Blockchain & Αποκεντρωμένες Εφαρμογές

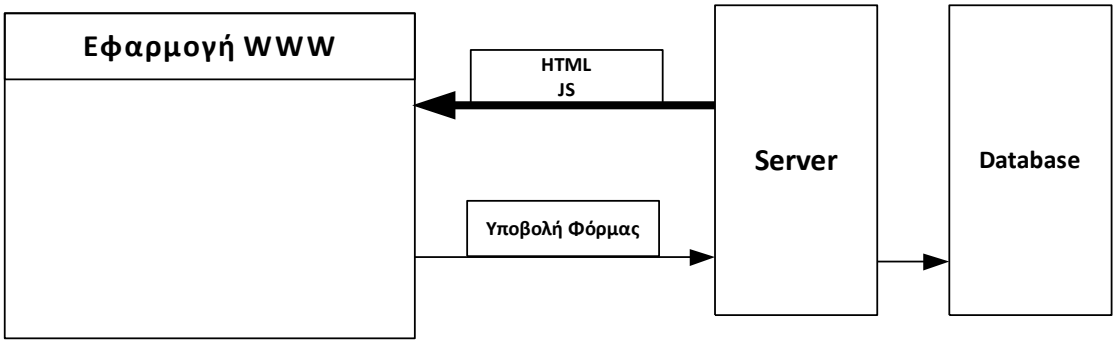
4η Εργαστηριακή Δραστηριότητα

Μαυρίδης Ιωάννης, Φουληράς Παναγιώτης
Μάστορας Θεόδωρος
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

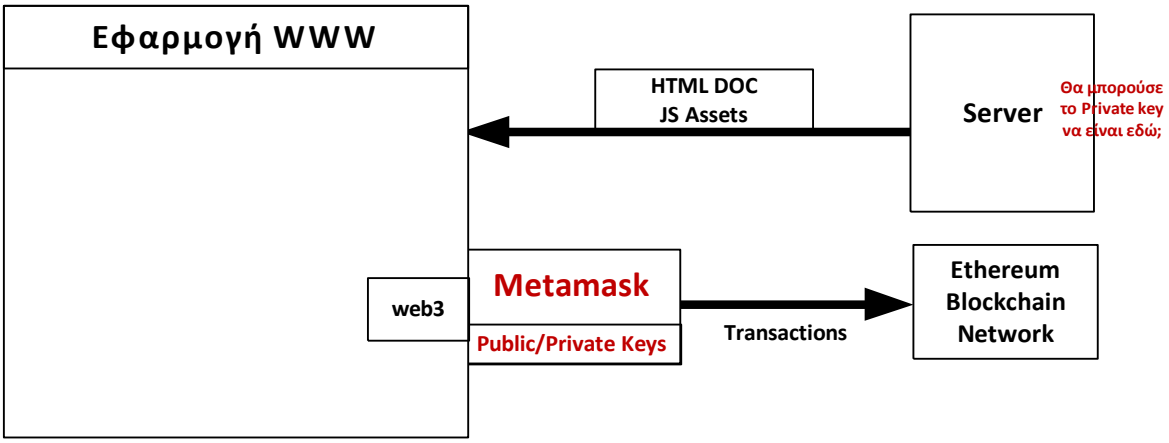
MULTIMEDIA SECURITY NETWORKING

A. Web και Web3.

Παραδοσιακή αρχιτεκτονική



Αρχιτεκτονική Ethereum



B. Συμβόλαιο λαχείου Lottery.sol.

1. Επιστρέψτε στον "File explorer" του Remix και δημιουργήστε ένα νέο αρχείο solidity με το όνομα " Lottery.sol".

Πληκτρολογήστε ή αντιγράψτε το παρακάτω πρόγραμμα Solidity:

link

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0;

contract Lottery {
    address public manager;
    address[] public players;

    event PlayerEntered(address player);
    event WinnerPicked(address winner);


    constructor() {
        manager = msg.sender;
```

```
    }

    function enter() public payable {
        require(msg.value >= .01 ether);

        players.push(msg.sender);
        emit PlayerEntered(msg.sender);
    }

    function random() private view returns (uint) {
        return uint(keccak256(abi.encodePacked(block.timestamp)));
    }

    function pickWinner() public restricted {
        uint index = random() % players.length;
        payable (players[index]).transfer(address(this).balance);
        emit WinnerPicked(players[index]);
        players = new address[](0);
    }

