



# Solidity

# Solidity

- Looks familiar to JavaScript
  - Behaves very differently!
  - Contracts look similar to classes
  - Functions are public by default
  - Static typing
- 
- Solidity source code should start with

```
pragma solidity ^0.5.3;
```
  - Good resource  
<https://solidity.readthedocs.io/en/develop/solidity-in-depth.html>

# Smart Contract Lifecycle

Write code in high-level language

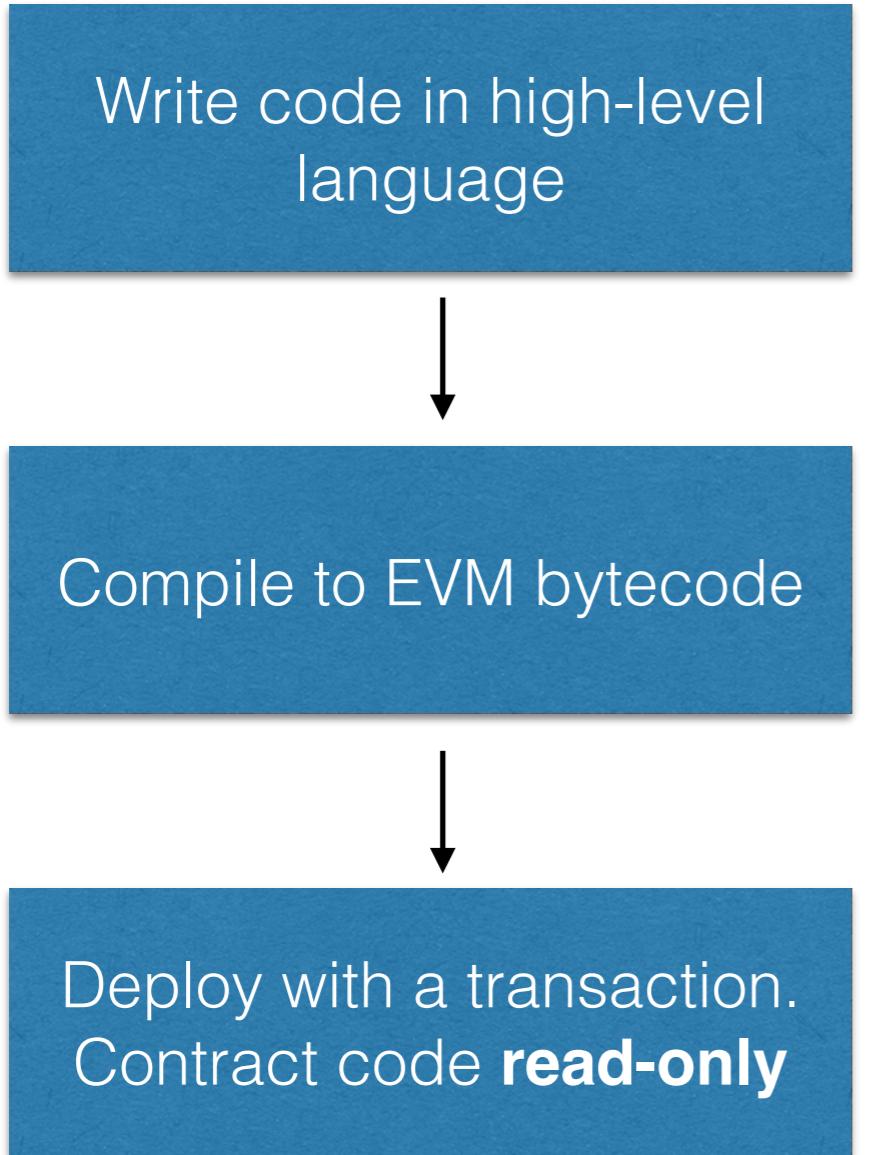
# Smart Contract Lifecycle

Write code in high-level language

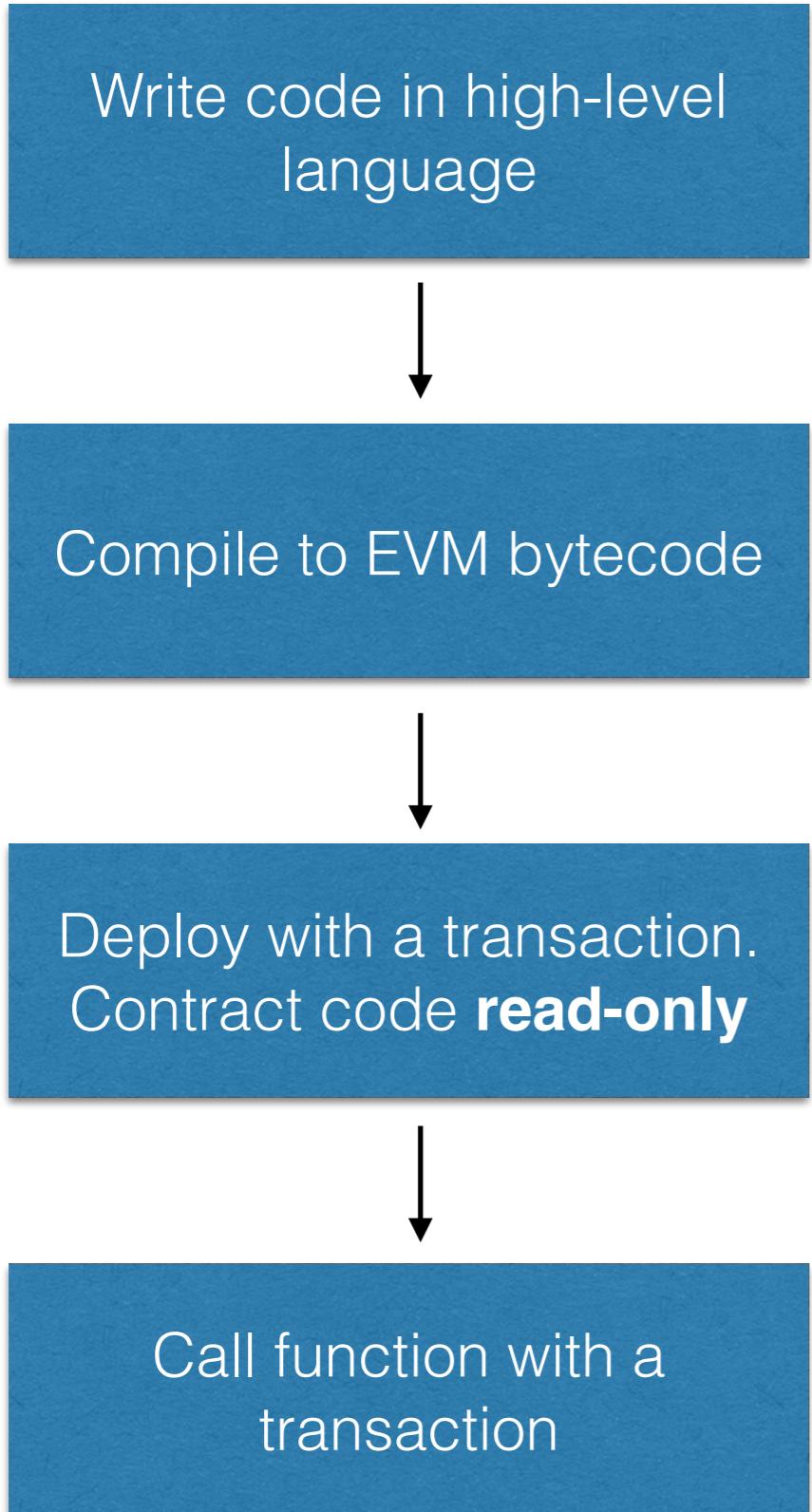


Compile to EVM bytecode

