* <MyButton />’un büyük harf ile başladığına dikkat edin. Bu onun bir React bileşeni olduğunu belirtir.
* HTML elemanları küçük harf ile başlamak zorundayken, React bileşenleri her zaman büyük harfle başlamak zorundadır.
* Biçimlendirmenizin hepsi return kelimesi ile aynı satırda değilse, biçimlendirmenizin parantez içine almak zorundasınız:
* Sınıf bileşenler yerine fonksiyonel bileşenler kullanın.
* Projelerde genellikle tek bir stil kullanılması tercih edilir. Bu nedenle, tırnak işareti seçimi genellikle ekip tarafından belirlenen kod stiline bağlıdır.
* export default bir componentin içinde olan bir çok fonksiyondan hangisinin varsayılan olarak çalıştırılacağını belirtir.
* Biçimlendirme sözdizimine JSX denir.
* React kaynakları
  + <https://react.dev/reference/react>
  + <https://react.dev/learn>
  + React typescript cheat sheat <https://react-typescript-cheatsheet.netlify.app/docs/basic/setup>
  + <https://create-react-app.dev/>
* Döndürmek isteğiniz elemanları <div>...</div> veya boş <>...</> gibi ortak bir üst eleman içine koymanız gerekmektedir:
* **Prop’lar**, JSX etiketine ilettiğiniz bilgilerdir. Örneğin, className, src, alt, width ve height, bir <img>’ye aktarabileceğiniz prop’lardan bazılarıdır.
* Prop’ları değiştiremezsiniz. Etkileşimli bir bileşen için, state ayarlaması yapmanız gerekecektir.

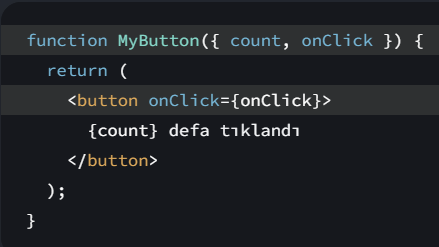


* Bileşeninizin bazı bilgileri ekran göstermek için “hatırlamasını” isteyeceksiniz. Örneğin, bir butona kaç defa tıklandığını takip etmek istiyorsunuz. Bunu yapmak için bileşeninize state ekleyin. (useState)
* const [count, setCount] = useState(0);
* useState’ten iki şey alacaksınız: mevcut state (count) ve state’i güncellemenizi sağlayan (setCount). Yaygın kullanış [something, setSomething] şeklindedir. Buton ilk defa görüntülendiğinde count değeri 0 olacaktır çünkü useState()’e 0 değerini ilettiniz.

