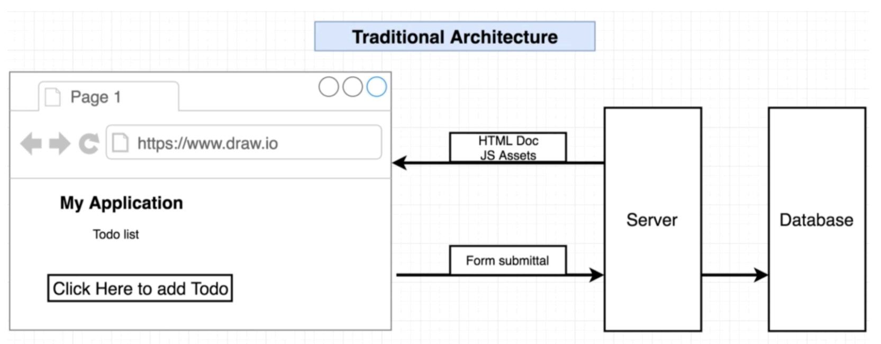
Ethereum and Solidity - The Complete Developer's Guide

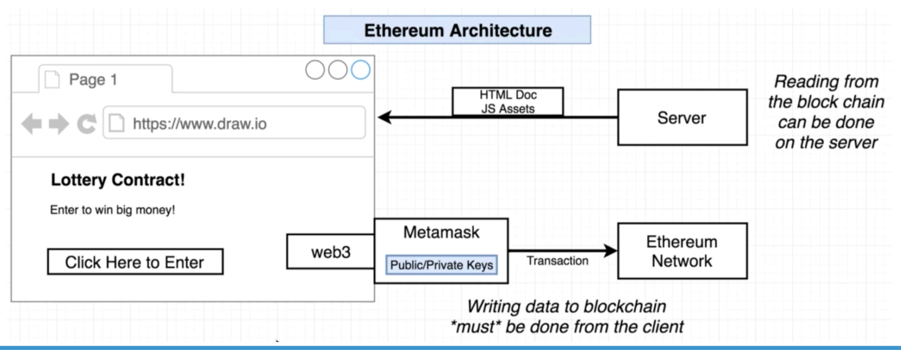
Section 4. Building Interactive Front-Ends (that works with the contract via web3)

