Transactions

Topics

- Basics of a transaction.
- Transferring custody via the locking of coins.
- Unspent Outputs (UTXO) and the chain-state.
- Change Outputs and Fees.
- Evolution of transaction types and scripts.
- Life-cycle of a Transaction, from created to confirmed.

Spooky Language

- The Mempool
- UTXO / Chain-state
- P2PKH, P2SH, P2W-PKH/SH, P2TR
- Base58 / Bech32
- Segregated Witness
- OP_DUP OP_HASH160 <PKH> OP_EQUALVERIFY OP_CHECKSIG

What is a Transaction?

- Transactions represent a transfer of value on the blockchain.
- A transaction can have multiple inputs and outputs.
- Each output stores a number value and locking script.
- Each input provides keys to unlock an output from a previous transaction.

Basics of a Transaction

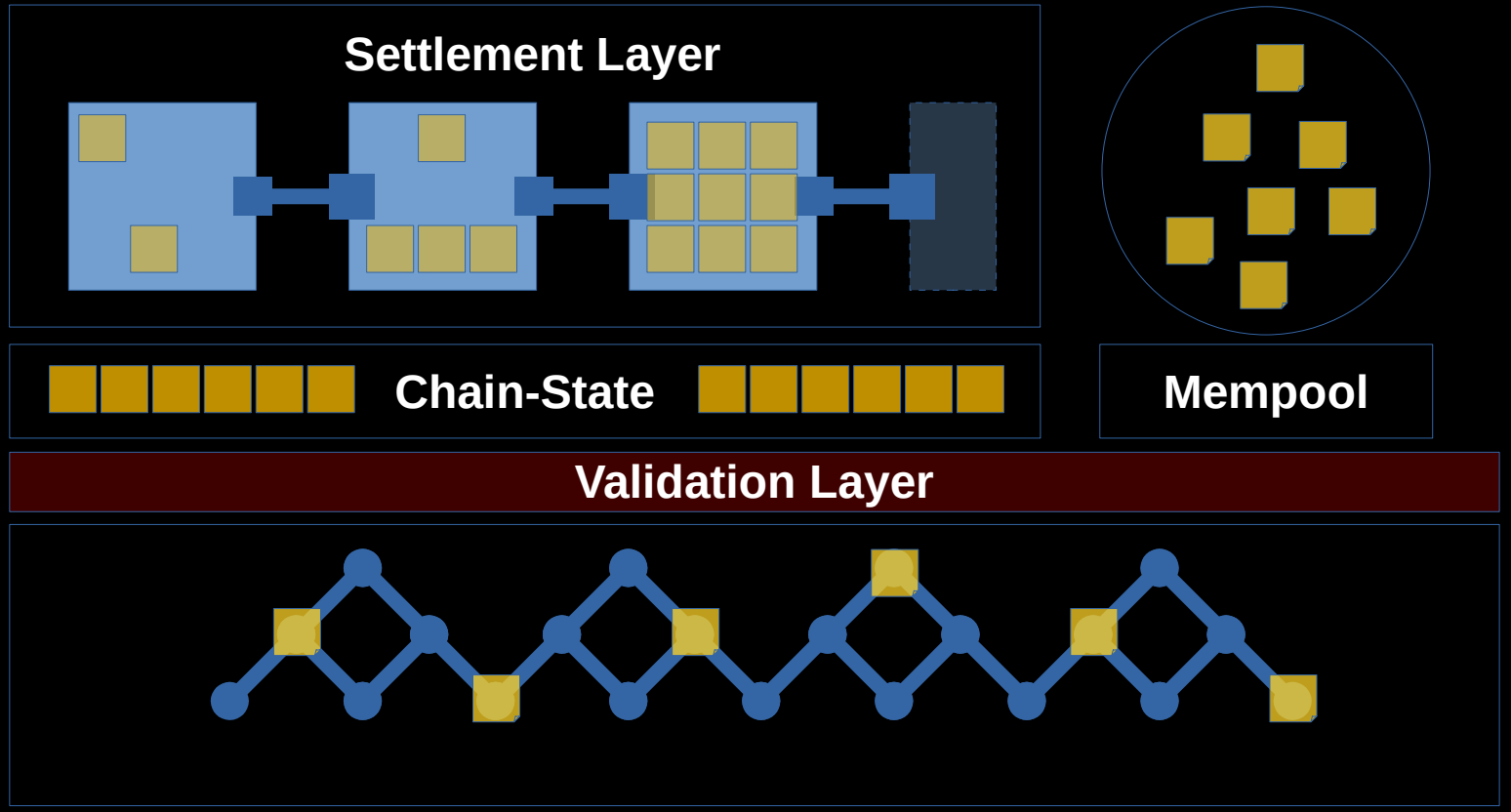
- Version
- Inputs
- Outputs
- Locktime

```
"version": 1
"vin": [
  {
    "txid": a1b2c3d4e5f6
    "vout": 0
    "scriptSig": [ signature, pubkey ]
    "sequence": 0xFFFFFFFF
  }
]
"vout": [
  {
    "value": 100 000 000
    "scriptPubkey": [ locking_script ]
  }
]
"locktime": 800000
```

