

NODE JS I

Ing. Henry Alberto Hernández Martínez MSc.

hahernandezm@udistrital.edu.co

Ingeniería en telecomunicaciones.

Universidad Distrital Francisco José de Caldas.



CONTENIDO

- Introducción
- Descripción de NODE JS
- Módulos
- Servidor HTTP



INTRODUCCIÓN



INTRODUCCIÓN

- **Node.js** es un entorno de tiempo de ejecución de JavaScript. Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.



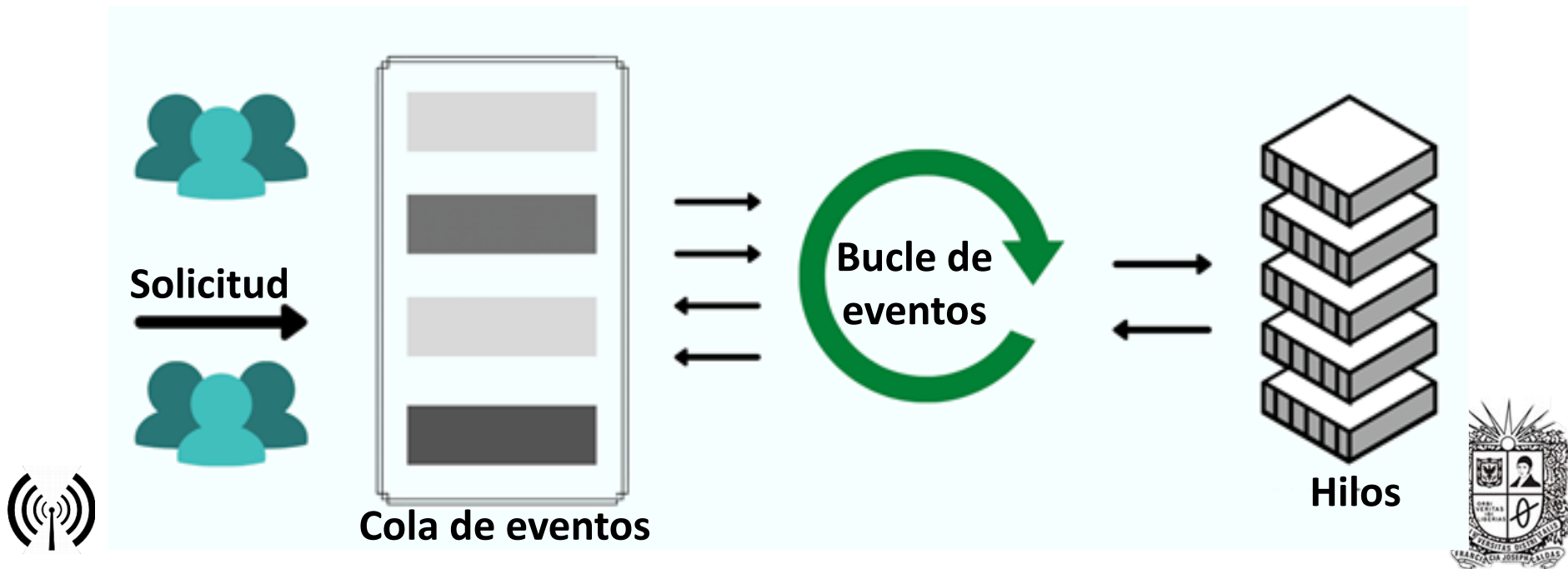
INTRODUCCIÓN

- **Node.js** es un entorno de tiempo de ejecución de JavaScript. Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.
- **JavaScript** *es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web.*



INTRODUCCIÓN

- Incorpora un modelo de entrada y salida sin bloqueo controlado por eventos que pueden ser leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP.



INTRODUCCIÓN

- El modelo de entrada y salida usando eventos ayuda al **manejo simultáneo de peticiones**.
- El administrador y el usuario incorporan estrategias de codificación similares.
- Con la implementación de plataformas de desarrollo de software como GitHub Inc., la **comunidad Node.js ha crecido** de forma exponencial y activa.



INTRODUCCIÓN

- Descargue e instale nodejs de la pagina (versión actual).



The screenshot shows the Node.js website's download section. The 'Actual' (Current) version is highlighted with a red dashed box. The 'LTS' (Long Term Support) version is also visible but not highlighted.

LTS	Actual	
Recomendado para la mayoría	Últimas características	
 Instalador Windows node-v17.0.1-x64.msi	 Instalador macOS node-v17.0.1.pkg	 Código Fuente node-v17.0.1.tar.gz

<https://nodejs.org/es/download/current/>



INTRODUCCIÓN

- Abra el terminal de windows y digite **node -v** y **npm -v** para verificar que la aplicación se haya instalado correctamente.

```
C:\Users\Hagui>node -v  
v17.0.1  
  
C:\Users\Hagui>npm -v  
8.1.0
```



INTRODUCCIÓN

- Abra el terminal de windows y digite **node -v** y **npm -v** para verificar que la aplicación se haya instalado correctamente.

```
C:\Users\Hagui>node -v
v17.0.1

C:\Users\Hagui>npm -v
8.1.0
```

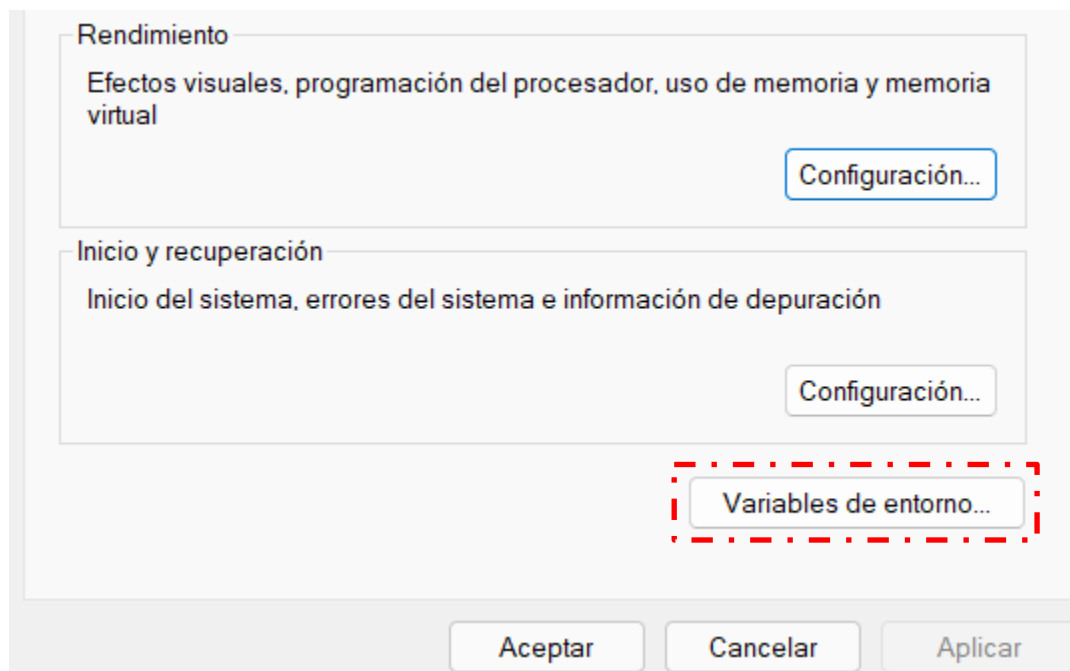
- Luego digite la palabra node y ejecute el comando `console.log('hola mundo')`

```
C:\Users\Hagui>node
Welcome to Node.js v17.0.1.
Type ".help" for more information.
> console.log('Hola mundo')
Hola mundo
undefined
```



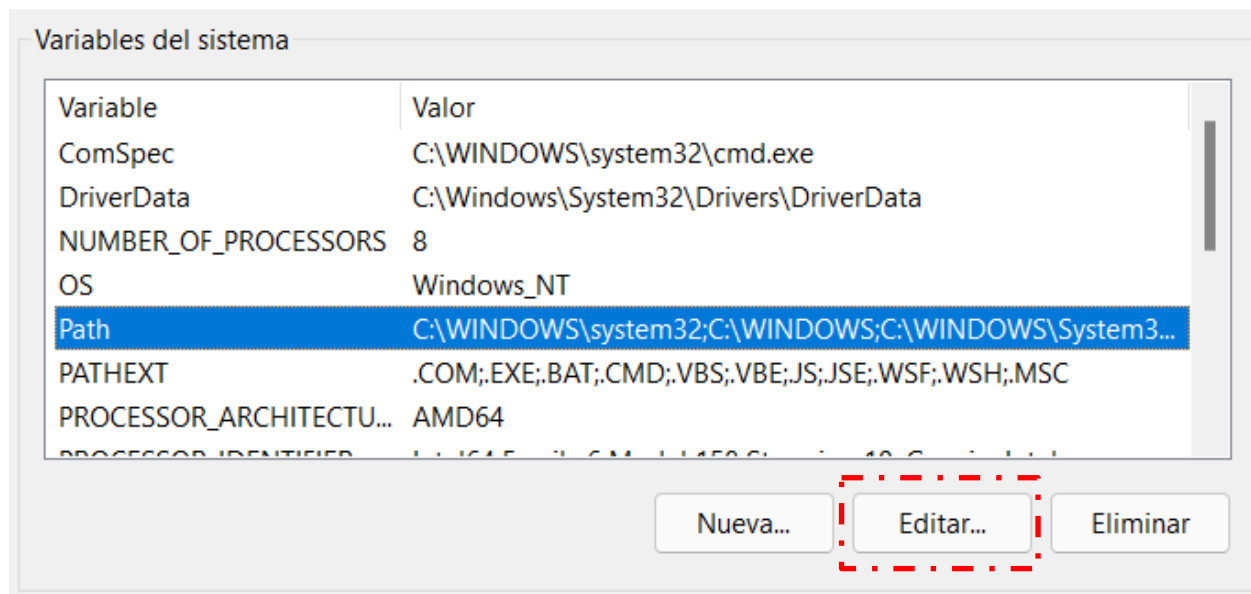
INTRODUCCIÓN

- Pulse ctrl+c para cerrar la consola y cierre la terminal de windows.
- Abra propiedades del sistema y haga clic en variables del entorno.

