Immutable ArrayBuffers for stage 2.7

Mark S. Miller Agoric
Peter Hoddie moddable
Richard Gibson Agoric
Jack-Works

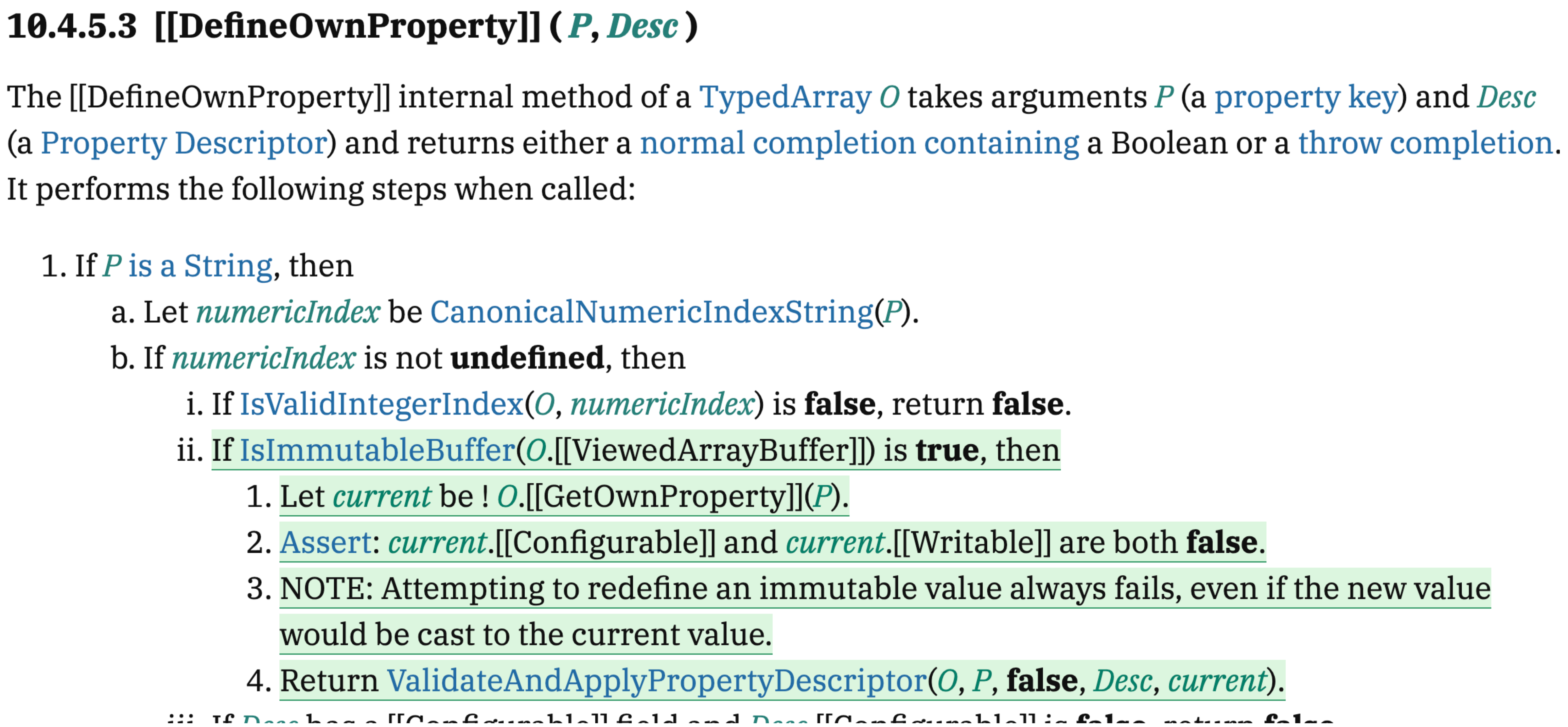
106th Plenary
February 2025
F5, Seattle, WA

TC
39

Recap: Proposed ArrayBuffer API

```
transfer(len?: number) :ArrayBuffer
transferToFixedLength(len?: number) :ArrayBuffer
resize(len: number) :void
slice(start?: number, end?: number) :ArrayBuffer
transferToImmutable() :ArrayBuffer
get immutable: boolean
get detached: boolean
get resizable: boolean
get byteLength: number
get maxByteLength: number
```

Recap: Freezable TypedArrays

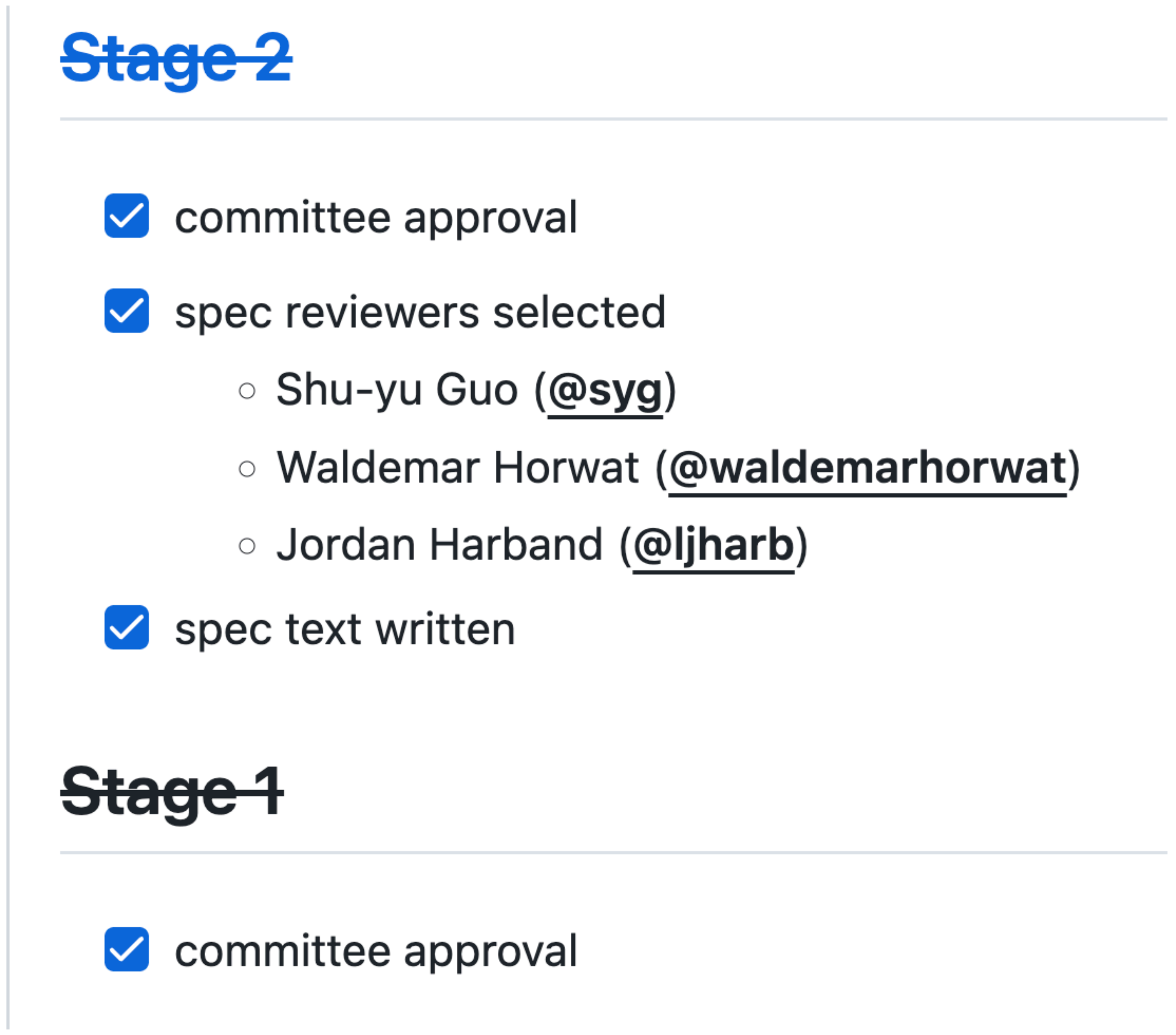


10.4.5.3 [[DefineOwnProperty]] (*P*, *Desc*)

The [[DefineOwnProperty]] internal method of a TypedArray *O* takes arguments *P* (a property key) and *Desc* (a Property Descriptor) and returns either a normal completion containing a Boolean or a throw completion. It performs the following steps when called:

```
1. If P is a String, then
    a. Let numericIndex be CanonicalNumericIndexString(P).
    b. If numericIndex is not undefined, then
        i. If IsValidIntegerIndex(O, numericIndex) is false, return false.
        ii. If IsImmutableBuffer(O.[[ViewedArrayBuffer]]) is true, then
            1. Let current be ! O.[[GetOwnProperty]](P).
            2. Assert: current.[[Configurable]] and current.[[Writable]] are both false.
            3. NOTE: Attempting to redefine an immutable value always fails, even if the new value
               would be cast to the current value.
            4. Return ValidateAndApplyPropertyDescriptor(O, P, false, Desc, current).
        iii. If Desc has a [[Configurable]] field and Desc.[[Configurable]] is false, return false.
```

