Next.js nedir ?

Next.js, React tabanlı bir framework'tür ve sunucu taraflı render (SSR), statik site üretimi (SSG), API yönlendirme gibi özellikler sunarak React uygulamalarını geliştirmeyi kolaylaştırır. Next.js, sayfa tabanlı rotalama sistemi ve varsayılan olarak sunucu tarafında render edilen sayfalarla, SEO dostu ve performanslı web uygulamaları oluşturmanıza olanak tanır. Ayrıca, dinamik yollar, API rotaları, otomatik kod bölme ve daha pek çok özellikle zenginleştirilmiştir.

Bu framework, modern web geliştirme ihtiyaçlarına cevap vermek için tasarlanmıştır ve projelerinizi daha hızlı ve verimli bir şekilde geliştirmenizi sağlar.

Next’i React’ten ayıran özellikler?

Next.js'i React'ten ayıran temel özellikler şunlardır:

1. **Sunucu Taraflı Render (SSR) ve Statik Site Üretimi (SSG):**
   * React, client-side (istemci tarafı) bir kütüphanedir, yani tüm render işlemleri tarayıcıda gerçekleşir. Next.js ise hem sunucu taraflı render (SSR) hem de statik site üretimi (SSG) yapabilir. Bu, sayfaların daha hızlı yüklenmesini ve daha iyi SEO performansı sağlamasını mümkün kılar.
2. **Sayfa Tabanlı Rotalama:**
   * Next.js, dosya sistemine dayalı bir sayfa yönlendirme sistemi kullanır. pages dizinine eklediğiniz her dosya otomatik olarak bir URL yolu haline gelir. React'te ise yönlendirme yapmak için manuel olarak react-router gibi bir kütüphane kurmanız gerekir.
3. **API Rotaları:**
   * Next.js, aynı proje içinde API rotaları oluşturmanıza olanak tanır. pages/api klasörüne eklenen her dosya, otomatik olarak bir API son noktası olarak hizmet eder. Bu, küçük backend işlevlerini kolayca eklemenizi sağlar.
4. **Otomatik Kod Bölme (Automatic Code Splitting):**
   * Next.js, her sayfa için bağımsız bir paket oluşturur, bu da her sayfanın yalnızca ihtiyaç duyduğu kodu yüklemesini sağlar. Bu, uygulamanızın performansını artırır. React'te bunu sağlamak için ek yapılandırmalar gerekebilir.
5. **Yerleşik CSS ve Sass Desteği:**
   * Next.js, CSS ve Sass dosyalarını doğrudan import etmenize ve kullanmanıza izin verir. React'te ise genellikle stil yönetimi için ek yapılandırmalar veya üçüncü parti kütüphaneler gereklidir.
6. **Gelişmiş Performans Optimizasyonları:**
   * Next.js, görüntü optimizasyonu, dinamik import, otomatik önbellekleme ve daha fazlası gibi performans artırıcı özellikler sunar. React'te bu tür optimizasyonlar genellikle manuel olarak yapılır.
7. **Tam Sayfa Yenileme Olmadan Yönlendirme:**
   * Next.js, Link bileşeni aracılığıyla tam sayfa yenileme olmadan sayfalar arasında geçiş yapmanızı sağlar, bu da kullanıcı deneyimini iyileştirir. Bu özellik React'te ek kütüphanelerle sağlanır.

Next.js, React üzerine eklenen bu özelliklerle, özellikle performans, SEO ve geliştirici deneyimi açısından büyük avantajlar sunar.

**React**

* **Client-Side Rendering (İstemci Tarafı İşleme):** React, bir client-side (istemci tarafı) kütüphanedir. Bu demektir ki, React ile geliştirilen bir web uygulamasında, sayfa içeriği tarayıcıda yani istemci tarafında oluşturulur. İlk olarak, tarayıcıya boş bir HTML iskeleti gönderilir, ardından JavaScript dosyaları yüklenir ve bu dosyalar tarayıcıda çalışarak sayfa içeriğini oluşturur.
  + **Sonuç:** Sayfa içeriği tamamen tarayıcıda oluşturulduğu için, sayfa yüklenmesi biraz zaman alabilir ve özellikle ilk yükleme esnasında kullanıcıya boş bir sayfa gösterilebilir. Bu da SEO açısından olumsuz bir durumdur, çünkü arama motorları sayfada içerik bulamayabilir.

