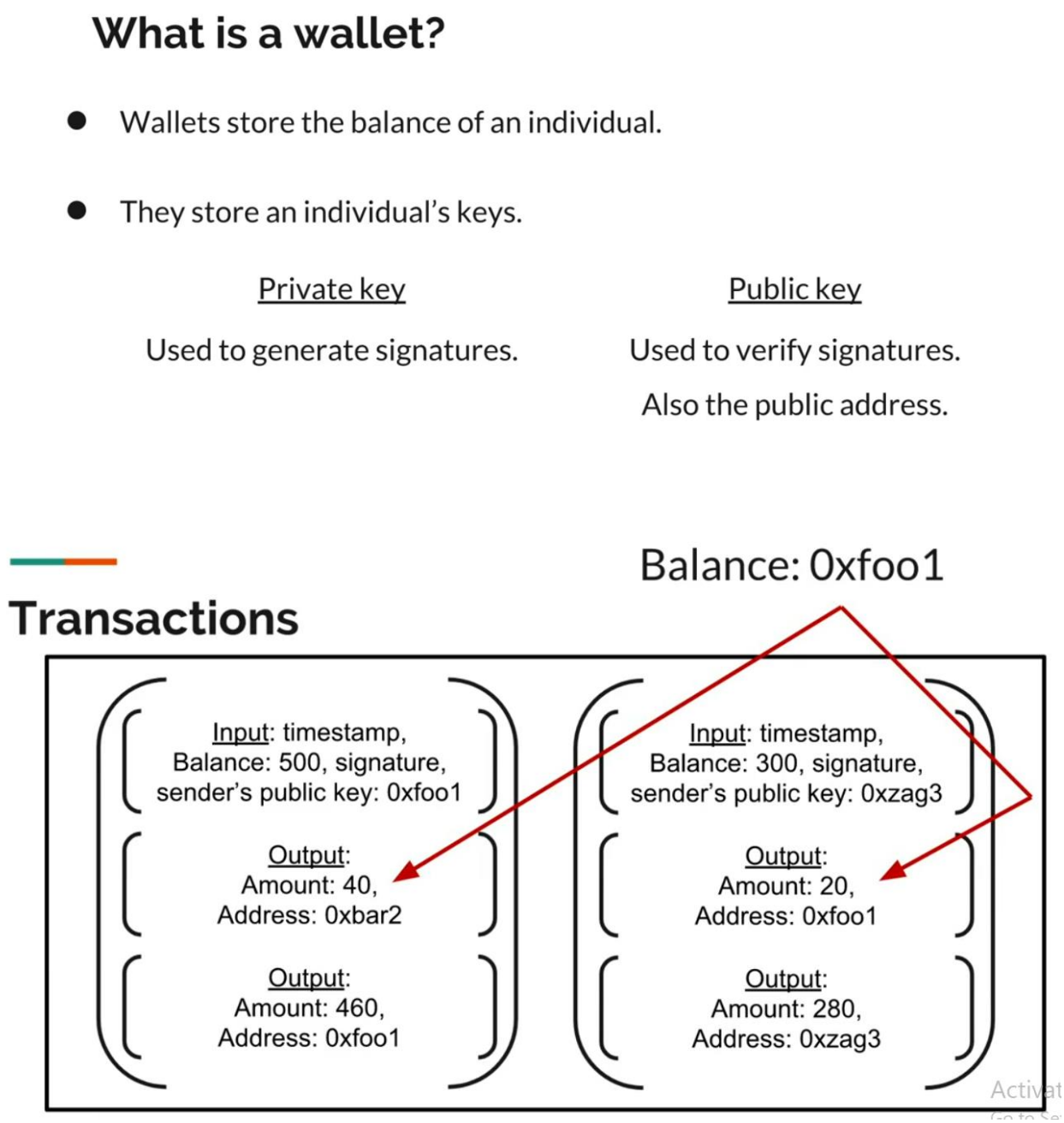
BlockChain

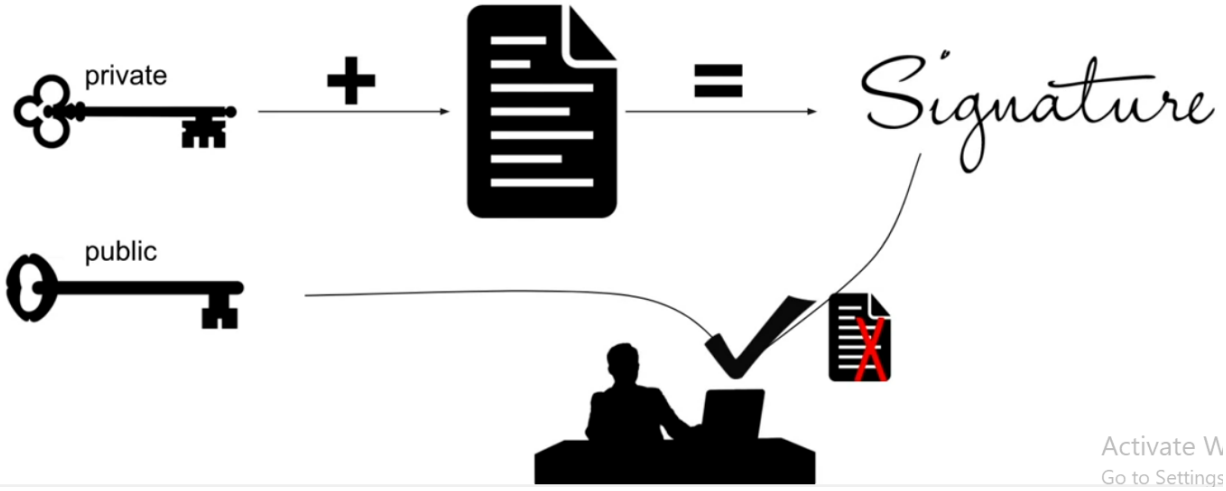
Wallets and Transactions on the Blockchain:

Wallets Keys and Transactions:



BlockChain

Digital Signatures



Blockchain-powered Cryptocurrencies

- Contain wallet objects.
- Keys for digital signatures and verification.
- Have transactions objects to represent currency exchange.

BlockChain

Create Wallet:

JS index.js 1 × JS config.js

wallet > JS index.js > Wallet

```js
const { INITIAL_BALANCE } = require('../config');


class Wallet {
    constructor(){
        this.balance = ;
        this.keyPair = null;
        this.publickey = null;
    }

    toString(){
        return `Wallet -
        publicKey: ${this.publickey.toString()}
        balance : ${this.balance}`
    }
}

module.exports = Wallet;
```

JS index.js 1 JS config.js ×

JS config.js > <unknown> > INITIAL_BALANCE

```js
const DIFFICULTY = 4;

const MINE_RATE = 3000;
 const INITIAL_BALANCE = 500;

module.exports = { DIFFICULTY, MINE_RATE, INITIAL_BALANCE};
```

BlockChain

Chain Util and Key Generation

npm i elliptic --save

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i elliptic --save

added 7 packages, and audited 377 packages in 2s

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

index.js > ...

```js
const Wallet = require('./wallet');
const wallet = new Wallet();
console.log(wallet.toString());
```

wallet > JS index.js > Wallet

```js
const ChainUtil = require('../chain-util');
const { INITIAL_BALANCE } = require('../config');


class Wallet {
    constructor(){
        this.balance = INITIAL_BALANCE;
        this.keyPair = ChainUtil.genKeyPair;
        this.publicKey = this.keyPair.getPublic().encode('hex');
    }

    toString(){
        return `Wallet -
        publicKey: ${this.publicKey.toString()}
        balance : ${this.balance}`
    }
}

module.exports = Wallet;
```

BlockChain

Create a Transaction

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i uuid --save

