to-regex-range

Pass two numbers, get a regex-compatible source string for matching ranges. Validated against more than 2.78 million test assertions.

Please consider following this project's author, Jon Schlinkert, and consider starring the project to show your :heart: and support.

Install

Install with npm:

What does this do?

This libary generates the `source` string to be passed to `new RegExp()` for matching a range of numbers.

Example

```
const toRegexRange = require('to-regex-
        range');
const regex = new
        RegExp(toRegexRange('15',
        '95'));
```

A string is returned so that you can do whatever you need with it before passing it to `new RegExp()` (like adding `^` or `$` boundaries, defining flags, or combining it another string).

Why use this library?

Convenience

Creating regular expressions for matching numbers gets deceptively complicated pretty fast.

For example, let's say you need a validation regex for matching part of a user-id, postal code, social security number, tax id, etc:

- regex for matching `1` => `/1/` (easy enough)
- regex for matching `1` through `5` => `/[1-5]/` (not bad…)
- regex for matching `1` or `5` => `/(1|5)/` (still easy…)
- regex for matching `1` through `50` => `/([1-9]|[1-4][0-9]|50)/` (uh-oh…)
- regex for matching `1` through `55` => `/([1-9]|[1-4][0-9]|5[0-5])/` (no prob, I can do this…)

Author

Jon Schlinkert

- GitHub Profile
- Twitter Profile
- LinkedIn Profile

Please consider supporting me on Patreon, or start your own Patreon page!

License

This file was generated by verb-generate-readme, v0.8.0, on April 07, 2019.

- regex for matching `1` through `555` => `/([1-9]|[1-9][0-9]|[1-4][0-9]{2}|5[0-4][0-9]|55[0-5])/` (maybe not…)
- regex for matching `0001` through `5555` => `/(0{3}[1-9]|0{2}[1-9][0-9]|0[1-9][0-9]{2}|[1-4][0-9]{3}|5[0-4][0-9]{2}|55[0-4][0-9]|555[0-5])/` (okay, I get the point!)

The numbers are contrived, but they're also really basic. In the real world you might need to generate a regex on-the-fly for validation.

Learn more

If you're interested in learning more about character classes and other regex features, I personally have always found regular-expressions.info to be pretty useful.

Heavily tested

As of April 07, 2019, this library runs >1m test assertions against generated regex-ranges to provide brute-force verification that results are correct.

Tests run in ~280ms on my MacBook Pro, 2.5 GHz Intel Core i7.

Optimized

Generated regular expressions are optimized:

- duplicate sequences and character classes are reduced using quantifiers
- smart enough to use ? conditionals when number(s) or range(s) can be positive or negative
- uses fragment caching to avoid processing the same exact string more than once

Usage

Add this library to your javascript application with the following line of code

```
const toRegexRange = require('to-regex-
        range');
```

The main export is a function that takes two integers: the `min` value and `max` value (formatted as strings or numbers).

```
const source = toRegexRange('15', '95');
//=> 1[5-9]|[2-8][0-9]|9[0-5]

const regex = new RegExp(`^${source}$`);
console.log(regex.test('14')); //=>
        false
```


Related projects

You might also be interested in these projects:

- expand-range: Fast, bash-like range expansion. Expand a range of numbers or letters, uppercase or lowercase. Used… more | homepage
- fill-range: Fill in a range of numbers or letters, optionally passing an increment or `step` to… more | homepage
- micromatch: Glob matching for javascript/node.js. A drop-in replacement and faster alternative to minimatch and multimatch. | homepage
- repeat-element: Create an array by repeating the given value n times. | homepage
- repeat-string: Repeat the given string n times. Fastest implementation for repeating a string. | homepage

Contributors

Commits	Contributor
63	jonschlinkert
3	doowb
2	realityking

Attribution

Inspired by the python library range-regex.

About

Contributing

Pull requests and stars are always welcome. For bugs and feature requests, please create an issue.

Running Tests

Running and reviewing unit tests is a great way to get familiarized with a library and its API. You can install dependencies and run tests with the following command:

```
$ npm install && npm test
```

Building docs

(This project's readme.md is generated by verb, please don't edit the readme directly. Any changes to the readme must be made in the .verb.md readme template.)

To generate the readme, run the following command:

```
$ npm install -g verbose/verb#dev verb-
        generate-readme && verb
```



```
console.log(regex.test('50')); //=> true
console.log(regex.test('94')); //=> true
console.log(regex.test('96')); //=>
        false
```