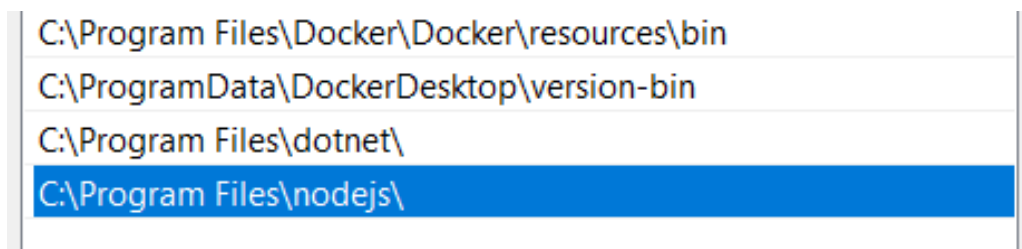


INTRODUCCIÓN

- Busque la opción path y haga clic en editar.



- Verifique que exista nodejs.

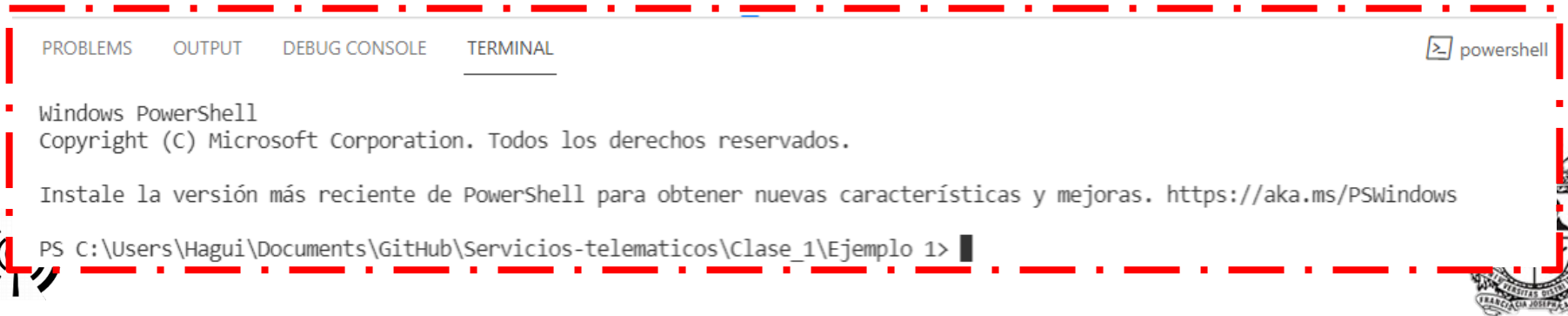


EJEMPLO 1



EJEMPLO 1

- Cree una carpeta nueva (en su repositorio), abra VS CODE y fije la ruta de la carpeta.
- Luego abra el terminal y cree un nuevo archivo que se llame Ejemplo1.js



EJEMPLO 1

- Digite en la primer línea el comando mostrado.

```
JS Ejemplo1.js ●  
JS Ejemplo1.js  
1 //Instrucciones  
2 console.log("Hola mundo node js")
```

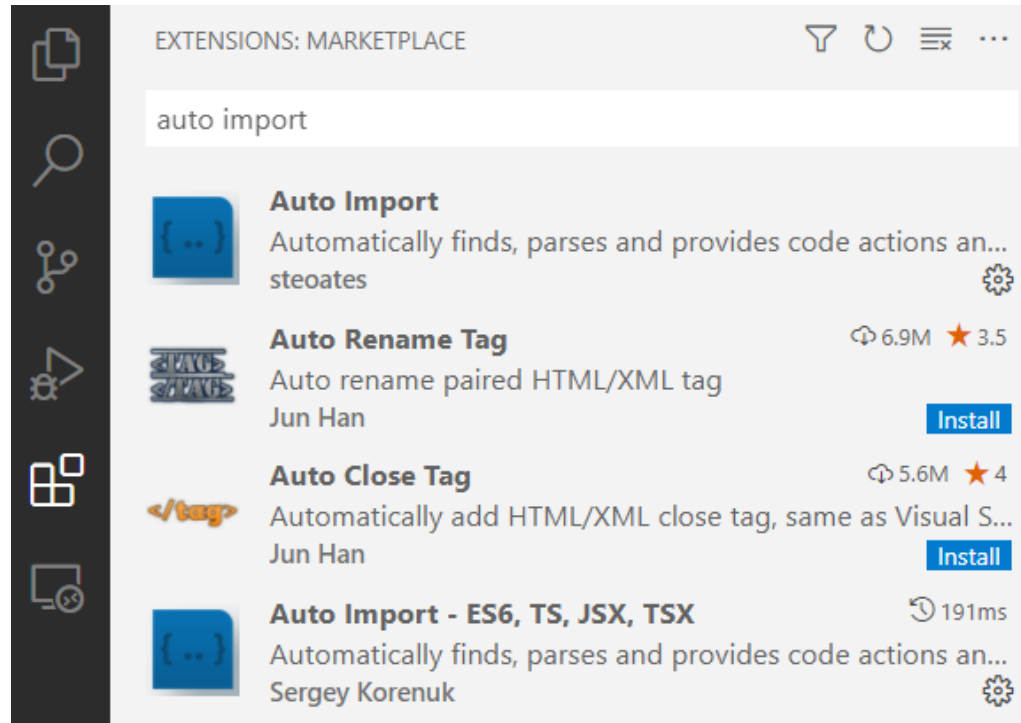
- Ahora digite en el terminal node Ejemplo1.js

```
PS C:\Users\Hagui\Documents\GitHub\Servicios-telematicos\Clase_1\Ejemplo 1> node ./Ejemplo1.js  
Hola mundo node js  
PS C:\Users\Hagui\Documents\GitHub\Servicios-telematicos\Clase_1\Ejemplo 1> █
```



EJEMPLO 1

- En la opción “Extensions” instale los complementos auto import y auto import – ES6, TS, JSX, TSX

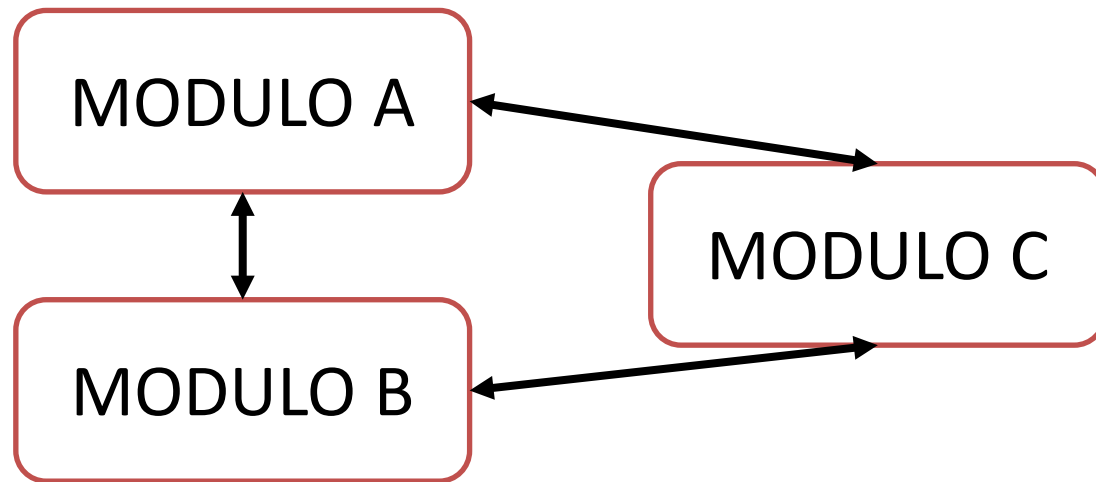


MÓDULOS



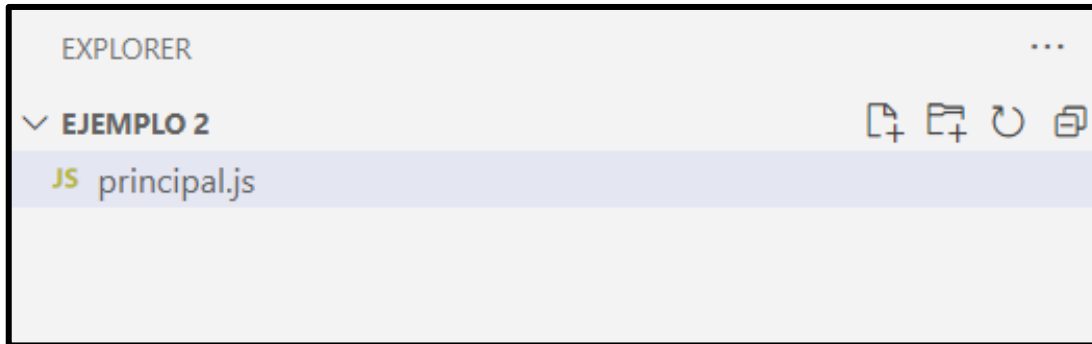
MÓDULOS

- Los módulos le permiten a un usuario comunicar scripts en node.js, ya que, el código desarrollado puede presentarse en más de un archivo (acción importa-exporta).



