WLParser

Version 0.3.1

LOLH

Table of Contents

Introduction

The program parses worklog daily and yearly files into an object.

1 Install the WLParser Package

The following shell scripts install the project's `package.json` file, install `jest` as a development dependency, install all `node` dependencies, and finally adds a '`test`' script to the `package.json`.

1.1 Install `package.json`

```
yarn init -yp
yarn add --dev jest
```

Listing 1.1: Install the WLParser Package and Dependencies

```
yarn init v1.19.0
success Saved package.json
Done in 0.04s.
yarn add v1.19.0
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 335 new dependencies.
info Direct dependencies
 jest@24.9.0
info All dependencies
 @babel/generator@7.6.4
 @babel/helper-function-name@7.1.0
 @babel/helper-get-function-arity@7.0.0
 @babel/helper-split-export-declaration@7.4.4
 @babel/helpers@7.6.2
 @babel/highlight@7.5.0
 @babel/plugin-syntax-object-rest-spread@7.2.0
 @babel/template@7.6.0
 @babel/traverse@7.6.3
 @cnakazawa/watch@1.0.3
 @jest/core@24.9.0
 @jest/reporters@24.9.0
 @jest/test-sequencer@24.9.0
 @types/babel__core@7.1.3
 @types/babel__generator@7.6.0
 @types/babel__template@7.0.2
 @types/babel__traverse@7.0.7
 @types/istanbul-lib-report@1.1.1
 @types/istanbul-reports@1.1.1
 @types/stack-utils@1.0.1
 @types/yargs-parser@13.1.0
```

```
abbrev@1.1.1
acorn-globals@4.3.4
acorn-walk@6.2.0
acorn@5.7.3
ajv@6.10.2
ansi-regex@4.1.0
aproba@1.2.0
are-we-there-yet@1.1.5
arr-flatten@1.1.0
array-equal@1.0.0
asn1@0.2.4
assign-symbols@1.0.0
astral-regex@1.0.0
async-limiter@1.0.1
asynckit@0.4.0
atob@2.1.2
aws-sign2@0.7.0
aws4@1.8.0
babel-jest@24.9.0
babel-plugin-jest-hoist@24.9.0
babel-preset-jest@24.9.0
balanced-match@1.0.0
base@0.11.2
bcrypt-pbkdf@1.0.2
brace-expansion@1.1.11
braces@2.3.2
browser-process-hrtime@0.1.3
browser-resolve@1.11.3
bser@2.1.0
buffer-from@1.1.1
cache-base@1.0.1
camelcase@5.3.1
capture-exit@2.0.0
caseless@0.12.0
chalk@2.4.2
chownr@1.1.3
ci-info@2.0.0
class-utils@0.3.6
cliui@5.0.0
co@4.6.0
code-point-at@1.1.0
collection-visit@1.0.0
color-convert@1.9.3
color-name@1.1.3
combined-stream@1.0.8
commander@2.20.0
concat-map@0.0.1
```

```
console-control-strings@1.1.0
convert-source-map@1.6.0
copy-descriptor@0.1.1
core-util-is@1.0.2
cross-spawn@6.0.5
cssom@0.3.8
cssstyle@1.4.0
dashdash@1.14.1
data-urls@1.1.0
debug@4.1.1
decamelize@1.2.0
decode-uri-component@0.2.0
deep-extend@0.6.0
deep-is@0.1.3
delayed-stream@1.0.0
delegates@1.0.0
detect-libc@1.0.3
detect-newline@2.1.0
diff-sequences@24.9.0
domexception@1.0.1
ecc-jsbn@0.1.2
emoji-regex@7.0.3
end-of-stream@1.4.4
error-ex@1.3.2
es-abstract@1.15.0
es-to-primitive@1.2.0
escape-string-regexp@1.0.5
escodegen@1.12.0
esprima@3.1.3
estraverse@4.3.0
expand-brackets@2.1.4
extend@3.0.2
extglob@2.0.4
extsprintf@1.3.0
fast-deep-equal@2.0.1
fast-levenshtein@2.0.6
fill-range@4.0.0
for-in@1.0.2
forever-agent@0.6.1
form-data@2.3.3
fs-minipass@1.2.7
fs.realpath@1.0.0
fsevents@1.2.9
gauge@2.7.4
get-caller-file@2.0.5
get-stream@4.1.0
getpass@0.1.7
```

```
globals@11.12.0
graceful-fs@4.2.2
growly@1.3.0
handlebars@4.4.3
har-schema@2.0.0
har-validator@5.1.3
has-unicode@2.0.1
has-value@1.0.0
has@1.0.3
hosted-git-info@2.8.5
html-encoding-sniffer@1.0.2
http-signature@1.2.0
iconv-lite@0.4.24
ignore-walk@3.0.3
imurmurhash@0.1.4
inflight@1.0.6
inherits@2.0.4
ini@1.3.5
invariant@2.2.4
is-accessor-descriptor@1.0.0
is-arrayish@0.2.1
is-data-descriptor@1.0.0
is-date-object@1.0.1
is-descriptor@1.0.2
is-fullwidth-code-point@2.0.0
is-generator-fn@2.1.0
is-plain-object@2.0.4
is-regex@1.0.4
is-stream@1.1.0
