Props < Context < Redux Nedir ?

"props", "context", ve "Redux" terimleri, React gibi bir JavaScript kütüphanesi olan React ile ilgili kavramlardır.

Props: Bir component'e dışarıdan gelen verinin tutulduğu yapıdır. React bileşenleri arasında veri ve fonksiyonları aktarmak için kullanılan bir mekanizmadır. Bir bileşenin, onun içerisine yerleştirildiği üst düzey bileşen ya da uygulama tarafından sağlanan veri ve fonksiyonlar, props aracılığıyla alt bileşenlere geçirilebilir. Props, verilerin bileşenler arasında akışını ve bilgi paylaşımını kolaylaştırır.

Context: Context, React'ta verileri bileşenler ağacının derinliklerine doğrudan geçirmek için kullanılan bir mekanizmadır. Bileşenler arasında props zinciri boyunca geçmek yerine, Context API sayesinde bir bileşenden başka bir bileşene verileri aktarmak daha kolay hale gelir. Bu, verilerin bileşenler arasında doğrudan iletilmesini sağlar ve props drilling denilen durumu önlemeye yardımcı olur.

Redux gibi durum yönetimi kütüphaneleri için kullanışlıdır, ancak Redux gibi kütüphaneler daha geniş ölçekli uygulamalar için daha kapsamlı bir çözüm sunar.

Redux: Redux, React veya başka bir JavaScript kütüphanesiyle kullanılmak üzere geliştirilen, genel durum yönetimi için bir JavaScript kütüphanesidir. Redux, uygulama durumunu merkezi bir depoda (store) yönetir ve bu depoya erişim ve değişiklik yapma mekanizması sağlar. Bileşenler, Redux ile depodan durumu alabilir ve gerektiğinde değişiklik yapabilirler.

Redux, büyük ve karmaşık uygulamalarda bileşenler arasında veri iletişimini kolaylaştırır ve uygulama durumunun tutarlılığını sağlamaya yardımcı olur. Ancak, Redux kullanmak, küçük ölçekli basit uygulamalarda gereksiz karmaşıklığa neden olabilir, bu nedenle küçük projelerde kullanımı gerekmeyebilir.

Özetle, props verileri bileşenler arasında iletmek için kullanılırken, context daha derin seviyelerde veri paylaşımını kolaylaştırmak için kullanılır ve Redux, uygulama durumunun merkezi bir depoda yönetimi için bir çözümdür.

Babel Nedir ?

Babel, JavaScript kodunu farklı tarayıcılarda ve çevrelerde çalıştırmak için kullanılan bir JavaScript dönüştürücüdür (transpiler). JavaScript, her sürümünde yeni özellikler ekleyen hızla gelişen bir programlama dilidir. Ancak, tarayıcılar ve çevreler farklı JavaScript sürümlerini desteklediğinden, kodun belirli bir tarayıcıda veya çevrede çalışması, diğerlerinde çalışmasından farklılık gösterebilir.

Babel, bu sorunu çözmek için kullanılır. Ana amacı, en yeni JavaScript dil özelliklerini kullanarak kod yazmayı sağlamak ve bu kodu daha eski tarayıcılarda ve çevrelerde uyumlu hale getirmektir. Böylece, geliştiriciler en son JavaScript dil özelliklerini rahatlıkla kullanabilir ve Babel, kodu daha eski sürümlerle uyumlu hale getirerek geniş bir kullanıcı kitlesine ulaşmayı sağlar.

Babel, JavaScript kodunu dönüştürmek için plugin'ler ve preset'ler kullanır. Plugin'ler, belirli bir dönüşüm işlemini gerçekleştiren küçük eklentilerdir. Örneğin, ok fonksiyonları veya yayılım operatörü gibi özelliklerin daha eski JavaScript sürümlerine dönüştürülmesi için plugin'ler mevcuttur. Preset'ler ise belirli bir dönüşüm grubunu içeren bir dizi plugin topluluğudur ve belirli bir JavaScript sürümünü hedefleyen dönüşümleri kolaylaştırır. Babel, React, Vue, Angular gibi modern JavaScript kütüphaneleri ve çerçeveleri ile kullanılarak, geliştiricilere güncel dil özelliklerini kullanma esnekliği sunar ve uygulamaların daha geniş bir kullanıcı kitlesi tarafından erişilebilir olmasını sağlar.

