4 - Tvorba volebního systému

# Výběr IDE (vývojové prostředí)

Solidity je vyšší programovací jazyk pro vytváření smart kontraktů. Jedná se o objektově orientovaný software.

Je vytvořen tak, aby cílil na virtuální stroj Ethereum (EVM).

Jazyky jako C++, Python a JavaScript byly inspirací pro vytvoření objektově orientovaného jazyka Solidity. Společné funkce, jako je podpora dědičnosti, knihovny a složité uživatelsky definované typy atd.

Vývojáři považují Solidity za užitečný při vytváření smluv jako je hlasování, crowdfunding, či aukce.

Solidity můžeme psát v řadě IDE prostředí, například EthFiddle nebo JetBrains

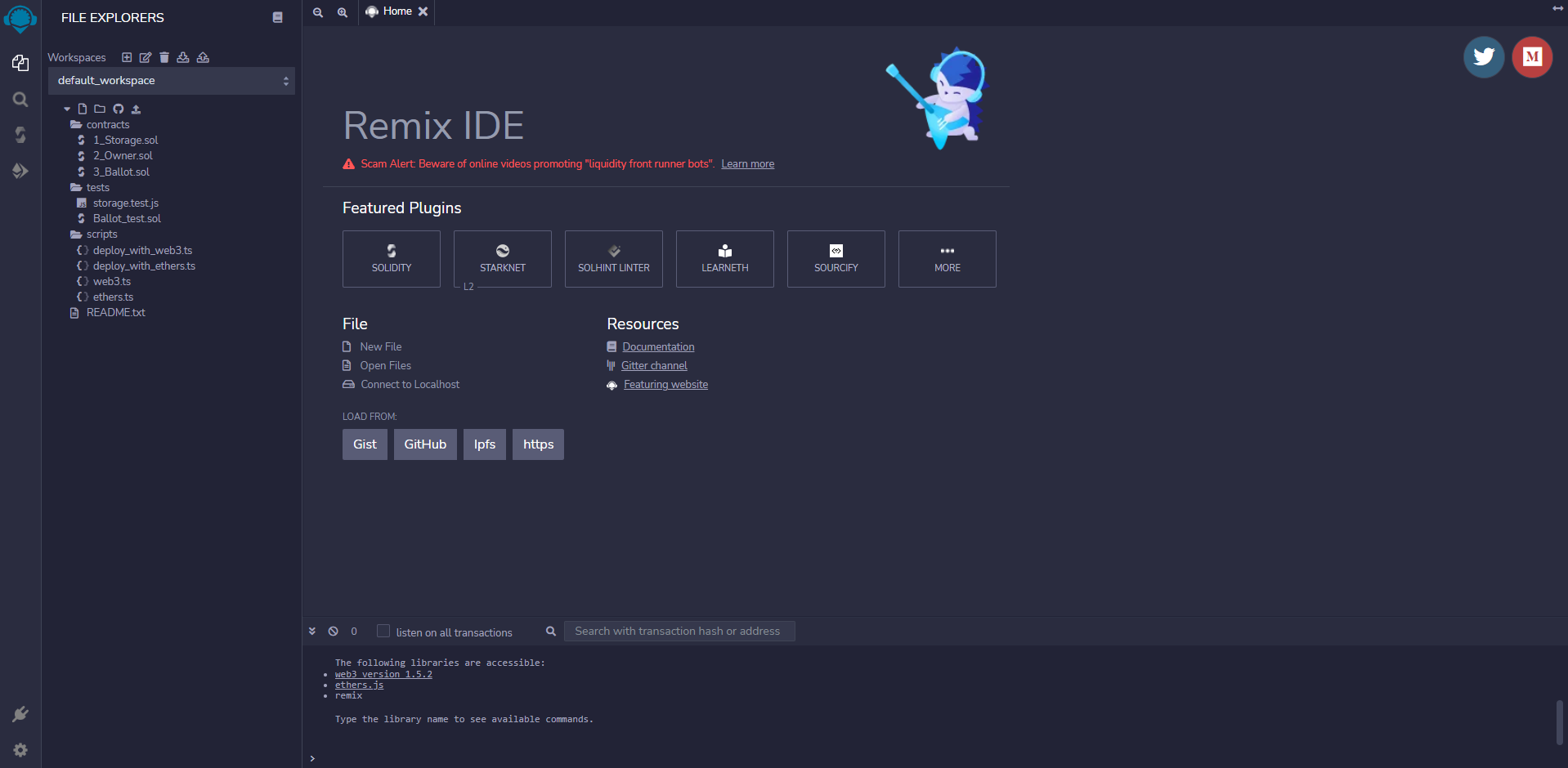
Budeme pracovat v IDE Remix.

