ติด..')
}
แสดงผล () {
กลับ (
<ดู style={styles.container}>
<Text>{this.state.name}</Text>
</ดู>
)
}
```

ตัวสร้างตั้งค่าสถานะวัตถุ

ที่มีชื่อคุณสมบัติ

วิธีวงจรชีวิตขั้นสุดท้าย

โทรแสดงผล ()

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 6


```
}
```

รูปแบบ const = StyleSheet.create ({
คอนเทนเนอร์: {
ขอบบน: 100,
ดิ้น: 1
}
})
หมายเหตุสิ่งที่ควรคำนึงถึงเมื่อเราพูดถึงวิธีการต่อไปนี้คือ
แนวคิดของการติดตั้ง เมื่อสร้างส่วนประกอบขึ้นมา ส่วนประกอบ React
วงจรชีวิตถูกสร้างอินสแตนซ์ ทริกเกอร์วิธีการที่ใช้ในการแสดงรายการ 1.1
ที่ด้านบนของไฟล์ คุณต้องมีReactจาก'react'เช่นเดียวกับView , Textและ
StyleSheet จาก'ตอบสนองพื้นเมือง' มุมมองเป็นองค์ประกอบพื้นฐานที่สำคัญที่สุด
สำหรับ
การสร้าง React Native component และ UI โดยทั่วไปและสามารถคิดได้
เหมือนa
div ใน HTML ข้อความช่วยให้คุณสร้างองค์ประกอบข้อความและเปรียบได้กับ
แท็กช่วง
ใน HTML StyleSheetให้คุณสร้างออบเจ็กต์สไตล์เพื่อใช้ในแอปพลิเคชัน สอง
คนนี้
แพ็คเกจ (reactและreact-native) มีให้ใช้งานเป็นโมดูล npm
เมื่อส่วนประกอบโหลดครั้งแรก คุณตั้งค่าสถานะวัตถุด้วยชื่อคุณสมบัติใน
ตัวสร้าง เพื่อให้ข้อมูลในแอปพลิเคชัน React Native เป็นไดนามิก จะต้องเป็นอย่าง
ใดอย่างหนึ่ง
ตั้งอยู่ในรัฐหรือส่งต่อเป็นอุปกรณ์ประกอบฉาก ที่นี่คุณตั้งค่าสถานะในตัวสร้างและ
จึงสามารถเปลี่ยนแปลงได้หากต้องการ โดยโทร
this.setState({

ชื่อ: 'ชื่ออื่น'
})
ซึ่งแสดงผลองค์ประกอบ การตั้งค่าตัวแปรในสถานะช่วยให้คุณอัปเดต
ค่าในส่วนอื่นของส่วนประกอบ
Render นั้นถูกเรียก: มันตรวจสอบอุปกรณ์และสถานะแล้วต้องคืนค่าเดียว
ตอบสนององค์ประกอบพื้นเมืองnullหรือเท็จ หากคุณมีองค์ประกอบย่อยหลาย
องค์ประกอบ จะต้อง
ห่อด้วยองค์ประกอบหลัก ในที่นี้ ส่วนประกอบ สไตล์ และข้อมูลจะรวมกันเป็น
สร้างสิ่งที่จะแสดงผลไปยัง UI
วิธีสุดท้ายในวงจรคือcomponentDidMount หากคุณต้องการทำ API . ใดๆ
การโทรหรือคำขอ AJAX เพื่อรีเซ็ตสถานะ ซึ่งมักจะเป็นวิธีที่ดีที่สุดในการทำ
เช่นนั้น ในที่สุด
UI แสดงผลไปยังอุปกรณ์ และคุณสามารถดูผลลัพธ์ได้

1.1.2 วงจรชีวิตปฏิกิริยา

เมื่อสร้างคลาส React Native เมธอดจะถูกสร้างอินสแตนซ์ที่คุณเชื่อมต่อได้
วิธีการเหล่านี้เรียกว่าวิธีวงจรชีวิต และเราจะกล่าวถึงในเชิงลึกในบทที่ 2
วิธีการในรายการ 1.1 คือconstructor , componentDidMountและ
renderแต่
มีอีกสองสามรายการและพวกเขาทั้งหมดมีกรณีการใช้งานของตัวเอง
วิธีวงจรชีวิตเกิดขึ้นพร้อมกันและช่วยจัดการสถานะของส่วนประกอบด้วย
เป็นรันโค้ดในแต่ละขั้นตอนของวิธีการ หากคุณต้องการ วิธีวงจรชีวิตที่จำเป็นเท่านั้นคือ
แสดงผล; ส่วนอื่นๆ ทั้งหมดเป็นตัวเลือก เมื่อทำงานกับ React Native คุณเป็น
พื้นฐาน
นับการทำงานด้วยวิธีวงจรชีวิตและข้อกำหนดเดียวกันกับที่คุณใช้กับ React

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

7
สิ่งที่ควรรู้

1.2 สิ่งที่คุณจะได้เรียนรู้

ในหนังสือเล่มนี้ เราจะครอบคลุมทุกสิ่งที่คุณจำเป็นต้องรู้เพื่อสร้างแอปพลิเคชันมือถือที่มีประสิทธิภาพ-
สำหรับ iOS และ Android โดยใช้ React Native framework เพราะ React Native
สร้างขึ้นโดยใช้ไลบรารี React เราจะเริ่มในบทที่ 2 โดยการครอบคลุมและทั่วถึง
อธิบายการทำงานของ React
จากนั้นเราจะพูดถึงการจัดสไตล์โดยสัมผัสถึงคุณสมบัติการจัดแต่งทรงผมส่วนใหญ่ที่มีใน
กรอบ. เนื่องจาก React Native ใช้ flexbox ในการจัดวาง UI เราจะเจาะลึกลงไป
flexbox ทำงานอย่างไรและหารือเกี่ยวกับคุณสมบัติ flexbox ทั้งหมด หากคุณเคยใช้ flexbox ใน CSS
สำหรับเลย์เอาต์บนเว็บ ทั้งหมดนี้คุณจะคุ้นเคย แต่โปรดจำไว้ว่า flexbox
การใช้งานโดย React Native นั้นไม่เหมือนกัน 100%
จากนั้นเราจะพูดถึงองค์ประกอบดั้งเดิมมากมายที่มาพร้อมกับเฟรมเวิร์ก

ออกจากกล่องและศึกษาวิธีการทำงานของแต่ละอย่าง ใน React Native ส่วนประกอบ
โดยพื้นฐานแล้วเป็นโค้ดที่มีฟังก์ชันเฉพาะหรือองค์ประกอบ UI และ can
ใช้งานง่ายในแอปพลิเคชัน ส่วนประกอบได้รับการคุ้มครองอย่างกว้างขวางตลอดทั้งนี้
จองเพราะเป็นส่วนประกอบสำคัญของแอปพลิเคชัน React Native
มีหลายวิธีในการใช้การนำทาง โดยแต่ละวิธีมีความแตกต่าง ข้อดี และ
ข้อเสีย เราจะหารือเกี่ยวกับการนำทางในเชิงลึกและครอบคลุมถึงวิธีสร้างการนำทางที่มี
ประสิทธิภาพโดยใช้
API การนำทางที่สำคัญที่สุด เราจะครอบคลุมไม่เพียง แต่การนำทางดั้งเดิม
API ที่ออกมาจากกล่องด้วย React Native แต่ยังรวมถึงโปรเจ็กต์ชุมชนอีกสอง
สามตัว
ects ใช้ได้ผ่าน npm
ต่อไป เราจะหารือในเชิงลึกทั้ง API ข้ามแพลตฟอร์มและเฉพาะแพลตฟอร์ม
สามารถทำได้ใน React Native และวิธีการทำงาน ได้เวลาเริ่มทำงานแล้ว
ด้วยข้อมูลโดยใช้คำขอเครือข่าย, AsyncStorage (รูปแบบของที่จัดเก็บในเครื่อง),
Firebase และ
เว็บซ็อกเก็ต จากนั้นเราจะเจาะลึกถึงสถาปัตยกรรมข้อมูลที่แตกต่างกันและดูว่าแต่ละ
สถาปัตยกรรมเป็นอย่างไร
ทำงานเพื่อจัดการกับสถานะของแอปพลิเคชัน สุดท้าย เราจะดูการทดสอบและข้อ
แตกต่างบางประการ-
วิธีการทดสอบใน React Native

1.3 สิ่งที่ควรรู้

เพื่อให้ได้ประโยชน์สูงสุดจากหนังสือเล่มนี้ คุณควรจะมีความรู้ระดับเริ่มต้นถึงระดับกลาง
ของจาวาสคริปต์ งานส่วนใหญ่ของคุณจะเสร็จสิ้นด้วยบรรทัดคำสั่ง ดังนั้น พื้นฐานภายใต้
จำเป็นต้องมีวิธีการใช้บรรทัดคำสั่งด้วย คุณควรเข้าใจด้วย
npm คืออะไรและทำงานอย่างไรในระดับพื้นฐานเป็นอย่างน้อย หากคุณกำลังจะสร้างใน
iOS ความเข้าใจพื้นฐานของ Xcode นั้นมีประโยชน์และจะทำให้สิ่งต่าง ๆ เร็วขึ้น แต่ไม่ใช่
ที่จำเป็น. ในทำนองเดียวกัน หากคุณกำลังสร้างสำหรับ Android ความเข้าใจพื้นฐานเกี่ยวกับ Android
Studio จะเป็นประโยชน์แต่ไม่จำเป็น
ความรู้พื้นฐานเกี่ยวกับคุณลักษณะ JavaScript ที่ใหม่กว่าที่นำมาใช้ใน ES2015
การเปิดตัวภาษาโปรแกรม JavaScript นั้นมีประโยชน์แต่ไม่จำเป็น บาง
ความรู้เชิงแนวคิดของเฟรมเวิร์ก MVC และสถาปัตยกรรมหน้าเดียวก็ดีเช่นกัน
แต่ไม่จำเป็น
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

1.4 ทำความเข้าใจว่า React Native ทำงานอย่างไร

มาดูกันว่า React Native ทำงานอย่างไร โดยพูดถึง JSX, threading model, React,
การไหลของข้อมูลแบบทิศทางเดียว และอื่นๆ

1.4.1 JSX

React และ React Native ต่างก็สนับสนุนการใช้ JSX JSX นั้นเป็นส่วนขยายของไวยากรณ์
กับ JavaScript ที่ดูคล้ายกับ XML คุณสามารถสร้างส่วนประกอบ React Native ได้
ไม่มี JSX แต่ JSX ทำให้ React และ React Native อ่านง่ายและง่ายขึ้นมาก บำรุงรักษา. JSX อาจดูแปลกในตอนแรก แต่ทรงพลังมากและคนส่วนใหญ่
เติบโตเพื่อรักมัน

1.4.2 การทำเกลียว

การทำงานของ JavaScript ทั้งหมด เมื่อโต้ตอบกับแพลตฟอร์มดั้งเดิม จะเสร็จสิ้นใน
เดือนกันยายน
arate thread ช่วยให้ส่วนต่อประสานกับผู้ใช้และแอนิเมชั่นทำงานได้อย่างราบรื่น
เธรดนี้เป็นที่ที่แอปพลิเคชัน React ใช้งานอยู่ และการเรียกใช้ API ทั้งหมด เหตุการณ์การสัมผัส
และมีการประมวลผลการโต้ตอบ เมื่อมีการเปลี่ยนแปลงในองค์ประกอบที่ได้รับการสนับสนุนโดยเนทีฟ
การอัปเดตจะถูกแบทช์และส่งไปยังฝั่งเนทีฟ สิ่งนี้จะเกิดขึ้นในตอนท้ายของแต่ละ iter-
การวนรอบเหตุการณ์ สำหรับแอปพลิเคชัน React Native ส่วนใหญ่ ตรรกะทางธุรกิจจะทำงานบน

เธรด JavaScript

1.4.3 ปฏิกิริยา

คุณสมบัติที่ยอดเยี่ยมของ React Native คือใช้ React React เป็น JavaScript แบบโอเพ่นซอร์ส
ห้องสมุดที่สนับสนุนโดย Facebook เดิมทีได้รับการออกแบบมาเพื่อสร้างแอปพลิเคชัน
และแก้ปัญหาบนเว็บ เฟรมเวิร์กนี้ได้รับความนิยมอย่างมากตั้งแต่
การเปิดตัวโดยมีบริษัทที่จัดตั้งขึ้นหลายแห่งใช้ประโยชน์จากการแสดงผลอย่างรวดเร็ว
การบำรุงรักษา และ UI ที่เปิดเผย เหนือสิ่งอื่นใด
การจัดการ DOM แบบดั้งเดิมนั้นช้าและมีราคาแพงในแง่ของประสิทธิภาพและ
ควรย่อให้เล็กสุด React ข้าม DOM ดั้งเดิมด้วยสิ่งที่เรียกว่า
virtual DOM: โดยพื้นฐานแล้ว สำเนาของ DOM จริงในหน่วยความจำที่
เปลี่ยนแปลงเฉพาะเมื่อ
เปรียบเทียบ DOM เสมือนเวอร์ชันใหม่กับ DOM เสมือนเวอร์ชันเก่า นี้
ลดจำนวนการดำเนินการ DOM ที่จำเป็นเพื่อให้ได้สถานะใหม่

1.4.4 การไหลของข้อมูลแบบทิศทางเดียว

React และ React Native เน้นการไหลของข้อมูลแบบทิศทางเดียวหรือทาง
เดียว เพราะว่า
วิธีสร้างแอปพลิเคชัน React Native โฟลว์ข้อมูลทางเดียวนี้ทำได้ง่าย

1.4.5 ความแตกต่าง

React ใช้แนวคิดในการทำให้แตกต่างและนำไปใช้กับองค์ประกอบดั้งเดิม ต้องใช้ UI
และ .ของคุณ
ส่งข้อมูลจำนวนน้อยที่สุดไปยังเธรดหลักเพื่อแสดงผลด้วยองค์ประกอบเนทีฟ

เต็นท์ UI ถูกแสดงอย่างเปิดเผยตามสถานะ และ React ใช้ความแตกต่างกับ
ส่งการเปลี่ยนแปลงที่จำเป็นข้ามสะพาน
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 9

9
ทำความเข้าใจว่า React Native ทำงานอย่างไร

1.4.6 การคิดในองค์ประกอบ

เมื่อสร้าง UI ใน React Native การคิดว่าแอปพลิเคชันของคุณเป็นอยู่นั้นมี
ประโยชน์
ที่ประกอบด้วยการรวบรวมส่วนประกอบ คิดถึงวิธีตั้งค่าเพจ คุณ
ได้ทำสิ่งนี้ตามแนวคิดแล้ว แต่ใช้แนวคิด ชื่อ หรือชื่อคลาส เช่น ส่วนหัว
ส่วนท้าย เนื้อหา แถบด้านข้าง และอื่นๆ ด้วย React Native คุณสามารถตั้งชื่อ
ส่วนประกอบเหล่านี้ได้
ที่เหมาะสมกับคุณและนักพัฒนาคนอื่นๆ ที่อาจกำลังใช้โค้ดของคุณอยู่
ง่ายต่อการนำคนใหม่เข้าสู่โครงการหรือมอบโครงการให้คนอื่น
สมมติว่าผู้ออกแบบได้มอบตัวอย่างจำลองที่แสดงในรูปที่ 1.1 Let's
คิดหาวิธีสร้างแนวคิดนี้เป็นส่วนประกอบ
สิ่งแรกที่ต้องทำคือแบ่งองค์ประกอบ UI ทางจิตใจออกเป็นสิ่งที่พวกเขาเป็นตัวแทน
ตัวอย่าง mockup มี header bar และใน header bar มี title และ a
รูปที่ 1.1 ตัวอย่างการออกแบบแอพ

หน้า 10

ปุ่มเมนู ด้านล่างส่วนหัวคือแถบแท็บ และภายในแถบแท็บจะมีแถบแยกสามตัว แท็บ ดูส่วนที่เหลือของแบบจำลองและคิดว่ารายการอื่นๆ คืออะไร เหล่านี้ รายการที่คุณระบุจะถูกแปลเป็นส่วนประกอบ นี่คือวิธีที่คุณควร

คิดเกี่ยวกับการเขียน UI เมื่อทำงานกับ React Native: แยกย่อยทั่วไป องค์ประกอบใน UI ให้เป็นส่วนประกอบที่ใช้ซ้ำได้ และกำหนดอินเทอร์เฟซตามนั้น เมื่อคุณต้องการองค์ประกอบในอนาคต องค์ประกอบดังกล่าวจะพร้อมใช้ซ้ำได้ การแบ่งองค์ประกอบ UI ออกเป็นส่วนประกอบที่ใช้ซ้ำได้นั้นดีสำหรับการใช้รหัสซ้ำและเช่นกัน

ทำให้รหัสของคุณชัดเจนและเข้าใจได้ ตัวอย่างเช่น แทนที่จะเป็น 12 บรรทัดของรหัสการดำเนินการส่วนท้ายองค์ประกอบอาจจะเรียกว่าส่วนท้าย กำลังดูรหัสที่สร้างขึ้นด้วยวิธีนี้ มันง่ายกว่ามากที่จะให้เหตุผลและรู้ว่าเกิดอะไรขึ้น

รูปที่ 1.2 แสดงให้เห็นว่าการออกแบบในรูปที่ 1.1 สามารถแยกออกเป็นฉันได้อย่างไร อธิบายไว้ ชื่อสามารถเป็นอะไรก็ได้ที่เหมาะกับคุณ บางส่วนของรายการคือ จัดกลุ่มเข้าด้วยกัน—ฉันแยกรายการตามหลักเหตุผลและจัดกลุ่มส่วนประกอบ-กางเต็นท์ตามแนวคิด

ต่อไปเรามาดูกันว่าสิ่งนี้จะมีลักษณะอย่างไรโดยใช้โค้ด React Native จริง ก่อนอื่นมาดูที่

องค์ประกอบ UI หลักปรากฏบนหน้าอย่างไร:

<Header />
<แถบแท็บ />
<ProjectList />
<ส่วนท้าย />

ต่อไปเรามาดูกันว่าองค์ประกอบย่อยมีลักษณะอย่างไร:

แถบแท็บ:

<TabBarItem />
<TabBarItem />
<TabBarItem />

รายการโครงการ:

// เพิ่มองค์ประกอบโครงการสำหรับแต่ละโครงการในรายการ:

<โครงการ />

ฉันใช้ชื่อที่ประกาศไว้ในรูปที่ 1.2 แต่อาจเป็นอะไรก็ได้ที่สมเหตุสมผล

คุณ.

1.5 ยอมรับจุดแข็งของ React Native

ดังที่ได้กล่าวไว้ก่อนหน้านี้ หนึ่งในจุดแข็งหลักที่ React Native กำลังทำก็คือมัน ใช้ React React เช่นเดียวกับ React Native เป็นโครงการโอเพ่นซอร์สที่ได้รับการสนับสนุนจาก Facebook เนื่องจาก

ในช่วงเวลาของการเขียนนี้ React มีดาวมากกว่า 100,000 ดวงและมากกว่า 1,100 ผลงาน

utors บน GitHub—นั่นคือความสนใจและการมีส่วนร่วมของชุมชนในโครงการเป็นอย่างมาก

ทำให้ง่ายต่อการเดิมพันในฐานะนักพัฒนาหรือผู้จัดการโครงการ เพราะปฏิกิริยาคือ พัฒนา บำรุงรักษา และใช้งานโดย Facebook มีวิศวกรที่มีความสามารถมากที่สุด ไม่สนใจโลกที่คอยดูแล ผลักดันไปข้างหน้า และเพิ่มคุณสมบัติใหม่ และมัน

คงจะไม่จากไปในเร็วๆ นี้

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

11

ยอมรับจุดแข็งของ React Native

1.5.1 ความพร้อมใช้งานของนักพัฒนา

ด้วยต้นทุนที่เพิ่มขึ้นและความพร้อมใช้งานที่ลดลงของนักพัฒนามือถือดั้งเดิม React Native เข้าสู่ตลาดด้วยข้อได้เปรียบที่สำคัญเหนือการพัฒนา Native: มันนำข้อดีของ

ความมั่งคั่งของนักพัฒนาเว็บและ JavaScript ที่มีความสามารถที่มีอยู่และมอบให้แก่พวกเขา

แพลตฟอร์มอื่นที่จะสร้างโดยไม่ต้องเรียนรู้ภาษาใหม่

1.5.2 ประสิทธิภาพการทำงานของนักพัฒนา

ตามเนื้อผ้า ในการสร้างแอปพลิเคชันมือถือข้ามแพลตฟอร์ม คุณจำเป็นต้องมีทั้ง Android

ทีมและทีม iOS React Native ให้คุณสร้าง Android, iOS และ (เร็ว ๆ นี้) แอปพลิเคชัน Windows ที่ใช้ภาษาการเขียนโปรแกรมเดียว, JavaScript และบางที

แม้แต่ทีมเดียว เวลาในการพัฒนาและต้นทุนการพัฒนาลดลงอย่างมาก

รูปที่ 1.2 โครงสร้าง App แบ่งออกเป็นส่วนประกอบต่างๆ

หน้า 12

ในขณะที่เพิ่มผลผลิต ในฐานะนักพัฒนาพื้นเมือง สิ่งที่ยอดเยี่ยมในการมาที่a แพลตฟอร์มเช่นนี้คือความจริงที่ว่าคุณไม่ได้ผูกติดอยู่กับการเป็นเพียง Android อีกต่อไป
หรือนักพัฒนา iOS เปิดประตูรับโอกาสมากมาย นี่เป็นข่าวดีสำหรับ
นักพัฒนา JavaScript เช่นกัน ทำให้พวกเขาใช้เวลาทั้งหมดอยู่ในสภาวะจิตใจเดียว เมื่อสลับไปมาระหว่างโครงการบนเว็บและมือถือ นอกจากนี้ยังเป็นชัยชนะสำหรับทีมที่เคยเป็น
ตามธรรมเนียมแล้วแยกระหว่าง Android และ iOS เพราะตอนนี้สามารถทำงานร่วมกันได้บน
ฐานรหัสเดียว เพื่อขีดเส้นใต้จุดเหล่านี้ คุณสามารถแบ่งปันสถาปัตยกรรมข้อมูลของคุณไม่เพียงแค่ข้ามแพลตฟอร์มเท่านั้น แต่ยังรวมถึงบนเว็บด้วย หากคุณใช้บางอย่างเช่น Redux (dis-
สาปแช่งในบทที่ 12)

1.5.3 ประสิทธิภาพ

หากคุณปฏิบัติตามโซลูชันข้ามแพลตฟอร์มอื่นๆ คุณอาจทราบถึงโซลูชันต่างๆ เช่น PhoneGap, Cordova และ Ionic แม้ว่าสิ่งเหล่านี้จะเป็นวิธีแก้ปัญหาที่ได้ผล sus คือประสิทธิภาพยังไม่ทันกับประสบการณ์ที่แอพเนทีฟมอบให้

นี่คือจุดที่ React Native ก็เปล่งประกายเช่นกันเพราะประสิทธิภาพมักจะไม่สังเกต-แตกต่างอย่างมากจากแอปมือถือดั้งเดิมที่สร้างโดยใช้ Objective-C/Swift หรือ Java

1.5.4 การไหลของข้อมูลทางเดียว

