Introducció a Node.js


Node.js és un entorn d'execució per a JavaScript, construït sobre el motor V8 de Google Chrome. Això permet executar aplicacions JavaScript en qualsevol sistema compatible, incloent-hi entorns de servidor, on JavaScript no tenia un paper destacat abans de l'aparició de Node.js. A continuació, explorarem les seues característiques principals i el seu impacte en el desenvolupament web.

1. Evolució de JavaScript

Com hem comentat, Node.js és un entorn que permet executar el llenguatge de programació JavaScript en el costat del servidor. Esta afirmació pot resultar mundana, però en realitat és una cosa sorprenent. Si tirem la vista arrere, el llenguatge JavaScript ha passat per diverses etapes o fases d'expansió successives:

1. **Primera etapa (anys 90):** es començaven a desenvolupar webs amb HTML, i el JavaScript que s'emprava llavors permetia afegir dinamisme a eixes pàgines, bé validant formularis, obrint finestres, o explorant el DOM (estructura d'elements de la pàgina), afegint o llevant continguts d'este. Era el que es coneixia com a HTML dinàmic o DHTML.
2. **Segona etapa (principis del segle XXI):** va arribar amb la incorporació de les comunicacions asíncrones, és a dir, amb AJAX. Es van desenvolupar llibreries com *Prototype* (primer) o *jQuery* (després) que van permetre actualitzar fragments de pàgines web, cridant des de JavaScript a documents del servidor, arreplegant la resposta i pegant-la en una zona concreta de la pàgina, sense necessitat de recarregar-la per complet.
3. **Tercera etapa:** esta següent etapa, vinculada a l'anterior, va tindre lloc amb l'aparició de distints frameworks JavaScript per al desenvolupament d'aplicacions web en el costat del client, o *frontend*. Mitjançant una sèrie de funcionalitats incorporades, i de llibreries externes, permeten dotar a l'aplicació client d'una estructura molt determinada amb uns afegits que faciliten, entre altres coses, la compartició de variables entre vistes o pàgines, o la generació dinàmica de contingut HTML en la pròpia vista. Parlem, fonamentalment, de frameworks com a Angular, React o Vue.
4. **Quarta etapa:** l'última etapa ha arribat amb l'expansió de JavaScript al costat del servidor. Fins a este moment només s'utilitzava en la part client, és a dir, fonamentalment en els navegadors, per la qual cosa només estàvem utilitzant i veient una versió reduïda o restringida del llenguatge. Créiem que JavaScript només servia per a validacions, exploració del contingut HTML d'una pàgina o càrrega de continguts en zones concretes. Però la realitat és que JavaScript pot ser un llenguatge complet, i això significa que podem fer amb ell qualsevol cosa que es pot fer amb altres llenguatges complets, com Java o C#: accedir al sistema de fitxers, connectar amb una base de dades, etc.

1.1. JavaScript en el servidor. El motor V8

Com hem vist, fins a 2009, JavaScript era un llenguatge de programació que s'utilitzava principalment en el costat del client, és a dir, en el navegador web. No obstant això, amb l'arribada de Node.js, va ser possible

utilitzar JavaScript també en el servidor. Això va ser possible gràcies a l'ús del motor V8, que originalment va ser desenvolupat per Google per al seu navegador Chrome.

Què és el motor V8?

V8 és un motor de JavaScript escrit en C++ i de codi obert. La seua funció principal és compilar i executar el codi JavaScript, transformant-lo en codi màquina optimitzat, el qual és entés directament pel processador. Este procés permet que les aplicacions JavaScript s'executen de manera molt ràpida i eficient.

A més de la compilació, V8 també maneja la gestió de la memòria, alliberant els recursos que ja no són necessaris durant l'execució del codi (això es coneix com *garbage collection*).

A l'ésser de codi obert i estar escrit en C++, V8 pot ser adaptat i estés per a afegir noves funcionalitats, la qual cosa permet que els programadors puguen utilitzar JavaScript en una varietat d'aplicacions més enllà del navegador.

