How to Become a Frontend Master? (pdf)

Last Updated Date: 13.03.2023
©Copyright: OnurDayibasi

INTRODUCTION

You do not think that you have deep knowledge in the Frontend area, although you have studied many learning contents, you are not confident in Frontend when you start a new job, you cannot develop your codes although you comprehend the existing architecture, LearnReactUI.dev is the website where you can meet all your relevant requirements in a single place.

Although the focus of the learning content on this site is *React*, the main goal here is to introduce UI development. The learning content is based on the essentials and mechanics of the real work done; therefore, regardless of the UI library -React, Vue, Angular, Svelte, etc.- you use, you will be able to learn many subjects at this website irrespective of the language.

Any person just at the beginning of his career may ask, “**Which learning route should I follow?**”

There is no prepared route applicable for every person since each individual's learning method and requirements are different. In the other learning courses, you can see roadmaps where the topics are arranged to be successive and grouped according to their simplicity. For example:

- https://roadmap.sh/frontend

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

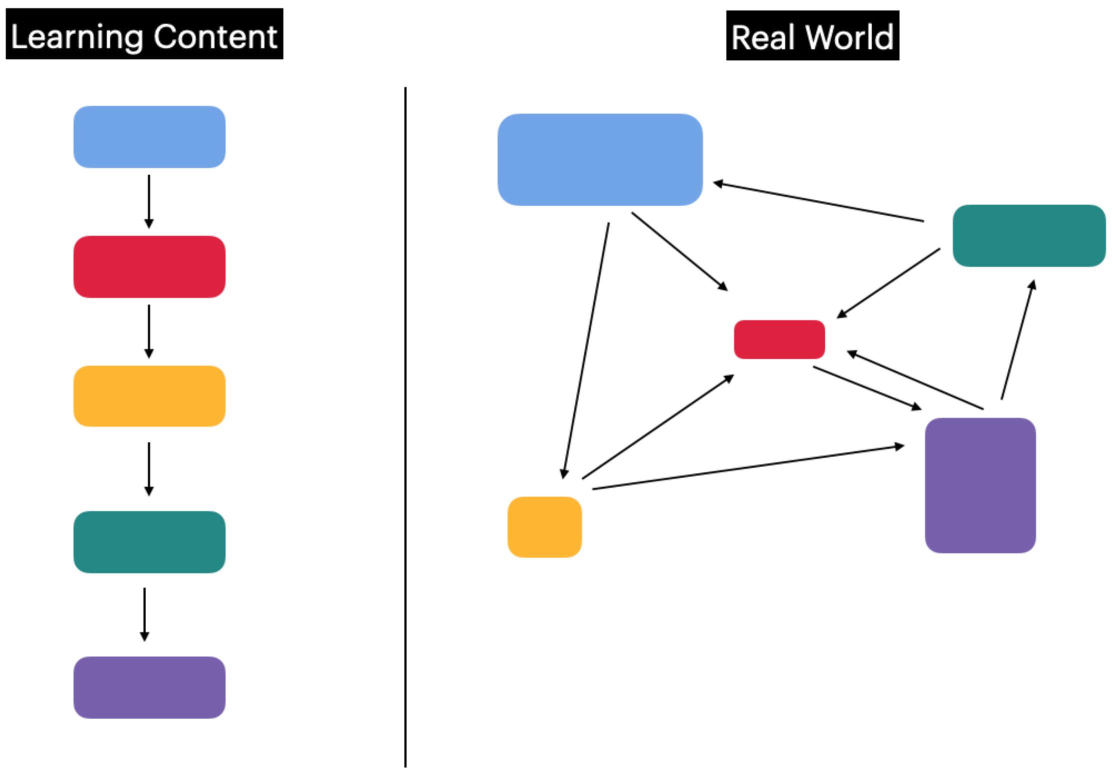
It seems an easy and quick process to gain expertise by following these sequential steps, but we encounter a quite different mechanism running in real life. You can see with several examples and in connection with the projects you develop that there are many interconnected atomic structures instead of the consecutive structures placed one under another and that the content of these concepts is much more, and the connections are stronger.

This can also apply to learning. “What is **Real Learning**?”

“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, and how you talk to your family, friends, and classmates about it. That’s what learning is. **“(Dennis Littky)**

For better clarification of the subject, an explanation is provided with the images below:

- The example given on the left side is the method commonly used in the market.
- The example on the right side shows the knowledge required to develop an application in the real world.



Connection Between the Learning Content and the Information in the Real World

Generally, we can see that the educational contents are arranged in a way to give the instructional concepts and lessons in a predefined and successive order. A single topic is focused on, and this topic is explained instead of describing how the interconnection should be made between these contents and which type of applications should be created.

But this is not the real world. In the real world, knowledge is in the form of interconnected nodes. You may need to know and use some information and concepts at a higher level, while it may be sufficient for you to know some on a limited scale. Some topics are provided on every platform, but you may find information on some topics on Stackoverflow or in the depths of the source codes of the GitHub open-source projects, or you may need to find a similar issue and implement it into your project.

When creating a component or a website, you need to use and combine all the information concerned in harmony. During this creation process, it is critically important for you to have full knowledge of many concepts, know the best practices and be aware of many hidden details. The essential point is that you must know how to harmonize the knowledge gained.

In other words, the knowledge you will gain in the specified courses may introduce, for example, the ingredients that can be used, the tools, and the tutorials related to the tools. Still, it will **NOT** give the meals (components) you can cook with all these ingredients and tools.

LearnReactUI.dev site enables you to reach your goal in a really fast way with the learning content, including examples close to real-world practices, and for doing this, it takes the learning method, which provides the aforementioned relations and depth as the basis.

AN EXAMPLE FROM THE REAL WORLD

An example from the real world will better explain the subject. Assuming the product team requests you to develop a component according to a UIX design:

Component Requirements

- The data should be aligned like a table, and at the same time, it should be possible to apply any CSS design desired.
- Pagination and Scroll should be available.
- For the text exceeding the cell in a row, the width or size of that cell should not be increased. Instead, (…ellipses) should be written, and a tooltip should appear.
- For the texts in multiple rows or over a certain number of characters, the continuation should be shown in detail as a tooltip.
- The data should be fetched from the server, and only the related sections should be rendered on the screen.
- Even if the number of rows is high, interaction with the user should be well, and smooth Scrolling should be possible.