Coins

- Each unspent transaction output (utxo) stores 8-bytes of value.
- This value can be seen as a spendable coin.
- Transactions can move, join and split coins.
- Coins trace back to their coinbase transaction.

Payment Network



Chain State

- The total collection of spendable coins on the blockchain is known as the chain-state.
- Transactions bid to make permanent changes to this state.
- Blocks settle and commit these changes.
- Non-spendable coins are also removed.

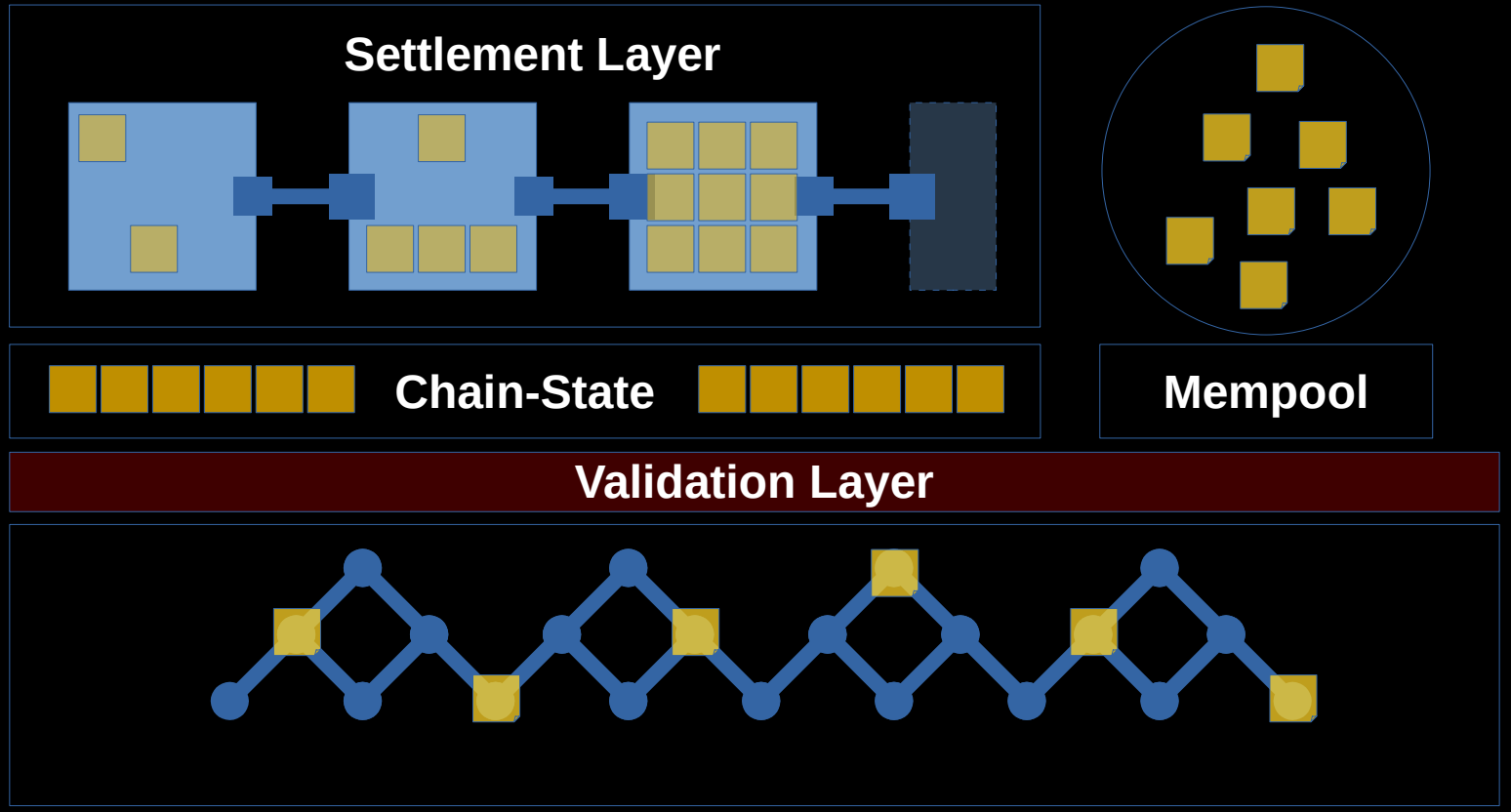
Mempool

- To save bandwidth, relayed transactions are cached.
- Nodes use this cache to validate incoming blocks.
- Nodes can request missing transactions from their network peers.
- The collective state of unconfirmed transactions on the network is known as the Mempool.

Settlement

- When a block is accepted, its transactions become settled.
- Each new block buries these transactions in computational work.
- Settled transactions have less validation rules.