Ethereum App Architecture

Traditional Architecture

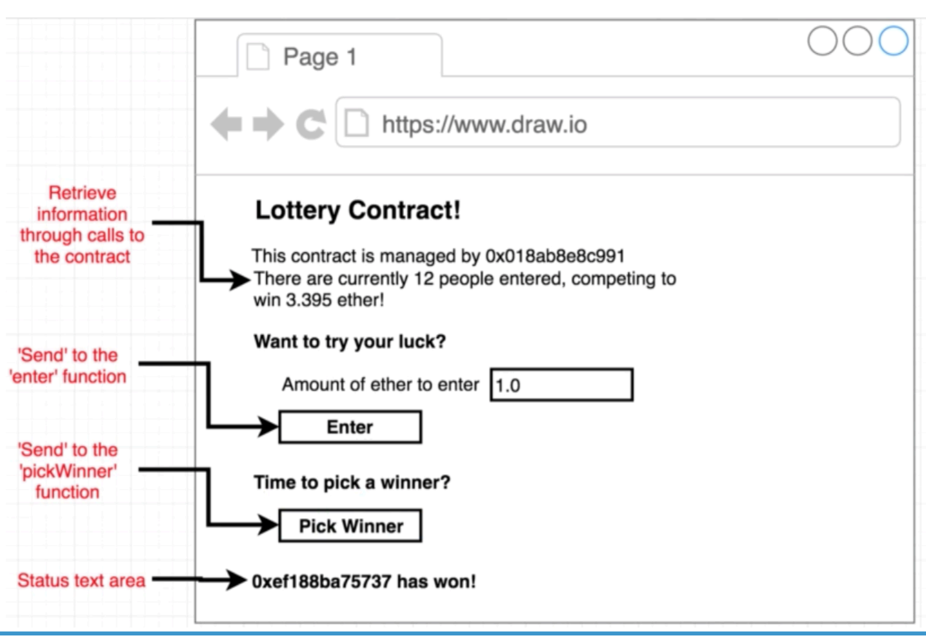


👍 Ethereum Architecture

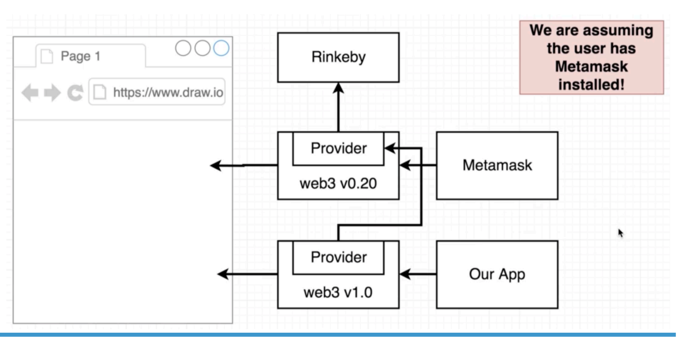


⚠️ Client should be smart - so she front-end framework should be used

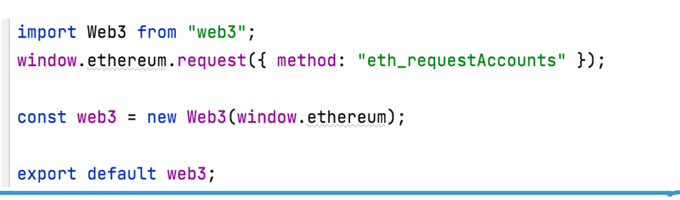
Lottery app overview


Page 1
https://www.draw.io
Lottery Contract!
This contract is managed by 0x018ab8e8c991
There are currently 12 people entered, competing to win 3.395 ether!
Want to try your luck?
Amount of ether to enter
1.0
Enter
Time to pick a winner?
Pick Winner
0xef188ba75737 has won!
Retrieve information through calls to the contract
'Send' to the 'enter' function
'Send' to the 'pickWinner' function
Status text area

Setup Web3



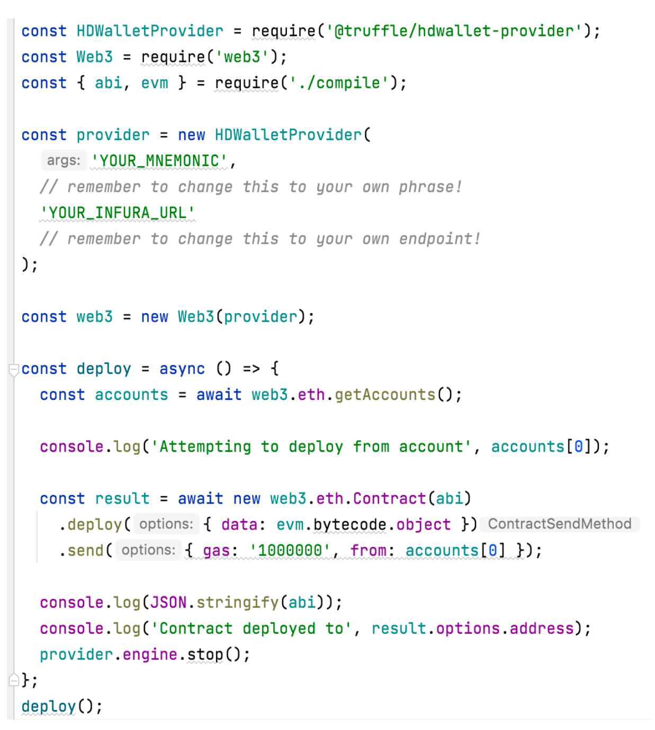
We need to "hijack" the Provider from Metamask's web3



```
import Web3 from "web3";
window.ethereum.request({ method: "eth_requestAccounts" });

const web3 = new Web3(window.ethereum);

export default web3;
```

❓ Won't it share access to the private keys?

