

Tehnici Web

CURSUL 10

Semestrul II, 2019-2020
Carmen Chirita

<https://sites.google.com/site/fmitehniciweb/>

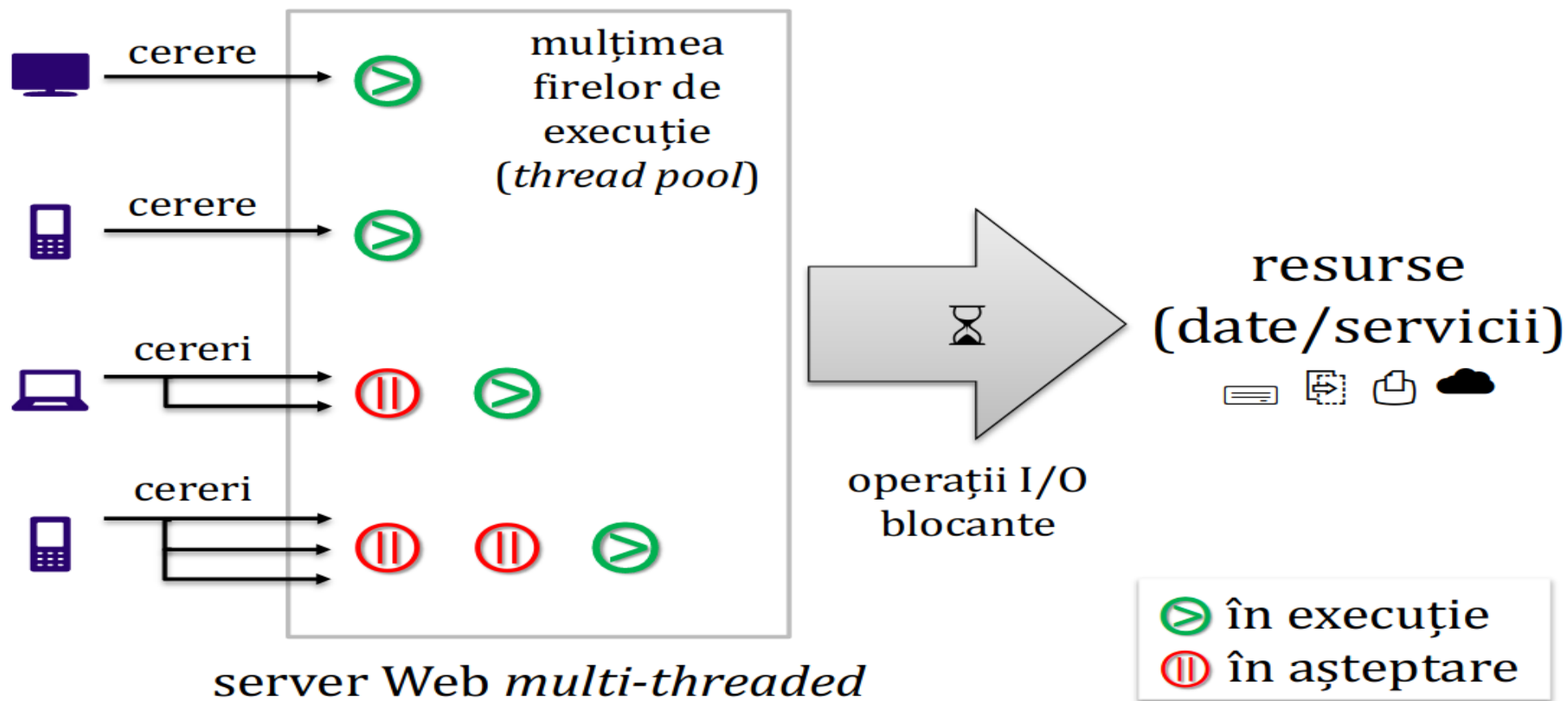
Server Web

- **Server Web** = program care ruleaza pe un calculator conectat la Internet si care furnizeaza clientilor la cerere diverse resurse Web
- Tim Berners-Lee în 1990 concepe primul **server Web** ruland pe calculatoare NeXT
- *Apache WebServer*—cel mai popular server Web. Prima versiune a fost lansata în 1995
- Alte servere web: Internet Information Services, Lighttpd, NGINX,...

Serverul Web -caracteristici

- deservește cereri multiple provenite de la clienți pe baza protocolului HTTP
- fiecare cerere e considerată independentă de alta, chiar dacă provine de la același client Web
- se creeaza un număr de fire de execuție (threads) la inițializare, fiecare fir interacționand cu un anumit client

Serverul Web



cererile multiple de la diverși clienți nu pot fi deservite simultan
(numărul firelor de execuție asociate unui proces este limitat)

Browser Web

- **Browser Web** = program software (client) ce permite utilizatorilor să se conecteze la un server Web în vederea explorării resurselor găzduite de acesta (text, grafică, video, etc.)
- Protocolul utilizat: **HTTP**
- Resursele sunt identificate printr-un Uniform Resource Locator (URL)
- Cele mai populare browsere Web: Google Chrome, Safari, Firefox, Internet Explorer, Opera

Browser Web-caracteristici

- Posibilitatea de a realiza interogări multiple către server
- Asigurarea securității transmițerilor de date
- Stabilirea de liste a site-urilor web favorite
- Memorarea istoricului navigării (history)
- Posibilitatea de a folosi mai multe ferestre de navigare
- Asigurarea suportului pentru diverse limbaje de programare folosite la realizarea paginilor Web dinamice (CSS,JavaScript)

Protocolul HTTP

- Protocolul HTTP(HyperText Transfer Protocol) = set de reguli de comunicare între un server și browser web.
- Dezvoltat în 1990 de Tim Berners-Lee
- Protocol de tip **cerere/raspuns**
- Portul standard de acces: **80**

Localizarea resurselor WEB

URI = sir de caractere care identifica o resursa prin nume sau locatie

URN = Uniform Resource Name

(identificare prin nume)

URN: ISBN: 9780062515872

URL = Uniform Resource Locator

(identificare prin locatie)

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html>

URL

`protocol:// host:port /location?query#fragment`

HTTP (port 80)

`http://webdesign.about.com/`

`http://search.about.com/?q=URL`

HTTPS = HTTP + securitate (port 443)

