Asenkron Callback Fonksiyonları

- Asenkron callback fonksiyonları, bir sunucu kaynağı için gelen HTTP yöntem çağrılarını yönetmek için kullanılır.
- Node.js, tüm ağ işlemlerini bloklamadan gerçekleştirir; bu, uygulamanın ağ işleminin tamamlanmasını beklerken zaman kaybetmemesini sağlar.

HTTP İsteği ve Callback Fonksiyonu

- Uygulama, HTTP.request fonksiyonunu çağırarak uzak bir web sunucusuna istek gönderir. Node.js, HTTP yanıtını almadan önce hemen bir sonuç döner.
- HTTP yanıtı alındığında, Node.js, tanımladığınız callback fonksiyonunu çağırarak yanıt mesajını işler.

Örnek Senaryo

- Uygulama, özel bir Node.js modülünü çağırır ve bu modül de HTTP.request fonksiyonunu kullanarak uzak sunucuya istek gönderir.
- Uzak sunucu yanıtını döndüğünde, Node.js modülündeki callback fonksiyonu, yanıt mesajını konsola yazdırır.

Callback Fonksiyonları

- Node.js, asenkron bir çerçeve olduğu için callback fonksiyonları kullanarak sonuçları geri döndürür.
- Callback fonksiyonları, hata nesnesini ilk parametre olarak alır. Eğer hata tanımlıysa, bu hata işlenir ve açık bağlantılar kapatılır.

Hata Yönetimi

- Callback fonksiyonu, ilk parametre olarak bir hata nesnesi alıyorsa, hata mesajı konsola yazdırılır.
- Eğer hata yoksa, sonuç kontrol edilir ve başarılı bir şekilde tamamlanan bir işlem sonucu konsola yazdırılır.

HTTP İstekleri

- Uygulama, bir Node.js modülüne HTTP isteği gönderir. İstek başarılı bir şekilde gönderildiğinde, kontrol Node.js modülüne geri döner.
- Uzak sunucu bir HTTP yanıtı gönderdiğinde, Node.js çerçevesi, tanımlanan callback fonksiyonunu çağırır.

Callback Fonksiyonları ve Nesting (İç İçe Geçme)

- **Callback**: Bir fonksiyonun, başka bir fonksiyona argüman olarak geçirilmesiyle oluşturulan ve belirli bir sonuç alındığında çalıştırılan fonksiyonlardır. Asenkron JavaScript kodu geliştirmemize yardımcı olurlar.

- **Nesting**: Birden fazla callback fonksiyonunun birbirinin içine yerleştirilmesi, yani iç içe geçmesi durumudur. Bu, kodun okunabilirliğini ve bakımını zorlaştırır ve "Callback Hell" (Callback Cehennemi) olarak adlandırılır.

Inversion of Control (IoC)

- **Inversion of Control**: Kontrol akışının, kodunuzun dışındaki bir üçüncü tarafa devredilmesi durumudur. Bu, genellikle callback'lerin üçüncü taraf koduna güvenmeyi gerektirir, bu da hata ayıklamayı zorlaştırır.
- Örneğin, bir butona birden fazla kez tıklanması durumunda, üçüncü taraf kodunun hatalı çalışması gibi sorunlar ortaya çıkabilir.

Çözüm Yöntemleri

- Callback hell ve IoC sorunlarını azaltmak için bazı yöntemler vardır:
    - **Yorumlar yazmak**: Kodunuzu daha anlaşılır hale getirmek için.
    - **Fonksiyonları küçük parçalara ayırmak**: Daha yönetilebilir hale getirmek için.
    - **Promises**: Asenkron işlemleri daha iyi yönetmek için kullanılabilir.
    - **Async/Await**: Daha okunabilir asenkron kod yazmak için bir başka yöntemdir.

Promise Nedir?

- Promise, asenkron bir yöntem tarafından döndürülen bir nesnedir.
- Üç durumu vardır: pending (beklemede), resolved (tamamlandı) ve rejected (reddedildi).

