Web Programlama I

Ders 03 - SPA Bileşenleri

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

Eğitmen: Ömür ŞAHİN

Sunum Planı

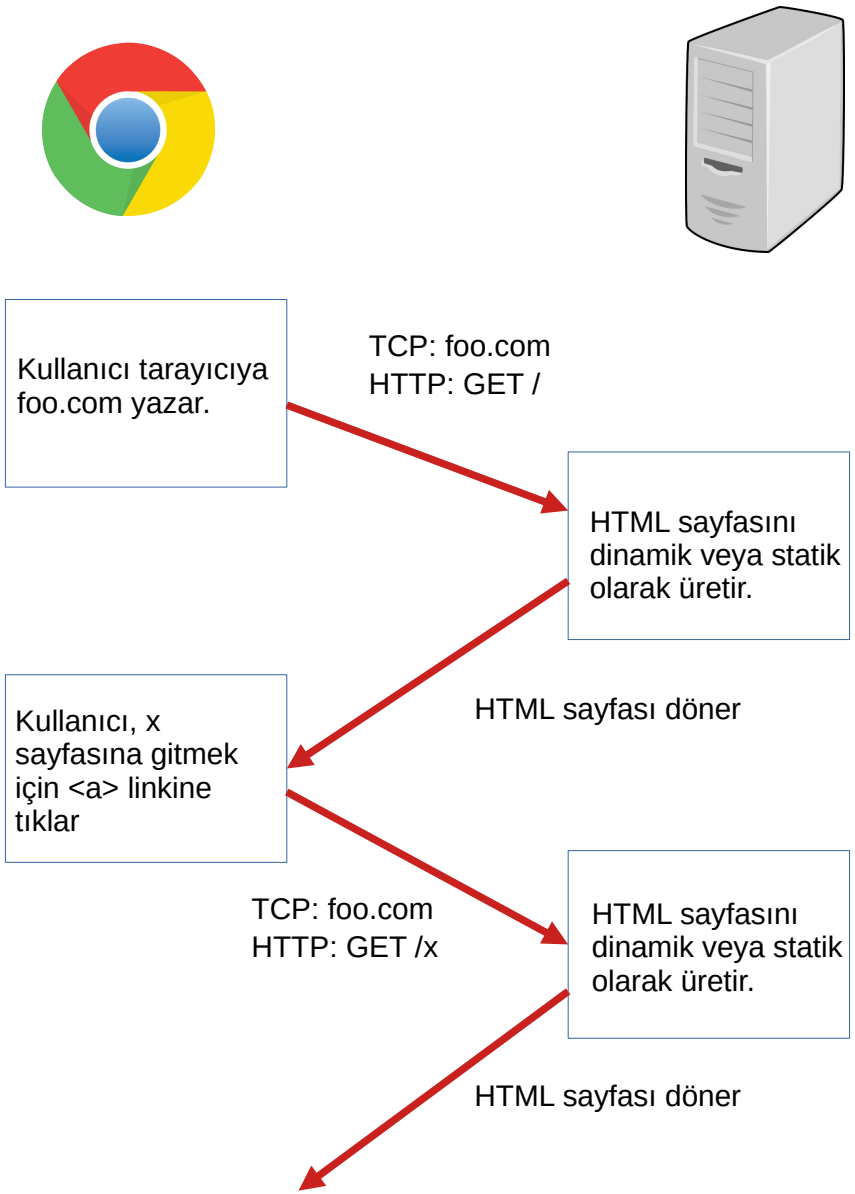
1-Genel Bakış

Genel Bakış

- Tek Sayfa Uygulamaları (Single Page Application, SPA) temel konsepti
- Neden doğrudan DOM manipülasyonu tavsiye edilmemektedir, neden kütüphane/framework kullanması daha iyidir?
- SPA componentlerini anlama
- React'a giriş

