17. Return *numValue*.

15.13.6.3.16 *TypedArray*.prototype [@@toStringTag]

The initial value of the @@toStringTag property is the string value of the name *TypedArray*.

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

15.13.6.4 Properties of *TypedArray* Instances

TypedArray instances are exotic Index Delegation objects. Each *TypedArray* instances inherits properties from the corresponding *TypedArray* prototype object. Each *TypedArray* instances have the following internal data properties: [[ViewedArrayBuffer]], [[TypedArrayElementKind]], [[ByteLength]], [[ByteOffset]], and [[ArrayLength]].

15.13.7 DataView Objects

15.13.7.1 Abstract Operations For DataView Objects

The abstract operation GetViewValue(view, requestIndex, isLittleEndian, type) used by functions on DataView instances is defined as follows:

1. Let *v* be ToObject(*view*)
2. ReturnIfAbrupt(*v*).
3. If *v* does not have [[ViewedArrayBuffer]] internal data property, then throw a **TypeError** exception.
4. Let *buffer* be the value of *v's* [[ViewedArrayBuffer]] internal data property.
5. If *buffer* is **undefined**, then throw a **TypeError** exception.
6. Let *numberIndex* be ToNumber(*requestIndex*)
7. Let *getIndex* be ToInteger(*numberIndex*).
8. ReturnIfAbrupt(*getIndex*).
9. If *numberIndex* ≠ *getIndex* or *getIndex* < 0, then throw a **RangeError** exception.
10. If *isLittleEndean* is not **undefined**, then let *isLittleEndian* be ToBoolean(*isLittleEndian*).
11. ReturnIfAbrupt(*isLittleEndean*).
12. Let *viewOffset* be the value of *v's* [[ByteOffset]] internal data property.
13. Let *viewSize* be the value of *v's* [[ByteLength]] internal data property.
14. Let *elementSize* be the Number value of the Element Size value specified in Table 36 for *type*.
15. If *getIndex* +*elementSize* > *viewSize*, then throw a **RangeError** exception.
16. Let *bufferIndex* be *getOffset*+*viewOffset*.
17. Return the result of the GetValueFromBuffer(*buffer*, *bufferIndex*, *type*, *isLittleEndean*).

The abstract operation SetViewValue(view, requestIndex, isLittleEndian, type, value) used by functions on DataView instances is defined as follows:

1. Let *v* be ToObject(*view*)
2. ReturnIfAbrupt(*v*).
3. If *v* does not have [[ViewedArrayBuffer]] internal data property, then throw a **TypeError** exception.
4. Let *buffer* be the value of *v's* [[ViewedArrayBuffer]] internal data property.
5. If *buffer* is **undefined**, then throw a **TypeError** exception.
6. Let *numberIndex* be ToNumber(*requestIndex*)
7. Let *getIndex* be ToInteger(*numberIndex*).
8. ReturnIfAbrupt(*getIndex*).
9. If *numberIndex* ≠ *getIndex* or *getIndex* < 0, then throw a **RangeError** exception.
10. If *isLittleEndean* is not **undefined**, then let *isLittleEndian* be ToBoolean(*isLittleEndian*).
11. ReturnIfAbrupt(*isLittleEndean*).
12. Let *viewOffset* be the value of *v's* [[ByteOffset]] internal data property.
13. Let *viewSize* be the value of *v's* [[ByteLength]] internal data property.
14. Let *elementSize* be the Number value of the Element Size value specified in Table 36 for *type*.
15. If *getIndex* +*elementSize* > *viewSize*, then throw a **RangeError** exception.
16. Let *bufferIndex* be *getOffset*+*viewOffset*.

Deleted: .

Deleted: After initialisation by a *TypedArray* constructor,

Deleted: [[TypedArrayData]]

