React Web Uygulamasının Mantıksal Modeli

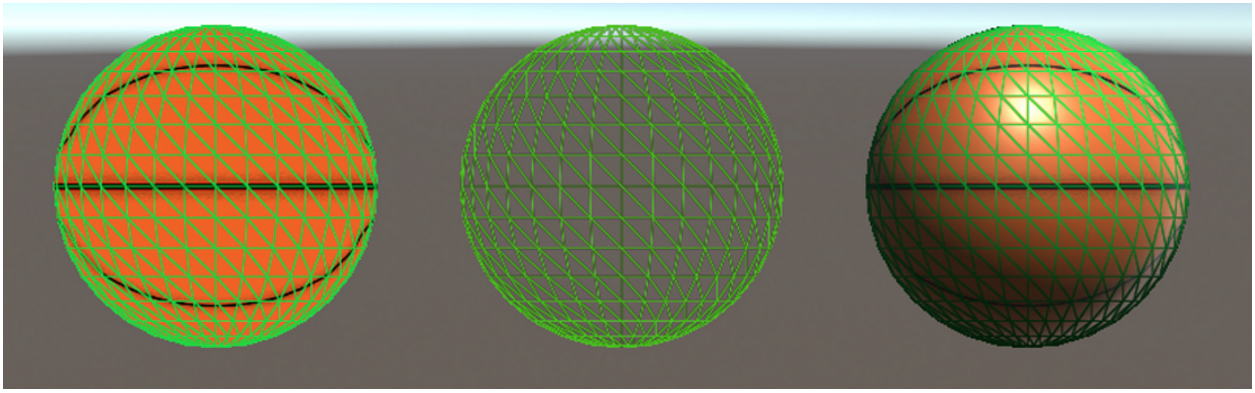
EK - Seviye 3 (Konsept Yapısında) Kullanabileceğiniz Kütüphaneler

- **Boilerplate Libraries/Frameworks**
  - **Framework**
  - **Routing**
- **State Management Libraries/Frameworks**
  - **Server State Management**
  - **Client State Management**
- **Pages Libraries/Frameworks**
- **Authentication Libraries/Frameworks**
- **Authorization Libraries/Frameworks**
- **Asset Management Libraries/Frameworks**
- **Styling Libraries/Frameworks**
- **Networking Libraries/Frameworks**
  - **Http**
  - **WebSocket**
  - **SSE**
- **Layouts Libraries/Frameworks**
- **Components Libraries/Frameworks**
  - **Component UI Libs**
  - **Visualization**
  - **Map UI Lib**
  - **Code Editor**
  - **Higlighter & Formatter**
  - **Table / Data Grid**
- **Code Quality Libraries/Frameworks**
  - **Tools**
  - **TypeSafe**
  - **Unit, DOM, Snapshot Tests**
- **Process Quality Libraries/Frameworks**
- **CI/CD Libraries/Frameworks**
  - **Platforms**
  - **Node Package Manager**
  - **Bundler**
  - **E2E Test**

Frontend uygulamaları aslında hepsi aynı mekanizma ile çalışıyor.

- Teknoloji ve Frontend Konseptleri
- Domain (Uygulamanın Yetenekleri)

Bu 2 yapı birbirine çok iyi harmanlayarak Frontend Web Uygulamalarını ortaya çıkarıyoruz. Aynı oyunlarda ve çizgi filmlerde olduğu gibi..



Senaryo Yazarlarının hikayeleri → Yönetmenler tarafından organize edilerek→ Animatörler , Görsel Efekt uzmanları aracılığı ile → Bileşenlere → Bileşenler arası etkileşime → daha sonra bir sahneye → sahneler birleşerek bir çizgi filme dönüştürülür.

Aslında Frontend Web uygulamasıda buna benzer bir yapıdır. Sadece bu yapıları iyi anlayıp kafanızda oturtursanız farklı farklı uygulamaların aslında benzer bir yapıda olduğunu algılayıp, teknik eleman olarak benzer örüntüler kurarsınız.

Şimdi size bu örüntüleri ve modelleri kendi kafamda oluşturduğum seviyeler üzerinden anlatacağım. Karşımdaki kişilere bu tarz konuları anlatırken, bu tip seviyeler üzerinden anlatmanın konunun rahat anlaşılmasını sağladığını farkettim.

