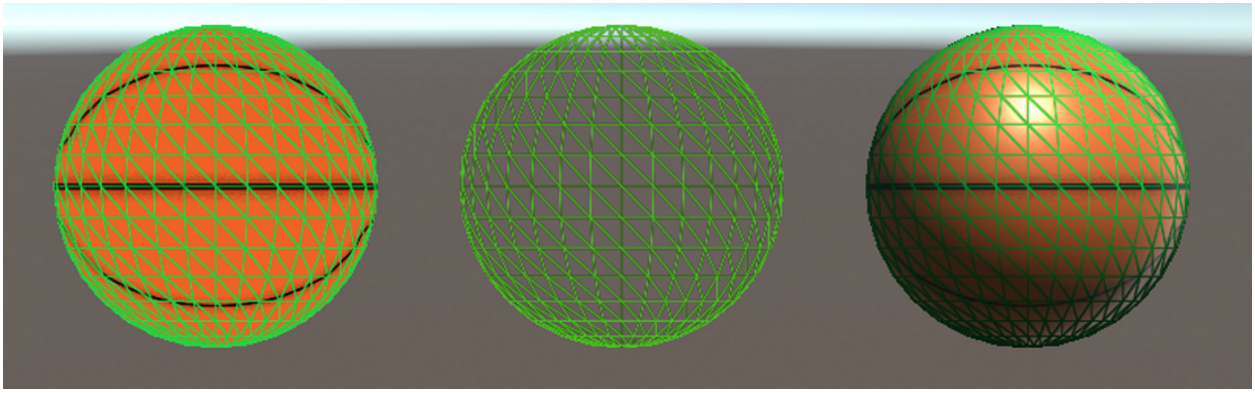
React Web Apps Conceptual Framework (pdf)

> **Son Güncellenme Tarihi:** 27.08.2023
> **©Copyright:** OnurDayibasi

In reality, all frontend programs use the same mechanism.

- Technology and Frontend Concepts
- Domain (Application Capabilities)

Frontend Web Applications are created by expertly combining these two structures. As in video games and cartoons.



The stories of the scriptwriters → are organised by the directors → through animators, visual effects specialists → into components → interaction between components → then into a scene → scenes are combined into a cartoon.

In fact, Frontend Web application is a similar structure. Only if you understand these structures well and sit in your head, you will perceive that different applications are actually in a similar structure and establish similar patterns as a technical staff.

Now I will explain these patterns and models to you through the levels I have created in my own mind. When I explain such subjects to the people I meet, I realised that explaining them through such levels makes it easier to understand the subject.

In this article, I will try to explain the subject by going a little deeper each time through the levels.

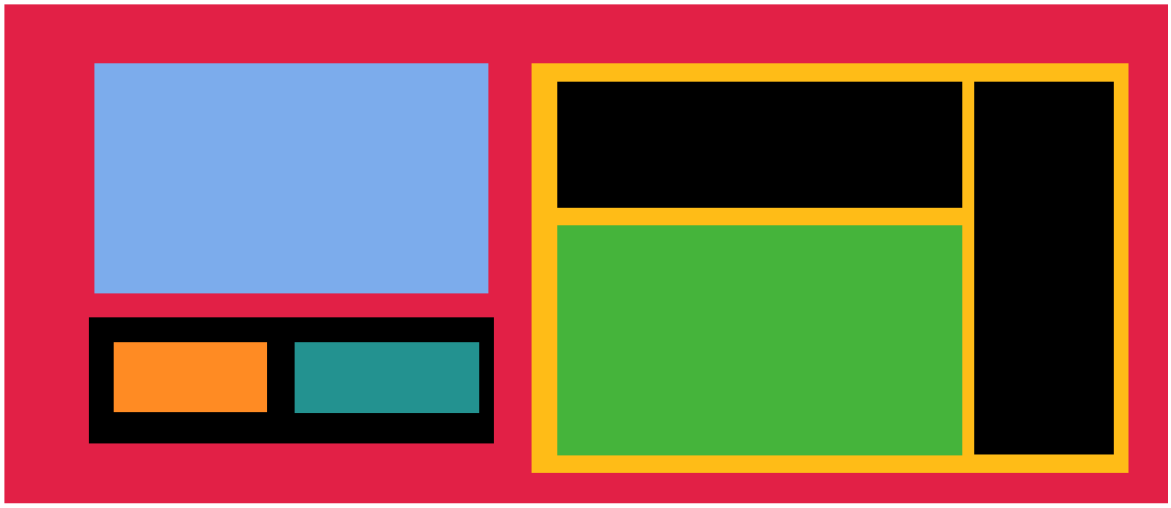
- Level 1 (Composition)
- Level 2 (Layered Structure)
- Level 3 (World of Concepts)