is-symbol@1.0.2
is-typedarray@1.0.0
is-windows@1.0.2
is-wsl@1.1.0
isarray@1.0.0
isexe@2.0.0
isstream@0.1.2
istanbul-lib-instrument@3.3.0
istanbul-lib-report@2.0.8
istanbul-lib-source-maps@3.0.6
istanbul-reports@2.2.6
jest-changed-files@24.9.0
jest-cli@24.9.0
jest-docblock@24.9.0
jest-each@24.9.0
jest-environment-jsdom@24.9.0
jest-environment-node@24.9.0
jest-leak-detector@24.9.0
```

```
jest-pnp-resolver@1.2.1
jest-resolve-dependencies@24.9.0
jest-serializer@24.9.0
jest-watcher@24.9.0
jest@24.9.0
js-tokens@4.0.0
jsdom@11.12.0
jsesc@2.5.2
json-parse-better-errors@1.0.2
json-schema-traverse@0.4.1
json-schema@0.2.3
json-stringify-safe@5.0.1
json5@2.1.1
jsprim@1.4.1
kind-of@3.2.2
kleur@3.0.3
left-pad@1.3.0
leven@3.1.0
levn@0.3.0
load-json-file@4.0.0
locate-path@3.0.0
lodash@4.17.15
loose-envify@1.4.0
makeerror@1.0.11
map-visit@1.0.0
merge-stream@2.0.0
mime-db@1.40.0
mime-types@2.1.24
minimatch@3.0.4
minimist@1.2.0
minipass@2.9.0
minizlib@1.3.3
mixin-deep@1.3.2
mkdirp@0.5.1
ms@2.1.2
nan@2.14.0
nanomatch@1.2.13
natural-compare@1.4.0
needle@2.4.0
neo-async@2.6.1
nice-try@1.0.5
node-int64@0.4.0
node-modules-regexp@1.0.0
node-notifier@5.4.3
node-pre-gyp@0.12.0
nopt@4.0.1
normalize-package-data@2.5.0
```

```
normalize-path@2.1.1
npm-bundled@1.0.6
npm-packlist@1.4.6
npm-run-path@2.0.2
npmlog@4.1.2
number-is-nan@1.0.1
nwsapi@2.1.4
oauth-sign@0.9.0
object-assign@4.1.1
object-copy@0.1.0
object-inspect@1.6.0
object-keys@1.1.1
object.getownpropertydescriptors@2.0.3
once@1.4.0
optimist@0.6.1
optionator@0.8.2
os-homedir@1.0.2
os-tmpdir@1.0.2
osenv@0.1.5
p-each-series@1.0.0
p-finally@1.0.0
p-limit@2.2.1
p-locate@3.0.0
p-reduce@1.0.0
p-try@2.2.0
parse-json@4.0.0
parse5@4.0.0
pascalcase@0.1.1
path-exists@3.0.0
path-is-absolute@1.0.1
path-key@2.0.1
path-parse@1.0.6
path-type@3.0.0
performance-now@2.1.0
pirates@4.0.1
pkg-dir@3.0.0
pn@1.1.0
posix-character-classes@0.1.1
process-nextick-args@2.0.1
prompts@2.2.1
psl@1.4.0
pump@3.0.0
qs@6.5.2
rc@1.2.8
react-is@16.10.2
read-pkg-up@4.0.0
read-pkg@3.0.0
```

```
readable-stream@2.3.6
remove-trailing-separator@1.1.0
repeat-element@1.1.3
request-promise-core@1.1.2
request-promise-native@1.0.7
request@2.88.0
require-directory@2.1.1
resolve-cwd@2.0.0
resolve-from@3.0.0
resolve-url@0.2.1
resolve@1.12.0
ret@0.1.15
rimraf@2.7.1
rsvp@4.8.5
safer-buffer@2.1.2
sane@4.1.0
sax@1.2.4
semver@5.7.1
set-blocking@2.0.0
set-value@2.0.1
shebang-command@1.2.0
shebang-regex@1.0.0
shellwords@0.1.1
signal-exit@3.0.2
sisteransi@1.0.3
snapdragon-node@2.1.1
snapdragon-util@3.0.1
source-map-resolve@0.5.2
source-map-support@0.5.13
source-map-url@0.4.0
spdx-correct@3.1.0
spdx-exceptions@2.2.0
split-string@3.1.0
sshpk@1.16.1
stack-utils@1.0.2
static-extend@0.1.2
stealthy-require@1.1.1
string_decoder@1.1.1
string.prototype.trimleft@2.1.0
string.prototype.trimright@2.1.0
strip-eof@1.0.0
strip-json-comments@2.0.1
symbol-tree@3.2.4
tar@4.4.13
test-exclude@5.2.3
tmpl@1.0.4
to-fast-properties@2.0.0
```

```
 to-object-path@0.3.0
 to-regex-range@2.1.1
 tough-cookie@2.5.0
 tunnel-agent@0.6.0
 tweetnacl@0.14.5
 uglify-js@3.6.2
 union-value@1.0.1
 unset-value@1.0.0
 uri-js@4.2.2
 urix@0.1.0
 use@3.1.1
 util-deprecate@1.0.2
 util.promisify@1.0.0
 uuid@3.3.3
 validate-npm-package-license@3.0.4
 verror@1.10.0
 w3c-hr-time@1.0.1
 walker@1.0.7
 whatwg-encoding@1.0.5
 whatwg-mimetype@2.3.0
 whatwg-url@6.5.0
 which-module@2.0.0
 which@1.3.1
 wide-align@1.1.3
 wordwrap@0.0.3
 wrap-ansi@5.1.0
 write-file-atomic@2.4.1
 ws@5.2.2
 xml-name-validator@3.0.0
 y18n@4.0.0
 yallist@3.1.1
 yargs-parser@13.1.1
Done in 7.49s.
```

