Natural Language to iCalendar Converter

COMP SCI 4TB3 PROJECT – GROUP 9

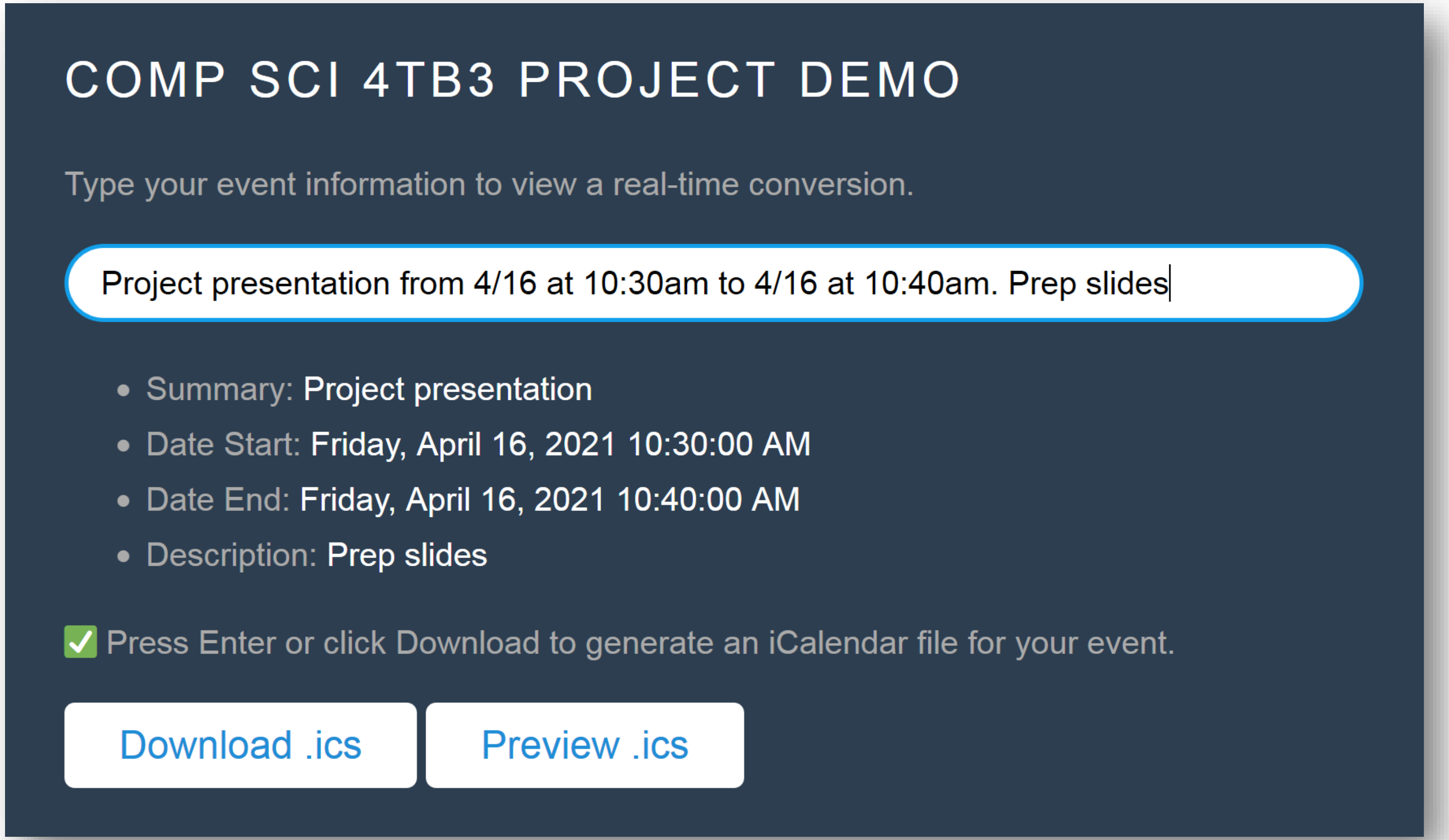
JOSHUA GUINNESS, ARKIN MODI, JASON KIM

APRIL 16, 2021

Natural Language to iCalendar Converter

Generates an iCalendar format event file from a natural language input.

Create new calendar events without filling separate fields as in a traditional calendar application.



COMP SCI 4TB3 PROJECT DEMO

Type your event information to view a real-time conversion.