V8 i Node.js

Node.js és un entorn que permet l'execució de JavaScript en el servidor utilitzant el motor V8. En este entorn, V8 no sols compila i executa JavaScript, sinó que també se li afigen funcionalitats addicionals a través de mòduls escrits en C++. Estos mòduls permeten, per exemple, accedir al sistema d'arxius o connectar a bases de dades, alguna cosa que no és possible fer directament des d'un navegador.

Gràcies a V8 i Node.js, JavaScript s'ha convertit en un llenguatge de propòsit general, sent utilitzat en aplicacions del món real com a servidors d'alta concurrència, microservicis i fins i tot en el desenvolupament d'aplicacions d'escriptori mitjançant ferramentes com *Electron*.

1.2. Requisits perquè JavaScript puga córrer en el servidor

Abans de l'aparició de Node.js, llenguatges com PHP, ASP.NET o JSP eren les principals opcions per al desenvolupament en el costat del servidor. Estos llenguatges comptaven amb característiques que JavaScript, en la seua forma original, no tenia, la qual cosa els permetia funcionar eficientment en entorns de servidor. Entre les característiques clau d'estos llenguatges que els permetien operar en el servidor s'inclouen:

- **Accés al sistema d'arxius:** estos llenguatges disposen de mecanismes integrats per a interactuar amb el sistema d'arxius del servidor. Això és essencial per a tasques com la lectura d'arxius de text, la manipulació de dades emmagatzemades en el servidor, o la pujada i emmagatzematge d'imatges.
- **Connexió amb bases de dades:** disposen d'interfícies i ferramentes per a connectar-se amb bases de dades, permetent la gestió de dades persistents i l'execució d'operacions com a consultes, insercions i actualitzacions.
- **Gestió de peticions i respostes HTTP:** poden manejar peticions HTTP de clients i enviar respostes adequades. Això és fonamental per a qualsevol aplicació web, on el servidor ha de respondre a sol·licituds de recursos o dades enviades des del navegador de l'usuari.

Res d'això era possible en JavaScript fins a l'aparició de Node.js. Este nou pas en el llenguatge li ha permés, per tant, conquistar també l'altre costat de la comunicació client-servidor per a les aplicacions web.

La **conseqüència** lògica d'esta evolució és que ara, utilitzant Node.js per al desenvolupament en el servidor, és possible desenvolupar una aplicació web completa amb un sol llenguatge: JavaScript. Abans que això fora possible, era indispensable conéixer, almenys, dos llenguatges: JavaScript per a la part de client i PHP, JSP, ASP.NET o un altre llenguatge per a la part del servidor.

2. Característiques principals de Node.js

Entre les principals característiques que oferix Node.js, podem destacar les següents:

- **API asíncrona i dirigida per esdeveniments:** Node.js oferix una API asíncrona, cosa que significa que no bloqueja l'execució del programa principal mentres espera respostes als seus mètodes. En lloc de detindre's, el programa continua executant-se i maneja la resposta quan esta es produïx, gràcies al seu model dirigit per esdeveniments. Això és crucial per al maneig eficient de múltiples operacions simultànies, com la lectura d'arxius o la connexió a bases de dades.
- **Execució de codi ràpida:** recordem que Node.js es recolza en el motor V8 de Google Chrome, implementat en C++, i que V8 compila JavaScript a codi màquina de manera eficient.
- **Model monofil escalable:** utilitza un únic fil per a atendre les peticions dels clients, a diferència d'altres servidors que permeten llançar múltiples fils en paral·lel. No obstant això, gràcies a la API asíncrona i dirigida per esdeveniments, Node.js pot atendre múltiples peticions amb eixe únic fil, consumint molts menys recursos que els sistemes multifil.
- **Eliminació de la necessitat de cross-browser:** s'elimina la necessitat de desenvolupar codi JavaScript compatible amb tots els navegadors, un desafiament comú en el desenrotllament del costat del client. En este cas, només hem de preocupar-nos que el nostre codi JavaScript siga correcte per a executar-se en el servidor.