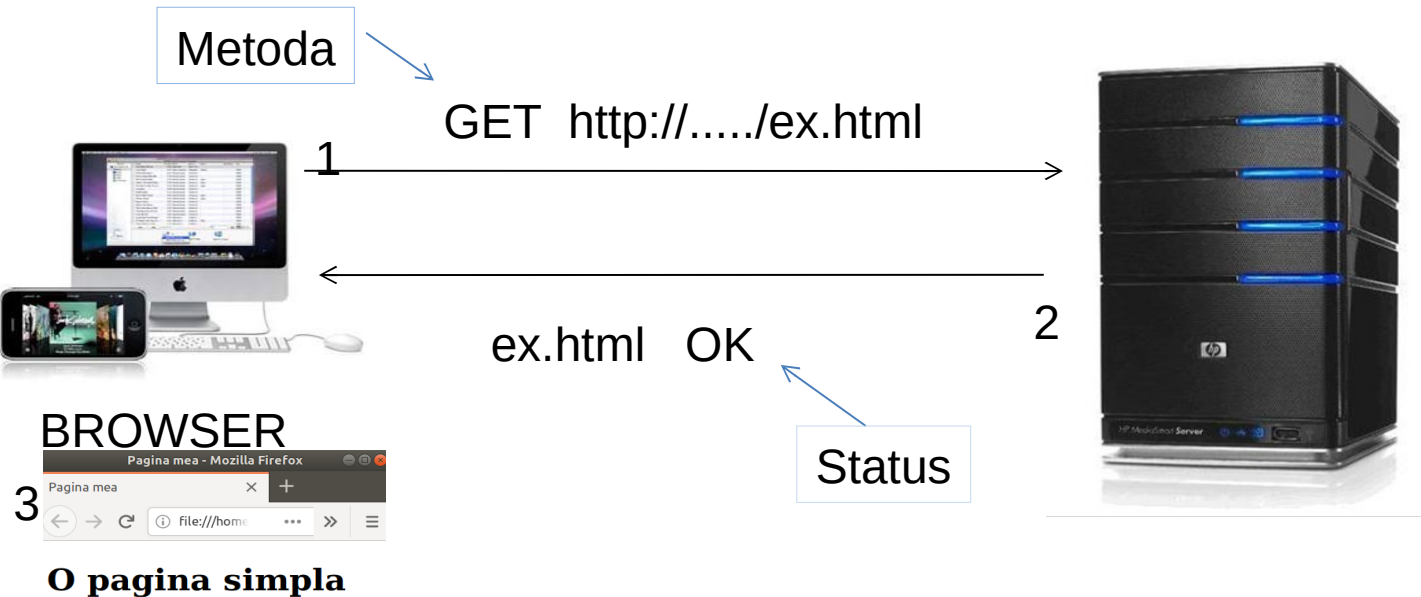
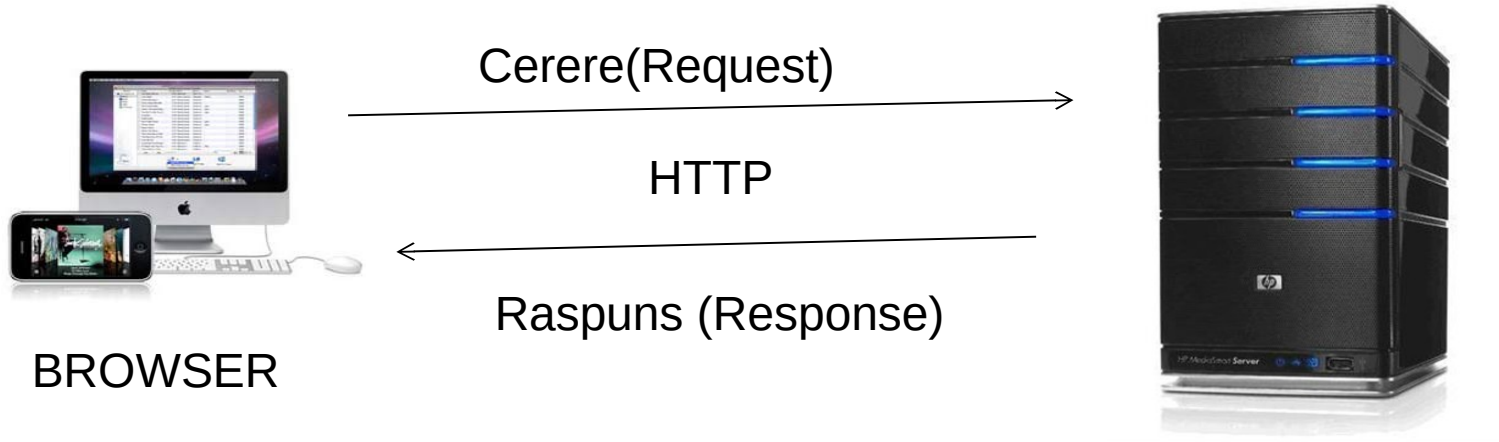
`https://web.stanford.edu/class/cs142/lectures/HTTP.pdf`

File URL = legatura la un fisier local

`file:///home/carmen/TEHNICI_WEB_CURSURI/date.xml`

Client

Server



HTTP Request

Sintaxa unei cereri HTTP

METHOD /path-to-resource HTTP/version-number

Header-Name-1: value

Header-Name-2: value

[optional request body]

HTTP Request-Exemplu

http://fmi.unibuc.ro/ro/pdf/2019/orar/orar_profesori_2019-2020_s1.pdf

GET /...orar_profesori_2019-2020_s1.pdf **HTTP/1.1**

Host: fmi.unibuc.ro

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: ro-RO,ro;q=0.8,en-US;q=0.6,en-GB;q=0.4,en;q=0.2

Accept-Encoding: gzip, deflate

Connection: keep-alive

Campuri-antet



Metode HTTP

GET - solicită un document/resursă de pe server

HEAD - solicită informații despre un document/resursă

POST - metodă utilizată pentru a transmite date către server
și a primi un raspuns

PUT - metodă utilizată pentru a actualiza/depune o resursă
pe server

DELETE- metodă utilizată pentru a șterge un
document/resursă pe server

Metode HTTP

- tradițional, browser-ul Web permite doar folosirea metodelor **GET** și **POST**
- o metoda este **sigură (safe)** când nu produce modificari în datele serverului
- GET și HEAD sunt *safe*
- POST, PUT, DELETE nu sunt *safe*

HTTP Response

Sintaxa unui raspuns HTTP

HTTP/version-number status-code message

Header-Name-1: value

Header-Name-2: value

[response body]

HTTP Response-Exemplu

http://fmi.unibuc.ro/ro/pdf/2019/orar/orar_profesori_2019-2020_s1.pdf

HTTP/1.1 200 OK

Date: Mon, 18 Nov 2019 18:23:27 GMT

Server: lighttpd/1.4.26

Content-Type: application/pdf

Content-Length: 1406

Last-Modified: Mon, 25 May 2015 15:34:17 GMT

Accept-Ranges: bytes

Coduri de stare

- 200 OK
- 301 Moved Permanently
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 503 Service Unavailable

Node.js

- permite dezvoltarea de aplicații Web la nivel de server în limbajul JavaScript
- oferă un mediu de execuție în linia de comandă, pe baza unor biblioteci C++ și a procesorului V8

