TypedArray Update/Issues

Allen Wirfs-Brock

Khronos??

- We're essentially superseding their work
- Do we need to be talking to them? John?

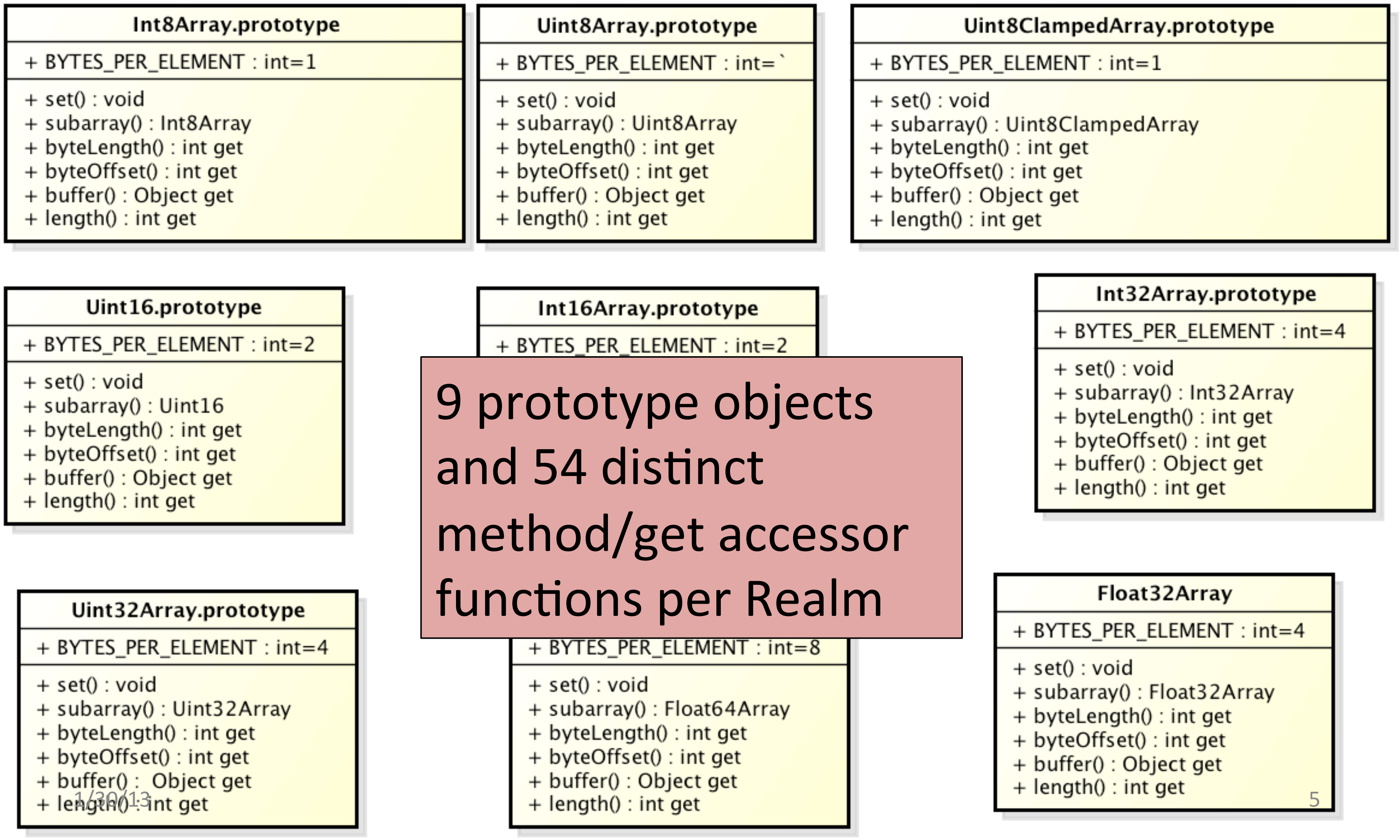
Integrate into ES spec.