## Remix

Remix je považován za jeden z nejlepších IDE pro Solidity. Zdá se, že je dnes nejvíce oficiálně podporovaným integrovaným vývojovým prostředím Ethereum smart-contracting.

**Výhody:**

* Podporuje výhradně prohlížeče
* Obsahuje dobrý kompilátor
* Neřeší komponenty na serverové straně
* Snadná tvorba smluv Ethereum pomocí Solidity

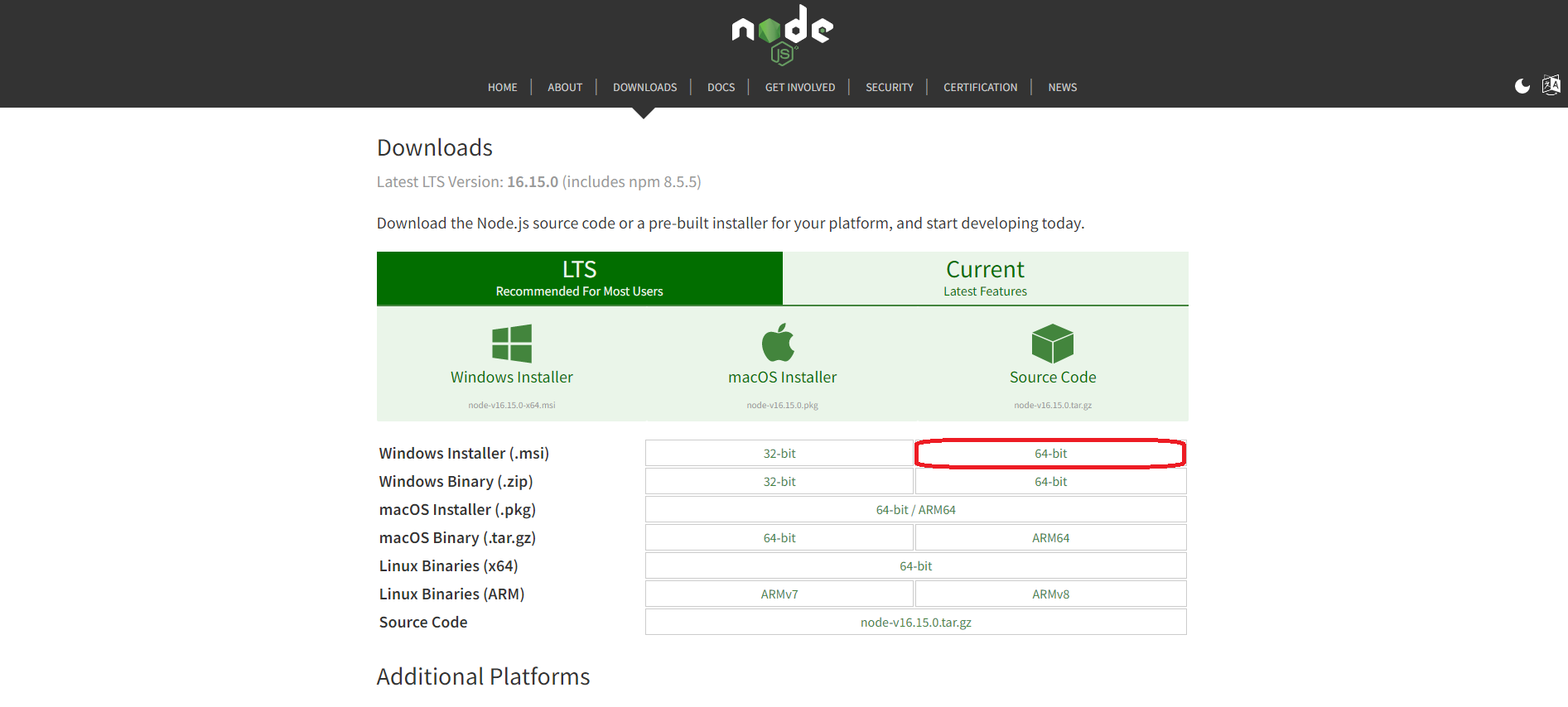


Obrázek 1 – Vývojové prostředí Remix

# Potřebné instalace

## Node.js

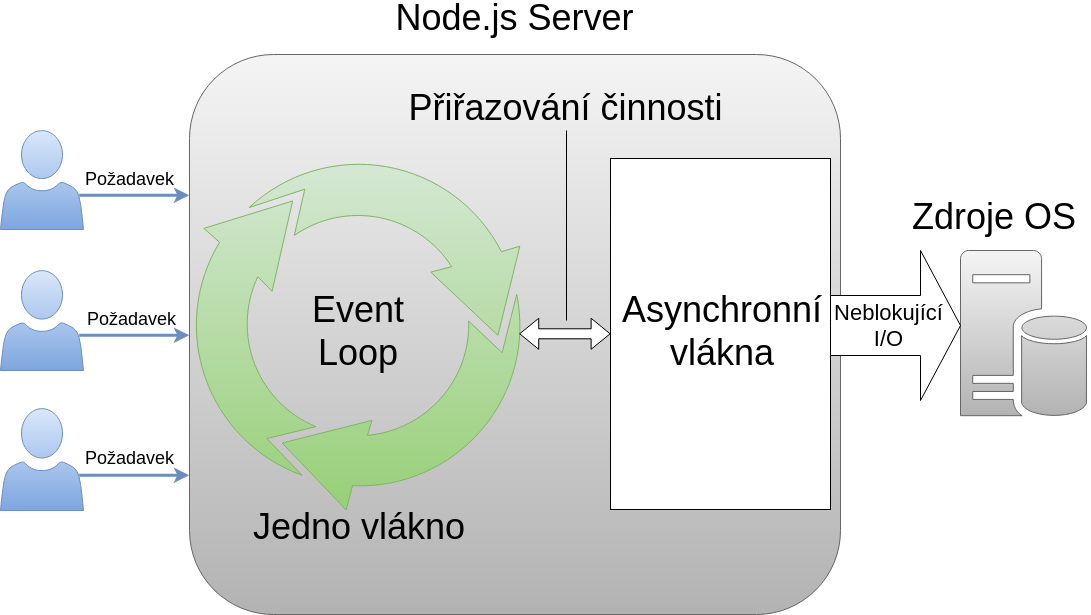
Odkaz na instalaci Node.js: <https://nodejs.org/en/download/>



Obrázek 2 – Stažení instalačního balíčku Node.js

Node.js je prostředí umožňující spouštět kód JavaScriptu mimo webový prohlížeč. Platforma je primárně určena pro běh samostatných serverů poslouchajících na dedikovaném TCP portu, což platí i pro webové frameworky. Používá se zejména pro programování na straně serveru a je primárně nasazen pro neblokující servery řízené událostmi, jako jsou tradiční webové stránky a back-end služby API. Toto má společné např. s jazykem PHP, který má stejné zaměření. JavaScript se tedy díky tomuto prostředí dá používat i na serveru a ne jen na druhém konci, u klienta. Avšak na rozdíl např. od zmíněného PHP je v Node.js kladen důraz na vysokou škálovatelnost, tzn. schopnost obsloužit mnoho připojených klientů naráz.

Jádrem architektury celého Node.js je tzv. smyčka událostí (event loop). Do ní vstupují všechny uživatelské požadavky jako události, které jsou poté přiděleny jednotlivým nezávislým vláknům.



