Array.fromAsync

avoid double construction of `this` value

Michael Ficarra • May 2023

3. Let *fromAsyncClosure* be a new Abstract Closure with no parameters that captures *C*, *mapfn*, and *thisArg* and performs the following steps when called:
    a. If *mapfn* is **undefined**, let *mapping* be **false**.
    b. Else,
        i. If IsCallable(*mapfn*) is **false**, throw a **TypeError** exception.
        ii. Let *mapping* be **true**.
    c. Let *usingAsyncIterator* be ? GetMethod(*asyncItems*, @@asyncIterator).
    d. If *usingAsyncIterator* is **undefined**, then
        i. Let *usingSyncIterator* be ? GetMethod(*asyncItems*, @@iterator).
    e. If IsConstructor(*C*) is **true**, then
        i. Let *A* be ? Construct(*C*).
    f. Else,
        i. Let *A* be ! ArrayCreate(0).
    g. Let *iteratorRecord* be **undefined**.
    h. If *usingAsyncIterator* is not **undefined**, then
        i. Set *iteratorRecord* to ? GetIterator(*asyncItems*, async, *usingAsyncIterator*).
    i. Else if *usingSyncIterator* is not **undefined**, then
        i. Set *iteratorRecord* to ? CreateAsyncFromSyncIterator(GetIterator(*asyncItems*, sync, *usingSyncIterator*)).
    j. If *iteratorRecord* is not **undefined**, then
        i. Let *k* be 0.
        ii. Repeat,
            1. ...
    k. Else,
        i. NOTE: *asyncItems* is neither an AsyncIterable nor an Iterable so assume it is an array-like object.
        ii. Let *arrayLike* be ! ToObject(*asyncItems*).
        iii. Let *len* be ? LengthOfArrayLike(*arrayLike*).
        iv. If IsConstructor(*C*) is **true**, then
            1. Let *A* be ? Construct(*C*, « 𝔽(*len*) »).
        v. Else,
            1. Let *A* be ? ArrayCreate(*len*).
        vi. Let *k* be 0.
        vii. Repeat, while *k* < *len*,

ptomato (Philip Chimento) 2 weeks ago

Member

I found this as well, while writing test262 tests. Here's a code snippet showing how it's observable from JS:

```
class MyArray {
  constructor(...args) {
    console.log('called with', args);
  }
}
await Array.fromAsync.call(MyArray, {
  length: 2,
  0: 1,
  1: 2
});
```

This logs:

```
called with
called with 2
```

Normative: avoid double construction of `this` value #41

Open by michaelficarra `tc39:main` ← `michaelficarra:patch-2`

Conversation 0 · Commits 1 · Checks 0 · Files changed 1

all commits · File filter · Conversations · Jump to

8 spec.html

```
@@ -122,16 +122,16 @@ <h1><ins>Array.fromAsync ( _asyncItems_ [ , _mapfn_ [ , _thisArg_ ] ] )</ins></h
122 122     1. Let _usingAsyncIterator_ be ? GetMethod(_asyncItems_, @@asyncIterator).
123 123     1. If _usingAsyncIterator_ is *undefined*, then
124 124       1. Let _usingSyncIterator_ be ? GetMethod(_asyncItems_, @@iterator).
125     -   1. If IsConstructor(_C_) is *true*, then
126     -     1. Let _A_ be ? Construct(_C_).
127     -   1. Else,
128     -     1. Let _A_ be ! ArrayCreate(0).
129 125   1. Let _iteratorRecord_ be *undefined*.
130 126   1. If _usingAsyncIterator_ is not *undefined*, then
131 127     1. Set _iteratorRecord_ to ? GetIterator(_asyncItems_, ~async~, _usingAsyncIterator_).
132 128   1. Else if _usingSyncIterator_ is not *undefined*, then
133 129     1. Set _iteratorRecord_ to ? CreateAsyncFromSyncIterator(GetIterator(_asyncItems_, ~sync~, _usingSyncIterator_)).
134 130   1. If _iteratorRecord_ is not *undefined*, then
    131     1. If IsConstructor(_C_) is *true*, then
    132       1. Let _A_ be ? Construct(_C_).
    133     1. Else,
    134       1. Let _A_ be ! ArrayCreate(0).
135 135     1. Let _k_ be 0.
136 136     1. Repeat,
137 137       1. If _k_ &ge; 2<sup>53</sup> - 1, then
```

one small issue:
champion appears absent

withdraw conditional stage 3?
ask for new champion(s)?
both?