Learning JavaScript at an intermediate level means diving deeper into the language's nuances, features, and best practices. Here is a guide to help you on your journey.

**Concepts to Focus On**

* Advanced DOM Manipulation
* Asynchronous Programming (Callbacks, Promises, and Async/Await)
* ES6+ Features (Arrow functions, destructuring, spread/rest operator, etc.)
* Closures and Scoping
* Modules and Namespacing
* Functional Programming Principles
* Object-Oriented Programming Principles
* Error Handling and Debugging
* Memory Management
* Testing (Unit Testing, E2E Testing)
* Frontend and Backend Frameworks (React, Angular, Node.js, etc.)

**Resources**

Books

1. "You Don't Know JS" (book series) by Kyle Simpson
2. "JavaScript: The Good Parts" by Douglas Crockford
3. "Eloquent JavaScript" by Marijn Haverbeke (Advanced Chapters)
4. "JavaScript Patterns" by Stoyan Stefanov

Online Courses

1. [Frontend Masters](https://frontendmasters.com/)
2. [Udemy](https://www.udemy.com/)

* "JavaScript: Understanding the Weird Parts"
* "Modern JavaScript From The Beginning"

1. [Pluralsight](https://www.pluralsight.com/)
2. [Egghead.io](https://egghead.io/)

Websites and Blogs

1. [Mozilla Developer Network (MDN) Web Docs](https://developer.mozilla.org/en-US/docs/Web/JavaScript)
2. [JavaScript Info](https://javascript.info/)
3. [2ality](http://2ality.com/) by Dr. Axel Rauschmayer
4. [David Walsh Blog](https://davidwalsh.name/)
5. [Hackernoon JavaScript Section](https://hackernoon.com/tagged/javascript)

YouTube Channels

1. [Academind](https://www.youtube.com/c/Academind)
2. [Traversy Media](https://www.youtube.com/user/TechGuyWeb)
3. [The Net Ninja](https://www.youtube.com/c/TheNetNinja)
4. [FunFunFunction](https://www.youtube.com/c/mpjme)

Practice Websites

1. [Exercism](https://exercism.io/tracks/javascript/exercises)
2. [LeetCode](https://leetcode.com/) (JavaScript section)
3. [HackerRank](https://www.hackerrank.com/domains/tutorials/10-days-of-javascript)
4. [Codewars](https://www.codewars.com/)

Frameworks

1. [React](https://reactjs.org/)
2. [Angular](https://angular.io/)
3. [Vue.js](https://vuejs.org/)
4. [Node.js](https://nodejs.org/en/)

Tools

* Debuggers: Chrome DevTools, Firebug, Visual Studio Code Debugger
* Linters: ESLint, JSHint
* Build Tools: Webpack, Babel

GitHub Repositories

* Follow open-source JavaScript projects
* Contribute to projects to learn by doing

**Plan**

1. **Week 1-2**: Brush up on advanced JavaScript fundamentals.
2. **Week 3-4**: Dive into ES6+ features and asynchronous programming.
3. **Week 5-6**: Study and apply design patterns.
4. **Week 7-8**: Implement small projects incorporating these advanced concepts.
5. **Week 9-10**: Dive into frameworks like React or Node.js.
6. **Week 11-12**: Study testing frameworks and write unit tests for your projects.

Remember, this journey is about depth, not just breadth. Consistent practice and project implementation are key to solidifying your intermediate skills in JavaScript.

Expert RoadMap

Achieving expert-level proficiency in JavaScript requires not only a strong understanding of the language itself but also its ecosystem, best practices, and the various tools and frameworks that are commonly used. Below is a roadmap, broken down into different categories, to guide you towards becoming an expert in JavaScript.

**Fundamentals:**

1. **Official Documentation**: Mozilla Developer Network (MDN) Web Docs is a go-to resource for accurate information.
2. **Books**:
   * "Eloquent JavaScript" by Marijn Haverbeke
   * "You Don't Know JS" (book series) by Kyle Simpson
   * "JavaScript: The Good Parts" by Douglas Crockford
3. **Online Courses**:
   * FreeCodeCamp's JavaScript Algorithms and Data Structures Certification
   * Udemy: "JavaScript: Understanding the Weird Parts" by Anthony Alicea
4. **Practice Sites**:
   * Exercism
   * HackerRank
   * LeetCode
   * Project Euler
5. **YouTube Channels**:
   * The Net Ninja
   * Traversy Media
   * Academind

**Advanced JavaScript:**

1. **Understanding ECMAScript**:
   * Keep an eye on the TC39 proposals and updates to stay up-to-date with the language features.
2. **Design Patterns and Architecture**:
   * Book: "JavaScript Design Patterns" by Addy Osmani
3. **Event Loop and Concurrency Model**:
   * Research how JavaScript's non-blocking architecture works.
4. **Memory Management**:
   * Learn about garbage collection, closures, and memory leaks.
5. **Performance Optimization**:
   * Google's Web.dev guides
   * Tools like Lighthouse for performance assessment

**Front-end:**

1. **Frameworks and Libraries**:
   * React (official docs, "The Road to React" book)
   * Angular (official docs, Angular University courses)
   * Vue.js (official docs, Vue Mastery courses)
2. **Package Managers**:
   * npm
   * yarn
3. **Build Tools**:
   * Webpack
   * Parcel
   * Rollup.js
4. **CSS Preprocessors and Styled-components**:
   * Sass, Less
   * Styled-components, Emotion
5. **State Management**:
   * Redux
   * MobX
   * Context API
6. **Static Site Generators**:
   * Next.js
   * Gatsby

**Back-end:**

1. **Node.js**:
   * Official Documentation
   * "Node.js Design Patterns" book
2. **Web Frameworks**:
   * Express.js
   * NestJS
   * Koa.js
3. **Databases**:
   * MongoDB
   * MySQL
   * PostgreSQL
4. **Authentication & Authorization**:
   * JSON Web Tokens (JWT)
   * OAuth
   * Passport.js
5. **Testing**:
   * Jest
   * Mocha & Chai
   * Cypress for end-to-end testing
6. **APIs**:
   * REST
   * GraphQL (Apollo Server, Relay)

**DevOps:**

1. **Docker and Kubernetes**
2. **Continuous Integration and Continuous Deployment (CI/CD)**:
   * Jenkins
   * GitHub Actions
   * GitLab CI
3. **Cloud Providers**:
   * AWS
   * Azure
   * Google Cloud Platform

**Soft Skills:**

1. **Code Reviews**
2. **Working with Teams**
3. **Problem-Solving**

**Follow Industry Leaders and Blogs:**

1. **Twitter**
2. **Medium**
3. **Dev.to**
4. **JavaScript Weekly newsletter**

**Open Source Contributions**

**Real-world Projects and Portfolio**

**Keep Learning and Teaching**

* Participate in meetups, webinars, and conferences.
* Write blogs, make YouTube tutorials, or even teach courses.

The resources mentioned are not exhaustive, and you might find other valuable resources along the way. The key to achieving an expert level in JavaScript is consistent learning and practical implementation.