Marlowe - 3 Steps to Working DApp

Steps

- Design
- Test
- Integrate

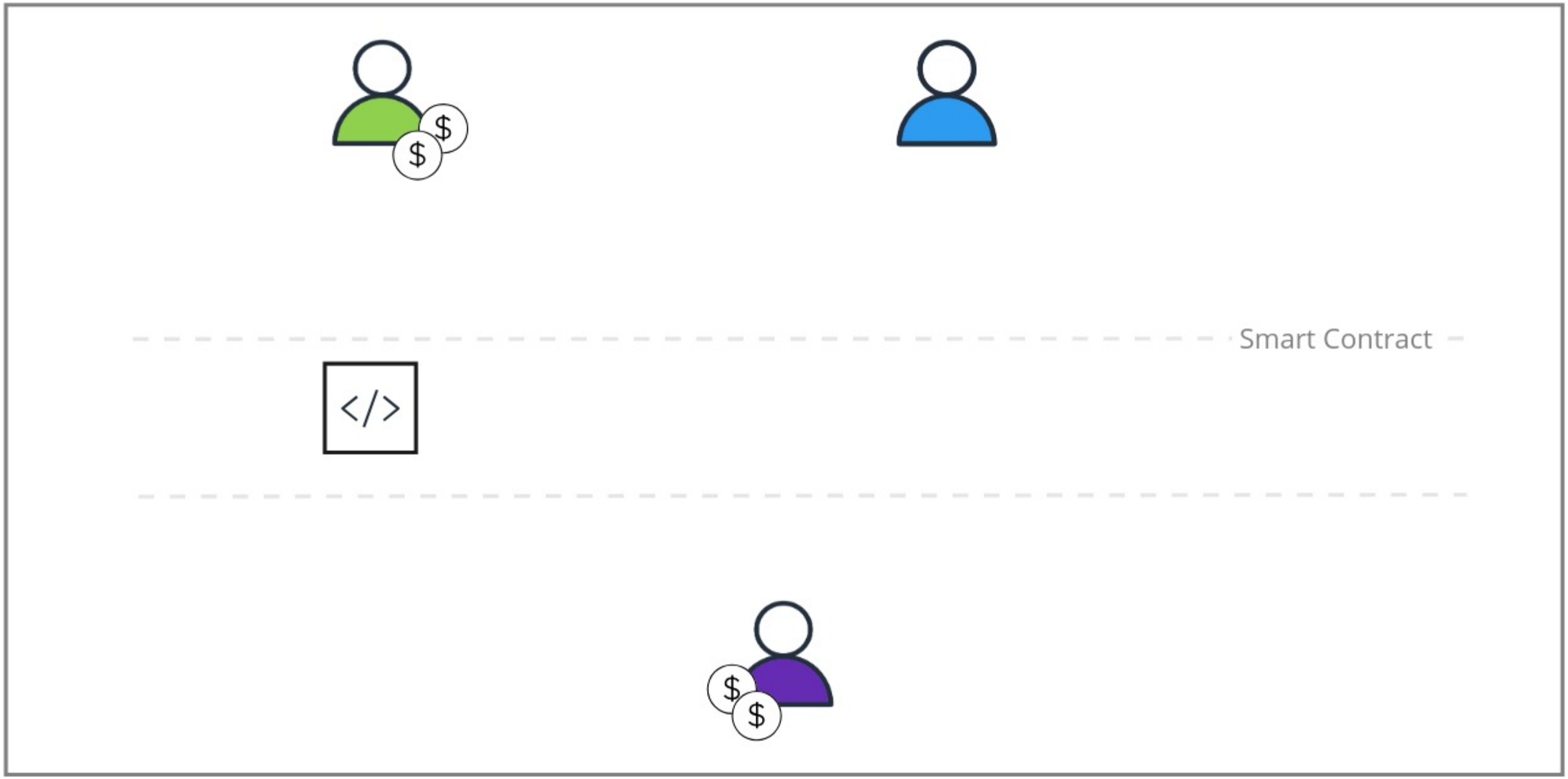
Project

On-Chain Betting Platform

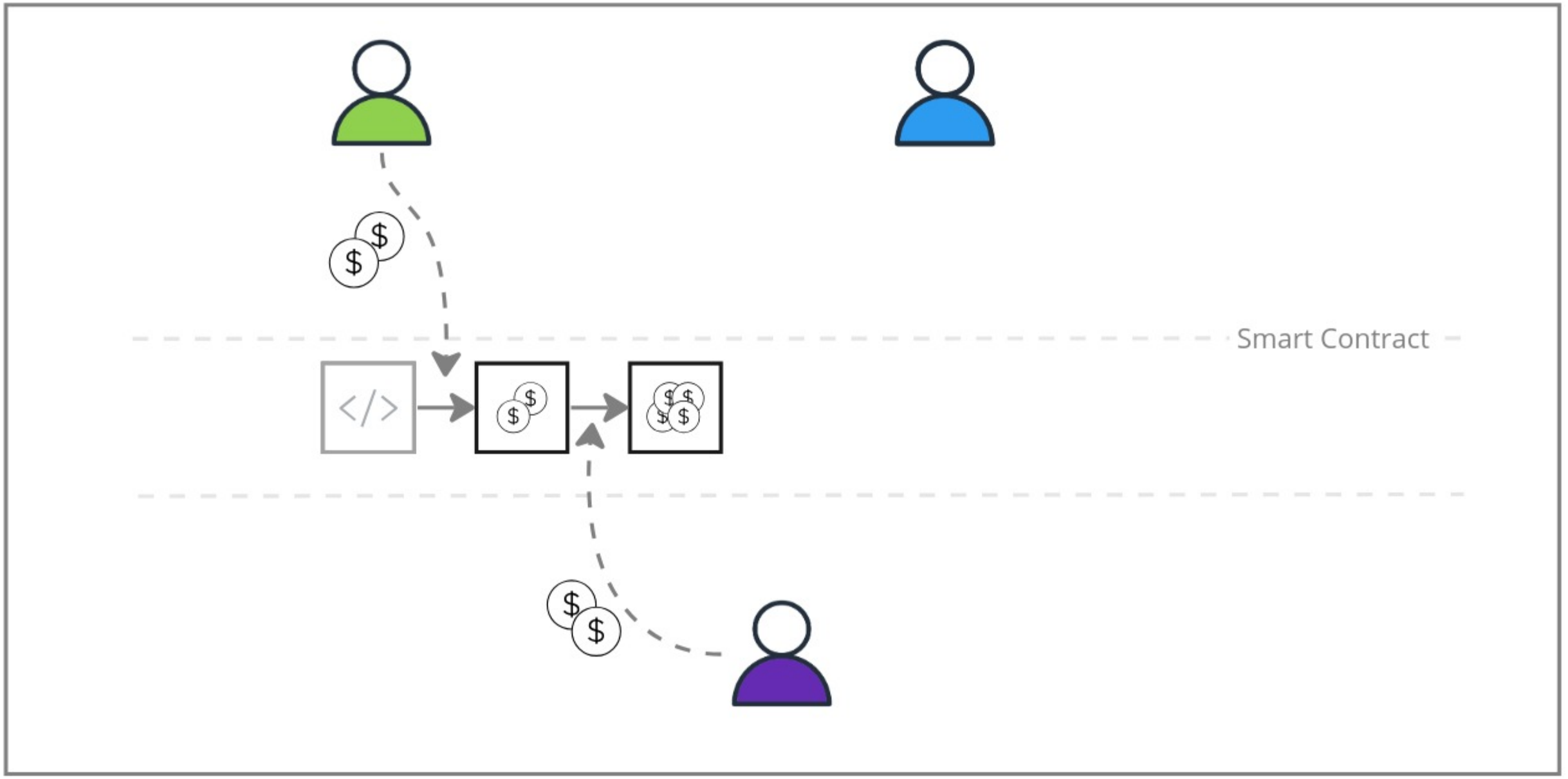
Steps

- **Design**
- Test
- Integrate

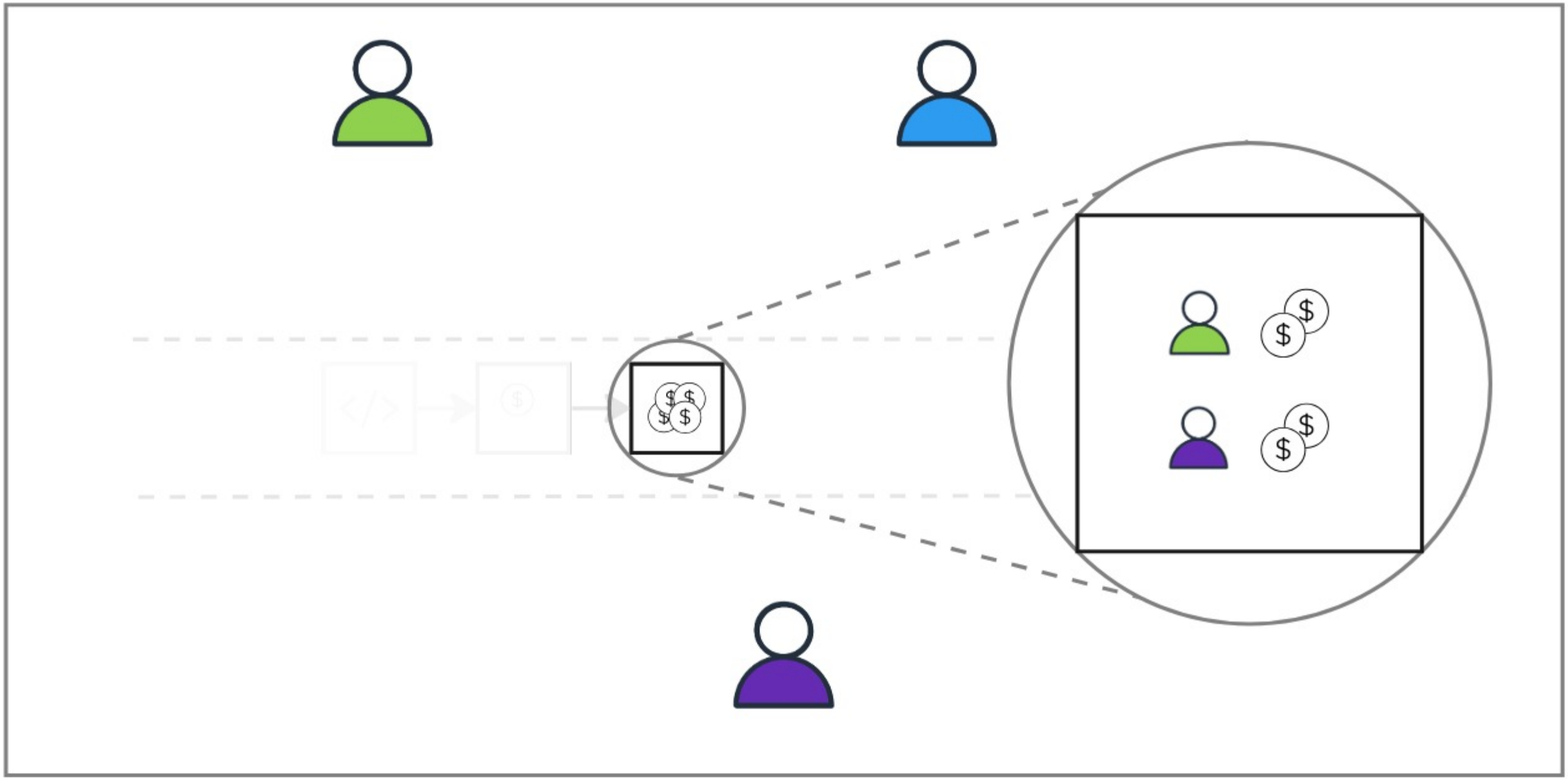
"On-Chain Betting"



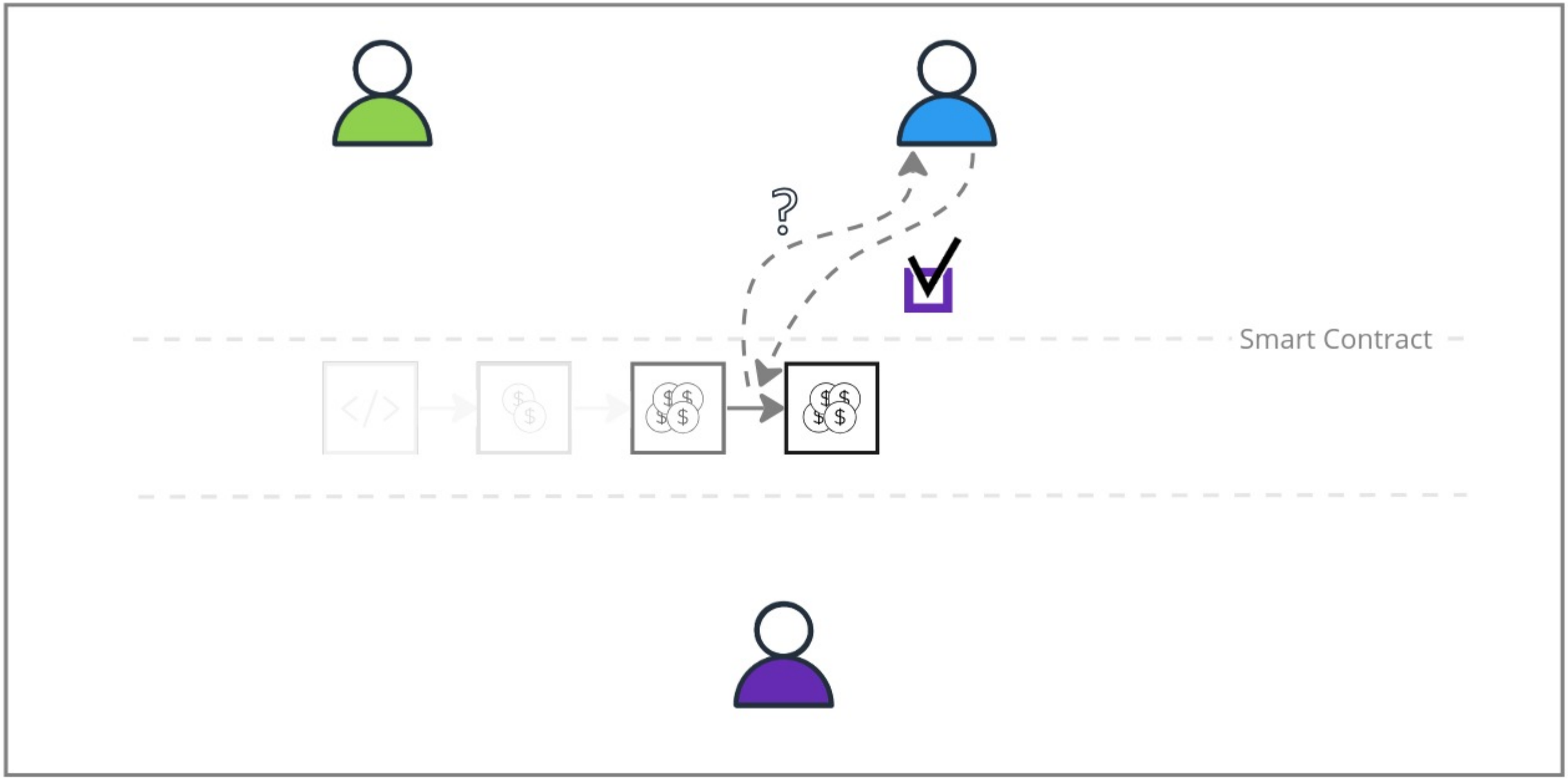
"On-Chain Betting"



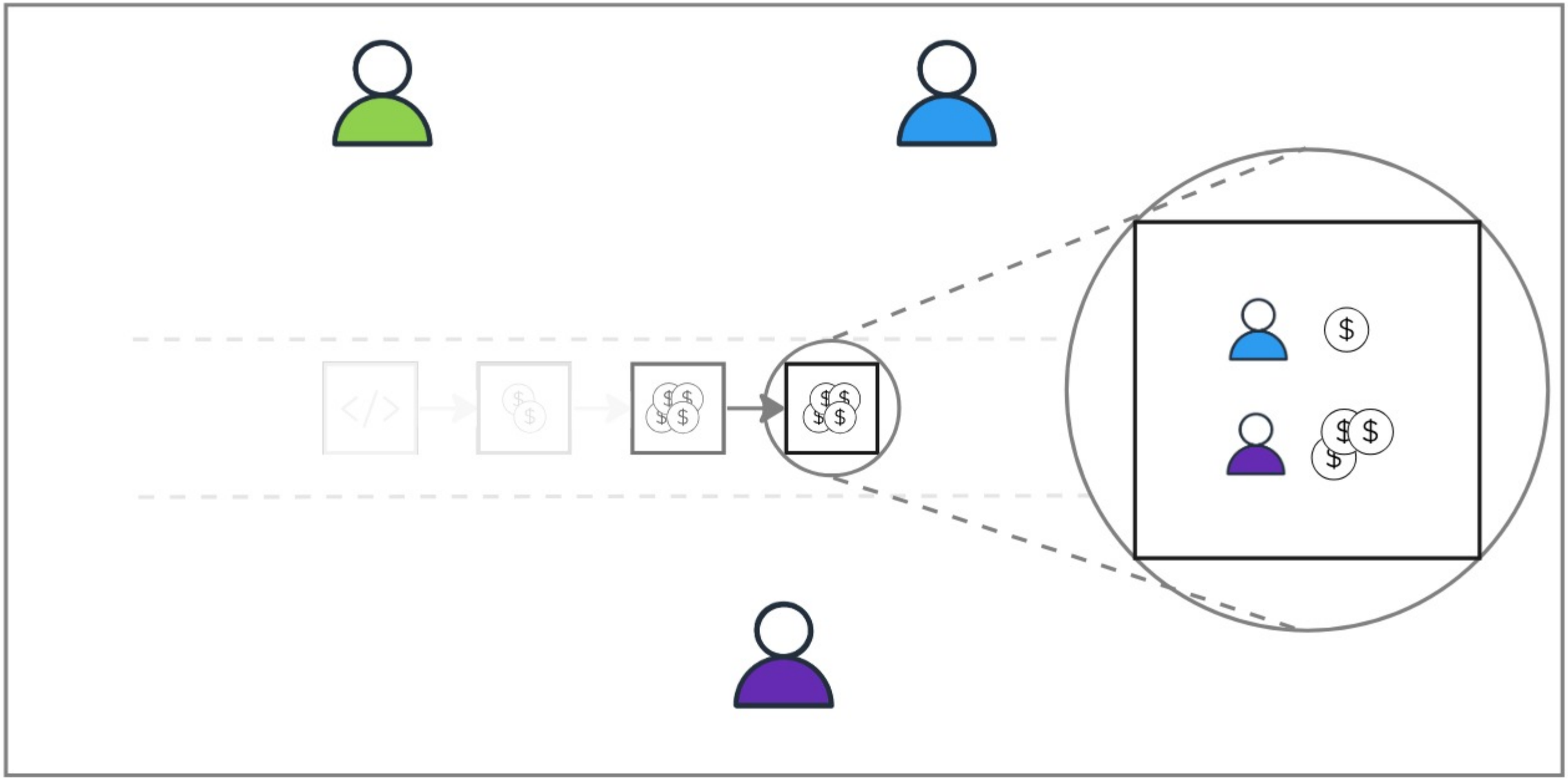
"On-Chain Betting"



