



Yazılım Altyapı

Next.js Sunumu

The React Framework for the Web



Next.JS nedir ?

- Next.js, React tabanlı bir framework olup, geliştiricilere sunucu tarafında render etme (SSR) ve statik site üretimi (SSG) gibi modern özellikler sunarak uygulamalarının performansını ve SEO uyumluluğunu artırma olanağı sağlar. Vercel tarafından geliştirilmiştir ve özellikle hızlı, dinamik web siteleri ve uygulamalar geliştirmeyi kolaylaştırır.
- Statik Site Üretimi (SSG): Bazı sayfaları derleme aşamasında statik olarak üretebilir, bu da sayfaların daha hızlı yüklenmesini sağlar. İçerik değişmeyen sayfalar için idealdir.
- Otomatik Rota Oluşturma: Next.js, dosya tabanlı bir rota sistemine sahiptir. "pages" klasörüne eklediğiniz her dosya otomatik olarak bir sayfa rotasına dönüşür.
- Hızlı Sayfa Yükleme ve Navigasyon: Next.js, istemci tarafında sayfa geçişlerini hızlı hale getiren "Link" bileşenini kullanır. Bu, SPA (Single Page Application) deneyimi sunarken performansı artırır.
- Yeniden Oluşturma (ISR): ISR (Incremental Static Regeneration), statik içeriklerin belirli aralıklarla yeniden oluşturulmasını sağlar. İçeriğiniz güncellendiğinde yeni bir yapı süreci olmadan değişiklikleri yansıtabilirsiniz.



Server-Side Rendering (SSR)

- **Nasıl Çalışır :** Server-Side Rendering, her sayfa isteğinde sunucunun sayfayı anında oluşturmasını ve istemciye göndermesini içerir. Yani kullanıcı bir sayfayı ziyaret ettiğinde, sunucu o an HTML’i render eder ve kullanıcıya gönderir. Bu işlem, sayfanın her talepte yeniden oluşturulmasını sağlar.
- **Verilerin Sunucu Tarafından Hazırlanması :** Kullanıcı sayfayı talep ettiğinde, sunucu veri tabanından veya API’den veriyi çekip sayfaya entegre eder. Böylece kullanıcıya tam bir HTML sayfası gönderilir.
- **Kullanım Durumları :** Kullanıcıya özel içerikler, dinamik veriye ihtiyaç duyan sayfalar, sıklıkla güncellenen ve anlık veri ihtiyacı olan sayfalar için idealdir.





Client-Side Rendering (CSR)

- **Nasıl Çalışır :** CSR, sayfa yüklenirken yalnızca temel HTML dosyasının sunucudan gelmesi ve geri kalan tüm verinin JavaScript üzerinden istemci (kullanıcı) tarayıcısında yüklenmesi anlamına gelir.
- **Next.js Projelerinde Kullanımı :** CSR, Next.js'de useEffect gibi React hook'ları ile veri getirme işlemlerinde kullanılır. Yani sayfa yüklenirken veri sunucudan çekilmez; sayfa yüklendikten sonra kullanıcı tarafından yüklenir.
- **SSG ve SSR ile Dengeleme :** Next.js projelerinde SSG veya SSR ile başlangıçta temel sayfa render edilip, CSR ile kullanıcıya spesifik veriler sayfa yüklendikten sonra sunulabilir. Böylece sayfanın hızlı açılması sağlanırken, dinamik veri de kullanıcıya ulaştırılır.
- **Avantajları :** Sayfa yüklenmesi sırasında kullanıcıya bir başlangıç HTML gönderilir ve veriler sonradan istemciye yüklenir. Bu, özellikle kullanıcının sürekli sayfa içi etkileşime geçtiği alanlar için uygundur.



