**[MS-ES5EX]:**

**Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)**



**Intellectual Property Rights Notice for Open Specifications Documentation**

* **Technical Documentation.** Microsoft publishes Open Specifications documentation forprotocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
* **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any otherterms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL’s, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
* **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
* **Patents.** Microsoft has patents that may cover your implementations of the technologiesdescribed in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft  [Open Specification Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the  [Communit](http://go.microsoft.com/fwlink/?LinkId=214448)y  [Promis](http://go.microsoft.com/fwlink/?LinkId=214448)e. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting  [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **Trademarks.** The names of companies and products contained in this documentation may becovered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
* **Fictitious Names.** The example companies, organizations, products, domain names, e-mailaddresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rightsother than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools orprogramming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.



*1 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**Revision Summary**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Date** |  |  | **Revision History** |  |  | **Revision Class** |  |  | **Comments** |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 09/08/2010 | |  | 0.1 | |  |  | New |  | Released new document. | | |  |
|  | |  |  | |  |  |  |  |  | | |  |
| 10/13/2010 | |  | 0.2 | |  |  | Minor |  | Clarified the meaning of the technical content. | | |  |
|  | |  |  | |  |  |  |  |  | | |  |
| 02/10/2011 | |  | 1.0 | |  |  | Minor |  | Clarified the meaning of the technical content. | | |  |
|  | |  |  | |  |  |  |  |  | | |  |
| 02/28/2011 | |  | 1.1 | |  |  | Minor |  | Clarified the meaning of the technical content. | | |  |
|  | |  |  | |  |  |  |  |  | | |  |
| 02/22/2012 | |  | 2.0 | |  |  | Major |  | Significantly changed the technical content. | | |  |
|  | |  |  | |  |  |  |  |  | | |  |
| 07/25/2012 | |  | 2.1 | |  |  | Minor |  | Clarified the meaning of the technical content. | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |



*2 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| [**1 Introduction ............................................................................................................**](#page7)**.** | | | **7** |
| 1.1 | [Glossary ..............................................................................................................](#page7). | | 7 |
| 1.2 | [References ...........................................................................................................](#page7). | | 7 |
| 1.2.1 | | [Normative References ......................................................................................](#page7). | 7 |
| 1.2.2 | | [Informative References ....................................................................................](#page7). | 7 |
| 1.3 | [Extension Overview (Synopsis) ...............................................................................](#page8). | | 8 |
| [1.3.1 Organization of This Documentation ..................................................................](#page9). | | | 9 |
| 1.4 | [Relationship to Standards and Other Extensions .......................................................](#page9). | | 9 |
| 1.5 | [Applicability Statement ..........................................................................................](#page9). | | 9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [**2 Extensions .............................................................................................................**](#page10)**.** | | | | **10** |
| [2.1 Extensions to Lexical Conventions .........................................................................](#page10). | | | | 10 |
| [2.1.1 Conditional Source Text Processing..................................................................](#page10). | | | | 10 |
|  | 2.1.1.1 | | [Global State ............................................................................................](#page11). | 11 |
|  | 2.1.1.2 | | [Conditional Processing Algorithm ...............................................................](#page11). | 11 |
| [2.1.2 Extensions to Numeric Literals ........................................................................](#page21). | | | | 21 |
| [2.1.3 Extensions to String Literals ...........................................................................](#page21). | | | | 21 |
| 2.2 | [Extensions to Types ............................................................................................](#page22). | | | 22 |
| 2.2.1 | | [SafeArray Type .............................................................................................](#page22). | | 22 |
| 2.2.2 | | [VarDate Type ................................................................................................](#page22). | | 22 |
| [2.3 Extensions to Type Conversion and Testing ............................................................](#page22). | | | | 22 |
| [2.4 Extensions to Executable Code and Execution Contexts ...........................................](#page23). | | | | 23 |
| [2.4.1 Extensions to Declaration Binding Instantiation .................................................](#page23). | | | | 23 |
| 2.5 | [Extensions to Expressions ....................................................................................](#page23). | | | 23 |
| [2.5.1 Extensions to typeof Operator .........................................................................](#page23). | | | | 23 |
| 2.6 | [Extensions to Statements ....................................................................................](#page24). | | | 24 |
| [2.6.1 Extension Grammar Production for Statement ...................................................](#page24). | | | | 24 |
| [2.7 Extensions to Function Definition ..........................................................................](#page24). | | | | 24 |
| [2.7.1 Function Definition Used As a Statement ..........................................................](#page24). | | | | 24 |
| [2.7.2 Event Handler Function Definitions ..................................................................](#page24). | | | | 24 |
| [2.8 Extensions to Native ECMAScript Objects ...............................................................](#page25). | | | | 25 |
| [2.8.1 Function Properties of the Global Object ...........................................................](#page26). | | | | 26 |
|  | 2.8.1.1 | | [ScriptEngine ...........................................................................................](#page26). | 26 |
|  | 2.8.1.2 | | [ScriptEngineBuildVersion ..........................................................................](#page26). | 26 |
|  | 2.8.1.3 | | [ScriptEngineMajorVersion .........................................................................](#page26). | 26 |
|  | 2.8.1.4 | | [ScriptEngineMinorVersion .........................................................................](#page26). | 26 |
|  | 2.8.1.5 | | [CollectGarbage ........................................................................................](#page26). | 26 |
| [2.8.2 Constructor Properties of the Global Object ......................................................](#page26). | | | | 26 |
| [2.8.3 Properties of Function Instances ......................................................................](#page27). | | | | 27 |
|  | 2.8.3.1 | | [The arguments Property ...........................................................................](#page27). | 27 |
|  | 2.8.3.2 | | [The caller Property...................................................................................](#page27). | 27 |
|  | [2.8.3.3 The [[Get]] (P) Method of a Function Object ...............................................](#page27). | | | 27 |
| [2.8.4 String.prototype HTML Wrapper Properties .......................................................](#page28). | | | | 28 |
|  | 2.8.4.1 | | [String.prototype.anchor(name) .................................................................](#page28). | 28 |
|  | 2.8.4.2 | | [String.prototype.big( ) .............................................................................](#page28). | 28 |
|  | 2.8.4.3 | | [String.prototype.blink( ) ...........................................................................](#page29). | 29 |
|  | 2.8.4.4 | | [String.prototype.bold( ) ............................................................................](#page29). | 29 |
|  | 2.8.4.5 | | [String.prototype.fixed( ) ...........................................................................](#page29). | 29 |
|  | 2.8.4.6 | | [String.prototype.fontcolor(color) ...............................................................](#page29). | 29 |
|  | 2.8.4.7 | | [String.prototype.fontsize(size) ..................................................................](#page29). | 29 |



*3 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.8.4.8 | | [String.prototype.italics( ) ..........................................................................](#page29). | | 29 |
| 2.8.4.9 | | [String.prototype.link(url) ..........................................................................](#page29). | | 29 |
| 2.8.4.10 | | [String.prototype.small( ) ........................................................................](#page29). | | 29 |
| 2.8.4.11 | | [String.prototype.strike( ) ........................................................................](#page29). | | 29 |
| 2.8.4.12 | | [String.prototype.sub( ) ...........................................................................](#page29). | | 29 |
| 2.8.4.13 | | [String.prototype.sup( ) ...........................................................................](#page29). | | 29 |
| [2.8.5 Properties of the Date Prototype Object ...........................................................](#page29). | | | | 29 |
| 2.8.5.1 | | [Date.prototype.getVarDate ( ) ...................................................................](#page29). | | 29 |
| [2.8.6 Properties of the RegExp Constructor ...............................................................](#page30). | | | | 30 |
| 2.8.6.1 | | [RegExp.input ..........................................................................................](#page30). | | 30 |
| 2.8.6.2 | | [RegExp.lastIndex .....................................................................................](#page30). | | 30 |
| 2.8.6.3 | | [RegExp.lastMatch ....................................................................................](#page30). | | 30 |
| 2.8.6.4 | | [RegExp.lastParen .....................................................................................](#page30). | | 30 |
| 2.8.6.5 | | [RegExp.leftContext ..................................................................................](#page30). | | 30 |
| 2.8.6.6 | | [RegExp.rightContext ................................................................................](#page30). | | 30 |
| [2.8.6.7 RegExp.$1 - RegExp.$9 ............................................................................](#page30). | | | | 30 |
| 2.8.6.8 | | [RegExp.$\_ ..............................................................................................](#page31). | | 31 |
| 2.8.6.9 | | [RegExp['$&'] ...........................................................................................](#page31). | | 31 |
| 2.8.6.10 | | [RegExp['$+'] .........................................................................................](#page31). | | 31 |
| 2.8.6.11 | | [RegExp["$`"] ........................................................................................](#page31). | | 31 |
| 2.8.6.12 | | [RegExp["$'"] .........................................................................................](#page31). | | 31 |
| [2.8.7 Properties of the RegExp Prototype Object .......................................................](#page31). | | | | 31 |
| 2.8.7.1 | | [RegExp.prototype.compile(pattern, flags) ...................................................](#page31). | | 31 |
| [2.8.8 Properties of RegExp Instances .......................................................................](#page32). | | | | 32 |
| 2.8.8.1 | | [options ...................................................................................................](#page32). | | 32 |
| 2.8.9 | [The Error Constructor ....................................................................................](#page32). | | | 32 |
| 2.8.9.1 | | [new Error () ............................................................................................](#page32). | | 32 |
| 2.8.9.2 | | [new Error(number, message) ....................................................................](#page32). | | 32 |
| [2.8.10 Properties of Error Instances .........................................................................](#page33). | | | | 33 |
| 2.8.10.1 | | [description ............................................................................................](#page33). | | 33 |
| 2.8.10.2 | | [number ................................................................................................](#page33). | | 33 |
| [2.8.11 Properties of NativeError Instances ................................................................](#page33). | | | | 33 |
| 2.8.11.1 | | [description ............................................................................................](#page33). | | 33 |
| 2.8.11.2 | | [number ................................................................................................](#page33). | | 33 |
| 2.8.12 | [The Debug Object ........................................................................................](#page33). | | | 33 |
| [2.8.12.1 Function Properties of the Debug Object ...................................................](#page34). | | | | 34 |
| [2.8.12.1.1 write ([ item1 [, item2 [, …]]]) ...........................................................](#page34). | | | | 34 |
| [2.8.12.1.2 writeln ([ item1 [, item2 [, …]]])) .......................................................](#page34). | | | | 34 |
| 2.8.13 | [Enumerator Objects .....................................................................................](#page34). | | | 34 |
| [2.8.13.1 The Enumerator Constructor Called as a Function ......................................](#page34). | | | | 34 |
| 2.8.13.2 | | [The Enumerator Constructor ...................................................................](#page34). | | 34 |
| 2.8.13.2.1 | | | [new Enumerator ([collection])............................................................](#page34). | 34 |
| [2.8.13.3 Properties of the Enumerator Constructor .................................................](#page35). | | | | 35 |
| 2.8.13.3.1 | | | [Enumerator.prototype .......................................................................](#page35). | 35 |
| [2.8.13.4 Properties of the Enumerator Prototype Object ..........................................](#page35). | | | | 35 |
| 2.8.13.4.1 | | | [Enumerator.prototype.constructor ......................................................](#page35). | 35 |
| 2.8.13.4.2 | | | [Enumerator.prototype.atEnd ( ) .........................................................](#page35). | 35 |
| 2.8.13.4.3 | | | [Enumerator.prototype.item ( ) ...........................................................](#page36). | 36 |
| 2.8.13.4.4 | | | [Enumerator.prototype.moveFirst ( ) ...................................................](#page36). | 36 |
| 2.8.13.4.5 | | | [Enumerator.prototype.moveNext ( ) ...................................................](#page36). | 36 |
| [2.8.13.5 Properties of Enumerator Instances ..........................................................](#page37). | | | | 37 |
| 2.8.14 | [VBArray Objects ..........................................................................................](#page37). | | | 37 |
| [2.8.14.1 The VBArray Constructor Called as a Function ...........................................](#page37). | | | | 37 |



