

ThetaScan.io's Smart Contract HQ

Help is divided into 4 sections compile, deploy, interact and Dapp builder.

[1. How to compile a smart contract in Remix.](#)

[2. How to deploy a smart contract on the Theta Blockchain.](#)

[3. How to interact with a smart contract on the Theta Blockchain.](#)

[4. How to build a Dapp that works on the Theta Blockchain.](#)

(All interactions on the Theta Blockchain require the MetaMask extension. [Click Here](#) for detailed directions to setup MetaMask.)

1. Compiling a smart contract with Remix.

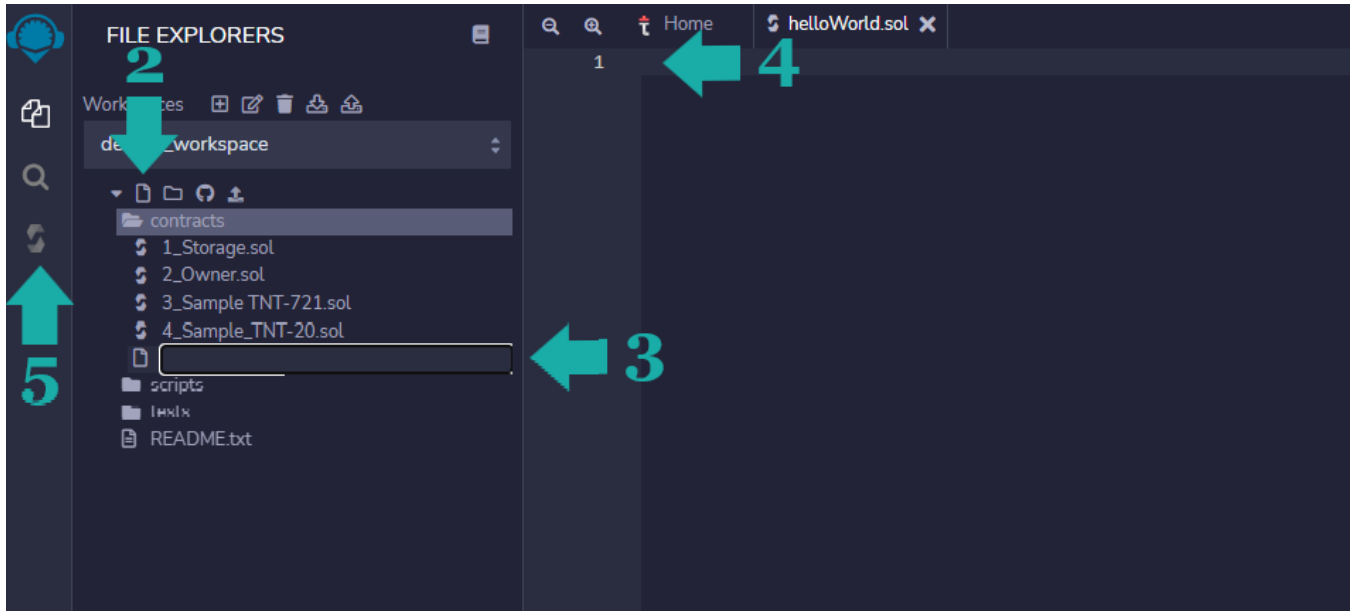
1.1 Click on "Remix for Theta" from the left menu.

