Demo Docs FAQ

Papa Parse 5

GitHub Help

Convert CSV to JSON

Delimited data can be parsed out of strings or files. Files that are parsed can be local or remote. Local files are opened with FileReader, and remote files are downloaded with XMLHttpRequest.

- Returns a parse results object (if not streaming or using worker).

PARSE LOCAL FILES

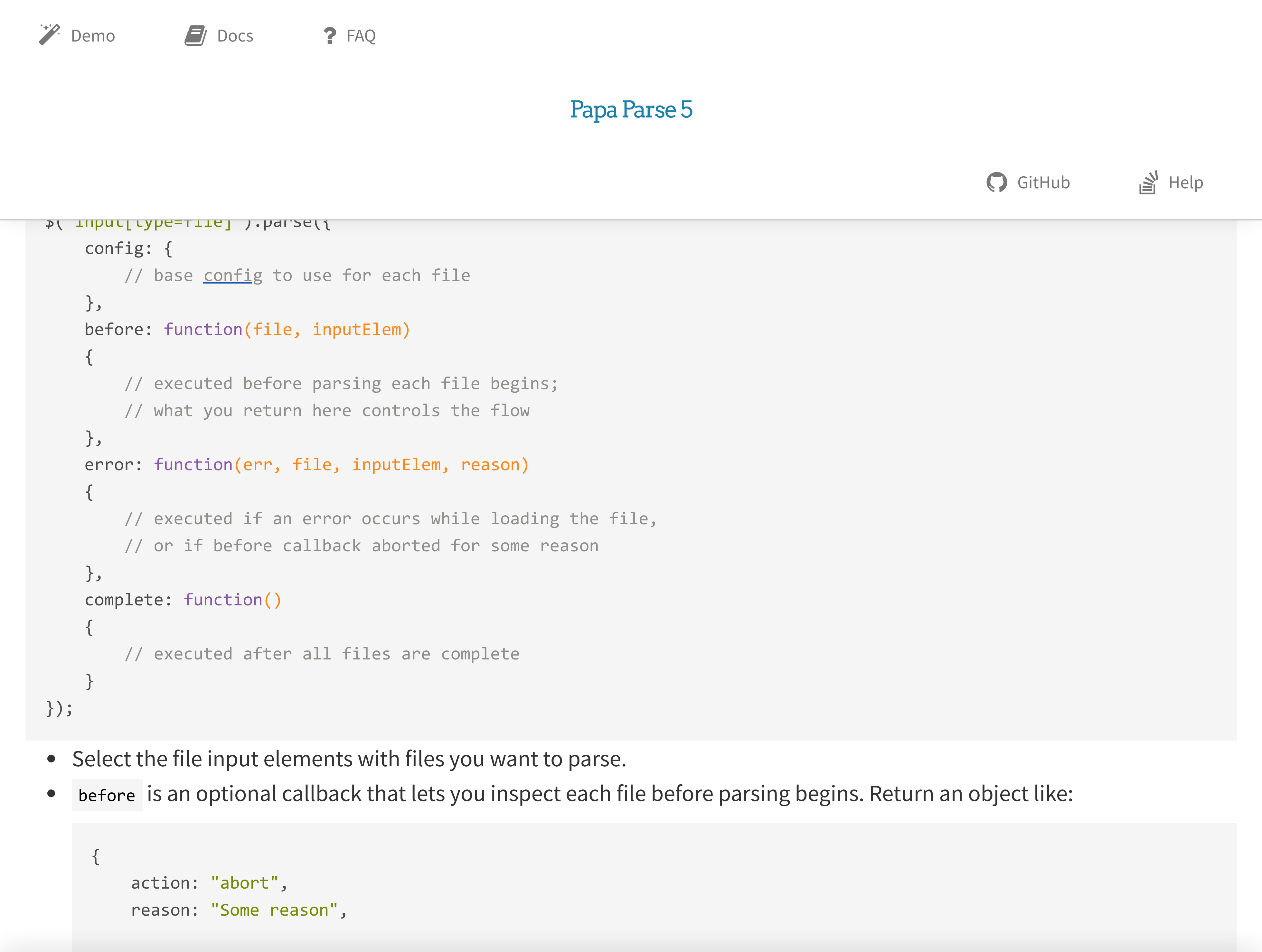
```
Papa.parse(file, config)
```

- `file` is a File object obtained from the DOM.
- `config` is a config object which contains a callback.
- Doesn't return anything. Results are provided asynchronously to a callback function.

PARSE REMOTE FILE

```
Papa.parse(url, {
    download: true,
    // rest of config ...
})
```

- `url` is the path or URL to the file to download.
- The second argument is a config object where `download: true` is set.



```
$('input[type=file]').parse({
    config: {
        // base config to use for each file
    },
    before: function(file, inputElem)
    {
        // executed before parsing each file begins;
        // what you return here controls the flow
    },
    error: function(err, file, inputElem, reason)
    {
        // executed if an error occurs while loading the file,
        // or if before callback aborted for some reason
    },
    complete: function()
    {
        // executed after all files are complete
    }
});
```

- Select the file input elements with files you want to parse.
- `before` is an optional callback that lets you inspect each file before parsing begins. Return an object like:

```
{
    action: "abort",
    reason: "Some reason",
```

callback in config for per-file results.

