JavaScript Roadmap

Foundations

Basic concepts and syntax

Core

Essential JavaScript principles

ES6+

Modern JavaScript features

Advanced

Complex JavaScript techniques

OOP

Object-oriented programming

Browser

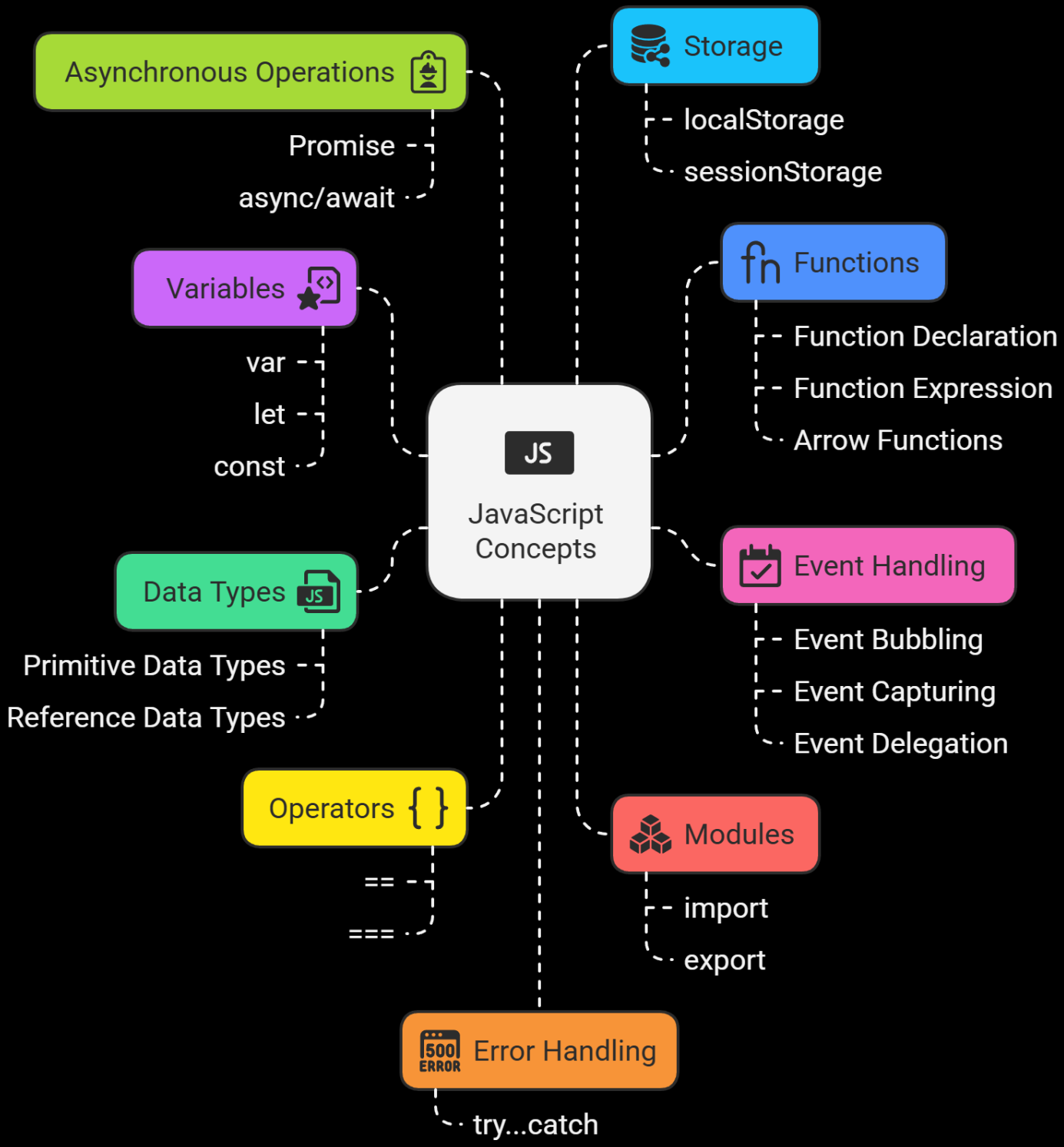
Web browser interactions

Mastery

Expert-level JavaScript skills

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

JavaScript Concepts and Differences



©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

Phase 1: Foundations

1. What is JavaScript?

- Learn definition: *Synchronous, single-threaded, multi-paradigm, object-oriented scripting language*.
- Understand its role: runs on both client (browser) and server (Node.js).
- Execution Context: memory component + code component.