    modifier restricted() {
        require(msg.sender == manager);
        _;
    }

    function getPlayers() public view returns (address[] memory) {
        return players;
    }
}
```

Το πρόγραμμα αυτό υλοποιεί ένα smart contract απλού λαχείου. Όποιος καλέσει την **enter** πληρώνοντας ποσό ίσο ή μεγαλύτερο από 1 λεπτό του Ether γίνεται υποψήφιος νικητής του λαχείου. Ακόμη κι αν πληρώσει ποσό πολύ μεγαλύτερο από 0,01 Ether, αποκτά μόνο ένα λαχνό (η διεύθυνσή του μπαίνει μόνο μια φορά στον πίνακα των υποψηφίων). Αν όμως **ξανακαλέσει** την enter παίρνει ακόμη ένα λαχνό (η διεύθυνσή του ξαναμπαίνει στον πίνακα) και έτσι αποκτά περισσότερες πιθανότητες νίκης.

Η συνάρτηση **getPlayers** επιστρέφει κάθε φορά τα ονόματα των υποψηφίων (επιστρέφει τον **πίνακα players**).

Την κλήρωση μπορεί να την εκτελέσει μόνο ο δημιουργός του συμβολαίου (μεταβλητή **manager**) καλώντας την συνάρτηση **pickWinner**. Όλο το ποσό που έχει δοθεί ως τότε μεταφέρεται τυχαία σε έναν από τους παίκτες. Η λίστα υποψηφίων που έχουν αγοράσει λαχείο αδειάζει. Μπορούν να συνεχίσουν να εγγράφονται παίκτες με το ποσό (έπαθλο) να αυξάνει και πάλι.

2. Πειραματιστείτε με το συμβόλαιο αφού το μεταγλωττίσετε και το δημοσιεύσετε στο τοπικό “Ganache”. Στο REMIX επιλέξτε “Injected Web3” και συνδέστε το Metamask με το Ganache.

C. Εγκατάσταση τοπικά μιας κενής react εφαρμογής για μετέπειτα δημοσίευση στον lite-server του node.js (http://localhost:3000/)

1. Σε παράθυρο γραμμής εντολών, δώστε την εντολή:

```
npm install -g create-react-app
```

Η εντολή αυτή εγκαθιστά καθολικά στον υπολογιστή τις απαραίτητες node.js βιβλιοθήκες για το framework της react. (Δεν θα χρειαστεί να την ξαναδώσετε στον συγκεκριμένο υπολογιστή)

2. **Έξω από τον φάκελο** (web), σε παράθυρο γραμμής εντολών, δώστε την εντολή:

```
create-react-app web
```

Δημιουργείται (στον τρέχοντα π.χ. στα έγγραφα) ο φάκελος "web" με τον server της react. Αν ο φάκελος υπάρχει ήδη θα πρέπει να είναι άδειος, διαφορετικά η εγκατάσταση δεν θα προχωρήσει. Εγκαθίστανται περισσότερα από 500MB αρχείων στον φάκελο. Η εντολή διαρκεί κάποιο χρόνο.

3. Δώστε

```
cd web
```

για να «βρεθείτε» εντός του νέου φακέλου.

4. Εγκαταστήστε την βιβλιοθήκη **web3** του node.js δίνοντας

```
npm install web3
```

5. Ενώ στη γραμμή εντολών βρίσκεστε ακόμα **μέσα στον φάκελο web**, εγκαταστήστε τη βιβλιοθήκη bootstrap για να μπορεί ο κώδικας να ενσωματώσει την αντίστοιχη μορφοποίηση CSS:

```
npm install -save bootstrap
```

6. Καθώς συνεχίζετε να βρίσκεστε **μέσα στον φάκελο web**, εγκαταστήστε τη βιβλιοθήκη web-vitals για να μπορεί ο κώδικας να εκτελεί μετρήσεις δεικτών απόδοσης της εφαρμογής (η εγκατάσταση της βιβλιοθήκης γινόταν αυτόματα μέχρι πρόσφατα):

```
npm install --save-dev web-vitals
```

Είστε σχεδόν έτοιμοι. Αν δώσετε την εντολή **`npm run start`** εντός του φακέλου web, ο server θα εκκινήσει και θα εμφανιστεί μια σελίδα με το λογότυπο της react. Αν θέλετε να τερματίσετε τον web server, στο παράθυρο γραμμής εντολών όπου δώσατε το **`npm run start`**, πατήστε **Ctrl+C**.

D. Υλοποίηση σε react της διασύνδεσης χρήστη (front end).

Στον φάκελο src βρίσκεται όλη η εφαρμογή. Στη δική μας περίπτωση, μπορούμε να θεωρήσουμε πως το αρχείο 'App.js' περιέχει όλη τη διασύνδεση (όλες τις ιστοσελίδες).

1. Με τη βοήθεια του VS Code, **μέσα στον φάκελο src** δημιουργήστε το αρχείο 'web3.js' με το παρακάτω περιεχόμενο:

link

