

WEB DEVELOPMENT

Complete Professional Course

Beginner to Intermediate • Self-Paced • Practical Career-Focused Training

HTML • CSS • JavaScript • React • Node.js • Express • SQL • APIs • Git • Deployment

Professional learning material designed for students, aspiring developers, freelancers and junior web professionals.

Table of Contents

Module	Topic
1	Web Development Foundations
2	HTML5 & Semantic Web
3	CSS3 & Responsive Design
4	JavaScript Fundamentals
5	Advanced JavaScript & Browser APIs
6	Git & GitHub Workflow
7	Modern Frontend Development
8	React.js Fundamentals
9	React.js Professional Development
10	Backend Development with Node.js
11	Express.js & REST APIs
12	Databases & SQL
13	Authentication & Web Security
14	Full-Stack Application Architecture
15	Testing, Debugging & Performance
16	Deployment, Hosting & DevOps Basics
17	Freelancing & Professional Web Projects
18	Complete Full-Stack Capstone & Career Preparation

How to Use This Course

Work through the modules in order. Complete the practical labs and mini projects before attempting the capstone. Keep all source code in a Git repository, document your decisions, and test every feature before deployment.

Recommended outcome: By the end of the course, learners should be able to create responsive websites, build interactive frontends, develop REST APIs, connect databases, implement authentication, use Git/GitHub, and deploy a complete full-stack application.

1. Web Development Foundations

- How the web works: browsers, servers, DNS, HTTP/HTTPS and hosting
- Frontend vs backend vs full-stack development
- Web development workflow, project planning and developer tools
- How to structure a professional website project

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

2. HTML5 & Semantic Web

- HTML document structure and semantic elements
- Headings, paragraphs, links, images, lists and tables
- Forms, validation, accessibility and metadata
- Semantic layout, SEO-friendly markup and reusable page structure

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

3. CSS3 & Responsive Design

- Selectors, cascade, inheritance and specificity
- Box model, typography, colors, spacing and positioning
- Flexbox, Grid and modern layout techniques
- Responsive design, media queries and mobile-first development

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

4. JavaScript Fundamentals

- Variables, data types, operators and control flow
- Functions, arrays, objects and modern ES6+ syntax
- DOM manipulation and browser events
- Debugging, error handling and clean JavaScript practices

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

5. Advanced JavaScript & Browser APIs

- Modules, destructuring, spread/rest and asynchronous JavaScript
- Promises, async/await and Fetch API
- Local storage, JSON and client-side state
- Form handling, validation and practical API consumption

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

6. Git & GitHub Workflow

- Repositories, commits, branches and merges
- Pull requests, collaboration and issue tracking
- Professional README files and versioning
- Basic Git workflow for freelance and team projects

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

7. Modern Frontend Development

- Component-based architecture and reusable UI
- Project structure, package managers and build tools
- Frontend forms, state, routing and API integration
- Performance, accessibility and maintainability

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

8. React.js Fundamentals

- React concepts, components and JSX
- Props, state, events and conditional rendering
- Lists, forms and reusable components
- React project structure and development workflow

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

9. React.js Professional Development

- Hooks and component lifecycle concepts
- Routing, API integration and loading/error states
- Reusable UI patterns and state management concepts
- Frontend security, performance and deployment readiness

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

10. Backend Development with Node.js

- Node.js runtime, npm and project setup
- Modules, environment variables and server architecture
- HTTP servers, middleware and request/response handling
- Building maintainable backend services

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

11. Express.js & REST APIs

- Express routing and middleware
- REST principles, CRUD operations and status codes
- Validation, error handling and API documentation
- Authentication concepts and secure API practices

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

12. Databases & SQL

- Relational database concepts and data modeling
- Tables, keys, relationships and normalization
- SQL SELECT, INSERT, UPDATE, DELETE, JOIN and aggregation
- Connecting backend applications to databases

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

13. Authentication & Web Security

- Sessions, tokens and authentication flows
- Password hashing, authorization and role-based access
- Input validation, secure cookies and environment secrets
- OWASP-aware web security and common vulnerabilities

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

14. Full-Stack Application Architecture

- Connecting React frontend with Node/Express backend
- Database-driven CRUD applications
- API contracts, error states and project organization
- Production-minded architecture and maintainability

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

15. Testing, Debugging & Performance

- Browser developer tools and systematic debugging
- Frontend and backend testing concepts
- Performance optimization, caching and asset optimization
- Accessibility, SEO basics and quality checklists

