




MeanJS



Presented by
VAISHALI TAPASWI


FANDS INFONET Pvt.Ltd.
www.fandsindia.com



Ground Rules

- Turn off cell phone. If you cannot please keep it on silent mode. You can go out and attend your call.
- If you have any questions or issues please let me know immediately.
- Let us be punctual.


www.fandsindia.com



LAMP Stack

- LAMP stack: Linux® for the operating system, Apache for the web server, MySQL for the database, and Perl (or Python, or PHP) for the programming language used to generate the HTML-based web pages

www.fandsindia.com



MEAN Stack

- MEAN stack: MongoDB, Express, AngularJS, Node.js.
- The MEAN stack represents a thoroughly modern approach to web development: one in which a single language (JavaScript) runs on every tier of your application, from client to server to persistence.

www.fandsindia.com

From LAMP to MEAN

- The MEAN (MongoDB, Express, AngularJS, Node.js) stack is a modern challenger to the long-popular LAMP stack for building professional websites with open source software. MEAN represents a major shift in architecture and mental models — from relational databases to NoSQL and from server-side Model-View-Controller to client-side, single-page applications.

www.fandsindia.com

What is Node.js?

- Node.js is a headless JavaScript runtime. It is literally the same JavaScript engine (named V8) that runs inside of Google Chrome, except that with Node.js, you can run JavaScript from the command line instead of in your browser.

www.fandsindia.com

What is unique about Node.js?

- JavaScript is used in client-side but node.js puts the JavaScript on server-side thus making communication between client and server happen in same language
- Servers are normally thread based but Node.JS is “Event” based. Node.JS serves each request in a evented loop that can handle simultaneous requests.

www.fandsindia.com

What can you do with Node ?

- It is a command line tool. You download a tarball, compile and install the source.
- It lets you Layered on top of the TCP library is a HTTP and HTTPS client/server.
- The JS executed by the V8 javascript engine (the thing that makes Google Chrome so fast)
- Node provides a JavaScript API to access the network and file system.

www.fandsindia.com

What we can't do with Node?

- ❑ Node is a platform for writing JavaScript applications outside web browsers. This is not the JavaScript we are familiar with in web browsers. There is no DOM built into Node, nor any other browser capability.
- ❑ Node can't run on GUI, but run on terminal

www.fandsindia.com

Threads VS Event-driven

| Threads | Asynchronous Event-driven |
|---|---|
| Lock application / request with listener-workers threads | Only one thread, which repeatedly fetches an event |
| Using incoming-request model | Using queue and then processes it |
| Multithreaded server might block the request which might involve multiple events | Manually saves state and then goes on to process the next event |
| Using context switching | No contention and no context switches |
| Using multithreading environments where listener and workers threads are used frequently to take an incoming-request lock | Using asynchronous I/O facilities (callbacks, not poll/select or O_NONBLOCK) environments |

www.fandsindia.com

Why node.js use event-based?

In a normal process cycle the web server while processing the request will have to wait for the IO operations and thus blocking the next request to be processed.

Node.JS process each request as events, The server doesn't wait for the IO operation to complete while it can handle other request at the same time.

When the IO operation of first request is completed it will call-back the server to complete the request.

www.fandsindia.com

Node.js VS Apache

- ❑ It's fast
- ❑ It can handle tons of concurrent requests
- ❑ It's written in JavaScript (which means you can use the same code server side and client side)
- ❑ Node.js faster than apache but it more hungry system's CPU and memory
- ❑ Node.js use event based programming, it make the server doesn't wait for the IO operation to complete while it can handle other request at the same time

www.fandsindia.com

Node Usage

- How do I use node's REPL?
 - REPL- 'Read-Eval-Print Loop'
 - Node
 - Write commands directly
- Execute Scripts
 - Node scriptfilename

www.fandsindia.com

Simple Node Code

- Write code in test.js

```
console.log(process.versions) ;
console.log(process.arch) ;
console.log(process.platform);
```
- Test Code

```
node test.js
```