`node server.js`

Node.js - caracteristici

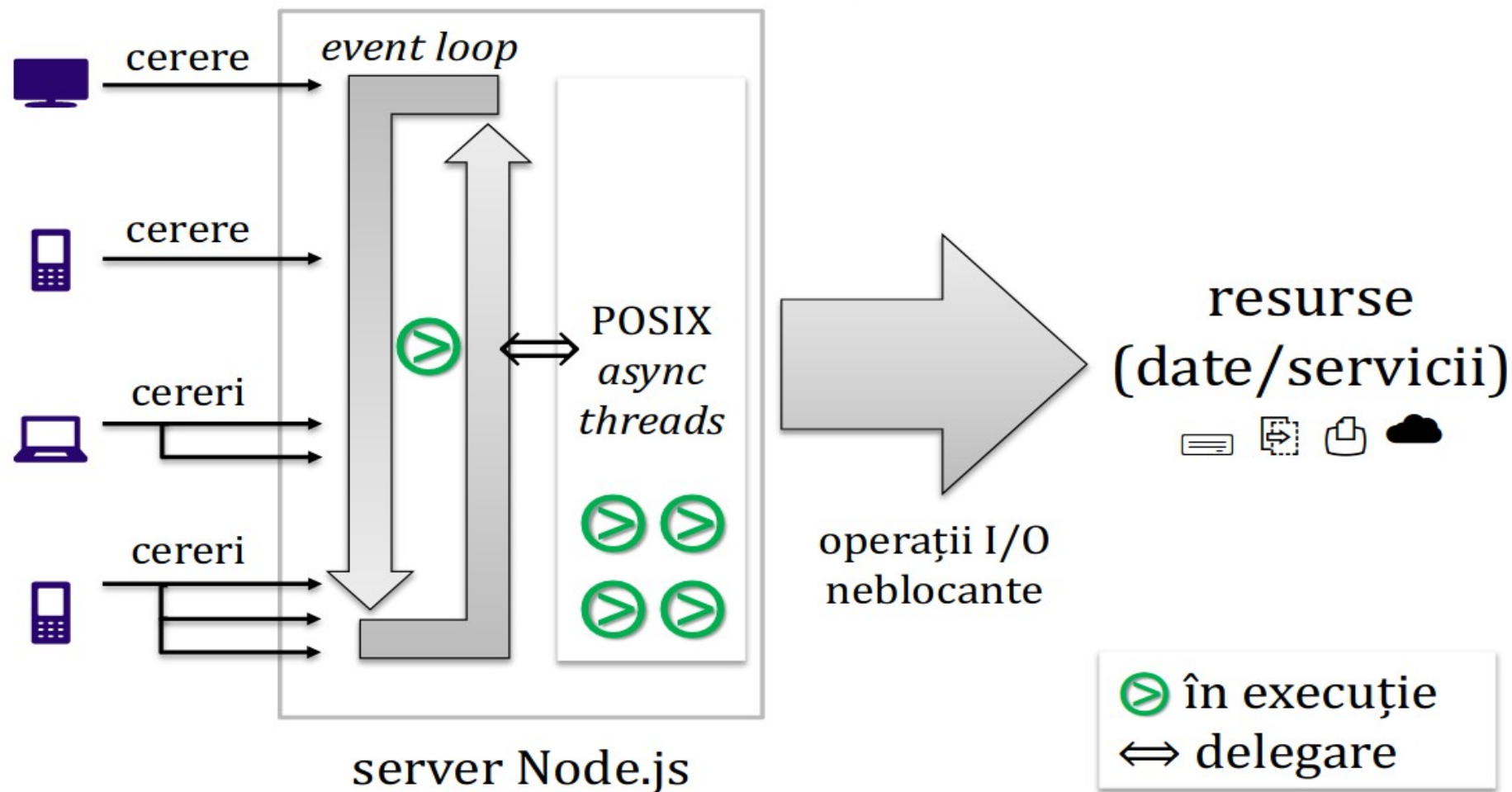
- mediul node.js e disponibil gratuit -open source-
- pentru platformele UNIX/Linux, Windows, MacOS

nodejs.org/en/download/

- operațiile de intrare/ieșire sunt asincrone
- o aplicație node.js rulează într-un singur proces

deosebire esențială față de serverele de aplicații Web tradiționale ce recurg la servere multi-process/threaded

Node.js

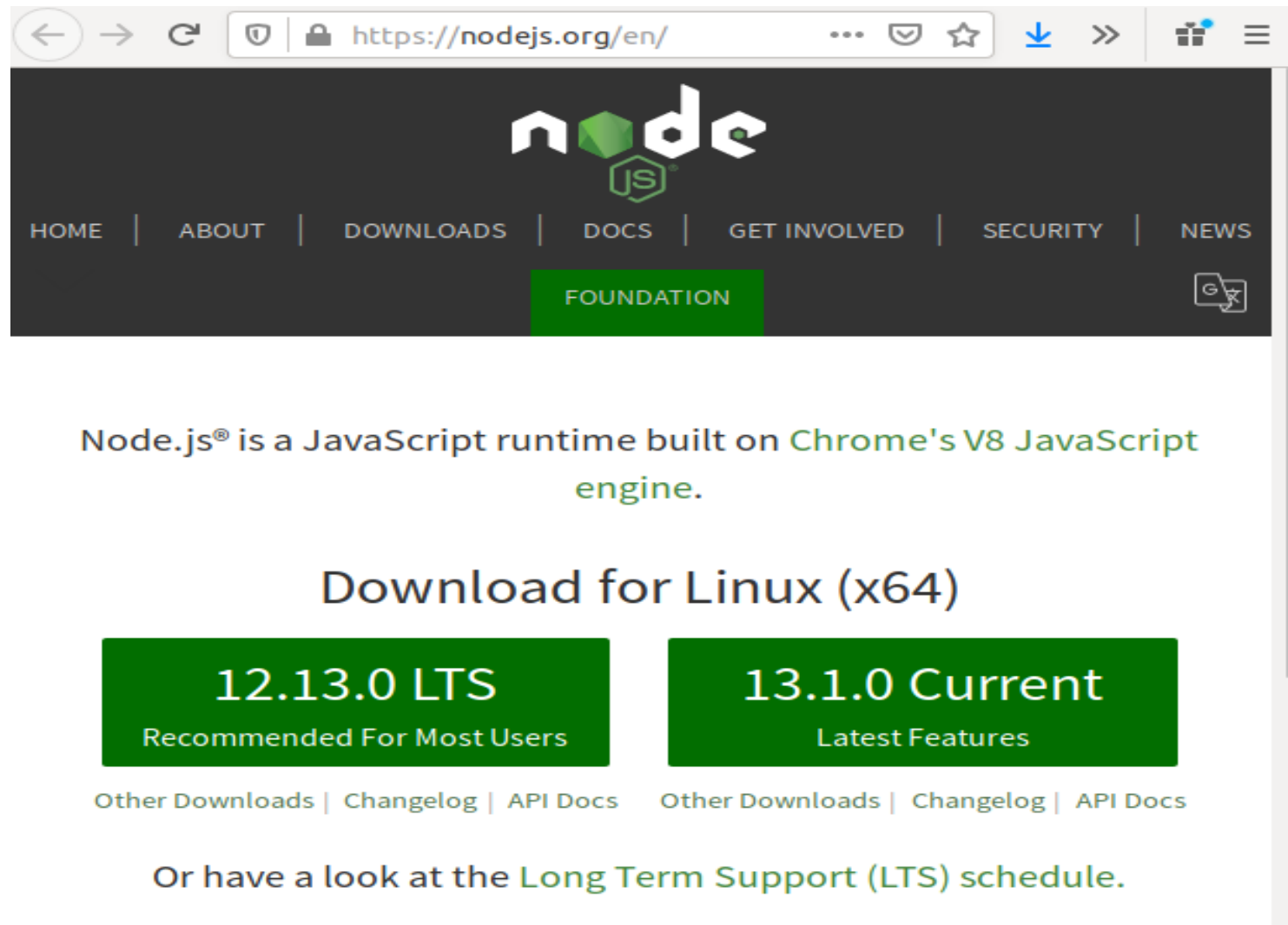


cererile multiple de la diverși clienți pot fi deservite simultan

Node.js și modulele asociate

- Node.js poate genera continut dinamic pe pagina
- Poate crea, deschide, citi, scrie, sterge si inchide fisiere pe server
- Node.js poate colecta date din formular
- Poate adauga, sterge, modifica date intr-o baza de date
- Poate crea sesiuni
- Poate face criptare/decriptare

Node.js -pagina oficiala



The image is a screenshot of the Node.js official website. At the top, there is a browser address bar showing the URL <https://nodejs.org/en/>. Below the address bar is a dark navigation bar with the Node.js logo in the center. To the left of the logo are links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, and NEWS. To the right of the logo is a green button labeled FOUNDATION and a small icon of a document with a magnifying glass. Below the navigation bar, the text reads: "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." Underneath this text is a heading "Download for Linux (x64)". Below the heading are two green buttons. The left button is labeled "12.13.0 LTS" and "Recommended For Most Users". The right button is labeled "13.1.0 Current" and "Latest Features". Below these buttons are two sets of links: "Other Downloads | Changelog | API Docs" for the LTS version and "Other Downloads | Changelog | API Docs" for the Current version. At the bottom, the text reads: "Or have a look at the Long Term Support (LTS) schedule."

