**State Nedir?**

Bir uygulama için state, o uygulamanın tüm belleğidir diyebiliriz. Yani, uygulama sürecindeki herhangi bir veri o anki state’i temsil etmektedir.

#### State Management Nedir?

# Uygulamadaki verilerden(yani state’lerden) herhangi biri değişkenlik gösterdiği taktirde, bu veriyi/state’i kullanan tüm noktalarda bu değişikliğin yansıması gerekmektedir. İşte bu state’lerin(yani verilerin) kullanıldığı farklı noktalar üzerinde anlık değişimlerin yansımasına ise State Management denmektedir.

State Management Temel olarak üç bileşene odaklanır; store, reducer ve Provider.

**store:** Global olarak saklamak istediğimiz verileri oluşturduğumuz depo alanı.Bir JS objesidir

**reducer:** Belirlenen global statelerin güncellemesinden sorumlu fonksiyondur. Güncel state değerini ve tetiklenirken gönderilen parametreleri argüman olarak dönen bir fonksiyondur.

**Provider:** Component’lerimizi sarmallayan (bir nevi yöneten) context bileşeni

State management yapılarına örnek olarak;

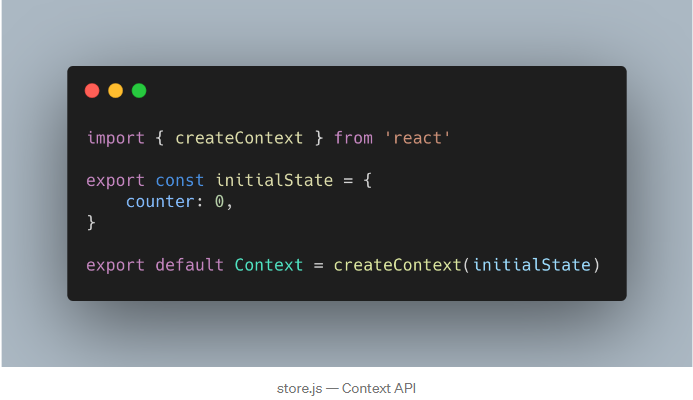
* [Redux](https://redux.js.org/)
* [Mobx](https://mobx.js.org/README.html)
* [Flux](https://facebook.github.io/flux/docs/overview)
* [Recoil](https://recoiljs.org/)

ya da React'in kendi [Context](https://reactjs.org/docs/context.html) yapısı verilebilir.

**Context API**

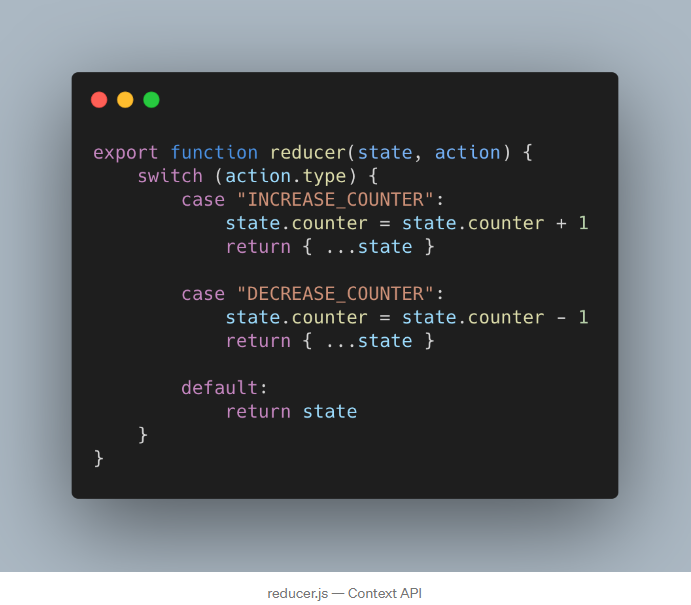
Context API yaklaşımı React’ın kendi saf context yapısını kullanarak global bir state management tasarlamaya dayanıyor. Aslında Redux’ın da temelinde de aynı dizayn mevcut fakat Redux bu yapıyı daha farklı bir şekilde işletiyor. Haliyle aslında Context API React’ın kendi “**geleneksel**” yoluyla global bir state yapısı kurmak diyebiliriz.