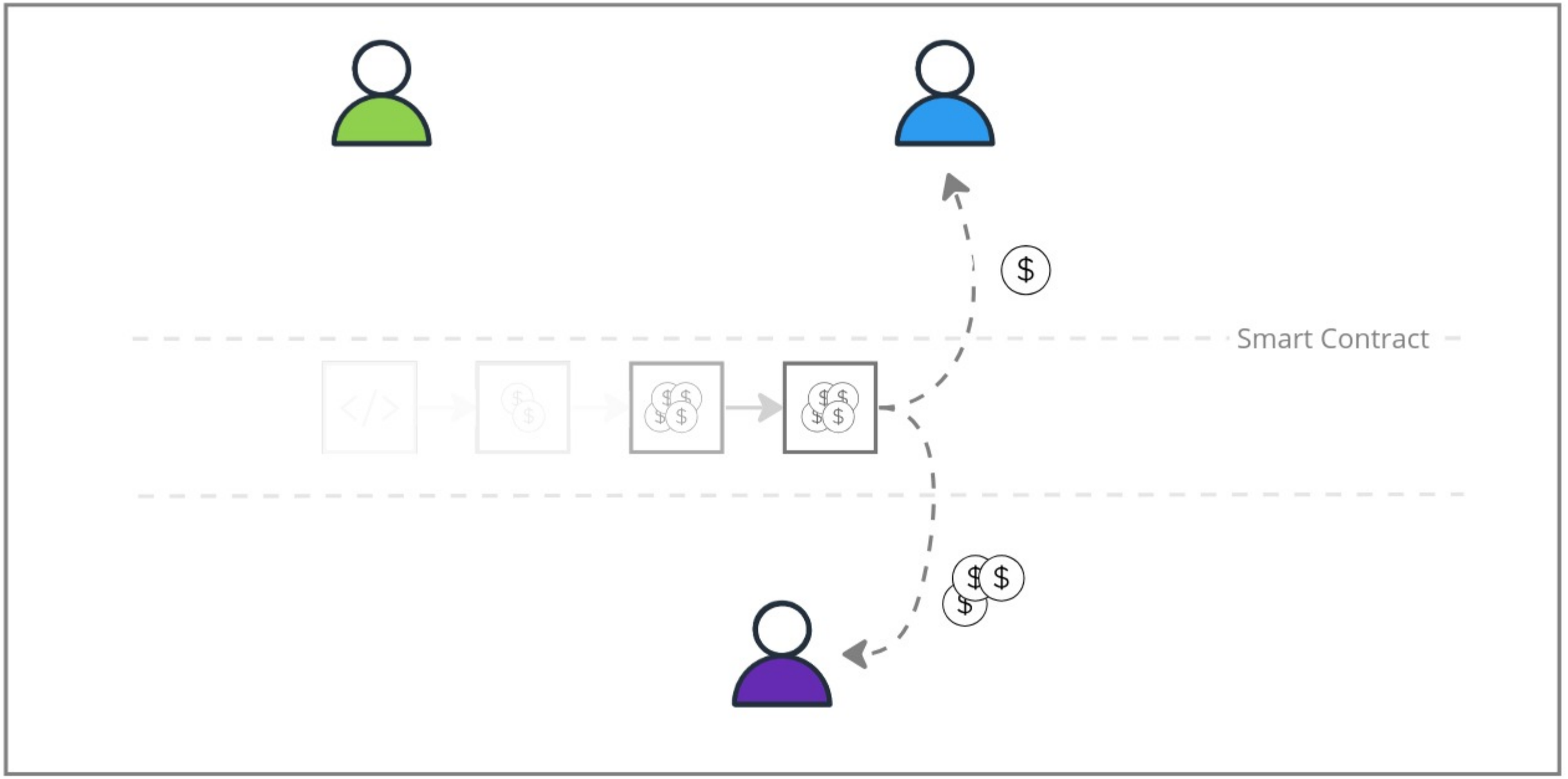
"On-Chain Betting"



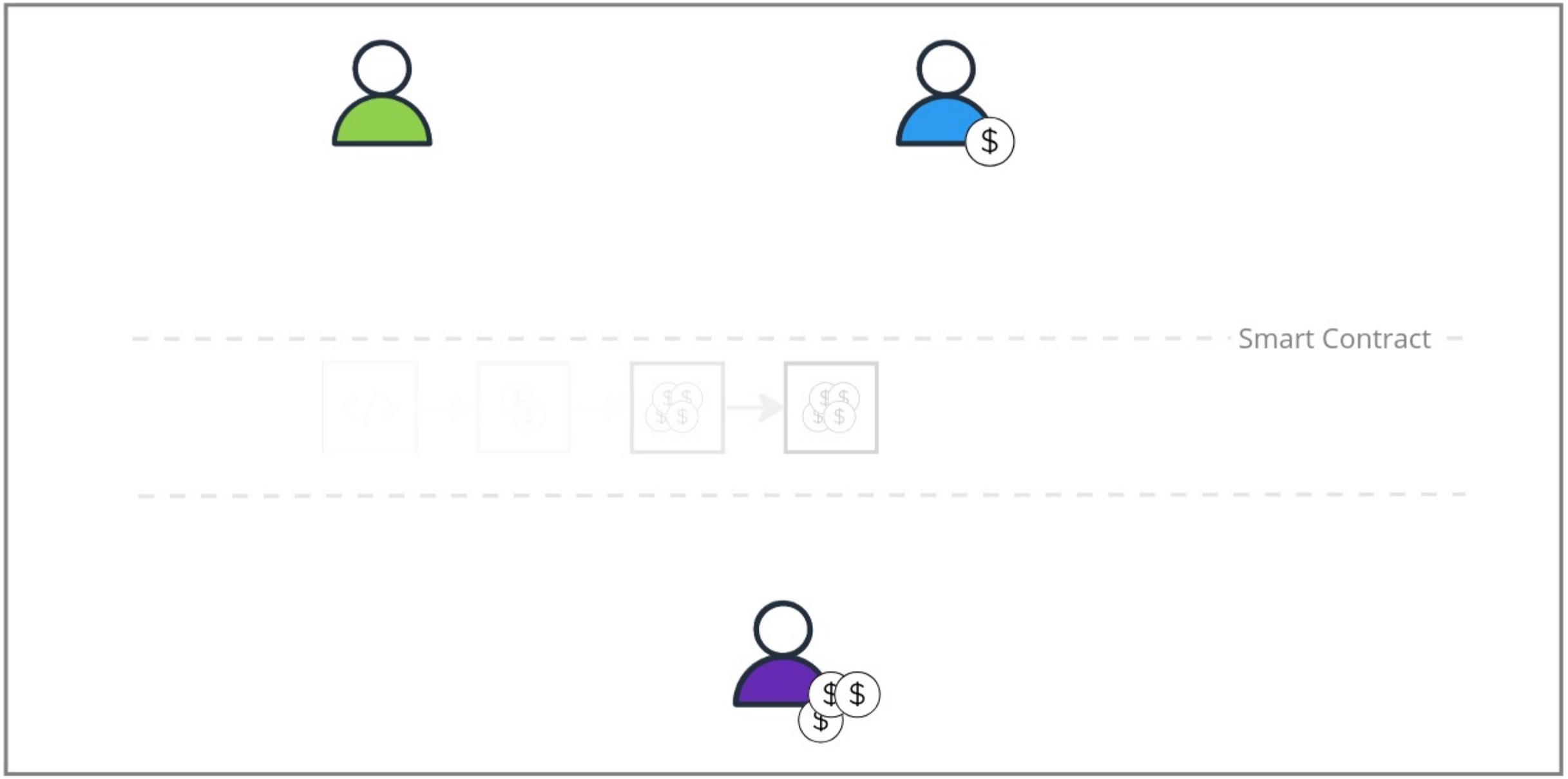
"On-Chain Betting"



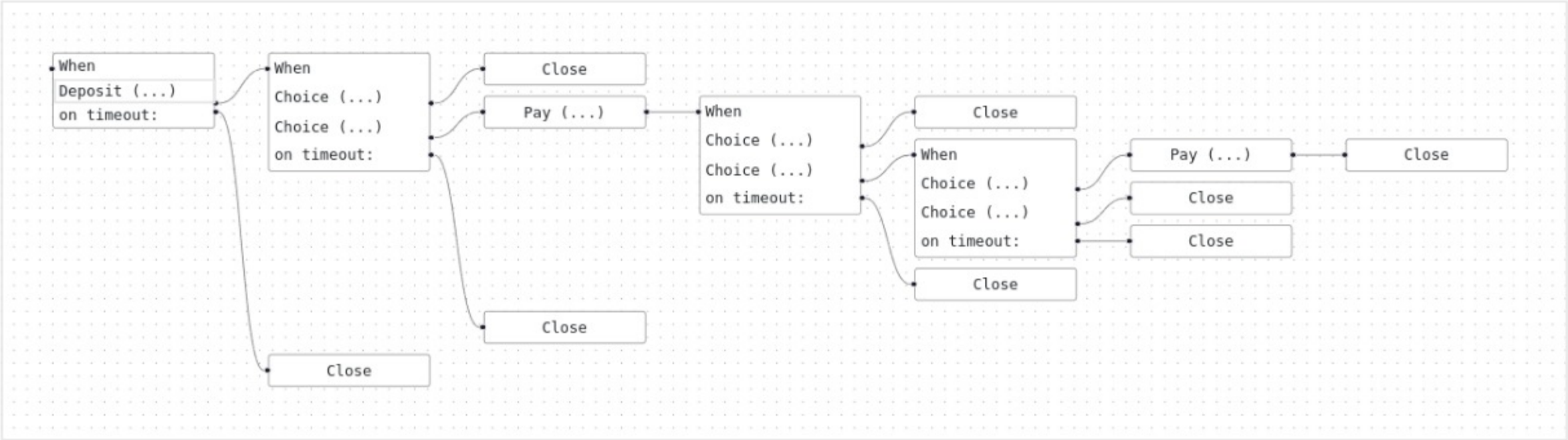
"On-Chain Betting"



"On-Chain Betting"

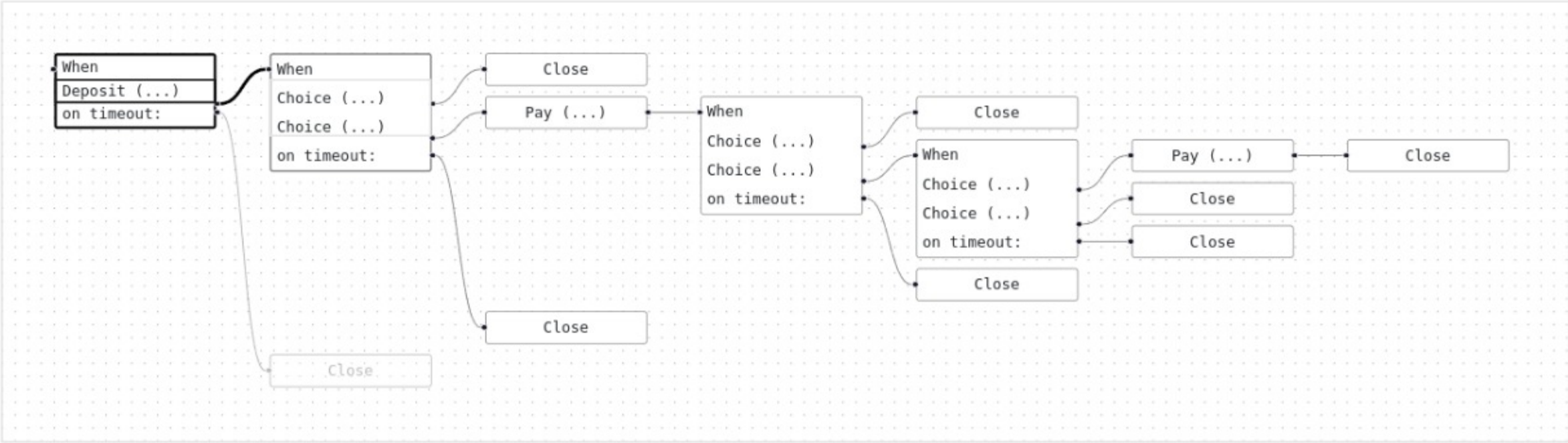


How Marlowe Operates



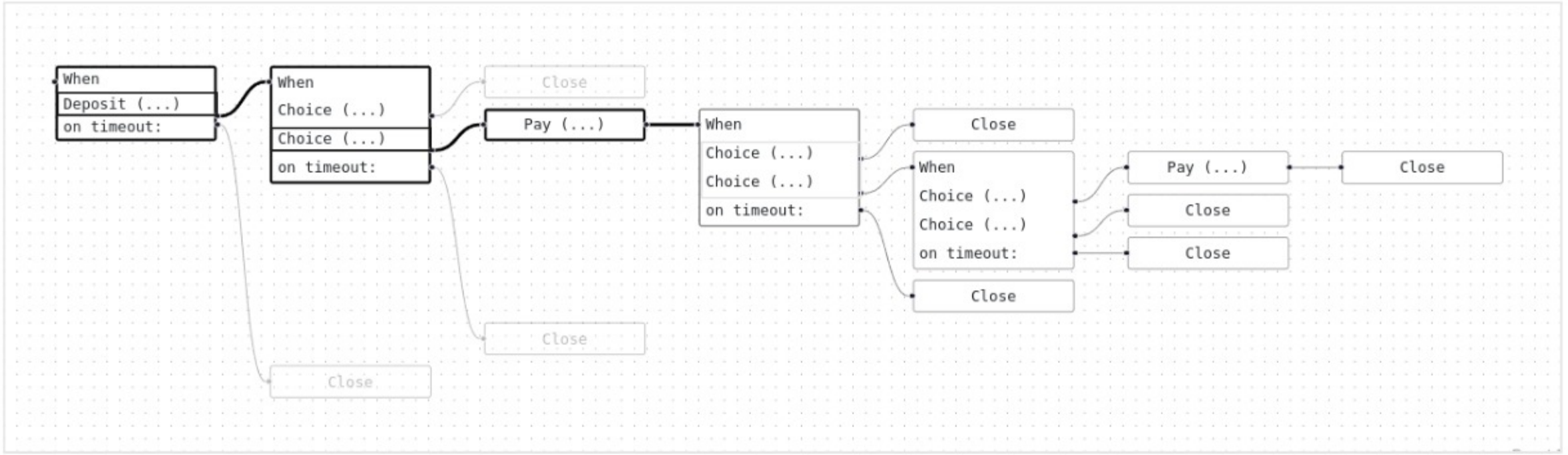
Contract structure is known upfront.

How Marlowe Operates



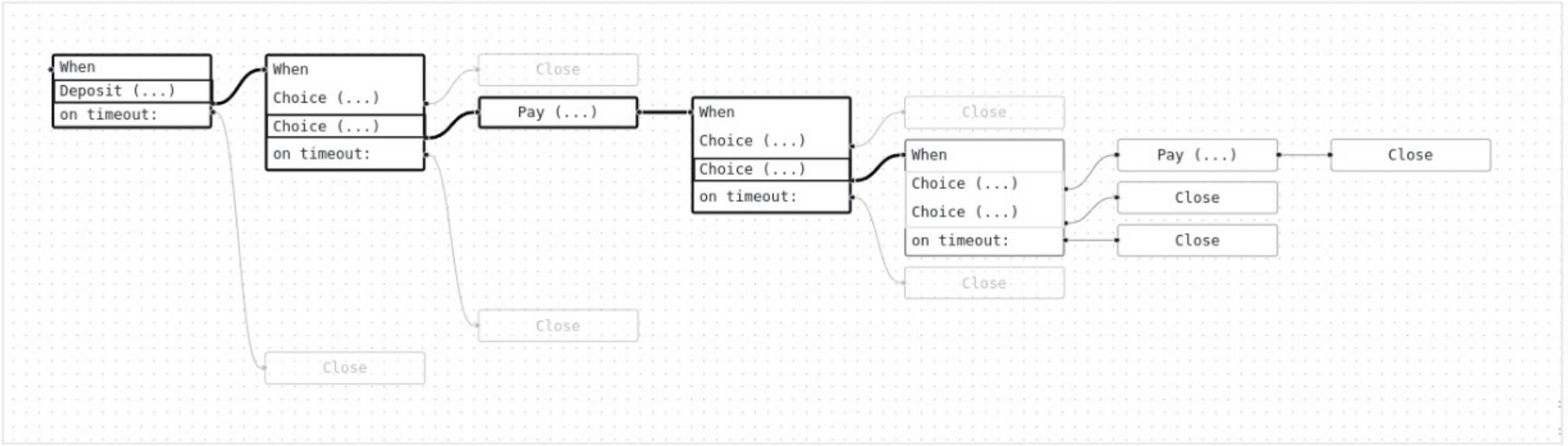
User inputs drives the execution.

How Marlowe Operates



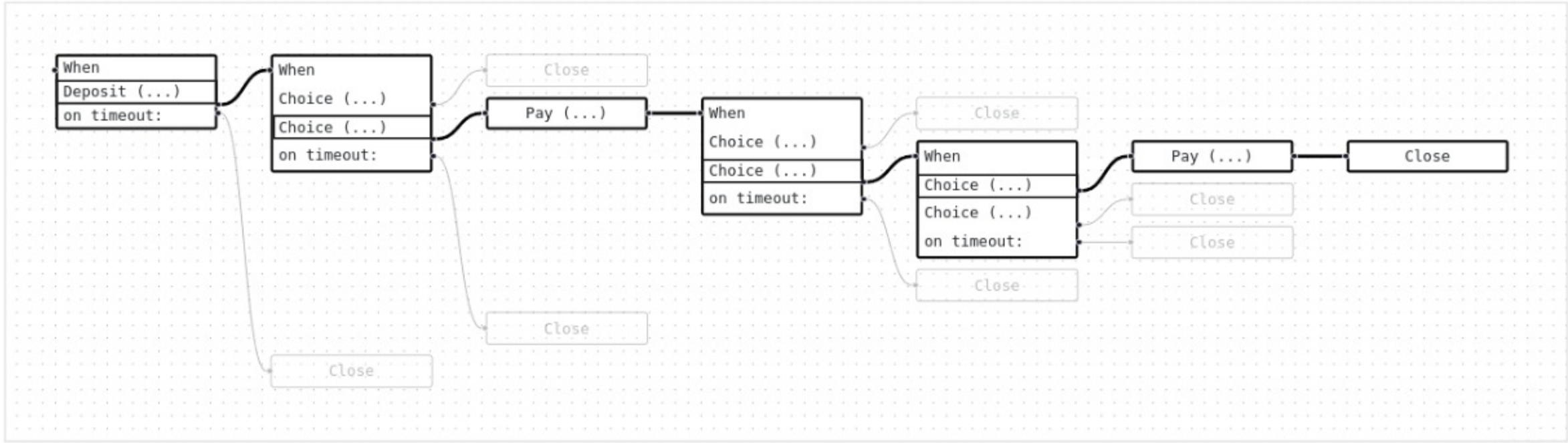
User inputs drives the execution.

How Marlowe Operates



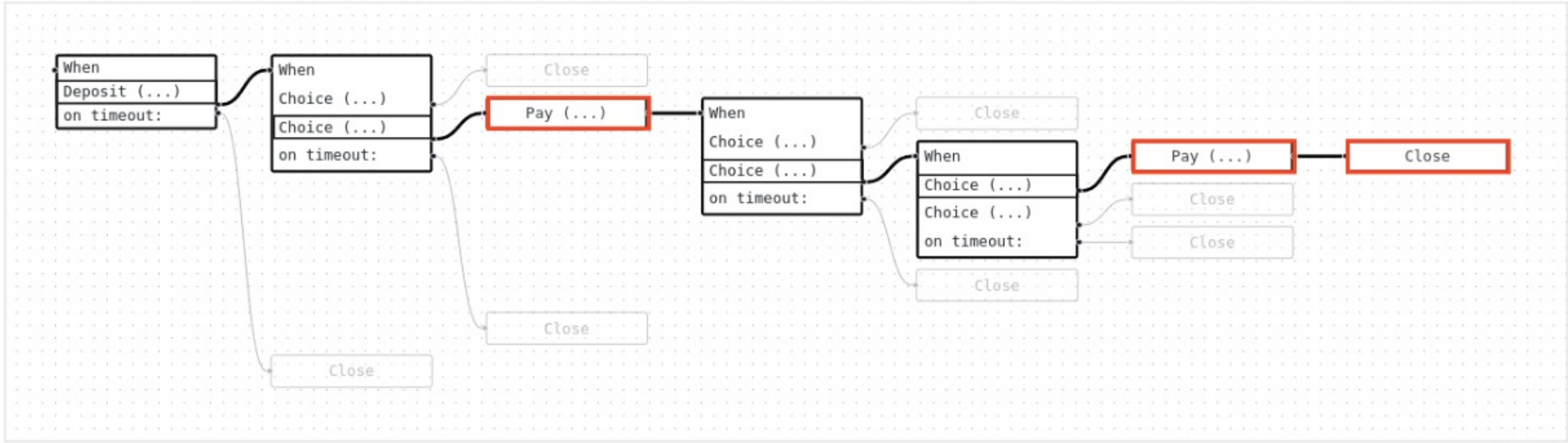
Circumstances drives the execution.

