React Context API nedir? Neden tercih edilebilir?

**React Context API** bir React uygulamasında bileşenler arasında global state (durum) paylaşmayı sağlayan bir yapıdır. Normalde bir bileşenden diğerine veri aktarmak için **props** kullanırız, ancak bu, bileşen ağacının derinliklerine indikçe karmaşıklaşabilir. Context API, bu tür durumlar için daha basit bir çözüm sunar ve uygulama genelinde herhangi bir bileşenden herhangi bir bileşene veri geçişini sağlar.

**React Context API Nedir?**

**Context**, verileri tüm bileşen ağacına ya da belirli bir alt ağaç içerisine doğrudan iletmemizi sağlayan bir mekanizmadır.

Global state yönetiminde props drilling (veriyi her seferinde tek tek alt bileşenlere prop olarak geçmek) sorununu çözer.

Genellikle **küçük ya da orta ölçekli** projelerde tercih edilir.

İki ana bileşenden oluşur:

**Provider**: Veriyi sağlayan bileşen. Bütün alt bileşenlerin erişebileceği bir state tanımlar.

**Consumer**: Veriyi tüketen bileşen. Bu bileşen Provider tarafından sağlanan veriyi kullanır.

**Neden Tercih Edilebilir?**

1. **Props Drilling Sorununu Çözmek**: Bir bileşenden diğerine veri aktarmak için her seviyede prop geçmek gerekmez. Veriyi doğrudan ihtiyacı olan bileşene ulaştırabiliriz.
2. **Küçük ve Orta Ölçekli Projelerde Yeterli Olabilir**: Redux gibi daha karmaşık state yönetim araçlarına gerek kalmadan basit bir yapıyla global state yönetimi yapılabilir.
3. **Kodun Okunabilirliğini Arttırır**: Bileşenler arası veri paylaşımı daha düzenli bir hale gelir ve kod daha temiz olur.
4. **Kolay Kullanım**: Context API, React'e dahili olduğu için ekstra bir kütüphane gerektirmez ve React ile doğal bir entegrasyonu vardır.

**Dezavantajları**

**Performans Sorunları**: Çok fazla veriyi bir Context'e koyarsanız, her veri değişikliğinde tüm ilgili bileşenlerin yeniden render edilmesi gerekebilir.

**Büyük Uygulamalarda Karmaşıklık**: Uygulamamız büyüdükçe, farklı context'ler arasında karmaşıklık artabilir. Bu durumda Redux veya Zustand gibi daha optimize edilmiş state yönetim çözümleri daha uygun olabilir.

Sonuç olarak, **React Context API**, global state yönetimi için basit ve etkili bir çözüm sunar ancak büyük projelerde yetersiz kalabileceği için dikkatli kullanılmalıdır.

Provider pattern ile nasıl Get ve Set işlemleri yapılır?

React Context API kullanarak Provider pattern ile **get** ve **set** işlemlerini yapabiliriz. Bu işlemler, genellikle Context’in içinde global bir state tanımlayarak, bu state’i sağlayan bir **Provider** bileşeni ve bu state’i kullanarak **get** ve **set**işlemlerini gerçekleştiren **Consumer** bileşenler aracılığıyla yapılır.

### Provider Pattern ile Get ve Set İşlemleri

Context API ile çalışırken, bir global state ve bu state’in güncellenmesi gerektiğinde **getter** ve **setter** işlevlerini sağlayan bir yapı kurabiliriz.

#### 1. Context’i Oluştur

İlk olarak, bir Context oluşturmamız gerekiyor. Bu context, hem state’i hem de bu state’i güncelleyecek fonksiyonları tutacak.

A blue background with text

Description automatically generated

#### 2. Provider Bileşeni Oluştur

Bu adımda, global state’i sağlayan bir **Provider** bileşeni oluşturacağız. **useState** hook’unu kullanarak bir state ve bu state’i güncelleyecek bir setter fonksiyonu tanımlarız. Ardından, bu state ve setter fonksiyonunu Provider’ın value prop’u ile aşağıdaki bileşenlere aktarırız.

A computer screen shot of code

Description automatically generated

Bu örnekte:

* value global state'tir. Bunu **get** işlemi gibi düşünebiliriz.
* setValue ise global state’i güncelleyebileceğimiz fonksiyondur. Bu da **set** işlemi gibi düşünülebilir.

#### 3. Context’i Kullanarak Get ve Set İşlemleri Yap

Alt bileşenlerde, useContext hook’u kullanarak bu global state’e erişip **get** ve **set** işlemlerini yapabiliriz.

A computer screen shot of a program code

Description automatically generated

Bu örnekte:

* value değişkeni ile global state'i **get** etmiş olduk.
* setValue fonksiyonu ile state’i **set** ettik (güncelledik).
* Bir düğmeye tıklayınca setValue çağrılarak global state değişiyor.

#### 4. Uygulamayı Birleştir

Tüm bileşenleri bir arada kullanarak global state’i sağlayan bir yapı oluştururuz.