Kod yapısını inceleyecek olursak;



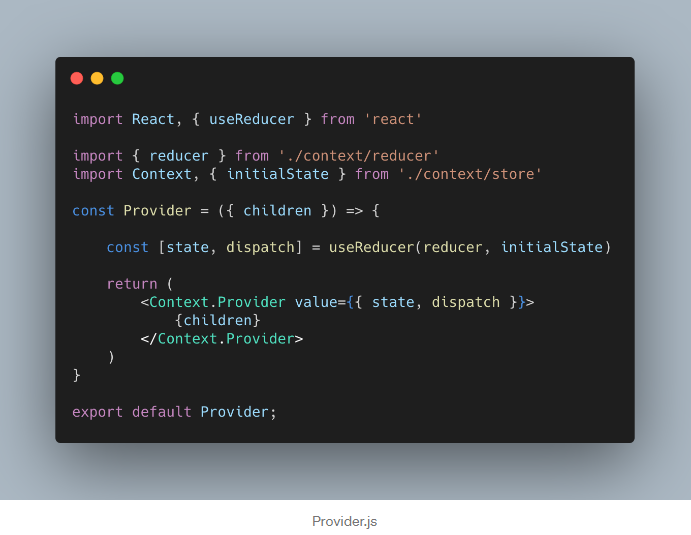
Context API için oluşturduğumuz **store**dosyamız bu şekilde. **initialState**adında oluşturduğumuz sabitimiz global olarak saklayacağımız state’lerimizi ve onların ilk değerlerini temsil ediyor.

Bir alt satırda ise Context adında bir değerimiz mevcut. React’tan çağırdığımız **createContext**metotu ile oluşturduğumuz bu değer genel olarak tasarladığımız global yapıyı kurmamıza imkan sağlayacak. İlerleyen örneklerde de göreceğiz.



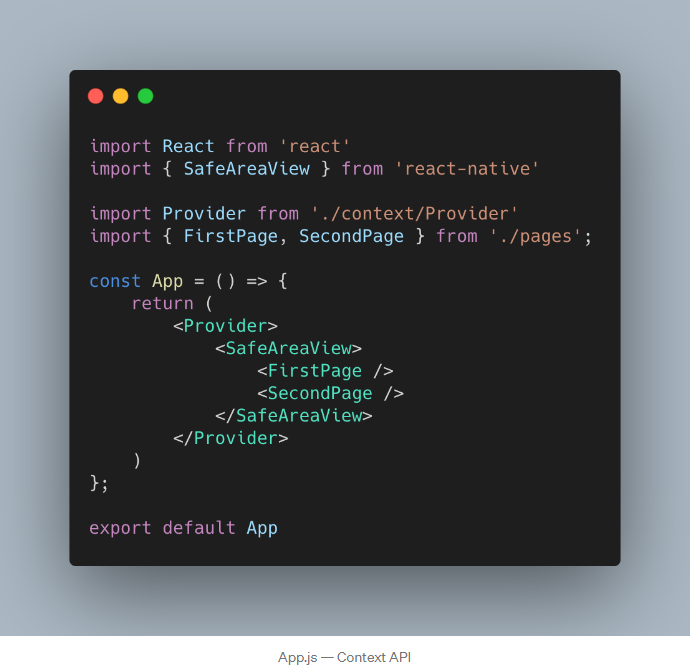
**reducer**yapısını global olarak tanımladığımız state’leri güncellemek için kurduğumuzu söylemiştik. Ekteki kodlardan anlaşıldığı üzere aslında export ettiğimiz şey “reducer” adında bir fonksiyon ve aldığı iki parametre var; **state** ve **action**.

