NODE JS

* Node.js es un entorno de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript.
* Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor.
* Está basado en el motor V8 de Google.
* Permite hacer aplicaciones que se ejecutan en el servidor.

Instalación

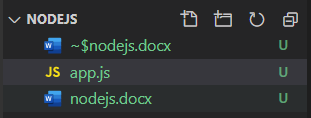
Ejecutar el instalable de node para nuestro sistema operativo.

Comprobar versión que tenemos instalada de node.

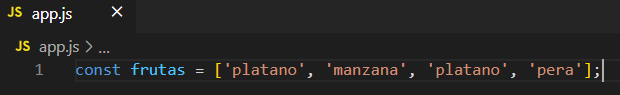


Creando una app (contador de fruta)

Creamos el fichero app.js



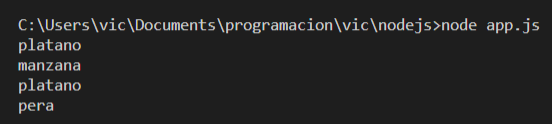
Definimos un array



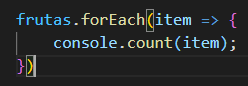
Lo iteramos con forEach

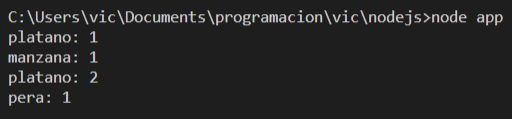


Para lanzar la app usamos **node** seguido del **nombre del fichero**



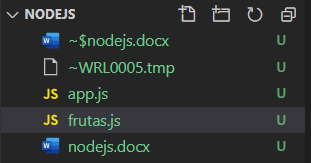
Podemos usar la función **console.count ()** para que nos devuelva la cantidad de piezas de cada fruta (hasta ese orden).



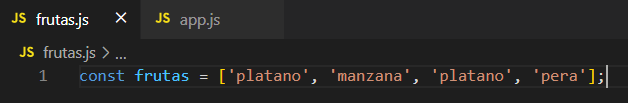


Exportaciones

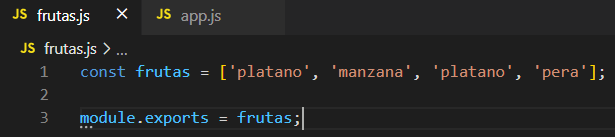
Creamos otro fichero llamado **frutas.js**



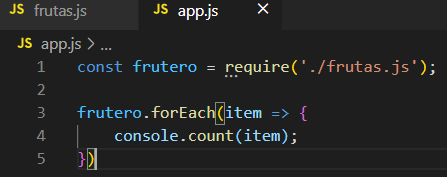
Cortamos y pegamos el array de frutas que tenemos en app.js

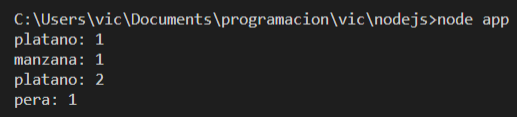


Para exportar el array usamos **module.exports** y dentro del igual añadimos lo que vamos a exportar, en este caso, nuestro array.

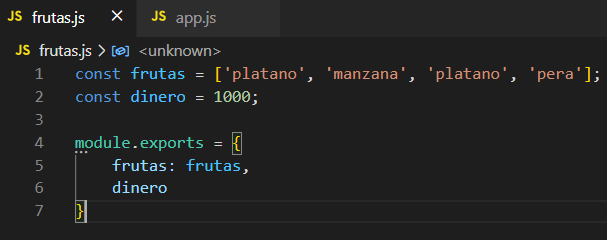


Para importarlo en el fichero que lo vamos a requerir, definimos una constante y en ella requerimos la ruta de nuestro fichero que contiene el array y lo iteramos de nuevo.

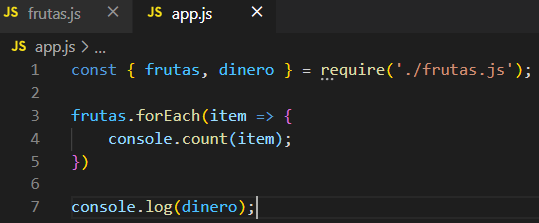


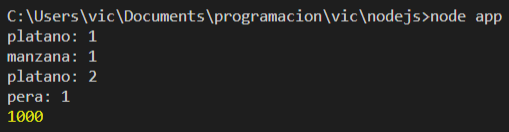


Si queremos exportar más de una constante, objeto o función lo haremos dentro de un objeto, y podemos hacerlo bien creando clave-valor o solo con el nombre de la constante (en este caso)