Concepts To Know for Meeting the Requirements

- The layout to be used due to the alignments in the visual of the component should be known: Table **(DOM Knowledge)**
- Assuming that details are present for each row of this table, and we need to show 2 cells as a whole in a collapsible structure, we need to know the HTML Table Colspan concept. **(DOM Knowledge)**
- To provide an animated collapsible with an accelerated spring motion, some knowledge of **(Maths and WebAnimations API)** is required.
- For creating font and icons or your CSS Animated Elements, **(CSS Knowledge) is required.**
- For fetching any data from a server, knowledge of (**Fetch, HTTPRequest, AJAX, and JSON**) is required.
- For storing that data in a memory or browser storage, knowledge of **(JS, Structures, and LocalStorage)** is required.
- For managing the data in a controlled way, knowledge of **(Data Layer Libraries, Redux, and React-Query)** is required.
- For rendering these to the screen, knowledge of **JSX, React, etc. (Vue, Svelte, Angular)** is required.
- For creating tables on the screen with full performance, it is required to know the **Virtualized** logic and Frontend caching.

And it is also very important for you to envisage and harmonize/combine many subjects not specified herein. For example:

- How do you do time planning during this process?
- Which development and automation tools do you utilize?
- What do you use for ensuring quality code, such as ESlint, Test Libraries, etc.?
- Which open-source libraries can you utilize, and how can you integrate them into your system?
- In which order and how long do you use them? Which processes do you interconnect?

As you can see, the real-world practices are somewhat different from the successive lectures in the courses and from learning through reading. We can ensure faster and deeper learning by learning the topics with real-life project examples.

For this purpose, we have founded the LearnReactUI.dev website. Let me shortly tell you why you should prefer the LearnReactUI.dev website in this regard.

1. Difference between LearnReactUI.dev and the Others

1.1 The Learning Content

First of all, it is a learning site that will prepare you for your project development process in the real world. For this purpose, you will find a course about the practices based on real project examples.

During my software development process for 20 years, I have taken place in UI development:

for applications running in different Backend environments

- On-Prem Machine, Near Real Time
- WebSocket and HTTP on Cloud

with teams of varying sizes

- Small Size Teams (5-20)
- Mid-size Teams (20-50)
- Very Large Teams (50-100)

in different sectors

- Military
- Insurance and Banking
- Governmental Institutions

- APM (Application Performance Monitoring)

by using different languages and libraries on UI

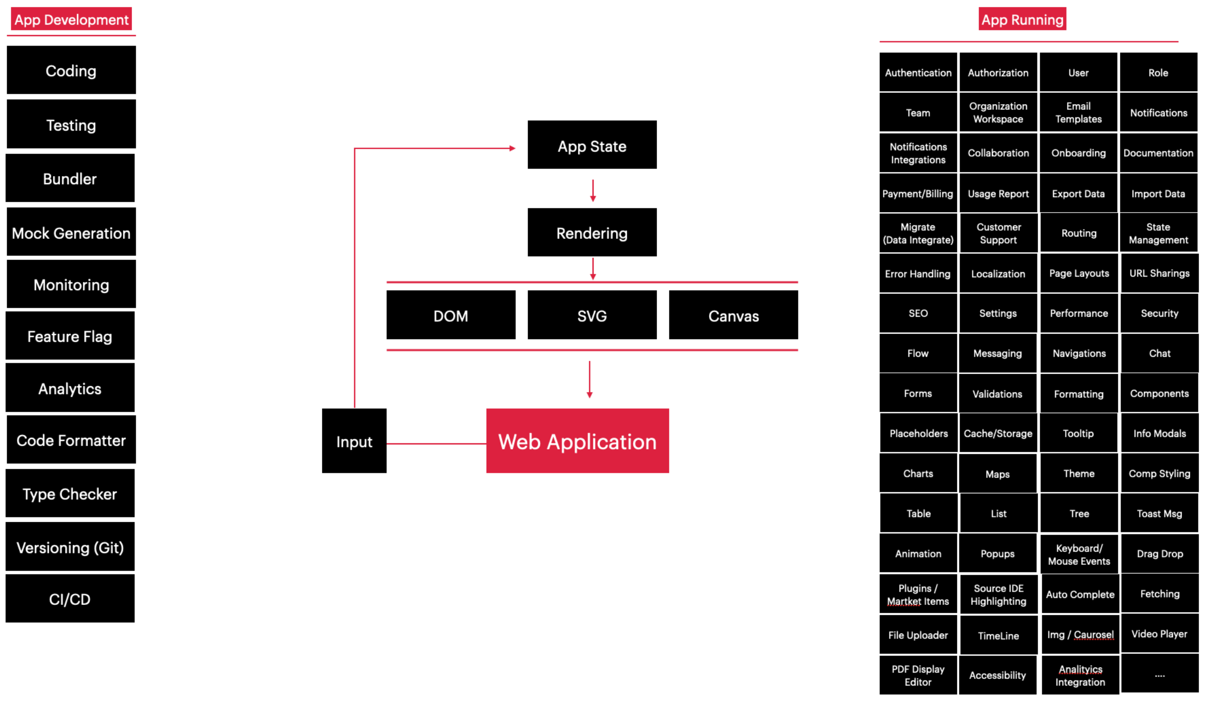
- Java, Swing, JavaFX
- JS, JQuery, ExtJS
- JS, React

I have gained lots of experience from these projects. In the learning content I will prepare, I will try to explain these experiences to you through examples close to reality by using up-to-date technologies and libraries.

Let’s Take a Closer Look at Web Applications:

The figure below is composed of three sections. You need to focus on these subjects in the UI Libraries independently from the daily technological changes while developing any Web Application.

- On the left side, the concepts on which the developers work during App Development are shown.
- The section in the middle describes the **App Running** logic.
- The concepts processed during the **App Running** phase are given on the right side.



LearnReactUI tries to explain the standards, the similar UI running mechanics, and the common architectures in this way.

- Standards (HTTP, HTML, CSS, … etc.)
- UI Running Mechanics (Fetch, Validate, Save the Screen Data, etc.)
- Color combinations, Layout structures, etc.
- Understanding the needs of running in different environments (Desktop, Mobile, Tablet …)
- Design Patterns
- Etc. …