React-Router Nedir ?

React Router, React tabanlı web uygulamaları için yönlendirme işlemlerini kolaylaştıran bir JavaScript kütüphanesidir. Geleneksel web uygulamalarında, kullanıcı farklı sayfalar arasında gezinirken tarayıcı adres çubuğu üzerindeki URL'ler değişir ve yeni bir sayfa yüklenir. Ancak, React gibi tek sayfa uygulamalarında (Single Page Application - SPA), tarayıcı URL'si değişse de sayfanın yeniden yüklenmemesi ve tüm içeriğin tek sayfa üzerinde dinamik olarak değiştirilmesi hedeflenir.

React Router, bu tür SPA'lar için sayfa yönlendirmesi ve gezinme işlemlerini yönetir. React Router, uygulamada bulunan bileşenleri belirli URL'lere eşleştirerek, kullanıcıların farklı sayfalar arasında gezinmelerini sağlar. React Router, temel olarak üç ana parçadan oluşur:

1) <BrowserRouter>: Tarayıcı geçmişi (browser history) kullanarak sayfa yönlendirmelerini ele alan ana bileşendir. Bu, URL'lerin değiştiği zaman uygulamada farklı bileşenlerin görüntülenmesini sağlar.

2) <Route>: Bir bileşeni belirli bir URL ile eşleştirmek için kullanılır. <Route> bileşeni, path prop'u ile belirli bir URL deseni belirler ve eşleşen URL ile ilişkilendirilen bir bileşeni render eder.

3) <Link>: Tarayıcıda gezinmek için kullanılan bir bileşendir. Kullanıcılar, <Link> bileşeni kullanarak belirli bir URL'ye tıkladığında, uygulama yeni bir sayfayı yeniden yüklemeden ilgili bileşeni görüntüler.

Axios Nedir ?

Axios, modern tarayıcılar ve Node.js üzerinde çalışan JavaScript uygulamalarında HTTP isteklerini yapmak için kullanılan popüler bir JavaScript kütüphanesidir. Axios, Promise tabanlı asenkron yapıda çalışır ve XMLHttpRequest ve Fetch API'sine alternatif olarak geliştirilmiştir. Bu sayede, sunucuyla veri alışverişi yapmak veya dış API'lere istek göndermek için kullanılabilir.

Axios, çeşitli platformlarda kullanılabilen basit ve kullanıcı dostu bir API sağlar ve isteklerin ve yanıtların işlenmesini kolaylaştırır. Axios, yüksek düzeyde yapılandırma seçenekleri sunarak, otomatik veri dönüşümleri, istek ve yanıt interceptor'leri, hata işleme gibi özelliklerin kullanımını kolaylaştırır.

Axios, popülerlik kazanmış ve JavaScript topluluğunda yaygın olarak tercih edilen bir HTTP istemci kütüphanesidir ve özellikle React, Vue ve Angular gibi çerçevelerle beraber kullanılarak web uygulamalarında veri alışverişi için yaygın olarak tercih edilmektedir.

Webpack Nedir ?

Webpack, modern web uygulamalarında kaynak dosyalarını (JavaScript, CSS, resimler, fontlar, vb.) paketleyen ve bu dosyaları optimize ederek, kullanıma hazır hale getiren açık kaynaklı bir JavaScript modül paketleme aracıdır. Webpack, uygulama geliştiricilerine birden fazla kaynak dosyasını birleştirme, dönüştürme, sıkıştırma ve optimize etme gibi işlemleri kolaylıkla yapma imkanı sağlar.

