Node.js

Objetivos

Apresentar a plataforma Node.

Criação de servidores com Node.

Módulo Express.

Exemplos de aplicações com Node.

O que é Node.js

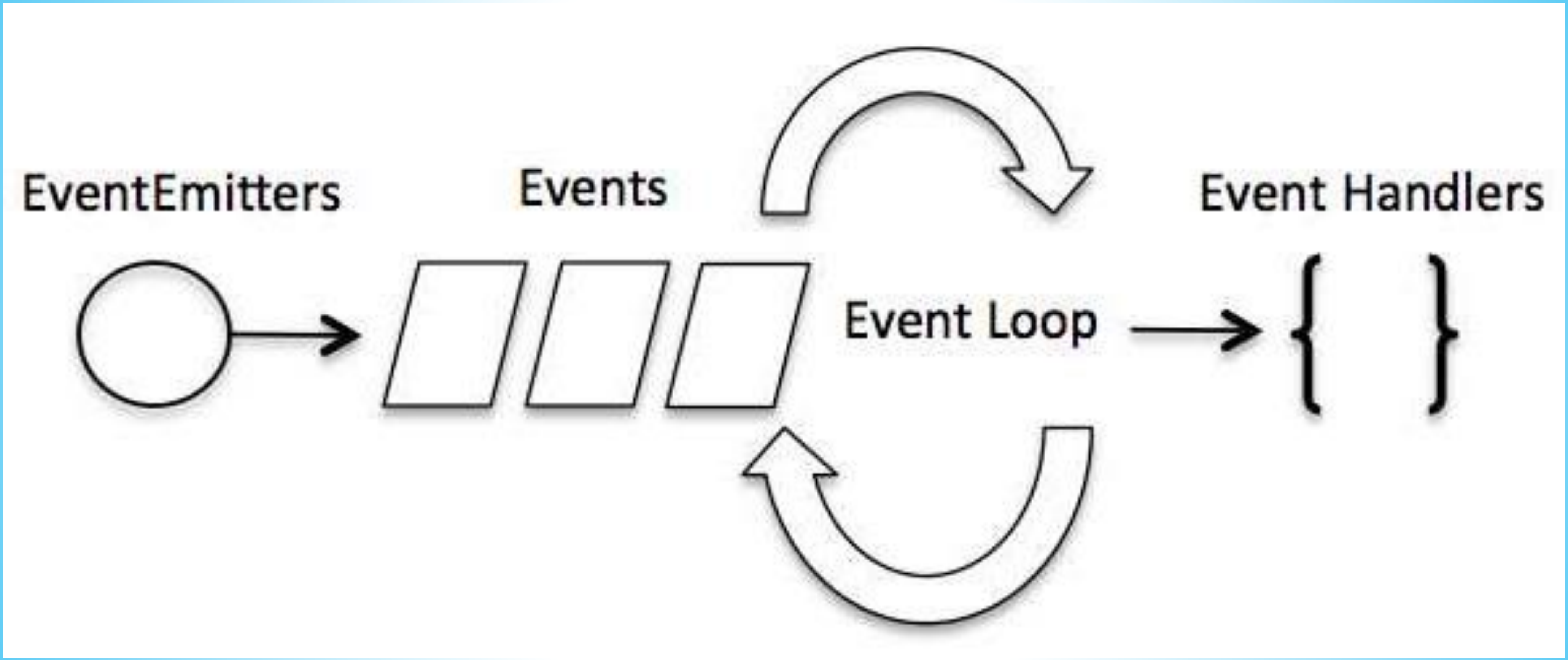
É uma plataforma que permite a execução de programas Javascript de forma assíncrona e dirigida a Eventos.

Não depende de um browser.

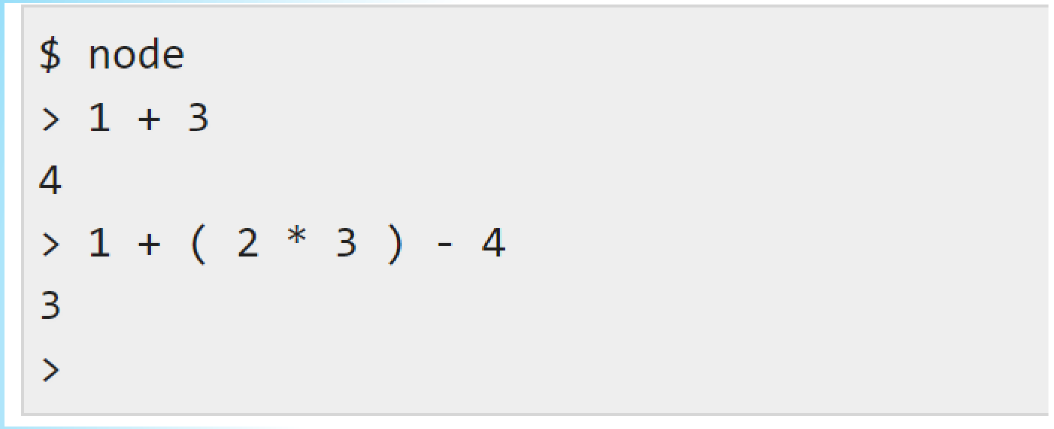
Permite a construção de aplicações para a Internet.