Para requerir directamente las constantes definidas en **frutas.js** podemos usar **destructuring** de js



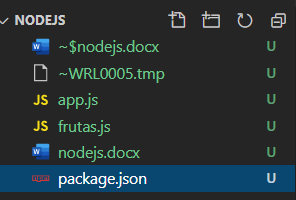


Package.json

Si ejecutamos en nuestra consola

npm init -y

Se nos creará un archivo el cual tendrá información sobre nuestro proyecto, lo más relevante en estos momentos serán sus dependencias y scripts





Dependencias y devDependencies:



* Aquí estará la lista de paquetes instalados a través de npm.
* devDependencies hace referencia a los paquetes que no se necesitan para producción, están destinados a instalarse solo en una máquina de desarrollo, no son necesarios para ejecutar el código en producción.

NPM (Node Package Manager)

 Es el administrador de paquetes estándar para Node.js.

<https://nodejs.dev/learn/an-introduction-to-the-npm-package-manager>

Instalando un paquete

¿Dónde instala npm los paquetes?

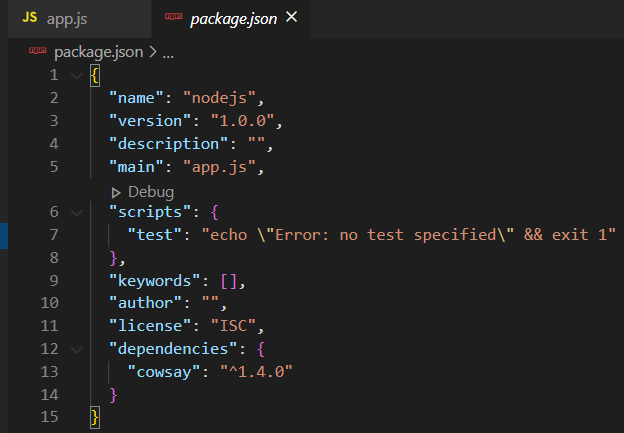
* Cuando instala un paquete con npm, puede realizar 2 tipos de instalación:
* Local: npm install <package-name> (se crea la carpeta node\_modules)
* Global: npm install -g <package-name> (se instala en el pc)

Instalamos la librería **cowsay**

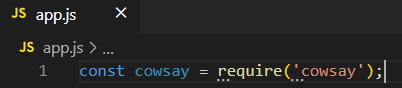
<https://www.npmjs.com/package/cowsay>



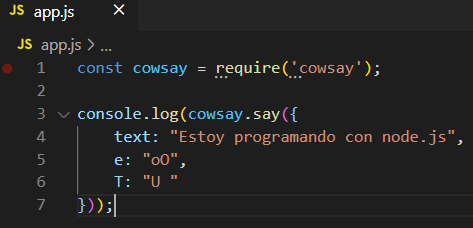
Podemos comprobar que se ha añadido a las dependencias



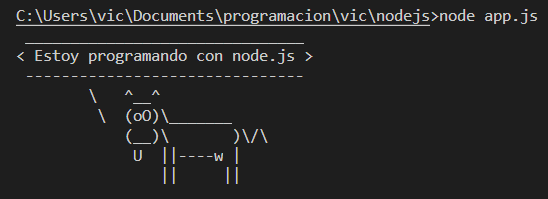
Como hicimos anteriormente, para poder hacer uso de la librería la requerimos



Seguimos la documentación y pegamos un console.log con las claves-valor para que funcione.



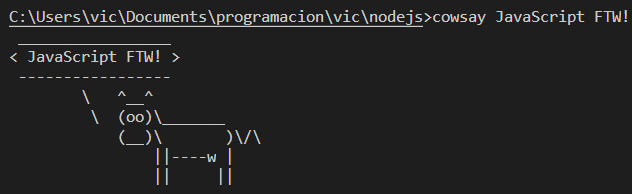
Lanzamos la app



Para ver como funciona al instalarla de manera global



Usamos la palabra reservada que nos indica la documentación seguido del texto que vamos a ver, así ya no es necesario depender de la node para hacer uso de la librería, normalmente solo instalaremos de manera global los CLI de los diferentes frameworks (Angular, React, Vue…)



NPX

Posiblemente se toparán con este comando a futuro (sobretodo si trabajan con React.js), bueno esto ejecuta un paquete de npm sin necesidad de instalarlo de forma global o local.