*4 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2.8.14.1.1 | | | [VBArray ( value) ..............................................................................](#page37). | | 37 |
| 2.8.14.2 | | [The VBArray Constructor ........................................................................](#page37). | | | 37 |
| 2.8.14.2.1 | | | [new VBArray ( value ) .......................................................................](#page37). | | 37 |
| [2.8.14.3 Properties of the VBArray Constructor ......................................................](#page37). | | | | | 37 |
| 2.8.14.3.1 | | | [VBArray.prototype ............................................................................](#page38). | | 38 |
| [2.8.14.4 Properties of the VBArray Prototype Object ...............................................](#page38). | | | | | 38 |
| 2.8.14.4.1 | | | [VBArray.prototype.constructor ...........................................................](#page38). | | 38 |
| 2.8.14.4.2 | | | [VBArray.prototype.dimensions ( ) .......................................................](#page38). | | 38 |
| [2.8.14.4.3 VBArray.prototype.getItem ( dim1 [, dim2, [dim3, …]]) ........................](#page38). | | | | | 38 |
| [2.8.14.4.4 VBArray.prototype.lbound ( [dimension] ) ...........................................](#page38). | | | | | 38 |
| 2.8.14.4.5 | | | [VBArray.prototype.toArray ( ) ............................................................](#page39). | | 39 |
| [2.8.14.4.6 VBArray.prototype.ubound ( [dimension] ) ..........................................](#page39). | | | | | 39 |
| 2.8.14.4.7 | | | [VBArray.prototype.valueOf ( ) ............................................................](#page40). | | 40 |
| [2.8.14.5 Properties of VBArray Instances ...............................................................](#page40). | | | | | 40 |
| 2.8.15 | [ActiveXObject Objects ..................................................................................](#page40). | | | | 40 |
| [2.8.15.1 The ActiveXObject Constructor Called as a Function ...................................](#page40). | | | | | 40 |
| 2.8.15.1.1 | | | [ActiveXObject ( name [, location])) ....................................................](#page40). | | 40 |
| 2.8.15.2 | | [The ActiveXObject Constructor ................................................................](#page41). | | | 41 |
| [2.8.15.2.1 new ActiveXObject (( name [, location]) ) ...........................................](#page41). | | | | | 41 |
| [2.8.15.3 Properties of the ActiveXObject Constructor ..............................................](#page41). | | | | | 41 |
| 2.8.15.3.1 | | | [ActiveXObject.prototype ....................................................................](#page41). | | 41 |
| [2.8.15.4 Properties of the ActiveXObject Prototype Object .......................................](#page42). | | | | | 42 |
| 2.8.15.4.1 | | | [ActiveXObject.prototype.constructor ...................................................](#page42). | | 42 |
| [2.8.15.5 Properties of ActiveXObject Instances .......................................................](#page42). | | | | | 42 |
| [2.9 Extensions to ECMAScript 5.1 ...............................................................................](#page42). | | | | | 42 |
| 2.9.1 | [Typed Arrays ................................................................................................](#page42). | | | | 42 |
| 2.9.1.1 | | [ArrayBuffer Objects .................................................................................](#page42). | | | 42 |
| [2.9.1.1.1 The ArrayBuffer constructor called as a function .....................................](#page42). | | | | | 42 |
| 2.9.1.1.2 | | | [The ArrayBuffer constructor .................................................................](#page42). | | 42 |
| 2.9.1.1.2.1 | | | | [New Array (len) ............................................................................](#page43). | 43 |
| [2.9.1.1.3 Properties of the ArrayBuffer constructor...............................................](#page43). | | | | | 43 |
| 2.9.1.1.3.1 | | | | [ArrayBuffer.Prototype ...................................................................](#page43). | 43 |
| [2.9.1.1.4 Properties of the ArrayBuffer Prototype Object .......................................](#page43). | | | | | 43 |
| 2.9.1.1.4.1 | | | | [ArrayBuffer.prototype.constructor ..................................................](#page43). | 43 |
| 2.9.1.1.5 | | | [Properties of ArrayBuffer Instances ......................................................](#page43). | | 43 |
| 2.9.1.1.5.1 | | | | [byteLength ..................................................................................](#page43). | 43 |
| 2.9.1.2 | | [TypeArray Objects ...................................................................................](#page44). | | | 44 |
| [2.9.1.2.1 The TypeArray Constructor Called as a Function .....................................](#page44). | | | | | 44 |
| 2.9.1.2.2 | | | [The TypeArray Constructor ..................................................................](#page44). | | 44 |
| [2.9.1.2.2.1 New TypeArray (arg0 [, arg1, [, arg2]) ...........................................](#page44). | | | | | 44 |
| [2.9.1.2.3 Properties of the TypeArray Constructor ................................................](#page46). | | | | | 46 |
| 2.9.1.2.3.1 | | | | [TypeArray.prototype .....................................................................](#page46). | 46 |
| [2.9.1.2.3.2 typeArray.BYTES\_PER\_ELEMENT ....................................................](#page46). | | | | | 46 |
| [2.9.1.2.4 Properties of the TypeArray Prototype Object .........................................](#page46). | | | | | 46 |
| 2.9.1.2.4.1 | | | | [TypeArray.prototype.constructor ....................................................](#page46). | 46 |
| 2.9.1.2.4.2 | | | | [TypeArray.prototype.set(Array [, offset] ) .......................................](#page46). | 46 |
| 2.9.1.2.4.3 | | | | [TypeArray.prototype.subarray(begin [, end] ) ..................................](#page47). | 47 |
| [2.9.1.2.5 Properties of TypeArray Instances ........................................................](#page48). | | | | | 48 |
| [2.9.1.2.5.1 [[DefineOwnProperty]] (P, Desc, Throw ) ........................................](#page48). | | | | | 48 |
| 2.9.1.2.5.2 | | | | [[[GetOwnProperty]] ( P) ................................................................](#page48). | 48 |
| 2.9.1.2.5.3 | | | | [length .........................................................................................](#page49). | 49 |
| 2.9.1.2.5.4 | | | | [byteLength ..................................................................................](#page49). | 49 |
| 2.9.1.2.5.5 | | | | [buffer ..........................................................................................](#page49). | 49 |



*5 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2.9.1.2.5.6 | | [byteOffset ...................................................................................](#page49). | 49 |
|  | 2.9.1.3 | [DataView Objects ....................................................................................](#page50). | | 50 |
|  | [2.9.1.3.1 The DataView Constructor called as a function .......................................](#page50). | | | 50 |
|  | [2.9.1.3.2 The DataView Constructor ...................................................................](#page50). | | | 50 |
|  | [2.9.1.3.2.1 New DataView (buffer [, byteOffset [, byteLength]]) .........................](#page50). | | | 50 |
|  | [2.9.1.3.3 Properties of the DataView Constructor .................................................](#page50). | | | 50 |
|  | 2.9.1.3.3.1 | | [DataView.prototype ......................................................................](#page50). | 50 |
|  | [2.9.1.3.4 Properties of the DataView Prototype Object ..........................................](#page51). | | | 51 |
|  | 2.9.1.3.4.1 | | [DataView.prototype.constructor .....................................................](#page51). | 51 |
|  | 2.9.1.3.4.2 | | [DataView.prototype.GetInt8(byteOffset) .........................................](#page51). | 51 |
|  | 2.9.1.3.4.3 | | [DataView.prototype.GetUInt8(byteOffset) .......................................](#page52). | 52 |
|  | 2.9.1.3.4.4 | | [DataView.prototype.GetInt16(byteOffset, littleEndian) ......................](#page52). | 52 |
|  | 2.9.1.3.4.5 | | [DataView.prototype.GetUInt16(byteOffset, littleEndian) ....................](#page52). | 52 |
|  | 2.9.1.3.4.6 | | [DataView.prototype.GetInt32(byteOffset, littleEndian) ......................](#page52). | 52 |
|  | 2.9.1.3.4.7 | | [DataView.prototype.GetUInt32(byteOffset, littleEndian) ....................](#page52). | 52 |
|  | 2.9.1.3.4.8 | | [DataView.prototype.GetFloat32(byteOffset, littleEndian) ...................](#page53). | 53 |
|  | 2.9.1.3.4.9 | | [DataView.prototype.GetFloat64(byteOffset, littleEndian) ...................](#page53). | 53 |
|  | 2.9.1.3.4.10 | | [DataView.prototype.SetInt8(byteOffset, value) ..............................](#page53). | 53 |
|  | 2.9.1.3.4.11 | | [DataView.prototype.SetUInt8(byteOffset, value) ............................](#page53). | 53 |
|  | 2.9.1.3.4.12 | | [DataView.prototype.SetInt16(byteOffset, value, littleEndian) ...........](#page53). | 53 |
|  | 2.9.1.3.4.13 | | [DataView.prototype.SetUInt16(byteOffset, value, littleEndian) .........](#page54). | 54 |
|  | 2.9.1.3.4.14 | | [DataView.prototype.SetInt32(byteOffset, value, littleEndian) ...........](#page54). | 54 |
|  | 2.9.1.3.4.15 | | [DataView.prototype.SetUInt32(byteOffset, value, littleEndian) .........](#page54). | 54 |
|  | 2.9.1.3.4.16 | | [DataView.prototype.SetFloat32(byteOffset, value, littleEndian) ........](#page54). | 54 |
|  | 2.9.1.3.4.17 | | [DataView.prototype.SetFloat64(byteOffset, value, littleEndian) ........](#page54). | 54 |
|  | [2.9.1.3.5 Properties of DataView Instances .........................................................](#page55). | | | 55 |
|  | 2.9.1.3.5.1 | | [byteLength ..................................................................................](#page55). | 55 |
|  | 2.9.1.3.5.2 | | [buffer ..........................................................................................](#page55). | 55 |
|  | 2.9.1.3.5.3 | | [byteOffset ...................................................................................](#page55). | 55 |
|  | [2.9.2 Properties of Error Constructor ........................................................................](#page55). | | | 55 |
|  | 2.9.2.1 | [stackTraceLimit .......................................................................................](#page55). | | 55 |
|  | [2.9.3 Properties of Error Instances ..........................................................................](#page55). | | | 55 |
|  | 2.9.3.1 | [stack ......................................................................................................](#page55). |  | 55 |
| **3** | [**Security Considerations .........................................................................................**](#page57)**.** | | | **57** |
| [**4 Appendix A: Product Behavior ...............................................................................**](#page58)**.** | | | | **58** |
| **5** | [**Change Tracking....................................................................................................**](#page59)**.** | |  | **59** |
| **6** | [**Index ....................................................................................................................**](#page61)**.** |  |  | **61** |



*6 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**1 Introduction**

This document describes extensions provided to the ECMAScript language that are implemented in IE9 Mode in Windows® Internet Explorer® 9 based on the ECMAScript Language Specification 5th Edition  [[ECMA-262/5],](http://go.microsoft.com/fwlink/?LinkId=185963) published December 2009. These are described in sections 2.1 through 2.8.

The implementation of the ECMAScript language in IE10 Mode in Windows® Internet Explorer® 10 is based on ECMAScript Language Specification 5.1 Edition  [[ECMA-262/51](http://go.microsoft.com/fwlink/?LinkId=240226)], published June 2011. All of the extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) described in this document for IE9 Mode also apply to IE10 Mode in Internet Explorer 10. In addition, section 2.9 describes extensions to the ECMAScript Language Specification 5.1 Edition  [[ECMA-262/51]](http://go.microsoft.com/fwlink/?LinkId=240226) in IE10 Mode; these extensions are not available in versions prior to Internet Explorer 10.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1. **Glossary**

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used asdescribed in  [[RFC2119](http://go.microsoft.com/fwlink/?LinkId=90317)]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1. **References**

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1. **Normative References**

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact  [dochelp@microsoft.co](mailto:dochelp@microsoft.com)m. We will assist you in finding the relevant information. Please check the archive site,  [http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C962](http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624)4, as an additional source.

[ECMA-262/5] ECMA International, "Standard ECMA-262 ECMAScript Language Specification", 5th Edition (December 2009),  [http://www.ecma-international.org/publications/files/ECMA-ST](http://go.microsoft.com/fwlink/?LinkId=185963)- [ARCH/ECMA-262%205th%20edition%20December%202009.pd](http://go.microsoft.com/fwlink/?LinkId=185963)f

[ECMA-262/51] ECMA International, "Standard ECMA-262 ECMAScript Language Specification", 5.1 Edition (June 2011),  [http://www.ecma-international.org/publications/standards/Ecma-262.ht](http://go.microsoft.com/fwlink/?LinkId=240226)m

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997,  [http://www.rfc-editor.org/rfc/rfc2119.tx](http://go.microsoft.com/fwlink/?LinkId=90317)t

1. **Informative References**

[MS-ES3] Microsoft Corporation, " [Microsoft JScript ECMAScript Language Specification 3rd Editio](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3%5d.pdf)n  [Standards Support Documen](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3%5d.pdf)t".

[MS-ES3EX] Microsoft Corporation, " [Microsoft JScript Extensions to the ECMAScript Languag](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3EX%5d.pdf)e  [Specification 3rd Editio](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3EX%5d.pdf)n".



*7 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

[MS-ES5] Microsoft Corporation, " [Internet Explorer ECMA-262 ECMAScript Language Specificatio](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES5%5d.pdf)n  [Fifth Edition Standards Support Documen](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES5%5d.pdf)t".

[MS-ES5EX] Microsoft Corporation, "Microsoft Internet Explorer Extensions to the ECMAScript Language Specification Fifth Edition".

1. **Extension Overview (Synopsis)**

The extensions described in this document were selected for their applicability to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963). These extensions are organized based on sections of  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) as follows.

Section 7,  [Lexical Convention](#page10)s

1. [Conditional Source Text Processin](#page10)g
2. 7.8.3,  [Numeric Literal](#page21)s
3. 7.8.4,  [String Literal](#page21)s

Section 8,  [Type](#page22)s

Section 9,  [Type Conversion and Testin](#page22)g

Section 10,  [Executable Code and Execution Context](#page23)s

Section 11,  [Expression](#page23)s

Section 12,  [Statement](#page24)s

Section 13,  [Function Definitio](#page24)n

Section 15,  [Standard Built-in ECMAScript Object](#page25)s

1. 15.1.2,  [Function Properties of the Global Objec](#page26)t
2. 15.1.4,  [Constructor Properties of the Global Objec](#page26)t
3. 15.3.5,  [Properties of Function Instance](#page27)s
4. [String.prototype HTML Wrapper Propertie](#page28)s
5. 15.9.5,  [Properties of the Date Prototype Objec](#page29)t
6. 15.10.5,  [Properties of the RegExp Constructo](#page30)r
7. 15.10.6,  [Properties of the RegExp Prototype Objec](#page31)t
8. 15.10.7,  [Properties of RegExp Instance](#page32)s
9. 15.11.2,  [The Error Constructo](#page32)r

10.15.11.5,  [Properties of Error Instance](#page33)s

11.15.11.6,  [Native Error Types Used in This Standar](#page33)d

12. [The Debug Objec](#page33)t

13. [Enumerator Object](#page34)s



*8 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

14. [VBArray Object](#page37)s

15. [ActiveXObject Object](#page40)s

1. **Organization of This Documentation**

This document is organized as follows:

* 1. **Conditional Source Text Processing**: Processing of source text by Internet ExplorerECMAScript.
  2. **Extensions to Types:** Types defined by Internet Explorer ECMAScript that supplement types of [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963).
  3. **Extensions to Statements:** A statement defined by Internet Explorer ECMAScript thatsupplements statements of  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963).
  4. **Extensions to Native ECMAScript Objects**: Object extensions defined by Internet ExplorerECMAScript are listed according to object at the highest level.
  5. **Properties:** The object properties defined by Internet Explorer ECMAScript, typically functions,methods, or data formats, are described at the next levels.

1. **Relationship to Standards and Other Extensions**

This document defines extensions to  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)]. Variations from  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) are defined in  [[MS-ES5]](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES5%5d.pdf).

The following documents describe variations and extensions from versions 3 and 5 of the ECMAScript Language:

|  |  |  |
| --- | --- | --- |
| Document | Reference | Title |
| Type |  |  |
|  |  |  |
| Variations | [[MS-ES3](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3%5d.pdf)] | Internet Explorer ECMA-262 ECMAScript Language Specification Standards |
|  |  | Support Document |
|  |  |  |
| Variations | [MS-ES5] | Internet Explorer ECMA-262 ECMAScript Language Specification (Fifth |
|  |  | Edition) Standards Support Document |
|  |  |  |
| Extensions | [[MS](file:///C:\Users\albert\Desktop\fronteers\ie-standards-docs\ms-es5ex-ie-extensions-to-the-ecma-262-ecmascript-language-specification-fifth-edition\%5bMS-ES3EX%5d.pdf)- | Microsoft JScript Extensions to the ECMAScript Language Specification |
|  | ES3EX] | Third Edition |
|  |  |  |
| Extensions | [MS- | Internet Explorer Extensions to the ECMA-262 ECMAScript Language |
|  | ES5EX] | Specification (Fifth Edition) |
|  |  |  |

1. **Applicability Statement**

This document specifies a set of extensions to the  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) specifications. The extensions in this document provide access to some features that are unique to Windows® Internet Explorer® 9 and Windows® Internet Explorer® 10 when it is operating in IE9 Mode.

Section 2.9 specifies extensions to the ECMAScript Language Specification 5.1 Edition  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=240226)- [262/51]](http://go.microsoft.com/fwlink/?LinkId=240226) in IE10 Mode.



*9 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**2 Extensions**

This section specifies extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) that are available in Windows® Internet Explorer® 9 in IE9 Mode.

Section 2.9 specifies extensions to the  [[ECMA-262/51]](http://go.microsoft.com/fwlink/?LinkId=240226) in Windows® Internet Explorer® 10 for IE10 Mode.

