**Solidity Units and Global Variables**

**Table of Contents**

* **Ether Units**
* **Time Units**
* **Introduction to Block and Transaction details**
* **Blockhash**
* **Block base fee**
* **Block chain id**
* **Block coinbase**
* **Block difficulty**
* **Block gas limit**
* **Current block number**
* **Current block timestamp**
* **Calculating remaining gas**
* **Fetching complete calldata**
* **Fetching message sender's address(current contract caller)**
* **Calculating number of wei units sent with message**
* **Total transaction gas price**
* **Transaction origin**

**Ether Units:**

Ether is the name of the currency that is used in the Ethereum blockchain. It is used to pay for work done in the EVM. A way to do this is to buy gas for ether. This is done as explained in the gas section.

Transactions are paid with ether.Similar to how one dollar is equal to 100 cent, one ether is equal to 1018 wei.

Denominations:

Metric units are used to measure how much ether there is in each type of unit. There is a name for each denomination that is different from the rest (some bear the family name of seminal figures playing a role in the evolution of computer science and cryptoeconomics). You can think of Wei as being the smallest amount of ether that you can get. Below is a list of the denominations and the value in Wei for each one of them. Following a common pattern, ether also refers to a unit of the currency, which is 1e18, or one quintillion Wei. The currency is not called "Ether," as many people think, and it is not a unit called "Ether."

| **Unit** | **Wei Value** | **Wei** |
| --- | --- | --- |
| **wei** | 1 wei | 1 |
| **Kwei (babbage)** | 1e3 wei | 1,000 |
| **Mwei (lovelace)** | 1e6 wei | 1,000,000 |
| **Gwei (shannon)** | 1e9 wei | 1,000,000,000 |
| **microether (szabo)** | 1e12 wei | 1,000,000,000,000 |
| **milliether (finney)** | 1e15 wei | 1,000,000,000,000,000 |
| **ether** | 1e18 wei | 1,000,000,000,000,000,000 |

Solidity Program to understand Ether units:

| // SPDX-License-Identifier: MIT pragma solidity ^0.8.10;  contract EtherUnits {  uint public oneWei = 1 wei;  // 1 wei is equal to 1  bool public isOneWei = 1 wei == 1;   uint public oneEther = 1 ether;  // 1 ether is equal to 10^18 wei  bool public isOneEther = 1 ether == 1e18;  } |
| --- |

All of the following assertions in Solidity will be equivalent to true

| assert(1 wei == 1); assert(1 szabo == 1e12); assert(1 finney == 1e15); assert(1 ether == 1e18); assert(2 ether == 2000 fenny); |
| --- |

**Time Units**

Solidity has time units, the same as Ether units. The lowest unit is the second, and we can use seconds, minutes, hours, days, and weeks as suffixes to show time.

All of the following assertions in Solidity will be equivalent to true

| assert(1 seconds == 1); assert(1 minutes == 60 seconds); assert(1 hours == 60 minutes); assert(1 day == 24 hours); assert(1 week == 7 days); |
| --- |

Take care when you use these units when you do calendar calculations. Because of leap seconds, not every year has 365 days, and not even every day has 24 hours. To keep an exact calendar library up to date, a third-party oracle must do the job. Leap seconds can't be predicted.

**Block and Transaction details**

The following code implements all major solidity global variables along with block and transaction details

### Block and Transaction Variables:

* blockhash(uint blockNumber) returns (bytes32): hash of the given block when blocknumber is one of the 256 most recent blocks; otherwise returns zero
* block.basefee (uint): current block’s base fee ([EIP-3198](https://eips.ethereum.org/EIPS/eip-3198) and [EIP-1559](https://eips.ethereum.org/EIPS/eip-1559))
* block.chainid (uint): current chain id
* block.coinbase (address payable): 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 calldata): complete calldata
* 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
* tx.gasprice (uint): gas price of the transaction
* tx.origin (address): sender of the transaction (full call chain)

| // SPDX-License-Identifier: MIT pragma solidity ^0.8.0;  contract TransactionVariables{    event loguint(uint);  event logbytes(bytes);  event logaddress(address);  event logbyte4(bytes4);  event logblock(bytes32);      function showBlockVariables() public payable{  emit logaddress(block.coinbase);  emit loguint(block.difficulty);  emit loguint(block.gaslimit);  emit loguint(gasleft());  emit loguint(tx.gasprice);  emit loguint(block.number);  emit loguint(block.timestamp);  emit logbytes(msg.data);  emit logbyte4(msg.sig);  emit logaddress(msg.sender);  emit loguint(msg.value);  emit logaddress(tx.origin);  emit logblock(blockhash(block.number));  } } |
| --- |
|  |