How Marlowe Operates



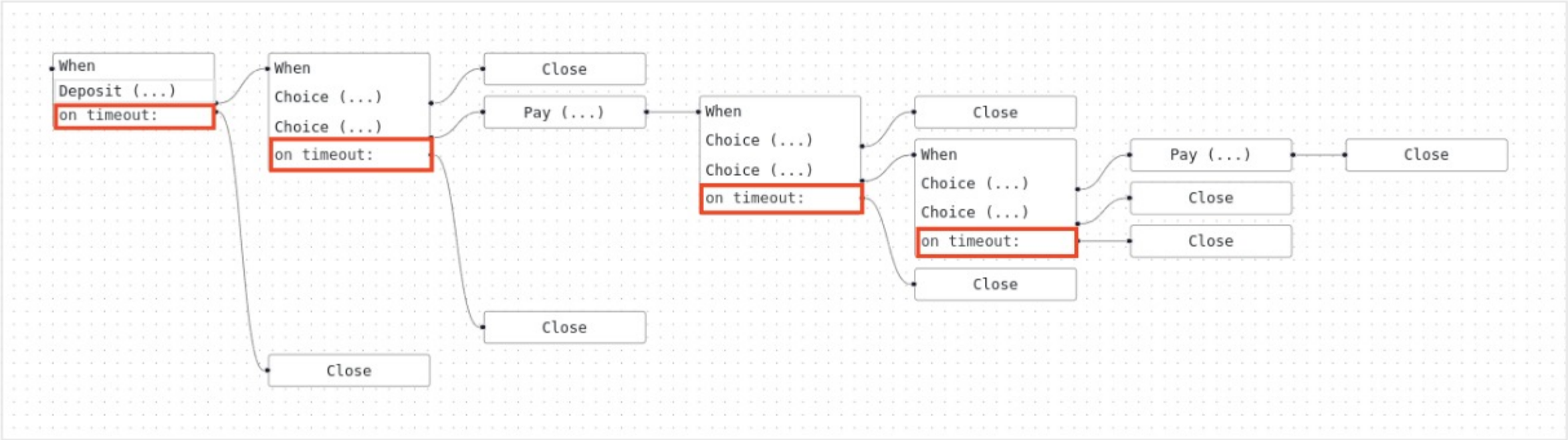
Path and values drive the cash flows.

How Marlowe Operates



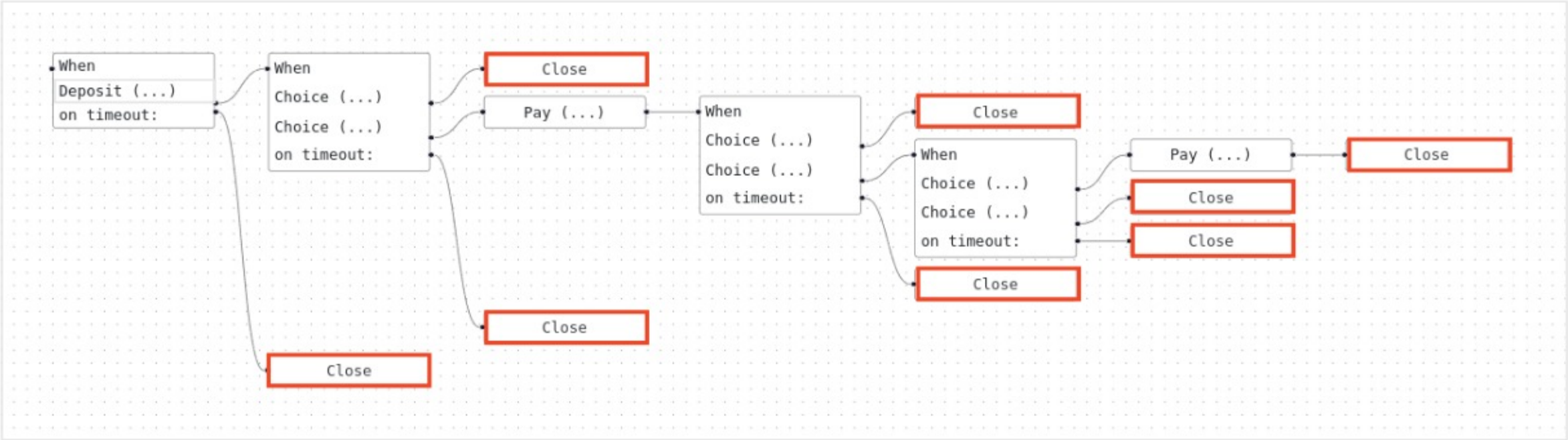
Path and values drive the cash flows.

How Marlowe Operates



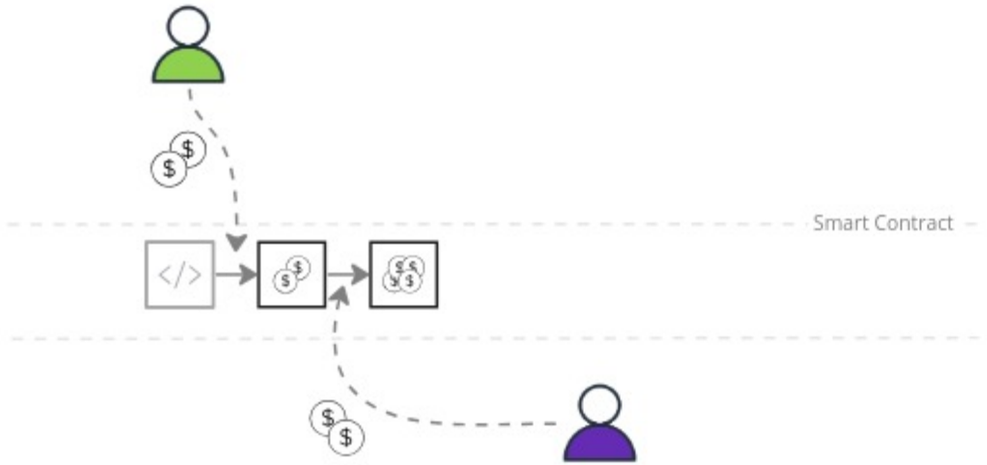
Time can influence the execution.

How Marlowe Operates

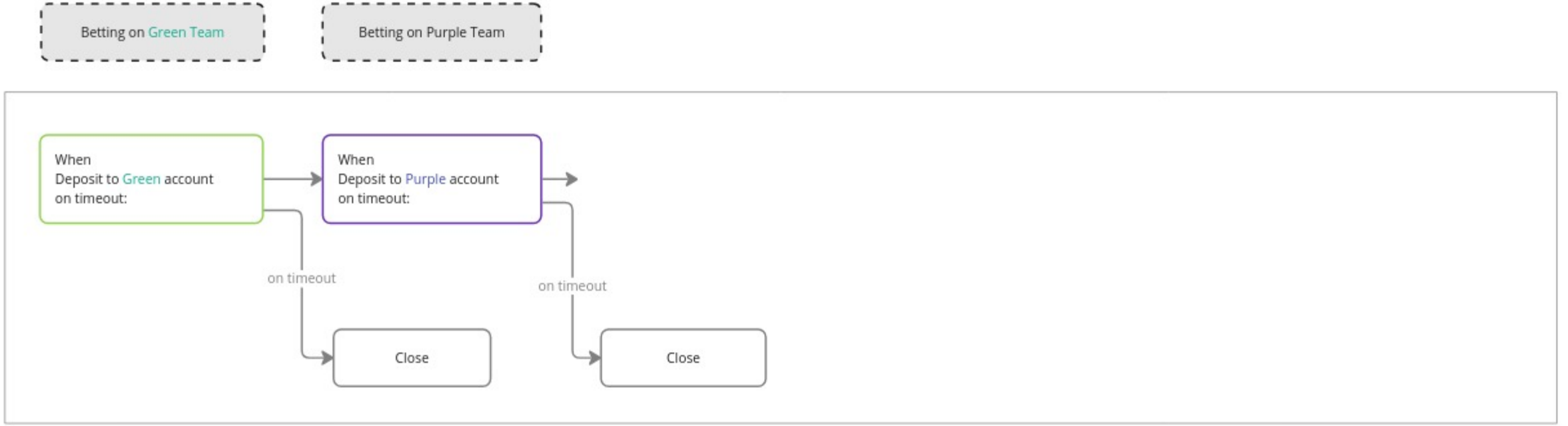


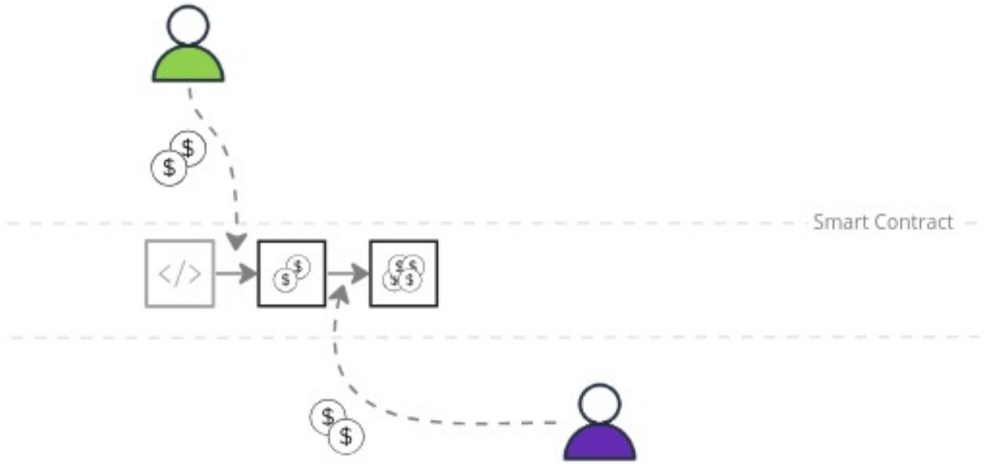
Every path is closed at the end.

Example: "On-Chain Betting"

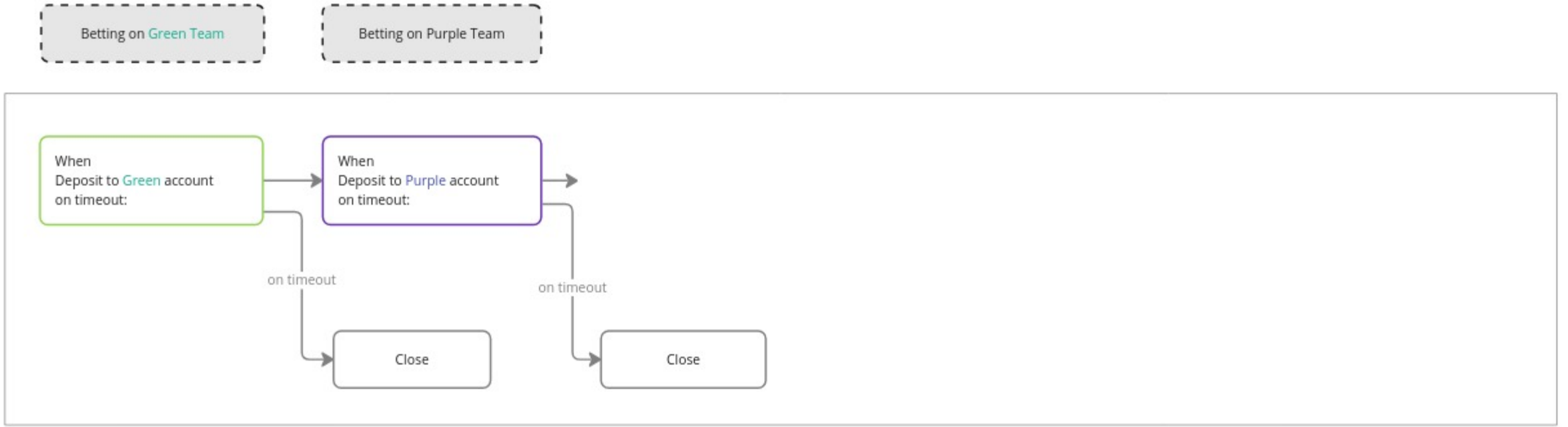


Example: "On-Chain Betting"



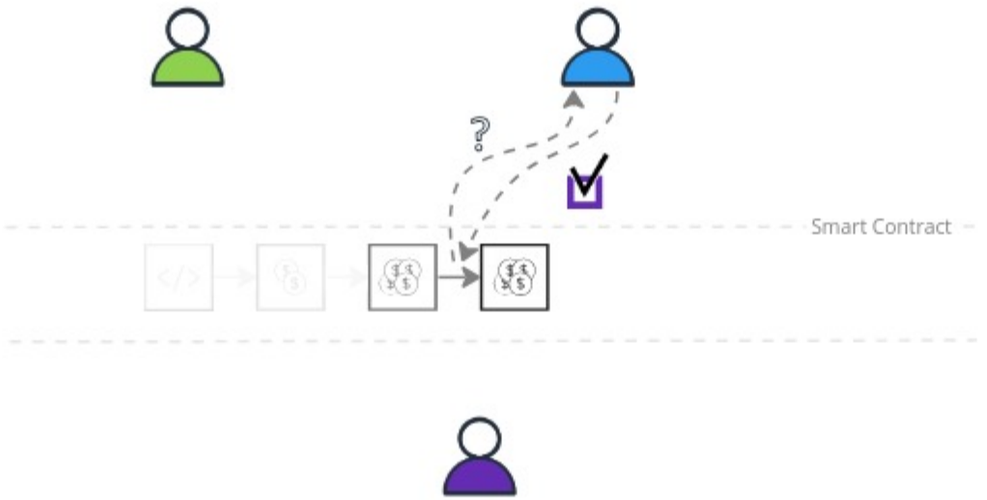


Example: "On-Chain Betting"

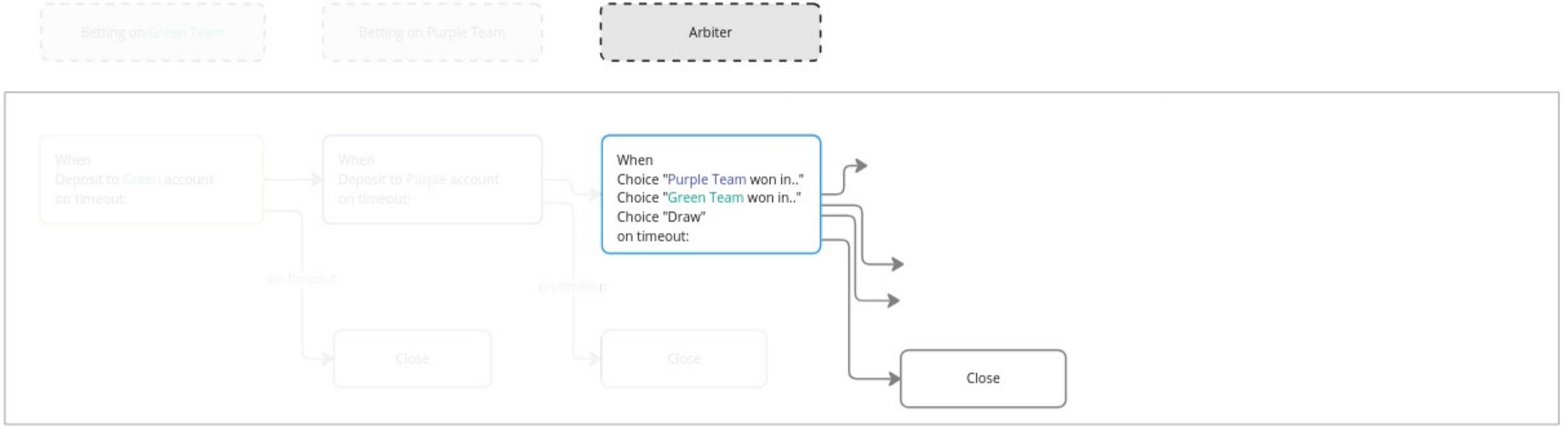


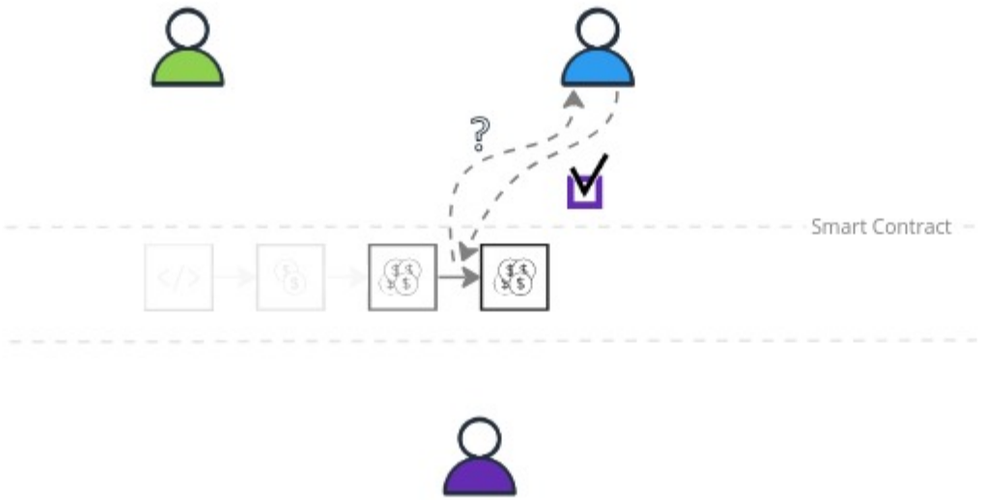
play.marlowe.iohk.io

Example: "On-Chain Betting"

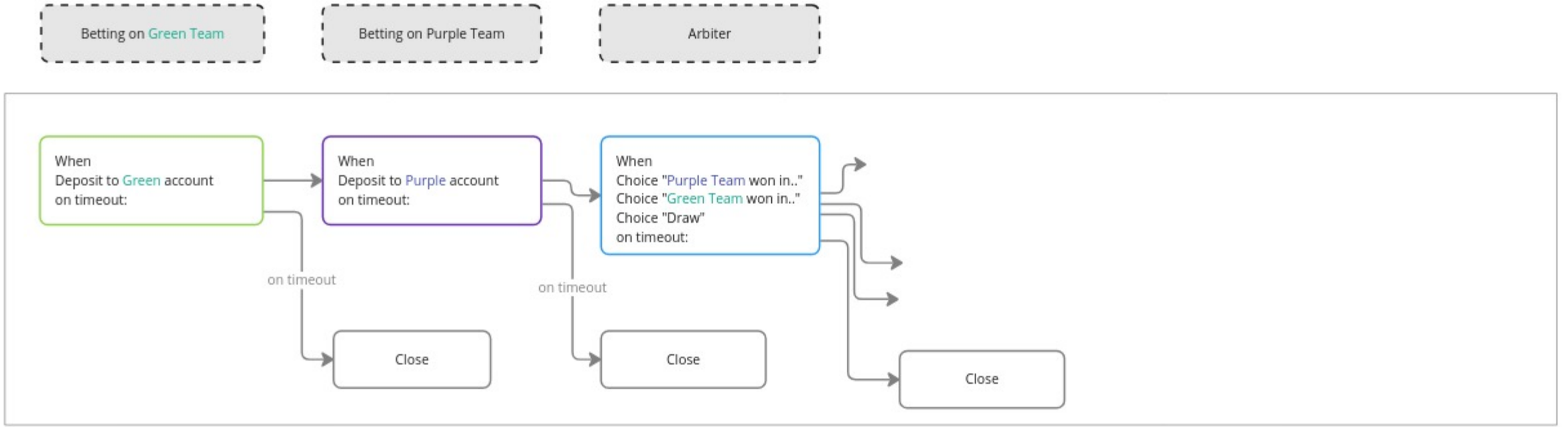


Example: "On-Chain Betting"