1.2 The Learning Process

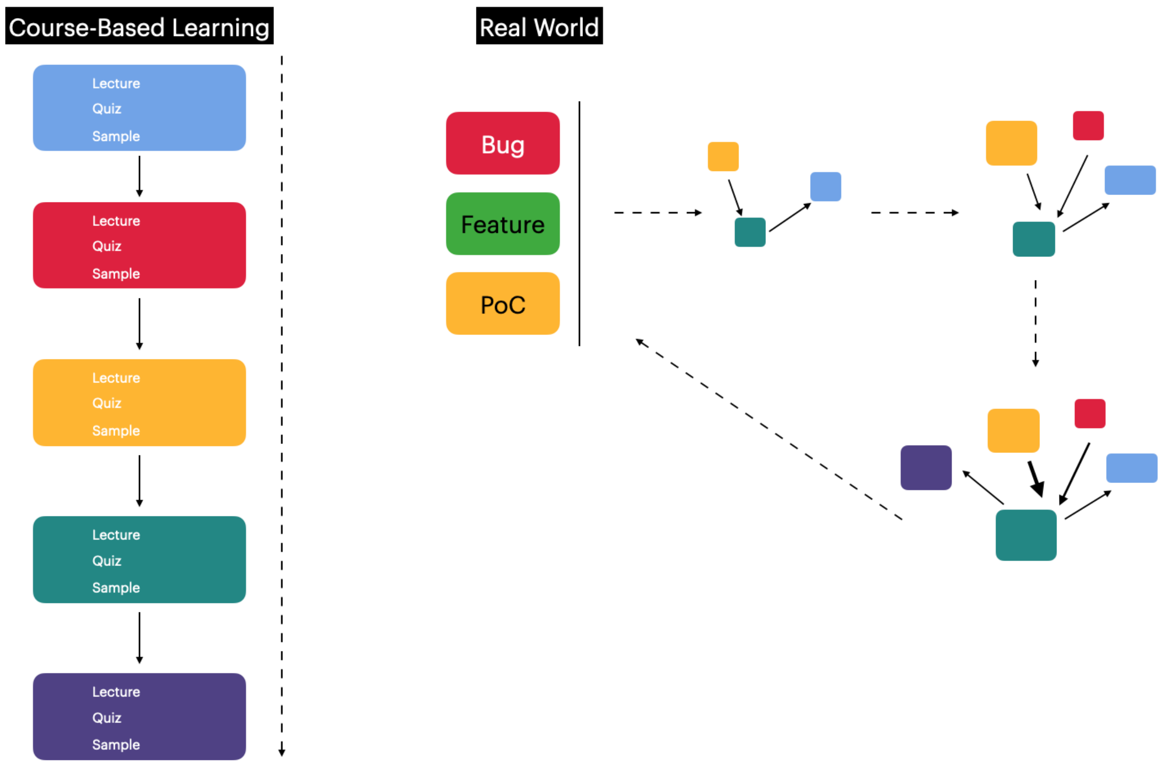
The courses in the learning sites are generally in a specific line and order, as mentioned above, and they are provided in the below-mentioned structure:

- Details of the Topic
- Quiz
- Example

While this method of learning could be a useful way to follow if you are not familiar with a topic and you are trying to learn it for the first time, you may have to deal with unnecessary details if you have some knowledge of that topic or you may forget many topics you remember at the beginning as the course progresses.

However, in the real world, we experience learning through Mistakes, the Development of Capabilities for Projects, or PoC (Proof of Concept) activities. During the development of projects, we also feel time pressure. We need to pay high attention in many respects during the development process:

- Not to damage the previous Legacy code
- To be in harmony with the architecture
- To complete the work within the given time
- To develop maintainable and reusable codes for the future



As mentioned above, the learning method in the Real World will be **more effective and educatory** than learning through educational content. Another thing to consider is your method of establishing knowledge and connections in time. While connections between the concepts are hard to occur in the method specified on the right side, the experiences you gain during your real work teach you naturally to make interconnections between these concepts and use them in harmony.

LEARNREACTUI.dev website demonstrates examples similar to the real application components, allowing you to learn the topic most effective and fastest.

In the meantime, it enables you to select any topic of study you want in relation to your work and the example of that topic. For example:

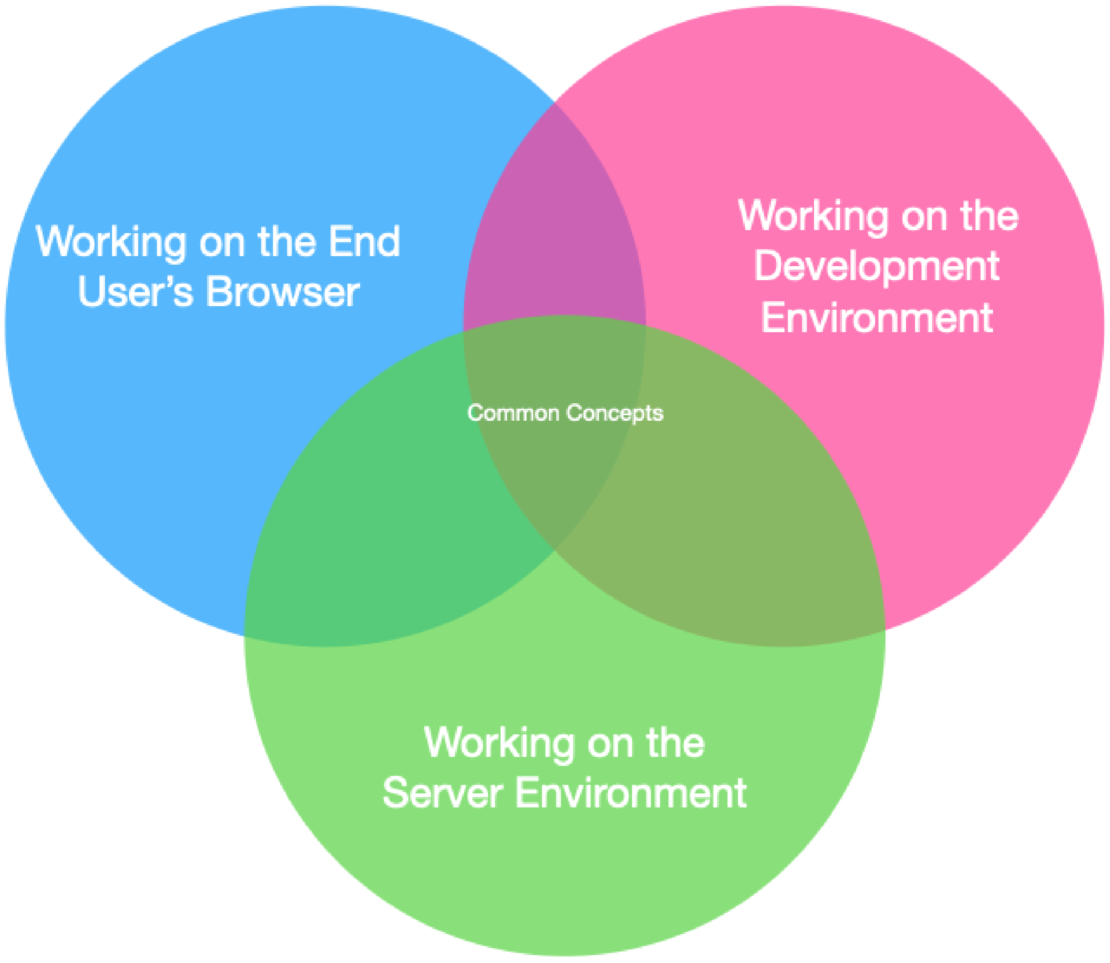
- Tables
- Charts and Dashboards
- Navigation

You can select the more important topic for you, start studying and learn it with a focus according to your purpose of learning.