Project presentation from 4/16 at 10:30am to 4/16 at 10:40am. Prep slides

- Summary: Project presentation
- Date Start: Friday, April 16, 2021 10:30:00 AM
- Date End: Friday, April 16, 2021 10:40:00 AM
- Description: Prep slides

✓ Press Enter or click Download to generate an iCalendar file for your event.

The iCalendar Format

Internet Calendaring and Scheduling Core Object Specification (.ics)

A standard for storing and exchanging calendar event data.

Supported by and can be imported into almost any calendar application.

"Discuss project by this monday at 4pm. Bring notes"

- SUMMARY: Discuss project
- DATE/TIME: Mon Apr 19 2021 4 PM
- DESCRIPTION: Bring notes

```
BEGIN:VCALENDAR
PRODID:Calendar
VERSION:2.0
BEGIN:VEVENT
UID:0@default
CLASS:PUBLIC
DTSTAMP;VALUE=DATE-TIME:20210413T042828
DTSTART;VALUE=DATE-TIME:20210419T160000
DTEND;VALUE=DATE-TIME:20210419T170000
SUMMARY;LANGUAGE=en-us:Discuss project
DESCRIPTION:Bring notes
TRANSP:TRANSPARENT
END:VEVENT
END:VCALENDAR
```

Implementation Details

Because there are many ways to provide event and date information through natural language, we have decided to limit the scope of acceptable input strings.

(**Event Summary**) (on | by | from | between) (**DateTime**) [. **Event Description**]

```
"Discuss project by this monday at 4pm. Bring notes"
```

Grammar

```
S ->                  Summary DateTime [". " Description]
Summary ->            [ Word ]+
DateTime ->           (' on ' | ' by ') AbsoluteDateTime | [' on ' | ' by ']
                      RelativeDateTime | (' from ' | 'between ') DateTimeRange
AbsoluteDateTime ->   ( ( DayOfMonth MonthName [Year] ) | ( [Year] MonthName DayOfMonth )
                      | DayOfMonth '/’ MonthNumber [ '/' Year ] )
                      [ 'at' ( AbsoluteTime | RelativeTime ) ]
RelativeDateTime ->   RelativeDate [ (' at ' | ' in the ') ( AbsoluteTime | RelativeTime ) ]
DateTimeRange ->      AbsoluteDateTime ( ' - ' | ' to ' | ' and ' ) AbsoluteDateTime
RelativeDate ->       'tomorrow' | 'today' | ( ( 'this' | 'next' ) DayOfWeek )
AbsoluteTime ->       HourTime [ ':' MinuteTime ] [ ' ' ] ( 'am'| 'pm' )
```

(cont'd)

Grammar

(cont'd)

```
DayOfWeek ->     'Mon' [ 'day' ] | ... | 'Sun' [ 'day' ]
DayOfMonth ->    1 | ... | 31
MonthNumber ->   1 | ... | 12
MonthName ->     'Jan' [ 'uary' ] | ... | 'Dec' [ 'ember' ]
Year ->          ( 2002 | ... | 2999 ) | ( 00 | ... | 99 )
RelativeTime ->  'morning' | 'noon' | 'afternoon' | 'evening' | 'night'
HourTime ->      1 | ... | 12
MinuteTime ->    1 | ... | 60
Description ->   [ Word ]*
Word ->          [ a-zA-Z0-9 ]+ | '!' | '?' | ' ' | '/' | '_' | ...
```

The Conversion Process

Natural Language Input

01

Split input into component substrings

02

Convert component substrings into matching iCalendar fields

03

Format output per iCalendar specification

iCalendar Format Output

01

Split input into component substrings

- Scan input string for optional date-description separator keyword
  - ". "
- Scan input string for summary-date separator keywords
  - " on " | " by " | " from " | " between "
- Scan date substring for optional date-time separators
  - " at " | " in the "
- Split the input string at separator positions, consuming the keywords
- Pass each component substring to its respective processing function

```
"Discuss project by this monday at 4pm. Bring notes
```

02

Convert component substrings into matching iCalendar fields

- Store event name component as the summary string
- Store the optional description component, if present

- Scan date substring for optional date-range separator keywords
  - " to " | " and " | "-"

- Split the date substring at date-range separator positions, creating separate start and end date substrings