การไหลของข้อมูลทางเดียวแยก React และ React Native ออกจากเฟรม JavaScript อื่น ๆ ส่วนใหญ่
ใช้งานได้และเฟรมเวิร์ก MVC ใด ๆ React รวมกระแสข้อมูลทางเดียวจากด้านบน
ส่วนประกอบระดับลงไปจนสุด (ดูรูปที่ 1.3) ทำให้แอปพลิเคชันง่ายขึ้นมาก
ในการให้เหตุผล เพราะมีแหล่งความจริงแหล่งเดียวสำหรับชั้นข้อมูลซึ่งต่างจากการที่
มันกระจัดกระจายเกี่ยวกับแอปพลิเคชัน เราจะดูรายละเอียดเพิ่มเติมในเล่มนี้

ข้อมูลถูกส่งผ่าน

สู่ระดับสูงสุด

ส่วนประกอบ.

ส่วนประกอบย่อยได้รับ

ข้อมูลนี้เป็นอุปกรณ์ประกอบฉาก

เมื่อข้อมูลระดับบนสุด

การเปลี่ยนแปลงส่วนประกอบลูก

รับข้อมูลใหม่

บ้าน

เรียกดู

บัญชี

หมวด 2

หมวด 1

หมวดหมู่ย่อย 2

หมวดหมู่ย่อย 1

ข้อมูล

หมวดหมู่ย่อย 2 รายการ

รูปที่ 1.3 วิธีการทำงานของกระแสข้อมูลทางเดียว

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 13

13

ยอมรับจุดแข็งของ React Native

1.5.5 ประสบการณ์ของนักพัฒนา

ประสบการณ์ของนักพัฒนาคือชัยชนะครั้งสำคัญสำหรับ React Native ถ้าคุณเคย
พัฒนา
สำหรับเว็บ คุณทราบถึงเวลาการรีโหลดที่รวดเร็วของเบราว์เซอร์ การพัฒนาเว็บมี
ไม่มีขั้นตอนการรวบรวม: เพียงรีเฟรชหน้าจอและการเปลี่ยนแปลงของคุณก็อยู่ที่นั่น นี้
มันไกล
ร้องไห้จากการรวบรวมเวลาอันยาวนานของการพัฒนาพื้นเมือง สาเหตุหนึ่ง
Facebook
ตัดสินใจที่จะพัฒนา React Native คือการเอาชนะเวลาคอมไพล์ที่ยาวนานของ
แอปพลิเคชัน Facebook เมื่อใช้เครื่องมือสร้าง iOS และ Android ดั้งเดิม ที่จะ
ทำให้
การเปลี่ยนแปลง UI หรือการเปลี่ยนแปลงอื่น ๆ นักพัฒนา Facebook ต้องรอเป็น
เวลานานในขณะที่
โปรแกรมที่คอมไพล์เพื่อดูผลลัพธ์ เวลาคอมไพล์นานส่งผลให้โปร-
ความเหนียวและต้นทุนนักพัฒนาที่เพิ่มขึ้น React Native แก้ปัญหานี้โดยให้คุณ

เวลาโหลดซ้ำของเว็บอย่างรวดเร็วรวมถึงเครื่องมือแก้ไขข้อบกพร่องของ Chrome และ Safari ทำให้
ประสบการณ์การดีบักให้ความรู้สึกเหมือนเว็บมาก
React Native ยังมีสิ่งที่เรียกว่า hot reloading ในตัว สิ่งนี้หมายความว่าอย่างไร
ในขณะที่กำลังพัฒนาแอปพลิเคชัน ลองนึกภาพว่าต้องคลิกเข้าไปในแอปของคุณสักสองสามครั้ง
เพื่อไปยังที่ที่คุณกำลังพัฒนา ขณะใช้การโหลดซ้ำ เมื่อคุณสร้างรหัส
เปลี่ยนไม่ต้องโหลดใหม่แล้วกดย้อนกลับผ่านแอพเพื่อไปที่ปัจจุบัน
สถานะ. เมื่อใช้คุณสมบัตินี้ คุณจะบันทึกไฟล์ และแอปพลิเคชันจะโหลดซ้ำเฉพาะส่วนประกอบ
คุณได้เปลี่ยนแปลง ให้คำติชมและอัปเดตสถานะปัจจุบันของ . ทันที
UI

1.5.6 การขนถ่าย

Transpilation มักเกิดขึ้นเมื่อสิ่งที่เรียกว่า transpiler ใช้ซอร์สโค้ดเป็นลายลักษณ์อักษร
ในภาษาโปรแกรมหนึ่งและสร้างรหัสที่เทียบเท่าในภาษาอื่น
ด้วยคุณสมบัติและมาตรฐาน ECMAScript ใหม่ที่เพิ่มขึ้น การถ่ายทอดข้อมูลจึงล้นหลาม
รวมถึงการใช้เวอร์ชันที่ใหม่กว่าและคุณลักษณะที่ยังไม่ได้ใช้งานของบาง
ภาษา ในกรณีนี้คือ JavaScript และการสร้างจาวาสคริปต์มาตรฐานแบบทรานสพิล, mak-
โค้ดที่ใช้งานได้โดยแพลตฟอร์มที่สามารถประมวลผลภาษาเวอร์ชันเก่าเท่านั้น

React Native ใช้ Babel ในการทำขั้นตอน transpilation นี้ และสร้างขึ้น
โดยค่าเริ่มต้น Babel
เป็นเครื่องมือโอเพ่นซอร์สที่ถ่ายทอดคุณสมบัติภาษา JavaScript ที่ล้ำหน้าที่สุด
tures เป็นรหัสที่สามารถใช้ได้ในปัจจุบัน ไม่ต้องรอให้ข้าราชการ
คุณสมบัติทางภาษาที่ได้รับการเสนอ อนุมัติ และดำเนินการก่อนคุณ
สามารถใช้งานได้ คุณสามารถเริ่มใช้คุณสมบัติได้ทันทีที่ทำให้มันกลายเป็น Babel ซึ่ง
ก็คือ
มักจะเร็วมาก คลาส JavaScript ฟังก์ชันลูกศรและการทำลายวัตถุเป็น
ตัวอย่างทั้งหมดของคุณสมบัติ ES2015 อันทรงพลังที่ไม่ได้มีอยู่ในทุกเบราว์เซอร์และ
รันไทม์ยัง; แต่ด้วย Babel และ React Native คุณสามารถใช้มันวันนี้โดยไม่
ต้องกังวล
ว่าพวกเขาจะทำงานหรือไม่ หากคุณชอบใช้ฟีเจอร์ภาษาล่าสุด คุณสามารถใช้
กระบวนการถ่ายโอนเดียวกันเพื่อพัฒนาเว็บแอปพลิเคชัน

1.5.7 ผลผลิตและประสิทธิภาพ

การพัฒนาอุปกรณ์พกพาแบบเนทีฟนั้นมีราคาแพงขึ้นเรื่อย ๆ ดังนั้นวิศวกรที่
สามารถส่งมอบแอปพลิเคชันข้ามแพลตฟอร์มและสแต็คจะมีมูลค่าเพิ่มขึ้น
และอยู่ในความต้องการ เมื่อ React Native—หรือสิ่งที่คล้ายกัน ถ้ามันมาพร้อม—
ทำให้
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

การพัฒนาเดสก์ท็อปและเว็บรวมถึงแอปพลิเคชันมือถือโดยใช้เฟรมเวิร์กเดียว
กระแสหลักจะมีการปรับโครงสร้างและคิดใหม่ว่าทีมวิศวกรเป็นอย่างไร
เป็นระเบียบ. แทนที่จะเป็นนักพัฒนาที่เชี่ยวชาญในบางแพลตฟอร์ม เช่น iOS
หรือเว็บ พวกเขาจะดูแลคุณสมบัติข้ามแพลตฟอร์ม ในยุคใหม่ของข้ามแพลตฟอร์มและ
ทีมวิศวกรรมข้ามสแตก นักพัฒนาส่งมอบมือถือ เว็บ และเดสก์ท็อป
การใช้งานจะมีประสิทธิภาพและประสิทธิผลมากขึ้น ดังนั้นจึงสามารถเรียกร้องได้
ค่าแรงที่สูงกว่านักพัฒนาเว็บทั่วไปที่สามารถให้บริการเว็บแอปพลิเคชันเท่านั้น
บริษัทที่จ้างนักพัฒนาเพื่อพัฒนามือถือจะได้รับประโยชน์จาก
ส่วนใหญ่มาจากการใช้ React Native การมีทุกอย่างที่เขียนในภาษาเดียวทำให้การ
จ้าง a
ง่ายกว่ามากและราคาไม่แพง ผลผลิตยังเพิ่มสูงขึ้นเมื่อทุกทีมมีข้อมูลตรงกัน
ทำงานภายใต้เทคโนโลยีเดียว ซึ่งช่วยลดความยุ่งยากในการทำงานร่วมกันและการ
แบ่งปันความรู้

1.5.8 ชุมชน

ชุมชน React และโดยการขยายชุมชน React Native เป็นหนึ่งใน
กลุ่มที่เปิดกว้างและช่วยเหลือดีที่สุดที่ฉันเคยมีปฏิสัมพันธ์ด้วย เมื่อฉันพบปัญหาฉัน
ไม่สามารถแก้ไขได้ด้วยตัวเองโดยการค้นหาออนไลน์หรือบน Stack Overflow
ฉันเอื้อมมือออกไป
โดยตรงกับสมาชิกในทีมหรือคนในชุมชนและไม่มีอะไรเลย
แต่ข้อเสนอแนะในเชิงบวกและความช่วยเหลือ

1.5.9 โอเพ่นซอร์ส

React Native เป็นโอเพ่นซอร์ส สิ่งนี้มีประโยชน์มากมาย ประการแรก นอกเหนือจาก
ทีมงาน Facebook นักพัฒนาหลายร้อยคนมีส่วนร่วมใน React Native ข้อบกพร่องชี้
ออกเร็วกว่าซอฟต์แวร์ที่เป็นกรรมสิทธิ์ซึ่งมีเฉพาะพนักงานเท่านั้น
ทีมงาน cific กำลังดำเนินการแก้ไขข้อบกพร่องและปรับปรุง โอเพ่นซอร์สมักจะเข้า
ใกล้
สิ่งที่ผู้ใช้ต้องการเพราะว่าผู้ใช้สามารถมีส่วนทำให้ซอฟต์แวร์ในสิ่งที่พวกเขาต้องการได้
ต้องการให้มันเป็น พิจารณาค่าใช้จ่ายในการซื้อซอฟต์แวร์ที่เป็นกรรมสิทธิ์ ค่าธรรมเนียม
ใบอนุญาต และการสนับสนุน
ต้นทุนพอร์ตโอเพ่นซอร์สยังชนะเมื่อวัดราคา

1.5.10 อัปเดตทันที

ตามเนื้อผ้า เมื่อเผยแพร่แอปเวอร์ชันใหม่ คุณอยู่ในความเมตตาของแอป
ขั้นตอนการอนุมัติร้านและกำหนดการ กระบวนการที่ยาวนานและน่าเบื่อนี้อาจใช้เวลาถึง
สอง
สัปดาห์ การเปลี่ยนแปลงแม้จะเล็กน้อยมากก็เจ็บปวดและต้องปล่อย
เวอร์ชันใหม่ของแอปพลิเคชัน
React Native เช่นเดียวกับเฟรมเวิร์กแอปพลิเคชันแบบไฮบริด ช่วยให้คุณสามารถ
ปรับใช้ mobile
แอปจะอัปเดตไปยังอุปกรณ์ของผู้ใช้โดยตรง โดยไม่ต้องผ่านการอนุมัติจากร้านแอป
กระบวนการ. หากคุณคุ้นเคยกับเว็บและวงจรการเผยแพร่อย่างรวดเร็ว คุณสามารถทำได้
ในตอนนี้
สิ่งเดียวกันกับ React Native และเฟรมเวิร์กแอปพลิเคชันไฮบริดอื่น ๆ

1.5.11 โซลูชันอื่นๆ สำหรับการสร้างแอปพลิเคชันมือถือข้ามแพลตฟอร์ม

React Native ไม่ใช่ตัวเลือกเดียวสำหรับการสร้างแอปพลิเคชันมือถือข้ามแพลตฟอร์ม หลาย-
มีตัวเลือกอื่น ๆ มากมายโดยมีตัวเลือกหลักคือ Cordova, Xamarin และ Flutter:

- Cordova นั้นเป็นเชลล์ดั้งเดิมรอบ ๆ เว็บแอปพลิเคชันที่อนุญาตให้
นักพัฒนาเพื่อเข้าถึง API ดั้งเดิมภายในแอปพลิเคชัน ไม่เหมือนเว็บทั่วไป

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

15
การสร้างและการใช้ส่วนประกอบพื้นฐาน
แอพพลิเคชั่น, แอพ Cordova สามารถนำไปใช้กับ App Store และ Google Play
เก็บ. ข้อดีของการใช้บางอย่างเช่น Cordova คือไม่มีอะไรมากไปกว่านี้อีกแล้ว
เพื่อเรียนรู้ว่าคุณเป็นนักพัฒนาเว็บอยู่แล้วหรือไม่: คุณสามารถใช้ HTML, JavaScript, CSS,
และกรอบงาน JavaScript ที่คุณเลือก ข้อเสียเปรียบหลักของคอร์โดวาคือ
คุณจะมีช่วงเวลาที่ยากลำบากในการจับคู่ประสิทธิภาพและ UI ที่ราบรื่นที่ React
ข้อเสนอดั้งเดิม: คุณพึ่งพา DOM เพราะคุณทำงานด้วยเป็นหลัก
เทคโนโลยีเว็บ

▪ Xamarin เป็นเฟรมเวิร์กที่ช่วยให้นักพัฒนาสามารถสร้าง iOS, Android, Windows,
และแอปพลิเคชัน macOS โดยใช้ codebase เดียวที่เขียนด้วย C# Xamarin คอม-
รวมเข้ากับแอพเนทีฟในรูปแบบต่างๆ ขึ้นอยู่กับแพลตฟอร์มที่กำหนดเป้าหมาย
Xamarin มีระดับฟรีที่ช่วยให้นักพัฒนาสร้างและปรับใช้แอปพลิเคชันมือถือ
และระดับการชำระเงินสำหรับบริษัทขนาดใหญ่หรือองค์กร Xamarin อาจจะอุทธรณ์
สำหรับนักพัฒนาที่เป็นเจ้าของภาษามากกว่าเพราะไม่มีความคล้ายคลึงกับเทคโนโลยีเว็บ
เช่น React Native และ Cordova
▪ Flutter เป็นเฟรมเวิร์กโอเพ่นซอร์สโดย Google ที่ใช้โปรแกรม Dart
ภาษาเพื่อสร้างแอปพลิเคชันที่ทำงานบนแพลตฟอร์ม iOS และ Android

1.6 ตอบสนองข้อเสียของ Native

ตอนนี้เราได้พูดถึงประโยชน์ของการใช้ React Native แล้ว เรามาดูเหตุผลกันสอง
สามข้อ
ลูกและสถานการณ์ที่คุณอาจไม่ต้องการเลือกกรอบ ขั้นแรกให้ตอบโต้
Native นั้นยังไม่บรรลุนิติภาวะเมื่อเทียบกับแพลตฟอร์มอื่นๆ เช่น Native iOS,
Android,
และคอร์โดวา ความเท่าเทียมกันของฟีเจอร์ยังไม่มีใน iOS หรือ Cordova
ดั้งเดิม ที่สุด
ขณะนี้มีฟังก์ชันในตัวแล้ว แต่อาจมีบางครั้งที่คุณต้องการฟังก์ชันที่
ยังไม่พร้อมใช้งาน และนี่หมายความว่าคุณต้องขุดลงไปในโค้ดเนทีฟเพื่อสร้างมันขึ้นมา
เอง
จ้างคนให้ทำหรือไม่ใช้คุณลักษณะนี้
อีกสิ่งหนึ่งที่ควรคำนึงถึงคือข้อเท็จจริงที่คุณและ/หรือทีมของคุณต้องเรียนรู้เกี่ยวกับคอม-

เทคโนโลยีใหม่มากมายหากคุณไม่คุ้นเคยกับ React คนส่วนใหญ่เห็นด้วยว่า React หยิบง่าย แต่ถ้าคุณเชี่ยวชาญด้าน Angular และ Ionic อยู่แล้ว เช่น
และคุณมีกำหนดส่งใบสมัครใกล้เข้ามา ก็ควรที่จะไปกับสิ่งที่คุณ
รู้อยู่แล้วแทนที่จะใช้เวลาในการเรียนรู้และฝึกอบรมทีมของคุณในรูปแบบใหม่
เทคโนโลยี นอกจากการเรียนรู้ React และ React Native แล้ว คุณต้องทำ
ความคุ้นเคยด้วย
ด้วย Xcode และสภาพแวดล้อมการพัฒนา Android ซึ่งอาจต้องใช้เวลาบ้าง
เคย.
สุดท้าย React Native เป็นนามธรรมที่สร้างขึ้นบน API ของแพลตฟอร์มที่มี
อยู่ เมื่อไหร่
iOS, Android เวอร์ชันใหม่กว่า และแพลตฟอร์มอื่นๆ ในอนาคตที่ออกวางจำหน่าย
อาจมี
เวลาที่ React Native จะล้าหลังในฟีเจอร์ใหม่ บังคับให้คุณสร้าง cus-
การใช้งาน tom เพื่อโต้ตอบกับ API ใหม่เหล่านี้หรือรอจนกว่า React Native
จะกลับมา
ความเท่าเทียมกันของคุณลักษณะกับรุ่นใหม่

1.7 การสร้างและการใช้ส่วนประกอบพื้นฐาน

ส่วนประกอบเป็นส่วนประกอบพื้นฐานของ React Native และสามารถ
เปลี่ยนแปลงได้ใน
ฟังก์ชั่นและประเภท ตัวอย่างของส่วนประกอบในกรณีใช้งานยอดนิยม ได้แก่ ปุ่ม
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

ส่วนหัว ส่วนท้าย และส่วนประกอบการนำทาง พวกเขาสามารถแตกต่างกันไปในประเภทจากทั้งหมด
ดูสมบูรณ์ด้วยสถานะและการทำงานเป็นองค์ประกอบเดียวที่
รับอุปกรณ์ประกอบฉากทั้งหมดจากผู้ปกครอง

1.7.1 ภาพรวมของส่วนประกอบ

อย่างที่ฉันได้กล่าวไปแล้ว แก่นของ React Native คือแนวคิดของส่วนประกอบ ส่วนประกอบคือ
การรวบรวมข้อมูลและองค์ประกอบ UI ที่ประกอบเป็นมุมมองและท้ายที่สุดคือแอปพลิเคชัน
React Native มีส่วนประกอบในตัวที่อธิบายว่าเป็นส่วนประกอบดั้งเดิมในนี้
หนังสือ แต่คุณสามารถสร้างส่วนประกอบที่กำหนดเองได้โดยใช้กรอบงาน เราจะเข้าสู่
เจาะลึกถึงวิธีการสร้าง สร้าง และใช้ส่วนประกอบ
ดังที่ได้กล่าวไว้ก่อนหน้านี้ส่วนประกอบ React Native นั้นสร้างโดยใช้ JSX ตาราง 1.1 แสดง
ตัวอย่างพื้นฐานบางประการของลักษณะ JSX ใน React Native เมื่อเทียบกับ HTML เท่าที่ทำได้
ดู JSX ดูเหมือน HTML หรือ XML

ตารางที่ 1.1 องค์ประกอบ JSX กับองค์ประกอบ HTML

ข้อความ
สวัสดีชาวโลก
<Text>สวัสดีชาวโลก</Text>
ดู

สวัสดีชาวโลก 2

<ดู>
<Text>สวัสดีชาวโลก 2</Text>
</ดู>
ไฮไลท์ที่สัมผัสได้ <ปุ่ม>
สวัสดีชาวโลก 2
</button >
<TouchableHighlight>
<Text>สวัสดีชาวโลก 2</Text>
</TouchableHighlight>

1.7.2 ส่วนประกอบดั้งเดิม

เฟรมเวิร์กนำเสนอส่วนประกอบดั้งเดิมที่พร้อมใช้งานทันที เช่นView , Textและรูปภาพท่ามกลางคนอื่น ๆ คุณสามารถสร้างส่วนประกอบโดยใช้ส่วนประกอบดั้งเดิมเหล่านี้เป็น
การก่อสร้างตึก. ตัวอย่างเช่น คุณสามารถใช้มาร์กอัปต่อไปนี้เพื่อสร้างปุ่ม
องค์ประกอบที่ใช้ตอบสนองพื้นเมืองTouchableHighlightและข้อความส่วนประกอบ

นำเข้า { ข้อความ TouchableHighlight } จาก 'react-native'
ปุ่ม const = () => (
<TouchableHighlight>
<Text>สวัสดีชาวโลก</Text>

</TouchableHighlight>
)
ส่งออกปุ่มเริ่มต้น
จากนั้นคุณสามารถนำเข้าและใช้ปุ่มใหม่ได้
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 17

17
การสร้างและการใช้ส่วนประกอบพื้นฐาน

นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ข้อความ, ดู } จาก 'react-native'
ปุ่มนำเข้าจาก './components/Button'
const หน้าแรก = () => (
<ดู>
<Text>ยินดีต้อนรับสู่ปุ่ม Hello World!</Text>
<ปุ่ม />
</ดู>
)
ต่อไป เราจะพูดถึงพื้นฐานของว่าส่วนประกอบคืออะไร ส่วนประกอบนั้นพอดีอย่างไร ลงในเวิร์กโฟลว์และกรณีการใช้งานทั่วไปและรูปแบบการออกแบบสำหรับการสร้าง

1.7.3 องค์ประกอบส่วนประกอบ

ส่วนประกอบมักจะประกอบขึ้นโดยใช้ JSX แต่ก็สามารถประกอบได้โดยใช้จาวาสคริปต์ ในส่วนนี้ คุณจะสร้างส่วนประกอบหลายวิธีในการดูทั้งหมด

ตัวเลือก คุณจะสร้างองค์ประกอบนี้:
<MyComponent />
ส่วนประกอบนี้จะส่งออก "Hello World" ไปที่หน้าจอ ตอนนี้เรามาดูวิธีการสร้างสิ่งนี้กัน
องค์ประกอบพื้นฐาน ส่วนประกอบสำเร็จรูปเพียงชิ้นเดียวที่คุณจะใช้สร้างแบบกำหนดเองนี้
องค์ประกอบคือองค์ประกอบมุมมองและข้อความที่กล่าวถึงก่อนหน้านี้ โปรดจำไว้ว่าดูส่วนประกอบที่
ตรวจวัดค่ามีความคล้ายคลึงกับเว็บ HTML และข้อความองค์ประกอบคล้ายกับ HTML
ลองดูวิธีสร้างส่วนประกอบสองสามวิธี แอปพลิเคชันทั้งหมดไม่จำเป็นต้องมีความสอดคล้องในคำจำกัดความขององค์ประกอบ แต่โดยปกติแนะนำให้คุณอยู่สอดคล้องและปฏิบัติตามรูปแบบเดียวกันสำหรับการกำหนดคลาสตลอดแอปพลิเคชันของคุณ

สร้างC LASS SYNTAX (ES5, JSX)

นี่คือวิธีสร้างส่วนประกอบ React Native โดยใช้ไวยากรณ์ ES5 คุณคงจะยังคงเห็นการใช้ไวยากรณ์นี้ในเอกสารและตัวอย่างที่เก่ากว่าบางส่วน แต่มันไม่ใช่ใช้ในเอกสารประกอบที่ใหม่กว่าและเลิกใช้แล้วในขณะนี้ เราจะเน้นที่คลาส ES2015 ไวยากรณ์สำหรับส่วนที่เหลือของหนังสือ แต่จะตรวจสอบไวยากรณ์createClassที่นี่ในกรณีที่คุณ
เจอมันในรหัสที่เก่ากว่า:

```
const React = ต้องการ ('react')
const ReactNative = ต้องการ ('react-native')
const { ดู, ข้อความ } = ReactNative
const MyComponent = React.createClass ({
```

เรนเดอร์ () {
กลับ (
<ดู>
<Text>สวัสดีชาวโลก</Text>
</ดู>)
}
})
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 18

18
C HAPTER 1 เริ่มต้นใช้งาน React Native
ไวยากรณ์ C LASS (ES2015, JSX)
วิธีหลักในการสร้างองค์ประกอบ React Native แบบเก็บสถานะคือการใช้คลาส ES2015 นี้
เป็นวิธีสร้างองค์ประกอบ stateful สำหรับส่วนที่เหลือของหนังสือและตอนนี้คือแนวทางที่แนะนำโดยชุมชนและผู้สร้าง React Native:
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ } จาก 'react-native'
คลาส MyComponent ขยาย React.Component {
เรนเดอร์ () {
กลับ (
<ดู>
<Text>สวัสดีชาวโลก</Text>
</ดู>)

```
}
}
```

S TATELESS (นำกลับมาใช้ใหม่) ส่วนประกอบ (JSX)

นับตั้งแต่เปิดตัว React 0.14 เราก็สามารถสร้างส่วนประกอบที่ไม่มีสถานะได้ เรายังไม่ได้ดำดิ่งสู่สถานะ แต่จำไว้ว่าส่วนประกอบไร้สัญชาตินั้นโดยพื้นฐานแล้วฟังก์ชันบริสุทธิ์ที่ไม่สามารถเปลี่ยนแปลงข้อมูลของตนเองและไม่มีสถานะของตนเองได้ นี้

ไวยากรณ์นั้นสะอาดกว่าคลาสหรือไวยากรณ์createClass มาก :

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ } จาก 'react-native'
const MyComponent = () => (
<ดู>
<Text>สวัสดีชาวโลก</Text>
</ดู>
)
หรือ
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ } จาก 'react-native'
ฟังก์ชัน MyComponent () {
ส่งคืน <View><Text>HELLO จาก STATELESS</Text></View>
}
```

สร้างE LEMENT (J AVA S CRIPT)

React.createElement ไม่ค่อยได้ใช้ และคุณอาจไม่จำเป็นต้องสร้าง React องค์ประกอบดั้งเดิมที่ใช้ไวยากรณ์นี้ แต่อาจมีประโยชน์หากคุณต้องการลองคิดดูว่าคุณกำลังสร้างส่วนประกอบอย่างไร หรือถ้าคุณกำลังอ่านโค้ดของคนอื่น นอกจากนี้ยังจะช่วยให้คุณดูว่า JavaScript รวบรวม JSX อย่างไร React.createElementใช้เวลา

อาร์กิวเมนต์เล็กน้อย:

React.createElement(ประเภท อุปกรณ์ประกอบฉาก ลูก) {}

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 19

19

การสร้างและการใช้ส่วนประกอบพื้นฐาน

ลองเดินผ่านพวกเขา:

- ประเภท——องค์ประกอบที่คุณต้องการแสดง
- อุปกรณ์ประกอบฉาก — คุณสมบัติใด ๆ ที่คุณต้องการให้ส่วนประกอบมี
- ลูก—— ส่วนประกอบลูกหรือข้อความ

ในตัวอย่างต่อไปนี้ คุณส่งผ่านมุมมองเป็นอาร์กิวเมนต์แรกไปยังอินสแตนซ์แรกของ React.createElement วัตถุว่างเป็นอาร์กิวเมนต์ที่สอง และองค์ประกอบอื่นเป็นอาร์กิวเมนต์สุดท้าย ในตัวอย่างที่สอง คุณส่งผ่านข้อความเป็นอาร์กิวเมนต์แรก an วัตถุว่างเป็นอาร์กิวเมนต์ที่สอง และ "สวัสดี" เป็นอาร์กิวเมนต์สุดท้าย:

```
คลาส MyComponent ขยาย React.Component {
เรนเดอร์ () {
กลับ (
React.createElement (ดู, {},
React.createElement (ข้อความ {} "สวัสดี")
)
)
}
}
```

นี้เหมือนกับการประกาศส่วนประกอบดังต่อไปนี้:

```
คลาส MyComponent ขยาย React.Component {
แสดงผล () {
กลับ (
<ดู>
<Text>สวัสดี</Text>
</ดู>
)
}
}
```

1.7.4 ส่วนประกอบที่ส่งออกได้

ต่อไป มาดูการใช้งาน React Native compo-
เน็ท คุณจะต้องสร้างส่วนประกอบทั้งหมดที่คุณสามารถส่งออกและใช้ในไฟล์อื่นได้:

```
นำเข้า React { ส่วนประกอบ } จาก 'react'
นำเข้า {
ข้อความ,
ดู
} จาก 'react-native'
คลาสโฮมขยายคอมโพเนนต์ {
เรนเดอร์ () {
กลับ (
<ดู>
<Text>สวัสดีจากหน้าแรก</Text>
</ดู>)
}
}
ส่งออกค่าเริ่มต้น Home
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc

หน้า 20

มาดูทุกส่วนที่ประกอบขึ้นเป็นองค์ประกอบนี้และอภิปรายว่าเกิดอะไรขึ้น

ฉันนำเข้า

รหัสต่อไปนี้นำเข้า React Native การประกาศตัวแปร:

```
นำเข้า React { ส่วนประกอบ } จาก 'react'
นำเข้า {
ข้อความ,
ดู
} จาก 'react-native'
```

ที่นี่คุณกำลังนำเข้า React โดยตรงจากไลบรารี React โดยใช้การนำเข้าเริ่มต้น และการนำเข้าส่วนประกอบจากไลบรารี React โดยใช้การนำเข้าที่มีชื่อ คุณยังใช้การนำเข้าที่มีชื่อเพื่อดึงข้อความและมุมมองลงในไฟล์ของคุณ

นำเข้าคำสั่งโดยใช้ ES5 จะมีลักษณะเช่นนี้

```
var React = ต้องการ ('react')
```

คำสั่งนี้โดยไม่ใช้การนำเข้าที่มีชื่อจะมีลักษณะดังนี้:

```
นำเข้า React = จาก 'react'
ส่วนประกอบ const = React.Component
นำเข้า ReactNative จาก 'react-native'
const Text = ReactNative.Text
const View = ReactNative.View
```

นำเข้าคำสั่งที่ใช้ในการฟังก์ชั่นนำเข้าวัตถุหรือตัวแปรที่ได้รับส่งออกจากโมดูล ไฟล์ หรือสคริปต์อื่น

ปฏิญญา C OMPONENT

รหัสต่อไปนี้ประกาศส่วนประกอบ:

class Home ขยายคอมโพเนนต์ { }

ที่นี่คุณกำลังสร้างอินสแตนซ์ใหม่ของคลาสReact Native Componentโดยขยายมัน

และตั้งชื่อมันว่าบ้าน ก่อนหน้านี้ คุณได้ประกาศReact.Component ; ตอนนี้คุณเพิ่งประกาศ-

ing Componentเนื่องจากคุณนำเข้าองค์ประกอบComponentในการทำลายวัตถุ-

คำสั่ง turing ให้คุณเข้าถึงComponentแทนที่จะต้องเรียกReact

ตัวแทน

T HE วิธีการ Render

ถัดไปดูที่ทำให้วิธีการ:

เรนเดอร์ () {

กลับ (

<ดู>

<Text>สวัสดีจากหน้าแรก</Text>

</ดู>)

}

สำหรับองค์ประกอบที่จะดำเนินการในการแสดงผลวิธีการและเนื้อหา

หลังจากที่ผลตอบแทนผลตอบแทนงบสิ่งที่แสดงผลบนหน้าจอ เมื่อการแสดงผลเรียกเมธอด มันควรส่งคืนองค์ประกอบลูกเดียว ตัวแปรหรือฟังก์ชันใดๆ นอกประกาศของทำให้ฟังก์ชั่นที่สามารถดำเนินการที่นี่ หากจำเป็นต้องดำเนินการใดๆ การคำนวณ ประกาศตัวแปรใดๆ โดยใช้สถานะหรืออุปกรณ์ประกอบฉาก หรือเรียกใช้ฟังก์ชันใดๆ ที่ไม่

หน้า 21

21

การสร้างและการใช้ส่วนประกอบพื้นฐาน

จัดการสถานะของส่วนประกอบ คุณสามารถทำได้ระหว่างวิธีการแสดงผล

และใบแจ้งกลับ

E XPORTS

ตอนนี้ คุณส่งออกส่วนประกอบเพื่อใช้ที่อื่นในแอปพลิเคชัน:

ส่งออกค่าเริ่มต้น Home

หากคุณต้องการใช้ส่วนประกอบในไฟล์เดียวกัน คุณไม่จำเป็นต้องส่งออกส่วนประกอบนั้น หลังจากที่มัน

ประกาศ คุณสามารถใช้มันในไฟล์หรือส่งออกเพื่อใช้ในไฟล์อื่น นอกจากนี้คุณยังสามารถ

ใช้module.exports = 'Home'ซึ่งเป็นไวยากรณ์ ES5

1.7.5 การรวมส่วนประกอบ

มาดูวิธีการรวมส่วนประกอบกัน ขั้นแรก ให้สร้างHome , Headerและ Footer com-

ponents ในไฟล์เดียว เริ่มต้นด้วยการสร้างองค์ประกอบหน้าแรก :

นำเข้า React { ส่วนประกอบ } จาก 'react'

นำเข้า {

ข้อความ,

ดู

} จาก 'react-native'
คลาสโฮมขยายคอมโพเนนต์ {
เรนเดอร์ () {
กลับ (
<ดู>
</ดู>)
}
}
ในไฟล์เดียวกัน ด้านล่างการประกาศโฮมคลาส ให้สร้างองค์ประกอบส่วนหัว :
ส่วนหัวของคลาสขยายส่วนประกอบ {
เรนเดอร์ () {
กลับ <ดู>
<Text>HEADER</Text>
</ดู>
}
}
ดูดี แต่มาดูวิธีการเขียนHeaderใหม่ให้เป็นส่วนประกอบที่ไม่เก็บสถานะ ดี
สนทนาว่าเมื่อใดและเหตุใดจึงควรใช้องค์ประกอบไร้สัญชาติกับ React . ปกติ
ชั้นเรียนพื้นเมืองในเชิงลึกในภายหลังในหนังสือ อย่างที่คุณจะเริ่มเห็น ไวยากรณ์และ
โค้ดคือ
สะอาดกว่ามากเมื่อคุณใช้ส่วนประกอบไร้สัญชาติ:
const ส่วนหัว = () => (
<ดู>
<Text>HEADER</Text>
</ดู>
)
ตอนนี้ แทรกส่วนหัวลงในองค์ประกอบหน้าแรก :
คลาสโฮมขยายคอมโพเนนต์ {
เรนเดอร์ () {

หน้า 22


```
กลับ (
<ดู>
<Header />
</ดู>
)
}
}
```

สร้างส่วนท้ายและมุมมองหลักด้วย:

```
const ส่วนท้าย = () => (
<ดู>
<Text>ส่วนท้าย</Text>
</ดู>
)
const หลัก = () => (
<ดู>
<ข้อความ> หลัก </Text>
</ดู>
)
```

ตอนนี้ วางสิ่งเหล่านั้นลงในใบสมัครของคุณ:

```
คลาสโฮมขยายคอมโพเนนต์ {
เรนเดอร์ () {
```

```
กลับ (
<ดู>
<Header />
<หน้าหลัก />
<ส่วนท้าย />
</ดู>
)
}
}
```

รหัสที่คุณเพิ่งเขียนมีความชัดเจนมาก หมายความว่ามันเขียนในลักษณะที่มันอธิบายสิ่งที่คุณต้องการทำและแยกส่วนได้ง่าย นี่คือสูง-ภาพรวมระดับของวิธีสร้างส่วนประกอบและมุมมองใน React Native แต่ควรให้แนวคิดที่ดีเกี่ยวกับวิธีการทำงานพื้นฐาน

1.8 การสร้างโครงการเริ่มต้น

เมื่อเราได้ดูรายละเอียดมากมายเกี่ยวกับ React Native แล้ว มาเจาะลึกกันมากขึ้นรหัส. เราจะเน้นที่การสร้างแอพโดยใช้ React Native CLI แต่คุณยังสามารถใช้สร้าง React Native App CLI เพื่อสร้างโครงการใหม่

1.8.1 สร้าง React Native App CLI

คุณสามารถสร้างโปรเจ็กต์ React Native โดยใช้ Create React Native App CLI ซึ่งเป็นโปรเจ็กต์ตัวสร้างที่ดูแลในที่เก็บ React Community GitHub ส่วนใหญ่โดยทีมงานเอ็กซ์โป Expo ได้สร้างโปรเจ็กต์ React Native App เพื่อให้นักพัฒนาสามารถเริ่มต้นใช้งาน React Native โดยไม่ต้องกังวลเกี่ยวกับการติดตั้ง Native SDK ที่เกี่ยวข้องกับการรันโปรเจ็กต์ React Native โดยใช้ CLI

หน้า 23

การสร้างโครงการเริ่มต้น

ในการสร้างโปรเจ็กต์ใหม่โดยใช้ Create React Native App ให้ติดตั้ง CLI ก่อน:

```
npm install -g create-react-native-app
```

ต่อไปนี้เป็นวิธีสร้างโครงการใหม่โดยใช้create-react-native-appจาก com-บรรทัดคำสั่ง:

```
create-react-native-app myProject
```

1.8.2 ตอบสนอง Native CLI

ก่อนที่เราจะดำเนินการต่อ ให้ตรวจสอบภาคผนวกของหนังสือเล่มนี้เพื่อยืนยันว่าคุณมีสิ่งที่จำเป็น-

เครื่องมือ sary ติดตั้งบนเครื่องของคุณ หากคุณไม่ได้ติดตั้ง SDK ที่จำเป็น คุณจะไม่สามารถสร้างโครงการแรกของคุณต่อไปโดยใช้ React Native CLI

ในการเริ่มต้นโครงการเริ่มต้น React Native และ React Native CLI ให้เปิด

บรรทัดคำสั่ง จากนั้นสร้างและนำทางไปยังไดเร็กทอรีว่าง เมื่อคุณ

ที่นั่น ติดตั้ง react-native CLI ทั่วโลกโดยพิมพ์ดังต่อไปนี้:

```
npm install -g react-native-cli
```

หลังจากติดตั้ง React Native บนเครื่องของคุณแล้ว คุณสามารถเริ่มต้นโปรเจ็กต์ใหม่ได้โดยพิมพ์-
ไอเอ็นจีตอบสนองพื้นเมือง initตามด้วยชื่อโครงการ:
react-native init myProject
myProject สามารถเป็นชื่อใดก็ได้ที่คุณเลือก CLI จะสร้างโครงการใหม่ในไม่ว่าคุณจะอยู่ในไดเรกทอรีใด เปิดโครงการในโปรแกรมแก้ไขข้อความ
อันดับแรก มาดูไฟล์และโฟลเดอร์หลักที่กระบวนการนี้สร้างขึ้นสำหรับคุณ:

- android — โฟลเดอร์นี้มีรหัสเฉพาะแพลตฟอร์ม Android ทั้งหมดและขึ้นอยู่กับ

เดนซี่ คุณไม่จำเป็นต้องเข้าไปในโฟลเดอร์นี้ เว้นแต่ว่าคุณกำลังใช้งาน
tom bridge เข้าสู่ Android หรือคุณติดตั้งปลั๊กอินที่เรียกหา deep
การกำหนดค่า

- ios — โฟลเดอร์นี้มีโค้ดและการอ้างอิงเฉพาะแพลตฟอร์ม iOS ทั้งหมด

คุณไม่จำเป็นต้องเข้าไปในโฟลเดอร์นี้ เว้นแต่ว่าคุณกำลังใช้บริดจ์แบบกำหนดเอง
ใน iOS หรือคุณติดตั้งปลั๊กอินที่เรียกการกำหนดค่าแบบลึกบางประเภท

- node_modules —React Native ใช้ npm (ตัวจัดการแพ็คเกจโหนด)

เพื่อจัดการ
การพึ่งพา การพึ่งพาเหล่านี้ได้รับการระบุและกำหนดเวอร์ชันใน .package
json และเก็บไว้ในโฟลเดอร์ node_modules เมื่อคุณติดตั้งชุดใหม่ใด ๆ-
จากระบบนิเวศน์ npm/node พวกเขาจะไปที่นี่ สามารถติดตั้งได้โดยใช้
npm หรือเส้นด้าย

- . owcon g —Flow (เปิดแหล่งที่มาโดย Facebook ด้วย) เสนอการ

ตรวจสอบประเภทสำหรับ
จาวาสคริปต์ Flow ก็เหมือน typescript ถ้าคุณคุ้นเคยกับมัน ไฟล์นี้เป็นคอน

สมมติโฟลว์หากคุณเลือกใช้

- .gitignore — นี่คือที่สำหรับเก็บเส้นทางของไฟล์ที่คุณไม่ต้องการในเวอร์ชันควบคุม.

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 24

- .watchmancon g —Watchman คือโปรแกรมดูไฟล์ที่ React Native ใช้ในการรับชม

ไฟล์และบันทึกเมื่อมีการเปลี่ยนแปลง นี่คือการกำหนดค่าสำหรับ Watchman เลขที่จำเป็นต้องเปลี่ยนแปลงสิ่งนี้ ยกเว้นในกรณีการใช้งานที่หายาก

- index.js —นี่คือจุดเริ่มต้นของแอปพลิเคชัน ในไฟล์นี้ App.js จะถูกนำเข้าและเรียกAppRegistry.registerComponent ซึ่งเป็นการเริ่มต้นแอป
- App.js —นี่คือการนำเข้าหลักเริ่มต้นที่ใช้ใน index.js ที่มีฐานโครงการ. คุณสามารถเปลี่ยนได้โดยการลบไฟล์นี้และแทนที่การนำเข้าหลักในดัชนี.js
- package.json — ไฟล์นี้มีการกำหนดค่า npm ของคุณ เมื่อคุณ npm ติดตั้งไฟล์ คุณสามารถบันทึกไว้ที่นี่เป็นการขึ้นต่อกัน คุณยังสามารถตั้งค่าสคริปต์เพื่อเรียกใช้งานที่แตกต่างกัน

รายการต่อไปนี้แสดง App.js