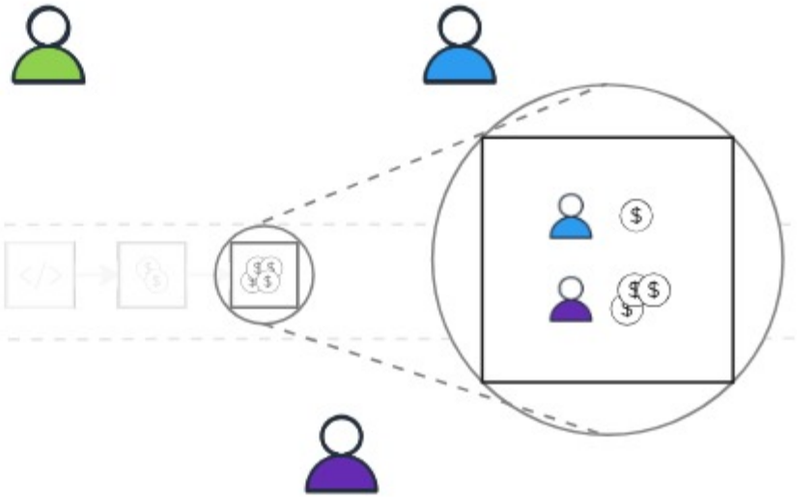


Example: "On-Chain Betting"

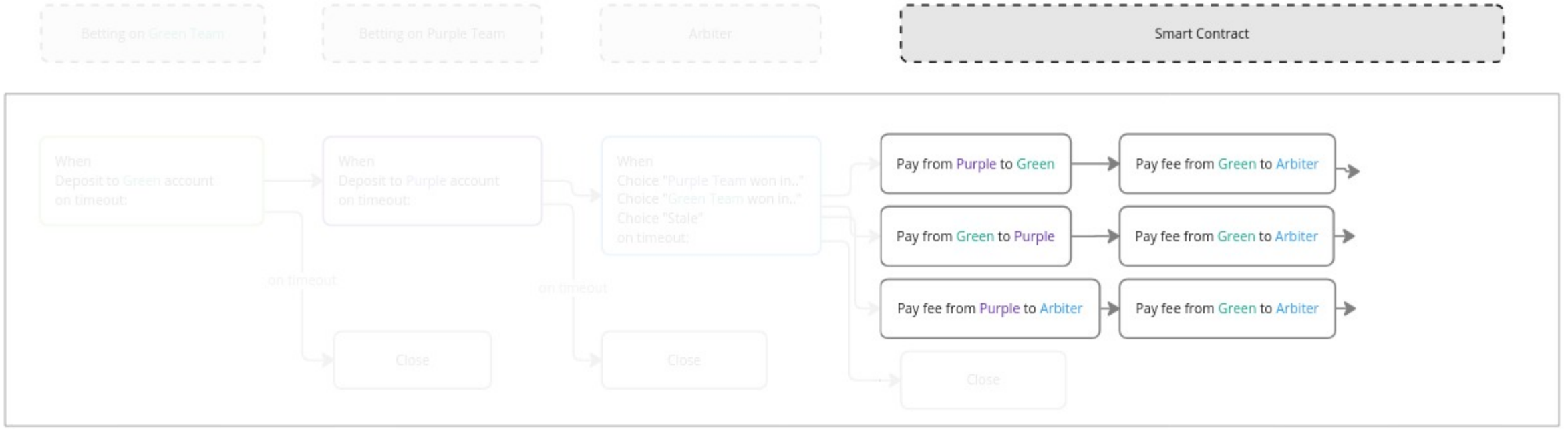


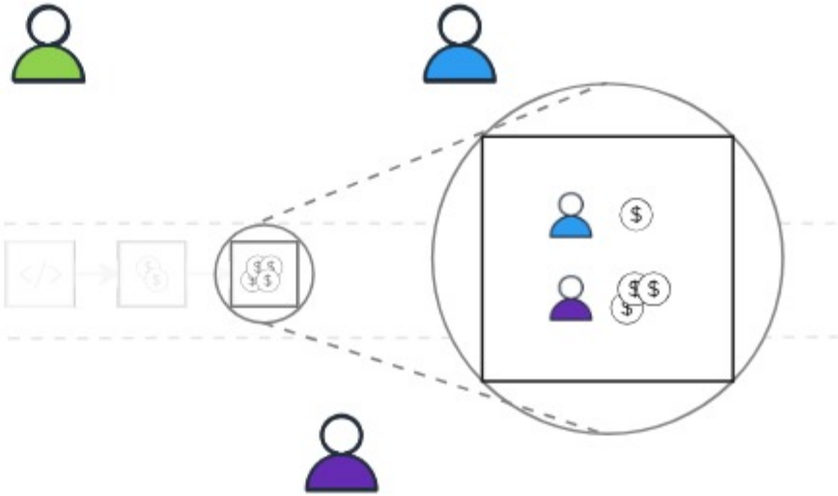
play.marlowe.iohk.io

Example: "On-Chain Betting"

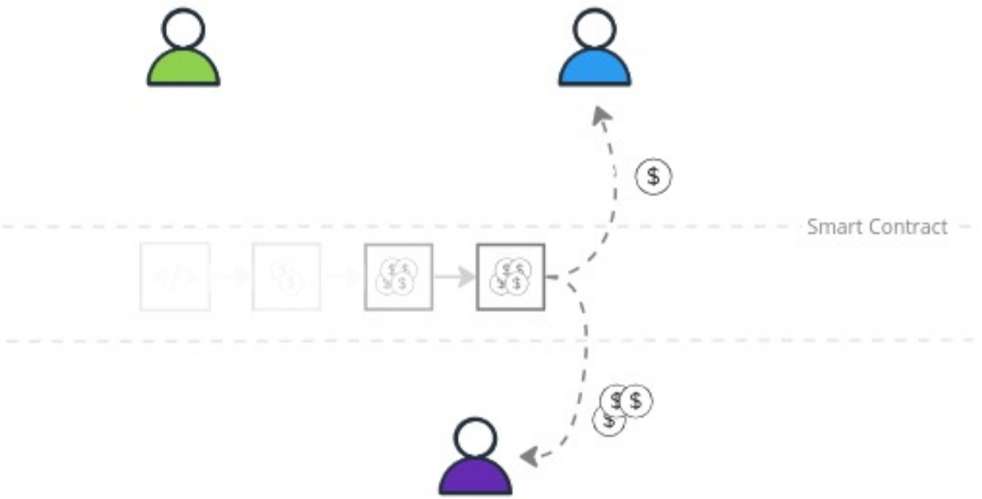


Example: "On-Chain Betting"

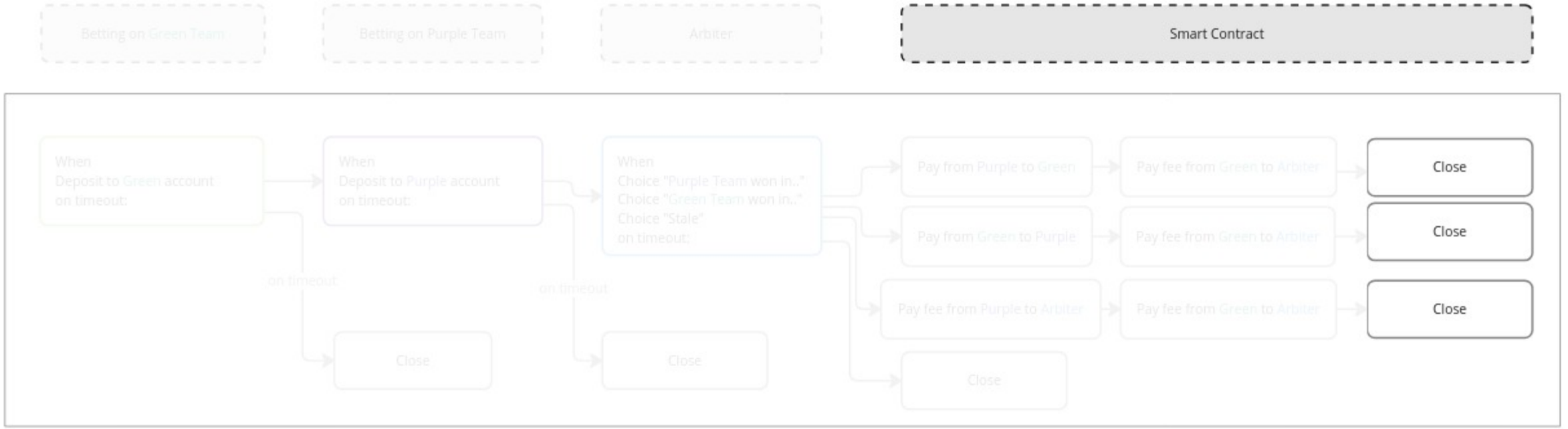


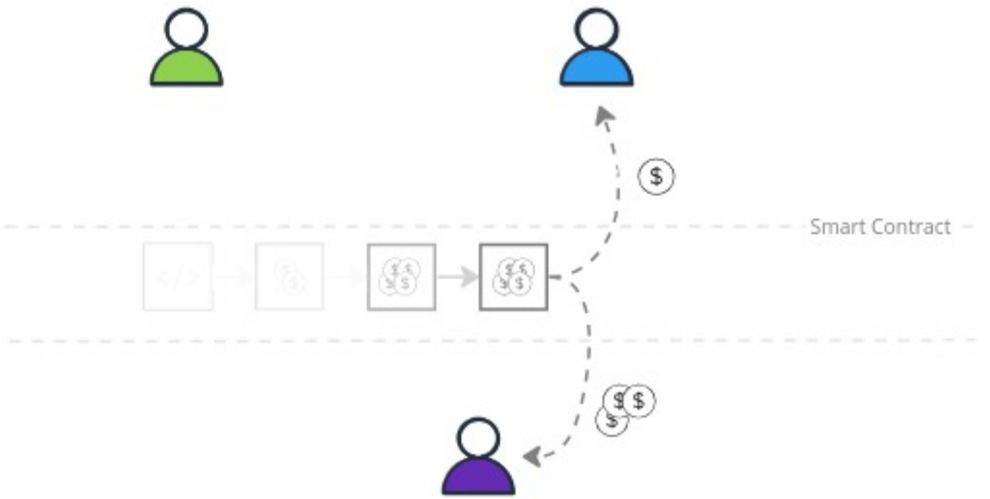


Example: "On-Chain Betting"

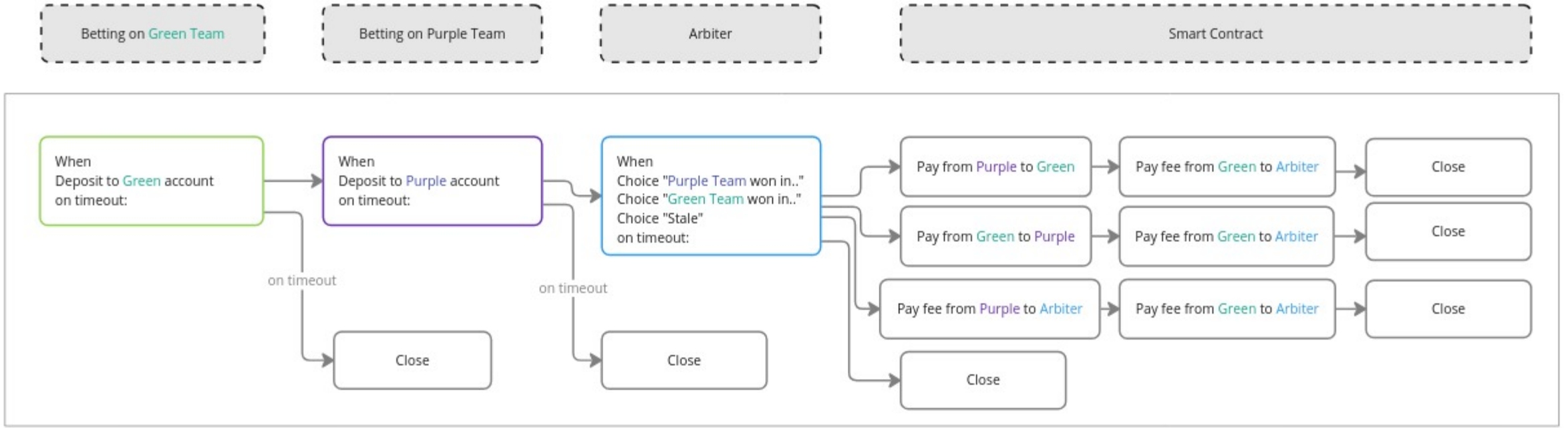


Example: "On-Chain Betting"





Example: "On-Chain Betting"



play.marlowe.iohk.io

Steps

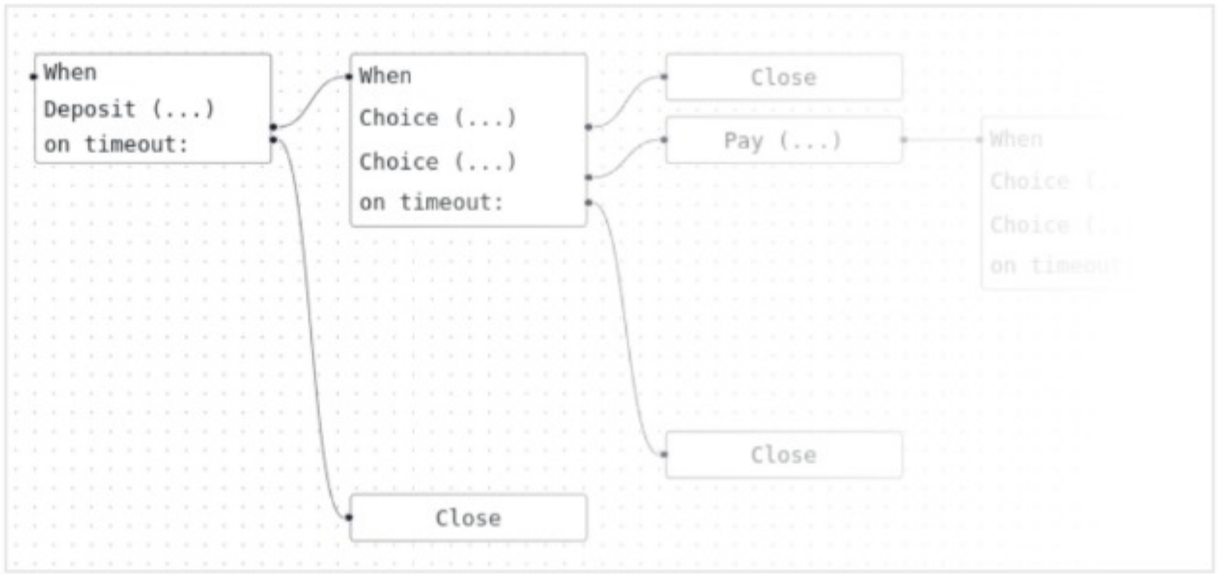
- **Design**
- Test
- Integrate

Steps

- Design
- **Test**
- Integrate

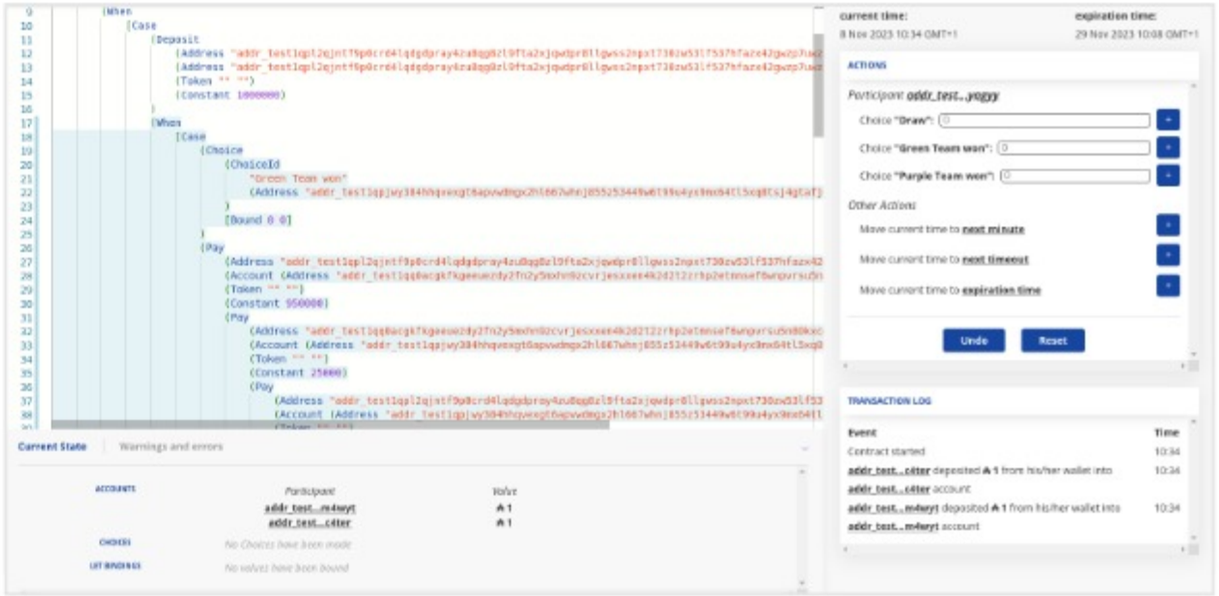
Test

- **Safety by construction**
- Static analysis
- Runner
- CLI tools
- Testing framework



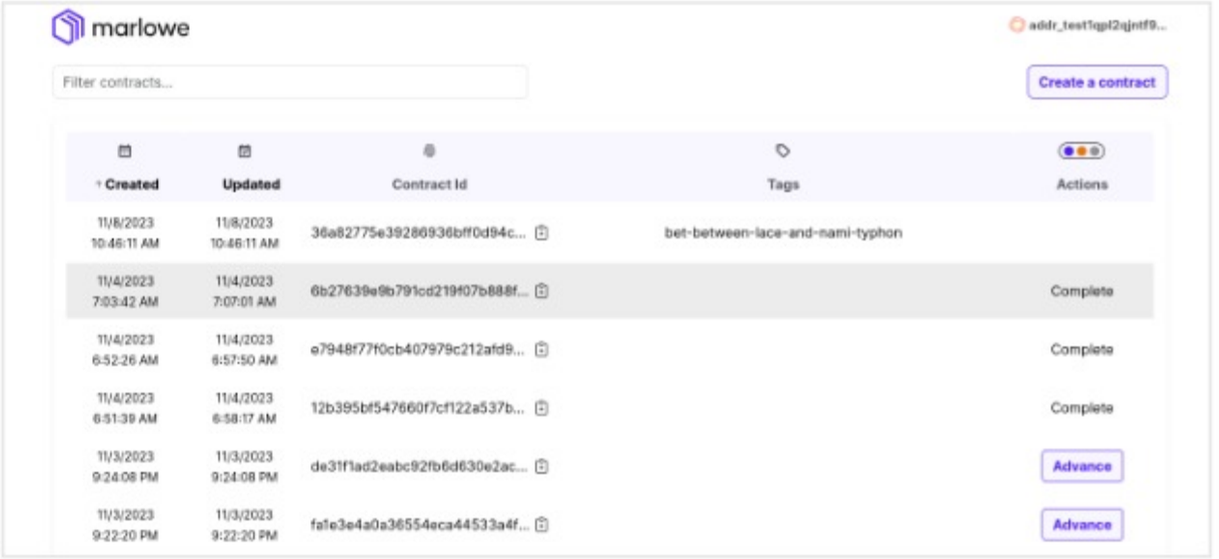
Test

- Safety by construction
- **Static analysis**
- Runner
- CLI tools
- Testing framework



Test

- Safety by construction
- Static analysis
- **Runner**
- CLI tools
- Testing framework