- Check whether date is relative or absolute by pattern matching keywords
  - "today" | "tomorrow" | ["this" | "next"] ("mon" ["day"] | … | "sun" ["day"])

- Check whether time, if present, is relative or absolute by pattern matching
  - "morning" | "noon" | "afternoon" | "evening" | "night"

```
"Discuss project  this monday  4pm  Bring notes"
summary string  relative date  absolute time  desc. string
```

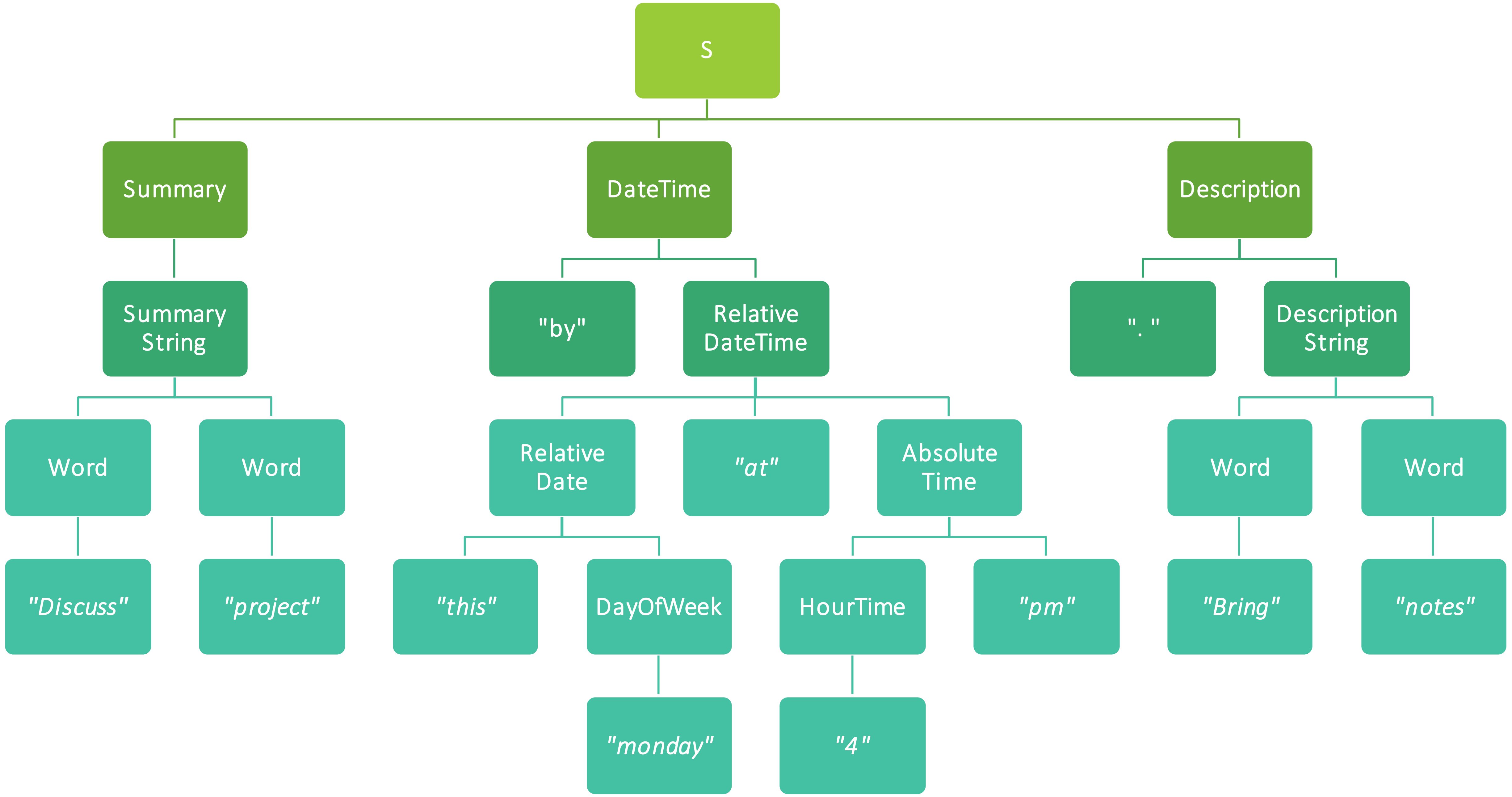
03

Format output per iCalendar specification

- Create date objects for event start and end dates
  - Single-day event if no end date
- Add time values to date objects
  - Default 1-hour event if no end time. All-day event if no start or end times
- For all-day events: adjust end date to midnight of next day, per iCal spec
- Validate existence of dates, check that start date precedes end date
- Build and output .ics file

