

**Introduction to WebAssembly**

**Course Number:** SCRPT-186  
**Duration:** 3 days

**Overview**

This Introduction to WebAssembly training gives attendees a comprehensive overview of the present and future of WebAssembly and prepares them to embrace the promise of standards-based, safe, fast, and portable code. Students learn how WebAssembly solves many of the problems facing client-side web development, while still allowing freedom of technology choice.

**Prerequisites**

Familiarity with modern HTML, CSS, and JavaScript development is presumed.

**Materials**

All WebAssembly training students receive comprehensive courseware.

**Software Needed on Each Student PC**

Software for each delivery is determined based on the client’s preferred environment and choice of either Rust or C/C++ as the development language for class.

**Objectives**

* Understand the purpose and consequences of WebAssembly's design
* Understand how it differs from other attempts to create safe, fast and portable code
* Leverage the elements of the Minimum Viable Product (MVP) WebAssembly platform
* Navigate the structure and elements of WebAssembly modules
* Use various tools in the WebAssembly ecosystem
* Understand the Wast text syntax
* Convert between WASM and WAST
* Leverage the JavaScript API
* Expose and call functions from WebAssembly
* Share memory between modules
* Provide dynamic linking between modules
* Convert legacy code to WebAssembly
* Work with C/C++
* Work with Rust
* Work with AssemblyScript
* Work inside and outside the browser
* Understand how WASI provides portability and security for applications
* Understand how the WebAssembly ecosystem is evolving

**Outline**

* Introduction
* History of safe, fast, portable initiatives
* NaCL/PNaCL
* Asm.js
* Path to WebAssembly Standards
* Minimum Viable Product
  + An overview of the version supported by all major browsers and Node
* WAST and WASM
  + WebAssembly formats
  + Converting between the formats
* WebAssembly Modules
  + Structural elements
* WebAssembly Memory
  + Reading and Writing to Shared Memory
  + Initializing and Growing Memory
* WebAssembly Tables
  + Invoking methods dynamically
* Working with C/C++
* Porting Legacy Code
* Working with Rust
* WebAssembly in the Server
  + Working with Node and Deno
* Working with AssemblyScript
* WebAssembly System Interface (WASI)
  + Introduction
  + Portability
  + Security
  + WAPM
* WebAssembly Green Field Projects
* The Future
  + Emerging topics
* Conclusion