Over 35 Years Of Technology Training

Document Generated: 04/28/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Intermediate

Course Duration: 3 Days

Next Course Date: June 8, 2026

Building AI-Powered Web Apps with JavaScript (TTAI4500)

About This Course:

Stop sending user data to servers, and start running AI on your users' devices.

This course teaches you to build the next generation of web applications: AI-powered, privacy-first, and lightning-fast. You'll learn to deploy machine learning models directly in the browser using JavaScript, eliminating the need for expensive Python backends and cloud GPU infrastructure.

Master real-time computer vision with MediaPipe, integrate large language models with Transformers.js, build autonomous AI agents with LangGraph, and optimize performance with WebGPU, all in JavaScript! You'll create applications that work offline, process sensitive data locally, and respond instantly without network delays.

Build chatbots that never expose conversations, image editors that keep photos private, recommendation engines that don't track users, and voice assistants that work without internet. Learn to architect AI systems that are faster (zero latency), cheaper (no server costs), and more secure (data stays local) than traditional cloud-based ML

Course Objectives:

By the end of this course, learners will be able to:

- Evaluate use cases where client-side JavaScript AI provides advantages over server-based solutions, including privacy-sensitive applications, offline functionality, reduced infrastructure costs, and real-time interactivity
- Implement core ML functionality using TensorFlow.js for training and inference in browser and Node.js environments
- Build computer vision applications using MediaPipe, OpenCV.js, and TensorFlow.js for face, hand, and pose tracking
- Integrate natural language processing using Transformers.js and LangChain.js for sentiment analysis, summarization, and conversational AI
- Create autonomous AI agents using frameworks like OpenAI Agents SDK, LangGraph, AutoGen, and CrewAI
- Preprocess and manipulate data using Danfo.js for ML-ready datasets
- Optimize model performance using ONNX.js, WebGPU, and WebAssembly for faster inference
- Visualize AI outputs and metrics using Plotly.js, Three.js, and WebAudio API
- Design architectures that maximize the benefits of client-side AI while understanding its limitations

Audience:

This course is designed for:

- Web developers seeking to add AI capabilities to their applications
- Full-stack JavaScript developers interested in ML integration
- Front-end engineers building intelligent user experiences
- Node.js developers implementing AI-driven features
- Software engineers transitioning from Python ML to JavaScript ML ecosystems
- Product developers looking to reduce infrastructure costs through client-side AI
- Privacy-focused engineers building applications with on-device processing

Prerequisites:

Students should have JavaScript programming experience including familiarity with ES6+ syntax, asynchronous programming (Promises/async-await), and basic web development concepts (DOM manipulation, browser APIs). A foundational understanding of machine learning concepts (neural networks, training, inference) is helpful but not required.

- Introduction to JavaScript / Modern JavaScript Essentials

Course Outline:

1. Introduction to AI & ML in JavaScript: Why Choose JavaScript?

A comprehensive exploration of the JavaScript ML ecosystem and strategic reasons for choosing JavaScript over Python for AI applications. This lesson establishes the foundation for understanding when and why client-side AI provides competitive advantages.

- The JavaScript AI revolution: from Python-only to browser-native ML
- Why JavaScript for AI: The compelling advantages
- Privacy-first architecture: sensitive data never leaves the user's device
- Zero-latency inference: no network round trips for real-time applications
- Cost elimination: offload compute to billions of client devices instead of expensive GPU servers
- Offline capability: AI features work without internet connectivity
- Universal deployment: reach any device with a browser, no app store required
- Seamless integration: add intelligence directly into existing web applications
- Developer accessibility: leverage existing JavaScript expertise
- Real-world use cases where JavaScript AI excels
- Face filters and AR effects (privacy + real-time)

- Medical image analysis (HIPAA compliance through local processing)
- Financial document processing (keeping sensitive data client-side)
- Accessibility features (speech-to-text, text-to-speech offline)
- Interactive creative tools (image generation, style transfer)
- Personalized recommendations without tracking
- Browser vs. Node.js environments for ML
- When NOT to use JavaScript AI: understanding the trade-offs and limitations
- Comparing JavaScript ML to Python ML workflows
- Lab: Deploy a pre-trained model in the browser and compare performance/cost to a server-based API approach

2. Core ML Libraries: Building Blocks of Intelligence

Explore the fundamental libraries that power machine learning in JavaScript, from comprehensive frameworks to lightweight neural network builders.