1.2 Add a 'test' Script to package.json and Update Version

Figure out how to dynamically update the version number; using a macro does not seem to work. This package uses Jest to test the modules. At this point, also change the version number.

```
const fs = require('fs');
const package = JSON.parse(fs.readFileSync('package.json', 'utf8'));
package.version = "0.3.1"
package.main = "index.js";
package.scripts = {"tests": "jest"};
fs.writeFileSync('package.json', JSON.stringify(package, null, 2));
```

Listing 1.2: Add *tests* script to package.json

```
cat package.json
```

Listing 1.3: The `package.json` file with added *tests* script

```
{
  "name": "wlparser",
  "version": "0.3.1",
  "main": "index.js",
  "repository": "ssh://git@github.com/wlharvey4/wlparser.git",
  "author": "Wesley Harvey <pinecone062@gmail.com>",
  "license": "MIT",
  "private": true,
  "devDependencies": {
    "jest": "^24.9.0"
  },
  "scripts": {
    "tests": "jest"
  }
}
```

1.3 Add an `index.js` Package File

```
1  /* WLParser/index.js */
2
3  module.exports = {
4      WLRecord: require('./lib/wlrecord'),
5      WLReader: require('./lib/wlreader'),
6      WLType:   require('./lib/wltype'),
7      WLChecks: require('./lib/wlchecks')
8  };
```

Listing 1.4: WLParser index.js File

1.3.1 TODO Auto-Update the Version Number

Figure out a way to auto-update the version number above.

2 The WLRecord Module

A proper worklog record conforms to the following structure:

```
1. datetime start: 2019-01-31T10:54:00
2. \tcase: 190301
3. \t\tSUBJECT --- VERB
4. \t\t\tTYPE: e.g. TIME | EXPENSE | ...
5. \s-------------------------------------------------------------------------
6. \sDETAIL : possible multiple lines in length
   \sDETAIL ...
7. \s-------------------------------------------------------------------------
8. datetime end: 2019-01-31T11:00:00
9. record separator: \n
```

Note: Record entries must be separated by a blank line.

