Fork me on GitHub

Parallel map (async.map)

When we want to apply the same function to every item in an array of values, we use `Array.map`

```
[1,2,3,4].map(function square(x) {
    return x * x;
});
```

What if we want to do the same for async transforms? For example, we have an array of item IDs and we want to get the item names.

Note: the items are large so we don't want to keep them entirely in memory.

Callbacks

Callback based functions don't return a value. That means we have to use a special tool again - namely `async.map`

```
async.map(ids, function(id, callback) {
    getFromStorage(id, function (err, res) {
        if (err) return callback(err);
        callback(null, res.name);
    })
}, function(err, results) {
    // results is an array of names
});
```

Promises

Promises are actual values - that means no special tools are required. We can simply use Array.map to get an array of promises for the names, then apply the usual tool on that array - Promise.all.

```
Promise.all(ids.map(function(id) {
    return getItem(id).then(function(result) {
        return result.name;
    });
})).then(function(results) {
    // results is an array of names
})
```

Notes

What if we can't run everything in parallel? Perhaps the operation is very expensive and therefore we would like to fetch the items in series instead of parallel. Then we will need to change our code a bit