UNIDAD 4: Node.js

1. [UNIDAD 4: Introducción](https://learn.nextu.com/mod/page/view.php?id=3540&pid=P_WEBDEV_V2)

Estudiaremos la escritura de servidores, desde cero, en node.js, el ecosistema JavaScript del lado del servidor que es destacado por su capacidad de procesar grandes cantidades de datos en tiempos muy bajos.

Los objetivos del Programa que corresponden a esta Unidad son:

* Usar metodologías sincrónicas y asincrónicas para gestionar datos desde el servidor.

Los objetivos del Curso 6 que corresponden a esta Unidad son:

* Describir el funcionamiento de la ejecución sincrónica y asincrónica
* Definir qué es una API y cómo funcionan los Servicios Web
* Implementar AJAX
* Comprender qué es y cómo funciona Node.js

1. Lección 1: Qué es Node.js y cómo funciona
   1. [Qué es Node.js](https://learn.nextu.com/mod/lesson/view.php?id=3541&pid=P_WEBDEV_V2)

Qué es Node.js?

Node.js es un entorno de ejecución multiplataforma, de código abierto, totalmente asíncrono, con una entrada y salida de datos basada en una arquitectura orientada a eventos.  
  
Node.js es JavaScript pero del lado del servidor, utiliza la máquina virtual de Google Chrome (V8) para la interpretación y ejecución del código de javascript. Node fue desarrollado con la misión de crear programas y servidores web de alto rendimiento y fácilmente escalables.  
  
Node.js es ideal para crear servidores que manejen muchas conexiones simultáneas y una concurrencia de usuarios/datos bastante alta.

¿Qué es V8?

V8 es el motor de JavaScript creado por Google para su navegador Chrome que se encarga de interpretar y ejecutar el código. Es un intérprete bastante poderoso con una característica especial, OPEN-SOURCE, es decir de código abierto, y cualquier persona puede descargar el código fuente de este intérprete y hacerlo funcionar en una aplicación. Node usa la máquina virtual V8 y le da otro propósito para usarlo en el servidor.

¿Por qué no utilizar el mismo lenguaje que usted emplea en el cliente para programar el servidor ?

Fue una de las preguntas con las que se inició la idea, la posibilidad de crear una arquitectura que contemplara el lenguaje Javascript tanto del lado del cliente como del servidor.

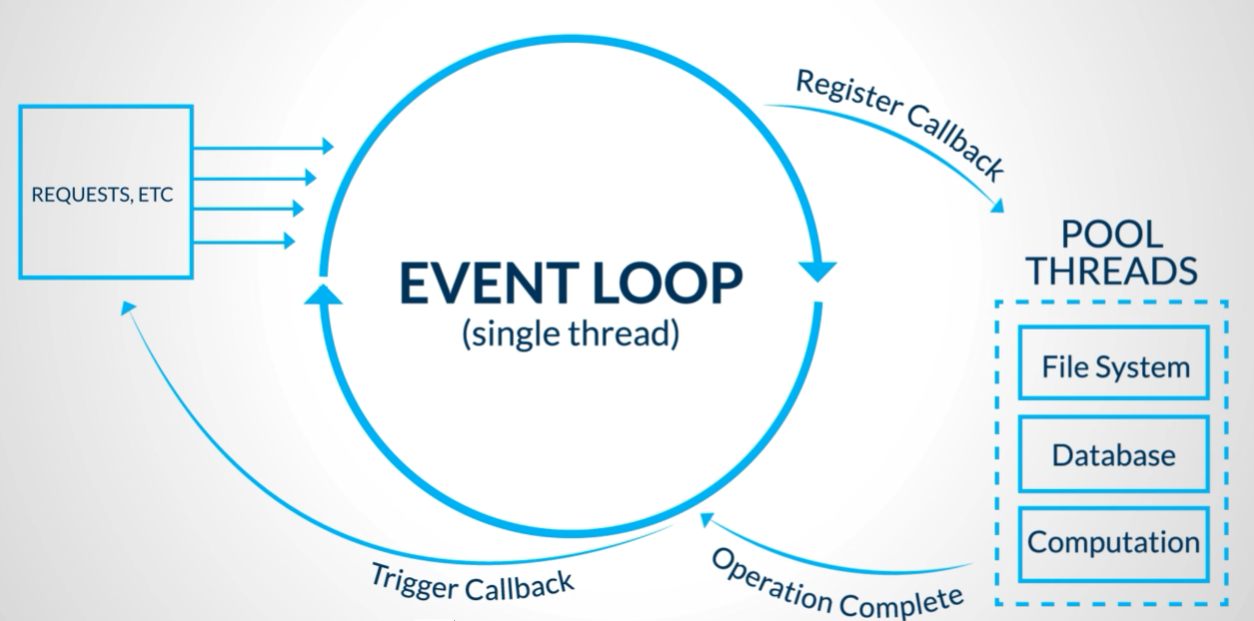
Programación orientada a eventos:

Hemos escuchado que existen varios paradigmas de programación muy populares como POO(Programación Orientada a Objetos), el cual es muy común encontrarlo en la mayoría de lenguajes más populares de la actualidad.  
  
Node usa el paradigma de Programación Orientada a Eventos. Al igual que en el lado del cliente estamos a la espera de acciones del usuario y sus correspondientes eventos, en el lado del servidor no es nada diferente, aunque no se están presionando botones ni llenando formularios, en otros niveles están sucediendo eventos, como establecimiento de conexiones, envío y recepción de datos y todas las demás actividades para las que un servidor debe estar en capacidad de responder.

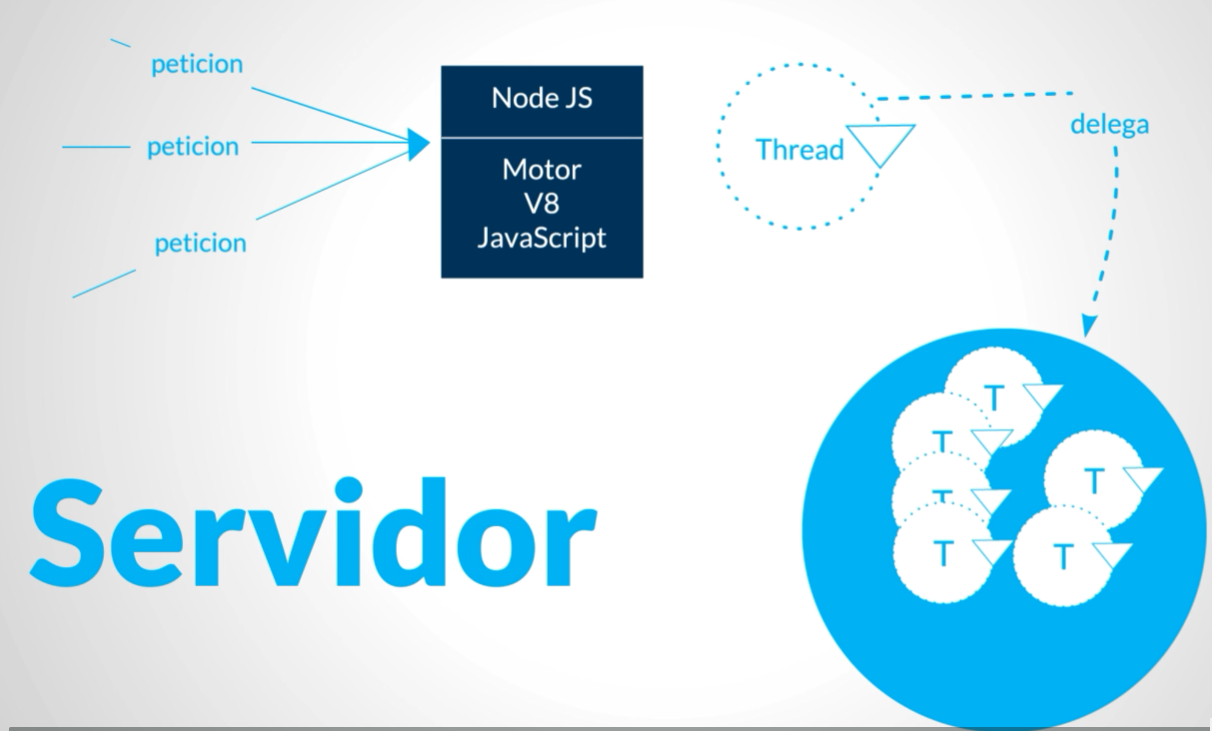
* 1. Cómo funciona Node.js

Eventos 🡪 Asíncrono

Trabaja con un único hilo de ejecución, las tareas y procesos es de un modelo llamado evento Loop.