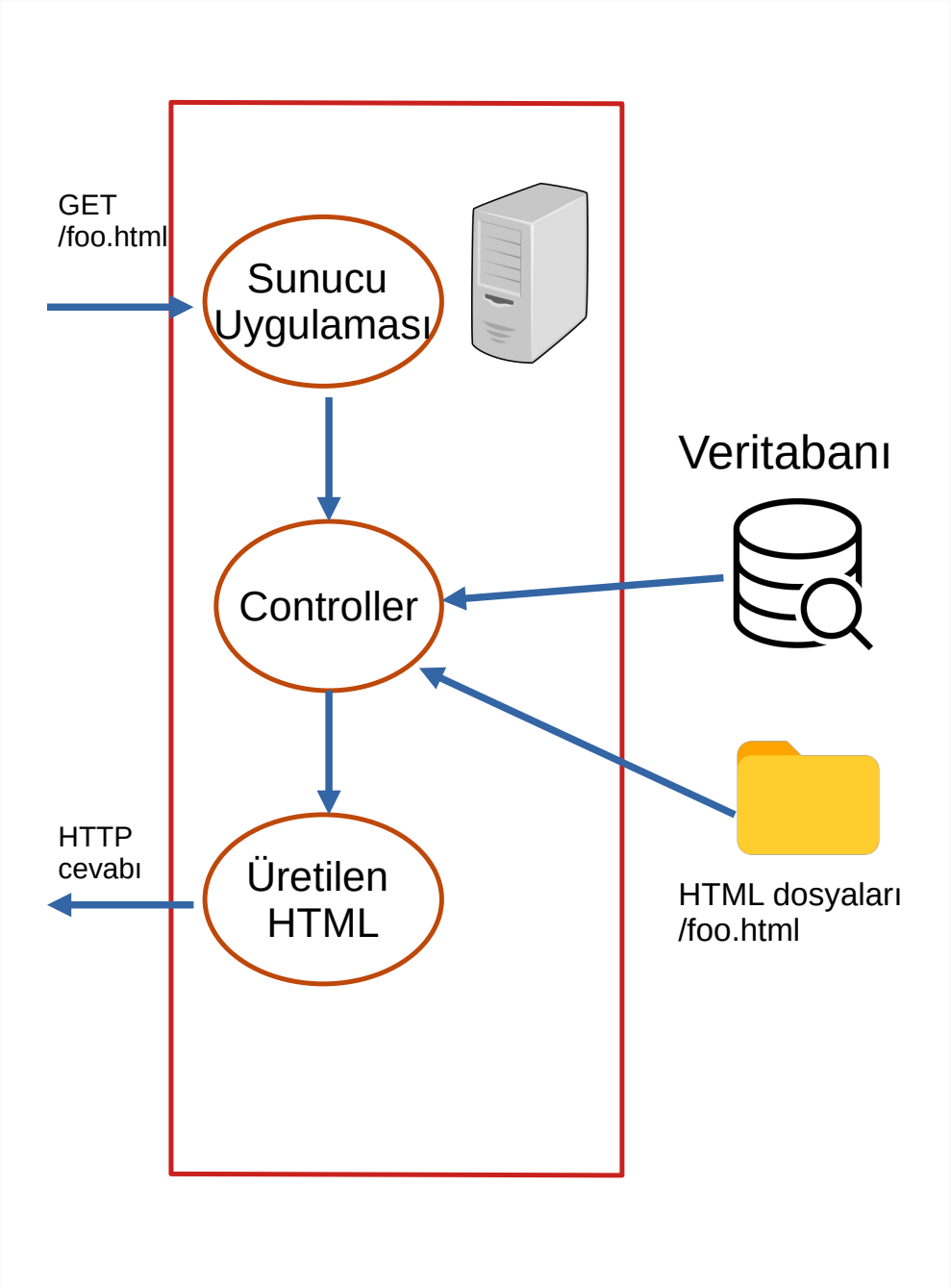
2-Geleneksel Web Uygulamaları

Geleneksel Web Uygulamaları



- Sayfalar arası etkileşim $< a >$ ve $< form >$ gibi taglarla gerçekleştirilir.
- Her bir istek GET, POST gibi metotlardan oluşmaktadır.
- HTML sayfasının tamamı sunucu tarafından üretilir.

Server Side Rendering



3-SPA

SPA

- Yalnızca tek bir HTML dosyasından oluşur.
- Bütün HTML içeriği JavaScript tarafından dinamik olarak üretilir.
  - Örneğin, DOM manipülasyonu ile.
- Sayfalar arası geçiş GUI'nin anında değiştirilmesi ile simüle edilir.

Veri çekme

- HTML dosyalar tarayıcı tarafından JS ile üretiliyor olsa da sunucu ile iletişime ihtiyaç duyulmaktadır.
  - Veri kaydetme/değiştirme
- Yeni bir HTML dosyası alınmaz.
- Veriler JavaScript Object Notation (JSON) formatındadır.
- JS DOM üzerindeki güncellemeleri JSON verilerine göre gerçekleştirir.
- Sunucular JSON veri sağlar.

SPA karmaşıklığı

- Frontend tarafında çok fazla JS kodu bulunmaktadır.
- Bütün tarayıcı eventlerinde veya durum değişikliklerinde DOM güncellemesini manuel yapmak çok ölçeklenebilir bir yaklaşım değildir.
- Bu türde karmaşık problemleri çözmek için tasarım paternlerine ve çeşitli araçlara ihtiyaç duyulmaktadır.

Kütüphane/Framework

- Frontend teknolojileri oldukça hızlı değişmektedir.
- Şu anda tamamen açık kaynak 3 temel kütüphane bulunmaktadır.
- **React**: Facebook geliştirmiştir.
- **Angular**: Google geliştirmiştir.
  - Oldukça ağır bir frameworktür.
  - Hala kullanılmakta ancak popülerliğini yitirmektedir.
- **Vue**: Bir geliştirici tarafından oluşturulmuştur.
  - Asyada oldukça popülerdir.
  - Bus faktör.