Static Site Generation (SSG)

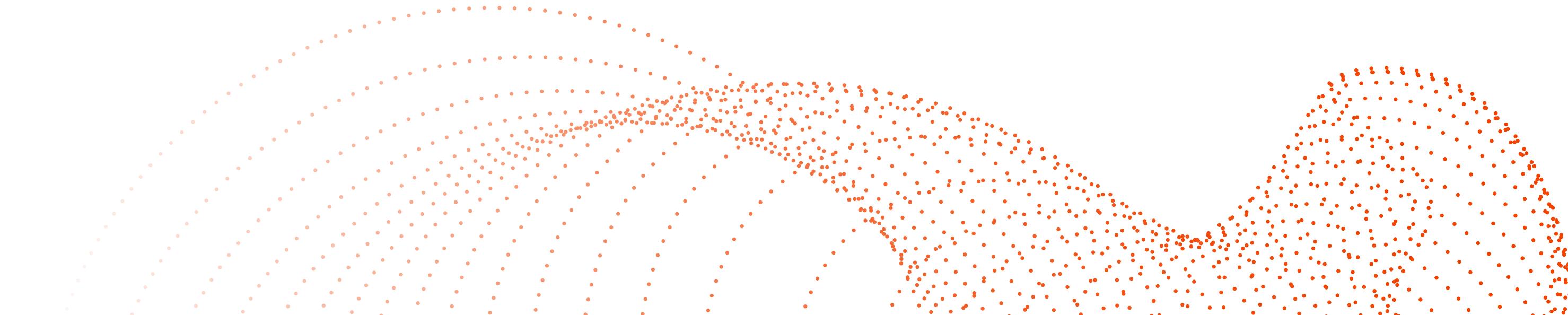
- **Nasıl Çalışır :** SSG, build (derleme) sırasında sayfaların önceden oluşturulup statik HTML dosyaları olarak depolanması anlamına gelir. Kullanıcı bir sayfaya eriştiğinde, bu statik HTML dosyası doğrudan kullanıcıya iletılır.
- **Cache ve Hız Optimizasyonu :** SSG'de sayfa içerikleri build sürecinde önceden hazırlanır ve cache'lenir. Sayfa her istek için yeniden oluşturulmaz, dolayısıyla yüksek performans sunar.
- **Kullanım Durumları :** Kullanıcıya özel içerikler, dinamik veriye ihtiyaç duyan sayfalar, sıkılıkla güncellenen ve anlık veri ihtiyacı olan sayfalar için idealdir.
- **Avantajları :** Sayfalar statik olarak hazırlandığı için yükleme süreleri çok hızlıdır. Özellikle değişimyen veya nadiren güncellenen içerikler için çok uygundur.





Incremental Static Regeneration (ISR)

- **Nasıl Çalışır :** ISR, SSG'nin statik yapılarını güncelleyebilir hale getirir. Next.js, belirli aralıklarla (örn: her 60 saniyede bir) sayfanın güncellenmiş bir kopyasını oluşturur ve yeni kullanıcı taleplerine bu güncel kopyayı sunar.
- **Veri Tutarlılığı ve Güncellik Sağlama :** ISR, hem statik içeriklerin avantajlarını kullanmak hem de içeriklerin güncel kalmasını sağlamak için kullanılır. Bu sayede içerik her zaman güncel olur ama SSG'nin hız avantajı da korunur.
- **Avantajları :** Sayfanın her istekle güncellenmesine gerek kalmaz, yalnızca belirli aralıklarda yenilenir. Bu, sunucu yükünü azaltır ve yüksek trafikli sitelerde performans sağlar.





Render Yöntemi	Kullanım Durumu	Avantajlar	Dezavantajlar
SSR	Dinamik içerik gerektiren, SEO odaklı sayfalar	Hızlı içerik sunumu, SEO dostu	Yüksek sunucu maliyeti, her istekte veri çekilmesi
SSG	Statik ve nadiren değişen içerikler	Hızlı yükleme, SEO dostu	İçerik güncellemeleri için siteyi yeniden build etme gerekliliği
ISR	Sık güncellenmesi gereken ancak her istek için dinamik veri çekmek istemeyen sayfalar	Statik hızda, periyodik olarak güncellenmiş içeri	Veri güncellemeleri belirli bir süreye bağlıdır
CSR	Kullanıcı etkileşimine dayalı, dinamik içerikler	Anlık veri gösterimi, esnek yapı	İlk yükleme performansı, SEO desteği yok



