

Hands-On Blockchain

This document provides additional links and insights you may find useful!

# Useful links

* Hands-On Github Repository: <https://github.com/Leo-Besancon/ISCTE-IUL-HandsOnBlockchain>
* Remix (Online IDE): <https://remix.ethereum.org>

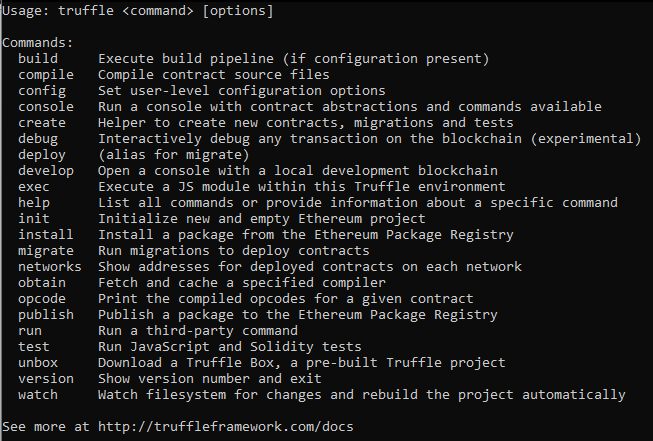
## Documentation

* Solidity: <https://solidity.readthedocs.io/en/v0.6.2/>
* Remix IDE: <https://remix-ide.readthedocs.io/en/latest/>
* Truffle: <https://www.trufflesuite.com/docs/truffle/overview>
* OpenZeppelin: <https://docs.openzeppelin.com/openzeppelin/>

# Get your first project running

To get familiar with Solidity code, you can look at the code for ERC20 implementation

## Truffle project



## Remix project

# Smart-contract development constraints

* Be careful, not every application needs to be a smart-contract! Blockchain is a powerful tool, but it has very specific use cases. For many projects, Blockchain isn’t the best-suited technology to implement it.
* <https://www.coindesk.com/three-smart-contract-misconceptions>

# Best practices

## Proxy contracts

* <https://medium.com/@blockchain101/the-basics-of-upgradable-proxy-contracts-in-ethereum-479b5d3363d6>

## Test your code

* <https://www.trufflesuite.com/docs/truffle/testing/testing-your-contracts>

## Audit and share your code

* Verify you smart-contract on Etherscan! Users will know exactly what code they are calling when they interact with your contracts.

## Single Page Application to interact with your smart-contract

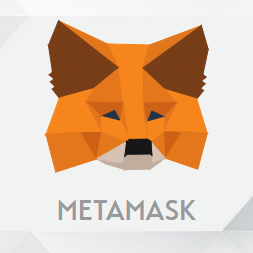
A Decentralized application is much more than just smart-contracts.

Users should be able to interact with your contracts without having to worry about many aspects, such as security details.

OpenZeppelin Starter kits: a boilerplate solution to develop your smart-contracts and a React Single Page Application to interact with them!

**npx oz unpack tutorial**

### Metamask: A browser extension to connect to Web3, and Ethereum



<https://metamask.io/>