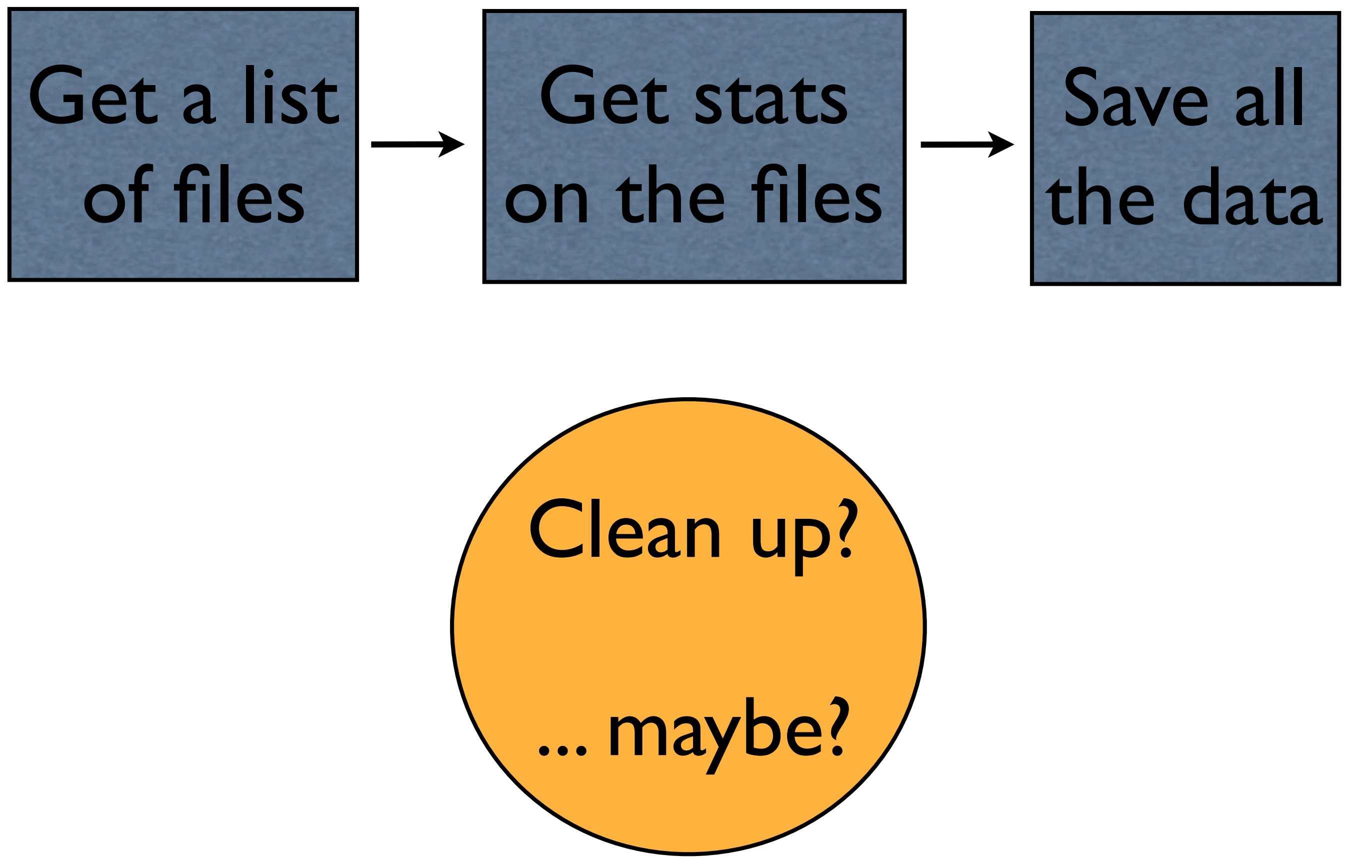
Asynchronicity using `async`

David Manning

Motivation

- Asynchronous design is one of Node.js strengths
- It also presents a significant learning curve for beginners
- ... et al.