```
import Web3 from 'web3';

const web3 = new Web3(window.ethereum);

export default web3;
```

2. Και πάλι **μέσα στον φάκελο src** δημιουργήστε το αρχείο 'lottery.js' με το παρακάτω περιεχόμενο:

link

```
import web3 from './web3';

const address = '0xd832133C60E3Fa999a44964c69eA0C5A798bfD36';

const abi = [

...

];

const lottery = new web3.eth.Contract(abi, address);

export default lottery;
```

Στο αρχείο αυτό **αντικαταστείστε** τη διεύθυνση και το **ABI** τμήμα με τα δικά σας.

3. Στο VS Code, μέσα στον φάκελο src ανοίξτε το αρχείο 'App.js' και τροποποιήστε το περιεχόμενό του στο παρακάτω:

link

```
import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.css';
import web3 from './web3';
import lottery from './lottery';

// Η κλάση App αποτελεί απόγονο της Component
// η οποία είναι θεμελιώδης στην react.
// Σε κάθε αλλαγή κάποιας μεταβλητή state της App,
// όλη η ιστοσελίδα (HTML) γίνεται refresh
// καλώντας αυτόματα τη render()...
// ...ΠΡΟΣΟΧΗ! ΔΕΝ γίνεται reload... ΜΟΝΟ refresh
class App extends Component {
  state = {
```

```
  manager: '',
  players: [],
  balance: '',
  value: '',
  message: '',
  currentAccount: ''
};

// Η componentDidMount() καλείται ΜΟΝΟ την πρώτη φορά
// που φορτώνει η ιστοσελίδα (είναι σαν την onLoad())
async componentDidMount() {
  try { // Αν υπάρχει εγκατεστημένο metamask
    // Ορισμός των state μεταβλητών
    const manager = await lottery.methods.manager().call();
    const players = await lottery.methods.getPlayers().call();
    const balance = await web3.eth.getBalance(lottery.options.address);
    this.setState({ message: '', manager, players, balance });
    try { // Επικοινωνία με το metamask
      const currentAccount = (await window.ethereum.request({ method: 'eth_requestAccounts' }))[0];
      this.setState({ message: '', currentAccount });
    } catch (error) { // Αν το metamask δεν έκανε accept το request
      this.setState({ message: 'Metamask has not connected yet' });
    }
  } catch (error) { // Αν το metamask δεν έχει εγκατασταθεί
    this.setState({ message: 'Metamask is not installed' });
  }
  // Set up event listeners only once
  if (!this.eventListenersSet) {
    this.setupEventListeners();
    this.eventListenersSet = true;
  }
}

setupEventListeners() {
  // Κάθε φορά που επιλέγεται άλλο πορτοφόλι στο metamask...
  window.ethereum.on('accountsChanged', (accounts) => {
    // ... να γίνεται refresh η σελίδα
    const currentAccount = accounts[0];
    this.setState({ currentAccount });
  });
}

// Όταν πατηθεί το κουμπί "Enter"
onSubmit = async event => {
  event.preventDefault();

  this.setState({ message: 'Waiting on transaction success...' });

  await lottery.methods.enter().send({ // Κλήση της "enter()" του συμβολαίου
    from: this.state.currentAccount,
    // gas: 100000, gasPrice: 1000000000, // μόνο στο Ganache
    value: web3.utils.toWei(this.state.value, 'ether')
  });

  this.setState({ message: 'You have been entered!' });
};

// Όταν πατηθεί το κουμπί "Pick a Winner"
onClick = async () => {
  this.setState({ message: 'Waiting on transaction success...' });

  await lottery.methods.pickWinner().send({ // Κλήση της "pickWinner()" του συμβολαίου
    // gas: 100000, gasPrice: 1000000000, // μόνο στο Ganache
    from: this.state.currentAccount
  });

  this.setState({ message: 'A winner has been picked!' });
};