```
/**
* ตัวอย่าง React Native App
* https://github.com/facebook/react-native
* @ไหล
*/
นำเข้า React { ส่วนประกอบ } จาก 'react';
นำเข้า {
แพลตฟอร์ม,
สไตล์ชีต,
ข้อความ,
ดู
} จาก 'react-native';
คำแนะนำ const = Platform.select ({
ios: 'กด Cmd+R เพื่อรีโหลด\n' +
'Cmd+D หรือเขย่าสำหรับเมนู dev',
android: 'แตะสองครั้งที่ R บนแป้นพิมพ์เพื่อโหลดซ้ำ\n' +
'เขย่าหรือกดปุ่มเมนูสำหรับเมนู dev',
});
ส่งออกแอปคลาสเริ่มต้นขยายคอมโพเนนต์<{}> {
เรนเดอร์ () {
กลับ (
<ดู style={styles.container}>
<รูปแบบข้อความ={styles.welcome}>
ยินดีต้อนรับสู่ React Native!
</Text>
<รูปแบบข้อความ={styles.instructions}>
ในการเริ่มต้น แก้ไข App.js
</Text>
<รูปแบบข้อความ={styles.instructions}>
{คำแนะนำ}
</Text>
```

หน้า 25

25
การสร้างโครงการเริ่มต้น
</ดู>
);
}
}
รูปแบบ const = StyleSheet.create ({
คอนเทนเนอร์: {
ดิ้น: 1,
justifyContent: 'ศูนย์',
alignItems: 'ศูนย์',
พื้นหลังสี: '#F5FCFF',
},
ยินดีต้อนรับ: {
ขนาดตัวอักษร: 20,
textAlign: 'ศูนย์',
ระยะขอบ: 10,
},
คำแนะนำ: {
textAlign: 'ศูนย์',
สี: '#333333',
ระยะขอบด้านล่าง: 5,
},
});

รหัสนี้ดูเหมือนกับสิ่งที่เราทำในส่วนที่แล้ว มีสองสาม
รายการใหม่ที่คุณยังไม่เคยเห็น:
สไตล์ชีต
แพลตฟอร์ม
แพลตฟอร์มคือ API ที่ให้คุณตรวจจับระบบปฏิบัติการประเภทปัจจุบันที่คุณอยู่ได้
ทำงานบน: เว็บ, iOS หรือ Android
StyleSheet เป็นนามธรรมเช่นสไตล์ชีต CSS ใน React Native คุณสามารถ
ประกาศ
สไตล์แบบอินไลน์หรือใช้สไตล์ชีต อย่างที่คุณเห็นในมุมมองแรก คอนเทนเนอร์ style
ถูกประกาศ:
<ดู style={styles.container}>
ซึ่งตรงกับ
คอนเทนเนอร์: {
ดิ้น: 1,
justifyContent: 'ศูนย์',
alignItems: 'ศูนย์',
พื้นหลังสี: '#F5FCFF',
}
ที่ด้านล่างของไฟล์ index.js คุณจะเห็น
AppRegistry.registerComponent('myProject', () => แอพ);
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

C HAPTER 1 เริ่มต้นใช้งาน React Native
นี่คือจุดเริ่มต้น JavaScript เพื่อเรียกใช้-
เรียกใช้แอป React Native ทั้งหมด ในไฟล์ดัชนี
เป็นที่เดียวที่คุณจะเรียกใช้ฟังก์ชันนี้
องค์ประกอบรูทของแอปควร
ลงทะเบียนด้วยAppRegistry.regis-
terComponent ระบบดั้งเดิมสามารถแล้ว
โหลดบันเดิลสำหรับแอพและเรียกใช้
แอพเมื่อพร้อม
ตอนนี้เราได้ไปมากกว่าสิ่งที่อยู่ใน
ไฟล์เรียกใช้โครงการใน iOS ซิมของคุณ
ulator หรือโปรแกรมจำลอง Android ของคุณ (ดูรูปที่
ใช่ 1.4) ในองค์ประกอบข้อความที่ประกอบด้วย
“ยินดีต้อนรับสู่ React Native” ป้อน “Wel-
มาที่ Hello World!” หรือข้อความอื่นๆ ของคุณ
ทางเลือก. รีเฟรชหน้าจอและคุณควร
ดูการเปลี่ยนแปลงของคุณ

สรุป

- React Native เป็นเฟรมเวิร์กสำหรับ build-
ing แอพมือถือดั้งเดิมใน JavaScript
โดยใช้ไลบรารี React JavaScript
- จุดแข็งบางส่วนของ React Native คือ
ประสิทธิภาพ ประสบการณ์ของนักพัฒนา

ence ความสามารถในการสร้างข้ามแพลตฟอร์ม
ด้วยภาษาเดียวทางเดียว
การไหลของข้อมูลและชุมชน คุณอาจ
พิจารณา React Native มากกว่าไฮบริด
ส่วนใหญ่เนื่องจากประสิทธิภาพ
และมากกว่า Native ส่วนใหญ่เป็นเพราะประสบการณ์ของนักพัฒนาและข้าม
แพลตฟอร์ม
ความสามารถด้วยภาษาเดียว

- JSX เป็นขั้นตอนของตัวประมวลผลล่วงหน้าที่เพิ่มไวยากรณ์ที่เหมือน XML
ให้กับ JavaScript คุณสามารถใช้ได้
JSX เพื่อสร้าง UI ใน React Native
- ส่วนประกอบเป็นส่วนประกอบพื้นฐานใน React Native พวกมันสามารถ
เปลี่ยนแปลงได้
ในการทำงานและประเภท คุณสามารถสร้างส่วนประกอบที่กำหนดเองเพื่อใช้คอม-
องค์ประกอบการออกแบบมอญ
- ส่วนประกอบที่ต้องใช้วิธีการของรัฐหรือวงจรชีวิตจะต้องสร้างโดยใช้ a
คลาส JavaScript โดยขยายคลาสReact.Component
- ส่วนประกอบไร้สัญชาติสามารถสร้างขึ้นได้โดยใช้ต้นแบบน้อยกว่าสำหรับ
ส่วนประกอบที่
ไม่จำเป็นต้องให้ทันกับสภาพของตัวเอง
- ส่วนประกอบขนาดใหญ่สามารถสร้างขึ้นได้โดยการรวมส่วนประกอบย่อยที่มีขนาด
เล็กลง

รูปที่ 1.4 React Native starter project: what
คุณควรเห็นหลังจากรันโครงการเริ่มต้น

ทำความเข้าใจปฏิกิริยา

บทนี้ครอบคลุม

- รัฐทำงานอย่างไรและเหตุใดจึงสำคัญ
- คุณสมบัติทำงานอย่างไรและเหตุใดจึงสำคัญ
- ทำความเข้าใจองค์ประกอบปฏิกิริยา

สเปค

- การใช้วิธีการ React lifecycle

เมื่อเราพูดถึงพื้นฐานแล้ว ก็ถึงเวลาดำดิ่งสู่ปัจจัยพื้นฐานอื่นๆ-
ชิ้นส่วนที่ประกอบขึ้นเป็น React และ React Native เราจะหารือถึงวิธีจัดการรัฐ
และข้อมูล และวิธีส่งข้อมูลผ่านแอปพลิเคชัน เราจะดำดิ่งลึกลงไปด้วย
โดยสาธิตวิธีการส่งผ่านคุณสมบัติ (อุปกรณ์ประกอบฉาก) ระหว่างส่วนประกอบและ
วิธีการ
จัดการอุปกรณ์ประกอบฉากเหล่านี้จากบนลงล่าง
หลังจากที่คุณมีความรู้เกี่ยวกับสถานะและอุปกรณ์ประกอบฉากแล้ว เราจะเจาะลึกลงไป
ใน
วิธีใช้วิธีวงจรชีวิต React ในตัว วิธีการเหล่านี้ช่วยให้คุณดำเนินการได้
การกระทำบางอย่างเมื่อมีการสร้างหรือทำลายส่วนประกอบ การทำความเข้าใจพวกเขา
คือ
กุญแจสำคัญในการทำความเข้าใจว่า React และ React Native ทำงานอย่างไร
และจะใช้งาน advan-
อายุของกรอบ วิธีวงจรชีวิตเป็นส่วนที่ใหญ่ที่สุดเช่นกัน
ของ React และ React Native
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 28

หมายเหตุคุณจะเห็นทั้ง React และ React Native อ้างอิงในบทนี้ เก็บไว้
ในใจว่าเมื่อผมพูดถึง React ผมไม่ได้พูดถึงสิ่งที่สเป-

cific กับ React Native แต่แนวคิดที่เกี่ยวข้องกับทั้ง React และ React พื้นเมือง. ตัวอย่างเช่น สถานะและอุปกรณ์ประกอบฉากทำงานเหมือนกันทั้งใน React และ React
Native เช่นเดียวกับวงจรชีวิต React และข้อกำหนดส่วนประกอบ React

2.1 การจัดการข้อมูลส่วนประกอบโดยใช้สถานะ

วิธีหนึ่งในการสร้างและจัดการข้อมูลใน React หรือ React Native component
คือการใช้สถานะ สถานะคอมโพเนนต์ถูกประกาศเมื่อสร้างส่วนประกอบและ
โครงสร้างเป็นวัตถุ JavaScript ธรรมดา สถานะสามารถอัพเดตได้ภายใน
ส่วนประกอบโดยใช้
ฟังก์ชันที่เรียกว่าsetStateที่เราจะเจาะลึกกันในไม่ช้านี้
อีกวิธีหนึ่งในการจัดการข้อมูลคือการใช้อุปกรณ์ประกอบฉาก อุปกรณ์ประกอบฉากจะ
ถูกส่งต่อเป็น
พารามิเตอร์เมื่อสร้างส่วนประกอบ ต่างจากรัฐไม่สามารถอัปเดตได้ภายใน
องค์ประกอบ

2.1.1 จัดการสถานะส่วนประกอบอย่างถูกต้อง

สถานะคือชุดของค่าที่ส่วนประกอบจัดการ React คิดว่า UI เป็นเรื่องง่าย
เครื่องจักรของรัฐ เมื่อสถานะของส่วนประกอบเปลี่ยนแปลงโดยใช้ฟังก์ชันsetState
React แสดงองค์ประกอบใหม่ หากองค์ประกอบย่อยใด ๆ กำลังสืบทอดสถานะนี้เป็น
อุปกรณ์ประกอบฉาก จากนั้นส่วนประกอบย่อยทั้งหมดจะถูกแสดงผลใหม่เช่นกัน
เมื่อสร้างแอปพลิเคชันโดยใช้ React Native การทำความเข้าใจว่าสถานะทำงาน
อย่างไร

พื้นฐานเพราะสถานะเป็นตัวกำหนดว่าองค์ประกอบเก็บสถานะแสดงผลและทำงานอย่างไร

สถานะคอมโพเนนต์คือสิ่งที่ช่วยให้คุณสร้างส่วนประกอบที่เป็นไดนามิกและโต้ตอบได้ประสงค์. ประเด็นหลักที่ต้องทำความเข้าใจเมื่อแยกความแตกต่างระหว่างสถานะและอุปกรณ์ประกอบฉากคือ

สถานะไม่แน่นอนในขณะที่อุปกรณ์ประกอบฉากจะไม่เปลี่ยนรูป

S ETTINGฉันNITIAL S TATE

สถานะเริ่มต้นขึ้นเมื่อสร้างส่วนประกอบใน Constructor หรือด้วย a ตัวเริ่มต้นคุณสมบัติ เมื่อเตรียมใช้งานสถานะแล้ว จะพร้อมใช้งานในองค์ประกอบเป็น this.state . รายการต่อไปนี้แสดงตัวอย่าง

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
คลาส MyComponent ขยาย React.Component {
รัฐ = {
ปี: 2016,
ชื่อ 'นาเดอร์ ดาบิต'
สี: ['สีน้ำเงิน']
}
เรนเดอร์ () {
กลับ (
<ดู>
<Text>ฉันชื่อ: { this.state.name }</Text>
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

29

การจัดการข้อมูลส่วนประกอบโดยใช้ state

```
<Text>ปีคือ: { this.state.year }</Text>
<Text>สีของฉันคือ { this.state.colors[0] }</Text>
</ดู>
)
}
}
```

ตัวสร้างฟังก์ชั่นที่เรียกว่าขณะนี้ระดับ JavaScript ถูกสร้างเป็น ปรากฏในรายการต่อไป นี่ไม่ใช่วิธี React lifecycle แต่เป็น JavaScript . ปกติ

วิธีการเรียน

```
นำเข้า React {Component} จาก 'react'
คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
ปี: 2016,
ชื่อ 'นาเดอร์ ดาบิต'
สี: ['สีน้ำเงิน']
}
}
เรนเดอร์ () {
กลับ (
<ดู>
<Text>ฉันชื่อ: { this.state.name }</Text>
```

<Text>ปีคือ: { this.state.year }</Text>
<Text>สีของฉันคือ { this.state.colors[0] }</Text>
</ดู>
)
}
}
ตัวสร้างและตัวเริ่มต้นคุณสมบัติทั้งคู่ทำงานเหมือนกันทุกประการและที่
วิธีการที่คุณใช้ขึ้นอยู่กับความชอบ

U PDATING รัฐ

สถานะสามารถอัปเดตได้โดยการเรียกthis.setState(object)ผ่านวัตถุด้วย
สถานะใหม่ที่คุณต้องการใช้ setStateรวมสถานะก่อนหน้ากับสถานะปัจจุบัน
ดังนั้นหากคุณส่งผ่านรายการเดียว (คู่คีย์-ค่า) ส่วนที่เหลือของรัฐจะยังคงเป็น
เดียวกันในขณะที่รายการใหม่ในสถานะจะถูกเขียนทับ
มาดูวิธีการใช้setState (ดูรายการ 2.3) ในการทำเช่นนั้น เราจะมาแนะนำ . ใหม่
วิธีการจัดการที่เรียกว่าสัมผัสonPress onPressสามารถเรียกได้สองสามประเภท
"tap-
pable” ตอบสนององค์ประกอบดั้งเดิม แต่ที่นี่คุณจะแนบไปกับองค์ประกอบข้อความ
เพื่อรับ
เริ่มต้นด้วยตัวอย่างพื้นฐานนี้ คุณจะเรียกใช้ฟังก์ชันที่เรียกว่าupdateYearเมื่อ
ข้อความ
กดเพื่ออัพเดตรัฐที่มีsetState ฟังก์ชันนี้จะถูกกำหนดก่อน
ฟังก์ชันการแสดงผลเพราะโดยปกติแล้วจะเป็นแนวทางปฏิบัติที่ดีที่สุดในการกำหนด
วิธีการที่กำหนดเองใดๆ มาก่อน
วิธีการเรนเดอร์ แต่โปรดจำไว้ว่าลำดับของคำจำกัดความของฟังก์ชัน
ไม่กระทบการใช้งานจริง

หน้า 30


```
นำเข้า React {Component} จาก 'react'
คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
ปี: 2016,
}
}
อัปเดตปี () {
this.setState({
ปี: 2017
})
}
เรนเดอร์ () {
กลับ (
<ดู>
<ข้อความ
onPress={() => this.updateYear()}>
ปีคือ: { this.state.year }
</Text>
</ดู>
```

)
}
}
รูปที่ 2.1 แสดงวิธีการอัพเดตสถานะทุกครั้งที่องค์ประกอบข้อความในรายการ 2.3 is กด ทุกครั้งที่setStateเรียกว่าตอบสนองจะ rerender ส่วนประกอบ (โทร ทำให้วิธีการอีกครั้ง) และส่วนประกอบเด็ก ๆ การเรียกthis.setStateคือ วิธีการเปลี่ยนตัวแปรรัฐและเรียกทำให้วิธีการอีกครั้งเพราะการเปลี่ยนแปลง ตัวแปรสถานะโดยตรงจะไม่ทริกเกอร์การเรนเดอร์องค์ประกอบซ้ำ ดังนั้นจึงไม่ การเปลี่ยนแปลงจะปรากฏใน UI ข้อผิดพลาดทั่วไปสำหรับผู้เริ่มต้นคือการอัปเดต สถานะ
ตัวแปรโดยตรง ตัวอย่างเช่น สิ่งต่อไปนี้ใช้ไม่ได้เมื่อพยายาม
รูปที่ 2.1 การไหลของ setState โดยมีลูกศรแสดงเมื่อข้อความ
องค์ประกอบถูกกด ทรัพย์สินปีของรัฐเริ่มต้นเป็น 2016 ใน
ตัวสร้าง ทุกครั้งที่กดข้อความ จะมีการตั้งค่าคุณสมบัติปีของรัฐ
ถึงปี 2560
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 31

31
การจัดการข้อมูลส่วนประกอบโดยใช้ state
เพื่ออัปเดตสถานะ— อ็อบเจ็กต์สถานะได้รับการอัปเดต แต่ UI ไม่ได้รับการอัปเดต เนื่องจากตั้งค่า-
รัฐไม่ได้เรียกว่าและส่วนประกอบที่ไม่ได้ rerendered:

```
คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
ปี: 2016,
}
}
อัปเดตปี () {
this.state.year = 2017
}
เรนเดอร์ () {
กลับ (
<ดู>
<ข้อความ
onPress={() => this.updateYear()}>
ปีคือ: { this.state.year }
</Text>
</ดู>
)
}
}
```

แต่มีเมธอดใน React ที่สามารถบังคับให้อัปเดตเมื่อตัวแปรสถานะมีถูกเปลี่ยนเหมือนในตัวอย่างก่อนหน้า วิธีนี้เรียกว่าforceUpdate ; ดูรายการ-หรือ 2.4. โทรforceUpdateสาเหตุRenderจะเรียกว่าในส่วนที่เรียกการแสดงผลซ้ำของ UI การใช้forceUpdateมักไม่จำเป็นหรือแนะนำแต่ควรทราบในกรณีที่คุณพบในตัวอย่างหรือเอกสารประกอบ ที่สุดของเวลานั้น การแสดงซ้ำนี้สามารถจัดการได้โดยใช้วิธีการอื่น เช่น การเรียกset-ระบุหรือส่งต่ออุปกรณ์ประกอบฉากใหม่

คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
ปี: 2016
}
}
อัปเดตปี () {
this.state.year = 2017
}
อัปเดต() {
this.forceUpdate()
}
เรนเดอร์ () {
กลับ (
<ดู>
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

<Text onPress={ () => this.updateYear() }>
ปีคือ: { this.state.year }
</Text>
<ข้อความ
onPress={ () => นี่. อัปเดต () }>บังคับอัปเดต
</Text>

</ดู>
)
}
}
ตอนนี้เรามาดูวิธีการทำงานกับ state โดยใช้สตริงพื้นฐานกันไปแล้ว มาดูบางส่วนกัน ประเภทข้อมูลอื่นๆ คุณจะแนบบูลีน อาร์เรย์ และอ็อบเจ็กต์เข้ากับสถานะและใช้งาน ในองค์ประกอบ นอกจากนี้คุณยังจะแสดงองค์ประกอบตามเงื่อนไขบูลีนใน รัฐ.

คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
ปี: 2016,
leapYear: จริง,
หัวข้อ: ['React', 'React Native', 'JavaScript'],
ข้อมูล: {
หนังสือปกอ่อน: จริง,
ความยาว: '335 หน้า',
ประเภท: 'การเขียนโปรแกรม'
}
}
}
เรนเดอร์ () {
let leapyear = <Text>นี่ไม่ใช่ปีอธิกสุรทิน!</Text>
ถ้า (this.state.leapYear) {
leapyear = <Text>นี่คือปีอธิกสุรทิน!</Text>
}
กลับ (

<ดู>
<ข้อความ>{ this.state.year }</Text>
<ข้อความ>ความยาว: { this.state.info.length }</Text>
<Text>ประเภท: { this.state.info.type }</Text>
{ ปีอธิกสุรทิน }
</ดู>
)
}
}

2.2 การจัดการข้อมูลส่วนประกอบโดยใช้อุปกรณ์ประกอบฉาก

อุปกรณ์ประกอบฉาก (ย่อมาจากคุณสมบัติ) เป็นค่าที่สืบทอดมาของส่วนประกอบหรือคุณสมบัติที่มี
ได้รับการถ่ายทอดจากองค์ประกอบหลัก อุปกรณ์ประกอบฉากอาจเป็นได้ทั้งแบบคงที่และแบบไดนามิก

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

ค่าเมื่อมีการประกาศ แต่เมื่อสืบทอด ค่าเหล่านั้นจะไม่เปลี่ยนรูป พวกเขาสามารถเปลี่ยนแปลงได้โดยการเปลี่ยนค่าเริ่มต้นที่ระดับบนสุดที่มีการประกาศเท่านั้น
และล่วงลับไปแล้ว เอกสารประกอบเรื่อง “Thinking in React” ของ React ระบุว่าอุปกรณ์ประกอบฉากนั้นดีที่สุด

อธิบายว่าเป็น "วิธีการส่งข้อมูลจากพ่อแม่สู่ลูก" ตาราง 2.1 ไฮไลท์บางส่วนของความแตกต่างและความคล้ายคลึงระหว่างอุปกรณ์ประกอบฉากและสถานะ

ตารางที่ 2.1 อุปกรณ์ประกอบฉากกับสถานะ

ข้อมูลภายนอก	ข้อมูลภายใน
ไม่เปลี่ยนรูป	เปลี่ยนแปลงได้
สืบทอดมาจากบิดามารดา	สร้างขึ้นในองค์ประกอบ
สามารถเปลี่ยนได้โดยองค์ประกอบหลัก	อัพเดทได้เฉพาะในคอมโพเนนท์
สามารถส่งต่อเป็นอุปกรณ์ประกอบฉากได้	สามารถส่งต่อเป็นอุปกรณ์ประกอบฉากได้
ไม่สามารถเปลี่ยนภายในส่วนประกอบได้	สามารถเปลี่ยนภายในส่วนประกอบได้

วิธีที่ดีในการอธิบายวิธีการทำงานของอุปกรณ์ประกอบฉากคือการแสดงตัวอย่าง รายการต่อไปนี้

ประกาศมูลค่าทางบัญชีและส่งต่อไปยังส่วนประกอบย่อยเป็นพร็อพคงที่

