

Be as proud of Sogang as Sogang is proud of you

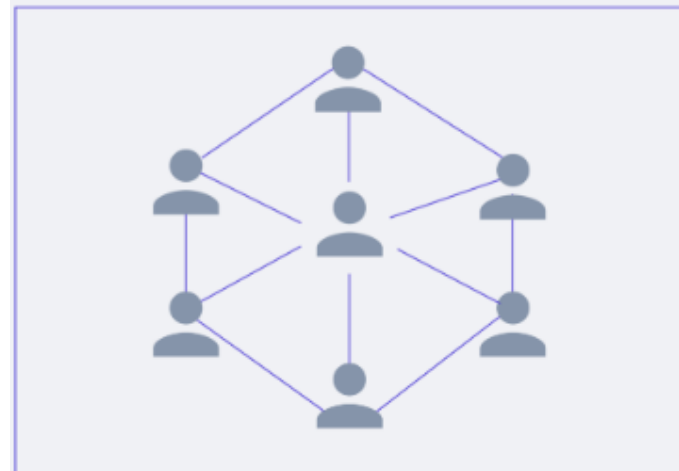
블록체인 이더리움



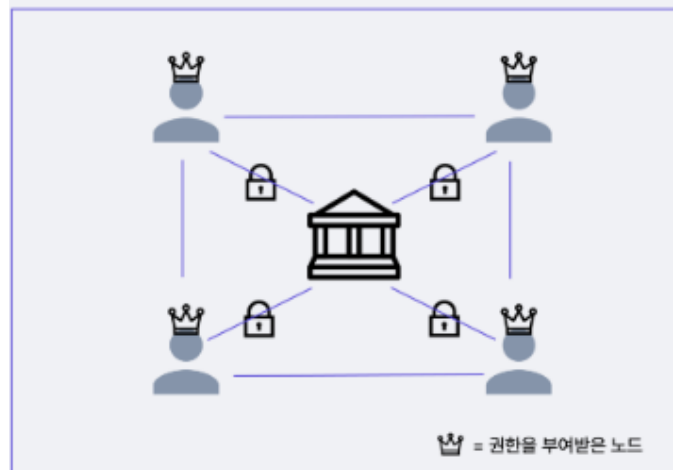
서강대학교
SOGANG UNIVERSITY

- 공개 블록체인 (Public Blockchain)
 - Permissionless
 - 개방형 블록체인
 - 누구나 참여하고 정보를 검토할 수 있음
 - 참여와 탈퇴를 개인이 자유롭게 결정
 - 네트워크 기여에 보상으로 주어지는 암호화폐 발행
 - 비트코인 (Bitcoin)과 이더리움 (Ethereum)

퍼블릭 블록체인



프라이빗 블록체인

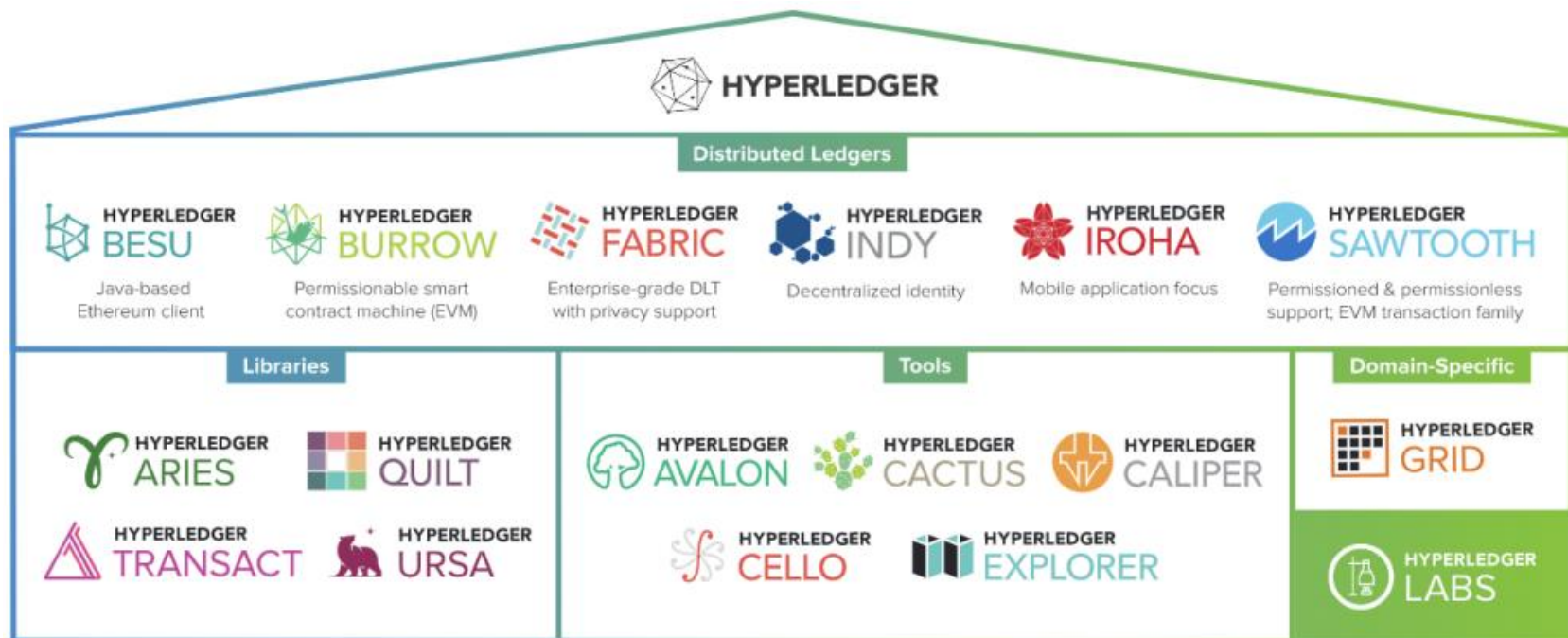


- 비공개 블록체인 (Private Blockchain)
 - Permissioned
 - 폐쇄형 블록체인, 운영자의 승인 필요
 - 일반적으로 특정 그룹이나 조직 내에서 사용, 외부에서 접근할 수 없는 네트워크로 구성
 - 참여자는 권한이 필요한 역할을 할 수 있으며, 모든 트랜잭션 정보에 대한 접근을 허용할지 여부를 제어할 수 있음
 - 민감한 데이터를 다루는 조직에서 특히 유용
 - 참여에 제한이 있고 목적에 따라 내부적으로 운영하므로 네트워크 유지를 위한 보상이 필요 없음
 - 암호화폐 발행은 필수 요건이 아님
 - Hyperledger Fabric

■ Hyperledger Fabric

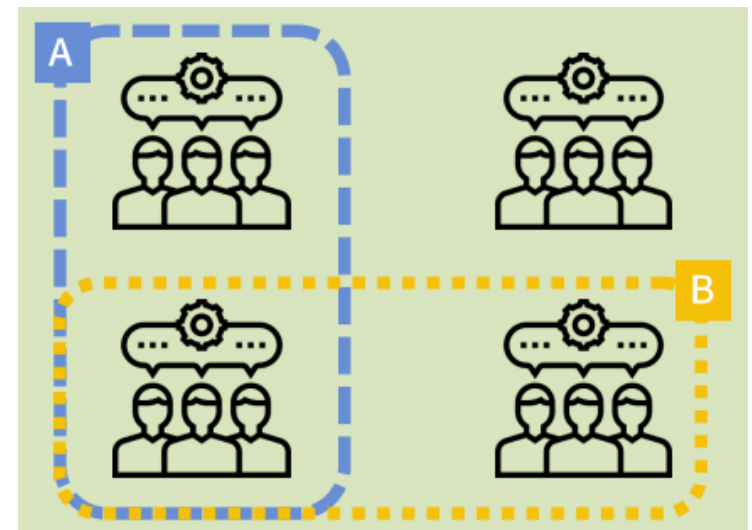
■ Hyperledger (<https://www.hyperledger.org/>)

- 리눅스 재단에서 주관 하에 2015년 12월에 시작된 블록체인 오픈소스 프로젝트
- 기업 및 조직에서 사용하기 위한 오픈 소스 분산 원장 기술을 개발하고 촉진하는 데 중점
- 다양한 블록체인 기술 및 프레임워크를 제공



■ Hyperledger Fabric

- Hyperledger 프로젝트 중 가장 널리 사용되는 것 중 하나로, 기업용 블록체인 솔루션을 위한 분산 원장 프레임워크
- Private Blockchain, Enterprise Blockchain, Permissioned Blockchain
- 합의 알고리즘 및 회원 서비스와 같은 구성 요소를 플러그 앤 플레이 방식으로 지원
- 참여자들을 관리하기 위해 멤버십 관리를 담당하는 모듈(Membership Service Provider) 지원
- 채널이라는 개념을 도입함으로써 블록체인 참여자들간의 프라이버시를 강화, 전체 시스템을 다수의 채널로 구분하여 multi-blockchain으로 운영 가능
- 체인코드(Chaincode)
 - 하이퍼레저 패브릭의 스마트 컨트랙트 프로그램



- IPO (Initial Public Offering)와 ICO (Initial Coin Offering)
 - 기업이 자금을 모으는 데 사용되는 두 가지 다른 금융 모델
- IPO (Initial Public Offering)
 - 기업이 비공개 회사에서 공개 회사로 전환하고 자신의 주식을 일반 투자자에게 처음으로 판매하는 과정
 - 주요 과정
 - 회사는 주식 시장에 상장하기 위해 증권 거래소와 협력
 - 회사는 IPO 준비를 위해 자세한 재무 정보 및 비즈니스 계획을 제출하고, 주식 가격과 수량을 결정
 - IPO 과정에서 회사 주식은 일반 투자자에게 판매되며, 회사는 자금을 조달
 - 규제와 보안: IPO는 엄격한 규제와 보안 요구 사항을 따라야 함. 정부 규제 기관과 증권 거래소에서 감독하며, 투자자 보호와 투명성을 강조
 - 투자자: IPO는 주로 전문 투자자, 기관 투자자 및 일반 투자자를 대상으로 하며, 공개된 회사의 주식을 보유하게 됨

■ ICO (Initial Coin Offering):

- 블록체인 및 암호화폐 기업이 자신의 프로젝트 또는 플랫폼을 위한 암호화폐 토큰을 발행하고 투자자에게 판매하는 과정
- 주요 과정
 - 기업은 프로젝트의 목적과 역량 등의 정보를 포함한 백서 발행
 - 기업은 자체 토큰을 발행하고 이를 투자자에게 판매. 이 토큰은 일반적으로 미래의 서비스 또는 생태계에서 사용될 것으로 기대됨.
 - ICO를 통해 모금된 자금은 프로젝트 개발, 마케팅, 운영 등에 사용
- 규제와 보안: ICO는 일반적으로 적은 규제가 적용되며, 법적 지위와 보안 문제에 대한 불확실성이 존재. 일부 국가 및 지역에서는 ICO를 규제하거나 금지
- 투자자: ICO는 기존의 투자자와 블록체인 및 암호화폐 개인 투자자에게 주로 제공됨. 투자자는 토큰을 보유하고 해당 플랫폼 또는 생태계에서 활용

■ 토큰 경제(Token Economy)

- 행동심리학에 시초를 둔 용어로 특정 행동을 이끌어내기 위해 토큰을 인센티브로 제공하고 해당 토큰은 유/무형의 가치와 교환됨으로써 그 행동을 강화하는 방법
- 블록체인 토큰 이코노미: 블록체인과 암호화폐 기술을 기반으로 한 새로운 경제 시스템
- 토큰은 참여를 장려하거나 보상하는 데 사용
- 특정 플랫폼, 프로젝트 또는 생태계에서 사용되는 토큰을 중심으로 구축되며, 이 토큰은 다양한 경제적 활동과 가치 교환을 위한 매개체로 사용
- 기존의 중앙 집중식 경제 모델과는 다르게 분산 원장과 스마트 계약을 활용하여 신뢰와 투명성을 증진하고 경제 활동을 자동화하며 프로토콜 수준에서 규칙을 정함
- 스마트 계약: 토큰 경제는 스마트 계약을 통해 프로그래밍 가능한 규칙을 정의하고 이행
- 보상 및 인센티브: 토큰 경제는 커뮤니티 구성원에 대한 보상과 인센티브를 제공. 이를 통해 사용자들은 플랫폼에 기여하고 자신의 활동을 증가시키며, 생태계의 성장을 지원

- Steemit
 - 블록체인 기반 SNS 플랫폼
 - 콘텐츠 창작자뿐만 아니라 콘텐츠 이용자도 콘텐츠에 대한 추천(voting)에 참여하여 보상을 받음