- Level 4 (Process, Concept and Tools)

Level 1 (Composition)

React allows you to develop applications by providing an infrastructure through **Component Model** and **Boundary Context**. In fact, everything is a component in this world. But the components are flexible enough to create very large corporate structures by covering each other.

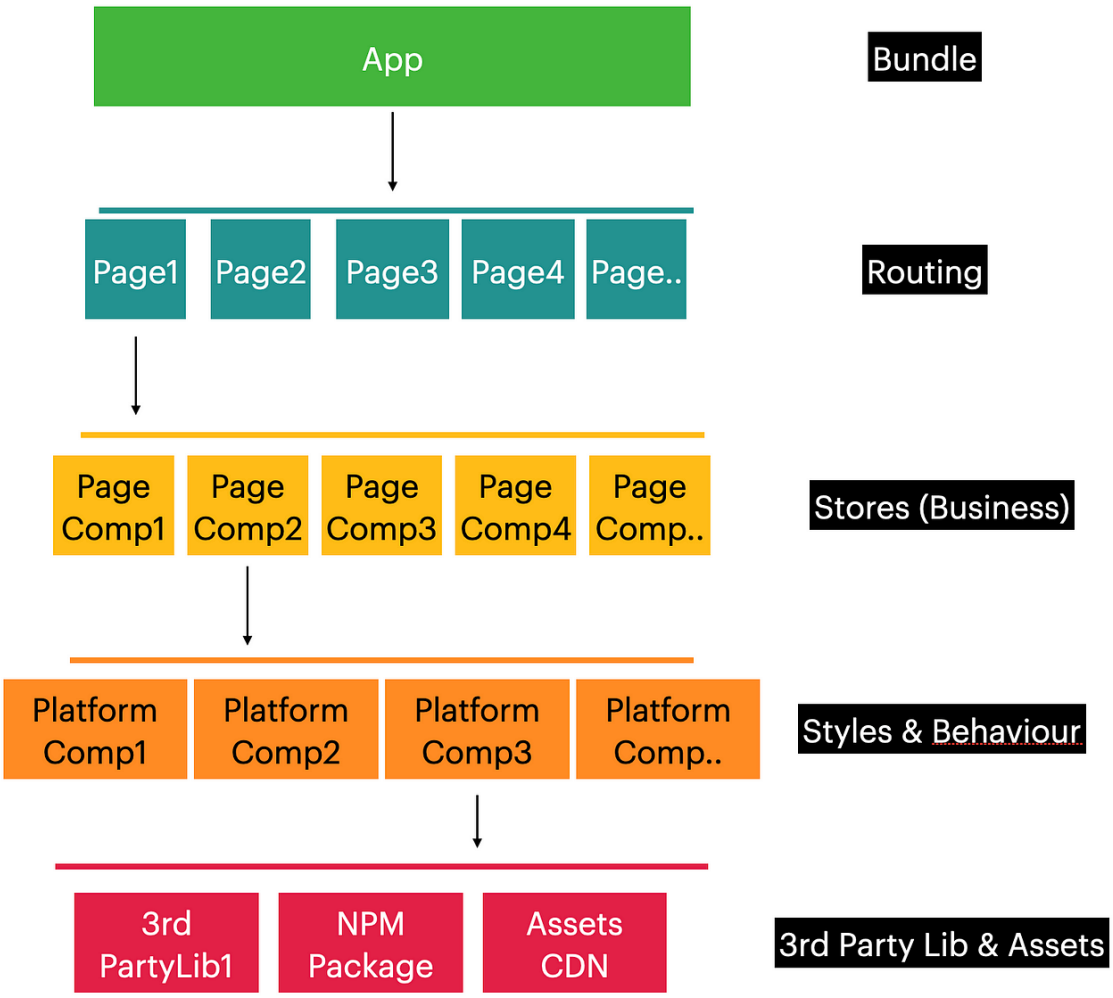
You can develop applications by creating the components you want nested. ⇒ (Composition)



