Nasıl Frontend Uzmanı Olurum ?

> **Son Güncellenme Tarihi:** 29.06.2022
>

Birçok eğitim içeriğinden faydalanmanıza rağmen Frontend alanında kendinizi derinleşmiş hissetmiyorsanız, işe başladığınızda Frontend konusunda kendinizden emin olamıyorsanız, mevcut mimariyi anlamanıza rağmen kendi kodlarınızı geliştiremiyorsanız, LearnReactUI.dev
bu konudaki tüm ihtiyaçlarınızı tek bir yerde karşılayabileceğiniz bir eğitim sitesidir.

Bu site içerisinde örnekler ve anlatımlar her ne kadar React odaklı olsa da asıl hedefi UI geliştirmeyi öğretmektir. Konu anlatımları asıl işin temellerine ve mekaniğine dayalıdır, bu nedenle hangi UI kütüphanesini -React, Vue, Angular, Svelte, vb.- kullanıyor olursanız olun sitemiz üzerinden birçok konuyu dilden bağımsız olarak öğrenme imkanı bulacaksınız.

Kariyerinin henüz başında olan birçok kişinin aklında “**Nasıl bir öğrenme rotası izlemem gerekiyor?**” sorusu olabilir.

Her bireyin öğrenme yöntemi ve ihtiyaçları farklı olduğundan herkes için uygulanabilir olan hazır bir rota mevcut değildir. Piyasada konuların birbiri arkasına dizildiği, kolaydan zora belirli konulara göre gruplamaların olduğu roadmap çizimleri görebilirsiniz. Örneğin;

- https://roadmap.sh/frontend
- https://frontendmasters.com/guides/learning-roadmap/

Bu sıralı adımları izleyerek uzmanlık seviyesine ulaşmak kolay ve hızlı bir süreç gibi görünse de gerçek hayatta farklı bir düzen ile karşı karşıya kalıyoruz. Birbiri altına sıralı yapılar yerine birbirine bağlı birçok atomik yapı olduğunu ve birtakım örnekler ile, geliştirdiğiniz projeler ile bağlantılı olarak bu kavramların içeriğinin daha çok olduğunu ve bağlantıların daha da fazla güçlendiğini görebilirsiniz.

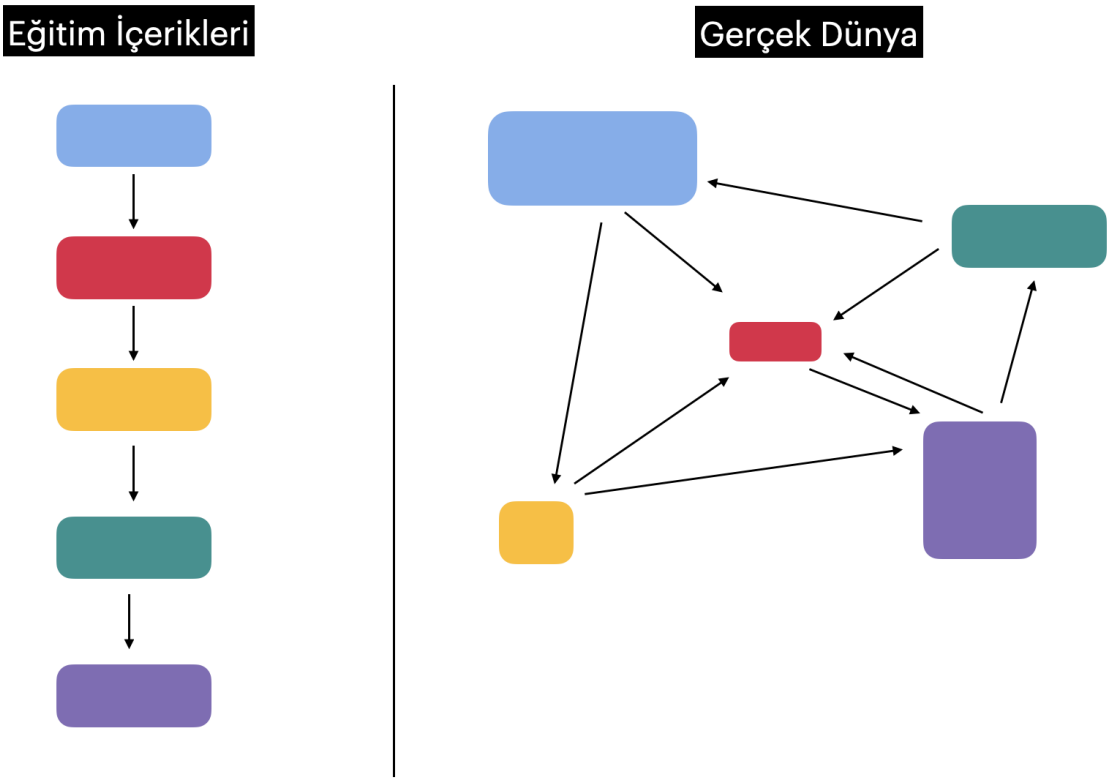
Bu konuyu öğrenmeye de uygulayabiliriz. **Gerçek Öğrenme** nedir?

> “Real learning is not memorizing knowledge. It’s understanding and knowing how to use and find knowledge. Learning is what you do with knowledge, how you integrate it, how you talk to your family, friends and classmates about it. That’s what learning is. **“(Dennis Littky)**

Dennis Littky’nin yukarıdaki sözlerini Türkçeye aktaracak olursak şu ifadeleri kullanabiliriz; “Bilgiyi ezberleyerek elde edemezsiniz. Onu nasıl kullanacağınızı ve nasıl bulabileceğinizi anlamalı ve bilmelisiniz. Öğrenme o bilgiyle ne yaptığınıza, onu nasıl entegre ettiğinize ve çevrenize nasıl aktardığınıza bağlıdır.”

Konunun daha iyi anlaşılabilmesi için aşağıdaki görseller üzerinden bir açıklama sunulmuştur:

- Solda verilen örnek piyasada yaygın olarak kullanılan öğrenme yöntemidir.
- Sağdaki örnek ise gerçek dünyada bir uygulamanın geliştirilmesi sırasında ihtiyaç duyulan bilgi birikimini göstermektedir.



Eğitim İçerikleri ve Gerçek Dünyadaki Bilgiler Arasındaki Bağlantı

Eğitim içeriklerinin genelde belirli bir sırayı takip eden kavramları, dersleri arka arkaya anlatmaya çalışan içerikler olduğunu görürüz. Bu içeriklerin birbirleri ile nasıl bağlanacağı ve nasıl uygulamalar oluşturulacağından çok, soyutlanmış olarak bir konuya odaklanılır ve bu konu anlatılmaya çalışılır.

Ama gerçek dünya bu şekilde değildir. Gerçek dünyada bilgi birikimleri birbirleri ile bağlantılı düğümler şeklindedir. Bazı bilgileri, konseptleri çok kullanmanız, çok bilmeniz gerekirken, bazı konuları daha az bilmeniz yeterli olabilir. Bazı konular her yerde anlatılırken, bazı konularda bilgileri Stackoverflow'da ya da GitHub açık kaynaklı projelerin kaynak kodlarının derinliklerinde bulabilirsiniz veya benzer bir sorunu bulup, kendi projenize uygulamanız gerekir.

Bir bileşen veya web sayfası oluştururken tüm bu bilgileri harmoni içerisinde kullanmanız ve birleştirmeniz gerekir. Bu oluşturma safhasında birçok konsepte hakim olmanız, en iyi pratikleri bilmeniz ve detaylarda gizli birçok noktadan haberdar

olmanız büyük önem taşır. En önemlisi de kapsamları nasıl harmanlayacağınızı biliyor olmanızdır.

Yani bu eğitimlerde elde edeceğiniz kapsam, örneğin pişireceğiniz yemeğin malzemelerini, araçları ve hangisinin ne işe yaradığını verse de bunlarla yapabileceğiniz yemeklerin (bileşenlerin) tarifini ver**Mi**yor olacaktır.

LearnReactUI.dev sitesi, gerçek dünya uygulamalarına yakın örnekler içeren anlatımları ile gerçekten hızlı bir şekilde hedefinize ulaşmanızı sağlar, bunu yaparken de yukarıda belirtilen ilişkileri ve derinliği öğrenebileceğiniz eğitim yöntemini baz alır.

GERÇEK DÜNYADAN BİR ÖRNEK

Konuyu gerçek bir örnek ile açıklamak daha anlaşılır olacaktır. Ürün ekibinin sizden bir UIX tasarımına göre bileşen geliştirmenizi istediğini varsayalım.

Bileşen İstekleri

- Veriler bir tablo gibi hizalı gözükmeli ve aynı zamanda kendi CSS tasarımını giydirebilmek mümkün olmalı.
- Pagination olmalı, Scroll olmalı.
- Bir satır içerisindeki hücreye metin sığmıyorsa bu hücrenin genişliği, boyutu büyümemeli. Bunun yerine (…ellipses) yazmalı ve tooltip çıkmalı.
- Mesaj uzunluğu çoklu satır veya belirli bir karakterin üzerinde ise bu detayda, tooltip olarak gösterilmeli.
- Verileri sunucudan çekmeli ve ekranda sadece ilgili kısımlar render edilmeli.
- Row sayısı çok olsa da kullanıcıya iyi bir etkileşim sunmalı, Scrolling akıcı bir şekilde gerçekleşmeli.