- 이더리움 개요
 - 이더
 - 계정(Accounts)
 - 가스(Gas)
 - EVM
- 스마트 컨트랙트
 - 솔리디티 언어
- Remix

■ 이더리움

- 비트코인 이후로 개발된 여러 블록체인 기반의 플랫폼들 중에 하나
- <https://ethereum.org/>



Vitalik Buterin

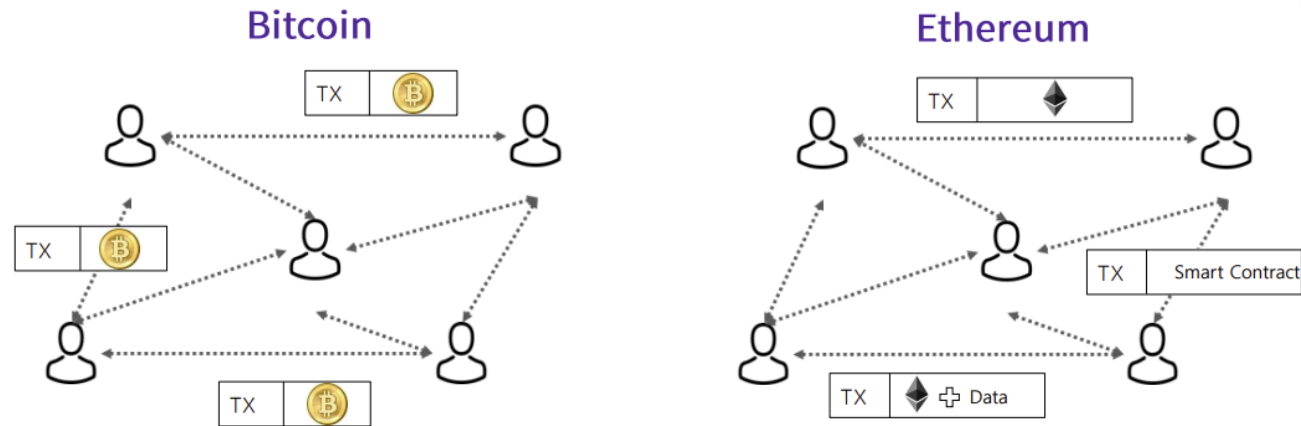
- 비트코인: 최초의 블록체인 기반 암호화폐 시스템
 - 1세대 블록체인: 암호화폐를 위한 플랫폼
- 이더리움
 - 2세대 블록체인: 스마트컨트랙트를 이용하여 다양한 분야의 분산 애플리케이션이 가능한 플랫폼
 - 비트코인의 블록체인 기술을 기반으로 설계
 - 2015년 비탈릭 부테린(Vitalik Buterin)이 개발
 - 정확히 프로그래밍한대로 동작하는 스마트컨트랙트를 실행시키는 분산 컴퓨팅 플랫폼
 - 이더(ETH): 이더리움 플랫폼에 의해 생성된 암호화폐
 - 수많은 ICO (Initial Coin Offering)의 플랫폼
 - 잔고를 갖는 계정(Accounts) 기반으로 설계

- 이더(ETH)
 - 이더리움 네트워크의 암호화폐
 - 송금 가능, 채굴에 대한 보상과 거래 수수료로 사용
 - 이더의 단위

	단위		Multiplier
1 ETH	Ether	1,000,000,000,000,000,000	10^{18}
0.001 ETH	finney	1,000,000,000,000,000	10^{15}
0.000 001 ETH	Szabo	1,000,000,000,000	10^{12}
0.000 000 001 ETH	Gwei	1,000,000,000	10^9
0.000 000 000 000 000 001 ETH	Wei	1	10^0

이더리움과 비트코인 차이

스마트컨트랙트 유무



잔고를 갖는 계정(Accounts) 기반으로 설계



■ 이더리움과 비트코인 차이

■ 튜링완전성(Turing-Completeness)

- 어떤 프로그래밍 언어나 추상 머신이 튜링머신과 동일한 계산 능력으로 문제를 풀 수 있다는 의미
- 튜링완전언어 + 무한한 저장공간 = 모든 계산 가능한 문제를 계산해내는 기계 = 튜링머신

■ 이더리움 참고

- 이더리움 백서(white paper): <https://ethereum.org/en/whitepaper/>
- 이더리움 황서(yellow paper): <https://ethereum.github.io/yellowpaper/paper.pdf>

■ 계정(Accounts)

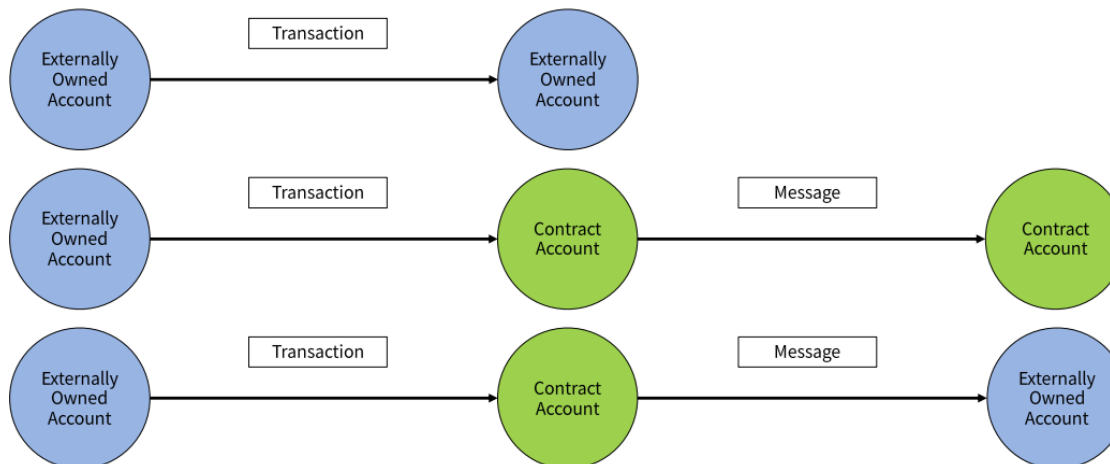
- 이더리움의 기본 단위는 계정 , 계정들을 모아 상태(State)를 구성
- 잔액 보유
- EOA(Externally Owned Account: 외부사용자,외부 소유 계정)
 - 사람이 직접 개인키(private key)로 관리
 - 거래를 생성하고 서명함으로써 메시지를 보낼 수 있다
- CA(Contract Account: 스마트 컨트랙트 계정)
 - 스마트 컨트랙트와 연결된 계정
 - 메시지를 받을 때마다 자신의 코드를 실행하여 내부 저장 공간의 데이터를 읽거나 쓰고, 다른 메시지를 보내고, 새로운 계약을 생성할 수 있다.
 - 컨트랙트 코드, 저장소

■ 거래(Transaction)

■ 거래(Transaction)

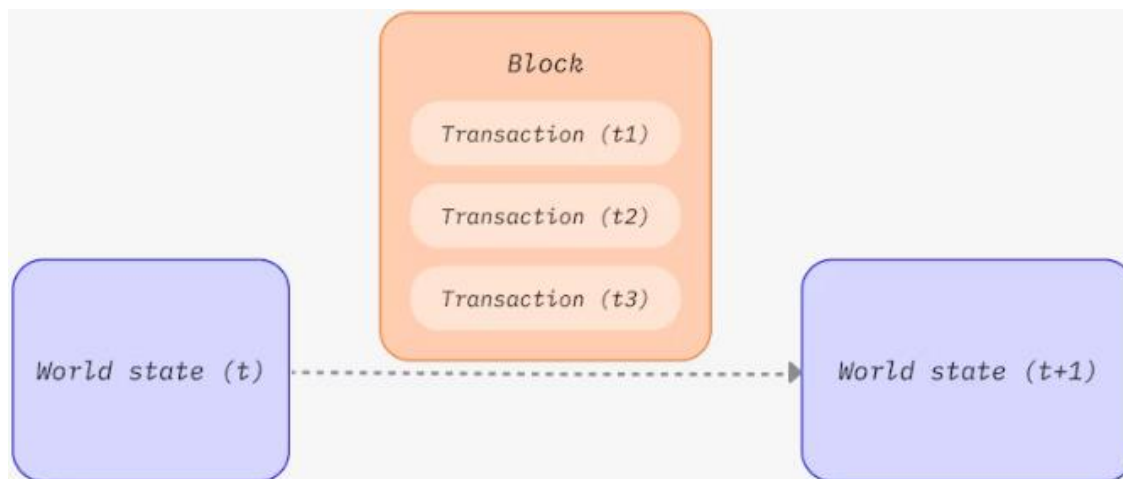
- 다른 계정으로 이더를 전송하거나, 스마트컨트랙트를 생성, 또는 컨트랙트의 함수를 호출
- 이더리움 네트워크의 상태를 업데이트
- EOA로부터 보내진 메시지를 저장하는 서명된 데이터패키지를 의미
 - 메시지 수신자, 발신자를 나타내는 서명, 발신자로부터 수신자에 보내지는 이더의 총액 등

■ 메시지(Message): 내부 트랜잭션, 계약은 다른 계약에 "메시지"를 보낼 수 있다



■ 블록

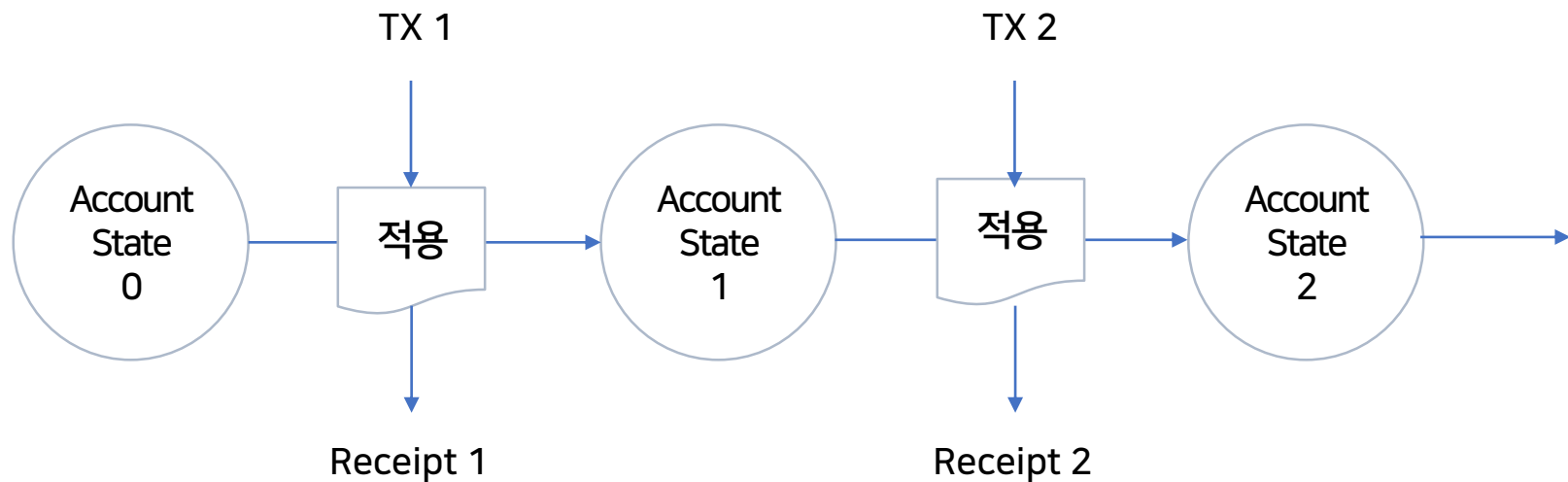
- 블록은 **이전 블록의 해시값**을 가지고 있는 트랜잭션의 묶음
- 블록 데이터에서 암호학적으로 파생되는 해시로 블록들을 연결하여 체인으로 만든다
- 어느 하나의 블록을 변경하면 이후의 모든 블록이 무효화되므로 부정(위변조) 방지
- 블록들은 일정한 간격으로 생성되어 체인에 연결: 이더리움은 약 17초, 이더리움 2.0은 12초
- 블록단위 동기화: 수십 또는 수백개의 트랜잭션이 한번에 합의, 동기화
- <https://ethereum.org/en/developers/docs/blocks/>