**Next.js**

* **Server-Side Rendering (Sunucu Tarafı İşleme):** Next.js, sunucu taraflı render (SSR) yapabilir. Bu, sayfanın içeriğinin sunucuda oluşturulup, tarayıcıya gönderilmesi anlamına gelir. Yani, kullanıcı tarayıcıda sayfayı açtığında, sayfa içeriği halihazırda sunucu tarafından oluşturulmuş ve tarayıcıya gönderilmiş olur.
  + **Sonuç:** Bu yaklaşım, sayfanın daha hızlı yüklenmesini sağlar ve kullanıcı hemen sayfa içeriğini görebilir. Ayrıca, sayfa içeriği sunucuda oluşturulup gönderildiği için, arama motorları sayfadaki içeriği daha kolay indeksleyebilir. Bu da SEO performansını iyileştirir.
* **Static Site Generation (Statik Site Üretimi):** Next.js ayrıca statik site üretimi (SSG) de yapabilir. Bu, belirli bir sayfanın HTML'sinin build (derleme) aşamasında oluşturulup, bu statik HTML dosyasının son kullanıcıya sunulması anlamına gelir. Bu yöntemle sayfa, önceden oluşturulmuş statik bir HTML dosyası olarak sunulur.
  + **Sonuç:** Statik sayfalar, son derece hızlı yüklenir ve SEO açısından çok etkilidir, çünkü sayfa içeriği her zaman hazır ve sunulmaya hazırdır.

**Özetle:**

React, sayfa içeriğini tarayıcıda oluşturur (client-side), bu da bazı durumlarda yavaş yüklenmelere ve SEO sorunlarına yol açabilir. Next.js ise hem sunucu tarafında hem de statik olarak sayfa içeriğini oluşturabilir, bu da daha hızlı yüklenme süreleri ve daha iyi SEO performansı sağlar.

SEO (Search Engine Optimization), Türkçe karşılığıyla Arama Motoru Optimizasyonu, web sitelerinin arama motorları sonuçlarında daha üst sıralarda yer almasını sağlamak için yapılan teknik ve içeriksel iyileştirmeler bütünüdür. SEO'nun temel amacı, arama motorları tarafından daha iyi anlaşılabilir ve kullanıcılar tarafından daha kolay bulunabilir web siteleri oluşturmaktır.

**SEO'nun Temel Bileşenleri:**

1. **Anahtar Kelime Araştırması:**
   * Kullanıcıların arama motorlarına yazdığı kelime ve ifadeleri (anahtar kelimeler) belirlemek ve bu kelimelere göre içerik üretmek. Bu, arama motorlarının web sitenizi bu anahtar kelimelerle ilişkilendirmesine yardımcı olur.
2. **İçerik Optimizasyonu:**
   * Web sitenizdeki içeriklerin, anahtar kelimelerle uyumlu, kullanıcılar için değerli ve bilgilendirici olmasını sağlamak. Ayrıca, içeriklerin güncel ve özgün olması önemlidir.
3. **Teknik SEO:**
   * Web sitesinin teknik yapısını optimize etmek. Bu, sayfa hızını artırma, mobil uyumluluk sağlama, site haritası oluşturma, düzgün URL yapıları kullanma ve tarama hatalarını düzeltme gibi teknik iyileştirmeleri içerir.
4. **Backlink Oluşturma:**
   * Diğer güvenilir ve alakalı web sitelerinden kendi sitenize gelen bağlantılar (backlinkler) edinmek. Bu, arama motorlarına web sitenizin güvenilir ve otoriter olduğunu gösterir.
5. **Kullanıcı Deneyimi (UX):**
   * Kullanıcıların web sitenizde kolayca gezinmesini sağlamak. Kullanıcı dostu bir arayüz, net navigasyon, hızlı yükleme süreleri ve mobil uyumluluk, kullanıcı deneyimini iyileştirir ve bu da SEO üzerinde olumlu bir etki yaratır.
6. **Yerel SEO:**
   * Özellikle yerel işletmeler için, belirli bir coğrafi bölgede arama sonuçlarında öne çıkmak için yapılan optimizasyon çalışmalarıdır. Google My Business kaydı ve yerel anahtar kelimeler kullanımı gibi teknikleri içerir.