Component Composition of Components

Level 2 (Layered Structure)

But if we divide these components into layers on an application basis and give a meaning to these layers, we can probably understand the subject more clearly. I have touched on this issue in detail in React Architecture 2 (Page Structure)



Layered Structure

Each layer has its own logic and function. In conclusion;

Application Layer: Bundling and deploying the application

Page Level: Groups pages on routing concepts.

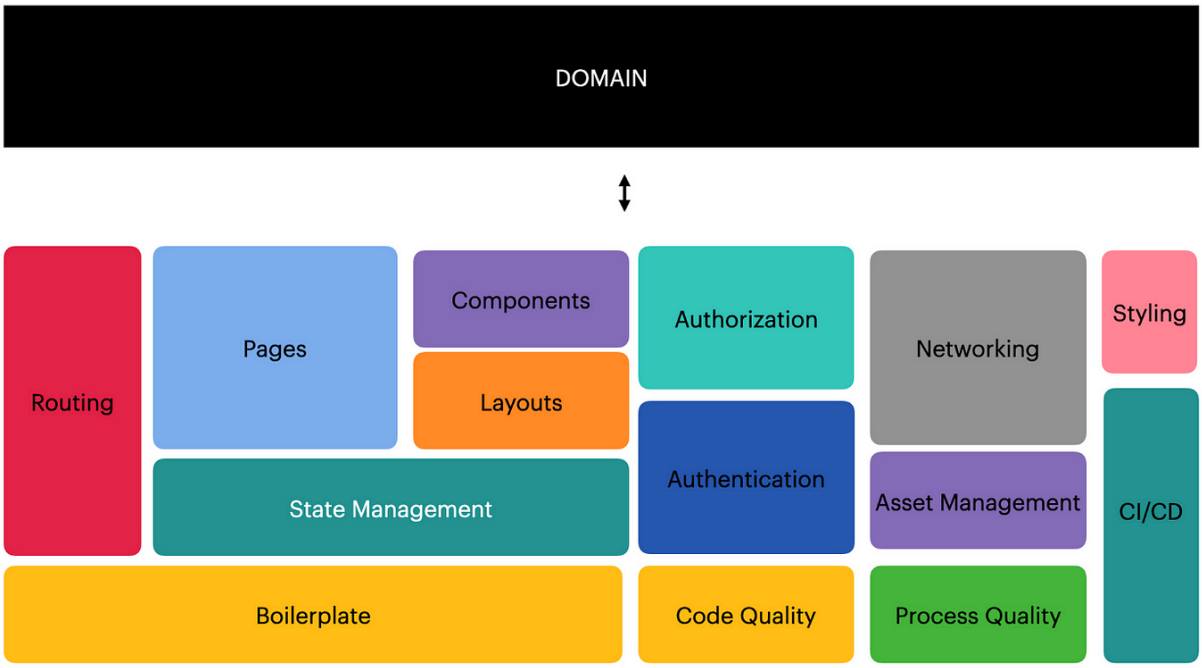
Page Comp: The layer where you manage Business Logics

Platform Comp: It is the component layer that provides visualization and interaction that only takes Props.

Dependency Lib/Assets: It is the layer with 3rd Party Lib, Assets that you use from outside of your project.

Level 3 (Concept Structure)

Now let's look a little more in terms of Frontend concepts within these structures, let's open the subject a little more.



In fact, Frontend Application is built on the harmonious work of different concepts that we call Frontend Application with each other and with Domain (Application Capabilities) as a whole. Let me talk briefly about these issues.

Boilerplate

Establishment of application development infrastructure and basic folder structure

- Create Boilerplate with CRA (Create React App)
- Folder and File Structure
- Alternative Boilerplate structures → Try Vite and Next.JS

Routing

SPA is responsible for page routing and rendering of related pages. You can use **React-Router, Next Router, TanStack Router** libraries in this section.