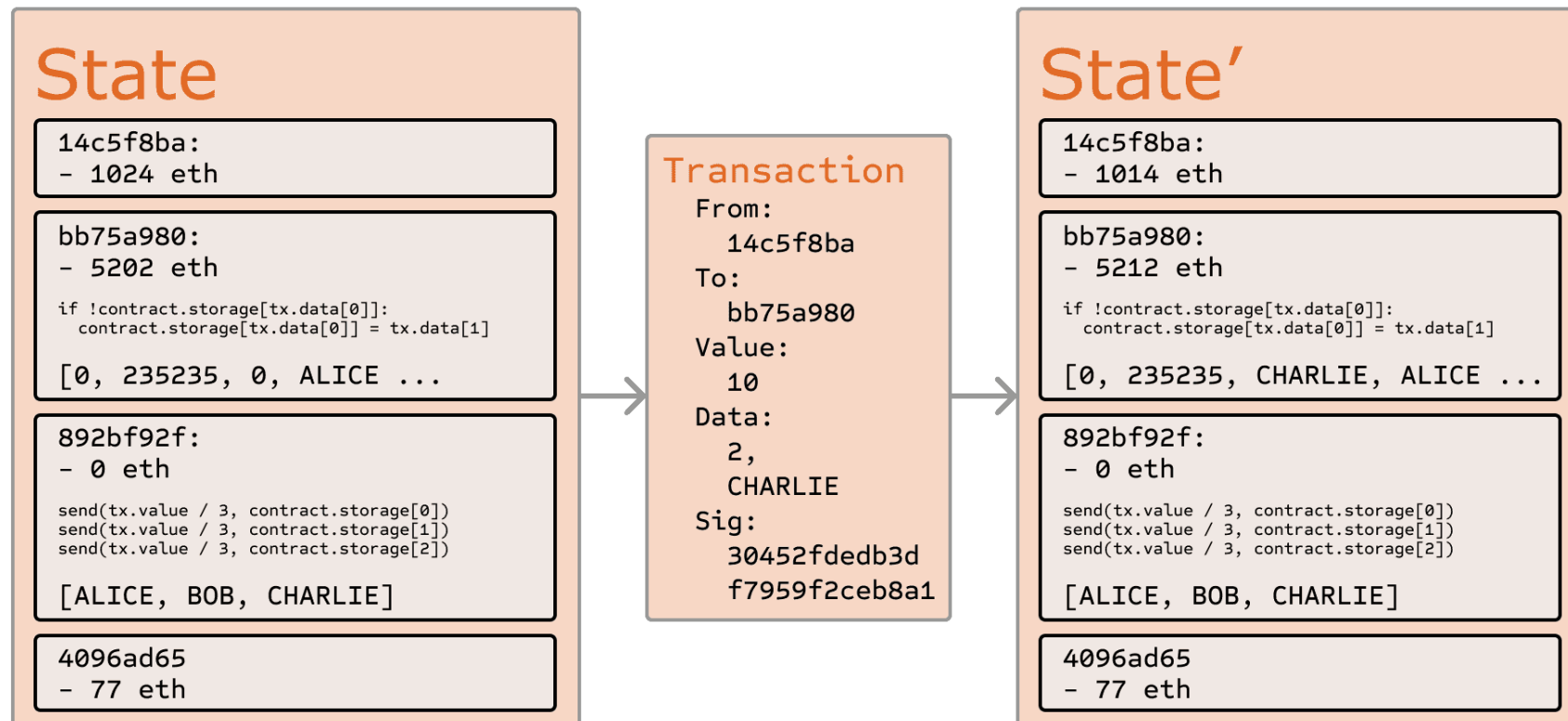
■ 이더리움 특징: State Transition Model

■ 이더리움 블록 정보

- Transactions
- Tx 실행결과 (Receipt)
- Account state (EOA와 CA)

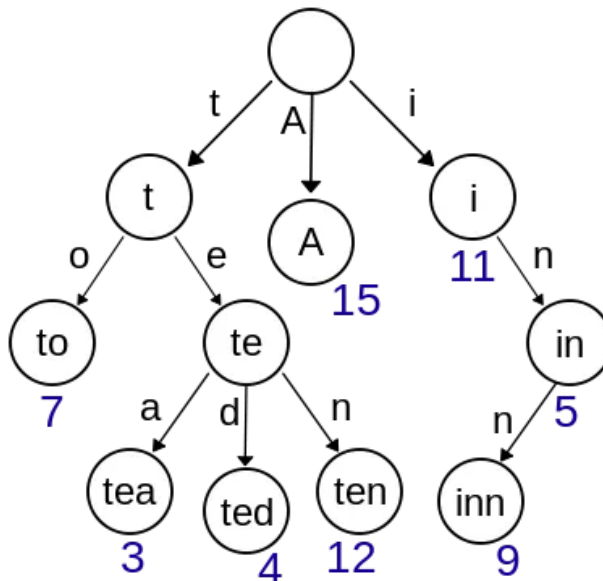


■ 트랜잭션 실행으로 상태 변경



■ 상태 저장: Merkle Patricia Trie

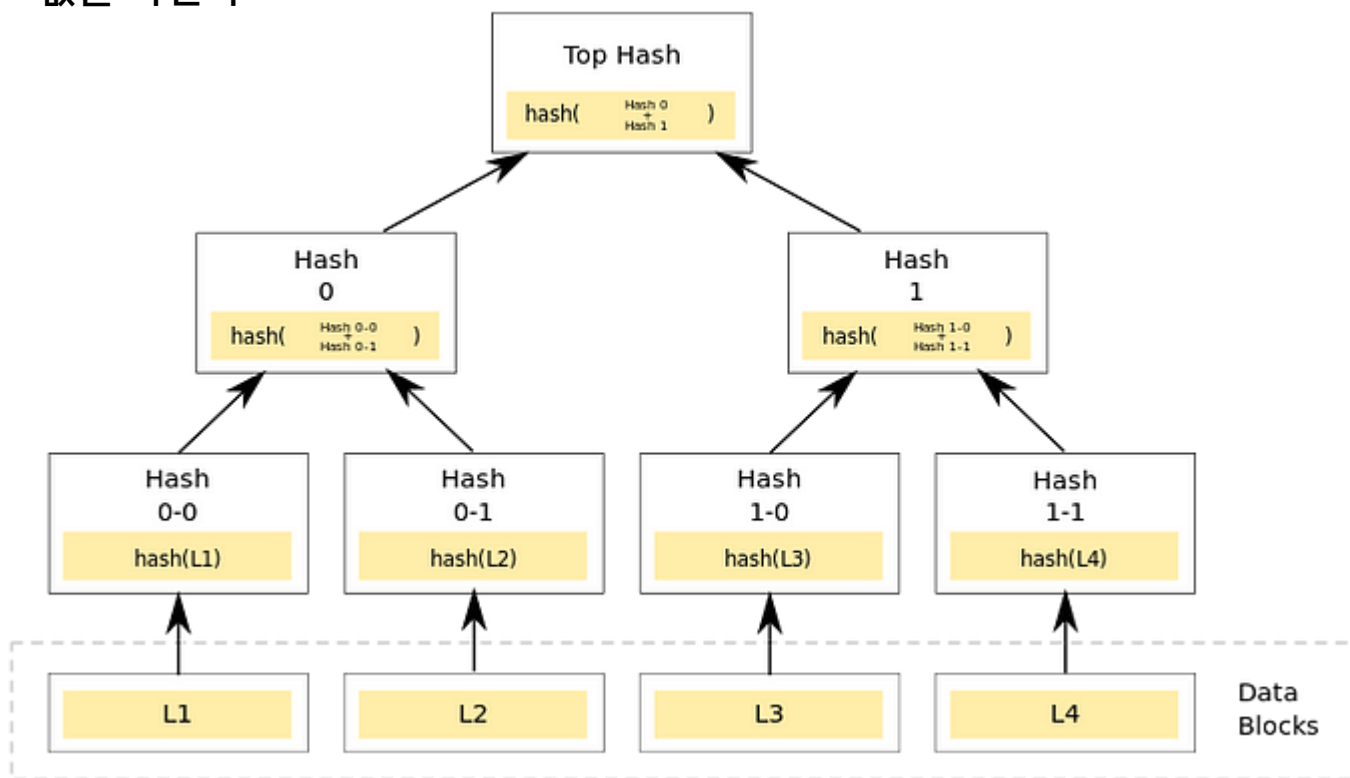
- 상태는 key와 value로 이루어짐
- 상태 정보를 효율적으로 저장, 수정, 삭제, 검색
- Patricia Tree(Trie)
 - path에 key를 집어넣어 공통된 prefix를 가지는 노드들은 같은 path를 가진다. 공통된 prefix를 찾는데 가장 빠르고, 적은 메모리로 구현할 수 있으며, 구현도 간단



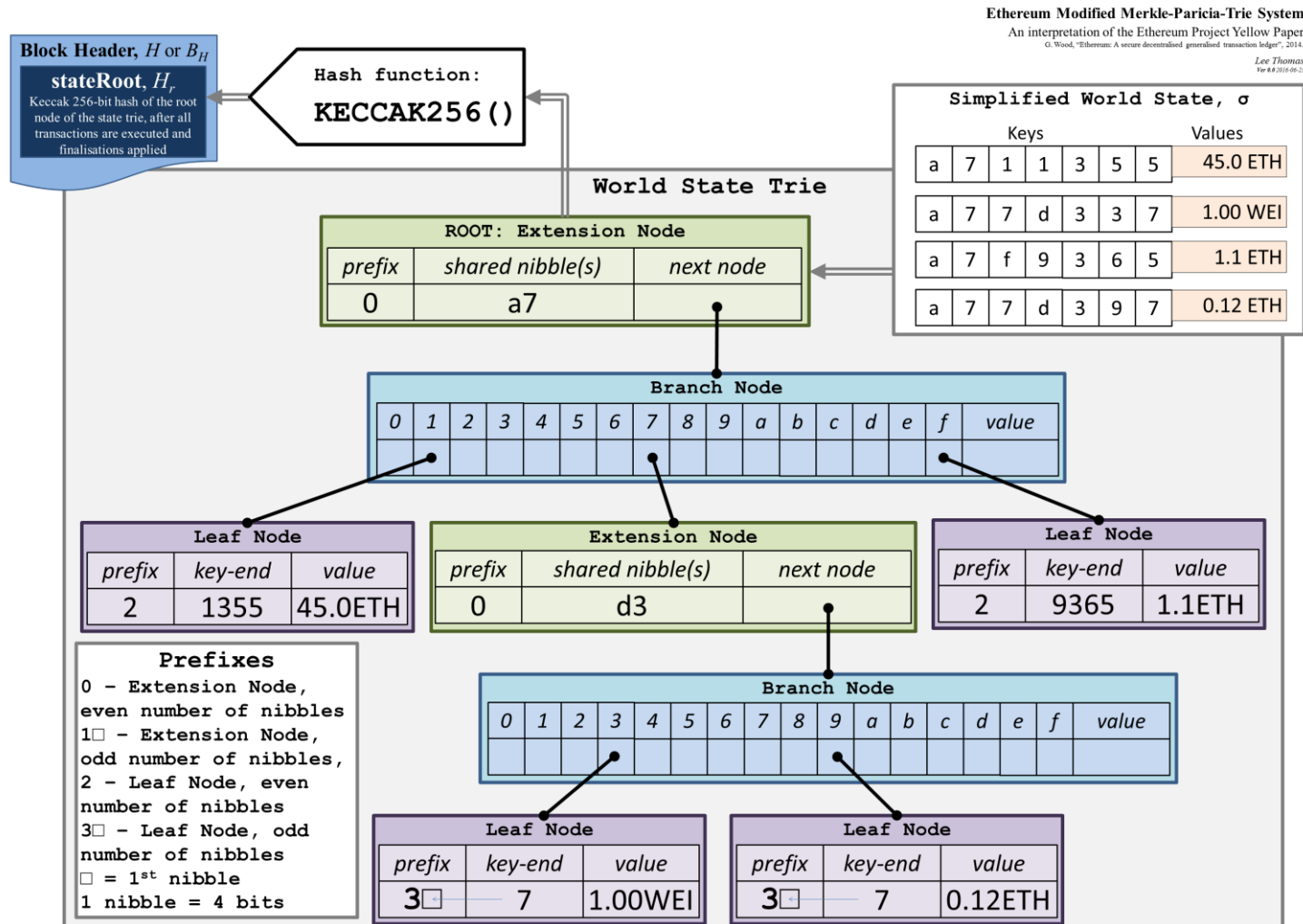
■ 상태 저장: Merkle Patricia Tree

• Merkle Tree

- Leaf의 부모는 leaf의 hash를 가지고, 그 부모는 자식들의 hash의 합을 다시 hash 한 값을 가진다



■ 상태 저장: Merkle Patricia Tree



■ PoS(Proof of Stake)

■ 이더리움2.0 이전: PoW(Proof of Work)

- 반복적인 과정을 통해서 어떤 난수 값을 찾아야 한다, 굉장한 에너지와 연산력이 필요
- 환경적인 측면에서 많은 논란이 대두
- PoS(지분 증명) 방식으로 전환

