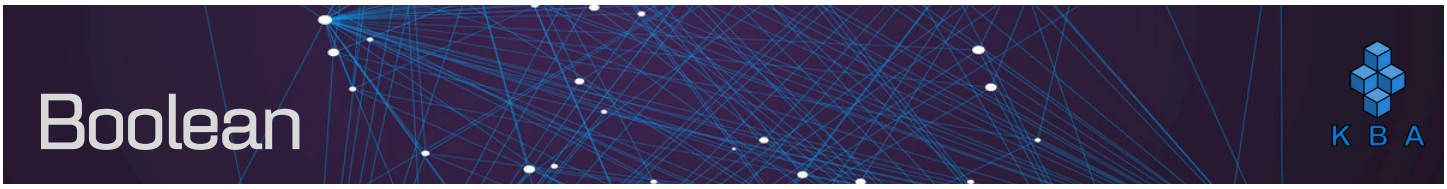
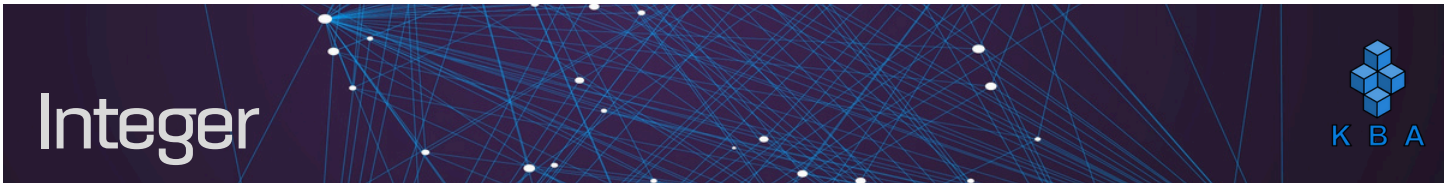


bool: The possible values are **true** and **false**.

Operators

- **!** (logical negation)
- **&&** (logical conjunction, “and”)
- **||** (logical disjunction, “or”)
- **==** (equality)
- **!=** (inequality)

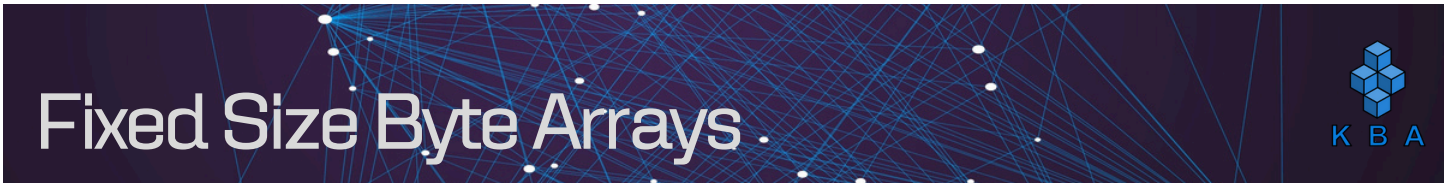
The operators **||** and **&&** apply the common short-circuiting rules. This means that in the expression **f(x) || g(y)** if **f(x)** evaluates to **true**, **g(y)** will not be assessed even if it may have side effects.



int/**uint**: signed and unsigned integers of various sizes. Keywords **uint8** to **uint256** in steps of **8** (unsigned of **8** to **256** bits) and **int8** to **int256**. **uint** and **int** are aliases for **uint256** and **int256** respectively.

Operators

- Comparison: **<=**, **<**, **==**, **!=**, **>=**, **>** (evalute to bool)
- Bit operators: **&**, **|**, **^** (bitwise exclusive or), **~** (bitwise negation)
- Shift operators: **<<** (left shift), **>>** (right shift)
- Arithmetic operators: **+**, **-**, unary **-**, *****, **/**, **%** (modulo), ****** (exponentiation)



The value types **bytes1**, **bytes2**, **bytes3**, ..., **bytes32** hold a sequence of **bytes** from one up to 32. byte is an alias of **bytes1**.