Bu yazıda da seviyeler üzerinden her seferinde biraz daha derine inerek konuyu anlatmaya çalışacağım.

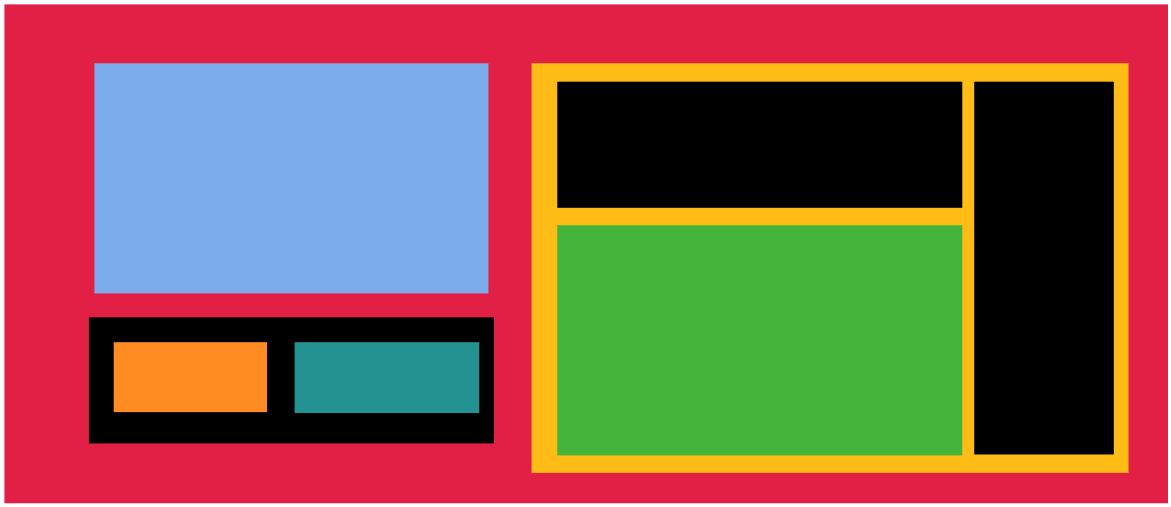
- Seviye 1 (İç İçe Bileşenler)
- Seviye 2 (Katmanlı Yapı)
- Seviye 3 (Konseptler Dünyası)

- Seviye 4 (Süreç, Konsept ve Araçlar)

Seviye 1 (İç İçe Bileşenler)

React, **Component Model** ve **Boundary Context** üzerinden bir altyapı sağlayarak sizin uygulama geliştirmenize olanak sağlar. Aslında her şey bir bileşendir bu dünyada. Ama bileşenler birbirlerine kapsayarak çok büyük kurumsal yapıları oluşturabilecek kadar da esnek bir yapıya sahiptir.

