<> **Code** | ⊙ Issues 6 | ⊷ Pull requests 4 | ▷ Actions | ⊞ Projects | 📖 Wiki | ⋯

⅄ d9f4e23cf4 ⌄

loader-utils / lib / **interpolateName.js** / <> Jump to ⌄

🐙 **Evilebot Tnawi** feat: the `resourceQuery` is passed to the `interpolateName` me... ⋯ ✓ | 🕐 History

🗑 **8 contributors** 🟦 🔵 🟤 🟢 ⚫ ⚪ 🔵 ⚫
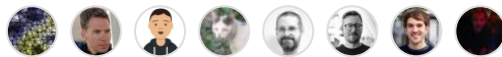
| 151 lines (119 sloc) | 3.69 KB | ⋯ |

```javascript
1   'use strict';
2
3   const path = require('path');
4   const emojisList = require('emojis-list');
5   const getHashDigest = require('./getHashDigest');
6
7   const emojiRegex = /[\uD800-\uDFFF]./;
8   const emojiList = emojisList.filter((emoji) => emojiRegex.test(emoji));
9   const emojiCache = {};
10
11  function encodeStringToEmoji(content, length) {
12    if (emojiCache[content]) {
13      return emojiCache[content];
14    }
15
16    length = length || 1;
17
18    const emojis = [];
19
20    do {
21      if (!emojiList.length) {
22        throw new Error('Ran out of emoji');
23      }
24
25      const index = Math.floor(Math.random() * emojiList.length);
26
27      emojis.push(emojiList[index]);
28      emojiList.splice(index, 1);
29    } while (--length > 0);
```

```javascript
  const emojiEncoding = emojis.join('');

  emojiCache[content] = emojiEncoding;

  return emojiEncoding;
}

function interpolateName(loaderContext, name, options) {
  let filename;

  const hasQuery =
    loaderContext.resourceQuery && loaderContext.resourceQuery.length > 1;

  if (typeof name === 'function') {
    filename = name(
      loaderContext.resourcePath,
      hasQuery ? loaderContext.resourceQuery : undefined
    );
  } else {
    filename = name || '[hash].[ext]';
  }

  const context = options.context;
  const content = options.content;
  const regExp = options.regExp;

  let ext = 'bin';
  let basename = 'file';
  let directory = '';
  let folder = '';
  let query = '';

  if (loaderContext.resourcePath) {
    const parsed = path.parse(loaderContext.resourcePath);
    let resourcePath = loaderContext.resourcePath;

    if (parsed.ext) {
      ext = parsed.ext.substr(1);
    }

    if (parsed.dir) {
      basename = parsed.name;
      resourcePath = parsed.dir + path.sep;
    }

    if (typeof context !== 'undefined') {
      directory = path
        .relative(context, resourcePath + '_')
```

```
 79          .replace(/\\/g, '/')
 80          .replace(/\.\.(\/)?/g, '_$1');
 81        directory = directory.substr(0, directory.length - 1);
 82      } else {
 83        directory = resourcePath.replace(/\\/g, '/').replace(/\.\.(\/)?/g, '_$1');
 84      }
 85
 86      if (directory.length === 1) {
 87        directory = '';
 88      } else if (directory.length > 1) {
 89        folder = path.basename(directory);
 90      }
 91    }
 92
 93    if (loaderContext.resourceQuery && loaderContext.resourceQuery.length > 1) {
 94      query = loaderContext.resourceQuery;
 95
 96      const hashIdx = query.indexOf('#');
 97
 98      if (hashIdx >= 0) {
 99        query = query.substr(0, hashIdx);
100      }
101    }
102
103    let url = filename;
104
105    if (content) {
106      // Match hash template
107      url = url
108        // `hash` and `contenthash` are same in `loader-utils` context
109        // let's keep `hash` for backward compatibility
110        .replace(
111          /\[(?:([^:\]]+):)?(?:hash|contenthash)(?::([a-z]+\d*))?(?::(\d+))?\]/gi,
112          (all, hashType, digestType, maxLength) =>
113            getHashDigest(content, hashType, digestType, parseInt(maxLength, 10))
114        )
115        .replace(/\[emoji(?::(\d+))?\]/gi, (all, length) =>
116          encodeStringToEmoji(content, parseInt(length, 10))
117        );
118    }
119
120    url = url
121      .replace(/\[ext\]/gi, () => ext)
122      .replace(/\[name\]/gi, () => basename)
123      .replace(/\[path\]/gi, () => directory)
124      .replace(/\[folder\]/gi, () => folder)
125      .replace(/\[query\]/gi, () => query);
126
127    if (regExp && loaderContext.resourcePath) {
```

```
128        const match = loaderContext.resourcePath.match(new RegExp(regExp));
129
130      match &&
131        match.forEach((matched, i) => {
132          url = url.replace(new RegExp('\\[' + i + '\\]', 'ig'), matched);
133        });
134    }
135
136    if (
137      typeof loaderContext.options === 'object' &&
138      typeof loaderContext.options.customInterpolateName === 'function'
139    ) {
140      url = loaderContext.options.customInterpolateName.call(
141        loaderContext,
142        url,
143        name,
144        options
145      );
146    }
147
148    return url;
149  }
150
151  module.exports = interpolateName;
```