- TensorFlow.js: architecture, training, and inference
- ML5.js: simplified ML for creative applications
- Brain.js: lightweight neural networks for quick prototyping
- Synaptic.js: understanding neural network fundamentals
- Choosing the right library for your use case
- Lab: Build a simple neural network classifier using TensorFlow.js and compare implementation with ML5.j

3. Data Preparation with Danfo.js

Learn to wrangle, clean, and prepare data for machine learning using JavaScript's answer to Pandas.

- Introduction to DataFrames in JavaScript
- Loading and exploring datasets
- Data cleaning and transformation techniques
- Feature engineering for ML models
- Integrating preprocessed data with TensorFlow.js
- Lab: Load a CSV dataset, perform exploratory data analysis, and prepare features for model training

4. Computer Vision: Making the Web See

Implement real-time computer vision capabilities in web applications, from face tracking to object detection—all without sending images to a server.

- Privacy advantages of client-side vision: processing sensitive visual data locally
- MediaPipe: real-time face, hand, and pose detection
- OpenCV.js: traditional computer vision techniques

- TensorFlow.js pre-trained models: image classification and object detection
- Processing video streams and webcam input
- Performance considerations for real-time vision
- Use case deep-dive: building privacy-preserving AR filters and medical imaging tools
- Lab: Build a real-time hand gesture recognition system using MediaPipe that works entirely offline

5. Natural Language Processing & Large Language Models

Integrate powerful language models directly into JavaScript applications for text analysis and generation without sending user data to external APIs.

- Privacy-first NLP: processing sensitive text client-side
- Transformers.js: running Hugging Face models in the browser
- Sentiment analysis and text classification
- Text summarization and question answering
- LangChain.js: building LLM workflows
- Retrieval-Augmented Generation (RAG) for context-aware applications
- Cost analysis: client-side inference vs. API-based solutions
- Lab: Create a client-side sentiment analyzer for confidential documents and implement a privacy-preserving RAG chatbot
- using LangChain.js

6. Agent Frameworks: Autonomous AI Systems

Design and implement AI agents that can reason, use tools, and complete complex tasks autonomously, running entirely in the browser or on edge devices.

- Introduction to AI agents and autonomous systems
- OpenAI Agents SDK: multi-tool agents
- LangGraph: state-based workflows and structured reasoning
- AutoGen (JS): multi-agent collaboration patterns
- CrewAI (JS): role-based cooperative agent systems
- Designing reliable agent architectures
- Deployment advantages: distributing agent intelligence
- Lab: Build a task-planning agent using LangGraph that runs client-side and can break down complex requests into actionable steps

7. Runtime Optimization & Performance

Maximize the speed and efficiency of ML models in production JavaScript environments to deliver real-time user experiences.

- ONNX.js: cross-platform model deployment
- WebGPU: GPU-accelerated inference and training in the browser
- WebAssembly (WASM): near-native performance for ML

- Model quantization and optimization techniques
- Benchmarking and profiling ML workloads
- Choosing the right runtime for your deployment target
- Performance case studies: achieving sub-50ms inference for real-time applications
- Lab: Convert a TensorFlow model to ONNX format and compare performance across different runtimes

8. Visualization & User Interaction

Create compelling visualizations and interactive experiences that showcase AI outputs and model behavior directly in the browser.

- Plotly.js: data visualization and ML dashboards
- Three.js: 3D visualization of neural networks and AI-generated content
- WebAudio API integration: audio ML applications
- Building interactive ML demos
- Real-time metric monitoring and display
- Lab: Create an interactive ML dashboard that visualizes model training progress and displays inference results

9. Privacy, Security, and Deployment Best Practices

Understand the architectural patterns and security considerations for deploying production AI applications in JavaScript.

- Privacy by design: architectural patterns that keep data local
- Client-side data encryption and secure model storage
- Handling sensitive model weights and intellectual property
- Progressive loading strategies for large models
- Caching and service workers for offline AI
- Browser compatibility and fallback strategies
- Monitoring client-side AI performance in production
- Lab: Implement a privacy-preserving AI application with encrypted model storage and offline capability

10. Putting It All Together: Building a Complete AI Application

Apply all learned concepts to design and build a full-featured AI-powered web application that demonstrates the advantages of JavaScript-based AI.

- Architecture patterns for AI-enabled web apps
- Combining multiple AI capabilities (vision + NLP + agents)
- Cost-benefit analysis: calculating savings from client-side deployment
- Client-side privacy considerations and compliance (GDPR, HIPAA)
- Deployment strategies and best practices
- Performance monitoring and optimization in production

- Case studies: successful JavaScript AI applications in production
- Lab: Build a comprehensive AI application that integrates computer vision, NLP, and agent frameworks (e.g., an intelligent document analyzer that keeps all data local, or a privacy-first smart assistant)