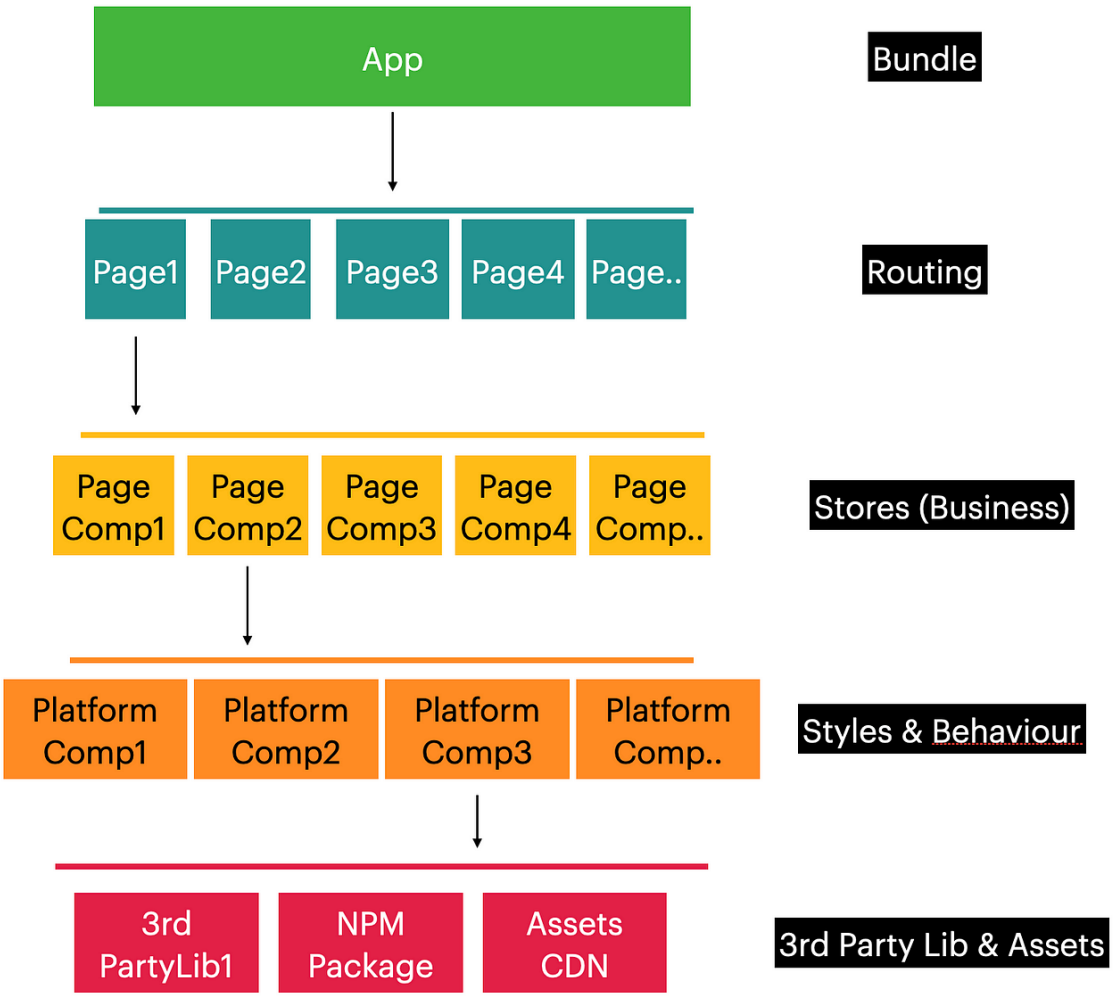
İç içe istediğiniz bileşenleri oluşturarak uygulamaları geliştirebilirsiniz.



Bileşenlerin Bileşenleri Kapsaması

Seviye 2 (Katmanlı Yapı)

Ama bu bileşenleri uygulama bazında katmanlara ayırsak ve bu katmanlara bir anlam versek herhalde konuyu daha net anlarız. Ben bu konuya **React Mimarisi 2 (Page Structure)** konusunda detaylıca değinmiştim



Frontend Katmanlı Mimari Yapısı

Her bir katmanın ayrı bir mantığı ve amacı bulunuyor. Özetle;

Uygulama Katmanında: Uygulamanın bundle edilmesi, deploy edilmesi

Page Seviyesinde: Routing konseptileri üzerinde sayfaları gruplar.