İstekleri Yerine Getirebilmek İçin Bilinmesi Gereken Konseptler

- Bileşenin görselliğindeki hizalar nedeniyle kullanılacak Layout bilinmelidir: Table **(DOM Bilgisi)**
- Bu tablonun her row'a ait detayı olsun, 2 hücre bir bütün olarak gözüksün istersek ve bunların Collapsable açılma özelliği olsun dersek bunu nasıl yapacağız. HTML Table colspan kavramı bilinmelidir. **(DOM Bilgisi)**
- Collapsable alanın animasyonlu ivmeli bir yay hareketi ile açılmasını sağlamak için biraz **(Matematik ve WebAnimations API)** bilinmelidir.
- Font ve Iconlar veya kendi CSS Animated Elemanlarını oluşturmak için **(CSS Bilgisi)** gereklidir.
- Veriyi sunucudan alabilmek için (**Fetch, HTTPRequest, AJAX, ve JSON Bilgisi**) gereklidir.
- Bu bilgiyi bellekte veya tarayıcı Storage içerisinde saklamak için **(JS, Structures, LocalStorage)** bilgisi gereklidir.
- Bu veriyi kontrollü bir şekilde yönetebilmek için **(Veri katmanı Kütüphaneleri, Redux, React-Query)** bilgisi gereklidir.
- Bunları ekrana render edebilmek için **JSX, React vb. (Vue, Svelte, Angular)** bilgisi gereklidir.
- Performanslı bir şekilde ekranda Tabloları çizmek için **Virtualized** mantığını, verinin nasıl Frontend kısmında cachelenmesi gerektiğinin bilinmesi gereklidir.

Ve burada yazmadığım birçok konuyu zihninizde tasarlayıp bunları doğru bir şekilde harmanlamanız/birleştirmeniz çok önemlidir. Örneğin;

- Bunu yaparken zaman planlamanızı nasıl yapıyorsunuz?
- Hangi geliştirme ve otomasyon araçlarından faydalanıyorsunuz?
- Kodun kalitesi için ESlint, Test Kütüphaneleri vb. neleri kullanıyorsunuz?
- Hangi açık kaynak kodlu kütüphanelerden faydalanabilirsiniz, bunları sisteminize nasıl entegre edebilirsiniz?
- Tüm bunları hangi sıra ile ve ne kadar süre kullanıyorsunuz? Hangi işlemleri birbirine bağlıyorsunuz?

Görüldüğü gibi gerçek dünyadaki uygulamalar, eğitimlerde sırası ile arka arkaya ders anlatılması ve okunarak öğrenilmesinden biraz farklılık gösteriyor. Gerçeğe yakın

proje örnekleri ile konuları anlatırsak daha hızlı ve derinlemesine bir öğrenme sağlayabiliriz.

Bu amaç doğrultusunda LearnReactUI.dev sitesini oluşturduk. Bu konuda neden LearnReactUI.dev sitesini tercih etmeniz gerektiğini kısaca anlatayım.

1. LearnReactUI.dev Sitesi İçerik Olarak Diğer Eğitim Sitelerinden Neden Farklıdır?

Öncelikle sizi gerçek dünyadaki proje geliştirme sürecinize hazırlayacak bir eğitim sitesidir. Bunun için gerçek proje örneklerinden gelen pratikler üzerine bir eğitim bulacaksınız.

20 yıllık yazılım geliştirme sürecinde UI konusunda **farklı diller ve kütüphaneler** kullanarak

- Java, Swing, JavaFX
- JS, JQuery, ExtJS
- JS, React

farklı sektörlerde

- Askeri
- Sigorta ve Bankacılık
- Devlet Kurumları
- APM (Application Performance Monitoring)

farklı büyüklükteki ekipler ile

- Küçük Boyutlu Ekipler (5-20)
- Orta Büyüklükteki ekipler (20-50)
- Çok Büyük Ekipler (50-100)

farklı Backend ortamlarında çalışan

- On Prem Makinede, Near Real Time

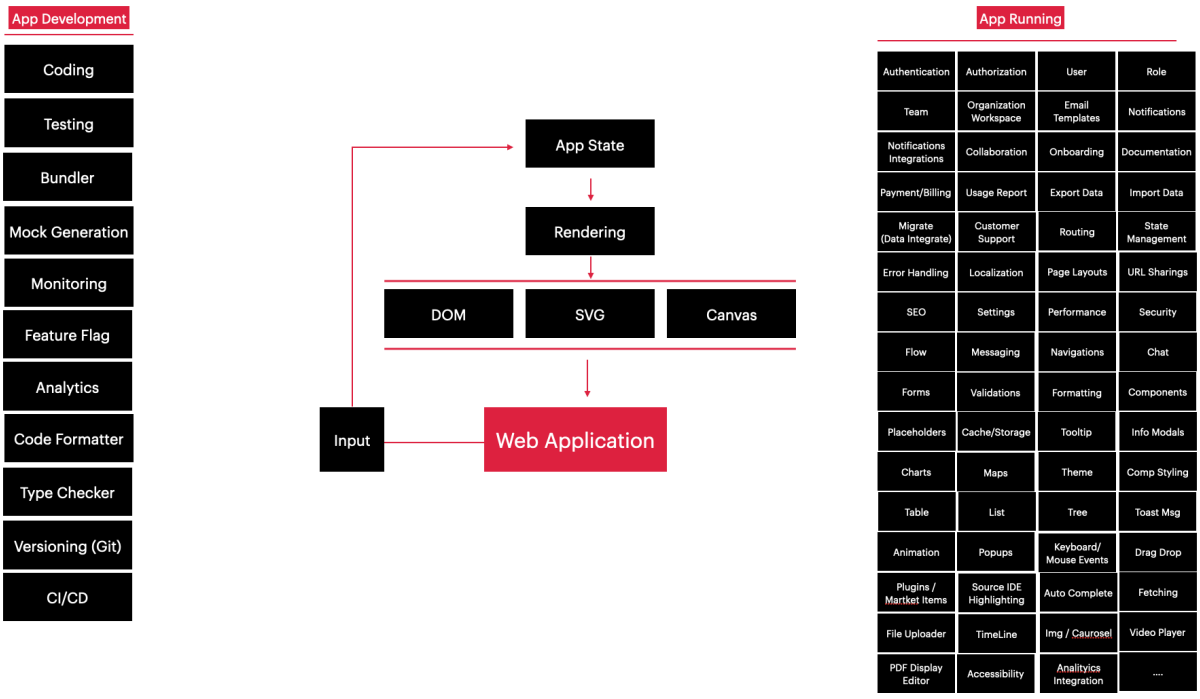
- Cloud Üzerinde HTTP ve WebSocket

uygulamaların UI geliştirme ekiplerinde yer aldım. Bu projelerden birçok deneyim elde ettim. Oluşturacağım bu eğitim içeriklerinde, bu deneyimleri güncel teknolojiler ve kütüphaneler kullanarak gerçeğe yakın örnekleri ile sizlere anlatmaya çalışacağım.

Web Uygulamasına Yakından Bakalım.

Aşağıdaki resim 3 kısımdan oluşuyor. Bunlar, Web Uygulaması geliştirirken UI Kütüphanelerinde günlük teknolojik değişimlerinden bağımsız olarak ilgilenmeniz gereken konulardır.

- Sol tarafta, biz geliştiricilerin **App Development** sırasında üzerinde çalıştığı konseptler mevcuttur.
- Orta kısım **Uygulama Çalışma Mantığını (App Running)** anlatmaktadır.
- Sağ tarafta ise bir Web App Çalışırken **(App Running)** işlenen konseptler bulunmaktadır.



LearnReactUI bu şekilde standartları, benzer UI çalışma mekaniklerini ve ortak mimarileri anlatmaya çalışır.

- Standartlar (HTTP, HTML, CSS, … vb.)
- UI Çalışma Mekanikleri (Ekran verilerini al, Validate Et, Kaydet vb.)
- Renk Uyumları, Layout yapıları vb.
- Farklı farklı ortamlarda çalışma gereklerini anlama (Desktop, Mobil, Tablet …)
- Tasarım Örüntüleri
- Vb. …

1.2 LearnReactUI.dev Eğitim Süreci Olarak Diğer Sitelerden Ne Gibi Farklılıklar İçerir?

Eğitim sitelerinde genelde dersler yukarıda bahsettiğimiz gibi belirli bir sıra ve düzende, aşağıdaki şekilde anlatılırken, yapısı genelde şu şekildedir:

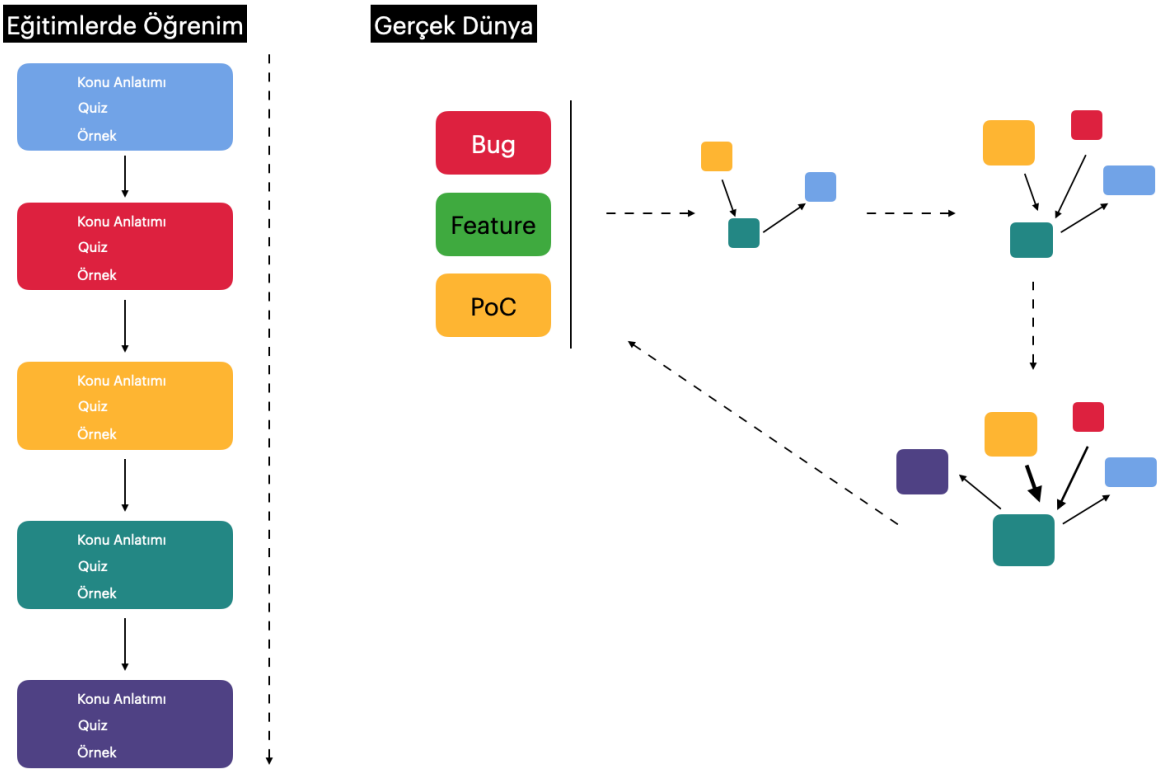
- Konu Anlatımı
- Quiz
- Örnek

Bu anlatım şekli, bir konuya yabancıysanız ve bu konuyu ilk kez öğrenmeye çalışıyorsanız izlenebilecek bir yol iken, bu konuda biraz bilgi sahibiyseniz sizi gereksiz detaylarla uğraştırabilir veya başlarda hatırladığınız birçok konuyu dersin diğer konularını dinledikçe unuttuğunuz bir hale dönüşür.

Gerçek dünyada ise öğrenme Hata, Projeye Yetenek Geliştirme veya PoC (Proof of Concept) çalışmaları ile karşımıza çıkar. Proje geliştirirken aynı zamanda üzerimizde bir zaman baskısı da vardır. Geliştirmeyi yaparken birçok konuya dikkat etmeniz gerekir:

- Önceki Legacy kodu bozmamak
- Mimariye uyumlu olmak

- Verilen zaman içerisinde işi tamamlamak
- Daha sonrası için bakımı yapılabilir ve tekrar kullanılabilir kod geliştirmek



Yukarıda bahsettiğim gibi Gerçek Dünyadaki öğrenme yöntemi, eğitim içerikleri ile sağlayacağınız öğrenmeden **çok daha eğitici ve akılda kalıcı** olacaktır. Bir diğer konu da zaman içerisindeki bilgi birikimi ve bağlantıları oluşturma yönteminizdir. Sağdaki yöntemde kavramlar arasındaki bağlantılar çok zor oluşurken, gerçek işte çalıştığınızda elde ettiğiniz deneyimler bu kavramları birbirine nasıl bağlayacağımızı ve bunları nasıl bir harmoni içerisinde kullanacağımızı bize doğal yollardan öğretir.

LEARNREACTUI.dev sitesi gerçek uygulama bileşenlerine benzer örnekleri anlatarak konuyu en etkili, hızlı ve akılda kalıcı olacak şekilde öğrenmenizi sağlar.

Bunu yaparken işinizde hangi konu üzerinde çalışmak istiyorsanız, bu konu ile örneği seçebilmenize olanak sağlar. Örneğin;

- Tablo
- Chart ve Dashboardlar

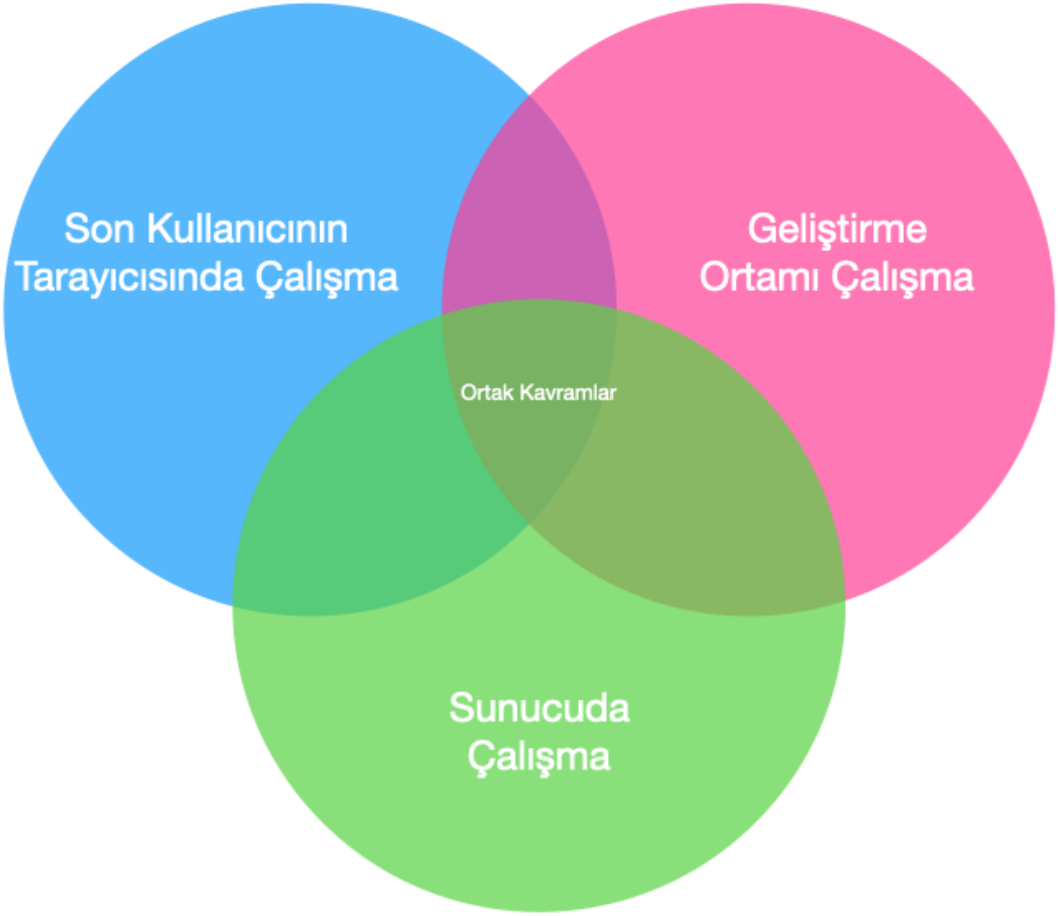
- Navigasyon

Hangisi sizin için daha önemli ise o konuya ilişkin örneği alıp üzerine çalışmaya başlayarak, odaklanmış bir şekilde amacınıza uygun öğrenmeyi gerçekleştirebilirsiniz.

1.3 LearnReactUI.dev Farklı Perspektiflerden Bakış Kazanımı Sağlar

Front end kapsamına farklı perspektiflerden bakabiliyor olmak biz geliştiricilere büyük avantajlar sağlar.

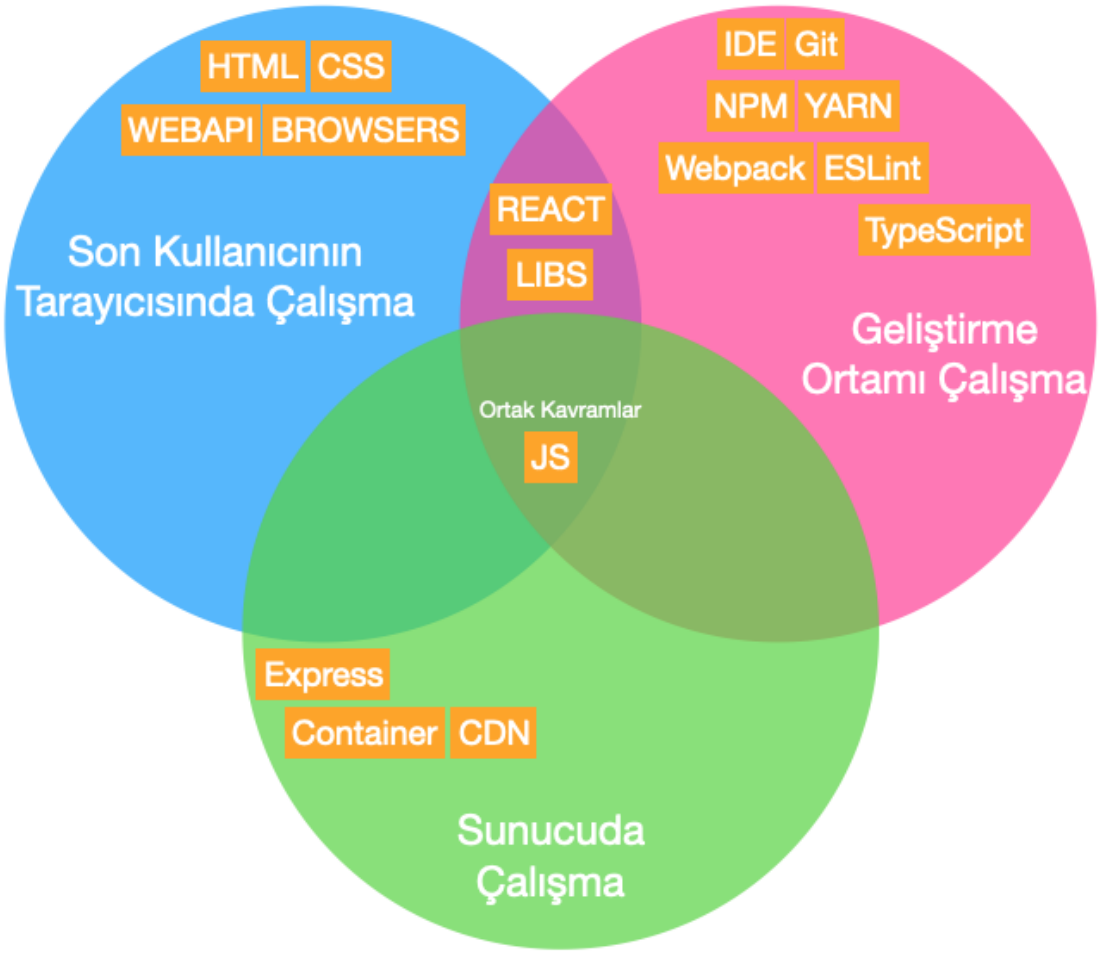
- Tarayıcı Ortamı
- Geliştirici Ortamı
- Sunucu Ortamı