Status Recap



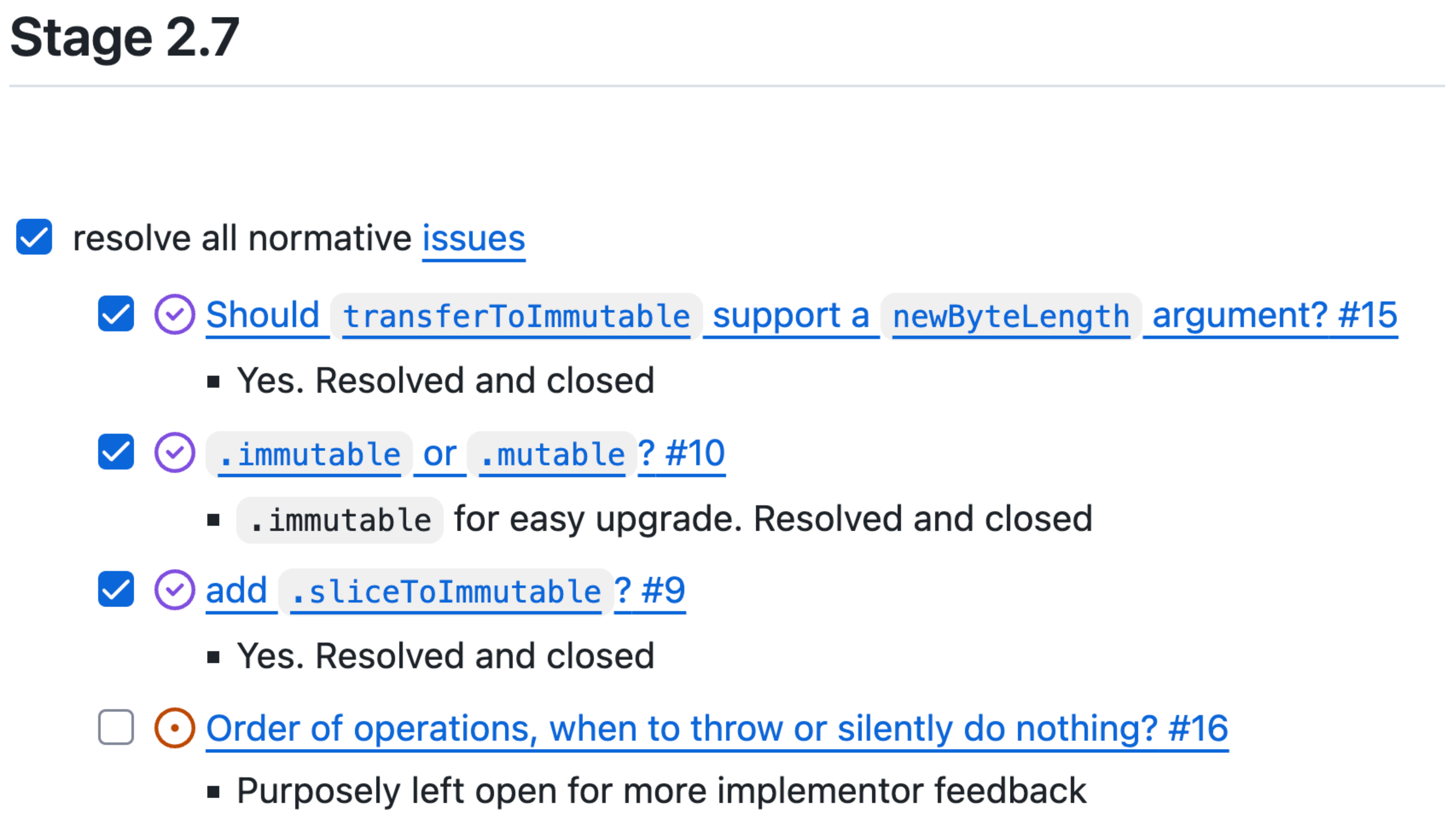
~~Stage 2~~

- [x] committee approval
- [x] spec reviewers selected
  - Shu-yu Guo (**@syg**)
  - Waldemar Horwat (**@waldemarhorwat**)
  - Jordan Harband (**@ljharb**)
- [x] spec text written

~~Stage 1~~

- [x] committee approval

Normative Issues



Stage 2.7

- [x] resolve all normative issues
  - [x] Should `transferToImmutable` support a `newByteLength` argument? #15
    - Yes. Resolved and closed
  - [x] `.immutable` or `.mutable`? #10
    - `.immutable` for easy upgrade. Resolved and closed
  - [x] add `.sliceToImmutable`? #9
    - Yes. Resolved and closed
  - [ ] Order of operations, when to throw or silently do nothing? #16
    - Purposely left open for more implementor feedback

Recap: Proposed ArrayBuffer API

```
transfer(len?: number) :ArrayBuffer
transferToFixedLength(len?: number) :ArrayBuffer
resize(len: number) :void
slice(start?: number, end?: number) :ArrayBuffer
transferToImmutable() :ArrayBuffer
get immutable: boolean
get detached: boolean
get resizable: boolean
get byteLength: number
get maxByteLength: number
```

