REACT JS OCHIQ KODLI JAVASCRIPT KUTUBXONASI AFZALLIKLARI VA QO'LLANILISHI.

Rustamova Malika Bahodirovna
Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Qarshi filiali
Abduxalilov G'olibjon Sanjarivich
Muhammad al-Xorazmiy nomidagi TATU Qarshi filiali talabasi

React Js server xizmat ko'rsatish (SSR) dan foydalaniladigan kuchli JavaScript kutubxonasi va qayta foydalanish mumkin bo'lgan foydalanuvchi interfeys komponentlarini yaratish uchun samarali, moslashuvchan va tavsiflovchi JavaScript kutubxonasi bo'ladi. Bu ochiq kodli kutubxona mobil va veb-ilovalar uchun dinamik va interaktiv foydalanuvchi interfeyslarini yaratish uchun ishlatiladi. React Js komponentga asoslangan dastur bo'lib, faqat dasturning tashqi ko'rinishi yoki oldingi qismi uchun mas'ul bo'lgan frontend kutubxonasi uchun foydalaniladi. Bu Facebookda dasturiy ta'minot muhandisi bo'lib ishlagan Jordan Walke tomonidan yaratilgan. U dastlab Facebook va keyinchalik WhatsApp tomonidan ishlab chiqilgan va Instagram tomonidan qo'llab-quvvatlanadi. Facebook 2011 yilda o'zining yangi ishlar bo'limida ReactJS-ni ishlab chiqdi, biroq u 2013-yil may oyida ommaga taqdim etilgan. ReactJS HTML DOM-da ma'lumotlarni to'ldirish uchun virtual DOM-ga asoslangan mexanizmdan foydalanadi. ReactJS ilovasi bir nechta komponentlardan iborat bo'lib, har bir komponent kichik, qayta foydalanish mumkin bo'lgan HTML kodini chiqarish uchun javobgar va bular har qanday React ilovasining yuragi hisoblanadi. React ilovasini yaratish uchun biz turli elementlarga mos keladigan React komponentlarini yozishimiz kerak. Reaktsiya bu ma'lumotlarni Document Object Model (DOM)ga oshkor qilish bilan bog'liq.

React JS quyidagi xususiyatlarga iborat.

JSX (JavaScript Syntax Extension):

JSX - bu HTML va JavaScript birikmasidir. Bu ReactJS tomonidan ishlatiladigan XML yoki HTML kabi sintez hisoblanadi. JSX kodlarni oson va tushunarli qiladi. Bu ES6 ni kengaytiradi, shuning uchun HTML kabi matn JavaScript reaksiya kodi bilan birga mavjud bo'lishi mumkin. HTML va JavaScript-ni bilsangiz, o'rganish oson.

```
const n="Home";
const ele = <h1>Welcome to {n}</h1>;
```

Komponentlar:

ReactJS ilovalari bir nechta komponentlardan iborat bo'lib, har bir komponentning o'z mantiqiy va boshqaruv elementlari mavjud. Shunday qilib, JavaScriptda yozilgan komponentlar mantig'i osonroq, tezroq ishga tushirish va qayta foydalanish imkonini beradi.

Virtual DOM:

Bu DOM (Document Object Model) ning qisqartmasi va Virtual DOM ob'ekti asl DOM ob'ektining ko'rinishidan boshqa narsa emas. U bir tomonlama ma'lumotlarni ulash kabi ishlaydi. Internetga o'zgartirishlar kiritilganda ilova birinchi navbatda barcha virtual

DOMlarni yangilaydi va haqiqiy DOM va Virtual DOM o'rtasidagi farqni topadi. Haqiqiy DOM va virtual DOM o'rtasidagi farqni topganda, u faqat yaqinda o'zgartirilgan DOM qismini yangilaydi va qolganlari o'zgarishsiz qoladi. Bu dasturni tezroq qila oladi va xotira ko'p sarflanmaydi.

Kengaytmalar yoki Toolkitlar

React, Flux, Redux, React Native va boshqalar bilan React JS kengaytirildi. Bu bizga yaxshi ko'rinadagi foydalanuvchi interfeysini yaratishga yordam beradi. React JS nafaqat mobil ilovalarni ishlab chiqishni qo'llab-quvvatlaydi, balki server tomonida renderlashni ham ta'minlaydi. To'liq UI ilovalarini yaratish uchun foydalanishimiz mumkin bo'lgan React-ning ko'plab kengaytmalari mavjud.

