

[Cross-Chain Development](#) >[Finance](#) >[Deploy with Fleek](#) >

Ctrl K

[GitHub](#)[Support](#)[Base Build](#)[Get Started](#)[Base Chain](#)[Base Account](#)[Base App](#)[Mini Apps](#)[OnchainKit](#)[Cookbook](#)[Showcase](#)[Learn](#)[Overview](#)[Anatomy of a Smart Contract](#)[Introduction to Solidity](#) >[Contracts and Basic Functions](#) ▾[Introduction to Contracts](#)[Hello World Guide](#)[Basic Types](#)[Basic Functions Exercise](#)[Deploying to a Testnet](#) ▾[Introduction to Solidity](#)

Overview

An overview of this module.

 [Copy page](#) ▾

...ercises found in [this](#) for [this](#) and contains background information, guided coding practices, and independent exercises.

This program is **not** suitable for people who are new to programming in general. While the explanations are thorough, they often rely on an expectation that you are familiar with the underlying concepts. We will not teach you what arrays are and how they are used, but we will show you how they work in this environment.

Prerequisites

Before these lessons, you should:

Have several years of experience as a programmer in an object-oriented language

Be familiar with the uses and properties of the Ethereum blockchain and the EVM

Ideally, be familiar with at least one curly-bracket programming language

Ask a question...

Ctrl+I

Objectives

By the end of this module, you should be able to:

Introduction to Solidity

Describe why languages like Solidity are used to write smart contracts

Relate an overview of the history (and pace of change) of Solidity and its strengths and weaknesses

Deploy and test the Storage.sol demo contract in Remix

Contracts and Basic Functions

Construct a simple "Hello World" contract

Categorize basic data types

List the major differences between data types in Solidity as compared to other languages

Compare and contrast signed and unsigned integers

Smart Contract Development

Overview

Anatomy of a Smart Contract

Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

Write a pure function that accepts argument and returns a value

Deploying Smart Contracts to a Testnet

Describe the uses and properties of the Ethereum testnet

Compare and contrast Ropsten, Rinkeby, Goerli, and Sepolia

Deploy a contract to the Sepolia testnet and interact with it in Etherscan

Control Structures

Control code flow with if, else, while, and for

List the unique constraints for control flow in Solidity

Storage in Solidity

Diagram how a contract's data is stored on the blockchain (Contract → Blockchain)

Order variable declarations to use storage efficiently

Diagram how variables in a contract are stored (Variable → Contract)

Arrays in Solidity

Construct then store and retrieve values in storage and memory arrays

Describe the difference between storage and memory arrays

Diagram how arrays are stored

Write a function that can return a filtered subset of an array

The Mapping Type

Smart Contract Development

Overview

Anatomy of a Smart Contract

Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

Smart Contract Development

Overview

Anatomy of a Smart Contract

Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

Construct a Map (dictionary) data type

Diagram the storage of the Mapping data type

Recall that assignment of the Map data type is not as flexible as for other data types/in other languages

Restrict function calls with the `msg.sender` global variable

Recall that there is no collision protection in the EVM and why this (probably) ok

Advanced Functions

Describe how pure and view functions are different than functions that modify storage

Categorize functions as public, private, internal, or external based on their usage

Use modifiers to efficiently add functionality to multiple functions

Utilize `require` to write a function that can only be used when a variable is set to 'True'

Structs

Construct a struct (user-defined type) that contains several different data types

Declare members of the struct to maximize storage efficiency

Describe constraints related to assignment of structs depending on the types they contain

Smart Contract Development**Overview**

Anatomy of a Smart Contract

Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

Inheritance

Write a smart contract that inherits from another contract

Imports

Import and use code from another file

Errors

Debug common solidity errors including execution reverted, out of gas, stack overflow, value overflow/underflow, index out of range, and so on

New Keyword

Write a contract that creates a new contract with the new keyword

Was this page helpful?

 Yes No Suggest edits Raise issue[◀ Deploy an Onchain App with Fleek](#)[Anatomy of a Smart Contract ▶](#)

base.org

Blog

Privacy Policy

Terms of Service

Cookie Policy

Smart Contract Development



[Overview](#)

Anatomy of a Smart Contract



Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