Excelente para atendimento assíncrono single thread.

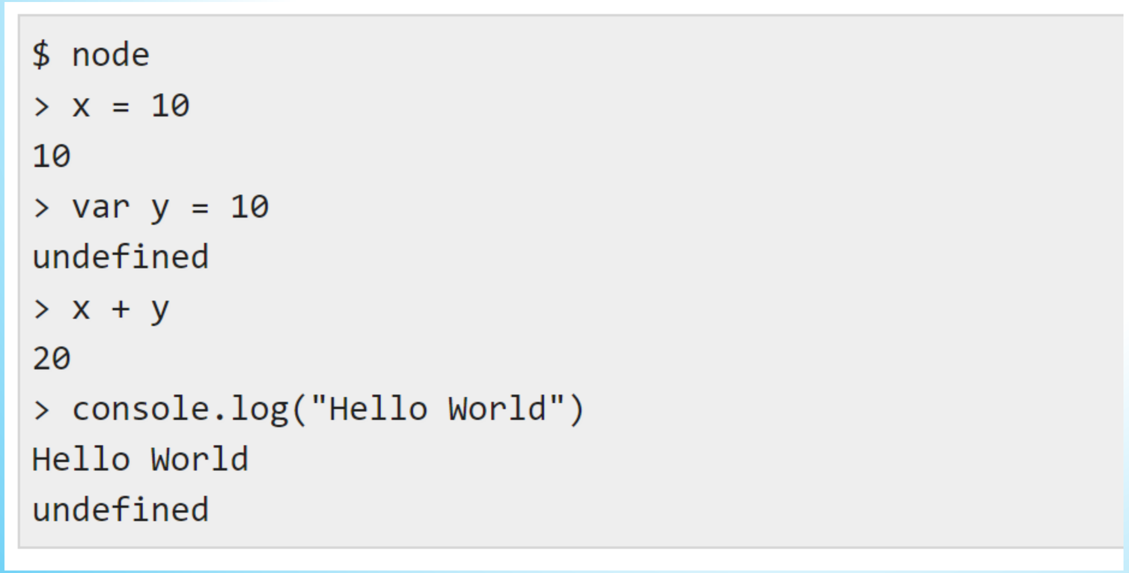
Não recomendável para aplicações CPU intensive.


EventEmitters
Events
Event Loop
Event Handlers
{ }

REPL – Read Eval Print Loop



```
$ node
> 1 + 3
4
> 1 + ( 2 * 3 ) - 4
3
>
```



```
$ node
> x = 10
10
> var y = 10
undefined
> x + y
20
> console.log("Hello World")
Hello World
undefined
```

Histórico

Criado por Ryan Dahl em 2009 em cima da Engine Javascript V8 do Google.

Características

Código aberto e gratuito.
Tem excelente desempenho.
Feito em Javascript.
Roda aplicações Javascript.
Extensível.