```
คลาส MyComponent ขยายส่วนประกอบ {
เรนเดอร์ () {
กลับ (
<BookDisplay book="React Native in Action" />
)
}
}
คลาส BookDisplay ขยายส่วนประกอบ {
```

เรนเดอร์ () {
กลับ (
<ดู>
<ข้อความ>{ this.props.book }</Text>
</ดู>
)
}
}
รหัสนี้สร้างสององค์ประกอบ: <MyComponent />และ <BookDisplay /> เมื่อไหร่
คุณสร้าง<BookDisplay />คุณส่งผ่านคุณสมบัติที่เรียกว่าbookและตั้งค่าเป็นสตริง
“ตอบสนองพื้นเมืองในการดำเนินการ”. สิ่งใดที่ผ่านเป็นทรัพย์สินในลักษณะนี้มีให้ในองค์ประกอบของเด็กเป็นthis.props
คุณยังสามารถส่งต่อตัวอักษรเหมือนกับที่คุณส่งตัวแปร โดยใช้วงเล็บปีกกาและ a ค่าสตริงดังแสดงต่อไป
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 34

คลาส MyComponent ขยายส่วนประกอบ {
เรนเดอร์ () {

```
กลับ (
<BookDisplay book={"React Native in Action"} />
)
}
}
คลาส BookDisplay ขยายส่วนประกอบ {
เรนเดอร์ () {
กลับ (
<ดู>
<ข้อความ>{ this.props.book }</Text>
</ดู>
)
}
}
```

D YNAMIC PROPS

ถัดไป ส่งคุณสมบัติไดนามิกไปยังส่วนประกอบ ในการทำให้วิธีการก่อน ส่งคืนคำสั่งประกาศหนังสือตัวแปรและส่งผ่านเป็นพร็อพ

```
คลาส MyComponent ขยายส่วนประกอบ {
เรนเดอร์ () {
ให้ book = 'ตอบสนอง Native ในการดำเนินการ'
กลับ (
<BookDisplay book={ book } />
)
}
}
คลาส BookDisplay ขยายส่วนประกอบ {
เรนเดอร์ () {
กลับ (
<ดู>
```

<ข้อความ>{ this.props.book }</Text>
</ดู>
)
}
}
ตอนนี้ ส่งคุณสมบัติไดนามิกไปยังส่วนประกอบโดยใช้สถานะ

คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง () {
ซุปเปอร์()
this.state = {
หนังสือ: 'React Native in Action'

หน้า 35

}
}
เรนเดอร์ () {
กลับ (
<BookDisplay book={this.state.book} />
)
}
}
คลาส BookDisplay ขยายส่วนประกอบ {
เรนเดอร์ () {

```
กลับ (
<ดู>
<ข้อความ>{ this.props.book }</Text>
</ดู>
)
}
}
```

ต่อไปเรามาดูวิธีการอัพเดทสถานะและค่าที่ส่งต่อลงมาเป็นเสาเพื่อBookDisplay จำไว้ว่าอุปกรณ์ประกอบฉากนั้นไม่สามารถเปลี่ยนแปลงได้ ดังนั้นคุณจะต้องเปลี่ยนสถานะของคอมโพเนนต์หลัก (MyComponent) ซึ่งจะจัดหาค่าใหม่ให้กับBook-แสดงพร็อพหนังสือและทริกเกอร์การเรนเดอร์ทั้งส่วนประกอบและคอมลูกponent การแบ่งแนวคิดนี้ออกเป็นส่วนๆ สิ่งที่ต้องทำมีดังนี้

1 ประกาศตัวแปรสถานะ:

```
this.state = {
หนังสือ: 'React Native in Action'
}
```

2 เขียนฟังก์ชันที่จะอัปเดตตัวแปรสถานะ:

```
updateBook() {
this.setState({
หนังสือ: 'ด่วนในการดำเนินการ'
})
}
```

3 ส่งฟังก์ชันและสถานะลงไปที่องค์ประกอบย่อยเป็นอุปกรณ์ประกอบฉาก:

```
<การแสดงหนังสือ
updateBook={ () => this.updateBook() }
book={ this.state.book } />
```

4 แนบฟังก์ชันกับตัวจัดการแบบสัมผัสในคอมโพเนนต์ย่อย:

```
<Text onPress={ this.props.updateBook }>
```

เมื่อคุณทราบชิ้นส่วนที่ต้องการแล้ว คุณสามารถเขียนโค้ดเพื่อนำไปใช้จริงได้ คุณจะใช้ส่วนประกอบจากตัวอย่างก่อนหน้านี้และเพิ่มฟังก์ชันการทำงานใหม่

```
คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 36


```
this.state = {
หนังสือ: 'React Native in Action'
}
}
updateBook() {
this.setState({
หนังสือ: 'ด่วนในการดำเนินการ'
})
}
เรนเดอร์ () {
กลับ (
<การแสดงหนังสือ
updateBook={ () => this.updateBook() }
book={ this.state.book } />
)
}
```

```
}
คลาส BookDisplay ขยายส่วนประกอบ {
เรนเดอร์ () {
กลับ (
<ดู>
<ข้อความ
onPress={ this.props.updateBook }>
{ this.props.book }
</Text>
</ดู>
)
}
}
```

D โครงสร้างอุปกรณ์ประกอบฉากและรัฐ

อ้างถึงรัฐและอุปกรณ์ประกอบฉากอย่างต่อเนื่องเนื่องจากthis.stateและ this.propsสามารถทำซ้ำได้-
ละเมิดหลักการ DRY (อย่าทำซ้ำตัวเอง) ที่พวกเราหลายคนพยายามปฏิบัติตาม
ในการแก้ไขปัญหานี้ คุณสามารถลองใช้การทำลายโครงสร้าง Destructuring เป็น
คุณลักษณะใหม่ของ JavaScript ที่
ถูกเพิ่มเข้ามาเป็นส่วนหนึ่งของข้อกำหนด ES2015 และพร้อมใช้งานในแอปพลิเคชัน
React Native NS
แนวคิดพื้นฐานคือคุณสามารถใช้คุณสมบัติจากวัตถุและใช้เป็นตัวแปรใน
แอพ:

```
คน const = { ชื่อ: 'เจฟฟ์' อายุ: 22 }
const { อายุ } = คน
console.log(age) #22
```

เขียนส่วนประกอบโดยใช้ destructuring ดังที่แสดงต่อไป

คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
หนังสือ: 'React Native in Action'
}
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 37

37
การจัดการข้อมูลส่วนประกอบโดยใช้อุปกรณ์ประกอบฉาก
}
updateBook() {
this.setState ({ หนังสือ: 'ด่วนในการดำเนินการ' })
}
เรนเดอร์ () {
const { book } = this.state
กลับ (
<การแสดงหนังสือ
updateBook={ () => this.updateBook() }
book={ book } />
)
}
}
คลาส BookDisplay ขยายส่วนประกอบ {
เรนเดอร์ () {
const { หนังสือ updateBook } = this.props

```
กลับ (
<ดู>
<ข้อความ
onPress={ updateBook }>
{ หนังสือ }
</Text>
</ดู>
)
}
}
```

คุณไม่จำเป็นต้องอ้างถึงthis.stateหรือthis.propsในองค์ประกอบเมื่อ
อ้างอิงหนังสือ; คุณได้นำตัวแปรbookออกจากสถานะและ
อุปกรณ์ประกอบฉากและสามารถอ้างอิงตัวแปรได้เอง สิ่งนี้เริ่มสมเหตุสมผลมากขึ้นและจะ
ทำให้โค้ดของคุณสะอาดขึ้นเมื่อสถานะและอุปกรณ์ประกอบฉากของคุณมีขนาดใหญ่ขึ้นและซับซ้อนมากขึ้น

P ROPS พร้อมส่วนประกอบที่ไม่มีสถานะ

เนื่องจากส่วนประกอบไร้สัญชาติต้องกังวลเกี่ยวกับอุปกรณ์ประกอบฉากและไม่มีของตัวเอง
ระบุว่ามีประโยชน์อย่างยิ่งเมื่อสร้างส่วนประกอบที่นำกลับมาใช้ใหม่ได้ มาดูกันว่าเป็นอย่างไร

อุปกรณ์ประกอบฉากใช้ในองค์ประกอบไร้สัญชาติ

หากต้องการเข้าถึงอุปกรณ์ประกอบฉากโดยใช้องค์ประกอบไร้สัญชาติ ให้ส่งอุปกรณ์ประกอบฉากเป็นอาร์กิวเมนต์แรกไปที่
ฟังก์ชั่น.

```
const BookDisplay = (อุปกรณ์ประกอบฉาก) => {
const { book, updateBook } = อุปกรณ์ประกอบฉาก
กลับ (
<ดู>
<ข้อความ
onPress={ updateBook }>
{ หนังสือ }
</Text>
</ดู>
)
}
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 38

คุณยังสามารถทำลายอุปกรณ์ประกอบฉากในอาร์กิวเมนต์ของฟังก์ชันได้

```
const BookDisplay = ({ updateBook หนังสือ }) => {
กลับ (
<ดู>
<ข้อความ
onPress={ updateBook }>
{ หนังสือ }
</Text>
</ดู>
```

)
}
มันดูดีกว่ามากและล้างโค้ดที่ไม่จำเป็นออกไปมากมาย! คุณควรใช้ state-
ส่วนประกอบน้อยลงทุกที่ที่คุณทำได้ ทำให้ codebase และตรรกะของคุณง่ายขึ้น
หมายเหตุส่วนประกอบไร้สัญชาติมักถูกอ้างถึงเป็นส่วนประกอบที่ใช้งานได้
เพราะสามารถเขียนเป็นฟังก์ชันใน JavaScript ได้

P ASSING Arrays และ OBJECTS AS PROPS

ชนิดข้อมูลอื่นๆ ทำงานตรงตามที่คุณคาดหวัง ตัวอย่างเช่น ในการส่งผ่านอาร์เรย์ คุณ
ส่งผ่านในอาร์เรย์เป็นเสา ในการส่งผ่านวัตถุ คุณต้องส่งผ่านวัตถุเป็นอุปกรณ์ประกอบ
ฉาก มาดูกัน
ในตัวอย่างพื้นฐาน

```
คลาส MyComponent ขยายส่วนประกอบ {
ตัวสร้าง (){
ซุปเปอร์()
this.state = {
leapYear: จริง,
ข้อมูล: {
ประเภท: 'การเขียนโปรแกรม'
}
}
}
เรนเดอร์ () {
กลับ (
<การแสดงหนังสือ
leapYear={ this.state.leapYear }
ข้อมูล={ this.state.info }
หัวข้อ={['React', 'React Native', 'JavaScript']} />
```

)
}
}
const BookDisplay = (อุปกรณ์ประกอบฉาก) => {
ปล่อยให้ปีอธิกสุรทิน
ให้ { หัวข้อ } = อุปกรณ์ประกอบฉาก
const { ข้อมูล } = อุปกรณ์ประกอบฉาก
หัวข้อ = หัวข้อ. แผนที่ ((หัวข้อ, ผม) => {
ส่งคืน <Text>{ หัวข้อ }</Text>
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 39

ปฏิกิริยาส่วนประกอบจำเพาะ
})
ถ้า (props.leapYear) {
leapyear = <Text>นี่คือปีอธิกสุรทิน!</Text>
}
กลับ (
<ดู>
{ ปีอธิกสุรทิน }
<Text>ประเภทหนังสือ: { info.type }</Text>
{ หัวข้อ }
</ดู>
)
}

2.3 ข้อกำหนดส่วนประกอบที่ทำปฏิกิริยา

เมื่อสร้างส่วนประกอบ React และ React Native คุณสามารถขอข้อมูลได้หลายอย่าง

fications และวิธีการวงจรชีวิตเพื่อควบคุมสิ่งที่เกิดขึ้นในองค์ประกอบของคุณ ในเรื่องนี้

เราจะพูดถึงพวกเขาและทำให้คุณเข้าใจถึงสิ่งที่แต่ละคนทำ

และเมื่อใดที่คุณควรใช้

อันดับแรก เราจะพูดถึงพื้นฐานของข้อกำหนดส่วนประกอบ ส่วนประกอบเฉพาะ ca-

โดยพื้นฐานแล้วจะกำหนดวิธีที่ส่วนประกอบควรตอบสนองต่อสิ่งต่าง ๆ ที่เกิดขึ้นในวงจรชีวิตของส่วนประกอบ ข้อกำหนดมีดังนี้:

- ทำให้วิธีการ
- วิธีการสร้าง
- สถิตวัตถุที่ใช้ในการกำหนดวิธีการคงใช้ได้กับชั้นเรียน

2.3.1 การใช้วิธีการเรนเดอร์เพื่อสร้าง UI

ทำให้วิธีเป็นวิธีเดียวในข้อกำหนดองค์ประกอบที่จำเป็น

เมื่อสร้างส่วนประกอบ ต้องส่งคืนองค์ประกอบลูกเดียวnullหรือ

เท็จ อิลิเมนต์ลูกนี้สามารถเป็นส่วนประกอบที่คุณประกาศได้ (เช่น View หรือ Text ส่วนประกอบ) หรือส่วนประกอบอื่นที่คุณกำหนด (อาจเป็นส่วนประกอบปุ่มคุณสร้างและนำเข้าสู่ไฟล์):

```
เรนเดอร์ () {
กลับ (
<ดู>
<Text>สวัสดี</Text>
</ดู>
)
```

}
คุณสามารถใช้ทำให้วิธีการที่มีหรือไม่มีวงเล็บ หากคุณไม่ใช้ paren-
วิทยานิพนธ์ ดังนั้นองค์ประกอบที่ส่งคืนแน่นอนจะต้องอยู่ในบรรทัดเดียวกับ
ผลตอบแทน
คำแถลง:
เรนเดอร์ () {
กลับ <View><Text>สวัสดี</Text></View>
}
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 40

ทำให้วิธีนี้ยังสามารถกลับองค์ประกอบอื่นที่ถูกกำหนดไว้ที่อื่น ๆ :
เรนเดอร์ () {
ส่งคืน <SomeComponent />
}
#หรือ
เรนเดอร์ () {
กลับ (
<SomeComponent />
)
}
นอกจากนี้คุณยังสามารถตรวจสอบเงื่อนไขในการแสดงผลวิธีการดำเนินการตรรกะและ
ผลตอบแทน

ส่วนประกอบตามมูลค่า:

เรนเดอร์ () {
ถ้า (บางสิ่ง === จริง) {
ส่งคืน <SomeComponent />
} else ส่งคืน <SomeOtherComponent />
}

2.3.2 การใช้ตัวเริ่มต้นและตัวสร้างคุณสมบัติ

สถานะสามารถสร้างได้ใน Constructor หรือใช้ตัวเริ่มต้นคุณสมบัติ ตัวเริ่มต้นคุณสมบัติ
เป็นข้อกำหนด ES7 สำหรับภาษา JavaScript แต่ทำงานนอกกรอบด้วย
ตอบโต้พื้นเมือง พวกเขาให้วิธีที่กระชับในการประกาศสถานะในคลาส React:

คลาส MyComponent ขยาย React.Component {
รัฐ = {
บางจำนวน: 1,
someBoolean: เท็จ
}

คุณยังสามารถใช้เมธอดConstructorเพื่อตั้งค่าสถานะเริ่มต้นเมื่อใช้คลาส NS
แนวคิดของคลาส เช่นเดียวกับฟังก์ชันคอนสตรัคเตอร์ ไม่ได้เจาะจงสำหรับ React
หรือ React
พื้นเมือง; มันเป็นข้อกำหนดของ ES2015 และเป็นเพียงวากยสัมพันธ์ที่ด้านบนของ
JavaScript
การสืบทอดตามต้นแบบที่มีอยู่สำหรับการสร้างและการเริ่มต้นวัตถุที่สร้างขึ้น
กับชั้นเรียน คุณสมบัติอื่น ๆ ยังสามารถตั้งค่าสำหรับคลาสคอมโพเนนต์ในตัวสร้าง
โดยการประกาศด้วยไวยากรณ์this.property (คุณสมบัติเป็นชื่อของ
คุณสมบัติ). คำหลักที่นี้หมายถึงเช่นชั้นปัจจุบันคุณอยู่ใน:

ตัวสร้าง (){

ซุปเปอร์()
this.state = {
someOtherNumber: 19,
someOtherBoolean: true
}
this.name = 'สวัสดีชาวโลก'
this.type = 'คลาส'
this.loaded = false
}
เมื่อใช้ Constructor เพื่อสร้าง React class คุณต้องใช้super keyword ก่อนที่คุณจะใช้คีย์เวิร์ดนี้ได้เพราะคุณกำลังขยายคลาสอื่น นอกจากนี้ ถ้าคุณต้องเข้าถึงอุปกรณ์ประกอบฉากใด ๆ ในตัวสร้าง พวกเขาจะต้องส่งผ่านเป็นอาร์กิวเมนต์ไปยัง
คอนสตรัคเตอร์และซูเปอร์คอล
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 41

41
ปฏิกิริยาวิธีวงจรชีวิต
การกำหนดสถานะตามอุปกรณ์ประกอบฉากมักจะไม่ใช่แนวปฏิบัติที่ดี เว้นแต่คุณจะตั้งใจ-
ตั้งค่าข้อมูลเมล็ดพันธุ์บางประเภทสำหรับการทำงานภายในของส่วนประกอบเพราะข้อมูลจะไม่สอดคล้องกันในองค์ประกอบต่างๆ หากมีการเปลี่ยนแปลง รัฐเป็นเพียงครี-
เมื่อติดตั้งหรือสร้างส่วนประกอบเป็นครั้งแรก หากคุณแสดงผลคอม-

ponent โดยใช้ค่า prop ที่แตกต่างกัน แล้วอินสแตนซ์ใดๆ ของส่วนประกอบนั้นที่มีติดตั้งแล้วจะไม่ใช้ค่า prop ใหม่ในการอัพเดทสถานะ

ตัวอย่างต่อไปนี้แสดงอุปกรณ์ประกอบฉากที่ใช้ในการตั้งค่าสถานะภายในคอน

อาจารย์ สมมติว่าคุณส่ง "Nader Dabit" เป็นอุปกรณ์ประกอบฉากในตอนแรก: the

คุณสมบัติ fullName ในรัฐจะเป็น “Nader Dabit” ถ้าส่วนประกอบนั้น reren-

dered ด้วย "Another Name" คอนสตรัคเตอร์จะไม่ถูกเรียกเป็นครั้งที่สอง ดังนั้น state

ค่าสำหรับfullNameจะยังคงเป็น “Nader Dabit”:

```
ตัวสร้าง (อุปกรณ์ประกอบฉาก) {
สุดยอด (อุปกรณ์ประกอบฉาก)
this.state = {
ชื่อเต็ม: props.first + ' ' + props.last,
}
}
```

2.4 ปฏิกิริยาวิธีวงจรชีวิต

วิธีการต่าง ๆ ถูกดำเนินการที่จุดเฉพาะในวงจรชีวิตของส่วนประกอบ: เหล่านี้คือ

เรียกว่าวิธีวงจรชีวิต การเข้าใจวิธีการทำงานมีความสำคัญเพราะว่า

อนุญาตให้คุณดำเนินการเฉพาะที่จุดต่าง ๆ ในการสร้างและทำลาย

ของส่วนประกอบ ตัวอย่างเช่น สมมติว่าคุณต้องการเรียกใช้ API ที่ส่งคืน

ข้อมูลบางอย่าง คุณอาจต้องการตรวจสอบให้แน่ใจว่าส่วนประกอบพร้อมที่จะแสดงผลนี้ data ดังนั้นคุณจะต้องทำการเรียก API เมื่อส่วนประกอบถูกติดตั้งในเมธอดที่เรียกว่า componentDidMount ในส่วนนี้ เราจะพูดถึงวิธีวงจรชีวิตและอธิบาย

พวกเขาทำงานอย่างไร

อายุการใช้งานของส่วนประกอบ React มีสามขั้นตอน: การสร้าง (การติดตั้ง) การอัปเดต และ
การลบ (ยกเลิกการต่อเชื่อม) ในระหว่างสามขั้นตอนนี้ คุณสามารถขอชีวิตสามชุด - วิธีวงจร:

- การติดตั้ง (การสร้าง) —เมื่อมีการสร้างส่วนประกอบ ชุดของวิธีวงจรชีวิตถูกทริกเกอร์และคุณมีตัวเลือกที่จะเชื่อมต่อกับส่วนใดส่วนหนึ่งหรือทั้งหมด: construc-
Tor , getDerivedStateFromProps ,การแสดงผลและ componentDidMount หนึ่ง
วิธีการที่คุณใช้จนถึงตอนนี้คือrenderซึ่งแสดงผลและส่งคืน UI
- การอัปเดต — เมื่อส่วนประกอบอัปเดต วิธีการวงจรการอัพเดทจะถูกเรียกใช้gered: getDerivedStateFromProps (เมื่ออุปกรณ์ประกอบฉากเปลี่ยน) shouldComponentUp-
วันที่ ,การแสดงผล , getSnapshotBeforeUpdate และ componentDidUpdate การปรับปรุง
สามารถเกิดขึ้นได้สองวิธี:

– เมื่อมีการเรียกsetState หรือforceUpdate ภายใน component
– เมื่ออุปกรณ์ประกอบฉากใหม่ถูกส่งต่อไปยังส่วนประกอบ

- Unmounting — เมื่อส่วนประกอบถูกถอดออก (ถูกทำลาย) วงจรชีวิตขั้นสุดท้าย
วิธีการจะถูกเรียก: componentWillUnmount

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

2.4.1 เมธอด getDerivedStateFromProps แบบคงที่

getDerivedStateFromProps เป็นเมธอดคลาสสแตติกที่เรียกว่าทั้งสองเมื่อ
คอม
ponent ถูกสร้างขึ้นและเมื่อได้รับอุปกรณ์ประกอบฉากใหม่ วิธีนี้ได้รับพร็อพใหม่
และสถานะล่าสุดเป็นอาร์กิวเมนต์และส่งกลับวัตถุ ข้อมูลในวัตถุคือ
ปรับปรุงให้เป็นรัฐ รายการต่อไปนี้แสดงตัวอย่าง