"Discuss project this monday 4pm Bring notes"

◦ SUMMARY;LANGUAGE=en-us:Discuss project
◦ DTSTART;VALUE=DATE-TIME:20210419T160000
◦ DTEND;VALUE=DATE-TIME:20210419T170000
◦ DESCRIPTION:Bring notes



"Discuss project by this monday at 4pm. Bring notes"

Converter Implementation

WRITTEN IN JAVASCRIPT (ABOUT 500 LINES)

RUNS CLIENT-SIDE WITHIN ANY BROWSER

WORKS AS A JAVASCRIPT LIBRARY

Natural Language to iCalendar Converter

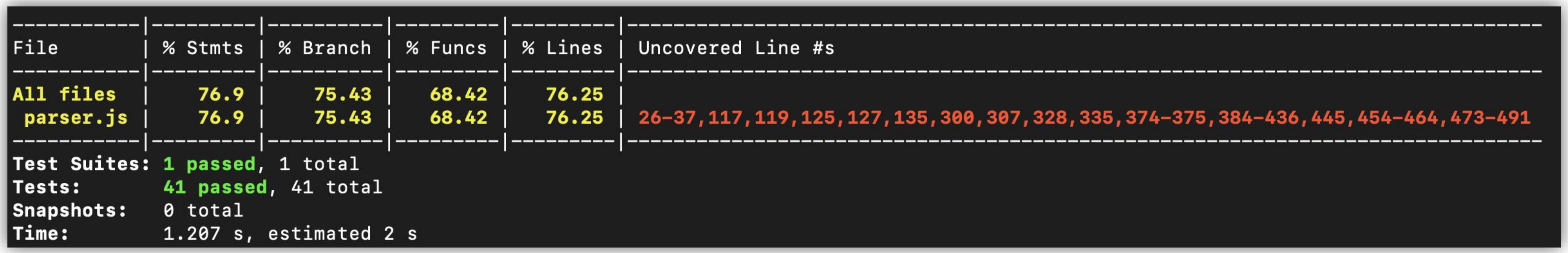
DEMONSTRATION

Automated Testing with Jest

Both acceptable and unacceptable inputs tested for valid event parsing and proper error handling

Up to 41 automated tests run between major code updates

Tests performed using Jest, a JavaScript testing framework



```
-----------|----------|-----------|----------|----------|-----------------------------------------------------------------------------
File       | % Stmts  | % Branch  | % Funcs  | % Lines  | Uncovered Line #s
-----------|----------|-----------|----------|----------|-----------------------------------------------------------------------------
All files  |     76.9 |     75.43 |    68.42 |    76.25 |
 parser.js |     76.9 |     75.43 |    68.42 |    76.25 | 26-37,117,119,125,127,135,300,307,328,335,374-375,384-436,445,454-464,473-491
-----------|----------|-----------|----------|----------|-----------------------------------------------------------------------------
Test Suites: 1 passed, 1 total
Tests:       41 passed, 41 total
Snapshots:   0 total
Time:        1.207 s, estimated 2 s
```

Unit Testing Results

Manual Testing

Ad-hoc manual tests were conducted throughout development using “real world” input strings via the demo page and browser developer tools

✓ “Complete project by Wed Apr 14 at 9:30 pm. Submit everything on Gitlab.”

- Accepted input because its structure is valid according to the grammar

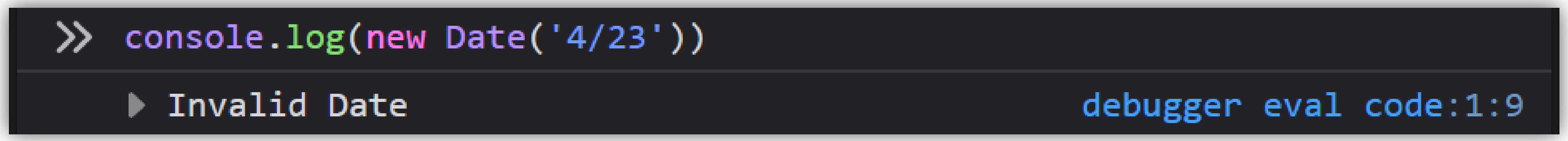
✗ “Complete project @ Apr14, 930p and submit everything on Gitlab”