Librería llamada libuv 🡪 múltiples hilos asíncronos

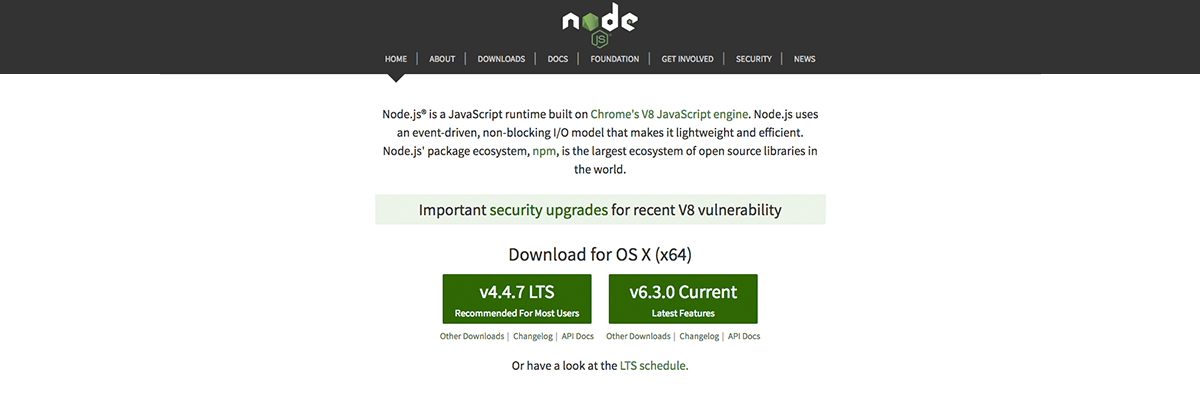


* 1. Descarga e instalación

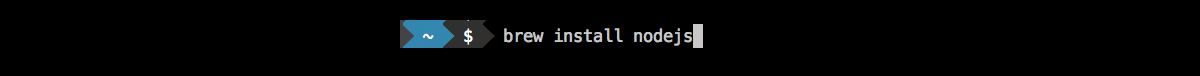
Descarga e Instalación de Node.js

Para tener Node.js en nuestra máquina, es necesario descargar el binario e instalarlo.  
  
Si estás en MacOS existen varias formas:

Utiliza Brew (el administrador de paquetes para MacOS), abre la terminal y digita:



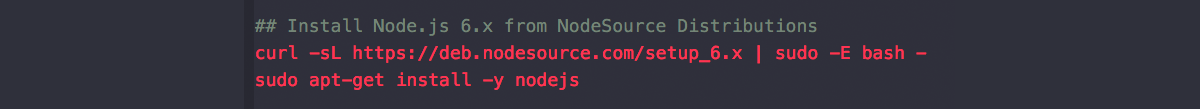
Otra forma es ir directamente a la página oficial: [https://nodejs.org](https://nodejs.org/en/) y descargar alguna de las versiones disponibles:



Actualmente existen dos versiones de Node.js disponibles, la versión 4.x es LTS (Long Term Support), es decir, una versión que estará estable por mucho tiempo y no sufrirá cambios drásticos que puedan romper el ecosistema, es la versión más recomendada para la mayoría de usuarios.  
  
La versión 6 es en la cual se incorporan y se prueban las nuevas características y funcionalidades y es posible que sufra muchos cambios y corrección de errores. Esto puede causar que dejen de funcionar algunas propiedades. No se recomienda para aplicaciones en producción.  
  
Si estás en Windows, para instalar Node.js debes descargar el archivo ejecutable desde la página oficial, aceptar los términos y condiciones y hacer clic en siguiente hasta finalizar la instalación.  
  
Si estás en linux - Debian o algún derivado es porque eres un usuario un poco más avanzado, la recomendación que hacemos es que instales Node.js via PPA, agregando el repositorio de NodeSource y luego descargando via apt-get.



Ahora sólo falta verificar que Node.js esté correctamente instalado en la máquina, para lo que debes ingresar a una terminal y digitar “node --version”:



La Terminal te devuelve la versión de node que está instalada.