Proposed ArrayBuffer API

transfer(len?: number) :ArrayBuffer

transferToFixedLength(len?: number) :ArrayBuffer

resize(len: number) :void

slice(start?: number, end?: number) :ArrayBuffer

transferToImmutable(*len?: number*) :ArrayBuffer

sliceToImmutable(start?: number, end?: number) :ArrayBuffer

get immutable: boolean

get detached: boolean

get resizable: boolean

get byteLength: number

get maxByteLength: number

Immutable ArrayBuffer Flavor

~~transfer(len?: number) :ArrayBuffer~~

~~transferToFixedLength(len?: number) :ArrayBuffer~~

~~resize(len: number) :void~~

slice(start?: number, end?: number) :ArrayBuffer

~~**transferToImmutable(*len?: number*) :ArrayBuffer**~~

sliceToImmutable(start?: number, end?: number) :ArrayBuffer

get immutable: true

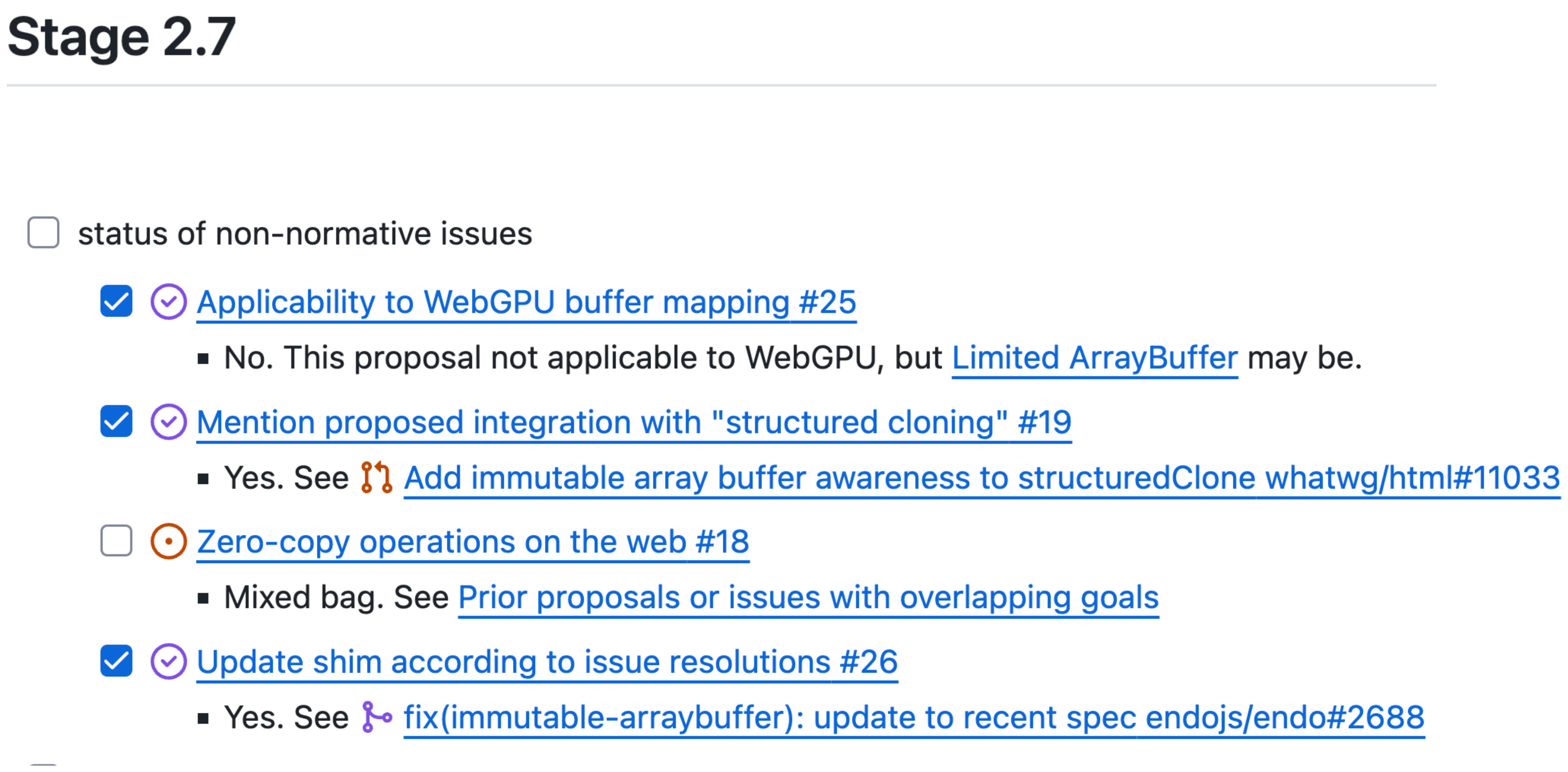
get detached: false

get resizable: false

get byteLength: number

get maxByteLength: same number

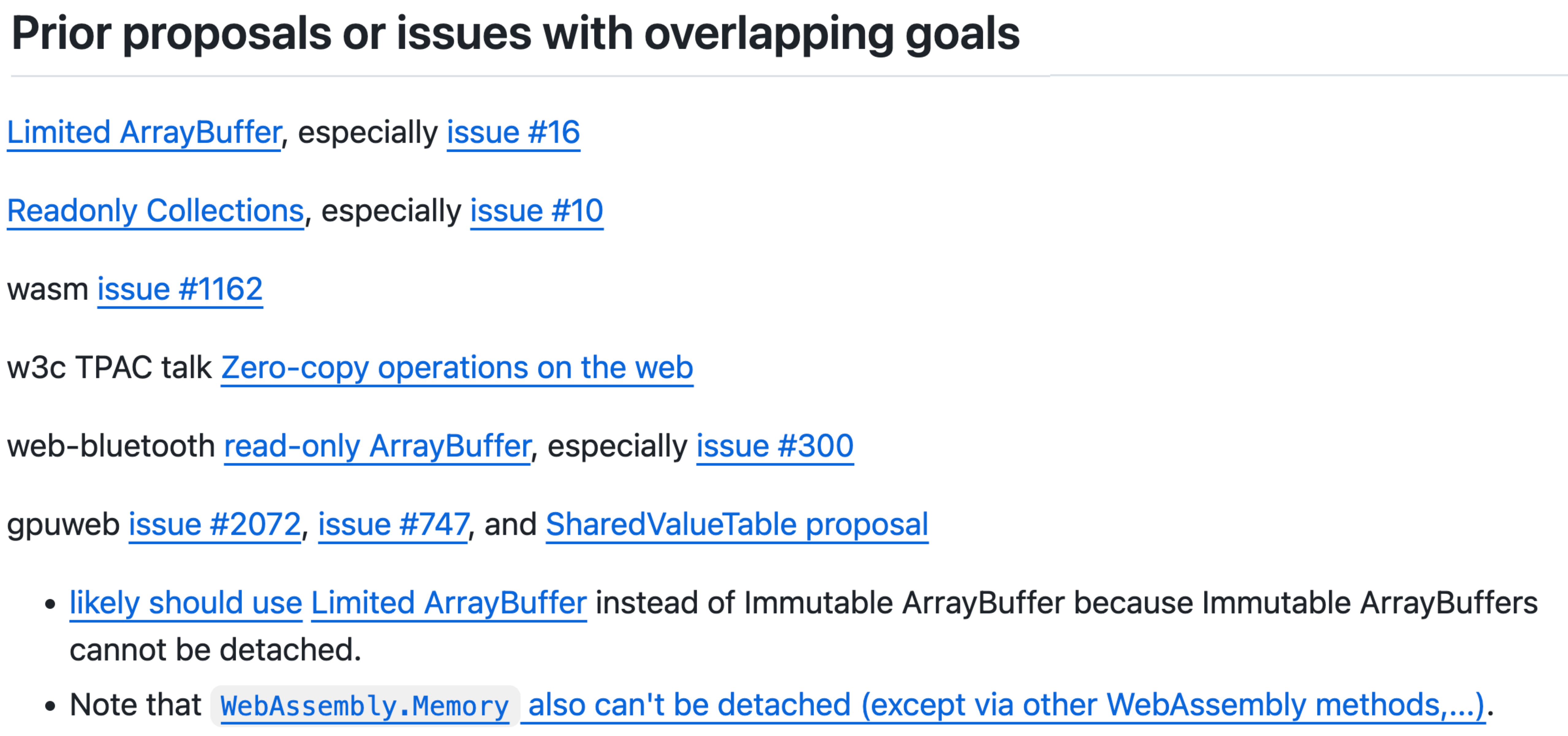
Non-Normative Issues



Stage 2.7

- [ ] status of non-normative issues
  - [x] ✓ Applicability to WebGPU buffer mapping #25
    - No. This proposal not applicable to WebGPU, but Limited ArrayBuffer may be.
  - [x] ✓ Mention proposed integration with "structured cloning" #19
    - Yes. See Add immutable array buffer awareness to structuredClone whatwg/html#11033
  - [ ] ● Zero-copy operations on the web #18
    - Mixed bag. See Prior proposals or issues with overlapping goals
  - [x] ✓ Update shim according to issue resolutions #26
    - Yes. See fix(immutable-arraybuffer): update to recent spec endojs/endo#2688

Zero-copy operations on the web 1/2



Prior proposals or issues with overlapping goals

Limited ArrayBuffer, especially issue #16

Readonly Collections, especially issue #10

wasm issue #1162

w3c TPAC talk Zero-copy operations on the web

web-bluetooth read-only ArrayBuffer, especially issue #300

gpuweb issue #2072, issue #747, and SharedValueTable proposal

- likely should use Limited ArrayBuffer instead of Immutable ArrayBuffer because Immutable ArrayBuffers cannot be detached.
- Note that `WebAssembly.Memory` also can't be detached (except via other WebAssembly methods,...).

Zero

Prior pr

Limited Arra

Readonly C

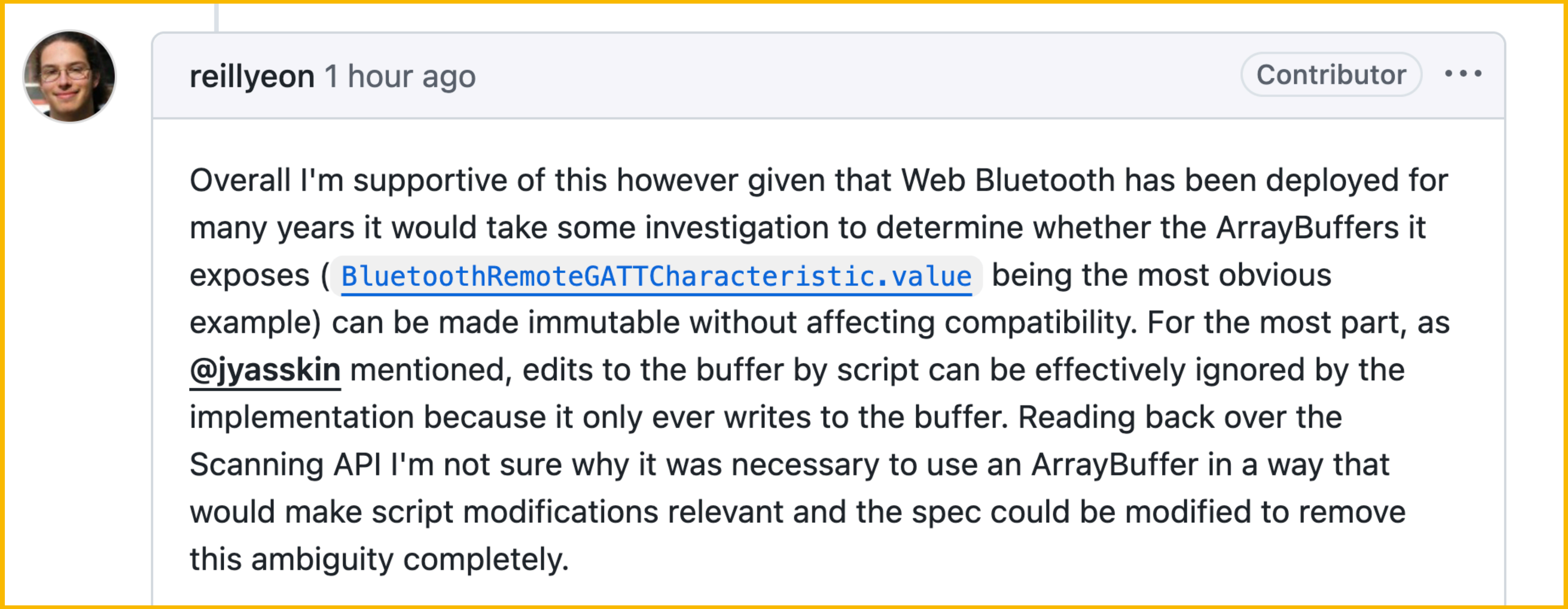
wasm issue #1

w3c TPAC talk Zero-copy operations on the web

web-bluetooth read-only ArrayBuffer, especially issue #300

gpuweb issue #2072, issue #747, and SharedValueTable proposal

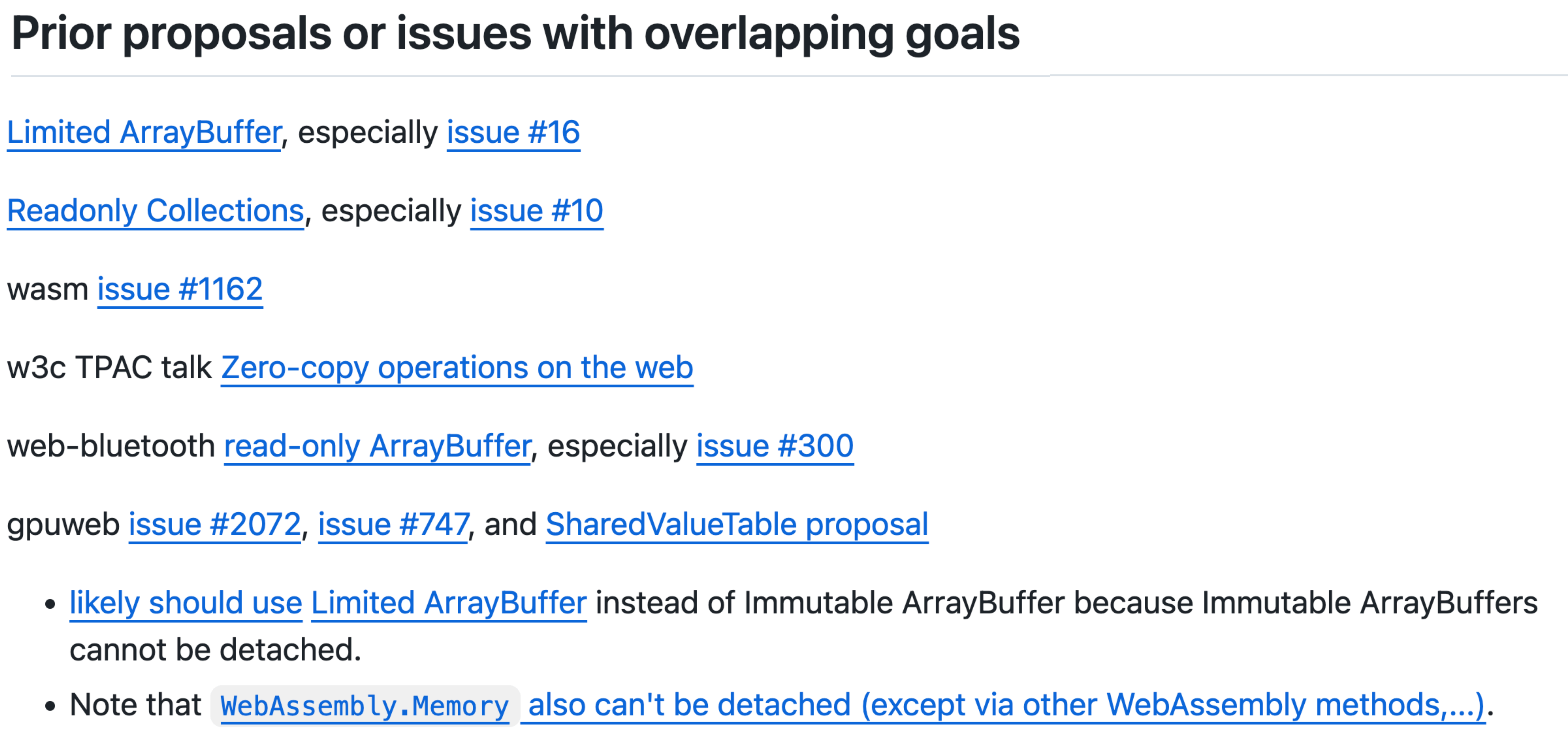
- likely should use Limited ArrayBuffer instead of Immutable ArrayBuffer because Immutable ArrayBuffers cannot be detached.
- Note that `WebAssembly.Memory` also can't be detached (except via other WebAssembly methods,...).



reillyeon 1 hour ago · Contributor

Overall I'm supportive of this however given that Web Bluetooth has been deployed for many years it would take some investigation to determine whether the ArrayBuffers it exposes (`BluetoothRemoteGATTCharacteristic.value` being the most obvious example) can be made immutable without affecting compatibility. For the most part, as **@jyasskin** mentioned, edits to the buffer by script can be effectively ignored by the implementation because it only ever writes to the buffer. Reading back over the Scanning API I'm not sure why it was necessary to use an ArrayBuffer in a way that would make script modifications relevant and the spec could be modified to remove this ambiguity completely.