Test

- Safety by construction
- Static analysis
- Runner
- **CLI tools**
- Testing framework

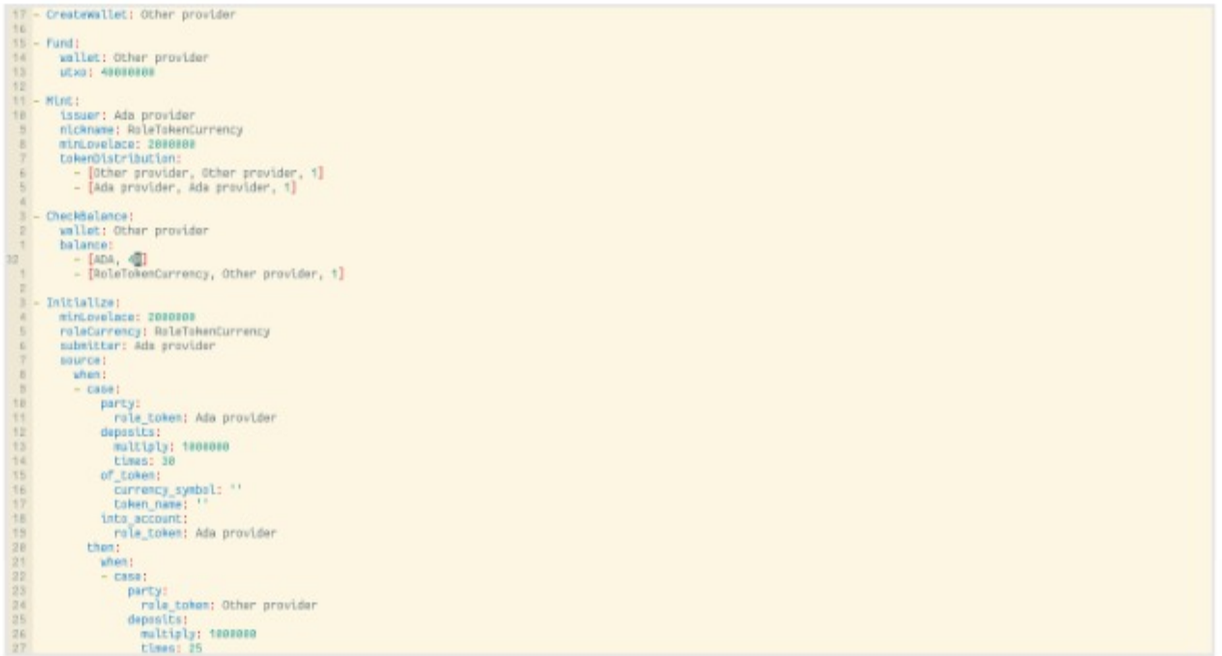
```
$ marlowe-cli run analyze --help
Usage: marlowe-cli run analyze (--mainnet | --testnet-magic INTEGER)
                               --socket-path SOCKET_FILE
                               --marlowe-file MARLOWE_FILE [--preconditions]
                               [--roles] [--tokens] [--maximum-value]
                               [--minimum-utxo] [--execution-cost]
                               [--transaction-size] [--best] [--verbose]

  [EXPERIMENTAL] Analyze a Marlowe contract.

Available options:
  --mainnet                Execute on mainnet.
  --testnet-magic INTEGER  Network magic. Defaults to the CARDANO_TESTNET_MAGIC
                           environment variable's value.
  --socket-path SOCKET_FILE
                           Location of the cardano-node socket file. Defaults to
                           the CARDANO_NODE_SOCKET_PATH environment variable's
                           value.
  --marlowe-file MARLOWE_FILE
                           JSON file with the state and contract.
  --preconditions          Whether to check preconditions for valid Marlowe
                           state.
  --roles                  Whether to check lengths of role names.
  --tokens                 Whether to check lengths of token names.
  --maximum-value          Whether to check the `maxValueSize` protocol limit.
  --minimum-utxo           Whether to check the `utxoCostPerWord` protocol
                           limit.
  --execution-cost         Whether to check the `maxTxExecutionUnits` protocol
                           limit.
  --transaction-size       Whether to check the `maxTxSize` protocol limit.
  --best                   Whether to compute tight estimates of worst-case
                           bounds, instead of generous estimates of those
                           bounds.
```

Test

- Safety by construction
- Static analysis
- Runner
- CLI tools
- **Testing framework**



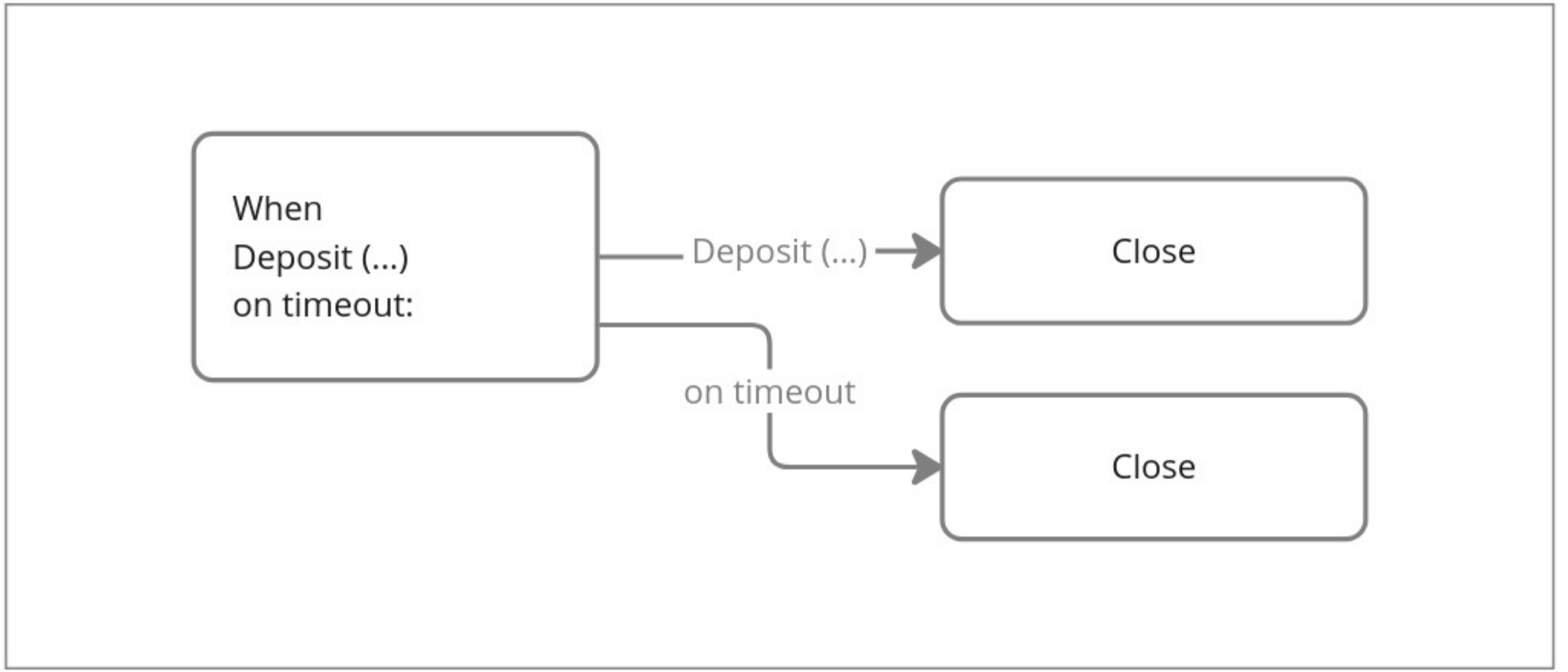
Steps

- Design
- **Test**
- Integrate

Steps

- Design
- Test
- **Integrate**

Dynamic Contract Generation



play.marlowe.iohk.io

Dynamic Contract Generation

```
{
  "when": [
    {
      "case": {
        "party": {
          "address": "addr_test1qq0acgkfkgeeuezdy2fn2y5mxhn9zcvrjesxxen4k2d2t2zrhp2e
              tmnsef6wnpvrsu5n80kxceafnxpv5te923agndxs5c4ter"
        },
        "deposits": 1000000,
        "of_token":
...
      }
      "then": "close",
    }
  ],
  "timeout": 1701252900000
  "timeout_continuation": "close",
}
```

Dynamic Contract Generation

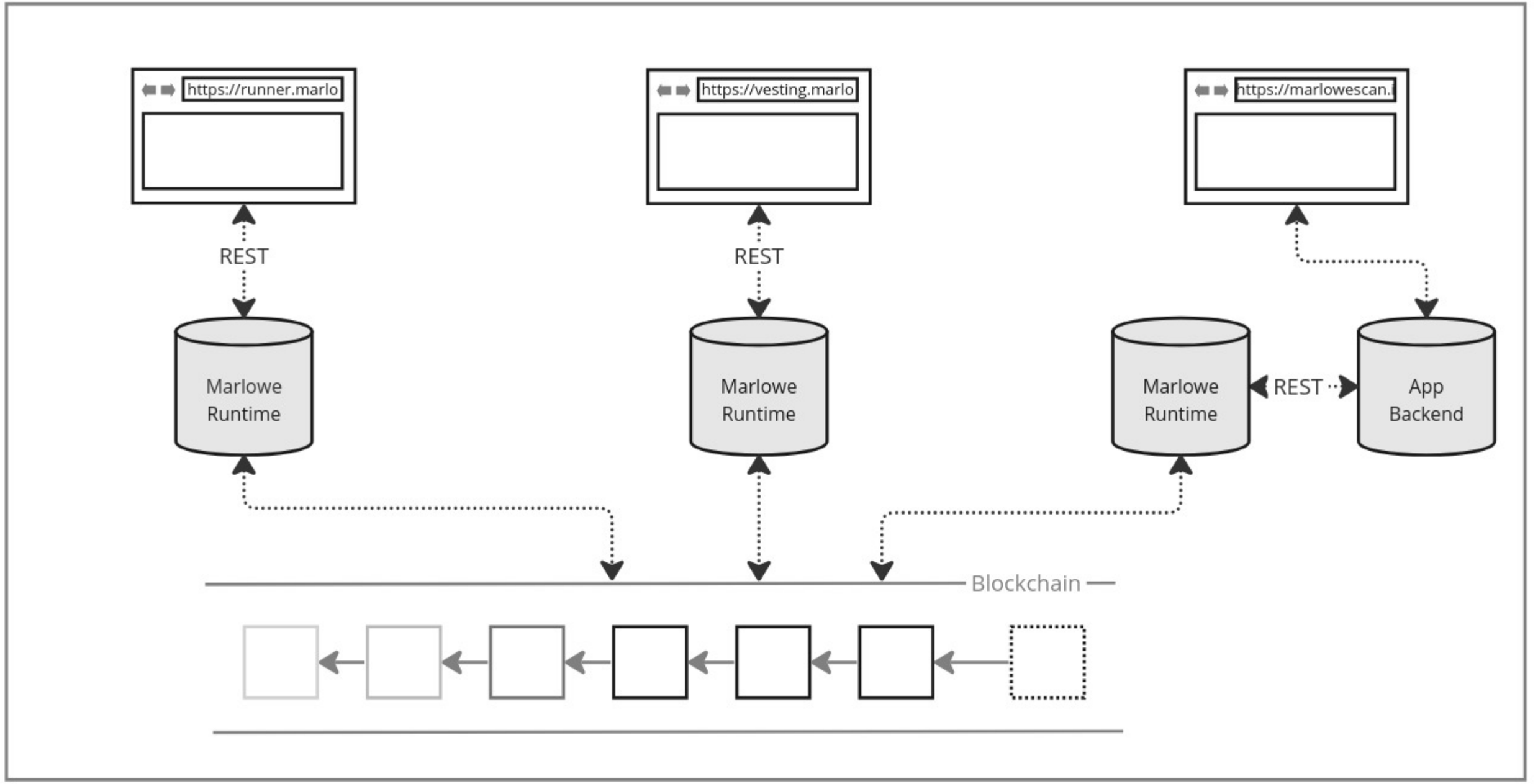
```
export const mkContract = (donor: Address, acceptor: Address): Contract => {
  const inTwentyFourHours = datetoTimeout(new Date(Date.now() + ...
  return {
    "when": [
      {
        "then": "close",
        "case": {
          "party": donor,
          "deposits": 100n,
...
    "timeout_continuation": "close",
    "timeout": inTwentyFourHours,
  }
};
```

- Full exercise
- marlowe-ts-sdk

Dynamic Contract Generation

- TS/JS: github.com/input-output-hk/marlowe-ts-sdk
- Rust: github.com/OlofBlomqvist/marlowe-rs
- Go: github.com/menabrealabs/marlowe
- Python: github.com/OlofBlomqvist/marlowe-py
- C# and more: projectcatalyst.io/funds/10/f10-developer-ecosystem-the-evolution/marlowe-runtime-sdks#about

Marlowe Runtime



Blockchain interaction

- Creation

```
1    await runtimeLifecycle.contracts.createContract({ contract, tags });
```

- Execution

```
1    const depositRequest = {inputs: [deposit]};
2    await runtimeLifecycle.contracts.applyInputs(contractId, depositRequest)
```

Marlowe and Your Dev Team

- Marlowe itself is small, safe and easy to learn
- Marlowe can be used from any programming language (Go, Rust, C#, Python, TypeScript)
- Marlowe blockchain interaction is provided through simple REST API

Marlowe and Your Bussines

- Low entry / prototyping cost
- Quick delivery - when problem fits Marlowe
- Easy to develop - UTxO agnostic, REST, online tools.
- Easy to maintain - no deep dev expertise required

DApps Prototypes

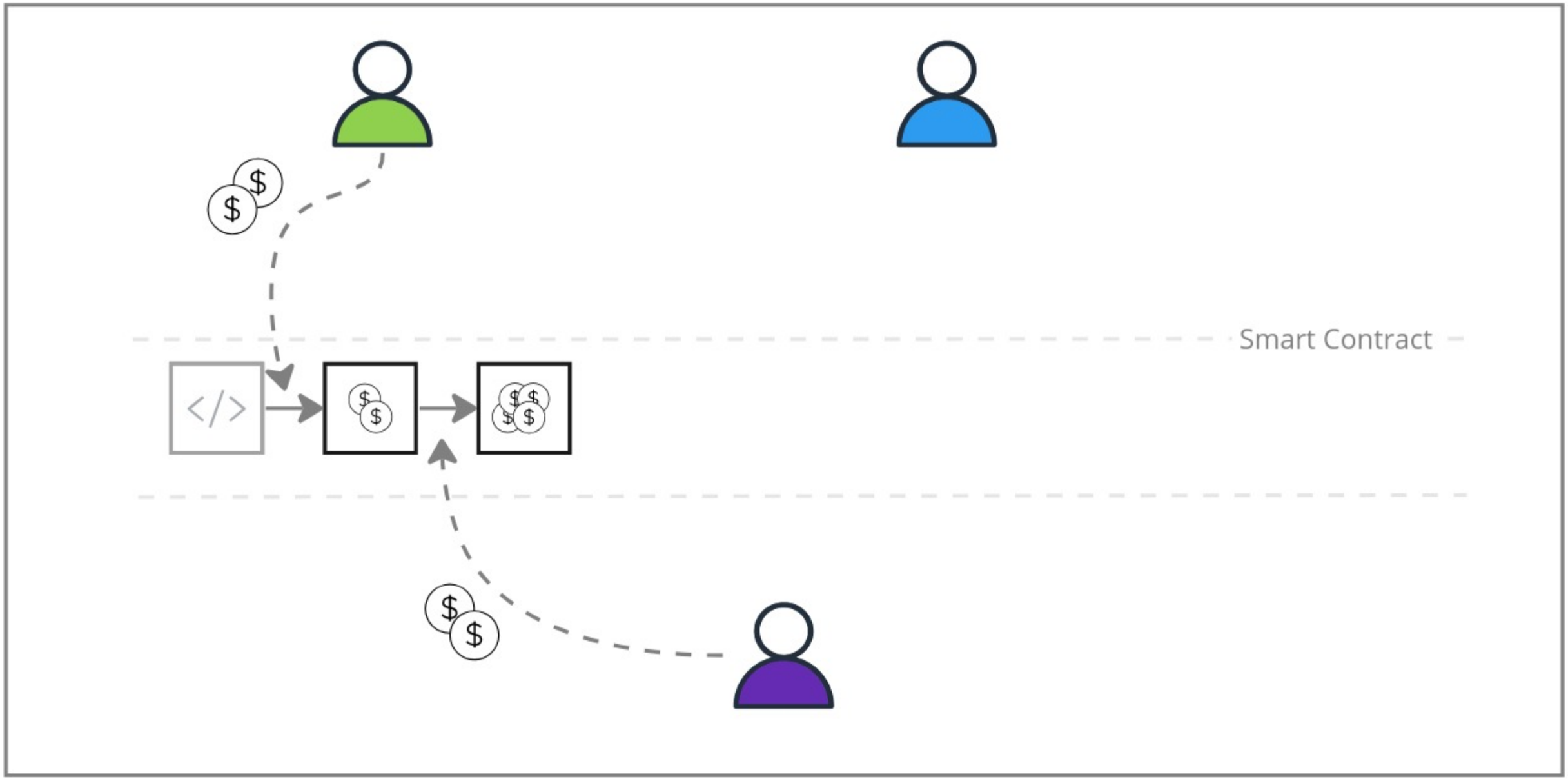
- Vesting app: vesting-preprod.scdev.aws.iohkdev.io
- Marlowe Runner: preprod.runner.marlowe.iohk.io
- Marlowe Scanner: marlowescan.com
- Workshop DApp: github.com/paluh/marlowe-workshop-dapp

Thank you

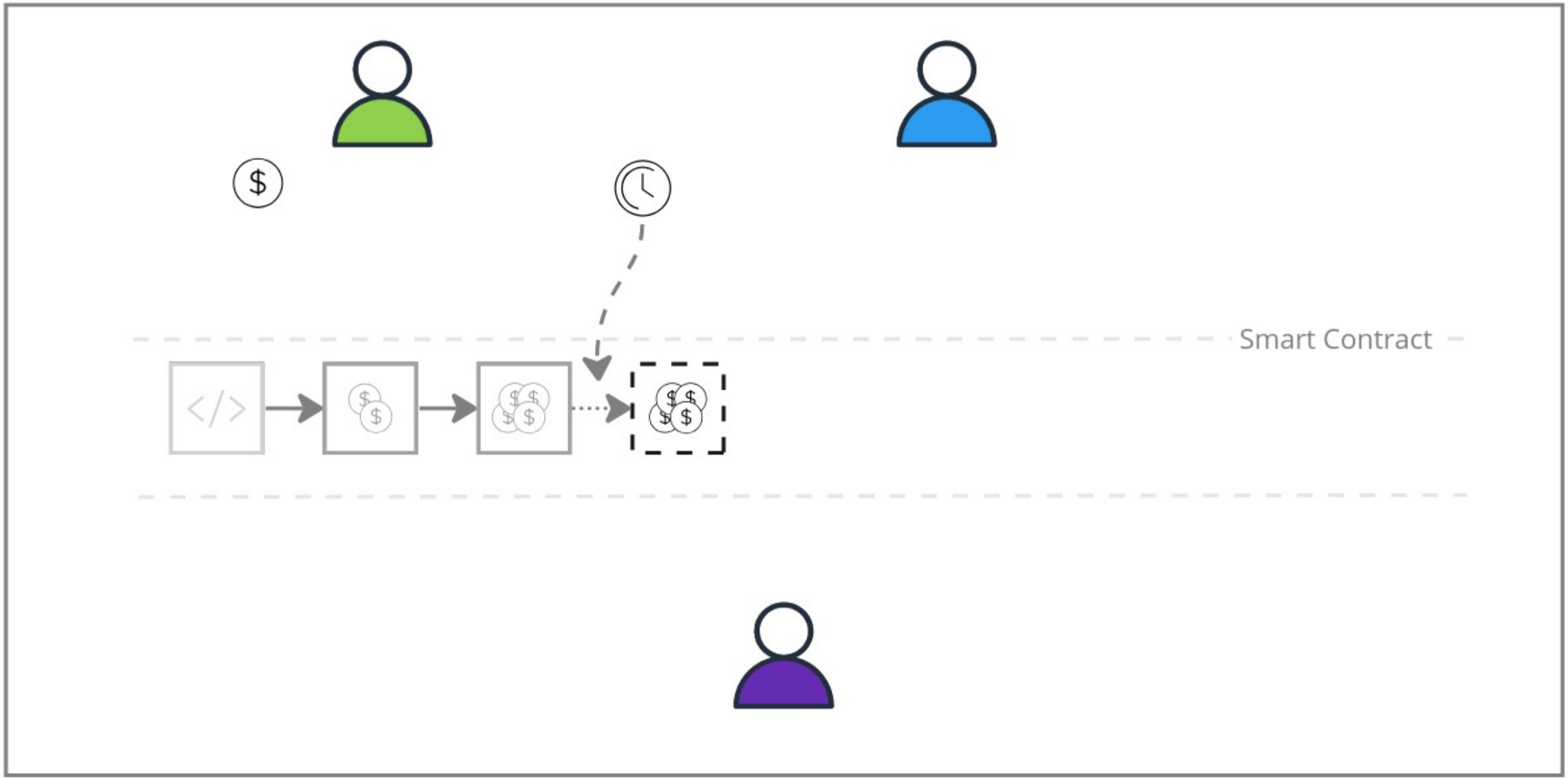
What are Smart Contracts?