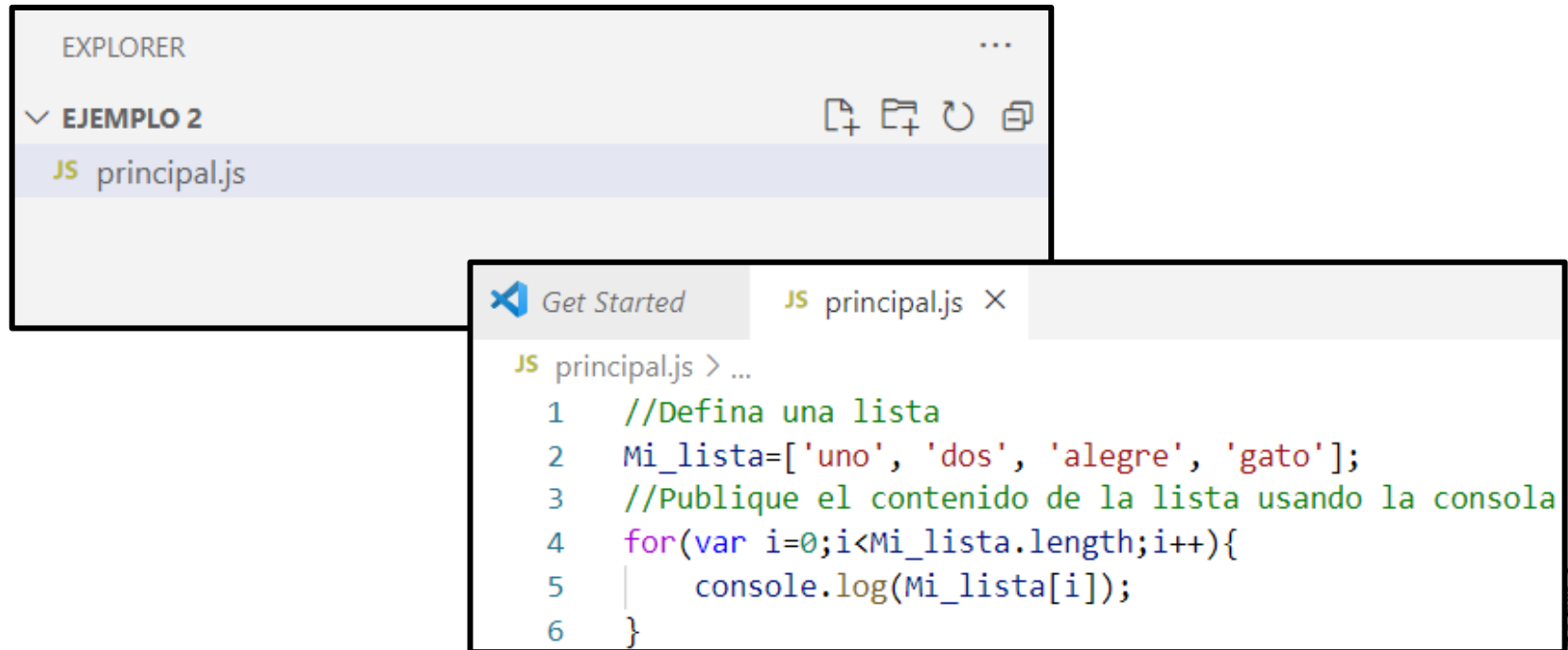
MÓDULOS

- Cree una carpeta nueva que se llame Ejemplo 2 y dentro de ella un archivo que se llame principal.js.



MÓDULOS

- Publique el contenido de una lista usando el terminal.



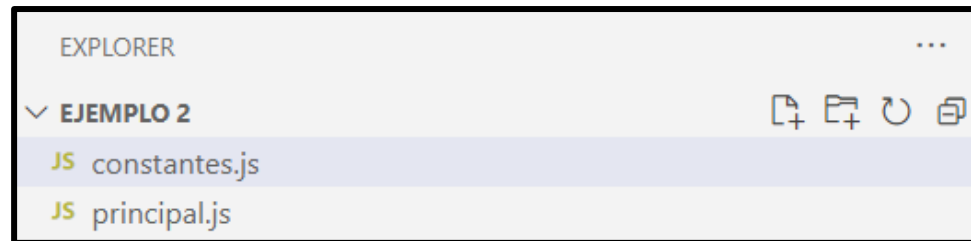
```
EXPLORER
▼ EJEMPLO 2
  JS principal.js

JS principal.js > ...
1  //Defina una lista
2  Mi_lista=['uno', 'dos', 'alegre', 'gato'];
3  //Publique el contenido de la lista usando la consola
4  for(var i=0;i<Mi_lista.length;i++){
5      console.log(Mi_lista[i]);
6  }
```



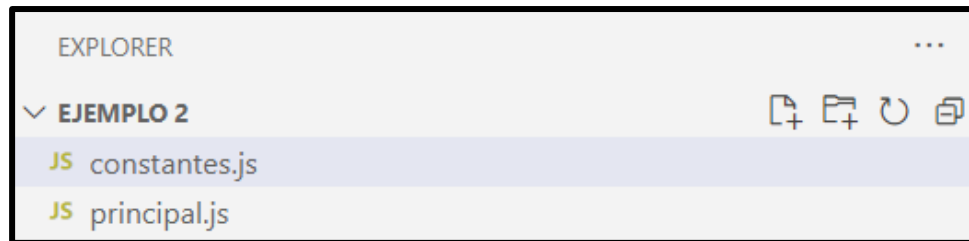
MÓDULOS

- Ahora cree un nuevo archivo que se llame constantes.js en el mismo directorio.

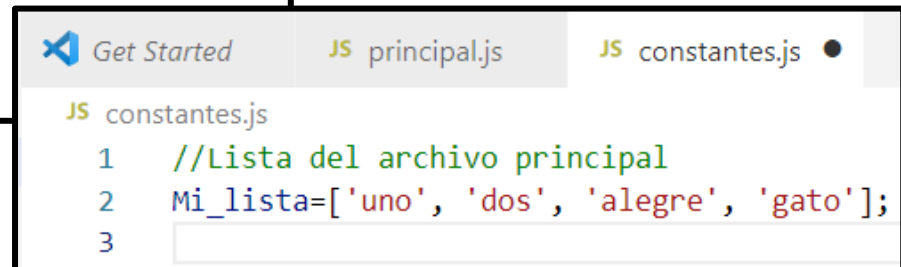
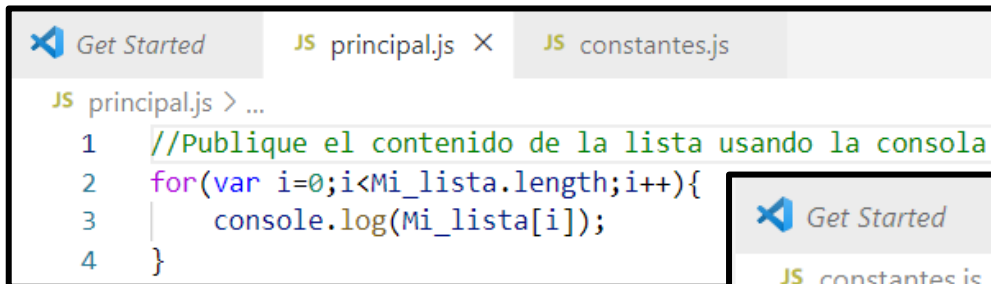


MÓDULOS

- Ahora cree un nuevo archivo que se llame constantes.js en el mismo directorio.

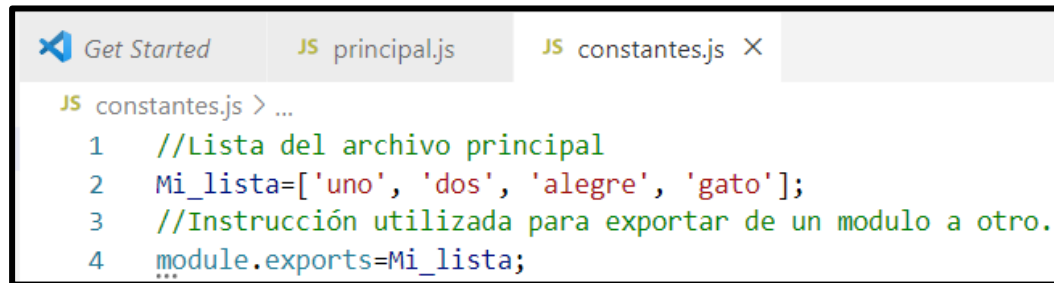


- Luego pase la lista del archivo principal al archivo constantes.



MÓDULOS

- Agregue la instrucción `module.export` en el archivo constantes y seleccione las variables a exportar.

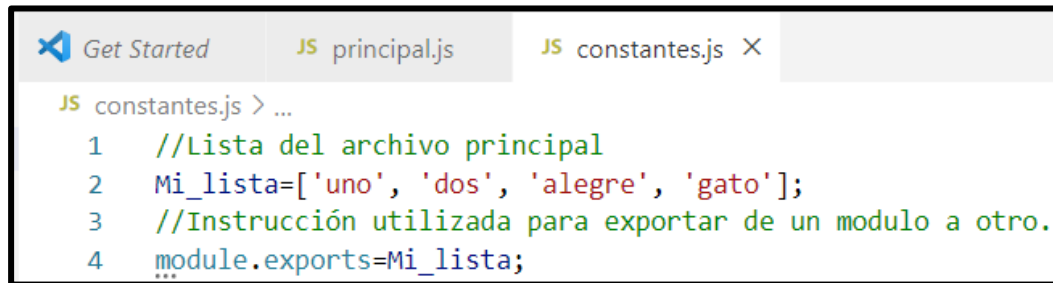


```
JS constantes.js > ...
1 //Lista del archivo principal
2 Mi_lista=['uno', 'dos', 'alegre', 'gato'];
3 //Instrucción utilizada para exportar de un modulo a otro.
4 module.exports=Mi_lista;
```



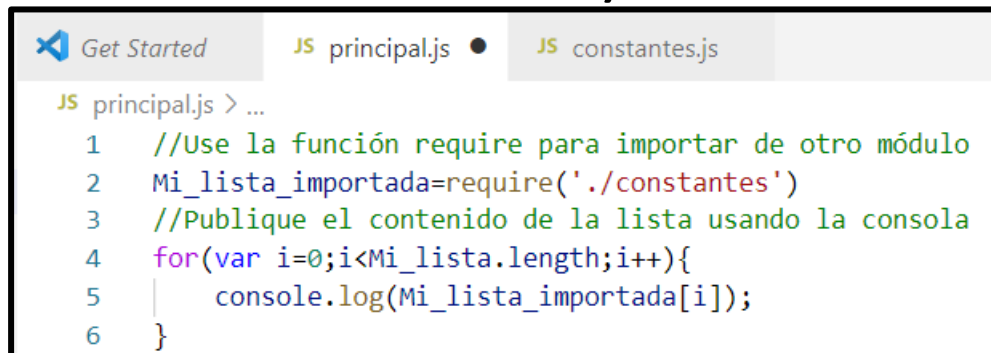
MÓDULOS

- Agregue la instrucción `module.export` en el archivo `constantes` y seleccione las variables a exportar.



```
Get Started JS principal.js JS constantes.js X
JS constantes.js > ...
1 //Lista del archivo principal
2 Mi_lista=['uno', 'dos', 'alegre', 'gato'];
3 //Instrucción utilizada para exportar de un modulo a otro.
4 module.exports=Mi_lista;
```

- Ahora en el archivo principal pruebe la opción `require`("ruta del archivo")

