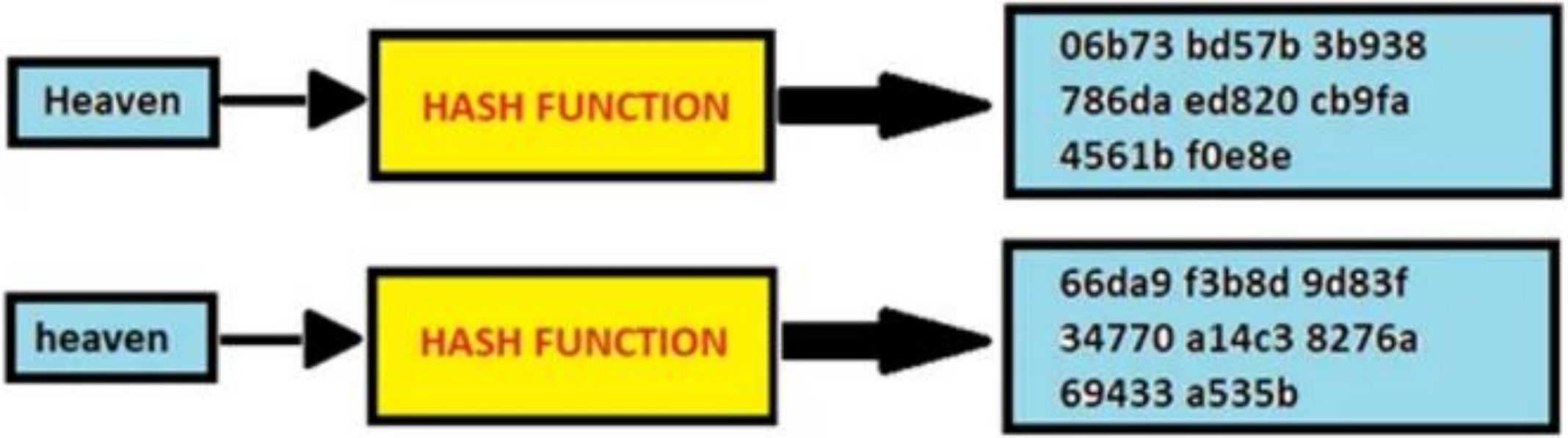
物聯網實務

10_12

廖裕評

SHA (Secure Hash Algorithm)安全雜湊演算法



https://www.encryptionconsulting.com/education-center/what-is-sha/

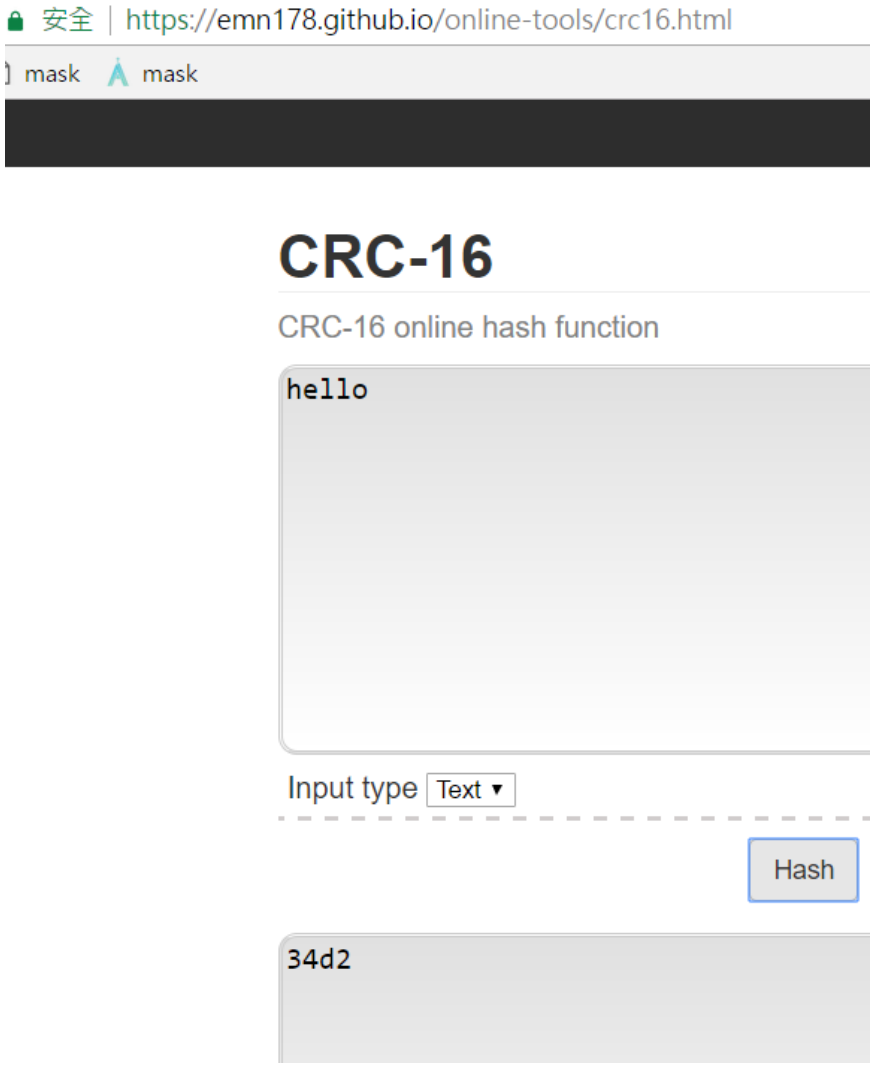
SHA函式對比

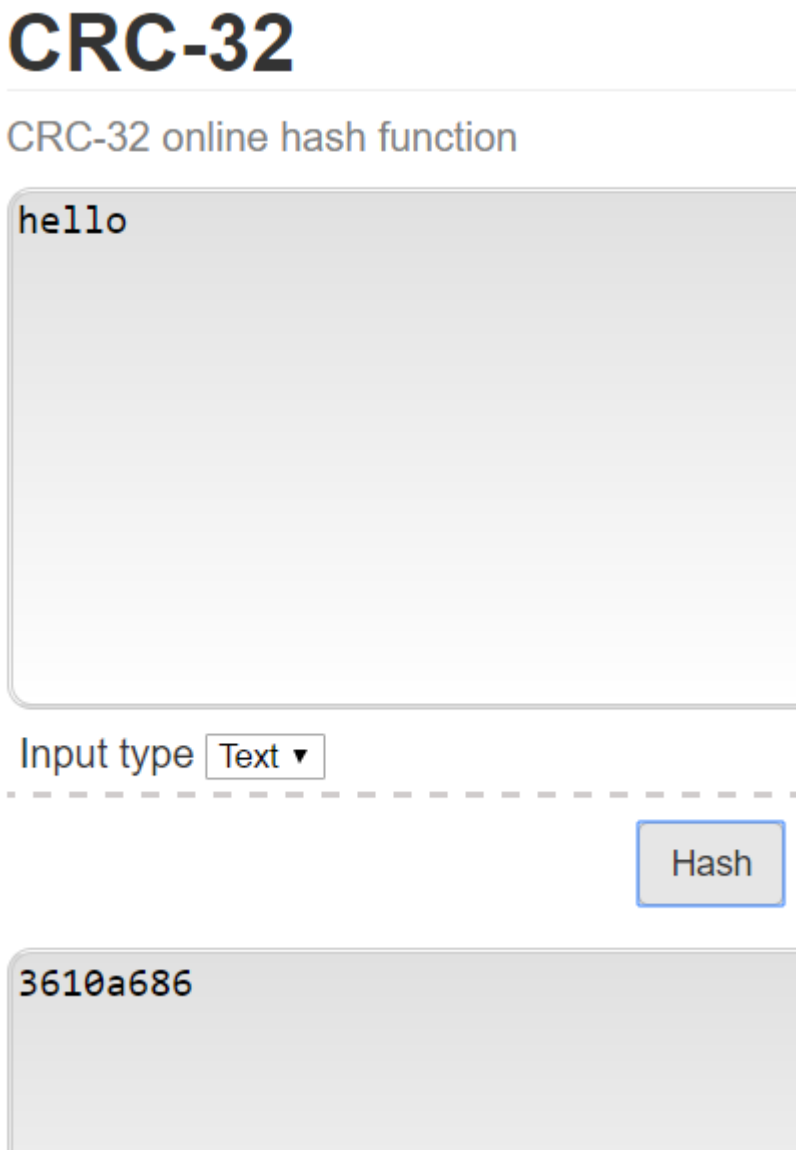
演算法和變體		輸出雜湊值長度（bits）	中繼雜湊值長度（bits）	資料區塊長度（bits）	最大輸入訊息長度（bits）	迴圈次數	使用到的運算子
MD5（作為參考）		128	128 (4 × 32)	512	無限[4]	64	And, Xor, Rot, Add (mod 2^{32}), Or
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or
SHA-1		160	160 (5 × 32)	512	$2^{64} - 1$	80	
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	無限[7]	24[8]	And, Xor, Rot, Not
	SHAKE128 *SHAKE256*	*d* (arbitrary) *d* (arbitrary)		1344 1088			

https://zh.wikipedia.org/zh-tw/SHA%E5%AE%B6%E6%97%8F

https://iter01.com/602330.html

hash





https://emn178.github.io/online-tools/crc32.html

MD2

MD2 online hash function

hello

Hash

a9046c73e00331af68917d3804f70655

MD4

MD4 online hash function

hello

Input type Text ▾

Hash

866437cb7a794bce2b727acc0362ee27

MD5

MD5 online hash function

```
hello
```

Input type Text ▾

Hash

```
5d41402abc4b2a76b9719d911017c592
```

SHA1

SHA1 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
```

SHA224

SHA224 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
ea09ae9cc6768c50fcee903ed054556e5bfc8347907f12598aa24193
```

SHA256

SHA256 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

SHA384

SHA384 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
59e1748777448c69de6b800d7a33bbfb9ff1b463e44354c3553bcdb9c666fa90125a3c79f
90397bdf5f6a13de828684f
```

SHA512

SHA512 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
9b71d224bd62f3785d96d46ad3ea3d73319bfbc2890caadae2dff72519673ca72323c3d99
ba5c11d7c7acc6e14b8c5da0c4663475c2e5c3adef46f73bcdec043
```

SHA512/224

SHA512/224 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
fe8509ed1fb7dcefc27e6ac1a80eddbec4cb3d2c6fe565244374061c
```

SHA512/256

SHA512/256 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
e30d87cfa2a75db545eac4d61baf970366a8357c7f72fa95b52d0accb698f13a
```

SHA3-224

SHA3-224 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
b87f88c72702fff1748e58b87e9141a42c0dbedc29a78cb0d4a5cd81
```

SHA3-256

SHA3-256 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
3338be694f50c5f338814986cdf0686453a888b84f424d792af4b9202398f392
```

SHA3-384

SHA3-384 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
720aea11019ef06440fbf05d87aa24680a2153df3907b23631e7177ce620fa1330ff07c0f
ddee54699a4c3ee0ee9d887
```

SHA3-512

SHA3-512 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
75d527c368f2efe848ecf6b073a36767800805e9eef2b1857d5f984f036eb6df891d75f72
d9b154518c1cd58835286d1da9a38deba3de98b5a53e5ed78a84976
```

Keccak-224

Keccak-224 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
45524ec454bcc7d4b8f74350c4a4e62809fcb49bc29df62e61b69fa4
```

Keccak-256

Keccak-256 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
1c8aff950685c2ed4bc3174f3472287b56d9517b9c948127319a09a7a36deac8
```

Keccak-384

Keccak-384 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
dcef6fb7908fd52ba26aaba75121526abbf1217f1c0a31024652d134d3e32fb4cd8e9c703
b8f43e7277b59a5cd402175
```

Keccak-512

Keccak-512 online hash function

```
hello
```

Input type Text ▾

Hash ☑ Auto Update

```
52fa80662e64c128f8389c9ea6c73d4c02368004bf4463491900d11aaadca39d47de1b013
61f207c512cfa79f0f92c3395c67ff7928e3f5ce3e3c852b392f976
```

Shake-128

Shake-128 online hash function

```
hello
```

Input type Text ▾

Output Bits: 256

Hash ☑ Auto Update

```
8eb4b6a932f280335ee1a279f8c208a349e7bc65daf831d3021c213825292463
```

Shake-256

Shake-256 online hash function

```
hello
```

Input type Text ▾

Output Bits: 512

Hash ☑ Auto Update

```
1234075ae4a1e77316cf2d8000974581a343b9ebbca7e3d1db83394c30f221626f594e4f0
de63902349a5ea5781213215813919f92a4d86d127466e3d07e8be3
```

Base32 Encode

Base32 online encode function

```
hello
```

Input type Text ▾

Encode ☑ Auto Update

```
NBSWY3DP
```

Base64 Encode

Base64 online encode function

```
hello
```

Input type Text ▾

Encode ☑ Auto Update

```
aGVsbG8=
```

Base64 Decode

Base64 online decode function

```
aGVsbG8K
```

Decode ☑ Auto Update

```
hello
```

Base32 Decode

Base32 online decode function

```
NBSWY3DP
```

Decode ☑ Auto Update

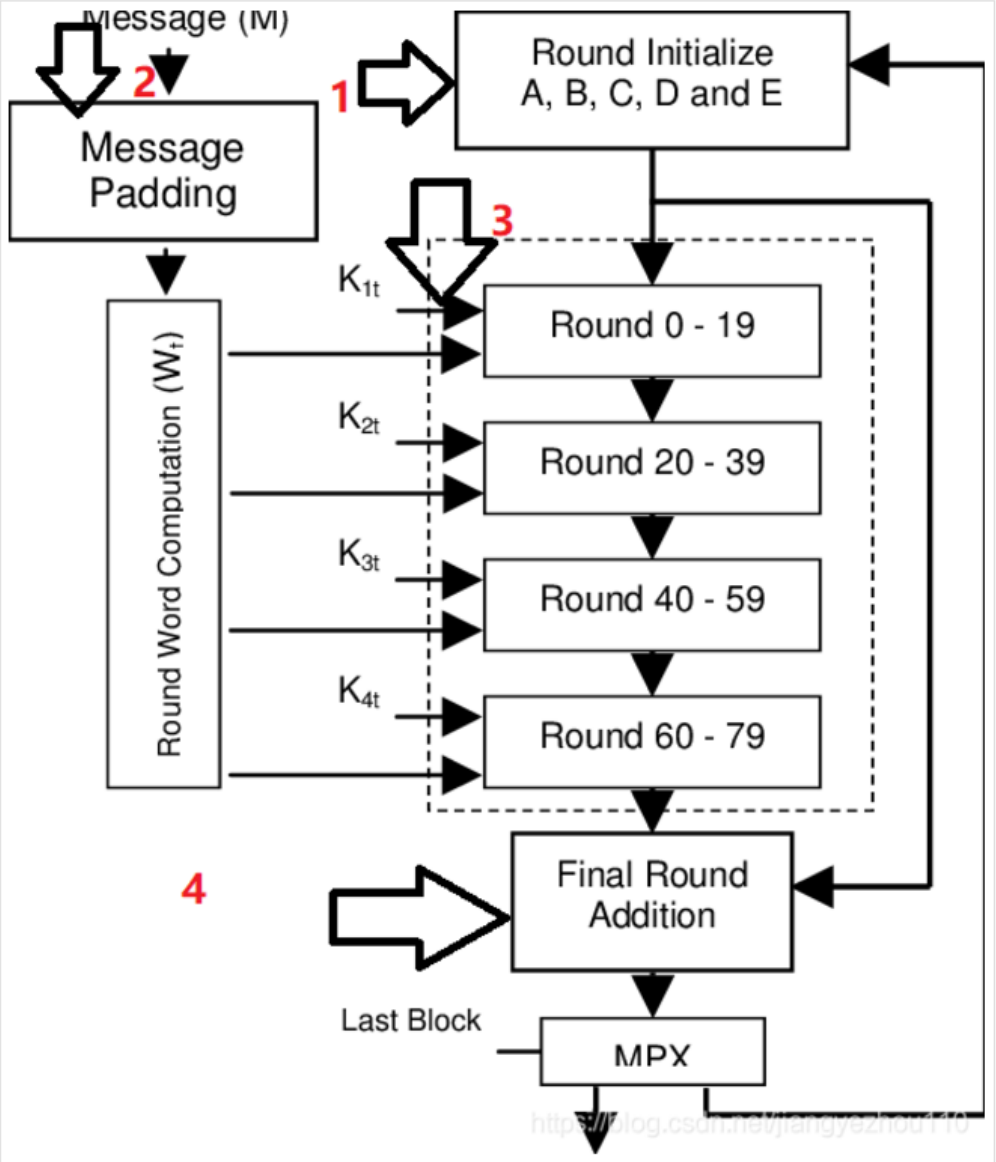
```
hello
```

Base64 encoder

文字（1 Byte）	A																							
位元	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位元（補0）	0	1	0	0	0	0	0	1	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
Base64編碼	**Q**						**Q**						**=**				**=**							
文字（2 Byte）	B								C															
位元	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
位元（補0）	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	**0**	**0**	0	0	0	0	0	0
Base64編碼	**Q**						**k**						**M**				**=**							

https://zh.wikipedia.org/zh-tw/Base64

SHA-1演算法在FPGA上的實作-1



https://www.uj5u.com/qukuanlian/245258.h

SHA-1演算法在FPGA上的實作-2

哈希計算（Hash Function）

哈希函式SHA-1的具體計算在上一篇中已經詳細介紹，這里不再贅述，直接給出代碼，

SHA-1 Function

在SHA-1演算法的計算輪次中，使用到了三種不同的函式，分別是Ch(x,y,z)、Parity(x,y,z)、Maj(x,y,z)，以及$ROTL^n(x)$具體的代碼實作展示如下：

```
// ch(x,y,z)
function [31:0] sha1_ch();
input [31:0] x,y,z;
begin
    sha1_ch = (x & y) ^ (~x & z);
end
endfunction
// parity(x,y,z)
function [31:0] sha1_parity();
input [31:0] x,y,z;
begin
    sha1_parity = x ^ y ^ z;
end
endfunction
// maj(x,y,z)
function [31:0] sha1_maj();
```

https://www.uj5u.com/qukuanlian/249799.html

crypto-js

SHA-1

The SHA hash functions were designed by the National Security Agency (NSA). SHA-1 is the most established of the existing SHA hash functions, and it's used in a variety of security applications and protocols. Though, SHA-1's collision resistance has been weakening as new attacks are discovered or improved.

