## **🧑‍💻 What You Should Know as a Web Developer (Advanced Level)**

These are foundational and practical insights, tools, and knowledge areas that **every modern web developer** should understand.

### **🔧 1. Core Web Technologies**

* **HTML5** – Semantic elements, accessibility, forms.
* **CSS3** – Flexbox, Grid, Media Queries, Custom Properties.
* **JavaScript (ES6+)** – Arrow functions, destructuring, async/await, promises, modules.

### **⚛️ 2. Frontend Frameworks**

* Deep understanding of frameworks like:  
  + **React.js** – hooks, context, component lifecycle.
  + **Vue.js** – composition API, directives.
  + **Next.js/Nuxt.js** – routing, SSR/SSG/ISR, API routes.

### **🗃️ 3. Package Management**

* Use of **npm** or **yarn** for dependency management.
* Understand **versioning**, package.json, and scripts.

### 

### **📦 4. Modules & Bundlers**

* **ES Modules**, import/export.
* Tools like **Vite**, **Webpack**, **Parcel** – used for module bundling, hot reload, and optimization.

### **🌐 5. HTTP & Networking Basics**

* **HTTP methods** – GET, POST, PUT, DELETE, PATCH.
* **Status codes** – 200, 201, 400, 401, 403, 404, 500.
* **Cookies vs. LocalStorage vs. SessionStorage.**
* **CORS**, Preflight requests, and headers.

### **⚙️ 6. Git & Version Control**

* Git basics – clone, commit, branch, merge, rebase.
* Collaboration using **GitHub**, **GitLab**, or **Bitbucket**.
* Pull requests, branching strategies (Git Flow).

### **🛠️ 7. Dev Tools & Debugging**

* **Chrome DevTools** – console, network tab, performance tab.
* Debugging with **breakpoints** and **error stack traces**.
* Using **Postman** or **Insomnia** to test APIs.

### **🗄️ 8. Database Knowledge**

* SQL (PostgreSQL, MySQL): Joins, indexes, transactions.
* NoSQL (MongoDB): Documents, collections, querying.
* ORMs: Prisma, Mongoose, Sequelize – abstraction over raw DB queries.

### **🔐 9. Authentication and Security**

* **Sessions vs JWT tokens**.
* **OAuth2**, Google/GitHub login.
* CSRF, XSS, SQL Injection – how to prevent them.
* Password hashing (bcrypt, argon2).

### **📦 10. Component-based Architecture**

* Thinking in **components** and **state**.
* Reusable, composable components.
* Use of UI libraries (ShadCN, Material UI, Tailwind UI).

### **🌍 11. Environment & Configuration**

* .env files for storing sensitive configs.
* Using dotenv or environment variables in deployment.

### 

### **🌐 12. APIs: REST & GraphQL**

* Designing and consuming RESTful APIs.
* Querying GraphQL APIs using Apollo or URQL.

### **📈 13. Web Performance & Optimization**

* Lazy loading, code splitting, tree shaking.
* Image optimization, responsive images (<picture> tag).
* Lighthouse audits & Core Web Vitals.

### **🚀 14. Hosting & Deployment**

* Platforms: Vercel, Netlify, Render, Railway, AWS.
* CI/CD workflows with GitHub Actions.
* Build and deploy static and dynamic full-stack apps.

### **📱 15. Responsive & Mobile-First Design**

* Viewport settings, fluid grids, adaptive breakpoints.
* Mobile testing, touch interactions.

### 

### 

### **🌎 16. SEO & Open Graph Protocol**

* Meta tags, Open Graph tags for social media sharing.
* Dynamic title & description rendering.

### **📘 17. Documentation & Code Quality**

* Writing clean, readable, commented code.
* Using JSDoc or TypeScript for type safety.
* Keeping README.md, folder structure organized.

### **🧪 18. Testing**

* **Unit Testing** – using Jest, Vitest.
* **Integration Testing** – testing API + UI together.
* **E2E Testing** – using Cypress or Playwright for full flows.

### **🧠 19. Problem Solving & System Design**

* Data structures & algorithms basics.
* Design scalable systems (e.g., file sharing, social media apps).
* Use diagrams (ERD, flowcharts, sequence diagrams).

### 

### 

### **📚 20. Keeping Up with Trends**

* Follow platforms like:  
  + [Frontend Mastery](https://www.frontendmasters.com/)
  + [JavaScript Weekly](https://javascriptweekly.com/)
  + [Smashing Magazine](https://www.smashingmagazine.com/)
  + [Dev.to](https://dev.to/)
  + [GitHub trending](https://github.com/trending)

## **🧳 Summary: What Makes a Good Web Developer at an Advanced Level?**

| **Skill** | **Description** |
| --- | --- |
| 🧠 Thinking in Components | Break UI into reusable chunks |
| 🌍 Understanding the Full Stack | Know how frontend connects to backend |
| 📦 Managing State & APIs | Data flow, caching, and API syncing |
| 🔐 Keeping Apps Secure | Understanding common threats & mitigations |
| 🛠 Writing Scalable Code | Modularity, readability, and maintainability |
| 🚀 Shipping Fast | Deploying, testing, and monitoring apps reliably |