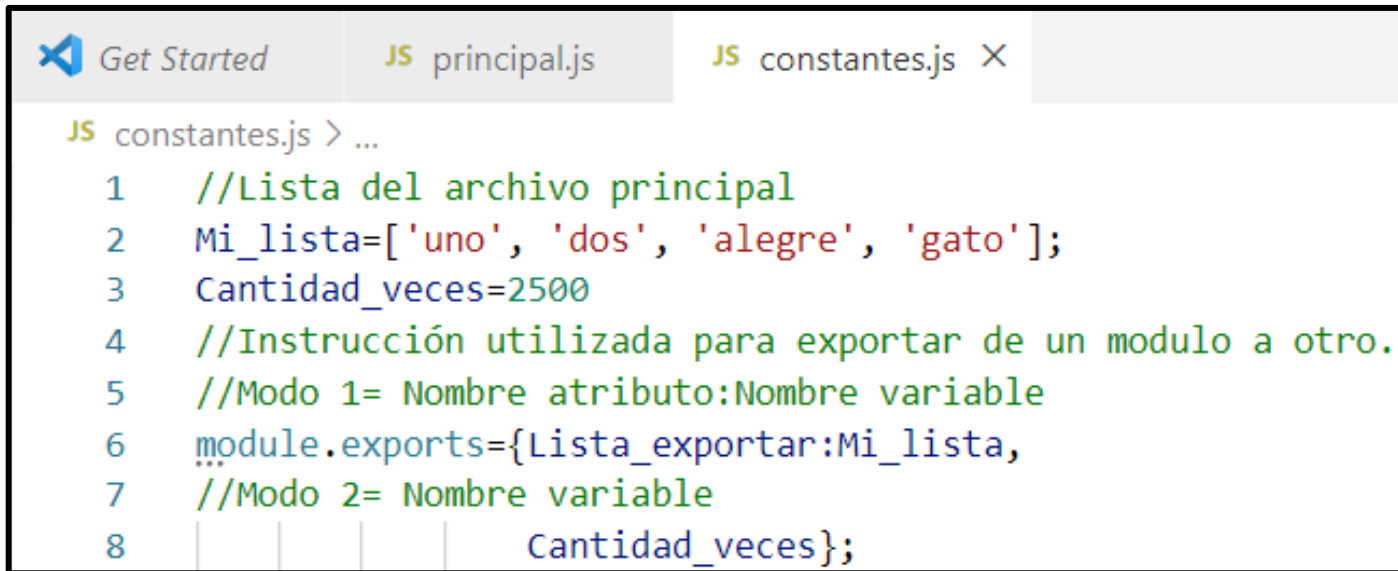


```
Get Started JS principal.js JS constantes.js
JS principal.js > ...
1 //Use la función require para importar de otro módulo
2 Mi_lista_importada=require('./constantes')
3 //Publique el contenido de la lista usando la consola
4 for(var i=0;i<Mi_lista.length;i++){
5     console.log(Mi_lista_importada[i]);
6 }
```



MÓDULOS

- En caso de que requiera exportar más de una variable del módulo use {Nombre_del_atributo:Variable, simplemente el nombre de la variable}

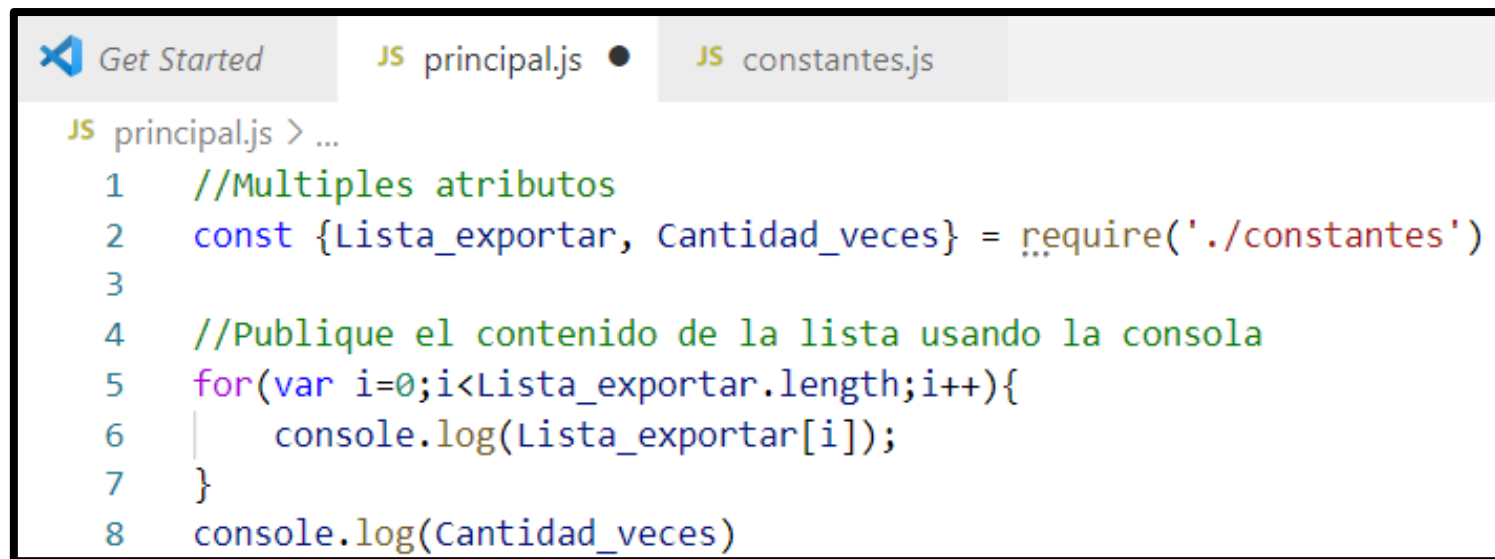


```
Get Started JS principal.js JS constantes.js X
JS constantes.js > ...
1 //Lista del archivo principal
2 Mi_lista=['uno', 'dos', 'alegre', 'gato'];
3 Cantidad_veces=2500
4 //Instrucción utilizada para exportar de un modulo a otro.
5 //Modo 1= Nombre atributo:Nombre variable
6 module.exports={Lista_exportar:Mi_lista,
7 //Modo 2= Nombre variable
8 Cantidad_veces};
```



MÓDULOS

- Ahora importar se realiza de forma similar solo se agrega la palabra reservada `const` de la siguiente forma.



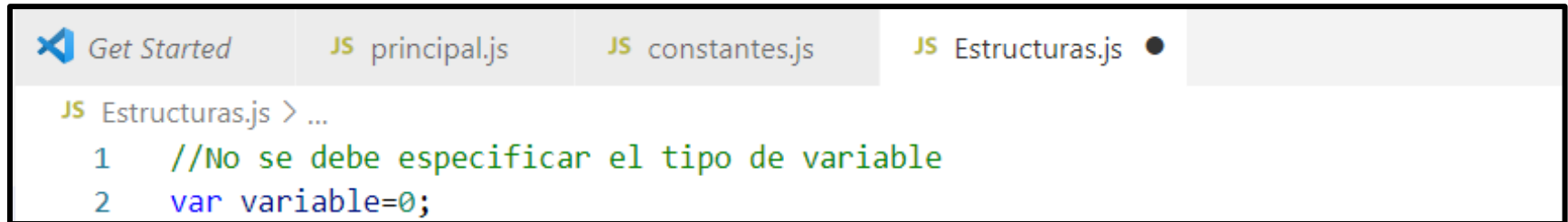
The screenshot shows a code editor with two tabs: 'principal.js' (active) and 'constantes.js'. The code in 'principal.js' demonstrates how to import a module using the `const` keyword. It includes comments in Spanish explaining the steps: importing multiple attributes, publishing the content to the console, and logging the quantity.

```
JS principal.js > ...
1  //Multiples atributos
2  const {Lista_exportar, Cantidad_veces} = require('./constantes')
3
4  //Publique el contenido de la lista usando la consola
5  for(var i=0;i<Lista_exportar.length;i++){
6      console.log(Lista_exportar[i]);
7  }
8  console.log(Cantidad_veces)
```



MÓDULOS

- Algunas estructuras son.



The screenshot shows a code editor with four tabs at the top: 'Get Started', 'JS principal.js', 'JS constantes.js', and 'JS Estructuras.js'. The 'JS Estructuras.js' tab is active and shows the following code:

```
JS Estructuras.js > ...  
1 //No se debe especificar el tipo de variable  
2 var variable=0;
```



MÓDULOS

- Algunas estructuras son.

```
Get Started JS principal.js JS constantes.js JS Estructuras.js ●
JS Estructuras.js > ...
1 //No se debe especificar el tipo de variable
2 var variable=0;
3 //Estructuras de control de flujo
4 //Condicionales if
5 if(variable==1){console.log('Se cumple la condición');}
6 else{console.log('No se cumple la condición');}
```



MÓDULOS

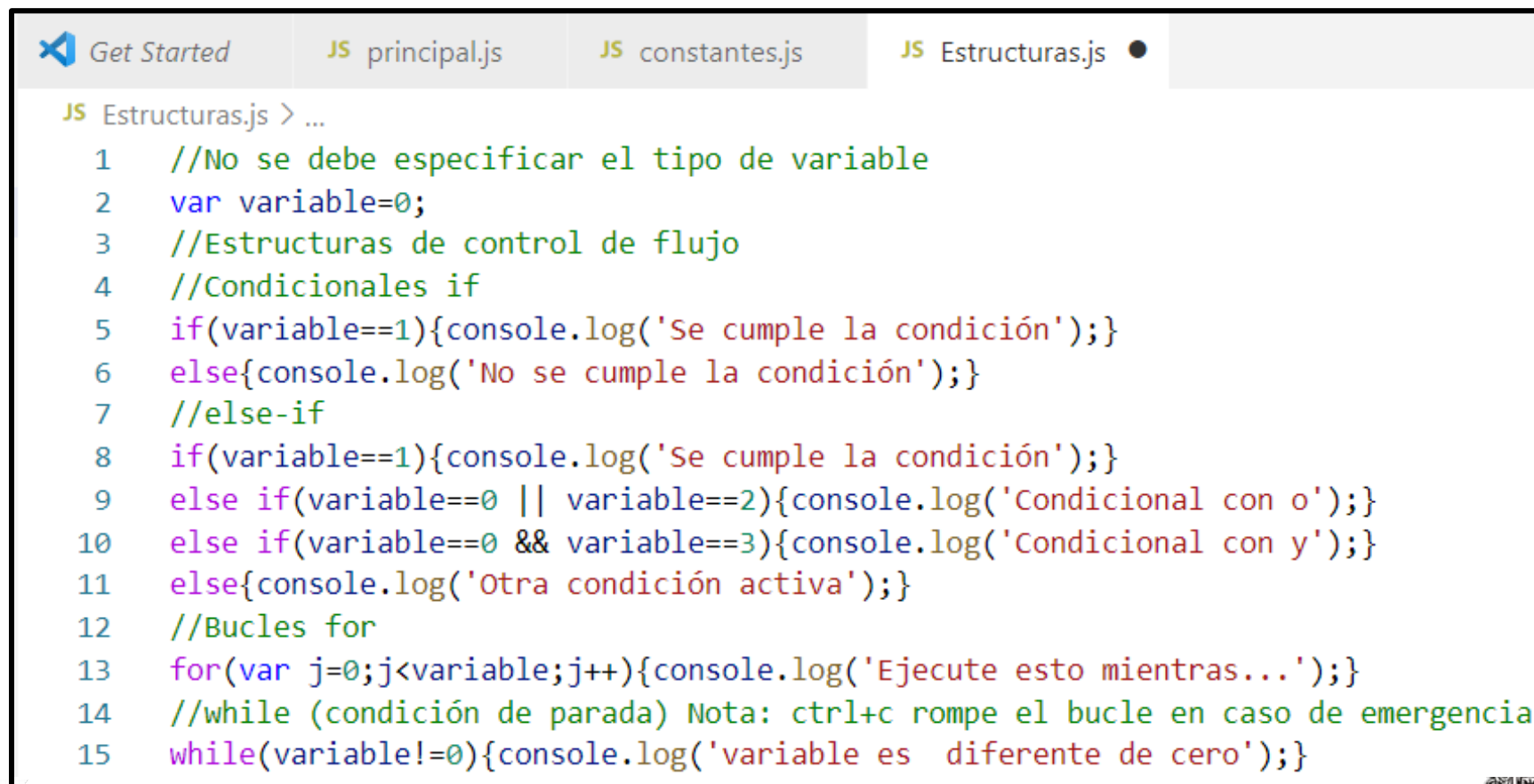
- Algunas estructuras son.

```
Get Started JS principal.js JS constantes.js JS Estructuras.js ●
JS Estructuras.js > ...
1 //No se debe especificar el tipo de variable
2 var variable=0;
3 //Estructuras de control de flujo
4 //Condicionales if
5 if(variable==1){console.log('Se cumple la condición');}
6 else{console.log('No se cumple la condición');}
7 //else-if
8 if(variable==1){console.log('Se cumple la condición');}
9 else if(variable==0 || variable==2){console.log('Condicional con o');}
10 else if(variable==0 && variable==3){console.log('Condicional con y');}
11 else{console.log('Otra condición activa');}
```