added 1 package, and audited 378 packages in 819ms

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

wallet > transactions.js > Transaction > constructor

```js
const ChainUtil = require('../chain-util');

class Transaction {
    constructor(){
        this.id = ChainUtil.id();
        this.input = null;
        this.outputs = [];
    }

    static newTransaction(senderWallet, recipient, amount){
        const transaction = new this();

        if(amount >senderWallet.balance){
            console.log(`Amount : ${amount} exceeds balance.`);
            return;
        }


        transaction.outputs.push(...[
            {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
            {amount, address: recipient}
        ])

        return transaction;

    }
}

module.exports = Transaction;
```

BlockChain

Test the Transaction:

chain-util.js > ChainUtil > genKeyPair

```javascript
const EC = require('elliptic').ec;
const uuidV1 = require('uuid/v1');
const ec = new EC('secp256k1');

class ChainUtil {
    static genKeyPair(){
        return ec.genKeyPair();
    }

    static id() {
        return uuidV1();
    }
}

module.exports = ChainUtil;
```

wallet > transaction.test.js > ...

```javascript
const Transaction = require('./transactions');
const Wallet = require('./index');


describe('Transcation', ()=> {
    let transaction, wallet, recipient, amount;

    beforeEach(() => {
        wallet = new Wallet();
        amount = 50;
        recipient = 'r3c1p13nt';
        transactions = Transaction.newTransaction(wallet, recipient,amount);

    });

    it('outputs the `amount` subtracted from the wallet balance', () =>{
        expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
        .toEqual(wallet.balance - amount);

    });

    it('outputs the `amount` added to the recipient', () =>{
        expect(transaction.outputs.find(output.address === recipient).amount)
        .toEqual(amount);
    });
})
```