The extensions are as follows:

[page10](#page10)  [Extensions to Lexical Convention](#page10)s

[page22](#page22)  [Extensions to Type](#page22)s

[page22](#page22)  [Extensions to Type Conversion and Testin](#page22)g

[page23](#page23)  [Extensions to Executable Code and Execution Context](#page23)s

[page23](#page23)  [Extensions to Expression](#page23)s

[page24](#page24)  [Extensions to Statement](#page24)s

[page24](#page24)  [Extensions to Function Definitio](#page24)n

[page25](#page25)  [Extensions to Native ECMAScript Object](#page25)s

1. **Extensions to Lexical Conventions**

The following section defines Internet Explorer ECMAScript extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) lexical conventions.

The extensions are as follows:

[page10](#page10)  [Conditional Source Text Processin](#page10)g

[page11](#page11)  [Global Stat](#page11)e

[page11](#page11)  [Conditional Processing Algorith](#page11)m

[page21](#page21)  [Extensions to Numeric Literal](#page21)s

[page21](#page21)  [Extensions to String Literal](#page21)s

1. **Conditional Source Text Processing**

When converting source text into input elements, Internet Explorer ECMAScript first does the processing necessary to remove or replace any conditional text spans and then does the input element conversion using the results of that processing as the actual source text input to the identification of lexical input elements.

Each *Program* (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 14), whether presented as either a discrete source text or as the argument to the eval built-in function, and each *FunctionBody* (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 13) processed by the standard built-in Function constructor ( [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.3.2.1) has conditional source text processing performed independently upon it.

*NOTE*



*10 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

*This specification defines conditional source text processing as if it were performed over an entire source text prior to any input element identification. It is an unobservable implementation detail whether this processing is actually performed in that manner or whether it is performed incrementally interweaved with input element identification.*

1. **Global State**

The following state is shared by the conditional source text processing of all independent source texts that make up an ECMAScript program (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 14). The state is initialized prior to the first such processing as follows:

1. *SubstitutionEnabled* **Boolean** flag with an initial value of **false**.

2. *CCvariables* A set of associations between string valued keys and values. The keysare strings. The values may be either ECMAScript **Number** ( [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 8.5) or **Boolean** ([[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963)section8.3) values. The initial associations are defined in the followingtable.

|  |  |
| --- | --- |
| Key | Initial Value |
|  |  |
| **"\_win32"** | Defined as **true** if this Internet Explorer ECMAScript implementation is a Microsoft |
|  | 32-bit–based implementation. Otherwise, this association is not initially defined. |
|  |  |
| **"\_win64"** | Defined as **true** if this Internet Explorer ECMAScript implementation is a Microsoft |
|  | 64-bit–based implementation. Otherwise, this association is not initially defined. |
|  |  |
| **"\_x86"** | Defined as **true** when running on a processor using the x86-based architecture. |
|  | Otherwise, this association is not initially defined. |
|  |  |
| **"\_ia64"** | Defined as **true** when running on a processor using the Itanium 64-bit |
|  | architecture. Otherwise, this association is not initially defined. |
|  |  |
| **"\_amd64"** | Defined as **true** when running on a processor using the x64 architecture. |
|  | Otherwise, this association is not initially defined. |
|  |  |
| **"\_jscript"** | **true** |
|  |  |
| **"\_jscript\_build"** | **Number** value that identifies the specific build of the Internet Explorer ECMAScript |
|  | implementation that is running. |
|  |  |
| **"\_jscript\_version"** | **Number** value that represents the version of the Internet Explorer ECMAScript |
|  | language implementation. The value 9 indicates that the implementation only |
|  | supports features of the Internet Explorer 9 ECMAScript language. |
|  |  |
| **"\_microsoft"** | Defined as true when running on a Microsoft ECMAScript implementation provided |
|  | by Microsoft. Otherwise, this association is not initially defined. |
|  |  |

1. **Conditional Processing Algorithm**

For each source text to be processed, let *source* be the original source text (a sequence of Unicode characters) and let *output* initially be an empty sequence of Unicode characters. Let *IfNestingLevel* be 0.

Processing of *source* proceeds by recognizing specific input elements from *source* and then taking specified actions. The processing is organized into several states. The specific input elements that are recognized and the subsequent semantic action that is taken varies among states. The semantic action taken for a recognized input element may include transitioning to a different state. Processing



*11 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

of a source text begins by recognizing *CCInputElementState0* if *SubstitutionEnabled* is **false** and

*CCInputElementState1* if *SubstitutionEnabled* is**true**.

The input elements for conditional processing are defined by the following grammar, which has Unicode characters as terminal symbols. Some rules of the grammar are defined using rules of the ECMAScript lexical grammar.

**Syntax**

NOTE:

*CCInputElementState0* is recognized during top-level conditional processing when

*SubstitutionEnabled* is**false**. When recognizing a *RegularExpressionLiteral* in this state, thecontextual distinction between *RegularExpressionLiteral* and *DivPunctuator* (see  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)] section 7) must be respected.

*CCInputElementState0* **::**

*RegularExpressionLiteralStringLiteralCCOnCCSet0CCIf0CCMultiLineComment0CCSingleLinecomment*

*0SourceCharacter*

*CCOn* **::**

**@** *CCOnId***/\*@** *CCOnId***//@** *CCOnId*

*CCOnId* **::**

**cc\_on** [lookahead*IdentifierPart*]

*CCSet0* **::**

**@set** [lookahead*IdentifierPart*]

*CCIf0* **::**

**@if** [lookahead*IdentifierPart*]

*CCMultiLineComment0* **::**

**/\*** [lookahead ≠*CCOnId*]*MultiLineCommentChars*opt **\*/**

*SingleLineComment0* **::**

**//** [lookahead ≠*CCOnId*]*SingleLineCommentChars*opt

**Semantics**

If *CCInputElementState0* cannot be recognized because there are no remaining characters in *source*, then Conditional Source processing is completed and the characters of the output supply theUnicode characters for subsequent input element processing. If *CCInputElementState0* cannot be recognized and there are characters in *source*, a **SyntaxError** exception is thrown.

The productions *CCInputElementState0* :: *RegularExpressionLiteral*, *CCInputElementState0* :: *StringLiteral*, *CCInputElementState0* :: *CCMultiLineComment0*, *CCInputElementState0* :: *CCSingleLinecomment0*, and *CCInputElementState0* :: *SourceCharacter* upon recognition performthe following actions:



*12 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Append to the end of output, in left-to-right sequence, the Unicode characters from *source* that were recognized by the production. Remove the recognized characters from *source*.
2. Use *CCInputElementState0* to recognize the next input element from *source*.

The production *CCInputElementState0* :: *CCOn* upon recognition performs the following actions:

1. Set SubstitutionEnable to **true**.
2. Append a <SP> character to the end of output. Remove the recognized characters from source.
3. Use CCInputElementState1 to recognize the next input element from source.

The production *CCInputElementState0* :: *CCSet0* upon recognition performs the following actions:

1. Set SubstitutionEnable to **true**.
2. Append a <SP> character to the end of output. Remove the recognized characters from source.
3. Use CCInputElementStateSetLHS to recognize the next input element from source.

The production *CCInputElementState0* :: *CCIf0* upon recognition performs the following actions:

1. Set SubstitutionEnable to **true**.
2. Append a <SP> character to the end of output. Remove the recognized characters from source.
3. Increment the value of IfNestingLevel by 1.
4. Use CCInputElementStateIfPredicate to recognize the next input element from source.

**Syntax**

NOTE:

*CCInputElementState1* is recognized during active conditional processing when *SubstitutionEnabled* is **true**. This may be at the top level or in the clause of an **@if** statement that represents the "true" condition. When recognizing a *RegularExpressionLiteral* in this state the contextual distinction between *RegularExpressionLiteral* and *DivPunctuator* (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 7) must be respected.

*CCInputElementState1* **::**

*RegularExpressionLiteralStringLiteralCCOnCCSet1CCIf1CCElif1CCElse1CCEnd1CC*Substitution*1CCSta rtMarkerCCEndMarkerCCMultiLineComment1CCSingleLinecomment1SourceCharacter*

*CCSet1* **::**

@set [lookahead IdentifierPart ] /\*@set [lookahead IdentifierPart ] //@set [lookahead IdentifierPart ]

*CCIf1* **::**

@if [lookahead IdentifierPart ] /\*@if [lookahead IdentifierPart ] //@if [lookahead IdentifierPart ]



*13 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

*CCElif1* **::**

@elif [lookahead IdentifierPart ] /\*@elif [lookahead IdentifierPart ] //@elif [lookahead IdentifierPart ]

*CCElse1* **::**

@else [lookahead IdentifierPart ] /\*@else [lookahead IdentifierPart ] //@else [lookahead IdentifierPart ]

*CCEnd1* **::**

@end [lookahead IdentifierPart ] /\*@end [lookahead IdentifierPart ] //@end [lookahead IdentifierPart ]

*CCSubstitution1* **::**

@ CCSubIdentifier /\*@ CCSubIdentifier //@ CCSubIdentifier

*CCStartMarker* **::**

/\*@

//@

*CCEndMarker* **::**

@\*/

*CCMultiLineComment1* **::**

/\* [lookahead ≠ @ ] MultiLineCommentCharsopt \*/

*SingleLineComment1* **::**

// [lookahead ≠ @] SingleLineCommentCharsopt

*CCSubIdentifer* **::**

[lookahead CCKeyword ] IdentifierName

*CCKeyword* **::**



*14 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

cc\_on setifelif elseend

**Semantics**

If *CCInputElementState1* cannot be recognized because there are no remaining characters in *source*, then Conditional Source processing is completed and the characters of the output supply theUnicode characters for subsequent input element processing. If *CCInputElementState1* cannot be recognized and there are characters in *source*, a **SyntaxError** exception is thrown.

The productions *CCInputElementState1* :: *RegularExpressionLiteral*, *CCInputElementState1* :: *StringLiteral*, *CCInputElementState1* :: *CCMultiLineComment1*, *CCInputElementState1* :: *CCSingleLinecomment1*, and *CCInputElementState1* :: *SourceCharacter* upon recognition performthe following actions:

1. Append to the end of output, in left-to-right sequence, the Unicode characters from *source* that were recognized by the production. Remove the recognized characters from *source*.
2. Use *CCInputElementState1* to recognize the next input element from *source*.

The productions *CCInputElementState1* :: *CCOn*, *CCInputElementState1* :: *CCStartMarker*,

*CCInputElementState1* :: *CCEndMarker* upon recognition perform the following actions:

1. Append a <SP> character to the end of output. Remove the recognized characters from *source*.
2. Use *CCInputElementState1* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCSet1* upon recognition performs the following actions:

1. Append a <SP> character to the end of output. Remove the recognized characters from *source*.
2. Use *CCInputElementStateSetLHS* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCIf1* upon recognition performs the following actions:

1. Append a <SP> character to the end of output. Remove the recognized characters from *source*.
2. Increment the value of *IfNestingLevel* by 1.
3. Use *CCInputElementStateIfPredicate* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCElif1* upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. If *IfNestingLevel* is 0, throw a **SyntaxError** exception.
3. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCElse1* upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. If *IfNestingLevel* is 0, throw a **SyntaxError** exception.
3. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCEnd* upon recognition performs the following actions:



*15 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Append a <SP> character to the end of output. Remove the recognized characters from *source*.
2. If *IfNestingLevel* is 0, throw a **SyntaxError** exception.
3. Decrement the value of *IfNestingLevel* by 1.
4. Use *CCInputElementState1* to recognize the next input element from *source*.

The production *CCInputElementState1* :: *CCSubstitution1* upon recognition performs the following actions:

1. Let var be the string of characters recognized as the *CCSubIdentifier* element of *CCSubstitution1*.
2. If the value of var is a key of *CCVariables*, then let the value be the associated value. Otherwise, let value be the string "NaN".
3. Let value be *ToString(value).*
4. Append the characters of the string value of value to the end of output.
5. Remove the recognized characters from *source*.
6. Use *CCInputElementStateIfPredicate* to recognize the next input element from *source*.

**Syntax**

NOTE:

*CCInputElementStateSetLHS* is recognized during active conditional processing of the body of an**@set** statement.

*CCInputElementStateSetLHS* **::**

*WhiteSpace*opt**@** *IdentifierName WhiteSpace*opt**=** *CCExpression*

**Semantics**

If *CCInputElementStateSetLHS* cannot be recognized a **SyntaxError** exception is thrown.

The production *CCInputElementStateSetLHS* :: *WhiteSpaceopt* @ *IdentifierName WhiteSpaceopt* =

*CCExpression* upon recognition performs the following actions:

1. Let *setName* be the string of characters recognized as the *IdentifierName* element of

*CCSubstitution1*.

1. Let *value* be the result of evaluating *CCExpression*.
2. Create an association within *CCVariables* where the key is the string value of *setName* and where the value is *value*. If an association with that key already exists, replace it.
3. Remove the recognized characters from *source*.
4. Use *CCInputElementState1* to recognize the next input element from *source*.

**Syntax**

NOTE:



*16 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

*CCInputElementStateIfPredicate* is recognized during active conditional processing of the predicateportion of an **@if** or **@elif** statement.

*CCInputElementStateIfPredicate* **::**

*WhiteSpace*opt**(** *CCExpression WhiteSpace*opt**)**

**Semantics**

If *CCInputElementStateIfPredicate* cannot be recognized, a **SyntaxError** exception is thrown.

The production *CCInputElementStateSetIfPredicate* :: *WhiteSpaceopt* ( *CCExpression WhiteSpaceopt*

) upon recognition performs the following actions:

1. Let *predicate* be the result of evaluating *CCExpression*.
2. Increment the value of *IfNestingLevel* by 1.
3. Set *SkippedIfNestingLevel* to 0.
4. Remove the recognized characters from *source*.
5. If *ToBoolean*(predicate) is true, then use *CCInputElementState1* to recognize the next input element from *source*.
6. Otherwise, use *CCInputElementStateFalseThen* to recognize the next input element from *source*.

**Syntax**

NOTE:

*CCInputElementStateFalseThen* is recognized during processing of false clauses of an**@if**statementfor which the true clause has not yet been processed. The current clause may be a "then" clause, an **@elif** clause, or an **@else** clause.

*CCInputElementStateFalseThen* **::**

@if [lookahead IdentifierPart ] @elif [lookahead IdentifierPart ] @else [lookahead IdentifierPart ] @end [lookahead IdentifierPart ] SourceCharacter

**Semantics**

If *CCInputElementStateFalseThen* cannot be recognized, a **SyntaxError** exception is thrown.

The production *CCInputElementStateFalseThen* :: *@if* [lookahead *IdentifierPart*] upon recognition performs the following actions:

1. Increment the value of *SkippedIfNestingLevel* by 1.
2. Remove the recognized characters from *source*.
3. Use *CCInputElementStateFalseThen* to recognize the next input element from *source*.