Smart Contract

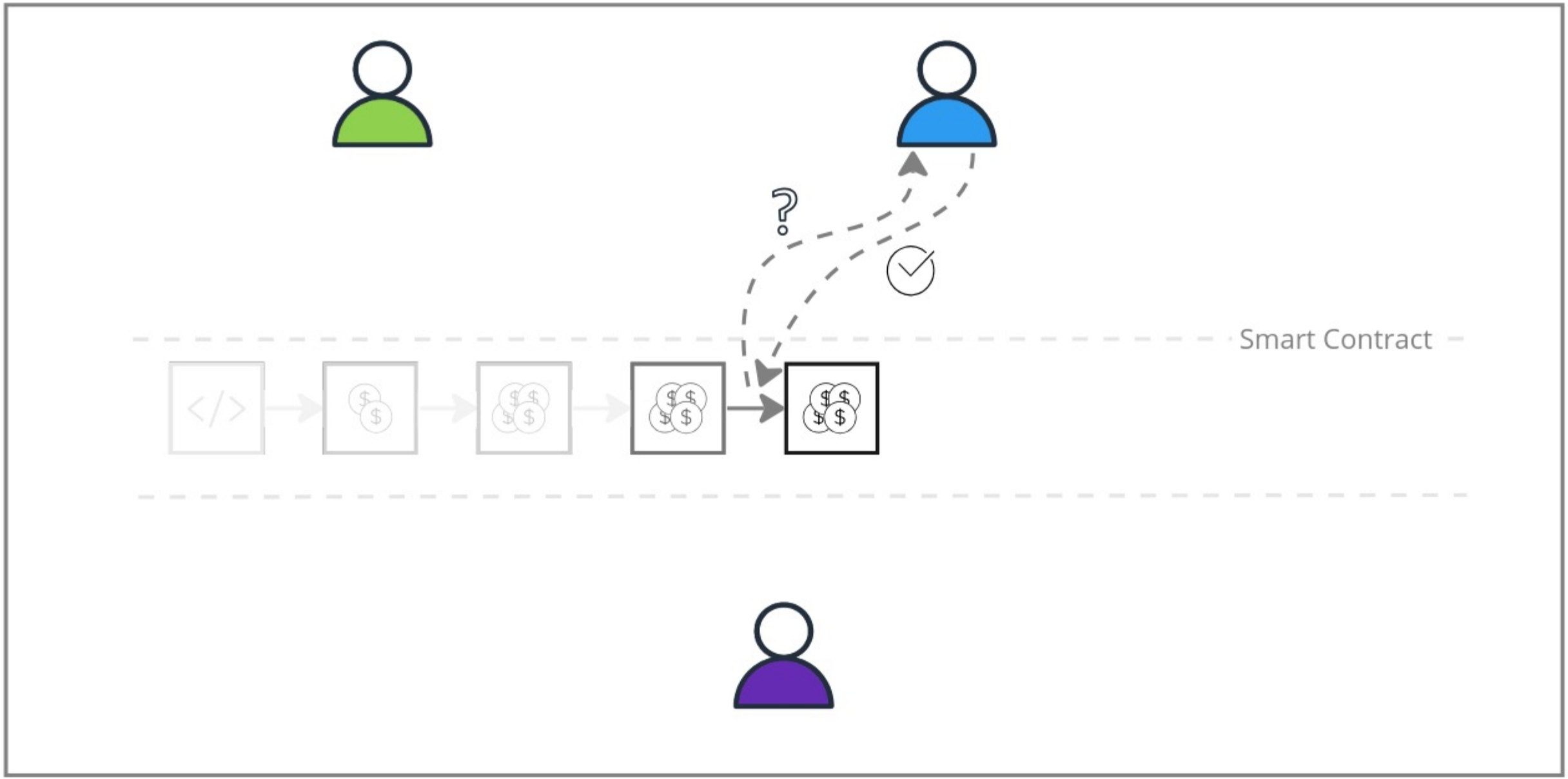
What are Smart Contracts?



What are Smart Contracts?



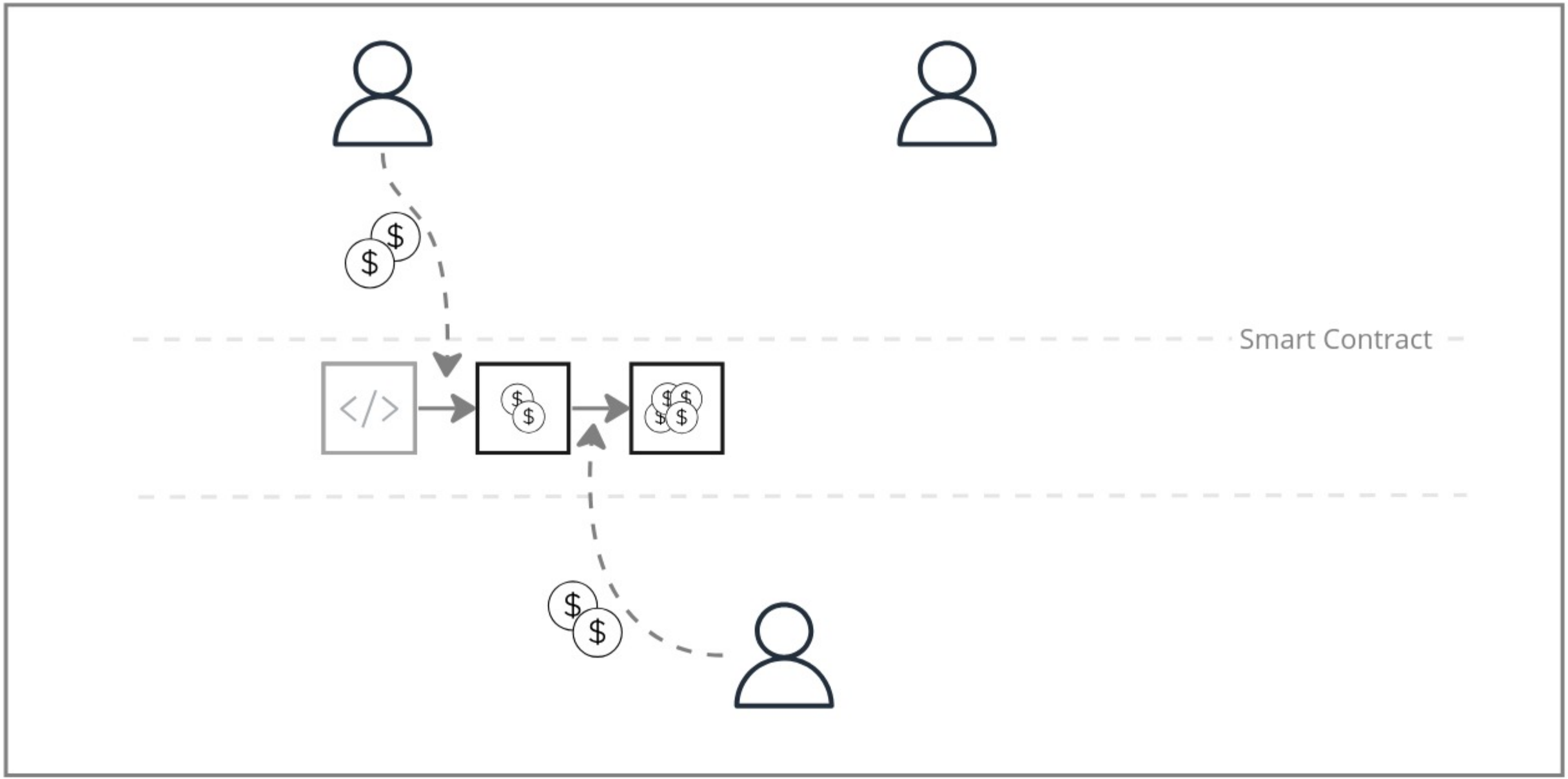
What are Smart Contracts?



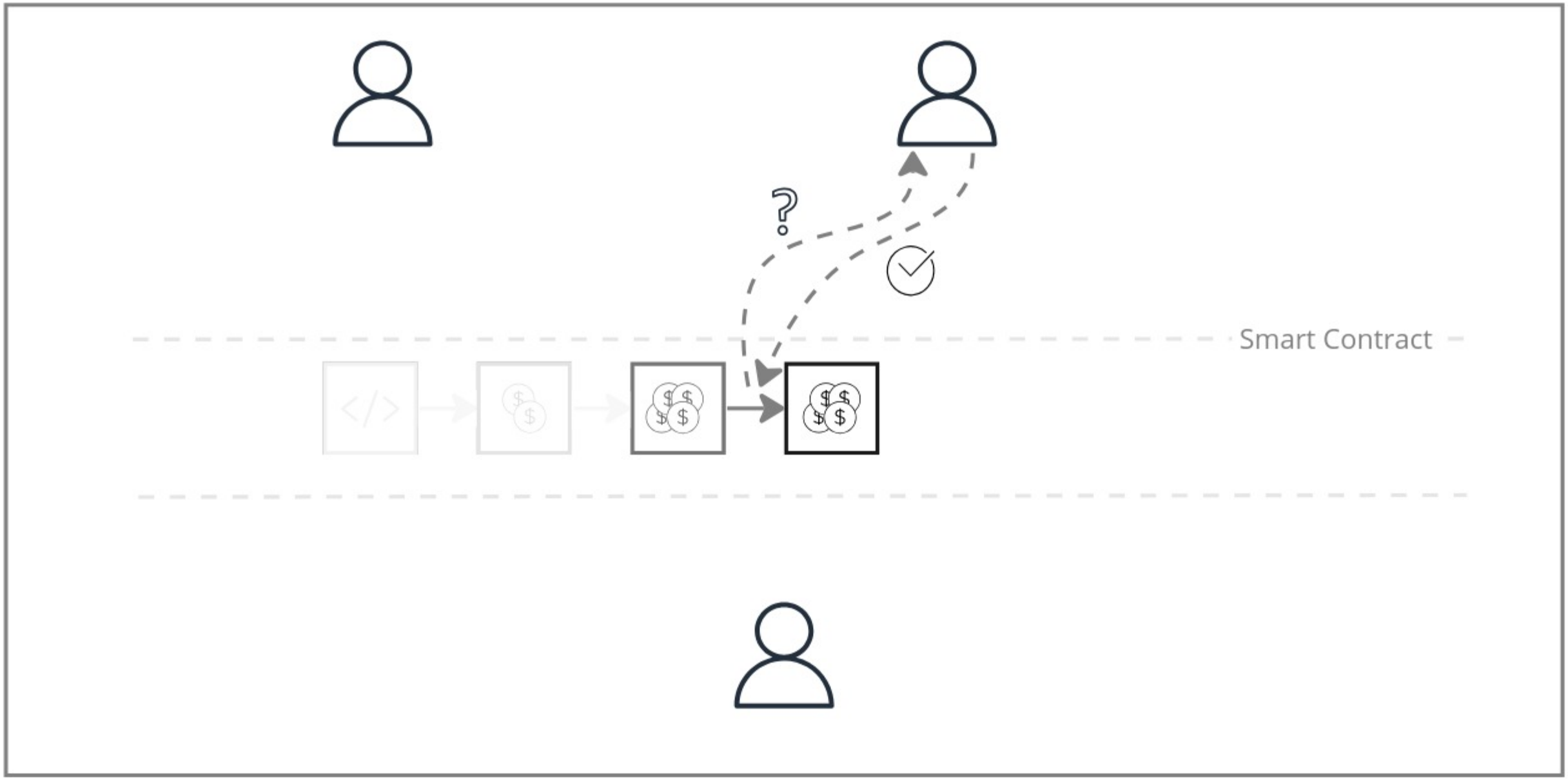
Example: "On-Chain Betting"

Smart Contract

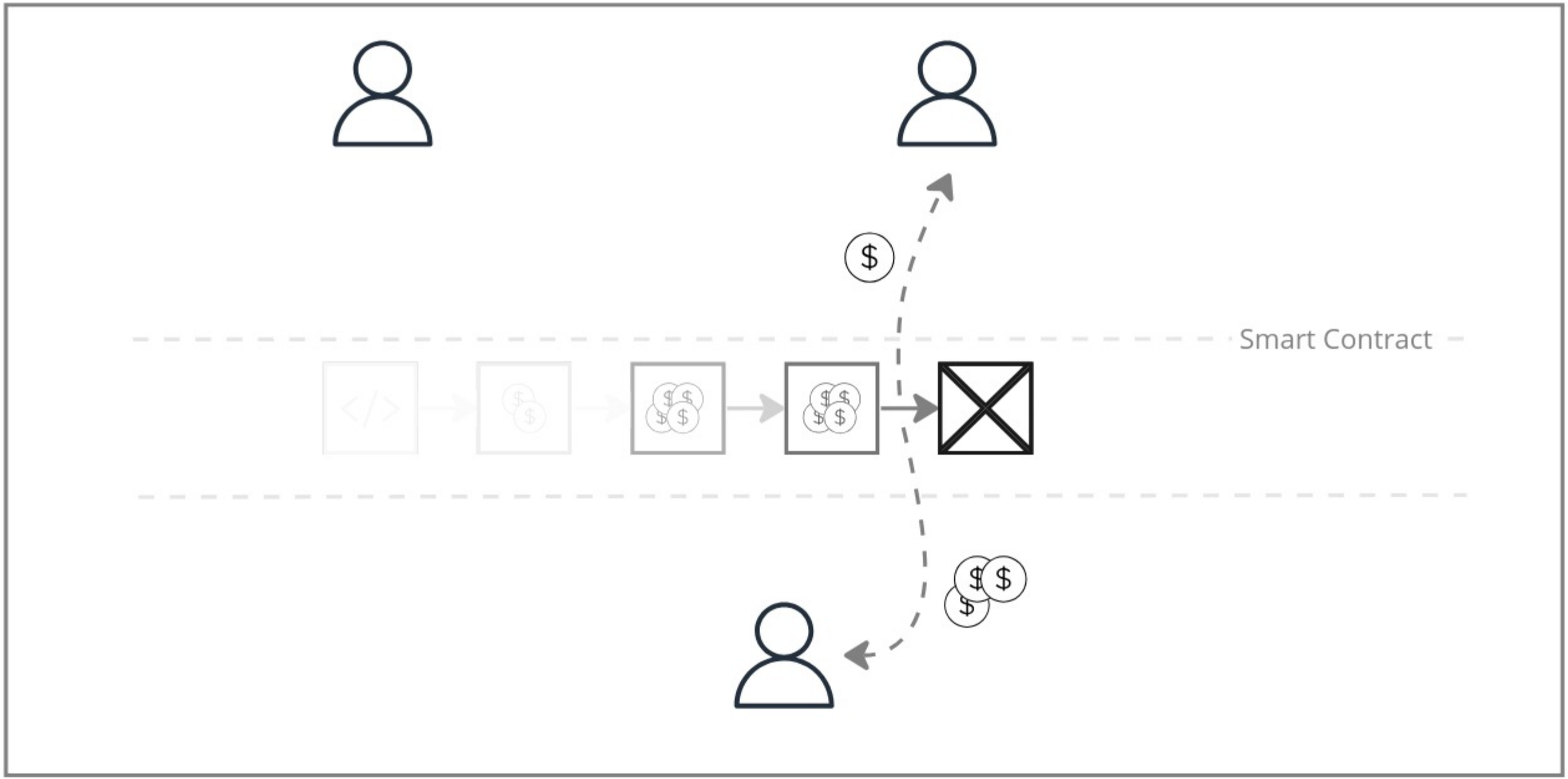
Example: "On-Chain Betting"



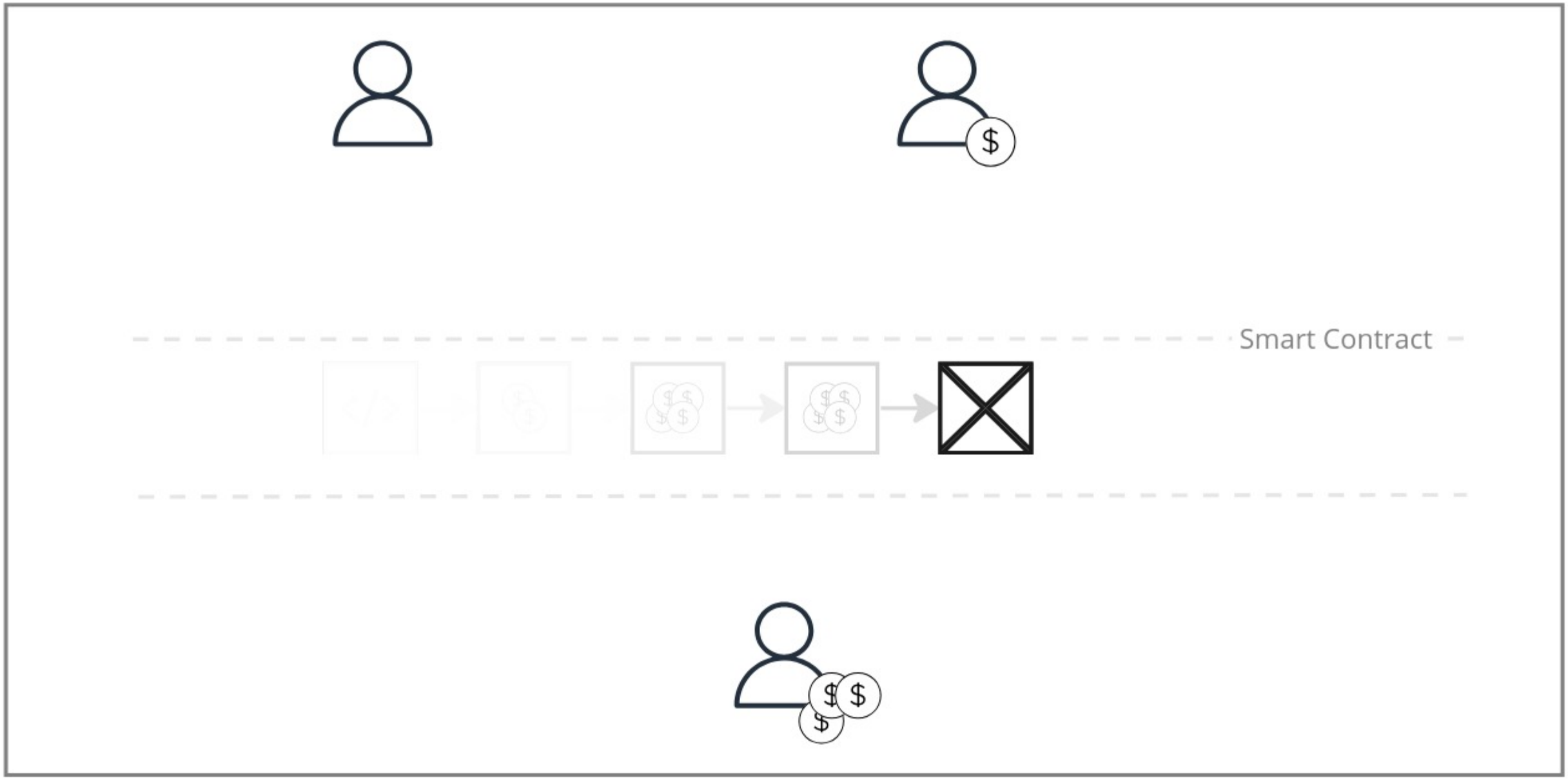
Example: "On-Chain Betting"



