1. **Web Development**
   1. **Frontend**
      1. HTML
         1. While using React JS there is not much use of HTML. But JSX is pretty much similar to HTML. If we have learnt HTML, it’s easier for us to switch to JSX. But since most of the templates out there is in HTML, we need to convert them to JSX.
            1. HTML to JSX: <https://magic.reactjs.net/htmltojsx.htm>
      2. CSS
         1. SCSS (Since I have already learnt CSS, it is easier for me to switch to SCSS rather than JSS/Styled components. Need to use unique class names in every file, otherwise they will conflict. Another trick is by nesting the class within a parent class. In this way only the parent class needs a unique name.)
            1. YouTube: <https://www.youtube.com/watch?v=_a5j7KoflTs>
            2. Install: Didn’t find compatible version of sass or node-sass for importing SCSS files to JS. So, I use vs extension to convert SCSS files to CSS and import the CSS file to JS.

Open VSCode Editor and Press ctrl+P, type **ext install live-sass**

* + - * 1. Use SCSS module for local scoping.
      1. Bootstrap (Makes my JSX code messy. Not useful for customization.)
         1. Docs: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>
         2. Install:

**npm install bootstrap** (Style using classes)

**npm install react-bootstrap** (Use styled components)

* + 1. JS
       1. React JS (Can make reusable components like navbar, sidebar to use on different pages, which was difficult in plain JS. In React JS we write code in ES6.)
          1. YouTube: <https://www.youtube.com/watch?v=Ke90Tje7VS0> (Here class components is used. But we will use function components and hooks at the end.)
          2. Install:

**npm install react** (To write JSX. This will also install react-dom which is needed for rendering JSX.)

**npm install create-react-app** (For creating react project)

**npm install react-router-dom** (For navigating to different pages and accessing different URLs)

* + - * 1. React hooks: <https://reactjs.org/docs/hooks-intro.html>

useState

useEffect: <https://www.youtube.com/watch?v=0ZJgIjIuY7U>

useRef: <https://www.youtube.com/watch?v=t2ypzz6gJm0>

useContext: <https://www.youtube.com/watch?v=5LrDIWkK_Bc>

useReducer: <https://www.youtube.com/watch?v=kK_Wqx3RnHk>

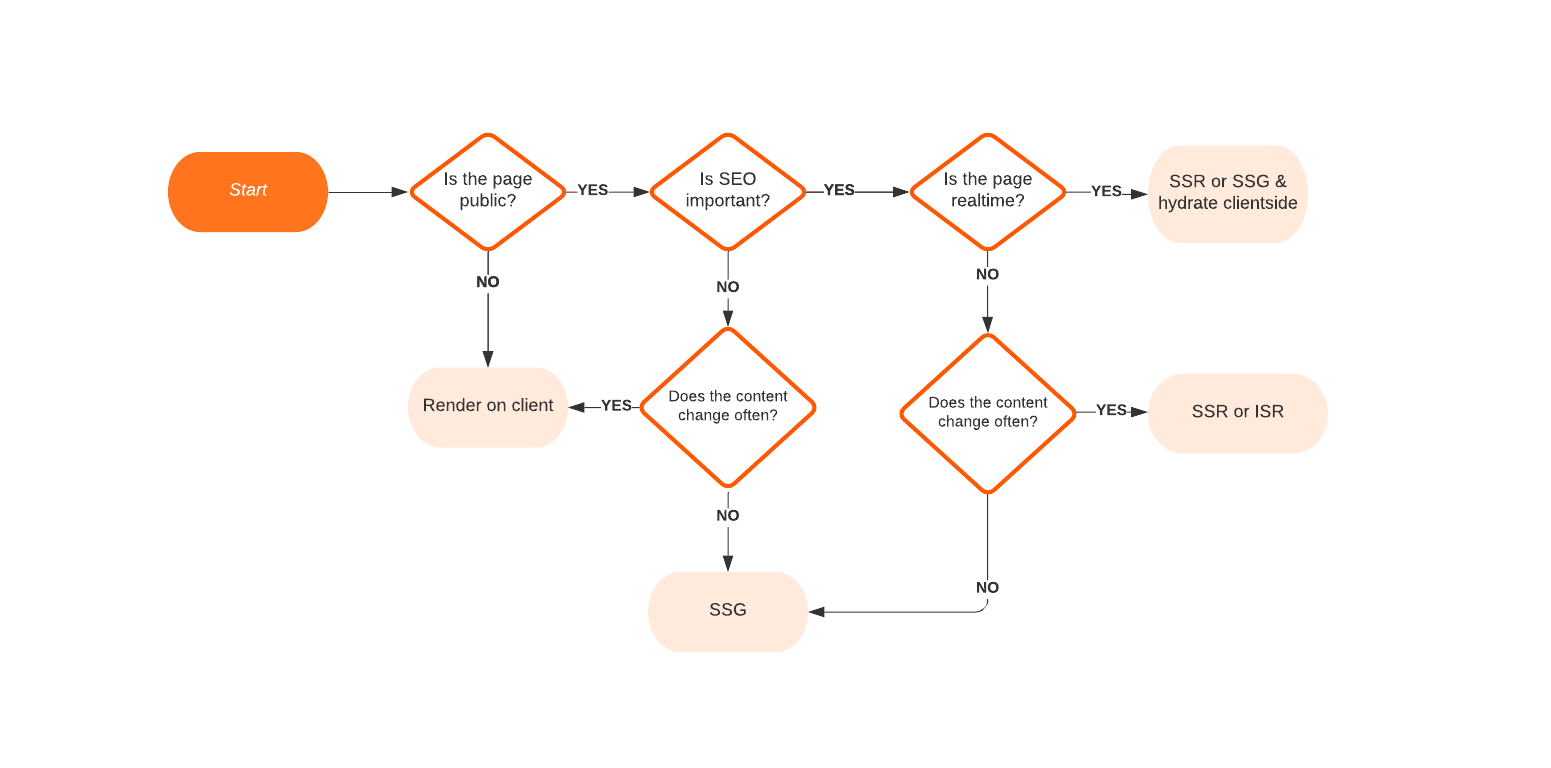
* + - 1. Next JS
         1. Install

**npm install next**

**npm install create-next-app**

* + - 1. Material UI (Styled components like Buttons)
         1. YouTube: <https://www.youtube.com/watch?v=vyJU9efvUtQ>
         2. Docs: <https://mui.com/material-ui/getting-started/installation/>
         3. Install: **npm install @mui/material @emotion/react @emotion/styled**
      2. Styled components: (Same issue. I don’t work with components that much.)
      3. SCSS Modules (Class names need to be in camel case)
      4. JSS (Class names need to be in camel case)
    1. Axios (For ajax calls to backend. Requires cors in the backend.)
       1. Docs: <https://axios-http.com/docs/intro>
       2. Install: **npm install axios**
  1. **Backend**
     1. **JS**
        1. **Node JS** (Besides running outside browser it has some functionality which is absent in plain JS. In Node JS we write code in Common JS.)
           1. **YouTube:** <https://www.youtube.com/watch?v=Oe421EPjeBE> (Include express JS too)
           2. **Install:** <https://nodejs.org/en/>
           3. **Nodemon:**  **npm install nodemon -D**
        2. **Express JS** (We will use express instead of http)
           1. **Install: npm install express**
     2. **Cors** (For request from client outside server’s domain)
        1. **npm install cors**
     3. **Socket.io** (For bidirectional connection between server and client)
     4. **Authentication**
        1. **JWT**
        2. **Bcrypto**

1. **UX**
   1. Profile link at top right corner
   2. Avoid dead end
   3. Scrolling is faster than clicking
   4. Don’t make pages too long
   5. Use blue color for links
   6. Make images clickable
   7. Button should look clickable
   8. Clickable elements should look clickable
   9. Scrollable elements should look scrollable
   10. Button that change or delete things should require more effort to click.
   11. Have a search field for big website.
   12. Search field should be a text box
   13. Search field should be long enough to see the entire query
   14. **Form**
       1. Show password eye button
       2. Label/Icon for input fields
       3. Caps lock warning (Optional)
       4. Show error before form submit
       5. **Login**
          1. Need an account? SIGN UP
          2. Auto focus on email. Make form accessible through keyboard.
          3. Forgot your password?
             1. Reset password field should remember which you already entered
          4. Not remember me. Use Log me out after
       6. **Sign Up**
          1. Already a user? LOGIN
          2. Show password requirements (If any)
          3. Keep the registration form short. Take only the necessary info. Email address, Password. Keep the other details for edit profile.
          4. No confirm password field
2. If any data is needed to be fetched to render a component, we should use  **or .**
3. For dynamic routes we should use if we want to use . Next will call for all the paths returned from **.**
4. If we don’t know all the paths possible in dynamic routes than we should use (Fallback: ’blocking’).
5. If our page data updates often then we should use ISR.
6. will run only at build time. will run at every render.



1. For my database project I will use ISR. Like for posts, profile, they are updated often.
2. fallback: ‘blocking’ All the paths returned by will be used to static site generation. But if any absent path is requested that will also be called at run time.
3. Won’t use CSS modules in next js project. For css ordering issue. Will use global css. S CSS file for each component.
4. Try BreadCrumb for easier navigation.