2.1. Qui utilitza Node.js?

Hi ha diverses empreses, algunes d'elles d'un pes rellevant a nivell internacional, que utilitzen Node.js en el seu desenvolupament. Per posar alguns exemples representatius, podem citar a Netflix, PayPal, Uber o la NASA, entre altres. En el cas d'Espanya, grans empreses de desenvolupament de programari com Everis, Accenture o Indra, també sol·liciten personal qualificat en Node.js per a cobrir llocs de treball. A més, en webs de referència com *InfoJobs*, se solen trobar fins a 4 o 5 vegades més ofertes de treball relacionades amb Node.js que amb altres frameworks populars com Laravel o Symfony.

3. Instal·lació de Node.js

A l'hora d'instal·lar Node.js podem optar per dues alternatives:

- Gestionar la instal·lació a través d'una eina anomenada **NVM** (*Node Version Manager*), que ens permet instal·lar/desinstal·lar i activar diferents versions de Node
- Realitzar la instal·lació amb el propi instal·lador de Node

> **Versió requerida**: recomanable tindre instal·lada l'actual versió LTS (*Long Term Support*), que és la que més suport a llarg termini tindrà. Aquesta és la web oficial de Node.js, on podem trobar informació

> sobre la versió actual, i el calendari d'implantació de les successives versions LTS, que seran les que principalment ens interessen.

3.1. Instal·lació mitjançant NVM

3.1.1. Instal·lació en Linux i Mac

Per a instal·lar NVM, hem de descarregar-ho amb la comanda `curl` o `wget`, segons s'explica en la pròpia web oficial en GitHub. Si optem per `wget`, la comanda és com segueix (en una sola línia):

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
```

En el cas de no disposar de la comanda `wget` instal·lat, podem o bé instal·lar-ho, o bé emprar aquesta altra comanda equivalent, amb l'ordre `curl` (també en una sola línia):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
```

> **NOTA**: el número de versió `v0.40.0` pot variar. És preferible consultar la web de GitHub per a obtindre la comanda actualitzat.

> **NOTA**: després d'executar la comanda anterior, serà necessari tancar el terminal i tornar-lo a obrir per a poder utilitzar la comanda `nvm`.

Ja tenim `nvm` instal·lat en el sistema. Ací vam mostrar algunes de les opcions més interessants que podem utilitzar:

- `nvm install node`: instal·la l'última versió disponible de Node
- `nvm install --lts`: instal·la l'última versió LTS disponible (opció recomanada per a aquest curs)
- `nvm install 18.20.4`: instal·la la versió especificada de Node
- `nvm uninstall 18.20.4`: desinstal·la la versió especificada de Node
- `nvm ls-remote`: mostra totes les versions disponibles per a instal·lar
- `nvm list`: mostra totes les versions instal·lades localment
- `nvm current`: mostra la versió actualment activa
- `nvm use 18.20.4`: marca la versió indicada com actualment activa
- `nvm use --lts`: marca com a activa l'última versió LTS instal·lada

En el nostre cas, instal·larem l'última versió LTS disponible, per la qual cosa executarem la comanda:

```
nvm install --lts
```

3.1.2. Instal·lació en Windows

Per a instal·lar Node.js en Windows, no disposem del mateix gestor *nvm* que en Linux o Mac. Com a alternativa, existeix alguna implementació paral·lela de *nvm* que podem fer servir, com aquesta. Podem descarregar un instal·lador (*nvm-setup.zip*) i executar-ho per a instal·lar aquest gestor. Després, des de línia de comandes tindrem disponibles, entre altres, aquestes opcions:

- `nvm install 18.20.4` : instal·la la versió especificada de Node
- `nvm uninstall 18.20.4` : desinstal·la la versió especificada de Node
- `nvm list` : mostra totes les versions instal·lades localment
- `nvm list available` : mostra totes les versions disponibles per a instal·lar amb aquesta adaptació de NVM.
- `nvm use 18.20.4` : marca com a activa la versió de Node especificada (prèviament instal·lada).

