
piexifjs Documentation

Release 2.0

hMatoba

Dec 24, 2019

Contents

1	About Piexifjs	3
1.1	Notice and Warning!	3
1.2	What for?	3
1.3	How to Use	3
1.4	Dependency	4
1.5	Environment	4
1.6	License	4
2	Functions	5
2.1	load	5
2.2	dump	6
2.3	insert	7
2.4	remove	7
3	Constants	9
3.1	To set exif values	9
3.2	To read exif keys	9
4	Helpers	11
4.1	degrees in <i>rational</i>	11
4.2	degrees in <i>number</i>	11
5	1 to 2	13
6	Samples	15
6.1	Insert Exif into jpeg	15
6.2	Read Exif Values	16
6.3	Generates Rotated JPEG	16
6.4	GPS Coordinates	18
6.5	Node.js	18
7	Appendices	21
7.1	Exif Data in Piexifjs	21
8	Development	23
9	Changelog	25
9.1	2.0.0(Not Completed)	25

9.2	1.03	25
9.3	1.02	25
9.4	1.01	25
9.5	1.0	25
10	Indices and tables	27
	Index	29

Exif manipulations with JavaScript. Writing, reading, and more... For client-side and Node.js. <https://github.com/hMatoba/piexifjs>

1.1 Notice and Warning!

We are implementing v2.0. This version would include a few big changes. If you won't ready to use, don't update this library.

- add some arguments type checks
- stop to support bower
- some data types are changed in exif object...?

Some names in this library will be changed until beta version.

1.2 What for?

Exif manipulations in JS. Writing, reading, and more...

1.3 How to Use

There are only just four functions.

- *load(jpegData)* - Get exif data as *object*.
- *dump(exifObj)* - Get exif binary as *string* to insert into JPEG.
- *insert(exifbytes, jpegData)* - Insert exif into JPEG.
- *remove(jpegData)* - Remove exif from JPEG.

and some utilities.

1.4 Dependency

No dependencies.

1.5 Environment

Both client-side and server-side. Piexifjs is transpiled as [Universal Module Definition](#).

1.6 License

The MIT License (MIT)

Copyright (c) 2015 hMatoba

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER 2

Functions

Warning: It could set any value in exif without actual value. For example, actual XResolution is 300, whereas XResolution value in exif is 0. Confliction might happen.

Warning: To edit exif tags and values appropriately, read official document from P167-. http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf

2.1 load

`piexif.load(jpegData)`

Get exif data as *object*. `jpegData` must be a *string* that starts with “data:image/jpeg;base64,”(DataURL), “\xff\xd8”, or “Exif”.

Arguments

- `jpegData` (*string*) – JPEG data

Returns Exif data({“0th”:object, “Exif”:object, “GPS”:object, “Interop”:object, “1st”:object, “thumbnail”:string})

Return type object

```
let exifObj = piexif.load(jpegData);
for (let ifd in exifObj) {
  if (ifd == "thumbnail") {
    continue;
  }
  console.log("-" + ifd);
  for (let tag in exifObj[ifd]) {
    console.log("  " + piexif.Tags[ifd][tag]["name"] + ":" + exifObj[ifd][tag]);
  }
}
```

(continues on next page)

(continued from previous page)

```
}  
}
```

2.2 dump

`piexif.dump(exifObj)`

Get exif binary as *string* to insert into JPEG.

Arguments

- **exifObj** (*object*) – Exif data({“0th”:0thIFD - object, “Exif”:ExifIFD - object, “GPS”:GPSIFD - object, “Interop”:InteroperabilityIFD - object, “1st”:1stIFD - object, “thumbnail”:JPEG data - string})

Returns Exif binary

Return type string

```
let zeroth = {};  
let exif = {};  
let gps = {};  
zeroth[piexif.TagValues.ImageIFD.Make] = "Make";  
zeroth[piexif.TagValues.ImageIFD.XResolution] = [777, 1];  
zeroth[piexif.TagValues.ImageIFD.YResolution] = [777, 1];  
zeroth[piexif.TagValues.ImageIFD.Software] = "Piexifjs";  
exif[piexif.TagValues.ExifIFD.DateTimeOriginal] = "2010:10:10 10:10:10";  
exif[piexif.TagValues.ExifIFD.LensMake] = "LensMake";  
exif[piexif.TagValues.ExifIFD.Sharpness] = 777;  
exif[piexif.TagValues.ExifIFD.LensSpecification] = [[1, 1], [1, 1], [1, 1], [1, 1]];  
gps[piexif.TagValues.GPSIFD.GPSVersionID] = [7, 7, 7, 7];  
gps[piexif.TagValues.GPSIFD.GPSDateStamp] = "1999:99:99 99:99:99";  
let exifObj = {"0th":zeroth, "Exif":exif, "GPS":gps};  
const exifbytes = piexif.dump(exifObj);
```

Properties of *piexif.TagValues.ImageIFD* help to make 0thIFD and 1stIFD. *piexif.TagValues.ExifIFD* is for ExifIFD. *piexif.TagValues.GPSIFD* is for GPSIFD. *piexif.InteropIFD* is for InteroperabilityIFD.