Primeiro servidor Http

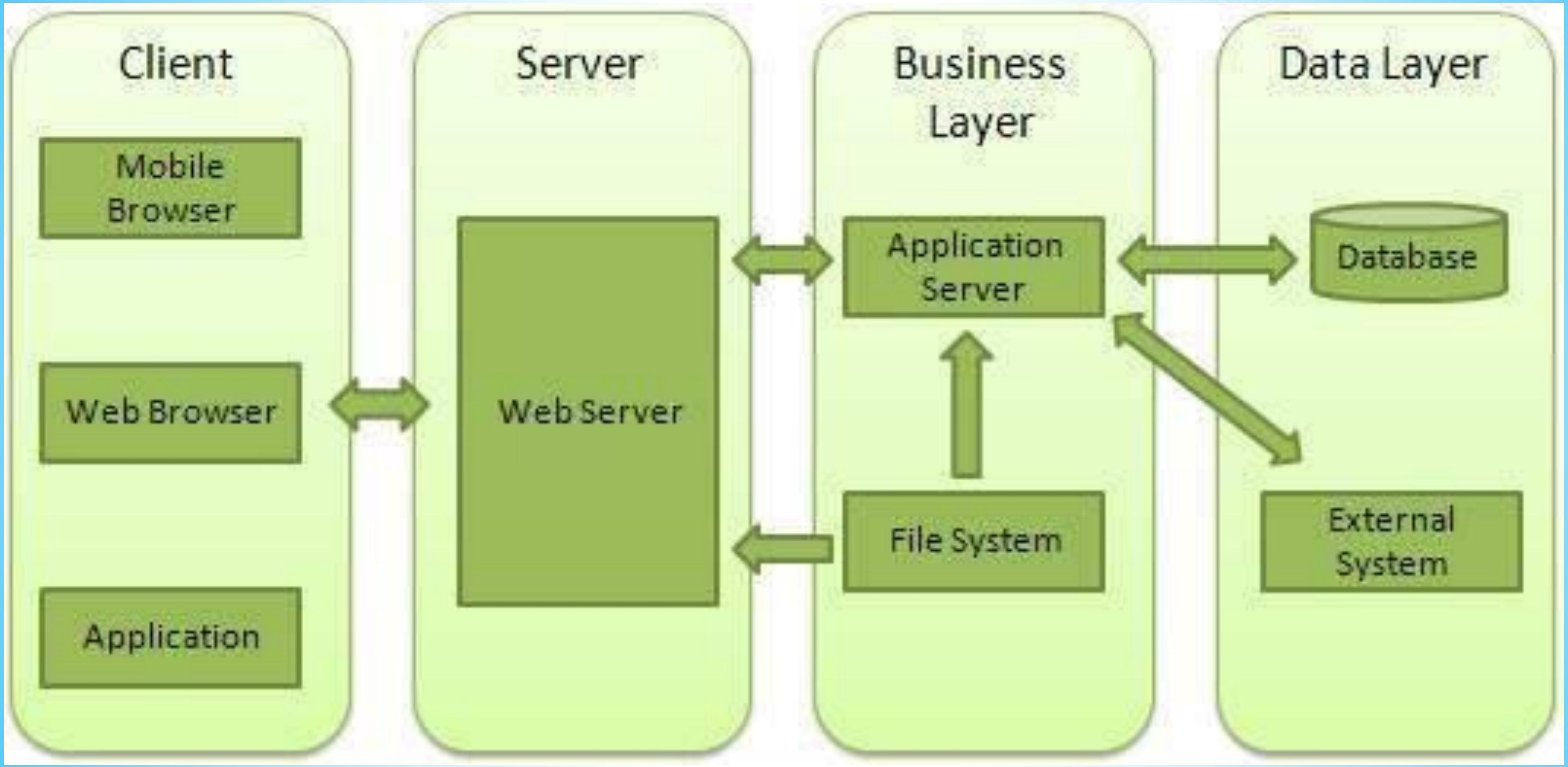
```
var http = require("http");  // carregar módulo http
http.createServer(
 function(request, response) {     // funcao anonima
 console.log(“request received”);
 response.writeHead(200, {"Content-Type": "text/plain"});
 response.write("Hello World");
 response.end();
  }).listen(8888);
console.log(“Server has started”);
```

....