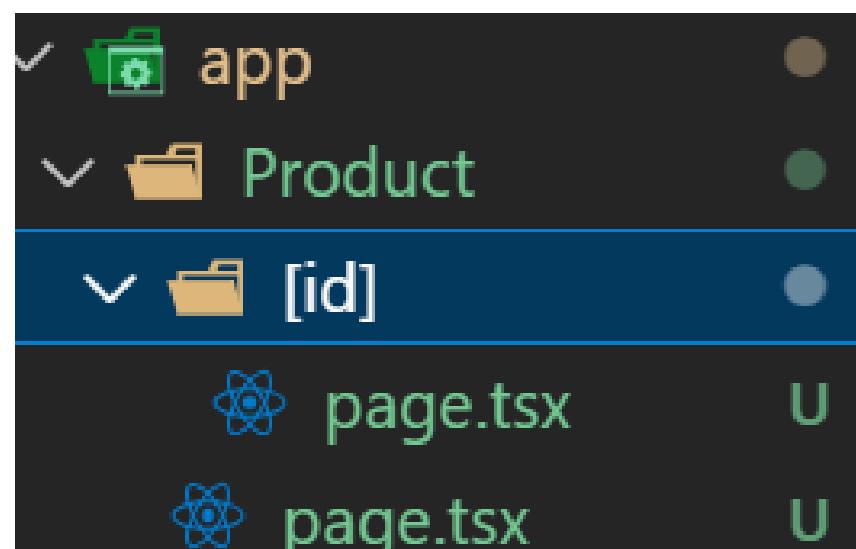
Next.JS Routing

- **Nasıl Çalışır :** Next.js'teki routing yapısı, dosya tabanlı yönlendirme (file-based routing) prensibine dayanır. Bu sistem, URL yapılarını doğrudan proje dosya yapısına yansıtarak, geliştiricilerin manuel olarak routing yapmasına gerek kalmadan sayfaları yönetmelerini sağlar.
- **Dosya Tabanlı Routing (File-Based Routing) :** Next.js, sayfaları dosya tabanlı bir yapı üzerinden yönlendirir. Yani, pages klasöründe oluşturduğunuz her dosya, bir rota olarak kabul edilir. Bu, geleneksel router yapılarından farklıdır çünkü URL yapısını doğrudan dosya adlarına bağlar.
- Sayfa Dosyaları: pages klasöründeki her .js, .ts, .jsx, .tsx dosyası, bir URL rotasına karşılık gelir. Örneğin:
 - pages/page.tsx → / (Ana sayfa)
 - app/Hakkimizda/page.tsx → /Hakkimizda
 - pages/Blog/page.tsx → /Blog



Dinamik Routing

- **Nasıl Çalışır :** Next.js, dinamik rotalar için de kolay bir yapı sunar. Dinamik URL parametrelerini almak için, dosya adında köşeli parantezler [] kullanılır. Bu, URL'deki değişkenleri yakalamanızı sağlar.
- **Dinamik Sayfalar:** Örneğin, bir yazının detay sayfası için:
- App/Product/[id] → /Product/1, /Product/2 gibi dinamik rotaları işler. Burada [id] bir parametreyi temsil eder.
- Dinamik parametreleri almak için useRouter hook'u kullanılabilir