www.fandsindia.com

Simple Http Server

```
var http = require('http');
http.createServer(function(request,response)
{
    response.writeHead(200);
    response.write("hello, this is HelloWorld
from Node");
    response.end();
}).listen(8080);
```

www.fandsindia.com

Server Node Code

```
var http = require('http');
var port = 9090;
var server = http.createServer(responseHandler);
server. listen(port);
console.log('Server running at
http://127.0.0.1:' + port + '/');
function responseHandler(req, res)
{
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('<html><body><h1>Hello
World</h1></body></html>');
}
```

www.fandsindia.com

Event Emitters

```
var server = require("http").createServer()
server.on("request", function(request, response) {
  console.log("> REQUEST STARTED")
  request.on("end", function() {
    console.log("> REQUEST CLOSED")
    response.writeHead(200, {"Content-Type": "plain/text"})
    response.end("Hello World\n")
    server.close() })
  response.on("close", function() {
    console.log("> RESPONSE CLOSED") }) })
server.on("close", function() {
  console.log("> SERVER CLOSED") })
server.on("listening", function() {
  console.log("> SERVER STARTED") })
server.listen(8080)
```

www.fandsindia.com

Custom Logger

```
var mylogger = exports;
mylogger.debugLevel = 'warn';
mylogger.log = function(level,
  message) {
  var levels = ['error', 'warn', 'info'];
  if (levels.indexOf(level) <=
    levels.indexOf(mylogger.debug
      Level)) {
    if (typeof message !== 'string') {
      message =
        JSON.stringify(message);
    };
    console.log(level+'
      '+message);
  }
}

var logger1 = mylogger;
logger1.debugLevel =
  'warn';
logger1.log('info',
  'Everything started
    properly.');
```

```
logger1.log('warn', 'Running
  out of memory...');
logger1.log('error', { error:
  'flagrant' });
```

www.fandsindia.com

How to access query string parameters

- ❑ var fs = require('fs');
- ❑ var http = require('http');
- ❑ var url = require('url');
- ❑ http.createServer(function (req, res) {
- ❑ var queryObject = url.parse(req.url, true).query;
- ❑ console.log(queryObject);
- ❑ res.writeHead(200);
- ❑ res.end('Feel free to add query parameters to the end of the url' + queryObject);
- ❑ }).listen(8080);

www.fandsindia.com

How to debug a node application

- ❑ Setup
 - npm install node-inspector –g
- ❑ Enable debugging
 - node --debug app.js
 - should print something along the lines of debugger listening on port 5858 to stderr.
 - node-inspector [--web-port=<custom port number>]

www.fandsindia.com

How do I read files in node.js?

```
fs = require('fs')
fs.readFile('yrdy.txt', 'utf8',
  function (err,data) {
    if (err) {
      return console.log(err);
    }
    console.log(data);
  });

var data =
  fs.readFileSync(""),
```

www.fandsindia.com

How do I search files and directories?

List files in current folder

```
fs = require('fs');
fs.readdir(process.cwd(),
  function (err, files) {
    if (err) {
      console.log(err);
      return;
    }
    console.log(files);
  });
```

Search Folder

```
npm install findit
var finder =
  =require('findit').find(__dirname);
finder.on('directory', function
  (dir) {
    console.log('Directory: ' + dir +
      '');
  });
//This listens for files found
finder.on('file', function (file) {
  console.log('File: ' + file);
});
```

www.fandsindia.com

ExpressJS

www.fandsindia.com

Express

- "A minimal and flexible node.js web application framework"
- It helps you build web apps
- Like any abstraction, Express hides difficult bits and says "don't worry, you don't need to understand this part".

