ECMAScript Proposal: Array.prototype.flat{Map,ten}

Michael Ficarra & Brian Terlson • 59th Meeting of TC39

Array.prototype.flatten([*depth*])

```
[[0], [1, 2], [3]].flatten()
```

```
⇒ [0, 1, 2, 3]
```

```
[0, [1, [2, [3, [4]]]]].flatten(3)
```

```
⇒ [0, 1, [2, [3, [4]]]].flatten(2)
```

```
⇒ [0, 1, 2, 3, [4]]
```

Array.prototype.flatMap (*callbackfn* [, *thisArg*])

```
[3, 2, 0, 5].flatMap(x => Array(x).fill(x))
```

```
⇒ [3, 2, 0, 5].map(x => Array(x).fill(x)).flatten()
⇒ [[3, 3, 3], [2, 2], [], [5, 5, 5, 5, 5]].flatten()
⇒ [3, 3, 3, 2, 2, 5, 5, 5, 5, 5]
```

Relevant Discussions

isConcatSpreadable vs Iterable

- isConcatSpreadable equivalent to `(xs) => [].concat(...xs)`
- Unexpected iterability: `["string", ["another"]].flatten()`
  - isConcatSpreadable: `["string", "another"]`
  - Iterable: `["s", "t", "r", "i", "n", "g", "another"]`
- Infinite depth flattening: `[[[[[[[["anything iterable"]]]]]]]].flatten(1/0)`
  - isConcatSpreadable: `["anything iterable"]`
  - Iterable: spins forever

throw vs auto-pure for non-spreadables

```
[0, [1, 2], 3].flatMap(x => x)
```

- Current proposal: `[0, 1, 2, 3]`
- Alternative: throw TypeError

treatment of holes

`[0, , 2].flatMap(x => [x])`

- Current proposal: `[0, 2]`
- Alternative 1: `[0, , 2]`
- ~~Alternative 2: `[0, undefined, 2]`~~

`[0, , 2].map(x => [x]).flatten()`

- `[0, 2]`

%TypedArrayPrototype%.flatMap

- probably a good idea
- don't want to include it in this proposal if it will cause delay
- should probably support returning Iterables
- should probably throw for non-Iterables