Payment Network



Unlocking an Output

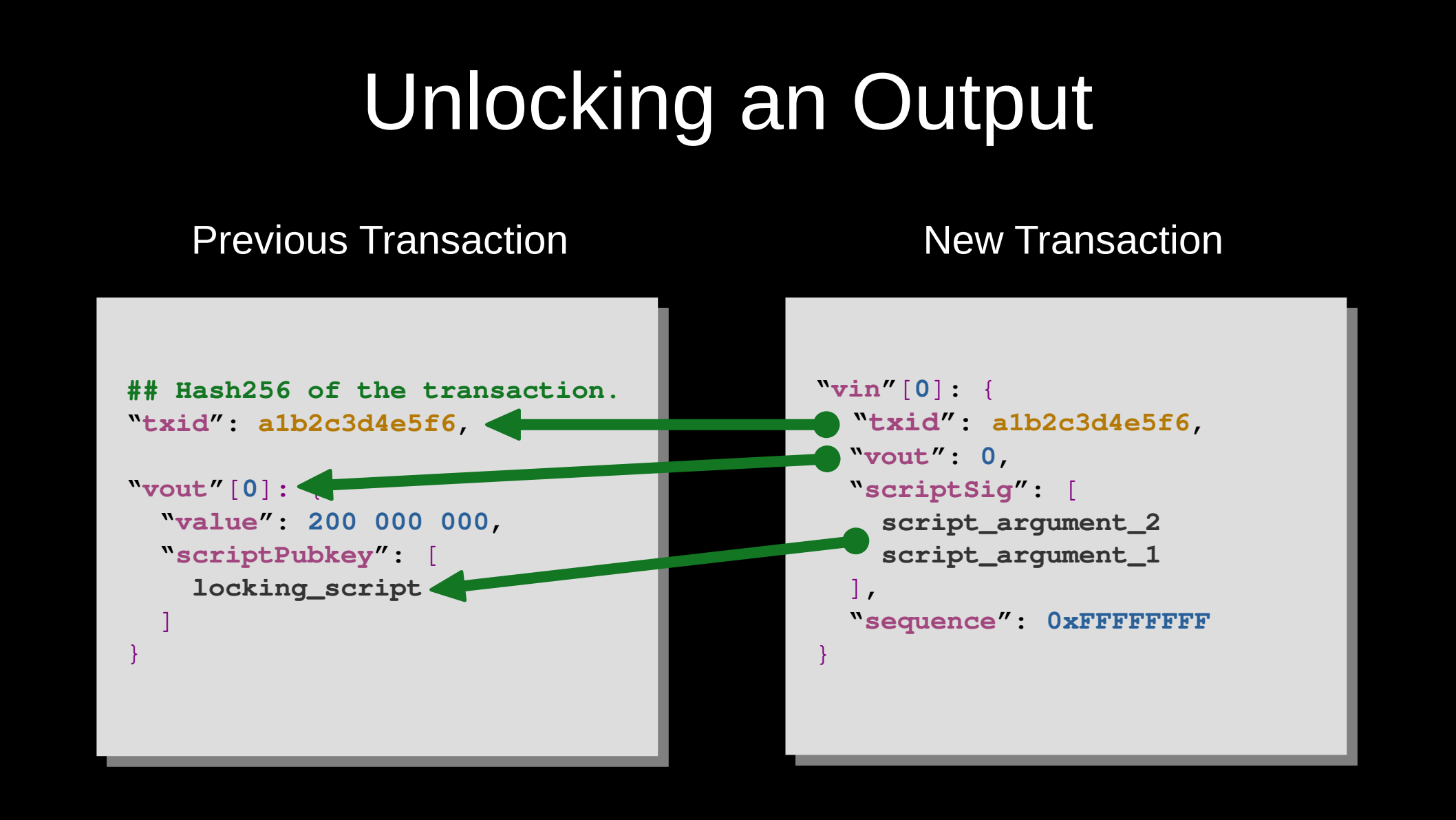
Previous Transaction

```
## Hash256 of the transaction.
"txid": a1b2c3d4e5f6,

"vout"[0]: {
  "value": 200 000 000,
  "scriptPubkey": [
    locking_script
  ]
}
```

New Transaction

```
"vin"[0]: {
  "txid": a1b2c3d4e5f6,
  "vout": 0,
  "scriptSig": [
    script_argument_2
    script_argument_1
  ],
  "sequence": 0xFFFFFFFF
}
```



Lock and Unlock

Custody Transfer of Coins

- Alice wants to send some coins to Bob.
- Bob provides Alice with a script to lock the coins.
- Alice prepares a transaction that:

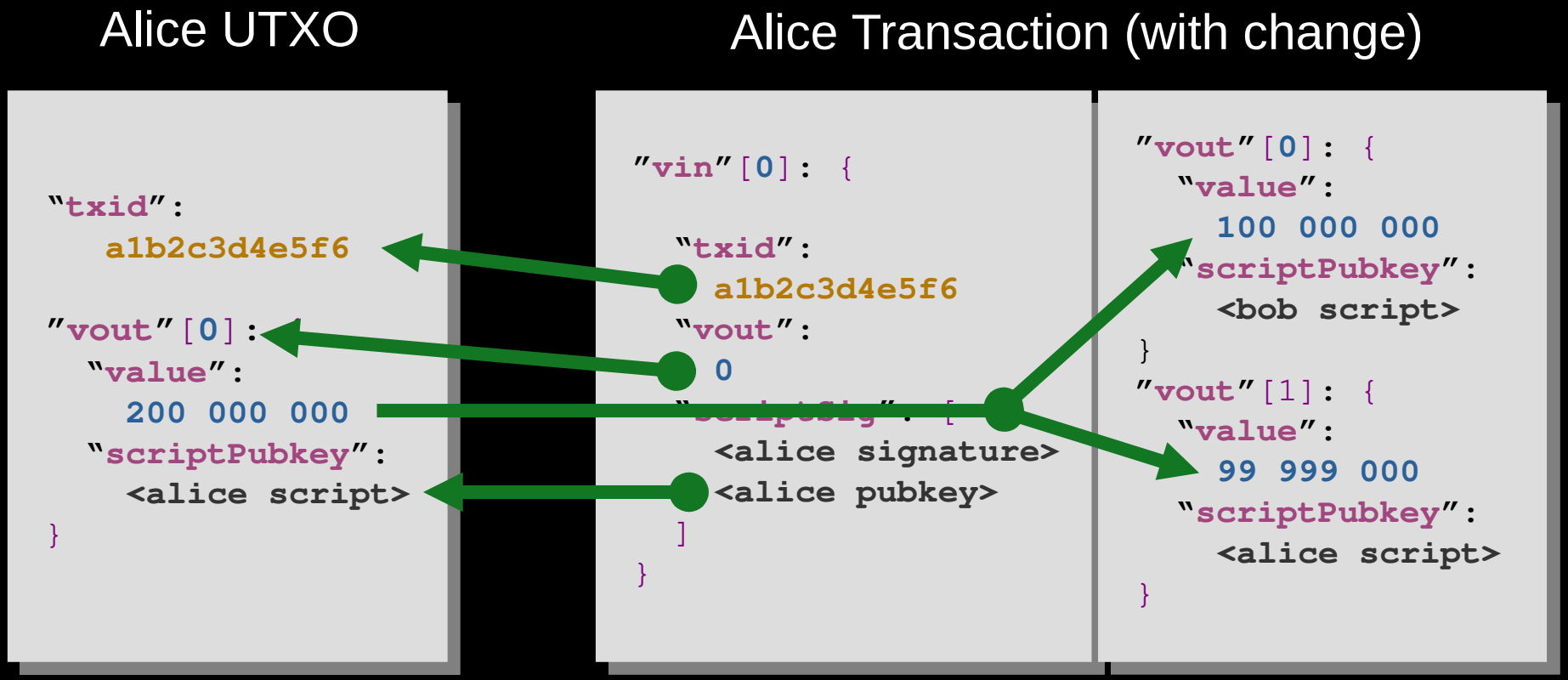
  - Selects some unspent coins.

  - Includes the keys to unlock those coins.

  - Spends x coins to a new output with Bob’s lock.

  - Spends y change to a new output with Alice’s lock.

Alice Sends To Bob



Change Outputs

- Coins cannot be partially spent.
- To pay an exact coin value, spend coins with greater value, return the rest as change.
- This is analogous to spending a large bill, and receiving change in return.

Fees

- Transaction fees are implicit bids for block space.
- Bids equal the spread between outputs minus inputs.
- Miners typically sort their bids by value per byte.
- Most nodes enforce a minimum fee on transactions in order to avoid spam on the network.

Payment Address

A payment address is simply the encoded version of a public key hash or script hash.

```
## Example of Base58 encoding.
“pubkey”  : 020fdeab2468c464bb2914a1d18fdd3b3cb3702890796a52d65ba2e59bd7905d90
“hash”    : 050688649b067200dbd2d402f103903a17b7ec39
“address” : mfyXVvFuXmXwd2iK9QedqR7u3fgRA4tQqx

## Example of Bech32 encoding.
“pubkey”  : 02024764db3bee1269ffd3b7c895a545236a476693dde99029ec28c3f107c91ac63d
“hash”    : 6f5699d0c49fd51f88456711337dc053bc6f2496
“version” : 0   (program script version)
“hrp”     : bc1 (network prefix and encoding version)
“address” : bc1qut0m3ltc2qkhtnxeawxpu28xlfcy5qv9rgv62r
```