Obrázek 3 - Jádro architektury Node.js

## Metamask

Instalace rozšíření do internetového prohlížeče viz předchozí kapitoly o Metamask.

Vytvoření peněženky a tvorba alespoň 3 účtů, budeme fungovat na testovací síti, tzv. Ropsten Testnet.

## Visual Studio Code

Visual Studio Code je editor zdrojového kódu vyvíjený společností Microsoft pro operační systémy Windows, Linux a macOS. Obsahuje podporu pro Git, zvýraznění syntaxe, kontextový našeptávač a podporu pro ladění a refaktorizaci. Zdrojový kód je svobodný software pod licencí MIT.

Budeme jej používat k tvorbě Github repositáře volební aplikace, a sice pro vývoj front-endové i back-endové části.

# Vytvoření projektu

1. Otevření příkazové řádky v repositáři *cmd*
2. Ověření, zda se správně nainstaloval node.js *node -v*
3. Ověření, zda se správně nainstaloval npm *npm -v*
4. Vytvoření souboru package.json *npm init*
5. Vše defaultně odsouhlasit *<ENTER>*
6. Otevření repositáře ve Visual Studio Code
7. Vytvoření souborů
   1. index.js v kořenovém adresáři
   2. složka public v kořenovém adresáři
   3. index.html ve složce public
   4. podsložka js ve složce public
   5. contract.js ve složce js

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 4 – Struktura a obsah repositáře Volebniapp

1. Instalace npm lokálně v repositáři  *npm install*

(Došlo k vytvoření souboru package-lock.json)