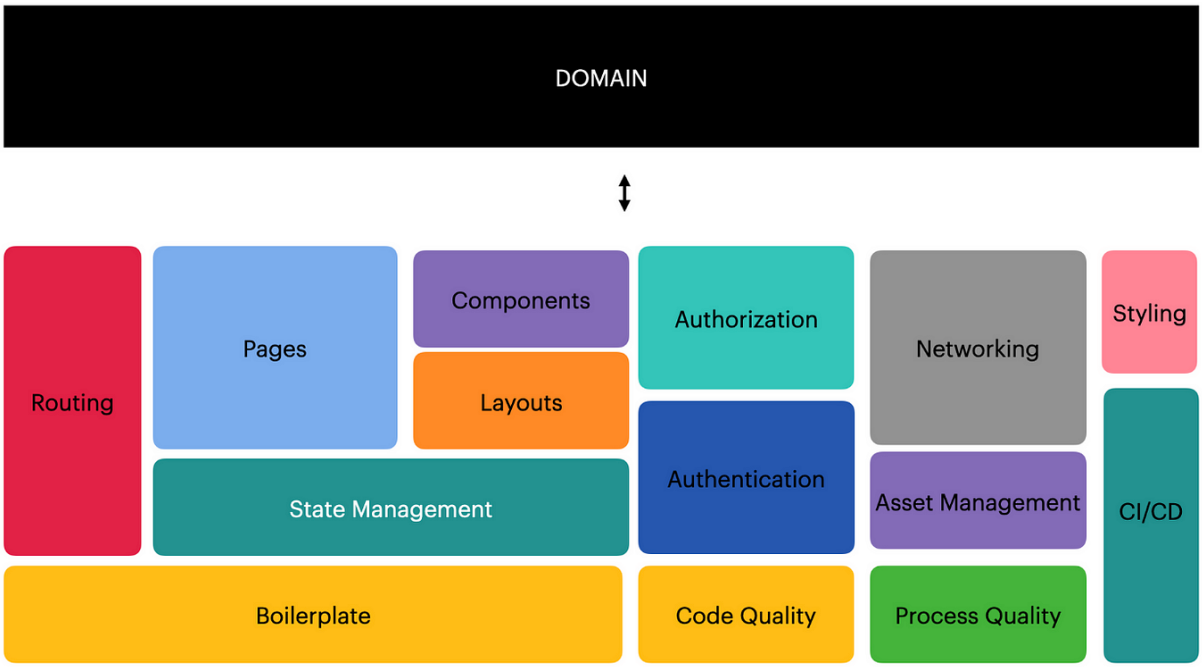
Page Comp: Business Logicleri yönettiğiniz katman

Platform Comp: Sadece Props alan görselliği ve etkileşimi sağlayan bileşen katmandır.

Dependency Lib/Assets: Projenizin dışarıdan kullandığınız 3rd Party Lib, Asset' lerin olduğu katmandır.

Seviye 3 (Konsept Yapısı)

Şimdi bu yapıların içerisinde biraz daha Frontend konseptleri açısından bakalım, konuyu biraz daha açalım.



Aslında Frontend Uygulaması dediğimiz farklı konseptilerin birbiri ile ve Domain (Uygulama Yetenekleri) ile bir bütün olarak ahenkli çalışması üzerine kurgulanır. Ben bu konular hakkında kısaca bahsedeyim.

Boilerplate

Uygulama geliştirme altyapısı ve temel klasör yapısın kurulması

- CRA (Create React App) ile BoilerPlate oluşturma
- Folder and File Structure
- Alternatif Boilerplate yapıları → Vite ve Next.JS denemek

Routing

SPA sayfa yönlendirmeleri ve bunun ile ilgili sayfaların render edilmesinden sorumludur. Bu kısımda **React-Router, Next Router , TanStack Router** kütüphanelerini kullanabilirsiniz.

- Dynamic Loading and Code Splitting
- Basic Routing and Navigation
- Routing with Redirections
- Helmet Title / Meta
- Data Passing Nested Tab Pages
- vb…

Pages

Bu kısımda tüm sayfa ve nested tab sayfaların yani Routing üzerinden yönetilen sayfaların yapısı ve yönetilmesi kısmını içeriyor..

State Management

Bu kısımda State yöntemi kütüphane kullanımlarını içeriyor, Global State, Client State ve Server State konuları

- Redux, Redux Toolkit, RTKQuery
- React Query
- Apollo GraphQL Client
- SWR
- Zustand, Jotai, Signal, vb..

Styling

Bir çok Styling altyapısı bulunuyor. Uygulamanın görselliğini bu Styling ve Theme aracılığı ile gerçekleştiriyoruz. Örneğin TailwindCSS, SaSS, Style in CSS Kütüphaneleri vb..

Asset Management

Bu kısım Icon, Image, Video dosyaları gibi görsel iletişim sağlacak içeriklerin iletişimini sağlayan bir yapıdır.

Layouts

Sayfaların içerisinde diğer Layoutların yerleşimi veya bileşenlerin yerleşiminden sorumlu Bileşenlerdir.

