Encode URL

Encode a URL to a percent-encoded form, excluding already-encoded sequences.

Installation

```
npm install encodeurl
```

API

```
var encodeUrl = require('encodeurl')
```

encodeUrl(url)

Encode a URL to a percent-encoded form, excluding already-encoded sequences.

This function accepts a URL and encodes all the non-URL code points (as UTF-8 byte sequences). It will not encode the "%" character unless it is not part of a valid sequence (`%20` will be left as-is, but `%foo` will be encoded as `%25foo`).

This encode is meant to be “safe” and does not throw errors. It will try as hard as it can to properly encode the given URL, including replacing any raw, unpaired surrogate pairs with the Unicode replacement character prior to encoding.

Examples

Encode a URL containing user-controlled data

```
var encodeUrl = require('encodeurl')
var escapeHtml = require('escape-html')

http.createServer(function onRequest
        (req, res) {
  // get encoded form of inbound url
  var url = encodeUrl(req.url)

  // create html message
  var body = '<p>Location ' +
        escapeHtml(url) + ' not found</
        p>'

  // send a 404
  res.statusCode = 404
  res.setHeader('Content-Type', 'text/
        html; charset=UTF-8')
```

```
  res.setHeader('Content-Length',
        String(Buffer.byteLength(body,
        'utf-8')))
  res.end(body, 'utf-8')
})
```

Encode a URL for use in a header field

```
var encodeUrl = require('encodeurl')
var escapeHtml = require('escape-html')
var url = require('url')

http.createServer(function onRequest
        (req, res) {
  // parse inbound url
  var href = url.parse(req)

  // set new host for redirect
  href.host = 'localhost'
  href.protocol = 'https:'
  href.slashes = true

  // create location header
  var location =
        encodeUrl(url.format(href))

  // create html message
  var body = '<p>Redirecting to new
        site: ' + escapeHtml(location)
        + '</p>'

  // send a 301
```

```
  res.statusCode = 301
  res.setHeader('Content-Type', 'text/
        html; charset=UTF-8')
  res.setHeader('Content-Length',
        String(Buffer.byteLength(body,
        'utf-8')))
  res.setHeader('Location', location)
  res.end(body, 'utf-8')
})
```

Similarities

This function is *similar* to the intrinsic function `encodeURI`. However, it will not encode:

- The `\`, `^`, or `|` characters
- The `%` character when it's part of a valid sequence
- `[` and `]` (for IPv6 hostnames)
- Replaces raw, unpaired surrogate pairs with the Unicode replacement character

As a result, the encoding aligns closely with the behavior in the WHATWG URL specification. However, this package only encodes strings and does not do any URL parsing or formatting.

It is expected that any output from `new URL(url)` will not change when used with this package, as the output has already been encoded. Additionally, if we were to encode before `new URL(url)`, we do not expect the before and after encoded formats to be parsed any differently.

Testing

```
$ npm test
$ npm run lint
```

References

- RFC 3986: Uniform Resource Identifier (URI): Generic Syntax
- WHATWG URL Living Standard

License