Convert JSON to CSV

Papa's `unparse` utility writes out correct delimited text strings given an array of arrays or an array of objects.

UNPARSE

```
Papa.unparse(data[, config])
```

- Returns the resulting delimited text as a string.
- `data` can be one of:

DEFAULT UNPARSE CONFIG WITH ALL OPTIONS

```
{
    quotes: false, //or array of booleans
    quoteChar: '"',
    escapeChar: '"',
    delimiter: ",",
    header: true,
    newline: "\r\n",
    skipEmptyLines: false, //other option is 'greedy', meaning skip delimiters, quotes, and whitespace.
    columns: null //or array of strings
}
```

UNPARSE CONFIG OPTIONS

Option	Explanation

also takes precedence over this.

`quoteChar`	The character used to quote fields.
`escapeChar`	The character used to escape `quoteChar` inside field values.
`delimiter`	The delimiting character. Multi-character delimiters are supported. It must not be found in Papa.BAD_DELIMITERS.
`header`	If `false` , will omit the header row. If `data` is an array of arrays this option is ignored. If `data` is an array of objects the keys of the first object are the header row. If `data` is an object with the keys `fields` and `data` the `fields` are the header row.
`newline`	The character used to determine newline sequence. It defaults to `"\r\n"` .

	the objects. If not set the keys of the first objects are used as column.
`escapeFormulae`	If `true`, field values that begin with `=`, `+`, `-`, `@`, `\t`, or `\r`, will be prepended with a `'` to defend against injection attacks, because Excel and LibreOffice will automatically parse such cells as formulae. You can override those values by setting this option to a regular expression

EXAMPLES

```
// Two-line, comma-delimited file
var csv = Papa.unparse([
    ["1-1", "1-2", "1-3"],
    ["2-1", "2-2", "2-3"]
]);

// With implicit header row
// (keys of first object populate header row)
var csv = Papa.unparse([
    {
        "Column 1": "foo",
        "Column 2": "bar"
```

```
var csv = Papa.unparse({
    "fields": ["Column 1", "Column 2"],
    "data": [
        ["foo", "bar"],
        ["abc", "def"]
    ]
});
```

The Parse Config Object

The `parse` function may be passed a configuration object. It defines settings, behavior, and callbacks used during parsing. Any properties left unspecified will resort to their default values.

```
    transformHeader: undefined,
    dynamicTyping: false,
    preview: 0,
    encoding: "",
    worker: false,
    comments: false,
    step: undefined,
    complete: undefined,
    error: undefined,
    download: false,
    downloadRequestHeaders: undefined,
    downloadRequestBody: undefined,
    skipEmptyLines: false,
    chunk: undefined,
    chunkSize: undefined,
    fastMode: undefined,
    beforeFirstChunk: undefined,
    withCredentials: undefined,
    transform: undefined,
    delimitersToGuess: [',', '\t', '|', ';', Papa.RECORD_SEP, Papa.UNIT_SEP],
    skipFirstNLines: 0
}
```

CONFIG OPTIONS

	must return a string which will be used as delimiter. In both cases it cannot be found in Papa.BAD_DELIMITERS.
`newline`	The newline sequence. Leave blank to auto-detect. Must be one of \r, \n, or \r\n.
`quoteChar`	The character used to quote fields. The quoting of all fields is not mandatory. Any field which is not quoted will correctly read.
`escapeChar`	The character used to escape the quote character within a field. If not set, this option will default to the value of `quoteChar`, meaning that the default escaping of quote character within a quoted field is using the quote character two times. (e.g. `"column with ""quotes"" in text"`)
`header`	If true, the first row of parsed data will be interpreted as field names. An array of field names will be returned in meta, and each row of data will be an object of values keyed by field name instead of a simple array. Rows with a different number of fields from the header row will produce an error. Warning: Duplicated field names will be automatically renamed to avoid values in previous fields having the same name to be overwritten. Renamed fields with original (or transformed by `transformHeader`) are stored in `ParseResult.meta.renamedHeaders`

	data must conform to the definition of a decimal literal. Numerical values greater than `2^53` or less than `-2^53` will not be converted to numbers to preserve precision. European-formatted numbers must have commas and dots swapped. It also accepts an object or a function. If it's an object, its values should be a boolean to indicate if dynamic typing should be applied for each column number (or header name if using headers). If it's a function, it should return a boolean value for each field number (or name if using headers) which will be passed as first argument.
`preview`	If > 0, only that many rows will be parsed.
`encoding`	The encoding to use when opening local files. If specified, it must be a value supported by the FileReader API.
`worker`	Whether or not to use a worker thread. Using a worker will keep your page reactive, but may be slightly slower.
`comments`	A string that indicates a comment (for example, "#" or "//"). When Papa encounters a line starting with this string, it will skip the line.