The production *CCInputElementStateFalseThen* :: *@elif* [lookahead *IdentifierPart*] upon recognition performs the following actions:



*17 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Remove the recognized characters from *source*.
2. If *SkippedIfNestingLevel* > 0, then use *CCInputElementStateFalseThen* to recognize the next input element from *source*.
3. Otherwise, use *CCInputElementStateIfPredicate* to recognize the next input element from *source*.

The production *CCInputElementStateFalseThen* :: *@else* [lookahead *IdentifierPart*] upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. If *SkippedIfNestingLevel* > 0, then use *CCInputElementStateFalseThen* to recognize the next input element from *source*.
3. Otherwise, use *CCInputElementState1* to recognize the next input element from *source*.

The production *CCInputElementStateFalseThen* :: *@end* [lookahead *IdentifierPart*] upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. If *SkippedIfNestingLevel* is 0, then go to step 6.
3. Decrement the value of *SkippedIfNestingLevel* by 1.
4. Use *CCInputElementStateFalseThen* to recognize the next input element from *source*.
5. Return.
6. Decrement the value of *IfNestingLevel* by 1.
7. Use *CCInputElementState1* to recognize the next input element from *source*.

The production *CCInputElementStateFalseThen* :: *SourceCharacter* upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. Use *CCInputElementStateFalseThen* to recognize the next input element from *source*.

**Syntax**

NOTE:

*CCInputElementStateFalseThen* is recognized during processing of false clauses of an**@if**statementfor which the true clause has already been processed. It is also used during processing of all clauses of a @if statement that is nested within a false clause of an enclosing **@if** statement. The current clause may be a "then" clause, an **@elif** clause or an **@else** clause.

*CCInputElementStateFalseIfTail* **::**

@if [lookahead IdentifierPart ] @elif [lookahead IdentifierPart ] @else [lookahead IdentifierPart ] @end [lookahead IdentifierPart ] SourceCharacter



*18 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**Semantics**

If *CCInputElementStateFalseIfTail* cannot be recognized, a **SyntaxError** exception is thrown.

The production *CCInputElementStateFalseIfTail* :: *@if* [lookahead *IdentifierPart*] upon recognition performs the following actions:

1. Increment the value of *SkippedIfNestingLevel* by 1.
2. Remove the recognized characters from *source*.
3. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.

The productions *CCInputElementStateFalseIfTail* :: *@elif* [lookahead *IdentifierPart*] and *CCInputElementStateFalseIfTail* :: *@else* [lookahead *IdentifierPart*] upon recognition perform thefollowing actions:

1. Remove the recognized characters from *source*.
2. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.

The production *CCInputElementStateFalseIfTail* :: *@end* [lookahead *IdentifierPart*] upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. If *SkippedIfNestingLevel* is 0, then go to step 6.
3. Decrement the value of Skipped*IfNestingLevel* by 1.
4. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.
5. Return.
6. Decrement the value of *IfNestingLevel* by 1.
7. Use *CCInputElementState1* to recognize the next input element from *source*.

The production *CCInputElementStateFalseIfTail* :: *SourceCharacter* upon recognition performs the following actions:

1. Remove the recognized characters from *source*.
2. Use *CCInputElementStateFalseIfTail* to recognize the next input element from *source*.

**Syntax**

*CCExpression* ::

CCLogicalANDExpression CCExpression WhiteSpaceopt ||

CCLogicalANDExpression

*CCLogicalANDExpression* ::

CCBitwiseORExpressionCCcLogicalANDExpression WhiteSpaceopt

* CCBitwiseORExpression



*19 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

*CCBitwiseORExpression* **::**

CCBitwiseXORExpressionCCBitwiseORExpression WhiteSpaceopt | CCBitwiseXORExpression

*CCBitwiseXORExpression* **::**

CCBitwiseANDExpressionCCBitwiseXORExpression WhiteSpaceopt ^ CCBitwiseANDExpression

*CCBitwiseANDExpression* **::**

CCEqualityExpressionCCBitwiseANDExpression WhiteSpaceopt & CCEqualityExpression

*CCEqualityExpression* **::**

CCRelationalExpressionCCEqualityExpression WhiteSpaceopt == CCRelationalExpressionCCEqualityExpression WhiteSpaceopt!= CCRelationalExpressionCCEqualityExpression WhiteSpaceopt ===

CCRelationalExpressionCCEqualityExpression WhiteSpaceopt !== CCRelationalExpression

*CCRelationalExpression* **::**

CCShiftExpressionCCRelationalExpression WhiteSpaceopt <

CCShiftExpressionCCRelationalExpression WhiteSpaceopt > CCShiftExpressionCCRelationalExpression WhiteSpaceopt <= CCShiftExpressionCCRelationalExpression WhiteSpaceopt >= CCShiftExpression

*CCShiftExpression* **::**

CCAdditiveExpressionCCShiftExpression WhiteSpaceopt << CCAdditiveExpressionCCShiftExpression WhiteSpaceopt >> CCAdditiveExpressionCCShiftExpression WhiteSpaceopt >>> CCAdditiveExpression

*CCAdditiveExpression* **::**

CCMultiplicativeExpressionCCAdditiveExpression WhiteSpaceopt +

CCMultiplicativeExpressionCCAdditiveExpression WhiteSpaceopt – CCMultiplicativeExpression

*CCMultiplicativeExpression* **::**

CCUnaryExpressionCCMultiplicativeExpression WhiteSpaceopt \*

CCUnaryExpressionCCMultiplicativeExpression WhiteSpaceopt /

CCUnaryExpressionCCMultiplicativeExpression WhiteSpaceopt % CCUnaryExpression

*UnaryExpression* **::**

CCPrimaryExpressionWhiteSpaceopt + CCUnaryExpressionWhiteSpaceopt -

CCUnaryExpressionWhiteSpaceopt ~ CCUnaryExpressionWhiteSpaceopt! CCUnaryExpression

*CCPrimaryExpression* **::**



*20 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

CCVariableCCLiteralWhiteSpaceopt ( Expression )

*CCLiteral* **::**

WhiteSpaceopt true [lookahead IdentifierPart ]

WhiteSpaceopt false [lookahead IdentifierPart ]

WhiteSpaceopt Infinity [lookahead IdentifierPart ]

WhiteSpaceopt NumericLiteral

*CCVariable* **::**

WhiteSpaceopt @ IdentifierName

**Semantics**

Unless otherwise specified in this section, the productions of *CCExpression* are evaluated using the same semantic rules as the analogous productions of the ECMAScript syntactic grammar for Expression in  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 11. However, only values of types **Number** and **Boolean** can occur during the evaluation of *CCExpression* productions, so any semantic steps that are relative to other types of values are not relevant.

The production *CCLitera*l :: *WhiteSpaceopt* true [lookahead *IdentifierPart*] is evaluated by returning the value true.

The production *CCLiteral* :: *WhiteSpaceopt* false [lookahead *IdentifierPart*] is evaluated by returning the value false.

The production *CCLiteral* :: *WhiteSpaceopt* Infinity [lookahead *IdentifierPart*] is evaluated by returning the value +∞.

The production *CCVariable* :: WhiteSpaceopt @ *IdentifierName* is evaluated by performing the following steps:

1. Let *var* be the string of characters recognized as the *IdentifierName* element of *CCVariable*.
2. If the value of *var* is a key of *CCVariables*, then let *value* be the associated value. Otherwise, let *value* be "NaN".
3. Return *value*.
4. **Extensions to Numeric Literals**

Internet Explorer ECMAScript supports the Numeric Literal extensions that are defined by  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) Annex B, section B.1.1.

1. **Extensions to String Literals**

Internet Explorer ECMAScript supports the String Literal extensions that are defined by  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) Annex B, section B.1.2.

In addition, the production *EscapeSequence* is extended to include the characters 8 and 9 as right-hand-side alternatives, as follows:

*EscapeSequence* **::**



*21 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

CharacterEscapeSequence OctalEscapeSequence HexEscapeSequence UnicodeEscapeSequence 8 9

The character values (CV) are defined as follows:

* 1. The CV of *EscapeSequence* **::** 8 is a character 8 (Unicode value 0038).
  2. The CV of *EscapeSequence* **::** 9 is a character 9 (Unicode value 0039).

1. **Extensions to Types**

The following section defines an Internet Explorer ECMAScript extension to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) types.

1. **SafeArray Type**

The **SafeArray** type is the set of all references to Microsoft COM SAFEARRAY data structures.

**SafeArray** values can be created only by host objects and host functions. SafeArray values can bemanipulated similarly to other ECMAScript data types.

1. **VarDate Type**

The **VarDate** type is the set of all references to Microsoft COM VARIANT data structures that have a VARTYPE enumeration value of VT\_DATE.

**VarDate** values can be created only by host objects and host functions, or by calling the **getVarDate** method by using the prototype property of the **Date** object: **Date**.**prototype**.**getVarDate**. **VarDate** values can be manipulated similarly to other ECMAScriptdata types.

1. **Extensions to Type Conversion and Testing**

The following extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) are necessary to support the **SafeArray** and **VarDate** extended types.

|  |  |  |
| --- | --- | --- |
| *Conversion operation* | *Argument* | *Operation* |
|  | *type* |  |
|  |  |  |
| **ToPrimitive** | **SafeArray** | Returns the input argument (no conversion is applied). |
|  |  |  |
| **ToPrimitive** | **VarDate** | Returns the input argument (no conversion is applied). |
|  |  |  |
| **ToBoolean** | **SafeArray** | Returns a value of **false**. |
|  |  |  |
| **ToBoolean** | **VarDate** | Returns a value of **false**. |
|  |  |  |
| **ToNumber** | **SafeArray** | Throws a **TypeError** exception. |
|  |  |  |
| **ToNumber** | **VarDate** | Returns the **Number** value that represents the internal |
|  |  | numerical value of the **VT\_Date** value. |
|  |  |  |
| **ToString** | **SafeArray** | Applies the following steps: |
|  |  |  |



*22 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

|  |  |  |
| --- | --- | --- |
| *Conversion operation* | *Argument* | *Operation* |
|  | *type* |  |

|  |  |  |
| --- | --- | --- |
|  |  | 1. Let *objValue* be **ToObject**(input *argument*). |
|  |  | 2. Returns the value of **ToString**(*objValue*). |
|  |  |  |
| **ToString** | **VarDate** | Returns a **String** value that contains a representation of the |
|  |  | **VarDate** value in the same representational format as |
|  |  | **Date**.**prototype**.**toString**. |
|  |  | For more information, see [[ECMA-262/5],](http://go.microsoft.com/fwlink/?LinkId=185963) Section 15.9.5.2. |
|  |  |  |
| **ToObject** | **SafeArray** | Creates a new **VBArray** object in the same manner as the |
|  |  | following ECMAScript expression: |
|  |  | new VBArray(argument) |
|  |  | In this case, *argument* is a value of type **SafeArray**. |
|  |  |  |
| **ToObject** | **VarDate** | Throws a **TypeError** exception. |
|  |  |  |
| **CheckObjectCoercible** | **SafeArray** | Returns with no return value. |
|  |  |  |
| **CheckObjectCoercible** | **VarDate** | Throws a **TypeError** exception. |

1. **Extensions to Executable Code and Execution Contexts**

The following section defines Internet Explorer ECMAScript extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) executable code and execution contexts.

The extensions are as follows:

[page23](#page23)  [Extensions to Declaration Binding Instantiatio](#page23)n

1. **Extensions to Declaration Binding Instantiation**

Internet Explorer ECMAScript allows a *FunctionDeclaration* language syntactic element to appear anywhere that a *Statement* can appear. *FunctionDeclaration* items are processed during step 5 of the Declaration Binding Instantiation algorithm (which is defined by  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)], section 10.5).

However, a *FunctionDeclaration* item that defines an event handler is excluded from the processing of step 5. Such a *FunctionDeclaration* item is evaluated when an ECMAScript *SourceElement* production is evaluated.

1. **Extensions to Expressions**

The following section defines Internet Explorer ECMAScript extensions to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) expressions.

1. **Extensions to typeof Operator**

Internet Explorer ECMAScript adds the following **typeof** operator results to Table 20 in  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5],](http://go.microsoft.com/fwlink/?LinkId=185963) section 11.4.3.

|  |  |
| --- | --- |
| *Typeof* ***val*** | *Result* |
|  |  |
| **SafeArray** | "unknown" |
|  |  |
| **VarDate** | "date" |
|  |  |



*23 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **Extensions to Statements**

The following section defines an Internet Explorer ECMAScript extension to  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)] statements.

1. **Extension Grammar Production for Statement**

Internet Explorer ECMAScript adds a *FunctionDeclaration* language syntactic element as an additional alternative to the *Statement* grammar production of  [[ECMA-262/5],](http://go.microsoft.com/fwlink/?LinkId=185963) section 12:

**Syntax Extension**

Statement**:**

*FunctionDeclaration*

A *FunctionDeclaration* element can occur in any context where a *Statement* production is required. The semantics of such declarations are specified in section  [2.4.1](#page23) of this document.

1. **Extensions to Function Definition**

The following section defines an Internet Explorer ECMAScript extension to  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) functions.

1. **Function Definition Used As a Statement**

**Semantic Extensions**

Internet Explorer ECMAScript allows a *FunctionDeclaration* element to be evaluated as a *Statement* production, as follows:

*FunctionDeclaration* **: function** *Identifier* **(** *FormalParameterList*opt**) {** *FunctionBody* **}**

When this production is evaluated, the following step is performed:

Return (**normal**, **empty**, **empty**).

1. **Event Handler Function Definitions**

Internet Explorer ECMAScript adds an additional alternative to the *FunctionDeclaration* grammar production of  [[ECMA-262/5],](http://go.microsoft.com/fwlink/?LinkId=185963) section 13, as follows:

**Syntax Extension**

*FunctionDeclaration* **:**

**function** *Identifier* **(** *FormalParameterList*opt **) {** *FunctionBody* **}**

**function** *ObjectPath* **::** *Identifier* **(** *FormalParameterList*opt **) {** *FunctionBody* **}**

*ObjectPath* **:**

*Identifier*

*ObjectPath NameQualifier Identifier*

*NameQualifier* **:**.

**Semantic Extensions**



*24 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

Internet Explorer ECMAScript allows a *FunctionDeclaration* element to be evaluated as a *Statement* production, as follows:

*FunctionDeclaration* **: function** *ObjectPath* **::** *Identifier* **(** *FormalParameterList*opt**) {** *FunctionBody* **}**

When this production is evaluated, the following steps are performed:

1. Let *p* be the result of the evaluation of *ObjectPath*.
2. Let *o* be **ToObject**(**GetValue**(*p*)).
3. If *o* is not a host object that supports event attachment, throw a **TypeError** exception.
4. Let *eventName* be a string that contains the text of *Identifier*.
5. Let *h* be the result of creating a new **Function** object, as specified in  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)], section 13.2, with the parameters specified by *FormalParameterList*opt and the body specified by

*FunctionBody*.

