---

title: "Waffle say hello world tutorial with hardhat and ethers" description: Make your first Waffle project with hardhat and ethers.js author: "MiZiet" tags: ["waffle", "smart contracts", "solidity", "testing", "hardhat", "ethers.js"] skill: beginner lang: en published: 2020-10-16

In this Waffle tutorial, we will learn how to set up a simple "Hello world" smart contract project, usinghardhat and ethers.js. Then we will learn how to add a new functionality to our smart contract and how to test it with Waffle.

Let's start by creating a new project:

`bash yarn init`

or

`bash npm init`

and installing required packages:

`bash yarn add -D hardhat @nomiclabs/hardhat-ethers ethers @nomiclabs/hardhat-waffle ethereum-waffle chai`

or

`bash npm install -D hardhat @nomiclabs/hardhat-ethers ethers @nomiclabs/hardhat-waffle ethereum-waffle chai`

Next step is creating a sample hardhat project by running`npx hardhat`.

```bash 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 8888888888 8888b. 888d888 .d88888 88888b. 8888b. 888888 888 888 "88b 888P" d88" 888 888 "88b "88b 888 888 888 .d888888 888 888 888 888 888 .d888888 888 888 888 888 888 888 Y88b 888 888 888 888 888 Y88b. 888 888 "Y888888 888 "Y88888 888 888 "Y888888 "Y888

Welcome to Hardhat v2.0.3

? What do you want to do? … ❯ Create a sample project Create an empty hardhat.config.js Quit ```

Select `Create a sample project`

Our project's structure should look like this:

`MyWaffleProject ├── contracts │ └── Greeter.sol ├── node_modules ├── scripts │ └── sample-script.js ├── test │ └── sample-test.js ├── .gitattributes ├── .gitignore ├── hardhat.config.js └── package.json`

## Now let's talk about some of these files: {#now-lets-talk}

- Greeter.sol - our smart contract written in solidity;

```solidity contract Greeter { string greeting;

constructor(string memory _greeting) public { console.log("Deploying a Greeter with greeting:", _greeting); greeting = _greeting; }

function greet() public view returns (string memory) { return greeting; }

function setGreeting(string memory _greeting) public { console.log("Changing greeting from '%s' to '%s'", greeting, _greeting); greeting = _greeting; } } ```

Our smart contract can be divided into three parts:

1. constructor - where we declare a string type variable called`greeting`,
2. function greet - a function that will return the`greeting` when called,
3. function setGreeting - a function that allows us to change the`greeting` value.
4. sample-test.js - our tests file

```js describe("Greeter", function () { it("Should return the new greeting once it's changed", async function () { const Greeter = await ethers.getContractFactory("Greeter") const greeter = await Greeter.deploy("Hello, world!")

```
await greeter.deployed()
expect(await greeter.greet()).to.equal("Hello, world!")

await greeter.setGreeting("Hola, mundo!")
expect(await greeter.greet()).to.equal("Hola, mundo!")
~~~

}) }) ```

Next step consists of compiling our contract and running tests: {#compiling-and-testing}

Waffle tests use Mocha (a test framework) with Chai (an assertion library). All you have to do is run `npx hardhat test` and wait for the following message to appear.

`bash ✓ Should return the new greeting once it's changed`

Everything looks great so far, let's add some more complexity to our project{#adding-complexity}

Imagine a situation where someone adds an empty string as a greeting. It wouldn't be a warm greeting, right?
Let's make sure that doesn't happen:

We want to use solidity's `revert` when someone passes an empty string. A good thing is that we can easily test this functionality with Waffle's chai matcher `to.be.revertedWith()`.

```js it("Should revert when passing an empty string", async () => { const Greeter = await ethers.getContractFactory("Greeter") const greeter = await Greeter.deploy("Hello, world!")

await greeter.deployed() await expect(greeter.setGreeting("")).to.be.revertedWith( "Greeting should not be empty" ) }) ```

Looks like our new test didn't pass:

```bash Deploying a Greeter with greeting: Hello, world! Changing greeting from 'Hello, world!' to 'Hola, mundo!' ✓ Should return the new greeting once it's changed (1514ms) Deploying a Greeter with greeting: Hello, world! Changing greeting from 'Hello, world!' to '' 1) Should revert when passing an empty string

1 passing (2s) 1 failing ```

Let's implement this functionality into our smart contract:

`solidity require(bytes(_greeting).length > 0, "Greeting should not be empty");`

Now, our setGreeting function looks like this:

`solidity function setGreeting(string memory _greeting) public { require(bytes(_greeting).length > 0, "Greeting should not be empty"); console.log("Changing greeting from '%s' to '%s'", greeting, _greeting); greeting = _greeting; }`

Let's run tests again:

```bash ✓ Should return the new greeting once it's changed (1467ms) ✓ Should revert when passing an empty string (276ms)

2 passing (2s) ```

Congrats! You made it :)

## Conclusion {#conclusion}

We made a simple project with Waffle, Hardhat and ethers.js. We learned how to set up a project, add a test and implement new functionality.
~~~

For more great chai matchers to test your smart contracts, check official Waffle's docs.