Node JS

- **NodeJS** és un entorn d'execució basat en **JavaScript.**
- Utilitza el motor de JavaScript de Chrome versió 8.
- Utilitza un esquema **orientat** totalment a **event**.
- **Cada aplicació s'executa en un únic thread.**
- Aquests punts fa que sigui un entorn d'execució molt

  lleuger, **ràpid** i **molt escalable**.
- Les aplicacions que corren en aquest entorn son aplicacions implementades

  en javaScript + HTML + llibreries de templates per generar HTML +CSS.
- https://nodejs.org/en/

Exemple d'aplicació en NodeJS: **ZipArchiver**

Aspectes de **JavaScript** que treballarem:

- Modules
- Objects
- Anonymous functions
- Callback functions

Exemple d'aplicació en NodeJS: **ZipArchiver**

Aspectes de **NodeJS** que treballarem:

- **NodeJS** Modules
- External Modules
- **package.json**: For configuring a NodeJS project.
- **NPM**: **N**ode **P**ackage **M**anager
  For managing NodeJS modules.

ZipArchiver: Configuració entorn d'execució.

1. Instal.lar nodeJS. Veure la guia d'instal.lació de materials.
2. Instal.lar npm. Veure la guia d'instal.lació anterior.
3. Crear el directori zipArchiver
4. Crear el fitxer: **package.json**

npm init

package.json

```
{
  "name": "zipArchiver",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node app.js"
  }
}
```

Versió de tots els mòduls externs a NodeJS que utilitza l'aplicació.

ZipArchiver: Configuració entorn d'execució.

5. Instal.lar el mòdul **archiver**
Mòdul amb la funcionalitat per manipul.lar **zip files**.

npm install archiver --save
cat package.json

```
{
  "name": "zipArchiver",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "archiver": "^1.3.0"
  }
}
```

Versió de tots els mòduls externs a NodeJS que utilitza l'aplicació.

ls -la
node_modules package.json

ls -la node_modules

```
r$ ls -la node_modules/
total 320
drwxrwxr-x 38 mc mc  4096 may  6 22:01 .
drwxrwxr-x  3 mc mc  4096 may  7 00:49 ..
drwxrwxr-x  3 mc mc  4096 may  6 22:01 archiver
drwxrwxr-x  2 mc mc  4096 may  6 22:01 archiver-utils
drwxrwxr-x  4 mc mc  4096 may  6 22:01 async
drwxrwxr-x  2 mc mc  4096 may  6 22:01 balanced-match
drwxrwxr-x  3 mc mc  4096 may  6 22:01 bl
```

ZipArchiver: Configuració entorn d'execució.

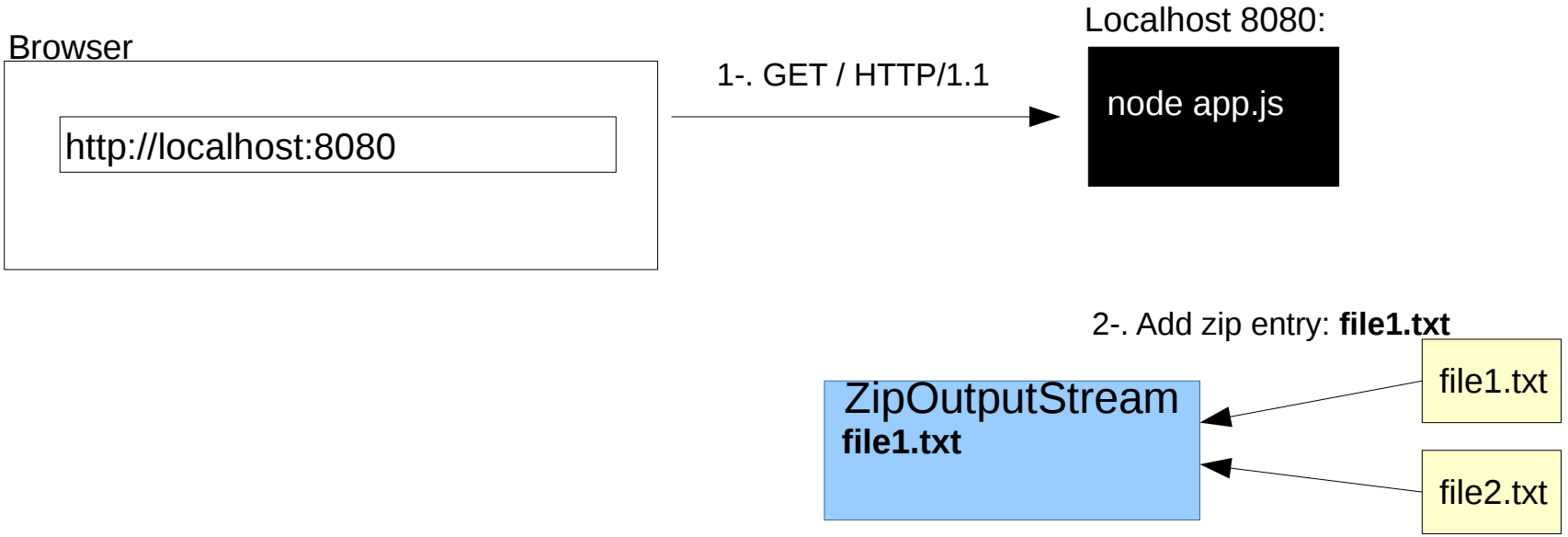
6. Apis que utilitzarem:
- NodeJS: https://nodejs.org/api/modules.html
- Archiver: https://archiverjs.com/docs/