1.3 LearnReactUI.dev Gives the Ability to Think Outside the Box

Looking from different perspectives regarding the Frontend provides major advantages to the developers.

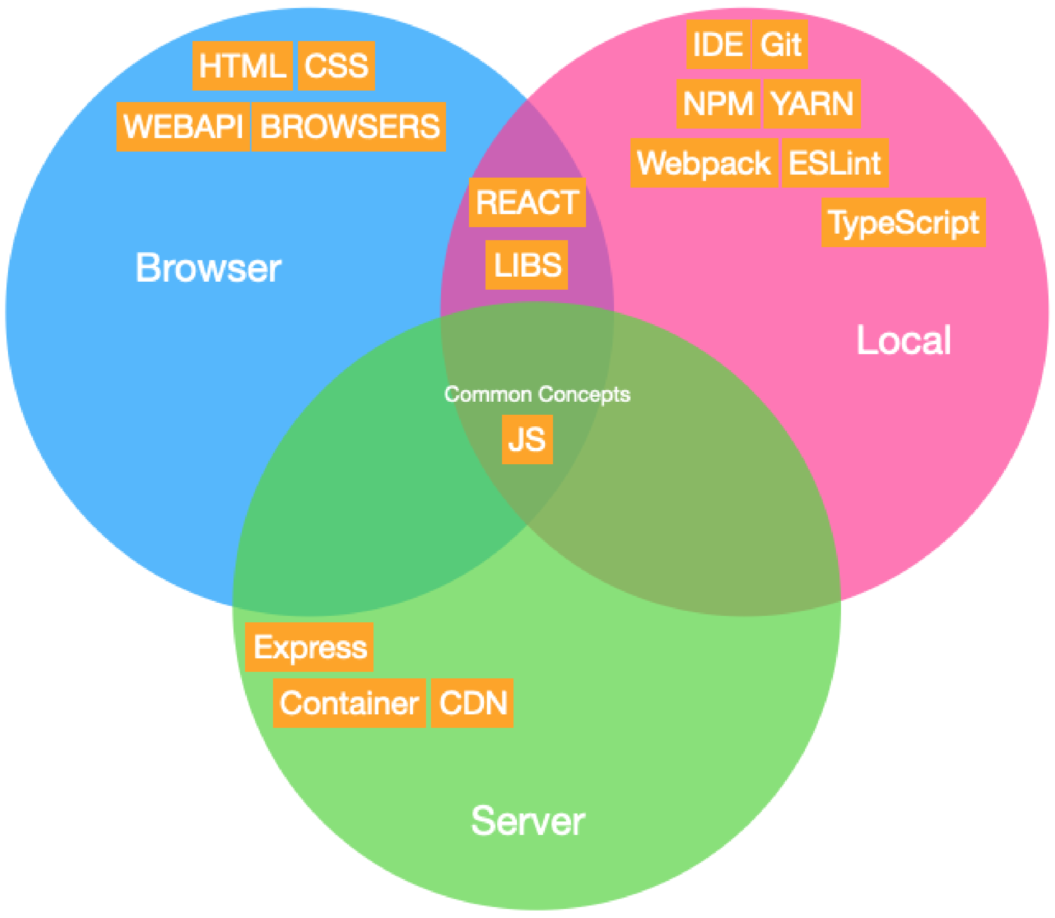
- Browser Environment
- Development Environment
- Server Environment


Working on the End User's Browser
Working on the Development Environment
Common Concepts
Working on the Server Environment

- Concerning Frontend, initially, the developers use tools and libraries **in the development processes in their environment**.
- In the following process, the related outputs **are *deployed* to specific servers,** and services and tools are used while rendering service **to the users from these servers.**
- In addition, there is a section where the user monitors on the browser the data, such as HTML, CSS, and JS, has taken from the server via the browsers and the other data (JSON, Audio, Video, Image, Base64) and where the user manages the user interactions.

In sum, three environments could be mentioned in total. When you learn the exact position of the technology and concepts within this cluster, you can better comprehend many topics.

In the figure below, you can see the positions of the concepts within the cluster more clearly.



- **Browser Environment:** Processing of files belonging to web applications such as HTML, CSS, JS and image and PDF, etc., in the browsers of your computer from the environments such as Server/File system/CDN.
- **Server Environment:** HTML, CSS, and JS files are rendered from the server. The server's responsibility here differs only from the level of providing REST over API to Template rendering and HTML generation. Your method of Rendering varies, and the server's responsibility and the Framework to be used vary accordingly.

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

- **Local Environment:** At the local, we run both the server and the browser environments. Apart from this, several development and high-level abstraction tools facilitate our life.

After these environments, the technologies and libraries and their running environments are explained with examples; it will be very simple and quick for you to comprehend and utilize a similar technology you encounter for the first time.

In sum, the LearnReactUI.dev website allows you to look at the subjects differently.

1.4 LearnReactUI.dev Uses Different Libraries and Shares the Related Experiences and Comparisons

Today's Web Application development process runs over open-source libraries. Especially in the React Ecosystem, there are thousands of different libraries. It is particularly important to use these libraries correctly when required and to make the right selections. The libraries to be used may differ from project to project according to the requirements.

For example, there is more than one library in the State management for React: Redux, Apollo GraphQL, React-Query, etc. Here, you need to know the scope of the State Management concept to understand which one of these libraries should be selected. LearnReactUI.dev ensures comprehension of the topic by describing the concepts in question with examples based on one of these libraries.

Every passing day new technology is developed. The important thing here is your comprehension of the concepts. For example, if you better know the **below-mentioned concepts** during Application and Server State management:

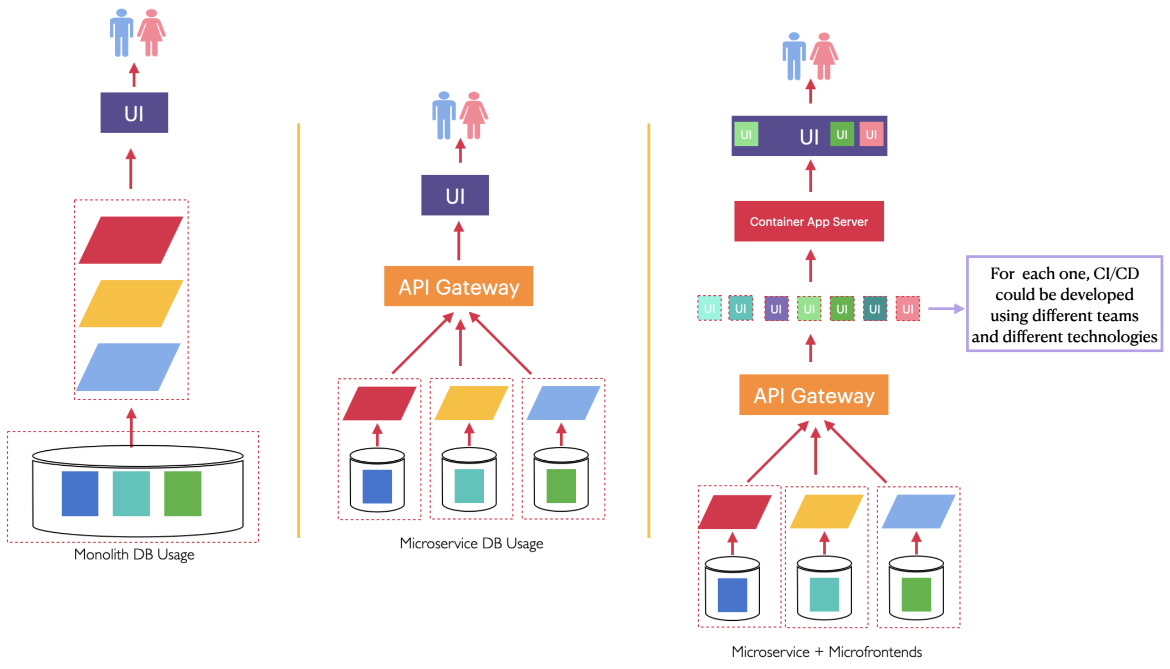
- Async Status
- Caching
- Callback Handling

- Mutation & Optimistic Update
- Background Updates
- Pagination /Incremental Loading
- Outdated Requests
- Deduping Request
- Garbage Collection ve Memory Implications

you can adapt to these new technologies in this respect faster and more detailed way.

1.5 Courses in LearnReactUI.dev Are Prepared Similarly to the Microservice Logic

Microservice has become popular initially on the Server side and then on the Frontend side. And what is Microservice? In the Microservice logic, the applications are designed as small applications instead of monolithic units. Therefore, these parts will not affect the application generally and will be scalable.


UI
Monolith DB Usage
UI
API Gateway
Microservice DB Usage
UI
UI
UI
UI
Container App Server
UI
UI
UI
UI
UI
UI
UI
For each one, CI/CD could be developed using different teams and different technologies
API Gateway
Microservice + Microfrontends

LearnReactUI.dev courses will similarly be in the form of small application parts. You can select the courses in the form of smaller parts like these application parts, and you can create the course close to your applications using the interrelated parts. Thus, you can adapt the available examples of learning to your projects quickly.

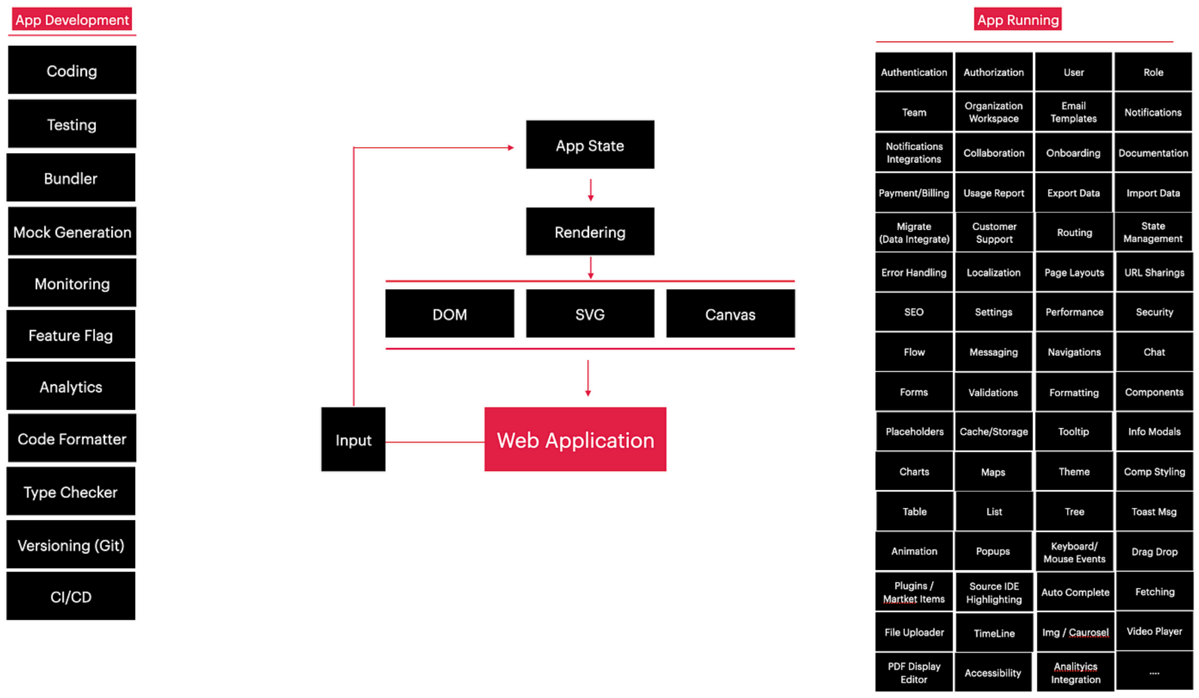
This situation also allows you to use only the examples connected to your goal by making smaller payments instead of making payments in blocks for the courses you attend.

2. TECHNICAL DETAILS

A detailed explanation of the aforementioned subjects is provided below. If you have sufficient technical information on these subjects, you may prefer not to read the following part of the document.

2.1 Web Application Structure, Processes, Concepts, and Tools

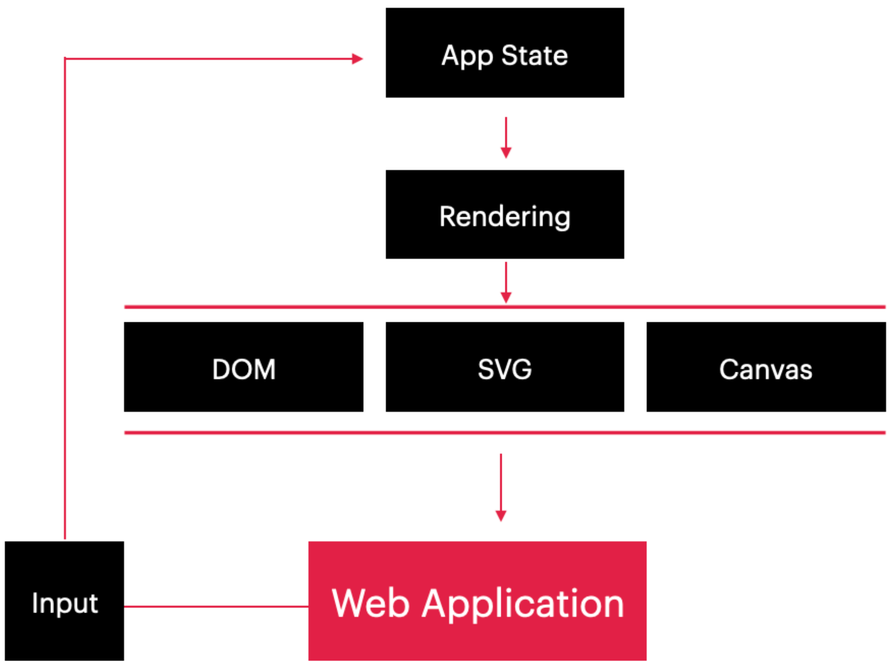
Technical details and descriptions of the Web Application structure, processes, concepts, and tools are provided below.