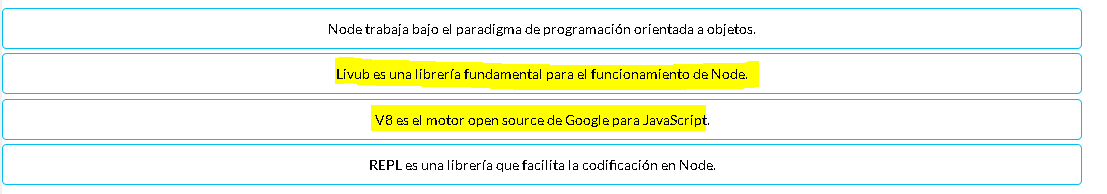
* 1. Node.js REPL

Que es un REPL – Readable Print to Loop

* 1. Actividad

Tiempo de pensar

En esta lección aprendiste que node es un entorno de ejecución multiplataforma que permite realizar gran cantidad de acciones del lado del servidor. Además, usa el lenguaje JavaScript lo cual es muy conveniente para los desarrolladores ya que no hay diferencia entre el lenguaje usado en el front-end y el usado en el back-end.  
  
Instrucciones:  
  
Lee con atención las opciones de respuesta, las cuales contienen algunas afirmaciones relacionadas con Node. Selecciona las 2 afirmaciones verdaderas.



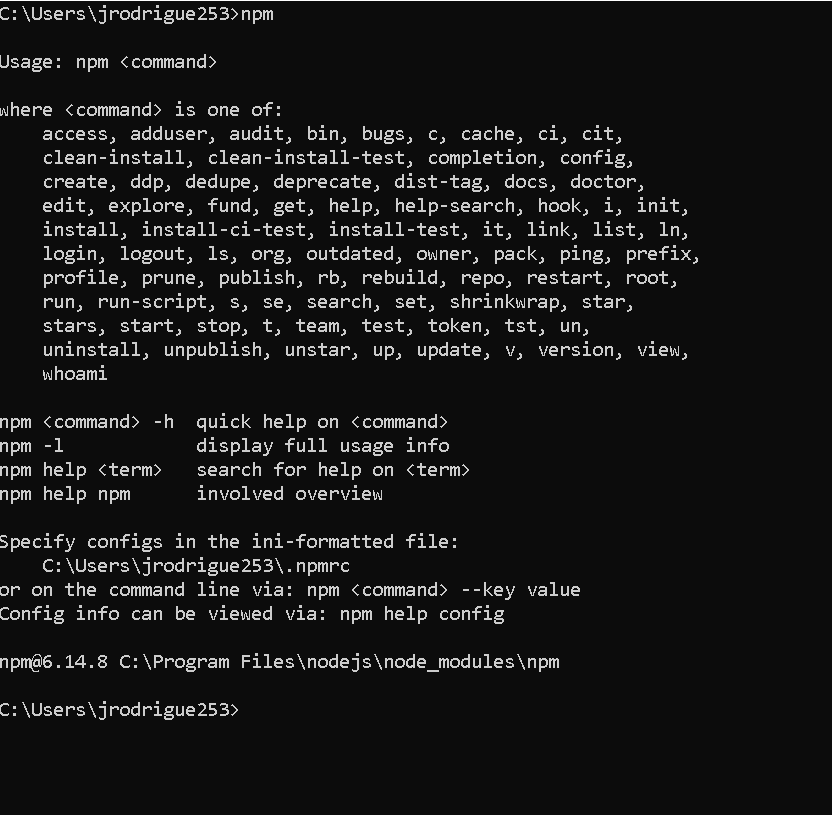
1. Lección 2: Node Package Manager (NPM)
   1. Qué es NPM

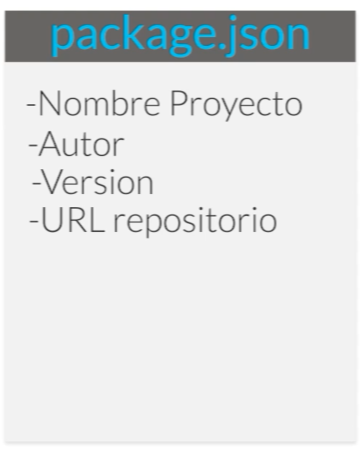
¿Qué es NPM?

No es conocido, a excepción de sus creadores, el verdadero significado de las siglas N.P.M. Lo que sí sabemos es que es un maravilloso gestor de paquetes y dependencias para JavaScript, diseñado para funcionar en la línea de comandos. Es un sistema que viene integrado con Node.js y se instala en el mismo momento que se instala Node en la máquina.  
  
Casualmente N.P.M. se acomoda perfectamente al nombre de Node Package Manager y así es como se conoce entre la comunidad de desarrolladores JavaScript. N.P.M. es una plataforma de código abierto, en la que miles de programadores de todo el mundo pueden crear librerías o paquetes como son conocidos en el ecosistema Node.js y publicarlos para que puedan ser descargados y usados por otras personas.  
  