7. Creem el fitxer de la nostra aplicació **app.js**

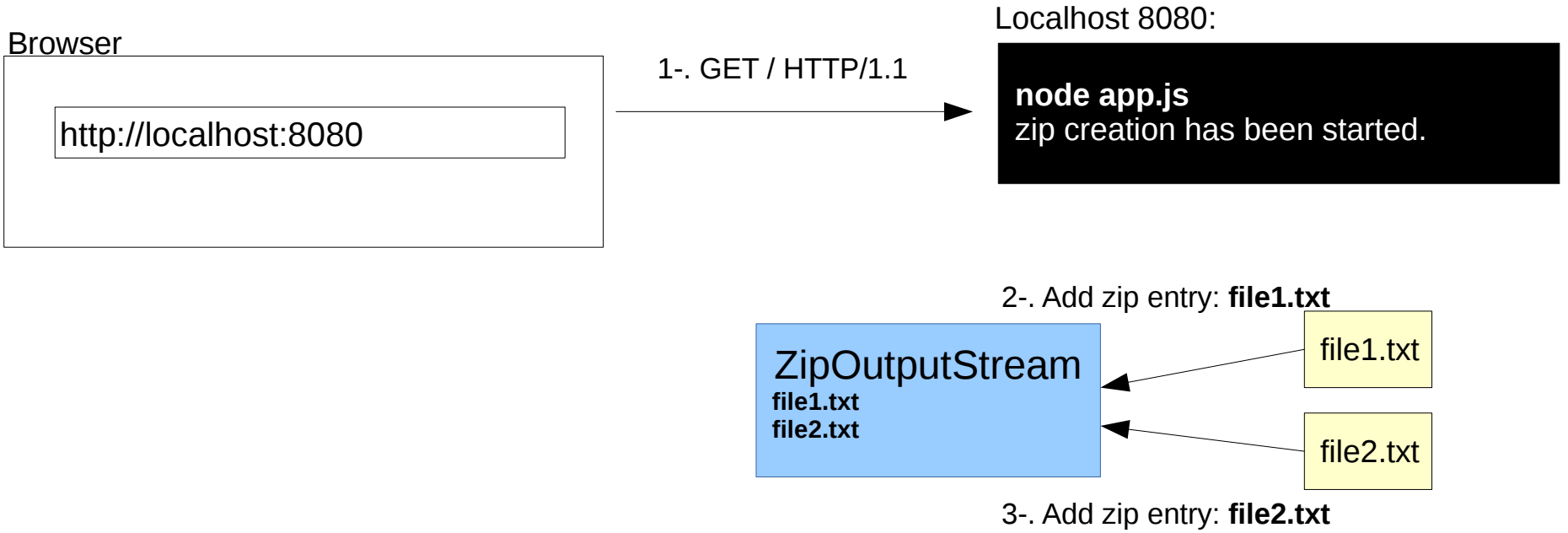
package.json

```
{
  "name": "zipArchiver",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "archiver": "^1.3.0"
  }
}
```

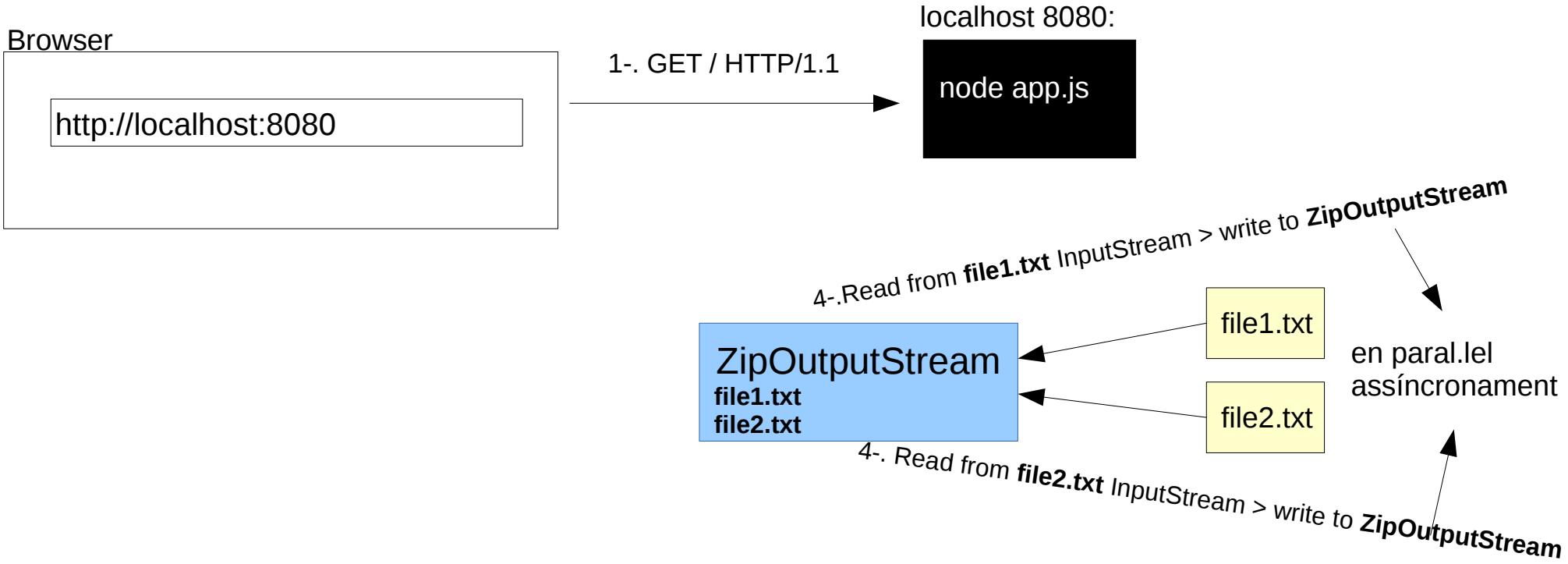
ZipArchiver: Què ha de fer?



