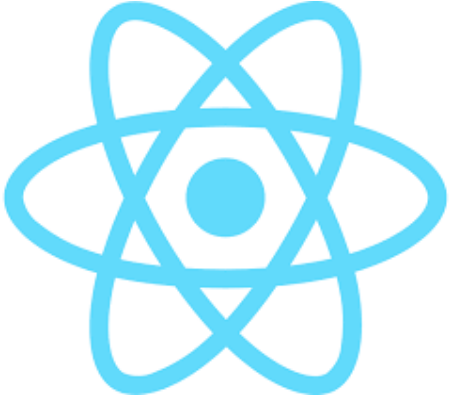


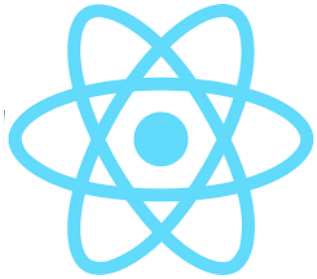
Koşullar, Döngüler, Diziler

Onur KULABAŞ

onurkulabas@yahoo.com

React Training

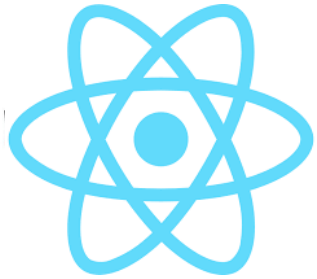
Istanbul, 2022



Koşullar

Koşullar tüm programlama dillerinde en temel kalıplardan biridir. Genelde **if – elseif – else – end** kalıbıyla, bazen de **switch : case** veya **x?y:z** gibi alternatif gösterimlerle ifade edilen ve en genel anlamıyla bir önerme (ifade) doğru ise farklı bir aksiyon alınmasını; yanlış ise farklı bir aksiyon alınmasını sağlarlar. React'da da basit kullanım örnekleri ile if – else kalıbını inceleyeceğiz.

if Kullanımı

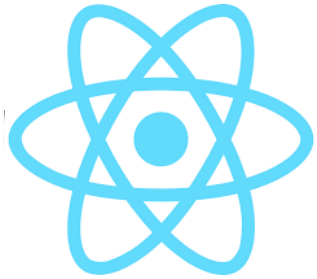


```
const myButtonClick = async() =>
    {
        if (name==="onur" && password==="1234")
        {
            console.log("OK");
            setResult("Giriş Başarılı");

            // console.log ile tarayıcı içindeki incele menusu altından console
bölümünde takip ve kontrol yapılabilir.
        }
        else
        {
            console.log("NOT OK");
            setResult("Hatalı kullanıcı adı veya şifre");
        }
    }
```

Önceki örnekteki myButtonClick fonksiyonunu yazarak devam edelim. If blokumuzda name alanı onur, password değeri 1234 ise işlemin başarılı, aksi halde başarısız olacağı gibi bir örnek senaryo üzerinden gidelim (Axios ile service call ederken gerçek bir kullanıcı adı-şifre kontrolü yapacağız).

if Kullanımı



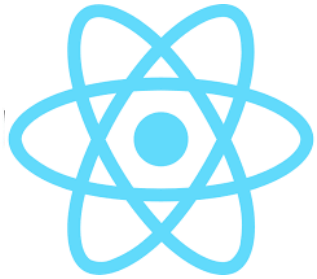
console.log satırı comment'de de yazdığımız gibi tarayıcı içindeki incele menusundeki console bölümü altından takip ve kontrol imkanı sağlamaktadır.

```
setResult("Giriş Başarılı");
```

Bu satırda ise daha önce de benzer örneğini yaptığımız gibi **setResult** objesi önceki örnekte yer alan result adlı değişkenin state'ini yönetmekte yani burada onun değerini set etmektedir. Önceki örnekte </form> ifadesi altına eklediğimiz {name}
{password} satırları yerine aşağıdaki ifadeyi koyalım.

```
<center>
  <br/>{result}
</center>
```

Böylelikle Gönder butonunun bir satır altında ve ekranın ortasına hizalı bir şekilde işlem sonucu (Giriş Başarılı / Hatalı Kullanıcı Adı veya Şifre) yazılacaktır.



== ile === Kullanım Farkı

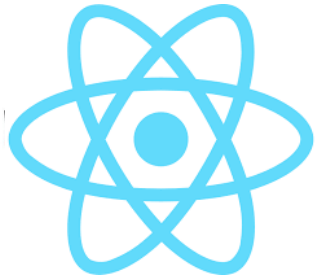
```
if (name==="onur" && password==="1234")
```

Örneğimizde == yerine === kullandık. 2 eşitlik yerine 3 eşitlik kullanmak ne demektir; farklarını inceleyelim.