```
ส่งออกแอปคลาสเริ่มต้นขยายคอมโพเนนต์ {
รัฐ = {
ผู้ใช้เข้าสู่ระบบ: false
}
คงที่ getDerivedStateFromProps (nextProps, nextState) {
ถ้า (nextProps.user.authenticated) {
กลับ {
ผู้ใช้เข้าสู่ระบบ: true
}
}
คืนค่า null
}
เรนเดอร์ () {
กลับ (
<ดู style={styles.container}>
{
```

```
this.state.userLogedIn && (
<AuthenticatedComponent />>
)
}
</ดู>
);
}
}
```

2.4.2 วิธีวงจรชีวิต componentDidMount

componentDidMount ถูกเรียกเพียงครั้งเดียว หลังจากที่โหลดส่วนประกอบแล้ว

วิธีนี้เป็นวิธีที่ดีในการดึงข้อมูลด้วยการเรียก AJAX ดำเนินการ setTimeout func-
และรวมเข้ากับเฟรมเวิร์ก JavaScript อื่นๆ

```
คลาส MainComponent ขยายส่วนประกอบ {
ตัวสร้าง () {
ซุปเปอร์()
this.state = { กำลังโหลด: จริง ข้อมูล: {} }
}
componentDidMount () {
#จำลองการโทรอาแจ็กซ์
setTimeout(() => {
this.setState({
กำลังโหลด: เท็จ
ข้อมูล: {ชื่อ: 'Nader Dabit' อายุ: 35}
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

```
})
}, 2000)
}
เรนเดอร์ () {
ถ้า (this.state.loading) {
return <Text>กำลังโหลด</Text>
}
const { ชื่อ, อายุ } = this.state.data
กลับ (
<ดู>
<Text>ชื่อ: {name}</Text>
<Text>อายุ: {age}</Text>
</ดู>
)
}
}
```

2.4.3 วิธีวงจรชีวิต shouldComponentUpdate

shouldComponentUpdate ส่งคืนบูลีนและช่วยให้คุณตัดสินใจได้ว่าเมื่อใดคอมโพเนนต์

แสดงผล หากคุณรู้ว่าสถานะใหม่หรืออุปกรณ์ประกอบฉากจะไม่ต้องการส่วนประกอบหรือส่วนประกอบใดๆ ของ

ลูกของมันที่จะแสดงผล คุณสามารถคืนค่าfalseได้ หากคุณต้องการให้องค์ประกอบแสดงผลใหม่

กลับมาจริง

```
คลาส MainComponent ขยายส่วนประกอบ {
shouldComponentUpdate (nextProps, nextState) {
if(nextProps.name !== this.props.name) {
คืนความจริง
}
คืนค่าเท็จ
}
เรนเดอร์ () {
ส่งคืน <SomeComponent />
}
}
```

2.4.4 วิธี componentDidUpdate lifecycle

componentDidUpdate ถูกเรียกใช้ทันทีหลังจากส่วนประกอบได้รับการอัพเดต

และแสดงผลใหม่ คุณได้รับสถานะก่อนหน้าและอุปกรณ์ประกอบฉากก่อนหน้าเป็นอาร์กิวเมนต์

```
คลาส MainComponent ขยายส่วนประกอบ {
componentDidUpdate (prevProps, prevState) {
ถ้า (prevState.showToggled === this.state.showToggled) {
this.setState({
showToggled: !showToggled
})
}
}
```

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc

หน้า 44

เรนเดอร์ () {
ส่งคืน <SomeComponent />
}
}

2.4.5 วิธี componentWillUnmount lifecycle

componentWillUnmount ถูกเรียกก่อนที่ส่วนประกอบจะถูกลบออกจากแอปพลิเคชัน
ชั่น ที่นี่ คุณสามารถดำเนินการล้างข้อมูลที่จำเป็น ลบผู้ฟัง และลบเวลา
ERS ที่ถูกตั้งขึ้นในcomponentDidMount

```
คลาส MainComponent ขยายส่วนประกอบ {
แฮนเดิลคลิก () {
this._timeout = setTimeout(() => {
this.openWidget();
}, 2000);
}
componentWillUnmount() {
clearTimeout(this._timeout);
}
เรนเดอร์ () {
ส่งคืน <SomeComponent
handleClick={() => this.handleClick()} />
}
```

}

สรุป

- State เป็นวิธีการจัดการข้อมูลในส่วนประกอบ React กำลังอัปเดตสถานะ
แสดงผล
UI ของส่วนประกอบและส่วนประกอบย่อยใดๆ ที่ใช้ข้อมูลนี้เป็นอุปกรณ์ประกอบฉาก
- คุณสมบัติ (อุปกรณ์ประกอบฉาก) คือวิธีที่ข้อมูลถูกส่งผ่านผ่านแอปพลิเคชัน React
Native
ไอออนบวกกับส่วนประกอบย่อย การอัปเดตอุปกรณ์ประกอบฉากจะอัปเดต
ส่วนประกอบใด ๆ โดยอัตโนมัติ
เต็นท์ที่ได้รับอุปกรณ์ประกอบฉากเดียวกัน
- ข้อมูลจำเพาะส่วนประกอบ React คือกลุ่มของวิธีการและคุณสมบัติใน React
องค์ประกอบที่ระบุการประกาศส่วนประกอบ เรนเดอร์เท่านั้น
วิธีที่จำเป็นเมื่อสร้างองค์ประกอบ React วิธีอื่นๆ และ
คุณสมบัติเป็นตัวเลือก
- มีสามขั้นตอนหลักในวงจรชีวิตของส่วนประกอบ React: การสร้าง (เมานต์-
ing) การอัปเดตและการลบ (ยกเลิกการต่อเชื่อม) แต่ละคนมีวงจรชีวิตของตัวเอง
วิธีการ
- วิธีวงจรชีวิตปฏิกิริยามีอยู่ในองค์ประกอบ React และดำเนินการที่
จุดเฉพาะในวงจรชีวิตของส่วนประกอบ พวกเขาควบคุมวิธีการส่วนประกอบ
ฟังก์ชั่นและการอัพเดท

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

สร้างแรกของคุณ

React Native app

บทนี้ครอบคลุม

- สร้างแอปสิ่งที่ต้องทำตั้งแต่เริ่มต้น
- การดีบักแบบเบา

เมื่อเรียนรู้กรอบ เทคโนโลยี ภาษา หรือแนวคิดใหม่ๆ ให้ดำน้ำโดยตรง ในกระบวนการ โดยการสร้างแอปจริงเป็นวิธีที่ดีในการเริ่มต้นการเรียนรู้อย่างมืออาชีพ ภายี ตอนนี้คุณเข้าใจพื้นฐานของวิธีการทำงานของ React และ React Native แล้ว มาทำกัน

นำชิ้นส่วนเหล่านี้มารวมกันเพื่อสร้างแอปแรกของคุณ: แอปสิ่งที่ต้องทำ ผ่านโปร- การสร้างแอพขนาดเล็กและการใช้ข้อมูลที่เราได้ทำไปแล้วจะเป็น

วิธีที่ดีในการเสริมสร้างความเข้าใจเกี่ยวกับวิธีใช้ React Native
คุณจะใช้ฟังก์ชันบางอย่างในแอปที่เรายังไม่ได้กล่าวถึงในเชิงลึกและ
ความแตกต่างด้านสไตล์บางอย่างที่เรายังไม่ได้พูดคุยกัน แต่ไม่ต้องกังวล แทนที่จะข้าม
สิ่งเหล่านี้
แนวคิดใหม่ๆ ทีละอย่างในตอนนี้ คุณจะสร้างแอปพื้นฐานแล้วเรียนรู้เกี่ยวกับคอน
รายละเอียดในบทต่อๆ ไป ใช้โอกาสนี้เพื่อเล่นกับแอพ as
คุณสร้างมันขึ้นมาเพื่อเรียนรู้ให้มากที่สุดเท่าที่เป็นไปได้ในกระบวนการ: อย่าลังเลที่จะ
ทำลายและแก้ไขสไตล์
และส่วนประกอบเพื่อดูว่าเกิดอะไรขึ้น
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 46

3.1 วางแอพสิ่งที่ต้องทำ

มาเริ่มสร้างแอพ todo กันเถอะ จะมีสไตล์และการใช้งานที่คล้ายคลึงกันกับ
แอปบนเว็บไซต์ TodoMVC (http://todomvc.com) รูปที่ 3.1 แสดงว่า
แอพจะเป็นยังไง
ดูเมื่อคุณทำเสร็จแล้ว เพื่อให้คุณได้แนวคิดว่าส่วนประกอบใดที่คุณต้องการและ
วิธีการจัดโครงสร้างพวกเขา ในบทที่ 1 รูปที่ 3.2 แบ่งแอปออกเป็นส่วนประกอบและ

ส่วนประกอบคอนเทนเนอร์ เรามาดูกันว่าสิ่งนี้จะมีลักษณะอย่างไรในแอพโดยใช้เครื่องมือพื้นฐาน

ส่วนประกอบของ React Native

<ดู>
<หัวข้อ />
<ป้อนข้อมูล />
<TodoList />
<ปุ่ม />
<แถบแท็บ />
</ดู>

แอพจะแสดงหัวเรื่อง การป้อนข้อความ ปุ่ม และแถบแท็บ เมื่อคุณเพิ่ม a todo แอพจะเพิ่มลงในอาร์เรย์ของ todos และแสดง todo ใหม่ใต้อินพุต แต่ละสิ่งที่ต้องทำจะมีสองปุ่ม: เสร็จสิ้น และ ลบ ปุ่มเสร็จสิ้นจะทำเครื่องหมายเป็นคอม-สมบูรณ์ และปุ่ม ลบ จะลบออกจากอาร์เรย์ของสิ่งที่ต้องทำ ที่ด้านล่างของหน้าจอ แถบแท็บจะกรองรายการสิ่งที่ต้องทำโดยพิจารณาจากว่ารายการนั้นสมบูรณ์หรือยังทำงานอยู่

รูปที่ 3.1 การออกแบบแอพ Todo

รูปที่ 3.2 แอพ Todo พร้อมคำอธิบาย

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

การเข้ารหัสแอพ todo

3.2 การเข้ารหัสแอปสิ่งที่ต้องทำ

มาเริ่มเขียนโค้ดแอปกันเลย สร้างโปรเจ็กต์ React Native ใหม่โดยพิมพ์react-
na-
tive init TodoApp ในเทอร์มินัลของคุณ (ดูรูปที่ 3.3) ตอนนี้ ไปที่ไฟล์ดัชนี
ของคุณ: if
คุณกำลังพัฒนาสำหรับ iOS เปิด index.iOS.js; และหากคุณกำลังพัฒนาสำหรับ
Android
เปิด index.Android.js รหัสสำหรับทั้งสองแพลตฟอร์มจะเหมือนกัน
หมายเหตุฉันใช้ React Native เวอร์ชัน 0.51.0 สำหรับตัวอย่างนี้ เวอร์ชั่นใหม่
กว่า
อาจมีการเปลี่ยนแปลง API แต่ไม่มีอะไรเสียหายสำหรับการสร้างสิ่งที่ต้องทำ
แอป. คุณสามารถใช้ React Native เวอร์ชันล่าสุดได้ แต่ถ้าคุณ
พบปัญหา ใช้เวอร์ชันที่ฉันใช้ที่นี่
ในไฟล์ดัชนี ให้นำเข้าส่วนประกอบแอป (ซึ่งคุณจะสร้างขึ้นเร็วๆ นี้) และลบ
จัดสไตล์พร้อมกับส่วนประกอบพิเศษที่คุณไม่ได้ใช้อีกต่อไป

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { AppRegistry } จาก 'react-native'
นำเข้าแอพจาก './app/App'
const TodoApp = () => <แอพ />
AppRegistry.registerComponent('TodoApp', () => TodoApp)
```

ที่นี่คุณจะนำมาในAppRegistryจากตอบสนองพื้นเมือง คุณนำแอพหลักเข้ามาด้วย
องค์ประกอบที่คุณจะสร้างขึ้นต่อไป
ในเมธอดAppRegistryคุณเริ่มต้นแอปพลิเคชัน AppRegistryเป็นรายการ JS

ชี้ไปที่การเรียกใช้แอป React Native ทั้งหมด ต้องใช้สองอาร์กิวเมนต์: appKey หรือชื่อ
ของแอปพลิเคชันที่คุณกำหนดเมื่อคุณเริ่มต้นแอป และฟังก์ชันที่ส่งกลับ
องค์ประกอบ React Native ที่คุณต้องการใช้เป็นจุดเริ่มต้นของแอป ในกรณีนี้,
คุณกำลังส่งคืนองค์ประกอบTodoApp ที่ประกาศในรายการ 3.2
ตอนนี้สร้างโฟลเดอร์ที่เรียกว่าแอพในรูทของแอปพลิเคชัน ในโฟลเดอร์แอพ Cre-
กินไฟล์ชื่อ App.js และเพิ่มรหัสพื้นฐานที่แสดงในรายการถัดไป
รูปที่ 3.3 การเริ่มต้นแอพ React Native ใหม่
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 48


```
นำเข้า React { ส่วนประกอบ } จาก 'react'
นำเข้า { ดู, ScrollView, StyleSheet } จาก 'react-native'
แอปคลาสขยายคอมโพเนนต์ {
เรนเดอร์ () {
กลับ (
<ดู style={styles.container}>
<ScrollView keyboardShouldPersistTaps='always'
style={styles.content}>
<ดู/>
</ScrollView>
```

</ดู>
)
}
}
รูปแบบ const = StyleSheet.create ({
คอนเทนเนอร์: {
ดิ้น: 1,
พื้นหลังสี: '#f5f5f5'
},
เนื้อหา: {
ดิ้น: 1,
paddingTop: 60
}
})
ส่งออกแอปเริ่มต้น
คุณนำเข้าองค์ประกอบใหม่ที่เรียกว่าScrollViewซึ่งปิดล้อมแพลตฟอร์มScroll-ดูและเป็นองค์ประกอบมุมมองที่เลื่อนได้โดยทั่วไป keyboardShouldPersistTaps
เพิ่ม prop เสมอ : prop นี้จะยกเลิกคีย์บอร์ดหากเปิดอยู่และอนุญาตให้ UI เพื่อประมวลผลเหตุการณ์onPress คุณต้องแน่ใจว่าทั้งScrollViewและ parent
มุมมองของ ScrollView มีดิ้น: 1 ค่า flex:1 เป็นค่าสไตล์ที่ทำให้คอมโพเนนต์จะเติมพื้นที่ทั้งหมดของคอนเทนเนอร์หลัก
ตอนนี้ ตั้งค่าสถานะเริ่มต้นสำหรับค่าบางอย่างที่คุณต้องการในภายหลัง คุณต้องมีอาร์เรย์เพื่อเก็บ todos ของคุณ ซึ่งคุณจะตั้งชื่อtodos ; ค่าที่จะถือสถานะปัจจุบันของ TextInput ที่จะเพิ่ม todos ชื่อ inputValue ; และค่าเก็บชนิดสิ่งที่ต้องทำที่คุณกำลังเข้าชม (ทั้งหมด, ปัจจุบันหรือ Active) ชื่อประเภท

ใน App.js ก่อนที่จะทำให้ฟังก์ชั่นเพิ่มคอนสตรัคและสถานะเริ่มต้นไป
คลาสและเริ่มต้นค่าเหล่านี้ในสถานะ

...
แอปคลาสขยายคอมโพเนนต์ {
ตัวสร้าง () {
ซุปเปอร์()
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 49

49
การเข้ารหัสแอพ todo
this.state = {
ค่าอินพุต: '',
สิ่งที่ต้องทำ: [],
ประเภท: 'ทั้งหมด'
}
}
เรนเดอร์ () {
...
}
}
...
ถัดไป สร้างองค์ประกอบส่วนหัวและกำหนดสไตล์ ในโฟลเดอร์แอป สร้าง
ไฟล์ชื่อ Heading.js นี่จะเป็นองค์ประกอบที่ไร้สัญชาติ

นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ, สไตล์ชีต } จาก 'react-native'
const ส่วนหัว = () => (
<ดู style={styles.header}>
<Text style={styles.headerText}>
สิ่งที่ต้องทำ
</Text>
</ดู>
)
รูปแบบ const = StyleSheet.create ({
หัวข้อ: {
ขอบด้านบน: 80
},
ส่วนหัวข้อความ: {
textAlign: 'ศูนย์',
ขนาดตัวอักษร: 72,
สี: 'rgba(175, 47, 47, 0.25)',
fontWeight: '100'
}
})
ส่งออกหัวเรื่องเริ่มต้น
ทราบว่าในจัดแต่งทรงผมของHeaderTextคุณผ่านการRGBAมูลค่าให้กับสี ถ้าคุณไม่ใช่
คุ้นเคยกับ RGBA ค่าสามค่าแรกประกอบกันเป็นค่าสี RGB และค่าสุดท้าย
ค่าแสดงถึงอัลฟาหรือความทึบ (สีแดง, สีฟ้า, สีเขียว, อัลฟา) คุณผ่านในอัลฟ่า
มูลค่า 0.25 หรือ 25% คุณยังตั้งค่าน้ำหนักแบบอักษรเป็น100ซึ่งจะทำให้ข้อความ
a
น้ำหนักทินเนอร์และดู

กลับไปที่ App.js นำองค์ประกอบHeadingมาวางไว้ในScroll-View แทนที่ View ที่ว่างเปล่าที่คุณวางไว้ในตอนแรก

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 50

เรียกใช้แอพเพื่อดูหัวเรื่องและเลย์เอาต์ของแอพใหม่: ดูรูปที่ 3.4 ในการเรียกใช้แอพใน iOS ของคุณ, การใช้งานตอบสนองพื้นเมืองวิ่ง iOS หากต้องการทำงานใน Android ให้ใช้react-native run-android
ในเทอร์มินัลของคุณจากรูทของแอปพลิเคชัน React Native

```
นำเข้า React { ส่วนประกอบ } จาก 'react'
นำเข้า {View, ScrollView, StyleSheet} จาก 'react-native'
นำเข้าหัวเรื่องจาก './Heading'
แอปคลาสขยายคอมโพเนนต์ {
...
เรนเดอร์ () {
กลับ (
<ดู style={styles.container}>
<ScrollView
keyboardShouldPersistTaps='เสมอ'
style={styles.content}>
<หัวข้อ />
</ScrollView>
```

</ดู>
)
}
}
...
รูปที่ 3.4 การรันแอพ

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 51

51

การเข้ารหัสแอพ todo

ถัดไป สร้างองค์ประกอบTextInputและกำหนดสไตล์ ในโฟลเดอร์แอป สร้างไฟล์ชื่อ Input.js

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, TextInput, StyleSheet } จาก 'react-native'
const อินพุต = () => (
<ดู style={styles.inputContainer}>
<ป้อนข้อความ
style={styles.input}
placeholder='ต้องทำอย่างไร'
ตัวยึดตำแหน่งTextColor='#CACACA'
selectionColor='#666666' />
</ดู>
)
รูปแบบ const = StyleSheet.create ({
```

คอนเทนเนอร์อินพุต: {
ระยะขอบซ้าย: 20,
ขอบขวา: 20,
เงาความทึบ: 0.2,
รัศมีเงา: 3,
shadowColor: '#000000',
shadowOffset: { กว้าง: 2, สูง: 2 }
},
ป้อนข้อมูล: {
ความสูง: 60,
พื้นหลังสี: '#ffffff',
ช่องว่างภายในด้านซ้าย: 10,
paddingRight: 10
}
})
ส่งออกอินพุตเริ่มต้น
คุณกำลังใช้องค์ประกอบ React Native ใหม่ที่เรียกว่าTextInputที่นี่ ถ้าคุณเป็นครอบครัว-
IAR กับการพัฒนาเว็บนี้เป็นคล้ายกับ HTML การป้อนข้อมูล คุณยังให้ทั้ง TextInput และด้านนอกดูสไตล์ของตัวเอง
TextInput ใช้อุปกรณ์ประกอบฉากอื่น ๆ สองสามอย่าง ที่นี่คุณระบุตัวยึดเพื่อแสดงข้อความ
ก่อนที่ผู้ใช้จะเริ่มพิมพ์placeholderTextColorที่จัดรูปแบบข้อความในที่พักและselectionColorว่ารูปแบบเคอร์เซอร์สำหรับTextInput
ขั้นตอนต่อไป ในหัวข้อ 3.4 จะเป็นการต่อฟังก์ชันเพื่อรับค่าของ TextInput และบันทึกไปยังสถานะขององค์ประกอบแอพ คุณจะเข้าไปที่ App.js และ

เพิ่มฟังก์ชันใหม่ที่เรียกว่าinputChangeด้านล่างตัวสร้างและเหนือการแสดงผล การทำงาน. ฟังก์ชันนี้จะอัปเดตค่าสถานะของinputValueด้วยค่าที่ผ่านในและตอนนี้จะออกจากระบบค่าinputValueให้คุณเพื่อให้แน่ใจว่าfunc-tion คือการทำงานโดยใช้console.log () แต่หากต้องการดูคำสั่ง console.log()ใน React Native คุณต้องเปิดเมนูนักพัฒนาซอฟต์แวร์ก่อน เรามาดูกันว่ามันทำงานอย่างไร