Por ejemplo si ejecutamos en la terminal CMD:

npx cowsay "Hola mundo"

Npx le permite ejecutar código creado con Node.js y publicado a través del registro npm.

Crear un servidor HTTP

Hypertext Transfer Protocol: El Protocolo de transferencia de hipertexto es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

* Intercambio de información entre cliente y servidor.
* El servidor queda a la espera de alguna solicitud HTTP ejecutada por el cliente y proporciona una respuesta.

Cuando visitamos un sitio web, hacemos una solicitud GET de HTTP, y el servidor nos devuelve por ejemplo un index.html con el sitio web.

Nosotros configuraremos estas respuestas en nuestro "servidor web o servidor http" con node.js

[#](https://bluuweb.github.io/node/01-fundamentos/#verbos-o-metodos-http)**Verbos o métodos HTTP**

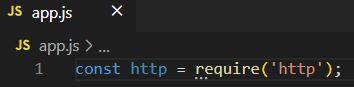
<https://developer.mozilla.org/es/docs/Web/HTTP/Methods>

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado.

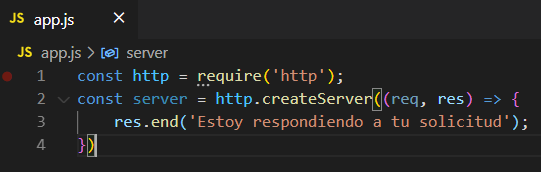
Los más populares son: GET, POST, PUT, DELETE.

En el siguiente apartado crearemos nuestro servidor web y pondremos en práctica cada uno de estos métodos.

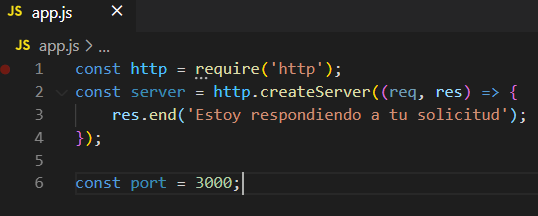
Este módulo de Http ya lo contiene NodeJs por definición, por lo que no es necesario instalarlo, solo requerirlo



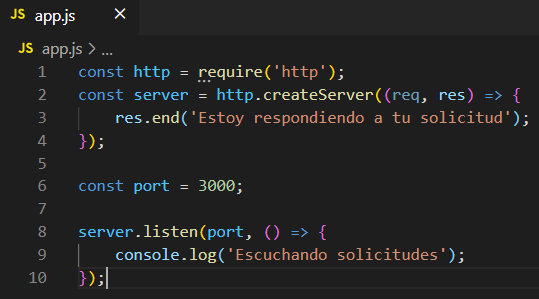
Creamos el servidor que será un método de **http** en la que tendremos siempre un requerimiento por parte del usuario (req) y una respuesa por parte del servidor (res)



Definimos un puerto para desplegar el servidor



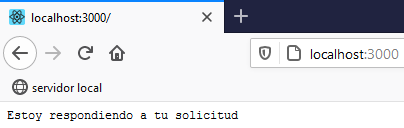
Para que el servidor pueda estar siempre a la escucha de las solicitudes que reciba, va a tener siempre como parámetro un puerto (el que definimos) y una función flecha.



Probamos que funcione



Para ver la respuesta del servidor lo hacemos a través del navegador



Nodemon

Esta herramienta se instala a través de npm y nos sirve para estar escuchando cambios en nuestra configuración de node.js y reinicia automáticamente el servidor.

Instalación global:

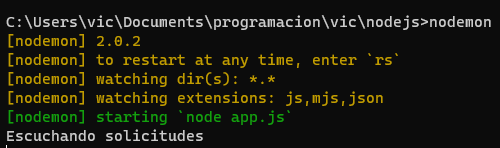
npm install -g nodemon

Ejecutar:

nodemon app.js

Así, con cualquier cambio que hagamos en nuestra app, no será necesario parar y lanzar manualmente el servidor.

Para arrancar el servidor local con nodemon



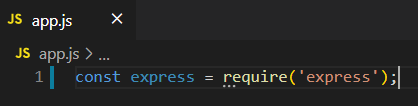
Express

* 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 (Framework).
* Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla.