**SEO'nun Faydaları:**

* **Arama Motorlarında Üst Sıralara Çıkma:** İyi bir SEO stratejisi, sitenizin arama motorlarında üst sıralarda yer almasını sağlar. Bu da daha fazla ziyaretçi çeker.
* **Artan Trafik:** Arama sonuçlarında üst sıralarda yer almak, organik (ücretsiz) trafik artışına neden olur.
* **Güvenilirlik ve Otorite:** Arama motorları tarafından üst sıralarda yer almak, sitenizin güvenilir ve otoriter bir kaynak olarak algılanmasını sağlar.
* **Daha İyi Kullanıcı Deneyimi:** SEO iyileştirmeleri, genellikle daha iyi bir kullanıcı deneyimi ile sonuçlanır, bu da ziyaretçilerin sitenizde daha uzun süre kalmasını sağlar.

SEO, uzun vadeli bir strateji olup, sabır ve sürekli iyileştirme gerektirir. Ancak doğru yapıldığında, arama motorlarından gelen organik trafiği artırarak işinizin büyümesine önemli ölçüde katkı sağlar.

**React:** Kullanıcı arayüzleri (UI) oluşturmak için kullanılan bir JavaScript kütüphanesidir. Bileşen tabanlı yapısı sayesinde, dinamik ve yeniden kullanılabilir arayüzler geliştirmeyi kolaylaştırır. React, genellikle tek sayfa uygulamaları (SPA) oluşturmak için kullanılır ve istemci taraflı render (client-side rendering) yapar.

**Next.js:** React üzerine kurulu bir framework'tür ve React uygulamalarına sunucu taraflı render (SSR), statik site üretimi (SSG) ve API yönlendirme gibi ek özellikler kazandırır. Next.js, SEO dostu, hızlı yüklenen ve daha performanslı web uygulamaları oluşturmayı sağlar.

app klasörü - (auth) + layout.tsx – sign-in klasörü + layout.tsx

sig-in klasörü – page.tsx

URL'de (auth) segmenti parantez içinde olduğu için bu segment URL'de doğrudan görünmez. Next.js, parantez içindeki segmentleri opsiyonel ve gizli segmentler olarak değerlendirir. Bu durumda, URL şöyle olacaktır:

**URL Yapısı:**

* **/sign-in**: Bu URL, app/(auth)/sign-in/page.tsx dosyasına karşılık gelir.

Yani, sign-in sayfasını ziyaret etmek için tarayıcıda şu URL'yi kullanırsınız:

**http://localhost:3000/sign-in**

**(auth)** segmenti URL'de görünmez, bu nedenle doğrudan sign-in sayfasına yönlendirilirsiniz.

**Örnek Yapı:**

Eğer bir Next.js uygulamasında kök (root) yapıyı tanımlıyorsanız, bu yapı tüm sayfaların temel düzenini belirler:

plaintext

Kodu kopyala

app/

|-- (root)/

|-- layout.tsx // Tüm sayfalar için ortak düzen

|-- page.tsx // Kök sayfa içeriği

Burada (root) veya kök segment, uygulamanın tüm sayfalarına uygulanacak genel bir düzen sağlar.

**Render etmek nedir ?**

**React Gibi Framework'lerde:** React ve benzeri kütüphanelerde "render etmek", bileşenlerin veya UI (Kullanıcı Arayüzü) elemanlarının ekranda gösterilmesi anlamına gelir. React, bileşenlerin durumuna (state) ve özelliklerine (props) göre kullanıcı arayüzünü yeniden oluşturur ve günceller.

App klasörü – api

Next.js projelerinde app klasörü altında bir api klasörü genellikle API endpoint'lerini (uç noktalarını) tanımlamak için kullanılır. Bu yapı, sunucu tarafında çalışacak API'leri oluşturmanıza olanak tanır. api klasörü, Next.js’in **API Routes** özelliğini kullanarak sunucu tarafında iş mantığınızı tanımlamanıza yardımcı olur.

