***What is React, and why do we need React?***

***What are the challenges in Modern Web Development?***

A)Modern Web Users are using the web from smart Devices more than 80 %

Mobiles, Tabs, etc.

**Challenges :**

1. **UX**
2. **Fluid User Experience**
3. **Loosely coupled and extensible.**

**1. We need a Unified User Experience**.

Unified Experience means an application must have the same experience, behaviour across any device.

**Example:**

We are watching YouTube on both PC/Laptop, it is working perfectly.

On Mobile through a browser (Mobile browsers are compact).

Earlier, we had to optimize(Compress) the YouTube site to fit on Mobile.

And Mobile Users are unable to use certain facilities. Some are working and some are not working.

Ex: MS Office on Pc/Laptop and Mobile are different.

**2. Fluid User Experience**

**Smooth Navigation** – Moving between pages or screens feels natural and fast.

**Quick Response** – Buttons, forms, and actions respond instantly with clear feedback.

**Easy to Use** – The design is simple and users understand what to do without confusion.

**Looks Good Everywhere** – Works well and looks consistent on all devices (mobile, tablet, desktop)

**3. Loosely coupled and extensible.**

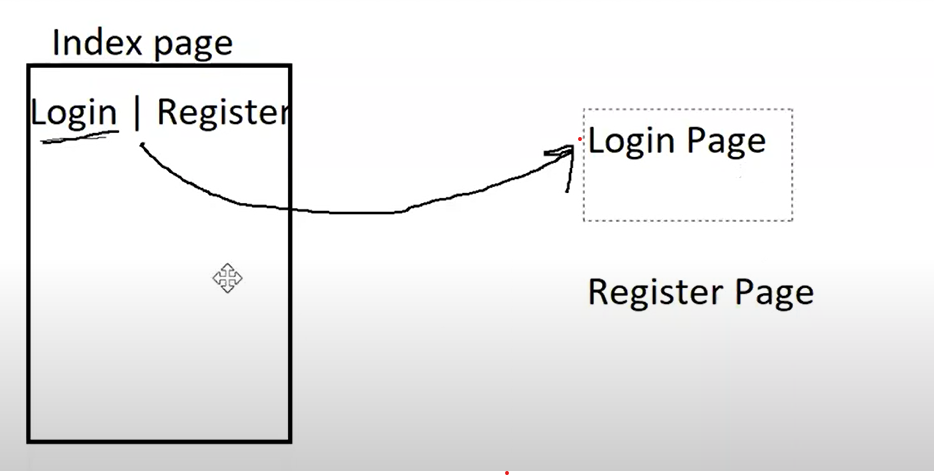
Without reinstalling the application, without stopping the application, you just keep the application idle for a few seconds.And add the new features to it.

This type of architecture is called loosely coupled and extensible.

***What is the Solution?***

Better to build an SPA (Single Page Application).

Example:

1)It is the normal way. All the pages are designed in differently and integrate into the index.html, and it will navigate to the particular page.

2)In this, we use the component in the SPA.

Like the login component, the Register Component will be integrated into the index.html.

***How to build an SPA?***

A)Can we build using JavaScript and jQuery? Yes

***What are the issues with JavaScript and jQuery?***

1. A lot of DOM Manipulations
2. A lot of references
3. A lot of Coding
4. A Lot of event handling
5. Explicit Ajax.

***What is the solution?***

A)Better use a library or a framework

React, Angular, Vue, Ember, Backbone, Knockout, etc.

* [React = Facebook]
* [Angular = Google]

***What is the difference between Angular & React?***

A) React is a library. Not a Framework.

Angular is not a Framework. it is a developer platform.

A developer platform means it will provide an end-to-end solution

For the developer.

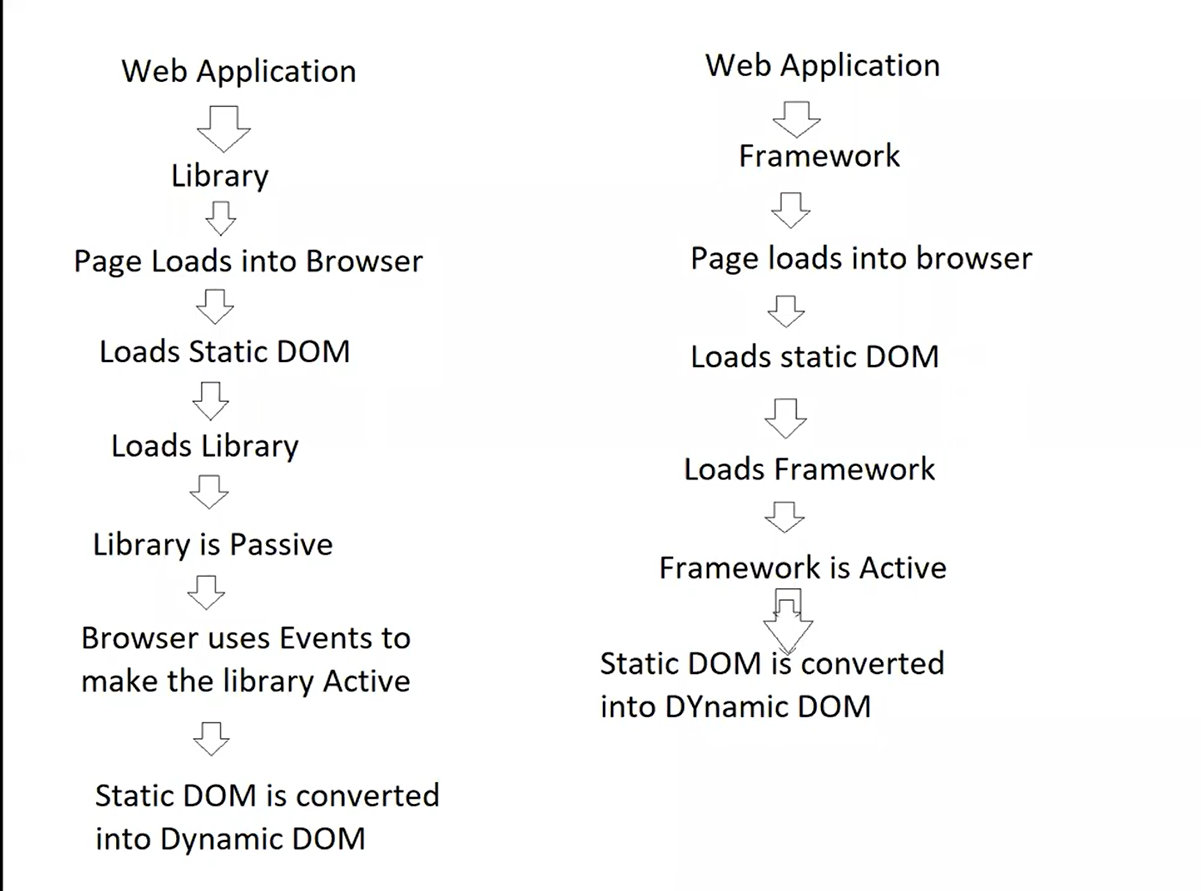
Actually, a developer needs some tools for building, debugging, testing, and deploying.

