ECMA Technical Report TR/00

ECMA 2000

ECMA/TC-39-TG1/2000/4

Standardizing Information and Communication Systems

ECMAScript 3rd Edition Compact Profile

Draft 0.4 November 2000

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

Brief history

This technical report defines ECMAScript Compact Profile as a subset of ECMA-262 (ECMAScript).

ECMAScript is based on several originating technologies, the most well known being JavaScript (Netscape) and Jscript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that company's Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 3.0.

The development of an ECMA standard began in November 1996. The first edition of ECMA-262 was adopted by the ECMA General Assembly of June 1997.

That ECMA Standard was submitted to ISO/IEC JTC1 for adoption under the fast-track procedure, and approved as international standard ISO/IEC 16262, in April 1998. The ECMA General Assembly of June 1998 approved the second edition of ECMA-262 to keep it fully aligned with ISO/IEC 16262. Changes between the first and the second edition are editorial in nature.

The third edition of ECMA-262 includes significant enhancements, for instance support for regular expressions, and improvements in regards to internationalisation. It was adopted by the ECMA General Assembly in December, 1999. This technical report defines the ECMAScript Compact Profile as a strict subset of the 3rd edition of ECMA-262. The Compact Profile is intended to meet the needs for resource-constrained environments, and has been developed with the help of Hewlett-Packard, IBM, Microsoft, Netscape, Nokia, Openwave, and Pixo.

1 Introduction

1.1 Scope

This Technical Report defines the ECMAScript Compact Profile (ES-CP) scripting language.

1.2 Conformance

A conforming implementation of ES-CP 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-CP is permitted to provide additional types, values, objects, properties, and functions beyond those described in this specification. In particular, a conforming implementation of ES-CP 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

ECMA-262 (ISO/IEC 16262 ECMAScript Language Specification)

3 Definitions

ES3	ECMAScript Language Specification, 3rd Edition
ES-CP	ECMAScript 3rd Edition Compact Profile

4 Overview

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

ECMAScript 3rd Edition is an object-oriented programming language for performing computations and manipulating computational objects within a host environment.

ECMAScript Compact Profile is a subset of ECMAScript 3rd Edition tailored to resource-constrained devices such as 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.

Operations which can potentially consume lots of memory, may exhaust the available memory in a resource constrained device. It is recommended that implementations of ES-CP provide a means for scripts to determine the amount of available memory, to allow the appropriate action to be taken when memory is in short supply.

Implementations of ES-CP are also recommended to provide a means for scripts to determine the version of host objects (see ES3 section 4.3.8).

5 Language

This section contains a normative description of ECMAScript 3rd Edition Compact Profile. Unless specifically noted, ES-CP adopts all requirements of ECMA-262 (ISO/IEC 16262, ECMAScript Language Specification).

5.1 Global built-in object

A conforming implementation of ES-CP SHALL support the Global built-in object, but is NOT REQUIRED to support the `Global.eval()` method, see ES3 section 15.1.2.1.

An implementation that does not support `Global.eval()` SHALL report an `EvalError` exception when `Global.eval()` is evaluated

NOTE

Runtime compilation is one of the most resource intensive features of ECMA-262,and many target environments for ES-CP will not support it. Global.`eval()` requires runtime compilation because the source text may not appear in the program (see ECMA-262, sections 15.1.2.1 and 10.1.2)."

5.2 Function Constructors

ES3 sections 15.3.2.1 and 15.3.1.1 define a mechanism for defining constructors for creating new objects. The constructor is specified as a set of arguments and an string containing the function body to be called. A conforming implementation of ES-CP MAY throw an `EvalError` exception if the function body is not a string literal.

NOTE

Runtime compilation is required to support the Function constructor and the Function function when the body is not a string literal.

5.3 Dynamic Addition of Properties

A conforming implementation of ES-CP is NOT REQUIRED to allow assignment to the properties of built-in objects, see ES3 section 4.3.7. If the implementation does not allow such assignments, then it SHALL throw a `ReferenceError` exception when evaluating assignments to properties of built-in objects.

NOTE

The rationale for not requiring support for the dynamic replacement of properties to built-in objects is to allow more efficient implementations of ES-CP based upon static compilation of built-in objects without risking that the objects are shadowed by dynamically added properties.