{} package.json > {} dependencies

```javascript
    "jest":{
      "testEnvironment": "node"
    },
```

BlockChain

wallet > JS transaction.test.js > describe('Transcation') callback > describe('transacting with an amount that exceeds the balance')

```javascript

    it('outputs the `amount` subtracted from the wallet balance', () =>{
        expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
        .toEqual(wallet.balance - amount);

    });

    it('outputs the `amount` added to the recipient', () =>{
        expect(transaction.outputs.find(output.address === recipient).amount)
        .toEqual(amount);
    });

    describe('transacting with an amount that exceeds the balance', ()=>{
        beforeEach(() =>{
            amount = 50000;
            transaction = Transaction.newTransaction(wallet, recipient, amount);
        });

        it('does not create the transaction', () =>{
            expect(transaction).toEqual(undefined);
        });
    });
})
```

Sign a Transaction:

BlockChain

JS chain-util.js > ChainUtil > hash

```javascript
const EC = require('elliptic').ec;
const SHA256 = require('crypto-js/sha256');
const { v1: uuidV1 } = require('uuid');
//const uuidV1 = require('uuid/v1');


// const uuidV1 = require('uuid');

//    console.log(uuidV1.v1());


// var uuid = require('uuid');
// const uuidV1 = require('uuid/v1');

const ec = new EC('secp256k1');

class ChainUtil {
    static genKeyPair(){
        return ec.genKeyPair();
    }

    static id() {
        return uuidV1();
    }

    static hash(data){
        return SHA256(JSON.stringify(data)).toString();
    }
}
```

BlockChain

wallet > index.js > Wallet > sign

```js
const ChainUtil = require('../chain-util');
const { INITIAL_BALANCE } = require('../config');


class Wallet {
    constructor(){
        this.balance = INITIAL_BALANCE;
        this.keyPair = ChainUtil.genKeyPair;
        this.publicKey = this.keyPair.getPublic().encode('hex');
    }

    toString(){
        return `Wallet -
        publicKey: ${this.publicKey.toString()}
        balance : ${this.balance}`
    }

    sign(dataHash){
        return this.keyPair.sign(dataHash);
    }
}

module.exports = Wallet;

//getPublic()
```

BlockChain

wallet > JS transactions.js > Transaction > newTransaction

```
        }


        transaction.outputs.push(...[
            {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
            {amount, address: recipient}
        ])

        Transaction.signTransaction(transaction, senderWallet);

        return transaction;

    }

    static signTransaction(transaction, senderWallet){
        transaction.input ={
            timestamp: Date.now(),
            amount: senderWallet.balance,
            address: senderWallet.publicKey,
            signature: senderWallet.sign(ChainUtil.hash(transaction.outputs))
        }
    }
}
```

Test the Transaction Input:

wallet > JS transaction.test.js > describe('Transcation') callback > it('inputs the balance of the

```

    it('inputs the balance of the wallet', ()=>{
        expect(transaction.input.amount).toEqual(wallet.balance);
    })
```

Verify Transactions:

JS chain-util.js > ChainUtil > verifySignature

```

    static verifySignature(publicKey, signature, dataHash){
        return ec,keyFromPublic(publicKey, 'hex').verify(dataHash, signature);
    }
}
```

BlockChain

wallet > JS transactions.js > Transaction > verifyTransaction

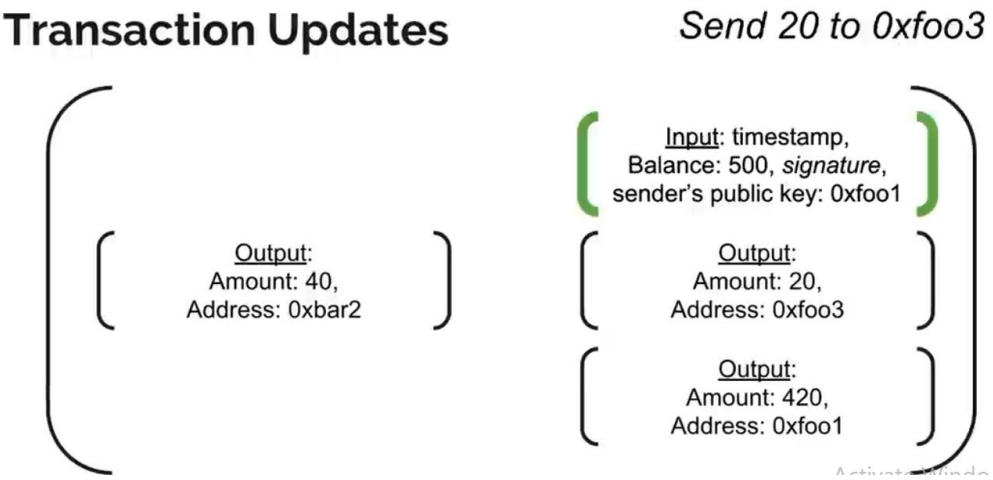
```
39
40      static verifyTransact any ransaction){
41          return ChainUtil.verigySignature(
42              transaction.input.address,
43              transaction.input.signature,
44              ChainUtil.hash(transaction.outputs)
45          );
46      }
```