If any technology provides an end-to-end solution from building to deploying everything, it is called a Developer platform.

***What is the difference between a library and a framework?***

A) The library is building the application.

The framework can be used to build and controlling the application's flow.



**For the Example of library:**



**For the Example of Framework:**



***What is React?***

1. React is a JavaScript library for building UI.
2. UI = SPA
3. Fluid UX
4. Unified
5. Loose-Coupled and Extensible.

***What are the Features of React?***

1. Faster
2. Better Performance
3. 1) Component-Based
4. Easy to build interfaces
5. Easy to reuse
6. Building faster
7. Easy to extend
8. Loosely coupled

2) Modular Library

* 1. Application-Specific Library
  2. Improves the performance of the application
  3. Lightweight application

3)Asynchronous library

1. Implicitly uses Ajax
2. Without reloading the complete page, new details are added to the page.

4) Uses Virtual DOM

1)What is DOM?

2)How does DOM render?

3)What is Shadow DOM?

4)What is Virtual DOM?

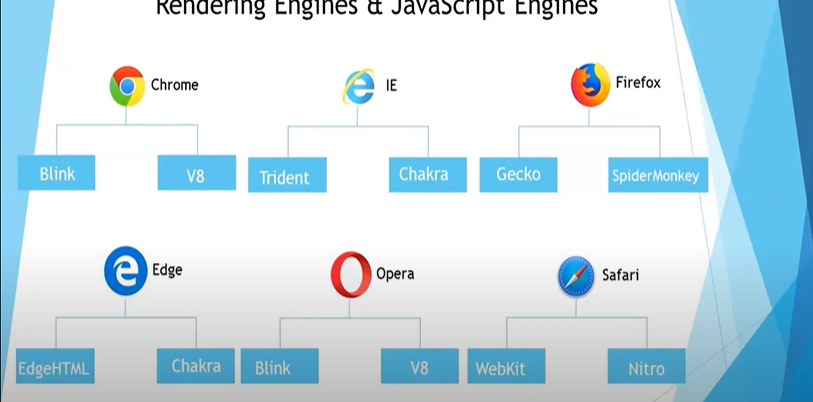
5) Difference between DOM, Shadow DOM, Virtual DOM?

*1)What is DOM?*

A)It is a hierarchy of elements presented in the Browser.