MÓDULOS

- Algunas estructuras son.



The screenshot shows a code editor with four tabs: 'Get Started', 'JS principal.js', 'JS constantes.js', and 'JS Estructuras.js' (which is active). The code in the active tab is as follows:

```
JS Estructuras.js > ...
1  //No se debe especificar el tipo de variable
2  var variable=0;
3  //Estructuras de control de flujo
4  //Condicionales if
5  if(variable==1){console.log('Se cumple la condición');}
6  else{console.log('No se cumple la condición');}
7  //else-if
8  if(variable==1){console.log('Se cumple la condición');}
9  else if(variable==0 || variable==2){console.log('Condicional con o');}
10 else if(variable==0 && variable==3){console.log('Condicional con y');}
11 else{console.log('Otra condición activa');}
12 //Bucles for
13 for(var j=0;j<variable;j++){console.log('Ejecute esto mientras...');}
14 //while (condición de parada) Nota: ctrl+c rompe el bucle en caso de emergencia
15 while(variable!=0){console.log('variable es diferente de cero');}
```



EJERCICIO 1



EJERCICIO 1

- En una aplicación hay dos módulos que almacenan información de 4 usuarios como la edad, la cantidad de veces que sale al día y el tiempo de actividad física. Realice un programa que de acuerdo al nombre de usuario presente su información relacionada.

```
//Algunas funciones de interes
module.exports={atributo_1:Nombre_variable, atributo_2}
const{atributo_1, atributo_2}=require('./ruta+Nombre archivo')
var Nombre_lista=['ele 1', 'ele 2', '...'];
if(condicion){console.log('tal cosa');}
for(var i=0; i<Nombre_lista.length; i++){console.log('tal cosa ...');}
```



NODE PACKAGE MANAGER –NPM–



-NPM-

- Es un repositorio utilizado para la publicación de proyectos Node.js de código abierto. Es decir, es una plataforma en línea donde cualquiera puede publicar y compartir herramientas escritas en JavaScript.



-NPM-

- Es un repositorio utilizado para la publicación de proyectos Node.js de código abierto. Es decir, es una plataforma en línea donde cualquiera puede publicar y compartir herramientas escritas en JavaScript.
- El proyecto debe contener un archivo llamado **package.json** para usar el NPM. Dentro de ese archivo se encuentran **metadatos** específicos para los proyectos.



-NPM-

- Algunos metadatos muestran características tales como.
 - El nombre del proyecto
 - La versión inicial
 - Descripción
 - El punto de entrada
 - Comandos de prueba
 - El repositorio [git](#)

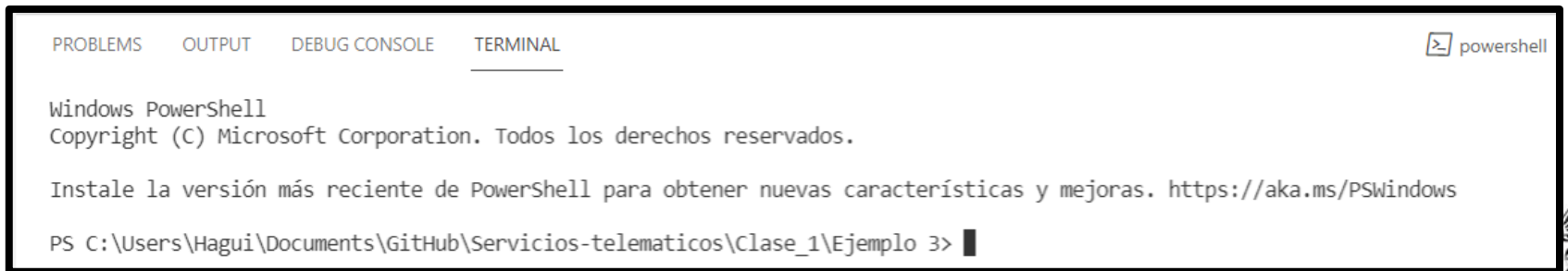
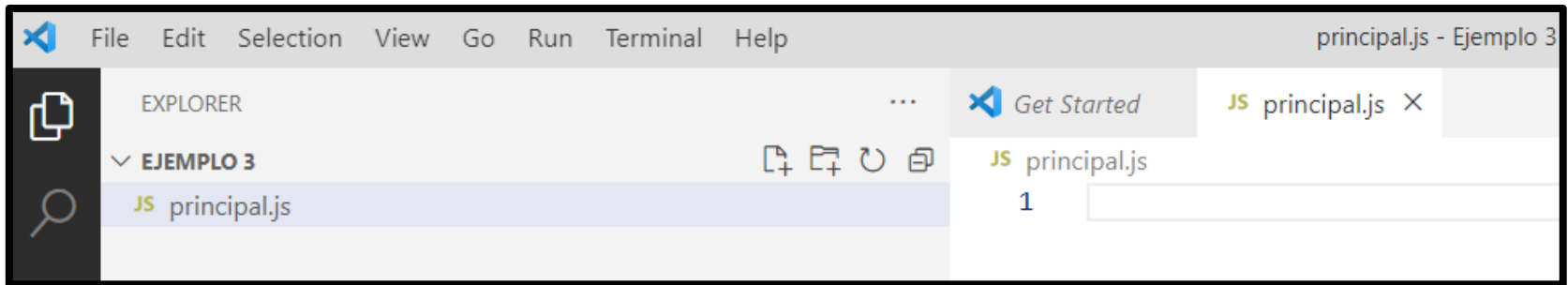


EJEMPLO 3



EJEMPLO 3

- Cree una carpeta que se llame Ejemplo 3 y dentro de ella un archivo principal.js e inicie la terminal.



EJEMPLO 3

- Cree una carpeta que se llame Ejemplo 3 y dentro de ella un archivo principal.js e inicie la terminal.
- Digite en el terminal el comando “npm init -y” y observe que se creará un nuevo archivo.



EJEMPLO 3

- Contenido del archivo.

```
{ } package.json > ...
```

```
1  { CARACTERÍSTICAS DEL PROYECTO
2    "name": "ejemplo-3",
3    "version": "1.0.0",
4    "description": "",
5    "main": "principal.js",
```

▷ Debug

PROGRAMA A EJECUTAR

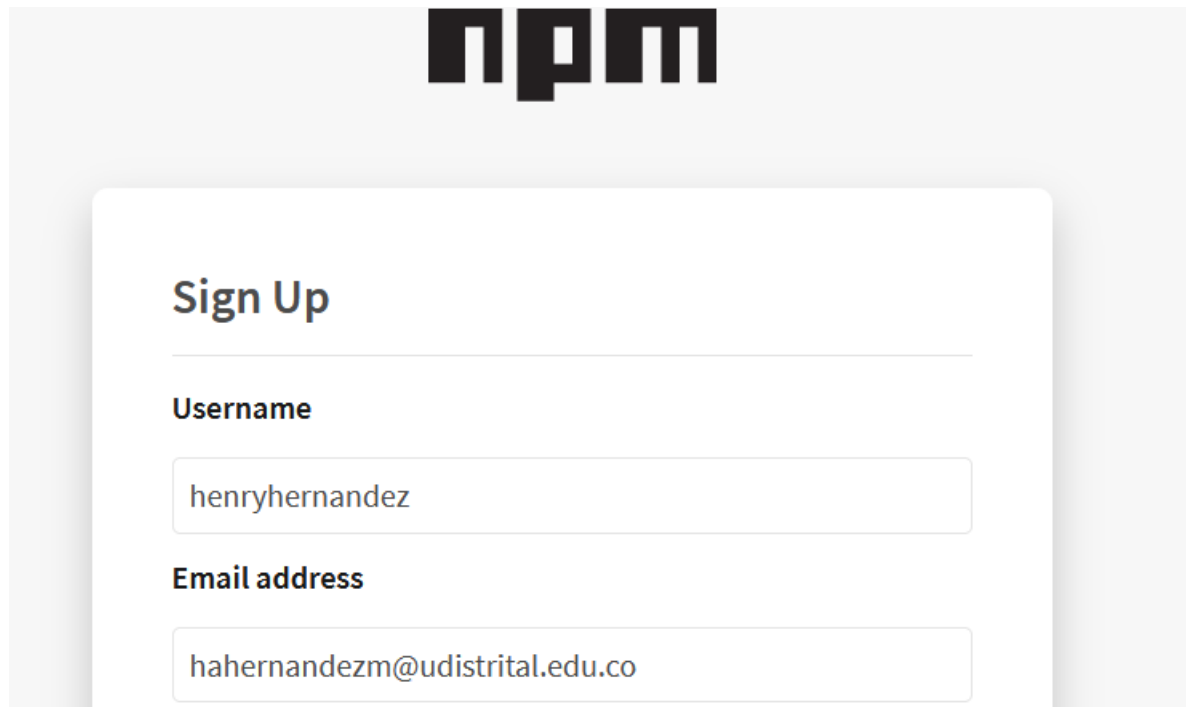
```
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC"
12 }
```

OTRAS CARACTERÍSTICAS



EJEMPLO 3

- Ingrese al repositorio npm <https://www.npmjs.com> y regístrese.