Formatted: Heading 3

Deleted: *TypeArray* instances inherit properties from the *TypeArray* prototype object and their [[Class]] internal data property value is "TypeArray". *TypeArray* instances also have the following properties.¶
<#>15.13.6.5.1 [[DefineOwnProperty]] (p, desc, throw)¶
<#>TypeArray objects use a variation of the [[DefineOwnProperty]] internal method used for other native 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:¶
<#> Let succeeded be the result of calling the default [[DefineOwnProperty]] internal method (8.12.9) on A passing P, Desc, and Throw as arguments.¶
<#> If succeeded is false, return false.¶
<#> If Desc contains a Value field, let newValue be Desc.Value¶
<#> Let convertedValue to ToType(newValue)¶
<#> Let index be ToUint32(P)¶
<#> Call the SetValueInBuffer internal operation with arguments A.buffer.[[NativeBuffer]], A.byteOffset, index, convertedValue, and Type.¶
<#> 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:¶
<#> Let size be the size in bytes of the type valueType.¶
<#> Let bytes be the array of bytes from nativeBuffer between offset byteOffset+(index*size) and offset byteOffset+((index+1)* × size)-1 inclusive.¶
<#> Let newValueBytes be the result of converting newValue to an array of bytes, using the platform endianness.¶
<#> Set each byte of bytes from the corresponding byte of newValueBytes.¶
<#>15.13.6.5.2 [[GetOwnProperty]] (P)¶
<#>TypeArray objects use a variation of the [[GetOwnProperty]] internal method used for other native ECMAScript objects (8.12.1). This special internal method provides access to named properties corresponding to the individual index values of the TypeArray objects.¶ ... [32]

Deleted: <#>The following sections define DavteaViews is derived from the Kronos specification. This material is a very early draft based upon the strawman at http://wiki.ecmascript.org/doku.php?id=strawman:typed_arrays . This material still needs significant work to fully integrate it into the ES6 spec. and also to integrate typed arrays with ES6 bina ... [33]

Comment [AWB16227]: Issue: the Khronos spec. says that undefined means false.

Comment [AWB16228]: Issue: the Khronos spec. says that undefined means false.

DRAFT

© Ecma International 2012

17. Return the result of the SetValueInBuffer(*buffer*, *bufferIndex*, *type*, *value isLittleEndean*).

NOTE The algorithms for GetViewValue and SetViewValue are identical except for their final steps.

15.13.7.2 The DataView Constructor

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. However, if the **this** value value passed in the call is an Object with an [[ViewedArrayBuffer]] internal data property whose value is **undefined**, it initializes the **this** value using the argument values. This permits `DataView` to be used both as factory method and to perform constructor instance initialization.

The `DataView` constructor is designed to be subclassable. It may be used as the value of an `extends` clause of a class declaration. Subclass constructors that intended to inherit the specified `ArrayBuffer` behaviour must include a `super` call to the `DataView` constructor to initialise subclass instances.

15.13.7.2.1 DataView(buffer, byteOffset=0, byteLength=undefined)

`DataView` called with arguments *buffer*, *byteOffset*, and *length* performs the following steps:

1. Let *O* be the **this** value.
2. If Type(*O*) is not Object or if *O* does not have an [[ViewedArrayBuffer]] internal data property or if the value of *O's* [[ViewedArrayBuffer]] internal data property is not **undefined**, then
    a. Let *F* be this function object.
    b. Let *argumentsList* be the *argumentsList* argument of the [[Call]] internal method that invoked *F*.
    c. Return the result of calling OrdinaryConstruct (*F*, *argumentsList*).
3. If Type(*buffer*) is not Object, then throw a **TypeError** exception.
4. If *buffer* does not have a [[ArrayBufferData]] internal data property, then throw a **TypeError** exception.
5. Let *numberOffset* be ToNumber(*byteOffset*).
6. Let *offset* be ToInteger(*numberOffset*).
7. ReturnIfAbrupt(*offset*).
8. If *numberOffset* ≠ *offset* or *offset* < 0, then throw a **RangeError** exception.
9. Let *bufferByteLength* be the value of *buffer's* [[ArrayBufferByteLength]] internal data property.
10. If *offset* ≥ *bufferByteLength*, then throw a **RangeError** exception.
11. If *byteLength* is **undefined**, then
    a. Let *viewByteLength* be *bufferByteLength* – *offset*.
12. Else,
    a. Let *numberLength* be ToNumber(*byteLength*).
    b. Let *viewLength* be ToInteger (*numberLength*).
    c. ReturnIfAbrupt(*viewLength*).
    d. If *numberLength* ≠ *viewLength* or *viewLength* < 0, then throw a **RangeError** exception.
    e. Let *viewByteLength* be *viewLength*.
    f. If *offset*+*viewByteLength* > *bufferByteLength*, then throw a **RangeError** exception.
13. If the value of *O's* [[ViewedArrayBuffer]] internal data property is not **undefined**, then throw a **TypeError** exception.
14. Set *O's* [[ViewedArrayBuffer]] to *buffer*.
15. Set *O*'s [[ByteLength]] internal data property to *newByteLength*.
16. Set *O*'s [[ByteOffset]] internal data property to *offset*.
17. Return *O*.