// Κάθε φορά που η σελίδα γίνεται refresh
render() {
```

```
    return (
      <div>
        <h2>Lottery Contract</h2>
        {/* Ό,τι βρίσκεται εντός των άγκιστρων είναι κώδικας JavaScript */}
        {/* Η σελίδα HLML λειτουργεί αυτόνομα, σαν να εκτελείται σε κάποιον server */}
        <p>
          This contract is managed by {this.state.manager}. There are currently{' '}
          {this.state.players.length} people entered, competing to win{' '}
          {web3.utils.fromWei(this.state.balance, 'ether')} ether!
        </p>


        <hr /> {/*  -------------------- Οριζόντια γραμμή -------------------- */}

        {/* Η φόρμα "Want to try your luck?" */}
        {/* Θα μπορούσε αντί να χρησιμοποιηθεί φόρμα, */}
        {/* να χρησιμοποιηθεί button... */}
        {/* και αντί .onSubmit να χρησιμοποιούνταν .onClick */}
        <form onSubmit={this.onSubmit}>
          <h4>Want to try your luck? Connected wallet address: {this.state.currentAccount}</h4>
          <div>
            <label>Amount of ether to enter</label>
            <input
              value={this.state.value}
              onChange={event => this.setState({ value: event.target.value })}
            />
          </div>
          <button>Enter</button>
        </form>

        <hr /> {/*  -------------------- Οριζόντια γραμμή -------------------- */}

        <h4>Ready to pick a winner?</h4>
        <button onClick={this.onClick}>Pick a winner!</button>

        <hr /> {/*  -------------------- Οριζόντια γραμμή -------------------- */}

      </div>
    );
  }
}

export default App;
```

4. Για να εκκινήσετε τον web server και να μοιράζετε τη σελίδα, στο παράθυρο γραμμής εντολών, δώστε την εντολή:

`npm run start`

Φροντίστε στο firewall να έχετε επιτρέψει τις εισερχόμενες κλήσει στο port 3000 (για περισσότερους φακέλους του lite-server ταυτόχρονα 3001, 3002 κ.λπ.) Δοκιμάστε την εφαρμογή ανοίγοντας ταυτόχρονα πολλούς browsers από διαφορετικά πορτοφόλια.

5. Καθώς πειραματίζεστε με την DApp παρατηρείτε πως δεν υπάρχει άμεση ενημέρωση όταν κάποιος **άλλος** αγοράσει έναν λαχνό ή όταν τέλος πραγματοποιηθεί η κλήρωση και προκύψει νικητής. Για να ορίσουμε τα συμβάντα του συμβολαίου να πυροδοτούνται εντός της react (ώστε να καλυφθεί και αυτή η απαίτηση) χρειάζονται δύο μικρά κομμάτια κώδικα.

Το πρώτο εντός της setupEventListeners():