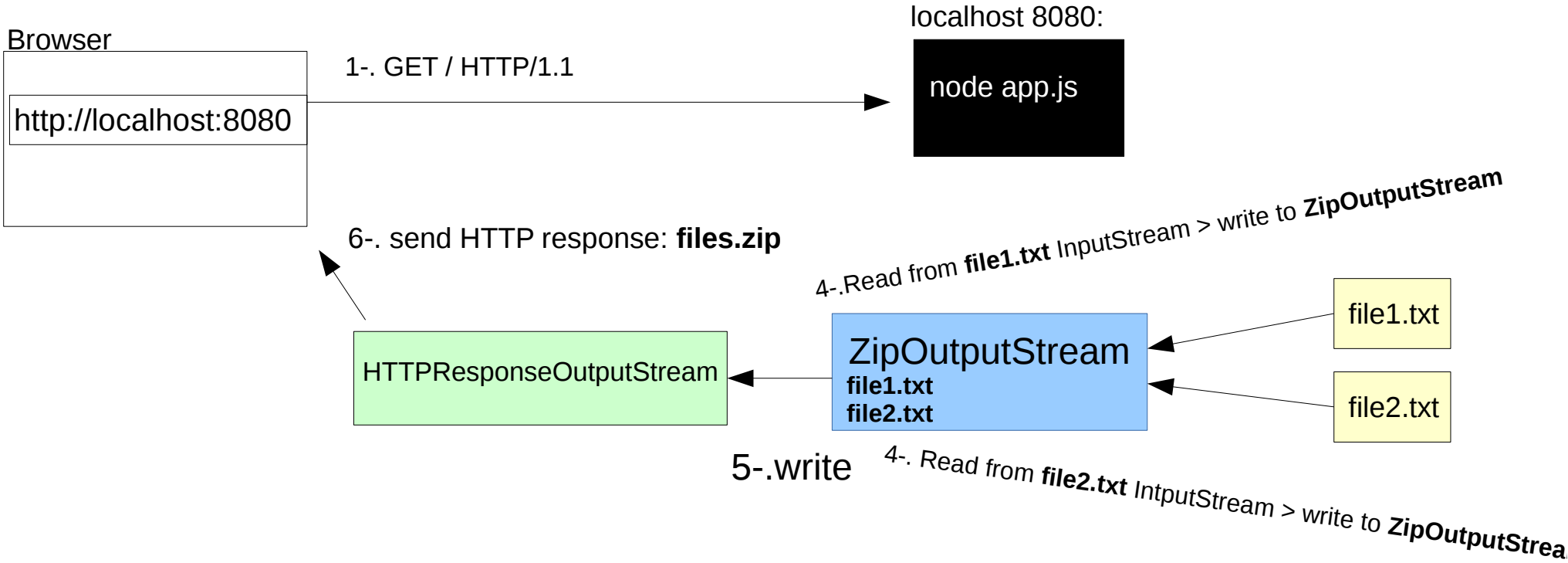
ZipArchiver: Què ha de fer?



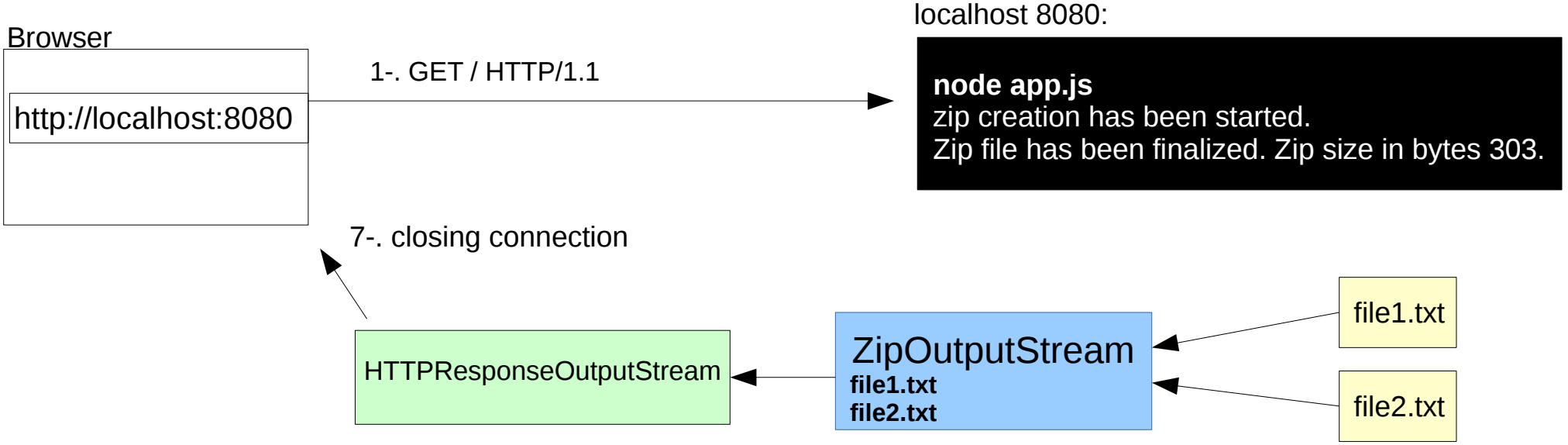
ZipArchiver: Què ha de fer?



ZipArchiver: Què ha de fer?



ZipArchiver: Què ha de fer?



ZipArchiver: app.js

app.js

```
var http = require('http'),
    archiver = require('archiver'),
    fs = require('fs');
```

Carreguem el mòdul **http** de la Api de NodeJS

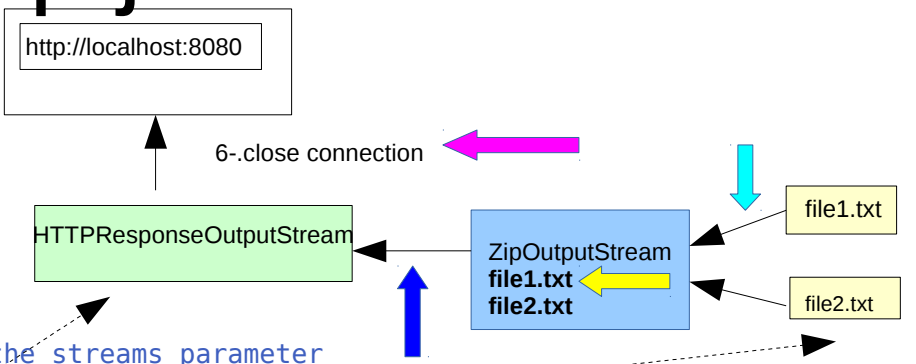
Carreguem el mòdul **archiver**, extern al NodeJS

Carreguem el mòdul **File System** de la Api de NodeJS

ZipArchiver: app.js

app.js

```
var http = require('http'), //http module
    archiver = require('archiver'), //archiver module
    fs = require('fs'); //file system module

/**
 * Function that pipes (sends) all the InputStreams in the streams parameter
 * to a ZipOutputStream connected to the HTTP Response Stream.
 * @streams Array with the InputStreams to be zipped.
 * @response OutputStream connected to the HTTP Response.
 */
function sendFilesInAZip( streams , response ){

  var zipOs = archiver('zip');
  //when compression finishes we execute the anonymous func. passed as a
  //parameter.
  zipOS.on('finish', function(){
    console.log('Zip file has been finalized. Zip  size in bytes: ' + zipOS.pointer());
    //closing the OutputStream connected to the HTTP Response.
    response.end();
  });

  //piping (connecting) the zip to the HTTP response output stream.
  zipOS.pipe(response);

  //for each InputStream to be zipped we add an entry into the
  //zip archive associating an entry name to the inputStream.
  streams.forEach(function(inputStream){
    //appending an entry in the archive from a FileInputStream
    zipOS.append(inputStream.stream, {name: inputStream.name});
  });

  // finalize the archive (we are done appending files but streams have to finish yet
  // to be compressed)
  zipOS.finalize();

  console.log('Zip creation has been started.'); } //end sendFilesInAZip
```



