Iterating Arrays with ForEach

`forEach` provides you with an alternative approach to iterating through items in an array, as opposed to using a `for` loop. With `forEach` there is no need for a counter variable or even a length test as it keeps track of the length of the array, it keeps track of the count automatically. As a result you can create code that's much cleaner and easier to read.

```
var steps = ["brainstorm", "narrow", "prototype", "test", "propose"];
steps.forEach(function(item) {
  console.log(item);
});
// "brainstorm"
// "narrow"
// "prototype"
// "test"
// "propose"
```

Here is a bit more complex example:

```
var steps = ["brainstorm", "narrow", "prototype", "test", "propose"];
steps.forEach(function(item, count) {
  switch(count) {
    case 0:
      console.log("First we " + item);
      break;
    case steps.length - 1:
      console.log("Finally we " + item);
      break;
    default:
      console.log("Then we " + item);
  }
});
// "First we brainstorm"
// "Then we narrow"
// "Then we prototype"
// "Then we test"
// "Finally we propose"
```

As you see the function that we are passing to `forEach` can accept a second optional argument that stores count, the index of a current element. This way we can keep track of what element is currently being processed.

Count With ForEach

You can actually use `forEach` with objects by taking advantage of the `keys` method on the base object. As we mentioned before, this method returns an array. You can use `forEach` on that array to loop through all of the items in an object.

```
var fruits = {};
fruits.apple = {"skin color":"red", "price":0.75};
fruits.orange = {"skin color":"orange", "price":0.65};
fruits.pear = {"skin color":"green", "price":0.95};
Object.keys(fruits).forEach(function(fruit) {
  console.log(fruits[fruit].price);
});
//0.75
//0.65
//0.95
```

'ForEach' Performance

One thing to keep in mind is the performance issues around `forEach`. It runs a little bit slower than `for` loops in the current JavaScript engines (about 50% slower), but I don't consider this a good reason not to use `forEach`. I always recommend coding for clarity and maintainability. That way you can adjust for performance later when the real world situation shows that it's really necessary.