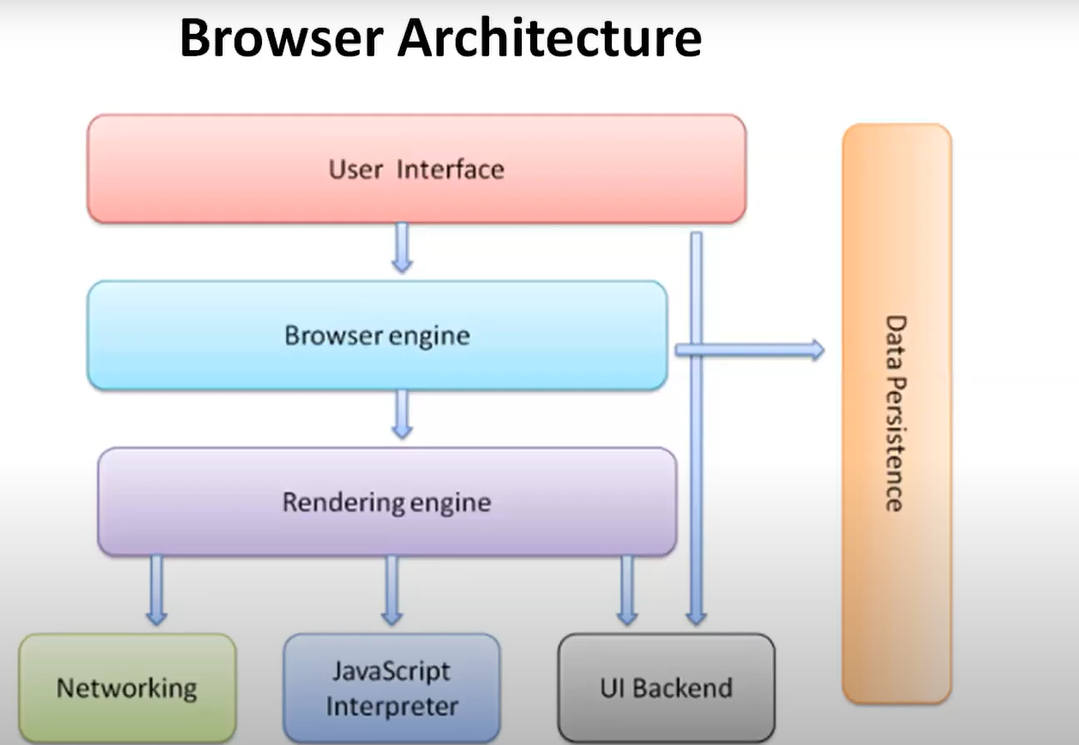
**Browser Engine:**

1. Chrome: Blink, V8
2. IE : Trident, chakra
3. Safari: WebKit, Nitro
4. Firefox: Gecko, Spider Monkey
5. Edge: Chromium, Chakra, EdgeHTML



**Architecture of Browser Engine:**

1. UI
2. Browser Engine
3. Render Engine
4. Data Persistence
5. Networking
6. JavaScript Interpreter
7. UI Backend



**Workflow of Browser Engine:**

Order :

1) The first HTML page converts to the Binary Format.

2) E.g., bytes 101010.

3)Bytes are again converted into Characters

4)Chars : < P > < b o d y >

5)Again, chars are converted into Tokens.

And this process is called Tokenization.

6)And again, Tokens are converted into Nodes.

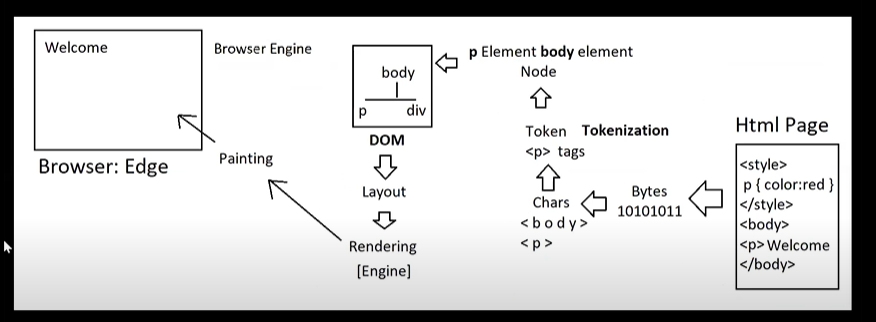
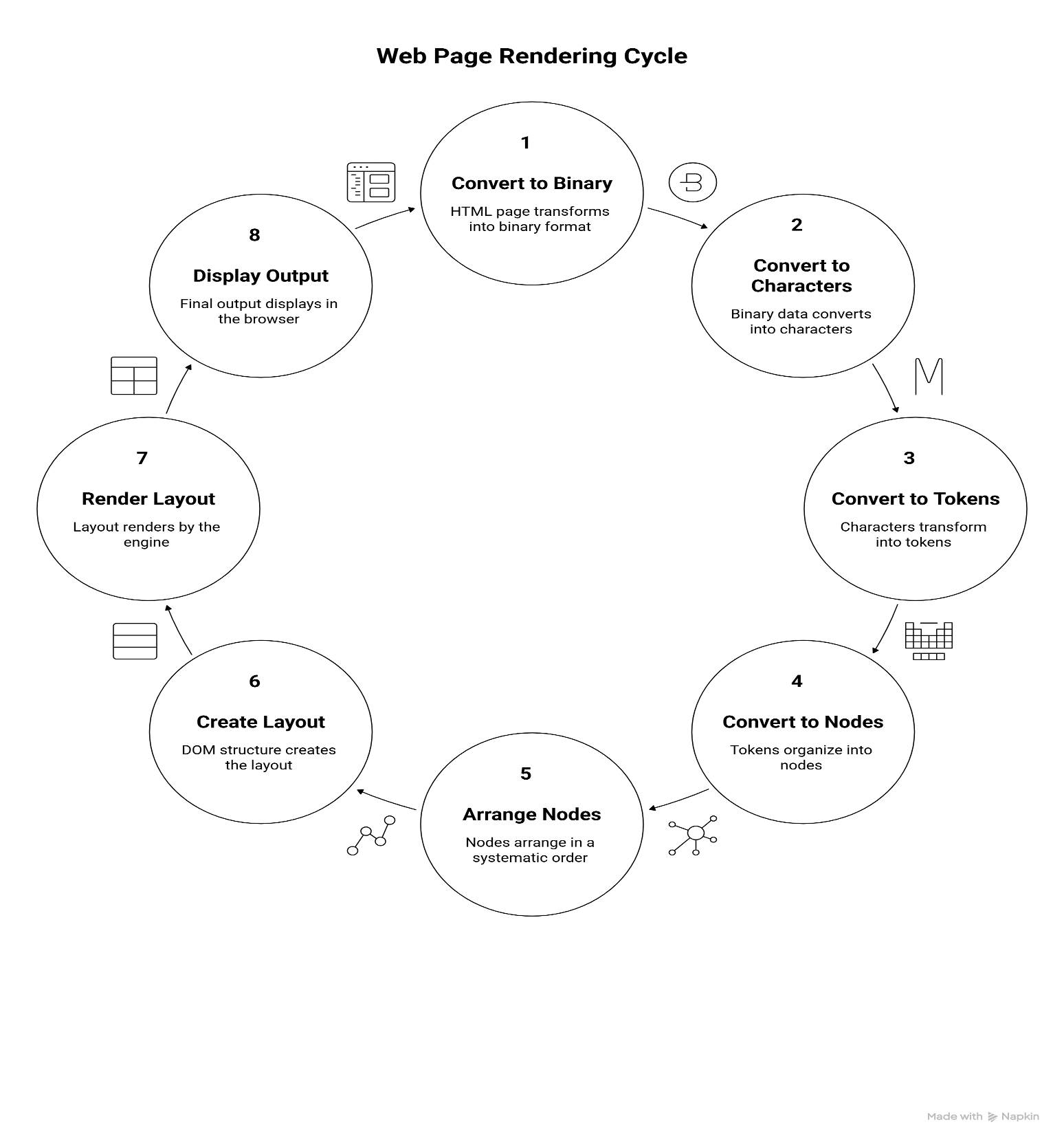
7)Nodes are arranged the elements in a systematic order.

