Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

AI-Assisted Development with JavaScript and TypeScript

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- Professional JavaScript or TypeScript development experience
- Working knowledge of Git and GitHub (issues, pull requests, branches, reviews)
- Comfort with the command line and the Node.js / npm ecosystem
- No prior AI tool experience required

Target Audience

Professional JavaScript and TypeScript developers — front end, back end, or full stack — who want a structured, end-to-end path from AI-curious to AI-proficient. Ideal for engineers and teams adopting AI coding tools across the full development lifecycle — writing, refactoring, testing, reviewing, and shipping JS/TS code — who also need to understand the security, licensing, and governance implications of doing so responsibly. This is a tool-agnostic course: students work hands-on with Claude Code, GitHub Copilot, Codex, Cursor, Windsurf, and Gemini CLI rather than betting on a single vendor, and the skills transfer to whichever assistant your organization adopts.

Description

No language ecosystem has more AI-generated code flowing through it than JavaScript and TypeScript — and none has more ways for that code to go subtly wrong, from loose typing to hallucinated npm packages to framework APIs that changed last quarter. This five-day intensive teaches professional JS/TS developers to get the productivity without the mess. Days one and two cover the foundations: how large language models generate code, the strengths and tradeoffs of the frontier models, and prompting and context engineering techniques that produce strictly-typed, modern TypeScript rather than careless `any`-riddled output. Days two and three move into the tools: inline assistance and chat-driven editing in the IDE, then full agentic workflows across Claude Code, GitHub Copilot, Codex, Cursor, Windsurf, and

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Gemini CLI — plan-then-execute sessions where the agent runs the type checker, linter, and test suite; grounding agents with project context files; and issue-to-PR automation. Day four applies AI to quality: generating and maintaining Vitest suites, component and end-to-end testing with Playwright, AI-assisted debugging, code review, and refactoring legacy JavaScript toward typed TypeScript. Day five addresses team-scale adoption: npm supply-chain risks, secret handling, licensing and IP questions, policy design, and honest productivity measurement. Students finish with a capstone that exercises the full workflow on a realistic TypeScript codebase.

Learning Outcomes

- Explain how LLMs generate code and choose the right frontier model for a given JS/TS task.
- Apply prompting and context engineering techniques that produce strictly-typed, modern TypeScript.
- Use IDE assistants (Copilot, Cursor, Windsurf) effectively for completion, chat, and multi-file edits.
- Drive agentic coding sessions with Claude Code, Copilot, Codex, Cursor, Windsurf, and Gemini CLI, with tsc, ESLint, and tests in the loop.
- Ground agents in project context with CLAUDE.md / AGENTS.md, rules files, and MCP servers.
- Generate, evaluate, and maintain Vitest and Playwright test suites with AI assistance.
- Use AI for debugging, code review, and migrating legacy JavaScript to typed TypeScript.
- Recognize JS/TS-specific AI failure modes: hallucinated packages, stale framework APIs, and type looseness.
- Apply security guardrails: npm supply-chain vetting, secret handling, and prompt-injection awareness.
- Contribute to team standards for licensing, governance, and measuring AI’s real productivity impact.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Software Requirements

Node.js 24 LTS or later, a GitHub account with Copilot access, Claude Code CLI with an Anthropic API key or subscription, Codex, Cursor, Windsurf, and the Gemini CLI (trials and free tiers acceptable), VS Code, Git, and the GitHub CLI (`gh`).

Training Topics

Foundations: How AI Writes Code

- How LLMs generate code: tokens, context windows, and training data
- The frontier models for coding: Claude, GPT, and Gemini compared
- Capability tradeoffs: speed, cost, context size, and reasoning
- Where AI excels in JS/TS work and where it reliably fails

Prompting and Context Engineering for JavaScript and TypeScript

- Writing prompts that produce strict, modern TypeScript — not `any` soup
- Steering toward current ECMAScript features and away from legacy patterns
- Context strategies: monorepos, tsconfig settings, and framework conventions
- Generics, utility types, and discriminated unions in prompts
- Keeping generated code aligned with fast-moving framework APIs

IDE Assistants in Daily JS/TS Work

- GitHub Copilot: completions, chat, edits, and Next Edit Suggestions
- Cursor: Composer, rules files, and project-level context
- Windsurf: Cascade, planning, and memories
- The wider landscape: Aider, Cline, JetBrains Junie, and Amazon Kiro
- Multi-file edits and chat-driven refactoring across modules
- Component generation: steering toward your design system and conventions
- JSDoc, type declarations, and documentation generation

Agentic Coding: Claude Code, Copilot, Codex, Cursor, Windsurf, and Gemini CLI

- The agentic loop: plan, act, observe, iterate
- The agent landscape compared: Claude Code, Copilot agent mode, Codex, Cursor CLI, Windsurf Cascade, and Gemini CLI

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

- Plan mode and the supervision spectrum: permission prompts to auto mode
- CLAUDE.md and AGENTS.md: package manager, tooling, and convention context
- The verify loop: agents running tsc, ESLint, and the test suite
- Issue-to-PR workflows: assigning work to cloud agents
- MCP servers: connecting agents to internal tools and data

AI-Driven Testing and Quality

- Generating Vitest suites: mocks, fixtures, and coverage gaps
- Component and end-to-end testing with Playwright and AI assistance
- AI-assisted debugging: stack traces, source maps, and async failures
- Agentic code review and pull request automation
- Migrating legacy JavaScript to TypeScript incrementally with agents

JS/TS-Specific Pitfalls and Verification

- Hallucinated npm packages and typosquatting risk
- Stale framework knowledge: verifying against current docs
- Type looseness in generated code: strict mode and lint rules as gates
- Bundle-size and performance review of generated code
- ESLint, Biome, and Prettier as quality gates

Security, Licensing, and Governance

- Prompt injection and untrusted content in agent workflows
- Secret leakage: env files, client-side exposure, and context hygiene
- npm supply-chain vetting: audit tooling and lockfile discipline
- Licensing and IP status of AI-generated code
- Team policy design: review requirements and audit trails
- Measuring real productivity impact honestly

Capstone Workshop

- Plan and execute a multi-step feature on a realistic TypeScript codebase with the coding agent of your choice
- Generate and harden a Vitest suite for an untested module
- Issue-to-PR exercise: agent-ready issue through reviewed merge
- Security review of an AI-generated changeset including dependency audit
- Q&A session