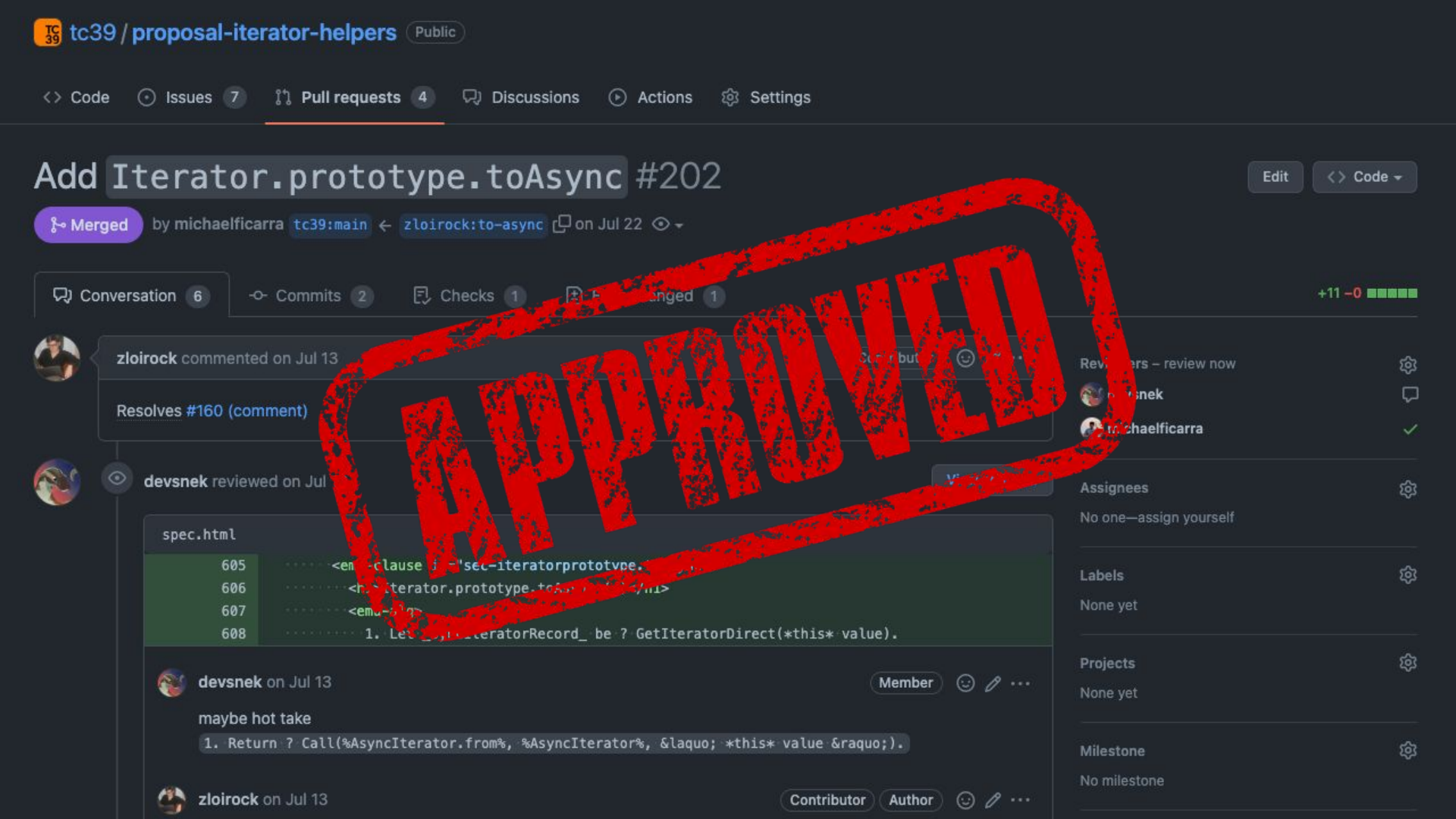
Iterator Helpers update

Michael Ficarra · TC39 · September 2022

with special thanks to Gus Caplan, Kevin Gibbons, and Yulia Startsev

aside: how I would like to receive feedback today

resolutions to previously discussed topics


tc39 / proposal-iterator-helpers
Public
Code
Issues 7
Pull requests 4
Discussions
Actions
Settings
Add Iterator.prototype.toAsync #202
Edit
Code
Merged
by michaelficarra tc39:main ← zloirock:to-async on Jul 22
Conversation 6
Commits 2
Checks 1
Changed 1
+11 −0
zloirock commented on Jul 13
Resolves #160 (comment)
devsnek reviewed on Jul
spec.html
605
606
607
608
1. Let
IteratorRecord_ be ? GetIteratorDirect(*this* value).
devsnek on Jul 13
Member
maybe hot take
1. Return ? Call(%AsyncIterator.from%, %AsyncIterator%, « *this* value »).
zloirock on Jul 13
Contributor
Author
APPROVED
Reviewers – review now
devsnek
michaelficarra
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone

fixes #207: add a counter parameter to Array analogues that have indices #211

Edit Code

Open main ← GH-207

Conversation 1 Commits 2 Checks 1 Files changed 1 +48 −52

michaelficarra commented 23 days ago Member

Fixes #207. Adds "counter" parameters to callbacks of methods that have an `Array.prototype` analogue that passes an index to its callback. Also removes `.indexed()` now that it's equivalent to `.map((a, i) => [i, a])`.

Feedback on #207 has been mixed. I am not yet convinced that this is an improvement, but I am also not opposed to this change if that's what the committee decides.

1

michaelficarra added 2 commits 23 days ago

`fixes #207: add a counter parameter to Array analogues that have them` 7b5e897

`remove no longer necessary .indexed() methods` 0aa02c4

