Deploying the Greeter Smart Contract

A deploy script is created for you by Hardhat automatically. Just change it to a more meaningful name.

```
mv scripts/sample-script.js scripts/deploy.js
```

Take a look at the script and then go ahead and deploy the contract. Since this is a hello world, there isn't anything we need to change.

```
npx hardhat run scripts/deploy.js --network localhost
```

We will get an output that tells us the address (or ID) of this contract.

```
Greeter deployed to: 0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
```

Keep track of this address. We'll need it to interact with our contract.

Accessing from React App

Use the following basic code to access the contract. We'll discuss the code in detail but for now, just put the following content in `src/App.js` file.

```
import './App.css';
import { useState } from 'react';
import { ethers } from 'ethers'   // acts like a backend for our Web3/DApp
import Greeter from './artifacts/contracts/Greeter.sol/Greeter.json'

// Update with the contract address logged out to the CLI when it was deployed
// !!!!!!!! Change this .....
const greeterAddress = "0x5FbDB2315678afecb367f032d93F642f64180aa3"

function App() {
  // store greeting in local state of react. Has nothing to do with the smart contract
  const [greeting, setGreetingValue] = useState()

  // request access to the user's account. This works regardless of the wallet you're u
  async function requestAccount() {
    await window.ethereum.request({ method: 'eth_requestAccounts' });
  }

  // call the smart contract, read the current greeting value
  async function fetchGreeting() {
    if (typeof window.ethereum !== 'undefined') {
      const provider = new ethers.providers.Web3Provider(window.ethereum)
      const contract = new ethers.Contract(greeterAddress, Greeter.abi, provider)
      try {
        const data = await contract.greet()
        console.log('data: ', data)
      } catch (err) {
        console.log("Error: ", err)
      }
    }
  }

  // call the smart contract, send an update
  async function setGreeting() {
    if (!greeting) return
    if (typeof window.ethereum !== 'undefined') {
      await requestAccount()
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = provider.getSigner()
      const contract = new ethers.Contract(greeterAddress, Greeter.abi, signer)
      const transaction = await contract.setGreeting(greeting)
      await transaction.wait()
      fetchGreeting()
    }
  }

  return (
    <div className="App">
      <header className="App-header">
        <button onClick={fetchGreeting}>Fetch Greeting</button>
        <button onClick={setGreeting}>Set Greeting</button>
        <input onChange={e => setGreetingValue(e.target.value)} placeholder="Set greeti
      </header>
    </div>
  );
}

export default App;
```

Go ahead and start the npm test server on your local machine. This will let us interact with out contract through a web frontend.

```
npm start
```

The app should open in your browser.

Open Developer console and then click on the `Get Greeting` button. Take a look at the console where hardhat will output transaction/read details for you.

Then, enter some text in the textbox and click on `Set Greeting`. You should get a popup in MetaMask asking you to select an account. Select the account you have balance in and connect. MetaMask will show you the gas required to run this transaction. Click ahead to run the transaction.

See the console for hardhat that looks like this:

```
eth_sendRawTransaction
  Contract call:       Greeter#setGreeting
  Transaction:         0xf69ac559e520e58f5e647820a3a14cbd1f18861a8bc1b71911dfbb2da8e33a
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            35330 of 35606
  Block #2:            0x098e048300fe12ae0812dd7c067bd274bd848481f57bcbbd94153f0b3b3666

  console.log:
    Changing greeting from 'Hello, Hardhat!' to 'New Greeting!'
```

Again, if you are re-doing the tutorial, reset the account in MetaMask to get rid of the stale state information in MetaMask.