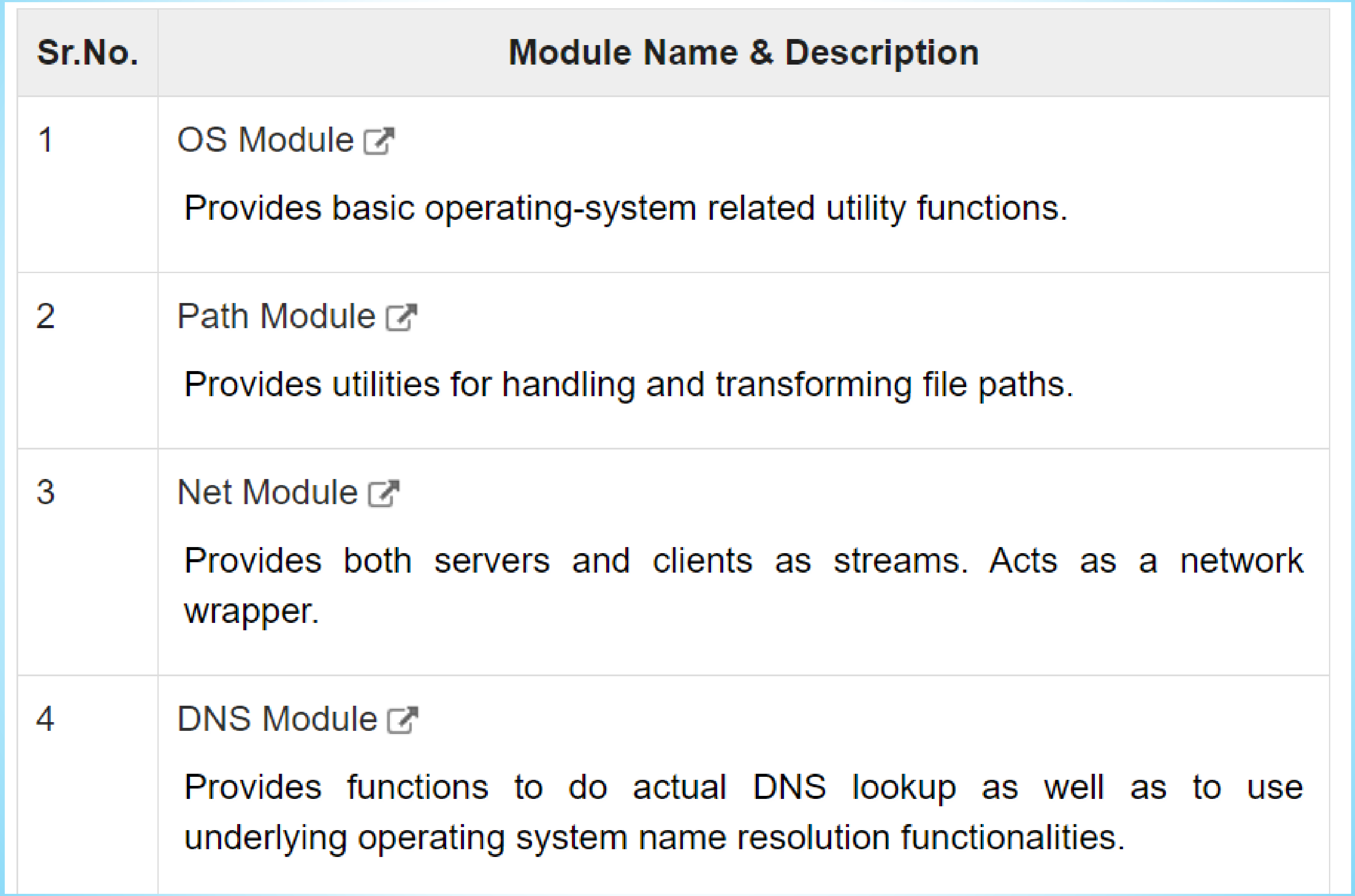
```
node  server.js
```

....

http://localhost:8888/


Client
Mobile Browser
Web Browser
Application
Server
Web Server
Business Layer
Application Server
File System
Data Layer
Database
External System

Módulos



Sr.No.	Module Name & Description
1	OS Module Provides basic operating-system related utility functions.
2	Path Module Provides utilities for handling and transforming file paths.
3	Net Module Provides both servers and clients as streams. Acts as a network wrapper.
4	DNS Module Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.

```
var http = require('http');
var fs = require('fs');
var url = require('url');
// Create a server
http.createServer( function (request, response) {
// Parse the request containing file name
 var pathname = url.parse(request.url).pathname;
// Print the name of the file for which request is made.
console.log("Request for " + pathname + " received.");
// Read the requested file content from file system
fs.readFile(pathname.substr(1), function (err, data) {
        if (err) { console.log(err);
        //HTTP Status: 404 : NOT FOUND
         // Content Type: text/plain
        response.writeHead(404, {'Content-Type': 'text/html'}); }
        else{
        //Page found
        // HTTP Status: 200 : OK
        // Content Type: text/plain
        response.writeHead(200, {'Content-Type': 'text/html'});
        // Write the content of the file to response body
        response.write(data.toString()); }
        // Send the response body
        response.end(); })
; }).listen(8081);
// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

```
var http = require('http');
var fs = require('fs');
var url = require('url');
// Create a server
http.createServer( function (request, response) {
// Parse the request containing file name
 var pathname = url.parse(request.url).pathname;
// Print the name of the file for which request is made.
console.log("Request for " + pathname + " received.");
// Read the requested file content from file system
fs.readFile(pathname.substr(1), function (err, data) {
        if (err) { console.log(err);
        //HTTP Status: 404 : NOT FOUND
         // Content Type: text/plain
        response.writeHead(404, {'Content-Type': 'text/html'}); }
        else{
        //Page found
        // HTTP Status: 200 : OK
        // Content Type: text/plain
        response.writeHead(200, {'Content-Type': 'text/html'});
        // Write the content of the file to response body
        response.write(data.toString()); }
        // Send the response body
        response.end(); })
; }).listen(8081);
// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Arquivo index.htm

```
<html>
<head>
 <title>Sample Page</title>
 </head>
<body> Hello World! </body>
</html>
```

Framework Express

O framework Express provê funcionalidades para a criação rápida de aplicações web.

Instalação:

```
$ npm install express --save
```

Npm é um instalador de pacotes (módulos) node.

Cont.

Instalar também:

- Body-parser
- Cookie-parser
- Multer

```
$ npm install body-parser –save
$ npm install cookie-parser –save
$ npm install multer --save
```

Hello World

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {  // req, res – request and response objects
                  res.send('Hello World'); })

var server = app.listen(8081, function () {
var host = server.address().address
var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)
})
```

Roteamento Básico

```
var express = require('express');
var app = express();
// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})
// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})
```

```
// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})
```

```
// This responds a GET request for abcd, abxcd, ab123cd, and so on


app.get('/ab*cd', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)