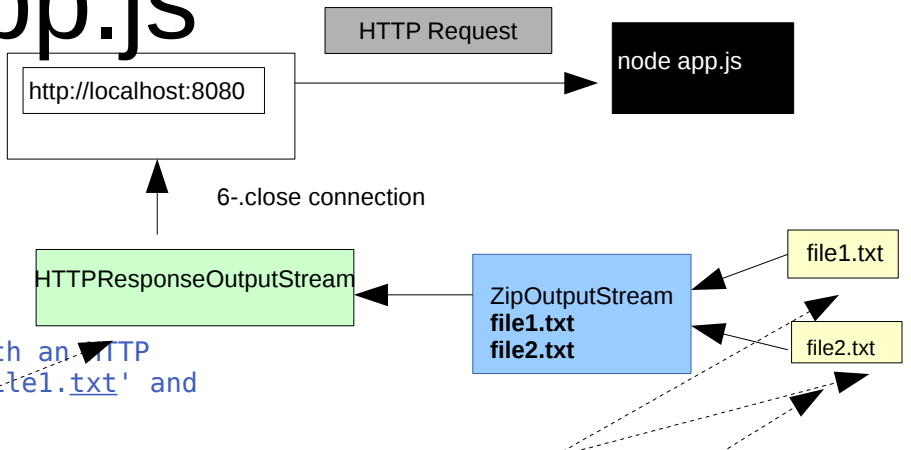
node app.js
zip creation has been started.
Zip file has been finalized. Zip size in bytes 303.

Tot i haver acabat **sendFilesInAZip**, quan finalitzi la compressió es cridarà a aquesta **callback function** que tancarà el HTTPResponseOutputStream.

En arribar al final de la funció pot ser encara no s'han acabat de comprimir els fitxers!!!!!!!!!!!!

ZipArchiver: app.js

app.js




```
/**
 * Function that when receives an HTTP request replies with an HTTP
 * response sending compressed in zip format the files 'file1.txt' and
 * 'file2.txt'
 */
function serverFunctionality( request , response ){
  //sending HTTP Response headers
  response.writeHead(200, {'Content-Type': 'application/zip',
    'Content-Disposition' : 'attachment; filename=files.zip' });

  //specifying the file names to be zipped and sent
  var names = ['file1.txt','file2.txt'];


  var streams = names.map(function(fileName){
    var inputStream = {
      name: fileName,

    stream : fs.createReadStream(fileName)
    };

    return inputStream;

  }); ➡ streams =[ Inpustream , Inpustream ]

  //sending all the inputStreams in the streams array to the ZipOutputStream
  //connected to the HTTP Response.
  SendFilesInAZip( streams , response );
} //end ServerFuncionality
```

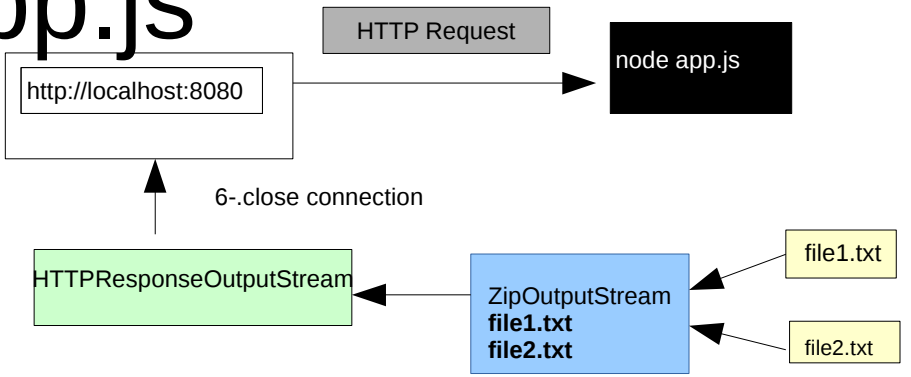
ZipArchiver: app.js

app.js

```
http.createServer(serverFunctionality).listen(8080);
console.log('Server running at http://localhost:8080/');
```

node app.js
Server running at http://localhost:8080
zip creation has been started.

Zip file has been finalized. Zip size in bytes 303.



Express

- Minimalist web framework for NodeJS
- Framework per desenvolupar aplicacions web seguint el patró **MVC** usant **NodeJS**.
- Api per desenvolupar aplicacions web fàcilment amb NodeJS.
- http://expressjs.com/
- Express utilitza llibreries basades en templates per a generar les **vistes** HTML a partir de les dades que calculem en el **model**.
  - La llibreria basada en templates que utilitzarem per a que express generi les vistes serà: **EJS** (**E**mbedded **J**ava**S**cript)
  - http://www.embeddedjs.com/getting_started.html

Express: Instal.lació

sudo apt-get install node-express

#instal.lar l'express generator a on tenim instal.lat el nodeJS
npm install express-generator -g

#crear des de zero el directori zipArchiverMVC amb l'express generator
express --view=ejs zipArchiverMVC