- Frontend ile ilgili olarak öncelikle geliştiriciler tarafından **kendi ortamlarındaki geliştirme süreçlerinde** kullanılan araçlar ve kütüphaneler mevcuttur.
- Sonraki süreçte bunların çıktıları belirli **sunuculara *deploy* edilmektedir ve bu sunuculardan kullanıcılara** servis verilirken kullanılan hizmetler, araçlar mevcuttur.
- Ayrıca, kullanıcının tarayıcılar aracılığı ile sunucudan çektiği HTML, CSS, JS gibi verileri ve diğer verileri (JSON, Audio, Video, Image, Base64) **tarayıcı üzerinde görüntülediği ve kullanıcı etkileşimlerini yönettiği** bir kısım da vardır.

Yani özetle 3 ortamdan bahsedilebilir. Teknoloji ve kavramların bu küme içerisinde nerede olduğunu tam olarak öğrendiğinizde birçok konuyu daha iyi kavramanız mümkündür.

Küme içerisindeki kavramların yer aldıkları alanları aşağıdaki şekilde daha net olarak görebiliriz.



- **Tarayıcı Ortamı:** Web uygulamalarına ait HTML, CSS, JS ve resim ve PDF benzeri dosyaların, Sunucu/Dosya sistemi/CDN gibi ortamlardan bilgisayarınızdaki tarayıcılarda işletilmesi.
- **Sunucu Ortamı:** Sunucudan HTML, CSS ve JS dosyaları sunulur. Burada sunucunun sorumluluğu sadece API üzerinden REST sağlama seviyesinden, Template rendering ile HTML üretme arasında farklılık göstermektedir. Amacınıza göre Rendering yönteminiz değişir, buna göre sunucu sorumluluğu ve kullanacağınız Framework değişkenlik gösterir.

```
CSR(ClientSide Rendering) <--  ..... --> SSR(ServerSide Rendering)
```

- **Geliştirici Ortamı:** Localde hem sunucu hem de tarayıcı ortamlarını çalıştırırız ama bunun haricinde bir takım geliştirme ve üst seviye soyutlama araçları

bulunur ki bunlar hayatımızı kolaylaştırır.

Bu ortamların neler olduğu, hangi teknoloji ve kütüphanenin nerede çalıştığı örnekler ile açıklandıktan sonra, artık buna benzer bir teknoloji ile karşılaştığınızda, bunu kafanızda şekillendirmeniz ve kullanmanız oldukça basit ve hızlı olacaktır.

Özetle LearnReactUI.dev sitesi konulara farklı perspektiflerden bakmanızı sağlar.

1.4 LearnReactUI.dev Farklı Kütüphaneler Deneyerek Bunlardan Elde Ettiği Deneyimleri ve Karşılaştırmaları Paylaşır

Günümüzdeki Web Uygulama geliştirme süreci açık kaynaklı kütüphaneler üzerinden işler. Özellikle React Ekosisteminde binlerce farklı kütüphane bulunur. Bu kütüphaneleri yeri gelince doğru şekilde kullanmak, doğru seçimleri yapmak oldukça önemlidir. Projeden projeye, ihtiyaçlara göre kullanacağınız kütüphaneler farklılık gösterebilir.

Örneğin React için State yönetiminde birden fazla kütüphane bulunur: Redux, Apollo GraphQL, React-Query vb. Burada bu kütüphanelerden hangisi seçeceğinizi anlamanız için State Management kavramının neleri içerdiğini anlıyor olmanız gerekir. LearnReactUI.dev söz konusu konseptleri bu kütüphanelerden biri üzerinden örnekler ile anlatarak konuyu kavramanızı sağlar.

Her geçen gün yeni bir teknoloji çıkıyor. Önemli olan konseptleri anlamanızdır. Örneğin, Uygulama ve Server State yönetme sırasında **aşağıda belirtilenleri** bilirseniz;

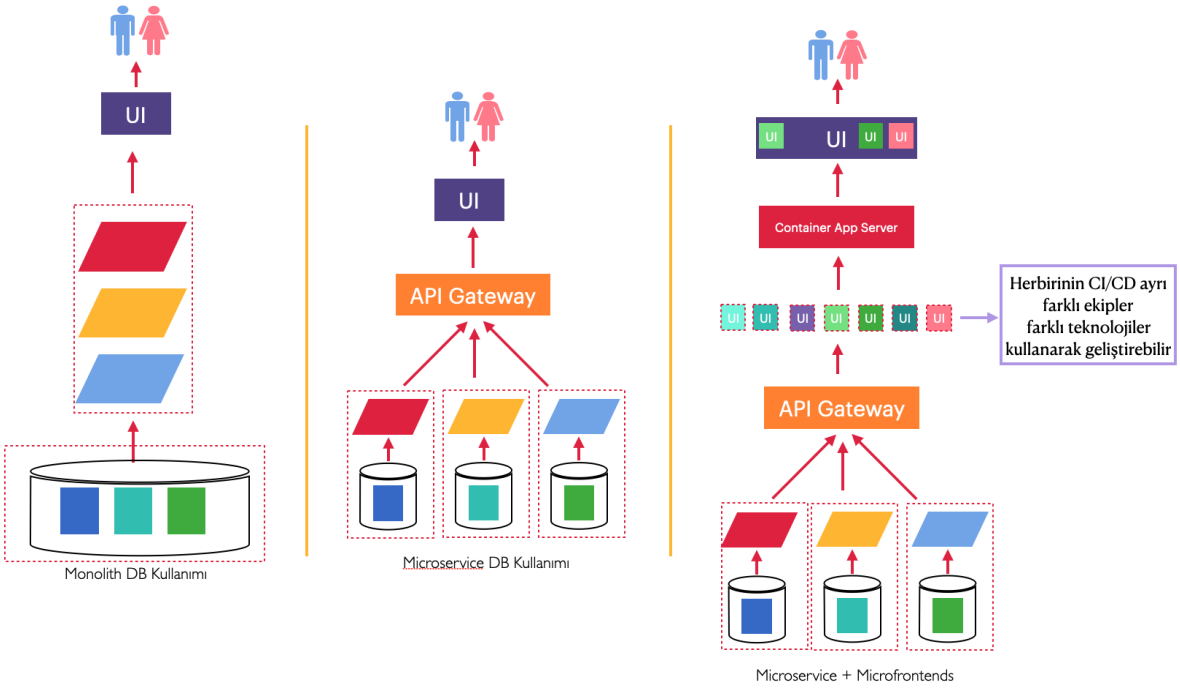
- Async Status
- Caching
- Callback Handling
- Mutation & Optimistic Update
- Background Updates

- Pagination /Incremental Loading
- Outdated Requests
- Deduping Request
- Garbage Collection ve Memory Implications

bu konu hakkında yeni çıkan teknolojilere adaptasyonunuz o kadar çabuk ve detaylı olur.

1.5 LearnReactUI.dev İçerisindeki Eğitimler Mikroservis Mantığına Benzer Şekilde Hazırlanır

Mikroservis son dönemlerde öncelikle Sunucu tarafında daha sonra da Frontend kısmında popüler olmuştur. Peki Mikroservis nedir? Mikroservis mantığında, uygulamalar monolitik değil de küçük küçük uygulamalar olacak şekilde tasarlanır; böylece, bu parçalar hem uygulamanın genelini etkilemez hem de ölçeklenebilir olur.



LearnReactUI.dev eğitimleri de benzer şekilde küçük uygulama parçaları halinde olacaktır. Bu uygulama parçaları gibi eğitimleri de daha küçük şekilde seçebilecek ve uygulamanıza yakın eğitimi birbiri ile ilişkili parçalardan oluşturabileceksiniz. Bu sayede, mevcut eğitim örneklerini kendi projelerinize hızlı bir şekilde adapte edebileceksiniz.

Bu durum aynı zamanda, faydalanabileceğiniz eğitimlere blok şeklinde toplu paralar ödemek yerine sadece hedefinize yönelik örnekleri kullanarak daha küçük ücretler ödeme imkanı sağlayacaktır.

💡 **NASIL FRONTEND UZMANI OLURUM ?**

LearnReactUI sitesine üye olarak bu hedef için ilk adımı atabilir ve bunun yanında, her zaman sizi ileriye taşıyacak içeriklere ulaşabilirsiniz. Bu yazının devamı olan Teknik Detay dokümanı her eğitimle birlikte güncellendikçe, ilgili bildirim e-postasını alacak ve istediğiniz eğitim içeriklerinden faydalanabilmeniz için sürekli bilgi sahibi olabileceksiniz.

2. TEKNİK DETAYLAR

Yukarıda belirtilen konuların teknik olarak detaylı açıklaması aşağıda verilmiştir. Bu konularda yeterli teknik bilgiye sahipseniz, dokümanın devam eden kısmını okumayı tercih etmeyebilirsiniz.

2.1 Web Uygulama Yapısı, Süreçler, Konseptler ve Araçlar

Web Uygulama yapısı, ilgili süreçler, konseptler ve araçlar ile ilgili teknik detay ve açıklamalar aşağıda verilmiştir.