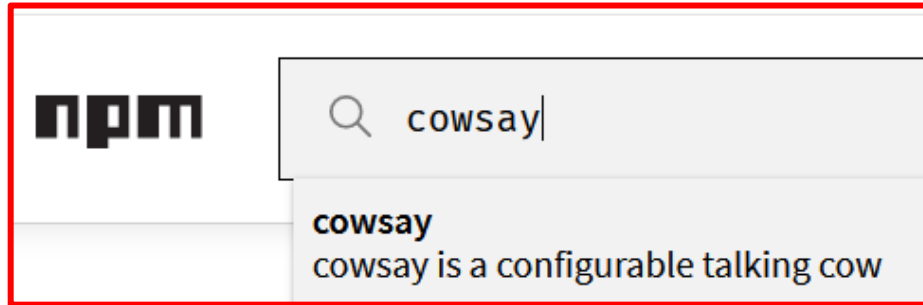


The image shows a screenshot of the npm website's sign-up form. At the top, the 'npm' logo is displayed in a large, bold, black font. Below the logo, the text 'Sign Up' is centered. Underneath, there are two input fields. The first is labeled 'Username' and contains the text 'henryhernandez'. The second is labeled 'Email address' and contains the text 'hahernandezm@udistrital.edu.co'.



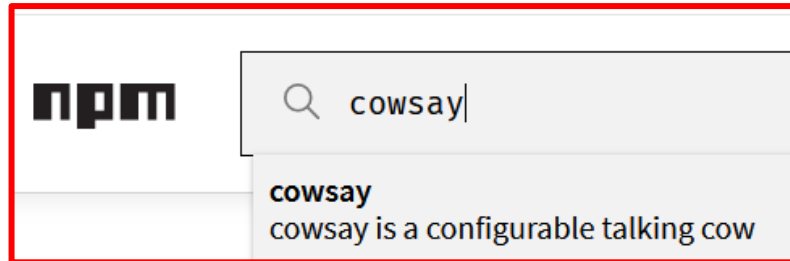
EJEMPLO 3

- Busque el repositorio cowsay.

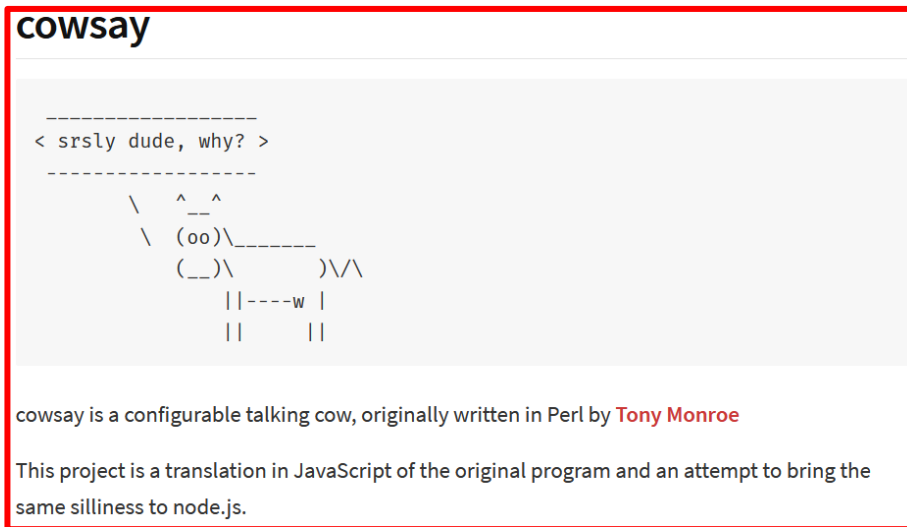


EJEMPLO 3

- Busque el repositorio cowsay.



- Como se observa el repositorio contiene la documentación del proyecto.



EJEMPLO 3

- Hay dos formas de instalación de un modulo.
 - La primera “npm install <paquete>” que es una instalación local.
 - La segunda “npm install -g <paquete>” que es una instalación de forma global (PC).



EJEMPLO 3

- Hay dos formas de instalación de un modulo.
 - En el terminal digite npm install cowsay.

```
PS C:\Users\Hagui\Documents\GitHub\Servicios-telematicos\Clase_1\Ejemplo 3> npm install cowsay
added 41 packages, and audited 42 packages in 5s
3 packages are looking for funding
  run `npm fund` for details
```

- En el archivo principal.js importe el paquete y ejecute una rutina del repositorio.

```
var cowsay = require("cowsay");

console.log(cowsay.say({
  text : "I'm a moooodule",
  e : "oO",
  T : "U "
}));
```

```
JS principal.js > ...
1 //Forma de importar el paquete
2 var cowsay = require("cowsay");
3 //Comando propio del repositorio
4 console.log(cowsay.say({
5   text : "I'm a student",
6   e : "oO",
7   T : "U "
8 }));
```



EJEMPLO 3

- Hay dos formas de instalación de un modulo.
 - En el terminal de Windows digite `npm install -g cowsay`.

```
C:\Users\Hagui>npm install -g cowsay

added 41 packages, and audited 42 packages in 3s

3 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\Hagui>cowsay hola mundo

  < hola mundo >
  -----
      \      ^__^
       (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```



EJEMPLO 3

- Al ejecutar la línea `npm init -y`, observe que el archivo `package.json` aparecen nuevas características.

```
10  "license": "ISC",
11  ✓  "dependencies": {
12    "ansi-styles": "^4.3.0",
13    "camelcase": "^5.3.1",
14    "ansi-regex": "^3.0.0",
15    "cliui": "^6.0.0",
16    "color-convert": "^2.0.1",
17    "color-name": "^1.1.4",
18    "cowsay": "^1.5.0",
19    "decamelize": "^1.2.0",
20    "emoji-regex": "^8.0.0",
21    "get-caller-file": "^2.0.5"
```

✓ EJEMPLO 3

✓ node_modules

- > .bin
- > ansi-regex
- > ansi-styles
- > camelcase
- > cliui
- > color-convert



EJEMPLO 3

- Al ejecutar la línea `npm init -y`, observe que el archivo `package.json` aparecen nuevas características.

```
10  "license": "ISC",
11  ✓  "dependencies": {
12    "ansi-styles": "^4.3.0",
13    "camelcase": "^5.3.1",
14    "ansi-regex": "^3.0.0",
15    "cliui": "^6.0.0",
16    "color-convert": "^2.0.1",
17    "color-name": "^1.1.4",
18    "cowsay": "^1.5.0",
19    "decamelize": "^1.2.0",
20    "emoji-regex": "^8.0.0",
21    "get-caller-file": "^2.0.5"
```

```
✓ EJEMPLO 3
  ✓ node_modules
    > .bin
    > ansi-regex
    > ansi-styles
    > camelcase
    > cliui
    > color-convert
```



EJEMPLO 3

- Si desean compartir información solo se comparten los archivos estáticos y el json.

```
{ } package-lock.json  
{ } package.json  
JS principal.js
```

- La instalación se realiza al ubicar la carpeta del proyecto y digitar el comando npm install o npm i.

```
Waiting for the debugger to disconnect...  
PS C:\Users\Hagui\Documents\GitHub\Servicios-telematicos\Clase_1\Ejemplo 3> npm i
```



EJERCICIO 2



EJERCICIO 2

- Instale el paquete yosay y publique un mensaje de bienvenida.
 - <https://www.npmjs.com>
 - `npm install <paquete>`
 - `const paquete = require("paquete")`
 - `npm init -y`

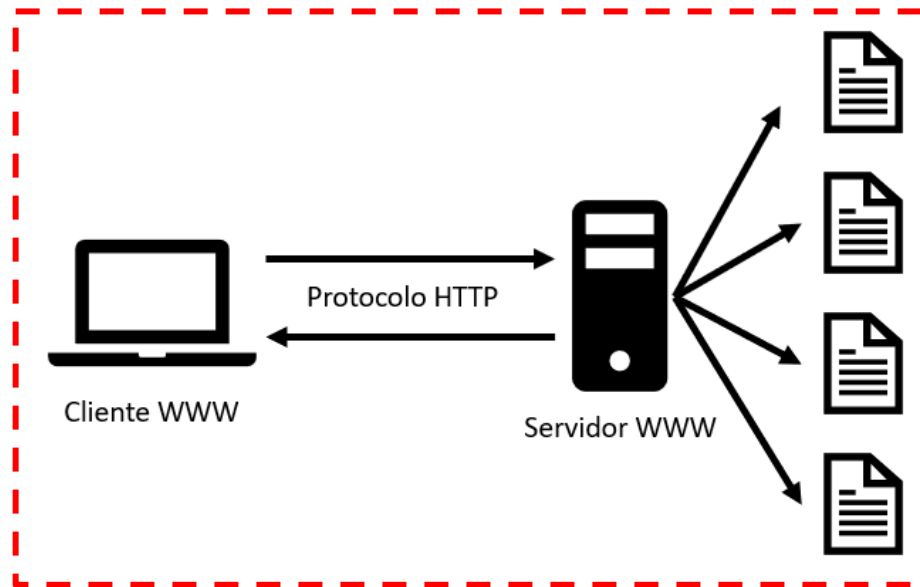


SERVIDOR HTTP



SERVIDOR HTTP

- Es un programa informático que procesa una aplicación del lado del servidor con un cliente, el cual genera una respuesta en un navegador web o aplicación al renderizar el código enviado por el servidor.



SERVIDOR HTTP

- Es un programa informático que procesa una aplicación del lado del servidor con un cliente, el cual genera una respuesta en un navegador web o aplicación al renderizar el código enviado por el servidor.
- Hypertext Transfer Protocol (HTTP) o Protocolo de Transferencia de Hipertexto en que permite la transferencia de información en la World Wide Web (Red mundial).



SERVIDOR HTTP

- Los verbos en HTTP indican la acción que se va a realizar sobre el servidor.
 - **PUT: Es una petición para almacenar una entidad creada.**
 - GET: Obtiene el recurso indicado, Ej. Consultar una pagina WEB.
 - POST: Es un método de creación que añade datos al servidor.
 - DELETE: Elimina un recurso indicado.
 - HEAD: Similar a GET, pero no se obtiene el cuerpo de respuesta, únicamente los metadatos de la cabecera.



SERVIDOR HTTP

- Los verbos en HTTP indican la acción que se va a realizar sobre el servidor.
 - PUT: Es una petición para almacenar una entidad creada.
 - GET: Obtiene el recurso indicado, Ej. Consultar una pagina WEB.
 - POST: Es un método de creación que añade datos al servidor.
 - DELETE: Elimina un recurso indicado.
 - HEAD: Similar a GET, pero no se obtiene el cuerpo de respuesta, únicamente los metadatos de la cabecera.



SERVIDOR HTTP

- Los verbos en HTTP indican la acción que se va a realizar sobre el servidor.
 - PUT: Es una petición para almacenar una entidad creada.
 - GET: Obtiene el recurso indicado, Ej. Consultar una pagina WEB.
 - POST: Es un método de creación que añade datos al servidor.
 - DELETE: Elimina un recurso indicado.
 - HEAD: Similar a GET, pero no se obtiene el cuerpo de respuesta, únicamente los metadatos de la cabecera.



SERVIDOR HTTP

- Los verbos en HTTP indican la acción que se va a realizar sobre el servidor.
 - PUT: Es una petición para almacenar una entidad creada.
 - GET: Obtiene el recurso indicado, Ej. Consultar una pagina WEB.
 - POST: Es un método de creación que añade datos al servidor.
 - **DELETE: Elimina un recurso indicado.**
 - HEAD: Similar a GET, pero no se obtiene el cuerpo de respuesta, únicamente los metadatos de la cabecera.



SERVIDOR HTTP

- Los verbos en HTTP indican la acción que se va a realizar sobre el servidor.
 - PUT: Es una petición para almacenar una entidad creada.
 - GET: Obtiene el recurso indicado, Ej. Consultar una pagina WEB.
 - POST: Es un método de creación que añade datos al servidor.
 - DELETE: Elimina un recurso indicado.
 - HEAD: Similar a GET, pero no se obtiene el cuerpo de respuesta, únicamente los metadatos de la cabecera.



EJEMPLO 4



EJEMPLO 4

- Cree una carpeta que se llame Ejemplo 4 y en ella cree un archivo nuevo llamado principal.js.



EJEMPLO 4

- Cree una carpeta que se llame Ejemplo 4 y en ella cree un archivo nuevo llamado principal.js.