Directori de l'aplicació

Llibreria de templates que volem usar per generar les views: **EJS**

Express: Instal.lació

sudo apt-get install node-express

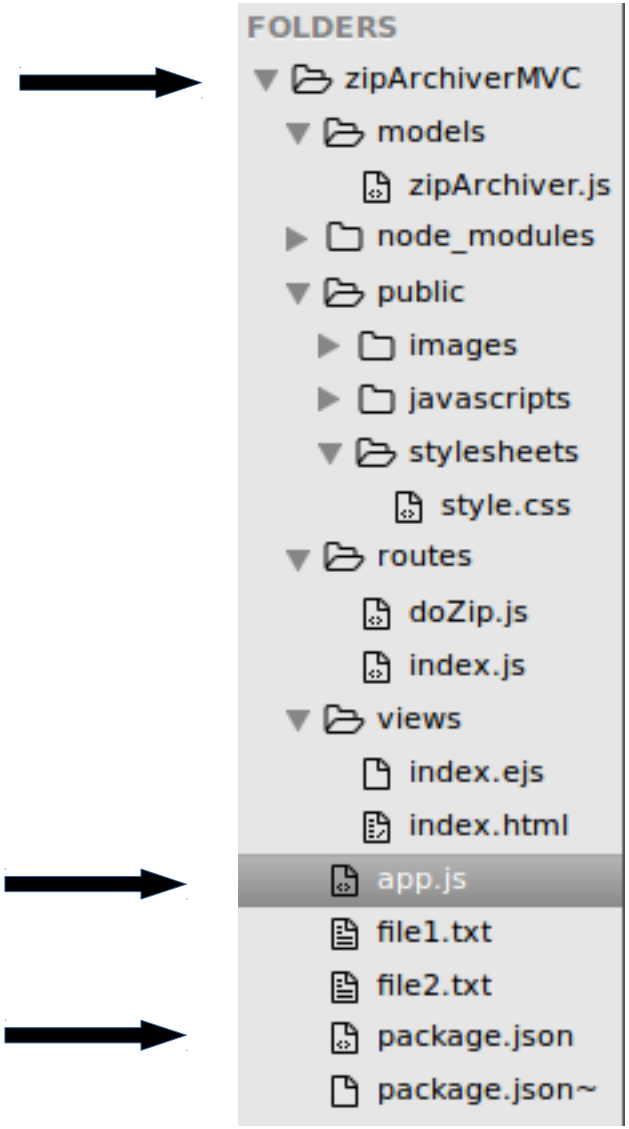
#instal.lar l'express generator a on tenim instal.lat el nodeJS
npm install express-generator -g

#crear des de zero el directori zipArchiverMVC
express --view=ejs zipArchiverMVC

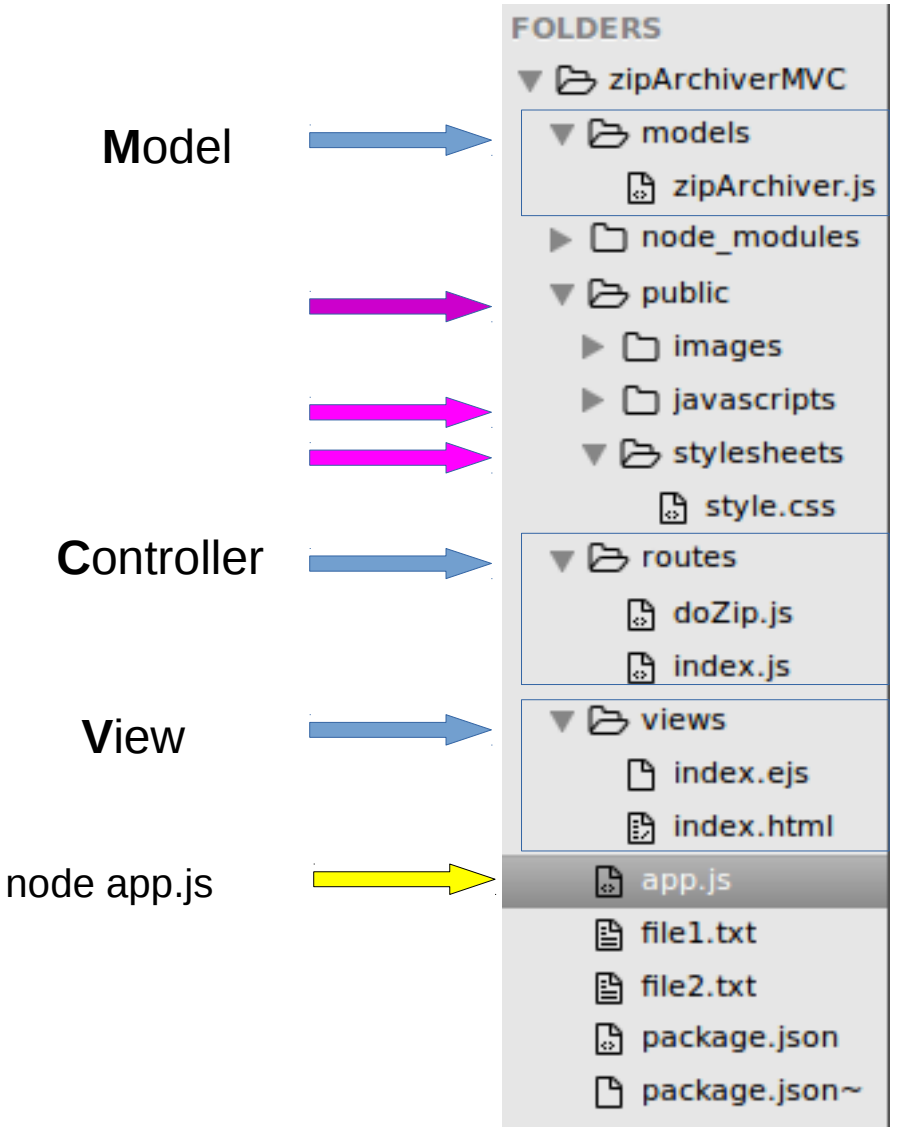
cd zipArchiverMVC
#per instal.lar els paquets del express.
npm install

```
$ ls node_modules/
connect  ejs  express  formidable  mime  mkdirp  qs
```

#instal.lar en el directori zipArchiverMVC el modul del archiver
npm install archiver --save



Express: Estructura de directoris



ZipArchiverMVC: app.js

app.js

```
/**
 * Module dependencies.
 */

var express = require('express'),
    routes = require('./routes');

var app = module.exports = express.createServer();

// Configuration

app.configure(function(){
  app.set('views', __dirname + '/views');
  app.set('view engine', 'jade');
  app.set("view options", { layout: false });
  app.use(express.bodyParser());
  app.use(express.methodOverride());
  app.use(app.router);
  app.use(express.static(__dirname + '/public'));
});

app.configure('development', function(){
  app.use(express.errorHandler({ dumpExceptions: true, showStack: true }));
});

app.configure('production', function(){
  app.use(express.errorHandler());
});

// Routes

app.get('/', routes.index);

app.listen(3000, function(){
  console.log("Express server listening on port %d in %s mode",
  app.address().port, app.settings.env);
});
```

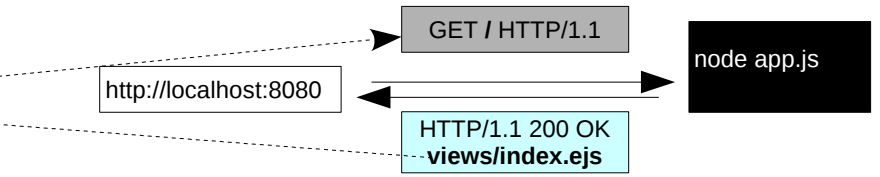
En la variable **routes** tenim el directori **routes**.