1.2 Click on create new file.

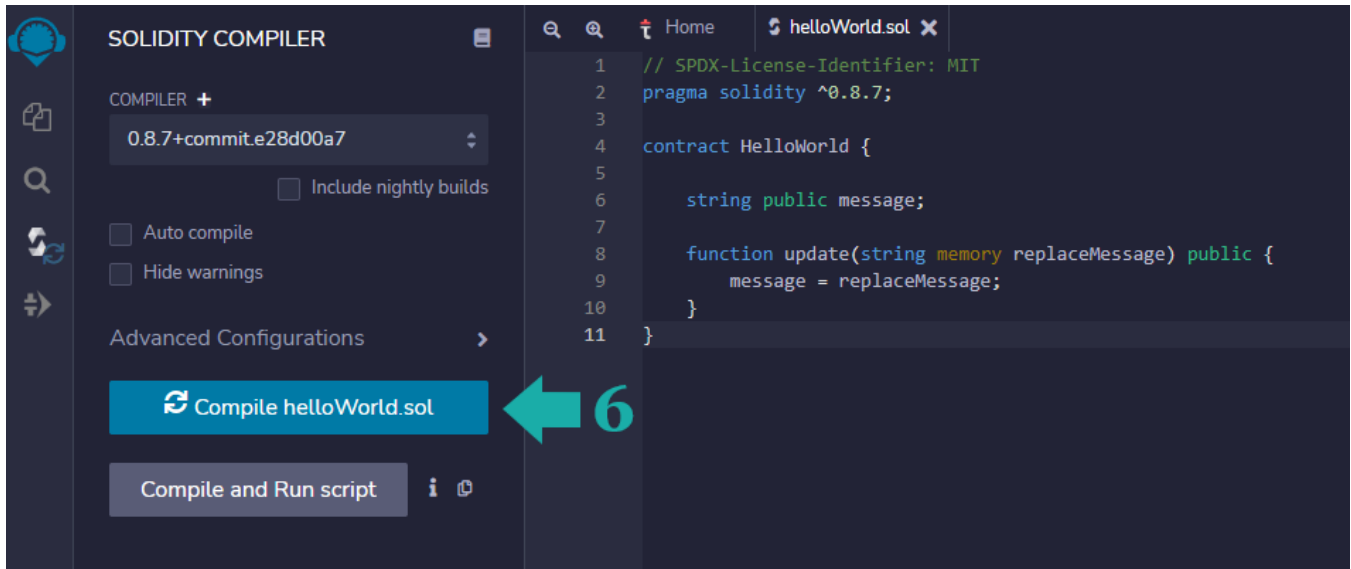
1.3 Name the file you just created (example: helloWorld.sol)

1.4 You can use one of our examples ([Click here](#)) or type in your contract. Copy and paste (ctrl + v) your contract here.

1.5 After your contract is entered click the solidity compile button to show the compiler.



1.6 Click the compile button to build your ABI and Bytecode. After it compiles, you will see it directly below this button.



You will need both sets of code to deploy your contract in the next section.