**api Klasörünün Amacı:**

1. **API Endpoint’leri Oluşturma:**
   * api klasörü, HTTP isteklerine yanıt verecek API endpoint’lerini tanımlamanıza olanak sağlar. Örneğin, veri almak, göndermek, güncellemek veya silmek için API uç noktaları oluşturabilirsiniz.
   * Her bir dosya veya klasör, bir endpoint olarak işlev görür. Örneğin, app/api/user/route.ts dosyası, /api/user yoluna gelen HTTP isteklerini işleyebilir.
2. **Sunucu Tarafı İşlemleri:**
   * API Routes, sunucu tarafında çalışır ve istemciden gelen istekleri işler. Bu, veritabanı sorguları yapma, üçüncü parti API'lere istek gönderme veya iş mantığı yürütme gibi işlemleri içerir.

**Örnek:**

Bir Next.js projesinde api klasörünün nasıl yapılandırılabileceğini göstermek için aşağıdaki örneği inceleyebilirsiniz:

plaintext

Kodu kopyala

app/

|-- api/

|-- user/

|-- route.ts // /api/user endpoint’i için iş mantığı

|-- posts/

|-- route.ts // /api/posts endpoint’i için iş mantığı

**1. Sayfalar (Pages):**

* **page.tsx / page.js:** URL'lere karşılık gelen sayfalar burada tanımlanır. Her dosya, bir URL yoluna karşılık gelir ve bu sayfaların içeriği page.tsx veya page.js dosyasında tanımlanır.
* **Örnek:** app/about/page.tsx dosyası /about URL'sine karşılık gelir.

**2. Layoutlar (Layouts):**

* **layout.tsx / layout.js:** Uygulamanızdaki belirli sayfalar için ortak bir düzen sağlar. İçerik bu layoutlar altında render edilir.
* **Örnek:** app/(auth)/layout.tsx dosyası, kimlik doğrulama sayfaları için ortak bir düzen sağlayabilir.

**3. API Routes (API Yönlendirmeleri):**

* **api/ Klasörü:** Sunucu tarafında çalışan API uç noktalarını tanımlar. Bu klasördeki dosyalar, HTTP isteklerine yanıt verir.
* **Örnek:** app/api/user/route.ts dosyası, /api/user yoluna gelen istekleri işler.

**4. Kök (Root) Segmentler:**

* **Parantez İçinde Klasörler (Örneğin, (auth))**: Özel düzenler ve segmentler tanımlar. Bu segmentler URL'de görünmeyebilir ve opsiyonel olabilir.
* **Örnek:** (auth) segmenti, kimlik doğrulama ile ilgili sayfalar için ortak bir düzen sağlayabilir.

**5. Statik Dosyalar ve Diğer Yapılandırmalar: bu projede ana dizinde**

* **public/ Klasörü:** Statik dosyalar (örneğin, görseller, fontlar) bu klasörde bulunur.
* **styles/ Klasörü:** Uygulamanızın CSS ve stil dosyalarını içerir.

**Parantez İçindeki Segmentler:** URL yapılarını dinamik veya opsiyonel şekilde yönetmenizi sağlar ve genellikle özel düzenler veya yapılandırmalar için kullanılır.

Next.js projelerinde, **components** (veya benzer bir isimde olan) klasörü, genellikle uygulamanın kullanıcı arayüzünü oluşturmak için kullanılan yeniden kullanılabilir bileşenleri içerir. Bu klasör, uygulamanızın farklı bölümlerinde kullanılan tüm küçük yapı taşlarını organize eder ve yönetir.

components/ |-- Button.tsx |-- Header.tsx |-- Footer.tsx

* **\*.ts:** TypeScript kodu içerir, ancak JSX kullanımı yoktur. İş mantığı, veri modelleri ve yardımcı fonksiyonlar için uygundur.
* **\*.tsx:** TypeScript kodu ve JSX içeren dosyalardır. React bileşenlerini tanımlamak ve kullanıcı arayüzü bileşenleri oluşturmak için kullanılır.

\*.tsx dosyaları, React projelerinde bileşenler ve arayüzler oluştururken \*.ts dosyaları ise iş mantığı ve veri işleme gibi görevler için kullanılır.

JSX, HTML ve JavaScript kodlarını birlikte yazmanıza izin verir ve bu, bileşenlerin oluşturulmasını ve düzenlenmesini daha okunabilir ve daha kolay hale getirir.

