React Nedir?

Reactjs son zamanlarda özellikle son bir yıldır yaygın olarak kullanılmaya başlanan bir javascript framework. İçeriğin zaman içinde değiştiği interaktif uygulamalar için oldukça etkili bir yönetilebilirlik sunan bu teknoloji, zamanla çok daha yaygın hale gelecektir.Halihazırda dünya genelinde kullanım oranına bakılırsa java script framework listesinde Angular Js sonrasında ikinci sırada yer alıyor.Kısaca bakmak gerekirse zamanında Facebook mühendisleri tarafından geliştirilmiş bir teknoloji. Yüzlerce geliştiricinin aynı zaman zarfında proje üzerinde çalışmasına kolay şekilde olanak veren, etkili bir DOM yönetimi sağlayan bir proje.

Neden React ?

React hızlı, basit ve hedeflediğimiz yüksek performansı bize sunabilecek bir kütüphane. Yeterli community desteğinin yanı sıra arkasında Facebook gibi büyük bir şirket var ve bunlar tercih sebeplerimizden sadece bir kaçı. Bazı eksikleri olsa da (hızlı bir şekilde eksiklerini kapatıyor) production'a çıkmak için makul bir seviyede. Dom manipülasyonundan uzak durmak istiyorsanız sizin için iyi bir seçenek. Çünkü dom'a her dokunuş performans kaybı demek. React view katmanında oldukça başarılı ve sadeliği ile iyi bir yol arkadaşı.

Virtual DOM nedir?

İsminden de anlaşılacağı gibi Virtual DOM, gerçek DOM'a karşılık gelen sanal bir DOM nesnesidir, yani render edilen DOM'un bir kopyasıdır. React her state değişikliğinde render edilen gerçek DOM'u bütünüyle tekrar oluşturmak yerine, state değişikliğini Virtual DOM'a yansıtmaktadır. Bu sayede gerçek DOM ile ve yeni sanal(virtual) DOM arasındaki farklılıkları tespit ederek, gerçek DOM'da yapılacak değişikleri hesaplar ve tek seferde sadece gerçek DOM'da değişen elemanları yeniden render eder. Bu şekilde state her değiştiğinde ana DOM'un tüm elemanlarıyla yeniden oluşturulması maliyetinden korunulmuş ve azami performans iyileştirmesi sağlanmaktadır. Gerçek DOM render edildiğinde, virtual DOM ile eşittir. Virtual DOM'da bir değişiklik meydana geldiğinde, gerçek DOM'a sadece değişikliğe konu alanlar yansıtılır ve her iki DOM tekrar birbirine eşitlenmiş olur. Aşağıdaki grafikte, bu döngü anlatılmaya çalışılmaktadır.

JSX

JSX kod yazımını kolaylaştırmak ve okunabilirliği arttırmak için geliştirilmiş bir JavaScript söz dizimidir. JSX söz dizimi kullanmadan React ile geliştirim yapmak zordurdiyebiliriz.XHTML'dekine benzer katı kurallar vardır. Açılan etiketler mutlaka kapatılmalıdır. Büyük küçük harfe dikkat edilmelidir.JSX ile XHTML benzerliğine dikkat çektik ancak yinede bazı JSX attributeleri bir kaç farklılığa sahiptir.

Örneğin:

class → className

for → htmlFor

React Elementleri

HTML yazarken, etiket adlarını, niteliklerini ve içeriğini (innerHTML) girersiniz. Ancak React, JavaScript'te bunu yapmanıza yardımcı olur. Aslında bazı Reaksiyon kodu, doğrudan babel JSX transpiler kullanılarak JavaScript ifadeleri olarak yazılan HTML etiketlerini içerir.

React Elements'i "programlı olarak oluşturulan" HTML etiketleri olarak düşünüyor ve onlarla çalışmayı doğrudan JavaScript programınızdan daha verimli hale getiriyor.

Element Yaratma

Bir React Element oluştururken temel fikir, HTML etiket adını, etiket özniteliklerini JSON biçiminde (yani, parantez ayraçları kullanarak) sağlamak ve etiket içeriğini sağlamaktır: oluşturulmakta olan etiketin innerHTML değeridir.

React, ana React nesnesinden kaynaklanan bir yöntem createElement'e sahiptir. Bu örnek, web sayfanıza React kütüphanesini dahil ettiğinizi varsaymaktadır.