Zero-copy operations on the web 1/2



Prior proposals or issues with overlapping goals

Limited ArrayBuffer, especially issue #16

Readonly Collections, especially issue #10

wasm issue #1162

w3c TPAC talk Zero-copy operations on the web

web-bluetooth read-only ArrayBuffer, especially issue #300

gpuweb issue #2072, issue #747, and SharedValueTable proposal

- likely should use Limited ArrayBuffer instead of Immutable ArrayBuffer because Immutable ArrayBuffers cannot be detached.
- Note that `WebAssembly.Memory` also can't be detached (except via other WebAssembly methods,...).

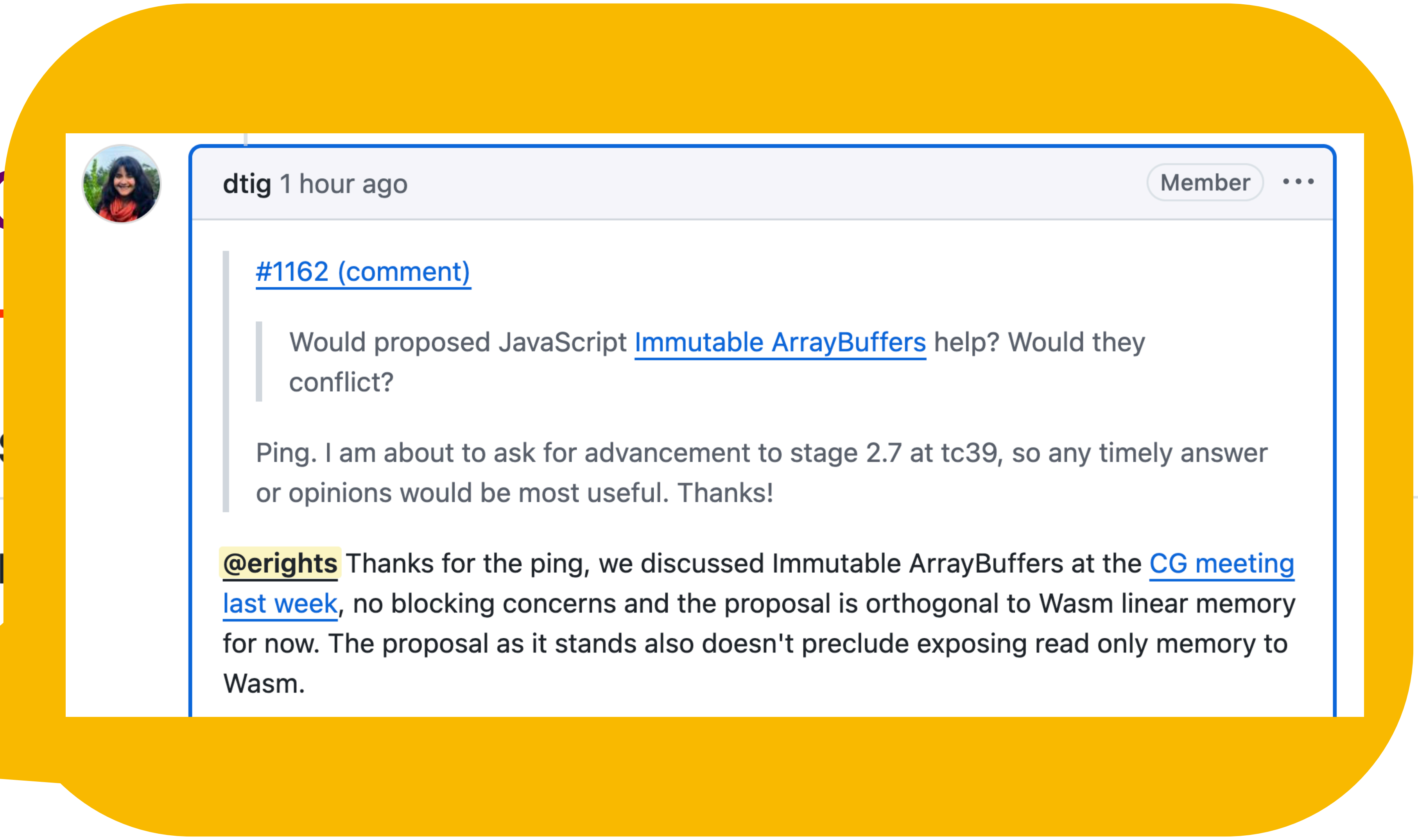
Zero-copy

Prior proposals or i

Limited ArrayBuffer, especial

Readonly Collections, esp

wasm issue #1162



dtig 1 hour ago Member

> #1162 (comment)
>
> > Would proposed JavaScript Immutable ArrayBuffers help? Would they conflict?
>
> Ping. I am about to ask for advancement to stage 2.7 at tc39, so any timely answer or opinions would be most useful. Thanks!

@erights Thanks for the ping, we discussed Immutable ArrayBuffers at the CG meeting last week, no blocking concerns and the proposal is orthogonal to Wasm linear memory for now. The proposal as it stands also doesn't preclude exposing read only memory to Wasm.

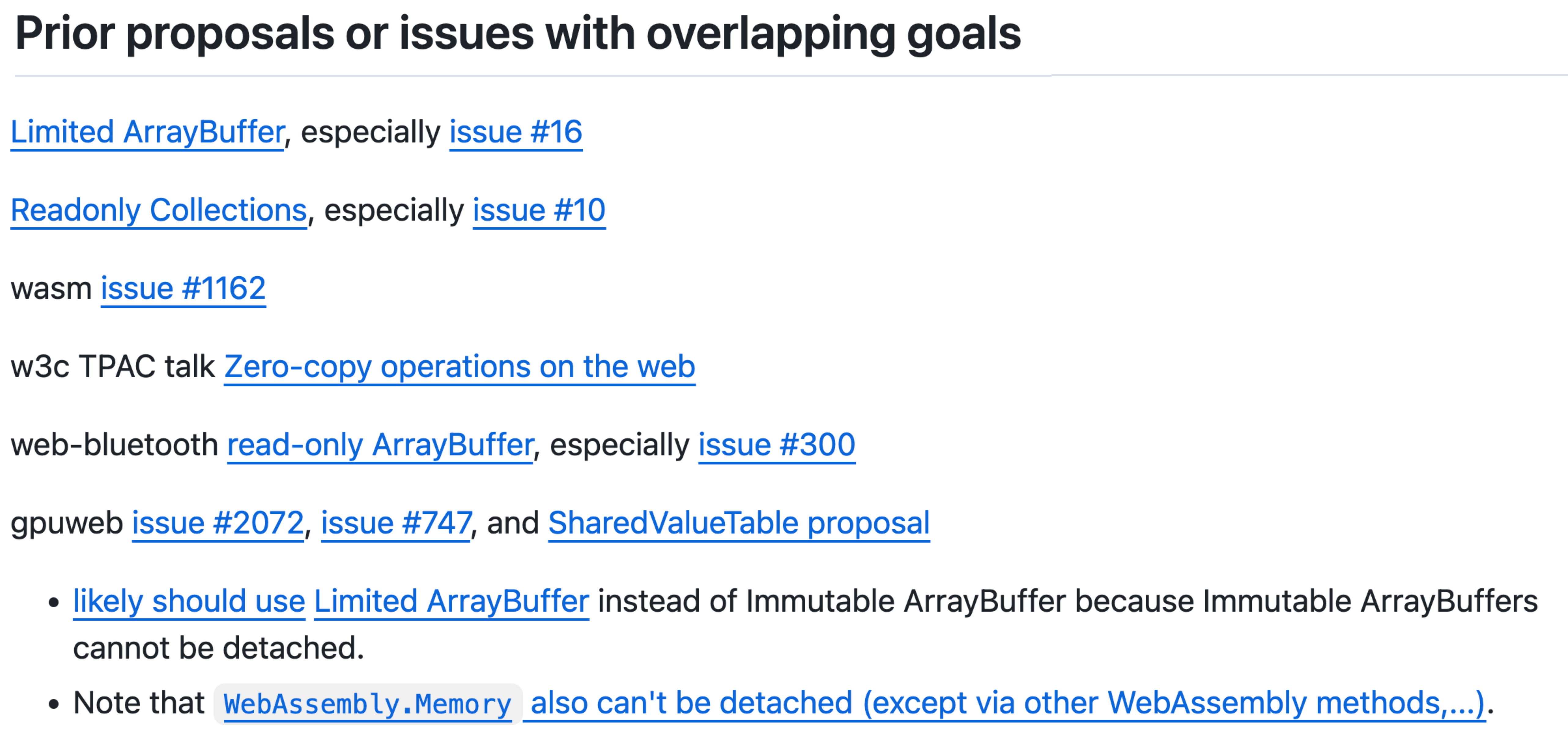
w3c TPAC talk Zero-copy operations on the web

web-bluetooth read-only ArrayBuffer, especially issue #300

gpuweb issue #2072, issue #747, and SharedValueTable proposal

- likely should use Limited ArrayBuffer instead of Immutable ArrayBuffer because Immutable ArrayBuffers cannot be detached.
- Note that `WebAssembly.Memory` also can't be detached (except via other WebAssembly methods,...).