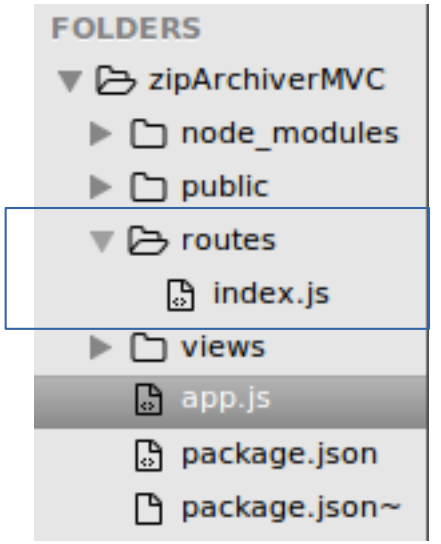
Creem servidor HTTP

Especifiquem el directori de les vistes

Canviar 'jade' per 'ejs'

Afegir aquesta línia de codi: com no usem 'jade' no ens calen layouts.





2-. Controller per a generar la view: index.js

routes/index.js

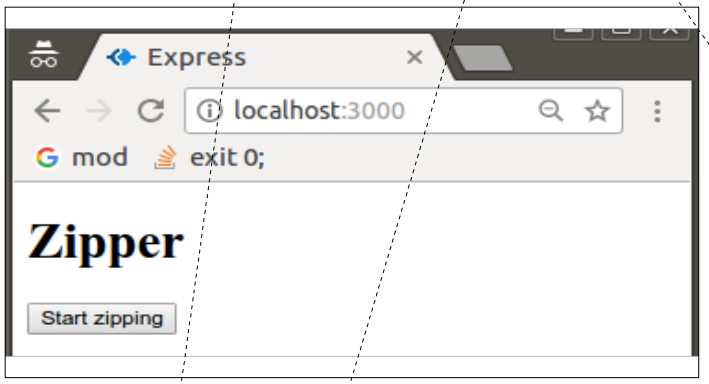
```
/*
 * GET home page.
 */

exports.index = function(req, res){
  res.render('index.ejs', { title: 'Express' })
};
```

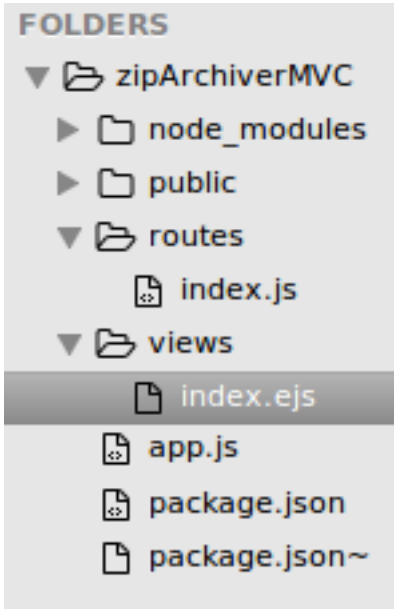
HTTP Request object

HTTP Response object

JSon

Això renderitza el template index.ejs





1-. app.js

```
app.get('/', routes.index);
```

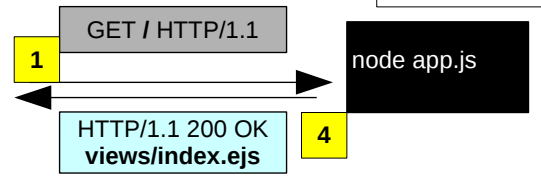
3-View

views/index.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title><%=title%></title>
    <meta charset="UTF-8">
  </head>
  <body>
   <h1>Zipper</h1>
    <form method="get" action="doZip">
      <input type="submit"
value="Start zipping"/>
    </form> </body> /html>
```

4-. NodeJS renderitza

```
<!DOCTYPE html>
<html>
  <head>
    <title>Express</title>
    <meta charset="UTF-8">
  </head>
  <body>
      <form>...</form>
  </body>
</html>
```

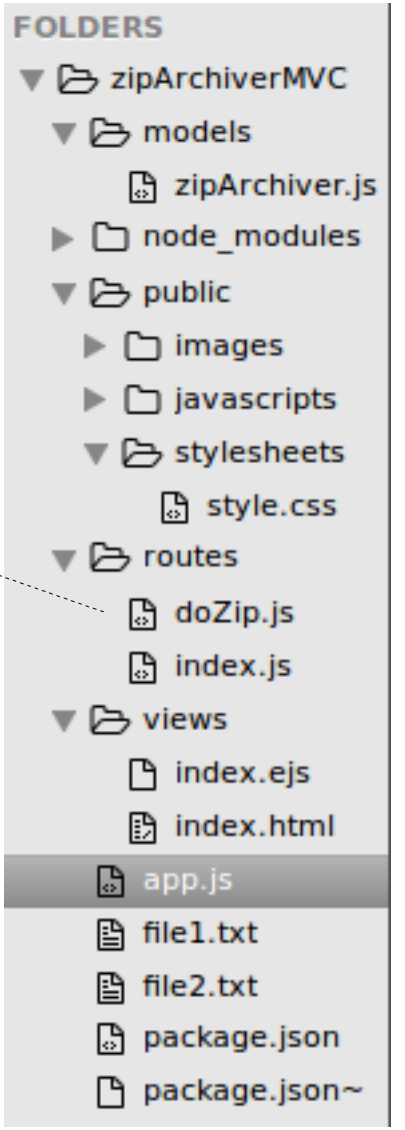


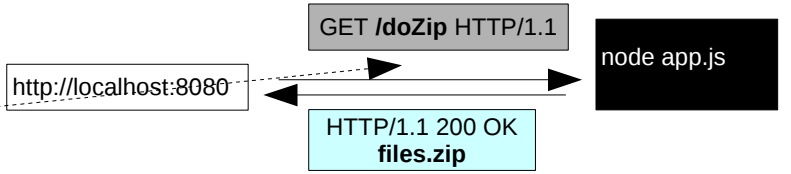
ZipArchiverMVC: app.js

app.js

```
/**
 * Module dependencies.
 */

var express = require('express'),
    routes = require('./routes'),
    zipCtrl = require('./routes/doZip');

var app = module.exports = express.createServer();