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Linux (x64)

Version	Description
12.13.0 LTS	Recommended For Most Users
13.1.0 Current	Latest Features

Other Downloads | Changelog | API Docs Other Downloads | Changelog | API Docs

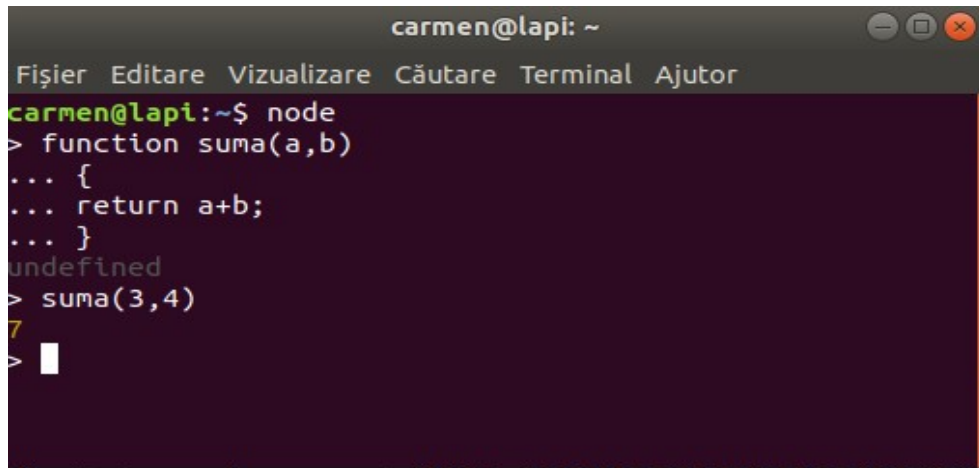
Or have a look at the Long Term Support (LTS) schedule.

Node.js Console - REPL

Node.js vine cu un mediu virtual numit REPL (Read-Eval-Print-Loop).

Este un mod rapid și ușor de testat codul Node.js / JavaScript simplu.

Lansare REPL: `node` (în linia de comanda)



```
carmen@lapi: ~  
Fișier Editare Vizualizare Căutare Terminal Ajutor  
carmen@lapi:~$ node  
> function suma(a,b)  
... {  
... return a+b;  
... }  
undefined  
> suma(3,4)  
7  
>
```

Node.js: module

Un modul este un bloc de cod JavaScript care de obicei are asociat un obiect ale carui metode pot fi invocate

Module standard predefinite (nu necesita instalare suplimentara):

- Privitoare la tehnologii Web: **http, https, url, querystring**
- Referitoare la fişiere: **fs, path**
- Vizând reţeaua: **net, dns, dgram, tls,...**
- Resurse privind sistemul de operare: **os, child_process**
- Alte aspecte de interes: **buffer, console, util, crypto**

Module globale (instalare folosind npm):
express, ejs, nodemailer, body-parser,...

Pentru a folosi un modul trebuie să-l includem cu funcţia **require()**

```
var module = require('module_name'); // întoarce un obiect asociat  
modulului respectiv
```


Modulul http

<https://nodejs.org/api/http.html>


- include functionalitati HTTP de baza
- permite receptionarea si transferarea datelor prin HTTP
- conține 5 clase:
 - http.Agent
 - http.ClientRequest
 - http.Server
 - http.ServerResponse
 - http.IncomingMessage

Modulul http

<https://nodejs.org/api/http.html>

crearea unui server Web: `createServer()`
realizarea de cereri HTTP: `request()` `get()`

```
var http=require("http");  
http.createServer(function(request,response){...}); //intoarce  
                                                    un obiect din clasa http.Server
```



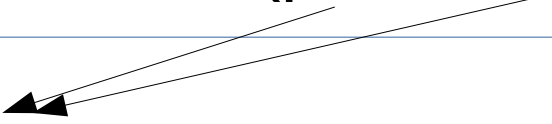
funcție care se va executa atunci când se realizează o cerere către server;
are ca parametrii obiectele **request** (din clasa IncomingMessage) și
response (din clasa ServerResponse)

Modulul http


servire de cereri HTTP – clasa `http.Server`

metode uzuale: `listen()` `setTimeout()` `close()`

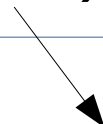
`server.listen(port, host, backlog, callback)`



portul: numeric
hostul: string



nr. de cereri acceptate
in paralel (implicit 511)



funcție care se executa când
pornește serverul

evenimente ce pot fi tratate:
`request` `connect` `close` `clientError` etc.

Modulul http

cerere emisă de client – clasa `http.ClientRequest`

metode uzuale:

`write()` `end()` `setTimeout()`

evenimente ce pot fi tratate:

`response` `connect` `continue` etc.

Modulul http

clasa `http.ClientRequest`

request.write(chunk[, encoding][, callback]) //trimite o parte din date
din corpul cererii

chunk: string; **encoding:** string (implicit „utf8”); **callback:** funcție care se va
executa după ce datele au fost trimise

request.end([data[, encoding]][, callback]) //incheie trimiterea cererii

data: string; **encoding:** string (implicit „utf8”); **callback:** funcție
care se va executa după trimiterea cererii

Modulul http

răspuns emis de server – clasa `http.ServerResponse`

metode uzuale:

`writeHead()` `getHeader()` `removeHeader()` `write()` `end()`

evenimente ce pot fi
tratate:

`close` `finish`

proprietăți folosite:

`statusCode` `headersSent`

Modulul http

clasa `http.ServerResponse`

response.write(chunk[, encoding][, callback])

\\trimite o parte din date către client

chunk: string; **encoding:** string (implicit „utf8”); **callback:** funcție care se va executa după ce datele au fost trimise

response.writeHead(statusCode[, statusMessage][, headers])

\\trimite un antet de răspuns

statusCode: numeric; **statusMessage:** string; **headers:** obiect

response.end([data[, encoding]][, callback]) \\răspunsul e complet

data: string; **encoding:** string (implicit „utf8”); **callback:** funcție care se va executa după ce răspunsul de la server este finalizat

salut.js

// Un program JavaScript care răspunde cu un mesaj de salut la toate cererile adresate de clienti Web

```
var http=require('http'); // folosim modulul 'http' predefinit
```

```
var server=http.createServer( // cream un server Web  
  // functie anonima ce trateaza o cerere si trimite un raspuns  
  function(request,response){  
    console.log("Am primit o cerere..");
```

```
  // stabilim valori pentru diverse campuri-antet HTTP  
    response.writeHead(200, {"Content-Type" : "text/html"});
```

```
  // emitem raspunsul propriu-zis conform tipului MIME (cod HTML)  
    response.end('<html><body><h1>Salutari din Node.js</h1></body></html>');  
  });  
server.listen(7000); // serverul este pornit si asculta cereri la portul 7000 al masinii locale  
  
console.log ('Serverul creat asteapta cereri la http://localhost:7000/');
```

```
carmen@lapi: ~/node-exemple  
Fișier Editare Vizualizare Căutare Terminal Ajutor  
carmen@lapi:~$ cd node-exemple  
carmen@lapi:~/node-exemple$ node salut.js  
Serverul creat asteapta cereri la http://localhost:7000/  
Am primit o cerere..
```



Modulul url

<https://nodejs.org/api/url.html>

- procesarea adreselor Web
(împarte o adresă web în părți care pot fi citite)

Metode oferite:

parse() format() resolve()

**url.parse(urlString[,
parseQueryString[,slashesDenoteHost]])**
\\transforma un URL string într-un URL obiect

Boolean (false implicit)

url.format(urlObject)
\\transforma un URL obiect într-un URL string

URL obiect-proprietati

- `urlObject.auth`
- `urlObject.hash`
- `urlObject.host`
- `urlObject.hostname`
- `urlObject.href`
- `urlObject.path`
- `urlObject.pathname`
- `urlObject.port`
- `urlObject.protocol`
- `urlObject.query`
- `urlObject.search`
- `urlObject.slashes`