2.1 Implement the WLRE Configuration File

The following code sets up the individual regular expressions for each component of a worklog entry. The export object is the combined regular expression for the entire worklog entry.

If an entry requires special processing at the time of parsing, as does the '`message`' entry below, which needs newlines stripped, place the component name in an array in position 0, and a lambda expression to run the entry through in position 1. The Wl$_r$*ecord* will look for an object in each component entry, and if one is found, run the entry through the lambda expression.

Note: The true structure of the 2nd, 3rd, and 4th lines include one, two and three tabs, respectively, at the beginning of the lines, but for the sake of avoiding problems with different editors handling SPACES vs TABS, I am using only SPACES, and not TABS.

```
 1  /* wlre.js */
 2
 3  /* KEYS OF A WLRECORD OBJECT */
 4  const WL_COMPONENTS = [
 5    'original',
 6    'start_date',
 7    'start_time',
 8    'caseno',
 9    'subject',
10    'verb',
11    'type',
12    ['message', m => m.replace(/\n/gm, '')],
13    'end_date',
14    'end_time',
15    'record_sep'
16  ];
17
18  /* REGEXP COMPONENTS OF A WORKLOG RECORD */
19  const DATETIME_RE = '^(\\d{4}-\\d{2}-\\d{2})T(\\d{2}:\\d{2}:00)$\\n';
20  const CASE_RE     = '^\\s+(.{6})$\\n';
21  const SUBJ_VERB_RE= '^\\s+(.*?) --- (.*?)$\\n';
22  const TYPE_RE     = '^\\s+(.*?)$\\n';
23  const MESSAGE_RE  = '^\\s-{78}$\\n^\\s(.*?)^\\s-{78}$\\n';
24  const RECORD_SEP  = '(^$)';
25  const FLAGS       = 'ms';
26
27  /* COMBINED REGULAR EXPRESSION FOR ENTIRE WORKLOG ENTRY */
28  const wlre = new RegExp(
29    DATETIME_RE  +
30    CASE_RE      +
31    SUBJ_VERB_RE +
32    TYPE_RE      +
33    MESSAGE_RE   +
34    DATETIME_RE  +
35    RECORD_SEP,
36    FLAGS
37  );
38
39  module.exports = {wlre, WL_COMPONENTS};
```

Listing 2.1: WLRE Configuration File for a WLRecord

2.2 Create the WLRecord Module

The following class receives a worklog record as a string and uses the `wlre` module to parse and return its components. Note that the parser must check each $\mathrm{WL}_{C}OMPONENT$

entry for an object (array in this case), and if one is found, then it runs the entry through the supplied lambda expression in the array's second position.

```
/* wlrecord.js */

const {wlre, WL_COMPONENTS} = require('./wlre.js');

class WLRecord {

    constructor(wlrecord) {
        const parsed = wlre.exec(wlrecord);

        if (!parsed) {
            throw ReferenceError(`wlrecord:\n${wlrecord}\nfailed to parse`)
        }

         parsed.forEach((e,i) => {
            let key = WL_COMPONENTS[i];
            if (typeof key === 'object') {  // see WL_COMPONENTS[7]
                e = WL_COMPONENTS[i][1](e); // run e through the supplied lambda funct
                key = WL_COMPONENTS[i][0];  // use the string key
            }

            this[key] = e;
        });
    }

    get entry() {
        return this;// Object.assign({}, this._record);
    }
}

module.exports = WLRecord;
```

Listing 2.2: The WLRecord Module

2.3 Test the WLRecord Module

NOTE: The following sample records sometimes shift to the right by a couple of spaces upon a save of the file. If a test record fails to parse, make sure there is no space before either datetime and there is one space only before the message detail and enclosing dotted lines.

```
1      const Parser = require('../lib/wlrecord');
2
3      const STR=
4  `2019-01-02T10:30:00
5             180704
6                     STATUS UPDATE --- SW PHONE
```

```
                          TIME
 ------------------------------------------------------------------------------
 Received call from SW; Client has been removed from his housing; SW is
 picking him up and will be finding a new place for him to stay.
 ------------------------------------------------------------------------------
2019-01-02T10:36:00

2019-01-02T14:00:00
        180203
                NEGOTIATION --- CLIENT PHONE
                        TIME
 ------------------------------------------------------------------------------
 Spoke with Client about terms of counter-offer.  Will propose flat amount of
 $900 per month.
 ------------------------------------------------------------------------------
2019-01-02T14:30:00

`;

describe('The WLRecord', () => {
    test('throws with an invalid entry', () => {
        expect(() => {
            new Parser('abc')
        }).toThrow(ReferenceError);
    });
    test('does not throw with a valid entry', () => {
        expect(() => {
            new Parser(STR);
        }).not.toThrow();
    });
});
```