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

3.3 การเปิดเมนูผู้พัฒนา

เมนูสำหรับนักพัฒนาเป็นเมนูในตัวที่พร้อมใช้งานโดยเป็นส่วนหนึ่งของ React Native; มันให้คุณ
เข้าถึงเครื่องมือแก้ไขข้อบกพร่องหลักที่คุณจะใช้ คุณสามารถเปิดได้ในโปรแกรมจำลอง iOS หรือ
ในโปรแกรมจำลอง Android ในส่วนนี้ ฉันจะแสดงวิธีเปิดและใช้งานการพัฒนาเมนูoper บนทั้งสองแพลตฟอร์ม

หมายเหตุหากคุณไม่สนใจเมนูนักพัฒนาหรือต้องการข้ามวินาทีนี้
สำหรับตอนนี้ ไปที่ส่วน 3.4 เพื่อสร้างแอป todo ต่อ

3.3.1 การเปิดเมนูนักพัฒนาในโปรแกรมจำลอง iOS

ในขณะที่โปรเจ็กต์กำลังทำงานอยู่ในตัวจำลอง iOS คุณสามารถเปิดเมนูนักพัฒนาใน
หนึ่งในสามวิธี:

- กด Cmd-D บนแป้นพิมพ์
- กด Cmd-Ctrl-Z บนแป้นพิมพ์
- เปิดเมนู Hardware > Shake Gesture ในตัวเลือกเครื่องจำลอง (ดู
รูปที่ 3.5)

เมื่อคุณทำ คุณควรเห็นเมนูนักพัฒนาที่แสดงในรูปที่ 3.6

หมายเหตุหาก Cmd-D หรือ Cmd-Ctrl-Z ไม่เปิดเมนู คุณอาจต้องแก้ไข
เชื่อมต่อฮาร์ดแวร์ของคุณกับแป้นพิมพ์ ในการดำเนินการนี้ ให้ไปที่ฮาร์ดแวร์ >
คีย์บอร์ด >

เชื่อมต่อฮาร์ดแวร์คีย์บอร์ดในเมนูจำลองของคุณ

รูปที่ 3.5 การเปิดผู้พัฒนาด้วยตนเอง
เมนู (โปรแกรมจำลอง iOS)

รูปที่ 3.6 React Native Developer menu
(เครื่องจำลอง iOS)

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

3.3.2 การเปิดเมนูนักพัฒนาในโปรแกรมจำลอง Android

เมื่อโปรเจ็กต์เปิดและทำงานในอีมูเลเตอร์ Android เมนูนักพัฒนาสามารถ

เปิดด้วยวิธีใดวิธีหนึ่งจากสามวิธี:

- กด F2 บนแป้นพิมพ์
- กด Cmd-M บนแป้นพิมพ์
- กดปุ่มฮาร์ดแวร์ (ดูรูปที่ 3.7)

เมื่อคุณทำ คุณควรเห็นเมนูนักพัฒนาที่แสดงในรูปที่ 3.8

3.3.3 การใช้เมนูนักพัฒนา

เมื่อเมนูนักพัฒนาเปิดขึ้น คุณจะเห็นตัวเลือกต่อไปนี้:

- โหลดซ้ำ (iOS และ Android) — โหลดแอปซ้ำ สามารถทำได้โดยการกด Cmd-R บนแป้นพิมพ์ (iOS) หรือกด R สองครั้ง (Android)
- ดีบัก JS จากระยะไกล (iOS และ Android) —เปิดเครื่องมือสำหรับนักพัฒนาของ Chrome และให้คุณ

รองรับการดีบักเต็มรูปแบบผ่านเบราว์เซอร์ (รูปที่ 3.9) ที่นี่คุณสามารถเข้าถึงได้ไม่เพียงแต่การบันทึกคำสั่งในโค้ดของคุณเท่านั้น แต่ยังรวมถึงเบรกพอยต์และอะไรก็ตามคุณคุ้นเคยกับการดีบักเว็บแอป (ยกเว้น DOM) ถ้าคุณจำเป็นต้องบันทึกข้อมูลใด ๆ ในแอปของคุณ ซึ่งมักจะเป็นที่ที่ต้องทำ

เมนูฮาร์ดแวร์

รูปที่ 3.7 การเปิดฮาร์ดแวร์ด้วยตนเอง

เมนู (โปรแกรมจำลอง Android)

รูปที่ 3.8 React Native Developer menu

(โปรแกรมจำลองแอนดรอยด์)

▪ เปิดใช้งาน Live Reload (iOS และ Android)—เปิดใช้งานการรีโหลดแบบสด เมื่อคุณทำ
การเปลี่ยนแปลงโค้ดของคุณ แอปทั้งหมดจะโหลดซ้ำและรีเฟรชในโปรแกรมจำลอง
▪ เริ่ม Systrace (iOS เท่านั้น)—Systrace เป็นเครื่องมือสร้างโปรไฟล์ สิ่งนี้จะทำให้คุณมีความคิดที่ดี
เวลาที่คุณใช้ไปในแต่ละเฟรม 16 ms ในขณะที่แอปของคุณทำงาน
หนึง บล็อกรหัสที่ทำโปรไฟล์ล้อมรอบด้วยเครื่องหมายเริ่มต้น/สิ้นสุดซึ่งต่อมาคือ
แสดงในรูปแบบแผนภูมิที่มีสีสัน ยังสามารถเปิดใช้งาน Systrace ด้วยตนเองจาก
บรรทัดคำสั่งใน Android หากคุณต้องการเรียนรู้เพิ่มเติม โปรดดูเอกสารสำหรับ a
ภาพรวมที่ครอบคลุมมาก
▪ เปิดใช้งาน Hot Reloading (iOS และ Android)—เพิ่มฟีเจอร์ที่ยอดเยี่ยมในเวอร์ชัน .22 ของ
ตอบโต้พื้นเมือง มันมอบประสบการณ์นักพัฒนาที่น่าทึ่ง ให้คุณมีความสามารถ
เพื่อดูการเปลี่ยนแปลงของคุณทันทีเมื่อไฟล์ถูกเปลี่ยนโดยไม่สูญเสียปัจจุบัน
สถานะของแอป สิ่งนี้มีประโยชน์อย่างยิ่งสำหรับการเปลี่ยนแปลง UI ลึกในแอปของคุณ
โดยไม่สูญเสียสถานะ ต่างจากการรีโหลดแบบสดเพราะเก็บกระแสไฟไว้
สถานะของแอปของคุณ อัปเดตเฉพาะส่วนประกอบและสถานะที่มีการเปลี่ยนแปลง
(การรีโหลดแบบสดจะรีโหลดแอปทั้งหมด ดังนั้นจึงสูญเสียสถานะปัจจุบัน)

▪ Toggle Inspector (iOS และ Android)—เรียกตัวตรวจสอบคุณสมบัติที่คล้ายกับสิ่งที่
ที่คุณเห็นในเครื่องมือสำหรับนักพัฒนาของ Chrome คุณสามารถคลิกองค์ประกอบและดูว่ามันอยู่ที่ไหนใน
ลำดับชั้นของส่วนประกอบ ตลอดจนรูปแบบใดๆ ที่ใช้กับองค์ประกอบ (รูปที่ 3.10)
▪ แสดง Perf Monitor (iOS และ Android)—เปิดกล่องเล็กๆ ที่มุมซ้ายบน-ไม่ใช่แอป โดยให้ข้อมูลบางอย่างเกี่ยวกับประสิทธิภาพของแอป ที่นี่
คุณจะเห็นจำนวน RAM ที่ใช้และจำนวนเฟรมต่อวินาที
ที่แอปกำลังทำงานอยู่ หากคุณคลิกที่ช่อง มันจะขยายเพื่อแสดง
ข้อมูลเพิ่มเติม (รูปที่ 3.11)
▪ การตั้งค่า Dev (โปรแกรมจำลอง Android เท่านั้น)—นำเสนอตัวเลือกการดีบักเพิ่มเติม
รวมถึงวิธีง่ายๆ ในการสลับระหว่างตัวแปรสภาพแวดล้อม__DEV__เป็น
จริงหรือเท็จ(รูปที่ 3.12)

รูปที่ 3.9 การดีบักใน Chrome

รูปที่ 3.10 การใช้ตัวตรวจสอบ (ซ้าย: iOS ขวา: Android)

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 55

3.4 กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

เมื่อคุณทราบวิธีการทำงานของเมนูนักพัฒนาแล้ว ให้เปิดและกด Debug JS จากระยะไกลเพื่อเปิดเครื่องมือ Chrome dev คุณพร้อมที่จะเริ่มบันทึกข้อมูลไปที่คอนโซล JavaScript

คุณจะต้องนำเข้าองค์ประกอบอินพุตไปยัง app/App.js และแนบวิธีการไปที่ TextInput ที่คุณจะให้เป็นเสาจะมีการป้อนข้อมูล คุณจะผ่าน inputValue เก็บไว้ในสถานะอินพุตเป็นอุปกรณ์ประกอบฉาก

...
นำเข้าหัวเรื่องจาก './Heading'
รูปที่ 3.11 Perf Monitor (ซ้าย: iOS, ขวา: Android)
รูปที่ 3.12 การตั้งค่า Dev (โปรแกรมจำลอง Android)
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 56


```
นำเข้าอินพุตจาก './Input'
แอปคลาสขยายคอมโพเนนต์ {
ตัวสร้าง () {
…
}
inputChange (ค่าอินพุต) {
console.log(' ค่าที่ป้อน: ' , inputValue)
this.setState ({ inputValue })
}
```

```
เรนเดอร์ () {
const { inputValue } = this.state
กลับ (
<ดู style={styles.container}>
<ScrollView
keyboardShouldPersistTaps='เสมอ'
style={styles.content}>
<หัวข้อ />
<อินพุต
inputValue={inputValue}
inputChange={(text) => this.inputChange(text)} />
</ScrollView>
</ดู>
)
}}
```

inputChange รับหนึ่งอาร์กิวเมนต์ ค่าของ TextInput และอัพเดตอินพุต-ราคาในรัฐที่มีค่ากลับมาจากTextInput

ตอนนี้ คุณต้องเชื่อมต่อฟังก์ชันกับTextInputในองค์ประกอบอินพุต

เปิด app/Input.js และอัปเดตคอมโพเนนต์TextInputด้วยinputChange . ใหม่

ฟังก์ชันและคุณสมบัติinputValue

```
...
const อินพุต = ({ inputValue, inputChange }) => (
<ดู style={styles.inputContainer}>
<ป้อนข้อความ
ค่า={inputValue}
style={styles.input}
placeholder='ต้องทำอย่างไร'
```

ตัวยึดตำแหน่งTextColor='#CACACA'
SelectionColor='#666666'
onChangeText={inputChange} />
</ดู>
)
...
คุณทำลายอุปกรณ์ประกอบฉากinputValueและinputChangeในการสร้างคน
ไร้สัญชาติ
ส่วนประกอบ. เมื่อค่าของTextInputเปลี่ยนไปฟังก์ชันinputChangeจะเป็น
เรียกว่าคุ้มค่าและถูกส่งไปยังส่วนของผู้ปกครองในการตั้งค่าสถานะของค่าอินพุต
คุณยังตั้งค่าของTextInputให้เป็นinputValueเพื่อให้คุณสามารถควบคุมและ .
ได้ในภายหลัง
รีเซ็ตTextInput onChangeTextเป็นเมธอดที่จะเรียกทุกครั้งที่มีค่า
ของTextInputองค์ประกอบที่มีการเปลี่ยนแปลงและจะถูกส่งผ่านค่าของ
TextInput
สร้างเมธอด inputChange ซึ่ง
รับ inputValue เป็นอาร์กิวเมนต์
ออกจากระบบค่า inputValue เป็น
ตรวจสอบให้แน่ใจว่าวิธีการทำงาน
ตั้งค่าสถานะด้วยค่าใหม่—เหมือนกับ
this.setState({inputValue: inputValue})
ส่งผ่าน inputValue เป็นคุณสมบัติ
ไปยังองค์ประกอบอินพุต
ส่งผ่าน inputChange เป็นคุณสมบัติ
ไปยังองค์ประกอบอินพุต

ทำลาย inputValue
และอินพุตเปลี่ยนอุปกรณ์ประกอบฉาก
ตั้งค่า onChangeText
วิธีการป้อนข้อมูลChange

หน้า 57

เรียกใช้โครงการอีกครั้งและดูว่าเป็นอย่างไร (รูปที่ 3.13) คุณกำลังบันทึกค่า
ของอินพุต ดังนั้นเมื่อคุณพิมพ์ คุณควรเห็นค่าที่ออกจากระบบคอนโซล
(รูปที่ 3.14)
รูปที่ 3.13 อัปเดตมุมมองหลังจากเพิ่ม TextInput
รูปที่ 3.14 ออกจากระบบค่า TextInput ด้วยวิธี inputChange

หน้า 58

ตอนนี้ค่าของinputValueถูกเก็บไว้ในสถานะแล้ว คุณต้องสร้าง a

ปุ่มเพื่อเพิ่มรายการไปยังรายการสิ่งที่ต้องทำ ก่อนที่คุณจะทำ ให้สร้างฟังก์ชันที่คุณจะผูกกับปุ่มเพื่อเพิ่มสิ่งที่ต้องทำใหม่ให้กับอาร์เรย์ของสิ่งที่ต้องทำที่กำหนดไว้ในตัวสร้าง เรียกใช้ฟังก์ชันนี้sendTodoและวางไว้หลังฟังก์ชันinputChangeและ before ทำให้ฟังก์ชั่น

```
...
ส่งสิ่งที่ต้องทำ () {
if (this.state.inputValue.match(/^\s*$/)) {
กลับ
}
สิ่งที่ต้องทำ = {
ชื่อเรื่อง: this.state.inputValue,
ดัชนีสิ่งที่ต้องทำ
สมบูรณ์: เท็จ
}
todoIndex++
const todos = [...this.state.todos, สิ่งที่ต้องทำ]
this.setState ({ todos, inputValue: " }, () => {
console.log('State: ', this.state)
})
}
...
```

ถัดไป สร้างtodoIndexที่ด้านบนของไฟล์ App.js ใต้คำสั่งนำเข้าล่าสุด

```
...
นำเข้าอินพุตจาก './Input'
ให้ todoIndex = 0
แอปคลาสขยายคอมโพเนนต์ {
...
```

เมื่อสร้างฟังก์ชันsubmitTodoแล้ว ให้สร้างไฟล์ชื่อ Button.js และ
ต่อสายฟังก์ชันเพื่อทำงานกับปุ่ม

นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ, สไตล์ชีต, TouchableHighlight } จาก 'react-native'
ตรวจสอบว่า inputValue ว่างเปล่าหรือ
มีเพียงช่องว่าง ถ้าว่าง
กลับมาโดยไม่ทำอะไรเลย
หาก inputValue ไม่ว่างเปล่า ให้สร้างและกำหนด todo
ตัวแปรอ็อบเจ็กต์ที่มีชื่อ, todoIndex และ a
บูลีนที่สมบูรณ์ (คุณจะสร้าง todoIndex ในไม่ช้า)
เพิ่ม todoIndex
ผลักดันสิ่งที่ต้องทำใหม่ไปที่
อาร์เรย์ที่มีอยู่ของ todos
ตั้งค่าสถานะของสิ่งที่ต้องทำให้ตรงกับ
อาร์เรย์ที่อัปเดตของ this.state.todos และ
รีเซ็ต inputValue เป็นสตริงว่าง
เมื่อตั้งค่าสถานะแล้ว คุณจะมีตัวเลือกที่จะผ่าน a
ฟังก์ชั่นโทรกลับ ที่นี่ ฟังก์ชันเรียกกลับจาก setState
ออกจากระบบเพื่อให้แน่ใจว่าทุกอย่างทำงาน
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

59

กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

ปุ่ม const = ({ submitTodo }) => (

<ดู style={styles.buttonContainer}>

<ไฮไลท์ที่สัมผัสได้

underlayColor='#efefef'

style={styles.button}

onPress={submitTodo}>

<รูปแบบข้อความ={styles.submit}>

ส่ง

</Text>

</TouchableHighlight>

</ดู>

)

รูปแบบ const = StyleSheet.create ({

คอนเทนเนอร์ปุ่ม: {

alignItems: 'flex-end'

},

ปุ่ม: {

ความสูง: 50,

ช่องว่างภายในด้านซ้าย: 20,

paddingขวา: 20,

พื้นหลังสี: '#ffffff',

ความกว้าง: 200,

ขอบขวา: 20,

ขอบบน: 15,

ขอบกว้าง: 1,

borderColor: 'rgba(0,0,0,.1)',

justifyContent: 'ศูนย์',

alignItems: 'ศูนย์'
},
ส่ง: {
สี: '#666666',
น้ำหนักแบบอักษร: '600'
}
})
ส่งออกปุ่มเริ่มต้น
ในองค์ประกอบนี้ คุณใช้TouchableHighlightเป็นครั้งแรก สัมผัสได้สูง-
แสงเป็นวิธีหนึ่งในการสร้างปุ่มใน React Native และเป็นพื้นฐาน
เทียบได้กับองค์ประกอบปุ่ม HTML
ด้วยTouchableHighlightคุณสามารถตัดมุมมองและทำให้ตอบสนองอย่าง
เหมาะสม
สัมผัสเหตุการณ์ เมื่อกดลงbackgroundColorเริ่มต้นจะถูกแทนที่ด้วยที่ระบุ
underlayColor คุณสมบัติที่คุณจะให้เป็นพร็อพ ที่นี่คุณระบุ underlay-
สีของ '#efefef' ซึ่งเป็นสีเทาอ่อน สีพื้นหลังเป็นสีขาว สิ่งนี้จะให้
ผู้ใช้รู้สึกดีว่าเหตุการณ์สัมผัสได้ลงทะเบียนแล้วหรือไม่ หากไม่
มีunderlayColor is
กำหนดไว้โดยค่าเริ่มต้นเป็นสีดำ
TouchableHighlight รองรับองค์ประกอบลูกหลักเพียงองค์ประกอบเดียว ที่นี่
คุณผ่านใน
ข้อความส่วนประกอบ หากคุณต้องการหลายองค์ประกอบใน
TouchableHighlightให้ห่อ
พวกเขาในครั้งเดียวดูและผ่านนี้ดูเป็นเด็กของTouchableHighlight
ทำลายฟังก์ชัน submitTodo ซึ่ง

ถูกส่งผ่านเป็นพร็อพไปยังส่วนประกอบ
แนบ sendTodo กับฟังก์ชัน onPress ที่มีให้
คอมโพเนนต์ TouchableHighlight ฟังก์ชันนี้จะเป็น
เรียกเมื่อแตะหรือกด TouchableHighlight
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 60

หมายเหตุนอกจากนี้ยังมีการจัดรูปแบบเล็กน้อยในรายการ 3.12 ไม่ต้องกังวล
เกี่ยวกับสไตล์เฉพาะในบทนี้: เราครอบคลุมรายละเอียดเหล่านี้ในบทที่ 4
และ 5. แต่จงมองดูพวกเขา เพื่อให้เข้าใจว่าการจัดสไตล์ทำงานอย่างไรในแต่ละ
องค์ประกอบ-
เน็ท สิ่งนี้จะช่วยได้มากในเชิงลึกในบทต่อๆ ไป เพราะคุณจะได้อยู่แล้ว
ได้สัมผัสกับคุณสมบัติการจัดแต่งทรงผมและวิธีการทำงาน
คุณได้สร้างองค์ประกอบปุ่มและเชื่อมต่อกับฟังก์ชันที่กำหนดไว้ใน
แอพ.js ตอนนี้นำส่วนประกอบนี้ไปยังแอพ (app/App.js) และดูว่าใช้งานได้
หรือไม่!

```
...
ปุ่มนำเข้าจาก './ปุ่ม'
ให้ todoIndex = 0
...
```

```
ตัวสร้าง () {
ซุปเปอร์()
this.state = {
ค่าอินพุต: '',
สิ่งที่ต้องทำ: [],
ประเภท: 'ทั้งหมด'
}
this.submitTodo = this.submitTodo.bind(นี่)
}
...
แสดงผล () {
ให้ { inputValue } = this.state
กลับ (
<ดู style={styles.container}>
<ScrollView
keyboardShouldPersistTaps='เสมอ'
style={styles.content}>
<หัวข้อ />
<อินพุต
inputValue={inputValue}
inputChange={(text) => this.inputChange(text)} />
<ปุ่ม submitTodo={this.submitTodo} />
</ScrollView>
</ดู>
)
}
```

คุณนำเข้าส่วนประกอบปุ่มและวางไว้ใต้ส่วนประกอบอินพุตใน ฟังก์ชั่นการแสดงผล submitTodo ถูกส่งผ่านไปยังปุ่มในฐานะคุณสมบัติที่เรียกว่า สิ่งนี้ ส่งสิ่งที่ต้องทำ

ตอนนี้ รีเฟรชแอป ควรมีลักษณะเหมือนรูปที่ 3.15 เมื่อคุณเพิ่มสิ่งที่ต้องทำ TextInput ควรล้าง และสถานะของแอปควรเข้าสู่ระบบคอนโซล โดยแสดงอาร์เรย์ของ todos กับ todo ใหม่ในอาร์เรย์ (รูปที่ 3.16)

นำเข้าส่วนประกอบปุ่มใหม่

ผูกเมธอดกับคลาสใน

ตัวสร้าง เพราะคุณคือ

การใช้คลาส ฟังก์ชันจะไม่เป็น

ผูกอัตโนมัติกับชั้นเรียน

วางปุ่มด้านล่าง

ส่วนประกอบอินพุตและผ่านใน

ส่งTodo เป็นพร็อพ

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 61

กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

ตอนนี้คุณกำลังเพิ่ม todos ลงในอาร์เรย์ของ todos คุณต้องสร้าง todos ลงในหน้าจอ. ในการเริ่มต้นใช้งาน คุณต้องสร้างสององค์ประกอบใหม่: TodoListและสิ่งที่ต้องทำ ToDoList จะแสดงรายชื่อของ Todos และจะใช้สิ่งที่ต้องทำส่วนประกอบสำหรับแต่ละ