})
```

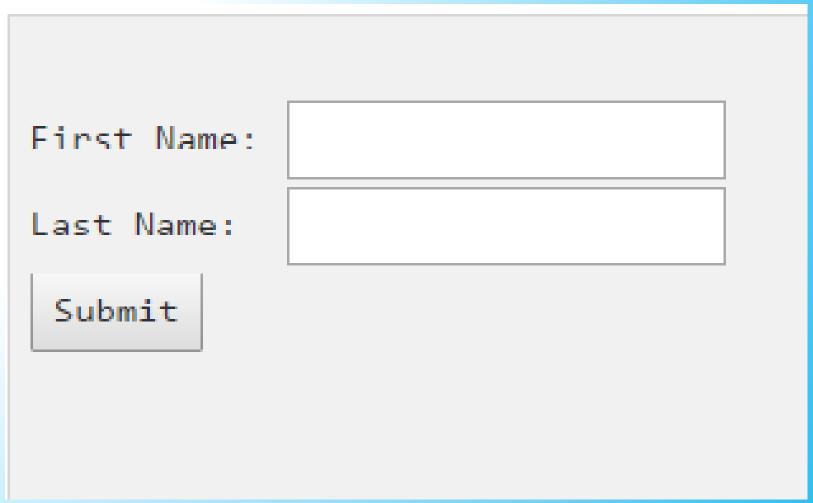
Prover arquivos estáticos

```
var express = require('express');
var app = express();
app.use(express.static('public'));  // prover arquivos no diretorio public
app.get('/', function (req, res) {
   res.send('Hello World');
})
var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

Método GET

Arquivo Index.htm

```
<html>
<body>
<form action="http://127.0.0.1:8081/process_get" method="GET">
First Name: <input type="text" name="first_name">  <br>

Last Name: <input type="text" name="last_name">
<input type="submit" value="Submit">
</form>
</body>
</html>
```



```
var express = require('express');
var app = express();

app.use(express.static('public'));
app.get('/index.htm', function (req, res) {
   res.sendFile( __dirname + "/" + "index.htm" );
})

app.get('/process_get', function (req, res) {
   // Prepare output in JSON format
   response = {
      first_name: req.query.first_name,
      last_name: req.query.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

Método POST

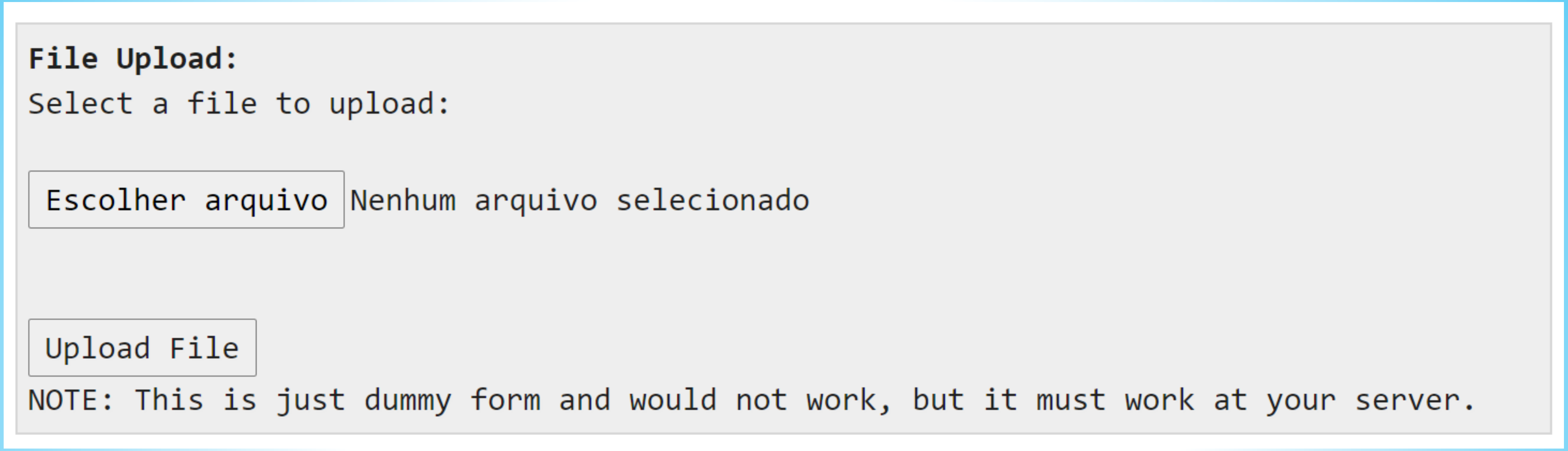
Arquivo index.htm

```
<html>
<body>
<form action="http://127.0.0.1:8081/process_post" method="POST">
First Name: <input type="text" name="first_name">  <br>
Last Name: <input type="text" name="last_name">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
// Create application/x-www-form-urlencoded parser
var urlencodedParser = bodyParser.urlencoded({ extended: false })
app.use(express.static('public'));
app.get('/index.htm', function (req, res) {
   res.sendFile( __dirname + "/" + "index.htm" );
})

app.post('/process_post', urlencodedParser, function (req, res) {
   // Prepare output in JSON format
   response = {
       first_name:req.body.first_name,
       last_name:req.body.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
})
var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