■ 32이더를 예치한 노드들이 검증자로서의 권한을 가지게 된다

■ 매 슬롯(12초 간격)마다 무작위로 선택된 검증자가 블록 제안자로 선정

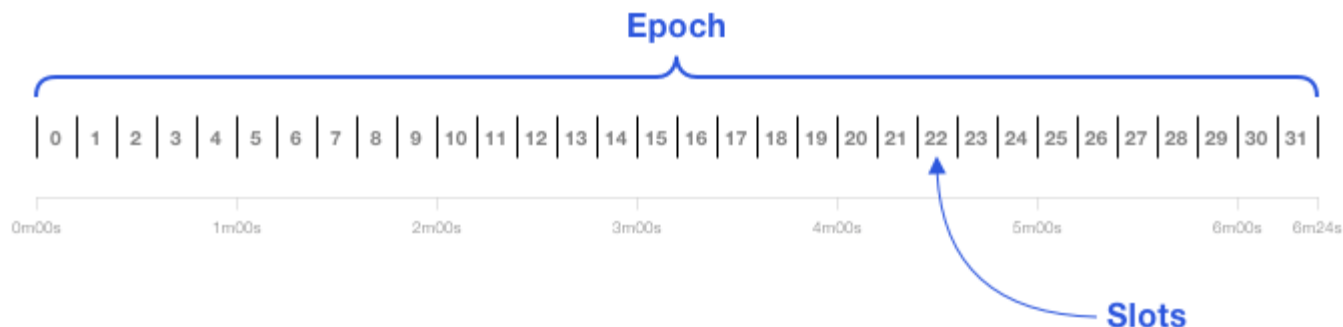
- 거래를 묶어서 실행하고 새로운 '상태'를 결정
- 이 정보를 블록으로 묶어 다른 검증자에게 전달

■ 새로운 블록에 대해 전달받은 다른 검증자들은 블록이 유효한지 판단. 유효하다면 자신의 자체 데이터베이스에 추가

- 만일 동일한 슬롯에서 두개의 충돌하는 블록을 받았을 경우 자동적으로 가장 많은 스테이킹 이더가 지원하는 블록이 선택되어진다

■ PoS

- 슬롯(slot): 블록제안, 검증자들의 검증이 진행되는 시간단위, 블록 생성 시간의 일관성
- 에포크(epoch): 32개의 슬롯으로 이루어진 주기



- block proposer: 각 슬롯(12초 간격)마다 랜덤하게 선택된 유효성 검사자, 새로운 블록을 생성하고 네트워크의 다른 노드로 보내는 역할
- Validator들 위원회: 매 슬롯에서는 일부 유효성 검사자로 구성된 위원회가 무작위로 선택, 투표 (Attestation) 를 함으로써 전달받은 블록 지지

■ Ether & Gas

■ Ether(ETH)

- 이더리움 네트워크에서 거래를 지불하는 데 사용되는 암호화 통화의 이름
- Ether는 Gas를 구매하는 데 사용(이더리움 내 계산 비용을 지불하는 데 사용)
- 이더리움의 기본 통화

■ Gas

- 이더리움을 구동하는 연료
- 누군가가 이더리움에서 무한 루프를 실행하고 메모리를 완전히 과부하시키는 것을 방지
- 네트워크에서 계산 리소스에 대한 비용을 지불하기 위한 측정(metric) 단위
- Gas Limit: 트랜잭션에서 사용할 수 있는 최대 가스량, 소비할 의사가 있는 최대량

■ 가스(gas)

- Gas는 이더리움 네트워크에서 특정 작업을 실행하는데 필요한 노력의 양을 측정하는 단위
- Gas는 각 operation 별로 정해짐(예, 송금거래는 21,000gas 필요)
- 이더리움 트랜잭션을 실행하려면 컴퓨팅 리소스가 필요하므로, 이에 따른 수수료를 내야 한다. 즉 가스는 이더리움 네트워크에서 어떤 작업을 수행하기 위해서 필요한 수수료
- 보안을 위해 필요: 악의적인 사용자의 스팸 방지, 무한 루프 방지
- 이더리움 생태계 유지: 이더리움 네트워크에서는 가스비(Gas Fee)라는 수수료를 징수해서 이를 네트워크 기여자에게 나눠주는 방식
- 가스비(gas fee) = 작업을 수행하는 데 사용되는 가스의 양 * 단위 가스당 비용(Gas price)
 - 네트워크 수수료(Network fee)
- 가스비는 이더리움의 기본 통화인 이더(ETH)로 지불
- Gas price는 일반적으로 ETH 단위인 gwei(0.000000001 ETH)로 표시

■ 가스(gas)

- 블록체인에 정보를 저장하고 계약을 실행하려면 비용 발생
- 스마트 컨트랙트의 길이와 정의된 명령어 코드에 따라서 소비되는 가스 양이 책정됨

Name	Value	Description
G_{zero}	0	Nothing paid for operations of the set W_{zero} .
$G_{jumpdest}$	1	Amount of gas to pay for a JUMPDEST operation.
G_{base}	2	Amount of gas to pay for operations of the set W_{base} .
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low} .
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid} .
G_{high}	$G_{callstipend}$	2300 A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{warmaccess}$	$G_{newaccount}$	25000 Paid for a CALL or SELFDESTRUCT operation which creates an account.
$G_{accesslistaddress}$	G_{exp}	10 Partial payment for an EXP operation.
$G_{accessliststorage}$	$G_{expbyte}$	50 Partial payment when multiplied by the number of bytes in the exponent for the EXP operation.
$G_{coldaccountaccess}$	G_{memory}	3 Paid for every additional word when expanding memory.
$G_{coldload}$	$G_{txcreate}$	32000 Paid by all contract-creating transactions after the <i>Homestead</i> transition.
G_{sset}	$G_{txdatazero}$	4 Paid for every zero byte of data or code for a transaction.
G_{sreset}	$G_{txdataanonzero}$	16 Paid for every non-zero byte of data or code for a transaction.
R_{sclear}	$G_{transaction}$	21000 Paid for every transaction.
	G_{log}	375 Partial payment for a LOG operation.
$R_{selfdestruct}$	$G_{logdata}$	8 Paid for each byte in a LOG operation's data.
$G_{selfdestruct}$	$G_{logtopic}$	375 Paid for each topic of a LOG operation.
G_{create}	$G_{keccak256}$	30 Paid for each KECCAK256 operation.
$G_{codedeposit}$	$G_{keccak256word}$	6 Paid for each word (rounded up) for input data to a KECCAK256 operation.
$G_{callvalue}$	G_{copy}	3 Partial payment for *COPY operations, multiplied by words copied, rounded up.
	$G_{blockhash}$	20 Payment for each BLOCKHASH operation.

■ 가스(gas)

■ 기본 수수료(base fee)

- 내부에서 계산되어 설정, 이전 블록 크기에 따라 변경
- 거래가 유효한 것으로 간주되려면 지불해야 하는 최소한의 금액
- 블록 생성후 검증자에게 주지 않고 소각

■ 우선 수수료(priority fee, Tip)

- 사용자가 설정한 값, 팁 개념과 유사
- 블록에 추가될 확률을 높여줌
- 검증자에게 보상으로 주어짐

■ 가스(gas)

■ 최대 수수료(Max fee): optional 매개변수

- 사용자가 트랜잭션 실행에 대해 지불할 의사가 있는 최대 한도를 지정
- 기본 수수료와 팁의 합계를 초과해야 함
- 추후 실제 사용하고 남은 차액은 반환

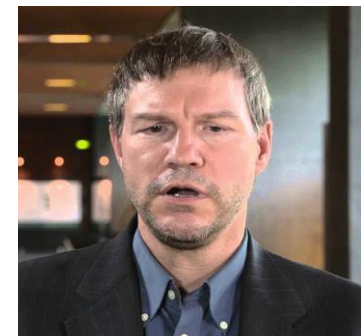
■ $\text{gas fee} = \text{units of gas used} * (\text{base fee} + \text{priority fee})$

■ 예) 1이더를 전송하는 송금 거래

- 21000(송금거래에 필요한 가스 단위), 10gwei(기본수수료), 2gwei(tip)
- $21,000 * (10 + 2) = 252,000 \text{ gwei} (0.000252 \text{ ETH})$
 - 송금 계좌에서 1.000252 ETH가 차감, 수신 계좌에 1.0000 ETH가 적립
 - 검증인은 0.000042 ETH의 팁 수령

■ 스마트 컨트랙트

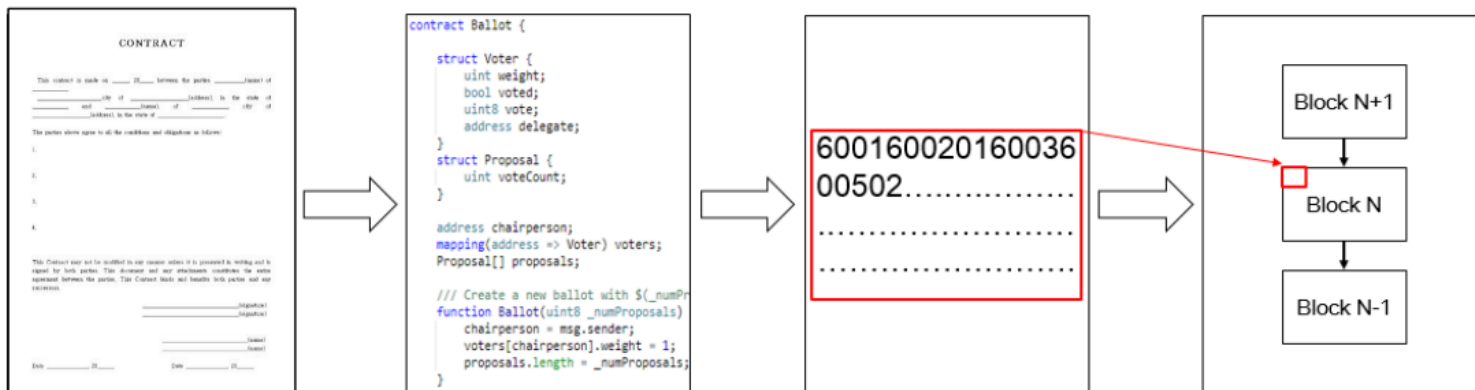
- 닉 재보(Nick Szabo)가 처음 고안한 개념
- 계약 조건을 실행하는 컴퓨터화된 거래 프로토콜
- 컴퓨터 코드로 구현되어 자동으로 실행되고, 사전에 정의된 조건이 충족되면 특정 작업이나 거래가 자동으로 이루어지는 시스템
 - 자동판매기와 비슷: 자동판매기에 미리 정해진 액수 이상의 돈을 투입하면, 자동으로 원하는 상품을 구매할 수 있듯이, 스마트 계약을 통해 일정한 조건이 충족되면 자동으로 계약이 실행되도록 한다
- 이더리움(Ethereum)이 개발됨으로써 실제로 구현
 - 스마트 컨트랙트 플랫폼(smart contract platform)



Nick Szabo

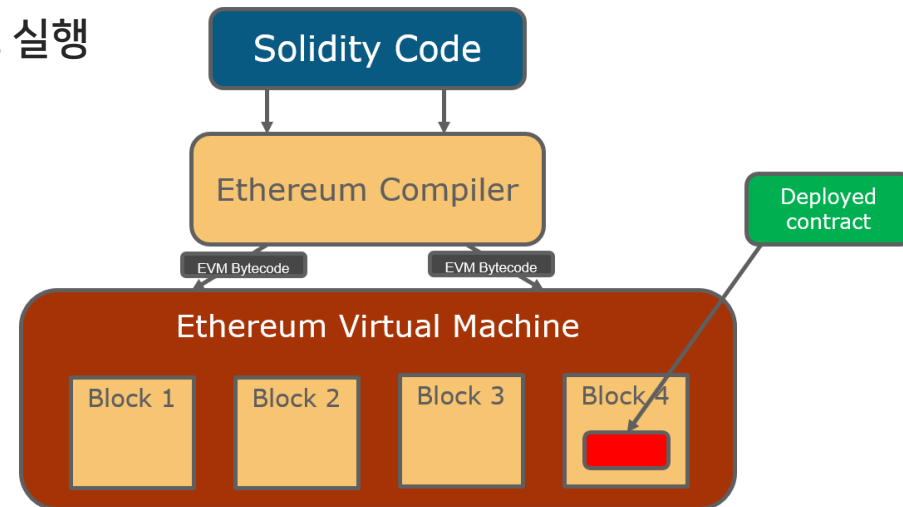
■ 이더리움 스마트 컨트랙트

- 이더리움상(EVM)에서 실행할 수 있는 프로그램, 디지털화된 약속
- '계약서' 내용의 로직을 프로그래밍을 통해서 프로그램화(상태와 함수를 갖는 프로그램)
- 계약서 내용이 작성된 소스코드를 컴퓨터가 이해할 수 있는 코드로 변환하여 블록체인에 저장
- 이더리움 가상머신(EVM; Ethereum Virtual Machine)이라는 독립된 실행 환경에서 실행
- 튜링 완전한(turing-complete) 언어를 사용하여 더 복잡한 처리 구현 가능
 - 조건문(if), 반복문(loop) 등의 로직 포함
- 수수료인 가스(gas)를 발생시키고 네트워크상에 수수료의 한계를 설정하여 무한루프를 막는다



■ 스마트컨트랙트 동작원리

- 솔리디티(고급언어)로 코드 작성
- 솔리디티 컴파일러(solc)에 의해 기계어(Ethereum bytecode)로 변환
- EVM(Ethereum Virtual Machine)이 바이트코드 실행