```
var hash = CryptoJS.SHA1("Message");
```

SHA-2

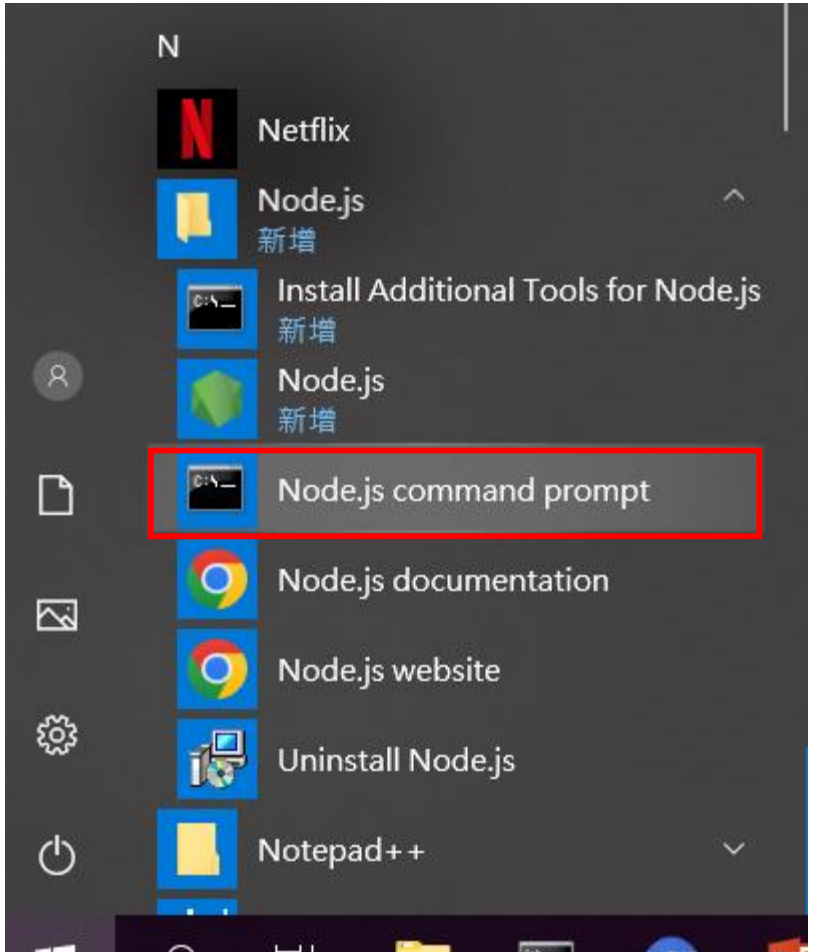
SHA-256 is one of the four variants in the SHA-2 set. It isn't as widely used as SHA-1, though it appears to provide much better security.

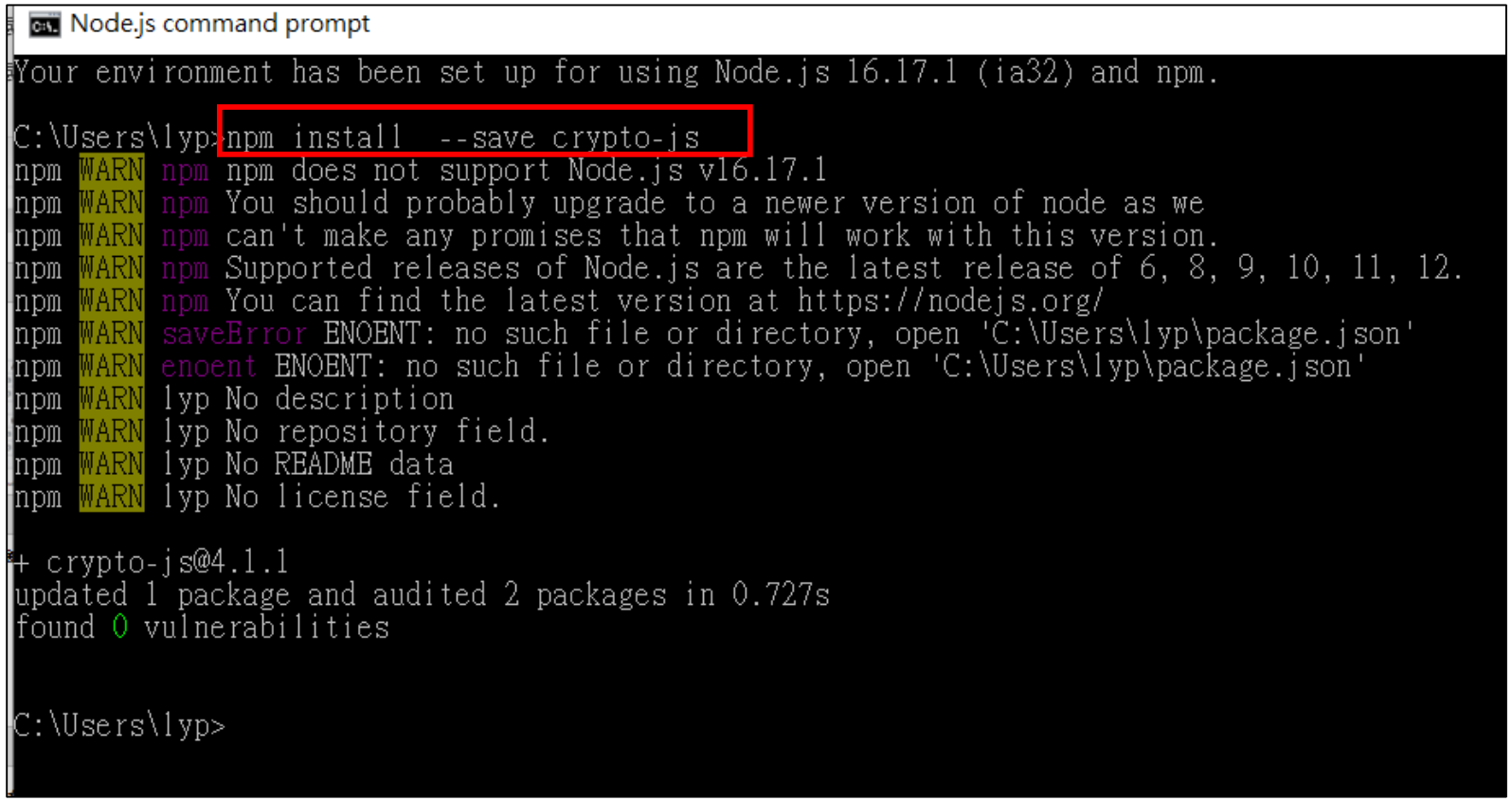
```
var hash = CryptoJS.SHA256("Message");
```

https://cryptojs.gitbook.io/docs/

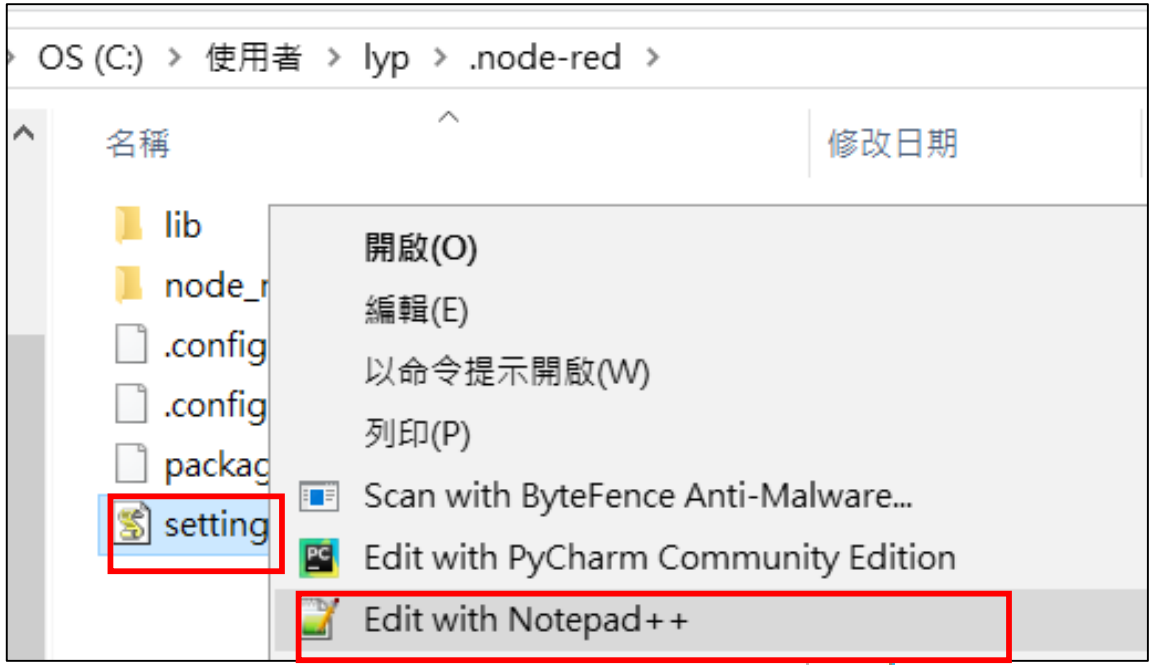
Install crypto-js

- npm install --save crypto-js





Edit settings.js ➡ Save settings.js




```
functionGlobalContext: {
    // os:require('os'),
    cryptojs:require('crypto-js')
},
```

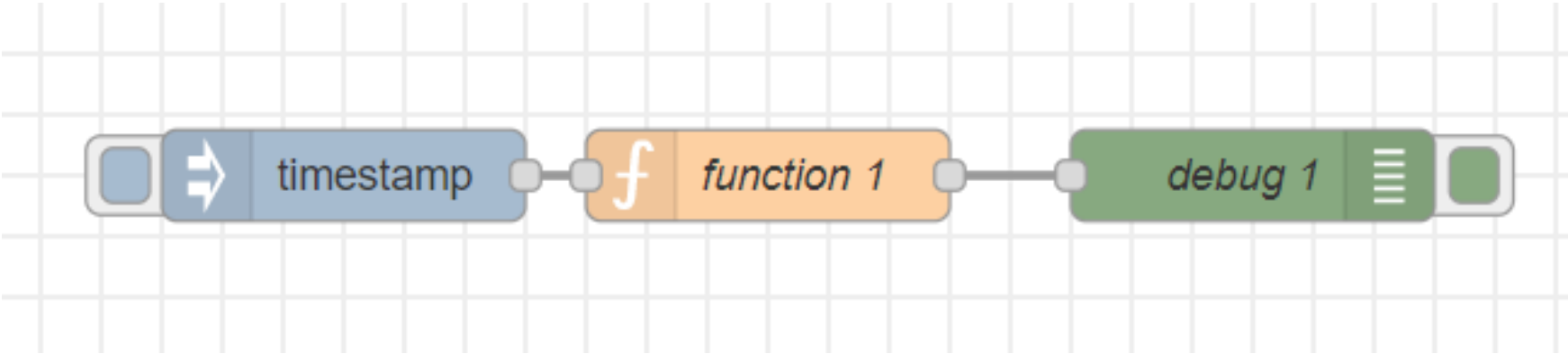
Running node-red

- node-red

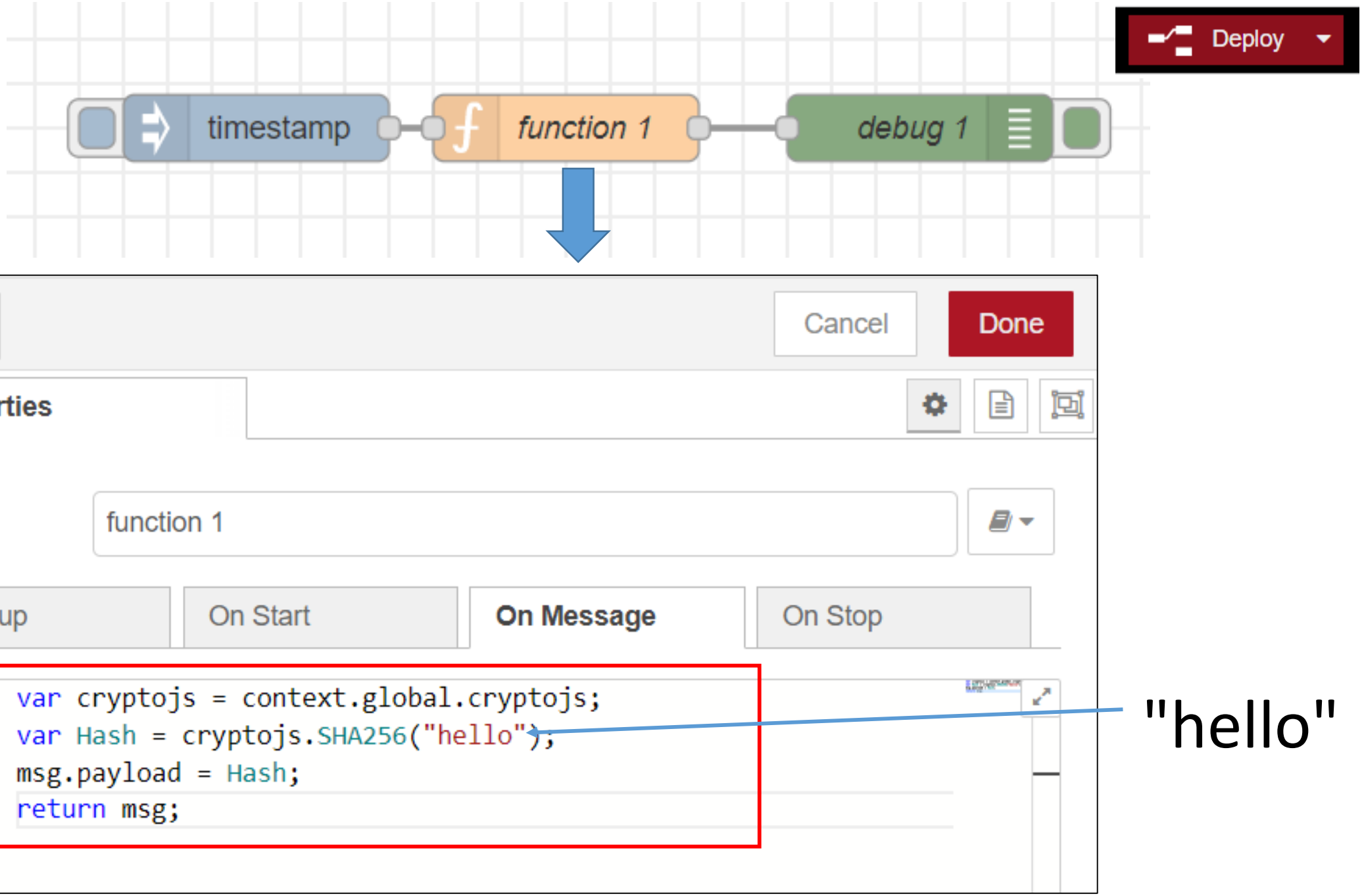
node-red

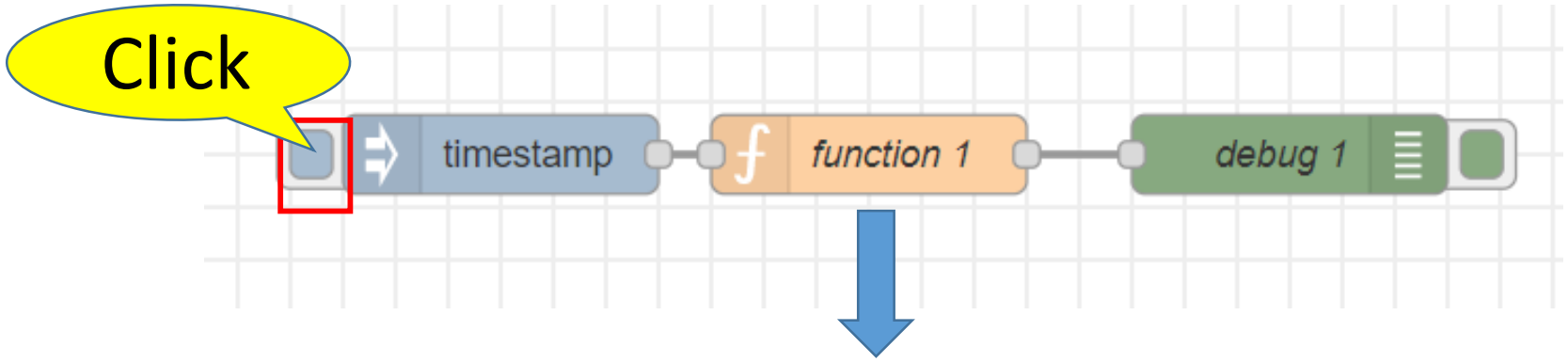
```
C:\Users\lyp>node-red
8 Oct 12:58:49 - [info]

Welcome to Node-RED
===================

8 Oct 12:58:49 - [info] Node-RED version: v3.0.2
8 Oct 12:58:49 - [info] Node.js  version: v16.17.1
8 Oct 12:58:49 - [info] Windows_NT 10.0.17763 ia32 LE
8 Oct 12:58:49 - [info] Loading palette nodes
8 Oct 12:58:49 - [info] Settings file  : C:\Users\lyp\.node-red\settings.js
8 Oct 12:58:49 - [info] Context store  : 'default' [module=memory]
8 Oct 12:58:49 - [info] User directory : \Users\lyp\.node-red
8 Oct 12:58:49 - [warn] Projects disabled : editorTheme.projects.enabled=false
8 Oct 12:58:49 - [info] Flows file     : \Users\lyp\.node-red\flows.json
8 Oct 12:58:49 - [info] Creating new flow file
8 Oct 12:58:49 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
```

Exercise 5-1



"hello" → SHA256 → "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824"


Deploy
timestamp
function 1
debug 1
Delete
Cancel
Done
Properties
Name
function 1
Setup
On Start
On Message
On Stop
var cryptojs = context.global.cryptojs;
var Hash = cryptojs.SHA256("hello");
msg.payload = Hash;
return msg;
"hello"




```
var cryptojs = context.global.cryptojs;
var Hash = cryptojs.SHA256("hello");
msg.payload = Hash;
return msg;
```

10/8/2022, 1:15:36 PM node: debug 1

msg.payload : Object

```
▾object
  ▾words: array[8]
      0: 754077114
      1: 1605411598
      2: 652753706
      3: -977673570
      4: 454434396
      5: 531055198
      6: 1929655138
      7: -1819568092
  sigBytes: 32
```

JavaScript Object toString()

Definition and Usage

The `toString()` method returns an object as a string.

The `toString()` method returns "[object Object]" if it cannot return a string.

`Object.toString()` always returns the object constructor.

The `toString()` method does not change the original object.

Note

Every JavaScript object has a `toString()` method.

The `toString()` method is used internally by JavaScript when an object needs to be displayed as a text (like in HTML), or when an object needs to be used as a string.

Normally, you will not use it in your own code.

https://www.w3schools.com/jsref/jsref_object_tostring.asp

Try it yourself

Examples

Using toString() on an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let text = fruits.toString();
```