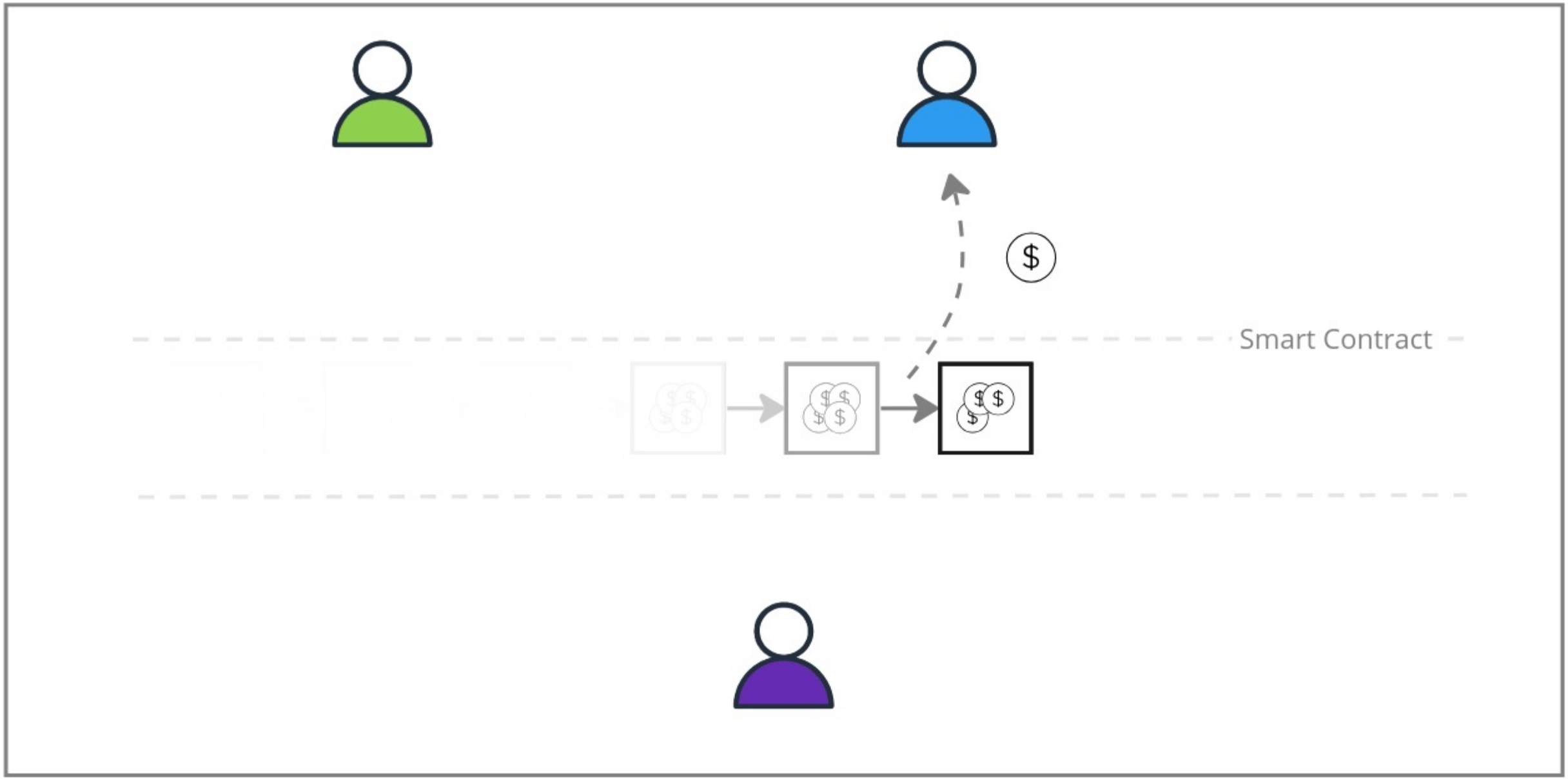
Example: "On-Chain Betting"



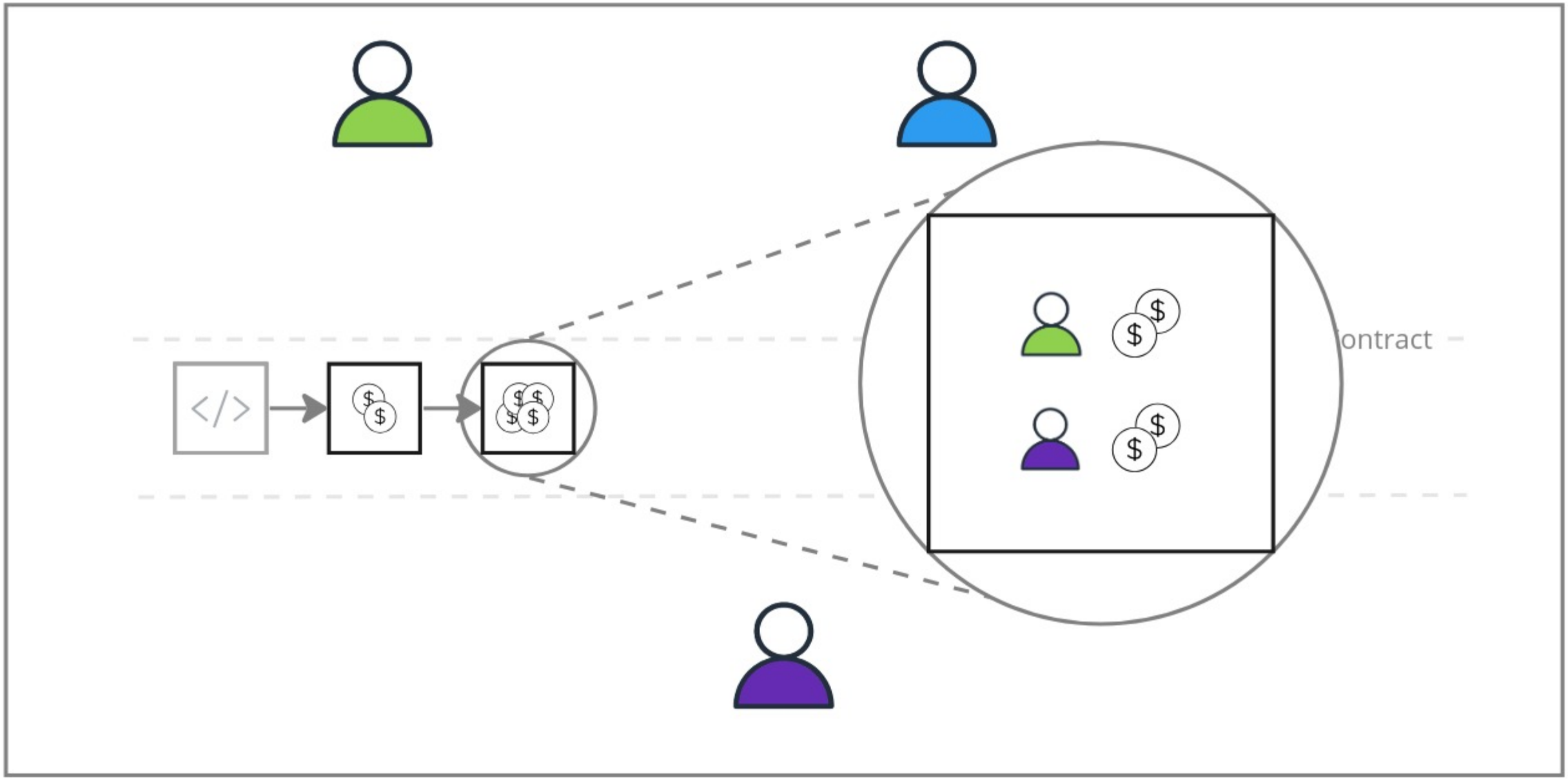
Example: "On-Chain Betting"



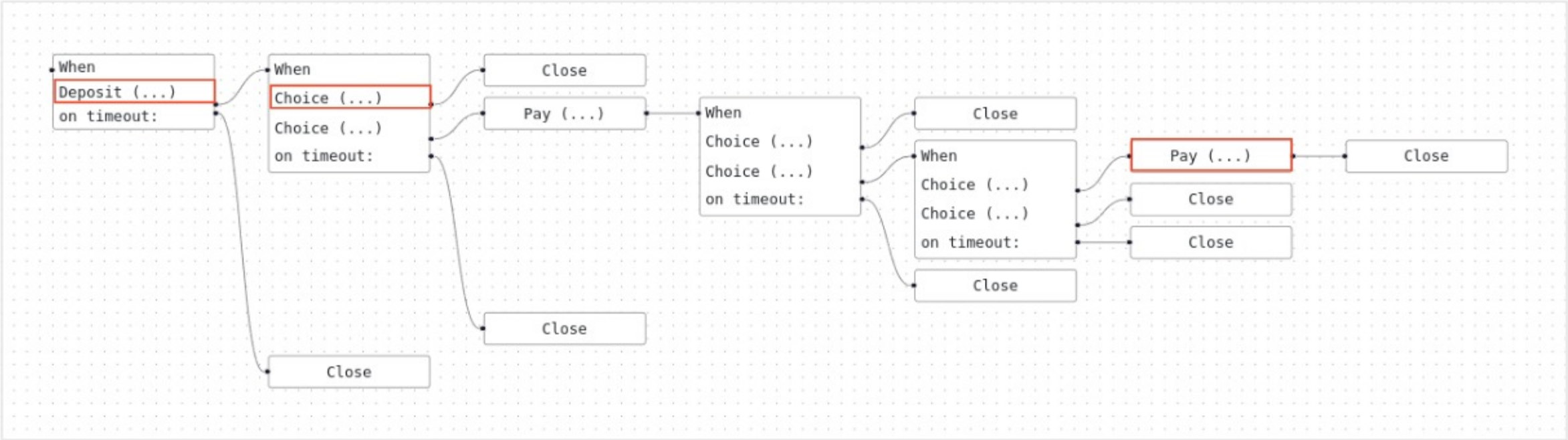
What are Smart Contracts?



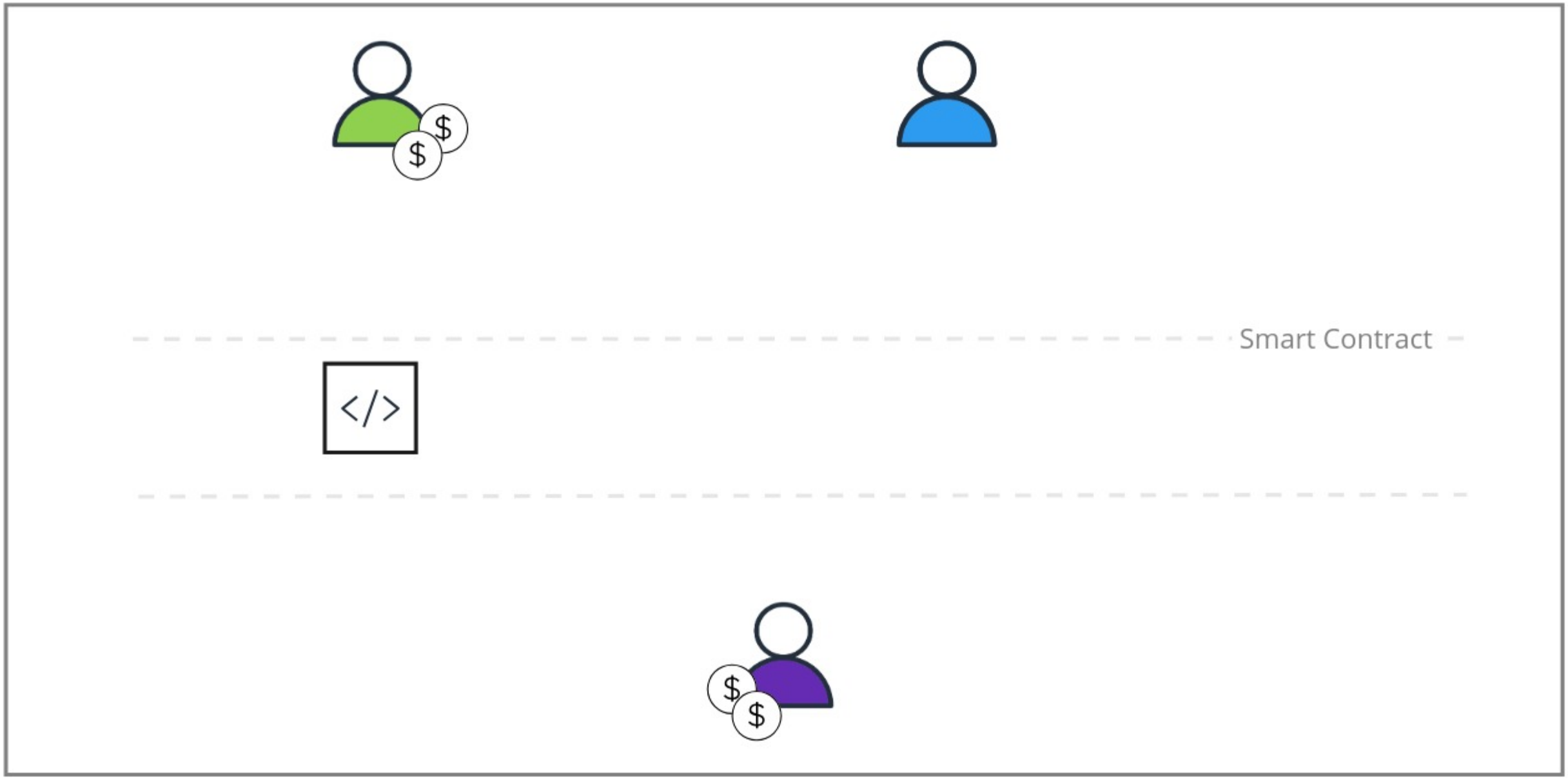
What are Smart Contracts?