■ 컴파일러

- 특정 프로그래밍 언어로 쓰여 있는 문서를 다른 프로그래밍 언어로 옮기는 언어 번역 프로그램
- 컴파일러는 고급 프로그래밍 언어를 실행 프로그램으로 만들기 위해 저급 프로그래밍 언어(예, 어셈블리 언어, object 코드, machine code)로 바꾸는 데 사용

■ 스마트 컨트랙트 특징

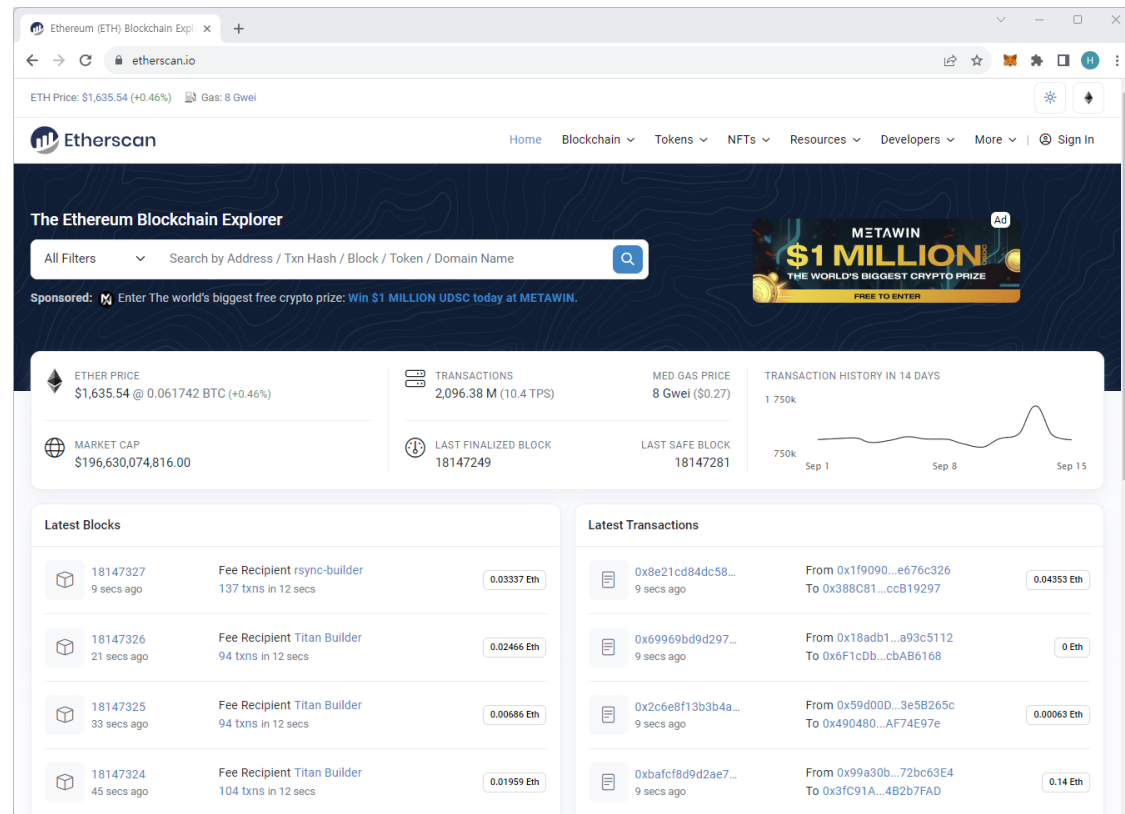
- 불변성: 블록체인에 배포된 스마트 컨트랙트는 수정 불가능
- 투명성: 블록체인에 배포된 스마트 컨트랙트의 내용은 전부다 공개됨
- 자동화: 조건이 충족되면 자동으로 실행됨

■ 스마트 컨트랙트 문제점

- 한번 배포되어 블록으로 생성된 스마트 계약은 수정이 불가능: 업그레이드나 버그 패치, 보안 취약점 수정 등이 어렵다
 - 업그레이드 가능한 스마트 계약(upgradable smart contract): delegatecall 기능을 이용하여 복수의 스마트 계약을 구성하고, 새로 배포한 스마트 계약을 델리게이트콜하는 방식

- EVM(Ethereum Virtual Machine, 이더리움 가상 머신)
 - 이더리움의 스마트 컨트랙트를 위한(동작하는, 실행되는) 런타임환경
 - 이더리움 노드에 존재하는 가상 컴퓨터로 스마트 컨트랙트를 구동시키기 위해 설계된 머신
 - 바이트 코드 기반
 - 가스 측정
 - 가스는 EVM에서 연산 자원의 사용을 측정하는 단위
 - 각 연산 명령어는 특정한 가스 비용을 가지며, 스마트 계약 실행 시 총 가스 사용량이 계산됨

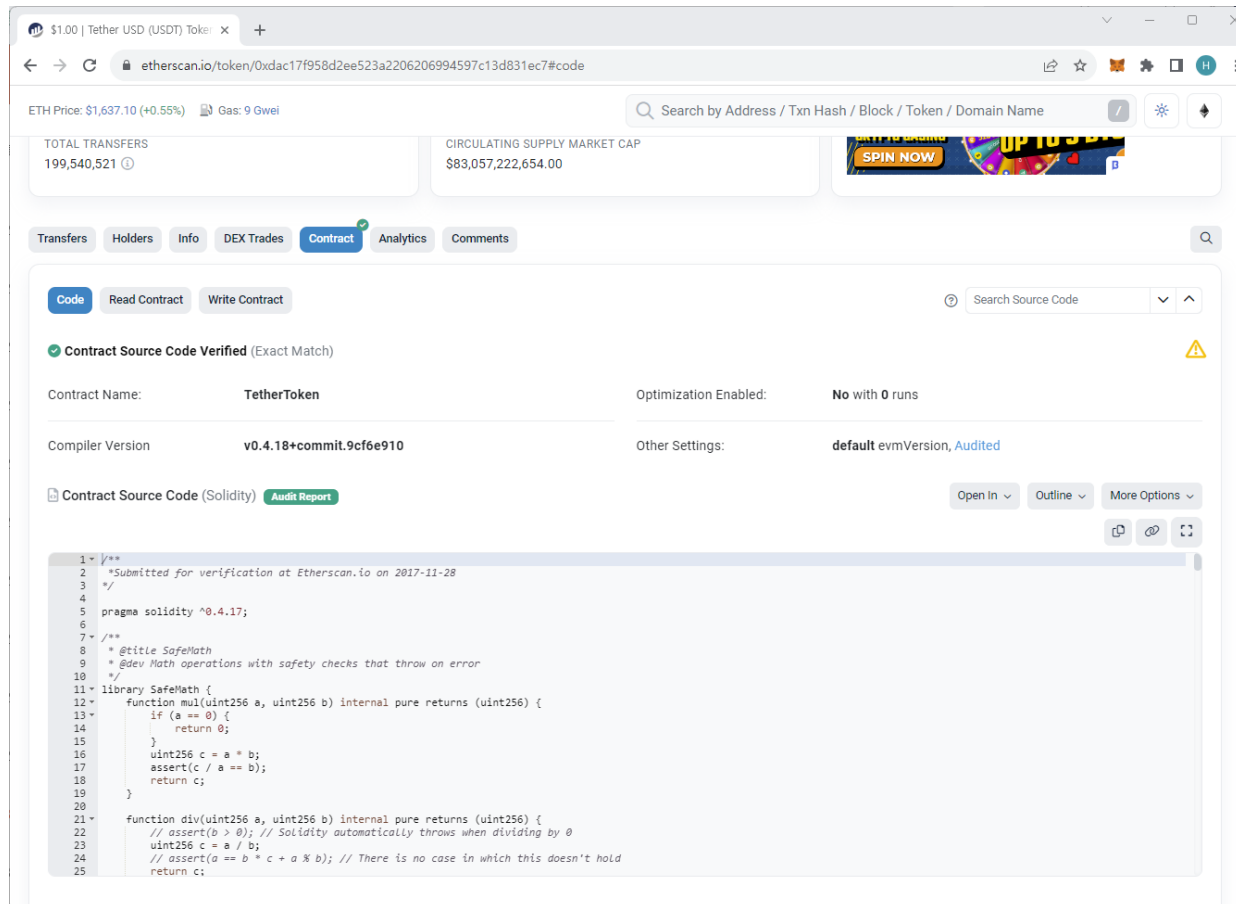
- 이더스캔(<https://etherscan.io/>)
 - 이더리움 블록체인에 관한 정보를 제공하고 검색할 수 있는 온라인 서비스 및 블록체인 탐색기
 - 이더리움 네트워크의 트랜잭션, 주소, 스마트 컨트랙트, 블록 및 기타 관련 데이터를 제공하여 사용자가 이더리움 블록체인을 탐색하고 모니터링할 수 있음



■ 이더스캔

■ 예) TetherToken 스마트 컨트랙트 확인

<https://etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7#code>



The screenshot displays the Etherscan.io interface for the TetherToken smart contract. The browser address bar shows the URL: `etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7#code`. The page header includes the ETH Price (\$1,637.10) and Gas price (9 Gwei). The contract details section shows the contract name as **TetherToken**, the compiler version as **v0.4.18+commit.9cf6e910**, and the optimization status as **No with 0 runs**. The **Contract Source Code (Solidity)** tab is selected, displaying the source code with a green checkmark indicating it is verified. The source code is as follows:

```
1 /**
2  *Submitted for verification at Etherscan.io on 2017-11-28
3  */
4
5  pragma solidity ^0.4.17;
6
7  /**
8   * @title SafeMath
9   * @dev Math operations with safety checks that throw on error
10  */
11  library SafeMath {
12      function mul(uint256 a, uint256 b) internal pure returns (uint256) {
13          if (a == 0) {
14              return 0;
15          }
16          uint256 c = a * b;
17          assert(c / a == b);
18          return c;
19      }
20
21      function div(uint256 a, uint256 b) internal pure returns (uint256) {
22          // assert(b > 0); // Solidity automatically throws when dividing by 0
23          uint256 c = a / b;
24          // assert(a == b * c + a % b); // There is no case in which this doesn't hold
25          return c;
26      }
27  }
```

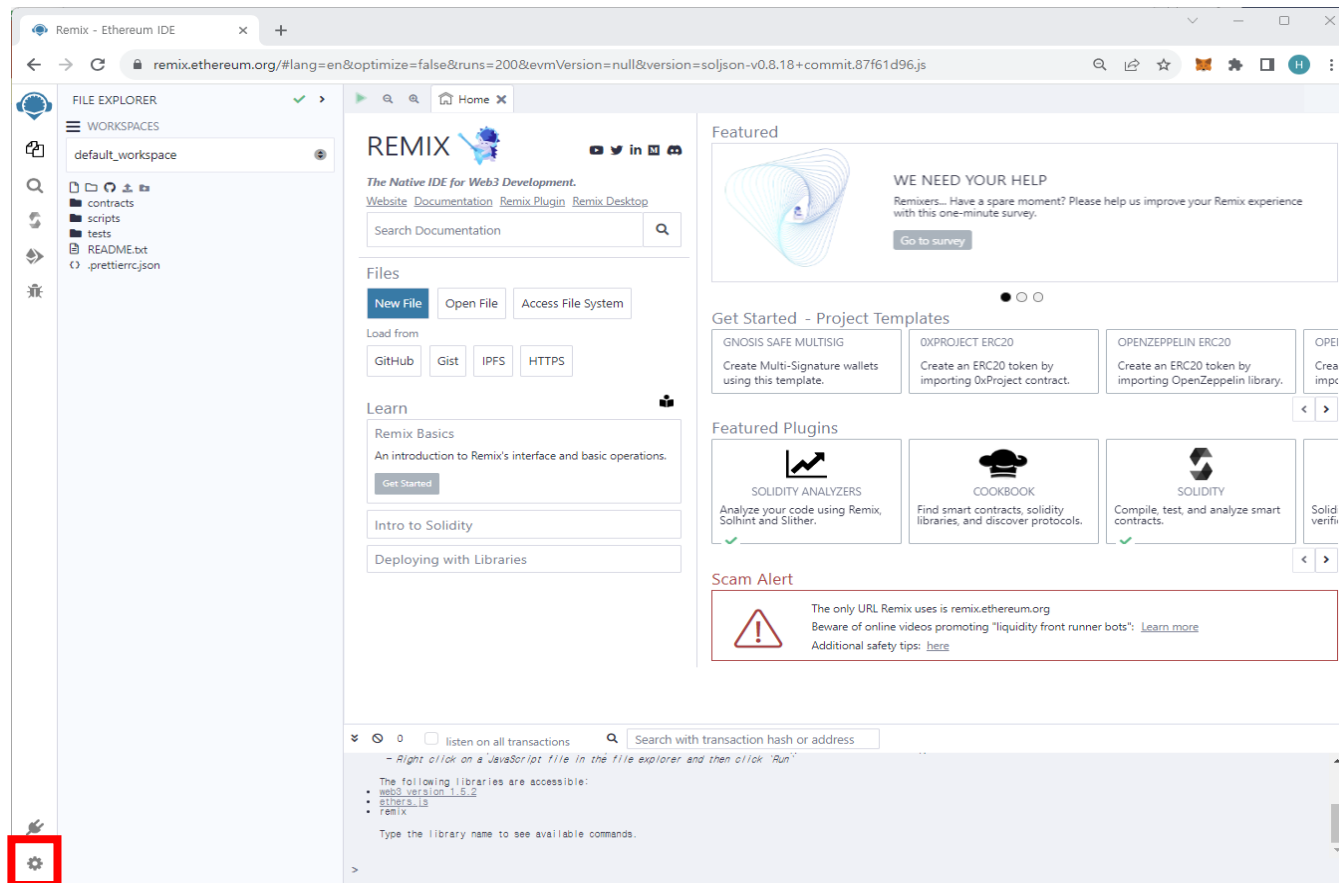
■ 스마트 컨트랙트 개발 언어

■ 솔리디티 언어(Solidity)