Try it Yourself »

https://www.w3schools.com/jsref/jsref_object_tostring.asp

Try it yourself

Run

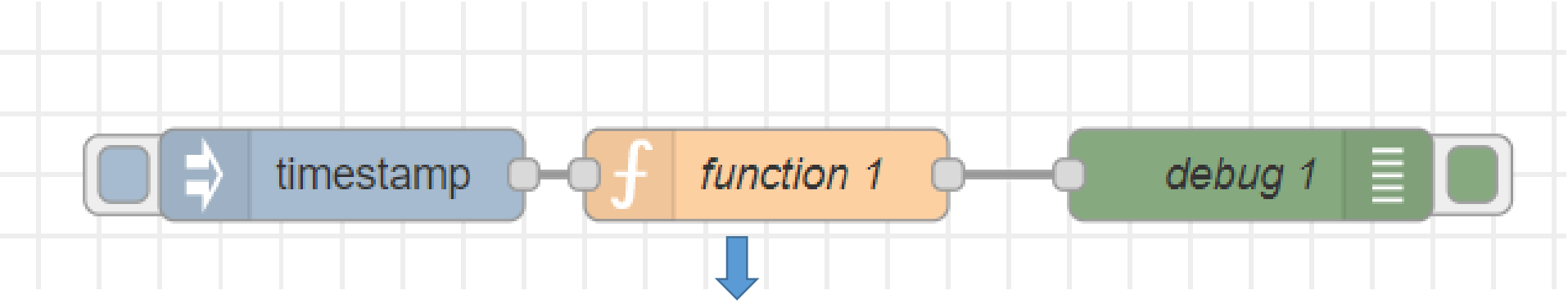
```html
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Objects</h1>
<h2>The tostring() Method</h2>

<p>The result of using toString() on an array:</p>
<p id="demo"></p>

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
const myArray = fruits.toString();

document.getElementById("demo").innerHTML = myArray;
</script>

</body>
</html>
```

JavaScript Objects

The tostring() Method

The result of using toString() on an array:

Banana,Orange,Apple,Mango



Setup | On Start | **On Message** | On Stop

```
1  var cryptojs = context.global.cryptojs;
2  var Hash = cryptojs.SHA256("hello");
3  msg.payload = Hash.toString();
4  return msg;
```

Hash

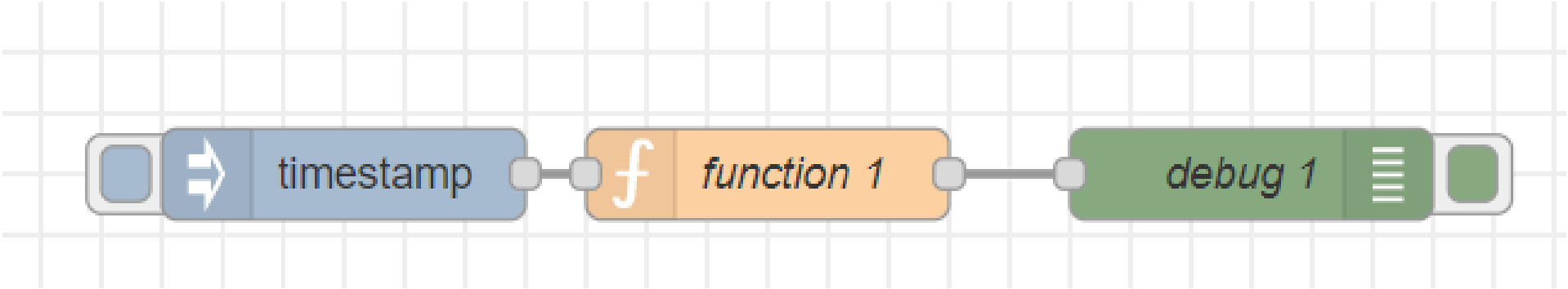
```
msg.payload : Object
▾ object
  ▾ words: array[8]
      0: 754077114
      1: 1605411598
      2: 652753706
      3: -977673570
      4: 454434396
      5: 531055198
      6: 1929655138
      7: -1819568092
  sigBytes: 32
```

```
Hash.toString();
```

10/8/2022, 1:17:36 PM node: debug 1

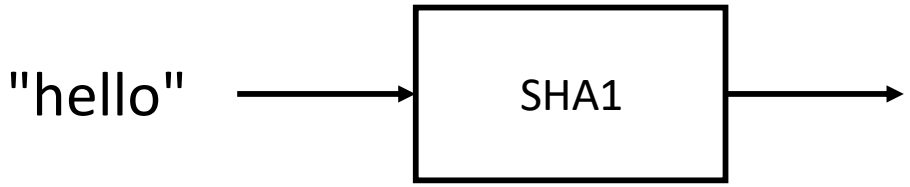
msg.payload : string[64]

"2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824"



- Try SHA1

```
var Hash = cryptojs.SHA1("hello");
```

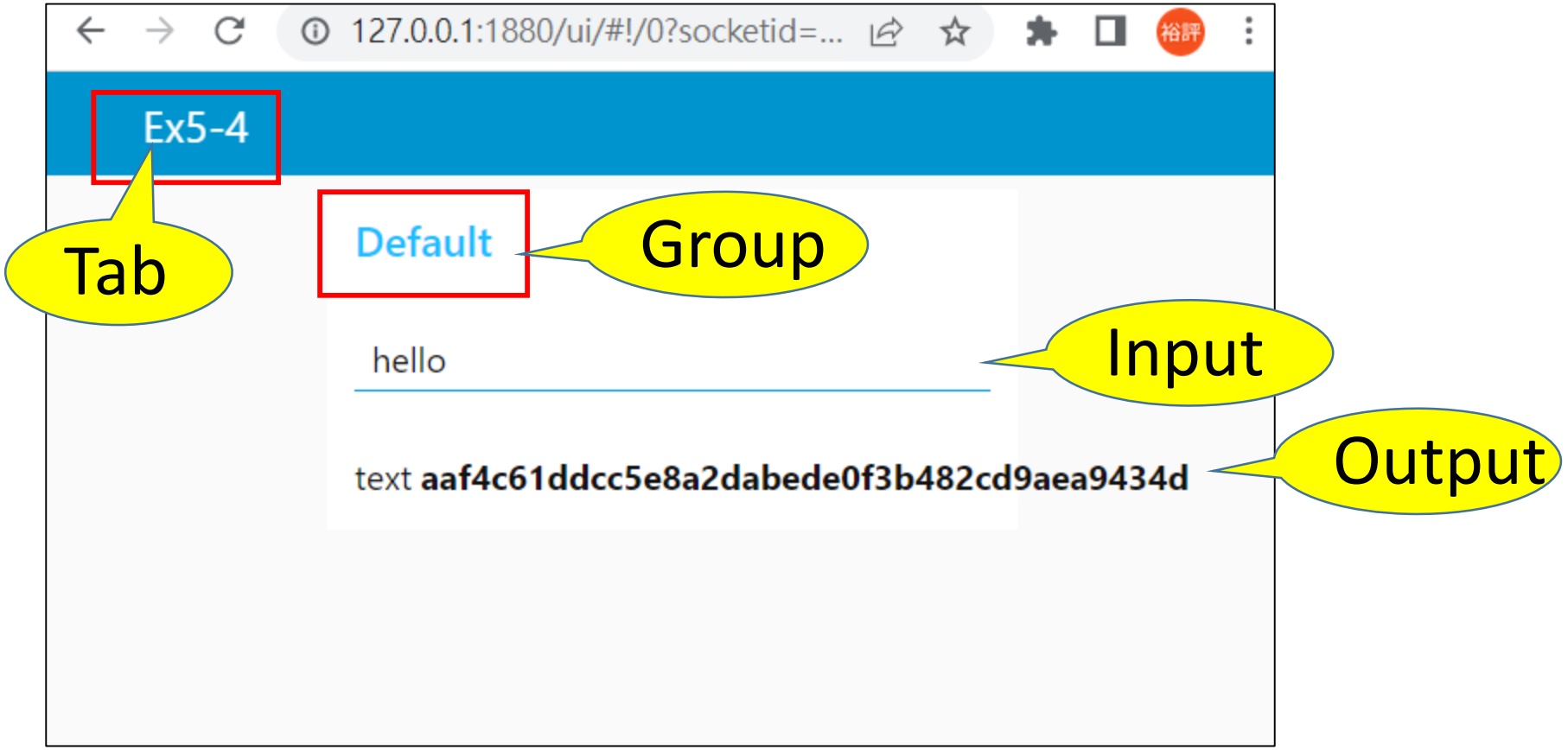


10/8/2022, 3:04:51 PM node: debug 1

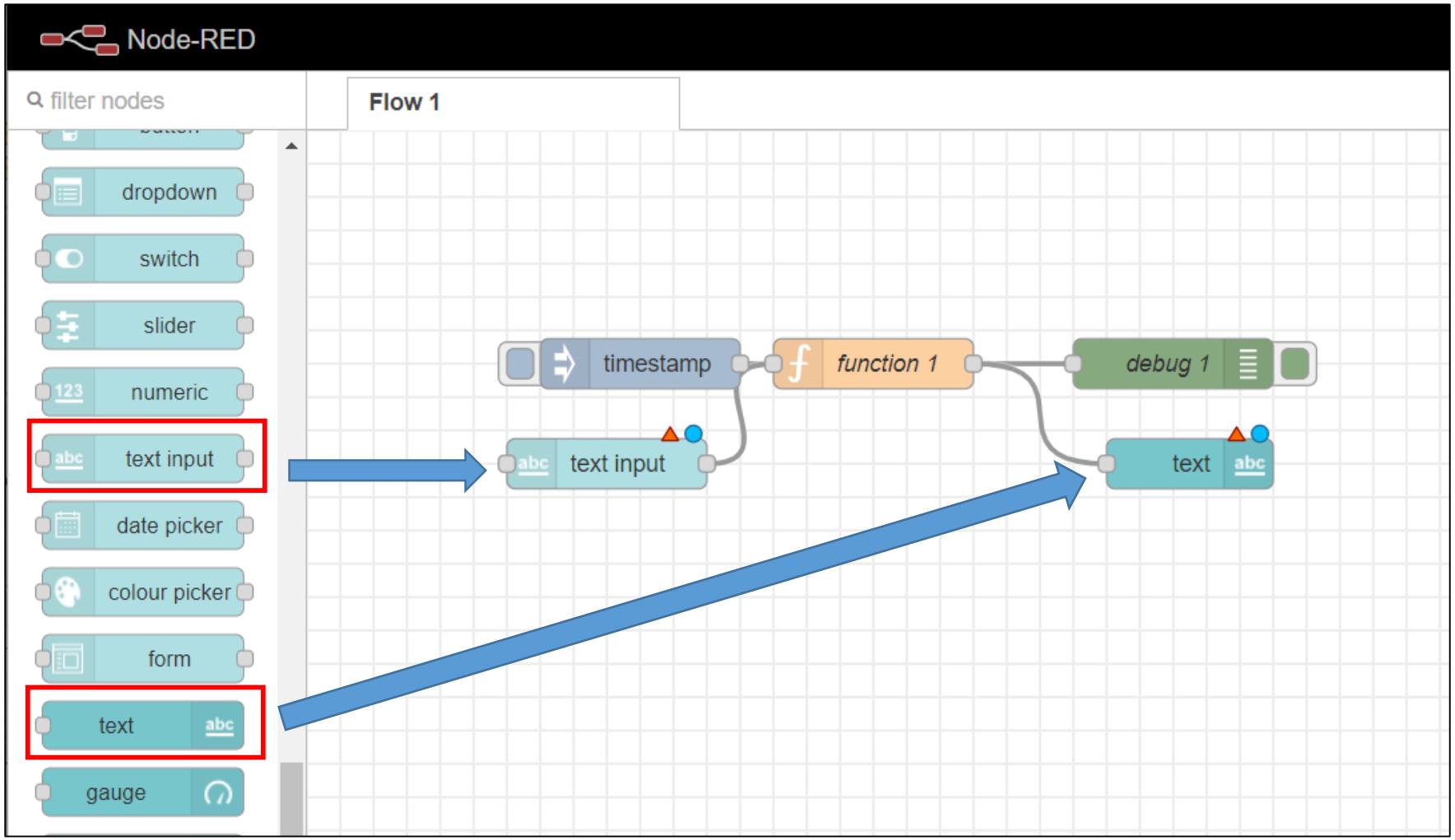
msg.payload : string[40]

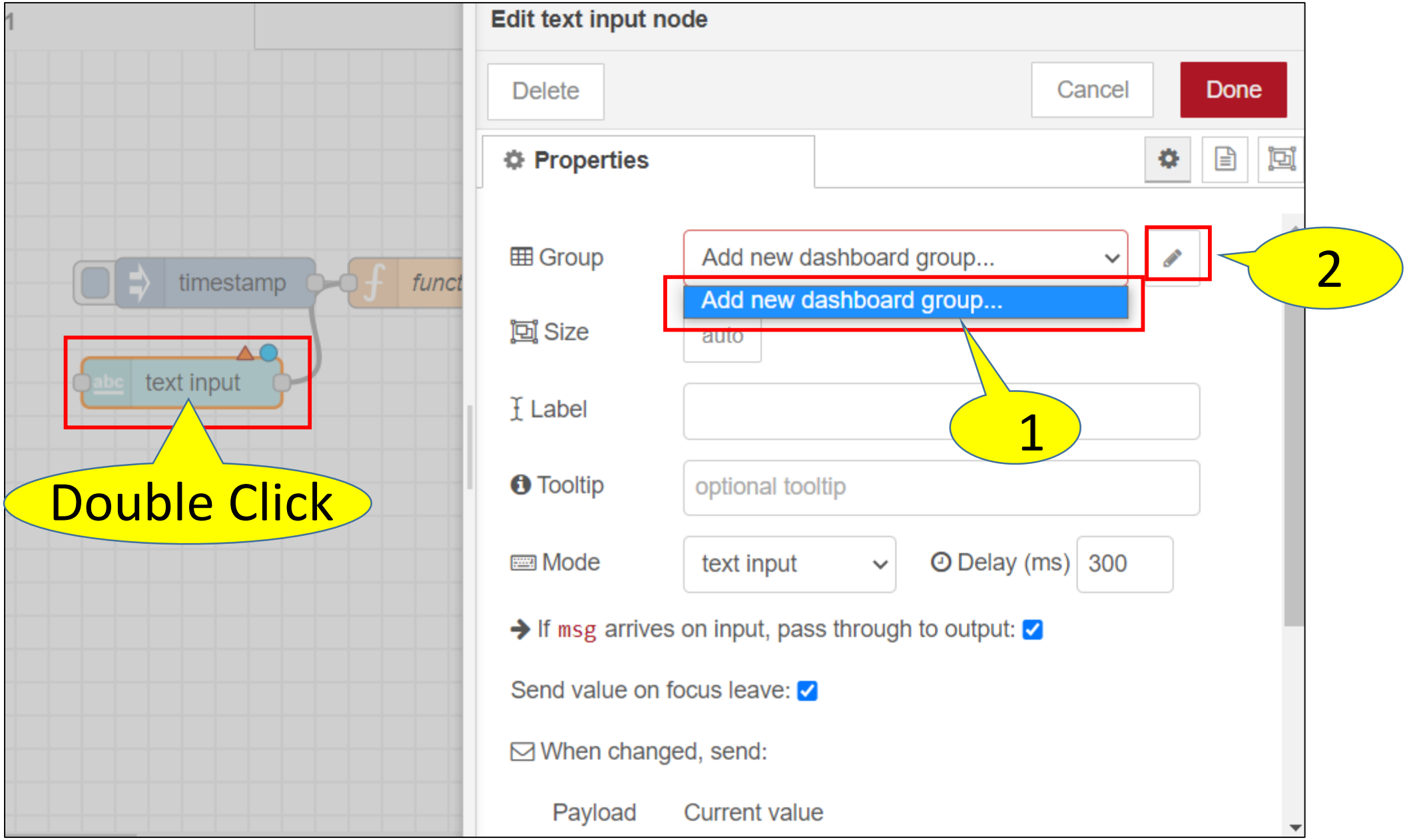
```
"aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"
```

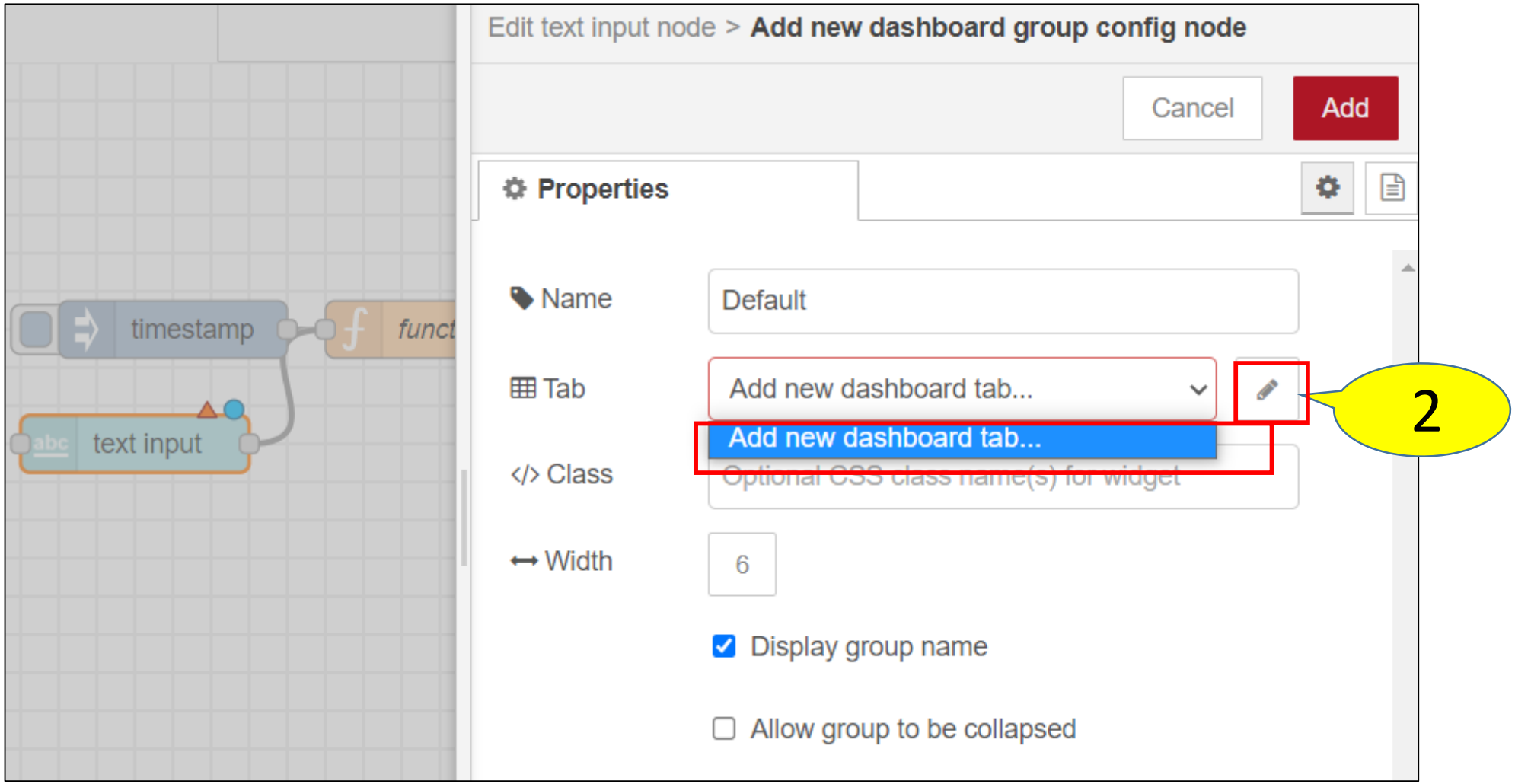
Exercise 5-4


127.0.0.1:1880/ui/#!/0?socketid=...
Ex5-4
Tab
Default
Group
hello
Input
text aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
Output

