

EXERCISE

Storage in Solidity ▾

Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise

Arrays in Solidity ▾

Arrays

Writing Arrays

Arrays Guide

Filtering Arrays

Fixed Size Arrays

Array Storage Layout

Exercise

The Mapping Type ▸

Advanced Functions ▸

Structs ▸

Inheritance ▸

Arrays in Solidity

Arrays Exercise

 Copy page ▾

Exercise - Demonstrate your knowledge of arrays.

Create a contract that adheres to the following specifications.

Contract

Review the contract in the starter snippet called `ArraysExercise`. It contains an array called `numbers` that is initialized with the numbers 1–10. Copy and paste this into your file.

```
contract ArraysExercise {  
    uint[] public numbers = [1,2,3,4,5,6,7,8,9,10];  
}
```



EXERCISE



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Arrays

Writing Arrays

Arrays Guide

Filtering Arrays

Fixed Size Arrays

Array Storage Layout

Exercise

Add the following functions:

Return a Complete Array

The compiler automatically adds a getter for individual elements in the array, but it does not automatically provide functionality to retrieve the entire array.

Write a function called `getNumbers` that returns the entire `numbers` array.

Reset Numbers

Write a `public` function called `resetNumbers` that resets the `numbers` array to its initial value, holding the numbers from 1-10.



We'll award the pin for any solution that works, but one that **doesn't** use `.push()` is more gas-efficient!



Remember, *anyone* can call a `public` function! You'll learn how to protect functionality in another lesson.

Append to an Existing Array

Write a function called `appendToNumbers` that takes a `uint[] calldata` array called `_toAppend`, and adds that array to the `storage` array called `numbers`, already

EXERCISE



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Arrays

Writing Arrays

Arrays Guide

Filtering Arrays

Fixed Size Arrays

Array Storage Layout

Exercise

present in the starter.

Timestamp Saving

At the contract level, add an `address` array called `senders` and a `uint` array called `timestamps` .

Write a function called `saveTimestamp` that takes a `uint` called `_unixTimestamp` as an argument. When called, it should add the address of the caller to the end of `senders` and the `_unixTimestamp` to `timestamps` .



You'll need to research on your own to discover the correct *Special Variables and Functions* that can help you with this challenge!

Timestamp Filtering

Write a function called `afterY2K` that takes no arguments. When called, it should return two arrays.

The first should return all timestamps that are more recent than January 1, 2000, 12:00am. To save you a click, the Unix timestamp for this date and time is `946702800` .

The second should return a list of `senders` addresses corresponding to those timestamps.

Resets



Q Search...

Ctrl K

GitHub

Support

Base Build



Get Started

Base Chain

Base Account

Base App

Mini Apps

OnchainKit

Cookbook

Showcase

Learn

How Storage Works

Storage Overview

Exercise



Arrays

Writing Arrays

Arrays Guide

Filtering Arrays

Fixed Size Arrays

Array Storage Layout

Exercise

Submit your Contract and Earn an NFT Badge! (BETA)

! Hey, where'd my NFT go!?

Testnets are not permanent! Base Goerli will soon be sunset, in favor of Base Sepolia.

As these are separate networks with separate data, your NFTs **will not** transfer over.

Don't worry! We've captured the addresses of all NFT owners on Base Goerli and will include them when we release the mechanism to transfer these NFTs to mainnet later this year! You can also redeploy on Sepolia and resubmit if you'd like!

EXERCISE



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Arrays **base docs**

Writing Arrays

Arrays Guide

Filtering Arrays

Fixed Size Arrays

Array Storage Layout

Exercise

Was this page helpful?



Yes



No



Suggest edits



Raise issue

← **Array Storage Layout**

Mappings Overview →

base.org

Blog

Privacy Policy

Terms of Service

Cookie Policy