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

16. Deployment, Hosting & DevOps Basics

- Production builds and environment configuration
- Hosting frontend and backend applications
- Domains, HTTPS, deployment workflow and logs
- Basic CI/CD concepts and production monitoring

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

17. Freelancing & Professional Web Projects

- Requirement gathering and scope definition
- Estimating work, milestones and deliverables
- Client communication, documentation and handover
- Portfolio projects and professional GitHub presentation

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

18. Complete Full-Stack Capstone & Career Preparation

- Plan and build a production-style web application
- Architecture, UI, API, database, authentication and deployment
- Testing, documentation and final presentation
- Developer portfolio, interview preparation and next-step roadmap

Professional Practice

Apply the concepts to a small working feature. Use meaningful names, reusable components, validation, clear folder structure, version control and concise documentation. Prefer accessibility, security and maintainability over shortcuts.

Learning Check

- Explain the main concept in your own words.
- Build a small feature without copying a complete solution.
- Debug at least one intentional error.
- Commit the work to Git with a meaningful message.

10 Practical Labs

Lab 1 — Build a semantic personal profile page with HTML.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 2 — Create a responsive landing page using CSS Flexbox/Grid.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 3 — Build a JavaScript form with validation and DOM updates.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 4 — Consume a public JSON API with Fetch and async/await.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 5 — Create a Git repository with branches and a professional README.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 6 — Build a React component-based task manager.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 7 — Create an Express REST API with CRUD endpoints.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 8 — Design a SQL database and implement CRUD queries.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 9 — Add authentication and protected routes to a full-stack app.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

Lab 10 — Deploy a frontend + backend application and verify production behavior.

Deliverable: source code, short README, screenshots or test notes, and a brief reflection.

5 Mini Projects

Mini Project 1: Responsive Business Website

Multi-page responsive site with navigation, services, contact form and basic SEO.

Mini Project 2: JavaScript Expense Tracker

Add/edit/delete expenses, totals, validation and local storage.

Mini Project 3: React Task Management App

Reusable components, filters, state management and API-ready structure.

Mini Project 4: REST API Service

Express CRUD API with validation, error handling and database integration.

Mini Project 5: Full-Stack Booking System

React frontend, Node/Express API, SQL database and authenticated user flow.

Final Capstone Project

Production-Style Full-Stack E-Commerce / Service Booking Platform

Build a complete web application that demonstrates the skills from the course. The project should include a responsive React frontend, Node.js/Express REST API, SQL database, authentication and authorization, validation, error handling, Git/GitHub workflow, testing, security controls, deployment and documentation.

- Requirements & user stories
- Responsive UI and reusable components
- Database schema and CRUD operations
- Authentication and protected routes
- REST API documentation
- Admin/user roles where appropriate
- Validation, security and error handling
- Deployment and environment configuration
- Testing and performance checklist
- Professional README and final presentation

100 Multiple-Choice Questions

Select the best answer. The answer key is provided at the end of this section.

1. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

2. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

3. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

4. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

5. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

6. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

7. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

8. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML

D. RENDER

9. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

10. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

11. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

12. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

13. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

14. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

15. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

16. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

17. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

18. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

19. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

20. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

21. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

22. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

23. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

24. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

25. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

26. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

27. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

28. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

29. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

30. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

31. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

32. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

33. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const

- C. class
- D. import

34. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

35. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

36. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

37. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

38. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

39. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

40. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

41. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

42. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

43. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

44. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

45. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

46. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

47. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

48. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

49. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

50. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

51. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

52. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

53. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

54. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

55. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

56. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

57. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

58. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH

C. FETCHHTML

D. RENDER

59. [Security] What is input validation intended to do?

A. Check data against expected rules

B. Increase image size

C. Delete logs

D. Disable HTTPS

60. [Deployment] What does HTTPS provide?

A. Encrypted transport between client and server

B. A database schema

C. A JavaScript compiler

D. A CSS reset

61. [HTML] Which element is normally used for the main heading of a page?

A.

B.

C.

D.

62. [CSS] Which layout system is designed for one-dimensional layouts?

A. Flexbox

B. SQL

C. DOM

D. JSON

63. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

A. let

B. const

C. class

D. import

64. [Git] Which command creates a local repository?