Streaming is necessary for large files which would otherwise crash the browser. You can call `parser.abort()` to abort parsing. And, except when using a Web Worker, you can call `parser.pause()` to pause it, and `parser.resume()` to resume.

`complete`

The callback to execute when parsing is complete. It receives the parse results. If parsing a local file, the File is passed in, too:

```
complete: function(results, file) {
	console.log("Parsing complete:", results, file);
}
```

When streaming, parse results are *not* available in this callback.

`error`

A callback to execute if FileReader encounters an error. The function is passed two arguments: the error and the File.

`download`

If true, this indicates that the string you passed as the first argument to `parse()` is actually a URL from which to download a file and parse its contents.

`downloadRequestBody`	Use POST request on the URL of the download option. The value passed will be set as the body of the request.
`skipEmptyLines`	If true, lines that are completely empty (those which evaluate to an empty string) will be skipped. If set to `'greedy'`, lines that don't have any content (those which have only whitespace after parsing) will also be skipped.
`chunk`	A callback function, identical to step, which activates streaming. However, this function is executed after every *chunk* of the file is loaded and parsed rather than every row. Works only with local and remote files. Do not use both chunk and step callbacks together. For the function signature, see the documentation for the step function.
`chunkSize`	Overrides `Papa.LocalChunkSize` and `Papa.RemoteChunkSize`. See configurable section to know the usage of both parameters.

	function receives as an argument the chunk about to be parsed, and it may return a modified chunk to parse. This is useful for stripping header lines (as long as the header fits in a single chunk).
`withCredentials`	A boolean value passed directly into XMLHttpRequest's "withCredentials" property.
`transform`	A function to apply on each value. The function receives the value as its first argument and the column number or header name when enabled as its second argument. The return value of the function will replace the value it received. The transform function is applied before dynamicTyping.
`delimitersToGuess`	An array of delimiters to guess from if the `delimiter` option is not set.
`skipFirstNLines`	To skip first N number of lines when converting a CSV file to JSON

A parse result always contains three objects: data, errors, and meta. Data and errors are arrays, and meta is an object. In the step callback, the data array will only contain one element.

RESULT STRUCTURE

```
{
    data:   // array of parsed data
    errors: // array of errors
    meta:   // object with extra info
}
```

- `data` is an array of rows. If header is false, rows are arrays; otherwise they are objects of data keyed by the field name.
- `errors` is an array of errors.
- `meta` contains extra information about the parse, such as delimiter used, the newline sequence, whether the process was aborted, etc. Properties in this object are not guaranteed to exist in all situations.

DATA

```
// Example (header: true)
[
    {
        "Column 1": "foo",
        "Column 2": "bar",
        "Column 1": "foo1",
    },
    {
        "Column 1": "abc",
        "Column 2": "def",
        "Column 1": "abc1",
    }
]
```

- If header row is enabled and more fields are found on a row of data than in the header row, an extra field will appear in that row called `__parsed_extra`. It contains an array of all data parsed from that row that extended beyond the header row.

ERRORS

```
// Error structure
{
    type: "",      // A generalization of the error
```

error type).

- Just because errors are generated does not necessarily mean that parsing failed. The worst error you can get is probably MissingQuotes.

META

```
{
    delimiter: // Delimiter used
    linebreak: // Line break sequence used
    aborted:   // Whether process was aborted
    fields:    // Array of field names
    truncated: // Whether preview consumed all input
    renamedHeaders: // Headers that are automatically renamed by the library to avoid duplication. {Column 1_1: 'Column
}
```

- Not all meta properties will always be available. For instance, `fields` is only given when header row is enabled.

There's a few other things that Papa exposes to you that weren't explained above.

READ-ONLY

Read-Only Property	Explanation
`Papa.BAD_DELIMITERS`	An array of characters that are not allowed as delimiters (`\r, \n, ", \ufeff`).
`Papa.RECORD_SEP`	The true delimiter. Invisible. ASCII code 30. Should be doing the job we strangely rely upon commas and tabs for.
`Papa.UNIT_SEP`	Also sometimes used as a delimiting character. ASCII code 31.
`Papa.WORKERS_SUPPORTED`	Whether or not the browser supports HTML5 Web Workers. If false, `worker: true` will have no effect.

	local computer. Default 10 MB.
`Papa.RemoteChunkSize`	Same as LocalChunkSize, but for downloading files from remote locations. Default 5 MB.
`Papa.DefaultDelimiter`	The delimiter used when it is left unspecified and cannot be detected automatically. Default is comma.

DOWNLOAD

Latest (master)

Lil' Papa

Fat Papa

COMMUNITY

Twitter

Stack Overflow