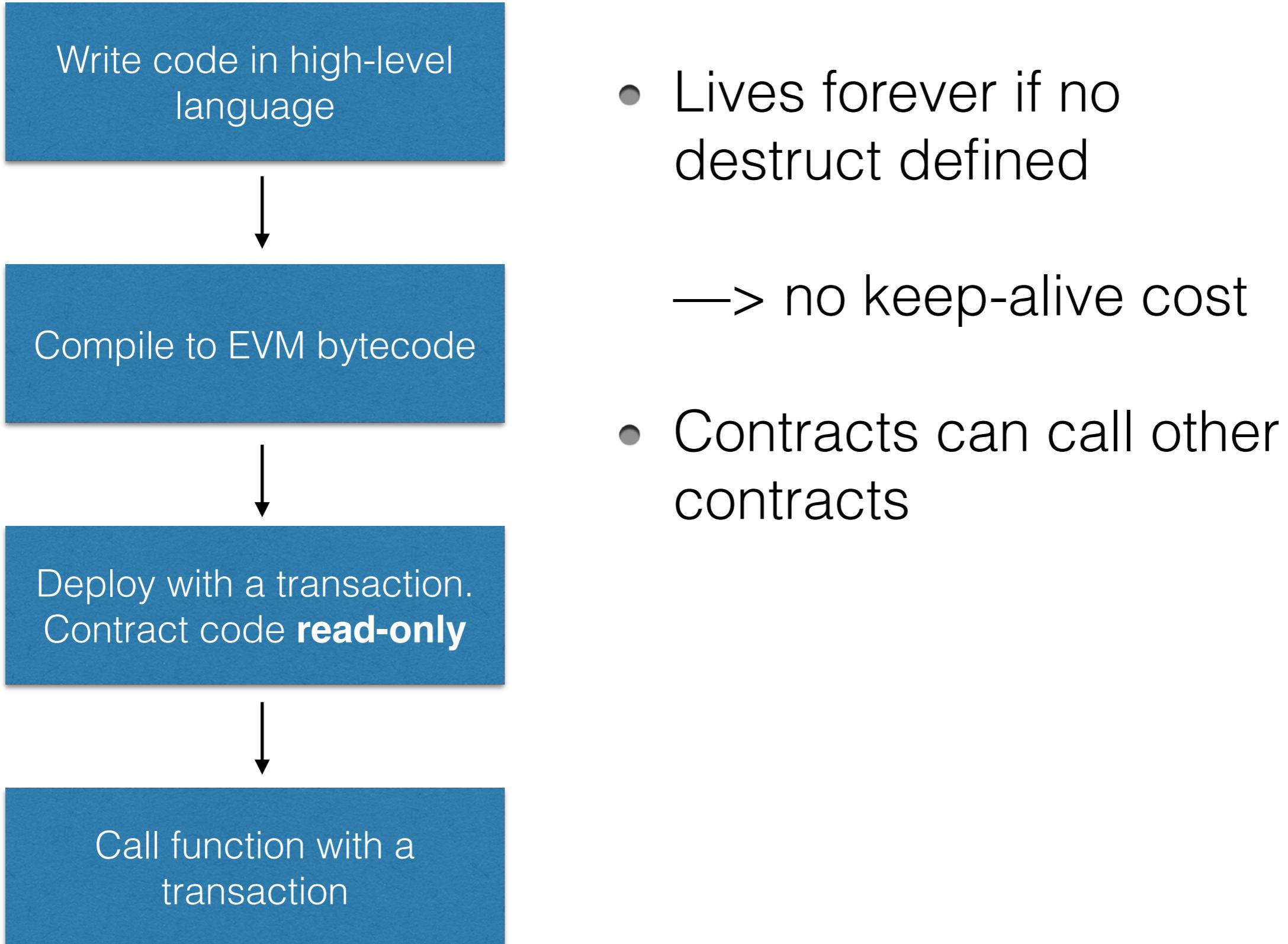
# Smart Contract Lifecycle



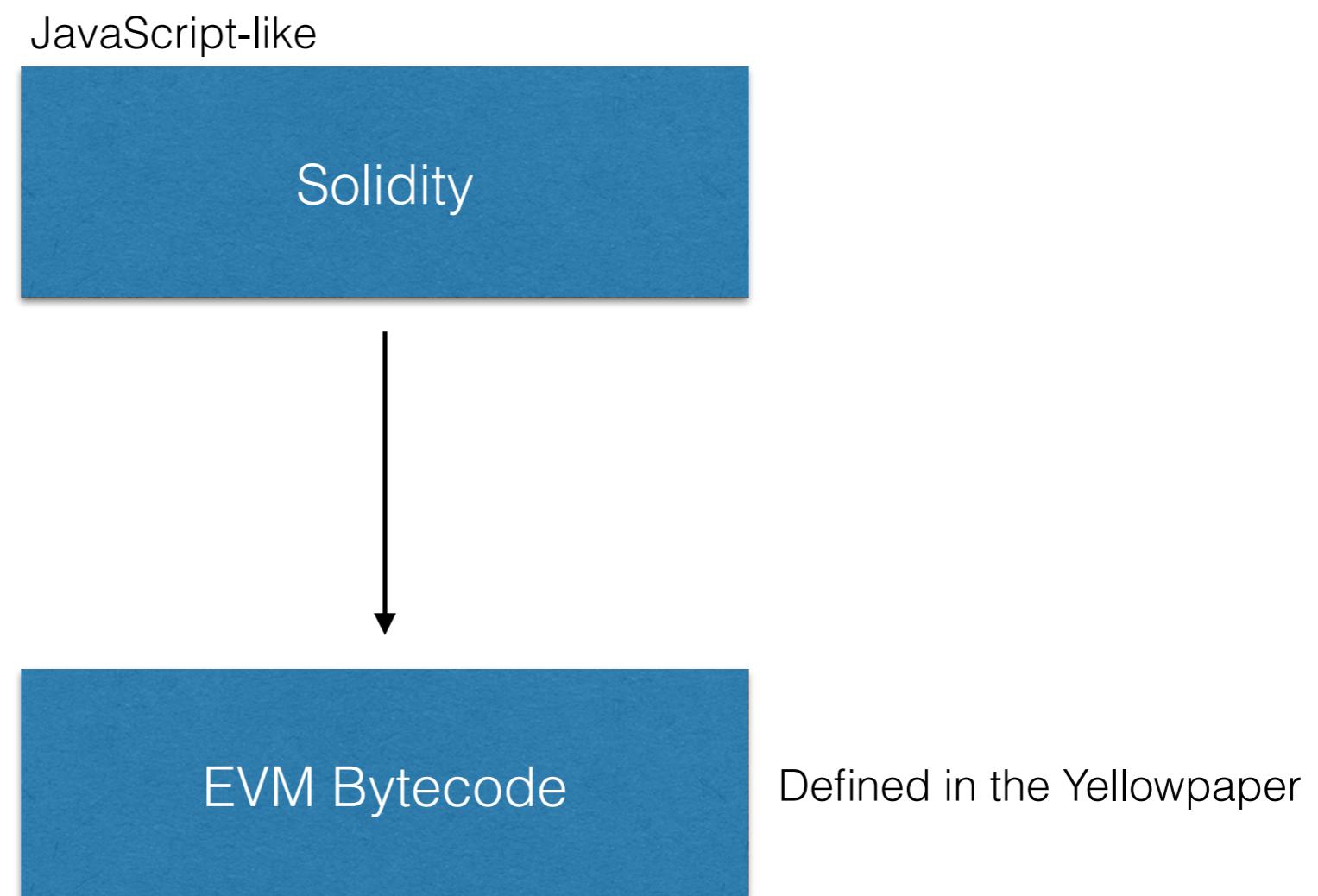
# Smart Contract Lifecycle



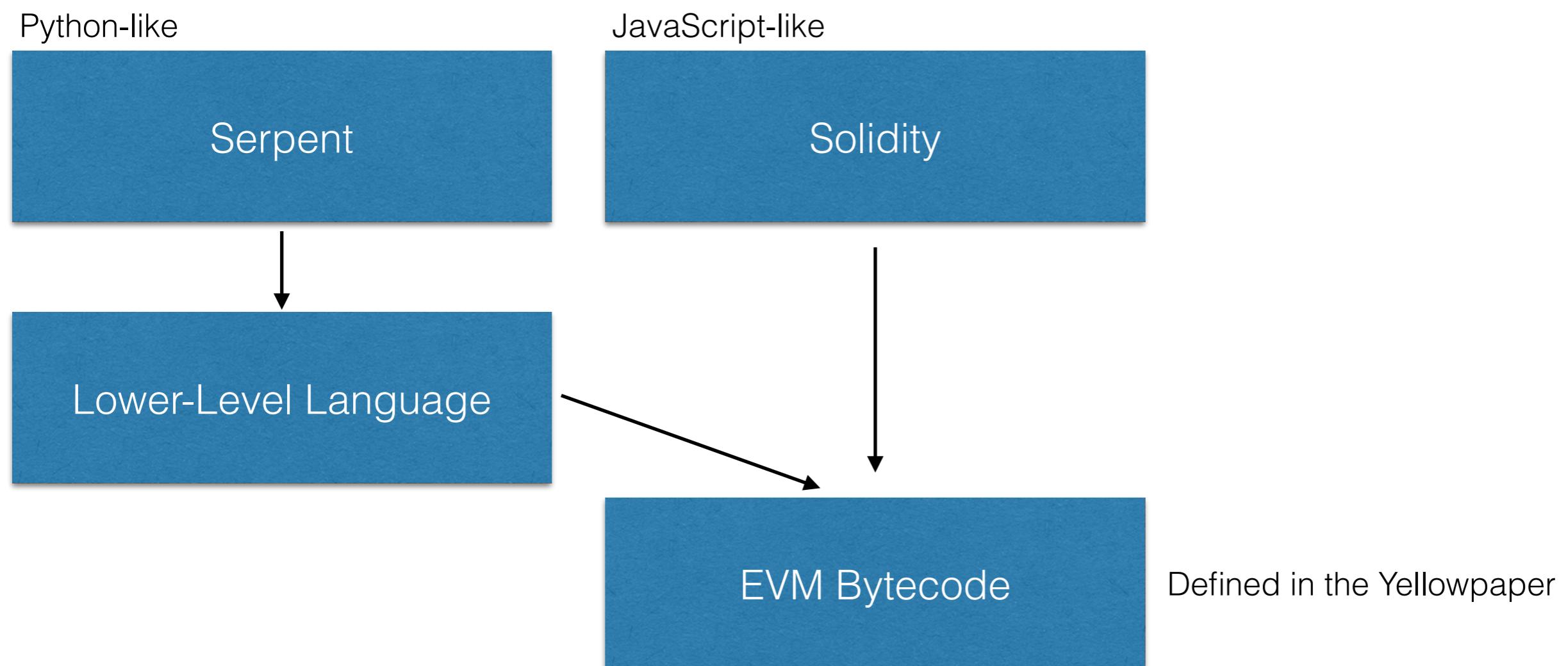
# Smart Contract Lifecycle



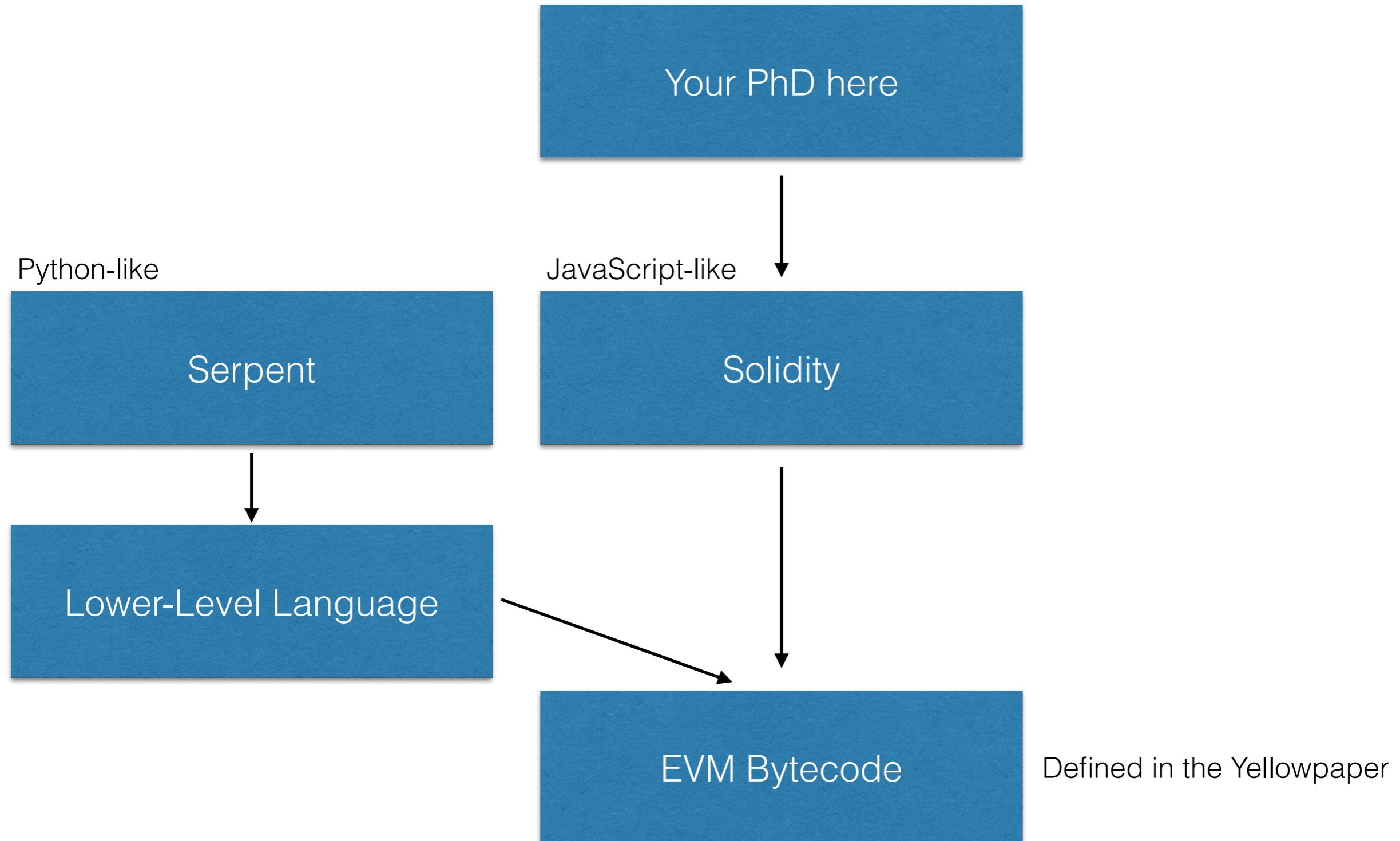
# Smart Contract Development



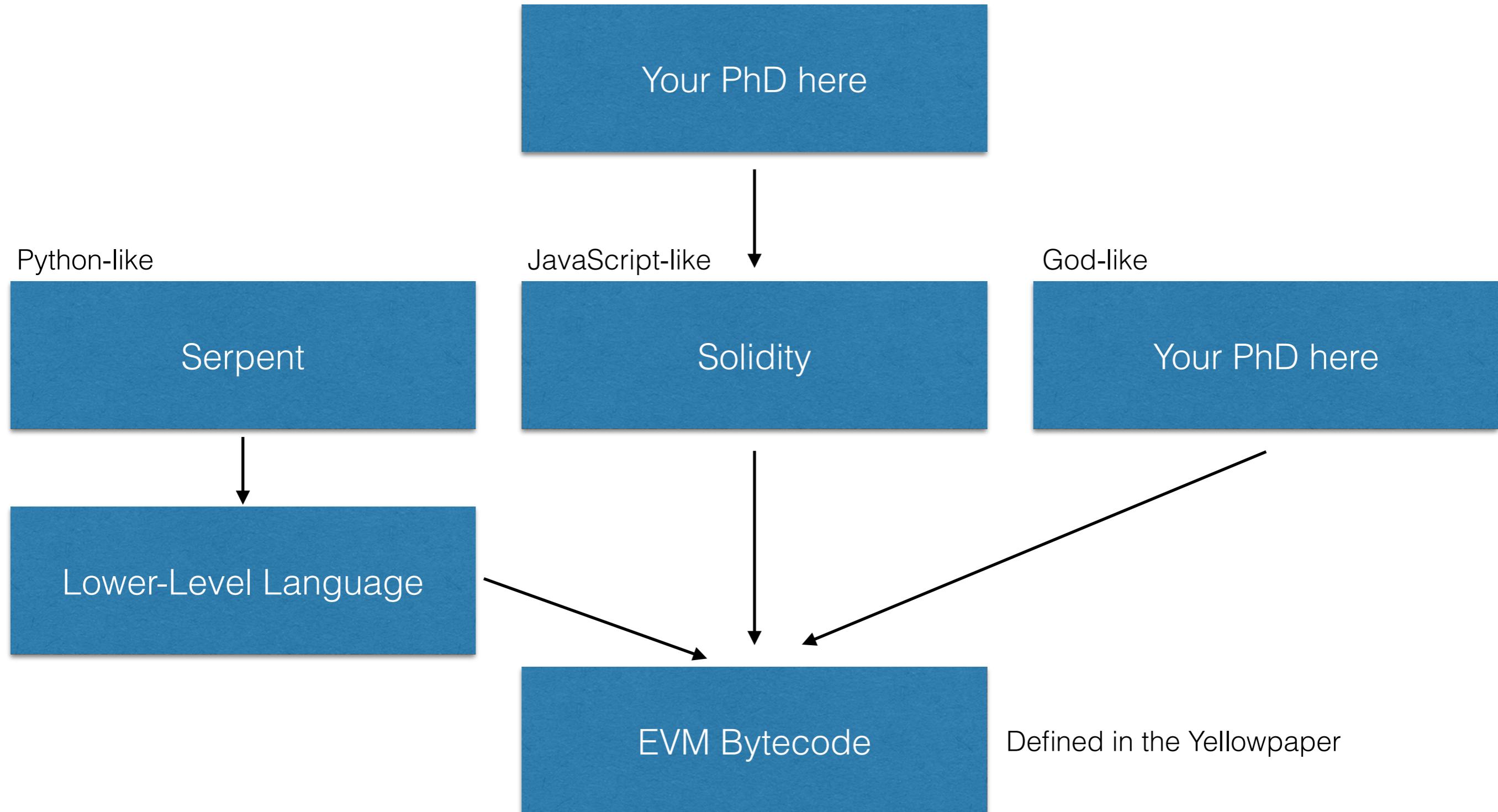