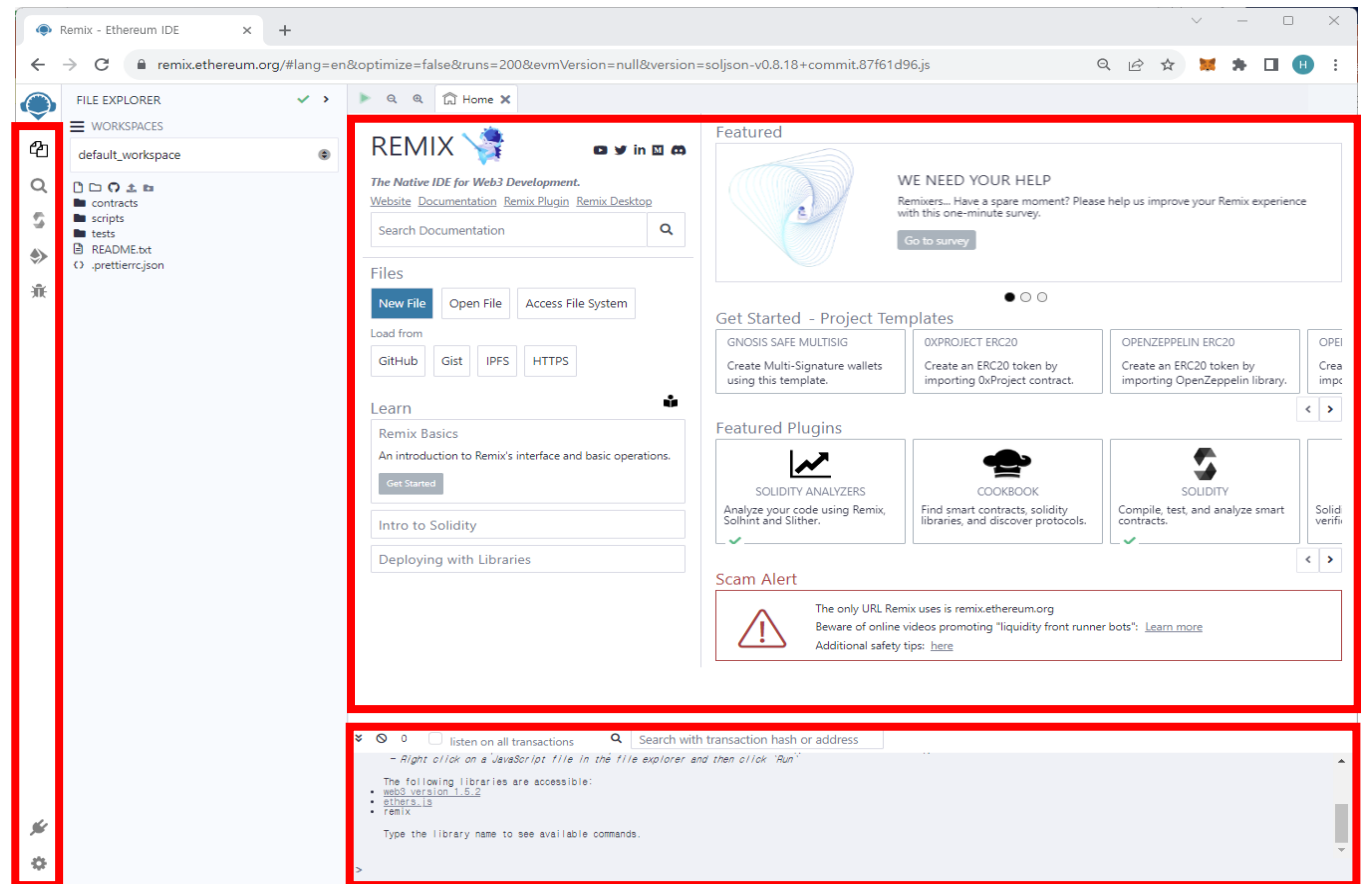
- 스마트컨트랙트 개발을 위한 프로그래밍 언어
- 객체 지향
- C++과 유사한 형태
- 가장 많이 사용되는 언어
- 상속 및 라이브러리 사용 가능
- 확장자: sol