Add “text Input” and “text”




Edit text input node
Delete
Cancel
Done
Properties
timestamp
funct
text input
Double Click
Group
Add new dashboard group...
Add new dashboard group...
2
Size
auto
Label
1
Tooltip
optional tooltip
Mode
text input
Delay (ms)
300
If msg arrives on input, pass through to output:
Send value on focus leave:
When changed, send:
Payload
Current value


Edit text input node > Add new dashboard group config node
Cancel
Add
Properties
timestamp
text input
Name
Default
Tab
Add new dashboard tab...
Add new dashboard tab...
Class
Optional CSS class name(s) for widget
Width
6
Display group name
Allow group to be collapsed
2

Edit text input node > Add new dashboard group config node > **Add new dashboard tab config node**

Cancel

Add

Add

Properties

Name: Ex5-4

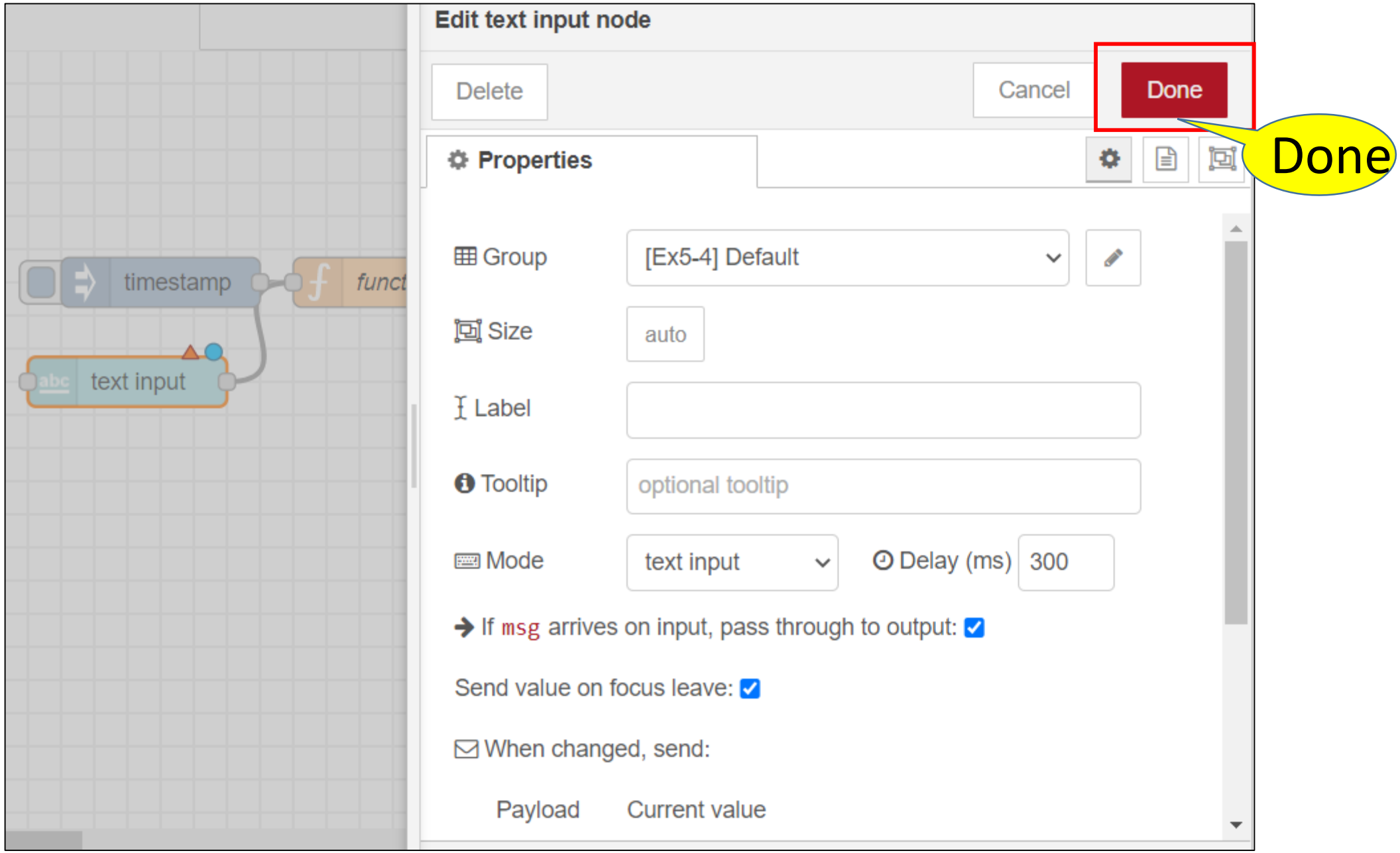
Icon: dashboard

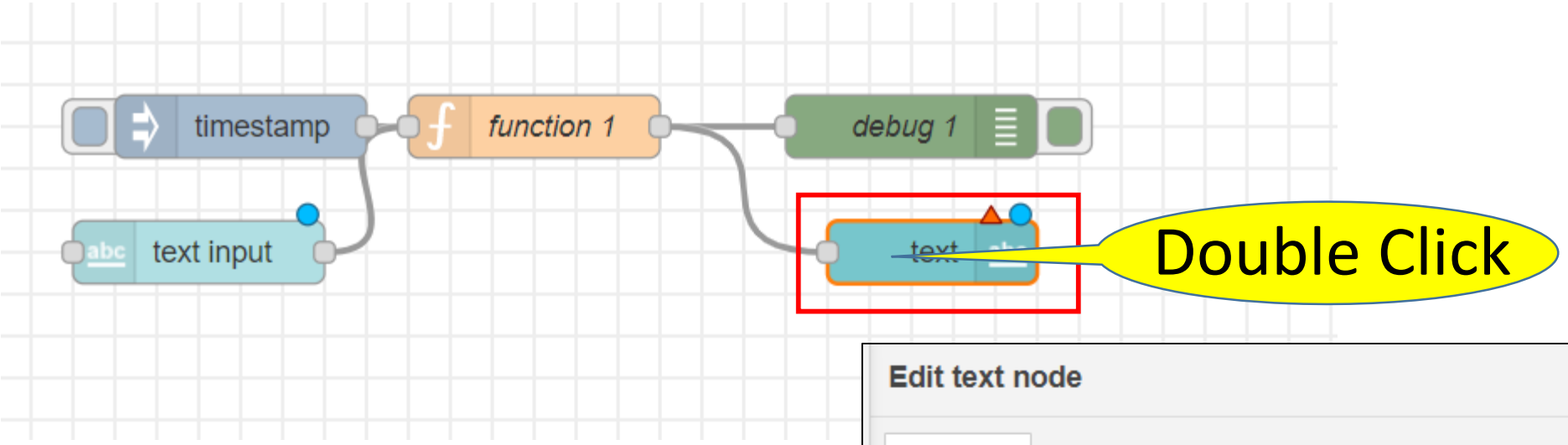
State: Enabled

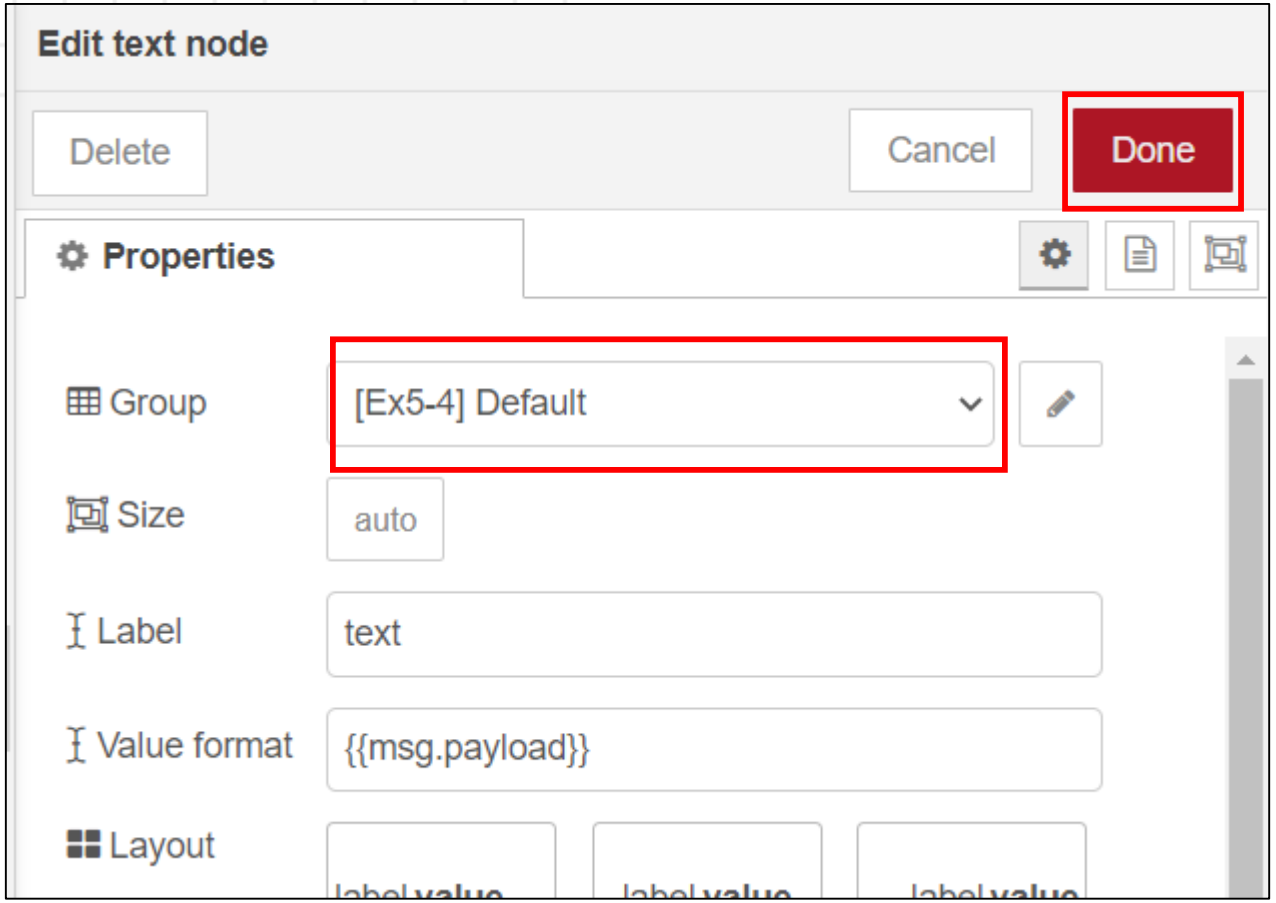
Nav. Menu: Visible

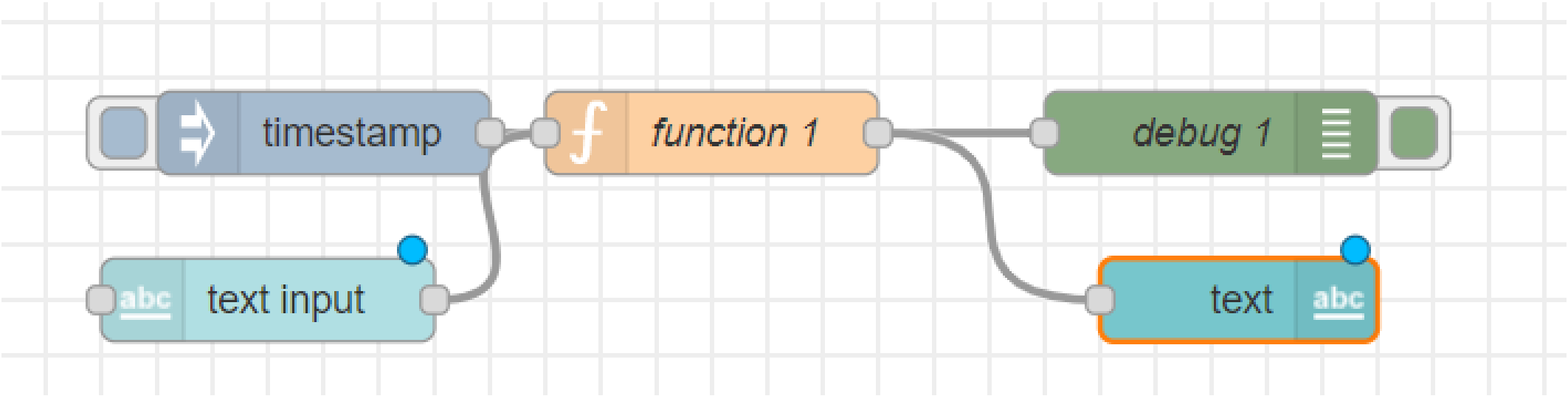
The **Icon** field can be either a Material Design icon *(e.g. 'check', 'close')* or a Font Awesome icon *(e.g. 'fa-fire')*, or a Weather icon *(e.g. 'wi-wu-sunny')*.

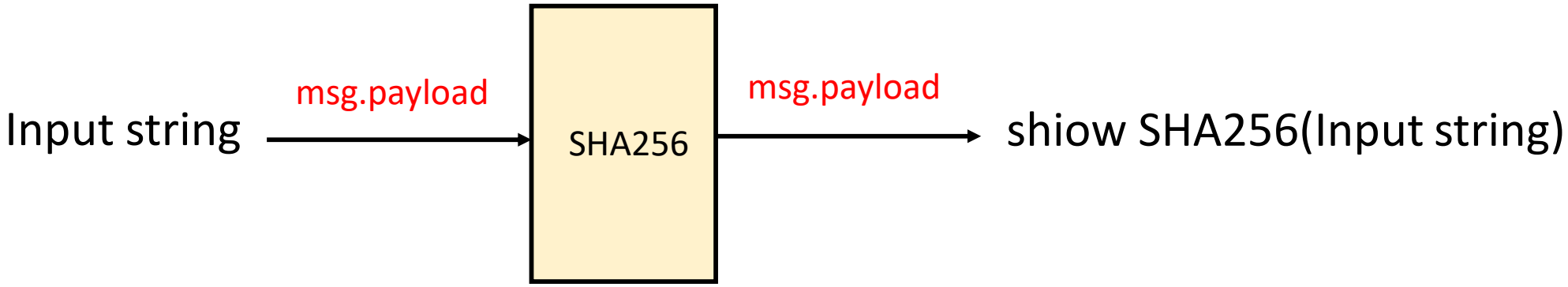
You can use the full set of google material icons if you add 'mi-' to the icon name. e.g. 'mi-videogame_asset'.

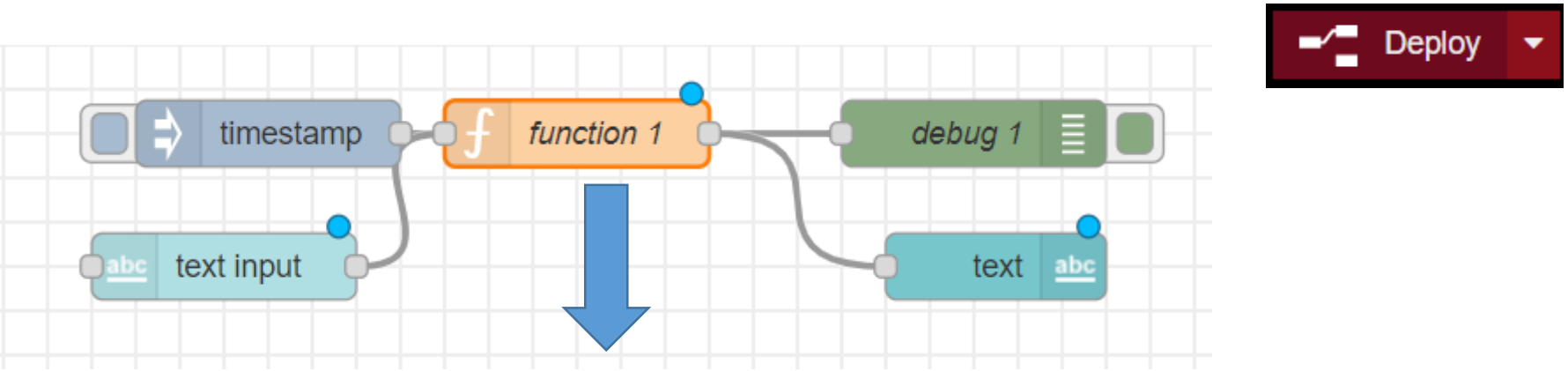

timestamp
funct
text input
Edit text input node
Delete
Cancel
Done
Done
Properties
Group
[Ex5-4] Default
Size
auto
Label
Tooltip
optional tooltip
Mode
text input
Delay (ms)
300
If msg arrives on input, pass through to output:
Send value on focus leave:
When changed, send:
Payload
Current value


timestamp
function 1
debug 1
text input
text
Double Click


Edit text node
Delete
Cancel
Done
Properties
Group
[Ex5-4] Default
Size
auto
Label
text
Value format
{{msg.payload}}
Layout