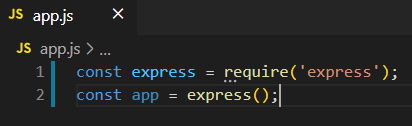
Para instalar Express

npm i express

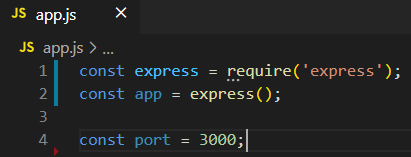
Requerimos express para poder usarlo



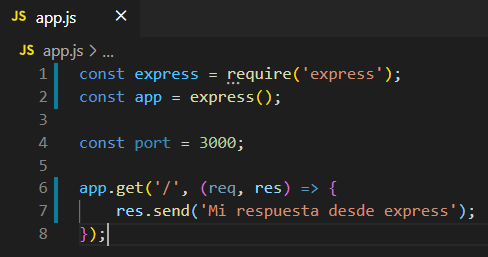
Para usar express como app lo definimos así



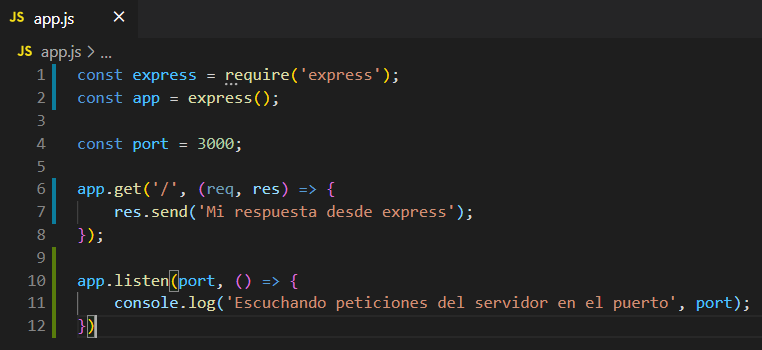
Definimos el puerto que vamos a utilizar



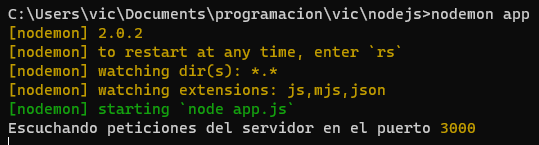
Hacemos la primera petición http



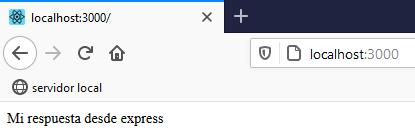
Para que el servidor esté continuamente escuchando las peticiones pasamos en primer parámetro el puerto de escucha



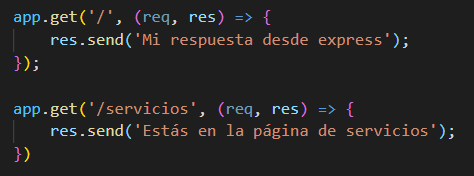
Arrancamos el servidor

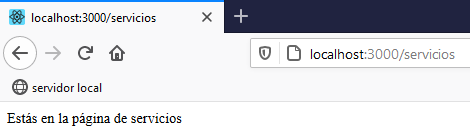


Desde el navegador



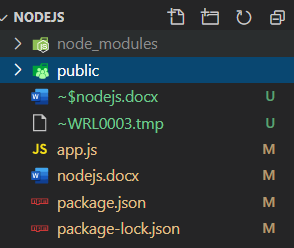
Si definimos otro end-point



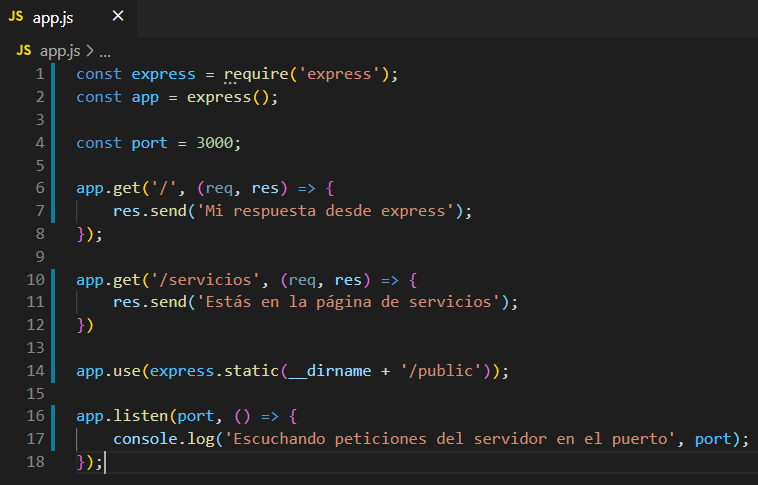


Archivos estáticos

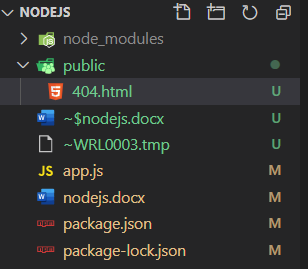
Creamos en la raíz una carpeta llamada **public** en la que guardaremos todo a lo que el usuario podrá acceder.