2.1.1 Web Uygulama Mekaniği (Web Application)

Öncelikle yukarıdaki resmin **ortasında bulunan Web Application kısmını** açıklayalım.

Web uygulaması, Tarayıcı üzerinde bulunan WebAPI sayesinde (DOM, Canvas, SVG) yapılarının üzerine AppState (Uygulama Durumunu) çizilmesi (rendering) ve kullanıcı/network vs. gibi gelen girdilerin (Input) dinlenerek AppState'in tekrar güncellenmesi ve ekrana yansıtılmasıdır.

2.1.2 Uygulama Geliştirme (App Development)

Frontend geliştiricilerin Web App Geliştirme aşamasında gerçekleştirdiği aktiviteler;

Coding: Yazılım geliştirme aşamasında yazılım mimarisinde birçok konu bulunuyor. Bunlar sizin daha temiz ve kaliteli ve tekrar kullanılabilir kod yazmanızı sağlayan konseptlerdir.

- Mimari Kavramlar
- Mimari Deneyim
- Mimari Kalite
- Mimari Örüntüler
- Tasarım Örüntüler
- Anti-Patterns

Testing: Kodun, Modülün veya Kütüphanenin, API'nin veya Uygulamanın farklı seviyelerde testini yaparak yazdığınız kodların doğruluğunu sağlamanız gerekir.

Bundling: Yazdığımız kodları (JS ve türevleri), styling (CSS ve türevlerini) Production Ready haline getirmeye sağlayan araçlar üzerindeki uzmanlığınızdır. Webpack, Rollup, ESBuild ve Vite araçlarını bunlara örnek verebiliriz.

Mock Generation: Frontend için yazmış olduğunuz kodlar bazen Back end, bazen ise kullanıcı vb. verilerini gerektirir. Bu verileri test ortamınız için üretmenizi sağlayacak araçların kullanımını içerir. Örneğin;

- Faker.js
- JSON Server
- MSW (Mock Service Worker)
- Pact.io

Monitoring: Frontend monitoring alanında, herhangi bir hata ve problemi yakalamak için kullanabileceğiniz kütüphane ve SaaS hizmetleridir. Örneğin;

- Log Rocket
- Sentry

Feature Flag: Uygulamanızda bazı yetenekleri açıp kapatabilecek bir altyapı ile geliştirme yapma ihtiyacınız olabilir. Örneğin A/B testing için;

- LaunchDarkly

Analytics: Müşteri demografik yapısını ve davranışlarını analiz etmenizi sağlayan bir takım analitik araçları, ürünü nasıl geliştirmeniz gerektiği yönünde birtakım bilgiler sağlar. Örneğin;

- Google Analytics
- Heap
- FullStory

Code Formatter: Kodun formatlanması ve güzel okunabilir gösterilmesi için bir takım plugin'ler ve araçlar kullanırız. Örneğin;

- Prettier
- Es-Lint

Type Checker, Annotations vb. Dil Desteği

- TypeScript
- Flow, Reason, Elm, ClojureScript, Flow, PureScript

Git Kullanımı: Versiyon yöntemi için Git yapısının nasıl çalıştığını, branch yapısını nasıl kullanacağınızı bilmeniz gereklidir.

CI/CD: Continuous Integration ve Development konusunda son dönemlerde GitHub Actions oldukça popülerdir. Bunun dışında Jenkins, Bamboo da kullanabilirsiniz.

2.1.3 Uygulama Konseptleri (Application Concepts)

Uygulama Konseptleri içerisinde birçok farklı konuyu barındırıyor. Bunları tek bir yazı içerisinde anlatmaya çalışmak yazıyı çok karmaşık hale getirecektir. Bu nedenle, burada sadece bu konseptlerin isimleri verilmiştir. İlerleyen yazılarda bunlar daha detaylı olarak ele alınacaktır.

- Authentication, Authorization
- User, Account, Role
- Organization, Workspace, Team
- Notifications, Email, SMS Templates
- Notification Integrations (WebHooks)
- Collaboration, Chat
- Onboardings, Documentation, Customer Support
- Payment, Billing, Usage Reports
- Data Migration, Export/Import Data
- Routing, Navigations
- State Management
- Error Handling, Localization, Accessibility
- Page Layouts, Tabs, Sidebars, Drawers
- SEO, URL Sharings
- Settings, Configurations
- Performance, Security, Messaging
- Business Flow
- Messaging, Chat
- Forms, Validations, Formatting, Components
- PlaceHolders, Tooltips, Info Modals
- Cache/Storages
- Warnings, Toast, Popups
- List, Table, Tree, Charts, Maps, Timeline

- Time Components, Calendar
- Keyboard/Mouse Inputs, Drag-Drops
- Plugin, Marketplace
- Animations
- IDE, Syntax Highlighting, Gutter
- Fetching, Websocket, SSE, WebRTC
- File Uploading, Camera
- Img, Carousel, Video Playing
- PDF Displaying and Editor
- Analytics Integration (Google Tag Manager, Heap, FullStory, Intercom)

Yukarıda bahsettiğim konseptlere ek olarak TailwindUI, PrimeBlock, ve TailwindUIKit gibi platformların UI bloklarını styling açısından hazır şablonlar halinde verdiğini görebilirsiniz.

2.1.3.1 Marketing

- **Page Sections:** Hero, Feature, CTA(Call to Action buttons), Pricing, Header, FAQs, Newsletter, Stats, Testimonials, Blog, Contact, Team, Content, Footers, Logo Clouds
- **Elements:** Headers, Banners, Flyout Menus
- **Feedback:** 404 Pages
- **Page Examples:** Landing, Pricing, Contact

2.1.3.2 Application UI

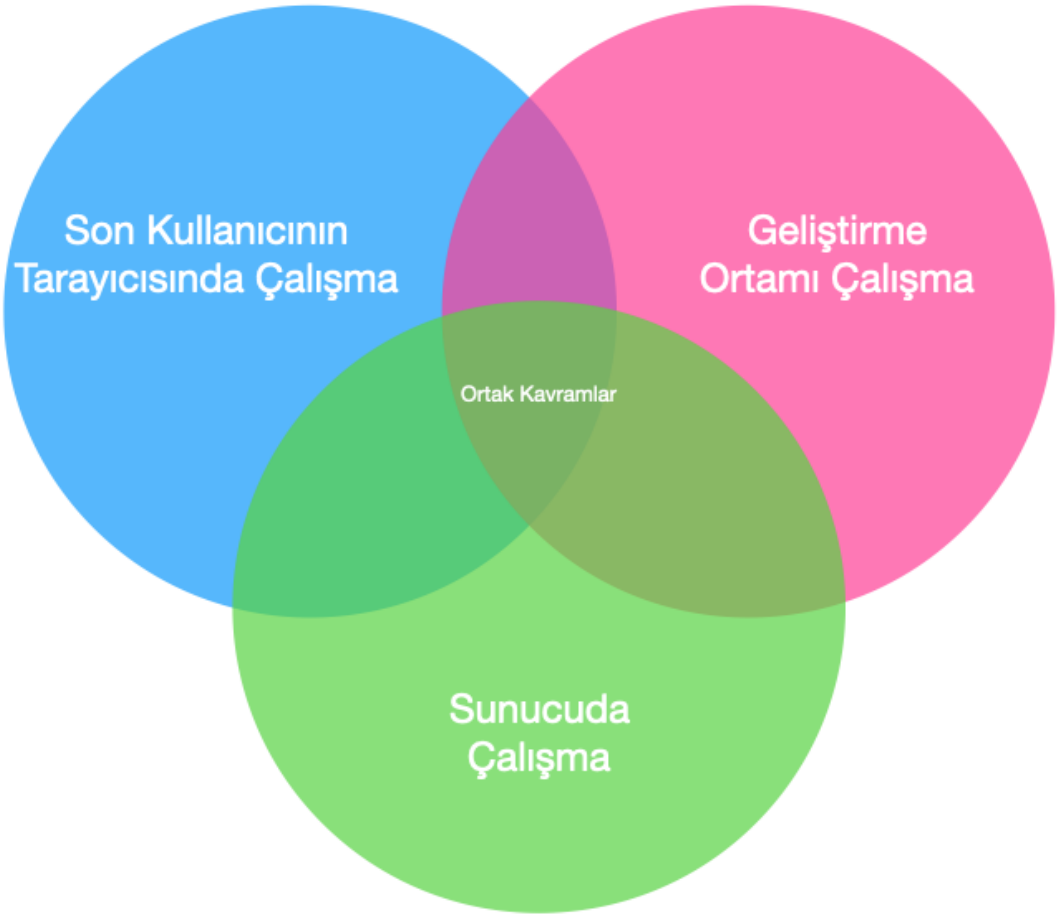
- **Application Shells:** Stacked, Sidebar, Multi-Columns
- **Headings:** Page, Card, Section
- **Data Display:** Description List, Stats, Calendars
- **Lists:** Tables, Stacked List, Greed List, Feeds

- **Form:** Form Layouts, Input Groups, Select Menus, Sign-in, and Registration, Text Areas, Radio Groups, Checkboxes, Toggles, Action Panels, Comboboxes
- **Feedback:** Alerts, Empty States
- **Navigation:** Navbars, Pagination, Tabs, Vertical Nav, Sidebar Nav, Breadcrumbs, Steps, Command Palettes
- **Overlays:** Modals, SlideOvers, Notifications
- **Elements:** Avatars, Dropdown, Badges, Buttons, ButtonGroups
- **Layouts:** Containers, Panels, List Containers, Media Objects, Dividers
- **Page Examples:** Home, Details, Settings

2.1.3.3 ECommerce

- **Components:** Product Overview, Product List, Category Preview, Product Quickviews, Product Features, Store Navigations, Prome Sections, Checkout Forms, Review, Order Summaries, Order History, Incentives
- **Page Examples:** Storefront, Product, Category, Shopping Cart, Checkout, Order Details, Order History, Shopping Carts, Category Filter

2.2 Web Uygulaması ve Geliştirme Ortamlarına Farklı Perspektiflerden Bakabilme



2.2.1 Tarayıcı (Browser) Ortamı

Bu ortam ister kullanıcının bilgisayarında ister cep telefonunda isterse iPad'de olsun, kullanıcılar bir tarayıcı üzerinden bu web sayfalarına, web uygulamalarına erişebilir.