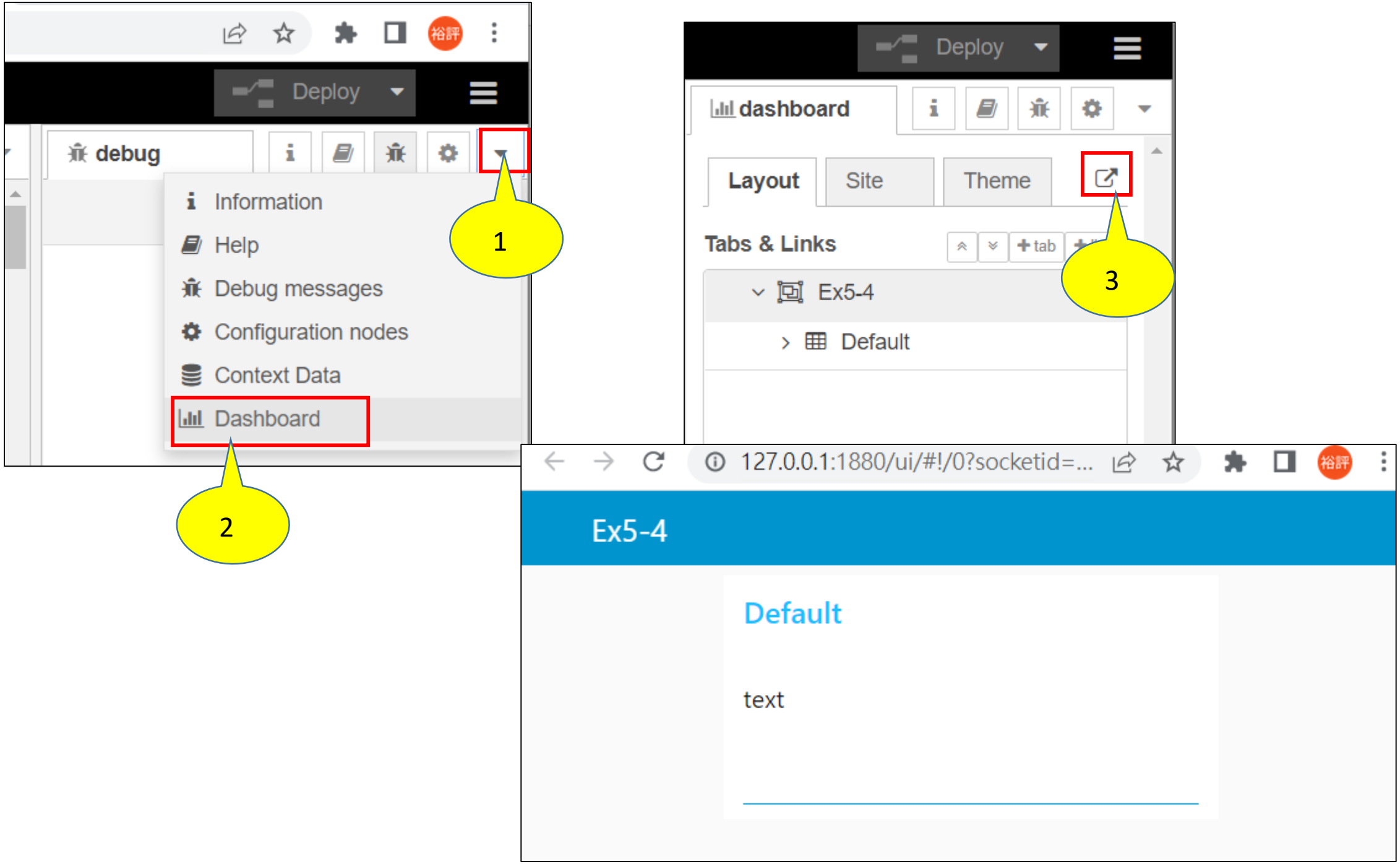

timestamp
function 1
debug 1
text input
text


Input string
msg.payload
SHA256
msg.payload
shiow SHA256(Input string)

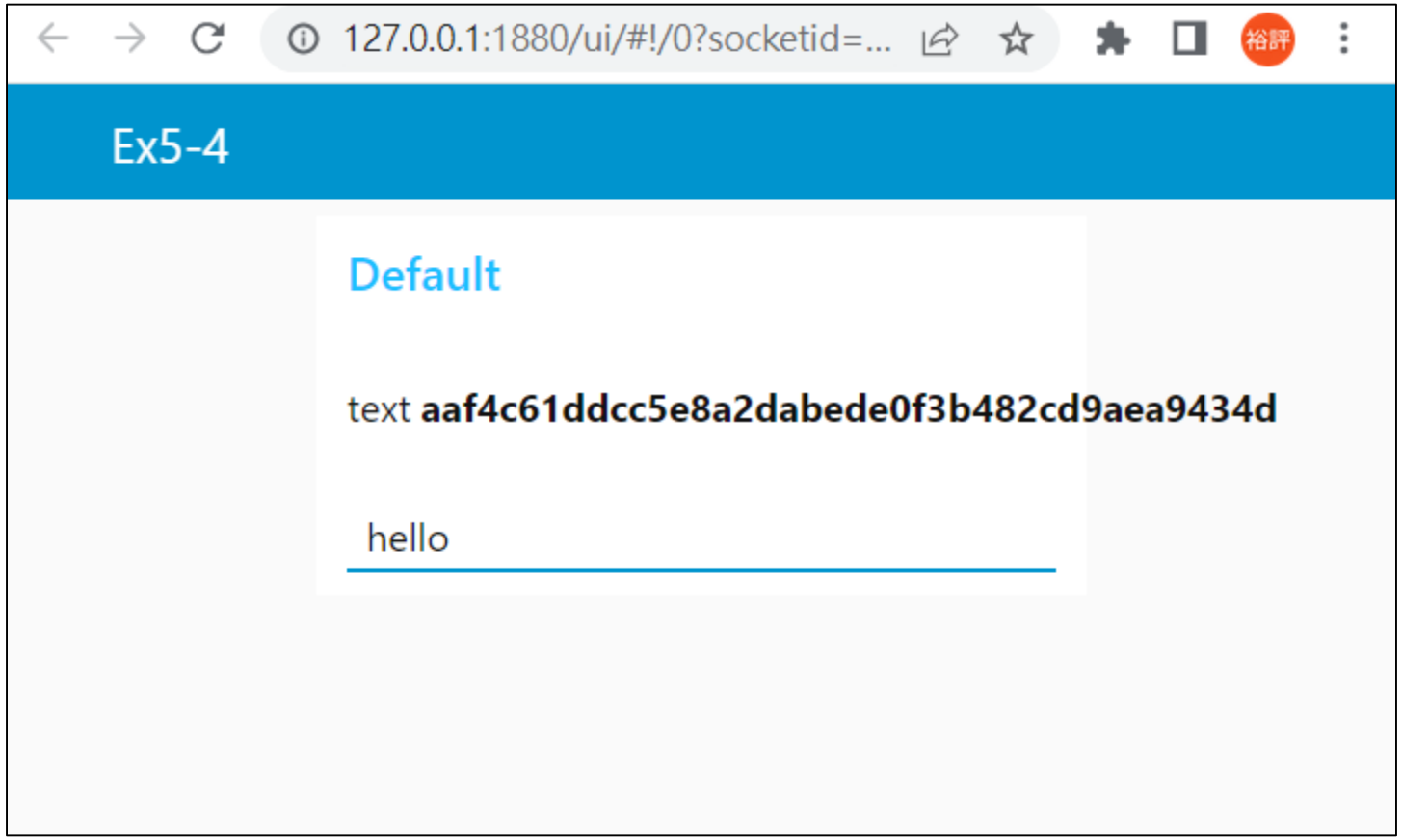


Setup | On Start | **On Message**

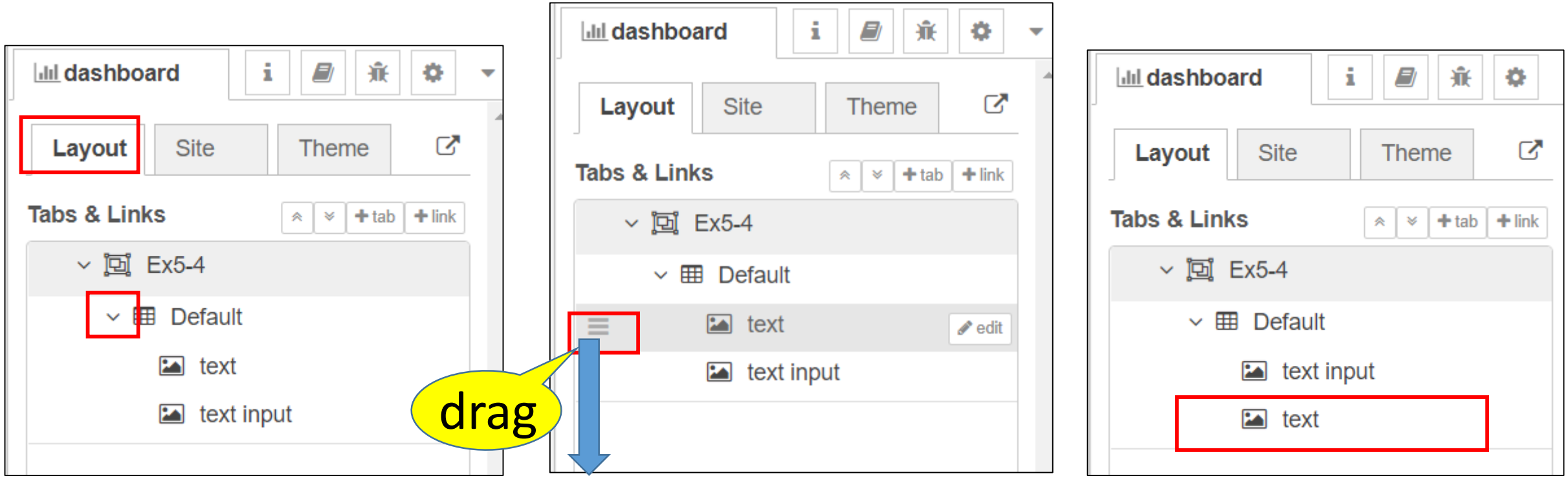
```
var inputtext=msg.payload;
var cryptojs = context.global.cryptojs;
var Hash = cryptojs.SHA1(inputtext);
msg.payload = Hash.toString();
return msg;
```


Deploy
debug
Information
Help
Debug messages
Configuration nodes
Context Data
Dashboard
1
2
Deploy
dashboard
Layout
Site
Theme
Tabs & Links
+ tab
Ex5-4
Default
3
127.0.0.1:1880/ui/#!/0?socketid=...
Ex5-4
Default
text

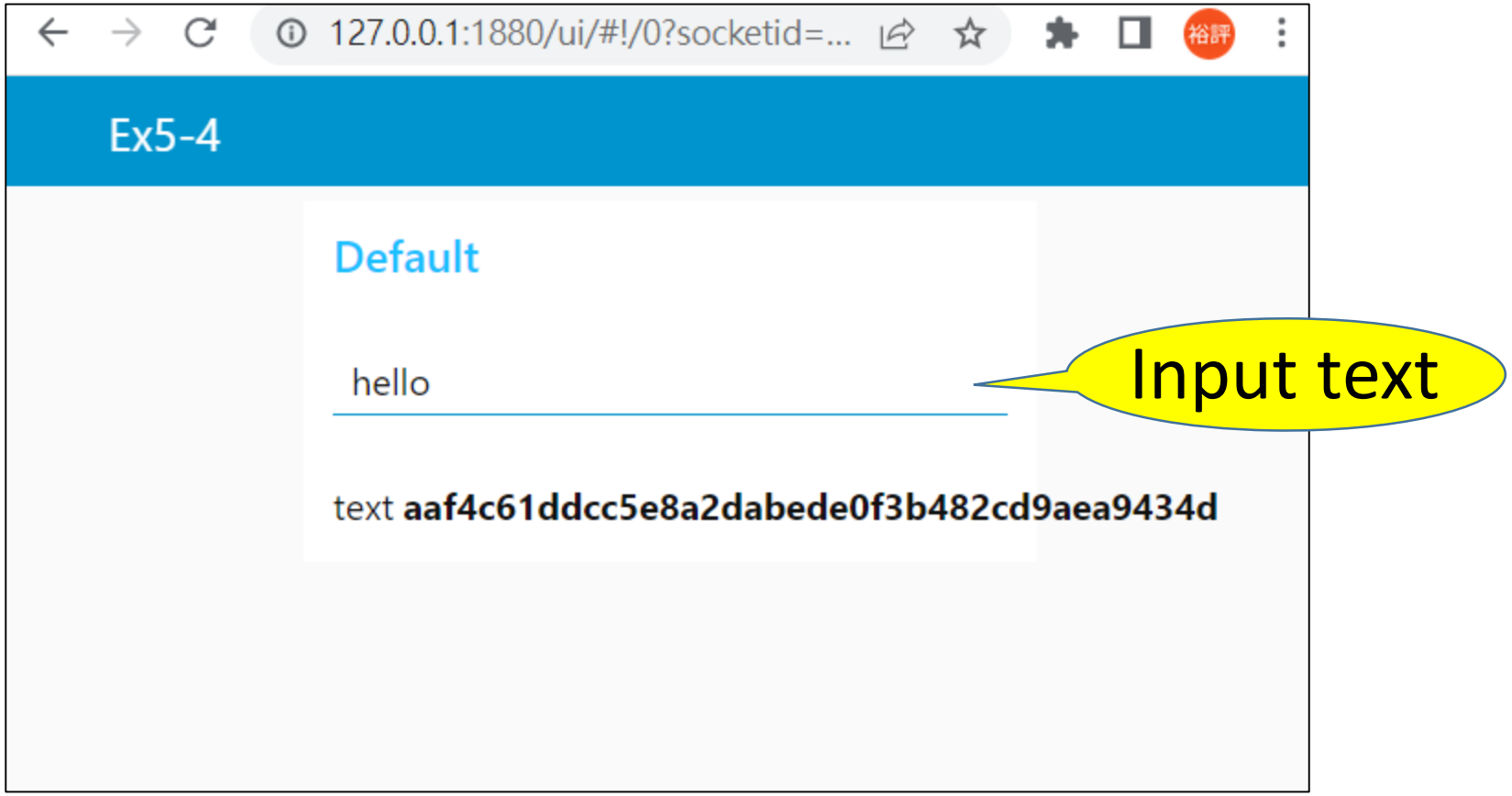
Go to dashboard



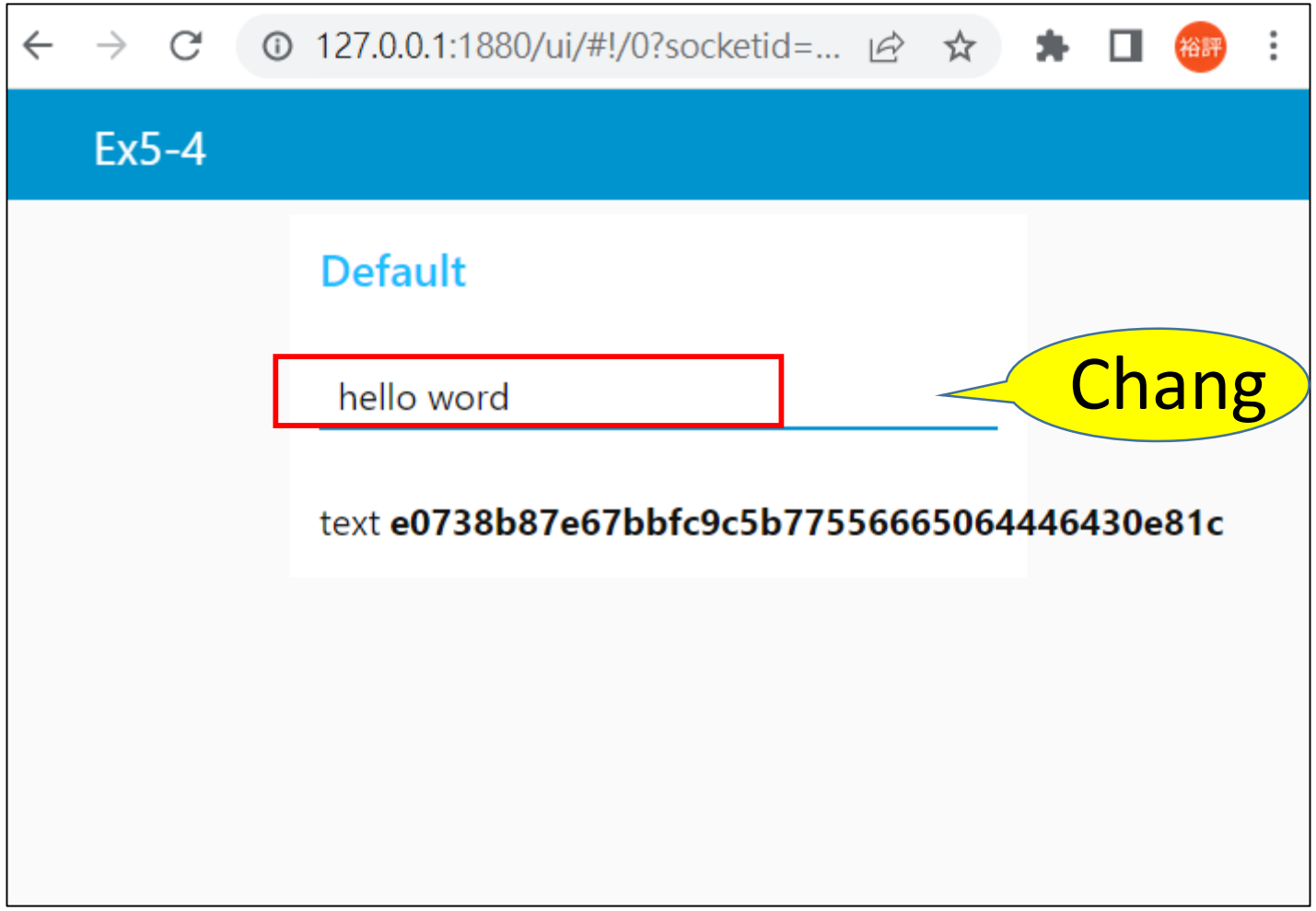
Change Layout



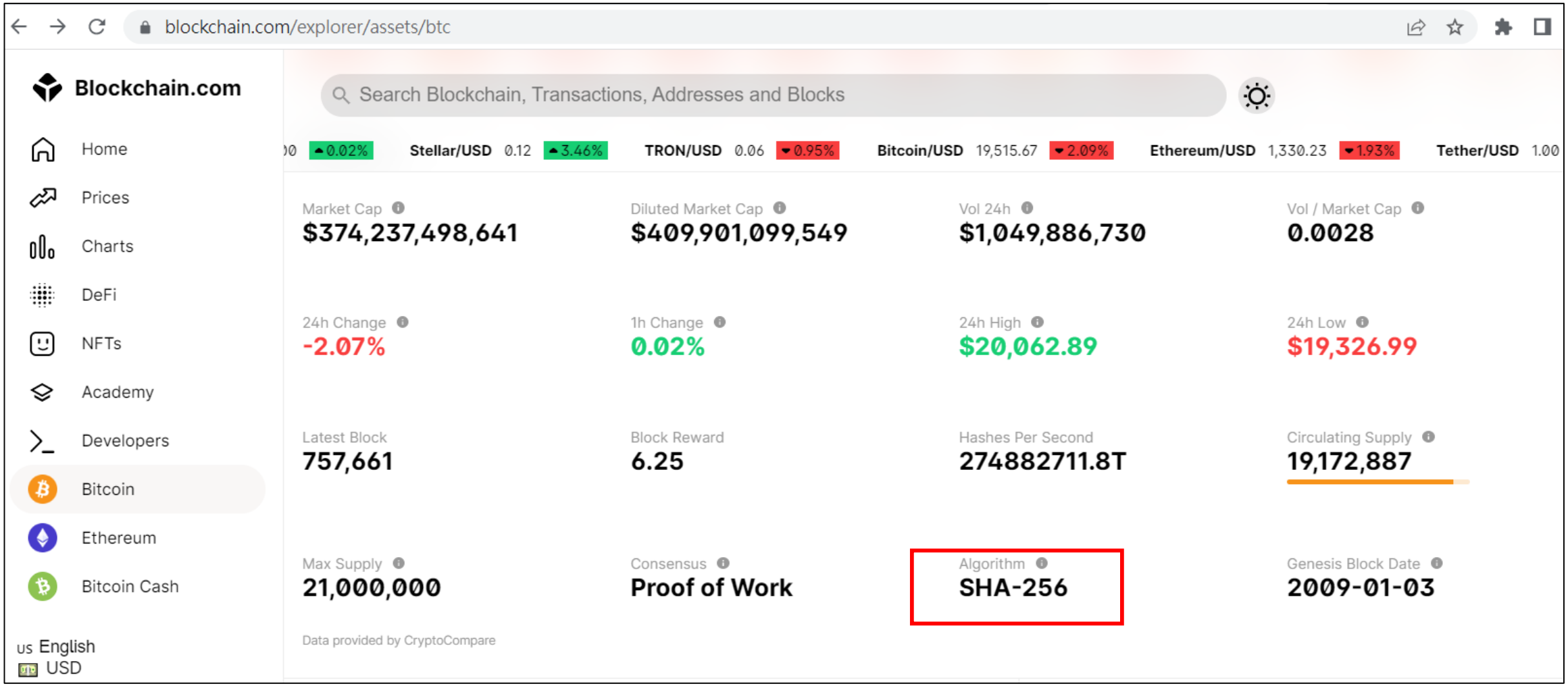
Go to dashboard



Chang the input text



Bitcoin (BTC)