```
JS principal.js > ...  
1 //El módulo http está integrado con node  
2 const http = require("http");  
3 const servidor = http.createServer
```



EJEMPLO 4

- Cree una carpeta que se llame Ejemplo 4 y en ella cree un archivo nuevo llamado principal.js.

```
JS principal.js > ...
1  //El módulo http está integrado con node
2  const http = require("http");
3  const servidor = http.createServer
4  //Cuando creen el servidor siempre da un requerimiento (req)
5  //y da una respuesta (res)
6  const server=http.createServer((req, res)=>{
7  |   res.end("Atendiendo una solicitud");
8  | });
```



EJEMPLO 4

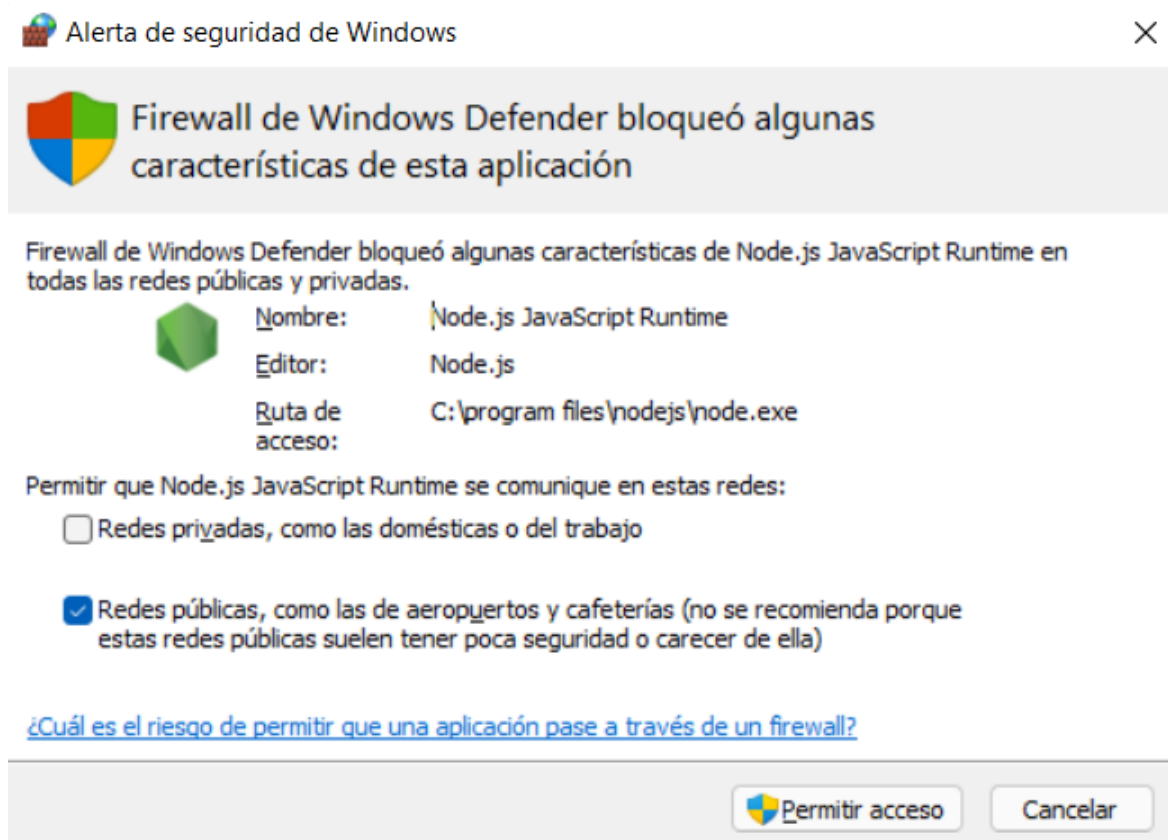
- Cree una carpeta que se llame Ejemplo 4 y en ella cree un archivo nuevo llamado principal.js.

```
JS principal.js > ...
1  //El módulo http está integrado con node
2  const http = require("http");
3  const servidor = http.createServer
4  //Cuando creen el servidor siempre da un requerimiento (req)
5  //y da una respuesta (res)
6  const server=http.createServer((req, res)=>{
7  |    res.end("Atendiendo una solicitud");
8  |});
9  //Nomenclatura de un puerto disponible para usar el servidor
10 const puerto=3000;
11 //El listener u oyente espera la solicitud del lado del cliente
12 server.listen(puerto, ()=>{
13 |    console.log("El servidor esta corriendo en el puerto "+puerto)
14 |});
```



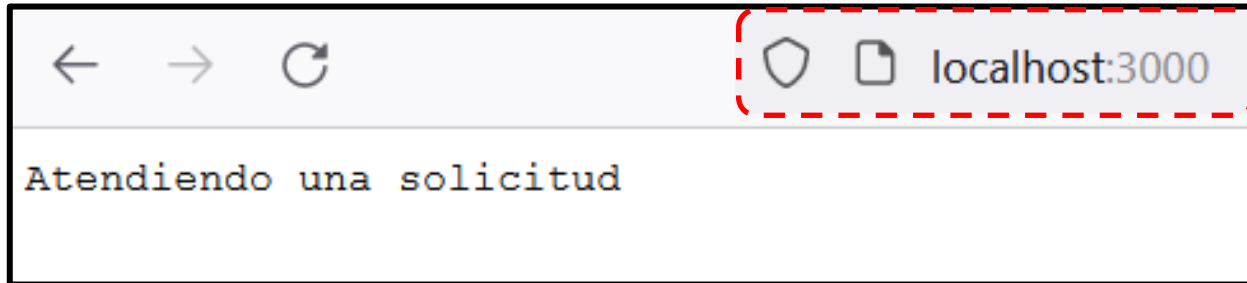
EJEMPLO 4

- Ejecute el script y otorgue los permisos requeridos.



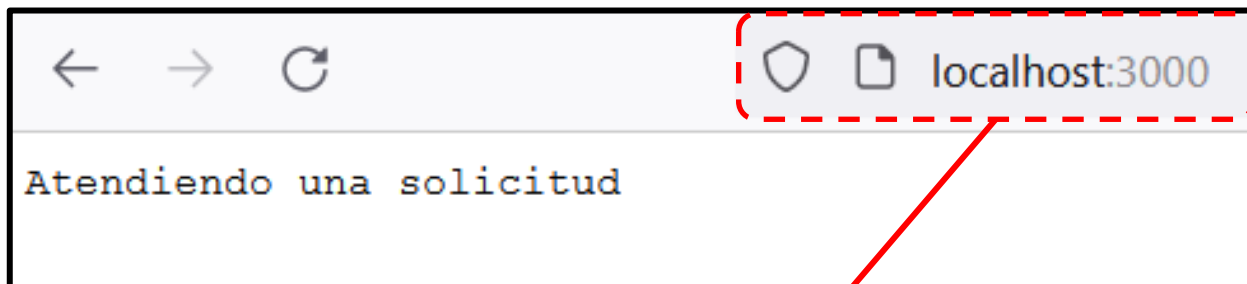
EJEMPLO 4

- Ahora en su navegador digite localhost:3000.



EJEMPLO 4

- Ahora en su navegador digite localhost:3000.



```
6  const server=http.createServer((req, res)=>{  
7    |    res.end("Atendiendo una solicitud");  
8  });
```



EJEMPLO 4

- Seleccione el terminal de node y pulse ctrl+c para terminar la ejecución del servidor .



EJERCICIO 3



EJERCICIO 3

- Instale el paquete nodemon de forma global. Cree un script que publique con la librería yosay la palabra hola en una pagina WEB. Luego ejecute el archivo js usando nodemon y no con node en el terminal. Finalmente, cambie la palabra hola por su nombre, guarde los cambios y verifique la respuesta dada por el servidor en el navegador.
- Ayuda en la siguiente diapositiva.



EJERCICIO 3

- Código particular.

JS principal.js > ...