a.React.js biz bir necha yil oldingi muammoni hal qilgan. O'sha paytda biz veb ishlab chiqish tendentsiyasi yuqori darajada interaktiv DOM manipulyatsiyasiga o'tayotganini angladik, bu odatda serverdan yuborilganidan ma'lumotlar juda ham farqli bo'lib keladi. Biz 100% JavaScript-da yaratilgan va DOM-ga qo'lda (jQuery bilan) kiritilgan foydalanuvchi interfeyslarini yaratishda tajriba o'tkazishga qaror qildik, chunki foydalanuvchilarning o'zaro ta'siri zarur edi. Bizning fikrimiz shundan iborat ediki, biz juda murakkab foydalanuvchi interfeysi komponentlarini (masalan, blog dizaynlari yoki taqdim etish shakllari) yaratishimiz va ularni JavaScript kutubxonalariga joylashtirishimiz mumkin edi. Keyin biz bir nechta JavaScript kutubxonalarini qo'shish va bosh sahifaga bo'sh ‹body› tegini qo'yish orqali butun saytni "yarata olamiz". JavaScript tajribamiz natijalari quyidagi sabablarga ko‘ra ijobiy bo‘lmadi: Asta-sekin juda ham ko'p yozilgan interfeys komponentlar yaratish ularni Komponentlarga bog'liqligini ko'rib chiqishni va ularni saralashni talab qilar edi Foydalanadigan har qanday bog'lash uchun ishlatiladigan ko'rsatilganlarni qayta foydalanmasligimiz kerak. Ilovalar ko'lami sifatida modullikni saqlash juda qiyin bo'ladi.

Yuqoridagi muammolar React JS yaxshi yechim bo'lishiga ishonamiz.

Reactdagi har bir komponent har o'zgarishlar davomiy o'tishini ko'rishimiz mumkin.

- Mounting- komponentingizning paydo bo'lishi
- Update– komponentingizning yangilanishi
- Unmount- komponentingizning yuq bo'lishi

Endi ularning qanday ishlashini ko'proq tushunamiz harakat qilamiz.

Bu metod Reactda eng ko'p qo'llaniladigan usul hisoblanadi.

Biz buni barcha React sinflarida ko'ramiz. Buning sababi shundaki, bu Reactdagi sinf ichidagi yagona talab qilinadigan metod.

U bizning komponentimizni UI ga bildirishga va aniqlashga bilan shug'ullanadi. Bu bizning komponentimizni o'rnatish va yangilash paytida sodir bo'ladi.Keling, React-da oddiy render() misolini ko'rib chiqaylik.

```
class First extends Component{
 render(){
   return <div>First {this.props.name}</div>
 }
}
```

Yuqoridagi misolda ko'rib turganimizdek, render() usuli foydalanuvchi interfeysida ko'rsatilgan JSX (JavaScript kengaytmasi) ni qaytaradi. Agar ushbu komponentda ko'rsatiladigan hech narsa bo'lmasa, u nullni ham qaytarishi mumkin. React bizning render() ni toza bo'lishini talab qiladi. Toza funksiyalar hech qanday nojo'ya ta'sirga ega bo'lmagan va bir xil kirishlar kiritilganda har doim bir xil chiqishni qaytaradigan funktsiyalardir. Bu shuni anglatadiki, biz show() ichida state() ni o'rnata olmaymiz. Biz view() ichidagi komponent holatini o'zgartira olmaymiz.

Agar biz boshqa davomiylik usullarida yuzaga kelishi kerak bo'lgan holatlarni o'zgartirishimiz kerak bo'lsa, render() ni toza saqlashimiz kerak.

Shuningdek, u render() ni Main(Asosiy) yangilanishlarisiz sodda va toza saqlash imkonini beradi.

componentDidMount():

Endi bizning komponentimiz o'rnatildi va tayyor bo'ldi, keyin keyingi React davomiylik sikli usuli componentDidMount() ishga tushadi.

Ushbu usul komponent yig'ilgan va tayyor bo'lganda chaqiriladi. Agar biz masofaviy so'nggi nuqtadan ma'lumotlarni yuklamoqchi bo'lsak, bu API uchun yaxshi joy. Render() usulidan farqli o'laroq, component DidMount() setState() ga ham ruxsat beradi. setState chaqiruvi bu yerda u o‘z holatini yangilaydi va uni boshqa render qilishga sabab bo‘ladi, lekin bu brauzer foydalanuvchi interfeysini yangilashdan oldin sodir bo‘ladi. Bu foydalanuvchi ikki tomonlama displeyli foydalanuvchi interfeysi yangilanishlarini ko'rmasligiga ishonch hosil qilish uchun amalga oshiriladi.

componentDidUpdate()

Ushbu hayot aylanish usuli yangilanish sodir bo'lganda chaqiriladi. Agar biz ushbu usulning umumiy qo'llanilishi haqida gapiradigan bo'lsak, bu DOMni prop yoki holat o'zgarishlariga javoban yangilashdir. Ushbu hayot tsiklida biz setState() ni chaqirishimiz mumkin, lekin biz avvalgi holatdan holat yoki rekvizit o'zgarishlarini tekshirish uchun uni bir holatga o'zgartirishimiz kerakligini yodda tutishimiz kerak. setState() dan noto'g'ri foydalanish cheksiz tsiklga olib kelishi mumkin.