Reviewers – review now

Suggestions

bakkot Request

devsnek Request

At least 1 approving review is required to merge this pull request.

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet

tc39 / proposal-iterator-helpers Public

Code Issues 7 Pull requests 4 Discussions Actions

fixes #115: @@toStringTag is an accessor no

Open main ← GH-115

Conversation 2 Commits 1 Checks 1 Files changed 1

michaelficarra commented 23 days ago

Fixes #115. Option 1: make a gross accessor.

👍 1 👎 2 👀 1

fixes #115: @@toStringTag is an accessor now

ljharb reviewed 23 days ago

spec.html

Add more commits by pushing to the **GH-115** branch on **tc39/proposal-iterator-helper**

This branch has not been deployed
No deployments

tc39 / proposal-iterator-helpers Public

Code Issues 7 Pull requests 4 Discussions Actions

fixes #115: @@toStringTag is writable now #2

Open main ← GH-115-2

Conversation 0 Commits 1 Checks 1 Files changed 1

michaelficarra commented 23 days ago

Fixes #115. Option 2: just make `@@toStringTag` writable.

👍 3

fixes #115: @@toStringTag is writable now

Add more commits by pushing to the **GH-115-2** branch on **tc39/proposal-iterator-help**

This branch has not been deployed
No deployments

At least 1 approving review is required by reviewers with write acc

1 successful check

tc39 / proposal-iterator-helpers Public

Code Issues 7 Pull requests 4 Discussions Actions Settings

fixes #173: add the new intrinsics as properties of something reachable #212

Edit Code

Open main ← GH-173

Conversation 4 Commits 2 Checks 1 Files changed 1 +20 −0

michaelficarra commented 23 days ago Member

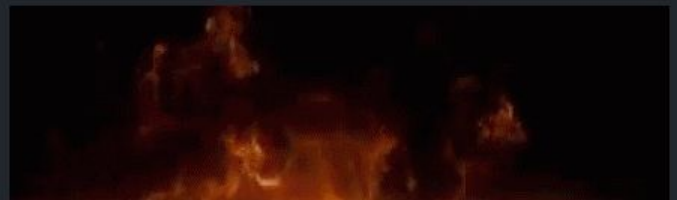
Fixes #173. I really do not want to merge this, but if the committee is so inclined, this should do it.

