```
      var itemResult =
        doThingsWith(arrayItem,
        lastItemResult);
      // results carried along from each
        to the next
      nextCallback(null, itemResult);
}})), function (err, finalResult) {
  // final callback
});
```

Acknowledgements

Hat tip to Caolan McMahon and Paul Miller, whose prior contributions this is based upon. Also Elan Shanker from which this rep is forked

License

MIT

a-sync-waterfall

Simple, isolated sync/async waterfall module for JavaScript.

Runs an array of functions in series, each passing their results to the next in the array. However, if any of the functions pass an error to the callback, the next function is not executed and the main callback is immediately called with the error.

For browsers and node.js.

Installation

- Just include a-sync-waterfall before your scripts.
- `npm install a-sync-waterfall` if you're using node.js.

Usage

- `waterfall(tasks, optionalCallback, forceAsync);`
- **tasks** - An array of functions to run, each function is passed a `callback(err, result1, result2, ...)` it must call on completion. The first argument is an error (which can be null)

and any further arguments will be passed as arguments in order to the next task.

- **optionalCallback** - An optional callback to run once all the functions have completed. This will be passed the results of the last task's callback.
- **forceAsync** An optional flag that force tasks run asynchronously even if they are sync.

Node.js:

```
var waterfall = require('a-sync-
        waterfall');
waterfall(tasks, callback);
```

Browser:

```
var waterfall = require('a-sync-
        waterfall');
waterfall(tasks, callback);

// Default:
window.waterfall(tasks, callback);
```

Tasks as Array of Functions

```
waterfall([
  function(callback){
    callback(null, 'one', 'two');
  },
  function(arg1, arg2, callback){
    callback(null, 'three');
  },
```

```
  function(arg1, callback){
    // arg1 now equals 'three'
    callback(null, 'done');
  }
], function (err, result) {
  // result now equals 'done'
});
```

Derive Tasks from an Array.map

```
/* basic - no arguments */
waterfall(myArray.map(function
        (arrayItem) {
  return function (nextCallback) {
    // same execution for each item,
        call the next one when done
    doAsyncThingsWith(arrayItem,
        nextCallback);
}}));

/* with arguments, initializer function,
        and final callback */
waterfall([function initializer
        (firstMapFunction) {
    firstMapFunction(null,
        initialValue);
  }].concat(myArray.map(function
        (arrayItem) {
    return function (lastItemResult,
        nextCallback) {
      // same execution for each item in
        the array
```