# Smart Contract Development



# Smart Contract Development



# Smart Contract Development



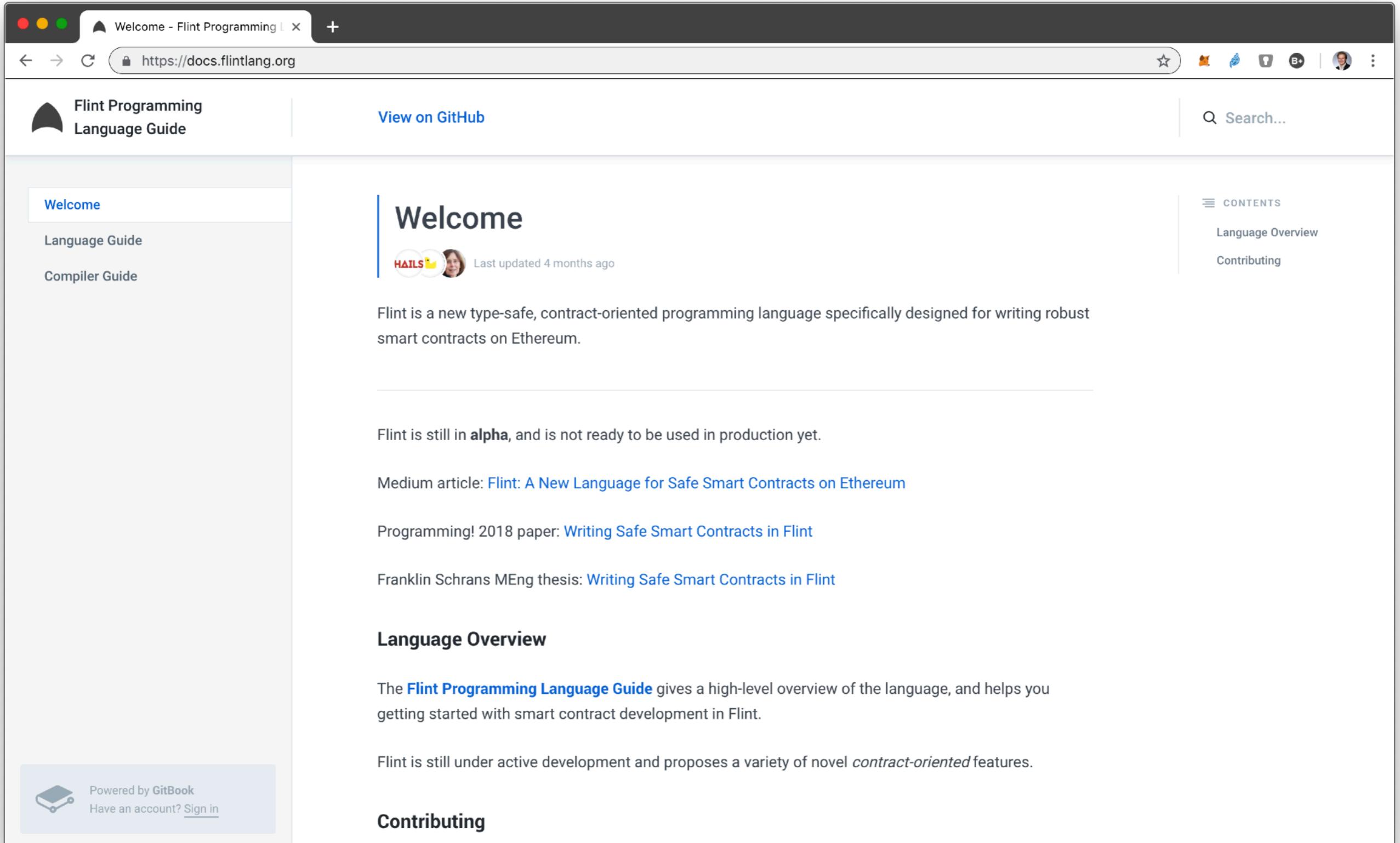
# ETHFiddle

The screenshot shows the EthFiddle web application interface. At the top, there's a header bar with the title "EthFiddle - Solidity IDE in the Browser" and a "Compile" button. Below the header is a navigation bar with links for "EthFiddle", "Security Audit", "Share", and "Login". To the right of the navigation bar are "Auto Compile" and "Solidity 0.4.18" dropdown menus. The main content area contains a code editor with the following Solidity code:

```
1 //Write your own contracts here. Currently compiles using solc v0.4.15+commit.bbb8ef6
2 pragma solidity ^0.4.18;
3 contract SimpleStore {
4     function set(uint _value) public {
5         value = _value;
6     }
7
8     function get() public constant returns (uint) {
9         return value;
10    }
11
12    uint value;
13 }
```

Below the code editor, there are two tabs: "Compiler Errors" (which is selected) and "Compiled output". At the bottom of the page, there are links for "Powered By Loom Network", "Recent Fiddles", and a blue "OPEN CHAT" button.