**JavaScript (JS)**, web sayfalarına dinamik ve etkileşimli özellikler eklemek için kullanılan bir programlama dilidir. Hem istemci tarafında (tarayıcıda) hem de sunucu tarafında (örneğin, Node.js ile) çalışabilir.

**TypeScript**, Microsoft tarafından geliştirilen bir programlama dilidir ve JavaScript'in üst kümesidir. TypeScript, JavaScript kodunun üzerine statik tip kontrolü ve diğer geliştirilmiş özellikler ekler.

metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

 **ui Klasörü:** Uygulamanızın kullanıcı arayüzü ile ilgili bileşenleri ve stillerini içerir. Bu, temel UI elemanları, stil ve tema bileşenleri, şablon bileşenleri ve yardımcı işlevleri içerebilir.

 **Amaç:** UI bileşenlerinin ve stillerinin düzenli bir şekilde saklanmasını ve yönetilmesini sağlar, böylece uygulamanızın görünümü ve hissiyatı tutarlı olur.

Constant - index.ts

 **constants Klasörü:** Uygulama genelinde kullanılan sabit değerleri ve yapılandırmaları saklar. Bu, sabit metinler, renk paletleri, API uç noktaları, konfigürasyon ayarları ve enum’ları içerebilir.

 **Amaç:** Sabit değerlerin merkezi bir yerde tutulmasını sağlar, böylece bu değerler uygulamanın farklı bölümlerinde kolayca erişilebilir ve yönetilebilir. Bu, kodun daha düzenli ve sürdürülebilir olmasını sağlar.

 **lib Klasörü:** Uygulamanızın iş mantığını destekleyen ve doğrudan kullanıcı arayüzü ile ilgili olmayan yardımcı fonksiyonlar, araçlar, API istemcileri, konfigürasyonlar ve modelleri içerir.

 **Amaç:** Kodu düzenli, modüler ve yeniden kullanılabilir kılmak için kullanılır. Bu klasör, uygulamanızın çeşitli yerlerinde tekrar eden işlevleri merkezi bir yerde toplar ve bu işlevlerin yönetimini kolaylaştırır.

 **public Klasörü:** Uygulamanızda doğrudan erişilebilen statik dosyaları içerir. Bu dosyalar tarayıcılar tarafından doğrudan yüklenebilir ve genellikle görsel ve medya dosyaları, ikonlar, favicon'lar, robots.txt gibi dosyaları içerir.

 **Amaç:** Statik dosyaların düzenli ve erişilebilir bir şekilde saklanmasını sağlar. Web sunucusunun bu dosyalara doğrudan erişmesini sağlar ve genellikle kullanıcıya sunulan içerikler için kullanılır.

 **types Klasörü:** TypeScript projelerinde veri yapılarını ve tür tanımlarını saklar. Arayüzler, tür tanımları, global tipler, veri modelleri, fonksiyon türleri ve enumerations gibi çeşitli tip tanımlarını içerir.

 **Amaç:** Kodun anlaşılmasını ve bakımını kolaylaştırır, tür güvenliğini sağlar ve uygulamanızın farklı bölümlerinde tutarlı veri yapıları kullanmanıza yardımcı olur.

**index.d.ts** dosyası, TypeScript projelerinde genellikle proje genelinde tanımlı tipleri ve arayüzleri merkezi bir noktada toplayan bir dosyadır. Bu dosya, TypeScript derleyicisine projede kullanılan türlerin ve arayüzlerin nasıl göründüğünü anlatır ve bu türlerin diğer dosyalar tarafından kolayca kullanılabilmesini sağlar.

**tsconfig.json** dosyası, TypeScript projelerinde derleyici ayarlarını ve yapılandırmasını belirleyen bir dosyadır. Bu dosya, derleyici seçeneklerini, hangi dosyaların dahil edileceğini veya hariç tutulacağını ve proje yapısını tanımlar. Özellikle, TypeScript'in nasıl çalışacağını ve projeyi nasıl derleyeceğini merkezi bir şekilde yönetir.

**tailwind.config.ts** dosyası, Tailwind CSS’in özelleştirilmesini sağlayan bir TypeScript dosyasıdır. Bu dosyada, renkler, fontlar, ekran boyutları gibi stil ayarları ve özelleştirmeler yapılır.