// Configuration

app.configure(function(){
  app.set('views', __dirname + '/views');
  app.set('view engine', 'ejs');
  app.set("view options", { layout: false });
  app.use(express.bodyParser());
  app.use(express.methodOverride());
  app.use(app.router);
  app.use(express.static(__dirname + '/public'));
});

app.configure('development', function(){
  app.use(express.errorHandler({ dumpExceptions: true, showStack: true }));
});

app.configure('production', function(){
  app.use(express.errorHandler());
});

// Routes

app.get('/', routes.index);
app.get('/doZip', zipCtrl.doZip);

app.listen(3000, function(){
  console.log("Express server listening on port %d in %s mode", app.address().port, app.settings.env);
});
```

En la variable **zipCtrl** tenim les funcions que exporta el fitxer **doZip.js**





2-. Controller per a la resposta HTTP que envia: file.zip

routes/doZip.js

```
exports.doZip =function(req, res) {
  console.log('routes/doZip');
  var zipArchiver = require("../models/zipArchiver");

  zipArchiver.serverFunctionality(res);
};
```

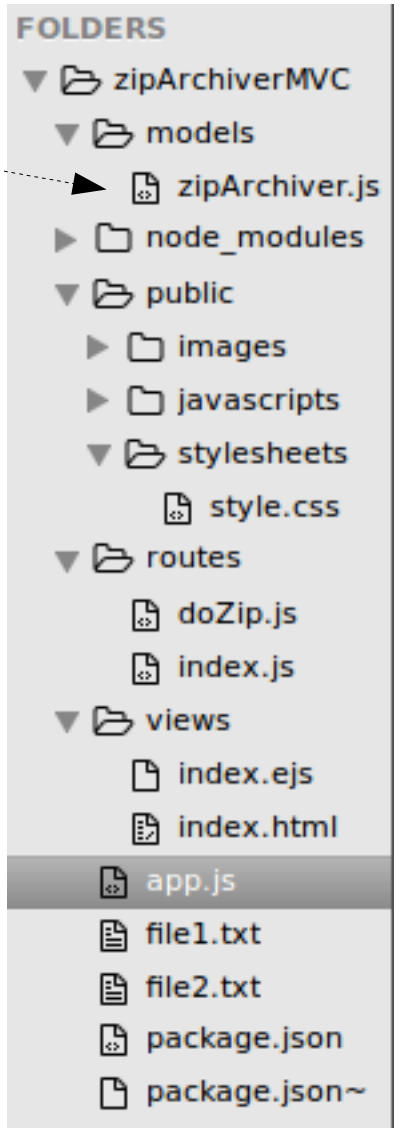
De l'arxiu **zipArchiver** executem la funció que exporta: **serverFunctionality(res)**

De totes les funcions que poden haver en aquest fitxer exporto doZip per a que es pugui usar fora d'aquest fitxer.

1-. app.js

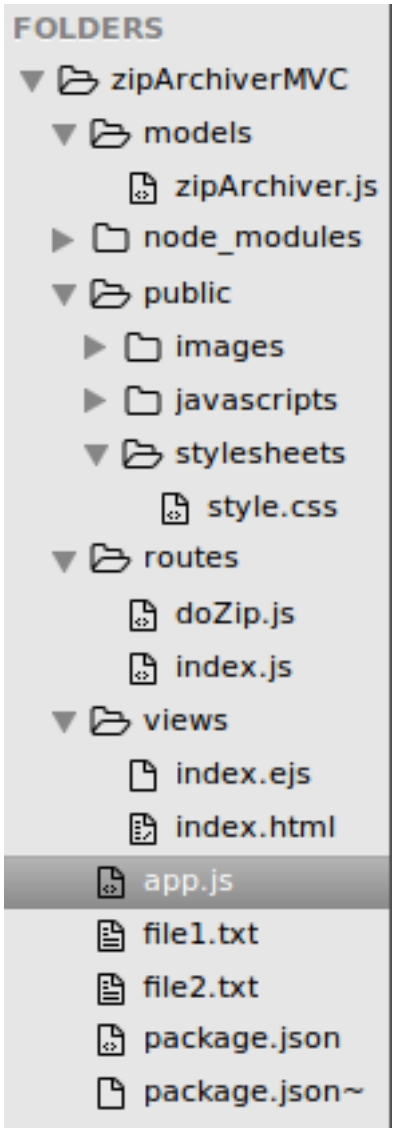
```
app.get('/doZip', zipCtrl.doZip);
```

En la variable **zipArchiver** tenim les funcions que exporta el fitxer **models/zipArchiver.js**



models/zipArchiver.js (1/2)

```
var archiver = require('archiver'), //archiver module
    fs = require('fs'); //file system module

/**
 * Function that pipes (sends) all the InputStreams in the streams parameter
 * to a ZipOutputStream connected to the HTTP Response Stream.
 * @streams Array with the InputStreams to be zipped.
 * @response OutputStream connected to the HTTP Response.
 */
function sendFilesInAZip(streams, response){
  var zipOS = archiver('zip');
  //when compression finishes we execute the anonymous func. passed as a
  //parameter.
  zipOS.on('finish', function(){
    console.log('Zip file has been finalized. Zip  size in bytes: ' + zipOS.pointer());
    //closing the OutputStream connected to the HTTP Response.
    response.end();
  });

  //piping (connecting) the zip to the HTTP response output stream.
  zipOS.pipe(response);

  //for each InputStream to be zipped we add an entry into the
  //zip archive associating an entry name to the inputStream.
  streams.forEach(function(inputStream){
    //appending an entry in the archive from a FileInputStream
    zipOS.append(inputStream.stream, {name: inputStream.name});
  });