https://www.blockchain.com/explorer/assets/btc

The cryptographic hash function SHA-256

General description

SHA-256 (secure hash algorithm, FIPS 182-2) is a cryptographic hash function with digest length of 256 bits. It is a keyless hash function; that is, an MDC (Manipulation Detection Code).

A message is processed by blocks of $512 = 16 \times 32$ bits, each block requiring 64 rounds.

Basic operations

- Boolean operations AND, XOR and OR, denoted by $\wedge$, $\oplus$ and $\vee$, respectively.
- Bitwise complement, denoted by $^{-}$.
- Integer addition modulo 2^{32}, denoted by $A + B$.

Each of them operates on 32-bit words. For the last operation, binary words are interpreted as integers written in base 2.

- $RotR(A, n)$ denotes the circular right shift of n bits of the binary word A.
- $ShR(A, n)$ denotes the right shift of n bits of the binary word A.
- $A \| B$ denotes the concatenation of the binary words A and B.

Functions and constants

The algorithm uses the functions:

https://bitcoin-info.guide/%E5%85%A5%E9%96%80%E6%8C%87%E5%BC%95/%E6%AF%94%E7%89%B9%E5%B9%A3%E9%81%8B%E4%BD%9C%E5%8E%9F%E7%90%86/SHA256%E8%A9%B3%E8%A7%A3

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://helix.stormhub.org/papers/SHA-256.pdf

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://bitcoin.org/bitcoin.pdf

Bitcoin Block #0

Mined on 1/04/2009, 02:15:05 View all Blocks

This is the Bitcoin genesis block it marks the birth of the Bitcoin network and was mined by the projects mysterious creator 'Satoshi Nakamoto'. Its 50 bitcoin coinbase reward is unspendable as it was omitted from the transaction database so any attempt to spend it would be rejected by the network. Whether this was intentional or not is unknown.

This block was mined on 1/04/2009, 02:15:05 by Satoshi. A total of 0.00 BTC ($0.00) were sent in the block with the average transaction being 0.0000 BTC ($0.00). Satoshi earned a total reward of 50.00 BTC $0.00. The reward consisted of a base reward of 50.00 BTC $0.00 with an additional 0.0000 BTC ($0.00) reward paid as fees of the 1 transactions which were included in the block.

Message Genesis

Details

Hash	00000-ce26f	Size	285
Depth	758,004	Version	0×1
Capacity	0.03%	Merkle Root	4a-3b
Distance	13y 9m 6d 15h 7m 1s	Difficulty	1.00
BTC	0.0000	Nonce	2,083,236,893
Value	$0.00	Bits	486,604,799
Value Today	$0.00	Weight	1,140 WU
Average Value	0.0000000000 BTC	Median Time	Jan 04, 2009, 2:15:05 AM
Median Value	50.00000000 BTC	Minted	50.00 BTC
Input Value	0.00 BTC	Reward	50.00000000 BTC
Output Value	50.00 BTC	Mined on	Jan 04, 2009, 2:15:05 AM

ref: https://blockchain.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

Bitcoin Block #1

Mined on 1/09/2009, 10:54:25 View all Blocks

This block was mined on 1/09/2009, 10:54:25 by Unknown. A total of 0.00 BTC ($0.00) were sent in the block with the average transaction being 0.0000 BTC ($0.00). Unknown earned a total reward of 50.00 BTC $0.00. The reward consisted of a base reward of 50.00 BTC $0.00 with an additional 0.0000 BTC ($0.00) reward paid as fees of the 1 transactions which were included in the block.

Details

Hash	00000-b6048	Size	215
Depth	758,004	Version	0×1
Capacity	0.02%	Merkle Root	0e-98
Distance	13y 9m 6d 15h 9m 1s	Difficulty	1.00
BTC	0.0000	Nonce	2,573,394,689
Value	$0.00	Bits	486,604,799
Value Today	$0.00	Weight	860 WU
Average Value	0.0000000000 BTC	Median Time	Jan 09, 2009, 10:54:25 AM
Median Value	50.00000000 BTC	Minted	50.00 BTC
Input Value	0.00 BTC	Reward	50.00000000 BTC
Output Value	50.00 BTC	Mined on	Jan 09, 2009, 10:54:25 AM
Transactions	1	Height	1
Witness Tx's	0	Confirmations	758,004
Inputs	1	Miner	Unknown

ref: https://blockchain.info/block/00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048)

Transaction View information about a bitcoin transaction

c4d90161f98da765b6baa1b079b2780fa586c25fcc7d67699a8d16ca9a161764

3Muzarg4bqatDiPt8QZuXP2vn5evVHi6Phs (0.0209609 BTC - Output)	→	1GTG33x1JfVrXX7RnW1gtvMUXnRxWj18qJ - (Unspent)	0.00951251 BTC
		3DXLCv4U2spXBWgDjybbamHHfgLcZ76bwi - (Spent)	0.01093079 BTC

9 Confirmations | 0.0204433 BTC

Summary	
Size	249 (bytes)
Weight	669
Received Time	2018-04-04 12:52:49
Lock Time	Block: 516592
Included In Blocks	516593 (2018-04-04 12:57:38 + 5 minutes)
Confirmations	9 Confirmations
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	0.0209609 BTC
Total Output	0.0204433 BTC
Fees	0.0005376 BTC
Fee per byte	215.904 sat/B
Fee per weight unit	80.359 sat/WU
Estimated BTC Transacted	0.00951251 BTC
Scripts	Hide scripts & coinbase

https://bitcoin-info.guide/%E5%85%A5%E9%96%80%E6%8C%87%E5%BC%95/%E6%AF%94%E7%89%B9%E5%B9%A3%E9%81%8B%E4%BD%9C%E5%8E%9F%E7%90%86/%E4%BA%A4%E6%98%93%E8%B2%BB%E7%94%A8%E8%A9%B3%E8%A7%A3

	礦池算力	全網算力	幣價	日理論收益	演算法
BTC	43.09 EH/s	300.01 EH/s	$ 19042.60	$ 0.0661 /T	SHA256d
LTC	60.04 TH/s	471.21 TH/s	$ 51.93	$ 2.35 /G	Scrypt
ETC	28.70 TH/s	153.48 TH/s	$ 24.10	$ 0.0026 /M	Etchash
ETHW	10.84 TH/s	44.50 TH/s	$ 7.81	$ 0.0025 /M	Ethash
ETHF	3930.85 GH/s	4992.09 GH/s	$ 0.9432	$ 0.0025 /M	Ethash
BCH	52.04 PH/s	2198.70 PH/s	$ 110.35	$ 0.0642 /T	SHA256d
CKB	39.21 PH/s	69.74 PH/s	$ 0.0039	$ 0.0006 /G	Eaglesong
RVN	427.34 GH/s	13.24 TH/s	$ 0.0324	$ 0.0082 /M	KawPow

https://www.f2pool.com/?lang=zh_TW

以太坊智慧合約程式碼缺陷致Qubit DeFi平台被竊取8000萬美元加密貨幣

https://www.techbang.com/posts/93909-ethereum-smart-contract-code-flaws-caused-the-qubit-defi

Qubit Finance platform hacked for $80 million worth of cryptocurrency

黑客從 Qubit DeFi 平台吸走了 8000 萬美元的加密貨幣

發表於 **2022年1月29日 星期六 下午 4:21:43**

去中心化金融通常被認為是防黑客的。 然而，黑客是一群智能人，他們設計了一些方法來攻破看似無敵的加密貨幣堡壘。 在黑客總結的最新黑客攻擊中，以太坊橋中使用的智能合約代碼中的一個缺陷已被利用。

最新的受害者是去中心化平台 Qubit Finance。 據 The Verge 報道，Qubit Finance 是高價值盜竊案的最新受害者，黑客周四竊取了約 8000 萬美元的加密貨幣。 據說這起盜竊案是 2022 年迄今為止最大的盜竊案。

Qubit Finance 已經承認了這次黑客攻擊。

Qubit Finance 已經在通過 Medium 發布的一份報告中承認了這一黑客行為。 該報告詳細介紹了這次襲擊，稱襲擊發生在美國東部時間 1 月 27 日晚上 5 點左右。

https://0xzx.com/zh-tw/2022012916212043440.html

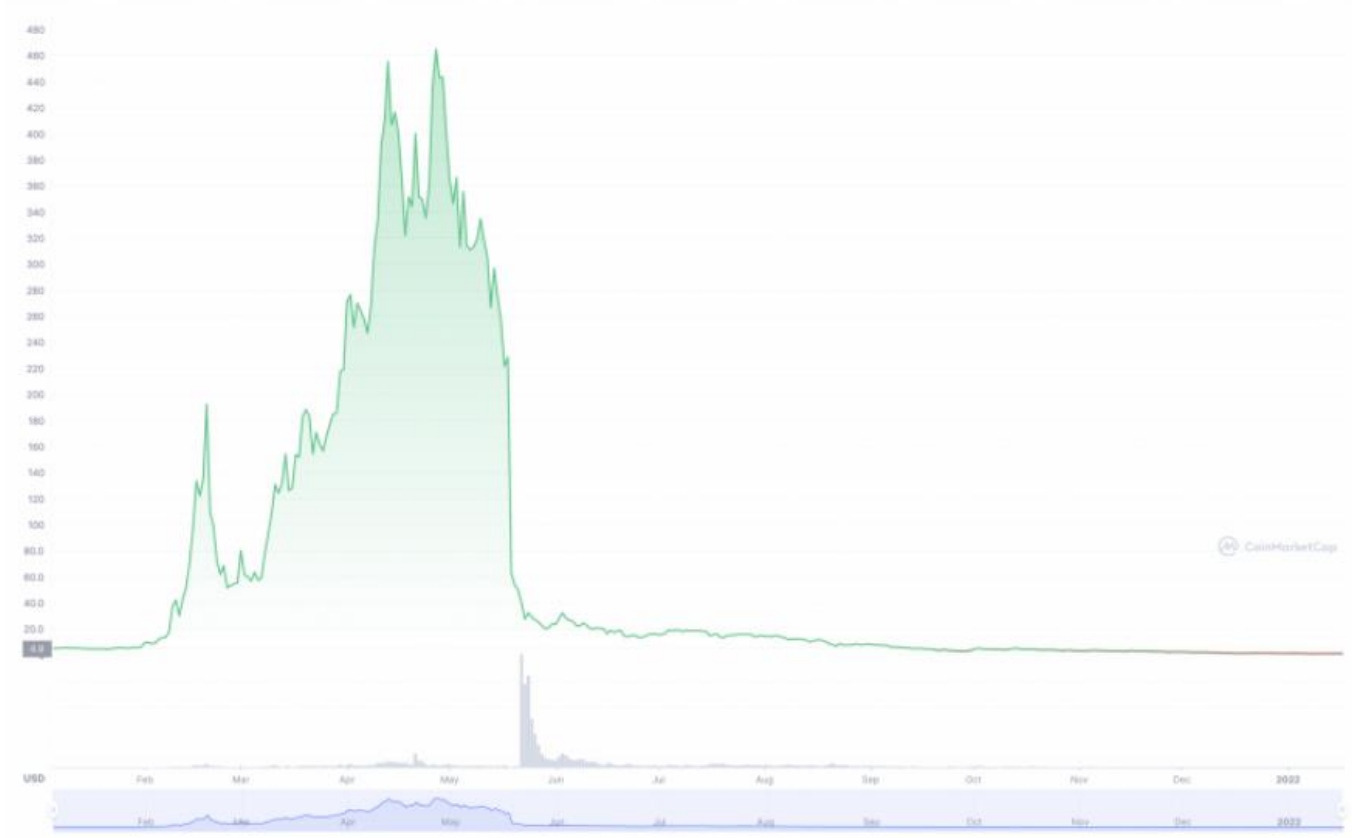
https://therecord.media/qubit-finance-platform-hacked-for-80-million-worth-of-cryptocurrency/

Qubit to USD chart

– Source : Coinmarketcap –

https://www.blocktempo.com/bsc-protocol-qubit-finance-got-flash-loan-attack-for-80-million-dollars/

BUNNY加密幣的初始迄今K線圖



BUNNY加密幣的初始迄今K線圖。（圖：CoinMarketCap）

https://www.storm.mg/lifestyle/4160597?page=2

加密貨幣再傳重大竊案！駭客入侵跨鏈橋BSC Token Hub，損失約32億

← **Thread**

CZ 🔶 **Binance** ✅ ···
@cz_binance

An exploit on a cross-chain bridge, BSC Token Hub, resulted in extra BNB. We have asked all validators to temporarily suspend BSC. The issue is contained now. Your funds are safe. We apologize for the inconvenience and will provide further updates accordingly.

7:51 AM · Oct 7, 2022 · Twitter Web App

3,787 Retweets **847** Quote Tweets **12.5K** Likes

https://www.storm.mg/lifestyle/4554962?itm_source_s=storm.mg&itm_medium_s=dable
https://protos.com/explained-how-600m-was-stolen-from-binances-bnb-chain/

一封「假錄取信pdf」害6億美元遭駭！駭客騙區塊鏈遊戲Axie Infinity工程師得逞

by James — 2022-07-07 in 區塊鏈商業應用, 即時新聞, 犯罪 AA

https://www.blocktempo.com/a-fake-job-offer-took-down-axie-infinity/

https://www.theblock.co/post/156038/how-a-fake-job-offer-took-down-the-worlds-most-popular-crypto-game

訂房資料外洩 百人遭假客服騙400萬

70

胡欣男 / 台北報導

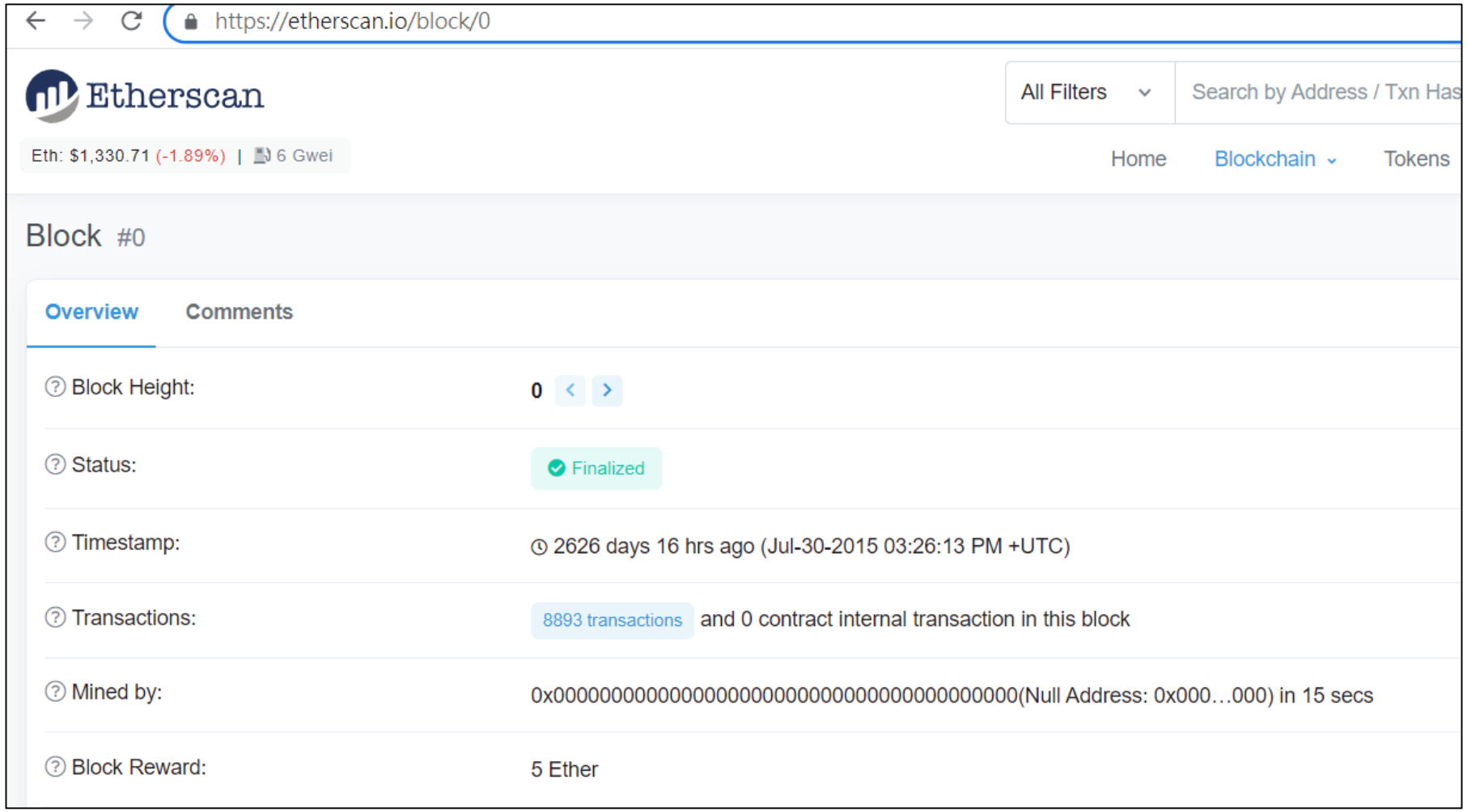
2022年10月11日 週二 上午4:10

13日開始邊境將解封，近月來，國內防疫已逐漸鬆綁，國人開始「報復性出遊」，不料疑似訂房網站個資外洩，9月迄今警政署165反詐騙諮詢專線，已接獲逾百起民眾訂房後遭「解除分期付款」手法詐騙，損失400多萬。刑事局表示，銀行客服不可能打電話請客戶操作ATM或網銀解除分期付款，民眾要提高警覺。