- Unacceptable input due to invalid separator keywords and malformed date-time values

Development Difficulties

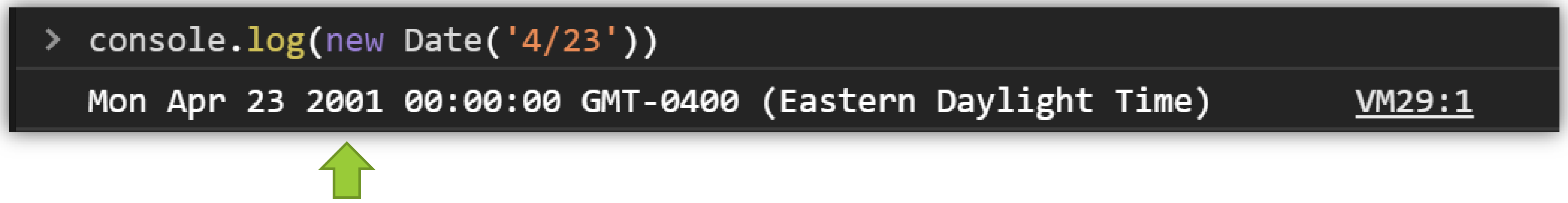
Built-in JavaScript date functionality is **implementation dependent** and varies between browsers

Mozilla Firefox - SpiderMonkey engine



```
>> console.log(new Date('4/23'))
   ▶ Invalid Date                                    debugger eval code:1:9
```

Google Chrome - V8 engine

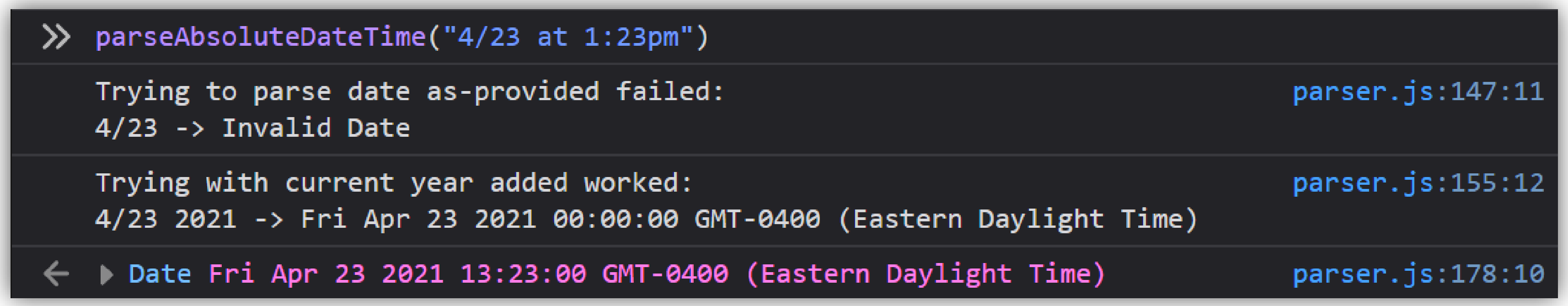


```
> console.log(new Date('4/23'))
  Mon Apr 23 2001 00:00:00 GMT-0400 (Eastern Daylight Time)          VM29:1
```

Development Difficulties

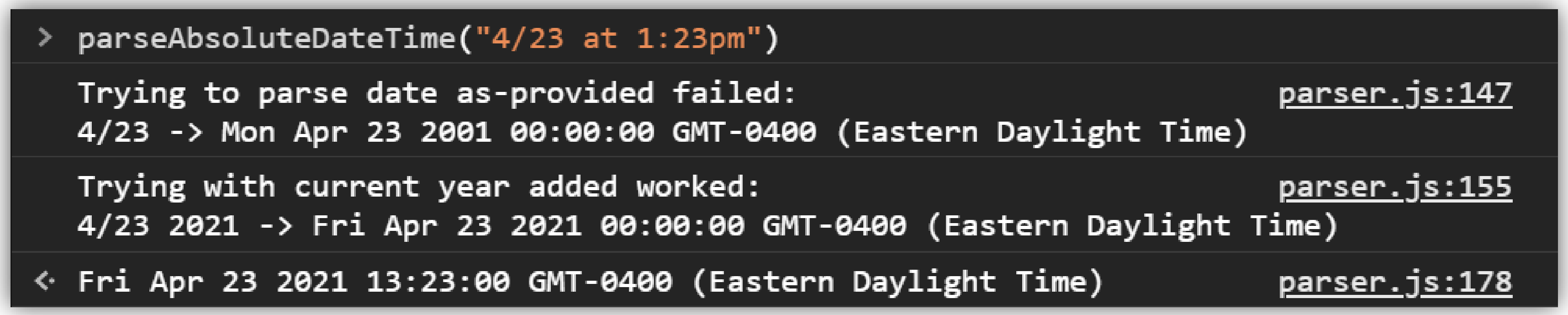
Our implementation detects the issue and adds an “implied year” value to work around this problem without using third-party date libraries