2.4 Check the `wlre` Regular Expression Implementation

```
const {wlre, WL_COMPONENTS} = require('../lib/wlre.js');

const STR=`2019-01-02T10:30:00
        180704
                STATUS UPDATE --- SW PHONE
                        TIME
 ---------------------------------------------------------------------------
 Received call from SW; Client has been removed from his housing; SW is
 picking him up and will be finding a new place for him to stay.
 ---------------------------------------------------------------------------
2019-01-02T10:36:00

2019-01-02T14:00:00
        180203
                NEGOTIATION --- CLIENT PHONE
                        TIME
 ---------------------------------------------------------------------------
 Spoke with Client about terms of counter-offer.  Will propose flat amount of
 $900 per month.
 ---------------------------------------------------------------------------
 2019-01-02T14:30:00

`;

const result = wlre.exec(STR);

const start_date = RegExp.$1;
const start_time = RegExp.$2;
const caseno     = RegExp.$3;
const subject    = RegExp.$4;
const verb       = RegExp.$5;
const type       = RegExp.$6;
const message    = RegExp.$7;
const end_date   = RegExp.$8;
const end_time   = RegExp.$9;
const record_sep = result[10];

const mess_sep_re = /\n/gm;
const mess = message.replace(mess_sep_re, '');

console.log(STR)
console.log('--------------');
console.log(`Start: ${start_date} T ${start_time}\nCase No: ${caseno}\nSubject: ${
console.log(`${mess}`);
console.log(`End: ${end_date} T ${end_time}`);
console.log(`RS: ${record_sep}`);
console.log('--------------');
```

Listing 2.3: Check WLRE

3 The WLReader Module

The Log Reader is a Stream Reader that reads records from a log file given the name of the log file and a record separator. The Reader emits a signal with each record read. The Reader emits a finished signal at the conclusion of reading all records.

3.1 Implementing the WLReader Module

```
 1   /* wlreader.js */
 2
 3 const EventEmitter = require('events').EventEmitter;
 4 const fs       = require('fs');
 5 const path     = require('path');
 6 const rl       = require('readline');
 7 const TODAY    = new Date();
 8 const YEAR     = TODAY.getUTCFullYear();
 9 const MIN_YEAR = 2016;
10 const REC_SEP  = /^$/;
11 const WORKLOG  = process.env.WORKLOG;
12 if (!WORKLOG)
13     throw new ReferenceError('Environment variable WORKLOG is undefined.');
14
15 class WLReader extends EventEmitter {
16     constructor(wl_year, rec_sep=REC_SEP) {
17         if (typeof wl_year !== 'number' ||
18             wl_year < MIN_YEAR          ||
19             wl_year > YEAR)
20             throw new RangeError(`Year '${wl_year}' must be between ${MIN_YEAR} an
21
22         if (!(rec_sep instanceof RegExp))
23             throw new AssertionError(`The record separator ('${rec_sep}') should b
24
25         super();
26
27         this._logfile = path.format({
28             dir: WORKLOG,
29             name: `worklog.${wl_year}`,
30             ext: '.otl'
31         });
32
33         this._rec_sep = rec_sep;
34         this._entry = '';
35
36         if (!fs.existsSync(this._logfile)) {
37             throw new Error(`Logfile '${this._logfile}' does not exist`);
38             process.exit(1);
39         }
40
41         this._rs = fs.createReadStream(this._logfile, {
42             encoding: 'utf8',
43             emitClose: true,
44         });
45     }
46
47     read() {
48         const rl_interface = rl.createInterface({
49             input: this._rs
50         });
51
```