En palabras concretas, N.P.M. es un gestor de módulos y paquetes para Node.js en el cual los desarrolladores crean, comparten y reutilizan módulos en las aplicaciones.  
  
N.P.M. cuenta con una cantidad de paquetes increíbles, los cuales nos ayudan a resolver muchos de los problemas que nos encontramos dia dia como desarrolladores Node.js

3.2 Iniciar con NPM

Npm en la línea de comandos



etc

3.3 Paquetes propios y crear un servidor simple en Node

Sumamente modular

Incluir un módulo:

Require(‘http’)

Instalar paquetes

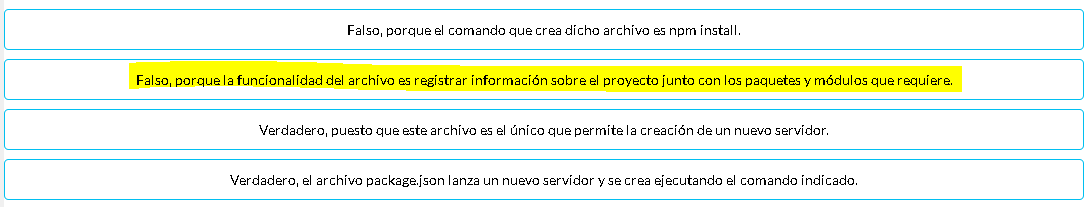


3.4 Paquetes NPM

3.5 Actividad

Tiempo de pensar

En esta lección tuviste la oportunidad de aprender sobre NPM y su función de administración de paquetes para Node. Saber usar esta herramientas en los proyectos de desarrollo implementados en Node, es fundamental puesto que en la mayoría de ocasiones tendrás la necesidad de usar paquetes de terceros en tu código.  
  
Instrucciones:  
  
Lee el enunciado a continuación y luego selecciona la respuesta correcta que indique si dicho enunciado es falso o verdadero y por qué.  
  
Enunciado:  
  
El archivo package.json se crea al ejecutar el comando “npm init” en el directorio del proyecto y sirve para lanzar un nuevo servidor.

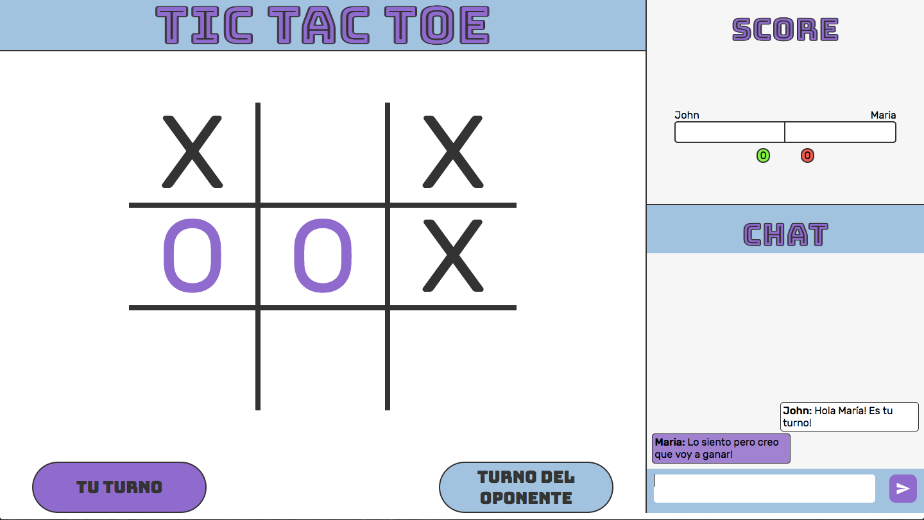


3.6 Ejercicio de Codificación

HORA DE  
**codificar**

45 MINUTOS

Node Package Manager es una herramienta muy útil al momento de crear proyectos web. Al realizar desarrollos del lado del servidor, podrás notar que siempre es necesario usar paquetes y módulos externos que complementan las funcionalidades a desarrollar. Adicionalmente NPM facilita la estructuración de los directorios y archivos que componen el proyecto, dando así orden y modularidad al desarrollo.  
  