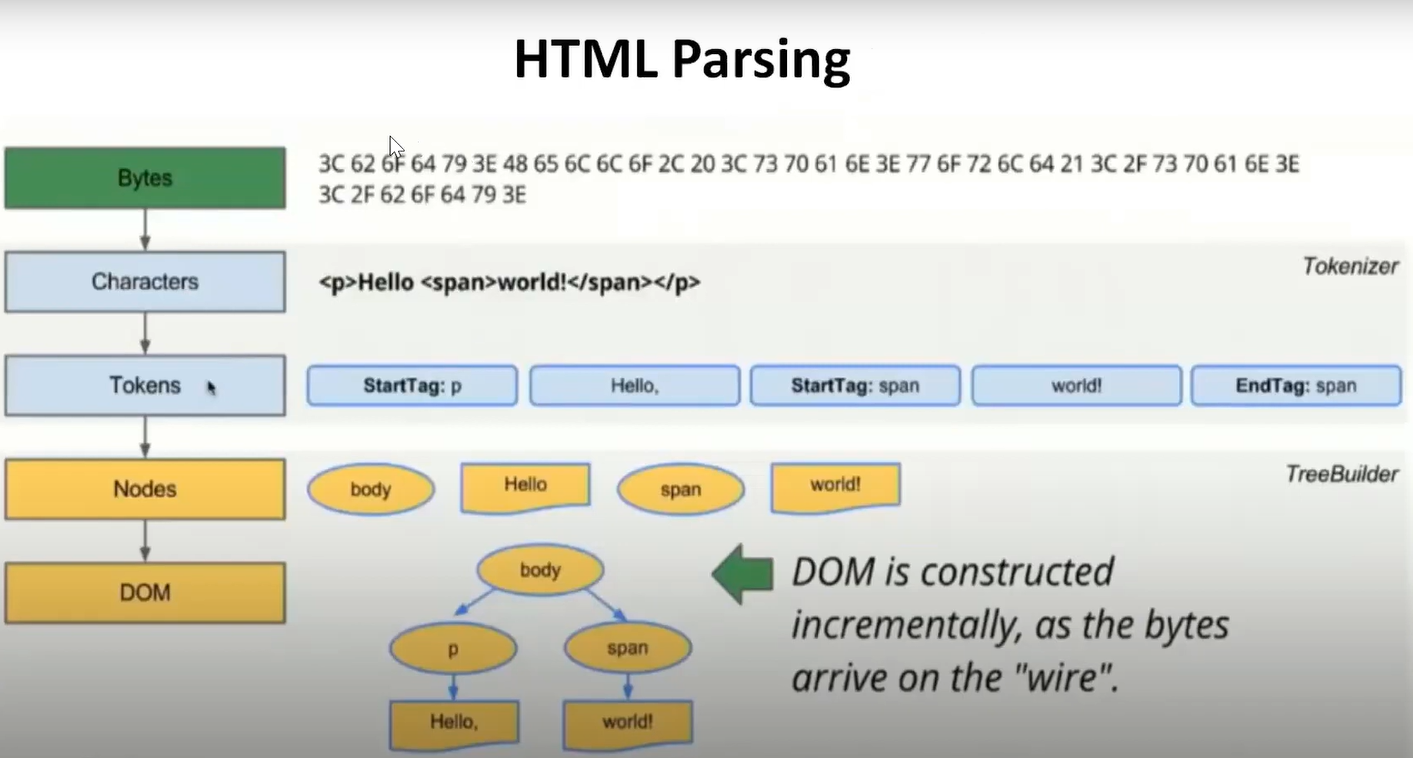
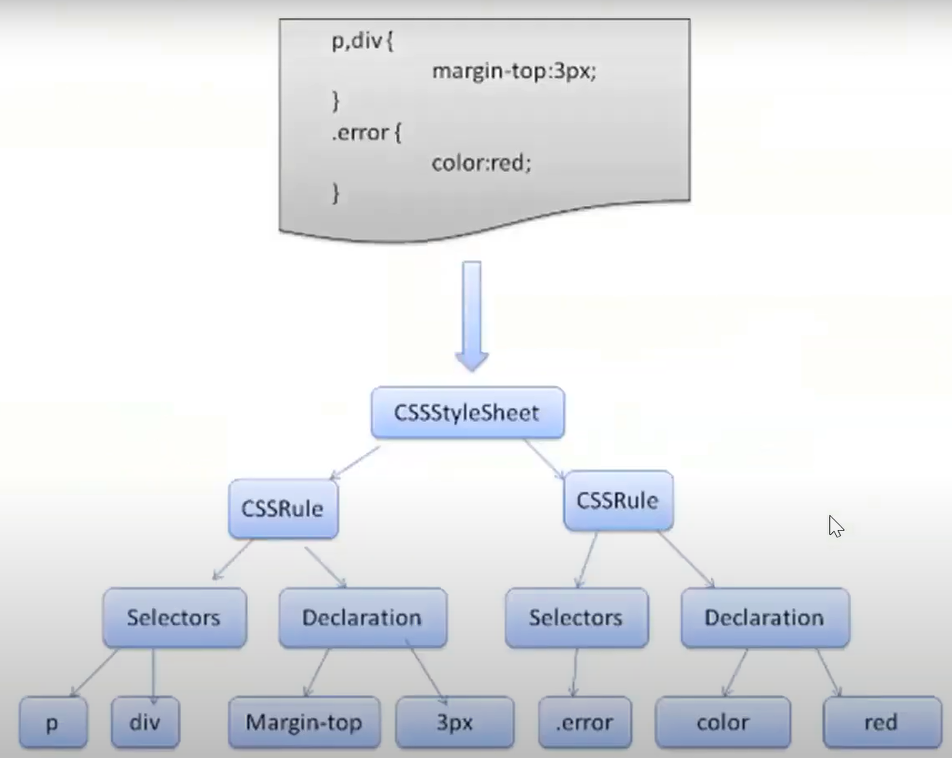
8)Like a DOM Structure