Deploy the contract publicly (to test network)

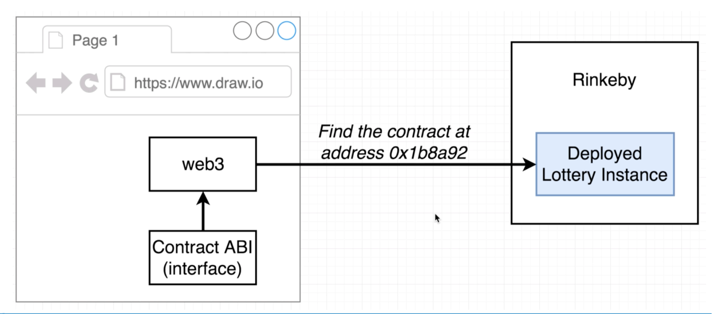


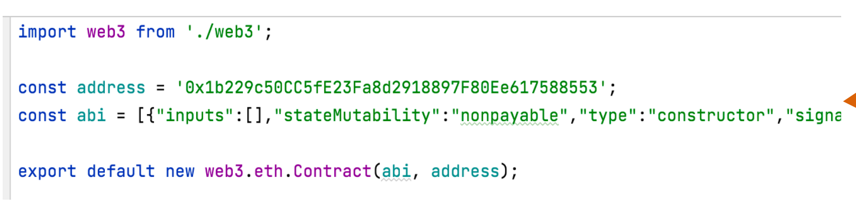
```
const HDWalletProvider = require('@truffle/hdwallet-provider');
const Web3 = require('web3');
const { abi, evm } = require('./compile');

const provider = new HDWalletProvider(
  args: 'YOUR_MNEMONIC',
  // remember to change this to your own phrase!
  'YOUR_INFURA_URL'
  // remember to change this to your own endpoint!
);

const web3 = new Web3(provider);

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

  console.log('Attempting to deploy from account', accounts[0]);

  const result = await new web3.eth.Contract(abi)
    .deploy( options: { data: evm.bytecode.object }) ContractSendMethod
    .send( options: { gas: '1000000', from: accounts[0] });

  console.log(JSON.stringify(abi));
  console.log('Contract deployed to', result.options.address);
  provider.engine.stop();
};
deploy();
```

deploy.js

- node deploy.js
- We now know the contract ADDRESS and ABI

Access local contract instance

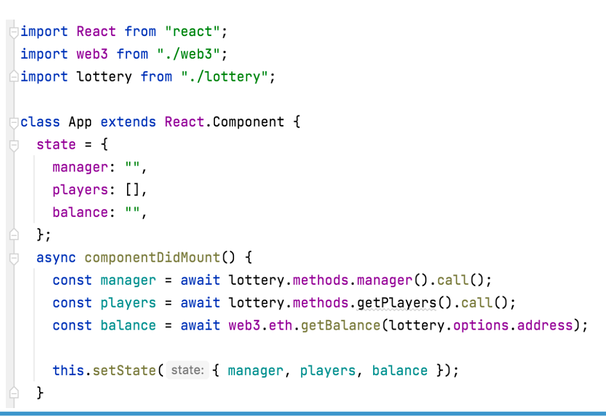

Page 1
https://www.draw.io
web3
Contract ABI (interface)
Find the contract at address 0x1b8a92
Rinkeby
Deployed Lottery Instance



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

const address = '0x1b229c50CC5fE23Fa8d2918897F80Ee617588553';
const abi = [{"inputs":[],"stateMutability":"nonpayable","type":"constructor","signa

export default new web3.eth.Contract(abi, address);
```

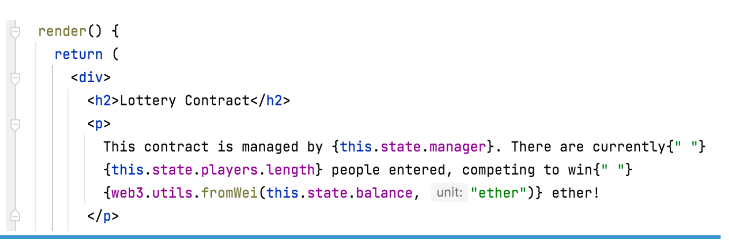
lottery.js

Render current contract data on component load



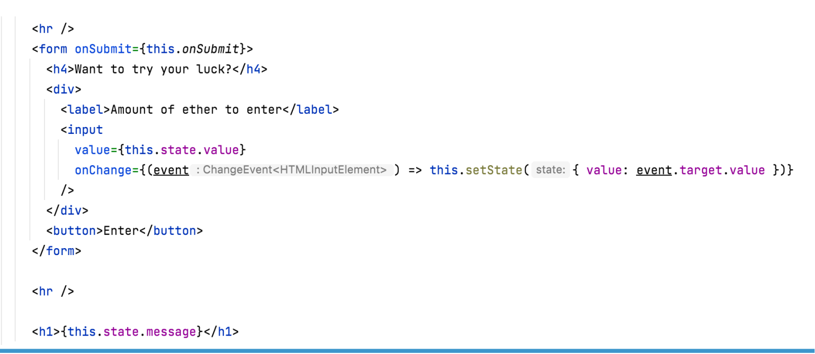
```
import React from "react";
import web3 from "./web3";
import lottery from "./lottery";

