Package: hashids (via r-universe)

September 4, 2024

Title Generate Short Unique YouTube-Like IDs (Hashes) from Integers

Version 0.9.0.9000

Description An R port of the hashids library. hashids generates
YouTube-like hashes from integers or vector of integers.
Hashes generated from integers are relatively short, unique and
non-seqential. hashids can be used to generate unique ids for
URLs and hide database row numbers from the user. By default
hashids will avoid generating common English cursewords by
preventing certain letters being next to each other. hashids
are not one-way: it is easy to encode an integer to a hashid
and decode a hashid back into an integer.

URL https://github.com/ALShum/hashids-r/, http://hashids.org

BugReports https://github.com/ALShum/hashids-r/issues

Depends R (>= 3.2.2)

License MIT + file LICENSE

LazyData true

Suggests testthat

Repository https://alshum.r-universe.dev

RemoteUrl https://github.com/alshum/hashids-r

RemoteRef HEAD

RemoteSha c696e0c0881dfbc2de03c786516294fbb142b7a9

Contents

`ascii_val` *Calculate the ascii value number of a character*

Description

Calculate the ascii value number of a character

Usage

```
ascii_val(char)
```

Arguments

`char`	character

Value

ascii value integer

`base16_to_dec` *Converts a base 16 string to a base 10 number. Because I couldn't get base R functions to work for big hex numbers.*

Description

Converts a base 16 string to a base 10 number. Because I couldn't get base R functions to work for big hex numbers.

Usage

```
base16_to_dec(str_16)
```

Arguments

`str_16`	base 16 number as a string.

Value

base 10 integer.

decode — *Decodes a hashid into the original integer or integer vector*

Description

Decodes a hashid into the original integer or integer vector

Usage

```
decode(hash_str, settings)
```

Arguments

`hash_str`	hashid string to decode into integer or integer vector
`settings`	Settings list generated by hashid_settings

Value

integer or integer vector

decode_hex — *Decodes a hashid into the original hexidecimal number*

Description

Decodes a hashid into the original hexidecimal number

Usage

```
decode_hex(hashid, settings)
```

Arguments

`hashid`	hashid to decode
`settings`	Settings list generated by hashid_settings

Value

hexidecimal number as a string

`dec_to_base16` *Converts a base 10 number to base 16 number. Because I couldn't get R's as.hexmode() to work for big integers.*

Description

Converts a base 10 number to base 16 number. Because I couldn't get R's as.hexmode() to work for big integers.

Usage

```
dec_to_base16(dec)
```

Arguments

`dec`	base 10 integer

Value

base 16 number as a string

`encode` *Encodes an integer or integer vector into a hashid string. All numbers must be non-negative integers.*

Description

Encodes an integer or integer vector into a hashid string. All numbers must be non-negative integers.

Usage

```
encode(int, settings)
```

Arguments

`int`	Integer or integer vector to encode
`settings`	Settings list generated by hashid_settings

Value

hashid string

`encode_hex` *Encodes a hexademical number into a hashid*

Description

Encodes a hexademical number into a hashid

Usage

```
encode_hex(hex_str, settings)
```

Arguments

`hex_str`	Hexadecimal number as string
`settings`	Settings list generated by hashid_settings

Value

hashid string

`enforce_min_length` *Enforces hashid minimum length by padding the hashid with additional characters.*

Description

Enforces hashid minimum length by padding the hashid with additional characters.

Usage

```
enforce_min_length(encoded, min_length, alphabet, guards, values_hash)
```

Arguments

`encoded`	encoded hashid
`min_length`	minimum length required for hashid
`alphabet`	set of letters used to generate hashid
`guards`	set of guards used to generate hashid
`values_hash`	value hashed used to select guard characters

Value

hashid with padded characters to insure minimum length

hash	*Maps an integer to a string. Generated string will be inversely proportional to alphabet length.*

Description

Maps an integer to a string. Generated string will be inversely proportional to alphabet length.

Usage

```
hash(number, alphabet)
```

Arguments

number	Integer to hash
alphabet	Possible letters for string.

Value

hashed string

hashid_defaults	*Default Values for hashid settings*

Description

Default alphabet, separators, and ratio of character separators and guards for hashid

Usage

```
DEFAULT_ALPHABET

DEFAULT_SEPS

RATIO_SEPARATORS

RATIO_GUARDS
```

Format

```
 chr "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"
```

Source

http://www.hashids.org

`hashid_settings` *A function to create a hashid settings list.*

Description

A function to create a hashid settings list.

Usage

```
hashid_settings(salt, min_length = 0, alphabet = DEFAULT_ALPHABET,
  sep = DEFAULT_SEPS)
```

Arguments

`salt`	An additional string to make hashids more unique.
`min_length`	Minimum length for hashid.
`alphabet`	String of characters for hashid.
`sep`	String of characters to use as separators.

Value

A list of parameters used in encoding and decoding.

`shuffle` *Permutes the characters in a string based on an inputted salt string.*

Description

Permutes the characters in a string based on an inputted salt string.

Usage

```
shuffle(string, salt)
```

Arguments

`string`	String to be permuted
`salt`	cryptograph salt string that is used to permute strings

Value

shuffled string

`split` *Splits a string based on a set of splitting characters*

Description

Splits a string based on a set of splitting characters

Usage

```
split(string, splitters)
```

Arguments

`string`	String to split
`splitters`	set of splitting characters as a string

Value

split vector of characters

`unhash` *Unhashes a string to an integer based on alphabet.*

Description

Unhashes a string to an integer based on alphabet.

Usage

```
unhash(hashed, alphabet)
```

Arguments

`hashed`	String to unhash
`alphabet`	Set of letters used for hashing

Value

Unhashed integer

Index