# Collecting and Iterating, the ES6 Way

**By** Jeff Posnick
Web DevRel @ Google

The ECMAScript 6 specification, while still in draft form, brings the promise of many exciting new tools to add to the JavaScript programmer's belt. New classes such as Set and Map offer native solutions to working with specific types of collections, and the for...of statement provides an elegant alternative to traditional ways of iterating over data.

**Set**s offer a way of keeping track of a collection of items in which each item can appear at most once. **Map**s offer more functionality than was previously possible by using **Object**s to associate keys with values—with a **Map**, your keys don't have to be strings, and you don't have to worry about accidentally choosing a key name that clashes with an **Object**'s method names. Lookup operations on native **Map**s and **Set**s are done in constant time, which offers efficiency gains over what's possible with simulated implementations.

The following sample demonstrates constructing a **Set**, and using **for...of** to iterate over its elements:

```
<pre id="log"></pre>

<script>
  function log() {
    document.querySelector('#log').textContent += Array.prototype.join.call(argum
  }

  log('Creating, using, and iterating over a Set:');
  var randomIntegers = new Set();
  // Generate a random integer in the range [1..10] five times,
  // and use a Set to keep track of the distinct integers that were generated.
  for (var i = 0; i < 5; i++) {
    randomIntegers.add(Math.floor(Math.random() * 10) + 1);
  }
  log(randomIntegers.size, ' distinct integers were generated.');
  log('The number 10 was ', randomIntegers.has(10) ? '' : 'not ', 'one of them.')
  log('Here\'s all of them:');
  // Use for...of to iterate over the items in the Set.
  // https://people.mozilla.org/~jorendorff/es6-draft.html#sec-iteration-statemen
  // The Set iterator yields a single value corresponding to each entry in the Se
  for (var item of randomIntegers) {
    log(item);
```

```
  }
</script>
```

Here's a corresponding sample that illustrates using and iterating over a **Map**:

```
<script>
  log('\nCreating and iterating over a Map:');
  // Maps can be initialized by passing in an iterable value (https://people.mozi
  // Here, we use an Array of Arrays to initialize. The first value in each sub-A
  // Map entry's key, and the second is the item's value.
  var typesOfKeys = new Map([
    ['one', 'My key is a string.'],
    ['1', 'My key is also a string'],
    [1, 'My key is a number'],
    [document.querySelector('#log'), 'My key is an object']
  ]);
  // You can also call set() to add new keys/values to an existing Map.
  typesOfKeys.set('!!!!', 'My key is excited!');

  // Use for...of to iterate over the items in the Map.
  // https://people.mozilla.org/~jorendorff/es6-draft.html#sec-iteration-statemen
  // There are several types of Map iterators available.
  // typesOfKeys.keys() can be used to iterate over just the keys:
  log('Just the keys:');
  for (var key of typesOfKeys.keys()) {
    log('  key: ', key);
  }

  // typesOfKeys.values() can be used to iterate over just the values:
  log('Just the values:');
  for (var value of typesOfKeys.values()) {
    log('  value: ', value);
  }

  // The default Map iterator yields an Array with two items; the first is the Ma
  // second is the Map entry's value. This default iterator is equivalent to type
  log('Keys and values:');
  // Alternative, ES6-idiomatic syntax currently supported in Safari & Firefox:
  // for (var [key, value] of typesOfKeys) { … }
  for (var item of typesOfKeys) {
    log('  ', item[0], ' -> ', item[1]);
  }
</script>
```

Some browsers, like Chrome, Internet Explorer, and Firefox have already added support for **Set**s and **Map**s. Native support complemented by polyfill libraries like es6-collections or es6-shim means that JavaScript developers can start building with these new collection types

today. There are no polyfills available for the `for...of` statement (though it is possible to transpile support via Traceur), but native support is available today in Chrome and Firefox.

**Update, September 2014**: Linked to an additional polyfill option, es6-shim

---



*Last updated July 2, 2018.*