- ES spec. conventions and semantics not WebIDL
  - Khronos spec. not necessarily tracking WebIDL evolution…
  - Eg, instanceof ??
- Lot's of implementation differences among browsers at MOP level to straighten out.
- TypedArrays are subclassable

Max Length

- Currently Kronos spec’s all lengths as Uint32
- Seems not very future friendly, especially for byte-sized element arrays
  - For example, a Uint8Array might map to a large real memory-mapped buffer bigger then 4GB

Khronos/W3C TypedArray Objects



Int8Array.prototype
+ BYTES_PER_ELEMENT : int=1
+ set() : void
+ subarray() : Int8Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Uint8Array.prototype
+ BYTES_PER_ELEMENT : int=`
+ set() : void
+ subarray() : Uint8Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Uint8ClampedArray.prototype
+ BYTES_PER_ELEMENT : int=1
+ set() : void
+ subarray() : Uint8ClampedArray
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Uint16.prototype
+ BYTES_PER_ELEMENT : int=2
+ set() : void
+ subarray() : Uint16
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Int16Array.prototype
+ BYTES_PER_ELEMENT : int=2

Int32Array.prototype
+ BYTES_PER_ELEMENT : int=4
+ set() : void
+ subarray() : Int32Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

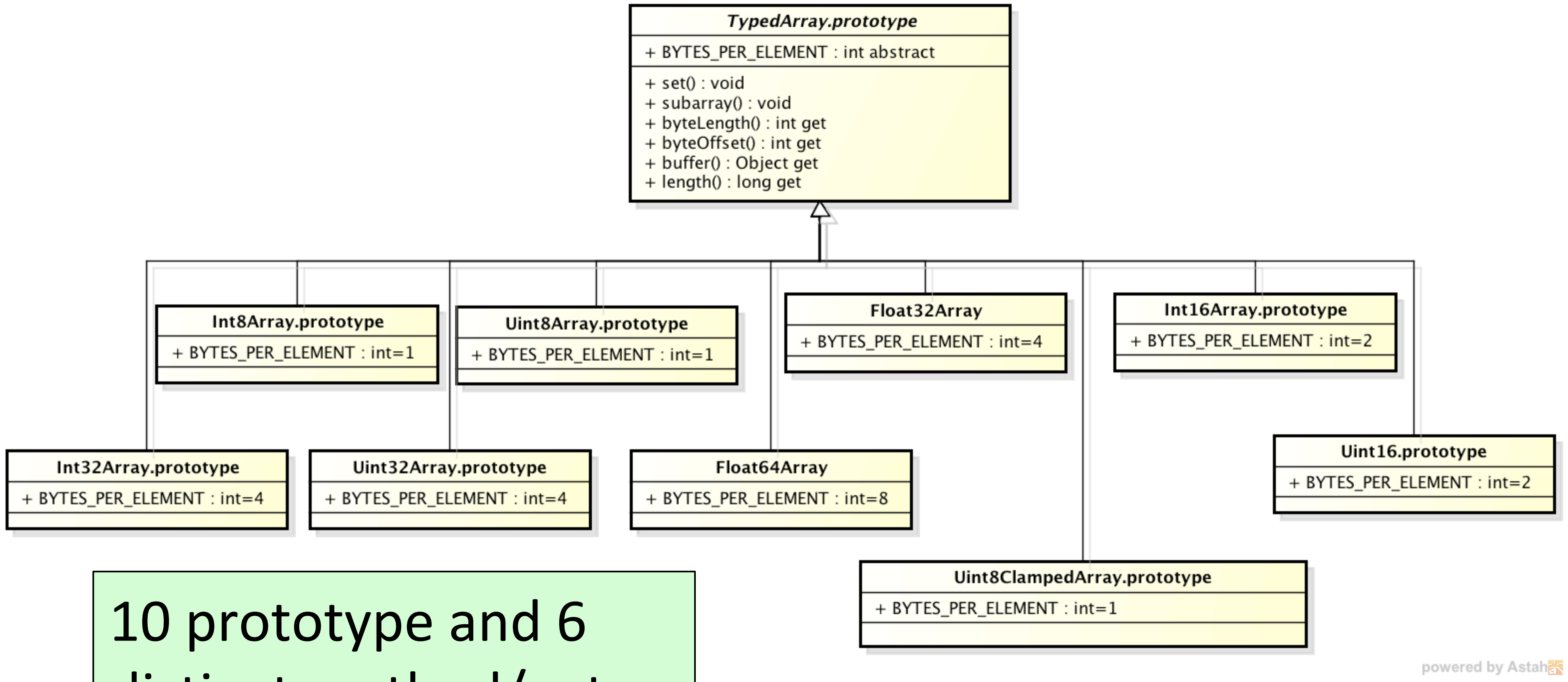
9 prototype objects and 54 distinct method/get accessor functions per Realm

