7.4.2 Keywords
The following tokens are ECMAScript keywords and may not be used as identifiers in ECMAScript programs.

Syntax

\[Keyword :: one of\]

- break
- for
- new
- typeof
- catch
- function
- return
- var
- continue
- if
- this
- void
- delete
- in
- throw
- while
- else
- instanceof
- try
- with

7.4.3 Future Reserved Words
The following words are used as keywords in proposed extensions and are therefore reserved to allow for the possibility of future adoption of those extensions.

Syntax

\[FutureReservedWord :: one of\]

- case
- default
- extends
- switch
- class
- do
- finally
- class
- enum
- import
- debugger
- export
- super

11.8 Relational operators

Syntax

\[RelationalExpression ::\]

- \[ShiftExpression\]
- \[RelationalExpression < ShiftExpression\]
- \[RelationalExpression > ShiftExpression\]
- \[RelationalExpression <= ShiftExpression\]
- \[RelationalExpression >= ShiftExpression\]
- \[RelationalExpression instanceof Identifier\]

Semantics
The result of evaluating \[RelationalExpression\] is always of type Boolean, reflecting whether the relationship named by the operator holds between its two operands.

11.8.6 The instanceof operator
The production \[RelationalExpression instanceof Identifier\] is evaluated as follows:

1. Evaluate \[RelationalExpression\].
2. Call GetValue(Result(1)).
3. Evaluate \[Identifier\].
4. Call GetValue(Result(3)).
5. If Result(2) is not an Object, return false.
6. If Result(4) is not a Function object, return false.
7. Call the [[Get]] method of Result(4) with parameter "prototype".
8. If Result(7) is not an Object, return false.
9. Let T be Result(7).
10. Let P be the value of the [[Prototype]] property of Result(2).
11. If P equals null, return false.
12. If P equals T, return true.
13. Let the new value of $P$ be the value of the [[Prototype]] property of $P$.