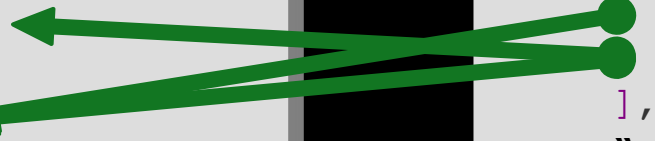
Pay-to-Pubkey Hash (P2PKH)

Previous Output (UTXO)

```
"vout"[0]: {
  "value": 100 000 000,
  "scriptPubkey": [
    OP_DUP,
    OP_HASH160,
    <20 byte hash>,
    OP_EQUALVERIFY,
    OP_CHECKSIG
  ]
}
```

New Transaction Input

```
"vin": [
  {
    "txid": a1b2c3d4e5f6,
    "vout": 0,
    "scriptSig": [
      signature
      public_key
    ],
    "sequence": 0xFFFFFFFF
  }
]
```



Pay-to-Script Hash (P2SH)

```
"vout"[0]: [
  {
    "value": 100 000 000,
    "scriptPubkey": [
      OP_HASH160,
      <20 byte hash>,
      OP_EQUAL
    ]
  }
]
```

```
"vin": [
  {
    "txid": a1b2c3d4e5f6,
    "vout": 0,
    "scriptSig": [
      signature,
      public_key,
      redeem_script
    ],
    "sequence": 0xFFFFFFFF
  }
]
```

Old Nodes

Returns TRUE

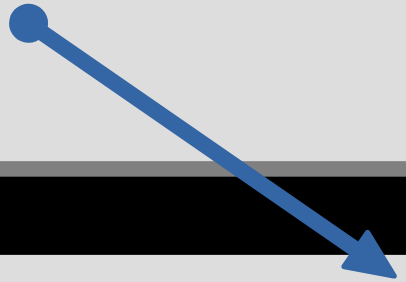
New Nodes (BIP 16)

Evaluate Redeem Script with Arguments

Pay-to-Witness (P2W-PKH)

```
"vout"[0]: {
  "value": 100 000 000,
  "scriptPubkey": [
    ## A 20-byte hash implies
    ## a P2PKH script.
    0, <pubkey hash>
  ]
}
```

```
"vin"[0]: {
  "txid": a1b2c3d4e5f6
  "vout": 0
  "scriptSig": []
  "sequence": 0xFFFFFFFF
  "txinWitness": [
    signature
    public_key
  ]
}
```



```
[ OP_DUP, OP_HASH160, <20 byte hash>, OP_EQUALVERIFY, OP_CHECKSIG ]
```