Webpack, modern JavaScript uygulamalarının yapılandırılmasını ve dağıtılmasını büyük ölçüde basitleştirir. Uygulamaların modüler hale getirilmesini ve bağımlılıklar arasındaki ilişkilerin yönetilmesini kolaylaştırır. Bu sayede, geliştiriciler daha verimli bir şekilde çalışabilir ve kodun daha hızlı yüklenmesi ve daha iyi performans elde edilmesi için uygulamanın boyutunu optimize edebilirler. Webpack'in temel özellikleri şunlardır:

- Modüllerin Desteklenmesi: Webpack, JavaScript uygulamalarının modüler bir şekilde yapılmasını ve CommonJS, AMD, ES6 modülleri gibi farklı modül sistemlerini destekler.
- Paketleme (Bundling): Webpack, birden fazla kaynak dosyasını tek bir dosya halinde birleştirir (bundler). Bu, sayfa yükleme süresini azaltır ve performansı artırır.

- Yerelleştirme (Localization): Birden fazla dil desteği sunan uygulamalarda, dil dosyalarını yönetmeyi kolaylaştırır.
- Loaders: Webpack, farklı dosya türlerini (CSS, resimler, fontlar, gibi) JavaScript ile kullanılabilir hale getiren yükleyiciler (loaders) kullanır.
- Plugins: Özel işlemler yapmak için kullanıcı tarafından oluşturulan ve Webpack ile entegre edilen eklentilerdir. Örneğin, sıkıştırma, kod analizi, hataları yönetme gibi işlemleri gerçekleştirebilirler.

Webpack, günümüzde React, Angular, Vue gibi popüler JavaScript çerçeveleriyle ve modern JavaScript araçlarıyla beraber sıkça kullanılmaktadır. Bu sayede, geliştiricilere güçlü bir yapılandırma ve dağıtım süreci sunar ve modern web uygulamalarının geliştirilmesini kolaylaştırır.

Typescript Nedir ?

TypeScript, Microsoft tarafından geliştirilen, JavaScript dilinin üst kümesi olan açık kaynaklı bir programlama dilidir. TypeScript, JavaScript'e ek olarak, statik tür sistemini ve daha fazla geliştirici aracını destekleyen bir dil olarak tasarlanmıştır. TypeScript, JavaScript'in geliştirme sürecini daha güvenli ve verimli hale getirmeyi amaçlayan bir dil olarak öne çıkar. Temel özellikleri şunlardır:

- Statiğe Dayalı Tip Sistemi: TypeScript, JavaScript'in zayıf tip sistemini güçlü bir tür sistemiyle genişletir. Bu sayede, değişkenlerin ve fonksiyonların türlerini belirleyebilir, hatalı tür kullanımlarını önleyebilir ve IDE'lerde otomatik tamamlama ve hata kontrolü gibi gelişmiş geliştirici araçlarından yararlanabilirsiniz.
- İleriye Dönük Uyumluluk: TypeScript, JavaScript'in tüm özelliklerini destekler ve JavaScript kodlarını doğrudan TypeScript kodlarına dönüştürmek mümkündür. Bu, mevcut JavaScript projelerini TypeScript'e geçişi kolaylaştırır.
- ES6 ve Sonraki Sürüm Desteği: TypeScript, ECMAScript 6 (ES6) ve sonraki sürümlerindeki özellikleri destekler. Bu sayede, en son dil özelliklerini kullanarak kod yazabilirsiniz.
- Gelişmiş Nesne Yönelimli Programlama (OOP) Desteği: TypeScript, sınıflar, miras, arayüzler ve diğer OOP kavramlarını destekler. Bu, daha büyük ve karmaşık projelerde kod organizasyonunu kolaylaştırır.
- Daha Hata Azaltma: TypeScript, statik tür kontrolü ve gelişmiş IDE entegrasyonu sayesinde hata oranını azaltır ve güvenli kod yazma sürecini kolaylaştırır.