1.7 Click on ABI to copy your ABI code to the clipboard.

1.8 Click on Bytecode to copy your Bytecode code to the clipboard.

The screenshot displays the Solidity Compiler interface in the Remix IDE. On the left sidebar, the compiler version is set to '0.8.7+commit.e28d00a7'. Below this, there are options for 'Include nightly builds', 'Auto compile', and 'Hide warnings'. A large blue button labeled 'Compile helloWorld.sol' is prominent. Underneath, there are buttons for 'Publish on Ipfes', 'Publish on Swarm', and 'Compilation Details'. At the bottom of the sidebar, there are two buttons: 'ABI' and 'Bytecode'. A red arrow with the number '7' points to the 'ABI' button, and another red arrow with the number '8' points to the 'Bytecode' button. The main editor area shows the Solidity code for a 'HelloWorld' contract, including a pragma statement and a function named 'update'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 contract HelloWorld {
5
6     string public message;
7
8     function update(string memory replaceMessage) public {
9         message = replaceMessage;
10    }
11 }
```

If you would like to learn more about Remix feel free to [click here](#) to watch an in depth video explaining how to use it.

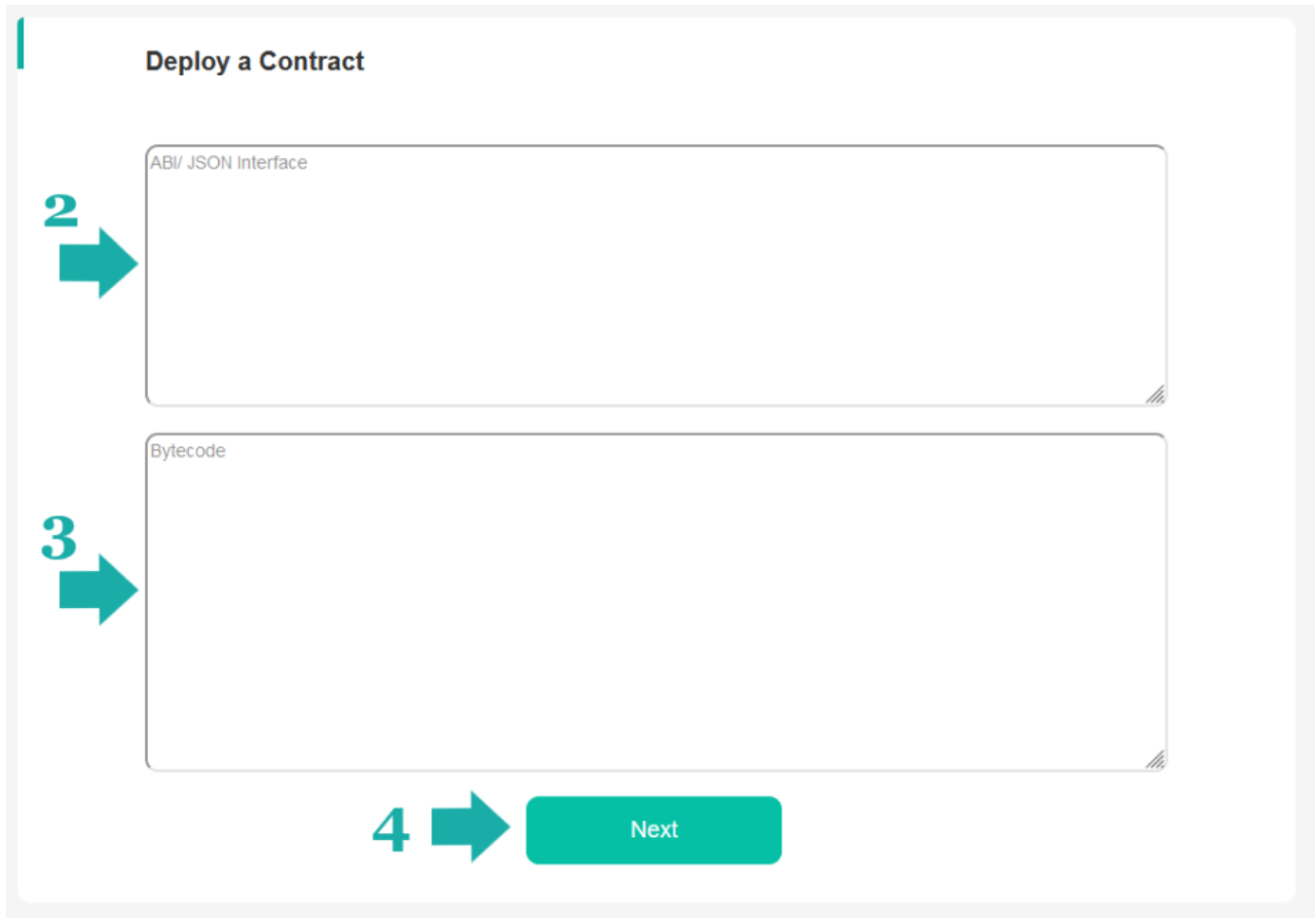
2. Deploying a smart contract.

2.1 Click on "Deploy Contract" from the left menu.