台中張姓男子9月訂了某飯店，沒多久就接獲自稱飯店客服的來電稱，「因請款信用卡資料有誤，造成重複扣款，須經銀行確認身分才能更正撤銷。」隨後又有自稱銀行員，致電要求他操作網路銀行查核。

https://tw.news.yahoo.com/%E8%A8%82%E6%88%BF%E8%B3%87%E6%96%99%E5%A4%96%E6%B4%A9-%E7%99%BE%E4%BA%BA%E9%81%AD%E5%81%87%E5%AE%A2%E6%9C%8D%E9%A8%99400%E8%90%AC-201000782.html

ETHEREUM Block #0

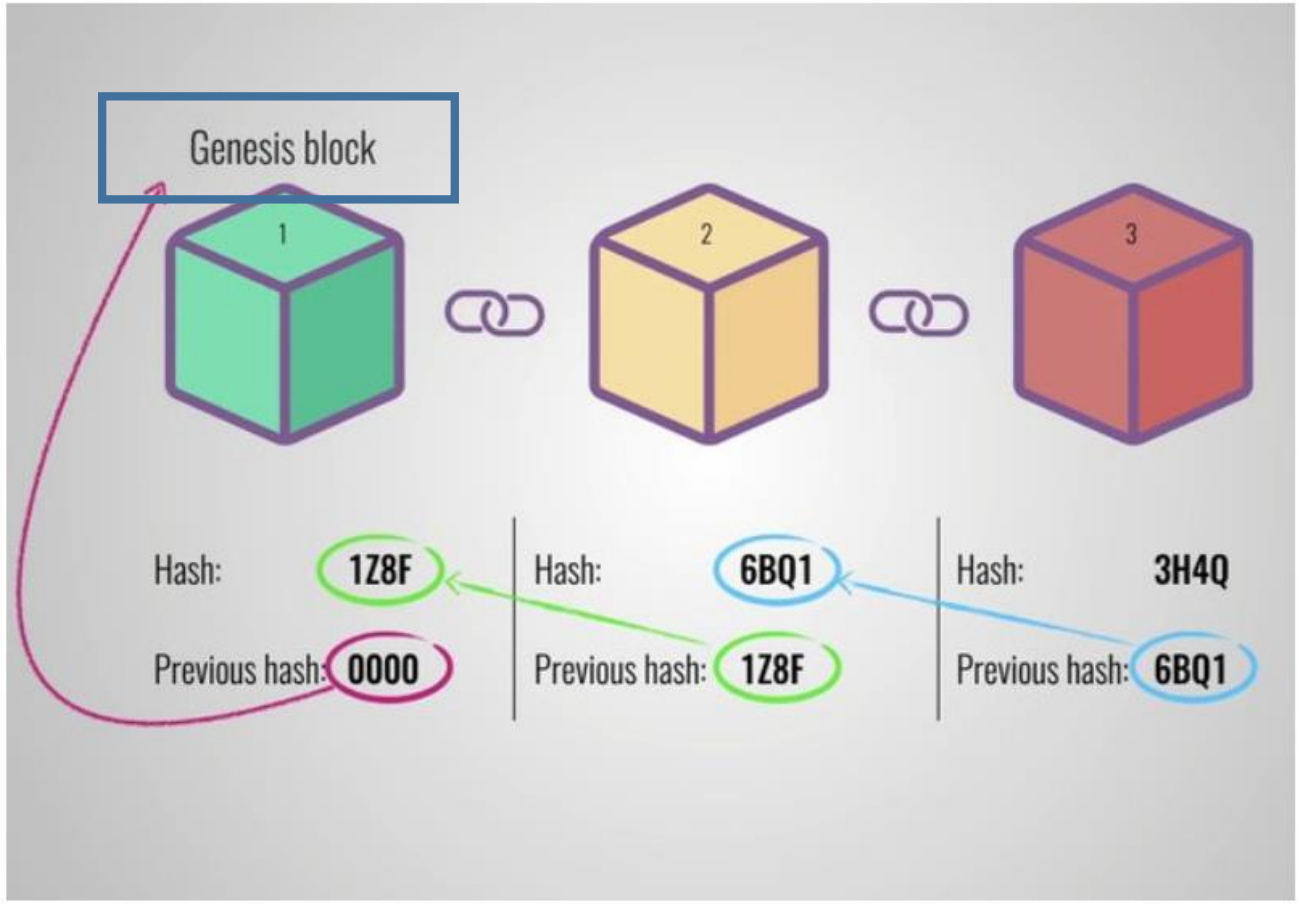


https://etherscan.io/block/0

ETHEREUM Block #0

Extra Data:	◆◆◆N4{N◆◆|◆p◆◆3◆◆◆i◆◆z8◆◆◆◆ (Hex:0x11bbe8db4e347b4e8c937c1
Ether Price:	N/A
Hash:	0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3
Parent Hash:	0x0000000000000000000000000000000000000000000000000000000000000000
Sha3Uncles:	0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
StateRoot:	0xd7f8974fb5ac78d9ac099b9ad5018bedc2ce0a72dad1827a1709da30580f0544
Nonce:	0x0000000000000042

Genesis block



https://steemit.com/hash/@wsmart/block-chain-hash-of-previous-block

Exercise 5-3

- Create a object

index

data

previoushash

time

Hash

msg.payload : Object

```
▾ object
  index: 0
  data: 0
  previoushash: "0000000000000"
  time: "Tue Oct 11 2022 12:59:59:37"
  Hash: "0e3e2e35e01efe3382b8773f803a1b8f9aac46e000abaed4075bb4e0821741cb"
```

```
Hash =
SHA256(index+ data +previoushash + time);
```

JavaScript Date Objects

Creating Date Objects

Date objects are created with the `new Date()` constructor.

There are **4 ways** to create a new date object:

```
new Date()
new Date(year, month, day, hours, minutes, seconds, milliseconds)
new Date(milliseconds)
new Date(date string)
```

https://www.w3schools.com/js/js_date_methods.asp

JavaScript Get Date Methods

Method	Description
getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getDay()	Get the **weekday** as a number (0-6)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the **time** (milliseconds since January 1, 1970)

https://www.w3schools.com/js/js_date_methods.asp

Try it yourself

The getMilliseconds() Method

The `getMilliseconds()` method returns the milliseconds of a date as a number (0-999):

Example

```
const d = new Date();
d.getMilliseconds();
```

Try it Yourself »

https://www.w3schools.com/js/js_date_methods.asp

Try it yourself

Run ›

Result Size: 498 x 569

Get your own website

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getSeconds()</h2>
<p>The getSeconds() method returns the seconds of a date
as a number (0-59):</p>

<p id="demo"></p>

<script>
const d = new Date();
document.getElementById("demo").innerHTML =
d.getSeconds();
</script>

</body>
</html>
```

JavaScript getSeconds()

The getSeconds() method returns the seconds of a date as a number (0-59):

29

https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getSeconds

Run

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getSeconds()</h2>
<p>The getSeconds() method returns the seconds of a date
as a number (0-59):</p>

<p id="demo"></p>

<script>
const d = new Date();
document.getElementById("demo").innerHTML =
d.toDateString() + " " + d.getHours() + ":" +
d.getMinutes() + ":" + d.getMinutes() + ":" +
d.getSeconds();
</script>

</body>
</html>
```

Try it yourself

Run ›

Result Size: 498 x 569

Get your own website

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getSeconds()</h2>
<p>The getSeconds() method returns the seconds of a date
as a number (0-59):</p>

<p id="demo"></p>

<script>
const d = new Date();
document.getElementById("demo").innerHTML =
d.toDateString() + " " + d.getHours() + ":" +
d.getMinutes() + ":" + d.getMinutes() + ":" +
d.getSeconds();
</script>

</body>
</html>
```

JavaScript getSeconds()

The getSeconds() method returns the seconds of a date as a number (0-59):

Tue Oct 11 2022 13:8:8:23

Run

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getSeconds()</h2>
<p>The getSeconds() method returns the seconds of a date
as a number (0-59):</p>

<p id="demo"></p>

<script>
const d = new Date();
document.getElementById("demo").innerHTML =d.getTime();
</script>

</body>
</html>
```

The getTime() Method

The `getTime()` method returns the number of milliseconds since January 1, 1970:

Example

```
const d = new Date();
d.getTime();
```

Try it Yourself »

d.getTime()

Run ›

Result Size: 596 x 567

Get your c

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getTime()</h2>
<p>The internal clock in JavaScript counts from midnight January 1,
1970.</p>
<p>The getTime() function returns the number of milliseconds since
then:</p>

<p id="demo"></p>

<script>
const d = new Date();
document.getElementById("demo").innerHTML = d.getTime();
</script>

</body>
</html>
```

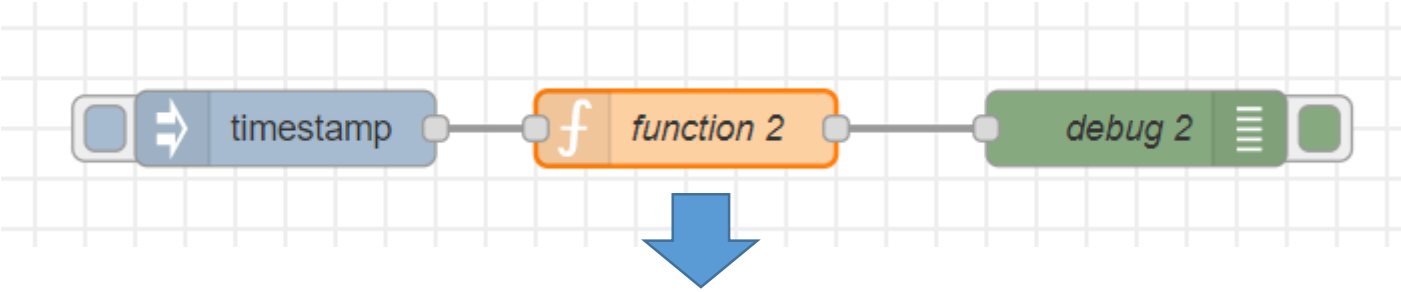
JavaScript getTime()

The internal clock in JavaScript counts from midnight January 1, 1970.

The getTime() function returns the number of milliseconds since then:

1665465291616

https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_gettime

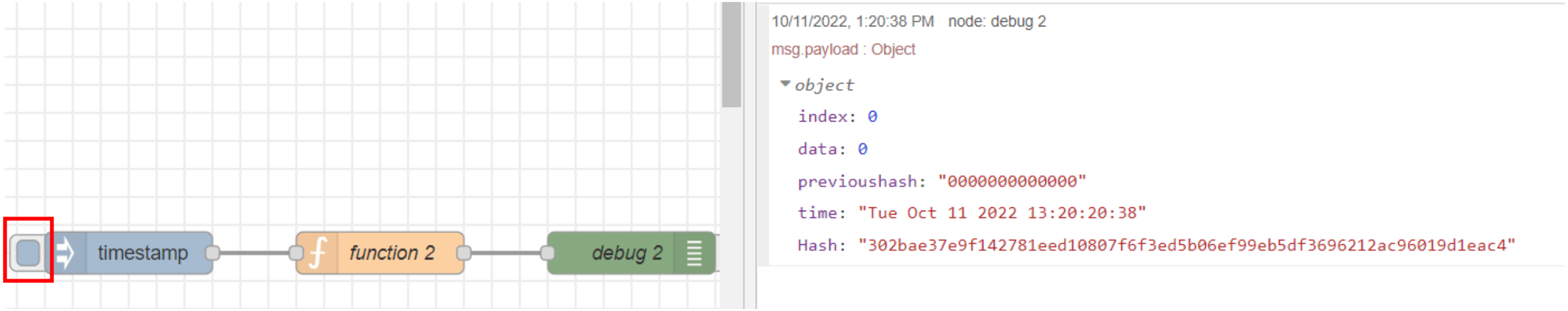


Name: function 2

Setup | On Start | **On Message** | On Stop

```
var cryptojs = context.global.cryptojs;
let data = 0;
let previoushash = "0000000000000";
const d = new Date();
var timestamp = d.getTime();
var time = d.toDateString() + " " + d.getHours() + ":" + d.getMinutes() + ":" + d.getMinutes() + ":" + d.getSeconds();
var index = 0;

msg.url = "https://xxxxxxx.firebaseio.com/" + "blockchainlyp/" + "0000000000000"+".json";


var Hash = cryptojs.SHA256(previoushash + index + data + time);

msg.payload = { "index": index, "data": data, "previoushash": previoushash, "time": time, "Hash": Hash.toString() };

return msg;
```

Triger the flow



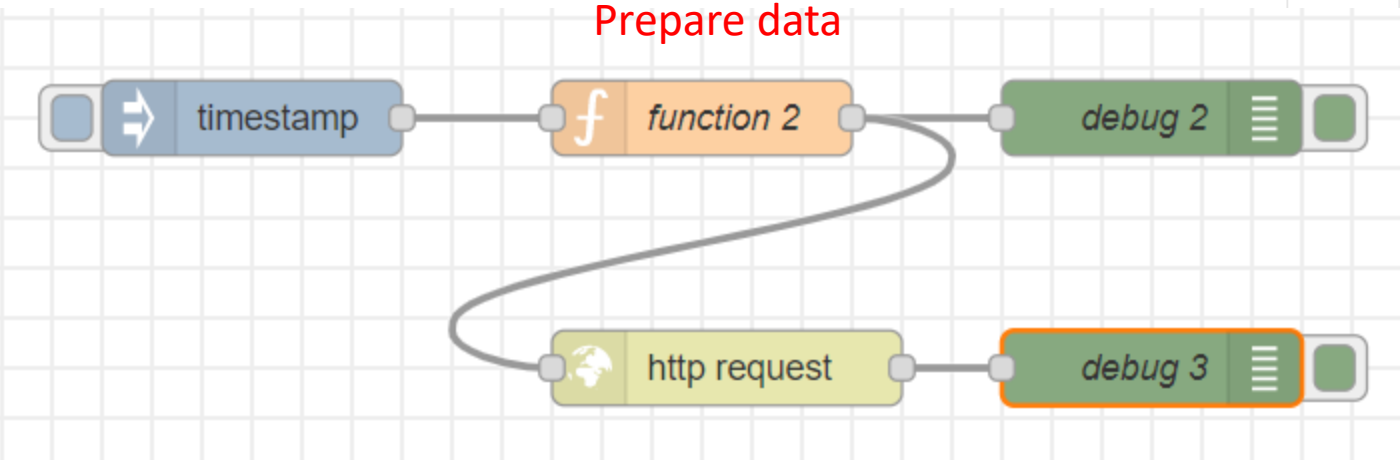
Exercise 5-4

- Create a genesis block

Firebase

```
blockchainlyp
  0000000000000
    Hash: "8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"
    data: 0
    index: 0
    previoushash: "0000000000000"
    time: "Tue Oct 11 2022 13:25:25:20"
```

Prepare data



HTTP PUT

Exercise 5-5

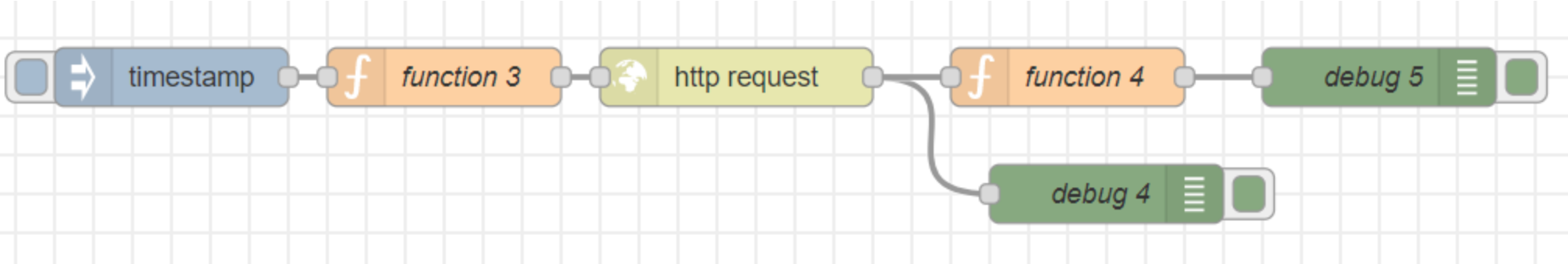
- Get the value of Hash of the last block
-

The Hash of the last block

10/11/2022, 2:19:32 PM node: debug 5

msg.payload : string[64]

"8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"



JavaScript Array Methods

JavaScript Array length

The length property provides an easy way to append a new element to an array:

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";
```

Try it Yourself »

fruits[4]=“Kiwi”;

https://www.panewslab.com/zh_hk/articledetails/D79768833.html

JavaScript Array length

Run ›

Result Size: 596 x 569

Get your own website

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Array Methods</h2>
<p>The length property provides an easy way to append new elements to
an array without using the push() method:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits[fruits.length] = "Kiwi";
document.getElementById("demo2").innerHTML = fruits;
</script>

</body>
</html>
```

JavaScript Array Methods

The length property provides an easy way to append new elements to an array without using the push() method:

Banana,Orange,Apple,Mango

Banana,Orange,Apple,Mango,Kiwi

fruits[0]

fruits[4-1]

fruits[4]

https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_change_add

Setup | On Start | **On Message**