3.2. Instal·lació des de la web de Node.js

També des de la pròpia web de Node tenim disponible un instal·lador per a Windows, que permet instal·lar el framework, encara que sense la possibilitat de tindre diferents versions coexistint, com ocorre amb NVM.

3.2.1. Instal·lació per a Windows i Mac

Per a Windows i Mac hem de fer clic sobre l'enllaç de descàrrega de la web oficial, que ja està preparat per al sistema operatiu que estem utilitzant. En la pàgina principal se'ns ofereixen dues versions alternatives i, com hem dit, el recomanable és emprar la versió LTS.

- Per a sistemes **Windows** el paquet és un instal·lador (arxiu *.msi*) que podem directament executar perquè s'instal·le. Acceptem l'acord de llicència, i confirmem cada pas de l'assistent amb els valors per defecte que apareguen.
- Per a sistemes **Mac US X**, el paquet és un arxiu *.pkg* que podem executar fent doble clic en ell, i seguir els passos de l'assistent com en Windows.

3.2.2. Instal·lació per a Linux

Si estem utilitzant un sistema Linux es recomana instal·lar Node.js des d'un repositori. Suposarem que utilitzem una distribució Debian (o Ubuntu, o similars). En aqueix cas podem escriure aquestes comandes en un terminal amb permisos de superusuario:

```
sudo apt-get update
sudo apt-get install -y curl
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -i nodejs
```

> **NOTA:** el número 20 haurem de substituir-lo per la versió de Node que vulguem instal·lar. En el cas de tindre la comanda `curl` ja prèviament instal·lat, només serà necessari executar les dues últimes instruccions.

3.2.3. Actualitzar des d'una versió prèvia

L'actualització des de versions prèvies és tan senzilla com descarregar el paquet de la nova versió i executar-lo. Automàticament se sobreescriurà la versió antiga amb la nova. En el cas de Linux, podem repetir la seqüència de comandes anterior canviant el paquet pel de la versió que siga.

3.3. Prova de la instal·lació

Una vegada instal·lat Node.js amb qualsevol dels mecanismes anteriorment exposats, podem comprovar que tot està correctament instal·lat amb la comanda `node -v` en el terminal, per a comprovar que ens mostra el número de versió adequat.

4. Visual Studio Code

Com IDE per a desenvolupar les nostres aplicacions emprarem **Visual Studio Code**, que és un dels IDEs més versàtils que existeixen hui dia per a desenvolupament web.

> **Versió requerida**: cap en particular, serveix amb l'última versió disponible.

Des de la web oficial de Visual Studio Code podem descarregar-ho per a la plataforma desitjada.

Linux (Debian)

