Browseth Documentation

Release 0.0.59

Braden Pezeshki, Ryan Le

Oct 02, 2018

Developer Documentation

CHAPTER 1

Utilities

The utility functions exposed in both the *ethers* umbrella package and the *ethers-utils*:

```
const utils = require('@browseth/utils');

// or

import utils from ('@browseth/utils');
```

1.1 Array Buffers

An Array Buffer is an Array Buffer.

utils.ab **. isBytes (value [, length])** Checks to see if value is bytes and if it matches optional length

utils.ab **. fromView (view)** Returns an Array Buffer from view

utils.ab **. fromBytes (value [, length])** Returns Array Buffer from bytes with optional length

utils.ab **. fromUtf8 (value)** Returns Array Buffer from fromUtf8

utils.ab **. fromUInt (value)** Returns Array Buffer from UInt

utils.ab **. toUf8 (value)** Converts Array Buffer into Utf8

utils.ab **. toTwos (value, size)** Converts Array Buffer into a two's compliment

utils.ab **. stripStart (value)** Strips out the start of an Array Buffer

utils.ab **. padStart (value, length [, fillByte])** Pads the start of an Array Buffer

utils.ab **. padEnd (value, length [, fillByte])** Pads the end of an Array Buffer

utils.ab **. concat (values)** Concats an array of Array Buffers

1.2 Address

Utilities for manipulating addresses

utils.address **. isValid (value)** Checks if the given value is a valid address

utils.address **. from (value)** Returns an address from bytes

utils.address **. fromAddressAndNonce (address, nonce)** Returns an address from an address and nonce

1.3 Crypto

utils.crypto **. keccak256 (value)** returns the keccak256 of a string

TODO: uuid is meant for internal use. Not working externally yet. utils.crypto . uuid (value)

returns the uuid of a string

1.4 Interval

utils.interval **. setUnrefedInterval (fn, delay [, args])** Sets an interval that dies when the function it's wrapped in is finished

utils.interval **. setUnrefedTimeout (fn, delay [, args])** Sets a timeout that dies when the function it's wrapped in is finished

1.5 Param

utils.param **. toData (value, length)** Converts parameters to hex

utils.param **. toQuantity (value)** Converts parameters to hex string quantity

utils.param **. toTag (value)** Converts value into a tag

utils.param **. isData (value [, length])** Checks if value is data of optional length

utils.param **. isQuantity (value)** Checks if value is a quantity

utils.param **. isTag (value)** Checks if value is a tag

utils.param **. fromData (value, length)** Converts value to uint8Array of length

utils.param **. fromQuantity (value)** Converts quantity to Big Number

utils.param **. fromTag (value)** Converts tag to Big Number

1.6 Rlp

utils.param **. encode (value)** Encodes value to Array Buffer

utils.param **. encodeLength (len, offset)** Encodes length to Array Buffer with offset

1.7 Block Tracker

Poll for blocks every 5 seconds until a block number is confirmed. Use this class to keep track of block(s).

Creating instances

new Browseth.utils **. BlockTracker (requestQueue [, confirmationDelay = 0])** Request queue is ... TODO. The confirmation delay is the minimum number of confirmed blocks until the block is considered confirmed.

1.8 Observable

Subscribe to value changes with callbacks

Creating instances

new Browseth.utils **. Observable (value)** Create new Observable object with the value to watch.

Prototype

prototype **. subscribe (fn)** Add function to list of callbacks on value change. returns function to used unsubscribe function

prototype **. set (newValue)** Set the new value to watch. Triggers subscribed functions

prototype **. get ()** Gets the current watched value.

Example

```
const observable = new Observable('123');
const unsubscribe = observable.subscribe(() => console.log('This is an example'));
observable.set('456'); // Sets new value and logs 'This is an example'
unsubscribe(); // unsubscribe earlier subscribed function
observable.set('78'); // Will set new value with no callbacks
observable.get(); // returns '78'
```

1.9 Emitter

Add events with callbacks and trigger those callbacks by emitting events.

Creating instances

new Browseth.utils **. Emitter ( )** Create new Emitter object.

Prototype

prototype **. on (event, fn)** Add event label and provide callback

prototype **. off (event, fn)** Remove function from an event

prototype **. onEvery (fn)** Provide callback for every emit

prototype **. emit (event [, params])** Emit an event and pass parameters to the callbacks

CHAPTER 2

Indices and tables

- genindex
- modindex
- search