File Upload

```html
<html>
   <head>
      <title>File Uploading Form</title>
   </head>

   <body>
      <h3>File Upload:</h3>
      Select a file to upload: <br />

      <form action = "http://127.0.0.1:8081/file_upload" method = "POST"
         enctype = "multipart/form-data">
         <input type="file" name="file" size="50" />
         <br />
         <input type = "submit" value = "Upload File" />
      </form>

   </body>
</html>
```


File Upload:

Select a file to upload:

NOTE: This is just dummy form and would not work, but it must work at your server.

```
// server.js
const express = require('express')
        , app = express()
        , path = require('path')
        , multer = require('multer');

// cria uma instância do middleware configurada
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, 'uploads/')
    },
    filename: function (req, file, cb) {
        cb(null, file.originalname);
    }
});

const upload = multer({ storage });
```

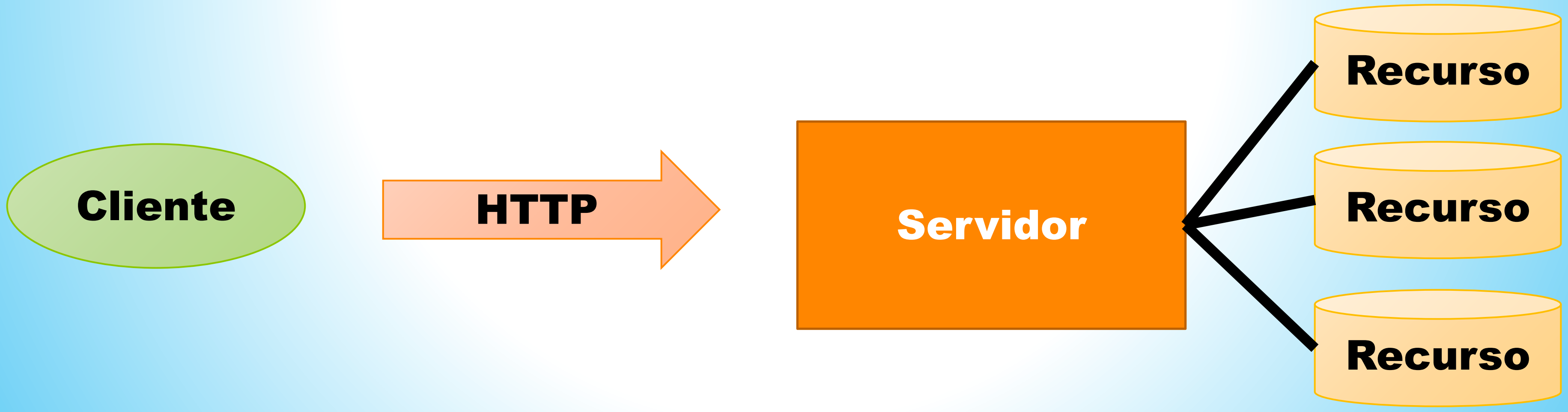
```
app.use(express.static('public'));

app.get('/index.htm', function (req, res) {
   res.sendFile( __dirname + "/" + "index.htm" );
})


// rota indicado no atributo action do formulário
app.post('/file_upload', upload.single('file'),
    (req, res) => res.send('<h2>Upload realizado com sucesso</h2>'));

app.listen(8081, () => console.log('App na porta 8081'));
```

Cangaceiro JavaScript

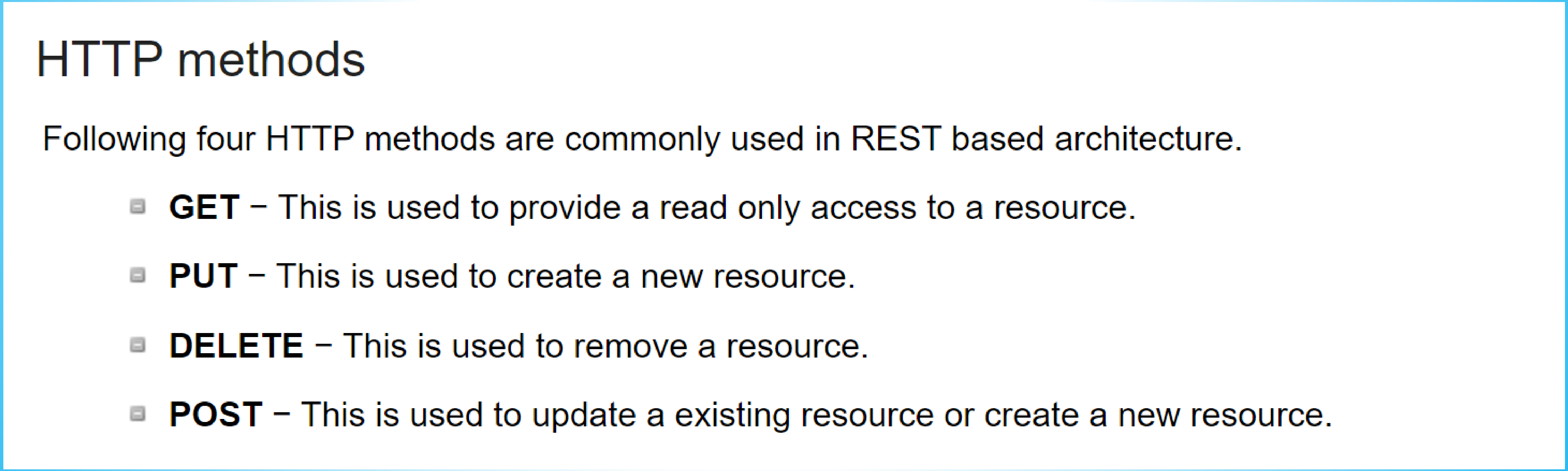
http://cangaceirojavascript.com.br/express-realizando-upload-multer/