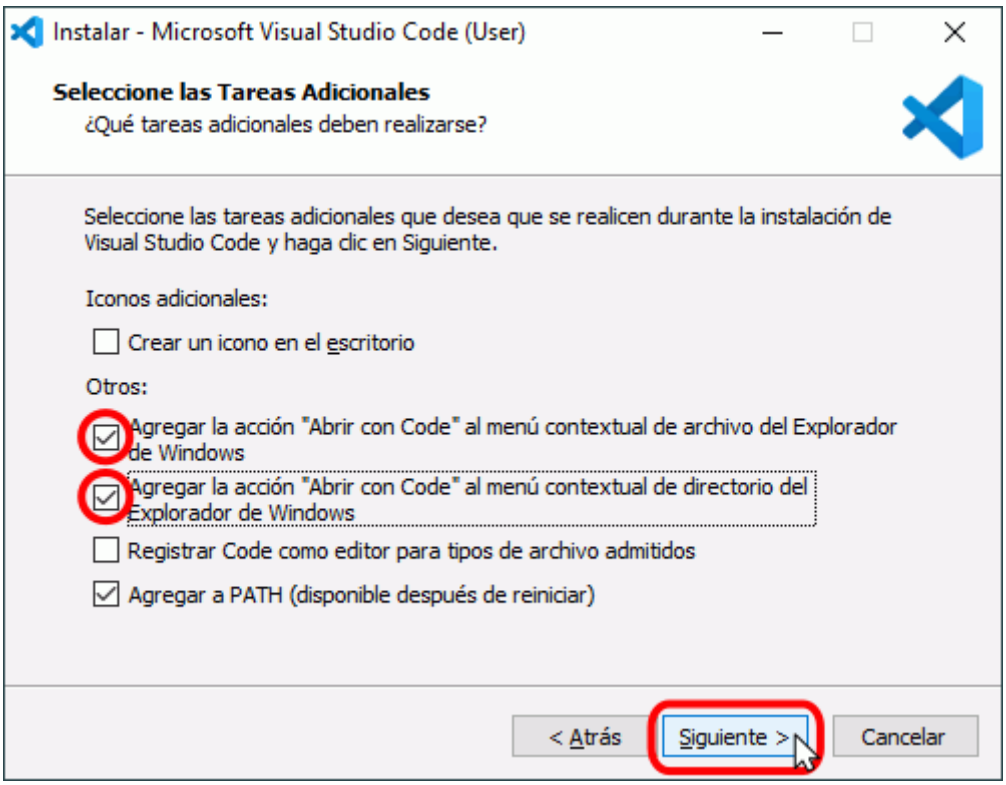
En el cas de Debian, Ubuntu o una distribució similar, descarregarem un arxiu *.deb*. Una vegada descarregat, accedim per terminal a la carpeta on estiga i executem aquesta comanda per a instal·lar-lo:

```
sudo dpkg -i nom_arxiu.deb
```

Es crearà automàticament un accés directe en el menú d'inici, dins de la secció de *Programació* en el cas de Lubuntu.

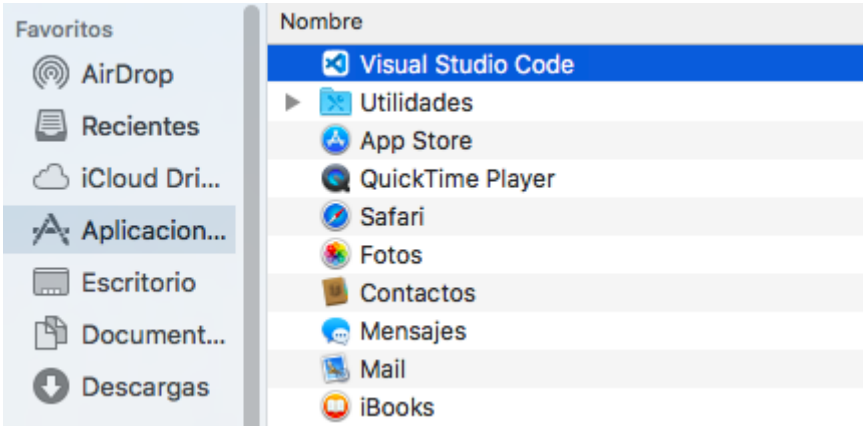
Windows

Per a Windows descarreguem l'instal·lador i seguim els passos. No hi ha molt a configurar; en tot cas, podem deixar marcada la casella per a afegir el menú contextual "*Obrir amb Code*" per a poder obrir arxius i carpetes amb VS Code des de l'explorador d'arxius directament, amb un clic dret.



Mac OSX

Per a **Mac OSX**, descarreguem l'aplicació i la podem executar directament. També podem moure-la a la carpeta de *Aplicacions* per a tindre-la localitzada.



5. Integració de Node.js i Visual Studio Code

Cada projecte Node que fem anirà contingut en la seua pròpia carpeta i, d'altra banda, Visual Studio Code i altres editors similars que puguem utilitzar treballen per carpetes (és a dir, els indiquem quina carpeta obrir i ens permeten gestionar tots els arxius d'aqueixa carpeta). Per tant, i per a centralitzar d'alguna forma tot el treball del curs, el primer que farem serà crear una carpeta anomenada "**ProjectesNode**" en el nostre espai de treball (per exemple, en la nostra carpeta personal). Dins d'aquesta carpeta, crearem dues subcarpetas:

- **Proves**, on guardarem tots els projectes de prova i exemple que fem durant les sessions.
- **Exercicis**, on emmagatzemarem els exercicis proposats de cada sessió per a entregar.