Note: ExifTag(34665), GPSTag(34853), and InteroperabilityTag(40965) in 0thIFD automatically are set appropriate value.

Note: JPEGInterchangeFormat(513), and JPEGInterchangeFormatLength(514) in 1stIFD automatically are set appropriate value.

Note: If ‘thumbnail’ is contained in *exifObj*, ‘1st’ must be contained – and vice versa. 1stIFD means thumbnail’s information.

2.3 insert

`piexif.insert(exifbytes, jpegData)`

Insert exif into JPEG.

Arguments

- **exifbytes** (*string*) – Exif binary
- **jpegData** (*string*) – JPEG data

Returns JPEG data

Return type string

```
const exifbytes = piexif.dump(exifObj)
const newJpeg = piexif.insert(exifbytes, jpegData)
```

2.4 remove

`piexif.remove(jpegData)`

Remove exif from JPEG.

Arguments

- **jpegData** (*string*) – JPEG data

Returns JPEG data

Return type string

```
const newJpeg = piexif.remove(jpegData)
```


3.1 To set exif values

0th IFD and 1st IFD: *piexif.TagValues.ImageIFD*

Exif IFD: *piexif.TagValues.ExifIFD*

GPS IFD: *piexif.TagValues.GPSIFD*

Interoperability IFD: *piexif.TagValues.InteropIFD*

```
let zerothIfd = {
  [piexif.TagValues.ImageIFD.ProcessingSoftware]: 'piexifjs',
  [piexif.TagValues.ImageIFD.XResolution]: [777, 1],
  [piexif.TagValues.ImageIFD.YResolution]: [777, 1],
  [piexif.TagValues.ImageIFD.Software]: "Piexifjs"
};
let exifIfd = {
  [piexif.TagValues.ExifIFD.DateTimeOriginal]: "2010:10:10 10:10:10",
  [piexif.TagValues.ExifIFD.LensMake]: "LensMake",
  [piexif.TagValues.ExifIFD.Sharpness]: 777,
  [piexif.TagValues.ExifIFD.LensSpecification]: [[1, 1], [1, 1], [1, 1], [1, 1]]
};
let gpsIfd = {
  [piexif.TagValues.GPSIFD.GPSVersionID]: [7, 7, 7, 7],
  [piexif.TagValues.GPSIFD.GPSDateStamp]: "1999:99:99 99:99:99"
};
let exifObj = {"0th": zerothIfd, "Exif": exifIfd, "GPS": gpsIfd};
```

3.2 To read exif keys

in *piexif.Tags*

```
let exifObj = piexif.load(exifBinary);
for (let ifd in exifObj) {
  if (ifd == "thumbnail") {
    continue;
  }
  console.log("-" + ifd);
  for (let tag in exifObj[ifd]) {
    console.log("  " + piexif.Tags[ifd][tag]["name"] + ":" + exifObj[ifd][tag]);
  }
}
```

4.1 degrees in *rational*

`piexif.helper.GPSHelper.degToDmsRational (degrees)`

Convert `degrees(number)` to `[[deg1, deg2], [min1, min2], [sec1, sec2]]`.

Arguments

- **degrees** (*number*) – degrees

Returns degrees as `[degrees, minutes, seconds]`

Return type Array

```
const degreesRat = piexif.helper.GPSHelper.degToDmsRational(63.2);
```

4.2 degrees in *number*

`piexif.helper.GPSHelper.dmsRationalToDeg (degrees, direction)`

Convert `degrees([[deg1, deg2], [min1, min2], [sec1, sec2]])` to *number*.

Arguments

- **degrees** (*Array*) – degrees in `[[deg1, deg2], [min1, min2], [sec1, sec2]]`
- **direction** (*string*) – “N”, “E”, “W”, or “S”

Returns degrees

Return type number

```
const degreesNum = piexif.helper.GPSHelper.dmsRationalToDeg([[60, 1], [10, 1], [10, 1]], "S");
```


