REACT.JS GİRİŞ

Açık kaynak dünyasına katkı sağlayan Facebook tarafından geliştirilen React.js, JavaScript ile önyüz bileşenleri oluşturma kütüphanesidir. Şuan piyasada popülerliği yüksektir.

React, içerisindeki state değişikliklerinde arayüzü tekrar oluşturur. Aslında arayüzün tekrar oluşturulması bir performans kaybı oluşturur ancak React bunu yaparken VirtualDOM adında sanal DOM mekanizması kullanır. DOM (Document Object Model) yazılım dünyasında dilleri barından bir standart olarak tanımlanır. DOM, HTML ile programlama dilleri arasında bir standart oluşturarak bu dillerin HTML den bilgi alıp, bilgi vermesine yardımcı olur. React render edilen DOM'un bir kopyasını VirtualDOM olarak tutar. İçerisinde ki herhangi bir state değişikliğinde DOM'da bir değişiklik oluyorsa sadece bu değişikliği VirtualDOM'a yansıtır ve bu durum render edilen DOM ile VirtualDOM arasında farklılık ortaya çıkarır. İşte React bu farklılıkları bularak DOM içerisinde sadece değişen alanları yeniden render eder ve bütün DOM'un tekrar render edilme maliyetinden korunulmuş ve azami performans iyileştirmesi sağlanmış olur.

React çalışma prensibinden bahsettikten sonra çalışma ortamını kurmaya başlayabiliriz. Hazır oluşturulan proje iskeletleriyle React ile kodlamaya başlamak çok kolay. En popüleri olan React'ın desteğini sunduğu *create-react-app* (deneyimli JavaScript geliştiricileri tarafından projelerin ortak ayarları belirlenerek oluşturulmuş projenize alt yapı sağlayan bir proje oluşturucu araç). (https://github.com/facebookincubator/create-react-app) Hazır iskeleti kullanabilmek için bilgisayarınızda kurulu olarak NodeJS ve beraberinde npm ya da alternatifi yarn package manager bulunması gerekiyor. Aşağıda ki komutları terminalde veya komut satırında çalıştırdığınızda benzer çıktıları görmeniz gerek.

```
node –version → v8.9.3

npm –version → 5.5.1
```

Herşeyin hazır olduğunu varsayarsak aşağıda ki komutları, komut satırında veya terminalde çalıştırarak yeni bir proje oluşturabiliriz. *my-app* kısmına kendi proje adınızı yazın.

```
npm install –g create-react-app

create-react-app my-app

cd my-app

npm start
```

"npm install –g create-react-app" komutu ile proje oluşturucu aracı indiriyoruz. "create-react-app my-app" komutumuzla yeni projemizi oluşturuyoruz. "cd my-app" komutu ile proje klasörüne gidip "npm start" komutu ile de localhost:3000 adresinden ayağa kaldırılmış olacak. Tarayıcınızın ekranında React logosu ve bir takım yazılar görmüş olacaksınız.

Render, Component, State, Props, Lifecycle kavramları nedir ?

Props ve state kavramalrını incelersek;

Props, react bileşenlerinde yukarıdan gelen veriyi tutan nesnedir. Props değerleri dış bileşenlerden external olarak gelir ve ilgili bileşen içersinde readonly'dir. Kısacası bileşenler arası iletişimi sağlar. Mesela üzerindeki yazıyı dışarıdan props aracılığıyla alan basit bir button bileşeni.

```
const MyButton = (props)=>{
   return(
      <button> { props.text } </button>
   )
 }
```

State, mevcut bileşen içinde kullanılacak ve değiştirilme durumu olan veriler için kullanılır.

Bileşenlerin(Component) güncellenip güncellenmeyeceğine manuel olarak karar verebilir veya bileşenin henüz sayfada yeni yükleniyorken yapacağınız işlere biz karar verebiliriz. Bileşenlerin yaşam döngüsünü tek tek inceleyecek olursak;

componentWillMount() → React bileşenleri render edilmeden(yani oluşturulmadan) hemen önceki adımdır. Bu aşamada hala state üzerinde değişiklik yapabilme imkanımız vardır.

render() → React bileşenlerin çağrıldığı yerde return içerisinde bulunan veriyi döner. Bunun dışında ki bir yerden ekrana herhangi bir veri döndürülmez.

componentDidMount() → React bileşenleri oluşturulur. Bu aşamada **ComponetDidMount** methodu tetiklenir.

Buraya kadar react bileşenlerinin oluşturulması sırasındaki kullanabilceğimiz methodlardı. Şimdi react bileşenlerinin oluşturulması(Mount) sırasındaki kullanabilceğimiz methodlara inceleyeceğiz.

componentWillReceiveProps(newProps) → Bileşenler property'lerini yukarıdan aşağıya doğru one-way-binding şeklinde aktarır. Props'lar üzerinde değişiklik olduğunda çalışır.

Bu adımda **ComponetWillReceiveProps** mothodu tetiklenir. Method içerisinden props değişikliklerine ulaşılabilir.

shouldComponentUpdate(newProps, newState) → React bileşenlerinin çalışması itibariyle props'lar ve state üzerinde değişiklik olduğunda otomatik olarak render işlemini çalıştırabilmektedir. Bu adımda **ShouldComponentUpdate** methodu tetiklenir. Method false degeri dönerse componetin rendering işlemi iptal edilmiş olur. (default olarak her zaman true döndürür)

componentWillUpdate(nextProps, nextState) → ComponetWillMount methodunda olduğu gibi react bileşenlerinin render edilmesinden hemen önceki adımdır. Farkı ise ilk render işleminde değil props veya state üzerinde değişiklik olduğunda çalışmasıdır.

componentDidUpdate(prevProps, prevState) → ComponentDidMount methodunda olduğu gibi react bileşenlerinin Dom üzerinde oluşturulduğunu anlatır. Bu aşamada **ComponentDidUpdate** methodu tetiklenir.

Son olarak UnMount (Ortadan Kaldırma) sırasında kullanılan tek bir method vardır.

componentWillUnmount() → React bileşenlerinin render edilip Dom'a basıldıktan sonra Dom üzerinden kaldırılması durumunda tetiklenecek methoddur. Bu method sayesinde bileşen uygulama hiyararşisi içerisinden kaldırılırken daha önce bileşen ile ilişkilendirilen nesnelerin temizlenmesi sağlanır.
Alttaki örnekde buu methodların kullanıldığı örnek bir kod parçasını paylaşıyorum

```
class Persons extends Component{

    constructor (props) {

        super(props);

        console.log('Persons.js içinde constructor');

     }
```

```
    componentWillMount(){
      console.log('Persons.js içinde componentWillMount');
    }
    componentDidMount() {
      console.log('Persons.js içinde componentDidMount');
    }
    componentWillReceiveProps(nextProps) {
      console.log('UPDATE Persons.js içinde componentWillReceiveProps', nextProps);
    }
    shouldComponentUpdate(nextProps, nextState) {
       console.log('UPDATE Persons.js içinde shouldComponentUpdate', nextProps,
nextState);
       return true;
    }
    componentWillUpdate(nextProps, nextState) {
      console.log('UPDATE Persons.js içinde componentWillUpdate', nextProps, nextState);
    }
    componentDidUpdate() {
      console.log('UPDATE Persons.js içinde componentDidUpdate');
    }
   render () {
      console.log('Persons.js içinde render()');
      return this.props.persons.map((person, index) => {
         return <Person
            click={() => this.props.clicked(index)}
            name={person.name}
            age={person.age}
            key={person.id}
            changed={(event) => this.props.changed(event, person.id)} />
      } );
  }
```