L'estructura de carpetes quedarà llavors com segueix:

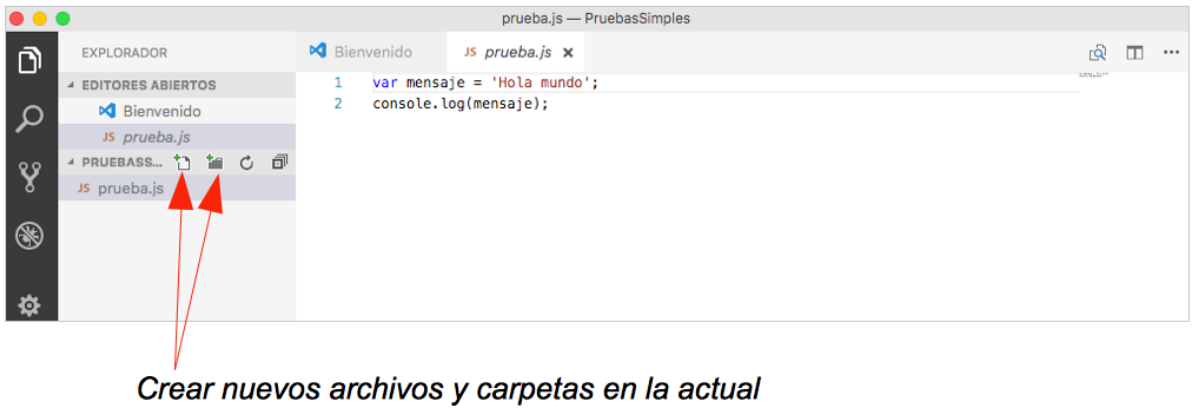
- ProjectesNode
  - Proves
  - Exercicis

5.1. Crear i editar un projecte bàsic

Dins de la carpeta *ProjectesNode/Proves*, crearem una altra subcarpeta anomenada "*ProvesSimples*" on definirem xicotets arxius per a provar alguns conceptes bàsics, especialment en les primeres sessions.

Una vegada creada la carpeta, obrim Visual Studio Code i anem al menú Arxiu > Obrir carpeta (o Arxiu > Obrir..., depenent de la versió de Visual Studio Code que tinguem). Triem la carpeta "*ProvesSimples*" dins de *ProjectesNode/Proves* i s'obrirà en l'editor. També podem obrir la carpeta arrossegant-la des d'algun explorador de carpetes fins a una instància oberta de Visual Studio Code.

De moment la carpeta està buida, però des del panell esquerre podem crear nous arxius i carpetes. Per a començar, crearem un arxiu "prova.js" amb el codi que es mostra a continuació:



5.2. Executar un arxiu Node des de Visual Studio Code

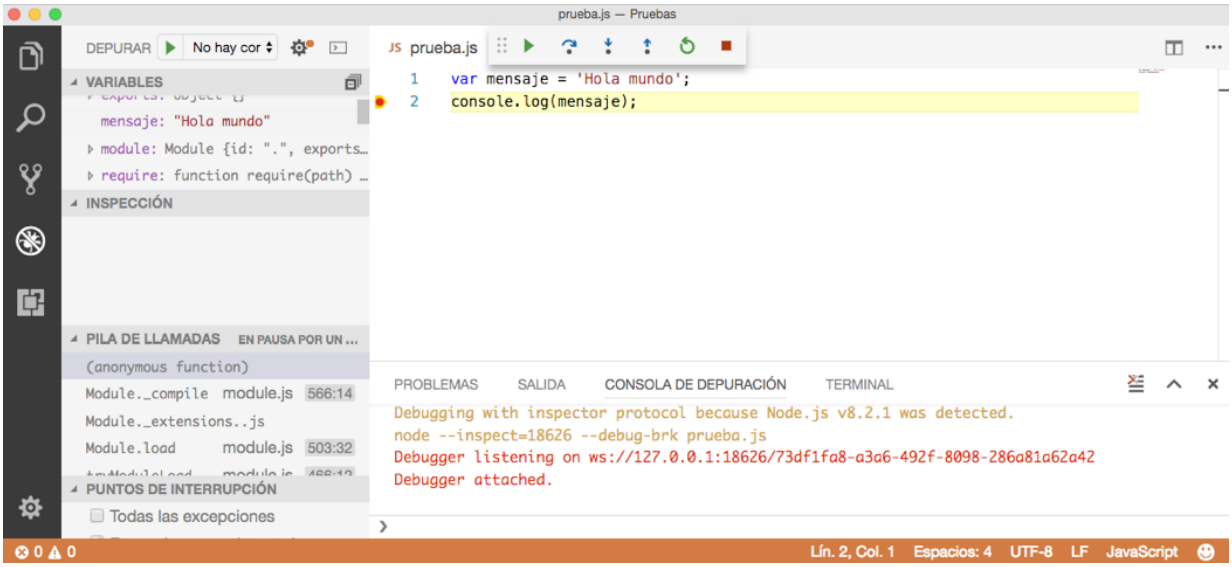
Visual Studio Code compta amb un terminal incorporat, que podem activar anant al menú *Veure* > *Terminal*. Apareixerà en un panell en la zona inferior. Observem com automàticament dit terminal se situa en la carpeta del nostre projecte actual, per la qual cosa podem directament escriure `node prova.js` en ell i s'executarà l'arxiu, mostrant el resultat en aquest terminal:

> **Exercici 1**:
>
> Crea l'estructura de carpetes indicada anteriorment, i crea l'arxiu `prova.js` en la carpeta *ProjectesNode/Proves/ProvesSimples*. Escriu el codi per a mostrar el missatge "Hola món" per consola, i prova a executar-ho des del terminal integrat de VS Code.

5.3. Depuració de codi

Visual Studio Code ofereix un depurador per a les nostres aplicacions Node. Per a entrar en manera depuració, fem clic en la icona de depuració del panell esquerre (el que té forma de bug o xinxa).

Podem establir *breakpoints* en el nostre codi fent clic en la línia en qüestió, en el marge esquerre (com en molts altres editors de codi) o afegint-hi la instrucció `debugger;` a la línia en qüestió. Després, podem iniciar la depuració amb `F5`, o bé amb el menú *Depurar* > *Iniciar depuració*, o amb la icona de la fletxa blava de play de la barra superior. En arribar a un breakpoint, podem analitzar en el panell esquerre els valors de les variables i l'estat de l'aplicació.



També podem continuar fins al següent breakpoint (botó de fletxa blava), o executar pas a pas amb la resta de botons de la barra de depuració. Per a finalitzar la depuració, fem clic en la icona del quadrat roig de stop (si no s'ha detingut ja l'aplicació), i tornem a la manera d'edició fent clic en el botó d'explorador del panell esquerre.

La "consola de depuració" que apareix en el terminal inferior realment és un terminal REPL (*Read Eval Print Loop*), cosa que significa que des d'ella podem accedir als elements del nostre programa (variables, objectes, funcions) i obtindre el seu valor o cridar-los. Per exemple, en el cas anterior, podríem teclejar "missatge" en el terminal i veure quant val aqueixa variable:

> **Exercici 2**:
>
> Crea una carpeta anomenada "**Exercici_Depuracio**" en la carpeta "*ProjectesNode/Exercicis*", obri-la amb Visual Studio Code i crea un arxiu anomenat `depuracio.js`. Dins, introdueix aquest codi i guarda l'arxiu:

```
var m = 1, n = 2;

for(i = 1; i <= 5; i++) {
    m = m * i;
    n = n + m * n;
}

console.log(n);
```

> Utilitza el depurador per a esbrinar el valor de la variable `n` després d'executar-se la línia 6 de codi (posa un *breakpoint* en la línia 8, per exemple).