CHAPTER 5

1 to 2

piexif.TAGS -> piexif.Tags

*piexif.*IFD -> piexif.TagValues.*IFD*

6.1 Insert Exif into jpeg

```
<input type="file" id="files" />
<script source="/js/piexif.js" />
<script>
function handleFileSelect(evt) {
    var file = evt.target.files[0];

    var zeroth = {};
    var exif = {};
    var gps = {};
    zeroth[piexif.TagValues.ImageIFD.Make] = "Make";
    zeroth[piexif.TagValues.ImageIFD.XResolution] = [777, 1];
    zeroth[piexif.TagValues.ImageIFD.YResolution] = [777, 1];
    zeroth[piexif.TagValues.ImageIFD.Software] = "Piexifjs";
    exif[piexif.TagValues.ExifIFD.DateTimeOriginal] = "2010:10:10 10:10:10";
    exif[piexif.TagValues.ExifIFD.LensMake] = "LensMake";
    exif[piexif.TagValues.ExifIFD.Sharpness] = 777;
    exif[piexif.TagValues.ExifIFD.LensSpecification] = [[1, 1], [1, 1], [1, 1], [1,
↪1]];
    gps[piexif.TagValues.GPSIFD.GPSVersionID] = [7, 7, 7, 7];
    gps[piexif.TagValues.GPSIFD.GPSDateStamp] = "1999:99:99 99:99:99";
    var exifObj = {"0th":zeroth, "Exif":exif, "GPS":gps};
    var exifbytes = piexif.dump(exifObj);

    var reader = new FileReader();
    reader.onload = function(e) {
        var inserted = piexif.insert(exifbytes, e.target.result);

        var image = new Image();
        image.src = inserted;
        image.width = 200;
        var el = $("<div></div>").append(image);
    }
}
```

(continues on next page)

(continued from previous page)

```
        $("#resized").prepend(el);

    };
    reader.readAsDataURL(file);
}

document.getElementById('files').addEventListener('change', handleFileSelect, false);
</script>
```

6.2 Read Exif Values

```
var reader = new FileReader();
reader.onloadend = function(e) {
    var exifObj = piexif.load(e.target.result);
    for (var ifd in exifObj) {
        if (ifd == "thumbnail") {
            continue;
        }
        console.log("-" + ifd);
        for (var tag in exifObj[ifd]) {
            console.log("  " + piexif.Tags[ifd][tag]["name"] + ":" +
↪exifObj[ifd][tag]);
        }
    }
};
reader.readAsDataURL(file);
```

6.3 Generates Rotated JPEG

```
function postJpeg (binStr) {
    var array = [];
    for (var p=0; p<data.length; p++) {
        array[p] = data.charCodeAt(p);
    }
    var u8array = new Uint8Array(array);

    var req = new XMLHttpRequest();
    req.open("POST", "/jpeg", false);
    req.setRequestHeader('Content-Type', 'image/jpeg');
    req.send(u8array.buffer);
}

function previewJpeg(evt) {
    var files = evt.target.files;
    var previewDiv = $("#preview");
    for (var i=0; i<files.length; i++) {
        var file = files[i];
        if (!file.type.match('image/jpeg.*')) {
            continue;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

var reader = new FileReader();
reader.onload = function(e) {
    var exif = piexif.load(e.target.result);
    var image = new Image();
    image.onload = function () {
        var orientation = exif["0th"][piexif.TagValues.ImageIFD.Orientation];

        var canvas = document.createElement("canvas");
        canvas.width = image.width;
        canvas.height = image.height;
        var ctx = canvas.getContext("2d");
        var x = 0;
        var y = 0;
        ctx.save();
        if (orientation == 2) {
            x = -canvas.width;
            ctx.scale(-1, 1);
        } else if (orientation == 3) {
            x = -canvas.width;
            y = -canvas.height;
            ctx.scale(-1, -1);
        } else if (orientation == 4) {
            y = -canvas.height;
            ctx.scale(1, -1);
        } else if (orientation == 5) {
            canvas.width = image.height;
            canvas.height = image.width;
            ctx.translate(canvas.width, canvas.height / canvas.width);
            ctx.rotate(Math.PI / 2);
            y = -canvas.width;
            ctx.scale(1, -1);
        } else if (orientation == 6) {
            canvas.width = image.height;
            canvas.height = image.width;
            ctx.translate(canvas.width, canvas.height / canvas.width);
            ctx.rotate(Math.PI / 2);
        } else if (orientation == 7) {
            canvas.width = image.height;
            canvas.height = image.width;
            ctx.translate(canvas.width, canvas.height / canvas.width);
            ctx.rotate(Math.PI / 2);
            x = -canvas.height;
            ctx.scale(-1, 1);
        } else if (orientation == 8) {
            canvas.width = image.height;
            canvas.height = image.width;
            ctx.translate(canvas.width, canvas.height / canvas.width);
            ctx.rotate(Math.PI / 2);
            x = -canvas.height;
            y = -canvas.width;
            ctx.scale(-1, -1);
        }
        ctx.drawImage(image, x, y);
        ctx.restore();

        var dataURL = canvas.toDataURL("image/jpeg", 1.0);
    }
}