TypeScript, özellikle büyük ölçekli uygulamalar geliştiren ekipler tarafından tercih edilir. TypeScript kodları, TypeScript Compiler (tsc) ile JavaScript'e dönüştürülür ve sonuç olarak bir JavaScript dosyası elde edilir. Bu JavaScript dosyası, modern tarayıcılar ve Node.js gibi platformlarda çalıştırılabilir.

Ecmascript Nedir ?

ECMAScript (kısaca ES), JavaScript'in resmi olarak standartlaştırılan bir versiyonudur. ECMAScript, JavaScript dilinin dil özelliklerini, syntax'ını, nesne modelini ve diğer davranışlarını tanımlayan ve belirleyen bir spesifikasyondur. ECMAScript, JavaScript dilinin evrimleşmesini ve gelişmesini yönlendiren standartlaştırma sürecidir.

JavaScript'in ilk sürümü, Netscape Communications Corporation tarafından 1995 yılında geliştirilmiştir. Daha sonra, diğer tarayıcı üreticileri de JavaScript'i desteklemeye başladılar. Ancak,

farklı tarayıcılar arasında dilin farklı versiyonlarının olması uyumluluk sorunlarına neden oldu. Bu durumda, JavaScript'in bir standartlaştırılması gereği ortaya çıktı.

ECMAScript, bu ihtiyacı karşılamak amacıyla, European Computer Manufacturers Association (Avrupa Bilgisayar Üreticileri Birliği) tarafından oluşturulan bir standartlaştırma organizasyonudur. ECMAScript spesifikasyonu, JavaScript'in dil özelliklerini belirleyerek, tüm tarayıcıların ve JavaScript motorlarının bu özellikleri takip etmelerini sağlar. Bu sayede, farklı tarayıcılarda ve ortamlarda çalışan JavaScript kodlarının tutarlılık ve uyumluluk sağlanmış olur.

ES6, JavaScript diline sınıflar, arrow fonksiyonlar, değişken sabitleri (const), temel dize şablonları, destructuring, ve Promise gibi birçok önemli dil özelliğini getirmiştir. ECMAScript sürümleri, JavaScript geliştiricilerine dildeki yenilikleri ve gelişmeleri kullanma imkanı sunar ve dilin sürekli gelişmesine katkı sağlar. JavaScript'in güncellemeleri ECMAScript spesifikasyonlarına göre yapılırken, JavaScript kodu, tarayıcılar ve diğer platformlar tarafından bu spesifikasyonları uygulamak üzere yorumlanır ve çalıştırılır.

Internatialization Nedir ?

Internationalization (kısaca i18n), bir yazılım veya web uygulamasının, farklı dil, kültür ve bölge ayarlarına sahip kullanıcıları tarafından kolayca anlaşılabilir ve kullanılabilir olması sürecidir. Internationalization, kullanıcı deneyimini geliştirmek ve kullanıcı tabanını genişletmek için uygulamaların çok dillilik ve çok kültürlülük desteği eklemesini içerir.

Internationalization işlemleri, uygulamada bulunan metinlerin, sayıların, tarihlerin, saatlerin ve diğer dil ve kültürel bağlamla ilişkili verilerin, farklı diller ve bölgeler için uygun şekilde desteklenmesini sağlar. Bu sayede, uygulama kullanıcıları kendi ana dilleri ve kültürlerine uygun olarak etkili bir şekilde etkileşime geçebilirler. Internationalization işlemleri aşağıdaki yöntemlerle gerçekleştirilir:

- Dil Desteği: Uygulama içinde kullanılan metinler ve içerikler, çevirilerinin sağlandığı dil dosyalarından alınır. Uygulama, kullanıcının tarayıcı ayarlarına veya tercihlerine göre doğru dil dosyasını yükler ve içeriklerin yerelleştirilmiş hallerini kullanıcıya sunar.
- Tarih ve Saat Biçimlendirme: Tarih ve saat değerleri, kullanıcının bölgesel ayarlarına uygun biçimlendirilir. Örneğin, Amerika'da ay/gün/yıl biçimi tercih edilirken, Avrupa'da gün/ay/yıl biçimi yaygındır.
- Para Birimi ve Sayı Biçimlendirme: Farklı bölgelerdeki para birimleri ve sayı biçimleri, kullanıcının yerel para birimine ve sayı biçimine göre düzenlenir.
- Görseller ve Grafikler: Uygulama içinde kullanılan görseller ve grafikler, kültürel farklılıklara uygun olarak değiştirilebilir.

