National University of Singapore
School of Computing
CS1101S: Programming Methodology (JavaScript)
Semester I, 2012/2013

JediScript
Week 4

Introduction

The language JediScript is the official language of CS1101S. You have never heard of JediScript? No wonder, because we invented it just for the purpose of this module. It is a strict sublanguage of JavaScript Version 1.8.5 and as such is fully supported by our development environment (Komodo Edit and Firefox).

It is defined in this document (and its successors); updated weekly when necessary. The missions, side quests, competitions and practical exams use the JediScript language of that week. Students will receive deductions of marks starting in Week 3 if they are using JavaScript construct that are not part of JediScript of that week.

Statements

A JediScript program is a statement. Statements are defined using Backus Naur Form (BNF) as follows:

⟨*statement*⟩ ::= `;`
| `var` ⟨*id*⟩ `=` ⟨*expression*⟩ `;`
| `if (` ⟨*expression*⟩ `) {` ⟨*statement*⟩ `} else {` ⟨*statement*⟩ `}`
| `function` ⟨*id*⟩ `(` ⟨*id-list*⟩ `) {` ⟨*statement*⟩`}`
| `switch (`⟨*expression*⟩`) {`
⟨*switch-cases*⟩
`default:` ⟨*statement*⟩ `}`
| ⟨*statement*⟩ ⟨*statement*⟩
| `return` ⟨*expression*⟩ `;`
| ⟨*expression*⟩ `;`

⟨*id-list*⟩ ::=
| ⟨*non-empty-id-list*⟩

⟨*non-empty-id-list*⟩ ::= ⟨*id*⟩
| ⟨*id*⟩ `,` ⟨*non-empty-id-list*⟩

Important note: There cannot be any newline character between `return` and ⟨*expression*⟩ `;`.

⟨*switch-cases*⟩ ::=
| `case` ⟨*expression*⟩ `:` ⟨*statement*⟩ `break;` ⟨*switch-cases*⟩

⟨*expression*⟩ ::= ⟨*number*⟩
| `true` | `false`
| ⟨*string*⟩
| ⟨*expression*⟩ ⟨*bin-op*⟩ ⟨*expression*⟩
| ⟨*un-op*⟩ ⟨*expression*⟩
| `function` [⟨*id*⟩] `(` ⟨*id-list*⟩ `)` { ⟨*statement*⟩}
| ⟨*id*⟩`(` ⟨*expr-list*⟩ `)`
| `(` ⟨*expression*⟩ `)(` ⟨*expr-list*⟩ `)`
| ⟨*expression*⟩ `?` ⟨*expression*⟩ `:` ⟨*expression* ⟩
| `(` ⟨*expression*⟩ `)`

⟨*bin-op*⟩ ::= `+` | `-` | `*` | `/` | `%` | `===` | `!==` | `>` | `<` | `>=` | `<=` | `&&` | `||`

⟨*un-op*⟩ ::= `!` | `-`

⟨*expr-list*⟩ ::=
::= ⟨*non-empty-expr-list*⟩

⟨*non-empty-expr-list*⟩ ::= ⟨*expression*⟩
| ⟨*expression*⟩ `,` ⟨*non-empty-expr-list*⟩

Identifiers

In JediScript, an identifier consists of digits (0,...,9) and letters (a,...z,A,...Z) and begins with a letter.

The following identifiers can be used, in addition to identifiers that are declared using `var` and `function`:

- `alert`
- `Math.`⟨*name*⟩

where ⟨*name*⟩ is any name specified in the JavaScript `Math` library, see `http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf` (PDF) pages 159 and following, and `http://bclary.com/2004/11/07/` (HTML). Examples:

- `Math.E`: Refers to the mathematical constant e,
- `Math.PI`: Refers to the mathematical constant π,
- `Math.sqrt`: Refers to the square root function.

Note that technically, `Math.`⟨*name*⟩ is not an identifier, but an operator combination, the operator being "`.`". We will learn more about this operator when learning about objects.

Numbers

Examples for numbers are `5432`, `-5432.109`, and `-43.21e-45`.

Strings

Strings are of the form `"`⟨*characters*⟩`"`, where the character `"` does not appear in ⟨*characters*⟩, and of the form `'` ⟨*characters*⟩ `'`, where the character `'` does not appear in ⟨*characters*⟩.

Typing

Expressions evaluate to numbers, boolean values, strings or function values.

Only function values can be applied using the syntax:

⟨*expression*⟩ ::= ⟨*id*⟩`(` ⟨*expr-list*⟩ `)`
| `(` ⟨*expression*⟩ `)(` ⟨*expr-list*⟩ `)`

The following table specifies what arguments JediScript's operators take and what results they return.

operator	argument 1	argument 2	result
+	number	number	number
+	string	any	string
+	any	string	string
-	number	number	number
*	number	number	number
/	number	number	number
%	number	number	number
===	number	number	bool
!==	number	number	bool
>	number	number	bool
<	number	number	bool
>=	number	number	bool
<=	number	number	bool
&&	bool	bool	bool
||	bool	bool	bool
!	bool		bool
-	number		number

Following `if` and preceding `?`, JediScript only allows boolean expressions.

Comments

In JediScript, any sequence of characters between "`/*`" and the next "`*/`" is ignored.
After "`//`" any characters until the next newline character is ignored.