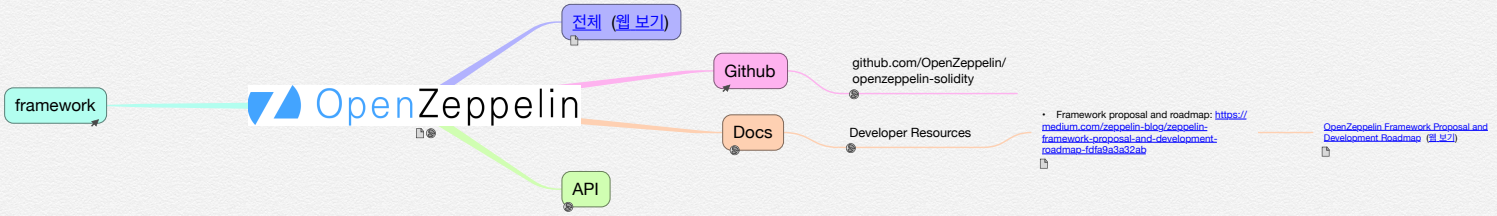
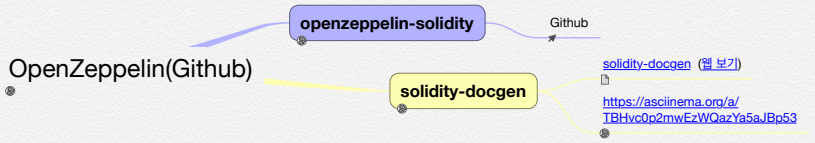
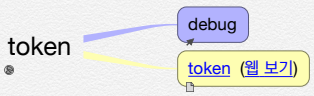
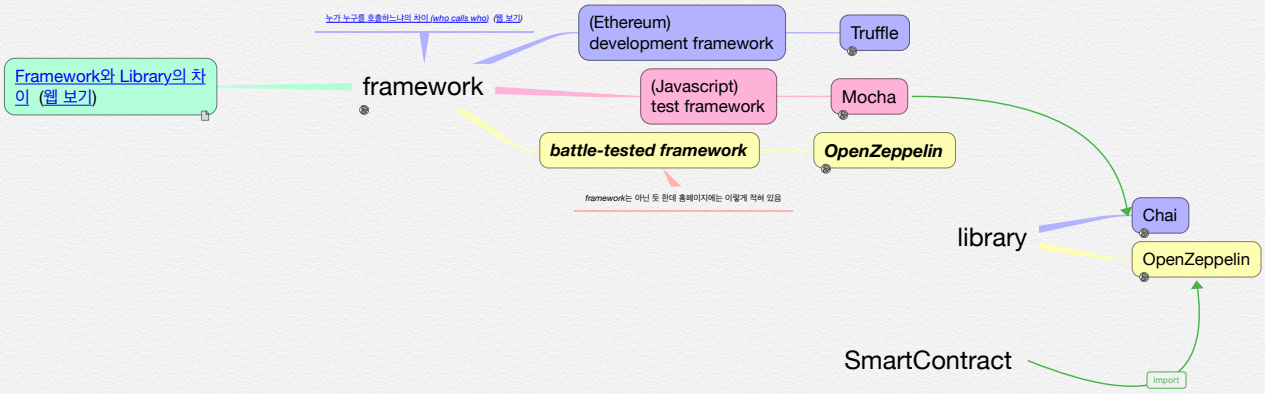
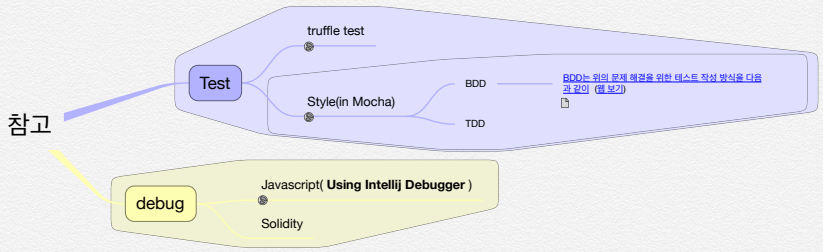

framework
OpenZeppelin
전체 (웹 보기)
Github
github.com/OpenZeppelin/
openzeppelin-solidity
Docs
Developer Resources
• Framework proposal and roadmap: https://medium.com/zeppelin-blog/zeppelin-framework-proposal-and-development-roadmap-fdfa9a3a32ab
OpenZeppelin Framework Proposal and Development Roadmap (웹 보기)
API