Keling, ushbu hayot aylanish usuli uchun odatiy foydalanish holatini ko'rsatadigan quyidagi misol bilan tushunaylik

```
componentDidUpdate(prevProps) {
 //do not forget to compare current props to the previous props
if (this.props.userName !== prevProps.userName) {
  this.fetchData(this.props.userName);
}
}
```

Yuqoridagi misolga ko'ra, biz hozirgi rekvizitlarni oldingi rekvizitlar bilan taqqoslaymiz.Bu qo'llab-quvvatlovchilarning hozirgi holatidan o'zgarganligini tekshirish uchun ishlatiladi. Bunday holda, qo'llab-quvvatlashlar o'zgartirilmasa, API chaqiruvini amalga oshirishning hojati yo'q.

component WillUnmount()

Ushbu usul komponentni tarqatish va yo'q qilishdan oldin qo'llaniladi. Agar tozalash choralarini ko'rishimiz kerak bo'lsa, buni ishlatish to’g’ri bo’ladi. Shuni yodda tutishimiz kerakki, biz Component Will Unmount davomiylik siklida komponent holatini o'zgartira

olmaymiz. Ushbu komponent boshqa hech qachon oshkor etilmaydi va shuning uchun biz ushbu davomiylik usuli davomida setState() ni chaqira olmaymiz.

```
componentWillUnmount () {
window.removeEventListener('resize', this.resizeListener)
}
```

Ushbu usul yordamida amalga oshiriladigan umumiy tozalash chorasida datalarni tozalash, api qo'ng'iroqlarini bekor qilish yoki xotiradagi har qanday keshni tozalash kiradi. Kamdan-kam uchraydigan davomiylik usullari. Endi bizda tez-tez ishlatiladigan React davomiylik usullari haqida yaxshi fikr bor. Bundan tashqari, Reactdan kam yoki umuman foydalanilmaydigan davomiylik usullari mavjud.

shouldComponentUpdate()

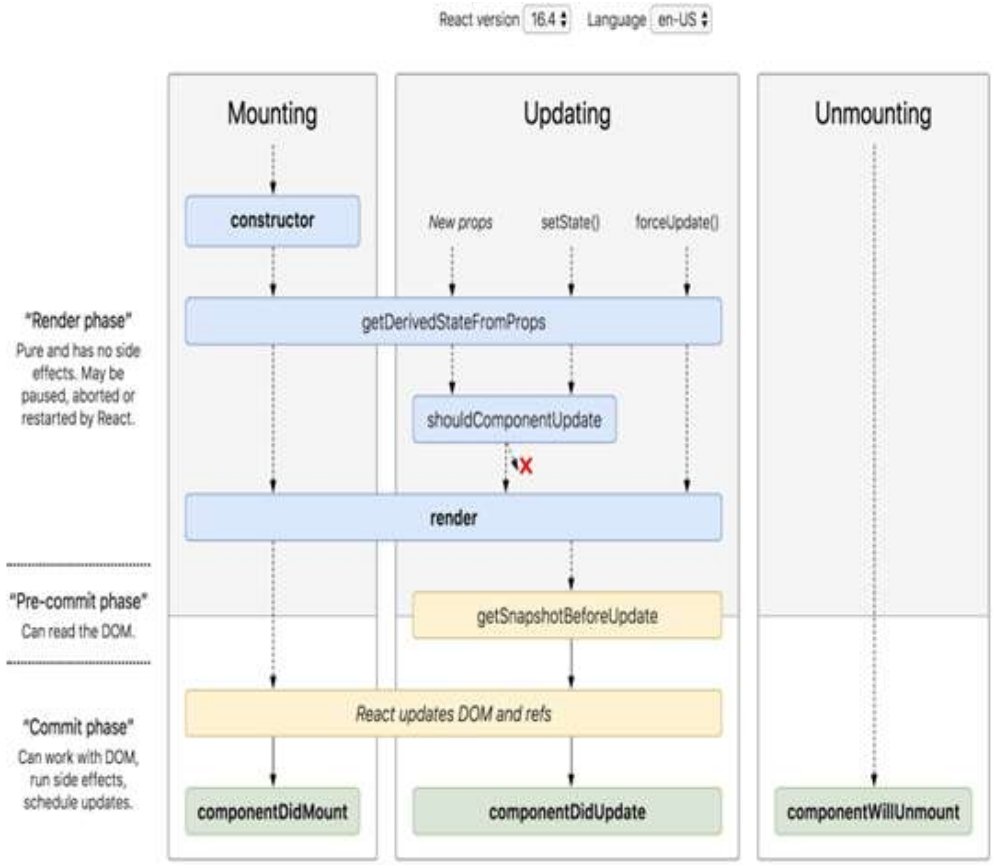
Bu davomiy aylanishi biz React bizning holatimizni yoki rekvizit o‘zgarishlarimizni ko‘rsatishini istamagan paytlarda foydali bo‘lishi mumkin. setState() chaqirilganda, komponent o’zgarmasligi bo'yicha uni yana ko'rsatadi. Should Component Update() usuli Reactga komponentga holat va prop o‘zgarishlari ta’sir qilmasligini bildirish uchun ishlatiladi. Shuni esda tutishimiz kerakki, ushbu davomiylik aylanish usuli juda kam ishlatilishi kerak va faqat ma'lum ishlarni optimallashtirish uchun mavjud. Ushbu jarayon aylanishida komponent holatini yangilay olmaymiz.

```
shouldComponentUpdate(nextProps, nextState)
{
return this.props.title !== nextProps.title ||
this.state.input !== nextState.input
}
```

