WS-4

1. Express nedir?

- Express.js, Node.js için en popüler web framework'üdür. Web uygulamaları ve API'ler oluşturmak için tasarlanmıştır ve Node.js için de facto standart sunucu framework'ü olarak adlandırılmıştır.

- Node.js'de bir uygulama için sıfırdan bir backend oluşturmak sıkıcı ve zaman alıcı olabilir. Bağlantı noktalarının ayarlanmasından rota işleyicilerine kadar tüm şablon kodların yazılması, bir uygulama için iş mantığını yazmak olan asıl önemli kısımdan uzaklaştırır. Geliştiriciler Express.js gibi web çerçevelerini kullanarak zamandan tasarruf edebilir ve diğer önemli görevlere odaklanabilir.

- Express, web ve mobil uygulamalar oluşturmak için geniş özellikler sağlayan bir node.js web uygulama çerçevesidir. Tek sayfalı, çok sayfalı ve hibrit web uygulaması oluşturmak için kullanılır.

- Sunucuları ve rotaları yönetmeye yardımcı olan Node js'nin üzerine inşa edilmiş bir katmandır.

2. Express.js’in popülerligi nereden geliyor?

- Express, API'leri ve web uygulamalarını kolaylıkla yapmak için oluşturuldu
- Kodlama süresinden neredeyse yarı yarıya tasarruf sağlar ve yine de web ve mobil uygulamaları verimlidir.
- Express'i kullanmanın bir başka nedeni de javascript ile yazılmış olmasıdır, çünkü javascript daha önce hiç kullanmamış olsanız bile kolay bir dildir.
- Express, pek çok yeni geliştiricinin web geliştirme alanına girmesini sağlar.
- Node js için bir express framework’ü oluşturmanın arkasındaki neden:
    1. Zaman açısından verimli
    2. Hızlı
    3. Ekonomik
    4. Öğrenmesi kolay
    5. Asenkron

3. Hiç baktınız mı, hangi ünlü şirketler Express kullanıyor, Express'in diğer frameworkler arasındaki yeri nedir? Hadi googlelayalım.

4. Express.js ile Node.js arasındaki farklar nelerdir?

Feature	Express.js	Node.js
Kullanim	Node.js'in yaklaşım ve prensiplerini kullanarak web uygulamaları yapmaya yarar.	Sunucu-tarafli, girdi-cıktı ve olay-sürümlü uygulamalar yapmakta kullanılır.
Özelliklerin seviyesi	Node.js'den daha fazla	Daha az özellik
Uterine kurulu olduğu blok	Node.js	Google'in V8 motoru
Yazildigi dil	JavaScript	C, C++, JavaScript
Framework/Platform	Framework'ü Node.js'e dayalıdır.	Yürütme platformu veya ortamı Javascript'in sunucu-tarafli yürütülmesi icin dizayn edilmiştir.
Controllers	Controller'i vardır	Controller'i yoktur.
Routing	Routing'i vardır	Routing'i yoktur
Middleware	Sunucu-tarafindaki fonksiyonların sistematik olarak düzenlenmesi icin middleware kullanır.	Boyle bir uygulaması yoktur.
Kodlama zamanı	Daha az zamana ihtiyac duyar.	Çok zaman alır

5. Middleware nedir ve fonksiyonları nelerdir?

- Middleware fonksiyonları, istek nesnesine (req), yanıt nesnesine (res) ve uygulamanın istek-yanıt döngüsündeki bir sonraki işleve erişimi olan işlevlerdir. Sonraki fonksiyon, Express yönlendiricisindeki bir işlevdir ve çağrıldığında, geçerli ara yazılımın ardından gelen ara yazılımı çalıştırır.

- Middleware fonksiyonları aşağıdaki görevleri gerçekleştirebilir:

    1. Herhangi bir kodu yürütmek.

    2. İstek ve yanıt nesnelerinde değişiklik yapma.

    3. İstek-yanıt döngüsünü sonlandırma.

    4. Yığındaki bir sonraki ara yazılımı çağırma.

    5. Geçerli ara katman işlevi istek-yanıt döngüsünü sonlandırmazsa, kontrolü bir sonraki ara katman işlevine aktarmak için next() işlevini çağırmalıdır. Aksi takdirde, istek askıda kalacaktır.

- https://expressjs.com/en/guide/writing-middleware.html#:~:text=Middleware

6. Package.json dosyasında gördüğümüz key’lerin anlamı ne?

- https://nodesource.com/blog/the-basics-of-package-json/

7. res.send() ile res.JSON() arasındaki fark nedir?

- res.send: Verileri gönderir ve isteği sonlandırır.
- res.json: Verileri JSON biçiminde gönderir ve isteği sonlandırır.
- res.send ve res.json arasında gerçek bir fark yoktur, her iki yöntem de neredeyse aynıdır. res.json sonunda res.send'i çağırır. Yanıt olarak aktarmanız gereken bir nesne veya dizi olduğunda, bunlardan herhangi birini kullanabilirsiniz.
- Ancak res.json ve res.send arasındaki temel fark, yanıt olarak nesne olmayanları geçirmeniz gerektiğinde ortaya çıkar. res.json, aslında geçerli bir JSON olmayan nesneleri (örn. null, undefined vb.) de dönüştürürken res.send bunları dönüştürmez.

8. app.use() ile app.get() arasındaki fark nedir?

app.use() method	app.get() method
Rotaları modüler hale getirmek için kullanılabilir (diğer web uygulamalarının kullanabileceği bir npm modülünden bir dizi rotayı açığa çıkarmak gibi).	Yöntem, GET yöntemini ortaya çıkarmak için kullanılır.
Middleware'i uygulamanıza bağlamak için tasarlanmıştır. Path bir bağlama yoludur ve middleware'i yalnızca kendisiyle başlayan tüm path'leri uygulamak üzere sınırlar.	get http tarafından talep edildiğinde belirli bir rotayı eşleştirmek ve işlemek için tasarlanmıştır.
Middleware fonksiyonu, istenen path'in tabanı path ile eşleştiğinde yürütülür.	HTTP GET isteklerini belirtilen geri arama işlevleriyle belirtilen yola yönlendirir.
Bu rota ile eşleşen tüm http isteklerine izin verecektir.	Yalnızca söz konusu rotaya yönelik http GET isteklerine izin verilecektir
Syntax: app.use([path,],callback[,callback…])	Syntax: app.get(path, callback)

- https://stackoverflow.com/questions/15601703/difference-between-app-use-and-app-get-in-express-js

9. Gelen istek yöntemini ve URL'yi konsola kaydetmek için bir middleware yazın.

```
const logger = (req, res, next) => {
  console.log(`
      ${req.method}
      ${req.url} `);
      next();
  };

const express = require('express');
const app = express();
app.use(logger);
```

10. express.js'de sorgu parametresi olarak aktarılan iki sayının toplamını hesaplayan bir fonksiyon yazın.

```
const sum = (req, res) => {
    let a = req.query.firstNumber
    let b = req.query.secondNumber

    let aN = parseInt(a)
    let bN = parseInt(b)

    console.log(aN + bN);

    return res.send({
      sum: aN + bN,
    })
  }
app.get('/', sum)
```