15.13.7.2.2 new DataView(... argumentsList)

`DataView` called as part of a new expression it performs the following steps:

4. Let *F* be the function object on which the `new` operator was applied.
5. Let *argumentsList* be the *argumentsList* argument of the [[Construct]] internal method that was invoked by the `new` operator.
6. Return the result of OrdinaryConstruct(*F*, *argumentsList*).

Deleted: <#>15.13.7.2 The DataView Constructor¶
When DataView is called as part of a new expression, it is a constructor: it initialises the newly created object.¶
<#>15.13.7.2.1 new DataView(buffer [, byteOffset [, byteLength]])¶
The [[Prototype]] internal data property of the newly constructed object is set to the original DataView prototype object, the one that is the initial value of DataView.prototype (15.13.3.3.1). The [[Class]] internal data property of the newly constructed object is set to "DataView". The [[Extensible]] internal data property of the newly constructed object is set to **true**.¶
The remaining proeprtiesproperties are set as follows:¶
<#> Let O be ToObject(buffer)¶
<#> If the [[Class]] internal data property of O is not "ArrayBuffer", raisethrow a **TypeError** exception.¶
<#> Let byteOffset be the result of calling ToUInt32 on byteOffset, if provided, or else 0.¶
<#> Let bufferLength be the result of calling [[Get]] on(*O* with property name, `"byteLength"`).¶
<#> Let byteLength be the result of calling ToUInt32 on byteLength, if provided, or else bufferLength – byteOffset.¶
<#> If byteOffset + byteLength is greater than bufferLength, throw raise a **RangeError** exception.¶
<#> The byteLength property of the newly constructed object is set to byteLength.¶
<#> The buffer property of the newly constructed object is set to O.¶
<#> The byteOffset property of the newly constructed object is set to byteOffset.¶

DRAFT

If `DataView` is implemented as an ordinary function object, its [[Construct]] internal method will perform the above steps.

15.13.7.3 Properties of the DataView Constructor

The value of the [[Prototype]] internal data property of the `DataView` constructor is the `Function` prototype object (15.3.3).

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

15.13.7.3.1 DataView.prototype

The initial value of `DataView.prototype` is the DataView prototype object (15.13.7.4).

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

15.13.7.3.2 DataView [@@create] ( )

The @@create method of a DataView function object *F* performs the following steps:

1. Let *F* be the **this** value.
2. Let *obj* be the result of calling OrdinaryCreateFromConstructor(*F*, `"%DataViewPrototype%"`, ([[ViewedArrayBuffer]] , [[ByteLength]], [[ByteOffset]])).
3. Return *obj*.

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

15.13.7.4 Properties of the DataView Prototype Object

The value of the [[Prototype]] internal data property of the DataView prototype object is the standard built-in Object prototype object (15.2.4). The DataView prototype object is an ordinary object. It does not have a [[ViewedArrayBuffer]], [[ByteLength]], or [[ByteOffset]] internal data property.

15.13.7.4.1 DataView.prototype.constructor

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

15.13.7.4.2 DataView.prototype.getInt8(byteOffset)

When the `getInt8` method is called with argument *byteOffset* the following steps are taken:

1. Let *v* be the this value.
2. Return the result of GetViewValue(*v*, *byteOffset*, **undefined**, `"Int8"`).

15.13.7.4.3 DataView.prototype.getUint8(byteOffset)

When the `getUint8` method is called with argument *byteOffset* the following steps are taken:

1. Let *v* be the **this** value.
2. Return the result of GetViewValue(*v*, *byteOffset*, **undefined**, `"Uint8"`).

15.13.7.4.4 DataView.prototype.getInt16(byteOffset, littleEndian=undefined)

When the `getInt16` method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.

Formatted: Font: (Default) Courier New, Bold, Complex Script Font: Courier New, Bold

Deleted: 15.13.3.4

Deleted: The [[Class]] internal data property of the newly constructed object is set to "Object". The [[Extensible]] internal data property of the newly constructed object is set to **true**.