www.fandsindia.com



Bottom layer: Node's HTTP server

```
// Require what we need
var http = require("http");
// Build the server
var app = http.createServer(function(request, response) {
  response.writeHead(200, {
    "Content-Type": "text/plain" });
  response.end("Hello world!\n");
});
// Start server
app.listen(1337, "localhost");
console.log("Server running at http://localhost:1337/");
```

www.fandsindia.com



Bottom layer: Node's HTTP server

```
var app = http.createServer(function(request, response) {
  // Build the answer
  var answer = "";
  answer += "Request URL: " + request.url + "\n";
  answer += "Request type: " + request.method + "\n";
  answer += "Request headers: " +
    JSON.stringify(request.headers) + "\n";

  // Send answer
  response.writeHead(200, { "Content-Type": "text/plain" });
  response.end(answer);
});
```

www.fandsindia.com



Page Navigation

```
// Homepage
if (req.url == "/") {
  res.end("Welcome to the homepage!");
}
// About page
else if (req.url == "/about") {
  res.end("Welcome to the about page!");
}
else {
  res.end("404 error! File not found.");
}
```

www.fandsindia.com



Middleware, the middle layer

□ Starting with express

```
// Require the stuff we need
var express = require("express");
var http = require("http");
// Build the app
var app = express();
// Add some middleware
app.use(function(request, response) {
  response.end("Hello world!\n");
});
http.createServer(app).listen(1337);
```

www.fandsindia.com

Top layer: routing

Routing is a way to map different requests to specific handlers

```
var express = require("express");
var http = require("http");
var app = express();
app.all("/*", function(request, response, next) {
  response.writeHead(200, { "Content-Type": "text/plain" });
  next(); });
app.get("/", function(request, response) {
  response.end("Welcome to the homepage!"); });
app.post("/about", function(request, response) {
  response.end("Welcome to the about page!"); });
app.get("/*", function(request, response) {
  response.end("404!"); });
http.createServer(app).listen(1337);
```

www.fandsindia.com

Simple Parameters (REST)

```
app.get("/hello/:who", function(req, res) {
  res.end("Hello, " + req.params.who +
    ".");
  // Fun fact: this has security issues
});
```

www.fandsindia.com

Get Request Parameters

- To pass parameters
 - `http://localhost:8080/about?username=test&password=W124`
- To access parameters
 - `var user_id = request.query.username;`
-

www.fandsindia.com

Post Request Parameters

- Pass as form variables
- To use
 - `var bodyParser = require("body-parser");`
 - Npm install
 - `app.use(bodyParser.urlencoded({ extended: false }));`
 - `var user_id = request.body.id;`

www.fandsindia.com

Request handling

- Express augments the request and response objects that you're passed in every request handler.
- In addition to sending text
 - `response.redirect("/hello/anime");`
 - `response.redirect("http://www.myanimelist.net");`
 - `response.sendFile("/path/to/anime.mp4");`

www.fandsindia.com

Views

- `// Start Express`
- `var express = require("express");`
- `var app = express();`
- `// Set the view directory to /views`
- `app.set("views", __dirname + "/views");`
- `// Let's use the Jade templating language`
- `app.set("view engine", "jade");`

www.fandsindia.com

Templating Languages

- Jade
- EJS
- HAML
- HandleBars
- Mustache

www.fandsindia.com

Database Integration - SQL

- Adding database connectivity capability to Express apps is just a matter of loading an appropriate Node.js driver for the database in your app.
- E.g. `npm install mysql`
- ```
var mysql = require('mysql');
var connection = mysql.createConnection({
 host: 'localhost', user: 'dbuser', password: 's3krete7' });
connection.query("select ...");
connection.connect();
```

[www.fandsindia.com](http://www.fandsindia.com)

## Database Integration - MondoDB

### □ Modules

- Mongoskin
  - the promise wrapper for node-mongodb-native.
- Mongooes
  - MongoDB object modeling designed to work in an asynchronous environment

[www.fandsindia.com](http://www.fandsindia.com)