Yuqoridagi misoldan shuni ko'rishimiz mumkinki, bu davomiylik sikli har doim savolga mantiqiy qiymatni qaytarishi kerak: "Men butun komponentimni yana ochishim kerakmi?" statik Get Derived State From Props Generic Name(). Bu yaqinda React jamoasi tomonidan kiritilgan yangi davomiy aylanish usuli hisoblanadi. Aytishimiz mumkinki, bu component WillReceiveProps() oldingi davomiy aylanish usulidan xavfsizroq va alternativa bo'ladi. U render() usulini chaqirishdan oldin chaqiriladi. Bu "this"(global) ga kirish imkoni bo'lmagan statik funktsiyadir. getDerivedStateFromProps() rekvizit o'zgarishlariga javoban holatini yangilash uchun ob'ektni qaytaradi. Agar holatga hech qanday o'zgarish bo'lmasa, null qaytarilishi mumkin. Bu holat komponent plaginlaridagi o'zgarishlarga bog'liq bo'lgan kamdan-kam hollarda qo'llaniladi.

```
static getDerivedStateFromProps(props, state)
{
if (props.currentRow !== state.lastRow)
{
return {
isScrollingDown: props.currentRow > state.lastRow,
lastRow: props.currentRow,
};
}
// Return null to indicate no change to state.
return null;
}
```

Shuni yodda tutishimiz kerakki, ushbu hayot aylanish usuli har bir transformatsiyada qo'llaniladi.‹Transition› komponenti oldingi va keyingi bola elementlarni solishtirish orqali qaysibola elementlarni ishlatishimiz va chiqarishni hal qilganda, bu usul foydali bo'lishi mumkin.



Ishlab chiqaruvchilar React JS ni qullab kelyapdi. Dastur ishlab chiquvchiga tahlil qilish imkonini beradi. Komponentlar ularga bitta sahifali dastur yaratishga imkonini beradi kamroq kod yozish orqali. U SEO do'stona sifatida ham tanilgan. Biz yaxshilab sarab olamiz va juda ham katta dastur ma'lumotlarini tanlanadi. Ushbu muhim afzalliklar tufayli React JS yuqori o'rinlarni qo'lga kiritdi bu esa ko'pchilikning e'tiborini tortadi. Ma'lumki, React JavaScriptga asoslangan frontenddir, ishlab chiquvchilarga va biznes egalari katta foyda keltiradigan frame workdir. Keyinchalik, bu framework tezda mashhurlikka erishdi va bugungi kunda eng maqul variant hisoblanadi. "ReactJS" ni o'rganish bu zamon talabi va mutlaqo mantiqiy dasturiy qismida ishlab chiquvchilar uchun juda zarur qulayliklarni ta'minlash, tez-tez o'zgarib turadigan ma'lumotlar bilan veb-ilovalar va foydalanuvchi interfeyslari ancha qisqa vaqt ichida katta hajmdagi ilovalar yaratish imkon berdi. Reactning afzalliklari shundan iboratki mustahkam, ilg'or, tez o'zgaruvchan, xavf-xatarsiz va foydalanuvchilar uchun qulayliktarafi bilan ancha ustundir va chunki React-dan foydalanadigan ishlab chiquvchilar va tashkilotlar bozorida dolzarb hisoblanadi, shuning uchun ular buni targ'ib qilamiz.

FOYDALANILGAN ADABIYOTLAR:

1.Cherry, S.M "Talk is cheap; text is cheaper [mobile messaging]"

2.Fu Kaifang "Design and implementation of an instant messaging architecture for mobile collaborative learning" Computing,

3.Butler, M "Android: Changing the Mobile Landscape",

4.https://react.dev/

5.https://legacy.reactjs.org/

6.https://react-redux.js.org/