Deleted: The internalabstract operation GetValue(byteOffset, isLittleEndian, type) used by functions on DataView instances is defined as follows:¶
<#> Let byteOffsetInt be ToUInt32(byteOffset)¶
<#> Let totalOffset be byteOffsetInt plus the result of calling [[Get]] on(this with parameter, `""byteOffset""`).¶
<#> Let byteLength be the result of calling [[Get]] on(this with parameter, `""byteLength""`).¶
<#> If totalOffset ≥>= byteLength, throw raise a **RangeError** exception.¶
<#> Let value be the result of calling the GetValueFromBuffer internalabstract operation (2.5.2) with arguments this.buffer.[[NativeBuffer]], totalOffset, 0 and type.¶
<#> Return value¶
The internalabstract operation SetValue(byteOffset, isLittleEndian, type, value) used by functions on DataView instances is defined as follows:¶
<#> Let byteOffsetInt be ToUInt32(byteOffset)¶
<#> Let totalOffset be byteOffsetInt plus the result of calling [[Get(this, `"byteOffset"`).Get]] on this with parameter "byteOffset"¶
<#> Let byteLength be the result of Get(this, `"byteLength"`).calling [[Get]] on this with parameter "byteLength"¶
<#> If totalOffset ≥>= byteLength, raisethrow a **RangeError** exception.¶
<#> Let value be the result of calling the SetValueInBuffer internalabstract operation (2.5.2) with arguments this.buffer.[[NativeBuffer]], totalOffset, 0, value and type.¶
<#> Return value¶
15.13.7.4.1

Formatted: Font: (Default) Courier New, Bold, Complex Script Font: Courier New, Bold

Deleted: Gets the Int8 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶

Deleted:

Deleted: true

Deleted: Gets the UInt8 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GetValue(byteOffset, **true**, UInt8)¶
15.13.7.4.4

DRAFT

© Ecma International 2012

2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Int16"**).

15.13.7.4.5 DataView.prototype.getUint16(byteOffset, littleEndian)

When the **getUint16** method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Uint16"**).

15.13.7.4.6 DataView.prototype.getInt32(byteOffset, littleEndian)

When the **getInt32** method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Int32"**).

15.13.7.4.7 DataView.prototype.getUint32(byteOffset, littleEndian)

When the **getUint32** method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Uint32"**).

15.13.7.4.8 DataView.prototype.getFloat32(byteOffset, littleEndian)

When the **getFloat32** method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Float32"**).

15.13.7.4.9 DataView.prototype.getFloat64(byteOffset, littleEndian)

When the **getFloat64** method is called with argument *byteOffset* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of GetViewValue(*v*, *byteOffset*, *littleEndean*, **"Float64"**).

15.13.7.4.10 DataView.prototype.setInt8(byteOffset, value)

When the **setInt8** method is called with arguments *byteOffset* and *value* the following steps are taken:

1. Let *v* be the this value.
2. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, **"Int8"**, *value*).

15.13.7.4.11 DataView.prototype.setUint8(byteOffset, value)

When the **setUint8** method is called with arguments *byteOffset* and *value* the following steps are taken:

Deleted: Gets the Int16 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GetValue(byteOffset, isLittleEndian, Int16)¶
15.13.7.4.5

Deleted: Gets the Uint16 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶ ... [34]

Deleted: Gets the Int32 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶ ... [35]

Deleted: Gets the Uint32 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶ ... [36]

Deleted: Gets the Float32 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶ ... [37]

Deleted: Gets the Float64 value at offset byteOffset in the DataView, using the provided endianness.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶ ... [38]

Deleted: Sets the Int8 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, **true**, Int8, ToInt8(value))¶
15.13.7.4.11

DRAFT

© Ecma International 2012

1. Let *v* be the **this** value.
2. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Uint8"`, *value*).

15.13.7.4.12 DataView.prototype.setInt16(byteOffset, value, littleEndian)

When the `setInt16` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Int16"`, *value*).

15.13.7.4.13 DataView.prototype.setUint16(byteOffset, value, littleEndian)

When the `setUint16` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Uint16"`, *value*).

15.13.7.4.14 DataView.prototype.setInt32(byteOffset, value, littleEndian)

When the `setInt32` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Int32"`, *value*).