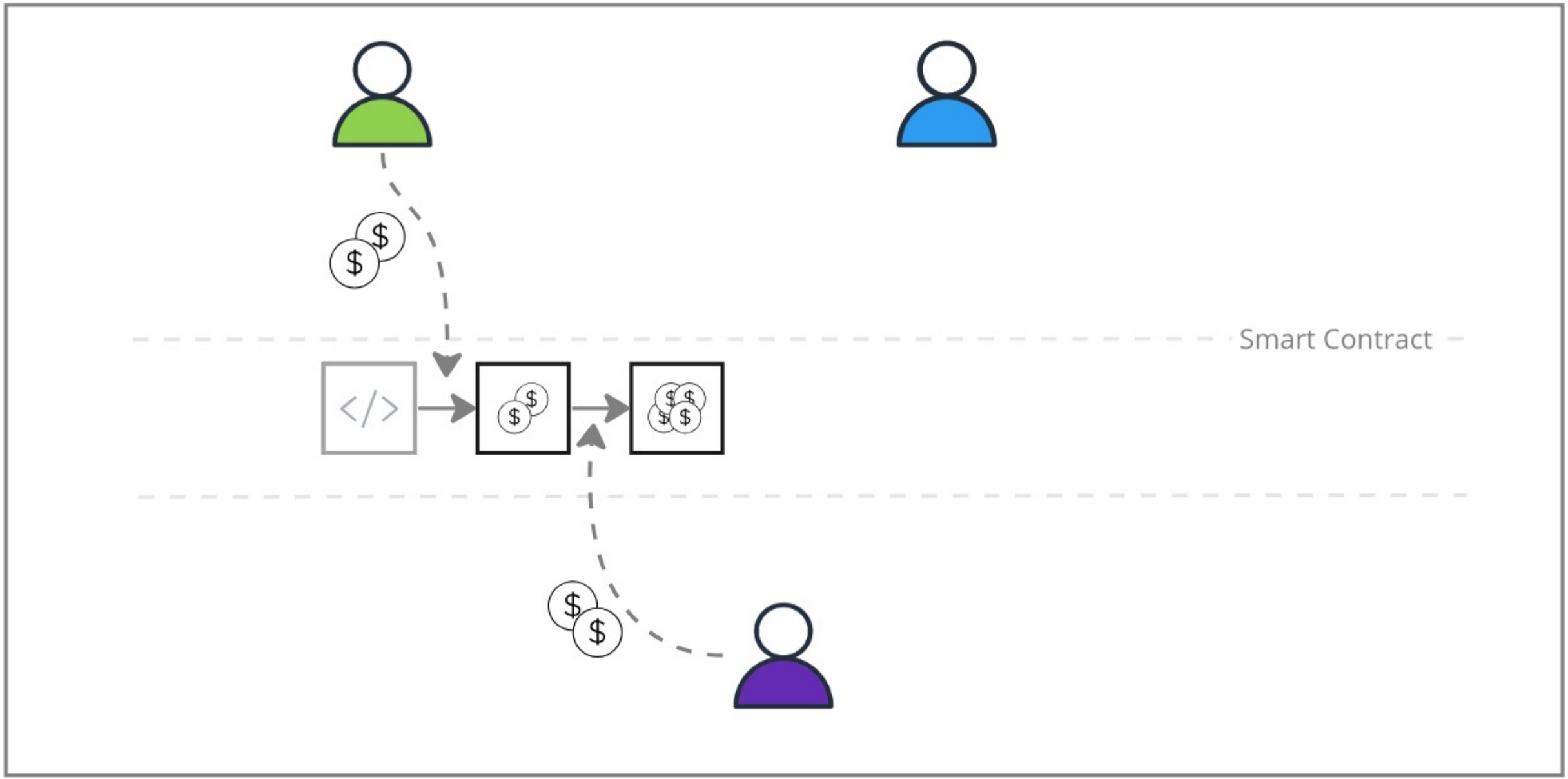
Anatomy of Marlowe



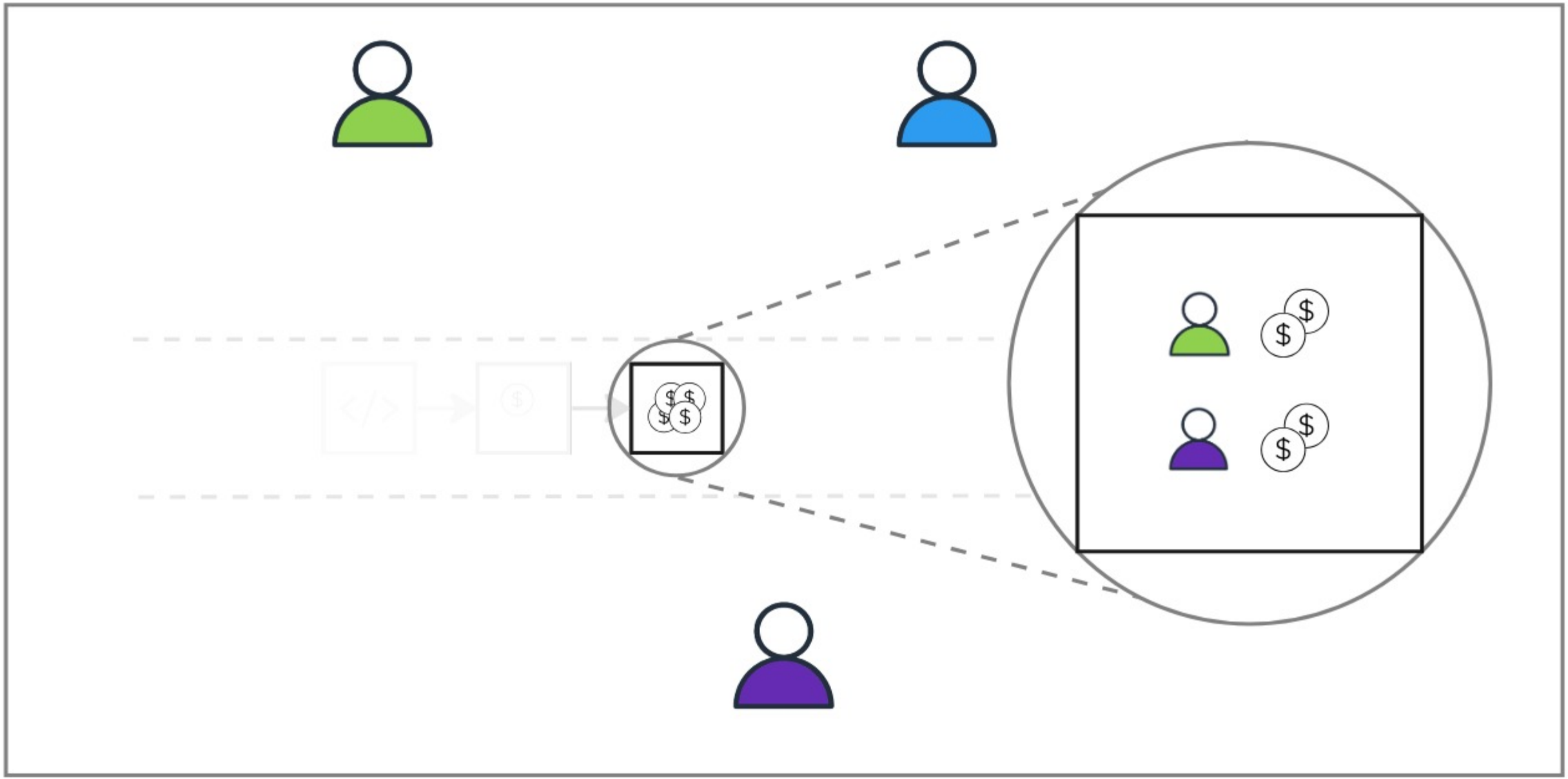
"Deposit" Locks The Assets



"Deposit" Locks The Assets



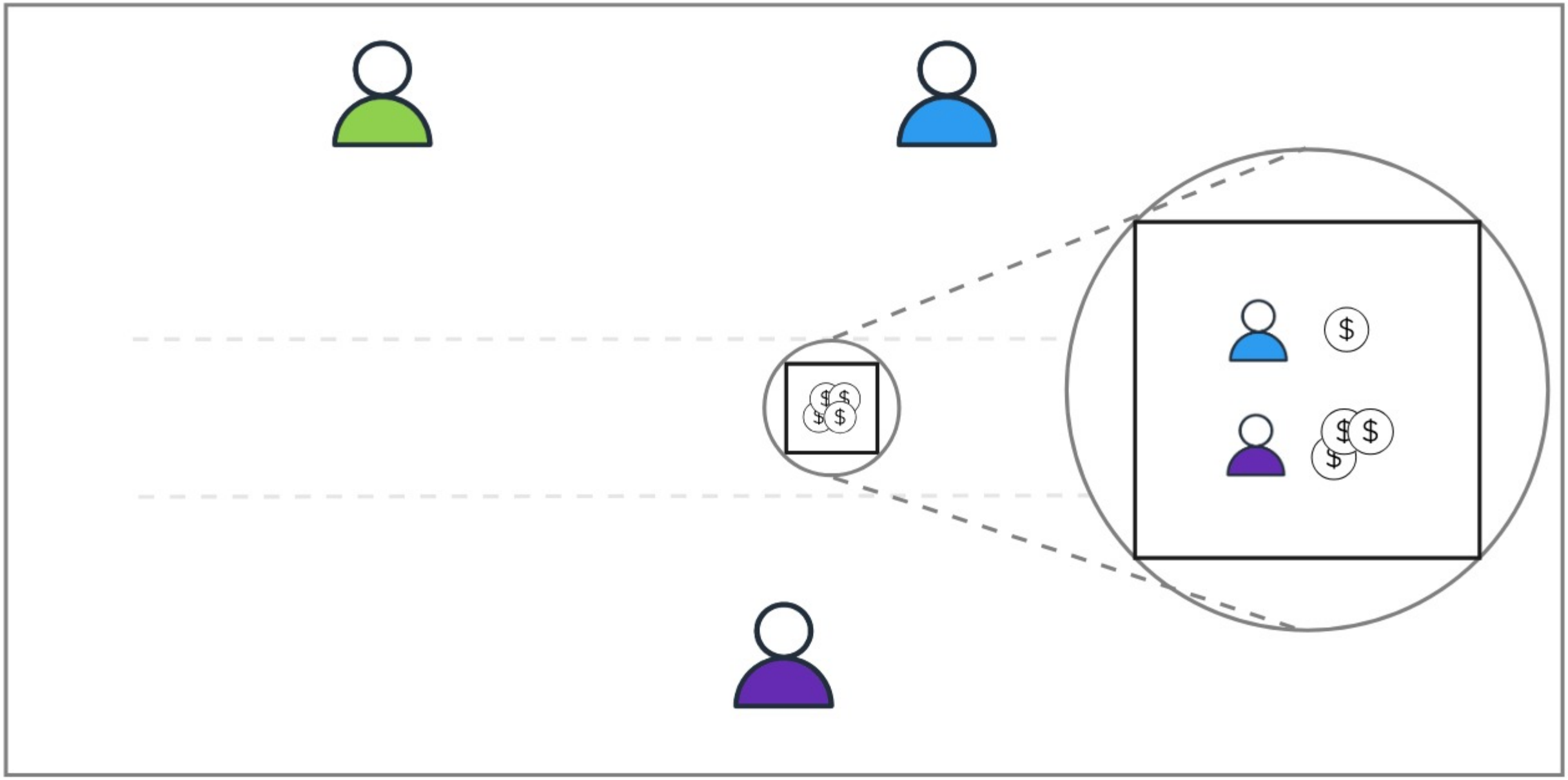
"Deposit" Locks The Assets



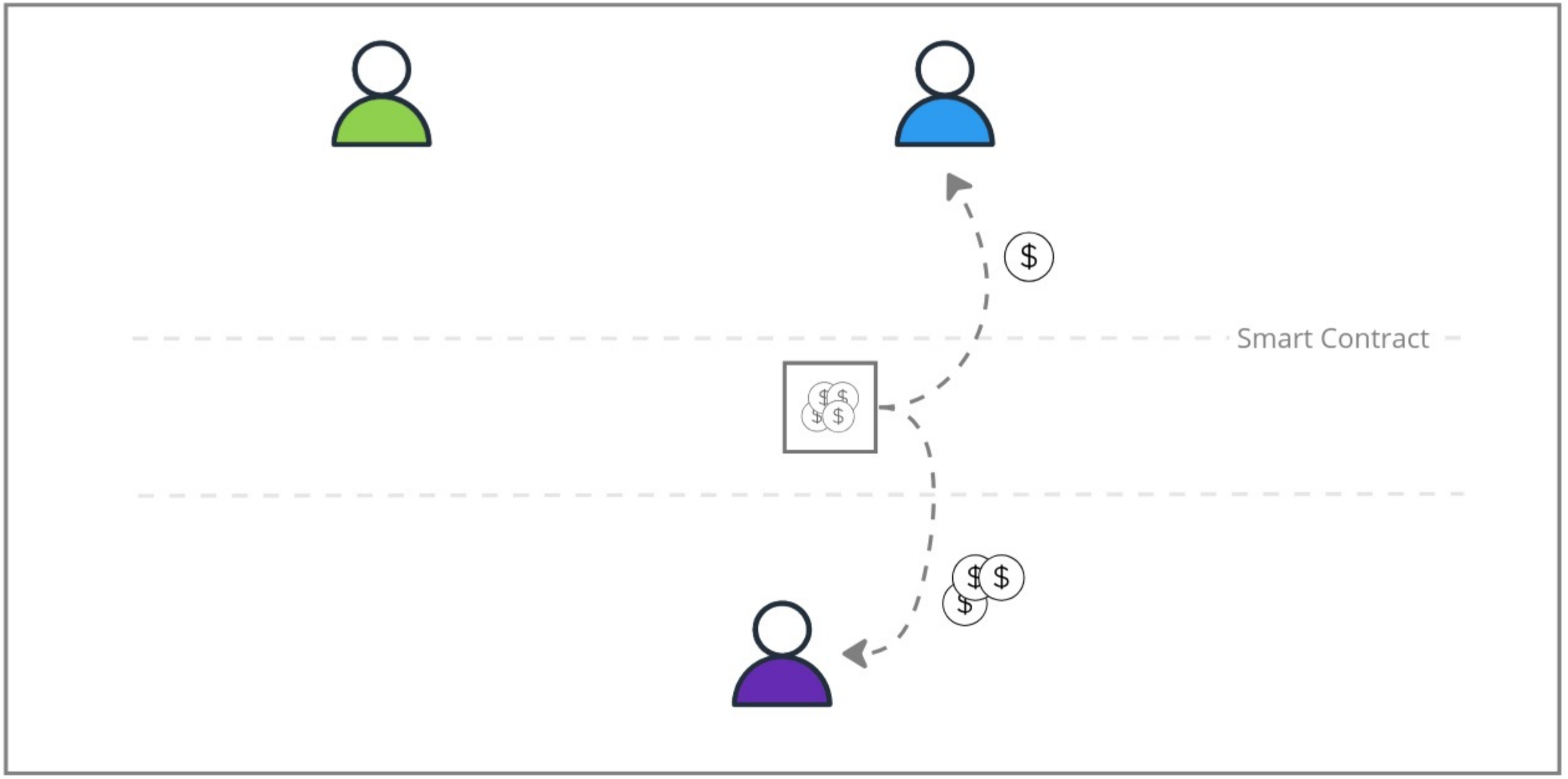
"Close" Releases The Assets

- When we reach "Close" contract releases remaining assets
- The assets are paid out according to account state

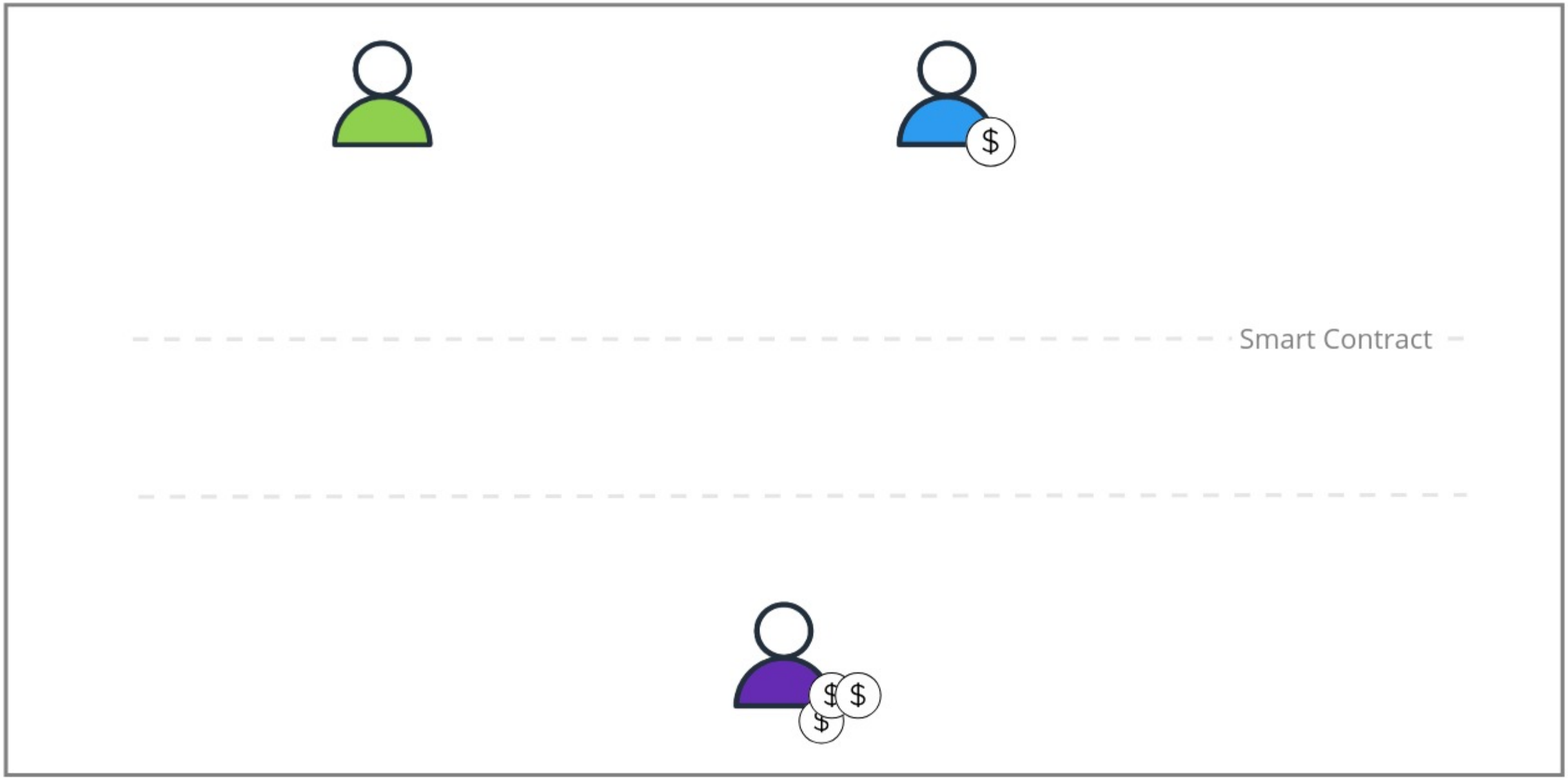
"Close" Releases The Assets



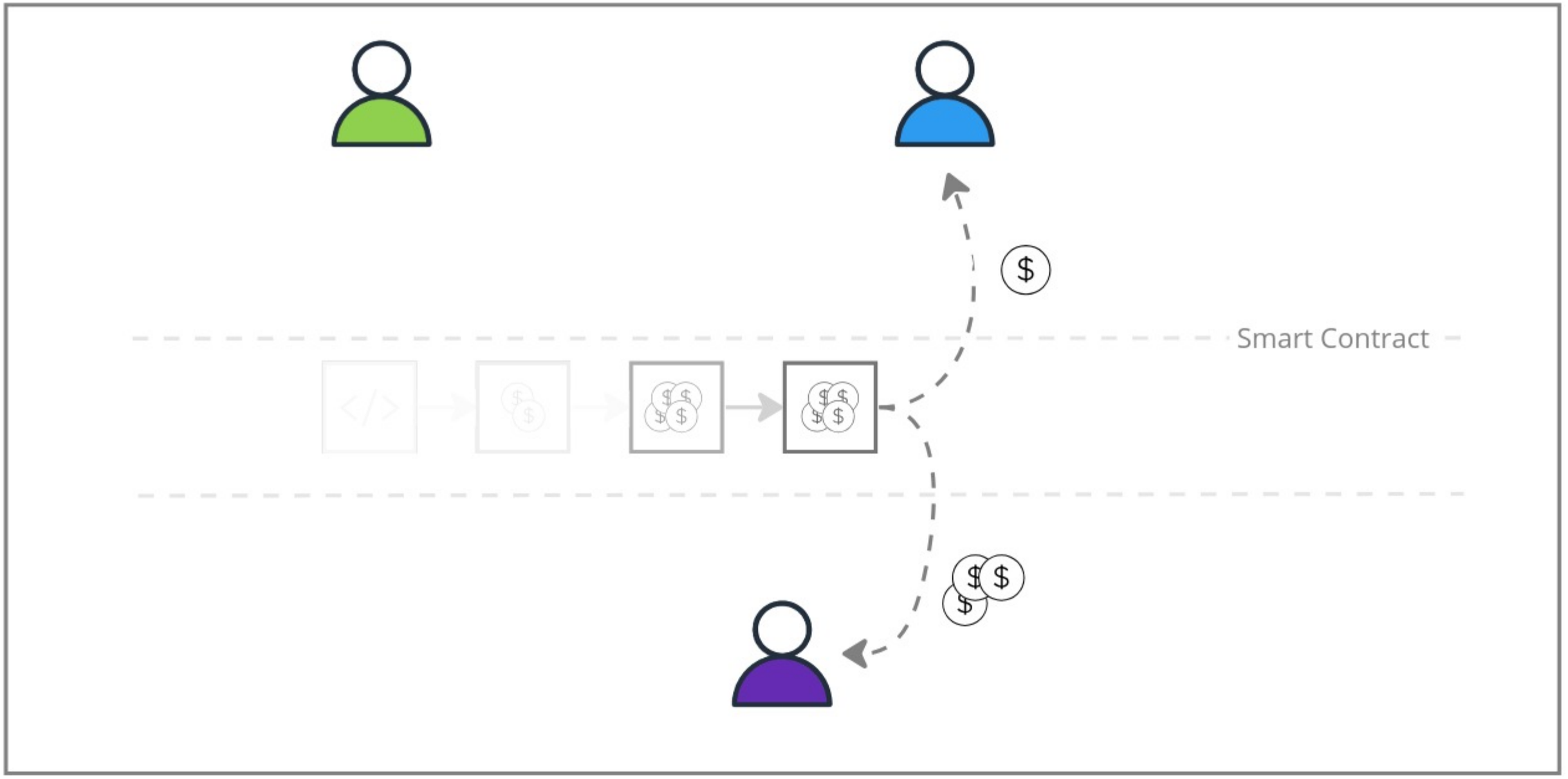
"Close" Releases The Assets



"Close" Releases The Assets



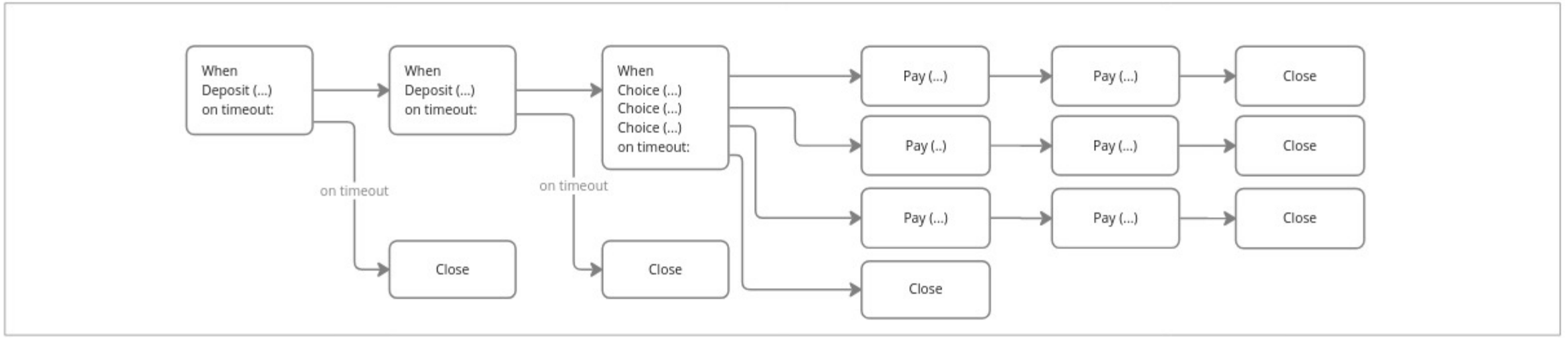
Example: "On-Chain Betting"



On-Chain Betting Benefits

- No up front deposits / account charging etc.
- Predictable cash flow
- Global reaching service
- Everything is transparent
- Arbiter (oracle) is the only trusted party in this setup
- Arbiter has reputation (whole business) at stake

Example: "On-Chain Betting"



play.marlowe.iohk.io