Operators

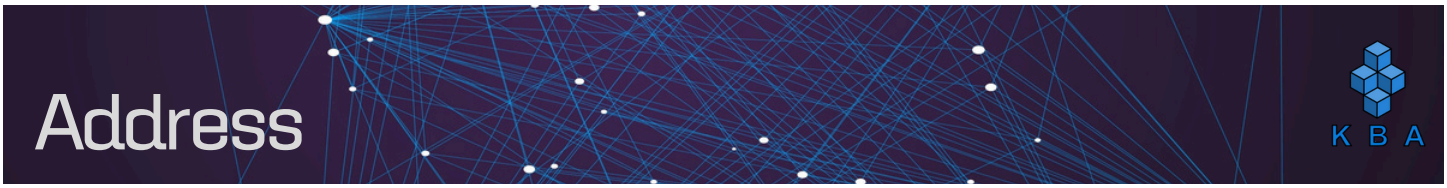
- Comparison: **<=**, **<**, **==**, **!=**, **>=**, **>** (evalute to **bool**)
- Bit operators: **&**, **|**, **^** (bitwise exclusive or), **~** (bitwise negation)

- Shift operators: **<<** (left shift), **>>** (right shift)
- Index access: If **x** is of type **bytesl**, then **x[k]** for **0 <= k < l** returns the **k**-th byte (read-only).

Members

.length yields the fixed length of the byte array (read-only).

- **bytes**: Dynamically sized byte array. Not a value type.
- **string**: Dynamically sized UTF-8 encoded string. Not a value type.



The address type can be of two varieties:

- **address**: Holds a 20-byte value (size of an Ethereum address)
- **address payable**: Same as **address**, but has additional members **transfer** and **send**.

Ether can be sent only to **address payable**, whereas a plain **address** cannot receive ether.

Type Conversions

Implicit conversions from **address payable** to **address** are allowed, whereas conversions from **address** to **address payable** must be explicit via **payable()**.

Explicit conversions to and from **address** are allowed for integers, integer literals, **bytes20**, and contract types.

Only expressions of type **address** and contract type can be converted to the type **address payable** via the explicit conversion **payable()**. For contract type, this conversion is only allowed if the contract can receive ether.

Operators

<=, **<**, **==**, **!=**, **>=** and **>**

Members of Address

It is possible to query the **balance** of an address using the property balance and to send ether (in units of wei) to a payable address using the **transfer** function.

- **address payable x = address(0x123);**
- **address myAddress = address(this);**
- **if (x.balance <10 && myAddress.balance >= 10) x.transfer(10);**

The **transfer** function fails if the balance of the current contract is not large enough or if the Ether balance is rejected by the receiving account. The **transfer** function reverts on failure. **send** is the low-level counterpart of **transfer**. If the execution fails, the current contract will not stop with an exception, but **send** will return **false**.

Type Information

- **type(C).name (string)**: the name of the contract
- **type(C).creationCode (bytes memory)**: creation bytecode of the given contract
- **type(C).runtimeCode (bytes memory)**: runtime bytecode of the given contract
- **type(I).interfaceId (bytes4)**: value containing the EIP-165 interface identifier of the given interface
- **type(T).min (T)**: the minimum value representable by the integer type **T**
- **type(T).max (T)**: the maximum value representable by the integer type **T**

Members of Address Types

- **<address>.balance(uint256)**: balance of address in wei
- **<address payable>.transfer(uint256 amount)**: send given amount of Wei to address, reverts on failure.
- **<address payable>.send(uint256 amount) returns (bool)**: send given amount of Wei to address, returns false on failure.

Block and Transaction Properties

- **blockhash(uint blockNumber) returns (bytes32)**: hash of the given block - only work for 256 most recent, excluding current blocks
- **block.coinbase (address payable)**: current block validator'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 payable)**: 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 payable)**: sender of the transaction (full call chain)

Ether Units

A literal number can take a suffix of **wei**, **finney**, **szabo**, or **ether** to specify a sub-denomination of **ether**, where ether numbers without a postfix are assumed to be **wei**.

- **assert(1 wei == 1);**
- **assert(1 szabo == 1e12);**
- **assert(1 finney == 1e15);**
- **assert(1 ether == 1e18);**

The only effect of the sub-denomination suffix is a multiplication by a power of **10**.