Map.prototype extensions

map, filter, and more
Design choices

- Directly on `Map.prototype` (see the `spec`)
Design choices

- Directly on `Map.prototype` (see the `spec`)
- On generic “map-like” `%CollectionPrototype%`
Design choices

- Directly on Map.prototype (see the spec)
- On generic “map-like” % CollectionPrototype%
- Binding :: operator and itertools? (not a proposal yet)
User-level API

1. Simple value map:

```javascript
new Map([['x', 10], ['y', 20]])
  .map((v, k, m) => v * 2);
```

=> new map: x: 20, y: 40
(keys are preserved)
User-level API

2. Entries map:

```javascript
new Map([['x', 10], ['y', 20]])
  .mapEntries((v, k, m) => {
    if (k == 'x') return ['z', v];
    return [k, v * 2];
  });
=> new map: z: 10, y: 40
(fully transformed map)
```
User-level API

1. Filter map:

```javascript
new Map([[‘x’, 10], [‘y’, 20]])
  .filter((v, k, m) => v > 10);
```

=> new map: y: 20
User-level API

Note:

The `map.map(...)` and `map.filter(...)` API is the same from user perspective in both case:

- `Map.prototype` storage
- `%CollectionPrototype%`

(storage is an implementation detail)
Protocol design choices

- Internal methods call to generate a new map (see the spec). Works on maps and objects that implement internal map slots.

- Call explicit user-level methods like **set**, and **get** (see this comment). Can be generic for any “map-like” object.
Explicit methods:

Object.defineProperty(%CollectionPrototype%, 'map', {
  value: function (fun) {
    let result = new this.constructor();
    for (let [key, val] of this) { // implies iterable
      result.set(key, val); // calls user-level “set”
    }
    return result;
  },
  configurable: true,
  enumerable: false,
  writable: true
});
import { map } from "itertools";

var newMap = oldMap::map(([k, v]) => [k + 1, v + 1]);

(see this comment)

- Not a proposal yet
- Can it even be considered potentially?
Overall

- To correlate with `map.forEach` better to be `map.map` and `map.filter`, not `map::map`

- Direct `Map.prototype` or `%CollectionPrototype%` - to be discussed (doesn’t affect user-level).