2.2.1.1 Tarayıcılar

- Chrome, Safari, IE, Firefox, Edge, Opera

  Peki bu tarayıcılar sunucudan aldıkları verileri işleyip bunların sonucunda web sayfaları ve etkileşimli uygulamalar geliştiriyorlar. Bu tarayıcılar nasıl çalışıyor? Kısaca **HTML, CSS, JS** işleyerek web sayfalarını oluşturuyor. Burada 2 tip engine kullanılıyor.

2.2.1.2 Javascript Engine

- V8 (Chrome) , SpiderMonker (Firefox), Chakra (IE), SquirrelFish (Safari), V8 (Opera)

  Javascript Engine JS dosyalarını işleyen engine'dir. Bu sizin UI business kodlarınızı işler veya WebAPI ile işletim sistemi ve diğer seviyelerdeki API'yi çağırarak Network, DOM, Animation, Bluetooth vb. sistemlerini yönetir.

2.2.1.3 Javascript (EcmaScript)

- **JS Tarihçesi** kitapçığında **let/const,** Arrow Functions, Inheritance, Default Parameters, Rest/Spread, Destruction Assignments, Template Literals, Loops, Async, Promise, EventLoop, Modules, Map, Set, Generators, vb. konularından genel şekilde bahsedilerek ES6 EcmaScript dil olarak nasıl geliştiği anlatılacaktır.
- **JS ile Fonksiyonel Programlama** kitapçığında daha kaliteli, hataya ve kırılganlığa dayalı yazılımları nasıl geliştirebileceğimiz konusunda bilgi verilecektir.

2.2.1.4 Browser Engine

- Blink (Chrome), Gecko (Firefox), Trident (IE), Webkit (Safari), Blink(Opera)

  DOM, CSS birleştirerek Layout oluşturup bunun için HTML elemanlarını Render eden Engine'dir. **Not:** DOM ve CSS günümüzde JS tarafından oluşturularak da kullanılabilmektedir. (React, Vue, Angular, StyleCSS, vb.)

2.2.1.5 CSS

Cascading Stylesheets: DOM Elemanlarına stil vermeyi sağlayan mekanizmadır. Bu sayede sayfalarımız daha güzel, belirli bir hizalamaya göre oluşturulur. CSS standartlarına baktığımızda aşağıdaki seçenekler mevcuttur.

- **CSS Building Blocks:** Cascade and inheritance, CSS selectors (Type, class, and ID selectors, Attribute selectors, Pseudo-classes/elements, Combinators), BoxModel (Margin, Border, Padding, Content), Outline, Background, Borders, Wring Modes, Overflow, Values & Units, Image/Media/Form Elements

- **CSS Text Styling:** Typography, Text, Font, Icon, Link, List

- **CSS Layout (display)**: Floats, Positioning, Flexbox, Grid, Responsive design

- **CSS Advanced:** 2D transform , 3D transform, Transition, Animation, Shadow, Gradient, Tooltip, Rounded Corners, Boundary, Color, Backgrounds, Web Fonts

  Aslında bu CSS tanımlamalarında, geliştiricilere **Preprocessor** kullanarak bu CSS verilerini yapılandırılmış ve tekrar kullanabilir şekilde yazma imkanı sunuluyor (SASS, LESS vb.).

  Ayrıca JS ile CSS oluşturma günümüzde çok popüler olup bu CSS-in-JS olarak adlandırılmaktadır. Bu yapıları kullanmanızı sağlayan bazı kütüphaneler mevcuttur.

- **CSS-in-JS Kütüphaneleri:**

  Styled Components, CSS Modules, Styled JSX, Emotion, vb.

  Peki bu CSS için oluşturulmuş hazır Framework (Alt çerçeve) ve Tema giydirilmiş UI bileşen kütüphaneleri mevcut mudur? Evet.

- **CSS Frameworks:**

  Bootstrap, Materialize CSS, Foundation, Semantic UI, Bulma, UIkit vb.

2.2.1.6 Uygulama/Web Sayfası Türleri:

- SPA (Single Page Application), SSG (Static Site Generation), SSR (Server Side Rendering) PWA (Progress Web App)

Uygulamaları şu açıdan gruplamak uygun olacaktır:

- **SPA:** Kullanıcı Cihazındaki Tarayıcının gücünü kullanarak çalışan, yani olabildiğince tek sayfa içerisinde (SPA) uygulamanın tüm ekranlarının ve bileşenlerinin olduğu Web Uygulama Hizmetleri'dir. Mümkün olduğu ölçüde, JS dosyalarının On The Fly şekilde yani o anda tarayıcı tarafından işletildiği ve bunun ile birlikte AJAX/HTTP Request yapılarak RESTfull servislerden JSON vb.

içeriklerin siteye yüklendiği ve sonrasında sayfaların render edildiği, etkileşimi yüksek uygulamalardır.

- **SSR ve SSG:** İkinci tip uygulamalar; bilgi paylaştığımız, CMS, Blog, Portal vb. gibi her bir URL'nin ayrı bir unique bilgi verdiği, Search Engine Crawler tarafından sayfaların içindeki verinin indekslenebildiği, teknik açıdan HTML ve CSS sunucu tarafında son haddine kadar oluşturulduğu, JS işleme açısından tarayıcıları çok az yoran, kullanıcı etkileşiminin çok olmadığı sayfalardır. Bu tür CMS sayfaları geçmişte çoğunlukla WordPress, Joomla, Drupal, phpBB, ile sayfaları Sunucuda LAMP Stack (Linux, Apache, MySQL, PHP/Perl/Python) üzerinden ilerliyordu. Bugün ise onların yerini SR JAMStack (JavaScript, APIs, and Markup) aldı.

- **PWA:** Progressive Web App kavramı Google tarafından ortaya atılmıştır. Standart Web Sayfaları ve Native Mobil Uygulamalar arasında hibrit bir uygulama modeli çıkaralım. Bu model, Web sayfası gibi çalışsa da internet olmadığı zamanlarda da Native uygulama özelliklerini göstermesi için ServiceWorker, LocalStorage ve diğer WebAPI kullanır.

2.2.1.7 WebAPI/BrowserAPI

- **Browser API Kullanımı:** DOM API, LocalStorage, Fetch, WebSocket, i18n, ServiceWorker, WebComponent, WebAnimation, WebVR, WebGL, WebRTC

  Günümüzde mobil cihazların artması, desktop uygulamalarının web ortamına kayması sebebiyle web sayfalarından beklentimiz artmış durumdadır, bu da bu beklentiyi karşılayacak Browser API'lerin ortaya çıkmasına sebep olmuştur.

- **HTML DOM API Kullanımı:** Javascript, JQuery, Mustache.js, Handlebars.js, Backbone.js, Ember.js, Angular.js, React.js, Vue.js, Polymer-Project, Svelte.js, Solid.js

  Mevcut doküman üzerinde HTML Elemanları ve Node değiştirerek, UI bileşenleri (Tablolar, Düğmeler, Seçim Kutuları, Form, Navigasyon Menüleri), yazılar, paragraflar ve ekranda gördüğünüz çoğu eleman ve bileşenin çizilmesini sağlar.

- **SVG DOM API Kullanımı:** JS, D3, Raphaël, amCharts, Chart.js, Rechart, GoogleChart, GoogleMaps, HighCharts vb.

  Scalable Vector Graphics, şekilleri vektörel olarak oluşturulması, diğer bir deyişle ekran çözünürlüğünden, zoom in/out kavramlarından etkilenmeden Çizim Kütüphaneleri, Chart kütüphaneleri, oluşturulabilmesidir.

- **Canvas API Kullanımı:** JS, Fabric.js, Processing.js, Two.js, Cytoscape.js, Paper.js

  Canvas altlık üzerine pixel pixel boyamaya imkan sağlar. Bu sayede daha performanslı boyama işlemleri bu API sayesinde gerçekleştirebilir. Google Maps gibi, API sayesinde Harita katmanlarını çok performanslı bir şekilde çizdirebilmektedir ve her türlü mobil cihazdan haritalara erişim sağlar.

- **History API**

  Browser Session History tutarak ileri ve geriye ilerleyebilmenizi sağlar.

- **URL API**

  Uniform Resource Locator şeklinde tanımlanmış olan adresi domains, hosts, and IP erişmeyi ve güncellemeye sağlayan API’dir.

- **Console API**

  Chrome, Safari, Firefox Geliştirici araçlarındaki console erişimi sağlayan API’dir. (console.log, console.debug etc.)

- **Drag-Drop API**

  Web elemanlarının mouse ile sürüklenip bırakılmasını sağlayacak API sağlar. (Not: Detay için Drag-Drop API yazımı okuyabilirsiniz.)

- **WebGL / Web Audio / Web Speech API Kullanımı:** JS, Pixi.js, Three.js

  Grafik kartı, Ses Kartı ve Mikrofon donanımlarını daha iyi kullanmayı sağlayan API’ler sağlar. Canvas Context üzerine GL sayesinde 3 boyutlu Rendering, Audio API sayesinde ses kanallarına erişim, stream veri gönderme, bunlar

üzerinde efekt uygulama ve görselleştirme imkanı verir. Speech API ise mikrofon üzerinden ses tanıma ve bu sesleri metine dönüştürmeyi sağlayan API'dir.