2.2 Copy and paste your ABI interface into the text box. You will use the code you copied from step 7 and 8 in the "Compiling a smart contract with Remix" section.

2.3 Copy and paste your Bytecode code into the text boxes.

2.4 Click Next



Deploy a Contract

ABI/ JSON Interface

2 →

Bytecode

3 →

4 → [Next](#)

Connect

2.5 Click on Connect in the upper right corner.

(If you are already connected your address and tFuel balance will be display.)

2.6 If you have no variables in your constructor the gas price will auto calculate. If you have variables enter them then you can estimate the gas needed to deploy the contract. (You do not have to use this button)

2.7 Click Deploy to send you contract to the Theta Network.

Deploy a Contract

Constructor Variables :None

Estimated Transaction Fee: 1.61 tFuel

Estimated Value in USD: \$0.088

Transaction Hash:

Contract Address:

6



Estimate Gas

Deploy



7

2.8 Metamask will open. This is your last chance to reject the transaction. If you are ready to send it to the blockchain click Confirm.

DETAILS DATA

EDIT

Estimated gas fee ⓘ 1.611952
1.611952 TFUEL
Site suggested Max fee: 1.611952 TFUEL

Total 1.611952
1.611952 TFUEL
Amount + gas fee Max amount: 1.611952 TFUEL

Reject Confirm

2.9 The hash will appear quickly. But it can take up to 15 seconds for the contract address to display. Once it displays click copy to copy it to your clipboard.
