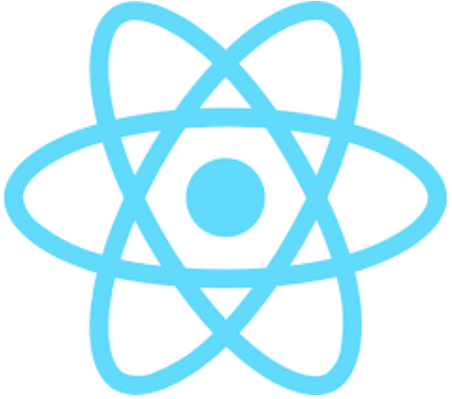


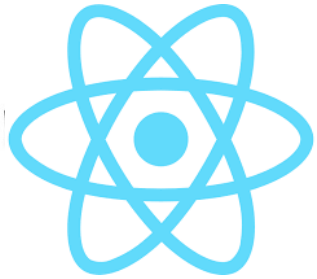
Formlar ve Data Binding

Onur KULABAŞ

onurkulabas@yahoo.com

React Training

Istanbul, 2022



Form Yapısı

Tüm web uygulamaları için **form** yapısı ve element'lere girilen **input** değerleri çok büyük öneme sahiptir. Geleneksel web uygulamalarında **Request** nesnesi ile form veya querystring metodları vasıtasıyla okunabilen input girdilerini **React ile çok daha dinamik** ve hızlı bir şekilde okuyabiliyoruz.

Form Kullanımı

```
<form>
  <label htmlFor="username">Username:</label>
  <input type="text" name="username" onChange={e=>setName(e.target.value)} />
  <br /><br />
  <label htmlFor="password">Last name:</label>
  <input type="text" name="password"  onChange={e=>setPassword(e.target.value)} />
  <br /><br />
  <input type="button" value="Gönder" onClick={()=>myButtonClick()} />
</form>
```

Yukarıdaki kodu **return ()** ifadesinin içine yerleştirelim. Bu örnekte JSX yazım şekline özgü htmlFor ve onChange, onClick kullanımlarını görüyoruz. Standart bir login formunu içeren örneğimizdeki **onChange** metotları ve **onClick** metodunun kullanım şeklini inceleyelim.

onChange - Detaylar

```
<input type="text" name="username" onChange={e=>setName(e.target.value)} />
```

Yukarıdaki kullanımda ilk kez arrow function kullanımını da görüyoruz. Bu modül içinde arrow function özelliklerini, kullanım şeklini ve standart function'larla farklarını ayrıca inceleyeceğiz. Bu örnekte username isimli text elementinin değerini **setName** objesi ile yöneteceğimizi ve **onChange** ile çağrıldığı için de **textbox içindeki her harekette** (yani her tuşa basıldığında veya diğer bir işlemde) **değerin güncelleneceğini** söyleyebiliriz. Aynı durum önceki sayfada kalan koddaki password isimli textbox ve setPassword isimli object arasında da mevcuttur.

onClick - Detaylar

```
<input type="button" value="Gönder" onClick={()=>myButtonClick()} />
```

Bu satırda da önceki 2 kullanım gibi arrow function kullanımı görüyoruz. Özetle butona basılınca adlı fonksiyon çağrılacaktır. Bu noktada yine geleneksel web uygulamaları ile olan bir farkı belirtelim. Kod satırında <form> şeklindeki kullanım eski tip web uygulamalarında <form action=…> şeklinde yapılır ve butona basıldığında genelde action içinde yazılan sayfaya post olunurdu. Artık sistem bu şekilde işlemiyor. Onun yerine **form element'lerinin durum değişikliklerini sürekli dinleyen ve sanal dom'u güncelleyerek** bilgileri sayfa post etmeden de alabilen bir yapı mevcut. Bu örnekte myButton fonksiyonu içinde okunan değerlerin login kontrolü ve duruma göre kullanıcıya hata mesajı döndürülmesi veya uygulama ana sayfasına yönlendirilmesi sağlanacaktır (Modül 10 dahilinde)

onClick - Detaylar

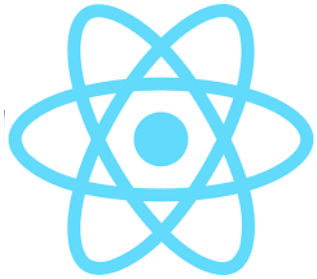
```
import React, { useState } from 'react’;
```

import kısmında bu satırın olduğundan emin olmalıyız. Çünkü değişken değerlerini okuyabilmek için useState bileşenini kullanacağız, bu kavramı daha önce görmüştük.

```
const [name, setName] = useState();
const [password, setPassword] = useState();
```

Yukarıdaki satırları return() satırlarının üst kısmına ekleyelim. Burada da yine daha önce benzerini gördüğümüz bir kullanım mevcut. Özetle **setName** objesi, **name** adlı değişkenin state’ini yani durumunu yönetecek. Koda geri dönüp bakarsak setName’in de aslında **username** isimli **textbox’ın onChange’ini** dinlediğini görebiliriz. Yani username isimli textbox’a bir şeyler yazıldıkça setName adlı obje, koddaki name isimli değişkene bu değeri set edecektir. İşte bu yöntemle textbox veya herhangi bir form element’inin değeri çok hızlı bir şekilde okunabilir. Aynı işlem, setPassword objesi üzerinden password değişkenine de uygulanmaktadır.

onClick - Detaylar

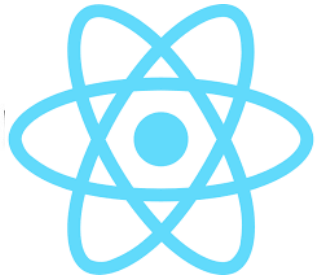


```
{name}<br/>
{password}
```

Yukarıdaki satırları return ifadesinin içinde </form> satırının altına ekleyelim. **{}** süslü parantez içinde **dinamik kodlar** yer alacaktır. Buradaki kullanımda değerleri daha önce set edilmiş olan name ve password isimli değişkenlerin değerleri ekrana yazdırılmaktadır.