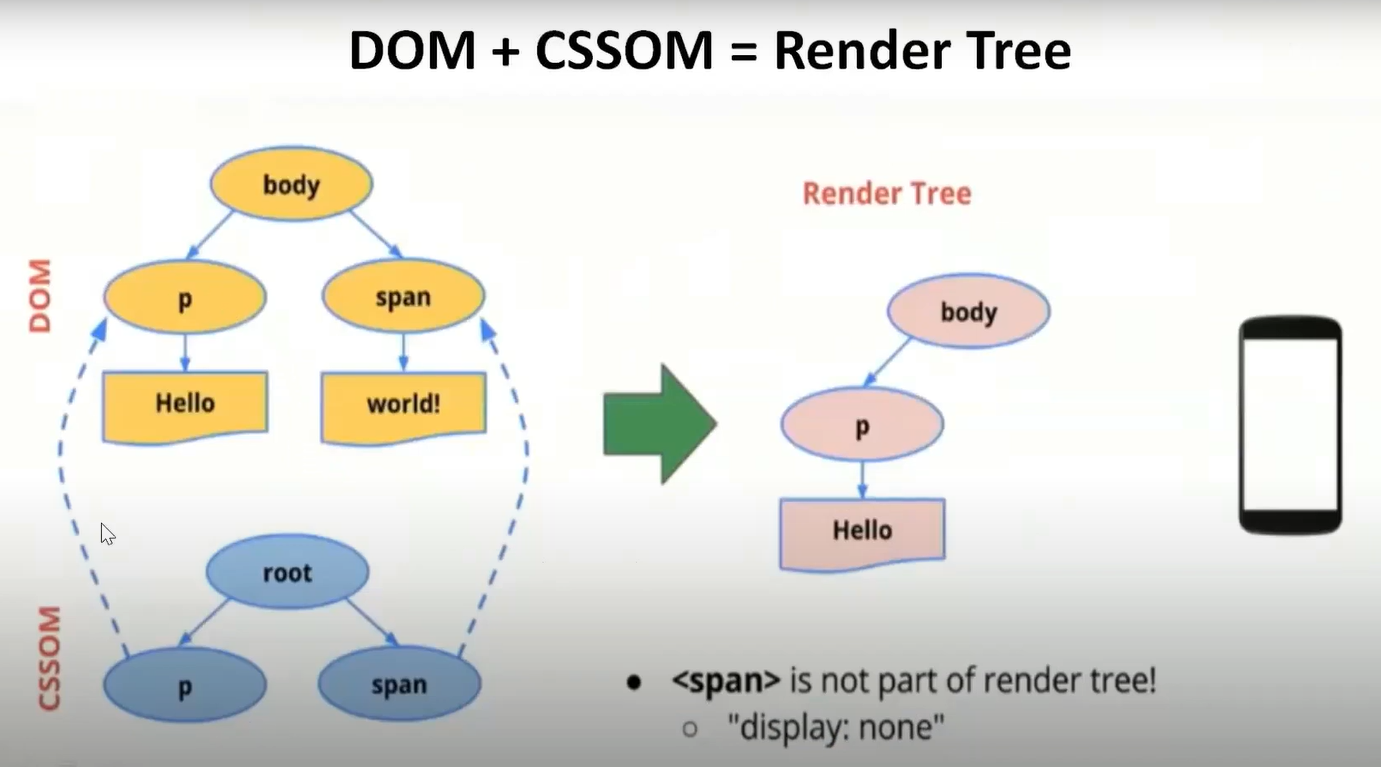
9)Dom gives you the Layout

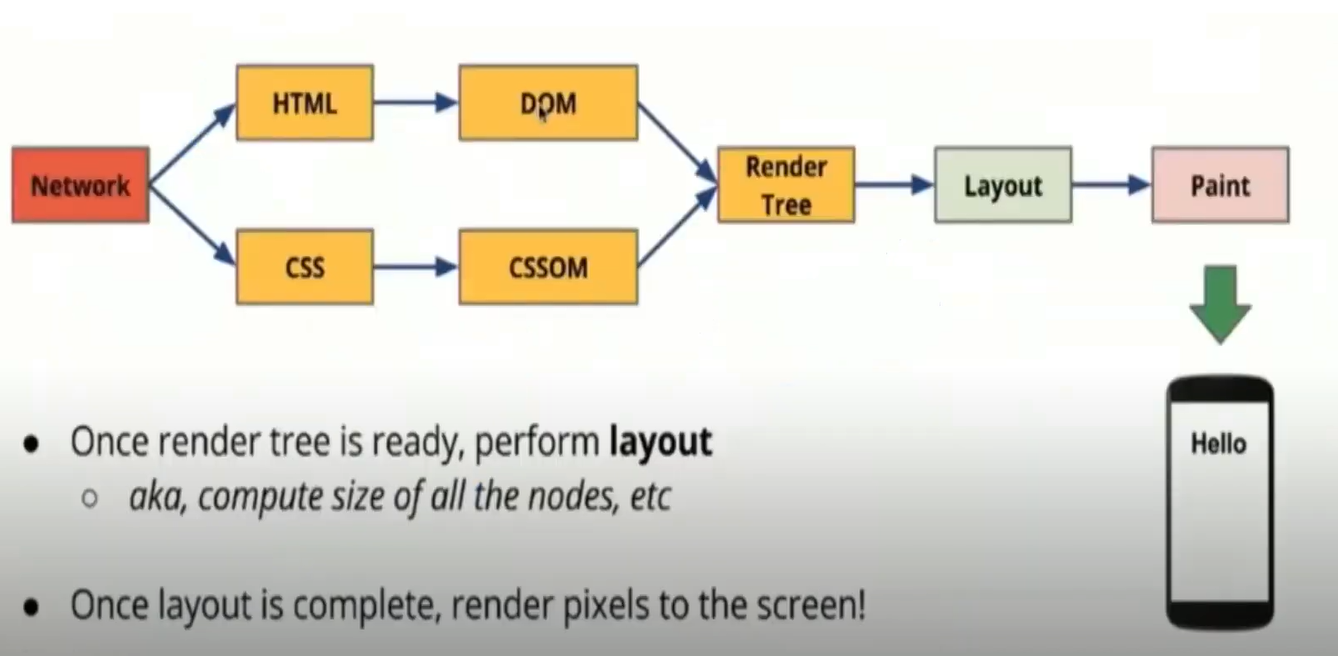
10)The layout will be passed to the Rendering engine.

