# Introduction to Ethereum's Solidity

Smart contract & Dapp programming

# Layout

- Tools
- Workflow
- Solidity
- Code
- Outro

# Tools

# Editors

## Tools

**https://github.com/tomlion/vim-solidity**

# Truffle

## Tools

**https://trufflframework.com**

# Testrpc

## Tools

**https://github.com/ethereumjs/testrpc**

# OpenZeppelin

## Tools

**https://openzeppelin.com**

# Web3.js

## Tools

**https://github.com/ethereum/web3.js**

# Remix

## Tools

**https://ethereum.github.io/browser-solidity/**

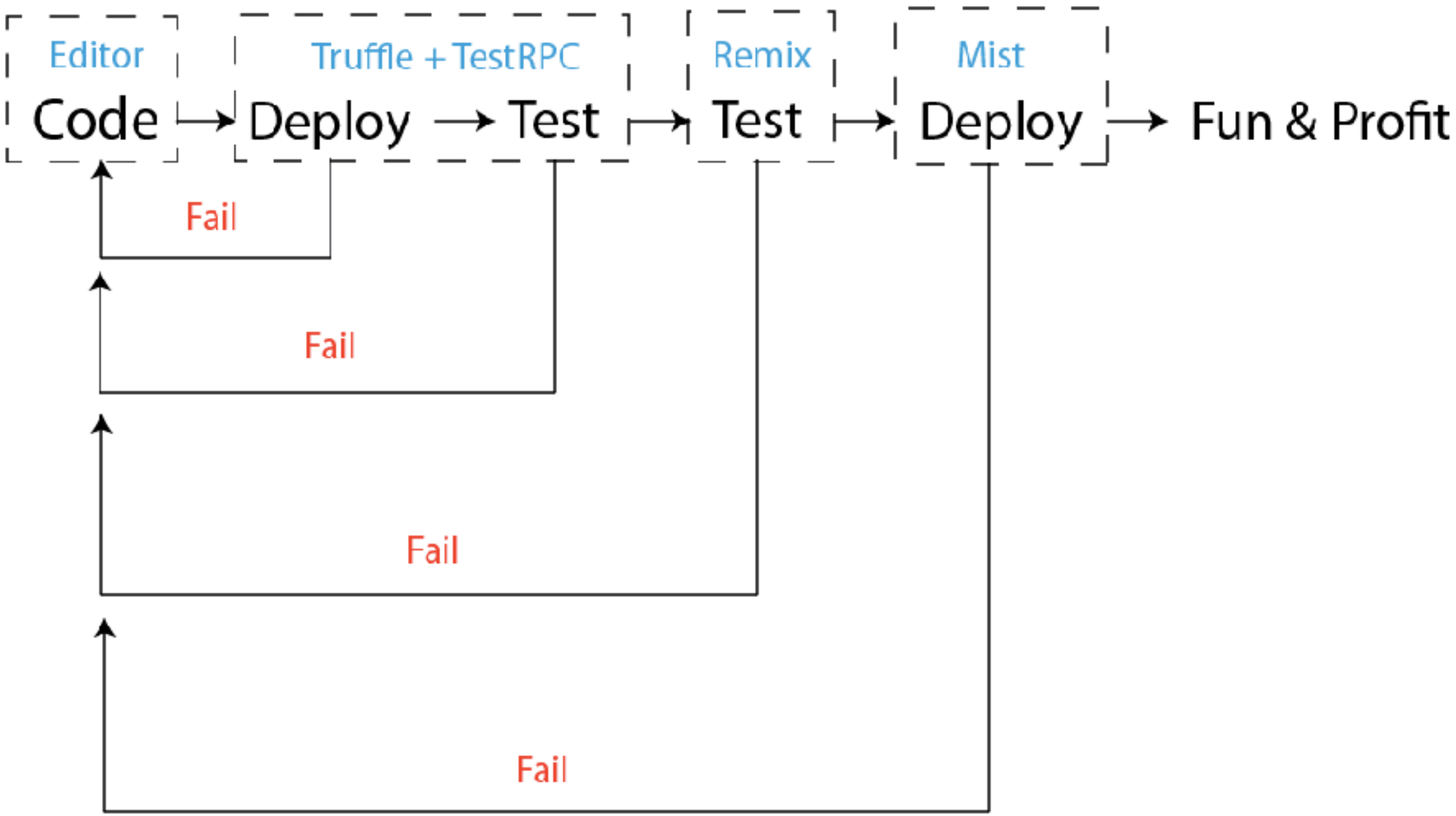
# Dapp

## Tools (Honorable Mention)

**https://dapp.readthedocs.io**

# Workflow

## Tools



# Introduction

Ethereum’s Solidity

# Sample Code

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) {
        storedData = x;
    }