3.2 Testing the WLReader Module

```
 1  /* log_reader.test.js */
 2
 3  const path    = require('path');
 4  const WLR     = require('../lib/wlreader');
 5  const WLRecord= require('../lib/wlrecord');
 6  const WORKLOG = process.env.WORKLOG;
 7  const YEAR    = (new Date()).getUTCFullYear();
 8  const REC_SEP = /^$/;
 9
10  describe('The WLReader', () => {
11      test('throws an error when the year is too early', () => {
12          expect(() => {
13              new WLR(2000, REC_SEP);
14          }).toThrow(RangeError);
15      });
16      test('throws an error when the year is in the future', () => {
17          expect(() => {
18              new WLR(YEAR+1, REC_SEP);
19          }).toThrow(RangeError);
20      });
21      test('reads a log file', () => {
22          expect(() => {
23              new WLR(2019, REC_SEP)
24          }).not.toThrow();
25      });
26      test('reads a log file using a default record separator', () => {
27          expect(() => {
28              new WLR(2016)
29          }).not.toThrow();
30      });
31      test('prints a log file', done => {
32          let entry;
33          const wlr = new WLR(YEAR);
34          wlr.on('entry', entry => {
35              entry = new WLRecord(entry);
36          }).on('done', done);
37
38          wlr.read();
39      });
40  });
```

Listing 3.2: Testing the WLReader

4 The WLType Module

The WLType class receives each of the worklog records from the WLReader, parses the record using the WLRecord, and emits messages for each type of record found, such as TIME, EXPENSE, PAYMENT, etc, as well as for each record under the message '`entry`'. The record itself is sent with each message.

4.1 Implementing the WLType Module

```
 1  /* wltype.js */
 2
 3  const EventEmitter = require('events').EventEmitter;
 4  const WLReader = require('./wlreader');
 5  const WLRecord = require('./wlrecord');
 6  const REC_SEP  = /^$/;
 7
 8  class WLType extends EventEmitter {
 9      constructor(wl_year, rec_sep=REC_SEP) {
10          super();
11          this._wlreader = new WLReader(wl_year, rec_sep);
12      }
13
14      parse() {
15          this._wlreader.on('entry', wlrecord => {
16              const record = new WLRecord(wlrecord);
17              this.emit('record', record);
18              this.emit(record.type, record);
19          }).on('done', () => this.emit('parsed'));
20
21          this._wlreader.read();
22      }
23  }
24
25  module.exports = WLType;
```

Listing 4.1: Implementing the WLType module

4.2 Testing the WLType Module

```
 1  /* wltype.test.js */
 2
 3  const WLType = require('../lib/wltype');
 4  const {wlre, WL_COMPONENTS} = require('../lib/wlre');
 5  const YEAR     = (new Date()).getUTCFullYear();
 6  const REC_SEP  = /^$/;
 7
 8  const keys = WL_COMPONENTS.map(
 9      c =>
10          typeof c === 'object' ?
11          c[0]                  :
12          c
13  );
14
15  describe('The WLType Module initialization', () => {
16      it('the WLType class initializes', () => {
17          expect(new WLType(YEAR, REC_SEP)).toBeInstanceOf(WLType);
18      });
19  });
20
21  describe('A WLType instance', () => {
22      let data;
23      beforeEach(() => {
24          data = new WLType(YEAR);
25      });
26      it('produces the event for TIME', done => {
27          data.on('TIME', time_record => {
28              expect(time_record).toHaveProperty('type');
29              done();
30          });
31          data.parse();
32      });
33      it('has all the keys in WL_COMPONENTS', done => {
34          data.on('record', record => {
35              const record_keys = Object.keys(record);
36              expect(record_keys).toEqual(keys);
37              done();
38          });
39          data.parse();
40      });
41  });
```

Listing 4.2: Testing the WLType module

5 The WLChecks Module

Finding checks and their information is a bedeviling problem with the Worklog setup as it is. This module is designed to find checks, parse out their information, create an object, and emit a message with the check information. This module uses the WLType module to find types that contain check information, such as '`EXPENSE`' and '`TRUST`' payments.