Promise Durumları

- Başlangıçta, promise durumu "pending" (beklemede) olarak başlar. İşlem tamamlandığında "resolved" (tamamlandı) olur; bir hata oluşursa "rejected" (reddedildi) olur.
- Örneğin, bir dosya okuma işlemi başarılıysa promise "resolved" olur, başarısızsa "rejected" olur.

Asenkron İşlemler ve Axios

- HTTP istekleri senkron olarak çağrıldığında bloklayıcı olabilir. Node.js ekosisteminde, HTTP isteklerini yönetmek için promise kullanan birçok paket vardır; bunlardan biri "axios" paketidir.
- Axios, bir promise nesnesi döndürür ve istek tamamlanana kadar durumu "pending" (beklemede) olarak kalır.

Promise Kullanımı

- Promise nesnesinin "then" metodu, promise tamamlandığında çağrılır. Eğer promise reddedilirse, "catch" bloğu çalıştırılır.

- Geçerli bir URL ile çağrıldığında, promise "resolved" olur ve yanıt konsola yazdırılır. Geçersiz bir URL ile çağrıldığında, promise "rejected" olur ve hata mesajı gösterilir.

JSON Nedir?

- JSON (JavaScript Object Notation), veri değişimi için standart bir formattır ve API (Application Programming Interface) veri alışverişinde yaygın olarak kullanılır.
- Node.js, JSON ile kolayca çalışabilir.

JSON Verisini Parse Etme

- JSON verisini bir JavaScript nesnesine dönüştürmek için JSON.parse metodunu kullanabilirsiniz.
- Bir JavaScript nesnesini JSON stringine dönüştürmek için JSON.stringify() metodunu kullanılır.

JSON ve JavaScript'in Önemi

- JSON (JavaScript Object Notation), veri serileştirme için de facto standarttır. Web geliştiricileri için JavaScript ve JSON bilgisi neredeyse bir gerekliliktir.
- JavaScript ile çalışırken, nesnelerle çalışmak aslında JSON ile çalışmak gibidir. RESTful web servislerinde de yaygın olarak JSON kullanılır.

JSON'un Kullanım Alanları

- JSON, API'leri tüketmek ve kendi API'lerinizin JSON döndürmesini sağlamak için gereklidir. Bu, diğer geliştiricilerin API'lerinizi kolayca kullanabilmesini sağlar.
- JSON, Node.js projelerinde gereksinimleri belirtirken sıkça kullanılır. Ayrıca, platform olarak hizmet (PaaS) sunumları veya Kubernetes iş yükleri için de geçerlidir.

JavaScript ve JSON'un Avantajları

- JSON ile çalışmak, bulut tabanlı REST API'lerde istek ve yanıt nesnelerinin genellikle JSON formatında olmasını sağlar.
- JavaScript, internetin dili olduğu için birçok uygulamada kullanılır. Mevcut bir projeye kod eklemek veya sıfırdan JavaScript yazmak oldukça faydalıdır.

Promises Nedir?

- **Promise**, asenkron bir işlemin başarısını veya başarısızlığını temsil eden bir nesnedir. Üç durumu vardır: **pending** (beklemede), **fulfilled** (tamamlanmış) ve **rejected** (reddedilmiş).
- Promise'ler, asenkron görevleri daha okunabilir ve yönetilebilir bir şekilde zincirleme imkanı sunar.

Async/Await Kullanımı

- **Async** ve **await**, Promise'lerin üzerine inşa edilen bir sözdizimidir. Bu yapı, asenkron kodun senkron koda benzer görünmesini sağlar, böylece okunması ve yazılması daha kolay hale gelir.

- Async fonksiyonları, bir Promise döner ve await, Promise'in çözülmesini bekler.

Axios ile HTTP İstekleri

- **Axios**, tarayıcı ve Node.js için Promise tabanlı bir HTTP istemcisidir. REST uç noktalarına asenkron HTTP istekleri göndermeyi kolaylaştırır.
- Axios, JSON verilerini otomatik olarak dönüştürür ve basit bir API sunar. Ayrıca, istekleri veya yanıtları kesme, istek iptali ve kapsamlı hata yönetimi gibi özellikler sunar.