- **Device API:** Ambient Light Sensor API, Geolocation API, Pointer Lock API, Proximity API, Device Orientation API, Screen Orientation API, Vibration API

  Web uygulamalarının donanım, sensör bilgilerine ulaşarak cihaz donanımından daha fazla şekilde faydalanmaya imkan sağlar.

- **Communication APIs:** WebRTC, NotificationAPI, Fetch, WebSocket, Network Information API, Web Notification API, Simple Push API, Bluetooth API, Beacon API

  Web sayfaları arasında Real Time Communication sağlayarak Video/Audio stream arasında peer to peer iletişim ve veri paylaşımı yapmayı sağlar. WebRTC, HTTP Request/Response mekanizma altyapısı için Fetch API, İki yönlü iletişim sağlayan client ile sunucu arasında aktif socket üzerinden haberleşme sunan WebSocket API, Local bildirimler için ServiceWorker ile birlikte Web Notifications API, sunucudan alınacak bildirimler için Push API kullanılabilir.

- **Data management APIs:** FileHandle API, IndexedDB, Storage API (Local Storage, Service Worker Registration, History States)

  Browser'da key-value olarak tutulan verilere, veritabanında tutulan verilere erişim sağlayan veya dosya sistemine erişebilecek API'ler sağlar.

- **Workers API:** Web Workers API, Service Workers API, Background Tasks

  Browser UI Look haricinde yan işleri yapabilecek Worker çalıştırmaya ve yönetmeyi sağlayan API'lerdir.

- **Performance API:** Performance API, Performance Timeline API, High Resolution Time, Frame Timing API

2.2.1.8 Web Components

- Custom Elements, Shadow DOM, HTML Templates, ES Modules

Vue, React, Svelte Ember vb. UI Kütüphaneleri tekrar kullanılabilir bileşenler oluşturulup bu bileşenlerin de birbirini kullanmasını sağlayarak daha büyük bileşen kütüphanelerinin oluşmasına olanak sağlamıştır. Sonuçta Web Components geliştirenlerin amacı bunun bileşen kütüphaneleri olmadan HTML standartlarında desteklenmesidir. Ve geliştiriciler tarafından bu yapıların benimsenmesi sonucunda popülerlik kazanabilme imkanı belki bulabilir. Şu anda rüzgar React, Vue ve Angular kütüphaneleri yönündedir.

2.2.1.9 WASM

- Tarayıcılar normalde JS Interpreter çalıştırılması üzerine kurulmuş bir yapıdır. EventLoop ile WebAPI üzerinden C Binding erişerek uygulamayı işletir. Compile edilmiş C/C++ veye Rust gibi low-level assembly benzeri dilleri çalıştırmak istersek, oyunlar, performans gerektiren uygulamalar için önümüzdeki bir çok engeli aşar. Tarayıcıların compile edilmiş WASM dosyalarını çalıştırabilme teknolojisine WebAssembly denmektedir. Bu sayede JS ve WASM dosyaları bir arada çalışabilecektir.

2.2.1.10 Store

- Redux , MobX, RxJS, NgRx, VueRx

  Modern Web App, uygulamanın state tutma ve sunucu ile etkileşimi sonucunda bunu yönetmek için Flux Örüntüsünün gerçekleştirimi olan bir sürü Store örneği bulunmaktadır.

2.2.1.11 Tarayıcı Dışında Bu Teknolojileri Kullanılma

Yukarıda son kullanıcı açısından çoğunlukla tarayıcılardan bahsedilmiştir ancak tarayıcı dışında Native ortamlarda da yukarıda bahsedilen teknolojileri kullanabilecek teknolojik altyapılar geliştirilmiştir. Bu sayede cross platform native uygulamalar geliştirmek mümkün olmuştur.

WebView bileşenini kullanarak altyapı sağlayan ortamlar şunlardır:

- PhoneGap Cordova (**Geliştirmeye Kapatıldı)**, NW.js, Electron

WebView bileşeni olmadan JS, CSS standartlarını kullanalım ama arkaplanda Native kodlar çalışsın diyen teknoloji yaklaşımları şu şekildedir:

- Flutter, React Native, NativeScript

Headless Browsers*:* Tarayıcı arayüzü olmadan Console/Ağ üzerinden Web sayfalarının test otomasyonlarını yapabilmek için sağlanan tarayıcılardır.

2.2.2 Geliştirici Ortamı

- Geliştirici ortamında kullandığımız kütüphane veya bulut servislerinin arttığını görebilirsiniz. Bu araçların ne yaptığına ve bunları nasıl kullanmanız gerektiğine hakim olmak gerekiyor.

2.2.2.1 Versiyon Control

- Git, GitHub, GitLab

2.2.2.2 Package Managers

- npm, yarn

2.2.2.3 Task Runners

- npm scripts, gulp

2.2.2.4 CSS Preprocessor

- SASS, PostCSS, LESS, Stylus

2.2.2.5 Module Bundlers

- Webpack, Parcel

2.2.2.6 Linter and Formatters

- Prettier, EsLint

2.2.2.7 Type Checker, Annotations vb. Dil Desteği

- TypeScript, Flow, Reason, Elm, ClojureScript, Flow, PureScript

2.2.2.8 Testing Tools

- Jest, Mocha, Storybook, Cypress, Enzyme, AVA, Jasmine, Puppeteer

2.2.2.9 Code Editor Tools

- VSCode, SublimeText, IntelliJ, Atom

2.2.3 Sunucu Ortamı ve Hosting

Sunucu ortamı HTML, CSS, JS ve diğer binary/base64 dosyalarının sunulduğu CDN, bunun yanında REST API üzerinden **JSON** sağlayan sunuculardır. Bu

sunucular farklı bulut sağlayıcı hizmetlerinden faydalanarak bu hizmetleri size sağlar. Frontend'in sunucuya erişimini de DNS sağlar.

2.2.3.1 Networking / DNS Ayarları

- DNS Ayarları
- HTTP I/O RPC,REST,GraphQL,Webhook
- REST'i nasıl kullanmalıyız? (Richardson Maturity Model)

2.2.3.2 Backend API

- **Standards:** REST, GraphQL, DNS, Domains, Ports
- **Tools:** Apollo, Relay, Swagger

2.2.3.3 Backend Frameworks

- Express, Next.js, Koa, Meteor, Sails, Feathers, Nuxt, Gatsby, VuePress, Jekklly, Hugo

2.2.3.4 Bulut / Hosting Servisleri

- AWS (Amplify, AppSync, S3, CDN), Netlify, Vercel, Google (Firebase)

2.3 Web Uygulama ve Server State Management (yönetme) konusunda belirtilen konseptlerden bazıları

2.3.1 Konseptler

2.3.1.1 Async Status

Sunucudaki verinin Client aktarılırken oluşabilecek bazı durumları (state) bulunur, bu durumlara göre ilgili UI bileşeninin davranışını değiştirmesi gerekir.

- success: İstek sonrasında yanıt başarılı bir şekilde istemciye(client) ulaştığında.
- error: İstek başarısız olup hata durumu oluştuğunda.
- loading: İstek daha network üzerinde ilerliyor ve bekleme durumunda olunduğunda.

- idle: Ekranda ilgili bileşenler gözükürken daha istek atılmamış olduğunda.

2.3.1.2 Caching

Cachelenmiş veri sayesinde, tekrar tekrar network request isteğinde bulunmayız.

- Bu, ekranda daha az loading durumu göstermemizi ve ekran geçişlerinin daha kaliteli ve akışkan olmasını sağlar.
- Daha az network trafiği olacağı ve backend daha az yoracağı için daha az kaynak kullanımı ve paradan tasarruf sağlar.

Zorluğu ise cache verinin geçersiz olduğunu anlayıp kullanıcıya yanlış veri göstermemek için bir takım ekstra mantıksal kodların yapılması gerekmesidir. Bu da ClientSide iş mantıklarını daha karmaşık hale getirir.

2.3.1.3 Callback Handling

Bu kısımda yaptığınız Call işleminin arkasından ekstra işlemler yapma ihtiyacı duyabilirsiniz. onSuccess veya onError durumlarını ayrıca ele alıp başka akışı başlatabilmesini isteyebiliriz.

Örneğin; **onSuccess** başka bir sayfaya yönlendirme, cache in-validate etme vb. gibi. Bunun için yaptığınız Redux dispatch kodunun çevresine
kendi **CallbackHandler'**ını yazarak bu yapıyı sağlayabilirsiniz.

2.3.1.4 Mutation & Optimistic Update

Sunucudaki verinin tümü veya bir parçası üzerinde birtakım değişiklikler yapıp bunun sonuçlarının UI'a yansıtılması işlemidir. Tabii burada objenin içerisinde bir değişiklik olması veya bir array elemanlı yapıda değişiklik olması vb. gibi durumlar çok farklı UI ihtiyaçları oluşturur.

- Yeni bir eleman mı eklendi?
- Yoksa eleman mı silindi?

Tüm bunların yansımasının iyi bir şekilde ele alınması gerekir. Bazı kısımlardaki güncellemelerde **Optimistic Updates** yöntemi kullanılabilir.

Optimistic Update, veriyi sunucuya gönderirken sanki başarılı gibi UI'da işlem yapıp, bir hata durumunda rollback işlemi yapar. Burada kullanıcıya Loading state göstermeden çok hızlı işler yapılıyormuş hissiyatı oluşturulur.

2.3.1.5 Background Updates

Arka planda sürekli olarak kontrol etmeniz gereken ve sisteme herhangi bir bildirim gelip gelmediği veya bir takım veri gruplarınızda güncelleme olup olmadığını ilişkin soruları sunucuya sorduğunuz mekanizmalardır.