Arquitetura REST

Conceito criado por Roy Fielding em 2000.
A idéia central é um modelo para criar aplicações distribuídas apenas usando o conceito de recursos e comandos HTTP.



Representational State Transfer (REST)



HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

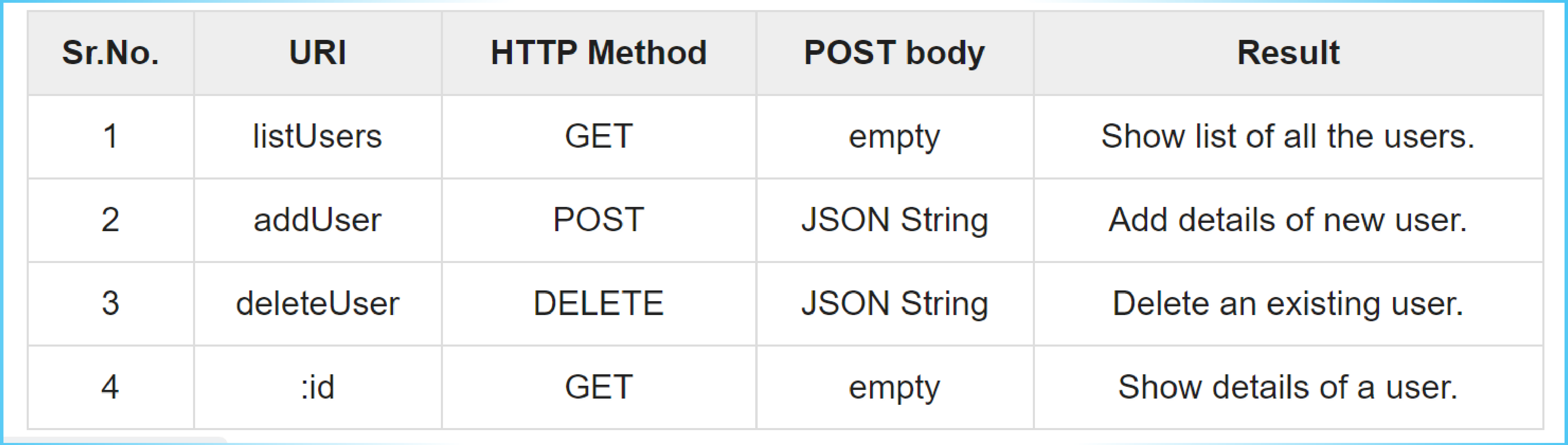
- **GET** – This is used to provide a read only access to a resource.
- **PUT** – This is used to create a new resource.
- **DELETE** – This is used to remove a resource.
- **POST** – This is used to update a existing resource or create a new resource.

Restful Web Service

Web service é uma coleção de protocolos e padrões para troca de dados e comandos entre sistemas.

Restful quando totalmente implementado com HTTP.

```
{
   "user1" : {
      "name" : "mahesh",
      "password" : "password1",
      "profession" : "teacher",
      "id": 1
   },

   "user2" : {
      "name" : "suresh",
      "password" : "password2",
      "profession" : "librarian",
      "id": 2
   },

   "user3" : {
      "name" : "ramesh",
      "password" : "password3",
      "profession" : "clerk",
      "id": 3
   }
}
```



Sr.No.	URI	HTTP Method	POST body	Result
1	listUsers	GET	empty	Show list of all the users.
2	addUser	POST	JSON String	Add details of new user.
3	deleteUser	DELETE	JSON String	Delete an existing user.
4	:id	GET	empty	Show details of a user.