15.13.7.4.15 DataView.prototype.setUint32(byteOffset, value, littleEndian)

When the `setUint32` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Uint32"`, *value*).

15.13.7.4.16 DataView.prototype.setFloat32(byteOffset, value, littleEndian)

When the `setFloat32` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Float32"`, *value*).

15.13.7.4.17 DataView.prototype.setFloat64(byteOffset, value, littleEndian)

When the `setFloat64` method is called with arguments *byteOffset* and *value* and optional argument *littleEndean* the following steps are taken:

1. Let *v* be the **this** value.
2. If *littleEndean* is not preset, then let *littleEndian* be **undefined**.
3. Return the result of SetViewValue(*v*, *byteOffset*, **undefined**, `"Float64"`, *value*).

Deleted: Sets the Uint8 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, **true**, Uint8, ToUint8(value))¶
15.13.7.4.12

Deleted: Sets the Int16 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, isLittleEndian, Int16, ToInt16(value))¶
15.13.7.4.13

Deleted: Sets the Uint16 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, isLittleEndian, Uint16, ToUint16(value))¶
15.13.7.4.14

Deleted: Sets the Int32 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, isLittleEndian, Int32, ToInt32(value))¶
15.13.7.4.15

Deleted: Sets the Uint32 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GetValue(byteOffset, isLittleEndian, Uint32, ToUint32(value))¶
15.13.7.4.16

Deleted: Sets the Float32 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, isLittleEndian, Float32, ToFloat32(value))¶
15.13.7.4.17

Deleted: Uint16

DRAFT

© Ecma International 2012

15.13.7.4.18 DataView.prototype[@@toStringTag]

The initial value of the @@toStringTag property is the string value **"DataView"**.

15.13.7.5 Properties of DataView Instances

DataView instances are ordinary objects that inherit properties from the DataView prototype object. Map instances each have a [[ViewedArrayBuffer]], [[ByteLength]], and [[ByteOffset]] internal data properties. DataView instances also have the following properties.

15.13.7.5.1 byteLength

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

15.13.7.5.2 buffer

The value of the buffer property is the ArrayBuffer accessed by the DataView object. This property has attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]:**false** }.

15.13.7.5.3 byteOffset

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

15.14 Map Objects

Map objects are collections of key/value pairs where both the keys and values may be arbitrary ECMAScript language values. A distinct key value may only occur in one key/value pair within the Map's collection. Distinct key values as discriminated using the a comparision algorithm that is selected when the Map is created.

A Map object can iterate its elements in insertion order. Map object must be implemented using either hash tables or other mechanisms that, on average, provide access times that are sublinear on the number of elements in the collection. The data structures used in this Map objects specification is only intended to describe the required observable semantics of Map objects. It is not intended to be a viable implementation model.

15.14.1 The Map Constructor Called as a Function

When **`Map`** is called as a function rather than as a constructor, it initialises its **this** value with the internal state necessary to support the **`Map.prototype`** internal methods. The **`Map`** constructor is designed to be subclassable. It may be used as the value in an **`extends`** clause of a class definition. Subclass constructors that intend to inherit the specified **`Map`** behaviour must include a **`super`** call to **`Map`**.

15.14.1.1 Map (iterable = undefined , comparator = undefined)

When the **`Map`** function is called with optional arguments *iterable* and *comparator* the following steps are taken:

1. Let *map* be the **this** value.
2. If Type(*map*) is not Object then, throw a **TypeError** exception.
3. If *map* does not have a [[MapData]] internal data property, then throw a **TypeError** exception.
4. If *map's* [[MapData]] internal data property is not **undefined**, then throw a **TypeError** exception.
5. If *iterable* is not present, let *iterable* be **undefined**.
6. If *iterable* is either **undefined** or **null**, then let *itr* be **undefined**.
7. Else,
   a. Let *hasValues* be the result of HasProperty(*iterable*, **"entries"**).
   b. ReturnIfAbrupt(*hasValues*).
   c. If *hasValues* is **true**, then

Deleted: Sets the Float64 value at offset byteOffset in the DataView.¶
<#> Let O be ToObject(this)¶
<#> Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**¶
<#> If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.¶
<#> Return GSetValue(byteOffset, isLittleEndian, Float64, ToFloat64(value))¶
15.13.7.5

Deleted: and their [[Class]] internal data property value is "DataView".