1. První spuštění aplikace na localhostu *node index.js*

(Možná kolize s firewallem -> povolit)

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 5 – Povolení Node.js ve Firewallu systému Windows

1. Propojení účtů peněženky Metamask s webovou aplikací

(nastavení -> connected sites)

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 6 – Propojení účtu s webovou aplikací

# Vytvoření hlasovacího smart contractu

1. Online vývojové prostředí na webu Remix.ethereum.org
2. Vytvoření souboru Voting.sol
3. Sepsání stěžejních bodů kontraktu
4. Nastavení kompilátoru kompatibilního s požadovanou verzí + zaškrtnutí „auto compile“

Obsah obrázku text, monitor, snímek obrazovky, obrazovka

Popis byl vytvořen automaticky

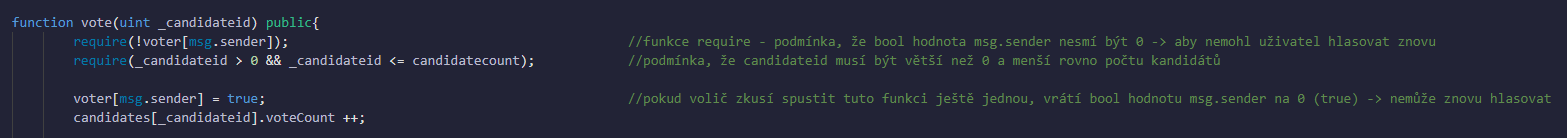
Obrázek – Stěžejní body kontraktu, výběr verze kompilátoru a volba Auto compile

1. Sepsání částí kódu jako deklarace parametrů kandidáta, provázání jejich funkcí pomocí mappingu, konstruktor pro spuštění kontraktu po zapsání do blockchainu a přidání nového kandidáta

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek – Přidání prvních částí kódu kontraktu



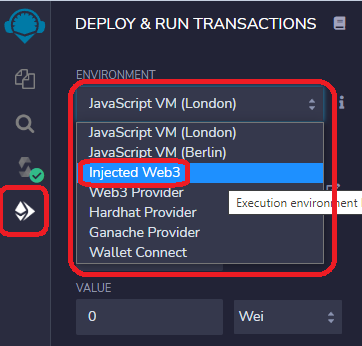
Obrázek – Přidání kódu podmínek hlasování

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek – Deklarace události hlasování a její emitace (spuštění)

1. *Následuje nasazené již vytvořeného kontraktu, je třeba zvolit správné prostředí (Injected Web3).*



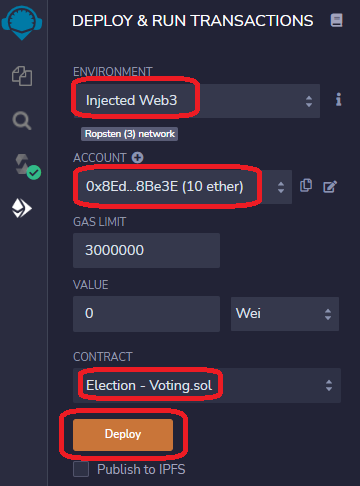
Obrázek – Výběr prostředí Injected Web3

1. Pro zapsání vytvořeného smart contractu do blockchainu Etherea je třeba (i když používáme Testnet) mít na peněžence alespoň nějaké množství Etherea. Na testnetu lze získat zdarma neomezené množství testovacích ETH mincí pro testování. Toto využívají běžně vývojáři v soukromých IT společnostech podílejících se na vývoji v oblasti kryptoměn a blockchainu. Můžeme využít hned několika webů, které přes Ropsten testnet pošlou na naši adresu peněženky ETH. Například [*https://faucet.egorfine.com/*](https://faucet.egorfine.com/)

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek – Účty s nabitými prostředky z Testnetu

1. V následujícím kroku již provedeme samotné nasazení smart contractu, respektive jeho zapsání do blockchainu za cenu tzv. Gas fee, viz předchozí kapitoly.

