

## Node.js ile Başlarken

### Tam yoğun uygulaması:

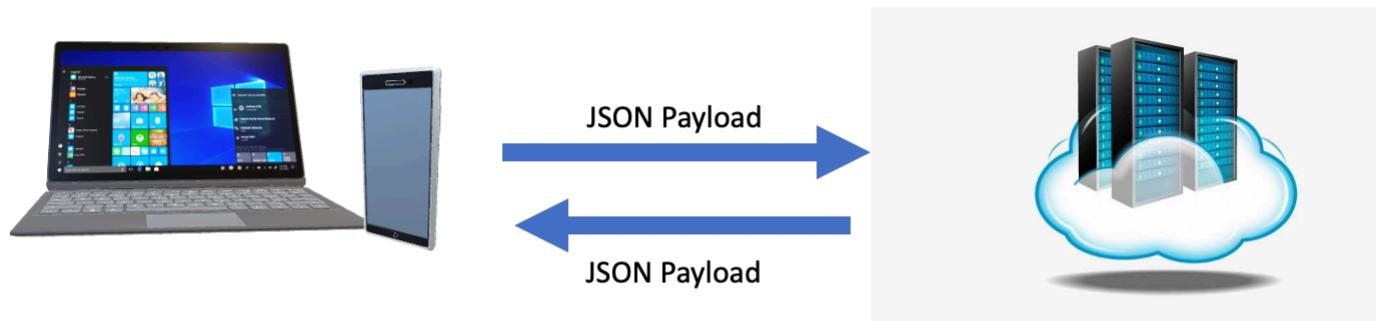
Tam yoğun uygulaması aşağıdaki bileşenleri içerir:

#### İstemci Tarafı:

Kullanıcıya yönelik web sitesi ve mobil uygulama.

#### Sunucu Tarafı:

İstemci tarafından gelen her türlü isteği işler ve yanıtları istemciye geri gönderir. Günümüzde, bulut web sunucusunu, uygulama sunucusunu ve veritabanını barındırır.



### Client

- HTML
- CSS
- Javascript
- React
- Etc.,

### Server

- Node.js
- Java
- Python
- Etc.,

### Açık Kaynak ve Çoklu Platform:

JavaScript, HTML sayfalarının istemci tarafında doğrulanması için ideal bir seçenekdir çünkü çok yönlü ve çoklu platform uyumluluğuna sahiptir. Kullanım kolaylığının fark edilmesiyle, JavaScript sunucu tarafi kodlaması için uyaranmıştır ve bu da Node.js'i ortaya çıkarmıştır. Node.js, bir çalışma zamanı olarak işlev görür ve ayrıca kullanım için özel lisans gerektirmez. Ayrıca, açık kaynak doğası sayesinde Node.js ile kullanılmak üzere birçok paket ve kütüphane geliştirilmiştir. Dahası, Node.js kodu, Linux, Windows ve Mac OSX gibi çeşitli işletim sistemlerinde sorunsuz bir şekilde çalışabilir.

### V8 Motoru:

Yazdığınız her bir kod parçası, işlenmesi ve makine tarafından okunabilir bir forma dönüştürülmesi gereken bir süreçten geçmelidir. Node.js alanında, JavaScript kodu Google V8 motoru kullanılarak yürütülür. Yüksek performansıyla tanınan V8, Google tarafından geliştirilen açık kaynak bir motordur ve tüm Google Chrome tarayıcılarına entegre edilmiştir. Microsoft Edge gibi modern tarayıcılar JavaScript için V8 motorunu kullanırken, Node.js sunucu tarafında V8'i kullanır.

### Olay Tabanlı, Asenkron, Engellemeden, Tek İşlemci:

Sunucu süreçleri “tek iş parçacıklı” veya “çoklu iş parçacıklı” olarak kategorize edilebilir. Tek iş parçacıklı bir ortamda, yalnızca bir komut belirli bir anda işlenirken, çoklu iş parçacıklı bir ortamda birden fazla komut aynı anda işlenebilir. Tek iş parçacıklı olmasına rağmen, Node.js asenkron ve engellemeden doğası sayesinde performans açısından öne çıkar. Bu, bir süreç yürütülürken programın tamamlanmasını beklemesi gerekmeden anlamına gelir. Node.js olay tabanlıdır; bu, ağdan okuma veya bir veritabanına veya dosya sisteme erişim gibi bir girdi/çıktı (I/O) işlemi gerçekleştirildiğinde bir olayın tetiklendiği anlamına gelir. İş parçacığını engellemek ve yanıt beklerken işlemci zamanını tüketmemek yerine, Node.js yanıt alındığında veya ilgili olay gerçekleştiğinde işlemleri yeniden başlatır. Bu engellemeden davranış, sunucunun yanıt verebilir durumda kalmasını ve çoklu görevleri eşzamanlı olarak yönetmesini sağlar, tipki çoklu iş parçacıklı bir ortam gibi.

### JSON Yükü

JSON, “anahtar-değer çiftleri” olarak biçimlendirilmiş JavaScript Nesne Notasyonu’nu ifade eder. Yük, istemci ile sunucu arasında iletilen veriyi temsil eder. İstemci sunucuya veri göndermesi gerektiğinde, bunu aşağıdaki örnekte gösterildiği gibi bir JSON nesnesi şeklinde yapar:

```
{  
  "name": "John",  
  "age": "24",  
  "email": "johnparker@gmail.com"  
}
```

Yukarıdaki nesne bir JSON’dur. Bu veri istek parçası olarak gönderildiğinde, değerleri `request.name` gibi ifadelerle kolayca çıkarılabilir.

Benzer şekilde, yanıt da JSON formatında gönderilir. Aşağıda bir örnek verilmiştir:

```
{  
  "status": "ok",  
  "message": "Successfully added"  
}
```

Node.js, bir sunucu oluşturmak için paketler sağlarken, Express framework'ü API'ler ve uç noktalar oluşturma sürecini basitleştirir. Bir API uç noktası, istemciden sunucuya yapılan bir isteğin belirli bir giriş noktasıdır.

### Sonraki adımlar

Bu kursta, Express çerçevesini kullanarak Node.js ile sunucu tarafı kodlamayı öğreneceksiniz.



# Skills Network