OpenZeppelin(Github)
openzeppelin-solidity
Github
solidity-docgen
solidity-docgen (웹 보기)
https://asciinema.org/a/TBHvc0p2mwEzWQazYa5aJBp53


token
debug
token (웹 보기)


Framework와 Library의 차이 (웹 보기)
framework
누가 누구를 호출하느냐의 차이 (who calls who) (웹 보기)
(Ethereum) development framework
Truffle
(Javascript) test framework
Mocha
battle-tested framework
OpenZeppelin
framework는 아닌 듯 한데 홈페이지에는 이렇게 적혀 있음
library
Chai
OpenZeppelin
SmartContract
import


참고
Test
truffle test
Style(in Mocha)
BDD
TDD
BDD는 위의 문제 해결을 위한 테스트 작성 방식을 다음과 같이 (웹 보기)
debug
Javascript(Using Intellij Debugger)
Solidity

/ token

https://github.com/OpenZeppelin/openzeppelin-solidity/tree/master/contracts/token

/ token / debug

ithoughts://open?topic=23AEC12E-EA30-408F-960C-BE17E7ADB914

/ token / token (웹 보기)

[token-iThoughtsX]

2e0713becbdfb7a3112139641402l3f8dc282cd1
2e0713b Francisco Giordano frangio.1@gmail.com on 2018. 9. 1. at AM 4:13
https://github.com/OpenZeppelin/openzeppelin-solidity/commit/2e0713becbdfb7a3112139641402l3f8dc282cd1

Rename ERC interfaces to I prefix (#1252)

- rename ERC20 to IERC20
- move ERC20.sol to IERC20.sol
- rename StandardToken to ERC20

ERC20.sol
Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
람보(원제:**First blood**)
팀 또는 게이머 중 제일 먼저 죽었다는 뜻. 줄여서 퍼블이라고도 한다.
https://namu.wiki/w/퍼스트%20블러드
https://ko.coinjinja.com/ico/firstblood
http://blog.naver.com/PostView.nhn?blogId=jruits&logNo=221016503577&categoryNo=12&parentCategoryNo=0&viewDate=&currentPage=23&postListTopCurrentPage=&from=postList

contract('ERC20',function([_,owner,recipient,anotherAccount]){

webpack:///./packages/truffle-core/lib/test.js:235
tests(accounts);
=> 여기에서 contract의 callback 함수를 실행
accounts는 array이기 때문에 []로 받을 수도 있음.
즉,
_ : 0번째의 element
owner : 1번째의 element
recipient : 2번째의 element
anotherAccount : 3번째의 element

this
proto

approve allowance
http://joojis.tistory.com/entry/Solidity-프로그래밍-3-이더리움-토큰의-구현
allowance는 balanceOf의 approval 버전입니다.

Q Mocha 로 작성한 test descript 만을 추출하는 방법?

test/token/ERC20/ERC20.test.js:509
...
```
        describe(description, function () {
          beforeEach('burning', async function () {
            const { logs } = await this.token.burn(owner, amount);
            this.logs = logs;
          });
```
=> burn 함수로 어떤 계정이든 burn할 수 있음

```
contract ERC20Mock is ERC20 {
...
  function burn(address _account, uint256 _amount) public {
    _burn(_account, _amount);
  }
```
=> Mock up 으로 만든 contract 에서 가능하도록 warrping 하였기 때문에 위와 같이 가능

```
contract ERC20Burnable is ERC20 {
...
  function burn(uint256 _value) public {
    _burn(msg.sender, _value);
  }
```
=> 자기 자신만 가능하도록 warrping 하는 것이 보안에 좋음

burn 은 ERC20의 공식 Interface는 아님
https://github.com/ethereum/EIPs/issues/20

_burnFrom
https://github.com/OpenZeppelin/openzeppelin-solidity/compare/master...jesoos:patch-1

Build Secure Smart Contracts in Solidity

OpenZeppelin is a battle-tested **framework** of **reusable** smart contracts for Ethereum and other EVM and eWASM blockchains.