# Flint - a language to replace Solidity



The screenshot shows a web browser window displaying the Flint Programming Language Guide at <https://docs.flintlang.org>. The page has a dark header with a logo, a search bar, and navigation links. The main content area features a "Welcome" section with a bio for HAILS and a summary of the language's purpose. It also includes links to Medium articles, a paper, and a thesis. A "Language Overview" section is present, along with a "Contributing" section and a sidebar for the "Compiler Guide". The footer indicates the site is powered by GitBook.

Welcome - Flint Programming [+](#)

<https://docs.flintlang.org>

Flint Programming Language Guide

View on GitHub

Search...

Welcome

Language Guide

Compiler Guide

CONTENTS

Language Overview

Contributing

## Welcome

HAILS Last updated 4 months ago

Flint is a new type-safe, contract-oriented programming language specifically designed for writing robust smart contracts on Ethereum.

Flint is still in **alpha**, and is not ready to be used in production yet.

Medium article: [Flint: A New Language for Safe Smart Contracts on Ethereum](#)

Programming! 2018 paper: [Writing Safe Smart Contracts in Flint](#)

Franklin Schrans MEng thesis: [Writing Safe Smart Contracts in Flint](#)

## Language Overview

The [Flint Programming Language Guide](#) gives a high-level overview of the language, and helps you getting started with smart contract development in Flint.

Flint is still under active development and proposes a variety of novel *contract-oriented* features.

## Contributing

Powered by [GitBook](#)  
Have an account? [Sign in](#)

## Solidity - Types

bool, uint8, uint16, ... uint256, int8, ... int256

address

string

byte[]

mapping(keyType ==> valueType)

- State variables reside in storage

## Solidity - Throw

```
uint8 numCandidates;
uint32 votingFee;
mapping(address => bool) hasVoted;
mapping(uint8 => uint32) numVotes;

// Cast a vote for a designated candidate
function castVote(uint8 candidate) {
    if (msg.value < votingFee)
        return;
    if (hasVoted[msg.sender])
        throw;

        hasVoted[msg.sender] = true;
        numVotes[candidate] += 1;
}
```

- Throw makes sure that only gas is consumed.

## Solidity - State Variables

```
pragma solidity ^0.5.3;

contract SimpleStorage {
    uint storedData; // State variable
    // ...
}
```

- State variables reside in storage

## Solidity - Functions

```
pragma solidity ^0.5.3;

contract SimpleAuction {
    function bid() public payable { // Function
        // ...
    }
}
```

- Internal/External function calls
- Visibility
  - External (can be called from other contracts)
  - Public (like external, can also be called internally)
  - Internal (can only be called internally)
  - Private (not visible in derived contracts)

## Solidity - Modifiers

```
contract Purchase {  
    address public seller;  
  
    modifier onlySeller() { // Modifier  
        require(msg.sender == seller);  
        _;  
    }  
  
    function abort() public onlySeller { // Mod. usage  
        // ...  
    }  
}
```

- Amend the semantics of functions

## Solidity - Events

```
contract SimpleAuction {  
    event HighestBidIncreased(address bidder, uint  
amount); // Event  
  
    function bid() public payable {  
        // ...  
        HighestBidIncreased(msg.sender, msg.value); //  
Triggering event  
    }  
}
```

- Events interface with EVM's logging capabilities
- Allows to call JavaScript callbacks in a dApp
- Function `bid()` is marked *payable* to receive funds

# Namecoin in Solidity

```
contract Namespace {  
  
    struct NameEntry {  
        address owner;  
        bytes32 value;  
    }  
  
    uint32 constant REGISTRATION_COST = 100;  
    uint32 constant UPDATE_COST = 10;  
    mapping(bytes32 => NameEntry) data;  
  
    function nameNew(bytes32 hash){  
        if (msg.value >= REGISTRATION_COST){  
            data[hash].owner = msg.sender;  
        }  
    }  
  
    function nameUpdate(bytes32 name, bytes32 newValue, address newOwner){  
        bytes32 hash = sha3(name);  
        if (data[hash].owner == msg.sender && msg.value >= UPDATE_COST){  
            data[hash].value = newValue;  
            if (newOwner != 0){  
                data[hash].owner = newOwner;  
            }  
        }  
    }  
  
    function nameLookup(bytes32 name){  
        return data[sha3(name)];  
    }  
}
```