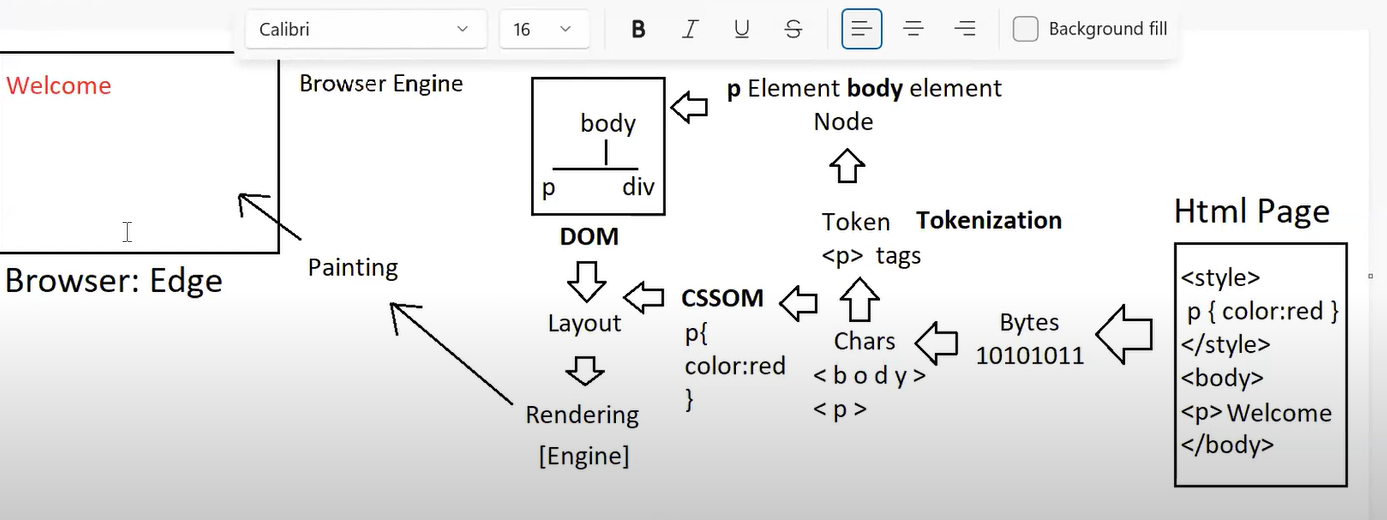
11)The rendering engine will undergo the painting

12)And Painting is responsible for displaying the final output in the Browser.