A computer screen with text and images

Description automatically generated

**Özet**

**Provider** bileşeni, bir global state ve bu state'i güncelleyen setter fonksiyonunu diğer bileşenlere sağlar.

Alt bileşenler useContext kullanarak bu state'e erişebilir (**get**) ve state'i güncelleyebilir (**set**).

Bu şekilde React Context API ile **Provider pattern** kullanarak bileşenler arasında **get** ve **set** işlemlerini gerçekleştirebiliriz.

Context API’nin sağladığı avantajlar ve kullanım senaryoları

**React Context API**, global state yönetimi için kullanılan, bileşenler arasında veri paylaşımını kolaylaştıran bir mekanizmadır. Props drilling (veriyi her bileşene prop olarak aktarmak) sorununu çözer ve genellikle küçük/orta ölçekli projelerde tercih edilir.

**Avantajlar:**

1. **Props Drilling’i Ortadan Kaldırır**: Veriyi tüm bileşen ağacında doğrudan paylaşarak, her seviyede props geçme ihtiyacını ortadan kaldırır.
2. **Basit Kullanım**: React’a entegre olduğu için öğrenmesi ve kullanması kolaydır, ek kütüphanelere gerek yoktur.
3. **Global State Yönetimi**: Redux gibi ağır çözümler yerine daha hafif bir yöntem sunar.
4. **Performans Optimizasyonu**: Gereksiz render’ları önleyerek performansı artırabilir.
5. **Modüler Yapı**: Farklı veri türleri için farklı Context’ler tanımlanabilir.

**Kullanım Senaryoları:**

* **Tema yönetimi** (light/dark mode),
* **Kullanıcı bilgileri** (oturum yönetimi),
* **Dil yönetimi** (i18n),
* **Sepet yönetimi** (e-ticaret),
* **Modal ve bildirim kontrolü**.

Context API, küçük projelerde global state yönetimini sadeleştirmek için etkili bir çözümdür.

**Zustand Nedir?**

**Zustand**, React uygulamaları için hafif ve minimal bir state yönetim kütüphanesidir. Redux gibi daha karmaşık state yönetim araçlarına alternatif olarak sunulan Zustand, basit API’si ve performans odaklı yapısıyla dikkat çeker. Zustand, global state yönetimini kolaylaştırır ve context API’ye kıyasla daha esnek ve performanslı bir yaklaşım sunar.

**Zustand**'ın avantajları:

* **Hafif ve basit**: Küçük projeler için gereksiz karmaşıklıklardan kaçınılır.
* **Reaktif**: Tüm bileşenlerin sadece ihtiyaç duyulan kısımları yeniden render edilir.
* **React hookları ile uyumlu**: useState gibi kullanımı kolaydır.
* **Side effect yönetimi** için async aksiyonlar destekler.

### ****Zustand Store’u Nasıl Dizayn Edilir?****

Zustand'da bir "store" (veri deposu), state ve state’i değiştiren fonksiyonlardan oluşur. Zustand store, bir JavaScript fonksiyonuyla oluşturulur ve bu fonksiyonda state ve set fonksiyonları tanımlanır.

#### 1. Zustand'ı Projeye Dahil Et

#### İlk olarak, Zustand kütüphanesini kuralım:

**npm install zustand**

#### 2. Basit Bir Store Oluşturma

#### Bir Zustand store’u create fonksiyonu ile oluşturulur. Bu store, state ve bu state'i güncelleyen fonksiyonları içerir.

A computer code with text

Description automatically generated

Bu örnekte:

* **count**: Global state.
* **increase**: count'u artıran fonksiyon.
* **decrease**: count'u azaltan fonksiyon.

### ****Get ve Set İşlemleri Nasıl Yapılır?****

Zustand store’unda state'e erişim (get işlemi) ve state'in güncellenmesi (set işlemi) useStore hook’u ile yapılır.

#### 3. Get İşlemi (State'e Erişim)

Store’dan state’i okumak için useStore hook’u kullanılır. Bu sayede herhangi bir bileşende state’e erişim sağlanabilir.

A computer screen shot of a code

Description automatically generated

#### 4. Set İşlemi (State'i Güncelleme)

#### State’i güncellemek için, store'da tanımladığınız güncelleme fonksiyonları kullanılır. Bu fonksiyonlara useStore ile erişebilirsiniz.

A computer screen shot of a program code

Description automatically generated

**Özet**

* **Zustand**, minimal bir state yönetim kütüphanesidir.
* Bir **store** oluşturmak için create fonksiyonu kullanılır ve state ile güncelleme fonksiyonları tanımlanır.
* State'e **get** işlemi ile erişmek için useStore hook'u kullanılır.
* State'i **set** işlemi ile güncellemek için store'daki fonksiyonlar kullanılır.

Bu şekilde Zustand ile hem state yönetimi yapılır, hem de verilerinizi uygulama genelinde kolayca paylaşıp güncelleyebiliriz.