    function get() constant returns (uint) {
        return storedData;
    }
}
```

# Initialization

- **Pragma** - pragmas are instructions for the compiler about how to treat the source code

- **Contract** - A contract in the sense of Solidity is a collection of code (its *functions*) and data (its *state*) that resides at a specific address on the Ethereum blockchain.

- `uint storedData;` declares a state variable called `storedData` of type `uint` (unsigned integer of 256 bits).

# Variables & Types

- Function Scoped (Global / Local variables)
- uint (unsigned)
- address (20 byte)
- enums, structs
- arrays (notation is reversed)
  As an example, an array of 5 dynamic arrays of uint is uint[][5]
  Can also be marked `public` to create a getter
- bytes, strings: special arrays (bytes cheaper than bytes[])
- constant - constant state variables

# Function Structure

```
function (<parameter types>) {internal|external} [pure|constant|
view|payable] [returns (<return types>)]
```

# Function Types

- **Internal** (default) - can only be called inside the current contract.
- **External** - can be passed via and returned from external function calls.
- (Calling a function `f` is internal `this.f` is external)

- **Contract functions themselves are `public` by default, only when used as the name of a type, the default is internal.**

# Function Types Cont’d

- **Pure:** promise not to read from or modify the state.
- **View:** promise not to modify the state.
- **Constant:** (alias to view)

# Notes on `constant`

- Functions that have `constant` typically read data, therefore the functions require a `return`

- Functions without `constant` typically write data therefore the function doesn’t need a `return`

# Mappings

- **Syntax**:
  mapping( <KeyType> => <ValueType> ) <variable>

- **KeyType** (keccak256): can be almost any type except for a mapping, a dynamically sized array, a contract, an enum and a struct.

- **ValueType**: can actually be any type, including mappings.

- **Default Values**: all zeros

- Only State Variables

- Not iterable

# Structs

```
struct Campaign {
    address beneficiary;
    uint fundingGoal;
    uint numFunders;
    uint amount;
    mapping (uint => Funder) funders;
}

uint numCampaigns;
mapping (uint => Campaign) campaigns;
```

# Special Variables & Functions

- **msg.value:** Amount of WEI being sent
- **msg.sender:** Address of agent sending message (initializer)
- <address>.balance
- <address>.transfer(<amount>);
- selfdestruct(address recipient)

# MOST Special of Functions

- function () payable {
  }
- a function has no name
- default function for contract (fall back)
- process payments sent to contract by default here

# Modifiers

- **Modifiers** modify (wrap) a function

- **Example**:
  function withdrawAllTheMoney() public onlyOwner {
    msg.send(balance); // pr0tect me plz
  }

- modifier onlyOwner {
    if(msg.sender != owner) {
      revert();
    }
    _;  // This is where the function goes
  }

# Events

- **Syntax:** event Name (<types>)

- Used as Debug/Logging/Event watching

# Pro-tips

- Explicitly declare types

- Explicitly convert types

# BORING!

# SHOW ME THE MONEY

**Put this code into Remix:**
**https://github.com/SynapseOrg/Introduction-to-Solidity/blob/master/TokenSale.sol**

# How to money?

- **EthLance!** https://ethlance.com/

# …but I need help?

- **Good:** https://solidity.readthedocs.io/
- **Good:** https://gitter.im/ethereum/solidity
- **Good:** https://www.reddit.com/r/ethdev/
- **Bad:** https://gitter.im/ethereum/remix
- **Bad:** #eth-dev on IRC

# …but I need MOAR halp

- **Come to Hack Days or Contact me!**
- **Twitter:** @dpg
- **Github:** dpgailey
- **Facebook:** Dan Gailey
- **Email:** dan@synapse.ai

# What’s next?

- Build Dapps
- Decentralize everything
- Disrupt existing monopolies
- Profit

# Synapse.ai

Decentralized Data + AI

**hello@synapse.ai**