(Save the contract address in a safe place you will need it in sections 3 and 4.)

Deploy a Contract


Constructor Variables :None

Estimated Transaction Fee: 1.61 tFuel

Estimated Value in USD: \$0.088

Transaction Hash: 0x3424a033e1f45d26e5cd192c345d2d4a31aa345df022f3eff21f341c3988b6f

Contract Address: 0xc031256fb3dab95678012d9ed423c93a4c4954c7

9

Copy

[Estimate Gas](#) [Deploy](#)

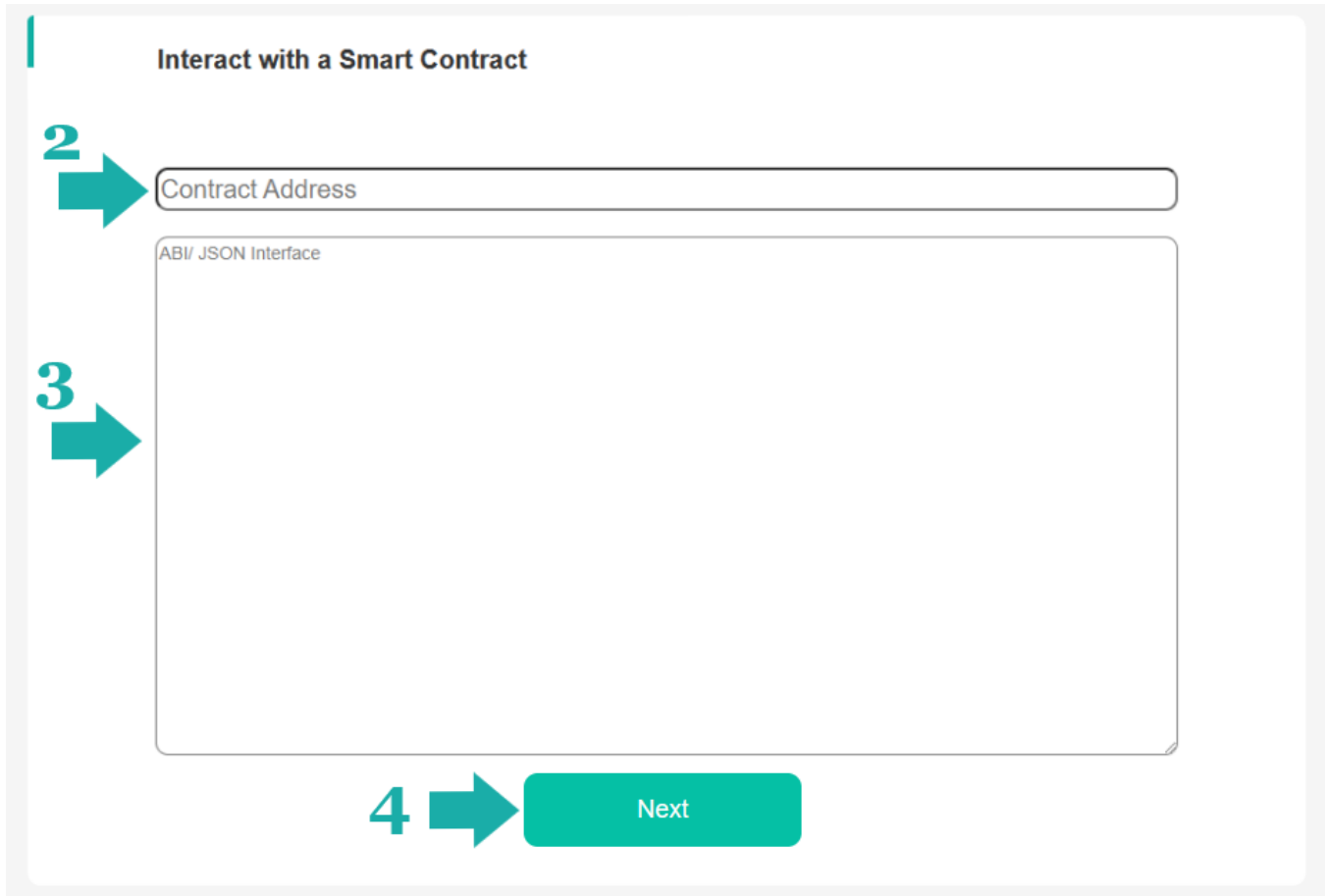
3. Interacting with a smart contract.

3.1 Click on "Interact With" from the left menu.

3.2 Copy and paste the contract address. (From deploy step 2.9)

3.3 Copy and paste the contract's ABI interface to interact with a contract. (From Remix step 1.7)

3.4 Click Next to interact with a contract.



Connect

3.5 Click on Connect in the upper right corner.

(If you are already connected your address and tFuel balance will be display.)

3.6 Select the function you would like to use with the contract.

3.7 The interface will appear.

3.8 Click interact to read or write to the contract. If the function is a read function click interact and the results will display, no more action is needed. If the function is a write function enter the inputs if needed and click interact.

Interact with a Smart Contract

update

6

replaceMessage

7

Interact

8

Transaction Hash:

3.9 Only if it is not a read function. Metamask will open. This is your last chance to reject the transaction. If you are ready to send it to the blockchain click Confirm.

[EDIT](#)

| | |
|--------------------------|--------------------------------|
| Estimated gas fee | 0.191808 |
| <i>Site suggested</i> | 0.191808 TFUEL |
| | Max fee: 0.191808 TFUEL |

| | |
|-------------------------|-----------------------------------|
| Total | 0.191808 |
| <i>Amount + gas fee</i> | 0.191808 TFUEL |
| | Max amount: 0.191808 TFUEL |

[Reject](#) [Confirm](#)


3.10 If it is a write function the transaction hash will appear below the interact button. If it is a call function no hash will display. (You can click View on Thetascan.io to see the transaction hash details on the block explorer in a new window.)

Interact with a Smart Contract

[update](#)

[Interact](#)

Transaction Hash: 0x81a8d45745fac5438e9dd32634f45f553223eeb6c3befe3a43d34d75e1ff183f [View on Thetascan.io](#)

 **10**

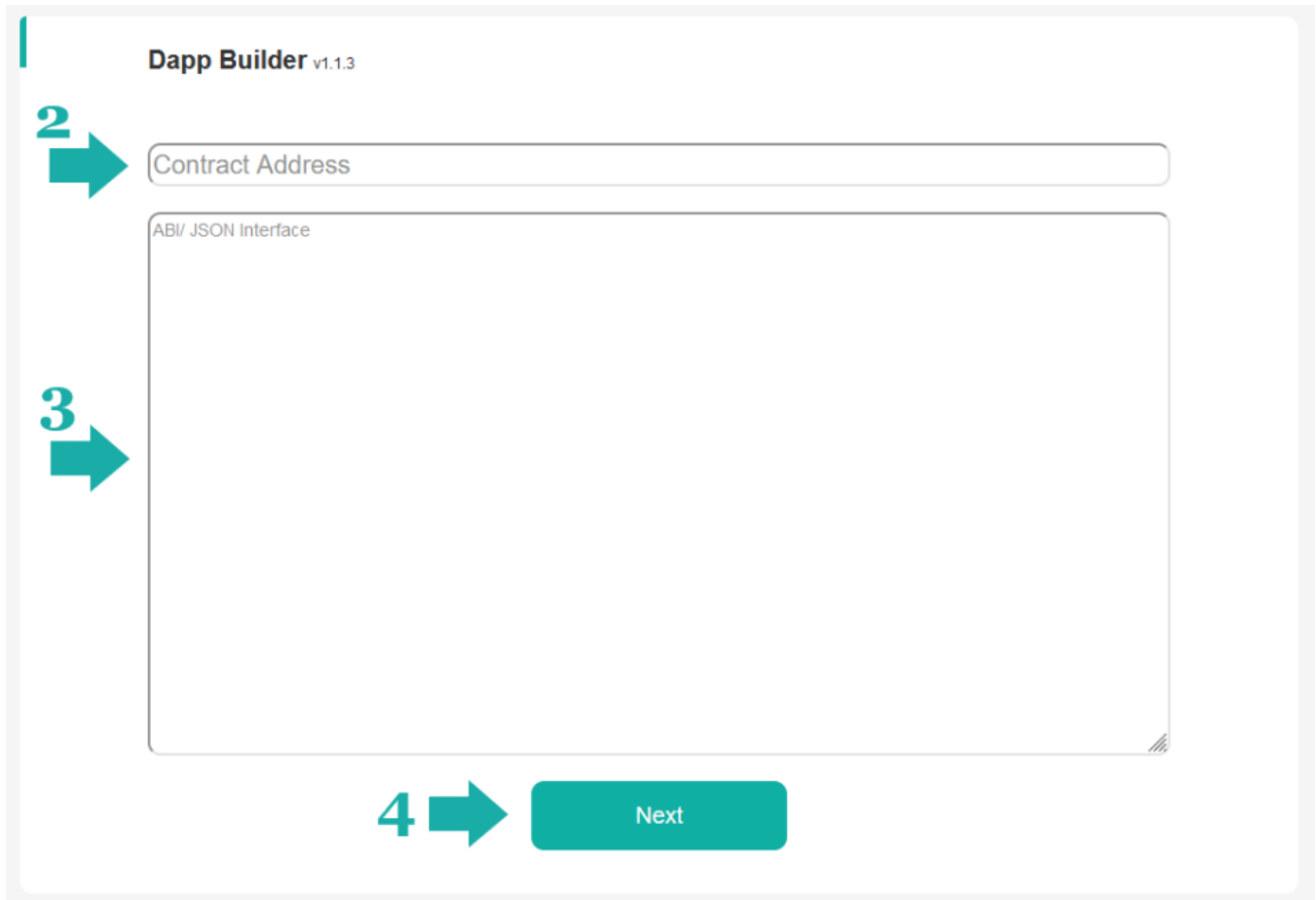
4. Building a Dapp on the Theta Blockchain.

4.1 Click on "Dapp Builder" from the left menu.

4.2 Copy and paste in the contract address (From deploy step 2.9)

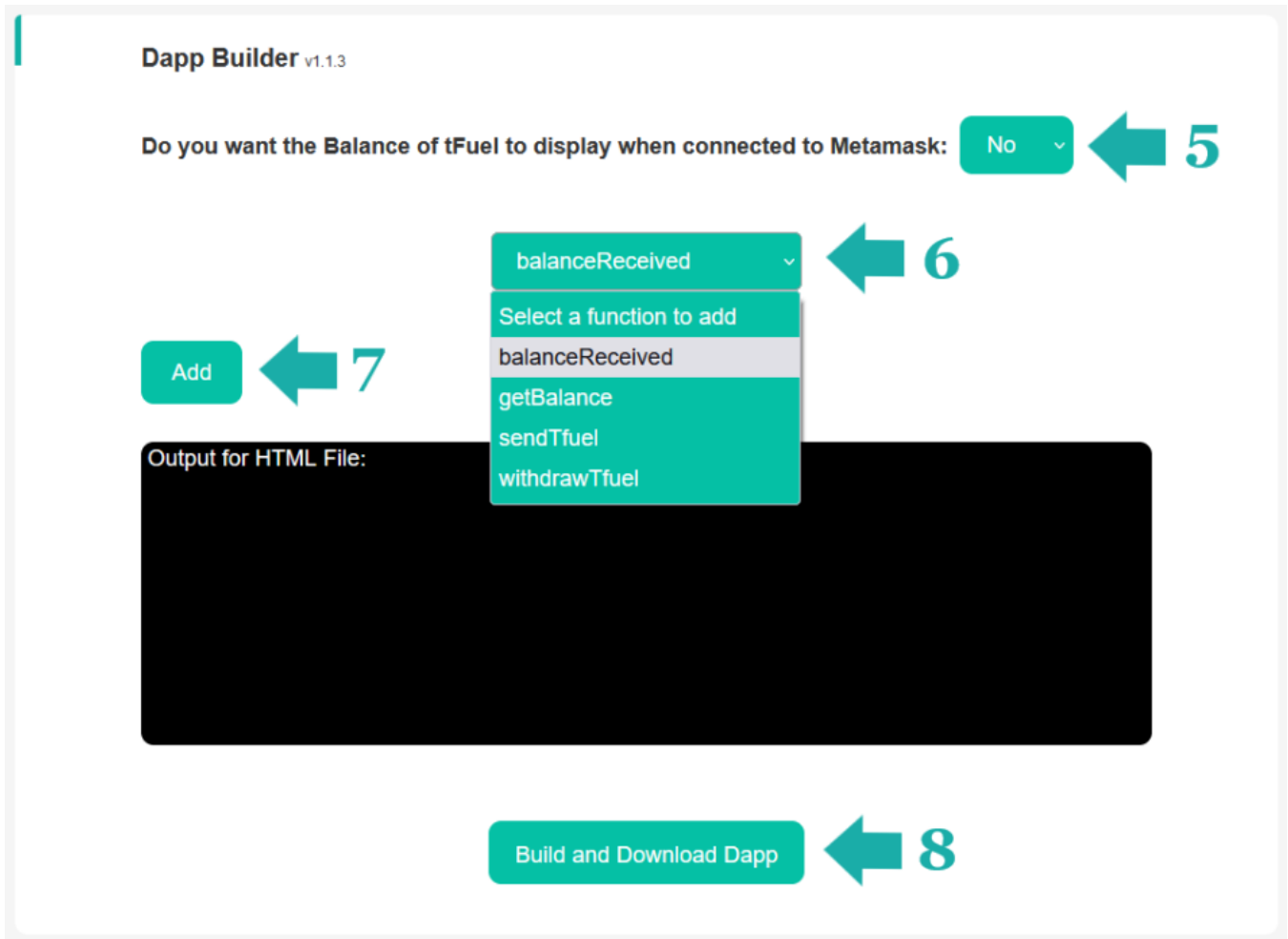
4.3 Copy and paste in the contract ABI. (From Remix step 1.7)

4.4 Click Next.