## Mongoose

Elegant MongoDB object modeling for Node.js

- `var mongoose = require('mongoose');`
- `mongoose.connect('mongodb://localhost/test');`
- `var Cat = mongoose.model('Cat', { name: String });`
- `var kitty = new Cat({ name: 'Zildjian' });`
- `kitty.save(function (err) {`
- `if (err) // ...`
- `console.log('meow');`
- `});`

[www.fandsindia.com](http://www.fandsindia.com)

## MongoDB

[www.fandsindia.com](http://www.fandsindia.com)

## MongoDB

MongoDB stores data in the form of documents, which are JSON-like field and value pairs. Documents are analogous to structures in programming languages that associate keys with values (e.g. dictionaries, hashes, maps, and associative arrays). Formally, MongoDB documents are BSON documents. BSON is a binary representation of JSON with additional type information. In the documents, the value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents.

[www.fandsindia.com](http://www.fandsindia.com)

## Starting with MongoDB

- Install
- Set up the MongoDB environment
  - Run server
    - `mongod --dbpath /dbpath/`
  - Run Client
    - Mongo
  - On mongo
    - use admin (where admin is name of database)

[www.fandsindia.com](http://www.fandsindia.com)

## Insert

The following operation inserts a new document into the users collection. The new document has four fields name, age, and status, and an `_id` field. MongoDB always adds the `_id` field to the new document if that field does not exist.

```
db.users.insert (
 {
 name: "sue",
 age: 26,
 status: "A"
 }
)
```

← collection  
← field: value  
← field: value  
← field: value  
} document

The following diagram shows the same query in SQL:

```
INSERT INTO users
(name, age, status)
VALUES ("sue", 26, "A");
```

← table  
← columns  
← values/row

## Update

In MongoDB, the `db.collection.update()` method modifies existing documents in a collection. The `db.collection.update()` method can accept query criteria to determine which documents to update as well as an options document that affects its behavior, such as the multi option to update multiple documents.

```
db.users.update(
 { age: { $gt: 18 } },
 { $set: { status: "A" } },
 { multi: true }
)
```

← collection  
← update criteria  
← update action  
← update option

The following diagram shows the same query in SQL:

```
UPDATE users
SET status = 'A'
WHERE age > 18
```

← table  
← update action  
← update criteria

## Remove

`db.collection.remove()` method deletes documents from a collection. The `db.collection.remove()` method accepts a query criteria to determine which documents to remove

```
db.users.remove(
 { status: "D" }
)
```

← collection  
← remove criteria

The following diagram shows the same query in SQL:

```
DELETE FROM users
WHERE status = 'D'
```

← table  
← delete criteria

[www.fandsindia.com](http://www.fandsindia.com)

## Find

Read operations, or queries, retrieve data stored in the database. In MongoDB, queries select documents from a single collection.

```
db.users.find(
 { age: { $gt: 18 } },
 { name: 1, address: 1 }
).limit(5)
```

← collection  
← query criteria  
← projection  
← cursor modifier

The next diagram shows the same query in SQL:

```
SELECT _id, name, address
FROM users
WHERE age > 18
LIMIT 5
```

← projection  
← table  
← select criteria  
← cursor modifier

[www.fandsindia.com](http://www.fandsindia.com)

## Query Behavior

### □ MongoDB queries exhibit the following behavior:

- All queries in MongoDB address a single collection.
- You can modify the query to impose limits, skips, and sort orders.
- The order of documents returned by a query is not defined unless you specify a sort().
- Operations that modify existing documents (i.e. updates) use the same query syntax as queries to select documents to update.
- In aggregation pipeline, the \$match pipeline stage provides access to MongoDB queries.
- MongoDB provides a `db.collection.findOne()` method as a special case of `find()` that returns a single document.

## QUESTION / ANSWERS



[www.fandsindia.com](http://www.fandsindia.com)

## THANKING YOU !



[www.fandsindia.com](http://www.fandsindia.com)