2.1.1 Web Application Mechanics

First of all, let's explain the **Web Application section in the middle of the figure** above.

The Web application is the rendering of the AppState on the (DOM, Canvas, SVG) structures via the WebAPI located on the Browser, the receipt of the coming input such as user/network, and re-update and display of the AppState.



2.1.2 App Development

Activities carried out by the Frontend developers during the Web App Development phase:

Coding: There are many topics related to software architecture in the software development phase. These concepts allow you to write more clear, more high-quality, and reusable codes.

- Architectural Concepts
- Architectural Experience

- Architectural Quality
- Architectural Patterns
- Design Patterns
- Anti-Patterns

Testing: You need to ensure the accuracy of the codes you write by testing the Code, Module, Library, API, or Application at different levels.

Bundling: It is your expertise on the tools that make the codes we write (JS and varieties) and the styling (CSS and varieties) Production Ready. Examples include the Webpack, Rollup, ESBuild, and Vite tools.

Mock Generation: The codes you write for the Frontend sometimes require the Backend data while they sometimes require the user, etc. data. It includes using tools that allow you to generate these data for your testing environment. For example:

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

Monitoring: They are the library and SaaS services you can use to diagnose any error and problem in the frontend monitoring area. For example:

- Log Rocket
- Sentry

Feature Flag: In the development phase, you may need an infrastructure enabling/disabling some of the functions in your application. For example, for A/B testing:

- LaunchDarkly

Analytics: It provides several analytical tools that allow you to analyze the demographic structure and behaviors of the customer and some information on how you should develop the product. For example:

- Google Analytics
- Heap
- FullStory

Code Formatter: We use several plugins and tools for formatting the code and displaying the same in a well-readable way. For example:

- Prettier,
- Es-Lint

Language Support: Type Checker, Annotations, etc.

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

Git Usage: You need to know the operation of the Git structure and the use of the branch structure for the version method.

CI/CD: GitHub Actions are recently so popular in relation to Continuous Integration and Development. In addition to this, you can also use Jenkins and Bamboo.

2.1.3 Application Concepts

Application Concepts include many different subjects. Describing them in a single article will make the article very complicated. Therefore, only the names of these concepts are given in this document. These will be elaborated on in more detail in the oncoming articles.

- 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 Sharing
- 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)

In addition to the abovementioned concepts, you can see that the platforms such as TailwindUI, PrimeBlock, and TailwindUIKit give the UI blocks of the platforms in the form of ready-made templates in terms of styling.

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

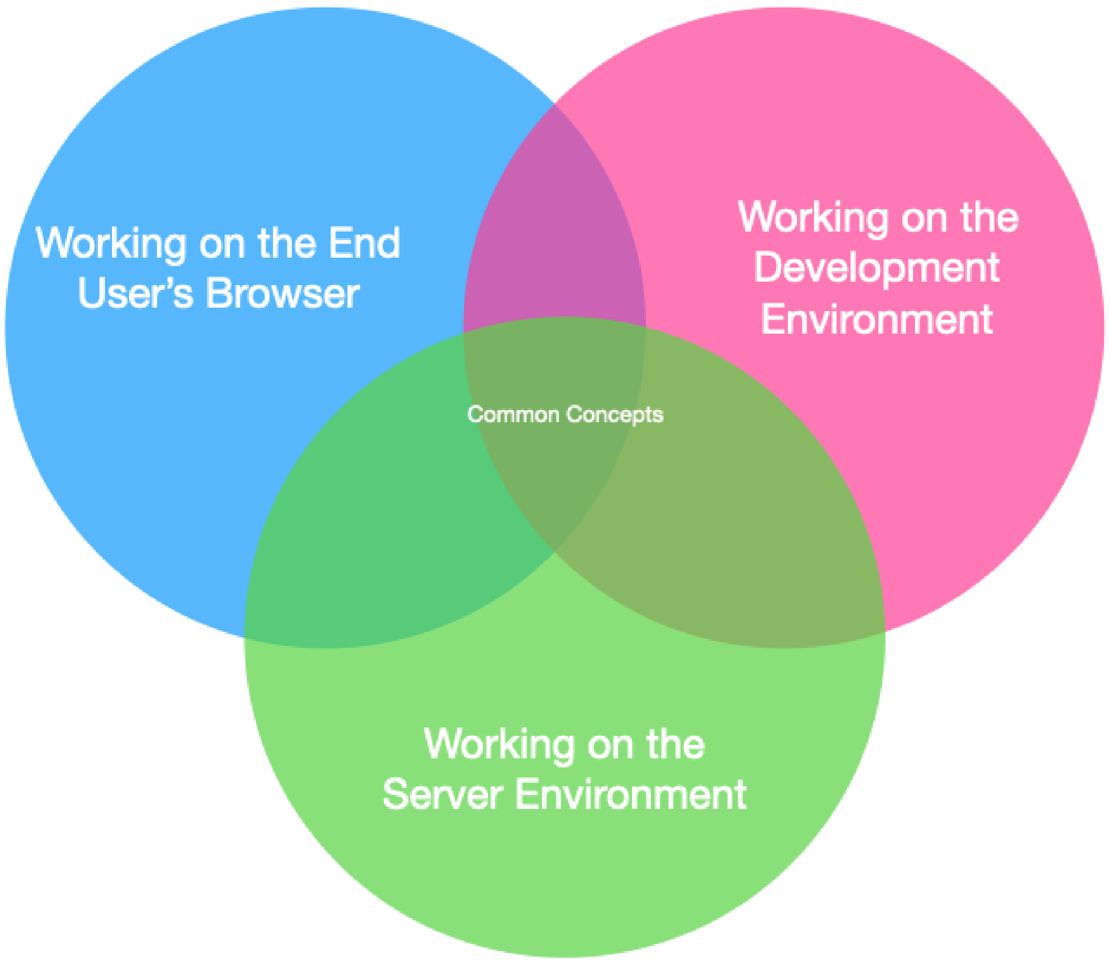
- **Application Shells:** Stacked, Sidebar, Multi-Columns
- **Headings:** Page, Card, Section
- **Data Display:** Description List, Stats, Calendars
- **Lists:** Tables, Stacked List, Greed List, Feeds
- **Form:** Form Layouts, Input Groups, Select Menus, Sign-in, and Registration, Text Areas, Radio Groups, Checkboxes, Toggles, Action Panels, Combo boxes
- **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, Button Groups
- **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 Looking at Web Application and Development Environments from Different Perspectives


Working on the End User's Browser
Working on the Development Environment
Common Concepts
Working on the Server Environment

2.2.1 Browser Environment

In this environment, whether on the user’s computer, mobile phone, or iPad, the users can access these websites and web applications via a browser.

