@isaacs/brace-expansion

A hybrid CJS/ESM TypeScript fork of brace-expansion.

Brace expansion, as known from sh/bash, in JavaScript.

Example

```
// => ['file2.jpg', 'file1.jpg',
        'file0.jpg']

expand('file{0..4..2}.jpg')
// => ['file0.jpg', 'file2.jpg',
        'file4.jpg']

expand('file-{a..e..2}.jpg')
// => ['file-a.jpg', 'file-c.jpg',
        'file-e.jpg']

expand('file{00..10..5}.jpg')
// => ['file00.jpg', 'file05.jpg',
        'file10.jpg']

expand('{{A..C},{a..c}}')
// => ['A', 'B', 'C', 'a', 'b', 'c']

expand('ppp{,config,oe{,conf}}')
// => ['ppp', 'pppconfig', 'pppoe',
        'pppoeconf']
```

API

```
import { expand } from '@isaacs/brace-
        expansion'
```


const expanded = expand(str)

Return an array of all possible and valid expansions of `str`. If none are found, `[str]` is returned.

Valid expansions are:

```
/^(.*,)+(.+)?$/
// {a,b,...}
```

A comma separated list of options, like `{a,b}` or `{a,{b,c}}` or `{,a,}`.

```
/^-?\d+\.\.-?\d+(\.\.-?\d+)?$/
// {x..y[..incr]}
```

A numeric sequence from `x` to `y` inclusive, with optional increment. If `x` or `y` start with a leading `0`, all the numbers will be padded to have equal length. Negative numbers and backwards iteration work too.

```
/^-?\d+\.\.-?\d+(\.\.-?\d+)?$/
// {x..y[..incr]}
```

An alphabetic sequence from `x` to `y` inclusive, with optional increment. `x` and `y` must be exactly one character, and if given, `incr` must be a number.

For compatibility reasons, the string `${` is not eligible for brace expansion.