Modulul url

<https://nodejs.org/api/url.html>

href							
protocol	auth		host		path		hash
			hostname	port	pathname	search	
						query	
" https: // user : pass @ sub.example.com : 8080 /p/a/t/h ? query=string #hash "			hostname	port			
protocol	username	password	host				
origin			origin	pathname	search	hash	
href							

- `urlObject.auth`
- `urlObject.hash`
- `urlObject.host`
- `urlObject.hostname`
- `urlObject.href`
- `urlObject.path`
- `urlObject.pathname`
- `urlObject.port`
- `urlObject.protocol`
- `urlObject.query`
- `urlObject.search`
- `urlObject.slashes`

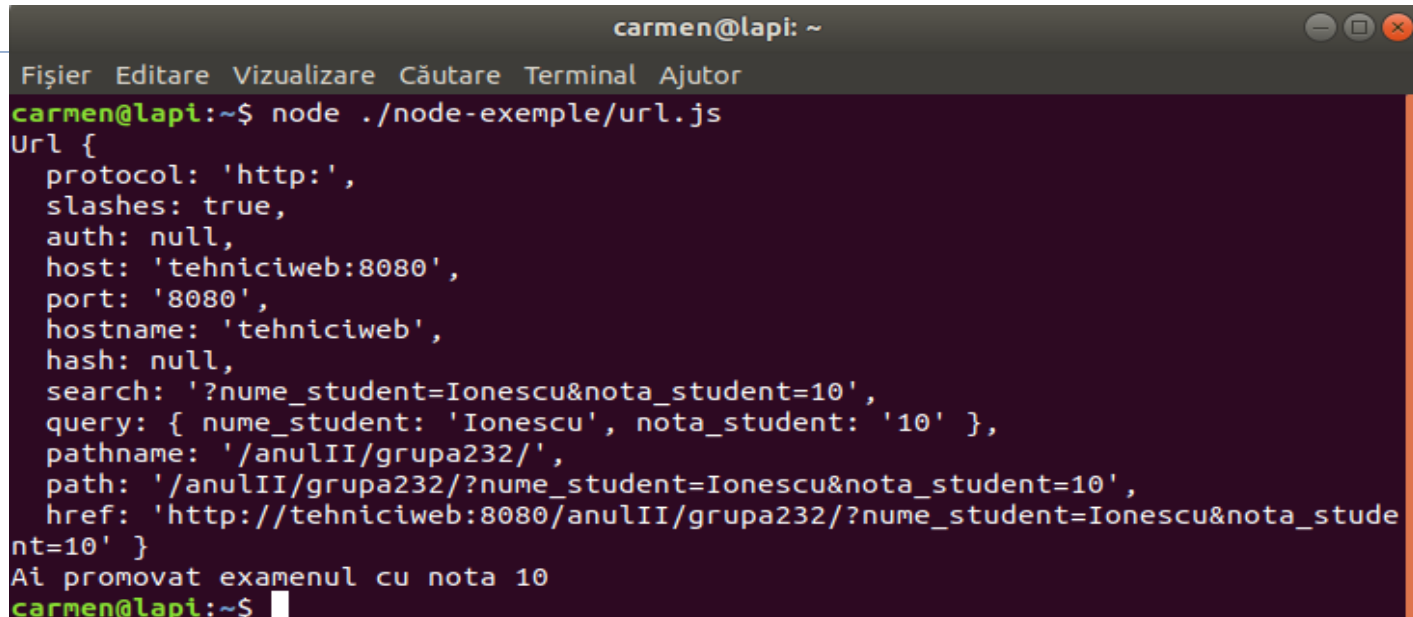
// Program ce ilustreaza procesarea URL-urilor

```
var url = require('url'); // folosim modulul 'url'
```

```
var adresa = url.parse('http://tehniciWeb:8080/anulII/grupa232/nume_student=Ionescu&nota_student=10',  
true); //parseaza un URL și întoarce un obiect URL cu fiecare parte a adresei
```

```
console.log(adresa);
```

```
if (adresa['query'].nota_student >= 5) {  
    console.log('Ai promovat examenul cu nota ' + adresa['query'].nota_student);  
} else {  
    console.log('Nu ai promovat examenul');  
}
```



```
carmen@lapi: ~  
Fișier Editare Vizualizare Căutare Terminal Ajutor  
carmen@lapi:~$ node ./node-exemple/url.js  
Url {  
  protocol: 'http:',  
  slashes: true,  
  auth: null,  
  host: 'tehniciweb:8080',  
  port: '8080',  
  hostname: 'tehniciweb',  
  hash: null,  
  search: '?nume_student=Ionescu&nota_student=10',  
  query: { nume_student: 'Ionescu', nota_student: '10' },  
  pathname: '/anulII/grupa232/',  
  path: '/anulII/grupa232/?nume_student=Ionescu&nota_student=10',  
  href: 'http://tehniciweb:8080/anulII/grupa232/?nume_student=Ionescu&nota_stude  
nt=10' }  
Ai promovat examenul cu nota 10  
carmen@lapi:~$
```

Tratarea erorilor în JavaScript

JavaScript dispune de un mecanism de tratare a erorilor (exista erori implicite și erori definite de utilizator cu ajutorul obiectului *Error*)

- lansarea unei erori

throw *expresie* //expresie este un argument de tip Error dar poate fi de alt tip (ex. string)

- prinderea unei erori

try { //instrucțiuni care pot genera erori }

catch(e) { // instrucțiuni de testare a tipului de eroare generat și de tratare a erorii }

finally { //cod care se executa la final }

Modulul fs

- permite operatii cu fisiere/directoare pe server (citire, creare, adaugare date, stergere, etc.)

Metode oferite:

`readFile()` `writeFile()` `open()` `appendFile()` `rename()` `unlink()`
variantele sincrone

`readFileSync()` `writeFileSync()` `appendFileSync()`

`fs.readFile(fileName [,options], callback)`

```
var fs = require('fs');
fs.readFile('fisier.txt','utf8', function (err, data) {
    if (err) throw err;
    console.log(data);
});
console.log('citire asincrona');
```

`fs.readFileSync(fileName [,options])`

```
var fs = require('fs');
var data =fs.readFileSync('dummyfile.txt',
'utf8');
console.log(data);
console.log('citire sincrona');
```

file.js

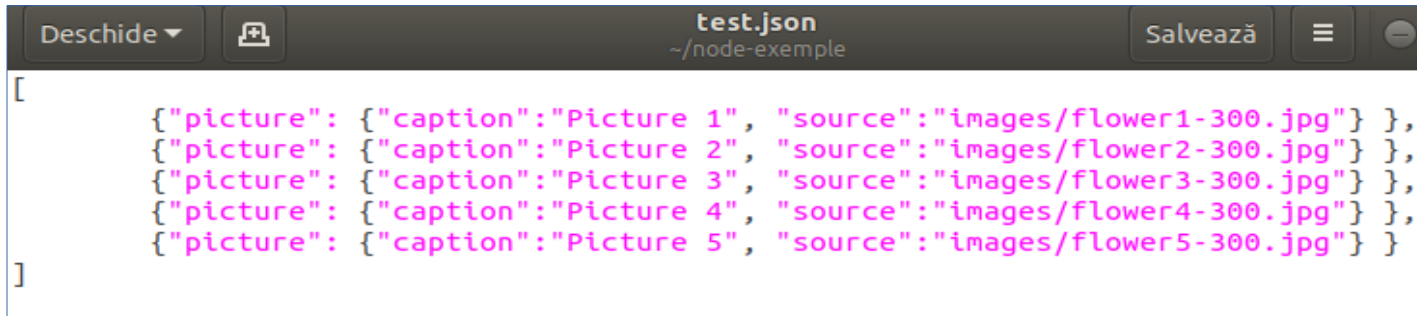
//citire date dintr-un fisier json și adaugarea lor într-un fisier html

