

**Introduction to the Zig Programming Language**

**Course Number:** PROG-116
**Duration:** 3 days

**Overview**

[Zig](https://ziglang.org/) is a general-purpose programming language and toolchain for creating robust, optimal, and reusable software. Zig improves on C with its customized memory control, null reference protection, and required error handling. Unlike Rust, Zig allows low-level memory control with syntax and memory model features to help avoid memory leaks. Finally, it provides interoperability with existing C libraries.

This Zig Programming training course teaches attendees how to leverage Zig's data types, control flow structures, code organization, memory management, and more.

**Prerequisites**

All students must have programming experience.

**Materials**

All Zig Programming training students receive comprehensive courseware.

**Software Needed on Each Student PC**

A complete, remote virtual environment is provided for training and is accessible via the Internet from any modern web browser.

**Objectives**

* Build and run Zig programs
* Explore features unique to Zig that set it apart from other languages
* Understand where Zig is a good choice for writing software
* Work with the Zig toolchain
* Explore how Zig can be used as a drop-in replacement for C
* Discover how Zig enables performance and safety
* Apply modern techniques for memory control, null reference handling, and error handling with a lower-level language.

**Outline**

* Introduction
	+ What is Zig?
	+ What Problems Does Zig solve?
	+ Zig Compared to C
	+ Zig Compared to Other Languages
	+ Zig Zen
* Getting Started
	+ Zig Toolchain
	+ Hello, Zig!
	+ Code and Debug with VSCode
	+ Zig Standard Library
	+ Zig Source Files
	+ Cross Compilation
* Language Features in Hello Zig
	+ Importing from the Standard Library
	+ Constants
	+ Define a Public “main” Function
	+ Try Statement
	+ Error Union Types
	+ String Interpolation
	+ Comments
* Zig Project Scaffolding
	+ Create a New Executable Project
	+ Create a New Library Project
	+ Build and Run
	+ Build and Test
* Console Apps
	+ Print Output to the Terminal
	+ Format Specifiers
	+ Anonymous Struct Literals
	+ Capture Input from the Terminal
	+ String Comparison
	+ While Loop
	+ Error Handling
* Data Types
	+ Integers
	+ Floats
	+ Arrays
	+ Pointers
	+ Slices
* Data Structures
	+ Struct
	+ Enum
	+ Union
* Variables
	+ Variable Name Rules
	+ Container Level Variables
	+ Compile-Time vs. Run-Time Variables
	+ Local Variables
* Control Flow
	+ Expressions and Operators
	+ While/For Loops
	+ Break/Continue Statements
	+ If Statement
	+ Switch Statement
	+ Try/Catch Statement
	+ Defer/ErrDefer Statement
* Functions
	+ What is a Function?
	+ Define a Function
	+ Call a Function
	+ Pass Parameters to a Function
	+ Immutable vs Mutable Parameters
	+ Importing Functions from Other Zig Code Files
* Strings
	+ UTF-8 Data Type
	+ Character Arrays
	+ Buffers
	+ Print Formatted Strings
	+ Capture Strings Console Input
	+ String Copy
	+ String Comparison
* Memory Control
	+ Memory Allocation Philosophy
	+ Memory Control vs. Memory Safety
	+ Choosing an Allocators
	+ Heap Allocation Failure
	+ Lifetime and Ownership
	+ Optional and Optional Pointers
	+ Null References
* Program a Zig “Object”
	+ Compared to C/C++/Python/JavaScript
	+ Anonymous Structs
	+ Anonymous Struct Literals
	+ Data Fields
	+ Constant Fields
	+ Error Enums
	+ Function Members
	+ Function Patterns
	+ Dynamic Memory Allocation
* Testing
	+ Zig’s built-in testing
	+ What can be tested?
	+ Assert Output with Expect
* Conclusion