* 1. Pass in the **VariableEnvironment** component of the running execution context as the *Scope*.
  2. Pass in a value of **true** as the *Strict* flag if the *FunctionDeclaration* element is contained in strict code or if its *FunctionBody* element is strict code.

1. Perform event handler attachment of *h* to *o* by using *eventName* as the event name.
2. Return (**normal**, **empty**, **empty**).

An event handler function *h* is attached to a host object *o* with *eventName n* as follows:

1. If *o* implements the  [IBindEventHandler](http://msdn.microsoft.com/en-us/library/56zc7scb(VS.85).aspx) COM interface ( [http://msdn.microsoft.com/en](http://msdn.microsoft.com/en-us/library/56zc7scb(VS.85).aspx)- [us/library/56zc7scb(VS.85).asp](http://msdn.microsoft.com/en-us/library/56zc7scb(VS.85).aspx)x), perform the following actions:
   1. Call the **BindHandler** COM method of *o*, passing arguments *n* and the function entry point *h*. This call hooks up the direct event.
   2. Return.
2. If *o* does not implement the **IBindEventHandler** COM interface, retain the information (*o*, *n*, and *h*), and defer the event binding until the script engine is placed into "connected" mode, as defined by the SCRIPTSTATE\_CONNECTED constant value of the Microsoft Windows Script Technologies SCRIPTSTATE enumeration ( [http://msdn.microsoft.com/en](http://msdn.microsoft.com/en-us/library/f7z7cxxa(VS.85).aspx)- [us/library/f7z7cxxa(VS.85).asp](http://msdn.microsoft.com/en-us/library/f7z7cxxa(VS.85).aspx)x). When the script engine is placed into the connected mode, the retained information is used to bind the event with an event sinking process. The event binding is performed immediately if the script is already in connected mode.
3. Return.

The **IConnectionPointContainer** COM interface ( [http://msdn.microsoft.com/en](http://msdn.microsoft.com/en-us/library/ms683857(VS.85).aspx)- [us/library/ms683857(VS.85).asp](http://msdn.microsoft.com/en-us/library/ms683857(VS.85).aspx)x) is used to perform the event binding in step 2, regardless of whether the binding is performed immediately or is deferred.

1. **Extensions to Native ECMAScript Objects**

Internet Explorer ECMAScript defines extensions to the native ECMAScript objects of  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)]. These extensions are described in the following sections.



*25 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **Function Properties of the Global Object**

Internet Explorer ECMAScript defines additional properties of the Global object of  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)]. These properties are described in the following sections.

1. **ScriptEngine**

When the ScriptEngine function is called, it returns a string value that specifies the implementation-defined name of the ECMAScript implementation that is executing the call. The Internet Explorer ECMAScript implementations within Internet Explorer 9 always return the string "JScript".

1. **ScriptEngineBuildVersion**

When the ScriptEngineBuildVersion function is called, it returns a value that uniquely identifies the specific build of the ECMAScript implementation that is executing the call.

1. **ScriptEngineMajorVersion**

When the **ScriptEngineMajorVersion** function is called, it returns a value that identifies the major revision level of the implementation, not the revision level of the ECMAScript or JavaScript language specification that is currently supported by the implementation.

An implementation of Internet Explorer ECMAScript that supports distinct document modes (that separately implement other versions of the language, such as JScript 5.7 and JScript 5.8 functionality) can return a single value that does not vary among modes. The return value cannot be used as a reliable indicator of the availability or lack of availability of specific language features.

The ECMAScript implementations within Internet Explorer 9 always return a value of 9, even when Internet Explorer 9 is operating in Quirks, IE7, or IE8 document modes.

1. **ScriptEngineMinorVersion**

When the ScriptEngineMinorVersion function is called, it returns a value that identifies the minor revision level of the implementation, not the revision level of the ECMAScript or JavaScript language specification that is currently supported by the implementation. This return value cannot be used as a reliable indicator of the availability or lack of availability of specific language features.

The ECMAScript implementation within Microsoft Internet Explorer 9 always returns a value of 0, even when Internet Explorer 9 is operating in Quirks, IE7, or IE8 document modes.

1. **CollectGarbage**

When the **CollectGarbage** function is called, the Internet Explorer ECMAScript implementation may attempt to reclaim unused or unneeded resources that are associated with the currently running application. Whether or not any action is actually taken depends on the current state of the execution environment and the resource management strategies and heuristics used by the implementation. An application may call this function to request that any such pending reclamation activities be completed immediately. However, an Internet Explorer ECMAScript implementation is not required to honor such a request.

1. **Constructor Properties of the Global Object**

Internet Explorer ECMAScript defines the following additional constructor properties of the Global object:



*26 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Debug
2. Enumerator
3. VBArray
4. ActiveXObject
5. **Properties of Function Instances**

Internet Explorer ECMAScript defines additional properties of Function instances of  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)]. These properties are described in the following sections.

1. **The arguments Property**

The value of the arguments property of a function instance is null. This property has the attributes { **[[Configurable]]**: **true**, **[[Writable]]**: **false**, **[[Enumerable]]**: **false** }. However, function instances also have a special [[Get]] internal method which in certain circumstances will return a value other than null when accessing the arguments property.

1. **The caller Property**

The value of the caller property of a function instance is null. This property has the attributes

{ **[[Configurable]]**: **true**, **[[Writable]]**: **false**, **[[Enumerable]]**: **false** }. However, function instances also have a special [[Get]] internal method which in certain circumstances will return a value other than null when accessing the caller property.

1. **The [[Get]] (P) Method of a Function Object**

When the [[Get]] method of *F* is called with value *P*, the following steps are taken:

1. If *P* is the string 'arguments', take the following steps:
   1. If an active execution context for *F* does not exist, go to step 3.
   2. Let *X* be the most recently created active execution context for *F*.
   3. If *X* is marked as having a partially accessible arguments object, let *A* be the original arguments object for *X*; otherwise, let *A* be the value of the property named 'arguments' of the variable object of *X*.

**Note:** JScript 5.x under Internet Explorer 9 (in all document modes) marks the currentexecution context as having a partially accessible arguments object when the function’s *FormalParameterList* contains the name 'arguments' or the function’s *FunctionBody* contains adirect reference to the function’s original arguments object or the function’s *FunctionBody* contains a direct call to **eval**.

* 1. Return *A*.

1. If *P* is the string 'caller', take the following steps:
   1. Let *X* be the most recently created active execution context for *F*.
   2. If *X* does not have an execution context to which it could normally exit, return **null**.
   3. Let *R* be the execution context which would become the current execution context if *X* exited normally (not via an exception).



*27 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

* 1. If *R* is an execution context for a built-in function or a host object function, return **null**.
  2. If *R* is an execution context for global code or for eval code, return **null**.
  3. *R* must be an execution context for function code, so let *rf* be the function object that containsthe call that caused *R* to be created.
  4. If *rf* is a strict mode **Function** object, throw a **TypeError** exception.
  5. Return *rf*.

1. Return the result of calling the default **[[Get]]** method ( [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 8.12.3), passing *P* as the argument.
2. **String.prototype HTML Wrapper Properties**

Internet Explorer ECMAScript defines **String**.**prototype** functions that wrap the string value of a **this** value with an HTML tag. The following abstraction is used to specify the behavior of thesefunctions.

The abstract operation **WrapWithHTML** is called with arguments **body**, **tag**, **attribute**, and **data**. The **tag** and **attribute** arguments must be strings; **attribute** and **data** may be omitted. The following steps are performed:

1. Append the character "<" to the characters of *tag*.
2. If *attribute* is not present, go to Step 7.
3. Append to *Result(1)* a single-space character followed by the characters of *attribute*.
4. Append to *Result(3)* the characters "=" and """.
5. Append to *Result(4)* the characters of the string returned by **ToString**(*data*).
6. Append to *Result(5)* the character """.
7. If *attribute* is present, use Result(6); otherwise, use Result(1).
8. Append to *Result(7)* the character ">".
9. Append to *Result(8)* the characters of the string returned by **ToString**(*body*). 10.Append to *Result(9)* the characters "<" and "/".

11.Append to *Result(10)* the characters of *tag*. 12.Append to *Result(11)* the character ">".

13.Return the string value of the characters from *Result(12)*.

1. **String.prototype.anchor(name)**

Return the result of **WrapWithHTML**(this **value**, "A", "NAME", **name**).

1. **String.prototype.big( )**

Return the result of **WrapWithHTML**(this **value**, "BIG").



*28 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **String.prototype.blink( )**

Return the result of **WrapWithHTML**(this **value**, "BLINK").

1. **String.prototype.bold( )**

Return the result of **WrapWithHTML**(this **value**, "B").

1. **String.prototype.fixed( )**

Return the result of **WrapWithHTML**(this **value**, "TT").

1. **String.prototype.fontcolor(color)**

Return the result of **WrapWithHTML**(this **value**, "FONT", "COLOR", **color**).

1. **String.prototype.fontsize(size)**

Return the result of **WrapWithHTML**(this **value**, "FONT", "SIZE", **size**).

1. **String.prototype.italics( )**

Return the result of **WrapWithHTML**(this **value**, "I").

1. **String.prototype.link(url)**

Return the result of **WrapWithHTML**(this **value**, "A", "HREF", **url**).

1. **String.prototype.small( )**

Return the result of **WrapWithHTML**(this **value**, "SMALL").

1. **String.prototype.strike( )**

Return the result of **WrapWithHTML**(this **value**, "STRIKE").

1. **String.prototype.sub( )**

Return the result of **WrapWithHTML**(this **value**, "SUB").

1. **String.prototype.sup( )**

Return the result of **WrapWithHTML**(this **value**, "SUP").

1. **Properties of the Date Prototype Object**

Internet Explorer ECMAScript defines an additional method of the **Date** prototype object of  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5].](http://go.microsoft.com/fwlink/?LinkId=185963) This method is described in the following section.

1. **Date.prototype.getVarDate ( )**

The **getVarDate** method is implemented as follows:

1. Let *t* be the time value.



*29 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. It the value of *t* is "NaN", return a **VarDate** value for which the value of **ToNumber** is "NaN".
2. Otherwise, return a **VarDate** value that corresponds to the time value *t*.
3. **Properties of the RegExp Constructor**

Internet Explorer ECMAScript defines additional properties of the RegExp constructor of  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5].](http://go.microsoft.com/fwlink/?LinkId=185963) These properties are described in the following sections.

1. **RegExp.input**

The initial value of **RegExp**.**input** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **true** }. The value of this

property may be modified by calls to **RegExp.prototype.exec**. The properties **RegExp**.**input** and **RegExp**.**$\_** always have the same value. When one is set to some value, the other is automaticallyalso set to that same value. Unlike most other **RegExp** constructor properties, this property is writable.

1. **RegExp.lastIndex**

The initial value of **RegExp**.**lastIndex** is the number −1. This property shall have the attributes

{ **DontEnum**, **DontDelete**, **ReadOnly** }. Even though this property is {**[[Writable]]**: **false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp.lastMatch**

The initial value of **RegExp**.**lastMatch** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]**: **false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp.lastParen**

The initial value of **RegExp**.**lastParen** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]**: **false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp.leftContext**

The initial value of **RegExp**.**leftContext** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]**: **false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp.rightContext**

The initial value of **RegExp**.**rightContext** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]: false**}, its value may be modified by calls to

**RegExp**.**prototype**.**exec**.

1. **RegExp.$1 - RegExp.$9**

The initial value of **RegExp**.**rightContext** is the empty string. This property shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even



*30 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

though these are {**[[Writable]]: false**} properties, their values may be modified by calls to

**RegExp**.**prototype**.**exec**.

1. **RegExp.$\_**

The initial value of each of the properties **RegExp.$1**, **RegExp**.**$2**, **RegExp**.**$3**, **RegExp**.**$4**,

**RegExp**.**$5**, **RegExp**.**$6**, **RegExp**.**$7**, **RegExp**.**$8**, and **RegExp**.**$9** is the empty string. Theseproperties shall have the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**,

**[[Writable]]**: **false** }. The value of this property may be modified by calls to

**RegExp**.**prototype**.**exec**. The properties **RegExp**.**input** and **RegExp**.**$\_** always have the samevalue. When one of these properties is set to some value, the other is automatically also set to that same value. Unlike most other **RegExp** constructor properties, this property is writable.

1. **RegExp['$&']**

The initial value of **RegExp**['$&'] is the empty string. This property shall have the attributes

{ **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]: false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp['$+']**

The initial value of **RegExp**['$+'] is the empty string. This property shall have the attributes

{ **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]: false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp["$`"]**

The initial value of **RegExp**["$`"] is the empty string. This property shall have the attributes

{ **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]: false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **RegExp["$'"]**

The initial value of **RegExp**["$'"] is the empty string. This property shall have the attributes

{ **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }. Even though this property is {**[[Writable]]: false**}, its value may be modified by calls to **RegExp**.**prototype**.**exec**.

1. **Properties of the RegExp Prototype Object**

Internet Explorer ECMAScript defines additional properties of the **RegExp** prototype object of  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) (see section 15.10.6). These properties are described in the following sections.

1. **RegExp.prototype.compile(pattern, flags)**

If *pattern* is an object *R* that has a **[[Class]]** property "RegExp" and *flags* is undefined, let *P* be the pattern used to construct *R*, and let *F* be the flags used to construct *R*. If *pattern* is an object *R* that has a **[[Class]]** property "RegExp", and *flags* is not undefined, throw a **SyntaxError** exception.

Otherwise, let *P* be "(?:)" if *pattern* is undefined and **ToString**(*pattern*) otherwise, and let *F* be the empty string if *flags* is undefined and **ToString**(*flags*) otherwise.

The global property of this **RegExp** object is set to a **Boolean** value that is **true** if *F* contains the character "g" and that is **false** otherwise.

The ignoreCase property of this **RegExp** object is set to a **Boolean** value that is **true** if *F* contains the character "i" and that is **false** otherwise.



*31 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

The multiline property of this **RegExp** object is set to a **Boolean** value that is **true** if *F* contains the character "m" and that is **false** otherwise.

If *F* contains any character other than "g", "i", or "m", throw a **SyntaxError** exception.

If the characters in *P* do not have the form *Pattern*, throw a **SyntaxError** exception. Otherwise, let the newly constructed object have a **[[Match]]** property obtained by evaluating ("compiling")

*Pattern*.

The source property of this **RegExp** object is set as follows:

1. When pattern is an object *R* that has a **[[Class]]** property of "RegExp", this **RegExp** object is set to the same string value as the value of the source property of pattern. Otherwise, the source property of this **RegExp** object is set to *P*.
2. The **lastIndex** property of this **RegExp** object is set to 0.
3. The **options** property of this **RegExp** object is set as described in section  [2.8.8.1](#page32) of this document.
4. This **RegExp** object is optimized using the assumption that it will be executed multiple times.
5. **Properties of RegExp Instances**

Internet Explorer ECMAScript defines an additional property of the **RegExp** instances of  [[ECMA](http://go.microsoft.com/fwlink/?LinkId=185963)- [262/5].](http://go.microsoft.com/fwlink/?LinkId=185963) This property is described in the following section.