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.8.2 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public {
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){
27         return number;
28     }
29 }
```

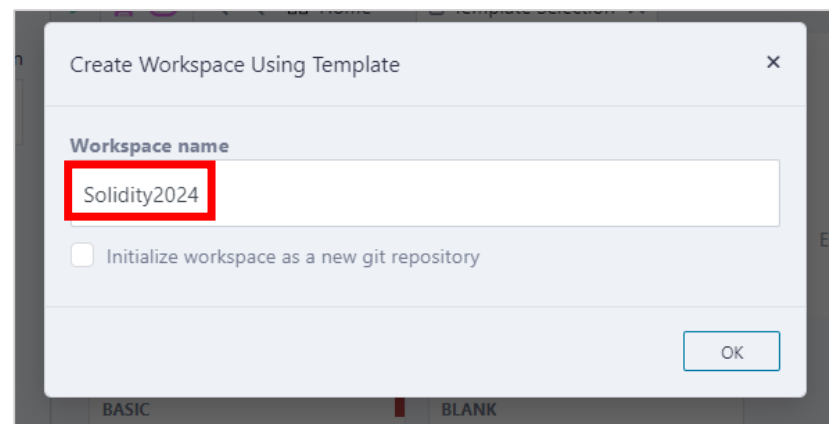
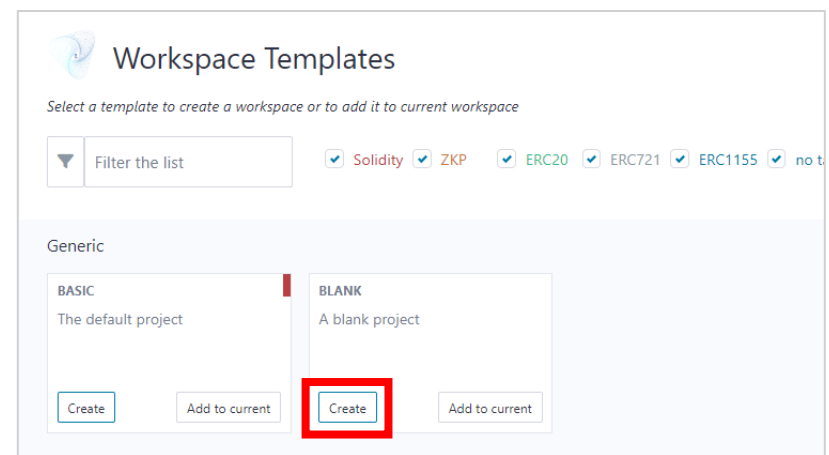
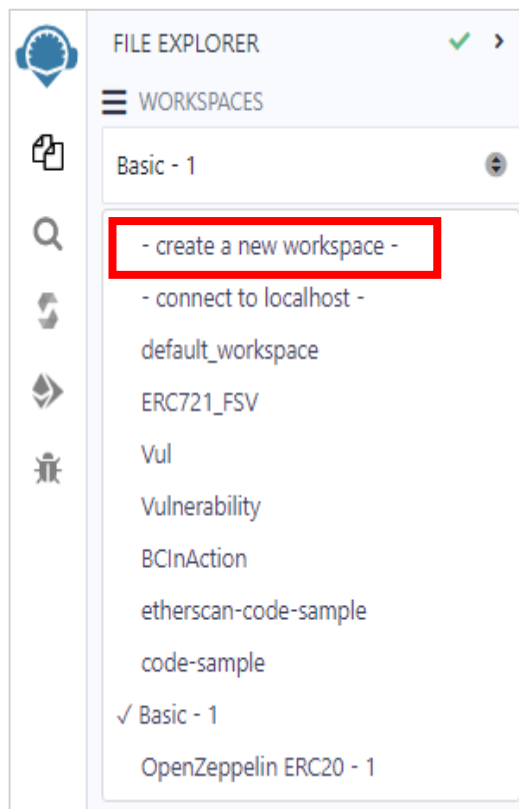
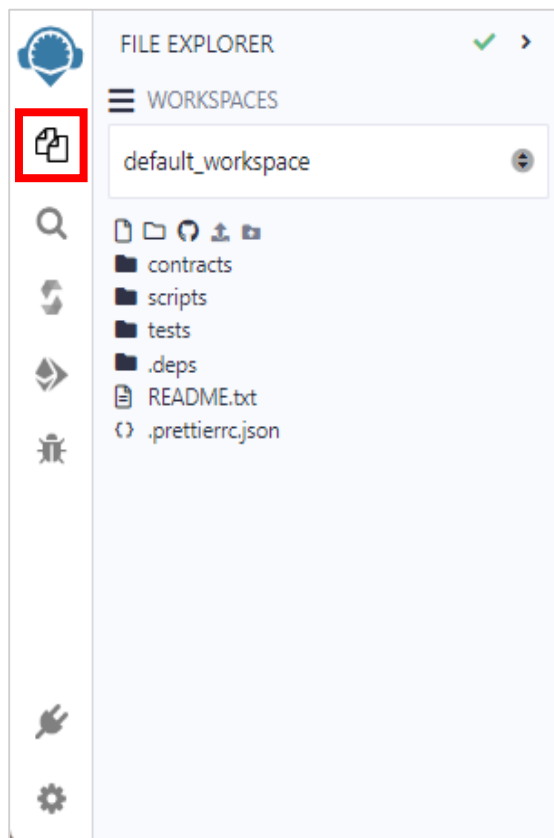
- 스마트 컨트랙트 개발 환경
 - 리믹스(Remix): <https://remix.ethereum.org/>
 - Settings -> Themes 변경



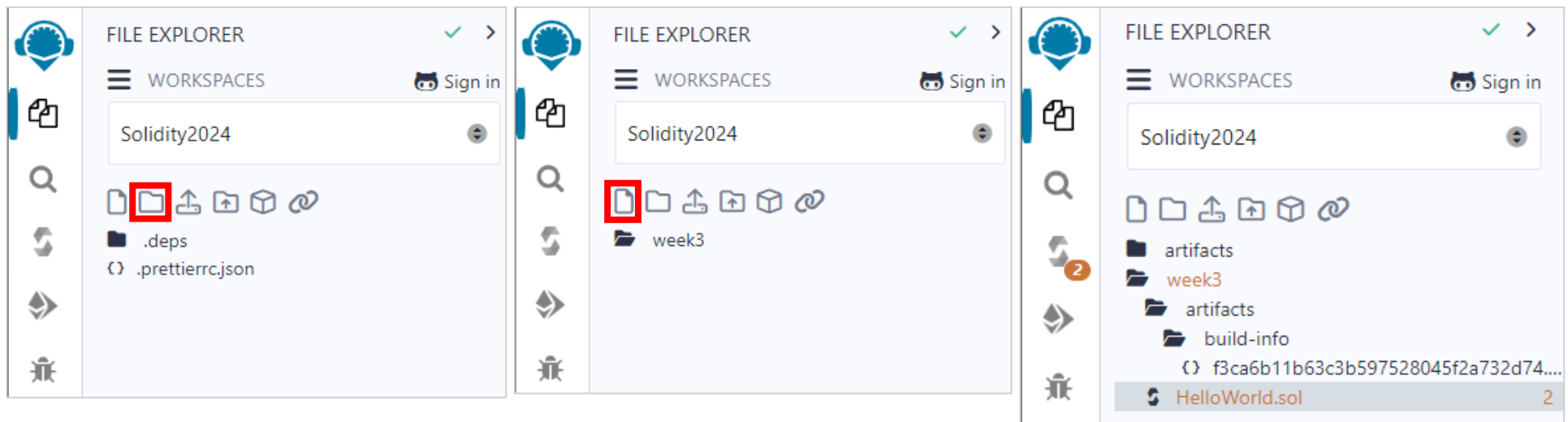
- 리믹스(Remix)
- 기본메뉴
- 메인화면
- 콘솔 화면



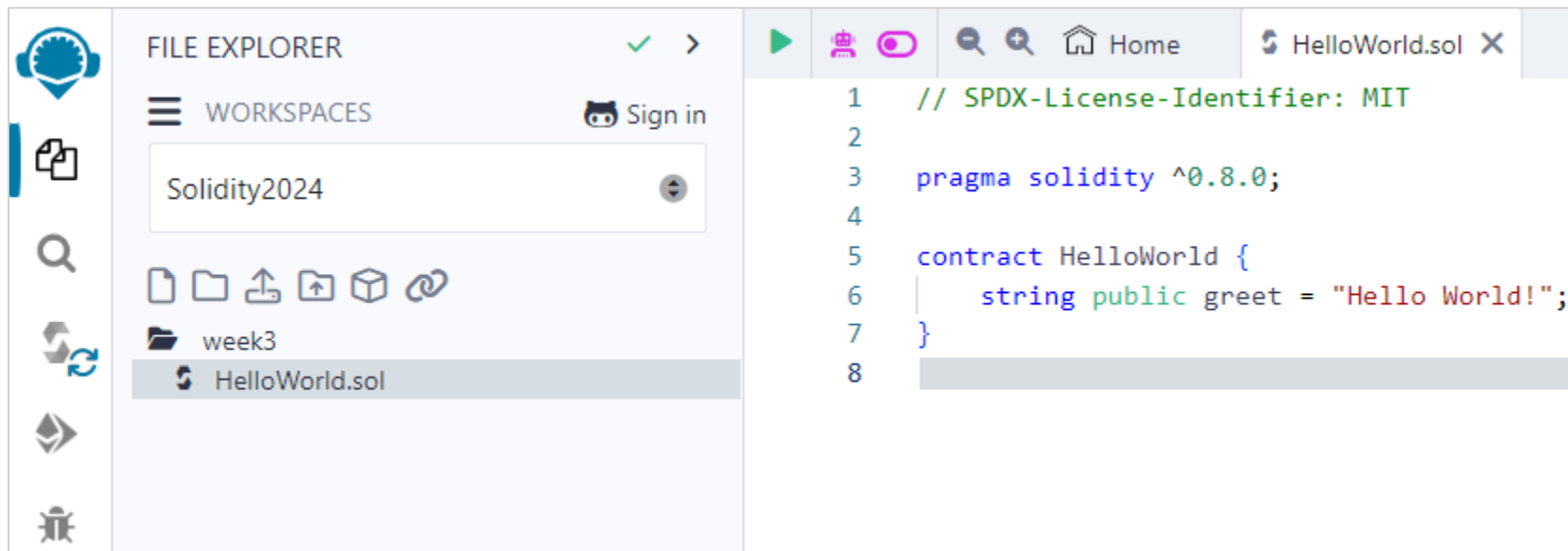
- 리믹스로 스마트컨트랙트 작성, 컴파일, 배포
 - 파일 익스플로러: 폴더와 스마트 컨트랙트 파일 생성
 - 워크스페이스 만들기



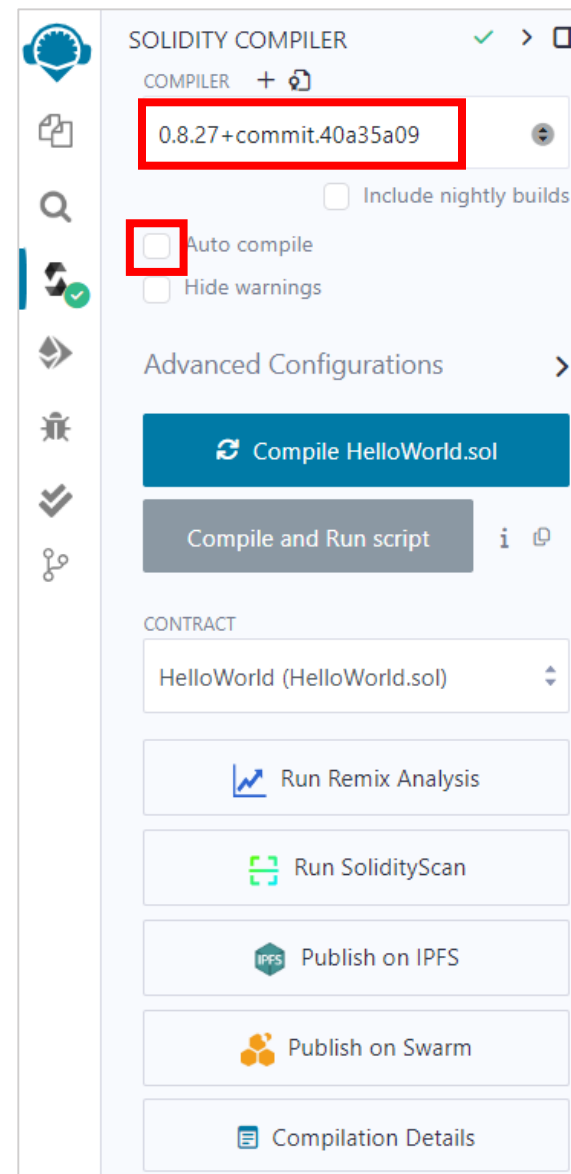
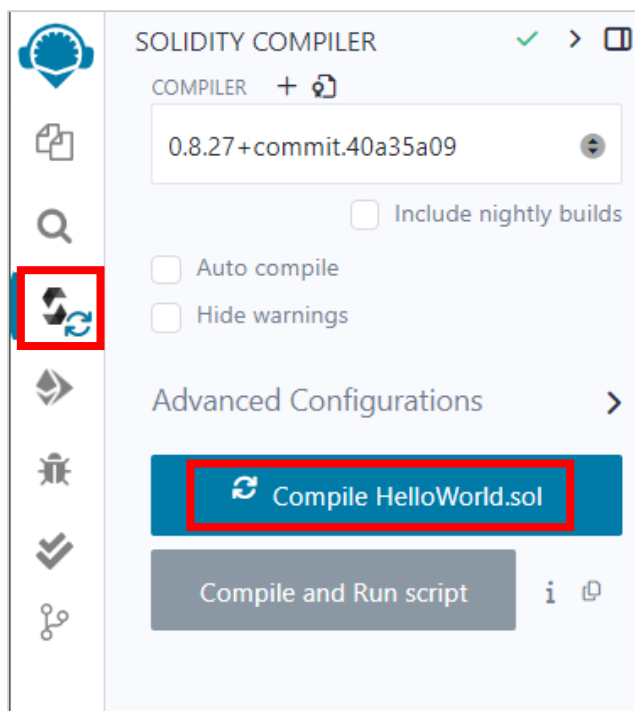
- 리믹스로 스마트컨트랙트 작성,컴파일,배포
 - 새로운 폴더 생성: Create New Folder → week3
 - 새로운 솔리디티 파일(확장자 sol) 생성: Create New File → HelloWorld.sol



- 리믹스로 스마트컨트랙트 작성, 컴파일, 배포
 - HelloWorld.sol 작성



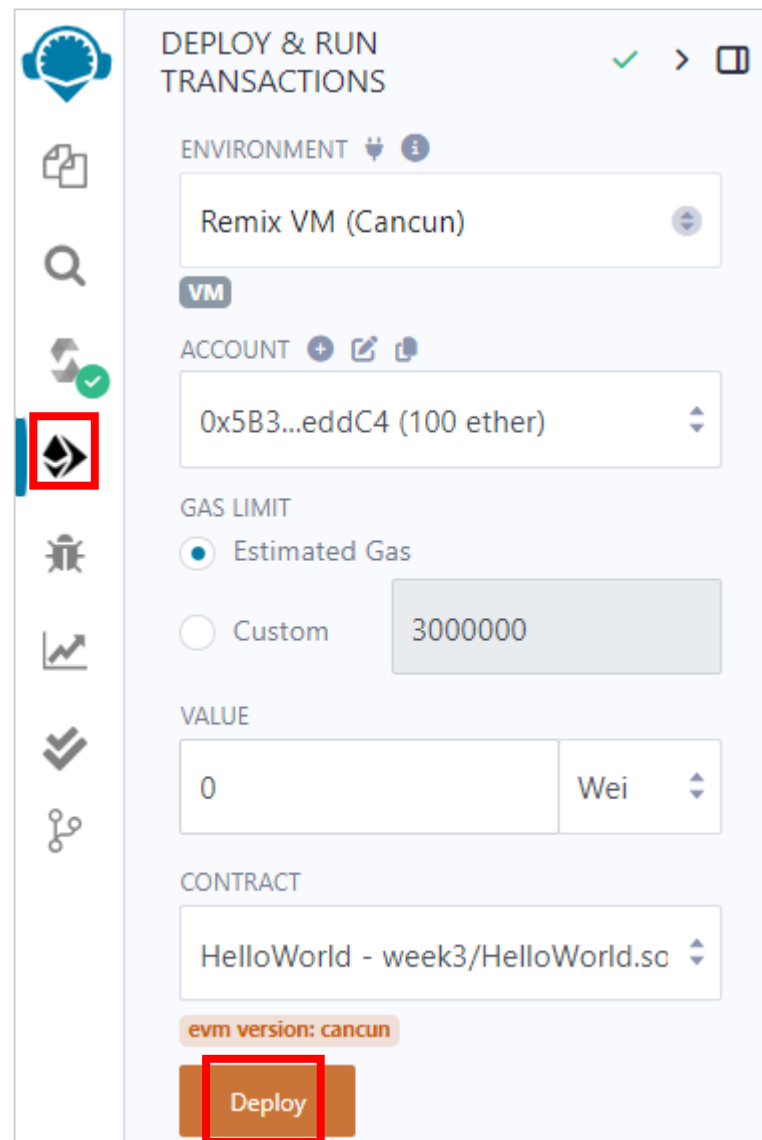
- 리믹스로 스마트컨트랙트 작성,컴파일,배포
 - HelloWorld.sol 컴파일
 - 컴파일러 버전 선택 가능



■ 리믹스로 스마트컨트랙트 작성, 컴파일, 배포

■ HelloWorld.sol 배포

- ENVIRONMENT
 - Remix VM: 리믹스의 로컬 블록체인
- ACCOUNT
 - 100이더를 각각 갖는 가상의 계정 제공
- GAS LIMIT
 - 트랜잭션이 소비할 수 있는 가스의 양 설정
- VALUE
 - 트랜잭션과 함께 전송할 이더
- CONTRACT
 - 배포할 스마트컨트랙트



- 리믹스로 스마트컨트랙트 작성,컴파일,배포
- HelloWorld.sol 배포

The screenshot displays the Remix IDE interface during the deployment of a smart contract. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing the 'Deploy' button and transaction details. The main editor displays the Solidity code for 'HelloWorld.sol'. The right sidebar shows the transaction details for the deployment, including the status, transaction hash, block hash, block number, contract address, and the 'from' and 'to' addresses.

DEPLOY & RUN TRANSACTIONS
evm version: cancan
Deploy
☐ Publish to IPFS
At Address Load contract from Address

Transactions recorded 1

Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network

Deployed/Unpinned Contracts
> HELLOWORLD AT 0XD91...3913

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.0;
4
5 contract HelloWorld {
```