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

Pages

This section includes the structure and management of all pages and nested tab pages, that is, the pages managed via Routing.

State Management

This section contains the State method library uses, Global State, Client State and Server State.

Styling

There are many Styling infrastructures. We realise the visuality of the application through this Styling and Theme. For example TailwindCSS, SaSS, Style in CSS Libraries etc..

Asset Management

This section is a structure that provides the communication of content that will provide visual communication such as Icon, Image, Video files.

Layouts

Components responsible for the placement of other Layouts or the placement of components within the pages.

- Page Layouts
- Component Layouts
- Dashboard Layouts.

- Responsive Structure

Container (Page Components + Container)

In this section, there are components specialised according to the domain within the pages. In this specialised component structure, custom components are customised according to the domain and fed with data dynamically. (Container Component approach is valid and can be used in existing Functional Programming structures...) does not create an obstacle to th

Components

They are domain independent components inside the pages. In this section, it is more logical to use our own component structures instead of using ready-made large component libraries

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

Authentication

You can use SaaS or In house application layer, Auth0, Cognito, Amplify, Firebase where we authenticate the user.

Authorization

After users enter the system, they need to be authorised about which page they can enter, which screens they can see, which operations they can perform. A sample library

CASL

Code Quality

There are different methods to ensure the quality of the code, some of them are

- Formatting the code (Prettier)
- Detection of errors by Linter (ESLint)
- Automation of these processes (Husky)
- Testing Methods (Playground)
- PR and PR Review process

Process Quality

In this section, the branch is opened with the correct names and certain operations can be done automatically when merging the branch.

- Branch Naming And Checker (Husky)
- GitHub Actions

Networking

In this section, XHTTPRequest, Fetch, WebSocket or Server-Sent Event structure that provides communication with the server can be used or libraries such as Axios, GraphQL Request provided by 3rd party libraries can be used on these structures.

