piexifjs Documentation

Release 1.0

hMatoba

Dec 24, 2019

Contents

1 About Piexifjs **3**
- 1.1 What for? . . . 3
- 1.2 How to Use . . . 3
- 1.3 Dependency . . . 3
- 1.4 Environment . . . 3
- 1.5 License . . . 4

2 Functions **5**
- 2.1 load . . . 5
- 2.2 dump . . . 6
- 2.3 insert . . . 7
- 2.4 remove . . . 7

3 Appendices **9**
- 3.1 Exif Data in Piexifjs . . . 9

4 Samples **11**
- 4.1 Insert Exif into jpeg . . . 11
- 4.2 Read Exif Values . . . 12
- 4.3 Generates Rotated JPEG . . . 12
- 4.4 GPS Coordinates . . . 14
- 4.5 Node.js . . . 14

5 Changelog **17**
- 5.1 1.0.6 . . . 17
- 5.2 1.03 . . . 17
- 5.3 1.02 . . . 17
- 5.4 1.01 . . . 17
- 5.5 1.0 . . . 17

6 Indices and tables **19**

Index **21**

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

CHAPTER 1

About Piexifjs

1.1 What for?

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

1.2 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.

1.3 Dependency

No dependencies.

1.4 Environment

Tested on IE11, Opera28, and PhantomJS 1.9.8. It runs on even Node.js.

1.5 License

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

```
var exifObj = piexif.load(jpegData);
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]);
```

(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

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

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

```
var exifbytes = piexif.dump(exifObj)
var 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

```
var newJpeg = piexif.remove("foo.jpg")
```

CHAPTER 3

Appendices

3.1 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 4

Samples

4.1 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.ImageIFD.Make] = "Make";
    zeroth[piexif.ImageIFD.XResolution] = [777, 1];
    zeroth[piexif.ImageIFD.YResolution] = [777, 1];
    zeroth[piexif.ImageIFD.Software] = "Piexifjs";
    exif[piexif.ExifIFD.DateTimeOriginal] = "2010:10:10 10:10:10";
    exif[piexif.ExifIFD.LensMake] = "LensMake";
    exif[piexif.ExifIFD.Sharpness] = 777;
    exif[piexif.ExifIFD.LensSpecification] = [[1, 1], [1, 1], [1, 1], [1, 1]];
    gps[piexif.GPSIFD.GPSVersionID] = [7, 7, 7, 7];
    gps[piexif.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);
        $("#resized").prepend(el);
```

(continues on next page)

(continued from previous page)

```
    };
    reader.readAsDataURL(file);
}

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

4.2 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);
```

4.3 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.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 == 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);
```

(continues on next page)

(continued from previous page)

```
                ctx.restore();

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

4.4 GPS Coordinates

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

4.5 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.ImageIFD.Make] = "Make";
zeroth[piexif.ImageIFD.XResolution] = [777, 1];
zeroth[piexif.ImageIFD.YResolution] = [777, 1];
zeroth[piexif.ImageIFD.Software] = "Piexifjs";
exif[piexif.ExifIFD.DateTimeOriginal] = "2010:10:10 10:10:10";
```

(continues on next page)

(continued from previous page)

```
exif[piexif.ExifIFD.LensMake] = "LensMake";
exif[piexif.ExifIFD.Sharpness] = 777;
exif[piexif.ExifIFD.LensSpecification] = [[1, 1], [1, 1], [1, 1], [1, 1]];
gps[piexif.GPSIFD.GPSVersionID] = [7, 7, 7, 7];
gps[piexif.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);
```

CHAPTER 5

Changelog

5.1 1.0.6

- Support parsing *SLONG*.

5.2 1.03

- Bug fix. Issues #19.

5.3 1.02

- Bug fix. To remove IFD pointer tag.

5.4 1.01

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

5.5 1.0

- Release.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index

P

`piexif.dump()` (*piexif method*), 6
`piexif.insert()` (*piexif method*), 7
`piexif.load()` (*piexif method*), 5
`piexif.remove()` (*piexif method*), 7