Mozilla Firefox



```
>> parseAbsoluteDateTime("4/23 at 1:23pm")
   Trying to parse date as-provided failed:                                 parser.js:147:11
   4/23 -> Invalid Date
   Trying with current year added worked:                                   parser.js:155:12
   4/23 2021 -> Fri Apr 23 2021 00:00:00 GMT-0400 (Eastern Daylight Time)
<- ▶ Date Fri Apr 23 2021 13:23:00 GMT-0400 (Eastern Daylight Time)        parser.js:178:10
```

Google Chrome



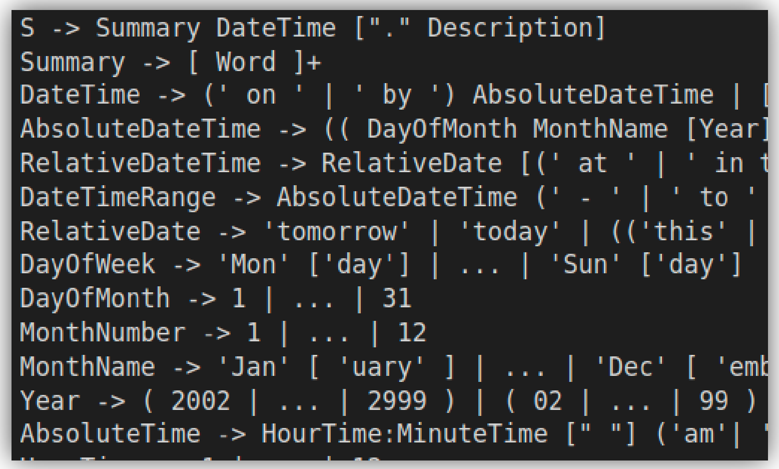
```
> parseAbsoluteDateTime("4/23 at 1:23pm")
  Trying to parse date as-provided failed:                               parser.js:147
  4/23 -> Mon Apr 23 2001 00:00:00 GMT-0400 (Eastern Daylight Time)
  Trying with current year added worked:                                 parser.js:155
  4/23 2021 -> Fri Apr 23 2021 00:00:00 GMT-0400 (Eastern Daylight Time)
<- Fri Apr 23 2021 13:23:00 GMT-0400 (Eastern Daylight Time)             parser.js:178
```

Further Development Difficulties

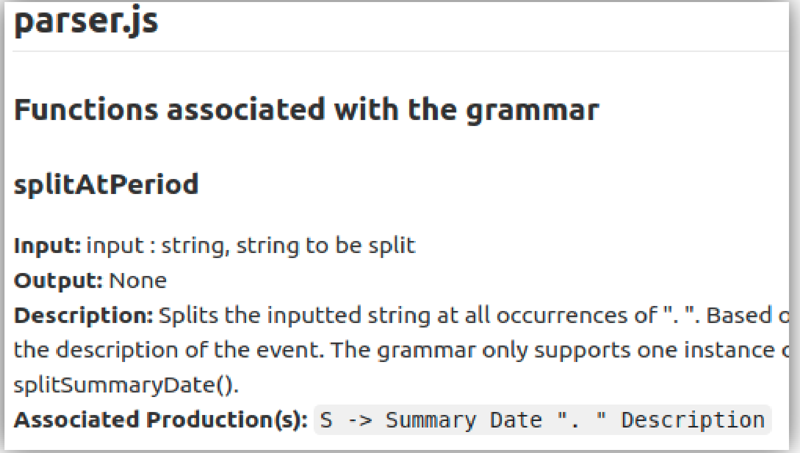
Problem: We faced difficulty dealing with the ambiguity of possible inputs; trying to support all sorts of input combinations led to the parser logic becoming unmanageable.

