

.add(key, value)

Add a new schema to the schemas object. As described in the constructor definition, `key` is a link prefix (`skype:`, for example), and `value` is a String to alias to another schema, or an Object with `validate` and optionally `normalize` definitions. To disable an existing rule, use `.add(key, null)`.

.set(options)

Override default options. Missed properties will not be changed.

License

[MIT](#)

linkify-it



Links recognition library with FULL unicode support. Focused on high quality link patterns detection in plain text.

[Demo](#)

Why it's awesome:

Full unicode support, *with astral characters!*

International domains support.

Allows rules extension & custom normalizers.

Install

Browserification is also supported.

Usage examples

Example 1

```
import linkifyit from 'linkify-it';
const linkify = linkifyit();

// Reload full tlds list & add
unofficial '.onion' domain.
linkify
.tlds(require('tlds'))
Reload with full tlds list
.tlds('onion',
true)
// Add
unofficial '.onion' domain
.add('git:',
'http:');
// Add 'git:'
protocol as "alias"
.add('ftp:', null)
//
Disable 'ftp:' protocol
.set({ fuzzyIP: true });
//
Enable IPs in fuzzy links
(without schema)
console.log(linkify.test('Site
github.com')); // true
console.log(linkify.match('Site
github.com')); // [
{
schema: ""},
//
```

2

- **index** - offset of matched text
- **lastIndex** - index of next char after match end
- **raw** - matched text
- **text** - normalized text
- **url** - link, generated from matched text

.matchAtStart(text)

Checks if a match exists at the start of the string. Returns `Match` (see docs for `match(text)`) or null if no URL is at the start. Doesn't work with fuzzy links.

.tlds(list, keepOld)

Load (or merge) new tlds list. Those are needed for fuzzy links (without schema) to avoid false positives. By default: 2-letter root zones are ok.

- `biz|com|edu|gov|net|org|pro|web|xxx|aero|asia|coop|info|museum|name|shop|pp` are ok.
- `encoded (x)n - -)` root zones are ok.

If that's not enough, you can reload defaults with more detailed zones list.

7

.test(text)

Searches linkifiable pattern and returns `true` on success or `false` on fail.

.pretest(text)

Quick check if link MAY BE can exist. Can be used to optimize more expensive `.test()` calls. Return `false` if link can not be found, `true` - if `.test()` call needed to know exactly.

.testSchemaAt(text, name, offset)

Similar to `.test()` but checks only specific protocol tail exactly at given position. Returns length of found pattern (0 on fail).

.match(text)

Returns Array of found link matches or null if nothing found.

Each match has:

- **schema** - link schema, can be empty for fuzzy links, or `//` for protocol-neutral links.

```
index: 5,  
lastIndex: 15,  
raw: "github.com",  
text: "github.com",  
url: "http://github.com",
```

Example 2. Add twitter mentions handler

```
linkify.add('@', {  
  validate: function (text, pos, self) {  
    const tail = text.slice(pos);  
  
    if (!self.re.twitter) {  
      self.re.twitter = new RegExp(  
        '^([a-zA-Z0-9_]){1,15}(?!_)(?  
        = $|' + self.re.src_ZPCc + ' )'  
      );  
    }  
    if (self.re.twitter.test(tail)) {  
      // Linkifier allows punctuation  
      // chars before prefix,  
      // but we additionally disable `@`  
      // ("@@mention" is invalid)  
      if (pos >= 2 && tail[pos - 2] ===  
        '@') {  
        return false;  
      }  
    }  
  }  
});
```

API

[API documentation](#)

```

return tail.match(self.re.twitter)
[0].length;
    }
    return 0;
},
normalize: function (match) {
    match.url = 'https://twitter.com/'
    + match.url.replace(/^@/, '');
}
};

```

- Creates new Linkifier instance with optional additional schemas. Can be called without new keyword for convenience.
- By default understands:
 - `http(s)://... , ftp://... , mailto:... & //...`
 - “fuzzy” links and emails (google.com, foo@bar.com).

- schemas is an object, where each key/value describes protocol/rule:
- **key** - link prefix (usually, protocol name with : at the end, skype: for example). Linkify - it makes sure that prefix is not preceded with alphanumeric char.
- **value** - rule to check tail after link prefix
 - *String* - just alias to existing rule
 - *Object*
 - *validate* - either a RegExp (start with ^, and don't include the link prefix itself), or a validator function which, given arguments *text*, *pos*, and *self*, returns the length of a match in *text* starting at index *pos*. *pos* is the index right after the link prefix. *self* can be used to access the linkify object to cache data.
 - *normalize* - optional function to normalize text & url of matched result (for example, for twitter mentions).
- **options:**
 - **fuzzyLink** - recognize URL-s without `http(s)://` head. Default true.
 - **fuzzyIP** - allow IPs in fuzzy links above. Can conflict with some texts like version numbers. Default false.
 - **fuzzyEmail** - recognize emails without `mailto:` prefix. Default true.
 - `—` - set true to terminate link with `--` (if it's considered as long dash).