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

Date în format JSON

```
fs.readFile('test.json',function (err, data) {  
    if (err) throw err;  
    var json=JSON.parse(data);    //transformare din string JSON într-un array JavaScript  
    fs.writeFileSync('test.html','<html><body>');  
    for(var i=0; i<json.length;i++)  
        fs.appendFileSync('test.html','<img src='+json[i].picture.source +'>');  
    fs.appendFileSync('test.html','</body></html>');  
  
    console.log('Operatie completa.');
```

});



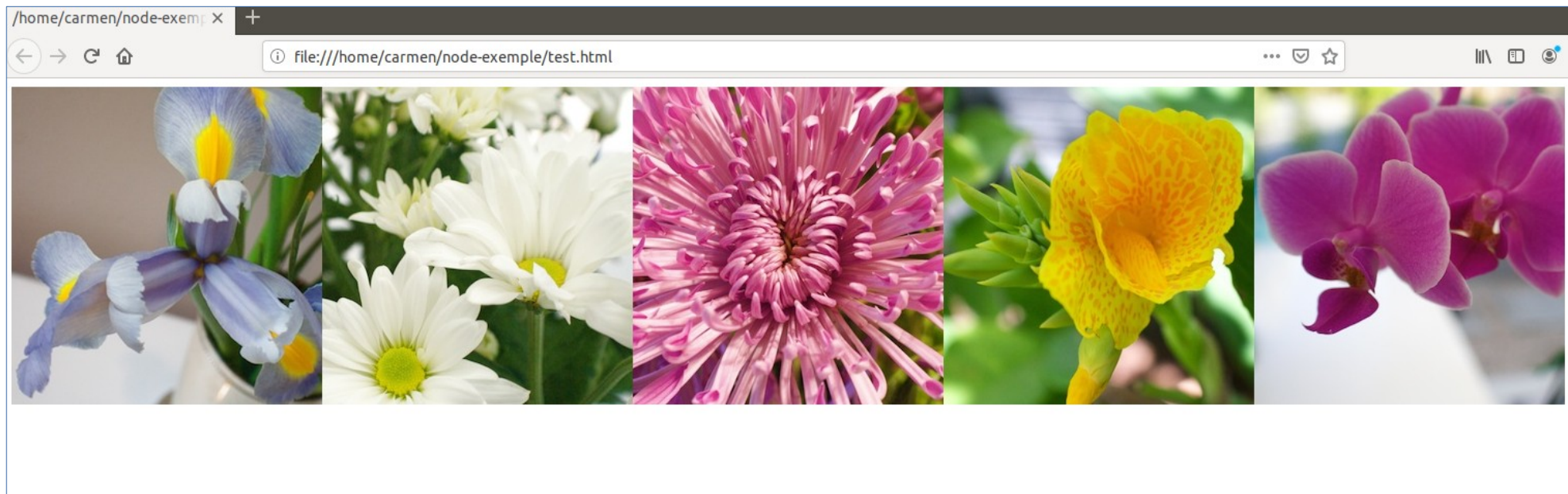
The screenshot shows a code editor window titled 'test.json' with the file path '~/.node-exemple'. The editor contains a JSON array of five objects, each representing a picture with a caption and a source path. The text is highlighted in pink.

```
[  
  {"picture": {"caption": "Picture 1", "source": "images/flower1-300.jpg"} },  
  {"picture": {"caption": "Picture 2", "source": "images/flower2-300.jpg"} },  
  {"picture": {"caption": "Picture 3", "source": "images/flower3-300.jpg"} },  
  {"picture": {"caption": "Picture 4", "source": "images/flower4-300.jpg"} },  
  {"picture": {"caption": "Picture 5", "source": "images/flower5-300.jpg"} }  
]
```

```
carmen@lapi: ~/node-exemple
Fișier Editare Vizualizare Căutare Terminal Ajutor
To run a command as administrator (user "root"), use "sudo <command>"
See "man sudo_root" for details.

carmen@lapi:~$ cd node-exemple
carmen@lapi:~/node-exemple$ node file.js
Operatie completa.
carmen@lapi:~/node-exemple$
```

Fișierul creat **test.html**



Module custom în Node.js

module create de utilizator
și incluse apoi în aplicație

Cuvântul cheie: **exports**

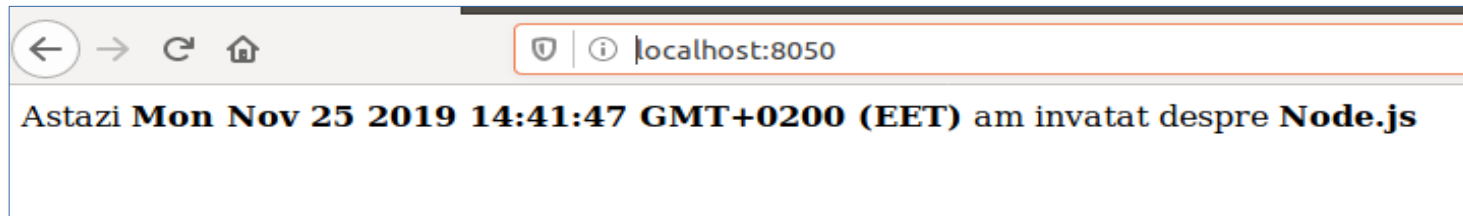
mymodule.js

```
module.exports = {  
  myDate : function () {  
    return Date();  
  },  
  myMessage: function() { return 'Node.js';}  
};
```

```
var http = require('http');  
var date=require('./mymodule');
```

custom.js

```
http.createServer(function(req, res) {  
  console.log('am primit un request');  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('<html><body>Astazi <b>' + date.myDate() +  
'</b> am invatat despre <b>' + date.myMessage() + '</b></body></html>');  
}).listen(8050);
```



Modulul crypto

-ofera metode pentru criptarea și decriptarea datelor (ex. pt. securizarea parolelor înainte de a fi stocate în baza de date)

createCipher() createDecipher() update() final()

```
var crypto = require('crypto'); //includem modulul crypto
var password = 'tehniciweb2019';
var cipher = crypto.createCipher('aes128', 'a password'); //creaza un obiect de tip
                                                         algoritm de cifrare
var encrypted = cipher.update(password, 'utf8', 'hex'); //criptarea parolei
encrypted += cipher.final('hex'); //finalizarea criptarii
console.log(encrypted); //9152f4b48a037fce5c269fbd56376fc4
```

```
var encrypt_password = '9152f4b48a037fce5c269fbd56376fc4';
var decipher = crypto.createDecipher('aes128','a password');
var decrypted = decipher.update(encrypt_password,'hex', 'utf8');
decrypted += decipher.final('utf8');
console.log(decrypted); //tehniciweb2019
```

Submiterea datelor dintr-un formular cu metoda GET și salvarea lor într-un fișier text

form.html

```
<!DOCTYPE html>
<form action="http://localhost:8080/cale" method="GET"
target="_blank">
.....
</form>
```

Nume:

Varsta:

Localitate:

Query string în url

localhost:8080/cale?name=Oana&age=23&city=Cluj

form.js

```
var http = require('http');
var url = require('url');
var fs = require('fs');
var server = http.createServer(function (req, res)
{
var url_parts=url.parse(req.url,true);
if(url_parts.pathname == '/cale'){
var query=url_parts.query;
fs.appendFileSync('date.txt', query.name + ',' + query.age + ',' + query.city+ '\n');
res.writeHead(200, {'Content-Type': 'text/plain'});
res.end(query.name + ' din ' + query.city + ' are ' + query.age + ' ani ');}
}).listen(8080);
```