- == operatörü kıyaslama işlemi öncesinde değişken değerlerini aynı türe çevirir (type coercion)
- === kıyaslama işlemi öncesinde herhangi bir dönüştürme işlemi yapmaz ve kıyaslanan 2 değerin birbirine tamamen eşitliğini kontrol eder.

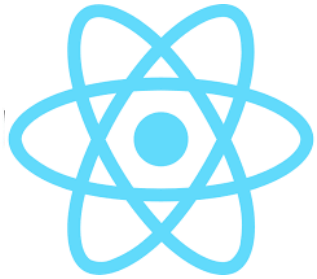
```
var one = 1;
var one_again = 1;
var one_string = "1";  // note: this is string

console.log(one ==  one_again);  // true
console.log(one === one_again);  // true
console.log(one ==  one_string); // true
console.log(one === one_string); // false
```



Döngüler

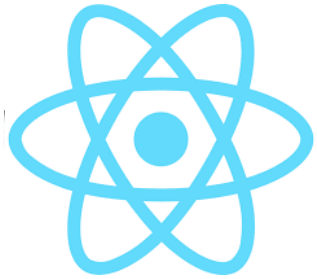
Döngüler de koşullar gibi tüm programlama dillerinde en temel kalıplardan biridir. Genelde **for, foreach, while, until** gibi kalıplarla yazılarak birden fazla değer/satırla çalışılması gereken durumlarda her bir değer veya satırda işlem yaparak sonraki değere veya satıra geçmek şekliyle çalışır. React'da farklı kullanım örnekleri ile döngüleri ve dizileri (array'leri) birlikte inceleyeceğiz.



Diziler (Arrays)

Array'ler de önceki kavramlar gibi tüm programlama dillerinde en temel kalıplardan biridir. **Aynı türde birden fazla** elemanı bir arada tutarak tek bir değişken gibi davranan ve **index** (sıra numarası) değerleri aracılığıyla her bir eleman değerini set eden veya geri döndüren yapılardır. Tek boyutlu veya çok boyutlu olabilirler. Şimdi React'da dizileri (array'leri) örnekleri ile birlikte inceleyeceğiz.

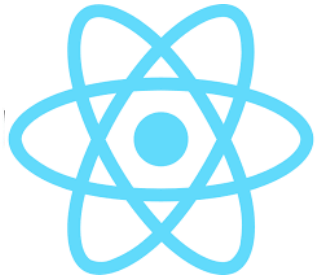
Dizi (Array) Örnek



```
const[myArray, setMyArray] = useState(["a","b"]);
```

Daha önceden tanıdığımız useState yapısını burada da kullanacağız. Bu senaryoda myArray adlı array'imizin ilk değerleri a ve b. Array'ler index numarasına diğer dillerde olduğu gibi **0'la başlarlar** ve index numaralarını köşeli parantezle alırlar. Yani burada **myArray[0]** değeri a, myArray[1] değeri ise b'dir.

Dizi (Array) Örnek



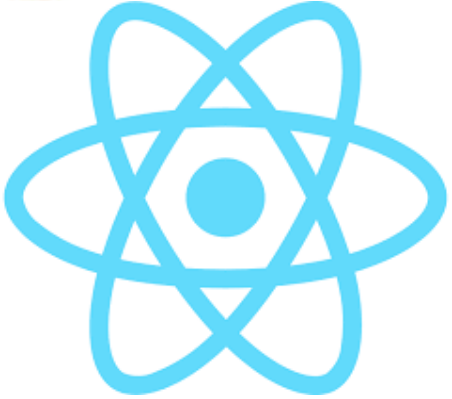
```
const[myArray, setMyArray] = useState([]);
```

Eğer useState içinde [] şeklinde kullanım sağlarsak myArray adlı array'imizin ilk değerleri boş olacaktır. Bu durumda setMyArray(…); içinden daha önceki değişkenlere değer set edildiği gibi yazmamız gerekecekti.

Dizi (Array) Örnek

```
{
    myArray.map(myArrayItem =>
        (
            <span>{myArrayItem[0]} - {myArrayItem[1]}<br/></span>
        )
    )
}
```

return ifadesinin içine yukarıdaki satırları yerleştirelim.
`myArray.map(myArrayItem => (…))` ifadesi esasen **foreach (myArrayItem in myArray) yazmak gibi**dir; biraz daha detaylı açıklarsak myArray adlı array'in tüm içeriğini myArrayItem içine tek tek index numarası vererek çağrılabilir şekilde map'ler yani eşler. Bu durumda ilk değer olarak a, ikinci değer olarak b değeri *a – b* şeklinde ekrana yansıyacaktır.



Teşekkürler ☺

Onur KULABAŞ

onurkulabas@yahoo.com

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