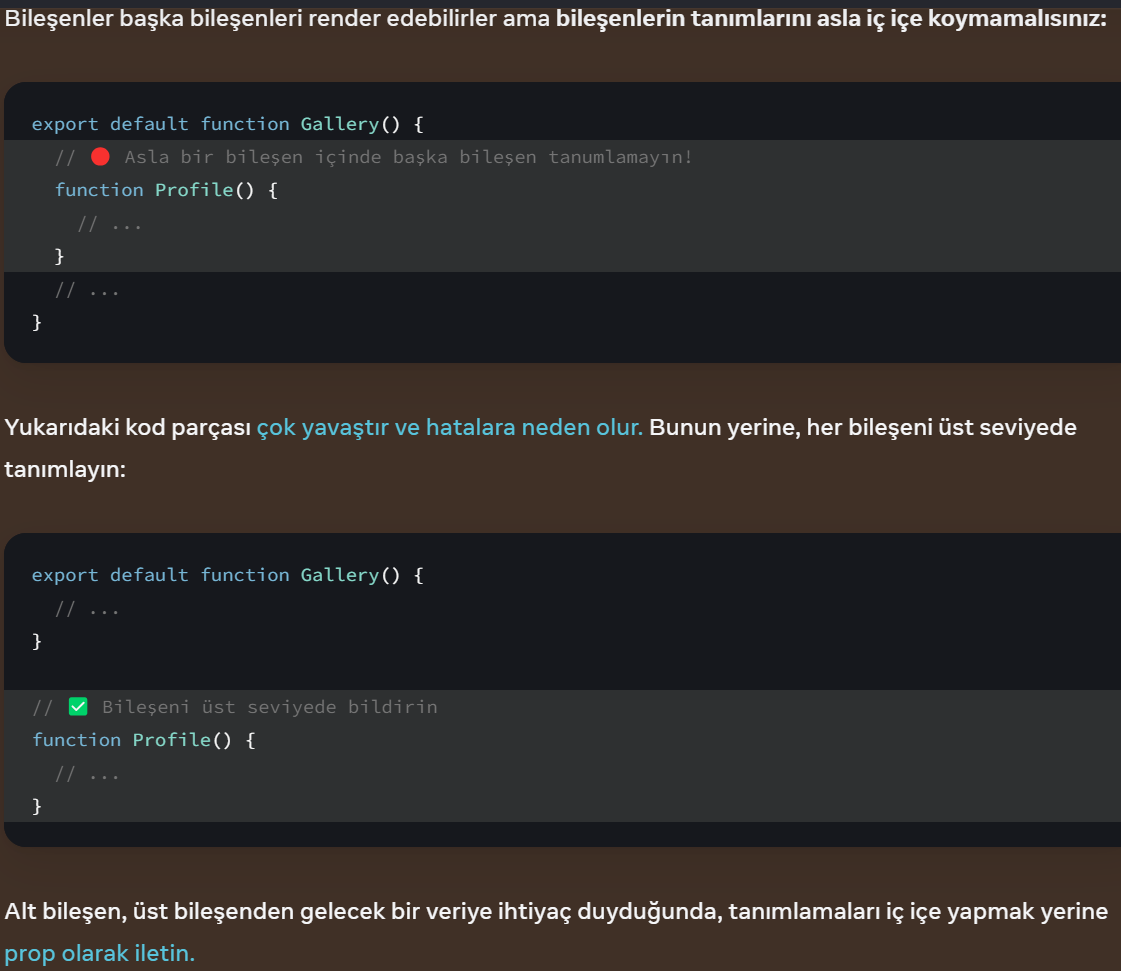


* use ile başlayan fonksiyonlara Hook denir. useState, React tarafından sağlanan bir Hook’tur.
* Eğer useState’i bir koşul ya da döngü içinde kullanmak istiyorsanız, yeni bir bileşen oluşturun ve onun içine koyun.
* Her iki MyButton bileşeninin aynı count değerini göstermesi ve aynı anda güncellenmesi için state’i bireysel butonlardan “yukarı”, hepsini içeren en yakın bileşene taşımanız gerekmektedir.
* Yeni count değeri her bir butona **prop** olarak iletilir, böylelikle hepsi yeni değeri gösterir.