Internationalization, bir uygulamanın global pazarlara açılmasını kolaylaştırır ve farklı dillerde ve kültürlerde daha geniş bir kullanıcı kitlesine ulaşmasına olanak sağlar. Bu nedenle, modern web ve yazılım uygulamaları genellikle internationalization desteği sunmaya önem verirler.

React Resuability Nedir ?

React reusability (yeniden kullanılabilirlik), React bileşenlerinin kodu olabildiğince genel ve modüler yaparak, aynı veya farklı projelerde tekrar kullanılabilir hale getirme yöntemidir. React, bileşen tabanlı bir JavaScript kütüphanesi olduğu için, bileşenlerin yeniden kullanılabilirliği önemli bir kavramdır. React reusability, aşağıdaki avantajları sağlar:

- Kod Tekrarını Azaltır: Bileşenleri yeniden kullanarak, aynı veya benzer işlevleri olan farklı bileşenler için aynı kodu defalarca yazmaktan kaçınılabilir. Bu, kod tabanını daha temiz ve sürdürülebilir hale getirir.
- Geliştirme Sürecini Hızlandırır: Yeniden kullanılabilir bileşenler, yeni projelerde veya farklı bölgelerde hızlı bir şekilde kullanılabilir ve geliştirme sürecini hızlandırır.
- Konsistans Sağlar: Aynı bileşenin farklı yerlerde kullanılması, tasarım tutarlılığını ve kullanıcı deneyimini artırır.
  React reusability sağlamak için şu adımlar takip edilebilir:
- Bileşenlerin Modüler Tasarımı: Bileşenleri tek bir işlevi gerçekleştirecek şekilde tasarlamak, kodun daha anlaşılır ve düzenlenir olmasını sağlar.
- Bileşenlerin Props ve State Kullanımı: Props, bileşenler arasında veri akışını sağlamak için kullanılırken, state bileşenin iç durumunu yönetmek için kullanılır. Props ve state, bileşenlerin yeniden kullanılabilirliğini artırır.
- Bileşenlerin Kapsamlı Testi: Bileşenlerin test edilebilir ve hatalardan arındırılmış olması, yeniden kullanılabilirliği sağlamak için önemlidir.
- Generic (Jenerik) Bileşenlerin Kullanımı: Generic bileşenler, çeşitli projelerde ve durumlarda kullanılabilen, çok amaçlı bileşenlerdir.
  React reusability, uygulama geliştirmeyi daha etkili ve verimli hale getirir. Uygun tasarlanmış ve yeniden kullanılabilir bileşenler, hem geliştiricilere hem de kullanıcılara daha iyi bir deneyim sunar.

Object Destructing Nedir ?

Object destructuring (nesne parçalama), JavaScript'te, bir nesnenin içerdiği özellikleri ayrıştırarak, değişkenlere atama işlemi yapmayı sağlayan bir dil özelliğidir. Bu yöntem, nesne özelliklerini daha kolay ve hızlı bir şekilde kullanmayı ve kodu daha temiz hale getirmeyi sağlar.

Nesne parçalama, özellikle büyük nesnelerde veya fonksiyonların döndürdüğü nesnelerde kullanışlıdır. Ayrıca, API çağrılarından elde edilen nesnelerin içerdiği bilgilere hızlı bir şekilde erişmek için de yaygın olarak kullanılır. Nesne parçalama, JavaScript kodunu daha kolay okunabilir ve anlaşılır hale getirirken, özellikle büyük nesnelerde kod tekrarını azaltır ve kodu daha temiz hale getirir.

Js Class Constructor Nedir ?