Deleted: 15.13.7.5.1

Deleted: ,

Deleted: which was fixed at creation.

Deleted: 15.13.7.5.2

Deleted: length

Deleted: of

Deleted: ,

Deleted: which was fixed at creation.

Deleted: 15.13.7.5.3

Deleted: ,

Deleted: which was fixed at creation.

Deleted: 15.14

Deleted: e

Deleted: also

Deleted: <#>15.14.1 Abstract Operations For Map Objects¶
<#>15.14.1.1 MapInitializsation¶
The abstract operation MapInitializsation with arguments *object* and *iterable* is used to initialize an object as a map. It performs the following steps:¶
<#>If Type(*obj*) is not Object, throw a **TypeError** exception.¶
<#>If *obj* alreadydoes not have has a [[MapData]] internal data property, throw a **TypeError** exception.¶
<#>If the result of calling the [[GetIsExtensible]] internal propertymethod of *obj* is **false**, throw a **TypeError** exception.¶
<#>If *iterable* is not **undefined**, then¶
<#>Let *iterable* be ToObject(*iterable*). ... [39]

Deleted: 15.14.21 .

Deleted: ize

Deleted: This premitspermits super invocation of the **`Map`** constructor by **`Map`** subclasses.

Deleted: 15.14.21.1 .

Deleted: [ ]

Deleted: *m*

Deleted: <#>If *m* is **undefined** or the intrinsic %MapPrototype%¶ ... [40]

Deleted: <#>Let *map* be the result of ToObject(*m*).¶ ... [41]

Deleted: n

DRAFT

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

15.13.6.5.1 [[DefineOwnProperty]] (p, desc, throw)

TypeArray objects use a variation of the [[DefineOwnProperty]] internal method used for other native 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:

Let succeeded be the result of calling the default [[DefineOwnProperty]] internal method (8.12.9) on A passing P, Desc, and Throw as arguments.

If succeeded is false, return false.

If Desc contains a Value field, let newValue be Desc.Value

Let convertedValue to ToType(newValue)

Let index be ToUInt32(P)

Call the SetValueInBuffer internal operation with arguments A.buffer.[[NativeBuffer]], A.byteOffset, index, convertedValue, and Type.

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:

Let size be the size in bytes of the type valueType.

Let bytes be the array of bytes from nativeBuffer between offset byteOffset+(index*size) and offset byteOffset+((index+1)* × size)-1 inclusive.

Let newValueBytes be the result of converting newValue to an array of bytes, using the platform endianness.

Set each byte of bytes from the corresponding byte of newValueBytes.

15.13.6.5.2 [[GetOwnProperty]] (P)

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

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

Let desc be the result of calling the default [[GetOwnProperty]] internal method (8.12.1) on A with argument P.

If desc is not undefined return desc.

If ToString(abs(ToInteger(P))) is not the same value as P, return undefined.

Let length be the result of a Get(A, `"length"`).calling [[Get]] on A with parameter "length"

Let index be ToInteger(P).

If length ≤ index, return undefined.

Let isLittleEndian be true if the platform endianness is little endian, else false.

Let value be the result of calling the GetValueFromBuffer internal operation with arguments A.buffer.[[NativeBuffer]], A.byteOffset, index, Type, and IisLittleEndian.

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:

Let size be the size in bytes of the type valueType.

Let bytes be the array of bytes from nativeBuffer between offset byteOffset+(index*size) and offset byteOffset+((index+1)* × size)-1 inclusive.

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.

If valueType is Float32 and rawValue is a Float32 representation of IEEE754 NaN, return the NaN Number value.

Else, if valueType is Float64 and rawValue is a Float64 representation of IEEE754 NaN, return the NaN Number value.

Else, return the Number value that that represents the same numeric value as rawValue

15.13.6.5.3 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 }.

15.13.6.5.4 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 }.

15.13.6.5.5 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 }.

15.13.6.5.6 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 }.

Page 368: [33] Deleted **Rev 16 Allen Wirfs-Brock** **6/23/2013 10:31:00 AM**

The following sections define DavteaViews is derived from the Kronos specification. This material is a very early draft based upon the strawman at http://wiki.ecmascript.org/doku.php?id=strawman:typed_arrays . This material still needs significant work to fully integrate it into the ES6 spec. and also to integrate typed arrays with ES6 binary data.