List Users

Let's implement our first RESTful API **listUsers** using the following code in a server.js file −

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/listUsers', function (req, res) {
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      console.log( data );
      res.end( data );
   });
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port
   console.log("Example app listening at http://%s:%s", host, port)
})
```

Add User

server.js

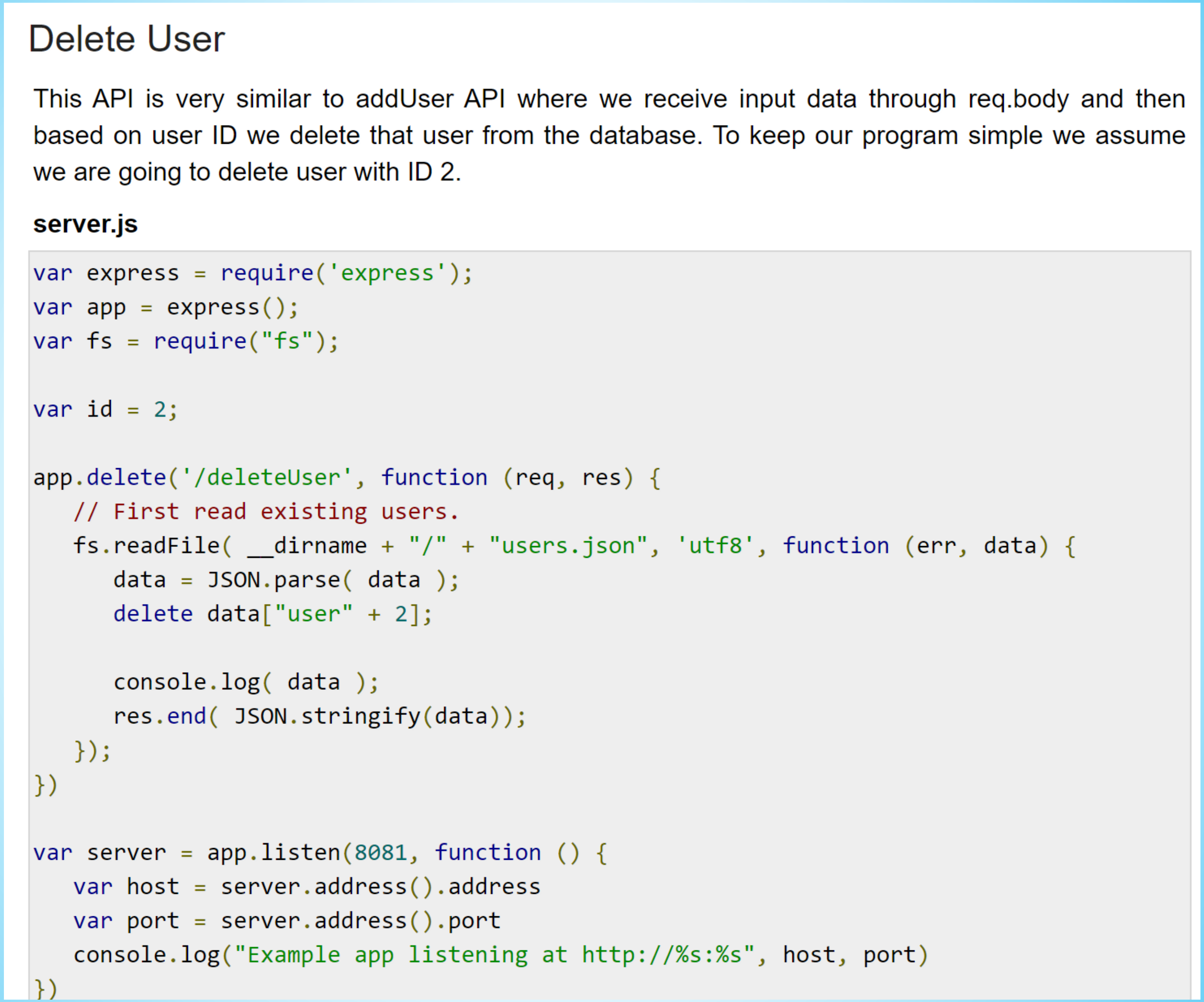
```
var express = require('express');
var app = express();
var fs = require("fs");

var user = {
   "user4" : {
      "name" : "mohit",
      "password" : "password4",
      "profession" : "teacher",
      "id": 4
   }
}

app.post('/addUser', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      data = JSON.parse( data );
      data["user4"] = user["user4"];
      console.log( data );
      res.end( JSON.stringify(data));
   });
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port
   console.log("Example app listening at http://%s:%s", host, port)
})
```

Show Detail

Now we will implement an API which will be called using user ID and it will display the detail of the corresponding user.

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/:id', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      var users = JSON.parse( data );
      var user = users["user" + req.params.id]
      console.log( user );
      res.end( JSON.stringify(user));
   });
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port
   console.log("Example app listening at http://%s:%s", host, port)
})
```



Delete User

This API is very similar to addUser API where we receive input data through req.body and then based on user ID we delete that user from the database. To keep our program simple we assume we are going to delete user with ID 2.

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

var id = 2;

app.delete('/deleteUser', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      data = JSON.parse( data );
      delete data["user" + 2];

      console.log( data );
      res.end( JSON.stringify(data));
   });
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port
   console.log("Example app listening at http://%s:%s", host, port)
})
```

Mais sobre Node

http://www.tutorialspoint.com/nodejs/index.htm

https://www.geeksforgeeks.org/http-cookies-in-node-js/

https://www.guru99.com/node-js-express.html

http://cangaceirojavascript.com.br/