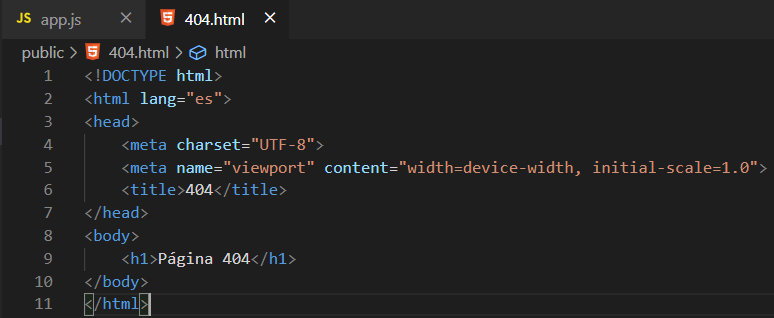
En nuestra app definimos un middleware (se denomina middleware porque estamos ejecutando una función antes de que se hagan las solicitudes) donde va a estar ubicada la carpeta **public**, usamos **\_dirname** porque es una **ruta dinámina**, es decir, cuando subamos nuestro proyecto a un hosting, es posible que tenga su propia raíz, de esta manera, a la carpeta **public** siempre se accederá de la misma manera desde nuestro equipo o hosting.



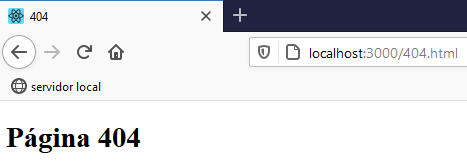
Para ver un ejemplo, vamos a añadir a la carpeta public un html 404



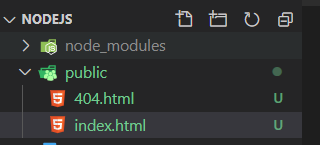
La definimos con un simple titulo H1



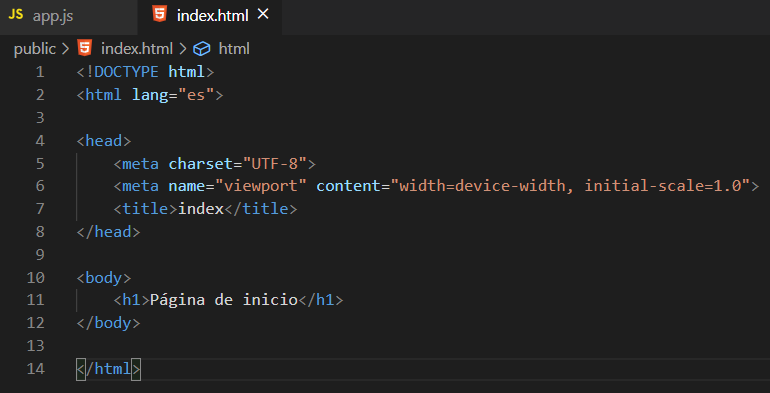
Si en el navegador indicamos la dirección de está página



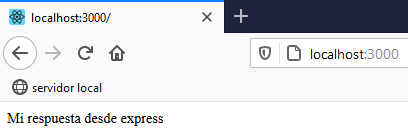
Ahora definimos una página index.html también dentro de **public**