Don't waste a lot of time reviewing this material until it is closer to a finished state.

Page 371: [34] Deleted **Rev 16 Allen Wirfs-Brock** **6/22/2013 8:28:00 PM**

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

Let O be ToObject(this)
Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**
If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.
Return GetValue(byteOffset, isLittleEndian, Uint16)

Page 371: [35] Deleted **Rev 16 Allen Wirfs-Brock** **6/22/2013 8:29:00 PM**

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

Let O be ToObject(this)
Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**
If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.
Return GetValue(byteOffset, isLittleEndian, Int32)

15.13.7.4.7

1.1.1.1.1

Page 371: [36] Deleted **Rev 16 Allen Wirfs-Brock** **6/22/2013 8:30:00 PM**

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

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

15.13.7.4.8

1.1.1.1.2

Page 371: [37] Deleted **Rev 16 Allen Wirfs-Brock** **6/22/2013 8:31:00 PM**

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

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

Return GetValue(byteOffset, isLittleEndian, Float32)

15.13.7.4.9

1.1.1.1.3

Page 371: [38] Deleted	Rev 16 Allen Wirfs-Brock	6/22/2013 8:32:00 PM

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

Let O be ToObject(this)
Let isLittleEndian be ToBoolean(littleEndian) if provided, else **false**
If the [[Class]] internal data property of O is not "DataView", throw raise a **TypeError** exception.
Return GetValue(byteOffset, isLittleEndian, Float64)

15.13.7.4.10

1.1.1.1.4

Page 373: [39] Deleted	Rev 14 Allen Wirfs-Brock	2/6/2013 4:22:00 PM

15.14.1 Abstract Operations For Map Objects

15.14.1.1 MapInitializsation

The abstract operation MapInitializsation with arguments *object* and *iterable* is used to initialize an object as a map. It performs the following steps:

If Type(*obj*) is not Object, throw a **TypeError** exception.
If *obj* alreadydoes not have has a [[MapData]] internal data property, throw a **TypeError** exception.
If the result of calling the [[GetIsExtensible]] internal propertymethod of *obj* is **false**, throw a **TypeError** exception.
If *iterable* is not **undefined**, then
    Let *iterable* be ToObject(*iterable*).
    ReturnIfAbrupt(*iterable*)
    Let *iterator* be the intrinsic symbol @@iterator.
    Let *itr* be the result of calling the Invoke(*iterableobj*, abstraction operation with *iterator*, *obj*, and an empty List as arguments).
    ReturnIfAbrupt(*itr*).
    Let *adder* be the result of calling the [[Get]] internal method of(*obj* with argument, **`"set"`**).
    ReturnIfAbrupt(*adderaddr*).
    If IsCallable(*adderaddr*) is **false**, throw a **TypeError** Exception.
Add a [[MapData]] internal data property to *obj*.
Set *obj*'s [[MapData]] internal data methodproperty to a new empty List.
If *iterable* is **undefined**, return *obj*.
Repeat
    Let *next* be the result of performing Invoke(*itr*, **`with arguments "next"`**), itr, and an empty arguments List.
    If IteratorComplete(*next*) is **true**, then return NormalCompletion(*obj*).
    Let *next* be ToObject(*next*).
    ReturnIfAbrupt(*next*).
    Let *k* be the result of calling the [[Get]] internal method of(*next* with argument,, **`"0"`**).
    ReturnIfAbrupt(*k*).
    Let *v* be the result of calling the [[Get]] internal method of(*next* with argument, **`"1"`**).
    ReturnIfAbrupt(*v*).
    Let *status* be the result of calling the [[Call]] internal method of *adder addr* with arguments *obj* as *thisArgument* and a List whose elements are *k* and *v* as *argumentsList*.
    ReturnIfAbrupt(*status*).

Page 373: [40] Deleted **Rev 13 Allen Wirfs-Brock** **12/7/2012 4:36:00 PM**

If *m* is **undefined** or the intrinsic %MapPrototype%

Let *map* be the result of the abstract operation ObjectCreate (15.2) with the intrinsic %MapPrototype% as the argument.

Else

Page 373: [41] Deleted **Rev 14 Allen Wirfs-Brock** **2/6/2013 1:21:00 PM**

Let *map* be the result of ToObject(*m*).

ReturnIfAbrupt(*map*).