FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Review report of a final thesis

Reviewer:	Ing. Jan Trávníček, Ph.D.
Student:	Bc. Jan Jindráček
Thesis title:	Usability improvements to JavaScript/ECMAScript
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Created on:	31 May 2021

Evaluation criteria

1. Fulfillment of the assignment

▸ [✓ [1] **assignment fulfilled**]
[2] assignment fulfilled with minor objections
[3] assignment fulfilled with major objections
[4] assignment not fulfilled

The task was to study the JavaScript programming language and to identify its design issues. The task is of medium to higher complexity due to the overall size of the JavaScript programming language.

2. Main written part 85/100 (B)

The text of the thesis is written in very good to excellent English. It is worth noting that the text is detailed and may serve as a basis for a JonScript user guide/documentation. I find all sections important and content-wise rich. I have identified some factual and typographic issues with the text that are mostly minor. More important issues relate to the evaluation of JonScript language as such, either to its ease-of-use or to its performance.
- The evaluation of "Ease of use" criteria is not supported by any quantifiable data. It is only deduced that JonScript should be easy to use based on the proposed simplifications/amendments of JavaScript/TypeScript. I'm convinced the language is simpler, nevertheless, the claim should be supported by hard data.
- It would be interesting to see how exactly JonScript is translated to JavaScript especially for the claimed performance costly operations. I believe at least a rudimentary analysis will be presented at the defence.

Factual issues:
- [set vs list] - "Function may have a set of parameters" - shouldn't it be a list of parameters? (Also, an article is missing.)
- [NaN] - "This value is never equal, greater, lesser, greater or equal, lesser or equal than

any other value except for itself. This value is unequal to every other value within the standard, including itself." The two statements seem to contradict each other on the equality of NaNs.
- [nil vs null vs undefined] - The text regarding JonScript even below the introduction of nil in section 2.1.7 still refers to undefined or null which are by my understanding not present?
- [operator overloading] - Reserving a fixed list of keywords in order to implement operator overloading is not optimal when a single keyword would be enough (C++).
- [default parameters] - It is not clear from the description whether default parameters may only be specified for the rightmost parameters or any. The same with templates.
- [try-catch-finally] - Description of finally in the try-catch-finally expression is overcomplicated by alternating between cases: finally is present, finally is not present and finally is present again. Also, it is not clear what the "this expression" references in "If the finally is present, the resulting value of either the try expression, or the catch executed function will be passed into the function defined after the finally keyword. This expression returns either the value returned by catch on expression, or the value passed into try clause, if finally is not present.".
- [property access] - It is rather strange to name "." as a keyword in the section Property access expression. Also, it is not clear whether the square brackets access to a property is allowed to contain an expression or it has to contain a string literal only.
- [short-circuiting] - Even though it was mentioned before, the description of logical expressions should state again that logical or and logical and short-circuit.
- [queue example] - A queue is not an algorithm but a data type.

Typography issues:
- The thesis text was not prepared to be compiled on an UTF8 system (nor any other for that matter; i.e. the latex code is missing the specification of the input encoding); The multiplication symbol (×) was inserted into the text in a raw way whereas it should be specified using latex command \times.
- The Czech abstract overflows the right text boundary a bit.
- The first chapter (Introduction) already contains an analysis of issues with JavaScript that in my opinion deserves a separate Chapter. (Which to some extend is represented by Chapter II.)
- The text commonly alternates between "I" and "we".
- Paragraphs are sometimes too long.
- The last example of section 1.3.2 seems to be without context.
- "... example found in this [12] documentation." is awkward - could be rephrased "... example found in documentation [12]."
- "The differences between null and undefined", undefined should probably be bold.
- "... and what the future development of JonScript." is an unfinished sentence fragment.
- Is it webassembly or WebAssembly?
- Related works chapter should likely be present sooner in the text. I'm aware it contains references to JonScript improvements, these, however, may be reversed.
- Description of private properties and public properties is counter-intuitively placed into unrelated descriptions. (Private properties and Public API).

Typos
- [there] - "Modern programming languages that are not considered functional programming languages support there features".
- [order of define and to] - "Next, we are going define to create a property on Module."
- [if if, objects vs object's] - "If if does not find anything, it finds said objects prototype and

tries to find the method there."
- [no vs not] - "This is due to the fact that a function, or a constructor will treat an undefined parameter as if it was no there."
- [from vs in] - "This mistake was not fixed from the language..."
- [missing period] - "Relational operators are simplified There is no triple equals."
- [an vs and] - "Throw expression is an unary prefix operator, which consists of the throw keyword an an expression."