- Page Layouts
- Component Layouts
- Dashboard Layouts..
- Responsive Structure

Container (Page Components + Container)

Bu kısımda sayfalar içerisinde domaine göre özelleşmiş bileşenler yer alır. Bu özelleşmiş bileşen yapısında domaine göre custom bileşenler özelleştirilir ve dinamik bir şekilde veri ile beslenirler. (Container Component yaklaşımı mevcut Functional Programming yapıları içinde geçerli ve kullanılabilir..) buna engel bir yapı oluşturmuyor.

Components

Sayfaların içerisinde yer alan domain bağımsız bileşenlerdir. Bu kısımda hazır büyük bileşen kütüphanelerini kullanmak yerine kendi bileşen yapılarımızı kullanmak daha mantıklı

- Naming
- Structure
- Props and State Usage
- Rendering Mechanics
- Typography
- Color Palette (BgColor, Color, Hover…)
- States
- Border, Margin, Padding, …
- Behaviour

Authentication

Kullanıcının doğrulmasını yaptığımız SaaS veya In house uygulama katmanı, Auth0, Cognito, Amplify, Firebase gibi yapılar kullanabilirsiniz.

Authorization

Kullanıcılar sisteme girdikten sonra hangi sayfaya girebileceği , hangi ekranları görebileceği , hangi işlemleri yapabileceği konusunda yetkilendirilmesi gerekiyordur. Örnek bir kütüphane CASL

Code Quality

Kodun kaliteli olması konusunda farklı farklı yöntemler bulunur, bunlardan bazıları

- Kodun Formatlanması (Prettier)
- Hataların Linter tarafından belirlenmesi (ESLint)
- Bu işlemlerin otomatikleştirilmesi (Husky)
- Testing Yöntemleri (Playground)
- PR ve PR Review süreci

Process Quality

Bu kısımda Branch doğru isimler ile açılması, branch birleştirilirken belli işlemlerin otomatik bir şekilde yapılabilmesinin sağlanmasıdır.

- Branch Naming And Checker (Husky)
- GitHub Actions

Networking

Bu kısımda sunucu ile iletişim sağlayan XHTTPRequest, Fetch, WebSocket veya Server-Sent Event yapısı kullanılabilir veya bu yapıların üzerine 3rd party kütüphanelerin sağladığı Axios, GraphQL Request gibi kütüphaneler kullanılabilir.

