marp
true

Arrays

- Arrays are our first reference type! 🎊
- Arrays act quite differently in storage vs memory/calldata 💿
- Arrays aren't used as frequently as mappings 😢
  - useful for ordered data or when you need iteration 🔢
  - unlimited size plus iteration can be a DOS vector ⛽

Reference Types

- Reference Types: `string` , `bytes` , `arrays` , `mappings` , and `structs`
- As an argument you must declare the memory location: `calldata` , `memory` or `storage`
- *Potentially* passed by reference, as opposed to value types

Let's take a look at the data locations!

Storage

In storage, arrays can be dynamic size or fixed size:

```
contract X {
  uint[3] favoriteNumbers;
  uint[] allNumbers;

  constructor() {
    // push is allowed on dyamic arrays
    allNumbers.push(1);

    // not allowed on fixed size arrays
    favoriteNumbers[0] = 1;
  }
}
```

Storage

For reference types, they can be passed by reference as a storage pointer:

```
import "forge-std/console.sol";
contract X {
  uint[3] favoriteNumbers;

  constructor() {
    modifyArray(favoriteNumbers);

    console.log(favoriteNumbers[0]); // 22
  }

  function modifyArray(uint[3] storage nums) private {
    nums[0] = 22;
  }
}
```

Calldata

Refers to the data passed into the function, read-only:

```
import "forge-std/console.sol";
contract X {
  function readArr(uint[3] calldata arr) external view {
    // cannot write to the array
    console.log(arr[0]);
  }
}
```

Memory

Temporary location, creates a copy of the reference type passed in:

```
import "forge-std/console.sol";
contract X {
  function readArr(uint[3] memory arr) external view {
    arr[0] = 5;
    console.log(arr[0]); // 5
  }
}
```