Solution: Limit scope by making the grammar stricter, so that there are fewer possible inputs and reduced ambiguity, which helped ensure that the project could be finished in time.

Documentation



```
S -> Summary DateTime ["." Description]
Summary -> [ Word ]+
DateTime -> (' on ' | ' by ') AbsoluteDateTime | [
AbsoluteDateTime -> (( DayOfMonth MonthName [Year]
RelativeDateTime -> RelativeDate [(' at ' | ' in t
DateTimeRange -> AbsoluteDateTime (' - ' | ' to '
RelativeDate -> 'tomorrow' | 'today' | (('this' |
DayOfWeek -> 'Mon' ['day'] | ... | 'Sun' ['day']
DayOfMonth -> 1 | ... | 31
MonthNumber -> 1 | ... | 12
MonthName -> 'Jan' [ 'uary' ] | ... | 'Dec' [ 'emb
Year -> ( 2002 | ... | 2999 ) | ( 02 | ... | 99 )
AbsoluteTime -> HourTime:MinuteTime [" "] ('am'| '
```

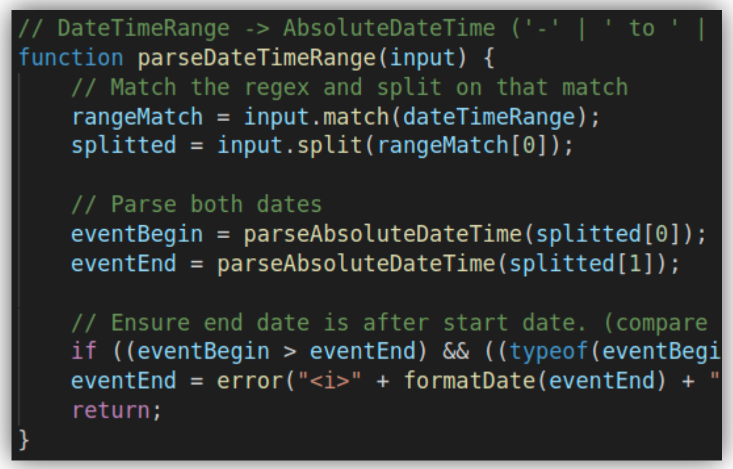


parser.js

Functions associated with the grammar

splitAtPeriod

Input: input : string, string to be split
Output: None
Description: Splits the inputted string at all occurrences of ". ". Based o
the description of the event. The grammar only supports one instance o
splitSummaryDate().
Associated Production(s): `S -> Summary Date ". " Description`



```
// DateTimeRange -> AbsoluteDateTime ('-' | ' to ' |
function parseDateTimeRange(input) {
    // Match the regex and split on that match
    rangeMatch = input.match(dateTimeRange);
    splitted = input.split(rangeMatch[0]);

    // Parse both dates
    eventBegin = parseAbsoluteDateTime(splitted[0]);
    eventEnd = parseAbsoluteDateTime(splitted[1]);

    // Ensure end date is after start date. (compare
    if ((eventBegin > eventEnd) && ((typeof(eventBegi
    eventEnd = error("<i>" + formatDate(eventEnd) + "
    return;
}
```

grammar.md

documentation.md

</code>

What We Learned

- The importance of limiting scope from the beginning
- Building a grammar and associated parser for a real-life scenario
- Further experience with the JavaScript language
- Using Developer Tools built into today's browsers for debugging
- Using the Jest framework for testing

Resources Used

iCalendar Specification (RFC 5545) - *Information on the .ics event file format and fields*

https://icalendar.org/RFC-Specifications/iCalendar-RFC-5545/

Mozilla Developer Network Documentation – *JavaScript functions, syntax and programming*

https://developer.mozilla.org/en-US/docs/Web/JavaScript

Jest – *JavaScript testing framework*

https://jestjs.io/

COMP SCI 4TB3 Lecture Notes – *Languages, grammars, regular expressions*

Emil Sekerinski, McMaster University

ics.js - *Assembles .ics file once the input has been processed by the parser*

https://github.com/nwcell/ics.js/

Pure CSS – *Styles the demo web page*

https://purecss.io/

Natural Language to iCalendar Converter

ANY QUESTIONS?