Consider this problem



We could do something like this:

```
fs.readdir(path, function (err, files) { // Get the directory listing
  if(err) {
    console.log('wat?: fs.readdir() failed!', err);
  } else {
    files.forEach( function (filename) { // For each file in the directory...
      fs.stat(path + filename, function (err, stats) { // ... get the stats
        if(err) {
          console.log('Oh no!: fs.stat() failed!', err);
        } else {
          db.put(filename, stats, function (err) { // Save the filename and its stats
            (err) ? console.log('Crap!: db:put() failed!', err)
                  : console.log('Logged stats for ' + filename);
          });
        }
      });
    });
  }
});
```

What about closing the database connection?

```
setTimeout(function () { db.close(); }, 100);
setTimeout(function () { db.close(); }, 100);
```

So maybe we can restructure?

```
fs.readdir(path, function (err, files) { // Get the directory listing
  if(err) {
    console.log('wat?: fs.readdir() failed!', err);
  } else {
    var batchRequest = [], numFiles = files.length;
    files.forEach( function (filename) { // For each file in the directory...
      fs.stat(path + filename, function (err, stats) { // ... get the stats
        if(err) {
          console.log('Oh no!: fs.stat() failed!', err);
        } else {  // Add another to the batch request and see if we're done
          batchRequest.push({ type: 'put', key: filename, value: stats });
          if (batchRequest.length == numFiles) db.batch(batchRequest,
            function (err) {
              if (err) console.log('Crap!: db.batch() failed!', err);
              db.close();
            }
          );
        }
      });
    });
  }
});
```

- Better, but we’re really just side-stepping the underlying issue
- What if we needed to index and store the data from multiple directories?
- Which event chain should close the door?
- Race conditions!
- Mixing flow control code with other logic is hard to maintain

What we need is a flow control strategy.

FLOMAX
relief MR
tamsulosin
28 CAPSULES
one a day
Effectively relieves:
frequent night-time urinating
urge to urinate every few hours
weak flow and feeling of unfinished urinating

?

Not that kind....

`async`

- Helps us organize and control asynchronous execution
- Prettifies your pyramids
- Still intelligible to other people
- Your other modules never have to know

async

- The async module has two main classes of functions: those for collections and those for flow control
- those for collections take an array of values and an asynchronous callback
- those for flow control take an array of functions to execute according to prescribed schedule

One way to rewrite with async

```
fs.readdir(path, function (err, files) {
  if (err) {
    console.log('wat?: fs.readdir() failed!', err);
  } else {
    async.each(files,
      function (filename, callback) {
        fs.stat(filename, function (err, stats) {
          if (err) {
            console.log('Oops!: fs.stat() failed!', err);
            callback(err);
          } else {
            console.log('Writing ' + filename + ' : ' + stats);
            db.put(filename, stats, callback);
          }
        });
      },
      function (err) {
        if (err) console.log('An error occurred!', err);
        db.close();
      }
    );
  }
});
```

Another (better) way to rewrite with async

```
async.waterfall([ // don't go chasin' this
  function (callback) {
    fs.readdir(path, callback);
  },
  function (files, callback) {
    async.map(files, fs.stat, function(err, results) {
      var i, batchRequest = [], numFiles = files.length;
      for(i = 0; i < numFiles; i++) {
        batchRequest.push({type: 'put', key: files[i], value: results[i]});
      }
      callback(err, batchRequest);
    });
  },
  function (batchRequest, callback) {
    db.batch(batchRequest, callback);
  }
], function(err) {
  if (err) {
    console.log("Oh no! Something went wrong!".red, err);
  }
  db.close();
});
```

Other patterns

```
async.series([ // the movie was better!
    function(callback){
        // do some stuff ...
        callback(null, 'one');
    },
    function(callback){
        // do some more stuff ...
        callback(null, 'two');
    }
],
// optional callback
function(err, results){
    // results is now equal to ['one', 'two']
});
```

Other patterns

```
async.parallel([ // till we never meet again
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even
though
    // the second function had a shorter timeout.
});
```