Smart Contracts

- Solidity offers many of the data structures that programmers are accustomed to from turing complete programming languages

- Smart contracts such as tokens with user bases larger than 100k are able to store relevant user info entirely on chain

Smart Contracts (memory and storage)

- Data Location: memory and storage

- There is always a default, however that can be overridden by appending either memory or storage to the type

Smart Contract (memory and storage)

- memory: the default data type for function parameters

- storage: the default for local variables and state variables

Smart Contract (memory and storage)

```
pragma solidity ^0.4.0;

contract C {
    uint[] x; // the data location of x is storage

    // the data location of memoryArray is memory
    function f(uint[] memoryArray) public {
        x = memoryArray; // works, copies the whole array to storage
        var y = x; // works, assigns a pointer, data location of y is storage
        y[7]; // fine, returns the 8th element
        y.length = 2; // fine, modifies x through y
        delete x; // fine, clears the array, also modifies y
        // The following does not work; it would need to create a new temporary /
        // unnamed array in storage, but storage is "statically" allocated:
        // y = memoryArray;
        // This does not work either, since it would "reset" the pointer, but there
        // is no sensible location it could point to.
        // delete y;
        g(x); // calls g, handing over a reference to x
        h(x); // calls h and creates an independent, temporary copy in memory
    }

    function g(uint[] storage storageArray) internal {}
    function h(uint[] memoryArray) public {}
}
```

Smart Contracts (memory and storage)

- Medium article that shows use of memory and storage in solidity:

https://medium.com/coinmonks/storage-vs-memory-in-solidity-8251847186fd