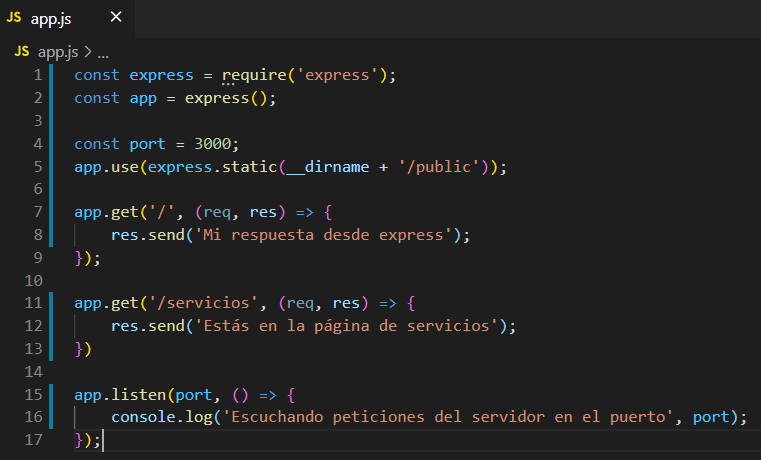
La definimos también con un simple H1



Si accedemos a la ruta principal desde el navegador no vemos ningún cambio



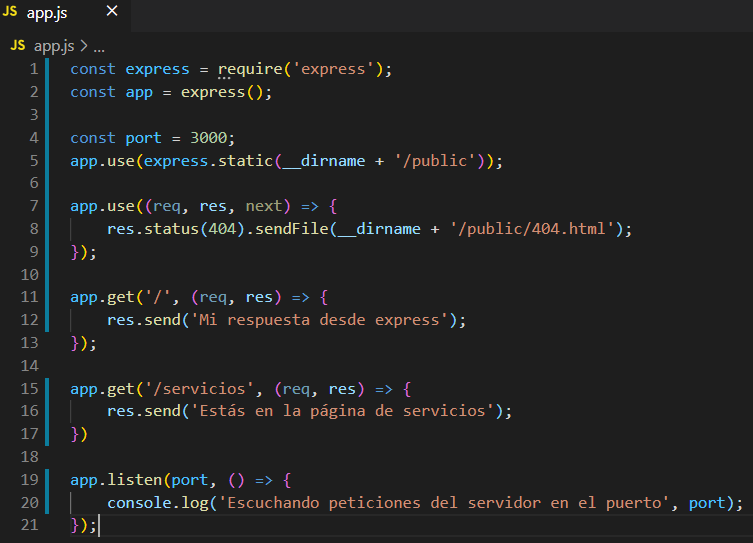
Si el middleware con la ruta estática la definimos antes de definir nuestra ruta



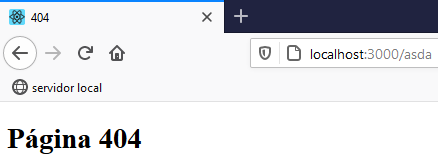
Ahora lo veremos así



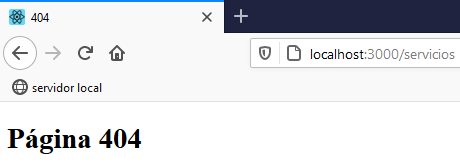
Podemos configurar un middleware para la página 404



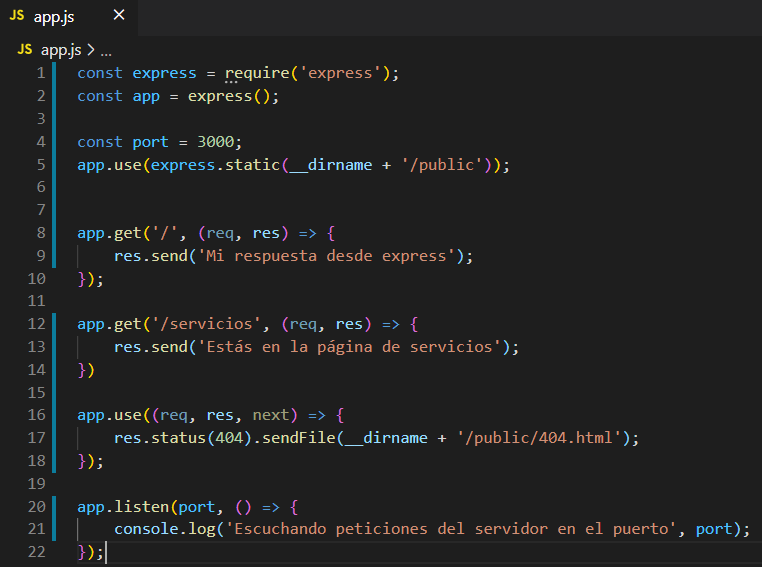
Si intentamos acceder a una ruta que no está dada de alta nos lee la página 404.html



Pero si accedemos a la página de servicios también nos aparece la página de 404.html y esta ruta si la tenemos dada de alta, esto es por el orden en el que hemos establecido el middleware.

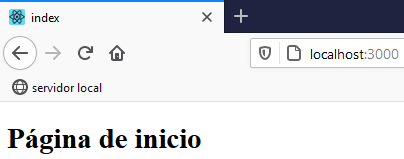


Si colocamos el middleware justo debajo de todas las solicitudes



Ahora si funcionan todas las rutas definidas





Pero si accedemos a una ruta no definida se aplica el middleware