The screenshot shows the "Dapp Builder v1.1.3" interface. It features a teal header bar with the title "Dapp Builder v1.1.3". Below the header, there are two main input areas: a text input field labeled "Contract Address" and a larger text area labeled "ABI/JSON Interface". A teal arrow with the number "2" points to the "Contract Address" field. Another teal arrow with the number "3" points to the "ABI/JSON Interface" area. At the bottom of the interface, there is a teal button labeled "Next" with a teal arrow and the number "4" pointing to it.

- 4.5 Choose how you want the connect button to appear. It can display just the user's address or the address with tFuel balance.
- 4.6 Choose a function you want in your Dapp. You can select just one function or one by one you can add them all. (After you select a function the Add will appear.)
- 4.7 Click "Add" Some functions will require additional information such as how much tFuel do you want this button to send to the contract when clicked. (If you added a function, you do not want in your Dapp refresh the page to start over.)
- 4.8 Click "Build and Download" to save your Dapp as an HTML file.



- 4.9 Copy and paste or upload the file into your web server. (It must be ran from a web server such as Apache or NGINX.)
- 4.10 Edit the HTML file to match your website by changing the CSS, adding images, etc...

*Thetascan.io will never ask for your private key. All interactions with the blockchain in the Smart Contract HQ use the Metamask Wallet to interact with it.