Uint32Array.prototype
+ BYTES_PER_ELEMENT : int=4
+ set() : void
+ subarray() : Uint32Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

+ BYTES_PER_ELEMENT : int=8
+ set() : void
+ subarray() : Float64Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Float32Array
+ BYTES_PER_ELEMENT : int=4
+ set() : void
+ subarray() : Float32Array
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : int get

Prototype hierarchy factoring



TypedArray.prototype
+ BYTES_PER_ELEMENT : int abstract
+ set() : void
+ subarray() : void
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : long get
Int8Array.prototype
+ BYTES_PER_ELEMENT : int=1
Uint8Array.prototype
+ BYTES_PER_ELEMENT : int=1
Float32Array
+ BYTES_PER_ELEMENT : int=4
Int16Array.prototype
+ BYTES_PER_ELEMENT : int=2
Int32Array.prototype
+ BYTES_PER_ELEMENT : int=4
Uint32Array.prototype
+ BYTES_PER_ELEMENT : int=4
Float64Array
+ BYTES_PER_ELEMENT : int=8
Uint16.prototype
+ BYTES_PER_ELEMENT : int=2
Uint8ClampedArray.prototype
+ BYTES_PER_ELEMENT : int=1
powered by Astah

10 prototype and 6 distinct method/get accessor functions per Realm

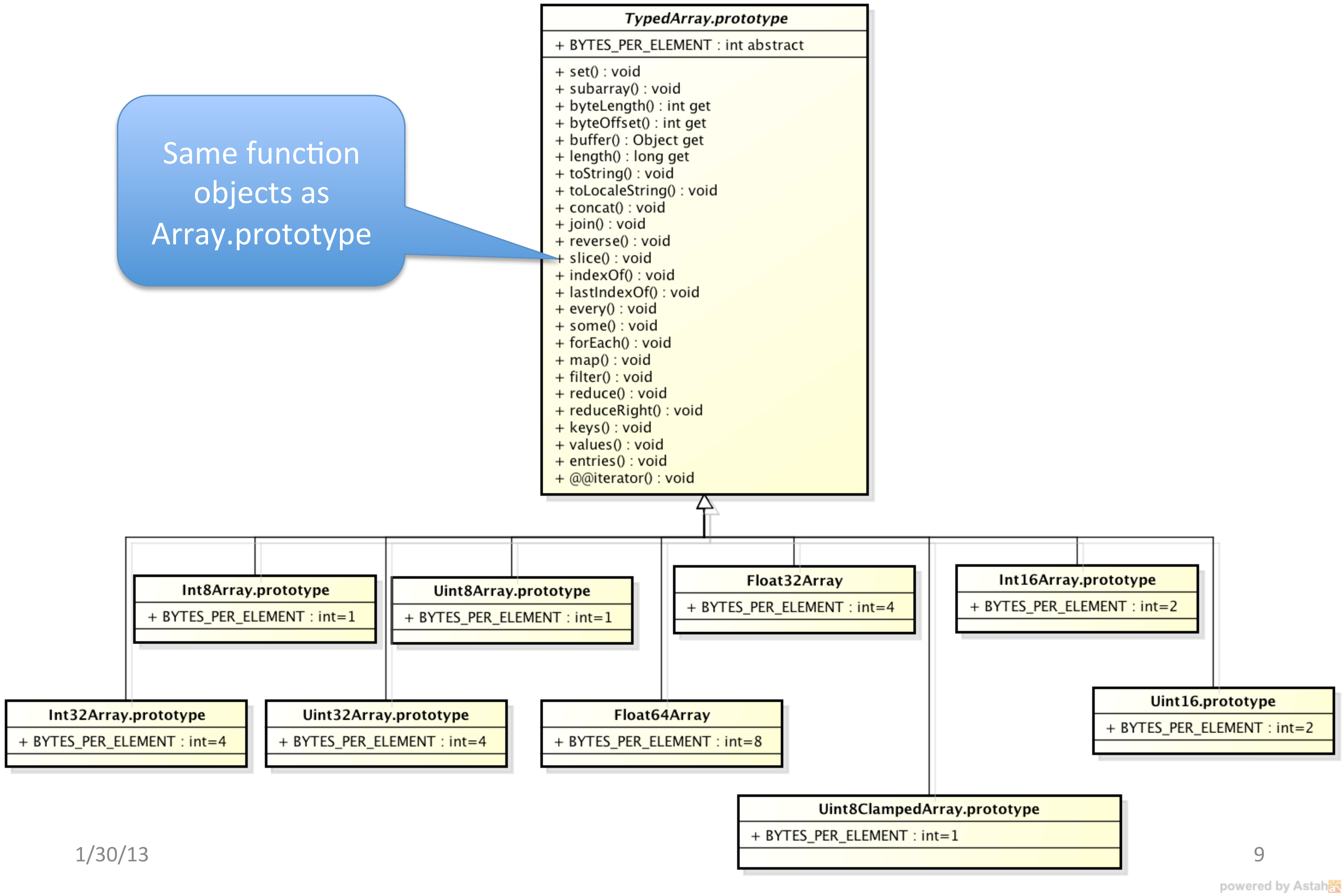
TypedArrays really do act like fixed length, numeric element JS Arrays

- So why not even better Array integration?
- Class methods?
  - TypedArray.of
  - TypedArray.from
- TypedArrays should be iterables?
  - @@iterator
  - @keys
  - @values
  - @entries