Zero-copy operations on the web 1/2



Prior proposals or issues with overlapping goals

Limited ArrayBuffer, especially issue #16

Readonly Collections, especially issue #10

wasm issue #1162

w3c TPAC talk Zero-copy operations on the web

web-bluetooth read-only ArrayBuffer, especially issue #300

gpuweb issue #2072, issue #747, and SharedValueTable proposal

- likely should use Limited ArrayBuffer instead of Immutable ArrayBuffer because Immutable ArrayBuffers cannot be detached.
- Note that `WebAssembly.Memory` also can't be detached (except via other WebAssembly methods,...).

Zero-copy operations on the web 2/2

Prior proposals or issues with overlapping goals

webidl Frozen Array

webcodecs issue #80, issue #104, and issue #212

web transport issue #131

- unlikely because Chrome (and likely others) copy when crossing address spaces.
- But possible: see Even when talking between different processes, each with their own address space, for a huge enough buffer ...

whatwg streams issue #495

- unlikely because, well, they are streams, not buffers.

w3c machine learning workshop issue #93

Zero-copy operations on the web 2/2

Prior proposals or issues with overlapping goals

webidl Frozen Array

webcodecs issue #80, i

Discussed CLDR w Shane over lunch…

web transport issue #131

- unlikely because Chrome (and likely others) copy when crossing address spaces.
- But possible: see Even when talking between different processes, each with their own address space, for a huge enough buffer ...

whatwg streams issue #495

- unlikely because, well, they are streams, not buffers.

w3c machine learning workshop issue #93

Zero-copy operations on the web 2/2

Prior proposals or issues with overlapping goals

webidl Frozen Array

webcodecs issue #80, issue #104, and issue #212

Follow on proposal…

web transport issue #131

- unlikely because Chrome (and likely others) copy when crossing address spaces.
- But possible: see Even when talking between different processes, each with their own address space, for a huge enough buffer …

whatwg streams issue #495

- unlikely because, well, they are streams, not buffers.

w3c machine learning workshop issue #93

Zero-copy operations on the web 2/2

Prior proposals or issues with overlapping goals

webidl Frozen Array

webcodecs issue #80, issue #104, and issue #212

web transport issue #131

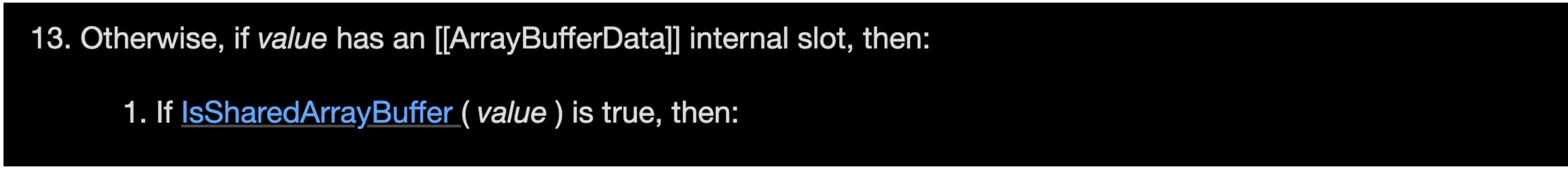
- unlikely because Chrome (and likely others) copy when crossing address spaces.
- But possible: see Even when talking between different processes, each with their own address space, for a huge enough buffer ...

whatwg streams issue #495

- unlikely because, well, they are streams, not buffers.

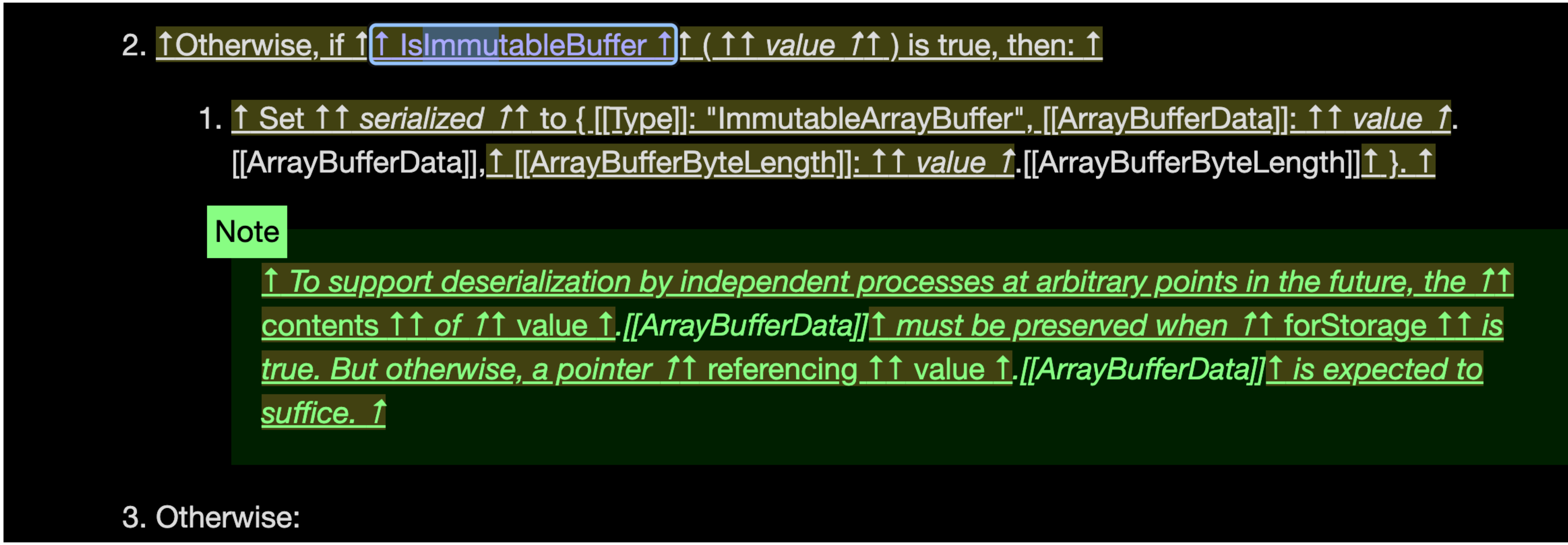
w3c machine learning workshop issue #93

Proposed mod to Structured Clone 1/3



13. Otherwise, if *value* has an [[ArrayBufferData]] internal slot, then:

    1. If IsSharedArrayBuffer (*value*) is true, then:



2. ↑Otherwise, if ↑↑ IsImmutableBuffer ↑↑ (↑↑ *value* ↑↑) is true, then: ↑

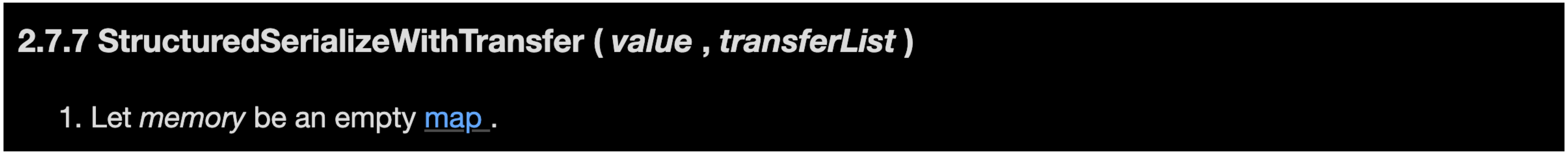
    1. ↑ Set ↑↑ *serialized* ↑↑ to { [[Type]]: "ImmutableArrayBuffer", [[ArrayBufferData]]: ↑↑ *value* ↑. [[ArrayBufferData]],↑ [[ArrayBufferByteLength]]: ↑↑ *value* ↑.[[ArrayBufferByteLength]]↑ }. ↑

    Note

    ↑ To support deserialization by independent processes at arbitrary points in the future, the ↑↑ contents ↑↑ of ↑↑ value *↑.[[ArrayBufferData]]↑ must be preserved when ↑↑* forStorage *↑↑ is true. But otherwise, a pointer ↑↑* referencing *↑↑* value *↑.[[ArrayBufferData]]↑ is expected to suffice. ↑*

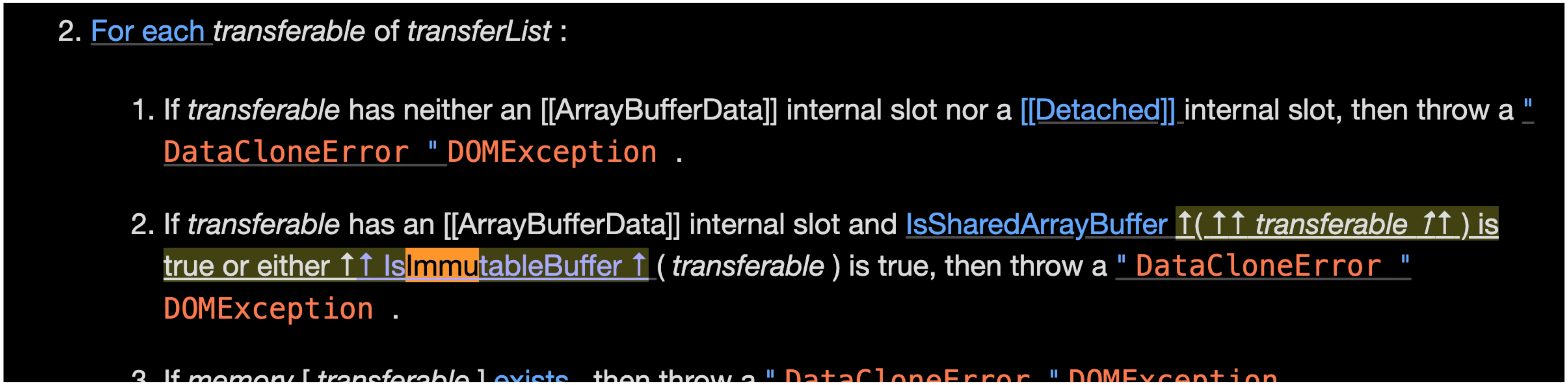
3. Otherwise:

Proposed mod to Structured Clone 2/3



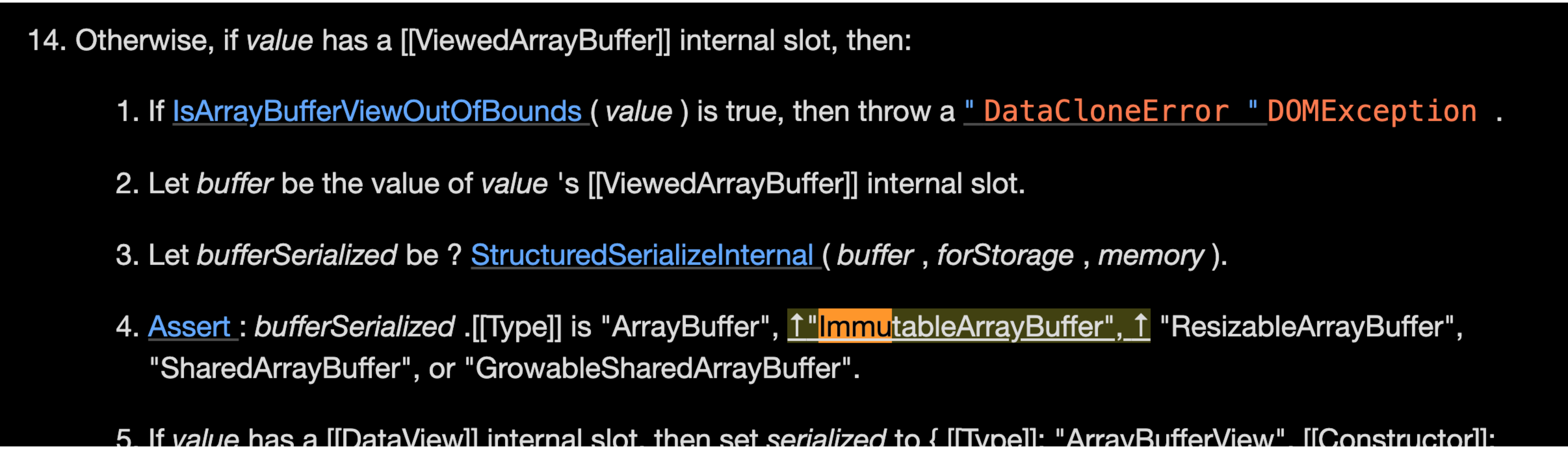
2.7.7 StructuredSerializeWithTransfer (*value* , *transferList*)

1. Let *memory* be an empty map .



2. For each *transferable* of *transferList* :
    1. If *transferable* has neither an [[ArrayBufferData]] internal slot nor a [[Detached]] internal slot, then throw a " `DataCloneError` " `DOMException` .
    2. If *transferable* has an [[ArrayBufferData]] internal slot and IsSharedArrayBuffer ↑(↑↑ *transferable* ↑↑) is true or either ↑↑ IsImmutableBuffer ↑ (*transferable*) is true, then throw a " `DataCloneError` " `DOMException` .
    3. If *memory* [*transferable*] exists , then throw a " `DataCloneError` " `DOMException`

Proposed mod to Structured Clone 3/3



14. Otherwise, if *value* has a [[ViewedArrayBuffer]] internal slot, then:

    1. If IsArrayBufferViewOutOfBounds (*value*) is true, then throw a " `DataCloneError` " `DOMException` .

    2. Let *buffer* be the value of *value* 's [[ViewedArrayBuffer]] internal slot.

    3. Let *bufferSerialized* be ? StructuredSerializeInternal (*buffer* , *forStorage* , *memory*).

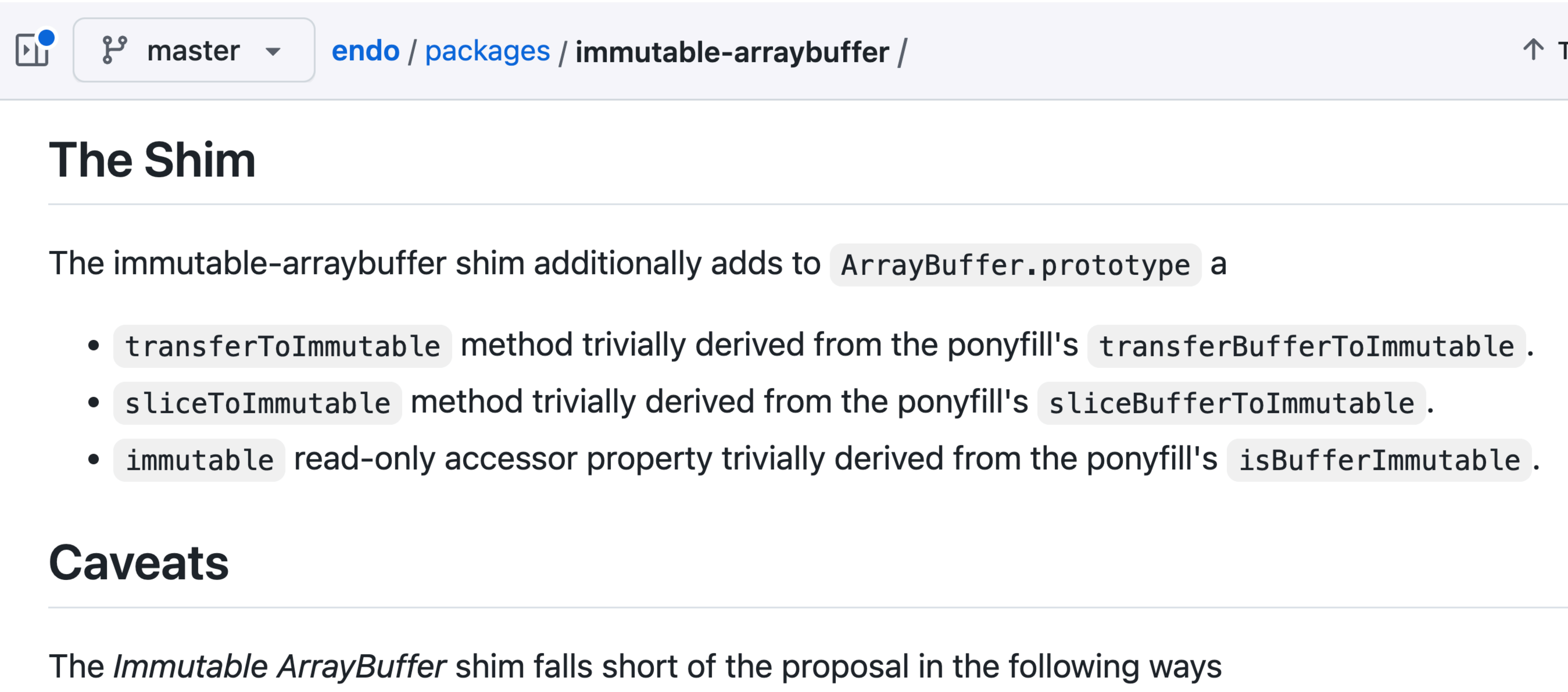
    4. Assert : *bufferSerialized* .[[Type]] is "ArrayBuffer", ↑"ImmutableArrayBuffer", ↑ "ResizableArrayBuffer", "SharedArrayBuffer", or "GrowableSharedArrayBuffer".

    5. If *value* has a [[DataView]] internal slot, then set *serialized* to { [[Type]]: "ArrayBufferView", [[Constructor]]:

Implementor Feedback

Stage 2.7

- [ ] receive implementer feedback
  - [x] XS implementation good. Does not suggest any changes.
  - [x] shim implementation and practical use is necessarily incomplete, but does not suggest any changes.
  - [ ] others...?



The Shim

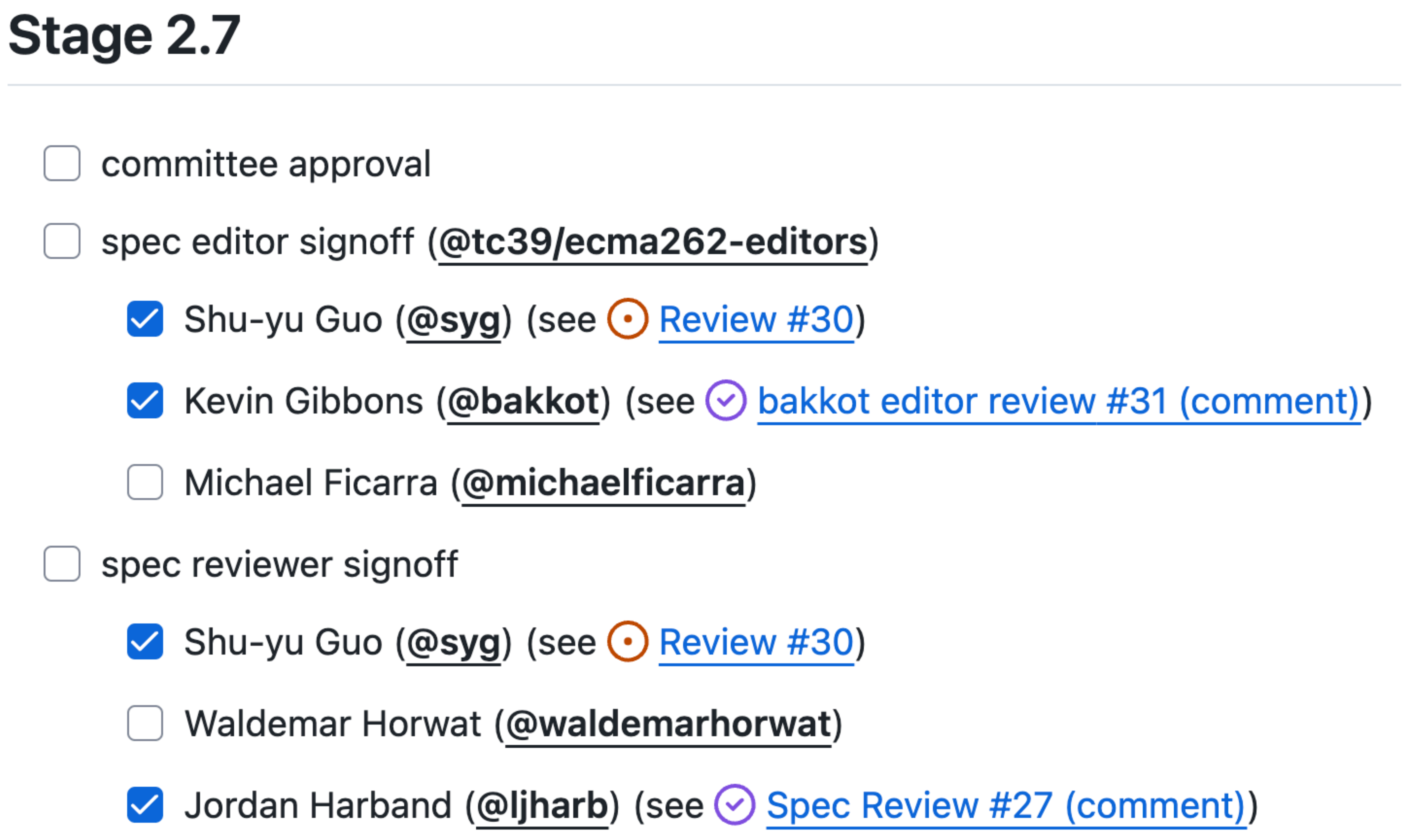
The immutable-arraybuffer shim additionally adds to `ArrayBuffer.prototype` a