Bu tür yapılarda;

- **Long Polling** dediğimiz belli aralıklarla fetch işlemi gerçekleştirebilir.
- **Server-Sent Events (SSE)** sunucu push teknolojisi ile istemcilere tek yönlü veriyi HTTP Connection ile geçirir.
- **WebSocket** üzerinden kurulmuş çift yönlü bir iletişimde Server istediği bilgileri istemciye geçirir.

2.3.1.6 Pagination /Incremental Loading

Aşağıdaki soruların yanıtlarına ulaşılır:

Pagination işlemleri sırasında tutulan sayfa bilgileri nasıl yönetilecek?

Bu sayfaların yüklenmesi async olarak nasıl gerçekleşecek?

Infinite Page Scrolling bunun UI ile etkileşimi nasıl olacak?

2.3.1.7 Outdated Requests

Süresi geçmiş istekler anlamına gelmektedir. Bunun farklı farklı nedenleri olabilir. Örneğin bu istekten daha sonra benzer bir istek atılmış veya kaynak üzerinde işlem yapılmak istenen kaynak öncesinde başka birileri tarafından silinmiş olabilir.

2.3.1.8 Deduping Request

Web uygulamaları kullanıcı aksiyonları sonrasında oluşan eventler doğrultusunda veya dış kaynaklardan gelen eventlerin başka eventleri üretmesi doğrultusunda çalışır.

Örneğin, siz kullanıcının her düğmeye basışında veya keyboard tuşlarına başısında network request gönderiyorsunuz ve bunlar arasında bir debouncing mekanizması yok. Özetle request cevabı sunucudan gelmeden arka arkaya request atan sisteminiz var ise bu UI da doğru veriyi gösterme aşamasında bir takım problemlere neden olacaktır. Hangisinin response valid olduğunu anlayıp diğerlerini elemeniz gerekir.

2.3.1.9 Garbage Collection ve Memory Implications

Kullanılmayan verilerin bellekten atılması ve tüm yukarıdaki işlemlerin belleğe olan etkisine ilişkindir.

2.4 Web Uygulaması Geliştirirken Kullanılabilen Açık Kaynak Kodlu Projeler

GitHub üzerinde popüler olan kütüphaneler belli başlıklar altında gruplanmıştır. Bu gruplama sırasında öncelikle 30K üzerindeki JS ve TS kütüphaneleri filtrelenmiştir.

Sorgu aslında 30K üzerindeki repoları listelemektedir:

JavaScript Repoları

```
https://github.com/search?q=stars%3A%3E30000+language%3AJavaScript&type=Repositories
```

TypeScript Repoları

```
https://github.com/search?p=1&q=stars%3A%3E30000+language%3ATypeScript&type=Repositories
```

Aşağıda 2021'de oluşturulmuş ve 2022 yılına göre güncellenmiş bir liste bulunmaktadır. Zaman içerisinde bu listelerde ufak değişiklikler ve farklılıklar olabilir ama geneli etkileme olasılığı düşüktür.

2.4.1 Örnek, Link Listesi, Best Practice yaklaşımları içeren Repolar

1. freeCodeCamp: freeCodeCamp.org's open source codebase and curriculum. Learn to code for free.
2. developer-roadmap: Roadmaps are being made interactive and have been moved to website.
3. javascript-algorithms: Algorithms and data structures implemented in JavaScript with explanations and links to further readings.
4. 30-seconds-of-code: Short JavaScript code snippets for all your development needs.
5. nodebestpractices: The Node.js best practices list.

6. Web-Dev-For-Beginners: Azure Cloud Advocates at Microsoft are pleased to offer a 12-week, 24-lesson curriculum all about JavaScript, CSS, and HTML basics.
7. realworld: “The mother of all demo apps” — Exemplary fullstack Medium.com clone powered by React, Angular, Node, Django, and man…
8. awesome-selfhosted: A list of Free Software network services and web applications which can be hosted on your own servers.
9. tech-interview-handbook: Materials to help you rock your next coding interview.
10. clean-code-javascript: Clean Code concepts adapted for JavaScript.
11. awesome-mac: □ Now we have become very big, Different from the original idea. Collect premium software in various categories.
12. 33-js-concepts: 33 concepts every JavaScript developer should know.
13. Leetcode: This essay records the course of and my emotion to this project from initialization to 10,000 stars.
14. algorithm-visualizer

2.4.2 UI Bileşenleri, Chart, Görselleştirme veya 3 Boyutlu Bileşenler Oluşturmanızı ve Render Etmenizi Sağlayan Kütüphaneler

1. vue: Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web.
2. react: A declarative, efficient, and flexible JavaScript library for building user interfaces.
3. d3: Bring data to life with SVG, Canvas and HTML.
4. react-native: A framework for building native apps with React.
5. three.js: JavaScript 3D library.
6. angular **(TS)**: One framework. Mobile & desktop.
7. angular.js: AngularJS — HTML enhanced for web apps!
8. jquery: jQuery JavaScript Library

9. Chart.js: Simple HTML5 Charts using the <canvas> tag
10. echarts **(TS)**: Apache ECharts is a powerful, interactive charting and data visualization library for browser t
11. svelte **(TS):** Cybernetically enhanced web apps
12. nestjs/nest: Nest is a framework for building efficient, scalable Node.js server-side applications. I
13. ionic-framework **(TS):** A powerful cross-platform UI toolkit for building native-quality iOS, Android, and Progressive Web Apps with HTML, CSS

2.4.3 Hazır CSS Layout veya CSS Bileşenleri Sağlayan Kütüphaneler

1. bootstrap: The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web.
2. ant-design **(TS):** A UI Design Language and React UI library
3. material-ui: Material-UI is a simple and customizable component library to build faster, beautiful, and more accessible React appl…
4. Font-Awesome: The iconic SVG, font, and CSS toolkit
5. Semantic-UI: Semantic is a UI component framework based around useful principles from natural language.
6. html5-boilerplate: A professional frontend template for building fast, robust, and adaptable web apps or sites.
7. materialize: Materialize, a CSS Framework based on Material Design

2.4.4 Utility Araç kütüphaneleri

1. axios: Promise based HTTP client for the browser and node.js
2. redux **(TS)**: Predictable state container for JavaScript apps
3. lodash: A modern JavaScript utility library delivering modularity, performance, & extras
4. moment: Parse, validate, manipulate, and display dates in javascript
5. react-router: Declarative routing for React and → Remix

6. anime: JavaScript animation engine
7. pdf.js, dayjs, immutable-js,video.js, clipboard.js, Leaflet

2.4.5 Kodun Daha Yüksek Seviyede (High Level) Yazılmasını Sağlayan Kütüphaneler

1. TypeScript **(TS):** TypeScript is a superset of JavaScript that compiles to clean JavaScript output.
2. webpack: A bundler for javascript and friends. Packs many modules into a few bundled assets. Code Splitting allows for loading…
3. yarn: Fast, reliable, and secure dependency management.
4. babel: Babel is a compiler for writing next generation JavaScript.
5. parcel: Blazing fast, zero configuration web application bundler
6. gulp

2.4.6 İçerik Oluşturma İçin Birtakım Sunum ve CMS İşlevleri Sağlayan Araçlar

1. reveal.js: The HTML Presentation Framework
2. markdown-here: *Markdown Here* is a Google Chrome, Firefox, Safari, Opera, and Thunderbird extension that lets you write email in Markdown and render them before sending. It also supports syntax highlighting (just specify the language in a fenced code block).
3. mermaid: Mermaid is a JavaScript based diagramming and charting tool that uses Markdown-inspired text definitions and a renderer to create and modify complex diagrams. The main purpose of Mermaid is to help documentation catch up with development.
4. impress.js: It’s a presentation framework based on the power of CSS3 transforms and transitions in modern browsers and inspired
5. Ghost: The #1 headless Node.js CMS for professional publishing

2.4.7 React, Vue, JS Kütüphanelerinin Detayları Olmadan Hızlı Uygulamalar Geliştirme Altyapısı (Framework) Sağlayan Platformlar

1. create-react-app: Create React apps with no build configuration.
2. vercel/next.js: The React Framework
3. socket.io: Realtime application framework
4. express: Fast, unopinionated, minimalist web framework for node.
5. gatsby: Build blazing fast, modern apps and websites with React
6. meteor: Meteor, the JavaScript App Platform
7. nuxt.js, nest

2.4.8 Kod Geliştirme Ortamı Veya Bu Ortamda Çalışacak Pluginler Sağlayan Platformlar

1. vscode**(TS)**: Visual Studio Code
2. atom/atom: The hackable text editor
3. code-server: Run VSCode on any machine anywhere and access it in the browser.
4. prettier: Prettier is an opinionated code formatter.
5. brackets

2.4.9 Fake Test Ortamı Oluşturabilmek İçin Araçlar, Sunucular ve Test Verileri Sağlayan Repolar

1. puppeteer**(TS):** Headless Chrome Node.js API
2. storybook **(TS):** The UI component explorer. Develop, document, & test React, Vue, Angular, Web Components, Ember, Svelte & more!
3. json-server Get a full fake REST API with zero coding in less than 30 seconds
4. faker.js, jest,

2.4.10 Runtime içeren JS Repolar

1. node: Node.js JavaScript runtime
2. nw.js: Call all Node.js modules directly from DOM/WebWorker and enable a new way of writing applications with all Web techno

Sonuç

Bu yazının sonuna geldik. Kitapçıktan fayda sağlayacağınızı umuyoruz. Buna benzer ücretli ve ücretsiz eğitim içerikleri geliştirildikçe mail yoluyla iletilecektir. Web sitesinden diğer eğitim içeriklerine göz atabilirsiniz.