2. Basics of JS Syntax

- Variables: var, let, const
- Comments (//, /* */)
- Data Types:
  - Primitives → number, string, boolean, undefined, null, symbol
  - Reference → object, array, function

3. Operators

- Arithmetic, Assignment, Comparison, Logical, Ternary, Bitwise, Type-check (typeof, instanceof).

Phase 2: Core Building Blocks

4. Data Structures in JS

- Arrays → methods, iteration, map, filter, reduce.
- Strings → manipulation & built-in methods.

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

- Objects → key-value access, destructuring.

5. **Control Flow**

- Conditional statements: if, else, switch.
- Loops → for, while, do...while, for...in, for...of.

6. **Functions**

- Declaration vs Expression
- Arrow functions
- Default parameters, Rest & Spread operators.

Phase 3: Modern JavaScript (ES6+)

7. **ES6+ Features**

- Template literals
- Destructuring assignment
- Modules (import / export)
- Spread/Rest operators
- let, const, class, super, extends

8. **Modularity & Functional Programming**

- Pure functions
- Higher-order functions
- Method chaining

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

Phase 4: Advanced Core Concepts

9. Asynchronous JavaScript

- Synchronous vs Asynchronous execution
- Callbacks & Higher-order functions
- Callback Hell → Pyramid of Doom
- Promises (creation, chaining, error handling)
- async/await

10. Behind the Scenes of JS

- Call Stack
- Hoisting
- Scope & Lexical Environment
- Scope Chain & Closures
- Function Currying
- 3 ways to add JS code (inline, internal, external).

Phase 5: Object-Oriented JS

11. OOP in JS

- this keyword
- Constructor functions & Classes
- Prototype & Prototype chaining
- call, apply, bind.

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

Phase 6: Browser & Environment

12. Browser-Specific Topics

- Window object
- Events (inline, DOM level, event listeners)
- Event bubbling, capturing, delegation.

13. Miscellaneous Concepts

- Truthy & Falsy values
- Deep Copy vs Shallow Copy
- JSON handling

Phase 7: Mastery & Best Practices

14. Design & Coding Practices

- Writing clean code
- Error handling (try...catch...finally)
- Debugging (console, breakpoints, dev tools)

15. Applied JavaScript

- DOM Manipulation
- Fetch API / AJAX
- LocalStorage & SessionStorage
- Modular projects with ES Modules

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

Phase 8: Testing & Debugging

- Debugging with Browser DevTools
- Common Errors (ReferenceError, TypeError, SyntaxError)
- Writing Unit Tests (Jest or Mocha basics)

Phase 9: Ecosystem & Frameworks

- NPM (packages, scripts)
- Bundlers (Vite, Webpack basics)
- Transpilers (Babel intro)
- Introduction to TypeScript (optional but recommended)
- Frontend Libraries: React (recommended after mastering JS)
- Backend with Node.js & Express (Intro)

By the end of this roadmap, a developer will not just know **syntax**, but also **how to apply JavaScript in real projects**.

JavaScript Interview Questions:

1. What is JavaScript and where does it run?
2. What are the primitive and reference data types?
3. What is the difference between var, let, and const?
4. What is the difference between == and ===?
5. What are truthy and falsy values in JavaScript?

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.

6. What is the difference between null and undefined?
7. Explain variable scope and hoisting in JavaScript.
8. What is the difference between function declaration and function expression?
9. What are arrow functions and how do they differ from normal functions?
10. Explain closures with an example.
11. What are template literals and destructuring assignments?
12. What are JavaScript modules (import / export)?
13. What is a Promise and how does async/await work?
14. What is the event loop in JavaScript?
15. What is the this keyword and how does it behave in different contexts?
16. What is a prototype and how does prototype chaining work?
17. What is event bubbling, capturing, and delegation?
18. What is the difference between localStorage and sessionStorage?
19. How do you convert a JavaScript object to JSON and back?
20. How do you handle errors in JavaScript (try...catch)?

©2025 Rishabh Singh [@itsindrajput]. Feel free to share.