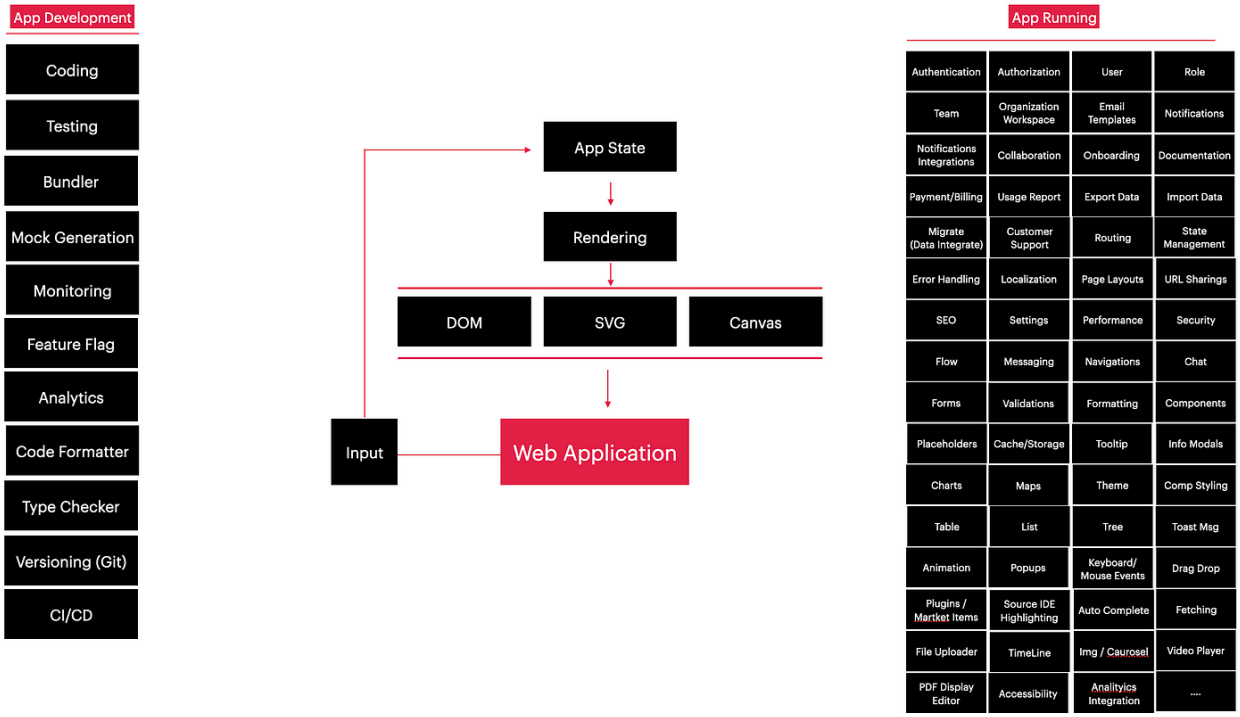
CI/CD and Cloud Platform

AWS, S3 ve Cloudfront (CDN) Frontend için oldukça ucuz bir Deployment ortamı sağlıyor. Bunun GitHub Actions ile birlikte aşağıdaki konuların yönetilmesi… .

- Automatic Deploy
- Manual Deploy

Seviye 4 (Süreç ve Ortamlar)

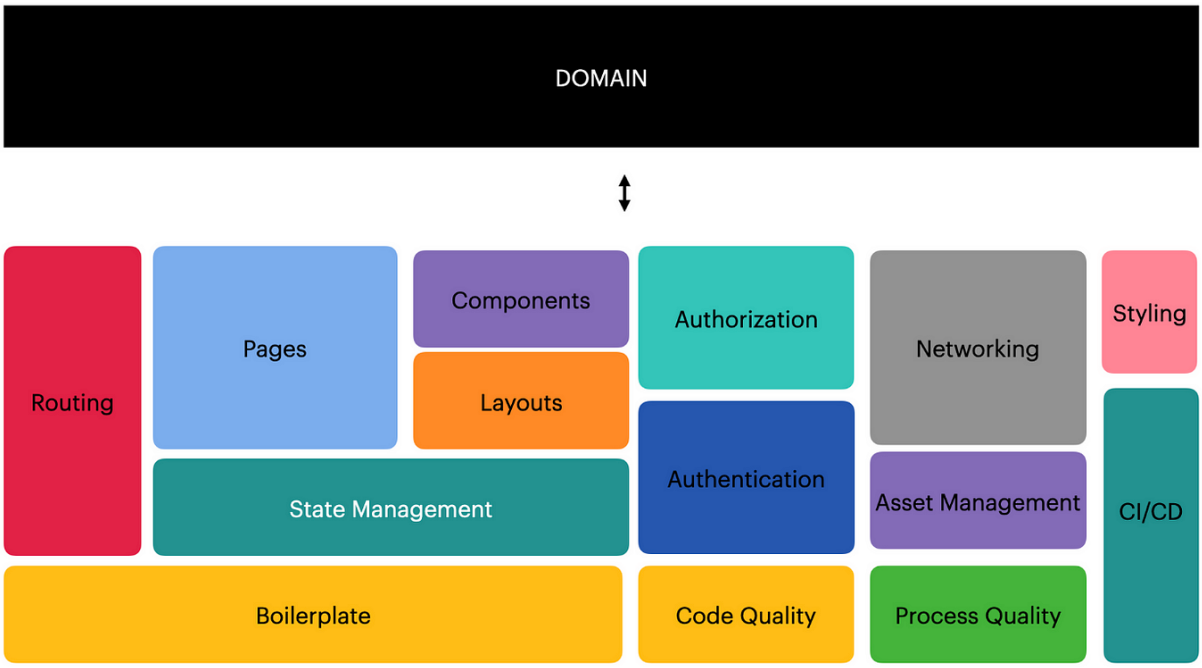
Aslında web uygulaması geliştirme araçları ve uygulamanın kendisi olarak 2 parçaya ayırabiliriz.