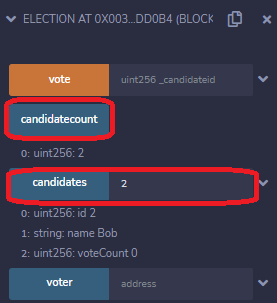
Obrázek – Nasazení vytvořeného kontraktu

Obsah obrázku text, monitor, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek – Úspěšně zapsaný vytvořený kontrakt v blockchainu

1. Můžeme si ověřit funčnost parametrů vytvořeného kontraktu tím, že stiskneme tlačítko candidatecount (zobrazí se 2, jelikož jsme vytvořili Alici a Boba), dále zvolit v kolonce candidates hodnotu 1 nebo 2 a po stisknutí tlačítka candidates se zobrazí detaily kandidáta Alice či Bob, respektive jeho základní parametry, které jsme jim nadefinovali



Obrázek – Ověření funkčnosti parametrů vytvořeného kontraktu

# Propojení s Front-endem

1. Otevřeme si repositář ve Visual Studiu
2. Otevřeme soubor contract.js ve složce public -> js
3. Soubor contract.js obsahuje dvě proměnné

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 16 – Soubor contract.js v prostředí Visual Studio Code

1. První proměnnou SmartContractABI získáme v prohlížečovém vývojovém prostředí Remix, a sice v kartě kompilátoru (předposlední karta), dole ve sloupci se nachází ABI kód ke zkopírování

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek – Získání dat proměnné SmartContractABI

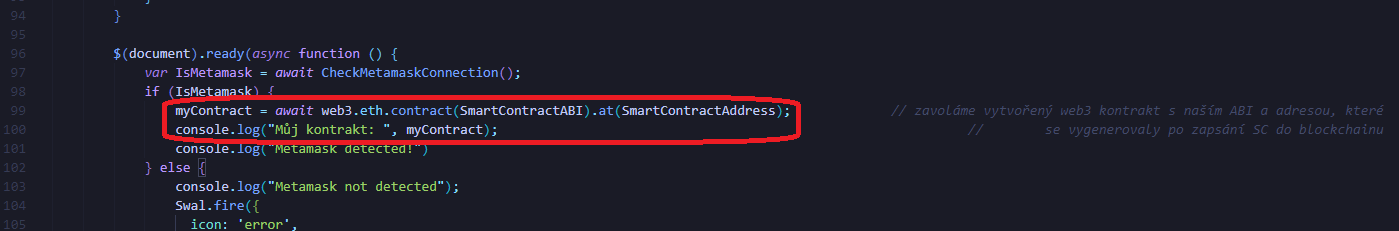
1. Druhou proměnnou SmartContractAddress získáme v prohlížečovém vývojovém prostředí Remix, a sice v kartě deploynutého kontractu zapsaného do blockchainu. Přesněji chceme zkopírovat adresu volby v blockchainu.

Obsah obrázku text, snímek obrazovky, monitor, černá

Popis byl vytvořen automaticky

Obrázek – Získání dat proměnné SmartContractAddress

1. V souboru contract.js na začátku skriptu na řádku 67 deklarujeme novou proměnnou „MyContract“ a v podmínce if (IsMetamask) {…} zavoláme vytvořený web3 kontrakt s naším ABI a adresou, které se vygenerovaly po zapsání Smart Contractu do blockchainu



Obrázek – Zavolání kontraktu z Remixu a jeho vypsání

1. V souboru index.html vytvoříme novou funkci „getCandidate“ s argumentem cad, kdy se hodnota cad stane zároveň argumentem vnitřní funkce (to slouží ke zkopírování parametrů vybraného kandidáta, například při výběru kandidáta 1 (Alice) dojde ke zkopírování parametrů Alice (např. ID atd.))

Obsah obrázku text, snímek obrazovky, obrazovka, černá

Popis byl vytvořen automaticky

Obrázek – Funkce getCandidate

1. Vyvolání funkce getCandidate(1) a getCandidate(2) rovněž v if podmínce IsMetamask na řádku 100 a 101

(došlo k přepsání sloupců Kandidát1 a Kandidát2 na Alice a Bob)

1. Vytvoření funkce OnClick tlačítka Hlasovat, která je svou strukturou velmi podobná předchozí vytvořené funkci getCandidate

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek – Funkce Vote

V této fázi je hlasovací aplikace se smart contractem vytvořena a její základní funkčnost je připravena k použití.

Pokud se připojíte na web a pod jiným účtem v Metamask zvolíte hlasování například pro Boba, pár vteřin (podle nastaveného GasFee limitu) potrvá, než se hals zapíše v blockchainu a poté se po znovunačtení stránky zobrazí u Boba započítaný hlas. Díky blockchainu a vámi napsanému smart contractu.

PS: Při pokusu hlasovat ještě jednou ze stejného Metamask účtu dojde k chybě, respektive Metamask zobrazí error a odkáže se na smart contract, kde jsme nadefinovali, že jeden uživatel, respektive účet Metamasku, nemůže hlasovat více než jedenkrát.

PS2: Visual Studio Code nabízí rozšíření „Solidity“ pro psaní souborů přímo v jazyce Solidity