Tepki Renk Seçici

Bu projede, butona tıklayarak rengi değiştireceksiniz. Ayrıca yaşam döngüsü yöntemlerini de kullanacaksınız.

Başlayalım!

Önce bu “src” klasörünü çalışan bir tepki yoluna koyun ve uygulamanızı çalıştırın. Bir renk seçici düğmesi göreceksiniz ama çalışmıyor.

Hedefleriniz;

• Butona tıkladığınızda sayfa rengi rastgele değişir,

• Sayfada <h1> öğesinde seçilen renkli baskılar,

• Renge göre (aydınlık olup olmamasına göre) <h1> elemanının yazı rengi siyah veya beyaz olarak değişir. Sınıfı değiştirerek bunu gerçekleştirin.

• Renge göre (aydınlık olup olmamasına göre), <button> elemanının yazı rengi ve arka plan rengi siyah veya beyaz olarak değişir. Sınıfı değiştirerek bunu gerçekleştirin.

isLight() ve formatColor() yöntemlerini değiştirme/güncelleme

1.

colorpicker.js dosyanızda; bir yapıcı yöntemi oluşturun ve color adında bir durum oluşturun. Bu renge “256”nın altındaki farklı sayıların değerlerine sahip bir dizi verin. Örnek: [42, 25, 146]

Bu, rgb renk şablonunuz olacaktır.

2.

Mevcut renginizi DOM'a yazdırın. formatColor() yöntemini kullanın.

İpucu: şöyle olmalı; “Rengininiz rgb(55, 42, 126)”.

3.

isLight() yöntemini kullanarak <h1> öğe sınıf adınızı değiştirin, .css dosyasından kullanılabilir sınıf adlarını seçin.

Elle duruma farklı renkler verin ve sonucu görün.

4.

Renginizi DOM gövdesine uygulamalısınız. Bu nedenle, bir applicationColor() yöntemi oluşturun ve rgb renk biçiminizi gövde öğesinin arka plan stiline atayın.

Bu applicationColor() yöntemini nasıl çağırabilirsiniz? Bir sonraki adıma geçildi.

5.

Farklı seçenekler olabilir;

• Bu yöntemi render içinde (dönmeden önce) çağırabilirsiniz.

• Butona tıklayarak bu metodu çağırabilir, bir butona onClick niteliği ekleyebilirsiniz,

• Bu yöntemi componentDidMount() yönteminizin içine ekleyebilirsiniz. Bu daha yaygın kullanımdır.

\*Bu yöntemleri deneyin ve farkı görün. Sonunda son seçeneği koru.

6.

applicationColor() yönteminizi componentDidMount() yönteminin içine eklediğinizde; sayfayı oluşturduktan sonra renk uygular. Renginizi elle değiştirin ve çalışıp çalışmadığını görün.

7.

Şimdi rastgele renk seçmemiz gerekiyor. Bir selectColor() yöntemi oluşturun. Rastgele renk dizisi oluşturmalı ve renk durumunu değiştirmelidir.

Sadece test etmek için: componentDidMount() yönteminden işlevinizi çağırın ve yönteminizin düzgün çalışıp çalışmadığını sayfanızı (h1 element comntent) kontrol edin.

Çalışıyorsa; componentDidMount() yönteminin içinde silin.

8.

Bu sefer metodunuzu Button bileşenine bir özellik olarak gönderin.

9.

Button.js dosyanızın içinde, button öğesine bir onClick niteliği verin ve props ile selectColor() yöntemini çağırın.

Düğmenizi deneyin rengi değiştirir

10.

Düğmeye tıklayarak durumunuzu değiştirmeyi başardınız. Ancak gövde arka plan rengini değiştirmeniz gerekir. Bu nedenle, her tıklamada applicationColor'ı çalıştırmanız gerekir.

Farklı seçenekler olabilir;

• selectColor() içinde applicationColor() yöntemini çağırabilirsiniz.

• Düğmeye tıklayarak bu yöntemi çağırabilir, bir düğmeye onClick (veya onFocus) niteliği ekleyebilirsiniz,

• Bu yöntemi componentDidUpdate() yönteminizin içine ekleyebilirsiniz. Bu daha yaygın kullanımdır.

\*Bu yöntemleri deneyin ve farkı görün. Sonunda son seçeneği koru.

11.

applicationColor() yönteminizi componentDidUpdate() yönteminin içine eklediğinizde; sayfadaki her değişiklikten sonra renk uygular. Bu yüzden her durum değişikliğinden sonra çalışması gerekir.

12.

Son olarak, düğme sınıfınızı .css dosyanızdaki uygun olanla değiştirin. Renge göre hafif mi değil mi?

Tebrikler!