CHEATSHEET: Global Variables

Here is a list of the global variables that we have in Solidity:

*This might change but the most recent solidity global variables you can find them here*[*@SolidityDocumentation*](https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=global%20variables)

* **block.blockhash(uint blockNumber) returns (bytes32)**: hash of the given block - only works for 256 most recent, excluding current, blocks - deprecated in version 0.4.22 and replaced by blockhash(uint blockNumber).
* **block.coinbase (address):** current block miner’s address
* **block.difficulty (uint)**: current block difficulty
* **block.gaslimit (uint)**: current block gaslimit
* **block.number (uint)**: current block number
* **block.timestamp (uint)**: current block timestamp as seconds since unix epoch
* **gasleft() returns (uint256)**: remaining gas
* **msg.data (bytes)**: complete calldata
* **msg.gas (uint)**: remaining gas - deprecated in version 0.4.21 and to be replaced by gasleft()
* **msg.sender (address)**: sender of the message (current call)
* **msg.sig (bytes4)**: first four bytes of the calldata (i.e. function identifier)
* **msg.value (uint)**: number of wei sent with the message
* **now (uint)**: current block timestamp (alias for block.timestamp)
* **tx.gasprice (uint)**: gas price of the transaction
* **tx.origin (address)**: sender of the transaction (full call chain)
* **addmod(uint x, uint y, uint k) returns (uint)**: compute (x + y) % k where the addition is performed with arbitrary precision and does not wrap around at 2\*\*256. Assert that k != 0 starting from version 0.5.0.
* **mulmod(uint x, uint y, uint k) returns (uint)**: compute (x \* y) % k where the multiplication is performed with arbitrary precision and does not wrap around at 2\*\*256. Assert that k != 0 starting from version 0.5.0.
* **keccak256(...) returns (bytes32)**: compute the Ethereum-SHA-3 (Keccak-256) hash of the (tightly packed) arguments
* **sha256(...) returns (bytes32)**: compute the SHA-256 hash of the (tightly packed) arguments
* **sha3(...) returns (bytes32)**: alias to keccak256
* **ripemd160(...) returns (bytes20)**: compute RIPEMD-160 hash of the (tightly packed) arguments
* **ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)**: recover the address associated with the public key from elliptic curve signature or return zero on error (example usage)
* **this**(**current contract’s type**): the current contract, explicitly convertible to Address
* **selfdestruct(address recipient):**destroy the current contract, sending its funds to the given Address
* **suicide(address recipient):**deprecated alias to selfdestruct