5.1 Implementing the WLChecks Module

```
 1  /* wl_check.js */
 2
 3  const EventEmitter = require('events').EventEmitter;
 4  const WLType       = require('./wltype');
 5  const REC_SEP      = /^$/;
 6  const CHECK_RE     = /^(\$\d+[,]?\d*\.\d{2})\s?::\s?(.*?)\s?::\s?(.*?)\s?::\s?(\w+
 7
 8  /* types that could return checks:
 9     - TRUST withdrawals
10     - EXPENSE
11  */
12
13  class WLChecks extends EventEmitter {
14      constructor(wl_year, rec_sep=REC_SEP) {
15          super();
16          this._wltype = new WLType(wl_year, rec_sep);
17      }
18
19      findChecks() {
20          this._wltype.on('TRUST', trust_record => {
21              if (trust_record.verb === 'WITHDRAWAL') {
22                  this._parseCheck(trust_record);
23              }
24
25          }).on('EXPENSE', expense_record => {
26              this._parseCheck(expense_record);
27
28          }).on('parsed', () => {
29              this.emit('checked', this._checks);
30
31          }).on('error', err => {
32              console.error(`Received an error: ${err.message}`);
33              throw(err);
34
35          });
36
37          this._wltype.parse();
38      }
39
40      _parseCheck(record) {
41          let check_info;
42          let check_data = {};
43          if ((check_info = CHECK_RE.exec(record.message))) {
44              check_data.type = record.type;
45              check_data.start_date = record.start_date;
46              check_data.checkno = check_info[5];
47              check_data.payee   = check_info[2];
48              check_data.acct    = check_info[4];
49              check_data.amount  = parseFloat(check_info[1].replace(/^\$/, ''));
50              check_data.purpose = check_info[3];
51              check_data.subject = record.subject;
```

5.2 Test the WLChecks Module

```
 1  /* wlchecks.test.js */
 2
 3  const WLChecks = require('../lib/wlchecks');
 4  const YEAR     = new Date().getUTCFullYear();
 5  //const REC_SEP  = /^$/;
 6
 7  let checks;
 8
 9  describe('WLChecks', () => {
10      it('checks is a WLChecks', () => {
11          expect(new WLChecks(2016)).toBeInstanceOf(WLChecks);
12      });
13  });
14
15  describe('A check', () => {
16      it('has the proper components', done => {
17          const checks = new WLChecks(2017);
18          checks.on('check', checkCheck);
19          checks.on('checked', done);
20          checks.findChecks();
21          function checkCheck(check) {
22              console.log(check);
23              expect(check).toHaveProperty('type', expect.any(String));
24              expect(check).toHaveProperty('start_date', expect.any(String));
25              expect(check).toHaveProperty('checkno', expect.any(String));
26              expect(check).toHaveProperty('payee', expect.any(String));
27              expect(check).toHaveProperty('acct', expect.any(String));
28              expect(check).toHaveProperty('amount', expect.any(Number));
29              expect(check).toHaveProperty('purpose', expect.any(String));
30              expect(check).toHaveProperty('subject', expect.any(String));
31              expect(check).toHaveProperty('caseno', expect.any(String));
32          }
33      });
34  });
```

Listing 5.2: Test the WLChecks module

Appendix A Makefile

```
DOCS = docs

.PHONY:        clean clean-dist clean-world clean-prod
.PHONY:        install-docs install-info install-pdf open-pdf
.PHONY:        texi prod dev check

clean:
        -rm *~

clean-dist: clean
        -rm -rf $(DOCS) lib/ __tests__/ node_modules/

clean-world:        clean-dist
        -rm *.{info,texi,pdf,json,lock,js}

clean-prod: clean
        -rm *{.texi,org} lib/check.js Makefile
        -rm -rf node_modules/ __tests__/

# Create a directory ready to be saved as branch:prod
prod: dev install-docs clean-prod
        git checkout -B prod
        git add -A .
        git commit -m "branch:prod"
        git push -f --tags origin prod

dev: clean-world
        git checkout dev

# Create a docs/ directory and move the .info and .pdf files into it
install-docs: install-info install-pdf

texi: WLParser.texi
WLParser.texi: WLParser.org
        emacs -Q --batch WLParser.org \
        --eval='(setq org-export-use-babel nil)' \
        -f org-texinfo-export-to-texinfo

info install-info: $(DOCS)/WLParser.info
$(DOCS)/WLParser.info: WLParser.org | docs-dir
        emacs -Q --batch WLParser.org \
        --eval='(setq org-confirm-babel-evaluate nil)' \
        --eval='(require '\''ob-shell)' \
        --eval='(require '\''ob-js)' \
        --eval='(org-babel-tangle-file "WLParser.org")' \
        --eval='(org-texinfo-export-to-info)'
        mv -v WLParser.info $(dir $@)

# Install a pdf file into a docs/ dir
pdf install-pdf: $(DOCS)/WLParser.pdf
```

Concept Index