- `transferToImmutable` method trivially derived from the ponyfill's `transferBufferToImmutable`.
- `sliceToImmutable` method trivially derived from the ponyfill's `sliceBufferToImmutable`.
- `immutable` read-only accessor property trivially derived from the ponyfill's `isBufferImmutable`.

Caveats

The *Immutable ArrayBuffer* shim falls short of the proposal in the following ways

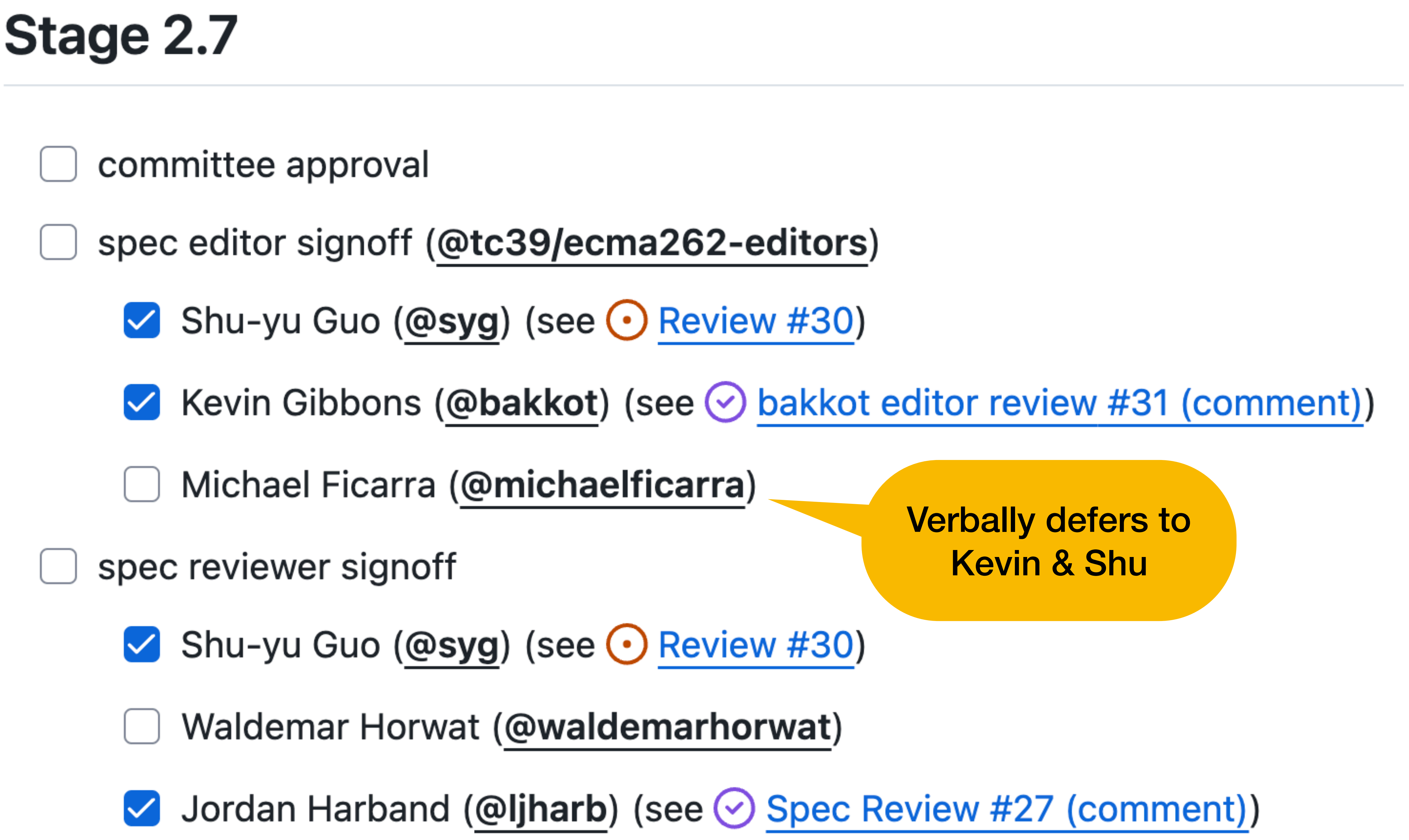
Approval Status



Stage 2.7

- [ ] committee approval
- [ ] spec editor signoff (**@tc39/ecma262-editors**)
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [x] Kevin Gibbons (**@bakkot**) (see bakkot editor review #31 (comment))
  - [ ] Michael Ficarra (**@michaelficarra**)
- [ ] spec reviewer signoff
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [ ] Waldemar Horwat (**@waldemarhorwat**)
  - [x] Jordan Harband (**@ljharb**) (see Spec Review #27 (comment))

Approval Status

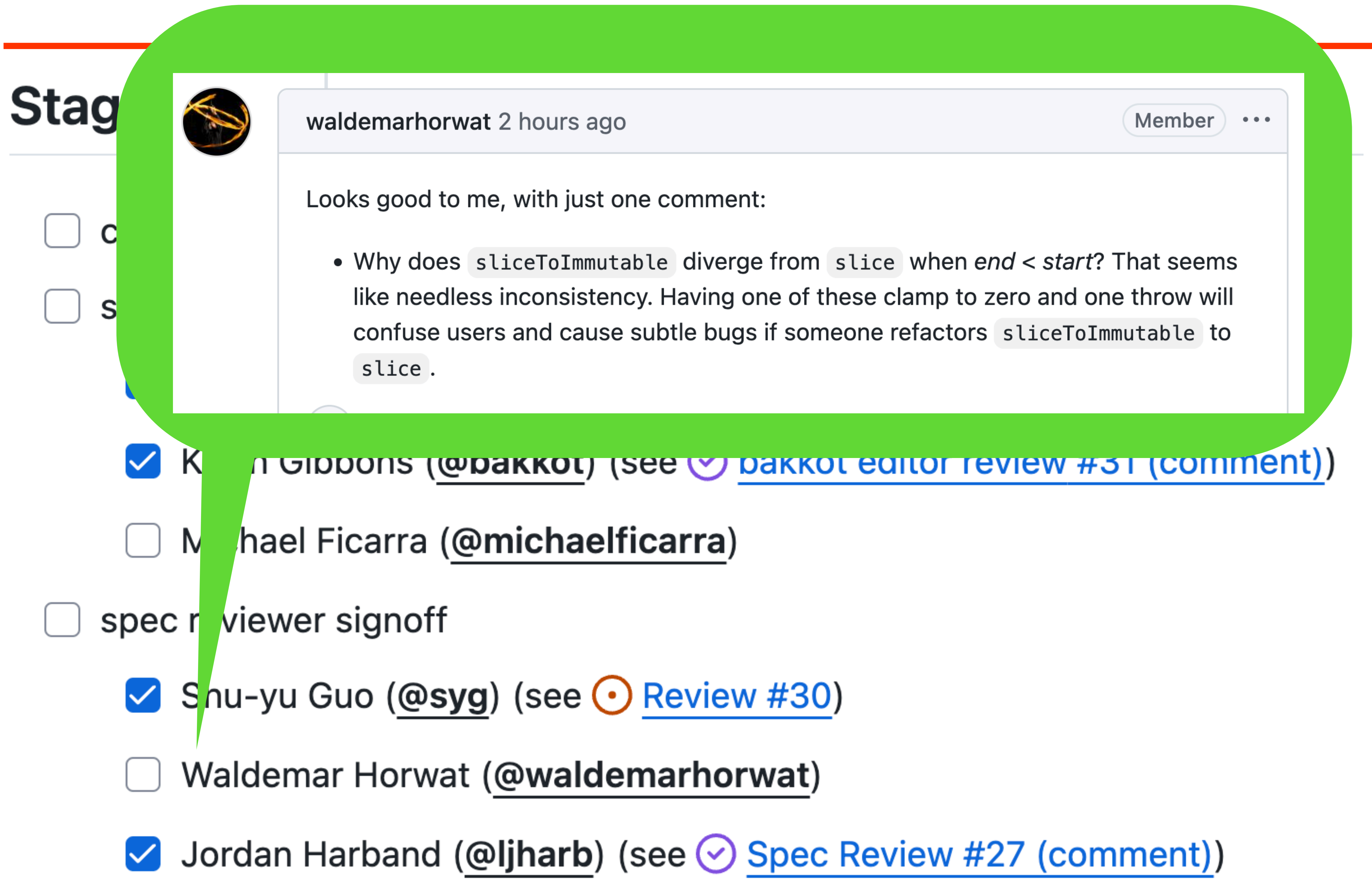


Stage 2.7

- [ ] committee approval
- [ ] spec editor signoff (**@tc39/ecma262-editors**)
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [x] Kevin Gibbons (**@bakkot**) (see bakkot editor review #31 (comment))
  - [ ] Michael Ficarra (**@michaelficarra**)
- [ ] spec reviewer signoff
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [ ] Waldemar Horwat (**@waldemarhorwat**)
  - [x] Jordan Harband (**@ljharb**) (see Spec Review #27 (comment))