2.2.1.1 Browsers

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

  These browsers process the data from the server and consequently develop websites and interactive applications. And how do these browsers run? Shortly, they create the websites by processing **HTML, CSS, and JS**. 2 types of engines are used for this purpose.

2.2.1.2 Javascript Engine

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

  Javascript Engine is the engine processing the JS files. It processes your UI business codes or manages the systems such as Network, DOM, Animation, Bluetooth, etc. by calling the operating system and other level API with WebAPI.

2.2.1.3 Javascript (EcmaScript)

- **JS History booklet** will describe in general the topics of let/const, Arrow Functions, Inheritance, Default Parameters, Rest/Spread, Destruction Assignments, Template Literals, Loops, Async, Promise, EventLoop, Modules, Map, Set, Generators, etc. and the language development of ES6 EcmaScript will be explained.
- In the **booklet on Functional Programming with JS,** information on developing higher-quality software based on errors and vulnerability will be provided.

2.2.1.4 Browser Engine

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

It is the engine that creates a Layout by combining DOM and CSS and renders the HTML elements for this purpose. **Note:** Today, DOM and CSS can also be used after being created by JS. (React, Vue, Angular, StyleCSS, etc.)

2.2.1.5 CSS

Cascading Stylesheets: It is the mechanism that allows the styling of DOM Elements. In this way, the pages can be created in a better way and according to an alignment. The following options are available in CSS standards.

- **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

In fact, in these CSS definitions, the developers are provided with the opportunity to write these CSS data in a structured and reusable way by using a **Preprocessor** (SASS, LESS, etc.).

In addition, creating CSS with JS is so popular nowadays and this is called CSS-in-JS. There are several libraries allowing the use of these structures.

- **CSS-in-JS Libraries:**

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

  And are there any ready-made Frameworks created for these CSS and Theme-applied UI component libraries? Yes.

- **CSS Frameworks:**

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

2.2.1.6 Types of Application / Web Sites:

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

The applications can be grouped in the following way:

- **SPA:** They are the Web Application Services that operate using the power of the Browser in the User's Device; in other words, they include all the screens and components of the (SPA) application in a single page as far as possible. These are highly interactive applications where the JS files are processed by the browser On The Fly as far as possible, and JSON, etc., content is loaded to the site from RESTfull services by making AJAX/HTTP Request, and then the pages are rendered.

- **SSR and SSG:** Secondary type applications are pages where we share information, each URL such as CMS, Blog, Portal, etc. provides unique information, and the data in the pages can be indexed by a Search Engine Crawler, technically created at the HTML and CSS server side to the maximum extent and which exhaust the browsers at the minimum level in terms of processing JS and where user interaction is not at a high level. These types of CMS pages were generally developed over WordPress, Joomla, Drupal, phpBB, and LAMP Stack (Linux, Apache, MySQL, PHP/Perl/Python) at the server. However, today SR JAMStack (JavaScript, APIs, and Markup) has taken its place.

- **PWA:** Progressive Web App concept was first brought up by Google. We can create a hybrid application model between Standard Web Pages and Native Mobile Applications. This model, even operating like a web page, uses ServiceWorker, LocalStorage, and other WebAPI Web to provide Native application features also when the Internet is not available.

2.2.1.7 WebAPI/BrowserAPI

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

  Today, our expectation from web pages is higher since mobile devices have increased and the web environment has replaced desktop applications. This has caused the development of Browser APIs that will meet the expectation concerned.

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

  It ensures the rendering of UI components (Tables, Buttons, Selection Boxes, Form, Navigation Menus), texts, paragraphs, and most of the elements and components displayed by changing the HTML Elements and the Node on the current document.

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

  It is the Scalable Vector Graphics, the creation of the shapes in a vectorial way, in other words, the creation of Drawings Libraries, Chart Libraries without being affected by the display solution, and zoom-in/out concepts.

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

  It allows painting pixel by pixel on a Canvas base. Thus, painting actions requiring higher performance can be made thanks to this API. Like Google Maps, it enables the drawing of Map layers thanks to API and accessing the maps from any device.

- **History API**

  It allows you to back or forward by keeping Browser Session History.

- **URL API**

  It is the API that is defined as the Uniform Resource Locator and which allows access to and update the address domains, hosts, and IP.

- **Console API**

  It is the API enabling access to the console at Chrome, Safari, and Firefox developer tools. (console.log, console.debug etc.)

- **Drag-Drop API**

  It provides an API that allows drag-drop of the web elements by mouse. (Note: For details, you can read my article on Drag-Drop API.)

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

  It provides APIs that allow better use of the Graphic Card, the Sound Card, and the Microphone hardware. It enables 3-dimensional Rendering on Canvas Context, access to sound channels via Audio API, sending stream data, applying effects on them, and rendering the same. Speech API is the API allowing voice recognition over the microphone and conversion of these sounds into text.

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

  It allows higher utilization of the device hardware by accessing the hardware and sensor information of the web applications.

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

  It ensures Real-Time Communication between web pages and enables peer-to-peer communication and data sharing between Video/Audio streams. Fetch API for WebRTC, HTTP Request/Response mechanism infrastructure, WebSocket API offering communication between the client and the server ensuring duplex communication, ServiceWorker, and Web Notifications API for local notifications and Push API for the notifications to be received from the server, can be used.

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

It provides APIs allowing access to the data kept as key value in the browser and the data kept in the database or APIs which can access the file system.

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

  These APIs ensure operating and managing Workers capable of performing the side works except the Browser UI Look.

- **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 etc. UI Libraries have enabled the creation of reusable components and the use of these components by each other and have consequently allowed the establishment of larger component libraries. Ultimately, the purpose of the Web Component developers is to support this at HTML standards without component libraries. And it may gain popularity as a result of the adaptation of these structures by the developers. Recently the wind blew toward the React, Vue, and Angular libraries.

2.2.1.9 WASM

- Browsers normally are structures established based on running a JS Interpreter. They operate the application by accessing C Binding over the EventLoop and WebAPI. If we want to operate low-level assembly like languages such as compiled C/C++ or Rust, it will eliminate many obstacles related to the games and high-performance applications. The technology allowing the browsers to run the compiled WASM files is called WebAssembly. Thanks to this, JS and WASM files can be operated together.

