TC39 met for months to hammer out ECMA-262, a standard defining a new scripting language named ECMAScript (often pronounced as "ek-ma-script").

The following year, the International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) also adopted ECMAScript as a standard (ISO/IEC-16262). Since that time, browsers have tried, with varying degrees of success, to use ECMAScript as a basis for their JavaScript implementations.

JAVASCRIPT IMPLEMENTATIONS

Though JavaScript and ECMAScript are often used synonymously, JavaScript is much more than just what is defined in ECMA-262. Indeed, a complete JavaScript implementation is made up of the following three distinct parts (see Figure 1-1):

- The Core (ECMAScript)
- The Document Object Model (DOM)
- The Browser Object Model (BOM)

ECMAScript

ECMAScript, the language defined in ECMA-262, isn't tied to web browsers. In fact, the language has no methods for input or output whatsoever. ECMA-262 defines this language as a base upon which more-robust scripting languages may be built. Web browsers are just one *host environment* in which an ECMAScript implementation may exist. A host environment provides the base implementation of ECMAScript and implementation extensions designed to interface with the environment itself. Extensions, such as the Document Object Model (DOM), use ECMAScript's core types and syntax to provide additional functionality that's more specific to the environment. Other host environments include NodeJS, a server-side JavaScript platform, and the increasingly obsolete Adobe Flash.

What exactly does ECMA-262 specify if it doesn't reference web browsers? On a very basic level, it describes the following parts of the language:

- Syntax
- Types
- Statements
- Keywords
- Reserved words
- Operators
- Global objects

ECMAScript is simply a description of a language implementing all of the facets described in the specification. JavaScript implements ECMAScript, but so does Adobe ActionScript.