Reduce the risk of vulnerabilities in your applications by using **standard, tested, community-reviewed code**.
https://openzeppelin.org

/전체 *(웹 보기)*

https://openzeppelin.org
[framework-iThoughsX]

Ethereum flavored WebAssembly(eWASM)
https://github.com/ewasm/design

현재 Ethereum의 Roadmap에는 EVM을 Web Assembly 기반인 **eWASM**으로 변경할 계획을 가지고 있습니다.
https://steemit.com/kr/@coinsight/eos

Source Code 분석 방법 중 test 프로그램 debugging을 선택
[참고-iThoughsX]

It provides implementations of standards ... , as well as Solidity components ... .
https://github.com/OpenZeppelin/openzeppelin-solidity#
Docs
API
https://github.com/OpenZeppelin/solidity-docgen

```
contracts
├─── access
├─── bounties
├─── crowdsale
├─── cryptography
├─── drafts
├─── examples
├─── introspection
├─── lifecycle
├─── math
├─── mocks
├─── ownership
├─── payment
├─── token
└─── utils
```

/ Github

ithoughts://open?topic=A5F24099-6BE4-4463-8589-61E72D1AD2CF

/ Github / github.com/OpenZeppelin/openzeppelin-solidity

https://github.com/OpenZeppelin/openzeppelin-solidity

/ Docs

https://openzeppelin.org/api/docs/open-zeppelin.html

/ Docs / Developer Resources

https://openzeppelin.org/api/docs/open-zeppelin.html#developer-resources

OpenZeppelin is a new smart contract development framework for the Ethereum Virtual Machine (EVM) focused on **security, modularity, and code reusability**.

Motivation
Smart contract security is hard.
...

Principles
The core development principles we want to base OpenZeppelin on are:
Security in Depth
We strive to provide **secure, tested, audited code**. To achieve this, we need to match intention with function. **Thus, documentation, code clarity, community review and security discussions are fundamental.**
Simple and Modular
Simpler code means easier audits, and better understanding of what each component does. We look for small files, small contracts, and small functions. If you can **separate** a contract into two **independent** functionalities you should probably do it.
Naming Matters(이름 지정 문제 ?)
We take our time with picking names. Code is going to be written once, and read hundreds of times. Renaming for clarity is encouraged.
Tests
Write tests for all your code. We encourage Test Driven Development so we know when our code is right. Even though not all code in the repository is tested at the moment, we aim to test every line of code in the future.
Check pre-conditions and post-conditions
A very important way to prevent vulnerabilities is to catch a contract's **inconsistent state** as early as possible. This is why we want functions to check pre- and post-conditions for executing its logic. When writing code, ask yourself what you are expecting to be true before and after the function runs, and express it in code.
Code Consistency
Consistency on the way classes are used is paramount to an easier understanding of the library. The codebase should be as unified as possible. Read existing code and get inspired before you write your own. Follow the style guidelines. Don't hesitate to ask for help on how to best write a specific piece of code.
Regular Audits
Following good programming practices is a way to reduce the risk of vulnerabilities, but professional code audits are still needed. We will perform regular code audits on major releases, and hire security professionals to provide independent review.

Implementation
We're starting with Solidity tools because Ethereum is currently the most popular smart contract development platform. Working with Solidity also makes our tools compatible

with Rootstock and some private blockchain systems like IBM's Blue Horizon.
But our vision of improving smart contract development security standards is platform agnostic. Our plans for the future involve working on tools for Bitcoin, Tendermint, Bloq Ora, or whatever platform developers are using to build smart contract apps.

/ Docs / Developer Resources / • Framework proposal … / OpenZeppelin Framework P…

https://blog.zeppelin.solutions/zeppelin-framework-proposal-and-development-roadmap-fdfa9a3a32ab

Manuel AraozFollow
Project lead at OpenZeppelin. CTO at Zeppelin Solutions
Sep 15, 2016

https://en.wikipedia.org/wiki/Codebase
Typically, a codebase includes only human-written source code files

Consistency on the way classes are used is paramount to an easier understanding of the library.
Q 여기서 말하는 classes ?

...
Read existing code and get inspired before you write your own.
...