State global olarak oluşturduğumuz state’lerin ta kendisi, action ise ilgili fonksiyonu çağırırken gönderdiğimiz parametreler. Biz eğer bir fonksiyona bir parametre göndereceksen buna **action.parametreADI**şeklinde erişebiliyoruz.



Son olarak oluşturduğumuz yapımız ise Provider component’imiz. Burada dikkat etmemiz gereken şey **useReducer** çağrısı. Bu hooks metotu Provider’a sağlayacağımız **state** ve **dispatch** değerlerini oluşturuyor ve aldığı parametreler ise state güncellemesi için yazdığımız reducer fonksiyonu ve store dosyamızda oluşturduğumuz initailState objesi. Bu fonksiyonun “state” ve “dispatch” geri dönüşü bileşenlerimizden global statelere erişmek için kullanacağımız değerlerin ta kendisi aslında.

Context yapımız şu noktada tamamlanmış oluyor. Geriye sadece Provider yapısını sarmallamak ve istediğimiz sayfadan Context bağlantısını yapmak kalıyor.



Uygulamamızın başlangıç sayfası ekte görüldüğü gibi. FirstPage ve SecondPage adını verdiğimiz componentleri birbirinden bağımsız iki sayfa olarak düşünebiliriz. **Herhangi bir props paylaşımı yapmıyorlar.**Haliyle ortak olarak kullandıkları hiç bir veri mevcut değil. Biz, her iki sayfadan da global state’imiz olan **counter**’a erişim yapacağız.



FirstPage adlı componentimiz **counter**adlı global state’imizi güncellediğimiz bir sayfamız. Global yapımıza bağlanmak için iki satırlık kod bizim için yeterli. İlk olarak Context’imizi import ediyoruz ve **useContext**hook’u ile **Provider**’da sağladığımız **state** ve **dispatch**yapılarımıza erişim sağlıyoruz.

**state**global olarak tuttuğumuz state’lerimiz. store’da **initalState**adını verdiğimiz obje olarak düşünebiliriz.

**dispatch**ise reducer’larımıza erişmemizi sağlayan fonksiyon diyebiliriz. Aldığı parametre ise reducer’da ki **action**parametresine karşılık geliyor. Dikkat ettiyseniz **type**adında bir değer gönderiliyor, bu switch-case yapısında oluşturduğumuz fonksiyonların kodları (switch’in case’leri) ve bunu reducer dosyamızı kontrol edersek görebiliriz.

Text

Description automatically generated

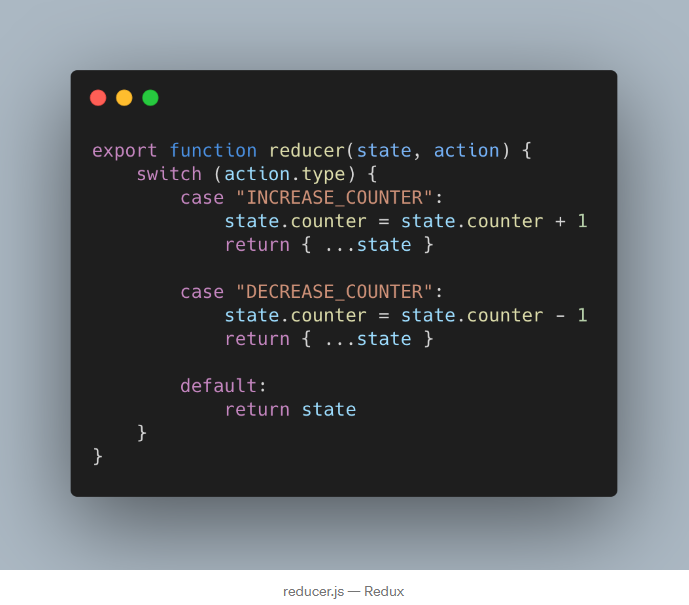
Graphical user interface, application

Description automatically generatedSecondPage’de ise sadece veri görüntülemesi yapacağız. Aynı şekilde Context’imizi çağırıp useContext ile bağlanıyor, state’e erişim sağlıyoruz.