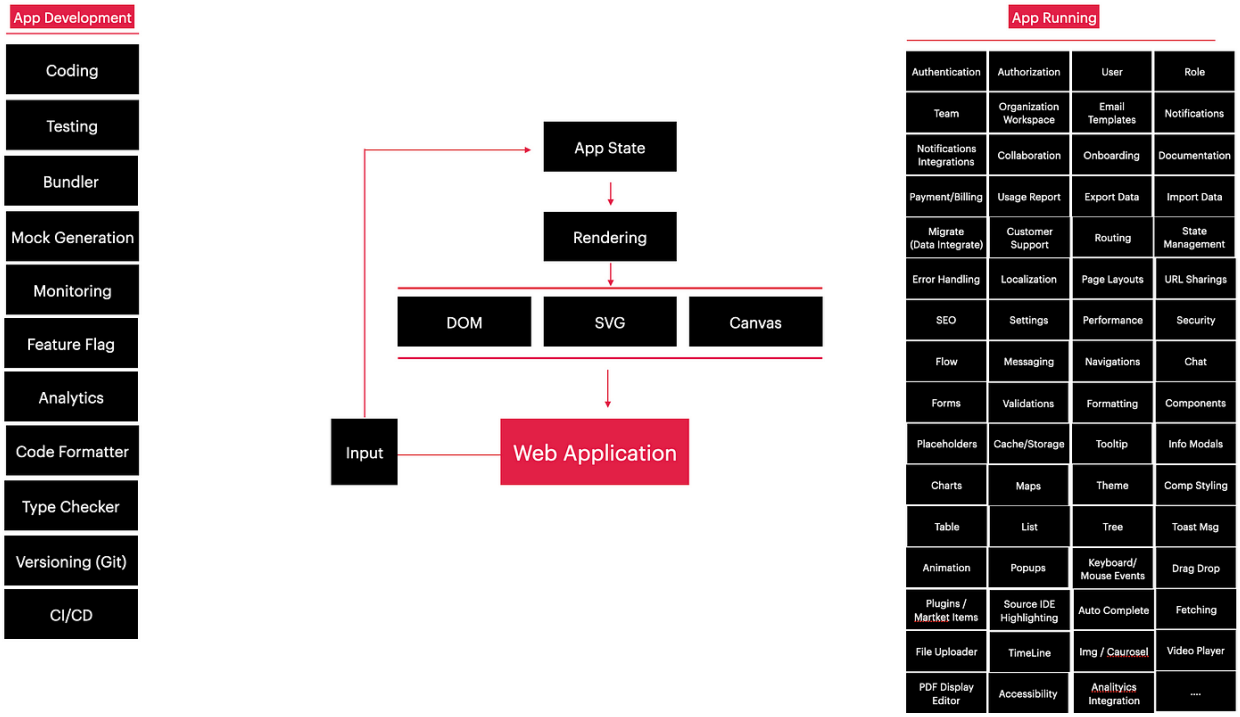
CI/CD and Cloud Platform

AWS, S3 and Cloudfront (CDN) provide a very cheap Deployment environment for Frontend. Managing the following issues together with GitHub Actions... .

- Automatic Deploy
- Manual Deploy

Seviye 4 (Process and Environments)

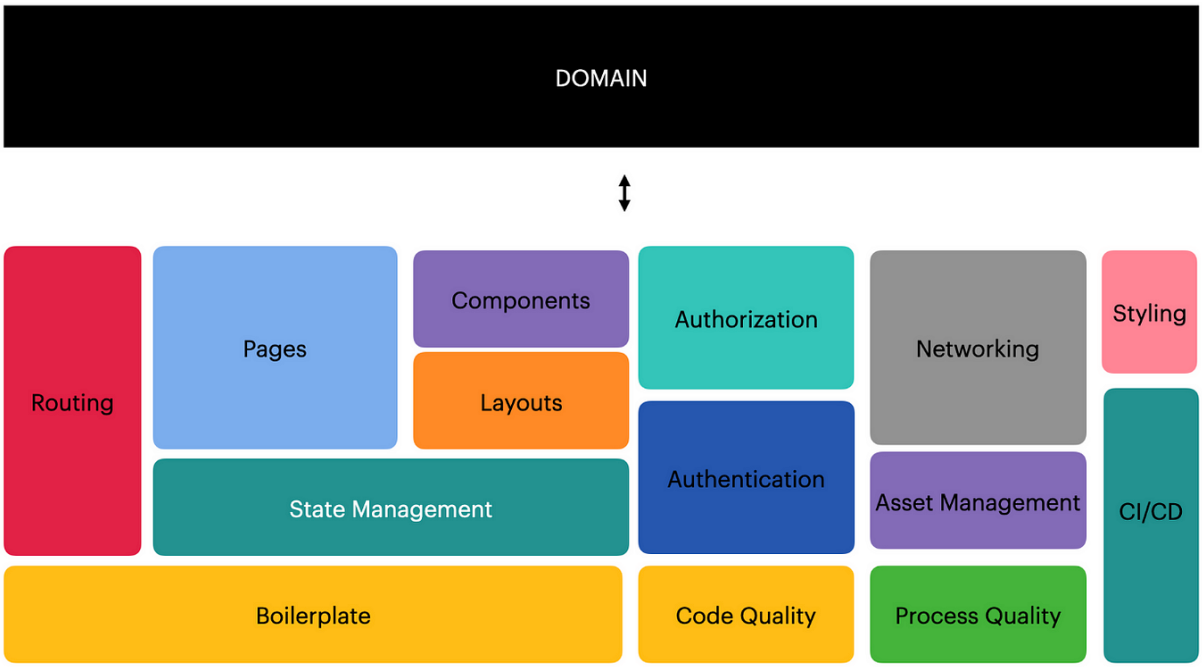
In fact, we can divide it into 2 parts as web application development tools and the application itself.



The left part you can see in the picture above contains topics related to application development, while the right part explains a lot of concepts that we will deal with during the application run. The picture there explains that the Web Application is actually a simple Rendering structure and updating this structure according to User Interactions.

Annex - Libraries You Can Use at Level 3 (Concept Structure)

We explained the concept structure above. Of course, you do not need to make this structure from scratch in your own project. There are many ready-made and libraries written for them. It is enough to know how to use these libraries.



Below I will talk about the libraries and frameworks that we can use in these concepts.

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