Furkan Kazım Çam – 21360859036 10.Hafta notları 9.Hafta raporu

Bu haftada ki derste hava durumu sayfasının API'sini düzenleyeceğiz.

App.js dosyasına gidip test için yeni bir URL yazalım.

app.get('/products', (req, res) => {

res.send({

products: []

})

})

nodemon src/app.js -e js,hbs kodunu terminale yazarak çalıştıralım. Ardından tarayıcıyı açıp localhost:3000/products URLsini girersek karşımızda statik ve değişmeyen bir sayfa görmemiz gerekir.

Belirli bir sonuç döndürmek istiyorsanız product sorgu dizesini kullanmamız gerekiyor. Sorgu dizesi URL'nin sonunda şu biçimde olmalıdır ?key=value use & to pass multiple key-value pairs like ?search=games&rating=5

Products route 'un içine req.query fonksiyonumuzu yazalım.

app.get('/products', (req, res) => {

console.log(req.query)

res.send({

products: []

})

})

localhost:3000/products?search=games&rating=5 URLsine giderseniz sonucun aynı fakat ekranda query dizisinin yazdırıldırıldığını görürsünüz.

app.get('/products', (req, res) => {

if (!req.query.search) {

res.send({

error: ‘You must provide a search term’

})

}

console.log(req.query.search)

res.send({

products: []

})

})

Localhost:3000/products ile çalıştırın.

Hata beklendiği gibi tarayıcıda görünüyor. Öte yandan, VS Kodunda ayarlanamıyor başlık istemciye mesaj gönderildikten sonra görünür. Bunun nedeni, bir gönderdikten sonra yanıt, kod yürütmeye devam eder ve başka bir yanıt gönderir. Yanıt gönderemiyoruz tek bir istek için iki kez. Bu nedenle, geri dönmemiz veya else koşulunu kullanmamız gerekir. Genellikle geri dönüş kullanılmış.

app.get(‘/products’, (req, res) => {

if (!req.query.search) {

return res.send({

error: ‘You must provide a search term’

})

}

console.log(req.query.search)

res.send({

products: []

})

})

Şimdi ise adres alınıyor adres yok ise hata döndürüyor.

```
app.get('/weather', (req, res) => {
if (!req.query.address) {
return res.send({
error: 'You must provide an address'
})
}
res.send({
forecast: 'It is snowing',
location: 'Bursa',
address: req.query.address
})
})
```

localhost:3000/weather?address=bursa şeklinde URL girip test edin. İsterseniz şehir kısmını değiştirip test edebilirsiniz.

Şuan elimizde zaten biraz geocoding ve forecast kodları var. Onları buraya kopyalayın. Bu iki dosyayı web-server/src klasörünün altına yerleştirn.Bazı modüller indireceğiz o yüzden CTRL-C ile nodemonu durdurun.

npm i request ile request modülünü indirebilirsiniz. nodemon src/app.js -e js,hbs kodunu çalıştırın. Göründüğü üzere artık geocoding ve forecast dosyalarını kullanbiliriyoruz. App.js'in üstüne

```
const geocode = require('./utils/geocode')
const forecast = require('./utils/forecast')
```

ekleyin.

```
app.get('/weather', (req, res) => {
if (!req.query.address) {
return res.send({
error: 'You must provide an address'
```

```
})
}
geocode(req.query.address, (error, { latitude, longitude, location }) => {
if (error) {
return res.send({ error })
}
forecast(latitude, longitude, (error, forecastData) => {
if (error) {
return res.send({ error })
}
res.send({
forecast: forecastData,
location,
address: req.query.address
})
})
})
res.send({
 forecast: 'It is snowing',
 location: 'Bursa',
 address: req.query.address
})
})
```

Kod satırlarını geocode'a yazın ve bursa ve izmir için test edin.