```
// Όποτε αγοραστεί λαχνός να ενημερώσεις τις players & balance
```

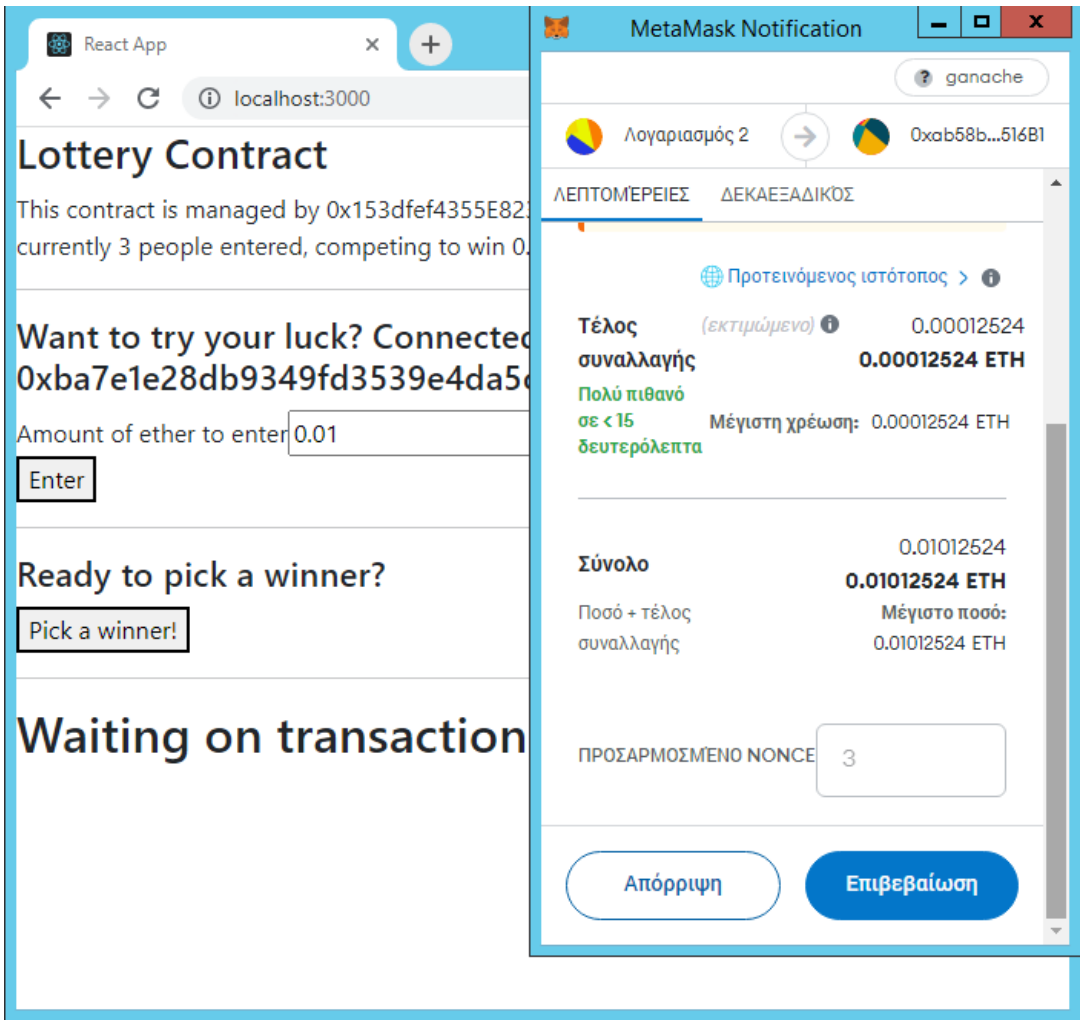
```
lottery.events.PlayerEntered().on('data', async (data) => {
    console.log(data.returnValues.player);
    const players = await lottery.methods.getPlayers().call();
    const balance = await web3.eth.getBalance(lottery.options.address);
    this.setState({ players, balance });
});

// Όποτε γίνει κλήρωση να ενημερώσεις τις players & balance
// και να δημιουργήσεις τη νέα state μεταβλητή lastWinner
lottery.events.WinnerPicked().on('data', async (data) => {
    console.log(data.returnValues.winner);
    const players = await lottery.methods.getPlayers().call();
    const balance = await web3.eth.getBalance(lottery.options.address);
    const lastWinner = data.returnValues.winner;
    this.setState({ lastWinner, players, balance });
});
```

Το δεύτερο στο τέλος της render():

```
    <h1>{this.state.message}</h1>
    {this.state.lastWinner &&
      <h3>Last Winner Address: {this.state.lastWinner}</h3>
    }
  </div>
);
```

Μπορείτε να κατεβάσετε το νέο αναθεωρημένο App.js **link**. Χωρίς να διακόψετε με Ctrl+C το τερματικό όπου εκτελείται ο server της React (αυτό με την “npm run start”), αντικαθιστώντας το παλιό αρχείο App.js στον φάκελο src, η σελίδα θα ενημερωθεί αυτόματα.



Μπορείτε να δείτε την εφαρμογή [link]

E. Approval.sol - Υλοποίηση της διασύνδεσης ΚΑΙ του συμβολαίου της προηγούμενης εργαστηριακής άσκησης σε react.

Με στόχο να εξοικειωθείτε με τη διαδικασία, επαναλάβετε σε έναν άλλο φάκελο π.χ. τον web_02 το front end της DApp του συμβολαίου Approval. Ξεκινήστε δημιουργώντας τον φάκελο από την αρχή δίνοντας σε παράθυρο γραμμής εντολών την εντολή:

create-react-app web_02

Αφού επαναλάβετε όλα τα βήματα όπως πριν, θα χρειαστείτε να φτιάξετε μόνοι σας, μέσα στον φάκελο src, το αρχείο approval.js. Τέλος θα γράψετε τον παρακάτω κώδικα στο αρχείο 'App.js':

link