สิ่งที่ต้องทำส่วนบุคคล เริ่มต้นด้วยการสร้างไฟล์ชื่อ Todo.js ในโฟลเดอร์แอป

รูปที่ 3.15 อัพเดทแอพด้วยปุ่ม

ส่วนประกอบ

รูปที่ 3.16 การบันทึกสถานะ

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 62


```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู, ข้อความ, สไตล์ชีต } จาก 'react-native'
const สิ่งที่ต้องทำ = ({ สิ่งที่ต้องทำ }) => (
<ดู style={styles.todoContainer}>
<Text style={styles.todoText}>
{todo.title}
</Text>
</ดู>
)
รูปแบบ const = StyleSheet.create ({
todoContainer: {
ระยะขอบซ้าย: 20,
ขอบขวา: 20,
พื้นหลังสี: '#ffffff',
ขอบด้านบนกว้าง: 1,
borderRightWidth: 1,
borderLeftWidth: 1,
borderColor: '#ededed',
ช่องว่างภายในด้านซ้าย: 14,
```

paddingTop: 7,
paddingด้านล่าง: 7,
เงาความทึบ: 0.2,
รัศมีเงา: 3,
shadowColor: '#000000',
shadowOffset: { ความกว้าง: 2, ความสูง: 2 },
flexDirection: 'แถว',
alignItems: 'ศูนย์'
},
สิ่งที่ต้องทำข้อความ: {
ขนาดตัวอักษร: 17
}
})
ส่งออก Todo เริ่มต้น
Todoองค์ประกอบที่จะใช้เวลาหนึ่งคุณสมบัติสำหรับตอนนี้มีสิ่งที่ต้องทำและทำให้ชื่อในส่วน
องค์ประกอบข้อความ คุณยังเพิ่มสไตล์ให้กับคอมโพเนนต์มุมมองและข้อความ
ถัดไป สร้างคอมโพเนนต์TodoList (app/TodoList.js)

นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู } จาก 'react-native'
นำเข้าสิ่งที่ต้องทำจาก './Todo'
const TodoList = ({ สิ่งที่ต้องทำ }) => {
todos = todos.map ((สิ่งที่ต้องทำ, i) => {
กลับ (
<สิ่งที่ต้องทำ
คีย์={todo.todoIndex}
สิ่งที่ต้องทำ={todo} />

หน้า 63

กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

```
)
})
กลับ (
<ดู>
{สิ่งที่ต้องทำ}
</ดู>
)
}
```

ส่งออก TodoList เริ่มต้น

ToDoListองค์ประกอบที่จะใช้เวลาหนึ่งคุณสมบัติสำหรับขณะนี้: อาร์เรย์ของ to-do จากนั้นคุณ map ทับ todos เหล่านี้และสร้างส่วนประกอบTodoใหม่(นำเข้าที่ด้านบนของไฟล์) สำหรับแต่ละสิ่งที่ต้องทำ การส่งสิ่งที่ต้องทำเป็นคุณสมบัติไปยังองค์ประกอบสิ่งที่ต้องทำ คุณยังระบุ-ify คีย์และส่งผ่านดัชนีของรายการสิ่งที่ต้องทำเป็นคีย์ไปยังแต่ละส่วนประกอบ สำคัญคุณสมบัติช่วย React ระบุรายการที่มีการเปลี่ยนแปลงเมื่อส่วนต่างกับ virtual DOM ถูกคำนวณ React จะเตือนคุณหากคุณไม่ดำเนินการนี้

สิ่งสุดท้ายที่คุณต้องทำคือนำเข้าส่วนประกอบTodoListลงในไฟล์ App.js และส่งผ่านใน todos เป็นพร็อพเพอร์ตี้

```
...
นำเข้า TodoList จาก './TodoList'
...
แสดงผล () {
const { inputValue, todos } = this.state
กลับ (
<ดู style={styles.container}>
<ScrollView
keyboardShouldPersistTaps='เสมอ'
style={styles.content}>
<หัวข้อ />
<อินพุต inputValue={inputValue} inputChange={(text) => this.
inputChange(ข้อความ)} />
<TodoList todos={todos} />
<ปุ่ม submitTodo={this.submitTodo} />
</ScrollView>
</ดู>
)
}
...
```

เรียกใช้แอป เมื่อคุณเพิ่มสิ่งที่ต้องทำ คุณจะเห็นป๊อปอัปในรายการสิ่งที่ต้องทำ (รูปที่ 3.17)

ขั้นตอนต่อไปคือการทำเครื่องหมายสิ่งที่ต้องทำว่าเสร็จสิ้น และการลบสิ่งที่ต้องทำ เปิด App.js และสร้างฟังก์ชัน toggleCompleteและdeleteTodoด้านล่างฟังก์ชัน sendTodo

toggleComplete จะสลับว่าสิ่งที่ต้องทำจะเสร็จสมบูรณ์หรือไม่ และ deleteTodo จะลบ

สิ่งที่ต้องทำ

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 64


```
ตัวสร้าง () {
...
this.toggleComplete = this.toggleComplete.bind (นี่)
this.deleteTodo = this.deleteTodo.bind(นี่)
}
...
deleteTodo (todoIndex) {
ให้ { todos } = this.state
todos = todos.filter((todo) => todo.todoIndex !== todoIndex)
this.setState({ สิ่งที่ต้องทำ })
}
```

รูปที่ 3.17 อัพเดทแอพด้วย
ส่วนประกอบ TodoList
ผูกวิธี toggleComplete
ถึงคลาสในตัวสร้าง
ผูกเมธอด deleteTodo กับ
คลาสในตัวสร้าง
deleteTodo ใช้ todoIndex

เป็นอาร์กิวเมนต์ กรอง todos
เพื่อส่งคืนทั้งหมดยกเว้นสิ่งที่ต้องทำด้วย
ดัชนีที่ผ่านเข้ามา
แล้วรีเซ็ตสถานะเป็น
สิ่งที่ต้องทำที่เหลืออยู่

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 65

65
กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

```
toggleComplete (todoIndex) {
ให้ todos = this.state.todos
todos.forEach((สิ่งที่ต้องทำ) => {
ถ้า (todo.todoIndex === todoIndex) {
todo.complete = !todo.complete
}
})
this.setState({ สิ่งที่ต้องทำ })
}
...
```

ในการขอใช้ฟังก์ชันเหล่านี้ คุณต้องสร้างส่วนประกอบปุ่มเพื่อส่งผ่านไปยัง
ทำ. ในโฟลเดอร์แอป ให้สร้างไฟล์ใหม่ชื่อ TodoButton.js

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { Text, TouchableHighlight, StyleSheet } จาก 'react-native'
```

```
const TodoButton = ({ onPress กรอกชื่อ }) => (
<ไฮไลท์ที่สัมผัสได้
onPress={onPress}
underlayColor='#efefef'
style={styles.button}>
<รูปแบบข้อความ={[
สไตล์.ข้อความ,
เสร็จสิ้น ? style.complete : null,
ชื่อ === 'ลบ' ? style.deleteButton : null ]}
>
{ชื่อ}
</Text>
</TouchableHighlight>
)
รูปแบบ const = StyleSheet.create ({
ปุ่ม: {
alignSelf: 'ปลายโค้งงอ',
ช่องว่างภายใน: 7,
borderColor: '#ededed',
ขอบกว้าง: 1,
รัศมีชายแดน: 4,
ขอบขวา: 5
},
ข้อความ: {
สี: '#666666'
},
เสร็จสิ้น: {
สี: 'เขียว',
fontWeight: 'ตัวหนา'
},
ลบปุ่ม: {
```

สี: 'rgba(175, 47, 47, 1)'
}
})
ส่งออก TodoButtton เริ่มต้น
toggleComplete ยังใช้ todoIndex เป็น
อาร์กิวเมนต์ และวนซ้ำผ่าน todos
จนกว่าจะพบสิ่งที่ต้องทำกับดัชนีที่กำหนด
มันเปลี่ยนบูลีนที่สมบูรณ์เป็น
ตรงข้ามกับการตั้งค่าปัจจุบันที่สมบูรณ์และ
แล้วรีเซ็ตสถานะของสิ่งที่ต้องทำ
ใช้เวลากด, เสร็จสมบูรณ์,
และตั้งชื่อเป็นอุปกรณ์ประกอบฉาก
ตรวจสอบว่าเสร็จสมบูรณ์ is
จริงและใช้สไตล์
ตรวจสอบว่าชื่อคุณสมบัติ
เท่ากับ “ลบ” และถ้าใช่ ให้ใช้สไตล์
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 66

ตอนนี้ ส่งต่อฟังก์ชันใหม่เป็นอุปกรณ์ประกอบฉากไปยังคอมโพเนนต์TodoList

```
แสดงผล () {
...
<TodoList
toggleComplete={this.toggleComplete}
deleteTodo={this.deleteTodo}
todos={todos} />
<ปุ่ม submitTodo={this.submitTodo} />
...
}
```

ถัดไป ส่งtoggleCompleteและdeleteTodoเป็นอุปกรณ์ประกอบฉากไปยังคอมโพเนนต์Todo

```
...
const TodoList = ({ สิ่งที่ต้องทำ, deleteTodo, toggleComplete }) => {
todos = todos.map ((สิ่งที่ต้องทำ, i) => {
กลับ (
<สิ่งที่ต้องทำ
deleteTodo={deleteTodo}
toggleComplete={toggleComplete}
คีย์={i}
สิ่งที่ต้องทำ={todo} />
)
})
...
```

สุดท้าย เปิด Todo.js และอัปเดตองค์ประกอบTodoเพื่อนำTodoButtonใหม่เข้ามา

ส่วนประกอบและการจัดรูปแบบบางอย่างสำหรับที่เก็บปุ่ม

นำเข้า TodoButton จาก './TodoButton'
...
const Todo = ({ สิ่งที่ต้องทำ, toggleComplete, deleteTodo }) => (
<ดู style={styles.todoContainer}>
<Text style={styles.todoText}>
{todo.title}
</Text>
<ดู style={styles.buttons}>
<TodoButton
ชื่อ='เสร็จสิ้น'
สมบูรณ์={todo.complete}
onPress={() => toggleComplete(todo.todoIndex)} />
<TodoButton
ชื่อ='ลบ'
onPress={() => deleteTodo(todo.todoIndex)} />
</ดู>
</ดู>
)

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 67

กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

รูปแบบ const = StyleSheet.create ({
...
ปุ่ม: {
ดิ้น: 1,
flexDirection: 'แถว',
justifyContent: 'flex-end',
alignItems: 'ศูนย์'
},
...
)}
คุณเพิ่มTodoButtonสองรายการ : อันหนึ่งชื่อ Done และอีกอันหนึ่งชื่อ Delete คุณยังผ่าน
toggleComplete และ deleteTodo เป็นฟังก์ชันที่จะเรียกว่าเป็น onPress ที่คุณกำหนด
ใน TodoButton.js หากคุณรีเฟรชแอปและเพิ่มสิ่งที่ต้องทำ ตอนนี้คุณจะเห็นใหม่ปุ่ม (รูปที่ 3.18)
หากคุณคลิก เสร็จสิ้น ข้อความของปุ่มควรเป็นตัวหนาและเป็นสีเขียว หากคุณคลิกลบเครื่องหมาย
สิ่งที่ต้องทำควรหายไปจากรายการสิ่งที่ต้องทำ
ตอนนี้คุณใช้งานแอปเกือบเสร็จแล้ว ขั้นตอนสุดท้ายคือการสร้างตัวกรองแถบแท็บที่จะแสดงสิ่งที่ต้องทำทั้งหมด เฉพาะ todos ที่สมบูรณ์ หรือเฉพาะ todos ที่ไม่สมบูรณ์ ถึง
เริ่มต้นใช้งาน คุณจะสร้างฟังก์ชันใหม่ที่จะกำหนดประเภทของสิ่งที่ต้องทำที่จะแสดง
รูปที่ 3.18 แอพที่มี TodoButtons
แสดง
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc

หน้า 68

ในตัวสร้าง คุณตั้งค่าตัวแปรประเภทสถานะเป็น'ทั้งหมด'เมื่อคุณสร้างแอปครั้งแรก ตอนนี้คุณจะสร้างฟังก์ชันชื่อsetTypeที่จะใช้ประเภทเป็นอาร์กิวเมนต์และอัปเดตประเภทในสถานะ วางฟังก์ชันนี้ไว้ใต้ปุ่มสลับฟังก์ชันที่สมบูรณ์ในแอพ.js

```
ตัวสร้าง () {
...
this.setType = this.setType.bind (นี่)
}
...
setType (ประเภท) {
this.setState ({ ประเภท })
}
...
```

ถัดไปคุณต้องสร้างTabBarและTabBarItemส่วนประกอบ ขั้นแรกให้สร้างคอมโพเนนต์ TabBar : เพิ่มไฟล์ในโฟลเดอร์แอปชื่อ TabBar.js

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { ดู StyleSheet } จาก 'react-native'
นำเข้า TabBarItem จาก './TabBarItem'
const TabBar = ({ setType ประเภท }) => (
<ดู style={styles.container}>
```

<TabBarItem type={type} title='ทั้งหมด'
setType={() => setType('All')} />
<TabBarItem type={type} border title='ใช้งานอยู่'
setType={() => setType('Active')} />
<TabBarItem type={type} border title='สมบูรณ์'
setType={() => setType('Complete')} />
</ดู>
)
รูปแบบ const = StyleSheet.create ({
คอนเทนเนอร์: {
ความสูง: 70,
flexDirection: 'แถว',
ขอบด้านบนกว้าง: 1,
borderTopColor: '#dddddd'
}
})
ส่งออก TabBar เริ่มต้น
ส่วนนี้จะใช้เวลาสองอุปกรณ์ประกอบฉาก: setTypeและประเภท ทั้งสองสืบทอดมาจาก
ส่วนประกอบหลักของแอพ
ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc
facebook.com/somsacki

หน้า 69

กำลังสร้างแอปสิ่งที่ต้องทำต่อไป

คุณกำลังนำเข้าองค์ประกอบTabBarItem ที่ยังไม่ได้กำหนด TabBarItemแต่ละอัน
องค์ประกอบที่ใช้เวลาสามอุปกรณ์ประกอบฉาก: ชื่อ , ประเภทและsetType สององค์ประกอบด้วย
ใช้เส้นขอบ prop (บูลีน) ซึ่งหากตั้งค่าจะเพิ่มสไตล์เส้นขอบด้านซ้าย
จากนั้น สร้างไฟล์ในโฟลเดอร์แอปชื่อ TabBarItem.js

```
นำเข้าปฏิกิริยาจาก 'ปฏิกิริยา'
นำเข้า { Text, TouchableHighlight, StyleSheet } จาก 'react-native'
const TabBarItem = ({ border, title, selected, setType, type }) => (
<ไฮไลท์ที่สัมผัสได้
underlayColor='#efefef'
onPress={setType}
สไตล์={[
style.item เลือก ? style.selected : null,
ชายแดน ? style.border : null,
พิมพ์ === ชื่อเรื่อง ? style.selected : null ]}>
<Text style={[ styles.itemText, type === title ? styles.bold : null ]}>
{ชื่อ}
</Text>
</TouchableHighlight>
)
รูปแบบ const = StyleSheet.create ({
รายการ: {
ดิ้น: 1,
justifyContent: 'ศูนย์',
alignItems: 'ศูนย์'
},
ชายแดน: {
```

```
borderLeftWidth: 1,
borderLeftColor: '#dddddd'
},
รายการข้อความ: {
สี: '#777777',
ขนาดตัวอักษร: 16
},
เลือก: {
พื้นหลังสี: '#ffffff'
},
ตัวหนา: {
fontWeight: 'ตัวหนา'
}
})
```

ส่งออกค่าเริ่มต้น TabBarItem

ในองค์ประกอบTouchableHighlightคุณตรวจสอบอุปกรณ์ประกอบฉากสองสามชิ้นและกำหนดสไตล์ตาม

เสา ถ้าเลือกเป็นจริงคุณจะให้มันสไตล์styles.selected ถ้าborderเป็นจริงคุณให้มันสไตล์styles.border ถ้าประเภทเท่ากับชื่อคุณให้styles.selected

ในองค์ประกอบข้อความคุณยังตรวจสอบเพื่อดูว่าtypeเท่ากับtitleหรือไม่ ถ้าใช่, เพิ่มสไตล์ที่เป็นตัวหนาให้กับมัน

ค้นหา Hybrid Mobile App https://tinyurl.com/HybridApp-BSc

facebook.com/somsacki

C HAPTER 3 การสร้างแอป React Native ครั้งแรกของคุณ

ในการปรับใช้TabBarให้เปิด app/App.js นำเข้าองค์ประกอบTabBarและ set
มันขึ้น นอกจากนี้คุณยังจะได้นำมาในชนิดที่จะทำให้การทำงานเป็นส่วนหนึ่งของ destructuring this.state

```
...
นำเข้า TabBar จาก './TabBar'
แอปคลาสขยายคอมโพเนนต์ {
...
แสดงผล () {
const { todos, inputValue, type } = this.state
กลับ (
<ดู style={styles.container}>
<ScrollView
keyboardShouldPersistTaps='เสมอ'
style={styles.content}>
<หัวข้อ />
<อินพุต inputValue={inputValue}
inputChange={(text) => this.inputChange(text)} />
<TodoList
ประเภท={ประเภท}
toggleComplete={this.toggleComplete}
deleteTodo={this.deleteTodo}
todos={todos} />
<ปุ่ม submitTodo={this.submitTodo} />
</ScrollView>
<TabBar type={type} setType={this.setType} />
</ดู>
)
```

}
...
ที่นี่ คุณนำองค์ประกอบTabBar เข้ามา จากนั้นคุณทำลายประเภทจากสถานะ
และส่งผ่านไม่เพียงไปยังองค์ประกอบTabBarใหม่แต่ยังรวมถึงองค์ประกอบ
TodoListด้วย คุณจะ
ใช้ตัวแปรประเภทนี้ในเวลาเพียงวินาทีเดียวเมื่อกรองสิ่งที่ต้องทำตามประเภทนี้ คุณ
ยังส่งฟังก์ชันsetTypeเป็นอุปกรณ์ประกอบฉากไปยังองค์ประกอบTabBar
สิ่งสุดท้ายที่คุณต้องทำคือเปิดส่วนประกอบTodoListและเพิ่มตัวกรอง
เพื่อส่งคืนเฉพาะ todos ของประเภทที่คุณต้องการคืนโดยอิงจากแท็บที่
เลือก เปิด TodoList.js ทำลายประเภทของอุปกรณ์ประกอบฉาก และเพิ่มสิ่งต่อไปนี้
ฟังก์ชัน getVisibleTodos ก่อนคำสั่ง return

...
const TodoList = ({ สิ่งที่ต้องทำ, deleteTodo, toggleComplete, ประเภท }) => {
const getVisibleTodos = (todos, ประเภท) => {
สวิตช์ (ประเภท) {
กรณี 'ทั้งหมด':
ส่งคืนสิ่งที่ต้องทำ
กรณี 'เสร็จสมบูรณ์':
ส่งคืน todos.filter((t) => t.complete)
กรณี 'ใช้งานอยู่':

ส่งคืน todos.filter((t) => !t.complete)
}
}
todos = getVisibleTodos (สิ่งที่ต้องทำประเภท)
todos = todos.map ((สิ่งที่ต้องทำ, i) => {
...
คุณใช้คำสั่งswitchเพื่อตรวจสอบประเภทที่ตั้งไว้ในปัจจุบัน หากตั้งค่า'ทั้งหมด'คุณส่งคืนรายการสิ่งที่ต้องทำทั้งหมด หากตั้งค่า'เสร็จสมบูรณ์'คุณจะกรองสิ่งที่ต้องทำและส่งคืนเท่านั้น
สิ่งที่ต้องทำที่สมบูรณ์ หากมีการตั้งค่า'ใช้งานอยู่'คุณจะกรองสิ่งที่ต้องทำและส่งคืนเฉพาะincom-
สิ่งที่ต้องทำมากมาย
แล้วคุณตั้งค่าTodosตัวแปรเป็นค่าที่ส่งกลับของgetVisibleTodos ตอนนี้คุณควรจะสามารถเรียกใช้แอพและเห็นTabBarใหม่(รูปที่ 3.19) TabBarประสงค์ตัวกรองตามประเภทที่เลือก
รูปที่ 3.19 แอพ todo สุดท้าย

หน้า 72

สรุป

- AppRegistry เป็นจุดเริ่มต้น JavaScript เพื่อเรียกใช้แอป React Native ทั้งหมด
- ตอบสนององค์ประกอบพื้นเมืองTextInput คล้ายกับ HTML การป้อนข้อมูล คุณสามารถ

ระบุอุปกรณ์ประกอบฉากหลายรายการรวมถึงตัวยึดเพื่อแสดงข้อความต่อหน้าผู้ใช้เริ่มพิมพ์placeholderTextColorที่จัดรูปแบบข้อความตัวแทน และ a เลือกสีที่รูปแบบเคอร์เซอร์สำหรับTextInput

- TouchableHighlight เป็นวิธีหนึ่งในการสร้างปุ่มใน React Native; มันคือคอมปา-

rable ไปยังองค์ประกอบปุ่ม HTML คุณสามารถใช้TouchableHighlight เพื่อห่อ

มุมมองและทำให้พวกเขาตอบสนองต่อเหตุการณ์สัมผัสได้อย่างถูกต้อง

- คุณได้เรียนรู้วิธีเปิดใช้งานเครื่องมือสำหรับนักพัฒนาทั้งในอีมูเลเตอร์ iOS และ Android
- การใช้คอนโซล JavaScript (มีให้ในเมนูนักพัฒนา) เป็นวิธีที่ดี

เพื่อแก้ไขข้อบกพร่องของแอปและบันทึกข้อมูลที่เป็นประโยชน์