**sentry.server.config.ts** dosyası, Next.js uygulamasında Sentry'nin sunucu tarafı yapılandırmasını belirler. Hata izleme ve performans izleme için gerekli ayarları içerir.

**sentry.edge.config.ts** dosyası, Sentry'nin Next.js uygulamasında Edge fonksiyonları için yapılandırmasını belirler ve hata ile performans izleme işlevlerini yönetir.

**sentry.client.config.ts** dosyası, Sentry'nin Next.js uygulamasında istemci tarafında hata ve performans izleme yapılandırmasını belirler.

**postcss.config.mjs** dosyası, PostCSS’in CSS işleme eklentilerini ve yapılandırmasını belirler.

**package-lock.json** dosyası, proje bağımlılıklarının kesin sürümlerini kaydeder ve bağımlılıkların tutarlı olmasını sağlar.

**package.json** dosyası, Node.js projelerinin yapılandırmasını ve bağımlılıklarını tanımlar.

**next.config.mjs Dosyası:** Next.js uygulamanızın yapılandırma ayarlarını ve özelleştirmelerini yönetir.

**components.json Dosyası:** Bileşenlerle ilgili meta bilgileri veya yapılandırmaları içeren JSON dosyasıdır.

**.gitignore Dosyası:** Git'in takip etmemesi gereken dosya ve klasörleri listeler.

**.eslintrc.json Dosyası:** ESLint’in yapılandırma ayarlarını ve kurallarını belirler.

**.env** dosyası, uygulama yapılandırma değişkenlerini ve ortam değişkenlerini saklar. Genellikle uygulamanın çalışma ortamına özgü ayarları içerir, örneğin API anahtarları, veritabanı bağlantı bilgileri ve diğer hassas veriler.

Yeni bir Next.js projesi oluşturmak için:

1. **Node.js ve npm/yarn Yükleyin.**
2. **Proje Oluşturun:**
   * npm: npx create-next-app my-next-app
   * yarn: yarn create next-app my-next-app
3. **Proje Klasörüne Geçin:** cd my-next-app
4. **Geliştirme Sunucusunu Başlatın:**
   * npm: npm run dev
   * yarn: yarn dev

node -v   
node sürümü öğrenme

proje kurulumunda

TypeScript?

ESLint?

Tailwind CSS?

`src/` directory? No

App Router? Yes

import alias (@/\*)? No

 **TypeScript:** JavaScript’in tip kontrolü sağlayan bir üst kümesi.

 **ESLint:** JavaScript kod kalitesini ve stilini denetleyen araç.

 **Tailwind CSS:** Yardımcı sınıflarla stil oluşturmanızı sağlayan CSS framework’ü.

 **src/ Directory:** Kaynak kodlarının yer aldığı dizin (bu seçenek "Hayır" olarak belirlenmiş).

 **App Router:** Next.js'de sayfa yönlendirme ve düzenleme işlevini yöneten yapı (bu seçenek "Evet" olarak belirlenmiş).

 **Import Alias (@/\*):** Kısayol ile import yapmayı sağlar (bu seçenek "Hayır" olarak belirlenmiş).

**Tailwind CSS**, bir **CSS framework'ü** olup, kullanıcılara **yardımcı sınıflar** (utility classes) sunarak stil uygulamalarını kolaylaştırır. Tasarım süreçlerini hızlandırır ve stil uygulamalarını daha tutarlı hale getirir. Tailwind CSS, düşük seviyeli CSS sınıfları sağlar ve bu sınıflar, HTML öğelerine doğrudan uygulanır.

İlk kurulumdaki next :

.env yok

component.json yok

next-env.d.ts var

sentry.client.config.ts yok

sentry.edge.config.ts yok

sentry.server.config.ts yok

app içinde :

favicon.ico

globals.css

layout.tsx

page.tsx

public :

next.svg

vercel.svg

Next.js, güçlü bir framework olsa da, basit projeler, sadece istemci tarafı işlemler, çok büyük statik içerikler veya öğrenme eğrisi gibi faktörler göz önüne alındığında, başka çözümler daha uygun olabilir. Her proje için ihtiyaçlarınıza en uygun aracı seçmek önemlidir.