Verbally defers to Kevin & Shu

Approval Status



waldemarhorwat 2 hours ago Member

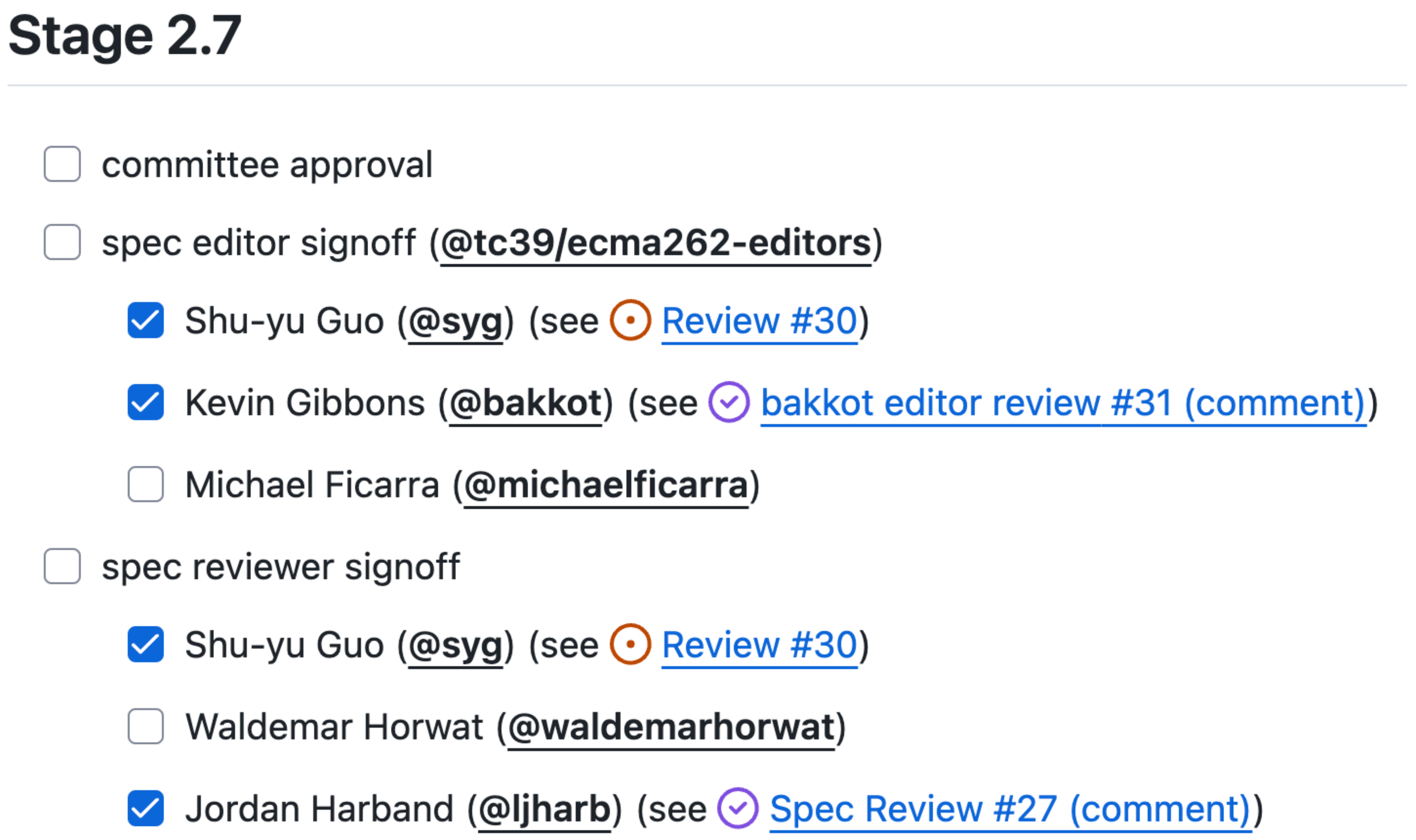
Looks good to me, with just one comment:

- Why does `sliceToImmutable` diverge from `slice` when *end* < *start*? That seems like needless inconsistency. Having one of these clamp to zero and one throw will confuse users and cause subtle bugs if someone refactors `sliceToImmutable` to `slice`.

Stag

- [ ] c
- [ ] s
  - [x] K n Gibbons (@bakkot) (see bakkot editor review #31 (comment))
  - [ ] M hael Ficarra (@michaelficarra)
- [ ] spec r viewer signoff
  - [x] S hu-yu Guo (@syg) (see Review #30)
  - [ ] Waldemar Horwat (@waldemarhorwat)
  - [x] Jordan Harband (@ljharb) (see Spec Review #27 (comment))

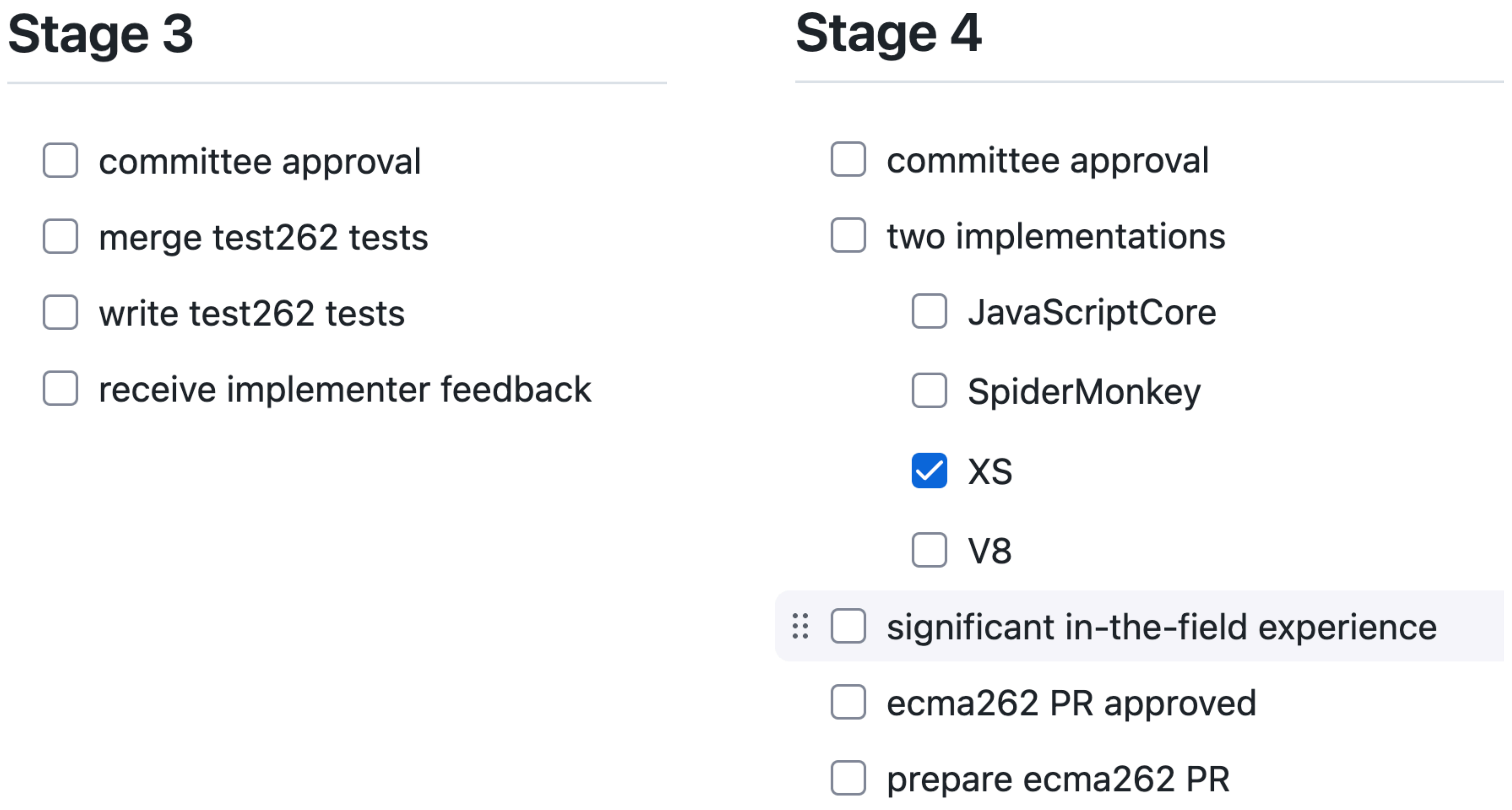
Approval Status



Stage 2.7

- [ ] committee approval
- [ ] spec editor signoff (**@tc39/ecma262-editors**)
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [x] Kevin Gibbons (**@bakkot**) (see bakkot editor review #31 (comment))
  - [ ] Michael Ficarra (**@michaelficarra**)
- [ ] spec reviewer signoff
  - [x] Shu-yu Guo (**@syg**) (see Review #30)
  - [ ] Waldemar Horwat (**@waldemarhorwat**)
  - [x] Jordan Harband (**@ljharb**) (see Spec Review #27 (comment))

Road to Future Stages



Stage 3

- [ ] committee approval
- [ ] merge test262 tests
- [ ] write test262 tests
- [ ] receive implementer feedback

Stage 4

- [ ] committee approval
- [ ] two implementations
  - [ ] JavaScriptCore
  - [ ] SpiderMonkey
  - [x] XS
  - [ ] V8
- [ ] significant in-the-field experience
- [ ] ecma262 PR approved
- [ ] prepare ecma262 PR

Questions? Stage 2.7?