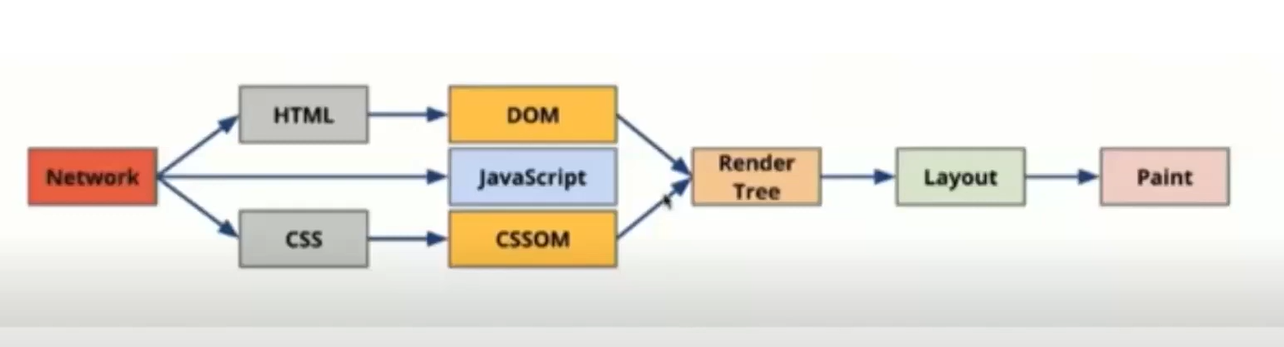




*Two object models are working in the background*

**DOM: Document Object Model of HTML**

**CSSOM: Cascading Style Sheet Object Model of CSS.**

****

**Workflow of Browser Engine:**

Markup 🡺 Bytes 🡺 Chars 🡺 Token 🡺 Node 🡺 DOM 🡺 Layout 🡺

Rendering 🡺 Painting 🡺 Output in Browser.

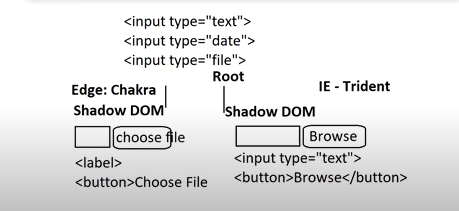
Styles 🡺 Bytes 🡺 Chars 🡺 CSSOM 🡺 Layout 🡺Rendering 🡺 Painting 🡺 Output in Browser.

***2)What is Shadow DOM?***

A)Shadow DOM is a DOM under any specific element in a page.

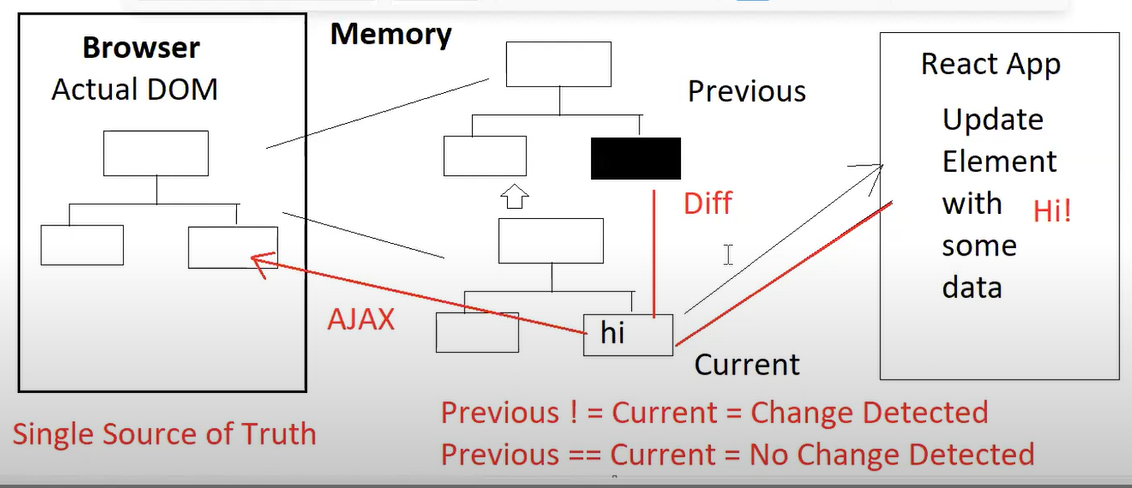
Root and Shadow hierarchy of building components.