Options

options.capture

Type: `boolean`

Deafault: `undefined`

Wrap the returned value in parentheses when there is more than one regex condition. Useful when you're dynamically generating ranges.

```
console.log(toRegexRange('-10', '10'));
//=> -[1-9]|-?10|[0-9]

console.log(toRegexRange('-10', '10', {
        capture: true }));
//=> (-[1-9]|-?10|[0-9])
```

options.shorthand

Type: `boolean`

Deafault: `undefined`

Use the regex shorthand for `[0-9]`:

```
console.log(toRegexRange('0',
        '999999'));
//=> [0-9]|[1-9][0-9]{1,5}

console.log(toRegexRange('0', '999999',
        { shorthand: true }));
//=> \d|[1-9]\d{1,5}
```

options.relaxZeros

Type: `boolean`

Default: `true`

This option relaxes matching for leading zeros when when ranges are zero-padded.

```
const source = toRegexRange('-0010',
        '0010');
const regex = new RegExp(`^${source}$`);
console.log(regex.test('-10')); //=>
        true
console.log(regex.test('-010')); //=>
        true
console.log(regex.test('-0010')); //=>
        true
console.log(regex.test('10')); //=> true
console.log(regex.test('010')); //=>
        true
console.log(regex.test('0010')); //=>
        true
```



```
toRegexRange('29', '51');
//=> 29|[3-4][0-9]|5[0-1]
```

Steps / increments

This library does not support steps (increments). A pr to add support would be welcome.

History

v2.0.0 - 2017-04-21

New features

Adds support for zero-padding!

v1.0.0

Optimizations

Repeating ranges are now grouped using quantifiers. rocessing time is roughly the same, but the generated regex is much smaller, which should result in faster matching.

Range	Result	Compile time
`toRegexRange(1, 1000)`	`[1-9]|[1-9][0-9]{1,2}|1000`	
`toRegexRange(1, 10000)`	`[1-9]|[1-9][0-9]{1,3}|10000`	*34µs*
`toRegexRange(1, 100000)`	`[1-9]|[1-9][0-9]{1,4}|100000`	*36µs*
`toRegexRange(1, 1000000)`	`[1-9]|[1-9][0-9]{1,5}|1000000`	*42µs*
`toRegexRange(1, 10000000)`	`[1-9]|[1-9][0-9]{1,6}|10000000`	*42µs*

Heads up!

Order of arguments

When the `min` is larger than the `max`, values will be flipped to create a valid range:

```
toRegexRange('51', '29');
```

Is effectively flipped to:

When `relaxZeros` is false, matching is strict:

```
const source = toRegexRange('-0010',
        '0010', { relaxZeros: false });
const regex = new RegExp(`^${source}$`);
console.log(regex.test('-10')); //=>
        false
console.log(regex.test('-010')); //=>
        false
console.log(regex.test('-0010')); //=>
        true
console.log(regex.test('10')); //=>
        false
console.log(regex.test('010')); //=>
        false
console.log(regex.test('0010')); //=>
        true
```

Examples

Range	Result	Compile time
`toRegexRange(-10, 10)`	`-[1-9]|-?10|[0-9]`	*132µs*
`toRegexRange(-100, -10)`	`-1[0-9]|-[2-9][0-9]|-100`	*50µs*
`toRegexRange(-100, 100)`	`-[1-9]|-?[1-9][0-9]|-?100|[0-9]`	*42µs*

Range	Result	Compile time
`toRegexRange(001, 100)`	`0{0,2}[1-9]|0?[1-9][0-9]|100`	*109µs*
`toRegexRange(001, 555)`	`0{0,2}[1-9]|0?[1-9][0-9]|[1-4][0-9]{2}|5[0-4][0-9]|55[0-5]`	*51µs*
`toRegexRange(0010, 1000)`	`0{0,2}1[0-9]|0{0,2}[2-9][0-9]|0?[1-9][0-9]{2}|1000`	*31µs*
`toRegexRange(1, 50)`	`[1-9]|[1-4][0-9]|50`	*24µs*
`toRegexRange(1, 55)`	`[1-9]|[1-4][0-9]|5[0-5]`	*23µs*
`toRegexRange(1, 555)`	`[1-9]|[1-9][0-9]|[1-4][0-9]{2}|5[0-4][0-9]|55[0-5]`	*30µs*
`toRegexRange(1, 5555)`	`[1-9]|[1-9][0-9]{2}|[1-4][0-9]{3}|5[0-4][0-9]{2}|55[0-4][0-9]|555[0-5]`	*43µs*

Range	Result	Compile time
`toRegexRange(111, 555)`	`11[1-9]|1[2-9][0-9]|[2-4][0-9]{2}|5[0-4][0-9]|55[0-5]`	*38µs*
`toRegexRange(29, 51)`	`29|[34][0-9]|5[01]`	*24µs*
`toRegexRange(31, 877)`	`3[1-9]|[4-9][0-9]|[1-7][0-9]{2}|8[0-6][0-9]|87[0-7]`	*32µs*
`toRegexRange(5, 5)`	`5`	*8µs*
`toRegexRange(5, 6)`	`5|6`	*11µs*
`toRegexRange(1, 2)`	`1|2`	*6µs*
`toRegexRange(1, 5)`	`[1-5]`	*15µs*
`toRegexRange(1, 10)`	`[1-9]|10`	*22µs*
`toRegexRange(1, 100)`	`[1-9]|[1-9][0-9]|100`	*25µs*
		31µs