A better future for comprehensions

Dave Herman
\[\text{[for } (x \text{ of } y) \text{ if } (p(x)) \ f(x)\]\n\[(\text{for } (x \text{ of } y) \text{ if } (p(x)) \ f(x))\]
• Parallel JS is moving in the direction of parallel pipelines – a natural fit for comprehensions.

• Three strikes and you refactor!

• LINQ: one comprehension syntax, unbounded number of (user-definable) traversable datatypes.
let a = for (x of a1)
    for (y of a2)
        if (y > x)
            { x, y };
let i = for (x of map1.keys())
    for (y of map2.keys())
        if (y > x)
            { x, y };
let p = for (x of a1.parallel())
    for (y of a2.parallel())
        if (y > x)
            { x, y };
• The LINQ idea (which is actually the Haskell idea): comprehensions desugar into generic combinators.

• Any datatype that supports those combinators automatically gets to play along.
• Defer comprehensions from ES6.
• Jafar and I will present an ES7 proposal for generalized comprehensions.
Iterator.prototype.
  zip
  filter
  map
  ...


table.keys().map(...)
  .filter(...)

// versus

import { map, filter } from "itertools";
filter(map(table.keys(), ...), ...);
(new Iterator({
    next() { ... }
})).map(...)

5 Jun 14 Resolutions
• Agree to defer comprehensions.

• No future-proofing placeholder objects (can be added later with low compatibility risk).

• May not generalize generator comprehensions since first RHS eagerly evaluated; more work to do.