Erratum for Final ES5 Specification (TC39-2009-043)

(Last Updated October 23, 2009)

8.6.2 Object Internal Properties and Methods

(third paragraph after Table 8)

Every ECMAScript object has a Boolean-valued [[Extensible]] internal property that controls whether or not named properties may be added to the object. If the value of the [[Extensible]] internal property is **false** then additional named properties may not be added to the object. In addition, if [[Extensible]] is **false** the value of the [[Class]] and [[Prototype]] internal properties of the object may not be modified. Once the value of an [[Extensible]] internal property has been set to **false** it may not be subsequently changed to **true**.

8.10.5 ToPropertyDescriptor (Obj)

5. If the result of calling the [[HasProperty]] internal method of *Obj* with argument "**`value`**" is **true**, then
    a. Let *value* be the result of calling the [[Get]] internal method of *Obj* with argument "**`value`**".
    b. Set the [[Value]] field of *desc* to *value*.

9.6 ToUint32: (Unsigned 32 Bit Integer)

The abstract operation ToUint32 converts its argument to one of 2^{32} integer values in the range 0 through $2^{32}-1$, inclusive. This abstraction operation functions as follows:

11.1.4 Array Initialiser

The production *ArrayLiteral* **`:[`** *ElementList* **`]`** is evaluated as follows:

1. Return the result of evaluating *ElementList*.

The production *ArrayLiteral* **`:[`** *ElementList* **`,`** $Elision_{opt}$ **`]`** is evaluated as follows:

1. Let *array* be the result of evaluating *ElementList*.
2. Let *pad* be the result of evaluating *Elision*; if not present, use the numeric value zero.
3. Let *len* be the result of calling the [[Get]] internal method of *array* with argument **`"length"`**.
4. Call the [[Put]] internal method of *array* with arguments **`"length"`**, ToUint32(*pad*+*len*), and **false**.
5. Return *array*.

11.3.1 Postfix Increment Operator

2. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*lhs*) is Reference is **true**
    - IsStrictReference(*lhs*) is **true**
    - Type(GetBase(*lhs*)) is Enviroment Record
    - GetReferencedName(*lhs*) is either **`"eval"`** or **`"arguments"`**

11.3.2 Postfix Decrement Operator

2. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*lhs*) is Reference is **true**
    - IsStrictReference(*lhs*) is **true**
    - Type(GetBase(*lhs*)) is Enviroment Record
    - GetReferencedName(*lhs*) is either **`"eval"`** or **`"arguments"`**

11.4.4 Prefix Increment Operator

2. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*expr*) is Reference is **true**
    - IsStrictReference(*expr*) is **true**
    - Type(GetBase(*expr*)) is Enviroment Record
    - GetReferencedName(*expr*) is either **`"eval"`** or **`"arguments"`**

11.4.5 Prefix Decrement Operator

2. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*expr*) is Reference is **true**
    - IsStrictReference(*expr*) is **true**
    - Type(GetBase(*expr*)) is Enviroment Record
    - GetReferencedName(*expr*) is either **`"eval"`** or **`"arguments"`**

11.13.1 Simple Assignment (=)

4. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*lref*) is Reference is **true**
    - IsStrictReference(*lref*) is **true**
    - Type(GetBase(*lref*)) is Enviroment Record
    - GetReferencedName(*lref*) is either **`"eval"`** or **`"arguments"`**

11.13.2 Compound Assignment (`op=`)

6. Throw a **SyntaxError** exception if the following conditions are all true:
    - Type(*lref*) is Reference is **true**
    - IsStrictReference(*lref*) is **true**
    - Type(GetBase(*lref*)) is Enviroment Record
    - GetReferencedName(lref) is either **`"eval"`** or **`"arguments"`**

15.4 Array Objects

An object, *O*, is said to be *sparse* if the following algorithm returns **true**:

1. Let *len* be the result of calling the [[Get]] internal method of *O* with argument **"length"**.
2. For each integer *i* in the range $0 \le i <$ ToUint32(*len*)
    a. Let *elem* be the result of calling the [[GetOwnProperty]] internal method of *O* with argument ToString(*i*).
    b. If *elem* is **undefined**, return **true**.
3. Return **false**.

15.4.4.11 Array.prototype.sort (comparefn)

(third paragraph)

Let *len* be the result of applying Uint32 to the result of calling the [[Get]] internal method of *obj* with argument "**`length`**".

(last item of first bulleted list)

- The result of calling the [[HasProperty]] internal method of *proto* with argument ToString(*j*) is **true**.

(starting at the last item of the second bulleted list)

- Any array index property of *obj* whose name is a nonnegative integer less than *len* is a data property whose [[Configurable]] attribute is **false**.

The behaviour of **`sort`** is also implementation defined if any array index property of *obj* whose name is a nonnegative integer less than *len* is an accessor property or is a data property whose [[Writable]] attribute is **false**.

Otherwise, the following steps are taken.

1. Perform an implementation-dependent sequence of calls to the [[Get]] , [[Put]], and [[Delete]] internal methods of *obj* and to SortCompare (described below), where the first argument for each call to [[Get]], [[Put]], or [[Delete]] is a nonnegative integer less than *len* and where the arguments for calls to SortCompare are results of previous calls to the [[Get]] internal method. The throw argument to the [[Put]] and [[Delete]] internal methods will be the value **true**. If *obj* is not sparse then [[Delete]] must not be called.

15.4.4.14 Array.prototype.indexOf (searchElement [, fromIndex])

9. Repeat, while $k < len$
    a. Let *kPresent* be the result of calling the [[HasProperty]] internal method of *O* with argument ToString(*k*).
    b. If *kPresent* is **true**, then

15.4.4.15 Array.prototype.lastIndexOf (searchElement [, fromIndex])

8. Repeat, while $k \ge 0$

a. Let *kPresent* be the result of calling the [[HasProperty]] internal method of *O* with argument ToString(*k*).
b. If *kPresent* is **true**, then

15.5.5.2 [[GetOwnProperty]] (P)

4. Let *str* be the String value of the [[PrimitiveValue]] internal property of *S*.

15.9.5 Properties of the Date Prototype Object

In following descriptions of functions that are properties of the Date prototype object, the phrase "this Date object" refers to the object that is the **this** value for the invocation of the function. Unless explicitly noted otherwise, none of these functions are generic; a **TypeError** exception is thrown if the **this** value is not an object for which the value of the [[Class]] internal property is **`"Date"`**. Also, the phrase "this time value" refers to the Number value for the time represented by this Date object, that is, the value of the [[PrimitiveValue]] internal property of this Date object.

Annex C

(4th bullet item)

- The identifier **`eval`** or **`arguments`** may not appear as the *LeftHandSideExpression* of an Assignment operator (11.13) or of a *PostfixExpression* (11.3) or as the *UnaryExpression* operated upon by a Prefix Increment (11.4.4) or a Prefix Decrement (11.4.5) operator.