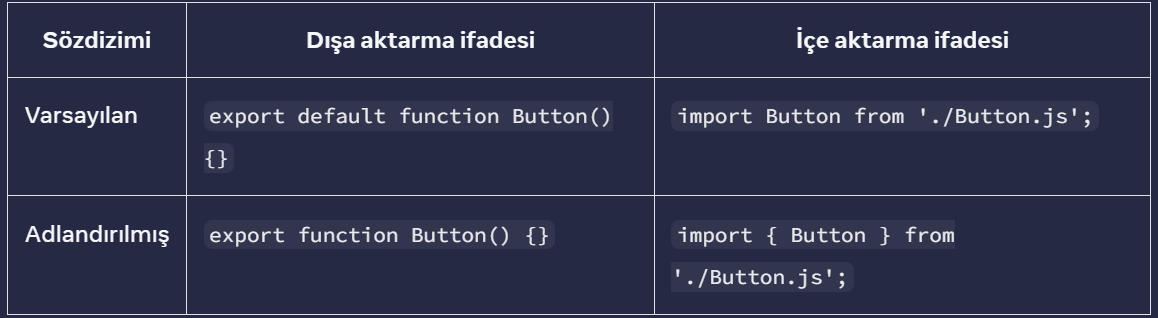




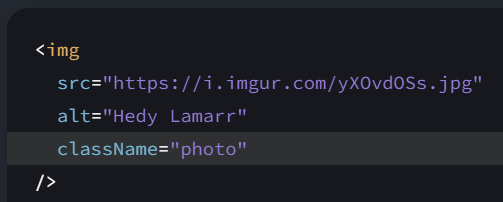
* State tanımlarken dikkat edilecek hususlar
  + Zaman içinde değişmeden mi duruyor? Eğer öyleyse, state değildir.
  + Prop’lar aracılığıyla üst bileşenden mi geliyor? Eğer öyleyse, state değildir.
  + Varolan state veya prop’lara dayalı olarak hesaplayabilir misiniz? Eğer öyleyse, kesinlikle state değildir!
* Eslint’in var olan ayarlarını kuralım
* Pretier kuralım
* useContext:
  + Bir temayı, dil ayarını veya kullanıcı bilgilerini tüm uygulama boyunca kolayca paylaşmak için kullanılabilir.
* useMemo:
  + React'te performans optimizasyonu yapmak için kullanılan bir Hook'tur. Bir bileşende hesaplanması maliyetli olan işlemlerin her render'da yeniden hesaplanmasını engelleyerek, yalnızca belirli bağımlılıkların değiştiği durumlarda bu hesaplamaların yapılmasını sağlar.
* useCallback:
  + useCallback, React'te performans optimizasyonu sağlamak amacıyla kullanılan bir Hook'tur. Özellikle bileşenlerin yeniden render edilmesi sırasında gereksiz fonksiyon yeniden oluşturulmalarını önlemek için kullanılır. useCallback, bir fonksiyonun bellekteki referansını hatırlar ve yalnızca belirli bağımlılıklar değiştiğinde bu fonksiyonu yeniden oluşturur.
* React developer tools’u kuralım
* Fonksiyonlar iç içe konulmamalıdır.



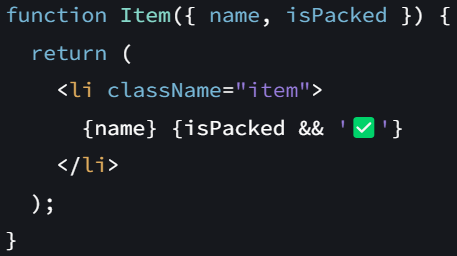
* Giderek daha fazla bileşeni iç içe yerleştirdikçe, bunları farklı dosyalara bölmeye başlamak genellikle mantıklıdır.
  + Next.js gibi dosya tabanlı yönlendirmeye sahip bir çatı (framework) kullanıyorsanız, kök bileşeniniz her sayfa için farklı olacaktır.



* Geliştiriciler eğer dosya tek bir bileşeni dışa aktarıyorsa genellikle varsayılan dışa aktarmayı, birden fazla bileşen ve değeri dışa aktarıyorsa adlandırılmış dışa aktarmayı kullanmaktadır.
  + Varsayılan ve adlandırılmış dışa aktarmalar arasındaki olası karışıklığı azaltmak için, bazı geliştirici ekipler yalnızca bir stile (varsayılan veya adlandırılmış) bağlı kalmayı veya bunları tek bir dosyada birlikte kullanmaktan kaçınmayı seçmektedir.
* JavaScript’in değişken adları konusunda sınırlamaları vardır. Örneğin, adları tire içeremez veya class gibi rezerve edilmiş sözcükler olamaz. Bu nedenle, React’te, birçok HTML ve SVG özellikleri camelCase ile yazılır. Örneğin, stroke-width yerine strokeWidth. class rezerve edilmiş sözcük olduğu için React’te, bunun yerine ilgili DOM özelliğinden sonra className yazarsınız:



* JavaScript && operatörü eğer ifademizin sol tarafı (koşulumuz) true (doğru)ise ifadenin sağ tarafındaki değeri döndürür.



* Çift süslü parantez kullanımı
  + JSX içinde bir JS nesnesi geçmek için, objeyi başka bir çift süslü parantez içine almanız gerekir: person={{ name: "Hedy Lamarr", inventions: 5 }}.
  + Bir iç içe stil gerektiğinde, nesneyi style özelliğine iç içe parantez ile iletebilirsiniz.
  + İç içe stil özellikleri camelCase kullanılarak yazılır. Örneğin, HTML’de <ul style="**background-color**: black"> yazımı bileşeninizde <ul style={{ **backgroundColor**: 'black' }}> şeklinde yazılır.