4-React

React Bileşenleri (Component)

- Componentler bir **state** ile tanımlanır ve bu state kullanılarak HTML **render** edilir.
- Web sayfası bir root component ile temsil edilir ve ağaç yapısı halinde children component'lar barındırır.
  - Her bir component'ın kendi state'i vardır ve yalnızca kendisini nasıl render edeceğini bilmektedir.
- Doğrudan render işlemi gerçekleştirilmez. Herhangi bir state değişimi olduğunda React otomatik olarak gerekli yerleri render eder.

Render Optimizasyonu

- Tarayıcı üzerinde çok fazla event bulunabilir (kullanıcı tıklamaları, fare hareketi vs.)
- React gerekli noktaları tekrar render ederken otomatik olarak optimize eder.
  - Örneğin birkaç ms içerisinde gerçekleşen bir güncellemeyi aynı anda gerçekleştirebilir.
- Virtual-DOM
  - Component state'i değiştiğinde HTML'in oldukça küçük bir noktasını etkilemiş olabilir.
  - React tüm HTML'i yeniden oluşturmaz. Yalnızca değiştirilen kısımları oluşturur.
  - Bir Virtual-DOM bellekte tutulur ve sadece mevcut arayüz VDOM'dan farklı olduğunda güncellenir.

JSX

- Bir React component'i HTML kodunu render() metodu ile oluşturur.
- HTML içeriğini bir JS string'i içerisinde tutmak oldukça hataya açıktır.
- **JSX** ise JS ve HTML kodlarını bir arada tutabileceğiniz React dosya formatıdır.
- Tarayıcılar JSX formatını tanımazlar o yüzden Babel kullanarak JS koduna dönüştürülür.
- Not: JSX dosyaları için ".jsx" uzantısını kullanacağız. ".js" dosya uzantısının kullanılmasında da sakınca yoktur ancak doğru bir yaklaşım değildir.

5-Babel ve Webpack ile React

Babel ile WebPack

- package.json içerisine kütüphaneler dahil edilmelidir.
- Webpack'in node_modules dışındaki dosyalar ile Babel'ı kullanabilmesi için webpack.config.js dosyasının düzenlenmesi gerekmektedir.

```
module: {
    rules: [
        {
            test: /\.jsx$/,
            exclude: /node_modules/,
            use: {
                loader: "babel-loader"
            }
        }
    ]
},
```

6-React Componentleri

React.Component (Class-based)

- class **App** extends React.Component
- **constructor(props)**
  - Her zaman super(props) çağırılır.
  - Başlangıç state durumu doğrudan atanmalıdır (**this.state = ...**).
- **render()**: State ve props'lara göre HTML üretimi
- **setState(newState)**: State değiştirileceği zaman setState() fonksiyonu kullanılır.
  - Buradaki değişim asenkron gerçekleşir. this.state anında değişmez.
  - **setState(prev =>newState)** kullanırsanız önceki değere de erişebilirsiniz. Örnek: setState(prev => (sayac: prev.sayac+1))

setState()

- setState() ile bir state değiştirildiğinde yeni bir girdi nesnesi sağlanmalıdır.
  - Veya önceki state'i girdi olarak alıp yeni bir state dönen fonksiyon
- Yeni state mevcut state ile "**birleştirilir**".
  - Eğer bir state'te çok fazla alan tanımlı ise yalnızca değişen alanlar verilebilir.
- Mevcut state doğrudan kullanılmamalıdır.
  - render() metodu state'deki değişimlere bağlı olarak asenkron olarak çalışmaktadır. Farklı setState çağrıları da birleştirilebilir.

7-React Yaşam Döngüsü

React Lifecycle

- **componentDidMount()**: constructor ve ilk render() metodu çağırımı sonrasında çağırılacak fonksiyonu override eder.
  - Constructor kısmında çağırıldığında uygulamayı yavaşlatan backend çağrıları gibi zaman alan, maliyetli başlangıç işlemlerinde oldukça kullanışlıdır.
- **componentWillUnmount()**: Componentin DOM'dan kaldırıldığında çalışan fonksiyonu override eder.
- **componentDidUpdate()**: state/prop değişikliklerine göre tekrar render edildiğinde çalışan fonksiyonu override eder.

JavaScript Problemleri

componentDidMount() yerine componenDidMount() yazarsanız ne olur?

JavaScript Problemleri

- React tarafından asla çağırılmayacak bir başka fonksiyon olarak tanımlanır.
- Override edilen fonksiyon için herhangi bir keyword kullanılmamaktadır.
- WebStorm gibi bazı IDE'ler bir metot kullanılmadıysa uyarıda bulunabilmektedir.

React Hook'lar

- Hook'lar 2019 yılında React'a dahil edilmiştir.
- Component'ları state kullanarak fonksiyon olarak tanımlanmasına olanak tanır.
- React component yazmada mevcut önerilen yöntem Hook kullanmaktır.
- 2022 yılı itibaryla React Hook kullanacağız.

useState, useEffect

- const [age, setAge] = useState(1)
- useEffect(function,[dep])