Yukarıda resimde göreceğiniz sol kısım uygulama geliştirme ile ilgili konuları içerirken, sağ kısım ise uygulama koşması sırasında ilgileneceğimiz bir sürü konsepti anlatıyor. Ordaki resim ise Web Uygulamasının aslında basit bir Rendering yapısı ve Kullanıcı Etkileşimlerine göre bu yapının güncellenmesinden ibaret olduğunu anlatıyor.

EK - Seviye 3 (Konsept Yapısında) Kullanabileceğiniz Kütüphaneler

Konsept yapısını yukarıda anlattık. Tabiki bu yapıyı kendi projenizde sıfırdan yapmanız gerekmiyor. Bunlar için yazılmışmış bir çok hazır ve kütüphane bulunuyor. Bu kütüphaneleri kullanmayı bilmeniz yeterli.



Bugün bu kısımlarda kullanabileceğimiz kütüphane ve framework' lerden bahsedeceğim.

Boilerplate Libraries/Frameworks

Framework

- Next.JS
- Remix
- Gatsby
- Vite — (use Rollup)
- Parcel
- Create React App

Routing

- React Router (Remix)
- Next.js Router
- Tanstack Router

State Management Libraries/Frameworks

Server State Management

- React Query
- SWR
- Apollo GraphQL Client
- RTK Query
- URLQ

Client State Management

- Props Drilling and useState
- React Context
- Redux Toolkit
- Zustand
- Jotai
- Signals

Pages Libraries/Frameworks

Bir kütüphaneye ihtiyaç duymaz, routing üzerinden oluşturulan kavramsal bir bileşen türüdür, diğer bileşenlere benzerlik gösterir.

Authentication Libraries/Frameworks

- Auth0
- Cognito

- Octa
- Keykloak
- Amplify
- Firebase
- Supabase
- ForgeRock
- Azure AD

Authorization Libraries/Frameworks

- CASL

Asset Management Libraries/Frameworks

- Google Fonts
- Icomoon

Styling Libraries/Frameworks

- CSS
- Inline Style
- SaSS
- TailwindCSS
- Styled Components
- Emotion

Networking Libraries/Frameworks

Http

- Fetch
- Axios

- GraphQL-Request
- ApolloClient

WebSocket

- Socket.io Client
- useWebSocket
- Websocket Link
- Redux WebSocket

SSE

- React Hooks SSE

Layouts Libraries/Frameworks

- React Grid Layout
- Golden Layout
- React Masonry
- FlexLayout React

Components Libraries/Frameworks

Component UI Libs

- Material-UI
- Ant Design
- Fluent UI
- Chakra UI
- Radix UI
- Blueprint
- Semantic UI React

- Evergreen
- Grommet
- Rebass
- Mantine
- Next UI
- ThemeUI
- PrimeReact

Visualization

- VisX (low-level visualization components)
- D3.js
- Rechart
- Flow.js
- Cytoscape
- Processing.js
- Three.js
- Nivo.js
- Vis.js

Map UI Lib

- Leaflet
- Google Map
- React Map GL
- Pigeon Maps
- React Simple Map

Code Editor

- Monoca Editor

- Code Mirror
- Ace Editor

Higlighter & Formatter

- React Syntax Highlighter
- JSON Viewer
- Log Viewer

Table / Data Grid

- Tanstack Table
- Metarial UI Table
- Antd Table
- React Table
- React Data Grid
- React Pivot Table

Code Quality Libraries/Frameworks

Tools

- Prettier
- Eslinter

TypeSafe

- PropTypes
- Flow
- TypeScript

Unit, DOM, Snapshot Tests

- Vitest

- Jest
- React Test Renderer
- React Testing Library

Process Quality Libraries/Frameworks

- Husky

CI/CD Libraries/Frameworks

Platforms

- GitHub Actions
- Jenkins
- GitLab CI/CD
- Bamboo
- Travis CI
- Circle CI
- BitBucket
- Teamcity
- Azure Devops

Node Package Manager

- npm
- yarn
- pnpm

Bundler

- webpack
- rollup

- parcel
- Turbopack

E2E Test

- Cypress
- Puppeteer
- Playwright