class App extends React.Component {
  state = {
    manager: "",
    players: [],
    balance: "",
  };
  async componentDidMount() {
    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( state: { manager, players, balance });
  }
```

balance is a BigNumber object

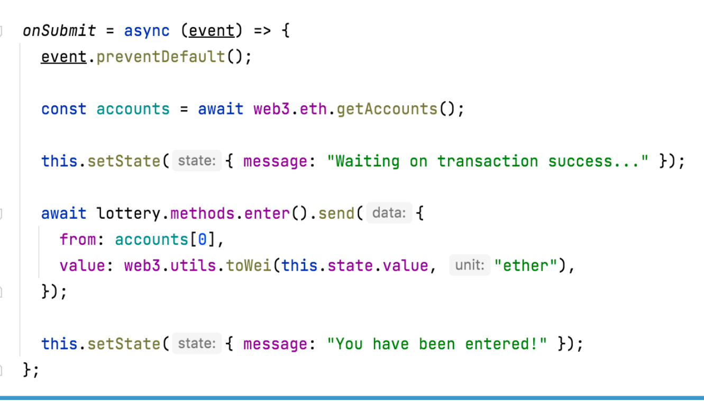


```
  render() {
    return (
      <div>
        <h2>Lottery Contract</h2>
        <p>
          This contract is managed by {this.state.manager}. There are currently{" "}
          {this.state.players.length} people entered, competing to win{" "}
          {web3.utils.fromWei(this.state.balance,  unit: "ether")} ether!
        </p>
```

Enter the contract form

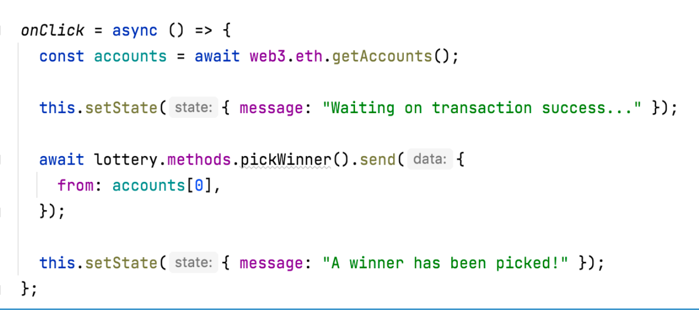


```
<hr />
<form onSubmit={this.onSubmit}>
  <h4>Want to try your luck?</h4>
  <div>
    <label>Amount of ether to enter</label>
    <input
      value={this.state.value}
      onChange={(event : ChangeEvent<HTMLInputElement> ) => this.setState( state: { value: event.target.value })}
    />
  </div>
  <button>Enter</button>
</form>

<hr />

<h1>{this.state.message}</h1>
```

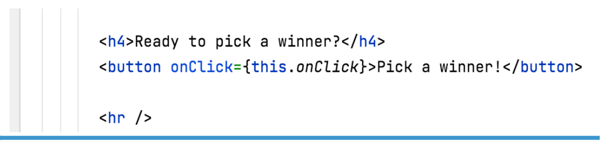


```
onSubmit = async (event) => {
  event.preventDefault();

  const accounts = await web3.eth.getAccounts();

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

  await lottery.methods.enter().send( data: {
    from: accounts[0],
    value: web3.utils.toWei(this.state.value,  unit: "ether"),
  });

  this.setState( state: { message: "You have been entered!" });
};
```

Picking a winner by the contract admin



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

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

  await lottery.methods.pickWinner().send( data: {
    from: accounts[0],
  });

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



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

<hr />
```