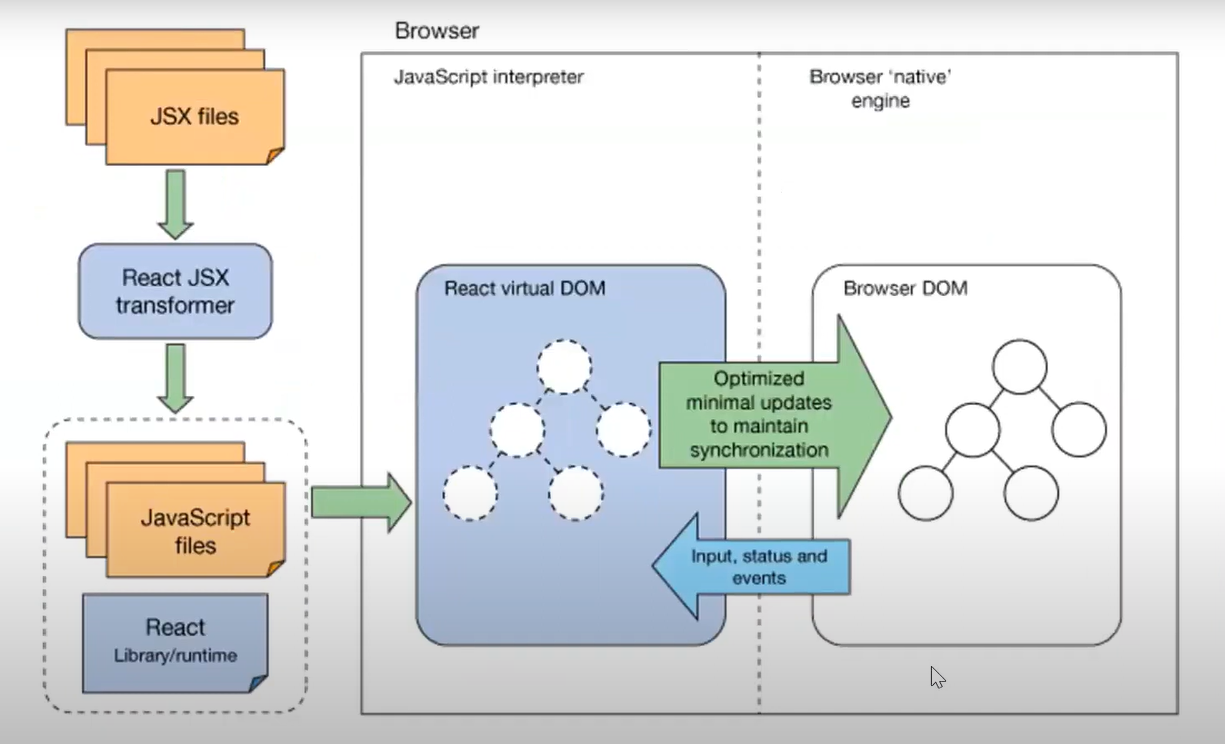
**Note**: **DOM** is for the page

 **Shadow DOM** is for a component in a page [Date, file, color].

***3)What is Virtual DOM?***

A)





***4) Difference between DOM, Shadow DOM, Virtual DOM?***

Below is a simple explanation of DOM, Shadow DOM, and Virtual DOM with diagrams and a comparison.

## 1. Simple Definitions

• **DOM**: The actual structure of the webpage loaded in the browser.

• **Shadow DOM**: A smaller, isolated DOM attached to elements for component-based development.

• **Virtual DOM:** A JavaScript-based representation of the DOM, used by libraries like React to update UI efficiently.

## 2. Comparison Table with Diagrams

|  |  |  |
| --- | --- | --- |
| **DOM** | **Shadow DOM** | **Virtual DOM** |
| Represents full document structure.  **Diagram:** <html>  <body>  <div>  <p>Hello</p>  </div>  </body> </html> | Scoped within an element.  **Diagram:** <custom-element>  #shadow-root  <style>p {color:red}</style>  <p>Hello</p> </custom-element> | JS copy of real DOM.  Diagram: virtualDOM = {  tag: 'div',  children: [  { tag: 'p', text: 'Hello' }  ] } |

**Cons of React :**

React is a library [can build but can’t control application flow].

Need support for various frameworks explicitly.

Need Support for various 3rd-party libraries.

Forms and Validation: Formik, Yup

API : Fetch, Axios, JQuery, Ajax

# React 3rd-Party Libraries

This document lists commonly used 3rd-party libraries in React, categorized by their purpose.

|  |  |  |
| --- | --- | --- |
| Category | Library Name | Purpose / Description |
| Forms | Formik | Form state management and validation support |
| Forms | React Hook Form | Lightweight form handling using React hooks |
| Forms | Final Form | Framework-agnostic form management library |
| Validation | Yup | Schema-based validation (commonly with Formik) |
| Validation | Zod | Type-safe schema validation (TS friendly) |
| Validation | Validator.js | String validation (e.g. email, URL, etc.) |
| API Handling | Axios | Promise-based HTTP client |
| API Handling | Fetch API | Native browser HTTP request tool |
| API Handling | SWR | Data fetching with caching (by Vercel) |
| API Handling | React Query | Server state management and data fetching |
| API Handling | SuperAgent | Lightweight AJAX client |
| API Handling | JQuery AJAX | Legacy library for AJAX requests |
| Routing | React Router DOM | Declarative routing for React apps |
| Routing | Wouter | Minimalist routing library |
| UI Libraries | Material-UI (MUI) | Google's Material Design components |
| UI Libraries | React Bootstrap | Bootstrap components as React components |
| UI Libraries | Ant Design | Rich enterprise-level UI components |
| UI Libraries | Chakra UI | Accessible and themeable component library |
| State Management | Redux | Predictable state container |
| State Management | Zustand | Lightweight state management with hooks |
| State Management | Recoil | State management by Facebook with atoms/selectors |
| State Management | MobX | Reactive state management |
| Animation | Framer Motion | Simple and powerful animations |
| Animation | React Spring | Physics-based animation library |
| Animation | GSAP | Industry-level animation tool |
| Form Validation UI | React Input Mask | Mask input for form fields |
| Form Validation UI | React Phone Input | Phone number input with country dropdown |
| Utilities | Lodash | Utility functions (e.g. debounce, cloneDeep) |
| Utilities | Moment.js | Date and time manipulation (deprecated, use Day.js) |
| Utilities | Day.js | Lightweight alternative to Moment.js |
| Testing | Jest | JavaScript testing framework |
| Testing | React Testing Library | Testing React components the user way |
| Testing | Enzyme | Testing utility for React (less popular now) |
| Charts | Recharts | Charting library using D3 under the hood |
| Charts | Chart.js + react-chartjs-2 | Popular chart visualizations |
| Charts | Victory | Modular chart components |
| File Uploads | React Dropzone | Drag-and-drop file uploader |
| File Uploads | React Uploady | Modern file uploading toolkit |
| Icons | React Icons | Popular icon sets (Font Awesome, Material, etc.) |
| Icons | Heroicons React | Official React icons for Tailwind CSS |