Ecma/TC39/2010/033

Overview

Array comprehensions were introduced in JavaScript 1.7. Comprehensions are a well-understood and popular language feature of list comprehensions, found in languages such as Python and Haskell, inspired by the mathematical notation of set comprehensions.

Array comprehensions are a convenient, declarative form for creating computed arrays with a literal syntax that reads naturally.

Examples

Filtering an array:

```
[ x for (x in a) if (x.color === 'blue') ]
```

Mapping an array:

```
[ square(x) for (x in values([1,2,3,4,5])) ]
```

Cartesian product:

```
[ [i,j] for (i in values(rows)) for (j in values(columns)) ]
```

Syntax

```
ArrayLiteral ::= ...
              |  "[" Expression ("for" "(" LHSExpression "in" Expression")")+ ("if" "(" Expression ")")? "]"
```

Translation

An array comprehension:

[$Expression_0$ **for (** $LHSExpression_1$ **in** $Expression_1$ **)** ... **for (** $LHSExpression_n$ **) if (** $Expression$ **)**$_{opt}$ **]**

can be defined by expansion to the expression:

```
let (result = []) {
    for (let LHSExpression_1 in Expression_1 ) {
        ...
        for (let LHSExpression_n in Expression_n ) {
            if ( Expression )_opt
                ArrayPush(result, Expression_0);
            }
        }
    }
=> result
}
```