3. Non-written part, attachments 90/100 (A)

Lexer:
- lexer rule fragments are used inconsistently: TYPE_OR, TYPE_AND in TYPE_OPERATORS vs PLUSMINUS_OPERATORS.
- some parser rules may be simplified while still permitting the same language, consider the following fragment (with some alternatives omitted from rule "type"):
type : VAR
| (VAR | accessType) templateType
| accessType ;
accessType : VAR OBJECT_OPERATORS (accessType | VAR);

May be redesigned to:
type : VAR templateType?
| accessType templateType?;
accessType : VAR OBJECT_OPERATORS (accessType | VAR);

And further on to:
type : accessType templateType?;
accessType : VAR (OBJECT_OPERATORS VAR)*;

Also some comments in the grammar are ambiguous:
ignoreableVar : (VAR QMARK? (COLON type)? (EQUALS expression)?) | IGNORE; // Much later we can remove the '?'
Which '?' may be removed?

I reckon the implementation is sufficiently tested with unit tests and using a sample application testing intercompatibility with other TypeScript libraries.

Even though the quiz was fun (hacking it anyway), I would expect some sort of interactive parse-and-run like form to try JonScript easily.

4. Evaluation of results, publication outputs and awards 90/100 (A)

JavaScipt is known to be inconsistent and with many pitfalls. Any attempt to improve it, practical or theoretical, is important. The implementation as such requires partially software engineering and partially computer science knowledge. I believe the implementation fully fulfilled the theoretical aspect. However, based on mostly performance evaluation, some bits of the computer science contribution are yet to be added. The project has potential to be useful in the future if the stated limitations/ performance issues are fixed and yet to be implemented features are implemented.

The overall evaluation 87 /100 (B)

The text is detailed, well written, and easily understandable. The amount of work put into the implementation of the parser and transcriptor is more than enough for a masters thesis in a software engineering study branch. I recommend the thesis for defence and I recommend evaluating it with 87 points, i.e. grade B (very good).

Questions for the defense

Considering the example of the pattern matching implemented by the operator is, how exactly the matching of a concrete Date works? Is it fixed to the three possibilities explicitly mentioned or any value of any type (even user-defined) convertible to Date will match?
Have you considered using different tools, for instance, alternatives of the parser generator, and what were the decisions against those?

Instructions

Fulfillment of the assignment

Assess whether the submitted FT defines the objectives sufficiently and in line with the assignment; whether the objectives are formulated correctly and fulfilled sufficiently. In the comment, specify the points of the assignment that have not been met, assess the severity, impact, and, if appropriate, also the cause of the deficiencies. If the assignment differs substantially from the standards for the FT or if the student has developed the FT beyond the assignment, describe the way it got reflected on the quality of the assignment's fulfilment and the way it affected your final evaluation.

Main written part

Evaluate whether the extent of the FT is adequate to its content and scope: are all the parts of the FT contentful and necessary? Next, consider whether the submitted FT is actually correct – are there factual errors or inaccuracies?

Evaluate the logical structure of the FT, the thematic flow between chapters and whether the text is comprehensible to the reader. Assess whether the formal notations in the FT are used correctly. Assess the typographic and language aspects of the FT, follow the Dean's Directive No. 26/2017, Art. 3.

Evaluate whether the relevant sources are properly used, quoted and cited. Verify that all quotes are properly distinguished from the results achieved in the FT, thus, that the citation ethics has not been violated and that the citations are complete and in accordance with citation practices and standards. Finally, evaluate whether the software and other copyrighted works have been used in accordance with their license terms.

Non-written part, attachments

Depending on the nature of the FT, comment on the non-written part of the thesis. For example: SW work – the overall quality of the program. Is the technology used (from the development to deployment) suitable and adequate? HW – functional sample. Evaluate the technology and tools used. Research and experimental work – repeatability of the experiment.

Evaluation of results, publication outputs and awards

Depending on the nature of the thesis, estimate whether the thesis results could be deployed in practice; alternatively, evaluate whether the results of the FT extend the already published/known results or whether they bring in completely new findings.

The overall evaluation

Summarize which of the aspects of the FT affected your grading process the most. The overall grade does not need to be an arithmetic mean (or other value) calculated from the evaluation in the previous criteria. Generally, a well-fulfilled assignment is assessed by grade A.