```
import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.css';
import web3 from './web3';
import approval from './approval';

class App extends Component {
  state = {
    manager: '',
    balance: '',
    value: '',
    account: '',
    receiver: '',
    contract: ''
  }; //χρησιμοποιούμε state για να αποθηκεύουμε συγκεκριμένες τιμές


  async componentDidMount() {
    const accounts = await web3.eth.getAccounts();
    this.setState({ account: accounts[0] });

    const balance = await web3.eth.getBalance(approval.options.address);
    this.setState({ balance });
  }


  onClick = async () => {
    const accounts = await web3.eth.getAccounts();
    this.setState({ account: accounts[0] })

    const contractAddress = await approval.methods.contractAddress().call();
    this.setState({ contract: contractAddress })

    const balance = await web3.eth.getBalance(approval.options.address);
    this.setState(balance);
  };

  onClick1 = async () => {

    const balance = await web3.eth.getBalance(approval.options.address);
    this.setState({ balance });

  };

  onClick2 = async () => {
    const accounts = await web3.eth.getAccounts();
```

```
    var newString = this.state.receiver.toString();

    this.setState({ message: 'Waiting on transaction success...' });
    await approval.methods.deposit(web3.utils.toChecksumAddress(newString)).send({
      from: accounts[0],
      value: web3.utils.toWei(this.state.value, 'ether')
    });
  };

  onClick3 = async () => {
    const { 0: manager } = await approval.methods.viewApprover().call();
    this.setState({ manager });
  }

  onClick4 = async () => {
    const accounts = await web3.eth.getAccounts();

    await approval.methods.approve().send({
      from: accounts[0]
    });;
  }

  render() {
    return (
      <div style={{
        display: 'block',
        padding: 30, alignContent: 'center'
      }}>

        <h1 class="text-center">DApp μεσάζοντα εγγυητή</h1> <br /><br />
        <div class="container" style={{ display: 'flex', justifyContent: 'center' }}>
          <div class="row" >
            <div class="col-sm" style={{ paddingRight: 30 }}>
              <button style={{ backgroundColor: "#24dd64", borderRadius: 20, }}
                      onClick={this.onClick}>-------Show/refresh contract details!------
</button>
            </div>
            <div class="col-sm" style={{ paddingLeft: 30 }}>
              <div>
                <label>Συνδεδεμένος λογαριασμός Metamask</label>
                <p>{this.state.account}</p>
              </div>
            </div>
          </div>
        </div>

        <br /><br />
        <h3 class="text-center">Αποστολή χρημάτων μέσω συμβολαίου: {this.state.contract}
<br />
          Θα πρέπει να δώσετε έγκριση της μεταφοράς από το Metamask.<br />
        </h3>
        <hr />
        <br /><br />


        <div class="container">
          <div class="row" style={{ width: "500px" }}>


            <h5>Διεύθυνση παραλήπτη χρημάτων</h5>
            <input style={placeholder}
              value={this.state.receiver}
              onChange={event => this.setState({ receiver: event.target.value })}
```

```
            />

            <h5>Χρηματικό ποσό (ETH)</h5>
            <input style={placeholder}
              value={this.state.value}
              onChange={event => this.setState({ value: event.target.value })}
            />

            <hr />

            <button style={button} onClick={this.onClick2}>Αποστολή</button> <br />
            <p>Πατώντας το κουμπί "Αποστολή" τα χρήματά σας θα πιστωθούν στο συμβόλαιο.</p>

            <hr />

          </div>
        </div>

        <div class="container">
          <div class="row" style={{ width: "370px" }}>

            <h4>Υπόλοιπο: {web3.utils.fromWei(this.state.balance, 'ether')} ether!</h4>
            <button style={button} onClick={this.onClick1}>Δείτε το υπόλοιπο</button>
            <br />
            <hr />

            <h4>Εγγυητής: {this.state.manager}</h4>
            <button style={button} onClick={this.onClick3}>Δείτε τον εγγυητή</button>
            Πατήστε το κουμπί για να δείτε τη διεύθυνση του Εγγυητή.


            <button style={button} onClick={this.onClick4}>Επιβεβαίωση συναλλαγής</button>
            Αν είστε ο εγγυητής, πατήστε το κουμπί για να εγκρίνετε τη συναλλαγή και τα
χρήματα να μεταφερθούν στον παραλήπτη.

          </div>
        </div>
      </div>
    );
  }
}

export default App;


//τα παρακάτω χρησιμοποιούνται για το styling
const button = {
  color: "white",
  backgroundColor: "DodgerBlue",
  padding: "10px",
  fontFamily: "Arial",
  borderRadius: 10,
};

const placeholder = {
  borderRadius: 10
}
```