

```

var maxFinder = {
  max: 0,
  find: function (numbers) {
    numbers.forEach(
      function (element) {
        if (element > this.max) {
          this.max = element;
        }
      }, this);
  }
};

maxFinder.find([2, 3, 4]);
// log the result
console.log(maxFinder.max);

```

But there is now an even more elegant solution with the arrow function syntax:

```

const maxFinder = {
  max: 0,
  find: function (numbers) {
    numbers.forEach(element => {
      if (element > this.max) {
        this.max = element;
      }
    });
  }
};

maxFinder.find([2, 3, 4]);
// log the result
console.log(maxFinder.max);

```

That makes the arrow functions the perfect candidates for anonymous functions in callbacks!

3.11. Sets and Maps

This is a short one: you now have proper collections in ES6. Yay \o/! We used to have dictionaries filling the role of a map, but we can now use the class `Map`:

```

const cedric = { id: 1, name: 'Cedric' };
const users = new Map();
users.set(cedric.id, cedric); // adds a user
console.log(users.has(cedric.id)); // true
console.log(users.size); // 1
users.delete(cedric.id); // removes the user

```