...
But our vision of improving smart contract development security standards is platform agnostic.
...

/ API

https://openzeppelin.org/api/docs/crowdsale_Crowdsale.html

/ framework

ithoughts://open?topic=0C3AFB61-9DE4-42C0-9DD9-402999AE7AC6

/ OpenZeppelin(Github)

https://github.com/OpenZeppelin

/ OpenZeppelin(Github) / openzeppelin-solidity

https://github.com/OpenZeppelin/openzeppelin-solidity

/ OpenZeppelin(Github) / openzeppelin-solidity / Github

ithoughts://open?topic=D653FF74-E06C-4A28-B76B-9242CEAA9C0D

/ OpenZeppelin(Github) / solidity-docgen

https://github.com/OpenZeppelin/solidity-docgen#solidity-documentation-generator

/ OpenZeppelin(Github) / solidity-docgen / solidity-docgen (웹 보기)

[solidity-docgen-iThoughtsX]

Prerequisites

git clone git@github.com:OpenZeppelin/openzeppelin-solidity.git ~/dev/smartcontractz
cd ~/dev/smartcontractz
mkdir docs
cd docs
docusaurus-init
mv docs-examples-from-docusaurus docs
cd website
mv blog-examples-from-docusaurus blog
solidity-docgen --exclude examples ~/dev/smartcontractz ~/dev/smartcontractz/contracts ~/dev/smartcontractz/docs
npm start
http://localhost:3000/docs/crowdsale_Crowdsale

/Users/roisun/dev/smartcontractz/docs/website/siteConfig.js:42
    // {doc: 'doc4', label: 'API'},
    {doc: 'crowdsale_Crowdsale', label: 'API'},

[참고]
Error: Solidity compiler not found. Please, add solc to path or define environment variable SOLC_PATH.

export SOLC_PATH=/usr/local/bin/solcjs

Error: Command line operation failed with code 1.
Standard error output: Invalid option selected, must specify either --bin or --abi
https://github.com/ConsenSys/mythril/issues/82
Hi @Warchant, solcjs doesn't support all command line options, you need to install the native solc.
❯ brew linkapps solidity
Error: Unknown command: linkapps
https://solidity.readthedocs.io/en/develop/installing-solidity.html

export SOLC_PATH=/usr/local/bin/solc or export SOLC_PATH=

/ OpenZeppelin(Github) / solidity-docgen / https://asciinema.org/a/TBHvc0p2mwEzWQazYa5aJBp53

https://asciinema.org/a/TBHvc0p2mwEzWQazYa5aJBp53

/ framework

https://openzeppelin.slack.com/archives/C4GP7FGM7/p1537245807000100

/ framework / (Ethereum) / Truffle

https://github.com/trufflesuite/truffle

/ framework / (Javascript) / Mocha

https://mochajs.org

/ framework / battle-tested framework / OpenZeppelin

https://openzeppelin.org

*/ framework / Framework*와 *Library*의 차이 *(*웹 보기*)*

http://mangkyu.tistory.com/4
http://you9010.tistory.com/147
누가 누구를 호출하느냐의 차이 (who calls who)
프레임워크에서는 프레임워크 코드가 우리 코드를 호출하고, 라이브러리에서는 우리 코드가 라이브러리를 호출한다.

/ library / Chai

http://www.chaijs.com

/ library / OpenZeppelin

https://github.com/OpenZeppelin/openzeppelin-solidity#

*/*참고 */ Test / truffle test*

https://truffleframework.com/docs/truffle/testing/writing-tests-in-javascript#writing-tests-in-javascript

*/*참고 */ Test / Style(in Mocha)*

https://mochajs.org/#interfaces

*/*참고 */ Test / Style(in Mocha) / BDD / BDD*는 위의 문제 해결을 위한 테스트 작성 방식을 다음과 같이 *(*웹 보기*)*

BDD는 위의 문제 해결을 위한 테스트 작성 방식을 다음과 같이 제시한다.
“구현(Implementation)을 테스트하는 것이 아니라, 동작(Behavior)을 테스트하는 것이다.”

*/*참고 */ debug / Javascript(Using Intellij Debugger)*

https://github.com/jesoos/truffle-intellij-debug


OpenZeppelin