Yeni bir konuya geçiyoruz. İsmi ise default parameters. Bir test dosyası ile öğrenelim. Dosya ismi default-params.js olsun.

const greeter = (name) => {

console.log(‘Hello ’ + name)

}

greeter(‘Can’)

node default-params.js çıktıyı kontrol edin ve fonksiyonu bir kere de parametresiz çalıştırın

greeter().

Göründüğü üzere default değerimiz undefined. Name değişkeni için default değer atayalım.

const greeter = (name= ‘user’, age) => {

console.log(‘Hello ’ + name)

}

greeter(‘Can’)

greeter()

Bu şekilde default değer atamayı öğrendik. Hava durumu uygulamasına devam edelim. localhost:3000/weather?address=! şeklinde çalıştır ve hata alın. Az önce öğrendiğimiz konuyu burada uygulayarak hatayı çözelim.

geocode(req.query.address, (error, { latitude, longitude, location }) => {

kod satırını

geocode(req.query.address, (error, { latitude, longitude, location } = {}) => {

dönüştürün ve sayfayı yenileyin.

App.js'e gidin. Verileri yakalamak için fetch kullanıcağız. Fetch node.js'in bir parçası değil fakat tarayıcılarda vardır. Yazacağımız js dosyası serverda değil arayüz tarafında çalışacaktır.

```
fetch('http://puzzle.mead.io/puzzle').then((response) => {

response.json().then((data) => {

console.log(data)

})

})
```

```
fetch('http://localhost:3000/weather?address=bursa').then((response) => {

response.json().then((data) => {

if (data.error) {

console.log(data.error)

} else {

console.log(data.location)

console.log(data.forecast)

}

})

})
```

Kaydedin ve sayfayı yenileyin. Sırada ise index.hbs dosyası var. <p> etiketleri arasına aşağıdaki kodu yazın.

```
<form>

<input placeholder="Location">

<button>Search</button>

</form>
```

App.js'e geri dönebiliriz. En sona const weatherForm = document.querySelector('form') kodunu ekleyin. Son olarak ise eventlistener eklememiz gerekiyor.

```
weatherForm.addEventListener('submit', () => {

console.log('testing')

})
```

Kaydedin çalıştırın fakat hata alıcağız. Sorun index.hbs dosyasındaki js dosyasının sırası ile ilgilidir. Script etiketini headerdan silip bodynin en son kısmına yazın. Ve çalıştırın.

```
const weatherForm = document.querySelector('form')

const search = document.querySelector('input')

weatherForm.addEventListener('submit', (e) => {

e.preventDefault()

const location = search.value

fetch('http://localhost:3000/weather?address=' + location).then((response) => {

response.json().then((data) => {

if (data.error) {

console.log(data.error)

} else {

console.log(data.location)

console.log(data.forecast)

}

})

})

})
```

Kaydedin ve değer vermeden çalıştırın.

İndex.hbs dosyasında form etiketinin kapatıldığı yerin hemen altına

```
<p id="message-1"> </p>

<p id="message-2"> </p>
```

App.js dosyasına ise yeni bir sorgu ekleyelim.

```
const messageOne = document.querySelector('#message-1')

const messageTwo = document.querySelector('#message-2')
```

eventlistener fonksiyonun altına ise aşağıdaki kodları yazın.

```
e.preventDefault()

const location = search.value

messageOne.textContent = 'Loading...'

messageOne.textContent = ''

fetch('http://localhost:3000/weather?address=' + location).then((response) => {

response.json().then((data) => {

if (data.error) {

messageOne.textContent = data.error

} else {

messageOne.textContent = data.location

messageTwo.textContent = data.forecast

}

})

})
```

Çalıştırın ve test edin.

En son olarak ise butona style vermek kaldı. Styles.css dosyasına gidin. Aşağıdaki kod satırlarını ekeyin.

```
input {

border: 1px solid #cccccc;

padding: 8px;

}

button {

cursor: pointer;

border: 1px solid #888888;

background: #888888;

color: white;

padding: 8px;

}
```