A. git init

B. git push

C. git clone

D. git merge

65. [React] What is JSX primarily used for?

A. Writing UI markup in JavaScript

B. Managing SQL tables

C. Hosting servers

D. Encrypting passwords

66. [Node.js] What is Node.js?

A. A JavaScript runtime

B. A database

C. A CSS framework

D. A browser

67. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

68. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

69. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

70. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

71. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

72. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

73. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

74. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

75. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

76. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

77. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

78. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

79. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

80. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

81. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

82. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

83. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const

- C. class
- D. import

84. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

85. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

86. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

87. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

88. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

89. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

90. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

91. [HTML] Which element is normally used for the main heading of a page?

- A.
- B.
- C.
- D.

92. [CSS] Which layout system is designed for one-dimensional layouts?

- A. Flexbox
- B. SQL
- C. DOM
- D. JSON

93. [JavaScript] Which keyword declares a block-scoped variable that can be reassigned?

- A. let
- B. const
- C. class
- D. import

94. [Git] Which command creates a local repository?

- A. git init
- B. git push
- C. git clone
- D. git merge

95. [React] What is JSX primarily used for?

- A. Writing UI markup in JavaScript
- B. Managing SQL tables
- C. Hosting servers
- D. Encrypting passwords

96. [Node.js] What is Node.js?

- A. A JavaScript runtime
- B. A database
- C. A CSS framework
- D. A browser

97. [Express] What is Express commonly used for?

- A. Building Node.js web servers/APIs
- B. Image editing
- C. SQL administration
- D. CSS compilation

98. [SQL] Which command retrieves rows from a table?

- A. SELECT
- B. PUSH
- C. FETCHHTML
- D. RENDER

99. [Security] What is input validation intended to do?

- A. Check data against expected rules
- B. Increase image size
- C. Delete logs
- D. Disable HTTPS

100. [Deployment] What does HTTPS provide?

- A. Encrypted transport between client and server
- B. A database schema
- C. A JavaScript compiler
- D. A CSS reset

MCQ Answer Key

1: A	2: A	3: A	4: A	5: A	6: A	7: A	8: A	9: A	10: A
11: A	12: A	13: A	14: A	15: A	16: A	17: A	18: A	19: A	20: A
21: A	22: A	23: A	24: A	25: A	26: A	27: A	28: A	29: A	30: A
31: A	32: A	33: A	34: A	35: A	36: A	37: A	38: A	39: A	40: A
41: A	42: A	43: A	44: A	45: A	46: A	47: A	48: A	49: A	50: A
51: A	52: A	53: A	54: A	55: A	56: A	57: A	58: A	59: A	60: A
61: A	62: A	63: A	64: A	65: A	66: A	67: A	68: A	69: A	70: A
71: A	72: A	73: A	74: A	75: A	76: A	77: A	78: A	79: A	80: A
81: A	82: A	83: A	84: A	85: A	86: A	87: A	88: A	89: A	90: A
91: A	92: A	93: A	94: A	95: A	96: A	97: A	98: A	99: A	100: A

50 Web Development Interview Questions & Answers

1. What is semantic HTML?

HTML that communicates the meaning and structure of content through appropriate elements.

2. What is responsive design?

A design approach that adapts layout and usability to different screen sizes and devices.

3. What is the CSS box model?

Content, padding, border and margin make up the box model.

4. Flexbox vs Grid?

Flexbox is mainly one-dimensional; Grid is designed for two-dimensional layouts.

5. What is the DOM?

The browser's object representation of an HTML document that JavaScript can read and modify.

6. What is event bubbling?

An event can propagate from a target element upward through its ancestors.

7. What is a Promise?

An object representing the eventual success or failure of an asynchronous operation.

8. Why use `async/await`?

It provides a readable syntax for working with Promise-based asynchronous code.

9. What is an API?

A defined interface through which software systems exchange data or functionality.

10. What is REST?

An architectural style commonly used for resource-oriented web APIs over HTTP.

11. What is JSON?

A lightweight text-based data format commonly used for structured data exchange.

12. What is React?

A JavaScript library for building component-based user interfaces.

13. What are React props?

Inputs passed from a parent component to a child component.

14. What is React state?

Data managed by a component that can change and trigger a UI update.

15. What is a React hook?

A function that lets components use React features such as state and effects.

16. What is Node.js?

A runtime that executes JavaScript outside the browser, commonly for servers.