File Conventions

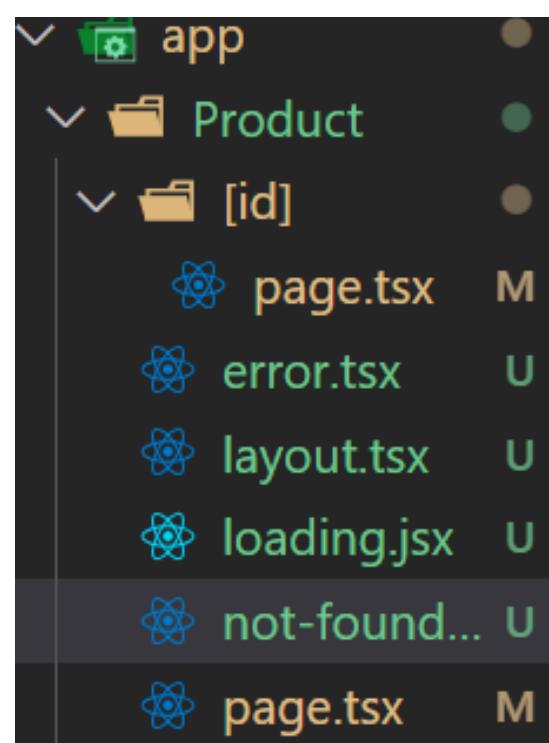
- **Nedir?** : Next.js 13 ile gelen App Router yapısında, sayfa ve bileşenlerin belirli durumlarını yönetmek için özel dosya konvansiyonları kullanılmaktadır. Bu dosyalar, uygulamanın farklı senaryolarına özgü içerikler gösterebilmek için kullanılır. Örneğin :
- **loading.js** dosyası, bir sayfa veya bileşen yüklenirken gösterilecek geçici içerikleri belirler. Bu dosya sayesinde, sayfa içerikleri yüklenirken kullanıcıya bir "Yükleniyor..." mesajı veya animasyonu gösterilebilir.
- **error.js** dosyası, sayfa veya bileşen bir hata ile karşılaştığında kullanıcıya özel bir hata mesajı veya alternatif bir içerik sunmak için kullanılır. Bu dosya, hata yönetimini daha kullanıcı dostu hale getirir.
- **not-found.js** dosyası, belirtilen sayfa veya içerik bulunamadığında gösterilecek özel bir içerik sunar.
- **layout.js** dosyası, sayfanın düzenini (layout) tanımlamak için kullanılır.





File Conventions

- **Next.js**, bu dosyaların hangi durumlarda çalışması gerektiğini bilir ve belirli bir sayfa veya bileşenle ilişkili dosyaları dinamik olarak yükler. Bu, loading.js, error.js ve not-found.js gibi dosyaların otomatik olarak çalışmasını sağlar. Uygulamanızda bu dosyaları doğru bir şekilde yerleştirmeniz yeterlidir; özel bir işlem yapmanız gereklidir.
- Özette, Next.js bu özel dosyaları otomatik olarak izler ve uygun durumlarda bu dosyaların içeriğini kullanıcıya sunar. Bu sayede, yükleme, hata ve bulunamadı gibi durumlar için her sayfa bileşeni için ayrı ayrı işlem yapmanız gereklidir.





Nested Server And Client Structure

- **Next.js** , client ve server render'ını iç içe kullanmaya olanak tanır. Bir sayfa, hem SSR (server-side rendering) hem de CSR (client-side rendering) bileşenlerini aynı anda içerebilir. Bu yapı, uygulamanın performansını optimize eder ve farklı gereksinimler için en uygun çözümü sunar.
- **Nasıl Çalışır :** Sayfanın temel yapısı ve statik veriler server-side render edilir. Bu sayede, kullanıcı sayfa yüklenliğinde hızlıca içerik görür. Dinamik, kullanıcı etkileşimi gerektiren bileşenler ise client-side render edilir. Bu sayede, kullanıcı sayfa ile etkileşime girdiğinde uygulama hızlıca yanıt verir ve yeni veriler gösterilir.
- server side bir componentin içerisinde kullandığımız client komponentte de bir server componenti kullanmak ister isek bu doğrultuda parent olan client componenti server componenti child olarak alacak şekilde güncellmemiz gerekmektedir.





State Management

- **Local State (Yerel Durum) Kullanımı**

Next.js, React tabanlı bir framework olduğu için, temel React state yönetimi ile çalışabiliriz. React'in kendi useState ve useReducer hook'ları, bileşenin içinde yerel durumları yönetmek için kullanılır.

- **Context API ile Global State Yönetimi**

React'in Context API'si, daha büyük uygulamalarda global durum yönetimi için kullanılabilir. Bu, birden fazla bileşenin aynı state'e erişebilmesini sağlar.

- **Redux ile Global State Yönetimi**

Daha büyük uygulamalarda, Redux gibi güçlü state yönetim kütüphaneleri kullanılabilir. Redux, global state'i merkezi bir şekilde yönetmek için popüler bir çözümdür.