2.2.1.10 Store

- Redux, MobX, RxJS, NgRx, VueRx

  Many Store examples are the realization of the Flux Pattern for management due to the Modern Web App's state storing and interaction with the server.

2.2.1.11 Use of These Technologies Except Browsers

Browsers are mostly specified in terms of the end user above, but apart from the browsers, technological infrastructures which can use the abovementioned technologies have been developed in the Native environments as well. In this way, it has become possible to develop cross-platform native applications.

The environments providing infrastructure **using the WebView component** are the following:

- PhoneGap Cordova (**Shut down to development)**, NW.js, Electron

The technological approaches serving the purpose of using the JS and CSS standards without the **WebView component** while the Native codes are running in the background are the following:

- Flutter, React Native, NativeScript

Headless Browser*s:* These are the browsers provided for making the test automation of the Web pages over the Console/Network without the browser interface.

2.2.2 Developer Environment

- You can see that the number of libraries or cloud services we use in the developer environment has increased. You need to have good knowledge of the features of these tools and their use.

2.2.2.1 Version 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 Language Support: Type Checker, Annotations, etc.

- 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 Server Environment and Hosting

The server environment provides CDN where the HTML, CSS, JS, and other binary/base64 files are delivered, as well as **JSON** over the REST API. These servers provide these services to you through different cloud provider services. Access to the Frontend to the server is also ensured by the DNS.

2.2.3.1 Networking / DNS Settings

- DNS Settings
- HTTP I/O RPC, REST, GraphQL, Webhook
- How to use the REST? (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 Cloud / Hosting Services

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

2.3 Some of the Concepts Specified on Web Application and Server State Management

2.3.1 Concepts

2.3.1.1 Async Status

Some states of the data in the server may occur during the transfer of the Client, and the related UI component is required to change its behavior according to these states.

- success: The response reaches the client successfully following the request.
- error: The request is unsuccessful, and an error occurs.
- loading: The request is still in progress on the network and in the pending state.
- idle: The request is not delivered yet while the relevant components are visible on the screen.

2.3.1.2 Caching

Thanks to the cached data, we do not need to make network requests repeatedly.

- In this way, the loading state is shown less on the screen, and the screen flows can be high quality and smooth.
- It ensures less use of the sources and money saving since there will be less network traffic, and the backend will cause less exhaustion.

The difficulty here is that some extra logical codes must be created to understand the invalidity of the cached data and not to show false data to the user. This makes the ClientSide work logic more complicated.

2.3.1.3 Callback Handling

You may need to do extra actions following the Call process you make in this section. We can handle the onSuccess or onError states separately and request the launching of another flow.

For example, **onSuccess** redirecting to another page, cache invalidation, etc. You can ensure this structure by writing around the Redux dispatch code you write its **CallbackHandler**.

2.3.1.4 Mutation & Optimistic Update

It is the process of making several alterations on the whole or a part of the data in the server and reflecting the relevant results to the UI. However, the situations such as a change in the object or a change in a structure with array elements, etc., lead to very different UI requirements.

- Is a new element added?
- Is an element deleted?

The reflection of all these should be evaluated elaborately. The **Optimistic Updates** method can be used for the updates in some sections.

Optimistic Update takes action in UI as if it is successful and makes a rollback action in case of an error. Here, a feeling like very fast processes are being made is created without showing the Loading State to the user.

2.3.1.5 Background Updates

These are the mechanisms that you should constantly check in the background and where you ask the server questions about whether the system received any notification or not or whether there is any update in some of your data groups or not.

In this type of structure:

- Fetching at certain intervals, which is called **Long Polling,** can be realized.
- **Server-Sent Events (SSE)** transfer the unidirectional data to the clients through HTTP Connection by using push technology.
- The Server delivers the information desired to the client in a duplex communication established over the **WebSocket**.

2.3.1.6 Pagination /Incremental Loading

Answers to the following questions are found:

How the page information kept during the pagination processes will be managed?

How the async loading of these pages will be realized?

How the interaction of the **Infinite Page Scrolling** with UI will be?

2.3.1.7 Outdated Requests

It means the expired requests. This can be based on many different reasons. For example, a similar request may have been made after this request or the source may have been deleted by others before the source requested to be processed.

2.3.1.8 Deduping Request

The web applications run in line with the events that occur following the user actions or in line with the generation of other events by the events from external sources.

For example, you send a network request each time the user presses the key or the keyboard keys and no debouncing mechanism is available between them. In summary, if you have a system that makes requests successively before the answer to the request comes from the server, this will cause some problems at the phase of showing the correct data. You need to understand which is the valid response and to eliminate others.

2.3.1.9 Garbage Collection ve Memory Implications

It is related to the deletion of the garbage data from the memory and the effect of all the processes above on the memory.

2.4 Open Source Projects That Can Be Used When Developing a Web Application

The libraries which are popular on GitHub are grouped under specific titles. In this grouping, initially the JS and TS libraries over 30K have been filtered.

The query lists the repositories over 30K:

JavaScript Repositories

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

TypeScript Repositories

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

Below there is a list that was created in 2021 and updated according to the year of 2022. In time, there may be minor changes and differences in these lists but it is not very likely to have an effect in general.

2.4.1 Repositories Including Example, List, Best Practice Methods

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 more
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 Libraries Allowing You to Create and Render UI Components, Charts, Visualization or 3-Dimensional Components

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 vizualization library for browser.
11. svelte **(TS):** Cybernetically enhanced web apps.
12. nestjs/nest: Nest is a framework for building efficient, scalable Node.js server-side applications.
13. ionic-framework **(TS):** A powerful cross-platform UI toolkit for building native-quality iOS, Android, and Progressive Web Apps with HTML, and CSS.

2.4.3 Libraries Providing Ready CSS Layout or CSS Components

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 apps.
4. Font-Awesome: The iconic SVG, font, and CSS toolkit.
5. Semantic-UI: Semantic is a UI component framework based on useful principles from natural language.

6. html5-boilerplate: A professional front-end 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 Tool libraries

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 Libraries Allowing High Level Code Writing

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 Tools Providing Some Presentation and CMS Functions for Content Creation

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 emails in Markdown and render them before sending them. It also supports syntax highlighting (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 is inspired.
5. Ghost: The #1 headless Node.js CMS for professional publishing.

2.4.7 Platforms Providing Fast Application Development Framework Without the Details of React, Vue, JS Libraries

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 Platforms Providing Code Development Environments or Plugins Operating in This Environment

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 Repositories Providing Tools, Servers, and Test Data for Creating Fake Testing Environments

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 JS Repositories Including Runtime

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.

Conclusion

You have reached the end of this article. We hope you will benefit from the booklet. Once similar paid and free learning content is developed, it will be sent to you via e-mail. You can also take a look at the other learning content on the website.