



Node.js ile Web Programlama Rapor - 7

Mahire Zühal Özdemir

İÇİNDEKİLER

ŞEKİLLER DİZİN	3
ES6 Object Property Shorthand ve Destructuring().....	4
ES6 Object Property Shorthand:	4
Object Destructuring:	5

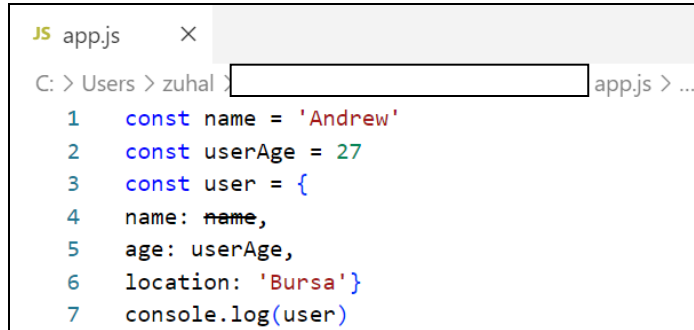
ŞEKİLLER DİZİNİ

Şekil 1: Object Shorthand	4
Şekil 2: Program Çıktısı	4
Şekil 3: Kısa Şekilde Tanımlama	4
Şekil 4: Program Çıktısı	4
Şekil 5: Object Destructuring	5
Şekil 6: Program Çıktısı	5
Şekil 7: Farklı Değişken İsmi İle Uygulama	5
Şekil 8: Yeni değişken	6
Şekil 9: Program Çıktısı	6
Şekil 10: Default Değişken	6
Şekil 11: Transaction Fonksiyonu	7

ES6 Object Property Shorthand ve Destructuring()

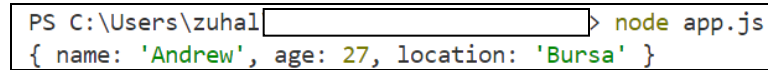
ES6 Object Property Shorthand:

Object Property Shorthand, bir nesne oluştururken bir değişkenin adını ve değerini aynı isimde kullanmaya olanak tanır. Yani, 'name: name' yerine sadece 'name' yazarak aynı sonucu elde edebilirsiniz. Bu, kodun daha kısa ve daha okunabilir olmasını sağlar.



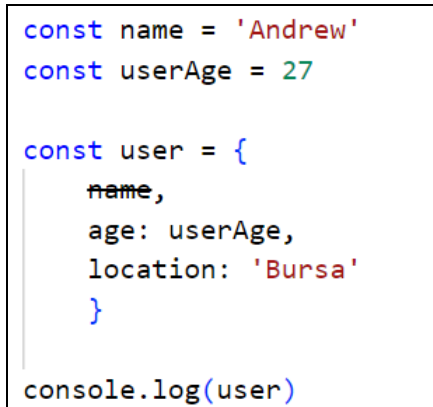
```
JS app.js  X
C: > Users > zuhal > app.js > ...
1  const name = 'Andrew'
2  const userAge = 27
3  const user = {
4    name: name,
5    age: userAge,
6    location: 'Bursa'
7  }
7  console.log(user)
```

Şekil 1: Object Shorthand



```
PS C:\Users\zuhal > node app.js
{ name: 'Andrew', age: 27, location: 'Bursa' }
```

Şekil 2: Program Çıktısı

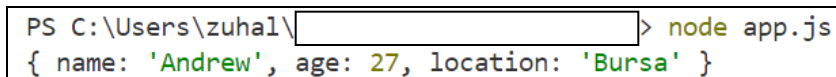


```
const name = 'Andrew'
const userAge = 27

const user = {
  name,
  age: userAge,
  location: 'Bursa'
}

console.log(user)
```

Şekil 3: Kısa Şekilde Tanımlama



```
PS C:\Users\zuhal > node app.js
{ name: 'Andrew', age: 27, location: 'Bursa' }
```

Şekil 4: Program Çıktısı

Object Destructuring:

Object Destructuring, bir nesnenin özelliklerini ayrıştırarak yeni değişkenler oluşturmayı sağlar. Bu, özellikle büyük ve karmaşık nesnelerde belirli özelliklere daha kolay erişim sağlar.

```
const product = {  
  label: 'Red notebook',  
  price: 3,  
  stock: 201,  
  salePrice: undefined  
}  
  
//const label = product.label  
//const stock = product.stock  
const {label, stock} = product  
console.log(label)  
console.log(stock)
```

Şekil 5: Object Destructuring

```
PS C:\Users\zuhal\ > node app.js  
Red notebook  
201
```

Şekil 6: Program Çıktısı

Değişken ismi değiştirilerek işlem yapılabilir.

```
const product = {  
  label: 'Red notebook',  
  price: 3,  
  stock: 201,  
  salePrice: undefined  
}  
  
//const label = product.label  
//const stock = product.stock  
const {label: productLabel, stock} = product  
console.log(productLabel)  
console.log(stock)
```

Şekil 7: Farklı Değişken İsmi İle Uygulama

Yeni değişkenler eklenerek işlem yapılabilir.

```
const product = {  
  label: 'Red notebook',  
  price: 3,  
  stock: 201,  
  salePrice: undefined  
}  
  
const {label: productLabel, stock, rating=5} = product  
console.log(productLabel)  
console.log(stock)  
console.log(rating)
```

Şekil 8: Yeni değişken

```
Red notebook  
201  
5
```

Şekil 9: Program Çıktısı

Birden fazla kez değişken tanımlanırsa default olan varsayılır.

```
const product = {  
  label: 'Red notebook',  
  price: 3,  
  stock: 201,  
  salePrice: undefined,  
  rating: 4.2  
}  
  
const {label: productLabel, stock, rating=5} = product  
console.log(productLabel)  
console.log(stock)  
console.log(rating)
```

Şekil 10: Default Değişken

Bir fonksiyon tanımlayalım: transaction. Bu fonksiyon, bir type parametresi ve bir myProduct nesnesi alıyor. Fonksiyon içinde, myProduct nesnesinden label özelliği ayrıştırılarak label adında bir yerel değişkene atanıyor.

```
const transaction = (type, myProduct) => {  
  const { label } = myProduct  
}  
  
transaction('order', product)
```

Şekil 11: Transaction Fonksiyonu

Bu şekilde, transaction fonksiyonunu çağırırken sadece istenen özelliklere doğrudan erişim sağlanabilir ve fonksiyon daha modüler hale gelir.