1. **options**

The value of the **options** property is a string that specifies the values of the **global**, **ignoreCase**, and **multiline** properties of this **RegExp** instance. If the value of the **ignoreCase** property is **true**, the string contains the character "i". If the value of the **global** property is **true**, the string contains the character "g". If the value of the **multiline** property is **true**, the string contains the character "i". When present, the characters appear in the order "igm". If all of the **global**, **ignoreCase**, and **multiline** properties have the value **false**, the value of this property is the empty string. Thisproperty shall have the attributes { **[[Enumerable]]: false, [[Configurable]]: false,** **[[Writable]]: false** }.

1. **The Error Constructor**

Internet Explorer ECMAScript defines additional behaviors of the **Error** constructor of  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963).

These behaviors are described in the following sections:

[page32](#page32)  [new Error (](#page32))

[page32](#page32)  [new Error (number, message](#page32))

1. **new Error ()**

When the **Error** constructor is called with no arguments, the call is equivalent to calling the **Error** constructor and passing the number zero as the only argument.

1. **new Error(number, message)**

When the **Error** constructor is called with two or more arguments, the following steps are taken:



*32 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. The **[[Prototype]]** property of the newly constructed object is set to the original **Error** prototype object, the one that is the initial value of **Error**.**prototype** (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.11.3.1).
2. The **[[Class]]** property of the newly constructed **Error** object is set to "Error".
3. Let *num* be **ToNumber**(*number*).
4. Let *msg* be **ToString**(*message*).
5. The **description** property of the newly constructed object is set to *msg*.
6. The **message** property of the newly constructed object is set to *msg*.
7. The **name** property of the newly constructed object is set to "Error".
8. The **number** property of the newly constructed object is set to *num*.
9. Return the newly constructed object.
10. **Properties of Error Instances**

Internet Explorer ECMAScript defines additional error instances inherited from the **[[Prototype]]** object of  [[ECMA-262/5].](http://go.microsoft.com/fwlink/?LinkId=185963) These error instances are described in the following sections.

1. **description**

The initial value of **description** is the same as the initial value of **message**.

1. **number**

An **Error** instance only initially has a number property if the first argument passed to the **Error** constructor was a number or could be converted to a number. The initial value of **number** is the number value passed to the constructor.

1. **Properties of NativeError Instances**

**Error** instances inherit properties from their **[[Prototype]]** object and **Error** prototype as specifiedpreviously. In addition, those **NativeError** instances that are created to represent a runtime error that is detected by the Internet Explorer ECMAScript implementation have the following properties.

1. **description**

An **Error** instance only initially has a description property if it is created by the Internet Explorer ECMAScript implementation in response to the occurrence of a runtime error. The initial value of **description** is the same as the initial value of **message**.

1. **number**

An **Error** instance only initially has a **number** property if it is created by the Internet Explorer ECMAScript implementation in response to the occurrence of a runtime error. The initial value of **number** is the number value passed to the constructor.

1. **The Debug Object**

The **Debug** object is a single object that has some named properties, all of which are functions.



*33 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

The value of the internal **[[Prototype]]** property of the **Debug** object is the **Object** prototype object (15.2.3.1). The value of the internal **[[Class]]** property of the **Debug** object is "Object".

The **Debug** object does not have a **[[Construct]]** property; it is not possible to use the **Debug** object as a constructor with the new operator.

The **Debug** object does not have a **[[Call]]** property; it is not possible to invoke the **Debug** object as a function.

1. **Function Properties of the Debug Object**

The **Debug** object inherits properties from the **Object** prototype object as specified previously, and also has the following properties.

1. **write ([ item1 [, item2 [, …]]])**

If a host-dependent debugging facility is available, **ToString** is called once, in order, on each item argument. The result of the call is passed to the debugging facility with the intent that the result be output to the user without the addition of any line terminator characters. The function returns **undefined** regardless of whether or not a debugging facility is present.

The **length** property of the **write** function is 0.

1. **writeln ([ item1 [, item2 [, …]]]))**

If a host-dependent debugging facility is available, **ToString** is called once, in order, on each item argument. The result of the call is passed to the debugging facility with the intent that the result be output to the user without the insertion of any line terminator characters between item results. A line terminator should be output after the last item or if there are no item arguments. The function returns **undefined** regardless of whether a debugging facility is present.

The **length** property of the **write** function is 0.

1. **Enumerator Objects**

**Enumerator** objects provide an alternative mechanism for iterating over the elements of **Array** instances and certain host objects.

For such objects, the order of enumeration is the same as occurs for the **for-in** statement (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 12.6.4).

1. **The Enumerator Constructor Called as a Function**

When **Enumerator** is called as a function rather than as a constructor, it returns **undefined**.

1. **The Enumerator Constructor**

When **Enumerator** is called as part of a new expression, it is a constructor: it initializes the newly created object.

1. **new Enumerator ([collection])**

When the **Enumerator** constructor is called with zero or one argument, the following steps are taken:



*34 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. If *collection* is not present, let *collection* be **undefined**, and then go to step 6.
2. If *collection* is an **Array** instance, go to step 5.
3. If *collection* is a host object that supports an implementation-dependent enumeration protocol, go to step 5.
4. Throw a **TypeError** exception.
5. The **[[EnumerationState]]** property of the newly created object is set to a state indicating that the enumeration is at the first item of the enumeration of *collection*. If *collection* has no enumerable items, the state will indicate that the end of the enumeration has been reached.
6. The **[[Collection]]** property of the newly created object is set to *collection*.
7. The **[[Prototype]]** property of the newly constructed object is set to the original **Error** prototype object, the one that is the initial value of **Enumerator**.**prototype** (see section  [2.8.13.3.1](#page35) of this document).
8. The **[[Class]]** property of the newly constructed **Enumerator** object is set to "Object".
9. Return the newly constructed object.
10. **Properties of the Enumerator Constructor**

The value of the internal **[[Prototype]]** property of the **Enumerator** constructor is the **Function** prototype object (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.3.4).

The value of the **length** property is 7 (seven). In addition, the **Enumerator** constructor has the following property.

1. **Enumerator.prototype**

The initial value of **Enumerator**.**prototype** is the **Enumerator** prototype object (see section  [2.8.13.4](#page35) of this document).

This property has the attributes { **[[Enumerable]]**:**false**, **[[Configurable]]**:**false**, **[[Writable]]**:**false** }.

1. **Properties of the Enumerator Prototype Object**

The **Enumerator** prototype object is itself an **Enumerator** object with a **[[Collection]]** property of undefined, and which does not have an **[[EnumerationState]]** property.

The value of the internal **[[Prototype]]** internal property of the **Enumerator** prototype object is the **Object** prototype object (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) Section 15.2.3.1).

1. **Enumerator.prototype.constructor**

The initial value of **Enumerator**.**prototype**.**constructor** is the built-in **Enumerator** constructor.

1. **Enumerator.prototype.atEnd ( )**
2. If the **this** object is not an **Enumerator** object, throw a **TypeError** exception.
3. Let *collection* be the value of the **this** object’s **[[Collection]]** property.



*35 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. If *collection* is undefined, return **true**.
2. Let *state* be the value of the this object’s **[[EnumerationState]]** property.
3. If *state* indicates that the end of the enumeration has been reached, return **true**.
4. Return **false**.
5. **Enumerator.prototype.item ( )**
6. If the **this** object is not an **Enumerator** object, throw a **TypeError** exception.
7. Let *collection* be the value of the **this** object’s **[[Collection]]** property.
8. If *collection* is undefined, return **undefined**.
9. Let *state* be the value of the this object’s **[[EnumerationState]]** property.
10. If *state* indicates that the end of the enumeration has been reached, return **undefined**.
11. Return the current enumeration item as indicated by **state**.
12. **Enumerator.prototype.moveFirst ( )**
13. If the **this** object is not an **Enumerator** object, throw a **TypeError** exception.
14. Let *collection* be the value of the **this** object’s **[[Collection]]** property.
15. If *collection* is undefined, return **undefined**.
16. Modify the **[[EnumerationState]]** property of the this object to a state indicating that the current enumeration of *collection* is now positioned at the original first item of the enumeration. If the current **[[EnumerationState]]** property indicates that *collection* has no enumerable items, the new *state* will indicate that the end of the enumeration has been reached.
17. Return **undefined**.
18. **Enumerator.prototype.moveNext ( )**
19. If the **this** object is not an **Enumerator** object, throw a **TypeError** exception.
20. Let *collection* be the value of the **this** object’s **[[Collection]]** property.
21. If *collection* is undefined, return **undefined**.
22. Let *state* be the value of the this object’s **[[EnumerationState]]** property.
23. If *state* indicates that the end of the enumeration has been reached, return **undefined**.
24. Modify *state* to a state indicating that the current enumeration of *collection* is now positioned at the next item beyond the current item of the enumeration. The new *state* may indicate that the end of the enumeration has been reached.
25. Update the **[[EnumerationState]]** property of the this object to *state*.
26. Return **undefined**.



*36 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **Properties of Enumerator Instances**

**Enumerator** instances inherit properties from their **[[Prototype]]** object as specified previously.In addition, **Enumerator** instances have an internal **[[Collection]]** property, and they may have an internal **[[EnumeratorState]]** property.

1. **VBArray Objects**

**Enumerator** objects provide an alternative mechanism for iterating over the elements of Arrayinstances and certain host objects.

For such objects, the order of enumeration is the same as the for-in statement (see  [[ECMA-262/5](http://go.microsoft.com/fwlink/?LinkId=185963)] section 12.6.4).

1. **The VBArray Constructor Called as a Function**

When **VBArray** is called as a function, it throws an exception if the argument is not a **SafeArray** value.

1. **VBArray ( value)**

When the **VBArray** function is called, the following steps are taken:

1. If *Type(value)* is **SafeArray**, return *value*.
2. Throw a **TypeError** exception.
3. **The VBArray Constructor**

When **VBArray** is called as part of a new expression, it is a constructor: it initializes the newly created object.

1. **new VBArray ( value )**

When the **VBArray** constructor is called with an argument value of zero or one, the following steps are taken:

1. If *Type(value)* is not **SafeArray**, throw a **TypeError** exception.
2. The **[[SArray]]** property of the newly created object is set to *value*.
3. The **[[Prototype]]** property of the newly constructed object is set to the initial value of the **VBArray prototype** object (see section[2.8.14.3.1](#page38)ofthis document).
4. The **[[Class]]** property of the newly constructed **Error** object is set to Object.
5. Return the newly constructed object.
6. **Properties of the VBArray Constructor**

The value of the internal **[[Prototype]]** property of the **VBArray constructo**r is the **Function** prototype object (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.3.4).

The value of the **length** property is 1. In addition, the **VBArray constructor** has the **VBArray.prototype** property (see section[2.8.14.3.1](#page38)ofthis document).



*37 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **VBArray.prototype**

The initial value of **VBArray.prototype** is the **VBArray prototype object** (see section  [2.8.14.4](#page38) of this document).

This property has the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **true** }.

1. **Properties of the VBArray Prototype Object**

The VBArray prototype object is **VBArray** object with a **[[SArray]]** property that is a **SafeArray** that references a **COM SAFEARRAY** with zero dimensions.

The value of the internal **[[Prototype]]** property of the **VBArray** prototype object is the **Object** prototype object (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.2.3.1).

1. **VBArray.prototype.constructor**

The initial value of **VBArray.prototype.constructor** is the built-in **VBArray** constructor.

1. **VBArray.prototype.dimensions ( )**
2. Call **ToObject**, passing the **this** value as the argument.
3. If *Result(1)* is not a **VBArray** instance, throw a **TypeError** exception.
4. Get the value of the **[[SArray]]** property of *Result(1)*.
5. Return the **Number** that is the number of dimensions of the **COM SAFEARRAY** referenced by Result(3).
6. **VBArray.prototype.getItem ( dim1 [, dim2, [dim3, …]])**
7. Call **ToObject**, passing the **this** value as the argument.
8. If *Result(1)* is not a **VBArray** instance, throw a **TypeError** exception.
9. Get the value of the **[[SArray]]** property of *Result(1)*.
10. If no arguments were passed to this call, or if the number of arguments passed is greater than *Result(3)*, throw a**RangeError**exception.
11. For each argument *dim1* through *dimN*, let *IdimX* be **ToInteger**(*dimX*) where *X* is the numeric suffix of the argument name.
12. For each of *Idim1* through *IdimN*, if *IdimX* is less than the **lower** bound of dimension *X* of the **COM SAFEARRAY** referenced by*Result(3)*, or if*IdimX*is greater than the **upper** bound ofdimension *X*, throw a **RangeError** exception.
13. Return the value of the element identified by array indices *Idim1* through *IdimN* in the **COM** **SAFEARRAY** referenced by*Result(3)*.

The **length** property of the **getItem** function is 1.

1. **VBArray.prototype.lbound ( [dimension] )**

1. Call **ToObject**, passing the **this** value as the argument.



*38 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. If *Result(1)* is not a **VBArray** instance, throw a **TypeError** exception.
2. Get the value of the **[[SArray]]** property of *Result(1)*.
3. If *dimension* is not defined, use a value of 1; otherwise, use **ToInteger**(*dimension*).
4. Get the **Number** that is the number of dimensions of the **COM SAFEARRAY** referenced by

*Result(3)*.

1. If *Result(4)* is less than 1 or greater than *Result(5)*, throw a **RangeError** exception.
2. Return the **Number** that is the lower bound of dimension number *Result(4)* of the **COM** **SAFEARRAY** referenced by*Result(3*).

The **length** property of the **lbound** function is 0.

1. **VBArray.prototype.toArray ( )**

The method copies all the elements of a multi-dimensional **COM SAFEARRAY** into a one-dimensional **ECMAScript** Array instance. When called with no arguments, **toArray** performs the following steps:

1. Call **ToObject**, passing the **this** value as the argument.
2. If Result(1) is not a **VBArray** instance, throw a **TypeError** exception.
3. Get the value of the **[[SArray]]** property of *Result(1)*.
4. Let *SA* be the **COM SAFEARRAY** referenced by *Result(3)*.
5. Let *dim* be the number of dimensions of *SA*.
6. If *dim* is zero, return a new **Array** object that is created as if by evaluating the expression new **Array**(0) using the original **Array** constructor object.
7. Let *size* be the total number of array elements of *SA*.
8. Let *A* be a new **Array** object that is created as if by evaluating the expression new **Array**(*size*) using the original **Array** constructor object.
9. Access the elements of *SA* in row-major order, and store the elements in the array-indexed properties for *A* starting with property 0.

10.Return *A*.