17. What is middleware in Express?

A function that can inspect or modify requests/responses or control the request pipeline.

18. What is CRUD?

Create, Read, Update and Delete—the common operations for persistent data.

19. Primary key?

A column or set of columns that uniquely identifies a database row.

20. Foreign key?

A field that references a key in another table to represent a relationship.

21. What is normalization?

Organizing relational data to reduce redundancy and improve integrity.

22. What is authentication?

Verifying who a user is.

23. What is authorization?

Determining what an authenticated user is allowed to do.

24. Why hash passwords?

To avoid storing plaintext passwords and reduce damage if stored credentials are exposed.

25. What is CORS?

A browser security mechanism governing cross-origin requests.

26. What is Git?

A distributed version control system.

27. What is a branch?

An independent line of development in a Git repository.

28. What is a pull request?

A workflow for proposing and reviewing changes before merging them.

29. What is deployment?

Making an application available in a target environment for users.

30. What is CI/CD?

Automated practices for integrating, testing and delivering software changes.

31. What is environment configuration?

Managing values such as API keys and database URLs outside source code.

32. Why use environment variables?

To keep environment-specific configuration and secrets out of committed code.

33. What is caching?

Storing reusable data closer to where it is needed to reduce repeated work.

34. What is lazy loading?

Loading resources only when they are needed, improving initial load performance.

35. What is accessibility?

Designing websites so people with different abilities can use them effectively.

36. What is SEO?

Practices that help search engines understand and discover web content.

37. What is XSS?

A vulnerability where untrusted content can execute scripts in a user's browser.

38. What is SQL injection?

A vulnerability caused by unsafe construction of database queries from untrusted input.

39. How can SQL injection be reduced?

Use parameterized queries/prepared statements, validation and least-privilege database accounts.

40. Why validate on both client and server?

Client validation improves UX; server validation is necessary because clients cannot be trusted.

41. What makes an API maintainable?

Clear routes, consistent responses, validation, error handling, documentation and tests.

42. How do you debug a web app?

Reproduce the issue, inspect logs/devtools, isolate the cause, fix it and add a regression test when appropriate.

43. Why use reusable components?

They reduce duplication and make interfaces easier to maintain consistently.

44. What is separation of concerns?

Keeping different responsibilities distinct so each part can be developed and changed more safely.

45. What should a professional README include?

Purpose, setup, environment requirements, commands, architecture notes and usage instructions.

46. What should you consider before deployment?

Security, environment variables, database migrations, logging, HTTPS, backups, performance and rollback options.

47. How do you improve frontend performance?

Optimize assets, reduce unnecessary work, use caching/lazy loading and measure bottlenecks.

48. How do you secure a web application?

Validate input, use secure authentication, protect secrets, apply authorization, HTTPS, safe headers and dependency updates.

49. What makes a good portfolio project?

A real problem, polished UI, working features, clear README, deployed demo and understandable source code.

Glossary

API — Application Programming Interface used for software-to-software communication.

DOM — Document Object Model.

HTTP — Protocol used for web communication.

HTTPS — HTTP protected by TLS encryption.

URL — Uniform Resource Locator.

JSON — JavaScript Object Notation.

CRUD — Create, Read, Update, Delete.

REST — Resource-oriented architectural style for web services.

SQL — Structured Query Language.

JWT — JSON Web Token, commonly used for stateless authentication flows.

CORS — Cross-Origin Resource Sharing.

CI/CD — Continuous Integration and Continuous Delivery/Deployment.

IDE — Integrated Development Environment.

DNS — Domain Name System.

CDN — Content Delivery Network.

SEO — Search Engine Optimization.

XSS — Cross-Site Scripting.

CSRF — Cross-Site Request Forgery.

ORM — Object-Relational Mapping.

SPA — Single Page Application.

Professional Completion Checklist

- HTML & semantic structure
- CSS & responsive layouts
- JavaScript fundamentals
- Async JavaScript & APIs
- Git & GitHub
- React fundamentals
- React application development
- Node.js & Express
- REST APIs
- SQL & database design
- Authentication & authorization
- Web security
- Testing & debugging
- Performance & accessibility
- Deployment & hosting
- Portfolio + GitHub README
- 5 mini projects completed
- 10 practical labs completed
- Final capstone deployed/documentated

Course completion standard: learners should be able to explain the concepts, build working projects, troubleshoot common issues, and present a deployed full-stack application with documentation.