0 Listen on all transactions Filter with transaction hash or address

[vm] from: 0x5B3...eddC4 to: HelloWorld.(constructor) value: 0 wei
data: 0x608...b0033 logs: 0 hash: 0x7e1...af4a8 Debug

status 0x1 Transaction mined and execution succeed

transaction hash 0x7e1d849b5e1301906d902462e84824881eb252ba4154482274976f42827af4a8

block hash 0xb82112e1fcbd6bce418783f57b4d90f64f41d7127b987807b813f8aac0f bec88

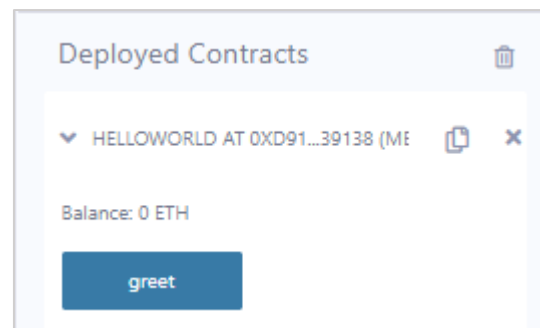
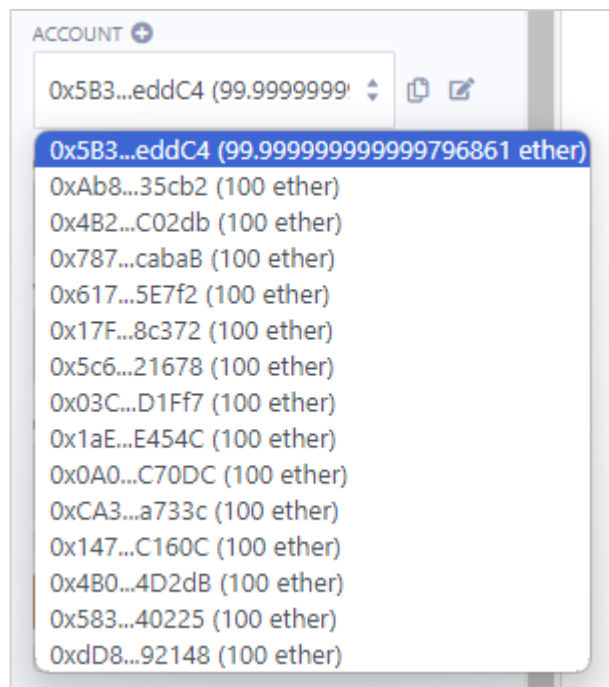
block number 1

contract address 0xd9145CCE52D386f254917e481eB44e9943F39138

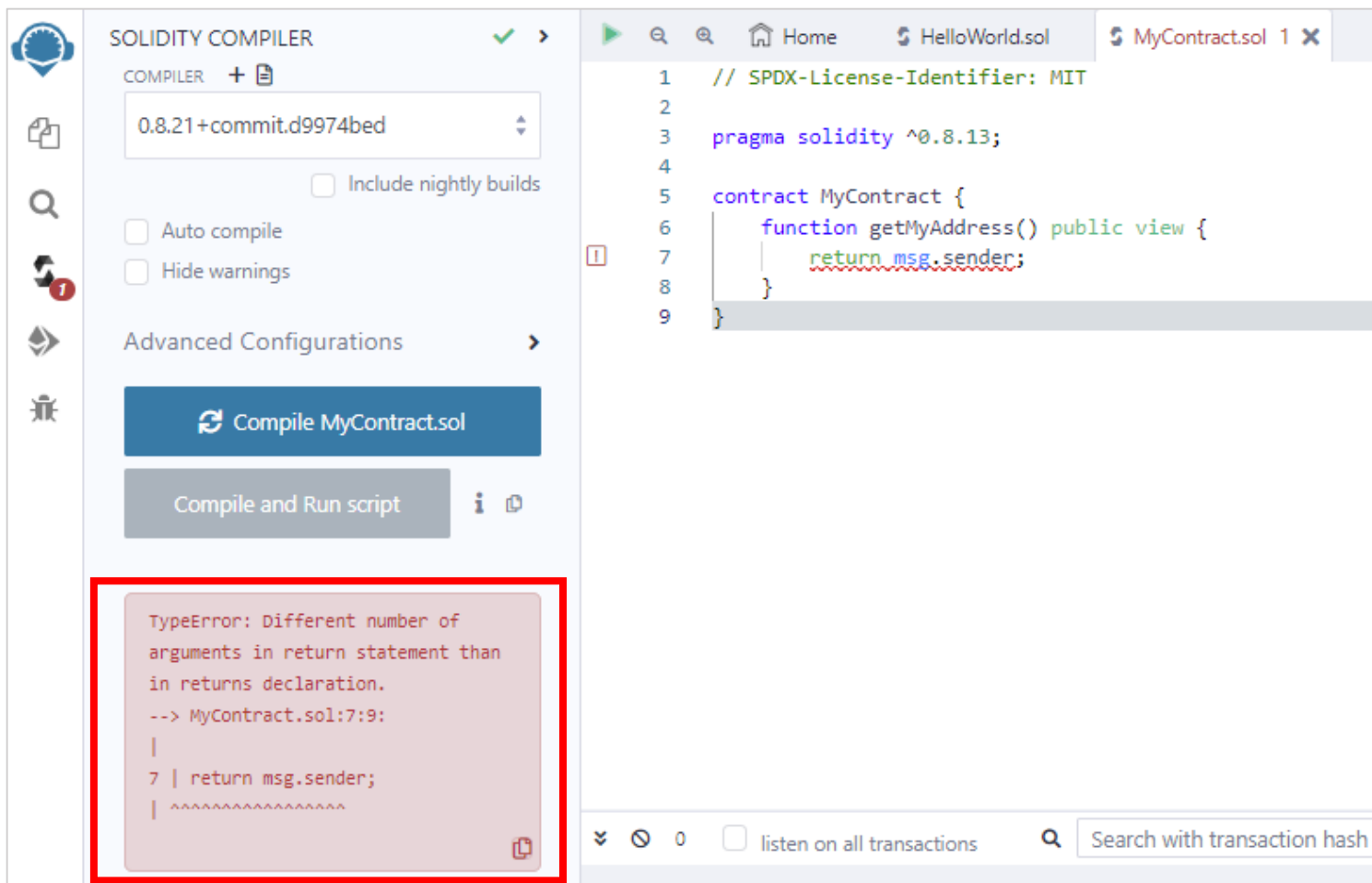
from 0x5B38Da6a701c568545dCf c803FcB875f56beddC4

to HelloWorld.(constructor)

- 리믹스로 스마트컨트랙트 작성, 컴파일, 배포
 - EOA(외부소유계정)과 CA(컨트랙트계정)



- 리믹스로 스마트컨트랙트 작성,컴파일,배포
- 컴파일 에러



The screenshot displays the Remix IDE interface. On the left, the 'SOLIDITY COMPILER' panel shows the compiler version '0.8.21+commit.d9974bed' and options for 'Auto compile' and 'Hide warnings'. A red box highlights the error message in the bottom-left corner of the compiler panel:

```
TypeError: Different number of
arguments in return statement than
in returns declaration.
--> MyContract.sol:7:9:
|
7 | return msg.sender;
|   ^^^^^^^^^^^^^^^
```

The main editor on the right shows the Solidity code for 'MyContract.sol':

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.13;
4
5 contract MyContract {
6     function getMyAddress() public view {
7         return msg.sender;
8     }
9 }
```

The error indicates a mismatch between the function signature 'getMyAddress()' and the return statement 'return msg.sender;'.

- 리믹스로 스마트컨트랙트 작성,컴파일,배포
- 컴파일 경고

The screenshot displays the Remix IDE interface. On the left, the 'SOLIDITY COMPILER' panel shows the version '0.8.21+commit.d9974bed' and options for 'Auto compile' and 'Hide warnings'. Below these are buttons for 'Compile MyContract.sol', 'Compile and Run script', and 'Publish on Ipfs'. The 'CONTRACT' dropdown is set to 'MyContract (MyContract.sol)'. At the bottom of the compiler panel, a warning is highlighted with a red box:

```
Warning: Function state mutability
can be restricted to view
--> MyContract.sol:6:5:
|
6 | function getMyAddress() public
  returns(address) {
  | ^ (Relevant source part starts
  here and spans across multiple
  lines).
```

On the right, the 'MyContract.sol' file is open, showing the following code:

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.13;
4
5 contract MyContract {
6     function getMyAddress() public returns(address) {
7         return msg.sender;
8     }
9 }
```

Below the code editor, the 'Transaction' panel shows the execution details for a call to 'HelloWorld.greet()'. The transaction was successful, with a gas cost of 3488 and a decoded output of 'Hello World!'.

■ 스마트 컨트랙트 기본구조

- SPDX 라이선스 식별자
 - <https://spdx.org/licenses/>
- Version Pragma
 - 솔리디티 컴파일러 버전 정보
 - `pragma solidity >=0.7.0 <0.9.0;`
- contract: 스마트 컨트랙트 작성 시작 키워드
 - `contract 컨트랙트이름 { ... }`
- 주석: 프로그램의 함수, 변수 등에 대한 설명
 - 블록단위
 - 행단위

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Ex2_1 {
    //행 단위 주석
    /*
        블록 단위 주석
    */
}
```

- 리믹스에서 작성한 스마트컨트랙트 저장
 - 다운로드: WORKSPACES, 폴더, 파일에서 오른쪽 마우스 클릭한 후 Download 가능
 - Remixd: 사용자 컴퓨터의 로컬 폴더 및 파일을 리믹스와 연동
 - Node.js(자바스크립트 런타임) 설치
 - <https://nodejs.org/ko>
 - 명령프롬프트 실행
 - node -v : node.js 버전 확인
 - npm -v : npm(Node Package Manager) 버전 확인
 - Remixd 설치
 - npm install -g @remix-project/remixd
 - remixd -v : 버전 확인
 - 폴더 연결(명령프롬프트 닫지 않기)
 - remixd -s <폴더경로> --remix-ide https://remix.ethereum.org

- 리믹스에서 작성한 스마트컨트랙트 저장
- Remixd: 사용자 컴퓨터의 로컬 폴더 및 파일을 리믹스와 연동

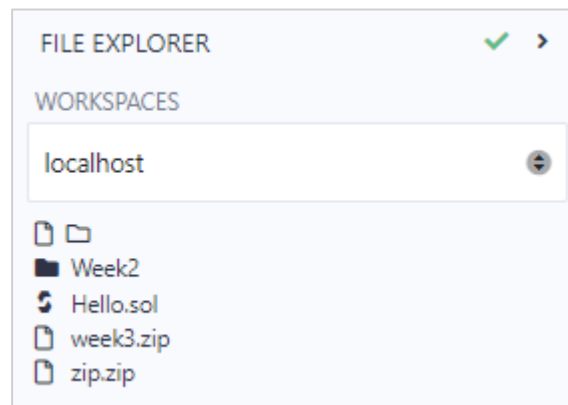
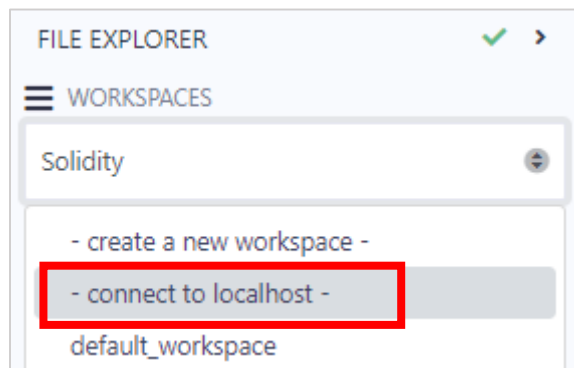
```
C:\Users\moonh>node -v
v18.15.0

C:\Users\moonh>npm -v
9.5.0

C:\Users\moonh>remixd -v
0.6.16

C:\Users\moonh>remixd -s C:\Work\remix_file --remix-ide https://remix.ethereum.org
[WARN] latest version of remixd is 0.6.17, you are using 0.6.16
[WARN] please update using the following command:
[WARN] yarn global add @remix-project/remixd
[WARN] You may now only use IDE at https://remix.ethereum.org to connect to that instance
[WARN] Any application that runs on your computer can potentially read from and write to all files in the directory.
[WARN] Symbolic links are not forwarded to Remix IDE

[INFO] Sun Sep 17 2023 16:15:48 GMT+0900 (대한민국 표준시) remixd is listening on 127.0.0.1:65520
[INFO] Sun Sep 17 2023 16:15:49 GMT+0900 (대한민국 표준시) slither is listening on 127.0.0.1:65523
```



■ 16진수(hexadecimal)

- 0부터 9까지의 수와 A에서 F까지의 로마 문자를 사용
- 이진법 표기의 4자리와 십육진법 한 자리가 일대일 대응
- prefix "0x"

■ 십육진수 F32의 십진수 환산

$$F32 = F \times 16^2 + 3 \times 16 + 2$$

$$= 15 \times 16 \times 16 + 3 \times 16 + 2 = 3840 + 48 + 2 = 3890$$

0 _{hex} = 0 _{dec} = 0 _{oct}	0	0	0	0
1 _{hex} = 1 _{dec} = 1 _{oct}	0	0	0	1
2 _{hex} = 2 _{dec} = 2 _{oct}	0	0	1	0
3 _{hex} = 3 _{dec} = 3 _{oct}	0	0	1	1
4 _{hex} = 4 _{dec} = 4 _{oct}	0	1	0	0
5 _{hex} = 5 _{dec} = 5 _{oct}	0	1	0	1
6 _{hex} = 6 _{dec} = 6 _{oct}	0	1	1	0
7 _{hex} = 7 _{dec} = 7 _{oct}	0	1	1	1
8 _{hex} = 8 _{dec} = 10 _{oct}	1	0	0	0
9 _{hex} = 9 _{dec} = 11 _{oct}	1	0	0	1
A _{hex} = 10 _{dec} = 12 _{oct}	1	0	1	0
B _{hex} = 11 _{dec} = 13 _{oct}	1	0	1	1
C _{hex} = 12 _{dec} = 14 _{oct}	1	1	0	0
D _{hex} = 13 _{dec} = 15 _{oct}	1	1	0	1
E _{hex} = 14 _{dec} = 16 _{oct}	1	1	1	0
F _{hex} = 15 _{dec} = 17 _{oct}	1	1	1	1