JavaScript sınıf yapısı (class), nesne tabanlı programlamayı destekleyen bir özelliktir. Bir sınıf, bir nesnenin yapısını ve davranışlarını tanımlayan bir şablondur. Sınıflar, JavaScript'te ES6 (ECMAScript 2015) ile birlikte tanıtılmıştır ve daha önceki sürümlerde prototip tabanlı kalıtım kullanılıyordu.

Bir JavaScript sınıfını tanımlamak için class anahtar kelimesi kullanılır. Sınıfın içinde özellikler (property) ve metotlar (method) tanımlanabilir. Sınıfın yapısını ve davranışlarını belirlemek için bir yapıcı metot da (constructor) tanımlanır.

Sınıfın yapıcı metodu (constructor), sınıfın bir nesnesi oluşturulduğunda otomatik olarak çağrılır. Yapıcı metot, nesnenin özelliklerini başlatmak için kullanılır ve nesnenin oluşturulma anında yapılacak diğer işlemleri gerçekleştirebilir. JavaScript sınıf yapısı, nesne tabanlı programlamada daha iyi yapılandırma, kalıtım ve kod organizasyonu sağlar. ES6 ile birlikte tanıtılan bu sınıf yapısı, JavaScript kodlarının daha okunabilir ve sürdürülebilir olmasını sağlar.

Js Class Super Keyword Nedir ?

JavaScript'de super anahtar kelimesi, bir alt sınıfta, üst sınıfın (ebeveyn sınıfın) yöntemlerine ve yapıcı (constructor) işlevine erişmek için kullanılan bir özelliktir. super anahtar kelimesi, ES6 (ECMAScript 2015) ile birlikte gelen sınıf tabanlı kalıtımı desteklemek için tanıtılmıştır.

super anahtar kelimesi, bir alt sınıfın yapıcısında (constructor) veya yöntemlerinde kullanılabilir. Bir alt sınıfın yapıcısı, super() metodu ile üst sınıfın yapıcısını çağırarak, üst sınıfın özelliklerini başlatmak için kullanılabilir. Ayrıca, alt sınıftaki bir yöntem içinde super.methodName() şeklinde kullanılarak, üst sınıftaki aynı isimli bir yöntemi çağırabilir. super anahtar kelimesi, alt sınıfın üst sınıfın yapıcısını veya yöntemlerini kolaylıkla çağırabilmesini sağlayarak, kalıtım ilişkisini yönetmeye yardımcı olur.

React bind ne için kullanıyoruz ?

React'ta bind işlemi, bileşen içinde kullanılan fonksiyonları, bileşenin doğru bağlamıyla (context) ilişkilendirmek için kullanılır. Bileşenlerdeki fonksiyonlar, o bileşene özgü özellikler ve durumları kullanabilir. Ancak, bu fonksiyonlar JSX içinde doğrudan kullanıldığında, bağlam (this) sorunları ortaya çıkabilir.

Bind işlemi, fonksiyonun doğru bağlamını bileşene bağlayarak bu tür bağlam sorunlarını çözmeye yardımcı olur. Böylece, fonksiyonlar doğru bileşen bağlamını kullanarak çalışır ve beklenen sonuçları elde ederiz. Arrow fonksiyonlar, otomatik olarak bağlamını koruduğu için, handleClick fonksiyonunu bind işlemi olmadan bileşene bağlamak mümkün olur. Bu yöntem daha yaygın olarak tercih edilen bir yöntemdir ve daha kısa ve okunabilir kod sağlar.

Component Did Mount

componentDidMount, React bileşenlerinin yaşam döngüsü (lifecycle) içinde yer alan bir yöntemdir. Bu yöntem, bileşenin ilk kez ekrana render edildiği zaman, yani bileşenin DOM'a monte edildiği anında çağrılır.