Test Transaction Verification:

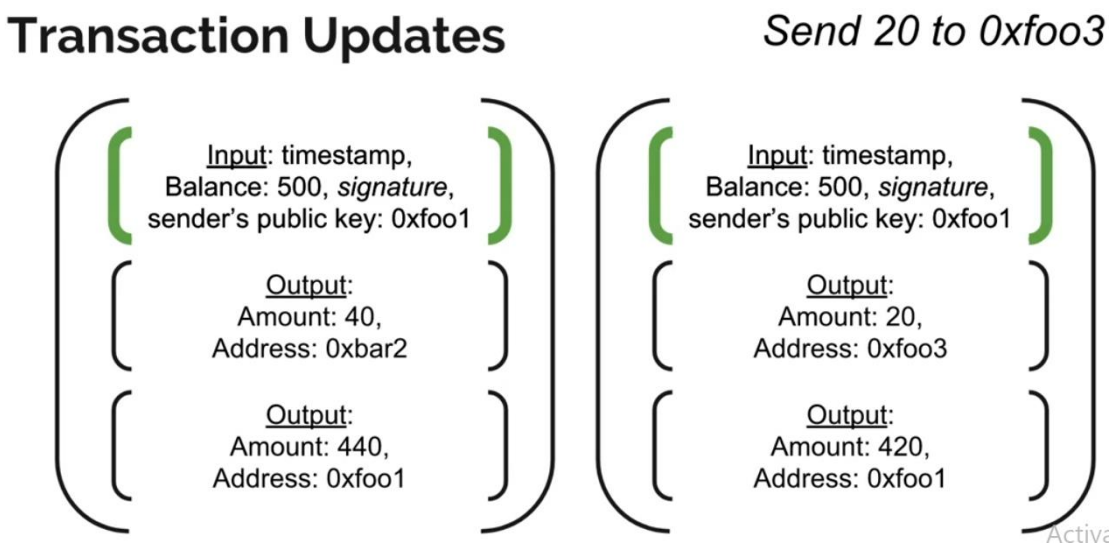
wallet > JS transaction.test.js > describe('Transcation') callback

```
33
34      it('validates a valid transaction', () => {
35          expect(Transaction.verifyTransaction(transaction)).toBe(true);
36      });
37
38      it('invalidates a corrupt transaction', () =>{
39          transaction.outputs[0].amount = 50000;
40          expect(Transaction.verifyTransaction(transaction)).toBe(false);
41      });
```

Transaction Updates:



BlockChain



wallet > JS transactions.js > Transaction > update

```
    update(senderWallet, recipient, amount){
        const senderOutput = this. outputs.find(output => output.address == senderWallet.publicKey);

        if (amount > senderOutput.amount) {
            console.log(`Amount : ${amount} exceeds balance.`);
            return ;
        }

        senderOutput.amount = senderOutput.amount - amount;
        this.outputs/push({amount, address: recipient});
        Transaction.signTransaction(this, senderWallet);

        return this;

    }
```

Test Transaction Updates:

BlockChain

wallet > JS transaction.test.js > ...

```
    describe('and updating a transaction', () =>{
        let nextAmount, nextRecipient;

        beforeEach(() =>{
            nextAmount = 20;
            nextRecipient = 'n3xt-4ddr355';
            transaction = transaction.update(wallet, nextRecipient, nextAmount);
        });

        it('subtrancts the next amount from the senders output', ()=>{
            expect(transaction.outputs.find(output => output.address === wallet.publicKey).amount)
            .toEqual(wallet.balance - amount - nextAmount);

        });

        it('outputs an amount for the next recipient', ()=>{
            expect(transaction.outputs.find(output => output.address == nextRecipient).amount)
            .toEqual(nextAmount);
        });

    });
});
```