Sonraki modülde okunan username ve password değerlerinin login kontrolü ve duruma göre kullanıcıya hata mesajı döndürülmesi veya uygulama ana sayfasına yönlendirilmesi sağlanacaktır

Arrow Functions

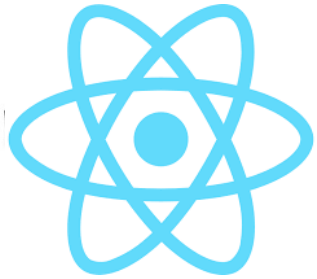


```
<input type="text" name="username" onChange={e=>setName(e.target.value)} />
<input type="button" value="Gönder" onClick={()=>myButtonClick()} />
```

Önceki örnekte gördüğümüz yukarıdaki 2 satırda kullanılan arrow function kullanımını biraz açalım. Arrow function'la regular function'ı basit bir örnek üzerinden karşılaştıralım.

```
// ES5
var add = function(x, y) {
    return x + y
};
// ES6
let add = (x, y) =>  x + y
```

Arrow Functions



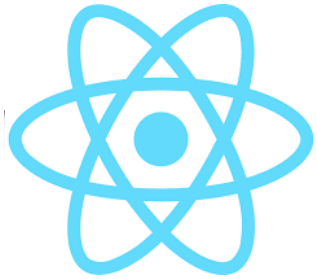
```
<input type="text" name="username" onChange={e=>setName(e.target.value)} />
<input type="button" value="Gönder" onClick={()=>myButtonClick()} />
```

Önceki örnekte gördüğümüz yukarıdaki 2 satırda kullanılan arrow function kullanımını biraz açalım. Arrow function'la regular function'ı basit bir örnek üzerinden karşılaştıralım.

1. Arrow function'larda **return** ifadesi implicit'tir; ekstra **yazılmaz**. Tek satır olacaksa {} ifadelerine de gerek yoktur.

```
// ES5
var add = function(x, y) {
   return x + y
};
// ES6
let add = (x, y) =>  x + y
```

Arrow Functions

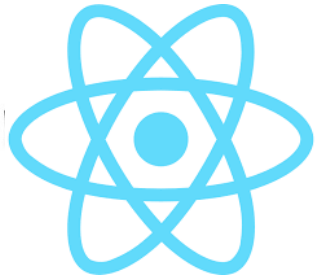


2. Arrow function'a **parametre** gönderimi aşağıdaki şekilde yapılabilir.

```
// Regular function
function regularFunction(a,b) {
    console.log(arguments)
}
regularFunction(1,2)
// Arguments[1,2]

// Arrow function
var arrowFunction = (...args) => {
    console.log(...args)
}
arrowFunction(1,2)
// 1 2
```

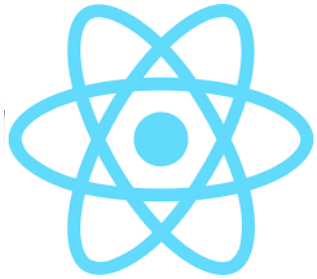
Arrow Functions



3. Arrow function'larda **this** keyword'u **kullanılamaz**. Bu nedenle object method'larında da kullanılmamalıdır (örneğin obje kaç kere tıklandı gibi…)

```
let user = {
   name: "GFG",
   gfg1:() => {
      console.log("hello " + this.name); // no 'this' binding here
   },
   gfg2(){
      console.log("Welcome to " + this.name); // 'this' binding works here
   }
 };
user.gfg1();
user.gfg2();
```

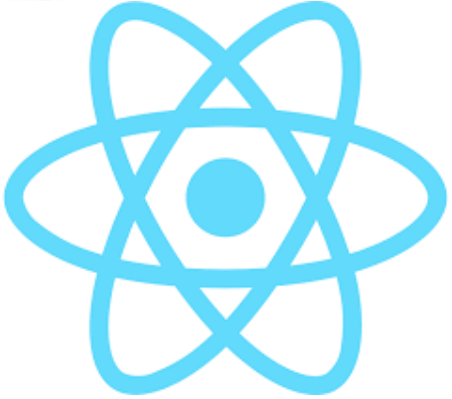
Arrow Functions



4. Arrow function'lar constructor method olarak kullanılamaz ve new keyword'ü ile çağrılamaz.

```
// Regular function
function add (x, y) {
   console.log(x + y)
}
let sum = new add(2,3);

// Arrow function
let add = (x, y) => console.log(x + y);
const sum = new add(2,4);
// TypeError: add is not a constructor
```



Teşekkürler ☺

Onur KULABAŞ

onurkulabas@yahoo.com

http://tr.linkedin.com/in/onurkulabas