Deschide ▾ **date.txt** ~/node-exemple/F... Salvează

Andrei,25,Brasov
Maria,10,Bucuresti
Bogdan,20,Bucuresti
Oana,23,Cluj

Submiterea datelor dintr-un formular cu metoda POST (mai simplu folosind Express)

<https://itnext.io/how-to-handle-the-post-request-body-in-node-js-without-using-a-framework-cd2038b93190>

```
var http = require('http');
var x = require('querystring');

var server = http.createServer((req, res) => {
  if (req.method === 'POST') {
    collectRequestData(req, result => {
      console.log(result);
      res.end('Ati introdus numele: ' + result.name);
    });
  }
});
server.listen(8080);

function collectRequestData(request, callback) {
  const FORM_URLENCODED = 'application/x-www-form-urlencoded';
  if(request.headers['content-type'] === FORM_URLENCODED) {
    let body = " ";
    request.on('data', chunk => {
      body += chunk.toString();
    });
    request.on('end', () => {
      callback(x.parse(body));
    });
  }
  else {
    callback(null);
  }
}
```

```
<!DOCTYPE html>
<form action="http://localhost:8080/cale" method="POST"
target="_blank">
.....
</form>
```

NPM (Node Package Manager)

- utilitar pentru administrarea pachetelor (instalare, update, dezinstalare, publicarea modulelor, etc.)
- se instaleaza odată cu Node.js
- comenzi specifice pentru operatii asupra modulelor

npm(site oficial) <https://www.npmjs.com/>

CLI documentation > CLI

- > **access** Set access level on published packages
- > **adduser** Add a registry user account
- > **audit** Run a security audit
- > **bin** Display npm bin folder
- > **bugs** Bugs for a package in a web browser maybe
- > **build** Build a package
- > **bundle** REMOVED
- > **cache** Manipulates packages cache
- > **ci** Install a project with a clean slate
- > **completion** Tab Completion for npm
- > **config** Manage the npm configuration files
- > **dedupe** Reduce duplication
- > **deprecate** Deprecate a version of a package
- > **dist-tag** Modify package distribution tags
- > **docs** Docs for a package in a web browser maybe
- > **doctor** Check your environments
- > **edit** Edit an installed package
- > **explore** Browse an installed package
- > **help-search** Search npm help documentation
- > **help** Get help on npm
- > **hook** Manage registry hooks
- > **init** create a package.json file
- > **install** Install a package
- > **link** Symlink a package folder
- > **logout** Log out of the registry
- > **ls** List installed packages
- > **npm** javascript package manager
- > **org** Manage orgs
- > **outdated** Check for outdated packages
- > **owner** Manage package owners
- > **pack** Create a tarball from a package
- > **ping** Ping npm registry
- > **prefix** Display prefix
- > **profile** Change settings on your registry profile
- > **prune** Remove extraneous packages
- > **publish** Publish a package
- > **rebuild** Rebuild a package
- > **repo** Open package repository page in the browser
- > **restart** Restart a package
- > **root** Display npm root
- > **run-script** Run arbitrary package scripts
- > **search** Search for packages
- > **shrinkwrap** Lock down dependency versions for publication
- > **star** Mark your favorite packages
- > **stars** View packages marked as favorites
- > **start** Start a package
- > **stop** Stop a package
- > **team** Manage organization teams and team memberships

- > **test** Test a package
- > **token** Manage your authentication tokens
- > **uninstall** Remove a package
- > **unpublish** Remove a package from the registry
- > **update** Update a package
- > **version** Bump a package version
- > **view** View registry info
- > **whoami** Display npm username

package.json

conține meta-date (numele modulului, versiune, autor,...) + informații privind dependențele de alte module

<https://docs.npmjs.com/files/package.json>

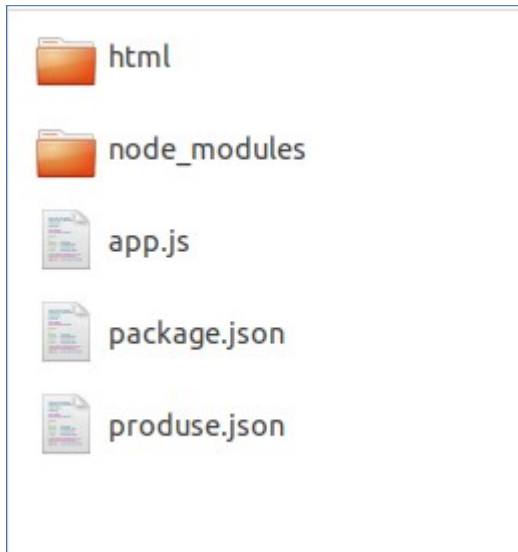
După crearea unui folder pentru proiect (aplicatie)

npm init //creaza fișierul package.json în folderul rădăcina al proiectului

npm install nume-modul --save //instaleaza un modul ca dependentă a proiectului (va apărea în package.json)

Exemplu

folderul **aplicatie_express**



```
Deschide ▼  package.json  Salvează  ≡
~/TehniciWeb/node/aplicatie_express

{
  "name": "aplicatie_express",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "connect-multiparty": "^2.2.0",
    "express": "^4.17.1"
  }
}
```

Trimiterea de emailuri folosind modulul **nodemailer**

npm install nodemailer --save //instalarea modulului

```
var nodemailer = require('nodemailer'); //folosirea modulului nodemailer
```

```
var transporter = nodemailer.createTransport({ //face autentificarea
  service: 'gmail',
  auth: {
    user: 'my.mail.node@gmail.com',
    pass: 'nodemailer'
  },
  tls:{rejectUnauthorized:false}
});
```

```
var mailOptions = { //optiunile mesajului
  from: 'my.mail.node@gmail.com',
  to: 'carmen_stama@yahoo.com',
  subject: 'Mesaj din Node.js',
  text: 'Hello!'
};
```

```
transporter.sendMail(mailOptions, function(error, info){ //trimite mail
  if (error) {
    console.log(error);
  } else {
    console.log('Mail trimis: ' + info.response);
  }
});
```

<https://nodemailer.com/usage/>

Trimiterea de emailuri folosind modulul **nodemailer**

Setari în Google Account



<https://nodemailer.com/usage/>

Express.js

Express reprezintă o platforma minimala și flexibila care oferă un set de funcționalități pentru dezvoltarea facila și rapidă de aplicații web

- este integrat cu diferite module pentru procesarea de cereri și de răspunsuri HTTP
- definește un sistem de rute prin intermediul căruia se stabilește ce acțiune este realizată în funcție de resursa care este solicitată precum și de metoda folosită
- permite redarea dinamica a paginilor HTML pe baza unor template-uri
- furnizează acces la informațiile stocate în diferite surse de date

Instalare: `npm install express --save`

Creare server cu `express`

```
var express = require('express'); //importam modulul express
```

```
var app = express(); // obiectul app corespunzător aplicației  
express are metode pentru:
```

- rutarea cererilor HTTP
- redarea HTML (template-uri folosind EJS)
- accesul la resurse statice (middleware-ul `express.static`)

```
app.listen(5000); //pornirea serverului la portul specificat
```

<https://expressjs.com/en/5x/api.html>

Express- rutare

- rutarea descrie modul în care aplicația răspunde diverselor cereri în funcție de URL-ul specificat și a metodei HTTP folosite
- rutele se definesc folosind metode ale obiectului aplicației Express (notat **app** in exemple) care corespund metodelor HTTP