Even Better Array Integration

- Other Array.prototype methods that will work just fine on TypedArrays
  - toString, toLocaleString, concat, join, reverse, slice, sort, indexOf, lastIndexOf, every, some, foreach, map, filter, reduce, reduceRight
- Only 5 Array.prototype methods won't work with TypedArrays
  - push, pop, shift, unshift, splice

Could add Array methods to TypedArray


Same function objects as Array.prototype
TypedArray.prototype
+ BYTES_PER_ELEMENT : int abstract
+ set() : void
+ subarray() : void
+ byteLength() : int get
+ byteOffset() : int get
+ buffer() : Object get
+ length() : long get
+ toString() : void
+ toLocaleString() : void
+ concat() : void
+ join() : void
+ reverse() : void
+ slice() : void
+ indexOf() : void
+ lastIndexOf() : void
+ every() : void
+ some() : void
+ forEach() : void
+ map() : void
+ filter() : void
+ reduce() : void
+ reduceRight() : void
+ keys() : void
+ values() : void
+ entries() : void
+ @@iterator() : void
Int8Array.prototype
+ BYTES_PER_ELEMENT : int=1
Uint8Array.prototype
+ BYTES_PER_ELEMENT : int=1
Float32Array
+ BYTES_PER_ELEMENT : int=4
Int16Array.prototype
+ BYTES_PER_ELEMENT : int=2
Int32Array.prototype
+ BYTES_PER_ELEMENT : int=4
Uint32Array.prototype
+ BYTES_PER_ELEMENT : int=4
Float64Array
+ BYTES_PER_ELEMENT : int=8
Uint16.prototype
+ BYTES_PER_ELEMENT : int=2
Uint8ClampedArray.prototype
+ BYTES_PER_ELEMENT : int=1
powered by Astah