/cc @erights

1

fixes #173: add the new intrinsics as properties of .from b9df749

devsnek commented 23 days ago Member



Reviewers – review now

bakkot

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

proposal summary (assuming all 3 PRs are accepted)

- Iterator()
- Iterator.from(O)
  - .WrapForValidIteratorPrototype
- Iterator.HelperPrototype
- Iterator.prototype
  - .constructor
  - .map(mapperWithCounter)
  - .filter(predicateWithCounter)
  - .take(limit)
  - .drop(limit)
  - .flatMap(mapperWithCounter)
  - .reduce(reducerWithCounter [, initialValue])
  - .toArray()
  - .forEach(effectWithCounter)
  - .some(predicateWithCounter)
  - .every(predicateWithCounter)
  - .find(predicateWithCounter)
  - [@@toStringTag]
  - .toAsync()

- AsyncIterator()
- AsyncIterator.from(O)
  - .WrapForValidAsyncIteratorPrototype
- AsyncIterator.HelperPrototype
- AsyncIterator.prototype
  - .constructor
  - .map(mapperWithCounter)
  - .filter(predicateWithCounter)
  - .take(limit)
  - .drop(limit)
  - .flatMap(mapperWithCounter)
  - .reduce(reducerWithCounter [, initialValue])
  - .toArray()
  - .forEach(effectWithCounter)
  - .some(predicateWithCounter)
  - .every(predicateWithCounter)
  - .find(predicateWithCounter)
  - [@@toStringTag]

other noteworthy changes

fixes #228: fail earlier when an iterable produces a broken iterator #232

Edit Code

Merged main ← GH-228 1 minute ago

Conversation 0 Commits 2 Checks 1 Files changed 1 +3 −0

michaelficarra commented 3 days ago Member

Fixes #228.

fixes #228: fail earlier when an iterable produces a broken iterator ✓ 1b75f9e

michaelficarra requested review from **bakkot** and **devsnek** 3 days ago

devsnek approved these changes 3 days ago View changes

bergus mentioned this pull request 2 days ago

GetIterator should check for `next` being a method tc39/ecma262#2903 Open

bakkot approved these changes 6 hours ago View changes

Reviewers – review now

bakkot ✓

devsnek ✓

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

new open questions

forEach return value (#217)

- currently: `undefined`
- possibly: "value" from IteratorResult object when its "done" is true
- argument in favour:
  - final values are useful; that's why the generator protocol has them
- argument against:
  - not part of the Iteration subset of the protocol used by for-of/for-await-of
  - already decided earlier that Iterator helpers do not preserve the full generator protocol
  - while technically part of the Iteration protocol, this value is not observable using existing facilities

should flatMap flatten iterables or iterators? (#229)

- currently: `.flatMap` calls `GetIterator` on the result of the passed callback
- possibly: assume the result is already an iterator
- argument in favour:
  - `X.prototype.flatMap` flattens `X`s
  - doesn't currently work with manually implemented iterators that are not also iterable
- argument against:
  - convenient for callbacks that already produce iterables
- considerations:
  - maybe accept both? (see open PR #233)
    - should `Iterator.from` and `AsyncIterator.from` be aligned with this logic?
  - what does async flatMap do when a sync iterator is returned?
  - are primitives (including strings!) iterable if they have a `Symbol.iterator` or should they fail?
    - what will we do about Tuples when they land?

allow interleaved next/return on produced async iterators (#223)

- in manual async iterators
  - you can call `.return` while `.next` is waiting on an async operation
  - so you can trigger cleanup/cancellation before the async operation to completes
- since `.map` has its own internal queue of calls to `.next`/`.return`
  - it never calls `.return` on the underlying iterator while it's waiting on `.next`
  - so `.map` limits the above use case as specified today
- also applies to many other async iterator helpers

Due to significant new design questions, no stage 3 today.

Hopefully next meeting.