1. **VBArray.prototype.ubound ( [dimension] )**
2. Call **ToObject**, passing the **this** value as the argument.
3. If *Result(1)* is not a **VBArray** instance, throw a **TypeError** exception.
4. Get the value of the **[[SArray]]** property of *Result(1)*.
5. If *dimension* is not defined, use a value of 1; otherwise, use **ToInteger**(*dimension*).
6. Get the **Number** that is the number of dimensions of the **COM SAFEARRAY** referenced by

*Result(3)*.



*39 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. If *Result(4)* is less than 1 or greater than *Result(5)*, throw a **RangeError** exception.
2. Return the **Number** that is the upper bound of dimension number *Result(4)* of the **COM** **SAFEARRAY** referenced by*Result(3)*.

The **length** property of the **ubound** function is 0.

1. **VBArray.prototype.valueOf ( )**
2. Call **ToObject**, passing the **this** value as the argument.
3. If *Result(1)* is not a **VBArray** instance, throw a **TypeError** exception.
4. Get the value of the **[[SArray]]** property of *Result(1)*.
5. Return *Result(3)*.
6. **Properties of VBArray Instances**

A **VBArray** instance inherits properties from the **[[Prototype]]** object, as specified in **VBArray.prototype.valueOf ( )** (see section[2.8.14.4.7](#page40)ofthis document). In addition, **VBArray** instances have an internal **[[SArray]]** property with a value that is the **SafeArray** from which the instance was constructed.

1. **ActiveXObject Objects**

**ActiveXObject** objects provide a mechanism for creating and interacting with host objects providedby Microsoft Windows ActiveX automation servers.

1. **The ActiveXObject Constructor Called as a Function**

When **ActiveXObject** is called as a function, it performs the same argument validation that it performs when it is called as part of a new expression. After successfully completing validation, it always raises an **Error** exception.

1. **ActiveXObject ( name [, location]))**

When the **ActiveXObject** function is called with one or more arguments, the following steps are taken:

1. Call **toPrimitive**(*name*, *hint Number*).
2. If the type of *Result(1)* is not **String**, raise a **TypeError** exception.
3. If *Result(1)* is an empty string, raise a **TypeError** exception.
4. If location is not present go to step 7.
5. Call **toPrimitive**(*location*, *hint Number*).
6. If the type of *Result(5)* is not **String**, raise a **TypeError** exception.
7. Raise an **Error** exception.



*40 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **The ActiveXObject Constructor**

When **ActiveXObject** is called as part of a new expression, it attempts to create a host object that corresponds to a Microsoft Windows ActiveX automation object.

1. **new ActiveXObject (( name [, location]) )**

When the **ActiveXObject** constructor is called with one or more arguments, the following steps are taken:

1. Call **toPrimitive**(*name*, *hint Number*).
2. If the type of *Result(1)* is not **String**, raise a **TypeError** exception.
3. If *Result(1)* is an empty string, raise a **TypeError** exception.
4. If *location* is not present, go to step 7.
5. Call **toPrimitive**(*location*, *hint Number*).
6. If the type of *Result(5)* is not **String**, raise a **TypeError** exception.
7. Attempt to create a host object than can be used to communicate with the application and application-specific object identified by the *Result(1)* String. If location was present, *Result(5)* identifies the server where the application resides; otherwise, the default server (the current machine) is used as the location of the application.
8. If any error occurs during step 7, such that the host object cannot be created, raise an **Error** exception.
9. Return *Result(7)*.

The format of the string values passed as arguments to this constructor are defined by the host operating system.

The object returned by this constructor is a host object. It is not an instance of **ActiveXObject**, and it does not inherit properties from the **ActiveXObject** prototype object or from **Object.prototype**. The specific properties of such objects will vary and are dependent upon the specific argument values passed to this constructor.

1. **Properties of the ActiveXObject Constructor**

The value of the internal **[[Prototype]]** property of the **ActiveXObject** constructor is the **Function** prototype object (see  [[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963) section 15.3.4).

The value of the length property is 1. In addition, the **ActiveXObject** constructor has the **ActiveXObject.prototype** property (see section[2.8.15.3.1](#page41)ofthis document).

1. **ActiveXObject.prototype**

The initial value of **ActiveXObject.prototype** is the **ActiveXObject** prototype object (see section section  [2.8.15.4](#page42) of this document).

This property has the attributes { **[[Enumerable]]**: **false**, **[[Configurable]]**: **false**, **[[Writable]]**: **false** }.



*41 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

The value of this property is not used by the **ActiveXObject** constructor. The value is not used as the **[[Prototype]]** value of host objects returned by **ActiveXConstructor**.

1. **Properties of the ActiveXObject Prototype Object**

The **ActiveXObject** prototype object is an **Object** instance, not an **ActiveXObject** instance.

The value of the internal **[[Prototype]]** property of the **ActiveXObject** prototype object is the **Object** prototype object (see[[ECMA-262/5]](http://go.microsoft.com/fwlink/?LinkId=185963)section15.2.3.1).

1. **ActiveXObject.prototype.constructor**

The initial value of **ActiveXObject.prototype.constructor** is the built-in **ActiveXObject** constructor.

1. **Properties of ActiveXObject Instances**

**ActiveXObject** has no instances. Objects created by the **ActiveXObject** constructor are hostobjects that have properties which are determined by the external application associated with the specific host object.

1. **Extensions to ECMAScript 5.1**

This section describes extensions to  [[ECMA-262/51]](http://go.microsoft.com/fwlink/?LinkId=240226) that are not available in versions prior to Windows® Internet Explorer® 10.

1. **Typed Arrays**

Typed arrays provide access to raw binary data and enables efficient byte-level programming ability to JavaScript developers. The functionality is implemented by the following three objects.

**ArrayBuffer**: The ArrayBuffer object provides the ability to create and work with an opaquebuffer of native memory.

**TypeArray**: Each of the TypeArray objects provides a view over an ArrayBuffer based on theelement Type, allowing typed access to the contents of the native buffer.

**DataView**: The DataView object provides unstructured access to the contents of an ArrayBuffer,reading and writing basic data types and fixed offsets in the buffer.

1. **ArrayBuffer Objects**

This section describes ArrayBuffer Objects.

1. **The ArrayBuffer constructor called as a function**

When ArrayBuffer is called as a function rather than as a constructor, it creates and initialises a new

ArrayBuffer object. Thus the function call ArrayBuffer(…) is equivalent to the object creation expression new ArrayBuffer (…) with the same arguments.

1. **The ArrayBuffer constructor**

When ArrayBuffer is called as part of a new expression, it is a constructor: it initialises the newly created object.



*42 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **New Array (len)**

The [[Prototype]] internal property of the newly constructed object is set to the original ArrayBuffer prototype object, the one that is the initial value of ArrayBuffer.prototype (16.1.3.1). The [[Class]] internal property of the newly constructed object is set to "ArrayBuffer". The [[Extensible]] internal property of the newly constructed object is set to true.

The length property of the newly constructed object is set to ToUInt32(len).

A fresh native buffer nativeBuffer of length bytes is allocated. The contents of this native buffer are zero initialized. If the requested number of bytes could not be allocated, a RangeError is raised. The [[NativeBuffer]] internal property of the newly constructed object is set to nativeBuffer.

1. **Properties of the ArrayBuffer constructor**

The value of the [[Prototype]] internal property of the ArrayBuffer constructor is the Function prototype object (15.3.4).

Besides the internal properties and the length property (whose value is 1), the ArrayBuffer constructor has the following properties:

1. **ArrayBuffer.Prototype**

The initial value of ArrayBuffer.prototype is the ArrayBuffer prototype object (16.1.4).

This property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.

1. **Properties of the ArrayBuffer Prototype Object**

The value of the [[Prototype]] internal property of the Array prototype object is the standard built-in Object prototype object (15.2.4). The [[Class]] internal property of the newly constructed object is set to "Object". The [[Extensible]] internal property of the newly constructed object is set to true.

1. **ArrayBuffer.prototype.constructor**

The initial value of ArrayBuffer.prototype.constructor is the standard built-in ArrayBuffer constructor.

1. **Properties of ArrayBuffer Instances**

ArrayBuffer instances inherit properties from the ArrayBuffer prototype object and their [[Class]] internal property value is "ArrayBuffer". ArrayBuffer instances also have the following properties.

1. **byteLength**

The byteLength property of this ArrayBuffer object is a data property whose value is the length of the ArrayBuffer in bytes, as fixed at construction time.

The length property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [Configurable]]: false }.



*43 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **TypeArray Objects**

For each *Type* in the following table, a separate *Type*Array constructor object, with corresponding prototype and instances as described below is available.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Array Name** | **Size** | **Description** | **Equivalent C Type** |
|  |  |  |  |  |
| Int8 | Int8Array | 1 | 8-bit 2's complement signed integer | signed char |
|  |  |  |  |  |
| Uint8 | Uint8Array | 1 | 8-bit unsigned integer | unsigned char |
|  |  |  |  |  |
| Int16 | Int16Array | 2 | 16-bit 2's complement signed integer | Short |
|  |  |  |  |  |
| Uint16 | Uint16Array | 2 | 16-bit unsigned integer | unsigned short |
|  |  |  |  |  |
| Int32 | Int32Array | 4 | 32-bit 2's complement signed integer | Int |
|  |  |  |  |  |
| Uint32 | Uint32Array | 4 | 32-bit unsigned integer | unsigned int |
|  |  |  |  |  |
| Float32 | Float32Array | 4 | 32-bit IEEE floating point | Float |
|  |  |  |  |  |
| Float64 | Float64Array | 8 | 64-bit IEEE floating point | Double |
|  |  |  |  |  |

1. **The TypeArray Constructor Called as a Function**

When *Type*Array is called as a function rather than as a constructor, it creates and initialises a new

*Type*Array object. Thus the function call *Type*Array(…) is equivalent to the object creationexpression new *Type*Array (…) with the same arguments.

1. **The TypeArray Constructor**

When *Type*Array is called as part of a new expression, it is a constructor: it initialises the newly created object.

1. **New TypeArray (arg0 [, arg1, [, arg2])**

The [[Prototype]] internal property of the newly constructed object is set to the original *Type*Array prototype object, the one that is the initial value of *Type*Array.prototype (16.2.3.1). The [[Class]] internal property of the newly constructed object is set to "*Type*Array". The [[Extensible]] internal property of the newly constructed object is set to true.

The remaining properties of the newly constructed object are set as follows:

1. If the argument arg0 is a Number:
   1. The length property of the newly constructed object is set to ToUInt32(arg0)
   2. The byteLength property of the newly constructed object is set to length multiplied by the size in bytes of *Type*.
   3. Let arrayBuffer be an object constructed as if by a call to the built-in ArrayBuffer constructor, as "new ArrayBuffer(byteLength)".
   4. The buffer property of the newly constructed object is set to arrayBuffer.
   5. The byteOffset property of the newly constructed object is set to 0.



*44 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Otherwise if the argument arg0 is an Object:
   1. Let *O* be the result of calling ToObject(arg0).
   2. Let *class* be the value of the [[Class]] internal property of *O*.
   3. If class is "ArrayBuffer":
      1. Let byteOffset be the result of calling ToUInt32 on arg1, if provided, or else 0.
      2. If byteOffset is not an integer multiple of the size in byte of Type, raise a RangeError exception.
      3. Let bufferLength be the result of calling [[Get]] on O with property name "byteLength".
      4. Let byteLength be the result of calling ToUInt32 on arg2, if provided, or else bufferLength – byteOffset.
      5. If byteOffset + byteLength is greater than bufferLength, raise a RangeError exception.
      6. Let length be the result of dividing byteLength by the size in bytes of Type.
      7. If ToUInt32(length) !== length, raise a RangeError exception.
      8. The length property of the newly constructed object is set to length.
      9. The byteLength property of the newly constructed object is set to byteLength.

10.The buffer property of the newly constructed object is set to O.

11.The byteOffset property of the newly constructed object is set to byteOffset.

1. Else:
   1. Let n to be the result of calling [[Get]] on V with property name "length".
   2. Let length be the result of calling ToUInt32(n).
   3. The length property of the newly constructed object is set to length.
   4. The byteLength property of the newly constructed object is set to length multiplied by the size in bytes of *Type*.
   5. Let arrayBuffer be an object constructed as if by a call to the built-in ArrayBuffer constructor, as "new ArrayBuffer(byteLength)".
   6. Initialize i to be 0.
   7. While i < length:
      1. Let x be the result of calling [[Get]] on arrayBuffer with property name ToString(i).
      2. Let indexDesc be a property descriptor.
      3. Set indexDesc.Writable to true.
      4. Set indexDesc.Enumerable to true.
      5. Set indexDesc.Configurable to false.



*45 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

* + 1. Set indexDesc.Value to x.
    2. Call [[DefineOwnProperty]] on the newly constructed object with arguments ToString(i), indexDesc, and false.
    3. Set i to i + 1.
  1. The buffer property of the newly constructed object is set to arrayBuffer.
  2. The byteOffset property of the newly constructed object is set to 0.

1. Otherwise:

Throw an exception

1. **Properties of the TypeArray Constructor**

The value of the [[Prototype]] internal property of the *Type*Array constructor is the Function prototype object (15.3.4).

Besides the internal properties and the length property (whose value is 3), the *Type*Array constructor has the following properties:

1. **TypeArray.prototype**

The initial value of *Type*Array.prototype is the *Type*Array prototype object (16.2.4).

This property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.

1. **typeArray.BYTES\_PER\_ELEMENT**

The initial value of *Type*Array.BYTES\_PER\_ELEMENT is the size in bytes of *Type*.

This property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.

1. **Properties of the TypeArray Prototype Object**

The value of the [[Prototype]] internal property of the TypeArray prototype object is the standard built-in Object prototype object (15.2.4). It’s [[Class]] is "*Type*Array".

1. **TypeArray.prototype.constructor**

The initial value of *Type*Array.prototype.constructor is the standard built-in *Type*Array constructor.

1. **TypeArray.prototype.set(Array [, offset] )**

Set multiple values in the TypedArray, reading from the array input., reading input values from the array. The optional offset value indicates the index in the current array where values are written. If omitted, it is assumed to be 0.

1. If this does not have class "*Type*Array", throw a TypeError.
2. Let offsetIndex be ToUInt32(offset)
3. Let *O* be the result of calling ToObject(array).



*46 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Let srcLength be the result of calling [[Get]] on O with property name "length".
2. Let targetLength be the result of calling [[Get]] on this with property name "length"
3. If srcLength + offset > targetLength, throw a RangeError.
4. Let temp be a new TypeArray created as if by a call to "new TypeArray(srcLength)"
5. Let k be 0
6. While k < srcLength
   1. Let v be the result of calling [[Get]] on src with property name toString(k)
   2. Call [[Put]] on temp with arguments ToString(k), v, and false

10.Let k be offset

11.While k < targetLength

1. Let v be the result of calling [[Get]] on temp with property name ToString(k-offset)
2. Call [[Put]] on temp with arguments ToString(k), v, and false
3. **TypeArray.prototype.subarray(begin [, end] )**

Returns a new TypedArray view of the ArrayBuffer store for this TypedArray, referencing the elements at begin, inclusive, up to end, exclusive. If either begin or end is negative, it refers to an index from the end of the array, as opposed to from the beginning.

