Gmail

**Siddharth Ram <siddharth.asterisk@gmail.com>**

**15 LPA+ web dev job checklist**
1 message

**Tanay Pratap from tanaypratap's letters** <tanaypratap@substack.com> Thu, May 13, 2021 at 12:37 PM
Reply-To: Tanay Pratap from tanaypratap's letters
<reply+lnd3e&hjh0g&&757c34a75f24f760f7b28e73837673a9d774f11764abc025df6ce94993b49e57@mg1.substack.com>
To: siddharth.asterisk@gmail.com

# 15 LPA+ web dev job checklist

Steps to be on top of the hill, no matter your college or background

Tanay Pratap
May 13

There's **no checklist** for 12LPA or higher-paying jobs. Currently, most of our placements are between 4.5-6.5LPA. Considering that our hiring partners **don't care about college, GPA,** or any background and that they offer these packages with simple interviews just because of projects in portfolios is a good turn in hiring space overall.

Anything **above 6.5LPA is considered an above-average package** in web-dev (except for top colleges). To get above average placements one needs to be above average as well. Treat the below list as something for the **4.5-15LPA+ range** and depending on how many skills you have outside your 5 projects and 3 blogs we might find a good fit for you.

Before going deep, I would like to point out that we can only send you for interviews, clearing is completely on you. With that clear, moving ahead.

Note: *If you're new here, and you don't know who's the "we" I'm talking about,* *it’s the roc8.careers team. We are revolutionizing tech hiring in India by making it move away from core DSA based rounds to project-based hiring. To know more check out https://roc8.careers.*

## knowledge base

1. **Problem Solving:** While the core of roc8 is on project-based hiring. The interviews for the company that pays anything above 9LPA would involve some PS/DS round. Don't expect something as complicated as Red Black Tree but at the same time knowing which data structure to use where, when to go for hashmaps and when to go for Linked Lists are important. Your **understanding of the data structures is important at this range.**
2. **Core JS Skill:** Using frameworks is good. And will get you a job, but if you're aiming for something top-notch, ask yourself, can I write plain JS? I am not talking about just vanillaJS based DOM scripting. I am talking about writing plain libraries or code in JavaScript. Can you write your own (toned-down versions of) Redux, React Router, your own polyfills, your own lodash, etc?
3. Basics of Optimising Web for Performance and how these things can help in making your web apps faster, the basics:

    1. Critical Rendering Path
    2. CDN
    3. Caching
    4. Bundling

    **Expectation:** Don't need to try everything on your app but having knowledge of the tools and techniques is a sure plus. Refer this to know about more topics: https://developer.mozilla.org/en-US/docs/Web/Performance

4. Basics of web security

    1. CSRF
    2. XSS
    3. HTTPS
    4. CORS

5. **Clean Coding:**

    1. variable and function naming
    2. file and folder structure
    3. proper error handling
    4. modularity
    5. usage of comments and documentation
    6. Tests added

6. Agile methodology: Software is developed in iterations and versions. Do you know what's agile? Do you open issues and PRs on your repo? Do you have some idea of Project Management? To start with you can use these features on Github itself: https://github.com/features/project-management/

## tangible skills

1. **CSS**

    1. vanilla CSS: Do not overlook CSS basics for frameworks when you're targeting this salary range. Given any screenshot, any interaction (hover, animation, etc) from a website you should be able to do it in plain CSS without any tool/framework.
    2. Design Sense: You don't need to be a designer but understanding typography, consistent padding, and margin, knowing about primary secondary color in web design, etc. are a huge plus. You should understand UX, usage of colors and animations to attract attention.
    3. PreProcessors

        1. PostCSS
        2. SaSS

    4. Frameworks (one or two of these) Expectation is that you understand how frameworks work. Not that you're fluent in these syntaxes. But the core of all frameworks is the ability to theme and consistency.

        1. Tailwind

        2. Bootstrap
        3. Material UI

    5. CSS-in-JS and modern tooling around it.

        1. Styled Components
        2. CSS Modules

2. **Typescript**

    1. Basics of TS
    2. Using TS in a React app. One or more projects completely built on React

3. **Testing**

    - TDD
    - Best practices
    - Jest
    - Unit Testing
    - Integration Testing

        - react-testing-library

    - E2E Testing

4. **Tooling**

    - DivOps (a playful naming for DevOps for FE)

        - npm scripts
        - Webpack and Optimisations

    - Github Actions

5. **Full Stack**

    1. Backend on NodeJS or any other language
    2. Anyone DB, and ODM. Preferably,

        1. NodeJS with MySQL + Sequelize or
        2. Mongo + Mongoose combo

3. Authentication

    1. JWT and middleware based auth
    2. PassportJS or something similar to do OAuth

    Note: Auth with Firebase is simple but you don't understand the basics of auth and thus, get limited to a tool.

4. Realtime Backend: Web Sockets. And consuming them on FE with JavaScript to make a real-time app.

6. **React Frameworks:** Knowledge of NextJS is a big plus and if you have projects to show in that you would stand out. Know Gatsby but not just for static sites. Have projects to **showcase your knowledge of JAMStack.**
7. **Project Variety:** Developing plugins for browsers, VSCode, bots for Discord, ESLint plugins, etc, or any such tool to showcase that you can work outside the MERN stack. It shows that you can read documentation aimed at devs and work with different APIs. Consider doing one or two such projects (outside the 5 you have submitted for roc8) to showcase these skills as well.
8. **a11y:** Basic understanding of semantic HTML. Knowing WCAG is a huge plus, even more, if you are able to use these in a React app. A good starting point would be: https://developer.mozilla.org/en-US/docs/Web/Accessibility

## communication

Teams work exceptionally well when members are helpful and expert at written and vocal communication. This is even more important in the remote work world in which we are living. Similar to how we use projects to showcase our programming skills, question is, **how can we show our communication skills?**

1. **Technical Blogging,** not just for the sake of roc8. Have consistent blogs documenting your learning journey for over a year. Even better if you're writing specifically on web dev and you have a newsletter.
2. Open Source: Issues, PRs, helping others on forums like Reddit
3. StackOverflow Reputation

4. **Technical Talks:** Not in the international conferences but definitely some in local meetups would be a lot helpful.
5. Propensity to guide others: Mentorship in college, Discord servers etc.
6. Developer Story and engagement on Twitter and LinkedIn
7. Clear and Concise Documentation for all your projects on Github.

> not just for the sake of roc8. Have consistent blogs documenting your learning journey for over a year.

## overall knowledge

Nobody expects you to know everything, but if you're aiming to be on top then people definitely expect you to be plugged into the ecosystem. This means that words like

- web assembly,
- PWA,
- React Native,
- Reconciliation
- GraphQL,
- Electron,
- SSR,
- SSG and many more, shouldn't sound foreign to you.
- Some idea about what's happening in other frameworks as well: VueJS, Angular, Preact, Svelte, etc

## next steps

For the next few days, use this list to calibrate yourself and see where you are. The question which might be popping up in your head if you're graduating would be, “Should I learn everything on the list and apply for jobs or go for jobs and keep learning?”. I have the answer to that as well but as always it's a detailed answer and I'll send it in the next newsletter.

Comment with your score. Or, comment with any question you have. But **do comment.** Having conversations would make this doc even more clear for you and for subsequent readers.

*If you liked this post from tanaypratap's letters, why not share it?*

Share