React bileşenlerinin yaşam döngüsü, bileşenin oluşturulduğu, güncellendiği, kaldırıldığı ve diğer durumlarda gerçekleşen olaylar dizisini ifade eder. Bu döngü boyunca, farklı yöntemler kullanılarak bileşenin belirli anlarda gerçekleşen olaylara tepki vermesi veya işlem yapması sağlanır.

componentDidMount yöntemi, bileşenin ilk kez ekrana monte edildiğinde yani DOM'a eklenirken çalıştırılması gereken kodlar için kullanılır. Genellikle, veri çağırma, abonelikler oluşturma, haritalar gibi üçüncü taraf kütüphaneleri başlatma ve diğer başlangıç işlemleri componentDidMount içinde yapılır.

componentDidMount yöntemi, genellikle bileşenin oluşturulduktan sonra bir defaya mahsus çalışacak başlangıç işlemleri için kullanılır. Bileşen oluşturulduğunda veya güncellendiğinde her seferinde çalıştırılacak işlemler için componentDidUpdate yöntemi kullanılır. Aynı şekilde, bileşen kaldırıldığında yapılacak temizlik işlemleri için componentWillUnmount yöntemi kullanılır.

Npm ile Yarn arasındaki farklar Nedir ?

Npm (Node Package Manager) ve Yarn, JavaScript ve Node.js projelerinde bağımlılık yönetimi için kullanılan iki popüler paket yönetim aracıdır. İkisi de projelerde dışa aktarılan kütüphaneleri (paketleri) yüklemek, güncellemek ve yönetmek için kullanılır. İşte Npm ve Yarn arasındaki bazı temel farklar:

- Performans ve Hız:

Yarn, projelerdeki paketleri daha hızlı bir şekilde indirir ve yükler. Paralel indirme ve kilitlenme mekanizmalarını kullanarak daha hızlı bir bağımlılık çözümleme süreci sunar.

Npm, gelişmiş sürümlerle birlikte performansını artırmış olsa da, Yarn kadar hızlı olmayabilir.

- Bağlılık Çözümleme (Dependency Resolution):

Yarn, paket bağımlılıklarını daha güvenilir ve tutarlı bir şekilde çözümler. Yarn, paketlerin bağımlılıklarını yarn.lock dosyasında saklayarak, tekrarlanabilir ve stabil bağımlılık çözümlerini destekler.

Npm, bağımlılıkları çözmek için package-lock.json dosyasını kullanır. Bu dosya, benzer bir işlevi yerine getirir, ancak Yarn'ın çözümleme mekanizmasından bazı farklılıkları vardır.

- Kullanıcı Arayüzü (CLI):

Npm ve Yarn, komut satırı arayüzleri (CLI) aracılığıyla kullanılır ve temel olarak aynı komutları destekler. Ancak Yarn, daha okunabilir ve anlaşılır çıktılar sağlayarak kullanıcı dostu bir deneyim sunar.

- Yama (Patch) Sürümleri:

Npm, npm update komutu ile yamaların güncellenmesine izin verirken, Yarn bu işlemi yapmaz. Yani, Npm'de yama güncellemeleri, Yarn'da ise yama sürümlerini güncelleme olanağı vardır.

- Cache (Önbellek) Yönetimi:

Yarn, indirilen paketleri önbellekte (cache) saklamak için daha gelişmiş bir altyapıya sahiptir ve tekrarlanan paket yüklemelerini daha hızlı hale getirir.

Npm, önbellek yönetimi için Yarn'a göre daha basit bir yaklaşım benimser.

- Proje Boyutu:

Npm, Node.js ile birlikte gelir ve projeler için doğal bir seçenek olarak düşük bir boyuta sahiptir.

Yarn, Npm'den daha büyük bir boyuta sahip olabilir, çünkü kendi ek özelliklerini içerir.

Genel olarak, Npm ve Yarn arasındaki farklar, performans, bağımlılık yönetimi, güvenilirlik ve kullanıcı deneyimi gibi alanlarda ortaya çıkar. Seçim, projenin ihtiyaçlarına, geliştirici tercihlerine ve ekosisteme bağlı olarak değişebilir. Her iki araç da aktif olarak kullanılmakta ve geliştirilmeye devam etmektedir.