```

(continues on next page)

(continued from previous page)

```
        var jpegBinary = atob(dataURL.split(",")[1]);

        var div = $("<div></div>");
        div.append(canvas);
        var button = $("<button>post this image</button>");
        button.click(function () {
            //postJpeg(jpegBinary);
        });

        previewDiv.prepend(div).prepend(button);
    };
    image.src = e.target.result;
};

reader.readAsDataURL(file);
}
}

document.getElementById("files").onchange = previewJpeg;
```

6.4 GPS Coordinates

```
var lat = 59.43553989213321;
var lng = 24.73842144012451;
gpsIfd[piexif.TagValues.GPSIFD.GPSLatitudeRef] = lat < 0 ? 'S' : 'N';
gpsIfd[piexif.TagValues.GPSIFD.GPSLatitude] = piexif.GPSHelper.degToDmsRational(lat);
gpsIfd[piexif.TagValues.GPSIFD.GPSLongitudeRef] = lng < 0 ? 'W' : 'E';
gpsIfd[piexif.TagValues.GPSIFD.GPSLongitude] = piexif.GPSHelper.degToDmsRational(lng);
```

6.5 Node.js

```
var piexif = require("piexifjs");
var fs = require("fs");

var filename1 = "in.jpg";
var filename2 = "out.jpg";

var jpeg = fs.readFileSync(filename1);
var data = jpeg.toString("binary");

var zeroth = {};
var exif = {};
var gps = {};
zeroth[piexif.TagValues.ImageIFD.Make] = "Make";
zeroth[piexif.TagValues.ImageIFD.XResolution] = [777, 1];
zeroth[piexif.TagValues.ImageIFD.YResolution] = [777, 1];
zeroth[piexif.TagValues.ImageIFD.Software] = "Piexifjs";
exif[piexif.TagValues.ExifIFD.DateTimeOriginal] = "2010:10:10 10:10:10";
exif[piexif.TagValues.ExifIFD.LensMake] = "LensMake";
exif[piexif.TagValues.ExifIFD.Sharpness] = 777;
exif[piexif.TagValues.ExifIFD.LensSpecification] = [[1, 1], [1, 1], [1, 1], [1, 1]];
```

(continues on next page)

(continued from previous page)

```
gps[piexif.TagValues.GPSIFD.GPSVersionID] = [7, 7, 7, 7];
gps[piexif.TagValues.GPSIFD.GPSDateStamp] = "1999:99:99 99:99:99";
var exifObj = {"0th":zeroth, "Exif":exif, "GPS":gps};
var exifbytes = piexif.dump(exifObj);

var newData = piexif.insert(exifbytes, data);
var newJpeg = Buffer.from(newData, "binary");
fs.writeFileSync(filename2, newJpeg);
```


7.1 Exif Data in Piexifjs

Each exif tag has appropriate type of the value. BYTE, ASCII, SHORT, or... See the document of Exif. http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf

Exif Type	JavaScript
BYTE	int
ASCII	string
SHORT	int
LONG	int
RATIONAL	[int, int]
UNDEFINED	string
SRATIONAL	[int, int]

If value type is number (BYTE, SHORT, LONG, RATIONAL, or SRATIONAL) and value count is two or more number, it is expressed with *Array*.

BYTE, SHORT, LONG	[int, int, ...]
RATIONAL, SRATIONAL	[[int, int], [int, int], ...]

Note: If value type is number and value count is one, *array* that is length one value (e.g. [int]) also be accepted.

CHAPTER 8

Development

start development.

```
npm install
```

Source codes are in “./src”.

Codes are written in TypeScript. Transpile with below comand.

```
npm run build
```

You can get *piexif.js* under */dist*.

Run tests.

```
npm run node-test  
npm run browser-test
```


9.1 2.0.0(Not Completed)

- Some name spaces are changed.
- Checking types for some argumetns.

9.2 1.03

- Bug fix. Issues #19.

9.3 1.02

- Bug fix. To remove IFD pointer tag.

9.4 1.01

- Bug fix. <https://github.com/hMatoba/piexifjs/issues/9>

9.5 1.0

- Release.

CHAPTER 10

Indices and tables

- `genindex`
- `modindex`
- `search`

P

`piexif.dump()` (*piexif method*), [6](#)
`piexif.helper.GPSHelper.degToDmsRational()`
 (*piexif.helper.GPSHelper method*), [11](#)
`piexif.helper.GPSHelper.dmsRationalToDeg()`
 (*piexif.helper.GPSHelper method*), [11](#)
`piexif.insert()` (*piexif method*), [7](#)
`piexif.load()` (*piexif method*), [5](#)
`piexif.remove()` (*piexif method*), [7](#)