Sintaxa unei rute: **app.METHOD(PATH, HANDLER)**

app.get(url,function(req,res){..});

get, post, put, delete

app.post(url,function(req,res){..});

app.put(url,function(req,res){..});

app.delete(url, function(req,res){..});

Functia care va procesa cererea (are ca parametrii obiectele **request** si **response**)

Obiectul de tip **request** <https://expressjs.com/en/5x/api.html#req>

- conține proprietati pentru procesarea cererii

```
app.get('/ceva', function(req, res) {...});
```

req.**query** - obiect continand parametrii din query

(ex. ?name=Maria&age=22 ⇒ {name: 'Maria', age:'22'})

req.**body** - obiect continand body-ul parsat

req.**get**(camp) - valoarea câmpului antet HTTP specificat

Obiectul de tip **response** <https://expressjs.com/en/5x/api.html#res>

- conține metode pentru setarea răspunsului HTTP

```
app.get('/ceva', function(req, res) {...});
```

res.**write(content)** - scrie în conținutul răspunsului

res.**status(code)** - setează status codul răspunsului

res.**end()** - încheie răspunsul

res.**end(msg)** - încheie răspunsul cu un conținut

res.**send(content)** - write() + end()

res.**redirect(url)** - redirectionare către alt url

```
app.use(function (req, res, next) {...});
```

- metoda **use** inregistreaza o funcție care contribuie la procesarea cererii

Exemple:

//functie care întoarce resurse statice dintr-un director

```
app.use(express.static('html'))
```

//functie care parseaza body-ul pentru form-uri submise cu post

```
app.use('/action',express.urlencoded({extended:true}))
```

//functie care intercepteaza un răspuns cu status 404

```
app.use(function(req,res){  
  res.status(404).send("<html><body>Page not found!</body><html>");  
});
```

```
var express = require('express'); //incarc modulul express
```

```
var app = express(); //obiectul aplicației express
```

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

```
app.use('/action', express.urlencoded({extended:true}));
```

app.js

```
app.get('/home',function(req,res){ //inregistrare handler pentru cereri get la calea '/home'  
res.send('<html><body>My home page!</body><html>');}) //cu răspuns în format html
```

```
app.get('/get',function(req,res){ //inregistrare handler pentru cereri get la calea '/get' cu  
res.redirect("formget.html");}); //redirectionare în raspuns
```

```
app.get('/action', function(req, res) { //inregistrare handler pentru cereri get la calea '/action'  
res.send(req.query.name + ' din ' //cu preluare date din form(cu get) in răspuns  
+ req.query.city + ' are ' + req.query.age + ' (de) ani');})
```

```
app.get('/post',function(req,res){ //inregistrare handler pentru cereri get la calea '/post'  
res.redirect("formpost.html");}); //redirectionare în raspuns
```

```
//inregistrare handler pentru cereri post la calea '/action' cu preluare date din form (cu post) în  
răspuns
```

```
app.post('/action',function(req, res) {res.send(req.body.name + ' din '  
+ req.body.city + ' are ' + req.body.age + ' (de) ani');})
```

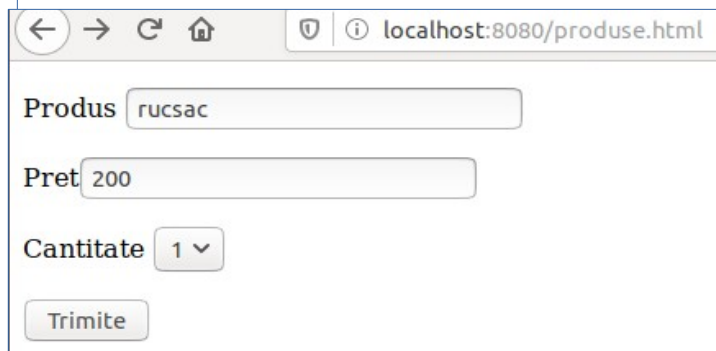
```
app.use(function(req,res){  
res.status(404).send("<html><body>Page not found!</body><html>");});
```

```
app.listen(8080, function() {console.log('listening')}); //porneste serverul pe portul 8080
```


La submiterea datelor dintr-un form acestea se introduc într-un fisier json

```
.....  
var fs=require('fs');  
  
//fs.writeFileSync("produse.json",JSON.stringify([]));  
  
app.use(express.static('html'));  
app.use('comanda',express.urlencoded({extended:true}));  
  
app.post('/comanda',function(req,res){  
var date= fs.readFileSync("produse.json");  
var ob=JSON.parse(date);  
ob.push(req.body);  
fs.writeFileSync("produse.json",JSON.stringify(ob));  
res.send("Produs adaugat:" + req.body.num);  
.....
```

app.js



localhost:8080/produse.html

Produs

Pret

Cantitate



Deschide  produse.json Salvează 

~/TehniciWeb/node...

```
[{"nume": "rucsac", "pret": "200", "cantitate": "1"},  
{"nume": "stilou", "pret": "50", "cantitate": "2"},  
{"nume": "caiet", "pret": "7", "cantitate": "5"},  
{"nume": "creion", "pret": "3", "cantitate": "3"},  
{"nume": "penar", "pret": "15", "cantitate": "4"}]
```

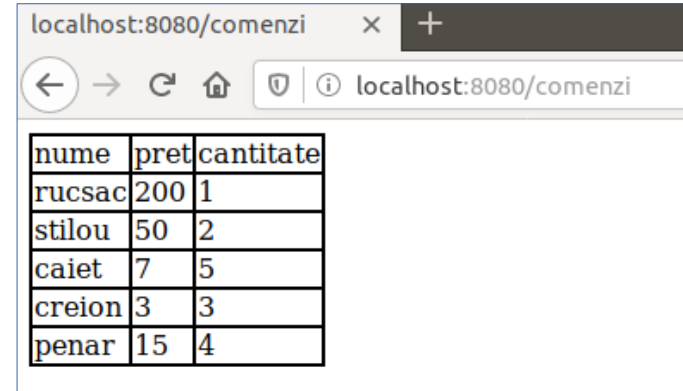
Citirea datelor dintr-un fisier json și inserarea lor într-un tabel html

```
.....
app.get('/comenzi', function(req, res) {
  var date = fs.readFileSync("produse.json");
  var comenzi = JSON.parse(date);
  res.status(200);
  res.write('<html><head><link rel="stylesheet"
href="stil.css"></head><body><table><tr><td>nume</td><td>pret</td>
<td>cantitate</td></tr>');
  for(c of comenzi) {
    res.write('<tr><td>');
    res.write(c.nume);
    res.write('</td><td>');
    res.write(c.pret);
    res.write('</td><td>');
    res.write(c.cantitate);
    res.write('</td></tr>');
  }
  res.write('</table></body></html>');
  res.end();
});
.....
```



A screenshot of a code editor window showing the contents of a file named 'produse.json'. The file contains a JSON array of five objects, each representing a product with its name, price, and quantity. The text is color-coded: 'nume' is pink, 'pret' is blue, and 'cantitate' is green.

```
[{"nume": "rucsac", "pret": "200", "cantitate": "1"},
{"nume": "stilou", "pret": "50", "cantitate": "2"},
{"nume": "caiet", "pret": "7", "cantitate": "5"},
{"nume": "creion", "pret": "3", "cantitate": "3"},
{"nume": "penar", "pret": "15", "cantitate": "4"}]
```



A screenshot of a web browser window showing the rendered HTML table. The browser's address bar displays 'localhost:8080/comenzi'. The table has three columns: 'nume', 'pret', and 'cantitate'. It contains five rows of data corresponding to the products in the JSON file.

nume	pret	cantitate
rucsac	200	1
stilou	50	2
caiet	7	5
creion	3	3
penar	15	4

app.js