HINTS

CONSEJOS CONCEPTUALES

- No es necesario que te aprendas de memoria todos los comandos de npm, ni su funcionamiento en específico. Siempre puedes recurrir a su documentación (https://docs.npmjs.com/cli/v7/commands), ahí encontrarás los comandos que hemos revisado, los cuales son solo unos pocos de los muchos que el CLI de npm puede aceptar.

- Lo que sí es importante, es empezar a reconocer la utilidad que npm tendrá para ti en el futuro, y como se relaciona con node y la estructura de un proyecto. Entender esta relación **es mucho más importante** que memorizar todos los comandos.

- También hay que entender que existen paquetes que te serán de ayuda en el ambiente **fuera del código de tu proyecto**. Un ejemplo de esto es nodemon: este paquete no influye en el resultado de la ejecución de tu programa, pues si se encuentra o no instalado, el resultado **será el mismo**. Su utilidad se encuentra fuera de tu proyecto, y ayuda a agilizar el desarrollo.

- Existen también otros paquetes que te serán de ayuda dentro de tu proyecto, tales como: **momentjs o lodash**, entre otros.

- Respecto a los cambios que se ven en el repositorio de git, al momento de instalar o desinstalar paquetes, recuerda que tu archivo package.json se actualizará **cada vez** que esto ocurra, para así mantener al día la integridad de dependencias de tu proyecto.

- Socket.io para aplicaciones web en tiempo real. Permite la comunicación bidireccional entre clientes web y servidores. Consta de dos partes: una biblioteca del lado del cliente, que se ejecuta en el navegador; y una biblioteca del lado del servidor para Node.js, ambas tienen una API casi idéntica. Al igual que node.js, se basa en eventos. Si quieres obtener más información, puedes visitar su página oficial: https://socket.io.

- Underscore.js es una biblioteca de JavaScript ligera, la cual ofrece funciones útiles para manipular datos. Dichas funciones están clasificadas en cinco grupos:

- Colecciones: funciones para recorrer, buscar, filtrar, ordenar, agrupar, contar, o hallar el máximo y el mínimo en un conjunto de elementos.
- Arrays: funciones para manipular arrays: obtener el primer o el último elemento, buscar elementos, hallar uniones o intersecciones entre arreglos, entre otros.
- Funciones: funciones para manipular el modo, el evento, o el momento en el cual se ejecutará una función.
- Utilidad: funciones útiles de uso variado. Permiten invocar una función una determinada cantidad de veces, obtener un número aleatorio, depurar cadenas de texto, obtener la hora actual, entre otros.

Para conocer más acerca de esta librería, puedes ir a su página oficial: http://underscorejs.org .

- Morgan es un middleware, utilizado con Express para la captura de solicitudes HTTP, éste muestra información importante de las request, como: métodos, tiempo de respuesta, endpoints, entre otros. Si quieres obtener más información, puedes ingresar al siguiente enlace: https://expressjs.com/en/resources/middleware/morgan.html

¿QUÉ ES UN MIDDLEWARE?

Es un concepto muy importante al desarrollar sobre Node.js, la plataforma para utilizar a JavaScript en el lado del servidor.

Un middleware es una función que se puede ejecutar antes o después del manejo de una ruta, es decir, es un bloque de código que se ejecuta entre la petición que hace el usuario (request), hasta que ésta llega al servidor.

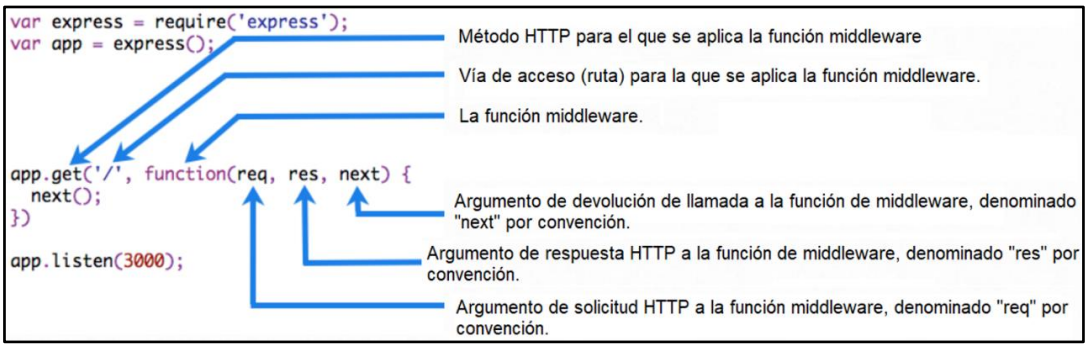
Suelen ser utilizadas como mecanismo para verificar niveles de acceso, antes de entrar en una ruta, manejo de errores, validación de datos, entre otros.

Estas funciones pueden realizar las siguientes tareas:

- Ejecutar cualquier código.
- Realizar cambios en la solicitud y los objetos de respuesta.
- Finalizar el ciclo de solicitud/respuestas.
- Invocar el siguiente middleware en la pila.

Si la función de middleware actual no finaliza el ciclo de solicitud/respuestas, debe invocar next() para pasar el control a la siguiente función. De lo contrario, la solicitud quedará colgada.

En la siguiente imagen se muestran los elementos de una llamada a función de middleware.



CONSEJOS PRÁCTICOS

- Recuerda que, para dividir tu código en distintos archivos, debes realizar un **proceso de dos pasos**. En primer lugar, definir explícitamente cuales los componentes de tu código que quieres exportar. Y la segunda parte es importar estos componentes dentro del archivo que vaya a hacer uso de ellos.

- Recuerda siempre que debes **especificar la ruta** en donde se encuentra el archivo a importar, para que así Node sea capaz de hacer uso de los componentes, pues de lo contrario, tu programa **no será capaz de correr**.

- Antes de importar un paquete o librería, debes leer la documentación oficial del paquete en cuestión, ya que todos tienen **distintas formas de uso,** y lo más recomendable es obtener la información desde la **fuente original**.

- Todas las documentaciones tienen sintaxis, y maneras distintas de expresar el uso de sus paquetes; esto puede ser particularmente complicado en un principio, pero a medida que leas más documentación de distintas librerías, **será más fácil entender** la explicación del autor para el uso de cada paquete.

- No siempre es fácil decidir cuándo dividir una porción de tu código. A medida que tus programas vayan creciendo, **irás notando ciertos patrones** dentro de tu manera de

programar, y verás en qué momento será mejor hacerlo. Si existe una función que estés utilizando en distintos archivos, puedes pensar en tenerla **escrita una sola vez,** y en un solo archivo para requerirla donde lo desees. Si al terminar de escribir una función o porción de código, notas que es **muy larga o difícil de leer**, puedes pensar en la forma de dividirlo en distintas partes para hacer más fácil su lectura y/o debugging.

Si bien los paquetes son una herramienta muy poderosa, tampoco significa que debas utilizar todos los que encuentres. En general, es ideal usar **solo lo necesario**, ya que paquetes instalados de más, harán crecer en tamaño a tu proyecto, y hacerlo más dependiente de otras fuentes que pueden estar constantemente en actualizaciones, lo cual puede provocar **inconsistencias**.