ECMA Technical Report TR/00

ECMA 2000

ECMA/TC-39/2000/13

Standardizing Information and Communication Systems

ECMAScript Edition 3 Mobile Profile

Draft 0.2 October 2000

Brief history

This technical report defines ECMAScript Mobile Profile as a subset of ECMAScript Language Specification Edition 3…

...blah, blah, blah

1 Introduction

1.1 Scope

This Technical Report defines the ECMAScript Mobile Profile (ES-MP) scripting language.

1.2 Conformance

A conforming implementation of ES-MP must provide and support all the types, values, objects, properties, functions, and program syntax and semantics described or normatively referenced in this specification.

A conforming implementation of ES-MP is permitted to provide additional types, values, objects, properties, and functions beyond those described in this specification. In particular, a conforming implementation of ES-MP is permitted to provide properties not described in this specification, and values for those properties, for objects that are described in this specification.

2 Normative References

ISO/IEC 16262 (1999), ECMAScript Language Specification Edition 3

3 Definitions

4 Overview

This section contains a non-normative overview of the ECMAScript Mobile Profile language.

ECMAScript Edition 3 is an object-based programming language for performing computations and manipulating computational objects within a host environment. ECMAScript Edition 3 is object based: objects provide basic language and host facilities and an ECMAScript program is a cluster of communicating objects.

ECMAScript Mobile Profile is a subset of ECMAScript Edition 3 tailored to battery-powered embedded devices. Therefore, special attention is paid to constraining ECMAScript features that require proportionately large amounts of system memory (both for storing and executing the ECMAScript language features) and continuous or proportionately large amounts of processing power.

5 Language

This section contains a normative description of ECMAScript Mobile Profile Edition 3 (ES-MP). Unless specifically noted, ES-MP complies with ISO/IEC 16262, ECMAScript Language Specification Edition 3.

5.1 Global built-in object

A conforming implementation of ES-MP MUST support the `Global` built-in object, but is NOT REQUIRED to support the `eval()` method.

The rationale for not requiring support or `eval()` is that compilation on the mobile device should not be required. In ECMAScript Edition 3, the `eval()` function scope chain is initiated and executed every time the program comes across the `eval()` function. Therefore, implementing eval context in the run-time execution would require compilation on the handset.

5.2 RegExp built-in object

A conforming implementation MUST support the `RegExp` object in which branches of a regular expression pattern MAY contain at least, but not more than, four nested groups and at least, but not more than, eight repetition operators.

The rationale for requiring support for the `RegExp` in such reduced fashion stems from the need to limit usage of stack memory in the client. Stack usage for the regular expressions is a function of the depth of which groups are nested and the number of repetition operations in the expression. Analysis has shown that allowing four levels of nested groups and eight repetition operators requires no more than few K of stack on a 32 bit processor.

Run time compilation of the `RegExp` (usually required when a regular expression pattern is changed or when it is not known a priori) can be handled by a native object and does not cause significant memory overhead.

If a program attempts to use the `RegExp` object in a non-supported way, a conforming implementation of ES-MP MUST throw a `TypeError` exception object.

5.3 The other built-in objects

Unless otherwise stated a conforming implementation MUST support the following additional built-in objects as specified in ECMAScript Language Specification Edition 3

- `Object`
- `Function`
- `Array`
- `String`
- `Boolean`
- `Math`
- `Number`
- `Date`
- `Error`

5.4 The Number type

A conforming implementation of ES-MP MUST support the double-precision 64-bit format IEEE 754 values as specified in ISO/IEC 16262, ECMAScript Language Specification Edition 3.

The rationale for not relaxing this requirement is based on an analysis that has determined that while the memory required to store and process a double-precision floating point library is approximately twice that of a single-precision floating point library, the memory and processing requirements of the double-precision floating point library is still dis-proportionately small compared to other required features.

5.5 Versioning

A conforming implementation of ES-MP MUST support the `getVersion()` method on any built-in object. This method returns the version number for the object and is useful for testing objects when passed as arguments. Hence, it remains the responsibility of the developer to take any appropriate action if an undesired version number is returned.

The rationale for requiring the `getVersion()` method is that it addresses the immediate need to support language versioning in the rapidly evolving wireless scripting languages.

5.6 Dynamic Addition of Properties

Conforming implementation is NOT REQUIRED to support dynamic shadowing of built-in properties of the global object. Such shadowing occurs when the name of an existing global object property is assigned to a reference of another property. To ensure that attempts to replace the value of the property has no effect, the [ReadOnly] attribute of the built-in global object properties SHOULD be set to **true**.

The rationale for not requiring support for the dynamic replacement of properties to the global object is to allow more efficient implementations of ES3-MP to be devised. The implementation should allow built-in objects to be statically compiled without risking that the objects are shadowed by dynamically added properties.