  // finalize the archive (we are done appending files but streams have to finish yet
  // to be compressed)
  zipOS.finalize();

  console.log('Zip creation has been started.');
} //end funcition sendFilesInAZip
```

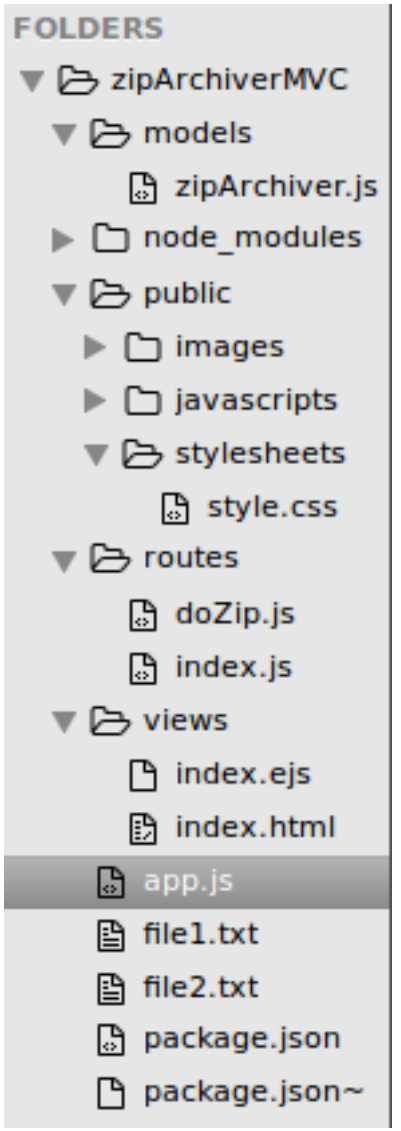


models/zipArchiver.js (2/2)

routes/doZip.js

```
zipArchiver.serverFunctionality(res);
```

```
/**
 * Function that when receives an HTTP request replies with an HTTP
 * response sending compressed in zip format the files 'file1.txt' and
 * 'file2.txt'
 */
function serverFunctionality(response){
  //sending HTTP Response headers
  response.writeHead(200, {'Content-Type': 'application/zip',
    'Content-Disposition' : 'attachment; filename=files.zip' });

  //specifying the file names to be zipped and sent
  var names = ['file1.txt','file2.txt'];

  //creating an input stream from each file. var streams will be an array
  //of inputStreams.
  var streams = names.map(function(fileName){
    var inputStream = {
      name: fileName,
      stream: fs.createReadStream(fileName)
    };

    return inputStream;

  });

  //sending all the inputStreams in the streams array to the ZipOutputStream
  //connected to the HTTP Response.
  sendFilesInAZip(streams, response);
} //end ServerFuncionality

exports.serverFunctionality = serverFunctionality;
```



De totes les funcions que hi han en zipArchiver.js només exporto **serverFunctionality** per a que es pugui usar fora d'aquest fitxer.

routes/doZip.js

```
exports.doZip =function(req, res) {
  console.log('routes/doZip');
  var zipArchiver = require("../models/zipArchiver");

  zipArchiver.serverFunctionality(res);
};
```

3

app.js

```
Var zipCtrl = require('./routes/doZip');
app.get('/doZip', zipCtrl.doZip);
```

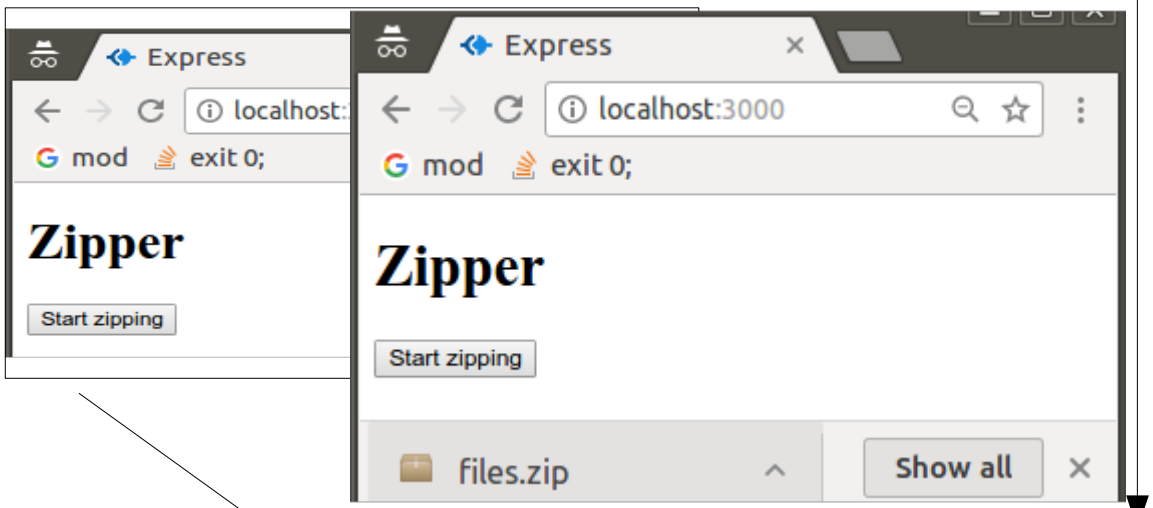
4

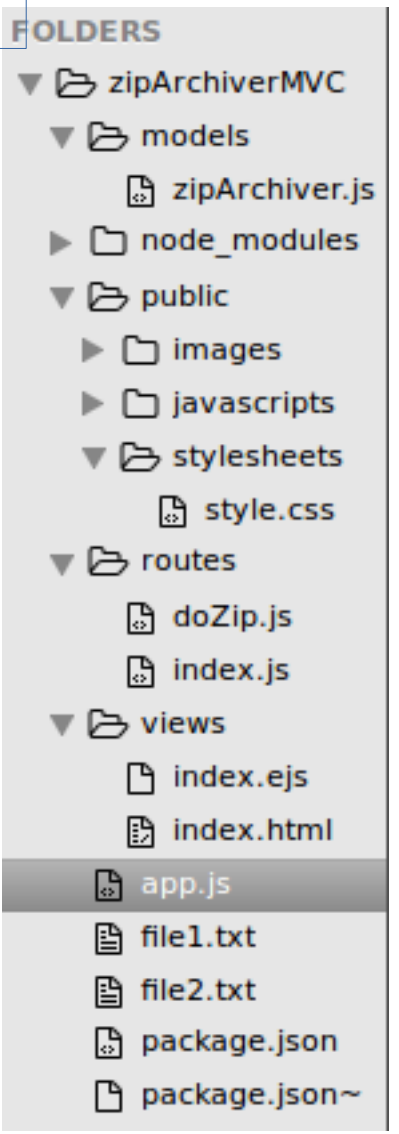
models/zipArchiver.js (2/2)

```
function serverFunctionality(response)
```

5

De totes les funcions que poden haver en aquest fitxer exporto doZip per a que es pugui usar for a d'aquest fitxer.





Click: Start Zipping

http://localhost:8080/doZip

1

GET **/doZip** HTTP/1.1

node app.js

2

HTTP/1.1 200 OK
files.zip