Publication date : 02/20/2024

Course : WebAssembly, boosting the performance of your web applications

Putting binary in the engine of your web browsers

Practical course - 3d - 21h00 - Ref. WAY

Price : 2150 CHF E.T.

★★★★★ 5 / 5

WebAssembly (WASM), an official W3C standard since 2019, makes it possible to write ultra-fast, ultra-light applications on the web. These applications can already be written in all kinds of existing source languages: C/C++, Rust, Go, Java, etc. All they need to do is be ported for access in a browser or secure container. This hands-on training course provides the keys to developing WASM code and compiling existing programs in WebAssembly.

Teaching objectives

At the end of the training, the participant will be able to:

- Understanding the architecture and environment of the W3C WebAssembly standard
- Master the binary instruction set of the WASM language and its textual representation, the WAT format
- Using the JavaScript API to interact with WASM modules
- Set up a C/C++ compilation with the Emscripten suite
- Developing with the AssemblyScript language
- Porting a C/C++ program or library to WASM

Intended audience

Developers, web developers, integrators, technical architects, technical solutions managers.

Prerequisites

Basic knowledge of HTML, languages such as JavaScript and C, and command languages such as shell, Bash or CMD (DOS).

Course schedule

PARTICIPANTS

Developers, web developers, integrators, technical architects, technical solutions managers.

PREREQUISITES

Basic knowledge of HTML, languages such as JavaScript and C, and command languages such as shell, Bash or CMD (DOS).

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

1 Introduction to WASM

- What issues does WebAssembly address?
- History of WASM.
- Architecture.
- Portability, safety, performance.
- Organization of the specification.
- Documentation.
- WASI, Bytecode Alliance.
- Module structure.

Hands-on work

Write simple WASM modules using WAT. Compile and run with wat2wasm and node.

2 WAT text language

- Description of a WAT development environment.
- Visual Studio Code extensions.
- The different module declarations.
- Comments, S-expressions.
- Functions and the instruction stack.
- Import/export a function or other artifact.
- Global objects.
- Linear memory.
- Pointer tables.
- The different instructions: loop, conditions, operations, trap.
- Start function "start".
- The JavaScript interface for using a WASM module.

Hands-on work

Write and compile a module in WAT offering some basic mathematical functions (factorial, Fibonacci, etc.). Run the file in Node and in a browser.

3 Runtimes WASM

- Runtime requirements.
- List of runtimes.
- Description of WASI.
- Runtime installation.
- Run WASM programs with runtimes.

Hands-on work

Compile a simple program written in Rust in WASM and run it on several runtimes (wasm3, wasmtime, etc.).

4 AssemblyScript

- Installation of the Node AssemblyScript module.
- Initiating a project with asinit.
- Garbage collector and memory.
- Programming with objects.
- Integration of a WASM library built in AssemblyScript.

Hands-on work

WASM module written in AssemblyScript, calculating the colors of the points of a Mandelbrot fractal, and integration of this library into a front end visualizing the fractal.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

5 The Emscripten tool

- Languages that can be ported to WASM.
- General presentation of Emscripten.
- Historical background.
- Official installation.
- Installation under Debian/Ubuntu with apt.
- The emcc compiler.
- The JavaScript envelope file.
- Compilation options.
- JavaScript call strategies (ccall, cwrap, etc.).

Hands-on work
Write a simple program in C, compile it in WASM and use it with Node and in a browser.

6 Bookstore portage

- Compile and configure with Emscripten and Autoconf.
- Compile and configure with Emmake and Emconfigure.
- Interaction with makefiles.
- MODULARIZE, EXPORTED_FUNCTIONS, EXPORTED_RUNTIME_METHODS compilation options.
- The virtual file system.
- Environment variables.

Dates and locations

REMOTE CLASS
2026 : 13 Apr., 24 June, 16 Nov.