FILE SYSTEM

DISK I/O

- `path` module for resolving paths
- `fs` module for manipulating files
  - Almost direct translation of the UNIX/POSIX APIs
  - Has a streaming abstraction
  - Random file access

PATHS

```
var path = require('path');
```

PATH OPERATIONS

01_path_operations.js

```
// Run these in the repl

path.join('/foo', 'bar', 'baz/asdf', 'quux', '..');
// '/foo/bar/baz/asdf'

path.normalize('/foo/bar//baz/asdf/quux/..');
// '/foo/bar/baz/asdf'

path.resolve('/foo/bar', './baz');
// '/foo/bar/baz'

path.resolve('wwwroot', 'static_files/png/', '../gif/image.gif');
// '/Users/tmpvar/wwwroot/static_files/gif/image.gif'

path.relative('/path/to/test', '/path/to/foo');
// '../foo'
```

PATH COMPONENT OPERATIONS

02_path_component_operations.js

```
path.dirname('/foo/bar/baz/asdf/quux.txt');
// '/foo/bar/baz/asdf'

path.basename('/foo/bar/baz/asdf/quux.txt');
// 'quux.txt'

path.basename('/foo/bar/baz/asdf/quux.txt', '.txt');
// 'quux'

path.extname('/foo/bar/baz/asdf/');
// ''

path.extname('/foo/bar/baz/asdf/quux.txt');
// '.txt'
```

FS MODULE

Provides an API for high level access to the underlying file system

FILE DESCRIPTORS

Provides access to an index on the file descriptor table.

SPECIAL FILE DESCRIPTORS

Available immediately when the process starts.

- 0 Standard input (process.stdin)
- 1 Standard output (process.stdout)
- 2 Standard error (process.stderr)

WRITE TO THE CONSOLE

```
process.stdout.write('hello world\n');
process.stderr.write('error!\n');
```

READ FROM STDIN

03_stdin.js

```
process.stdin.pipe(process.stdout);

process.stdin.on('end', function() {
  process.stdout.write('\ndone\n');
});
```

REGULAR FILE DESCRIPTORS

Handles to files and directories on disk

THE FS MODULE

Provides an API for high level access to the underlying file system.

```
var fs = require('fs');

console.log(Object.keys(fs));
```

QUERY A FILE OR DIRECTORY

04_stat.js

```
var fs = require('fs');
var file = process.argv[2] || __filename;

fs.stat(file, function(err, stats) {
  if (err) throw err;

  console.log(stats, stats.isFile(), stats.isDirectory());
});
```

READING FROM A FILE

05_fs_read.js:

```
var fs = require('fs');
var file = process.argv[2] || __filename

fs.open(file, 'r', function opened(err, fd) {
  if (err) { throw err }

  var buffer = new Buffer(1024),
      length = buffer.length,
      filePosition = parseInt(process.argv[3] || 100, 10);

  fs.read(fd, buffer, 0, length, filePosition, function read(err, bytes)
      if (err) throw err;
      console.log('just read ' + bytes + ' bytes');
      if (bytes > 0) {
        console.log(buffer.slice(0, bytes).toString());
      }
    }
  );
});
```

Random read access

```
$ node 05_fs_read.js 05_fs_read.js 400
$ node 05_fs_read.js 05_fs_read.js 350
```

WRITING TO A FILE

Create/Append to file

06_fs_write.js:

```
var fs = require('fs');
var path = require('path');

fs.open(path.join('support', 'out.txt'), 'a', function opened(err, fd) {
  if (err) throw err;

  var buffer = new Buffer('writing this string');
  var length = buffer.length;

  fs.write(fd, buffer, 0, length, null, function wrote(err, written) {
      if (err) { throw err; }
      console.log('wrote ' + written + ' bytes');
    }
  );
});
```

```
$ node 06_fs_write.js
```

READ FILE PART 2

Read an entire file

07_fs_readfile.js

```
var fs = require('fs');
var file = process.argv[2] || __filename

fs.readFile(file, function(err, bytes) {
  console.log(bytes.toString());
});
```

```
$ node 07_fs_readfile.js
```

WRITE FILE PART 2

Append a buffer to a file

08_fs_writefile.js

```
var fs = require('fs');
var path = require('path');
var file = path.join('support', 'out.txt');
var buffer = new Buffer('writing this string');

fs.writeFile(file, buffer, { flag: 'a' }, function(err) {
  if (err) throw err;

  console.log('wrote ' + buffer.length + ' bytes');
});
```

READ FILE PART 3

Create read stream

09_fs_readstream.js

```
var fs = require('fs');
var file = process.argv[2] || __filename

fs.createReadStream(file).pipe(process.stdout);
```

COPY

Stream a file somewhere else on the filesystem

10_fs_pipe.js

```
var fs = require('fs');
var path = require('path');
var src = process.argv[2] || __filename;
var dest = process.argv[3] || path.join(__dirname, 'support', 'copy.js')

fs.createReadStream(src).pipe(fs.createWriteStream(dest));
```

SUMMARY

Node provides a rich set of tools for working with the file system

- `path`
- `fs`
  - POSIX-like methods
  - Helper methods
  - Streams
- Most `fs` methods have an corresponding `fs.*Sync` method
- See the docs