Örnek:

```
React.createElement(ElementName,      // "p", "div", "span", and so on...
                    ElementAttributes,    // { attr1: "value", ... }
                    ElementContent);       // "Text, or other HTML."


var P = React.createElement("p",
                            { className: "paragraph" },
                              "Hello from React."
                            });
```

React bileşeninin yaşam döngüsü

componentWillMount() → bileşen oluşturulmadan önce

componentDidMount() → bileşen oluşturulduktan sonra

componentWillReceiveProps(newProps) → yeni props eklenmeden önce

shouldComponentUpdate(newProps, newState) → bileşenin güncellenip güncellenmeyeceği kararı verilir. (default olarak her zaman true döndürür)

componentWillUpdate(nextProps, nextState) → bileşen güncellenmeden önce

componentDidUpdate(prevProps, prevState) → bileşen güncellendikten sonra

componentWillUnmount() → bileşen DOM'dan silinmeden önce

render() → bileşenin render işlemi (return edilecek)

React bileşenleri

React bileşenlerinin tekrar kullanılabilirliği, dinamikliği ve erişimi sağlamak için props, state refs adında 3 alana sahip.

Props: Nesneye yönelimli programlamada kurucu metoda verilen parametreler gibi düşünebiliriz. Sonradan değişmeyecek veriler React bileşeni oluşturulurken bileşene bağlanır(bind).

Refs: DOM'daki React bileşenlerine erişmemiz için kullanılır. Küçük bir örnekle bileşen oluşturulduktan sonra içeriğini değiştirelim.

State: Bileşen içinde değişecek veriler için kullanılır. Stateler değiştiğinde React ne yapar? başlığından öğrendiğimiz gibi React bu değişiklikleri algılar ve yeniden render işlemine gider.

Events

Clipboard Events:

onCopy,onCut ,onPaste

Composition Events:

onCompositionEnd,onCompositionStart,onCompositionUpdate

Keyboard Events:

onKeyDown , onKeyPress ,onKeyUp

Focus Events:

onFocus, onBlur

Form Event:

onChange, onInput, onInvalid, onSubmit

Mouse Events:

onClick , onContextMenu, onDoubleClick, onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp

Selection Events:

onSelect

Touch Events:

onTouchCancel, onTouchEnd, onTouchMove, onTouchStart

UI Events:

onScroll

Wheel Events:

onWheel

Media Events:

onAbort onCanPlay onCanPlayThrough onDurationChange onEmptied onEncrypted

onEnded onError onLoadedData onLoadedMetadata onLoadStart onPause onPlay

onPlaying onProgress onRateChange onSeeked onSeeking onStalled onSuspend

onTimeUpdate onVolumeChange onWaiting

Image Events:

onLoad, onError

Animation Events:

onAnimationStart ,onAnimationEnd ,onAnimationIteration

Transition Events:

onTransitionEnd

Other Events:

onToggle

Paket Yönetme Yöntemleri

Bundle işlemleri için webpack, gulp ve grunt gibi araçlardan daha esnek(inanılmaz derecede!) ve daha güçlü bir araç. Webpack code split yapmak için oldukça iyi bir bundle management tool. Plugin desteği

sayesinde bir çok işi zahmetsiz bir şekilde halletmeniz mümkün.

Karşılaşılabilecek Problemler

Bu kadar teknolojiyi ve aracı bir arada kullanıp ortaya bir ürün çıkarmanın başlı başına emek ve zaman gerektirdiğini tecrübe etmiş biri olarak javascript dünyasının çeşitliliği ve beraberinde getirdiği sorunlarla baş etmenin kolay olmadığını söyleyebilirim.

Özellikle kullanılan tüm teknolojinin open source olması bir arada kullanılabilirliği sürdürmekte zorlanmanıza neden olabiliyor. Bağımlılıklar, bağlılıklar geliştirmenin ilk fazlarında değil tüm aşamalarında dikkat gerektiriyor.

Hazır component'lar bulma konusunda sıkıntılar yaşayabilirsiniz ve oturup bir çok şeyi yeniden yapmanız gerekebilir.

Zamanla stabilite oluşmaya başladıkça teknolojilerin bir arada kullanımı daha kolaylaşacaktır.Tasarımsal süreçlerin en az geliştirme kadar önemli bir konu olduğunu unutmamalısınız. Eğer proje bittiğinde sadece bir iki component reusable ise ortada bir problem var demektir. Tasarım sürecini planlarken component'ların tekrar kullanılabilirliğine dikkat edilmeli. Aksi taktirde aynı işleve sahip ancak farklı tasarımları olan bir çok component geliştirmiş olursunuz.