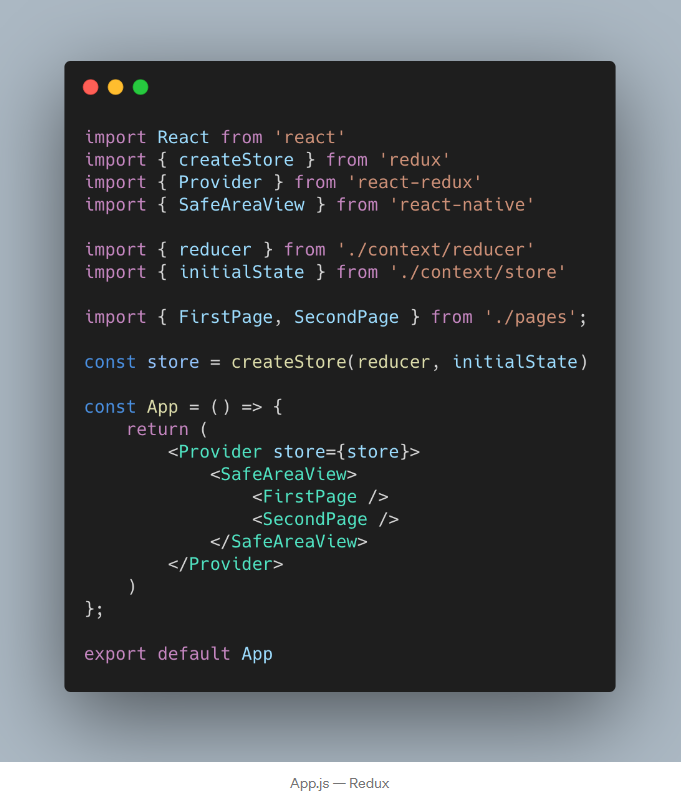
# **Redux:**

Öncelikle projemize redux ve react-redux paketlerini dahil etmemiz gerekecek. Bu paketler sayesinde context yönetimini uygulayabiliyoruz.

**store** yapımızda bu sefer sadece global olarak tutmak istediğimiz veriler mevcut. Aynı şekilde bir obje formatında tutuluyor. Bir fark yok.

****

**reducer** yapımız ise birebir aynı. Aynı uygulamayı yapacağımız için fonksiyonlarımız da aynı formatta, fakat yapısal olarak Context’ten hiç bir farklılığı yok.



Redux için ayırca bir Provider yazmaya gerek yok çünkü Context yapısından oluşturulmuş Provider’ı bize react-redux paketi otomatik olarak veriyor. Aslında buna benzer bir yapıyı Context’te de kurabilirdik, yani Provider dosyasını ayrı bir Provider.js olarak oluşturmaktansa direkt App.js’in içerisinde de <Context.Provider> şeklinde kullanabilirdik fakat biz o örnekte aradaki bağı ayırdık.

Redux ile global state kurulumu bu kadar. Gelelim sayfalardan erişimlere..

****

FirstPage component’imiz bildiğiniz üzere counter’ımızı görüntüleyip güncelleme yaptığımız bir component. Context’ten farklı olarak **state** ve **dispatch**’e ayrı ayrı erişim sağlıyoruz. Bunun için react-redux paketinden **useSelector**ve **useDispatch**hooklarını kullanıyoruz. Kullanımları ise görselimizde olduğu gibi. useSelector ile istediğimiz state’e erişimi sağlıyor, useDispatch ile fonksiyon çağrısı yapacağımız nesnemizi oluşturuyoruz

SecondPage’imiz ise bildiğiniz üzere sadece **counter**state’imizi görüntülediğimiz bir componentimiz. Herhangi bir dispatch işlemi mevcut değil.

Redux yapısının kurulumu bu kadar. Context API ile kuruduğumuz düzenin aynısını Redux ile kurmuş olduk.

.