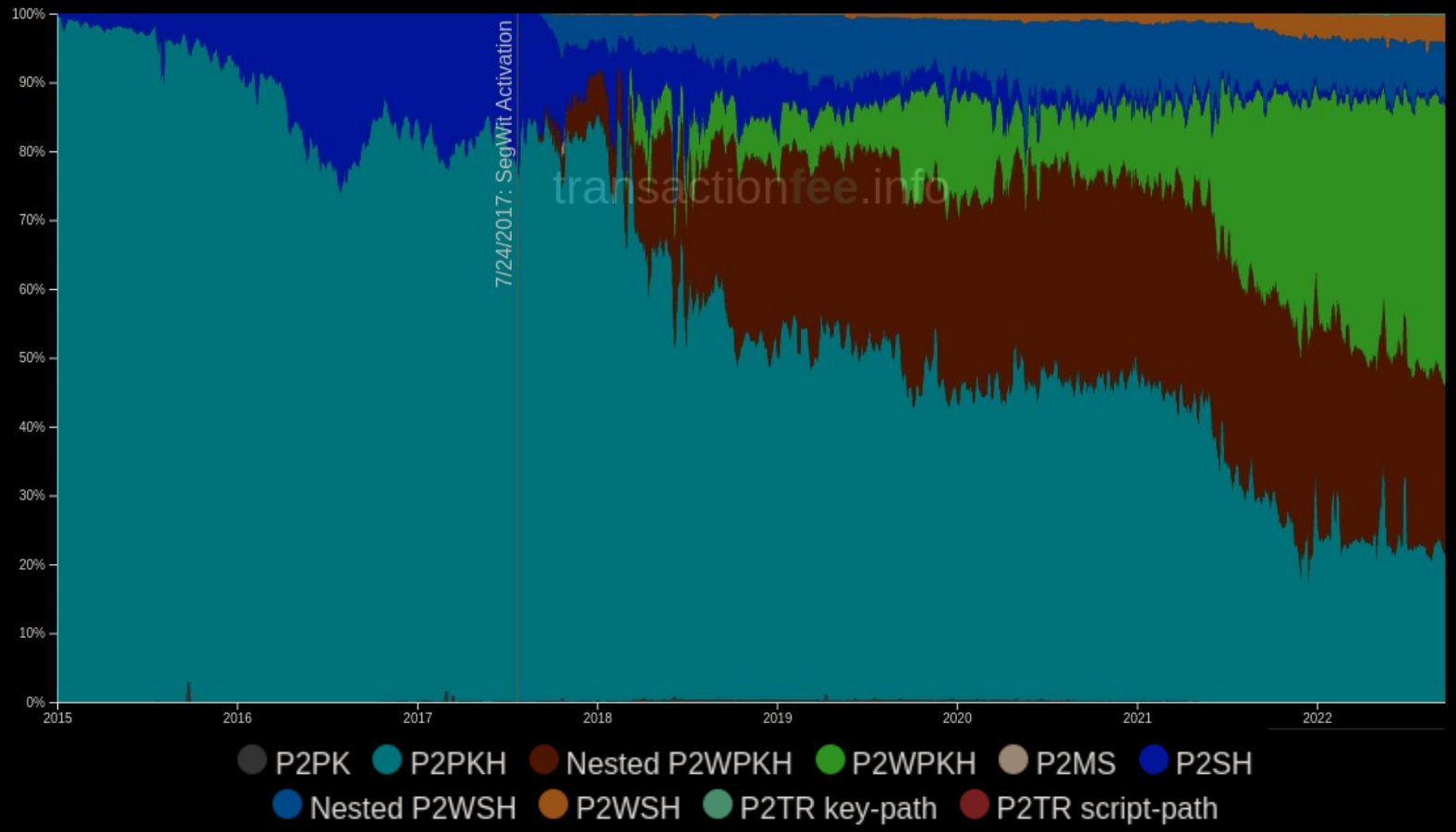
Pay-to-Witness (P2W-SH)

```
"vout"[0]: {
  "value": 100 000 000,
  "scriptPubkey": [
    ## A 32-byte hash
    ## implies a script.
    0, <script hash>
  ]
}
```

```
"vin"[0]: {
  "txid": a1b2c3d4e5f6,
  "vout": 0,
  "scriptSig": [],
  "sequence": 0xFFFFFFFF,
  "txinWitness": [
    <script argument 2>
    <script argument 1>
    <script>
  ]
}
```

```
[ OP_ADD, OP_5, OP_EQUAL ]
```

Popularity of Transaction Types



Transaction Life-cycle

- Bob gives Alice an address / locking script.
- Alice creates, signs, and broadcasts a transaction to her peers.
- Her peers verify the transaction and relay it across the network.
- The transaction propagates into the memory pool of miners.
- Miners add the transaction to their block to collect fees.
- A valid block is then broadcast across the network.
- Nodes relay the new block, and update their chain-state.
- Bob can now confirm the coins are in his custody.

Future Topics

- Decoding a raw transaction.
- Parsing variable words lengths and opcodes.
- Calculating Public Key / Script Hashes.
- Constructing a signature hash.
- Calculated fields: TXID, Hash, Size, Weight.
- Decoding Segregated Witness flag and data.

Smart Contracts with Bitcoin Script

Coming up next ...