```
1  const http = require("http");
2  const servidor = http.createServer
3  const server=http.createServer((req, res)=>{
4  |    res.end("Atendiendo una solicitud v2");
5  |  });
6  const puerto=3000;
7  server.listen(puerto, ()=>{
8  |    console.log("El servidor esta corriendo en el puerto "+puerto)
9  |  });
```

- En caso de que no funcione nodemon ejecute VSC como administrador y digite en el terminal **Set-ExecutionPolicy Unrestricted**



EXPRESS



EXPRESS

- Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.



EJEMPLO 5



EJEMPLO 5

- Cree un nuevo proyecto en node e instale el paquete express usando la terminal.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Hagui\Documents\GitHub\Servicios-telematicos\Clase_1\Ejemplo 5> npm i express

added 50 packages, and audited 51 packages in 4s
```



EJEMPLO 5

- De manera similar al caso anterior...

```
1 //Importe el módulo express
2 const express = require("express");
3 //Variable que almacena el objeto para usar los métodos de la clase
4 const app = express();
```



EJEMPLO 5

- De manera similar al caso anterior...

```
1 //Importe el módulo express
2 const express = require("express");
3 //Variable que almacena el objeto para usar los métodos de la clase
4 const app = express();
5 const puerto = 5000;
6 //El primer atributo de get indica la ruta
7 ✓ app.get('/', (req,res)=>{
8   res.send("Hola curso usando express")
9 });
```



EJEMPLO 5

- De manera similar al caso anterior...

```
1  //Importe el módulo express
2  const express = require("express");
3  //Variable que almacena el objeto para usar los métodos de la clase
4  const app = express();
5  const puerto = 5000;
6  //El primer atributo de get indica la ruta
7  ✓ app.get('/', (req,res)=>{
8    res.send("Hola curso usando express")
9  });
10 //Luego se agrega el oyente al servidor
11 ✓ app.listen(puerto, () => {
12   |   console.log("Ejecutando express");
13   | });
```



EJEMPLO 5

- De manera similar al caso anterior...



EJEMPLO 5

- El servidor puede atender múltiples solicitudes (modifique el ejemplo actual).

```
6 //Solicitud 1 atendida en la raíz
7 ✓ app.get('/', (req,res)=>{
8   res.send("Hola curso usando express")
9 });
10 //Solicitud 2 atendida en el espacio servicios
11 ✓ app.get('/curso', (req,res)=>{
12   res.send("Hola curso usando express v2")
13 });
```



EJEMPLO 5

- El servidor puede atender múltiples solicitudes (modifique el ejemplo actual).

```
6 //Solicitud 1 atendida en la raíz
7 ✓ app.get('/', (req,res)=>{
8   res.send("Hola curso usando express")
9 });
10 //Solicitud 2 atendida en el espacio servicios
11 ✓ app.get('/curso', (req,res)=>{
12   res.send("Hola curso usando express v2")
13 });
```



localhost:5000

Hola curso usando express



localhost:5000/curso

Hola curso usando express v2

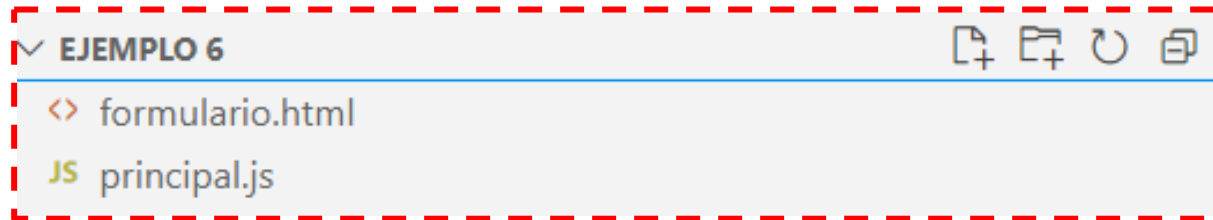


EJEMPLO 6



EJEMPLO 6

- Cree un nuevo proyecto en node, instale el paquete express usando la terminal y cree dos archivos uno llamado principal.js y el otro formulario.html.



EJEMPLO 6

- Abra el archivo formulario.html, digite la palabra html y seleccione la opción marcada para crear una plantilla.



The screenshot shows a code editor with the file 'formulario.html' open. On line 1, the text 'html' is entered. A dropdown menu is visible, listing three options: 'html', 'html:5', and 'html:xml'. Each option is preceded by a small wrench icon. The 'html:5' option is highlighted with a red rectangular box. The text 'Emmet Abbreviat...>' is visible at the end of the menu.

```
<> formulario.html
1  html
   html
   html:5
   html:xml
```



EJEMPLO 6

- Como se observa la aplicación genera un código por defecto.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10  |
11  </body>
12  </html>
```



EJEMPLO 6

- Agregue la siguiente estructura.

```
<body>  
  <form action="/" method="post">  
    <label>Escriba un mensaje:</label>  
    <input type="text" id="msn" name="msn">  
    <button type="submit">Envíe su mensaje</button>  
  </form>  
</body>
```



EJEMPLO 6

- Guarde los cambios y cree un servidor en el archivo principal.

```
1  const express = require("express");  
2  const app = express();  
3  const puerto = 5000;  
4  //Módulo para juntar palabras y construir una ruta.  
5  const path = require('path');  
6  //Extrae información de la solicitud entrante.  
7  const bodyParser = require('body-parser');
```



EJEMPLO 6

- Guarde los cambios y cree un servidor en el archivo principal.

```
1  const express = require("express");
2  const app = express();
3  const puerto = 5000;
4  //Módulo para juntar palabras y construir una ruta.
5  const path = require('path');
6  //Extrae información de la solicitud entrante.
7  const bodyParser = require('body-parser');
8  //Recupera información en forma de texto unicamente.
9  app.use(bodyParser.urlencoded({extended: false}));
10 //Solicitud atendida y redirigida al archivo formulario.
11 app.get('/', (req,res)=>{
12     //__dirname indica la ubicación del proyecto
13     res.sendFile(path.join(__dirname, '/formulario.html'));
14 });
```



EJEMPLO 6

- Guarde los cambios y cree un servidor en el archivo principal.

```
1  const express = require("express");
2  const app = express();
3  const puerto = 5000;
4  //Módulo para juntar palabras y construir una ruta.
5  const path = require('path');
6  //Extrae información de la solicitud entrante.
7  const bodyParser = require('body-parser');
8  //Recupera información en forma de texto unicamente.
9  app.use(bodyParser.urlencoded({extended: false}));
10 //Solicitud atendida y redirigida al archivo formulario.
11 app.get('/', (req,res)=>{
12     //__dirname indica la ubicación del proyecto
13     res.sendFile(path.join(__dirname, 'formulario.html'));
14 });
15 //Captura de información usando post
16 app.post('/', (req,res)=>{
17     var mensaje=req.body.msn;
18     console.log(mensaje)
19     res.send('El mensaje era '+mensaje);
20 });
21 app.listen(puerto, () => {console.log("Ejecutando express");});
```



EJEMPLO 6

- Guarde los cambios y cree un servidor en el archivo principal.

```
15 //Captura de información usando post
16 app.post('/', (req,res)=>{
17     var mensaje=req.body.msn;
18     console.log(mensaje)
19     res.send('El mensaje era '+mensaje);
20 });
21 app.listen(puerto, () => {console.log("Ejecutando express");});
```

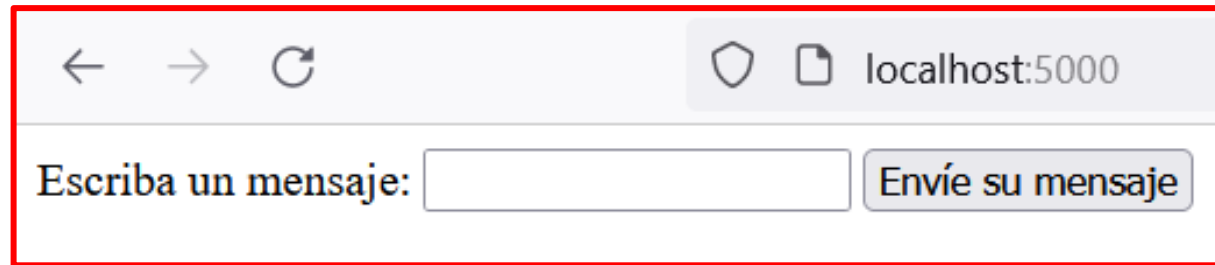
```
<body>
  <form action="/" method="post">
    <label>Escriba un mensaje:</label>
    <input type="text" id="msn" name="msn">
    <button type="submit">Envíe su mensaje</button>
  </form>
</body>
```



**Toma el atributo
name**



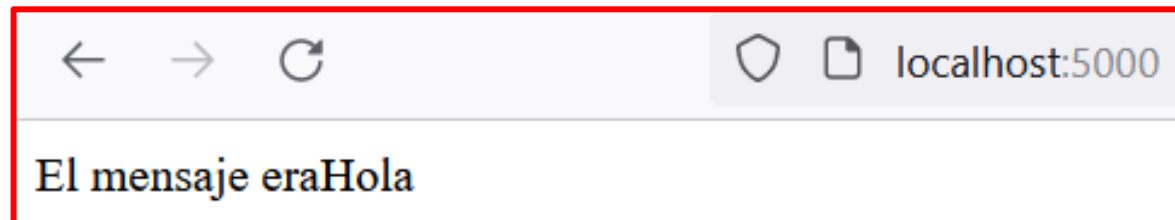
EJEMPLO 6



- Finalmente al ejecutar y pulsar el botón verán lo siguiente.



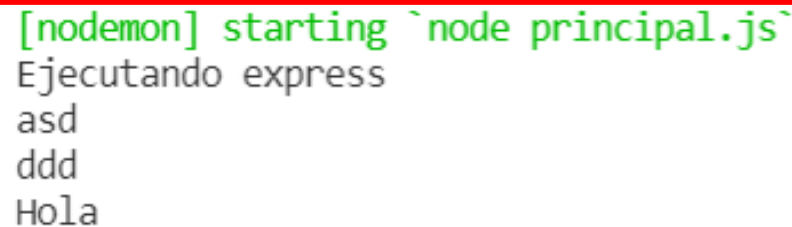
← → ↻   localhost:5000

Escriba un mensaje: Envíe su mensaje



← → ↻   localhost:5000

El mensaje eraHola



```
[nodemon] starting `node principal.js`  
Ejecutando express  
asd  
ddd  
Hola
```



EJERCICIO 4



EJERCICIO 4

- Usando express tome la edad de una persona en un formulario y publique en una pagina ese valor multiplicado por 2.



EJERCICIO



EJERCICIO

- Cree una aplicación que le permita registrar y visualizar la información de inventario de una empresa y realizar consultas de la siguiente manera:
 - Información general de los productos registrados (terminal).
 - Una ruta muestra el nombre de los productos con serial par (WEB).
 - Una ruta muestra el nombre de los productos con serial impar (WEB).
 - Otra ruta muestra los productos con serial ordenado de forma ascendente o descendente (Terminal).
- ***Nota: el formulario solo permite ingresar los datos de una persona por ejecución.***