```
msg.url = "https://xxxxxxx.firebaseio.com/" + "blockchainlyp.json";
return msg;
```

Set URL

HTTP GET

Get the Hash of the last block


```
var revstr=msg.payload;
var obj=JSON.parse(revstr);
var revvalues=Object.values(obj);
var len = revvalues.length;
var lastvalue = revvalues[len - 1];
msg.payload = lastvalue.Hash;
return msg;
```

returns an array of a given object's own enumerable property values

Get the length of revalues array

Get the last element of revalues array

Get the value of Hash

Exercise 5-6

- Prepare the data for next block

block #0

10/11/2022, 3:14:33 PM node: debug 4

msg.payload : string[180]

"{"0000000000000":
{"Hash":"8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1","data":
0,"index":0,"previoushash":"0000000000000","time":"Tue Oct 11 2022 13:25:25:20"}}"

10/11/2022, 3:14:33 PM node: debug 5

msg.payload : Object

object

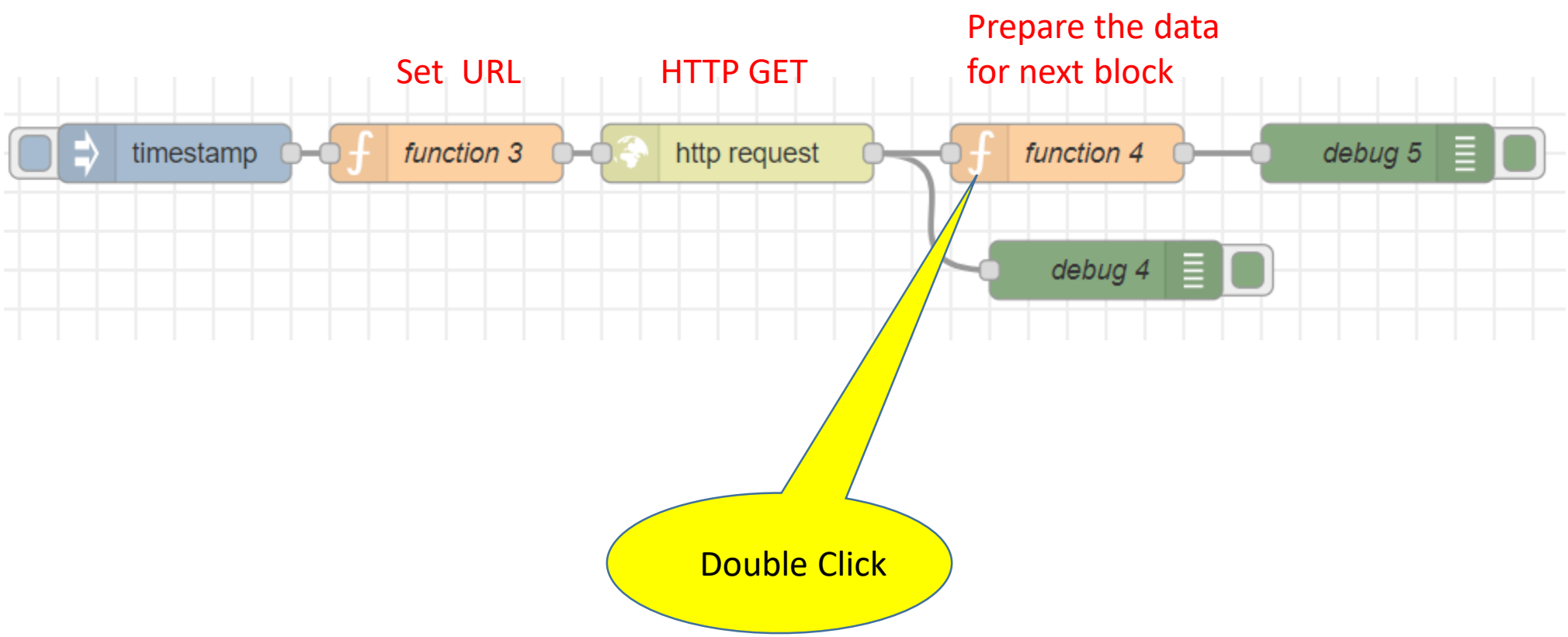
index: 1

data: 28

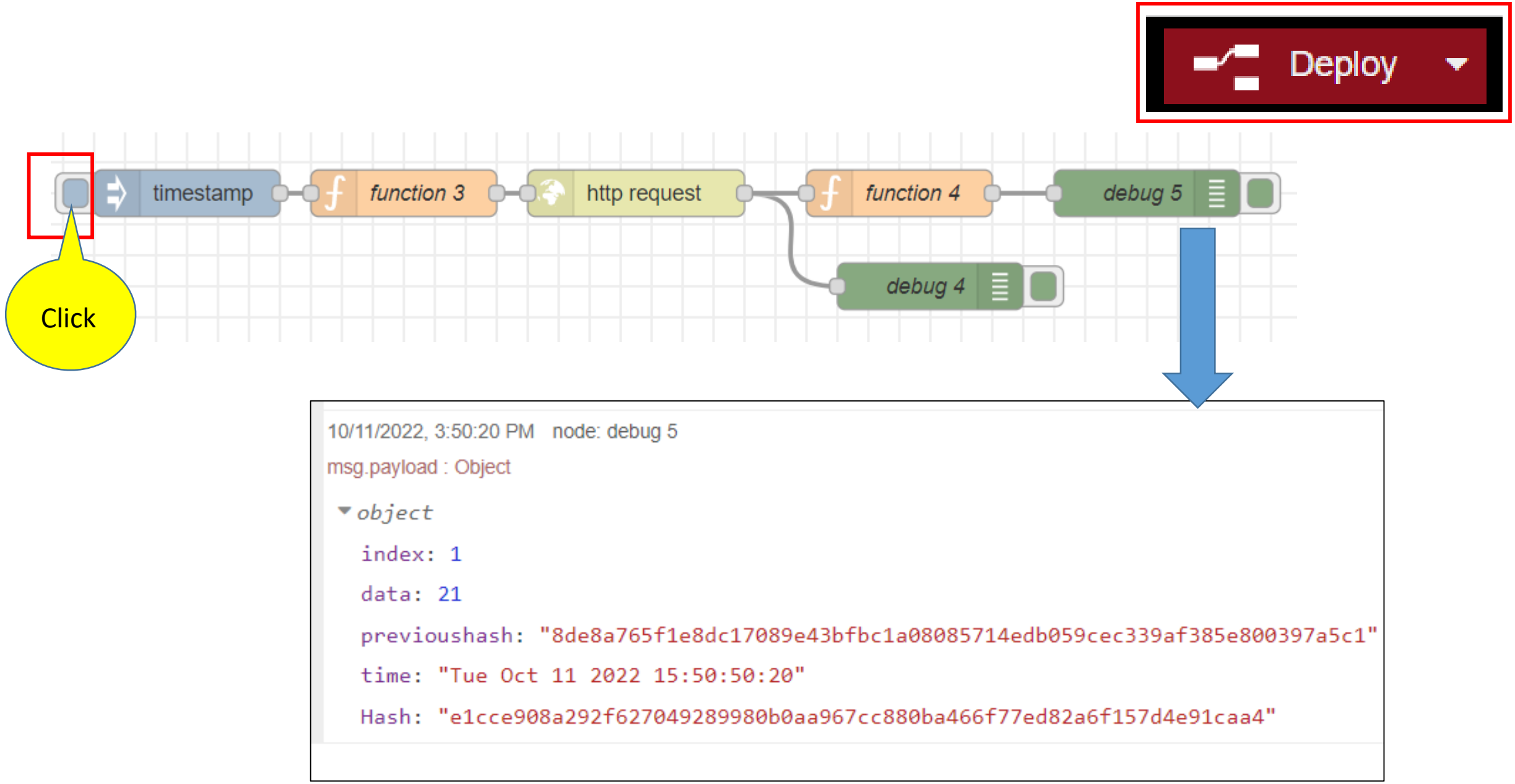
previoushash:
"8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"

time: "Tue Oct 11 2022 15:14:14:33"

Hash:
"7acc700ca17369312dd236fad860960a07d7dc324fca4274bc7eae8f99cc72c7"


Set URL
HTTP GET
Prepare the data
for next block
timestamp
function 3
http request
function 4
debug 5
debug 4
Double Click

```
var revstr=msg.payload;
var obj=JSON.parse(revstr);
var revvalues=Object.values(obj);
var len = revvalues.length;
var lastvalue = revvalues[len - 1];
var previoushash = lastvalue.Hash;

var cryptojs = context.global.cryptojs;
let data = Math.round((Math.random()*100));

const d = new Date();
var timestamp = d.getTime();
var time = d.toDateString() + " " + d.getHours() + ":" + d.getMinutes() + ":" +
d.getMinutes() + ":" + d.getSeconds();
var index = len;

msg.url = "https://xxxxxx.firebaseio.com/" + "blockchainlyp/" + timestamp + ".json";

var Hash = cryptojs.SHA256(previoushash + index + data + time);
msg.payload = { "index": index, "data": data, "previoushash": previoushash, "time": time,
"Hash": Hash.toString() };
return msg;
```


Deploy
timestamp
function 3
http request
function 4
debug 5
debug 4
Click
10/11/2022, 3:50:20 PM node: debug 5
msg.payload : Object
object
index: 1
data: 21
previoushash: "8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"
time: "Tue Oct 11 2022 15:50:50:20"
Hash: "e1cce908a292f627049289980b0aa967cc880ba466f77ed82a6f157d4e91caa4"

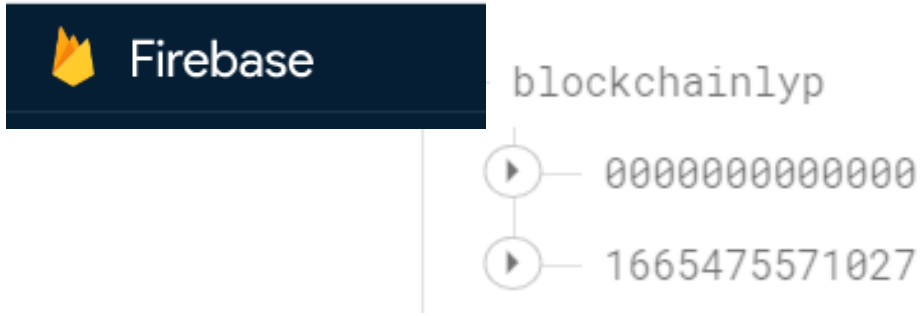
Exercise 5-7

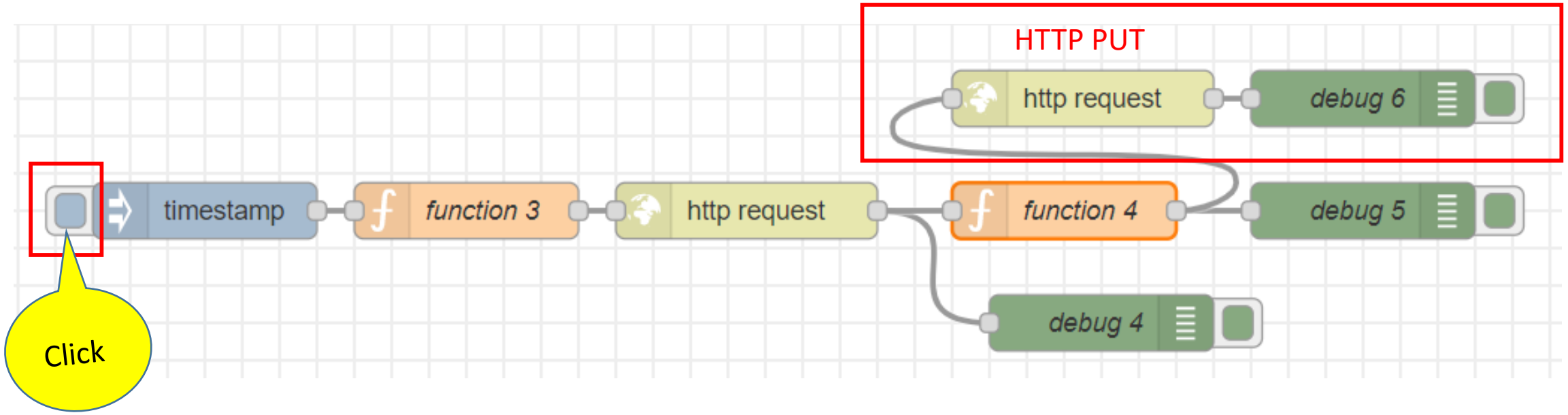
- Write a block to your database.

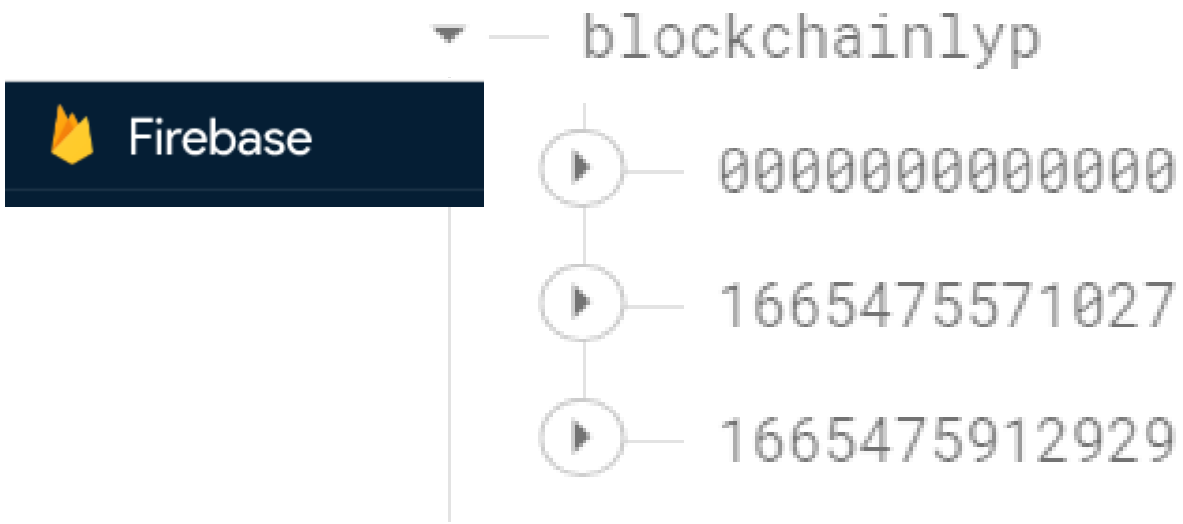

```
blockchainlyp
  0000000000000
    Hash: "8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"
    data: 0
    index: 0
    previoushash: "0000000000000"
    time: "Tue Oct 11 2022 13:25:25:20"
  1665475571027
    Hash: "eaf686411a53dba1f699bd88a08433b77f464fdfac2c9637c34571767d6dc243"
    data: 75
    index: 1
    previoushash: "8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"
    time: "Tue Oct 11 2022 16:6:6:11"
```

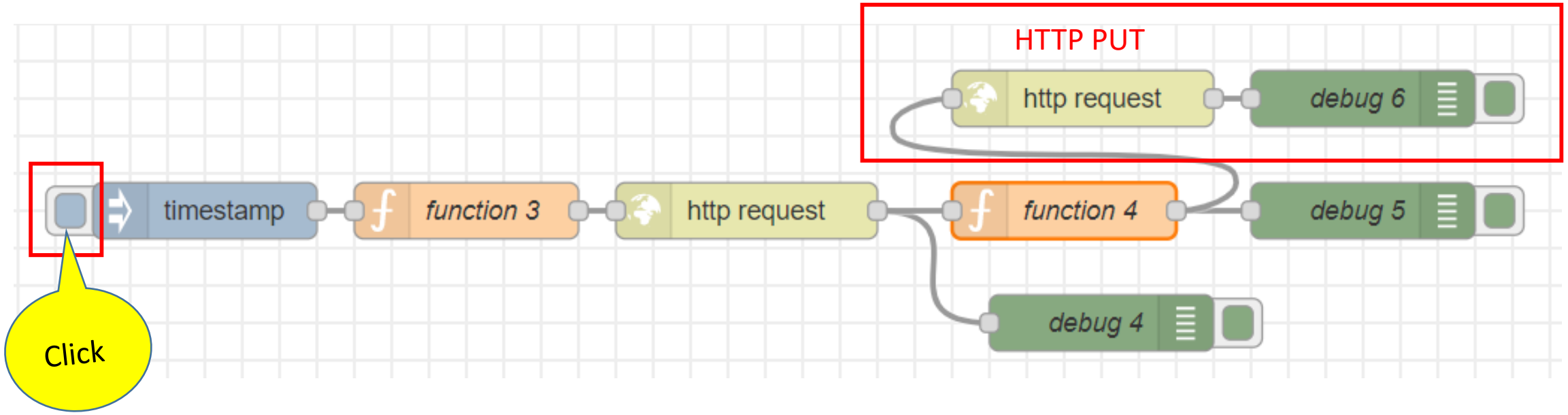
HTTP PUT


Deploy


Firebase
blockchainlyp
0000000000000
1665475571027


HTTP PUT
http request
debug 6
timestamp
function 3
http request
function 4
debug 5
debug 4
Click


blockchainlyp
Firebase
0000000000000
1665475571027
1665475912929


HTTP PUT
http request
debug 6
timestamp
function 3
http request
function 4
debug 5
debug 4
Click

Firebase

```
1665475571027
  Hash: "eaf686411a53dba1f699bd88a08433b77f464fdfac2c9637c34571767d6dc243"
  data: 75
  index: 1
  previoushash: "8de8a765f1e8dc17089e43bfbc1a08085714edb059cec339af385e800397a5c1"
  time: "Tue Oct 11 2022 16:6:6:11"
1665475912929
  Hash: "76c34ed69316c3cb696644a4616943d0ef66f313ec7571bdf57c4edf446b622e"
  data: 47
  index: 2
  previoushash: "eaf686411a53dba1f699bd88a08433b77f464fdfac2c9637c34571767d6dc243"
  time: "Tue Oct 11 2022 16:11:11:52"
```

Reference

- https://academy.binance.com/en/glossary/genesis-block
- https://bitcoin-info.guide/%E5%85%A5%E9%96%80%E6%8C%87%E5%BC%95/%E6%AF%94%E7%89%B9%E5%B9%A3%E9%81%8B%E4%BD%9C%E5%8E%9F%E7%90%86/%E4%BD%95%E8%AC%82%E5%8D%80%E5%A1%8A%E9%8F%882
- https://tdr.lib.ntu.edu.tw/handle/123456789/21350?mode=full
- https://ithelp.ithome.com.tw/articles/10215088
- https://sourceforge.net/p/bitcoin/code/133/tree/trunk/main.cpp#l444