Solidity

* Solidity is a **contract-oriented, high-level programming language** for implementing smart contracts.
* Solidity is highly influenced by **C++, Python and JavaScript** and has been designed to target the Ethereum Virtual Machine (EVM).
* Solidity is **statically typed, supports inheritance, libraries and complex user-defined types** programming language.
* **Ethereum is a decentralized** ie. blockchain platform that runs **smart contracts** i.e. applications that run exactly as programmed without any possibility of **downtime, censorship, fraud or third-party interference.**
* The **Ethereum Virtual Machine, also known as EVM**, is the **runtime environment for smart contracts in Ethereum.**
* The Ethereum Virtual Machine focuses on providing **security and executing untrusted code** by computers all over the world.
* The EVM specialised in **preventing Denial-of-service attacks.**
* The Ethereum Virtual Machine has been designed to serve as a **runtime environment for smart contracts based on Ethereum.**
* A smart contract is a **computer protocol** intended to **digitally facilitate, verify, or enforce the negotiation or performance** of a contract.
* Smart contracts allow the **performance of credible transactions** **without third parties.** These transactions are **trackable and irreversible.**
* Solidity is a **statically typed language**, which means that **the state or local variable type** needs to be **specified during declaration.**
* Each declared variable always have a **default value** based on its **type**.
* There is no concept of **"undefined" or "null".**
* Solidity supports **three** types of variables.
* **State Variables** − Variables whose values are **permanently stored in a contract storage.**
* **Local Variables** − Variables whose values are **present till function is executing.**
* **Global Variables** − Special variables exists in the **global namespace** used to get information about the blockchain.
* **Scope of local variables is limited to function** in which they are defined but **State variables can have three types of scopes.**
* **Public** − Public state variables can be accessed **internally** as well as **via messages.** For a public state variable, an automatic getter function is generated.
* **Internal** − Internal state variables can be **accessed only internally** from the **current contract or contract deriving from it** without using this.
* **Private** − Private state variables can be accessed **only internally from the current contract** they are defined not in the derived contract from it.
* In Solidity, an array can be of **compile-time fixed size or of dynamic size.**
* For storage array, it can have **different types of elements** as well.
* In case of memory array, element type can not be mapping and in case it is to be used as function parameter then element type should be an **ABI type.**
* All arrays consist of **contiguous memory locations.** The lowest address corresponds to the first element and the highest address to the last element.
* **Enums restrict a variable to have one of only a few predefined values.** The values in this enumerated list are called enums.
* With the use of enums it is possible **to reduce the number of bugs in your code.**
* To access any member of a structure, we use the **member access operator (.).**
* The member access operator is coded **as a period between the structure variable name and the structure member** that we wish to access.
* In solidity we can use **wei, finney, szabo or ether** as a suffix to a literal to be used to convert various ether based denominations. **Lowest unit is wei and 1e12 represents 1 x 1012.**
* Style Guide helps to maintain **code layout consistent and make code more readable.**
  + **Indentation** − Use **4 spaces instead of tab** to maintain indentation level. Avoid mixing spaces with tabs.
  + **Two Blank Lines Rule** − Use **2 Blank lines between two contract definitions.**
  + **One Blank Line Rule −** Use **1 Blank line between two functions.** In case of only declaration, no need to have blank lines.
  + **Maximum Line Length −** A single line should not cross **79 characters** so that readers can easily parse the code.
  + **Wrapping rules −** First argument be in **new line without opening parenthesis.** Use single indent per argument. Terminating element ); should be the last one.
  + Elements should be layout in following order.
* Pragma statements
* Import statements
* Interfaces
* Libraries
* Contracts
* Within Interfaces, libraries or contracts the order should be as −
* Type declarations
* State variables
* Events
* Functions
* **View functions** ensure that they will not modify the state. A function can be declared as view.
* **Pure functions** ensure that they not read or modify the state. A function can be declared as pure.
* **Withdrawal pattern** ensures that direct transfer call is not made which poses a security threat.
* By Default, a **contract state is read-only** unless it is specified as public.
* We can **restrict who can modify** the contract's state or call a contract's functions using modifiers.