## sha

- sha1()

### sha1

Secure Hash Algorithm 1

```
#bytes-to-hex(sha1("Hello World!"))
```

```
2ef7bde608ce5404e97d5f042f95f89f1c232871
```

**Parameters**

```
sha1(
  message: str,
  iv: array
) -> bytes
```

**message** `str`

Message to hash

**iv** `array`

Initial vector

Default: `sha1-default-iv`

## misc

- hmac()

### hmac

Hash-based Message Authentication Code

```
#bytes-to-hex(hmac("Key", "Hello World!"))
```

```
0bec6dbeb923f906fa3ec96433e00fa12fb91dec
```

**Parameters**

```
hmac(
  key: str bytes,
  message: str bytes,
  hash-func: function,
  block-size: number
) -> bytes
```

**key** `str` or `bytes`

Hashing key

**message** `str` or `bytes`

Message to hash

**hash-func** `function`

Hashing function

Default: `sha1`

**block-size** `number`

Block size

Default: `64`

## base

- b32-decode()
- b32-encode()

### b32-decode

Decodes a base32-encoded value

```
#str(b32-decode("LFHVKUCJ"))
```

```
YOUPI
```

**Parameters**

```
b32-decode(encoded: str) -> bytes
```

### b32-encode

Encodes a value in base32

```
#b32-encode(bytes("YOUPI"))
```

```
LFHVKUCJ
```

**Parameters**

```
b32-encode(decoded: bytes) -> str
```

## utils

- bin-to-int()
- bytes-to-hex()
- circular-shift()
- xor-bytes()
- z-fill()

### bin-to-int

Converts an array of bits into an integer

```
#let bits = (0, 0, 1, 0, 1, 0, 1, 0)
#bin-to-int(bits)
```

```
42
```

**Parameters**

```
bin-to-int(bin: array) -> number
```

**bin** `array`

Bit array

### bytes-to-hex

Converts a byte array to a hexadecimal string

```
#let b = bytes((0xfa, 0xca, 0xde))
#bytes-to-hex(b)
```

```
facade
```

**Parameters**

```
bytes-to-hex(bytes: bytes) -> str
```

### circular-shift

Rotates a number to the left (wrapping the leftmost bits to the right)

```
#let a = 42
#let b = circular-shift(a, n: 20)
#let c = circular-shift(b, n: 11)
#b, #c
```

```
44040192, 21
```

**Parameters**

```
circular-shift(
  x: number,
  n: number
) -> number
```

**x** `number`

Number to rotate

**n** `number`

Shift amount

Default: 1

### xor-bytes

Applies the XOR operation between two byte arrays

```
#let a = bytes((0b010, 0b0111))
#let b = bytes((0b011, 0b0101))
#array(xor-bytes(a, b)).map(
  b => z-fill(str(b, base: 2), 3)
)
```

```
("001", "010")
```

#### Parameters

```
xor-bytes(
  bytes-a: bytes,
  bytes-b: bytes
) -> bytes
```

**bytes-a** `bytes`

First byte array

**bytes-b** `bytes`

Second byte array

### z-fill

Pads a string with 0s on the left to reach a certain length

```
#z-fill("1011", 8)
```

```
00001011
```

#### Parameters

```
z-fill(
  string: str,
  length: number
) -> str
```