1. If this does not have class "*Type*Array", throw a TypeError.
2. Let srcLength be the result of calling [[Get]] on this with property name "length"
3. Let beginInt be ToInt32(begin)
4. If beginInt < 0, let beginInt be srcLength + beginInt
5. Let beginIndex be min(srcLength, max(0, beginInt))
6. Let endInt be ToInt32(end) if end was provided, else srcLength.
7. If endInt <0,let endInt be srcLength + endInt
8. Let endIndex be max(0,min(srcLength, endInt))
9. If endIndex < beginIndex, let endIndex be beginIndex

10.Return a new TypeArray with the following values for it’s proeprties:

1. The length property of the newly constructed object is set to endIndex - beginIndex
2. The byteLength property of the newly constructed object is set to length multiplied by the size in bytes of *Type*.
3. The buffer property of the newly constructed object is set to this.buffer.
4. The byteOffset property of the newly constructed object is set to this.offset + beginIndex.



*47 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **Properties of TypeArray Instances**

*Type*Array instances inherit properties from the *Type*Array prototype object and their [[Class]]internal property value is "*Type*Array". *Type*Array instances also have the following properties.

1. **[[DefineOwnProperty]] (P, Desc, Throw )**

*Type*Array objects use a variation of the [[DefineOwnProperty]] internal method used for othernative ECMAScript objects (8.12.9).

When the [[DefineOwnProperty]] internal method of A is called with property P, Property Descriptor Desc and Boolean flag Throw, the following steps are taken:

1. Let succeeded be the result of calling the default [[DefineOwnProperty]] internal method (8.12.9) on A passing P, Desc, and Throw as arguments.
2. If succeeded is false, return false.
3. If Desc contains a Value field, let newValue be Desc.Value
4. Let convertedValue to To*Type*(newValue)
5. Let index be ToUInt32(P)
6. Call the SetValueInBuffer internal operation with arguments A.buffer.[[NativeBuffer]], A.byteOffset, index, convertedValue, and *Type*.
7. Return true.

The internal operation SetValueInBuffer takes five parameters, a native buffer nativeBuffer, an integer byteOffset, an integer index, a value of type *Type* newValue, and a Type valueType. It operates as follows:

1. Let size be the size in bytes of the type valueType.
2. Let bytes be the array of bytes from nativeBuffer between offset byteOffset+(index\*size) and offset byteOffset+((index+1)\*size)-1 inclusive.
3. Let newValueBytes be the result of converting newValue to an array of bytes, using the platform endianness.
4. Set each byte of bytes from the corresponding byte of newValueBytes.
5. **[[GetOwnProperty]] ( P)**

*Type*Array objects use a variation of the [[GetOwnProperty]] internal method used for other nativeECMAScript objects (8.12.1). This special internal method provides access to named properties corresponding to the individual index values of the *Type*Array objects.

When the [[GetOwnProperty]] internal method of A is called with property name P, the following steps are taken:

1. Let desc be the result of calling the default [[GetOwnProperty]] internal method (8.12.1) on A with argument P.
2. If desc is not undefined return desc.
3. If ToString(abs(ToInteger(P))) is not the same value as P, return undefined.



*48 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. Let length be the result of a calling [[Get]] on A with parameter "length"
2. Let index be ToInteger(P).
3. If length ≤ index, return undefined.
4. Let isLittleEndian be true if the platform endianness is little endian, else false.
5. Let value be the result of calling the GetValueFromBuffer internal operation with arguments A.buffer.[[NativeBuffer]], A.byteOffset, index, *Type*, and littleEndian.
6. Return a Property Descriptor { [[Value]]: value, [[Enumerable]]: true, [[Writable]]: true, [[Configurable]]: false }

The internal operation GetValueFromBuffer takes three parameters, a native buffer nativeBuffer, an integer byteOffset, an integer index, a Type valueType, and a boolean isLittleEndian. It operates as follows:

1. Let size be the size in bytes of the type valueType.
2. Let bytes be the array of bytes from nativeBuffer between offset byteOffset+(index\*size) and offset byteOffset+((index+1)\*size)-1 inclusive.
3. Let rawValue be the result of convert the array bytes to a value of type valueType, using little endian if isLittleEndian is true, otherwise big endian.
4. If valueType is Float32 and rawValue is a Float32 representation of IEEE754 NaN, return the NaN Number value.
5. Else, if valueType is Float64 and rawValue is a Float64 representation of IEEE754 NaN, return the NaN Number value.
6. Else, return the Number value that that represents the same numeric value as rawValue
7. **length**

The value of the length property is the length of the TypeArray object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **byteLength**

The value of the byteLength property is the length of the TypeArray object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **buffer**

The value of the buffer property is the length of the TypeArray object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **byteOffset**

The value of the byteOffset property is the length of the TypeArray object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.



*49 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **DataView Objects**

This section describes DataView Objects.

1. **The DataView Constructor called as a function**

When DataView is called as a function rather than as a constructor, it creates and initialises a new

DataView object. Thus the function call DataView(…) is equivalent to the object creation expression new DataView(…) with the same arguments.

1. **The DataView Constructor**

When DataView is called as part of a new expression, it is a constructor: it initialises the newly created object.

1. **New DataView (buffer [, byteOffset [, byteLength]])**

The [[Prototype]] internal property of the newly constructed object is set to the original DataView prototype object, the one that is the initial value of DataView.prototype (16.1.3.1). The [[Class]] internal property of the newly constructed object is set to "DataView". The [[Extensible]] internal property of the newly constructed object is set to true.

The remaining proeprties are set as follows:

1. Let O be ToObject(buffer)
2. If the [[Class]] internal property of O is not "ArrayBuffer", raise a TypeError.
3. Let byteOffset be the result of calling ToUInt32 on byteOffset, if provided, or else 0.
4. Let bufferLength be the result of calling [[Get]] on O with property name "byteLength".
5. Let byteLength be the result of calling ToUInt32 on byteLength, if provided, or else bufferLength

– byteOffset.

1. If byteOffset + byteLength is greater than bufferLength, raise a RangeError exception.
2. The byteLength property of the newly constructed object is set to byteLength.
3. The buffer property of the newly constructed object is set to O.
4. The byteOffset property of the newly constructed object is set to byteOffset.
5. **Properties of the DataView Constructor**

The value of the [[Prototype]] internal property of the DataView constructor is the Function prototype object (15.3.4).

Besides the internal properties and the length property (whose value is 3), the DataView constructor has the following properties:

1. **DataView.prototype**

The initial value of DataView.prototype is the DataView prototype object (16.1.4).

This property has the attributes { [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.



*50 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **Properties of the DataView Prototype Object**

The value of the [[Prototype]] internal property of the DataView prototype object is the standard built-in Object

prototype object (15.2.4). The [[Class]] internal property of the newly constructed object is set to **"Object"**. The [[Extensible]] internal property of the newly constructed object is set to **true**.

The internal operation GetValue(byteOffset, isLittleEndian, type) used by functions on DataView instances is defined as follows:

1. Let byteOffsetInt be ToUInt32(byteOffset)
2. Let totalOffset be byteOffsetInt plus the result of calling [[Get]] on this with parameter "byteOffset"
3. Let byteLength be the result of calling [[Get]] on this with parameter "byteLength"
4. If totalOffset >= byteLength, raise a RangeError
5. Let value be the result of calling the GetValueFromBuffer internal operation (9.2.5.2) with arguments this.buffer.[[NativeBuffer]], totalOffset, 0 and type.
6. Return value

The internal operation SetValue(byteOffset, isLittleEndian, type, value) used by functions on DataView instances is defined as follows:

1. Let byteOffsetInt be ToUInt32(byteOffset)
2. Let totalOffset be byteOffsetInt plus the result of calling [[Get]] on this with parameter "byteOffset"
3. Let byteLength be the result of calling [[Get]] on this with parameter "byteLength"
4. If totalOffset >= byteLength, raise a RangeError
5. Let value be the result of calling the SetValueInBuffer internal operation (9.2.5.2) with arguments this.buffer.[[NativeBuffer]], totalOffset, 0, value and type.
6. Return value
7. **DataView.prototype.constructor**

The initial value of DataView.prototype.constructor is the standard built-in DataView constructor.

1. **DataView.prototype.GetInt8(byteOffset)**

Gets the Int8 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
3. Return GetValue(byteOffset, true, Int8)



*51 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **DataView.prototype.GetUInt8(byteOffset)**

Gets the UInt8 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
3. Return GetValue(byteOffset, true, UInt8)
4. **DataView.prototype.GetInt16(byteOffset, littleEndian)**

Gets the Int16 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Int16)
5. **DataView.prototype.GetUInt16(byteOffset, littleEndian)**

Gets the Uint16 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Uint16)
5. **DataView.prototype.GetInt32(byteOffset, littleEndian)**

Gets the Int32 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Int32)
5. **DataView.prototype.GetUInt32(byteOffset, littleEndian)**

Gets the Uint32 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Uint32)



*52 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **DataView.prototype.GetFloat32(byteOffset, littleEndian)**

Gets the Float32 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Float32)
5. **DataView.prototype.GetFloat64(byteOffset, littleEndian)**

Gets the Float64 value at offset byteOffset in the DataView, using the provided endianness.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Float64)
5. **DataView.prototype.SetInt8(byteOffset, value)**

Sets the Int8 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
3. Return GetValue(byteOffset, true, Int8, ToInt8(value))
4. **DataView.prototype.SetUInt8(byteOffset, value)**

Sets the Uint8 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
3. Return GetValue(byteOffset, true, Uint8, ToUint8(value))
4. **DataView.prototype.SetInt16(byteOffset, value, littleEndian)**

Sets the Int16 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Int16, ToInt16(value))



*53 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

1. **DataView.prototype.SetUInt16(byteOffset, value, littleEndian)**

Sets the Uint16 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Uint16, ToUint16(value))
5. **DataView.prototype.SetInt32(byteOffset, value, littleEndian)**

Sets the Int32 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Int32, ToInt32(value))
5. **DataView.prototype.SetUInt32(byteOffset, value, littleEndian)**

Sets the Uint32 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Uint32, ToUint32(value))
5. **DataView.prototype.SetFloat32(byteOffset, value, littleEndian)**

Sets the Float32 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.
4. Return GetValue(byteOffset, isLittleEndian, Float32, ToFloat32(value))
5. **DataView.prototype.SetFloat64(byteOffset, value, littleEndian)**

Sets the Float64 value at offset byteOffset in the DataView.

1. Let O be ToObject(this)
2. Let isLittleEndian be ToBoolean(littleEndian) if provided, else false
3. If the [[Class]] internal property of O is not "DataView", raise a TypeError.



*54 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

4. Return GetValue(byteOffset, isLittleEndian, Float64, ToFloat64(value))

1. **Properties of DataView Instances**

DataView instances inherit properties from the **DataView** prototype object and their [[Class]] internal property value is "DataView". DataView instances also have the following properties.

1. **byteLength**

The value of the **byteLength** property is the length of the **DataView** object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **buffer**

The value of the **buffer** property is the length of the **DataView** object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **byteOffset**

The value of the **byteOffset** property is the length of the **DataView** object, which was fixed at creation. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

1. **Properties of Error Constructor**

Internet Explorer 10 ECMAScript defines an additional property on Error constructor of [ECMA-262/5]. The additional property is described in the following section.

1. **stackTraceLimit**

The initial value of **stackTraceLimit** is the numeric value 10. This property has the attributes { [[Enumerable]]:true, [[Configurable]]:true, [[Writable]]:true }.

1. **Properties of Error Instances**

Internet Explorer ECMAScript defines additional error instances inherited from the [[Prototype]] object of  [[ECMA-262/5].](http://go.microsoft.com/fwlink/?LinkId=185963) This error instance is described in the following section.

1. **stack**

The initial value of **stack** is undefined. This property has the attributes { [[Enumerable]]:true, [[Configurable]]:true, [[Writable]]:true }. When an error is thrown the stack property is set to contain a string value which describes the stack frames formatted as described below.

"<Error Type>: <Error Description>

at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>) at FunctionName (<Fully qualified file/URL>:<line#>:<col#>)



*55 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

at FunctionName (<Fully qualified file/URL>:<line#>:<col#>)"

The number of stack frames shown is controlled by the **stackTraceLimit** property defined on the Error constructor.



*56 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**3 Security Considerations**

There are no additional security considerations.



*57 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**4 Appendix A: Product Behavior**

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

Windows® Internet Explorer® 9

Windows® Internet Explorer® 10

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.



*58 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**5 Change Tracking**

This section identifies changes that were made to the [MS-ES5EX] protocol document between the February 2012 and July 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

A document revision that incorporates changes to interoperability requirements or functionality.

An extensive rewrite, addition, or deletion of major portions of content.

The removal of a document from the documentation set.

Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

New content added.

Content updated.

Content removed.

New product behavior note added.

Product behavior note updated.

Product behavior note removed.

New protocol syntax added.

Protocol syntax updated.

Protocol syntax removed.

New content added due to protocol revision.

Content updated due to protocol revision.

Content removed due to protocol revision.

New protocol syntax added due to protocol revision.



*59 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

Protocol syntax updated due to protocol revision.

Protocol syntax removed due to protocol revision.

New content added for template compliance.

Content updated for template compliance.

Content removed for template compliance.

Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

**Protocol syntax** refers to data elements (such as packets, structures, enumerations, andmethods) as well as interfaces.

**Protocol revision** refers to changes made to a protocol that affect the bits that are sent overthe wire.

The changes made to this document are listed in the following table. For more information, please contact  [protocol@microsoft.com](mailto:protocol@microsoft.com).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Major** |  |  |  |
|  | **Tracking number (if applicable)** |  | **change** |  |  |  |
|  |  |  |
| **Section** | **and description** |  | **(Y or N)** |  | **Change type** |  |
|  |  |  |  |  |  |  |
| 1 | Updated document to remove beta tagging. |  | N | | Content updated. |  |
| Introduction |  |  |  |  |  |  |
|  |  |  |  |  |  |  |



*60 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*

**6 Index**

**A**

[Applicability](#page9) 9

**C**

[Change tracking](#page59) 59

**G**

[Glossary](#page7) 7

**I**

[Implementer - security considerations](#page57) 57

[Informative references](#page7) 7

[Introduction](#page7) 7

**N**

[Normative references](#page7) 7

**O**

[Overview (synopsis)](#page8) 8

**P**

[Product behavior](#page58) 58

**R**

[References](#page7) 7  [informative](#page7) 7  [normative](#page7) 7

**S**

[Security - implementer considerations](#page57) 57

**T**

[Tracking changes](#page59) 59



*61 / 61*

*[MS-ES5EX] — v20120725*

*Internet Explorer Extensions to the ECMA-262 ECMAScript Language Specification (Fifth Edition)*

*Copyright © 2012 Microsoft Corporation.*

*Release: July 25, 2012*