En este ejercicio iniciaremos con el desarrollo de un juego en línea de Tic Tac Toe, que completaremos con los ejercicios de las siguientes lecciones. De este modo, la solución de cada ejercicio se convierte en la base sobre la cual debes trabajar en el siguiente. El front-end del proyecto ya está preparado y tu misión es desarrollar el Back-End con Node.js, haciendo uso de las herramientas que irás aprendiendo en esta unidad.



Contenido

Para el desarrollo de este ejercicio debes descargar el archivo base llamado codBase.zip. En él encontrarás una carpeta llamada src/ en la cual se encuentran los directorios client/ y server/, los cuales contienen los archivos del front-end y del back-end respectivamente. Adicionalmente encuentras el archivo scripts.txt que necesitarás para el desarrollo de este ejercicio.

[Descargar archivo base](https://s3.amazonaws.com/nextu-content-production/Desarrollador_Web/06_BackEnd/Ejercicios/WEB16S_C6U4L2_Ejercicio1/solucion/codBase.zip)

Instrucciones

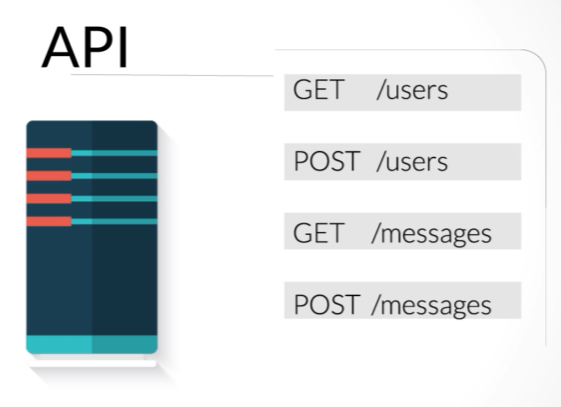
1. Crear una carpeta general en donde guardarás todo el proyecto y en su interior pega la carpeta src/ del código base.
2. Ejecutar el comando npm init desde la terminal, sobre la carpeta recién creada para generar el archivo package.json.
   * Instalar, usando npm, las siguientes dependencias:
   * babel-preset-es2015
   * babelify
   * browserify
   * express
   * jquery
   * socket.io
   * socket.io-client
   * watchify
3. Abrir el archivo package.json y en la propiedad scripts añadir las propiedades y valores indicados en el archivo scripts.txt. Recuerda usar el formato json para escribirlos adecuadamente.
4. Importar, en el archivo index.js dentro del directorio server, los siguientes módulos:
   * http
   * express
   * socket.io
5. Definir el puerto que usará el servidor y crear una aplicación express, en el mismo archivo del paso anterior.
6. Indicar a la aplicación express que los archivos estáticos están en la carpeta public.
7. Ejecutar el comando npm install para que se instalen todas las dependencias necesarias que aún no se hayan instalado.
8. Ejecutar el comando npm start y verificar que no han habido errores.  
     
   Si vas a desarrollar el proyecto en Windows ten presente que debes instalar:
   * babel-cli de manera global, usando el comando: npm install -g babel-cli
   * browserify de manera global, usando el comando: npm install -g browserify

Buenas Prácticas

Al momento de incluir módulos y paquetes para el desarrollo de las funcionalidades de un sistema, es muy recomendable separar los paquetes que se van a usar en la fase de desarrollo de los paquetes que se usarán en el proceso de producción, debido a que ésto evitará la instalación e inclusión de paquetes innecesarios en ambos ámbitos, y optimizando la carga del servidor.

1. Lección 3: Crear una aplicación Node.js
   1. Explicación y estructura del proyecto

Arquitectura: Ejemplo de un chat simple



Utilizando un JSON, lo más conveniente es usar una DB.

Estructura:



* 1. Construyendo la API
  2. Almacenamiento de los datos

/server/storage/index.js

Modulo filesystem que es parte de NodeJS

SaveData

getData

\_\_dirname 🡪 Función de NodeJS que almacena la ruta del directorio actual

Operador Ternario, ejemplo

var dataPath = dataType == 'users' ?

\_\_dirname + path.join('/data/users.json'):

\_\_dirname + path.join('/data/messages.json')



Promises 🡪 son una de las características de JS para el manejo de tareas asíncronas, es una forma de controlar los tiempos de ejecución.

Reject() 🡪 Cuando hay error

Resolve() 🡪 Cuando resuelve

* 1. Creando el frontend



Para detectar cuando un usuario envía un mensaje y renderizarlo en el HTML vamos a necesitar una serie de Funciones (en tiempo real), a través de la tecla enter o mediante el botón enviar:

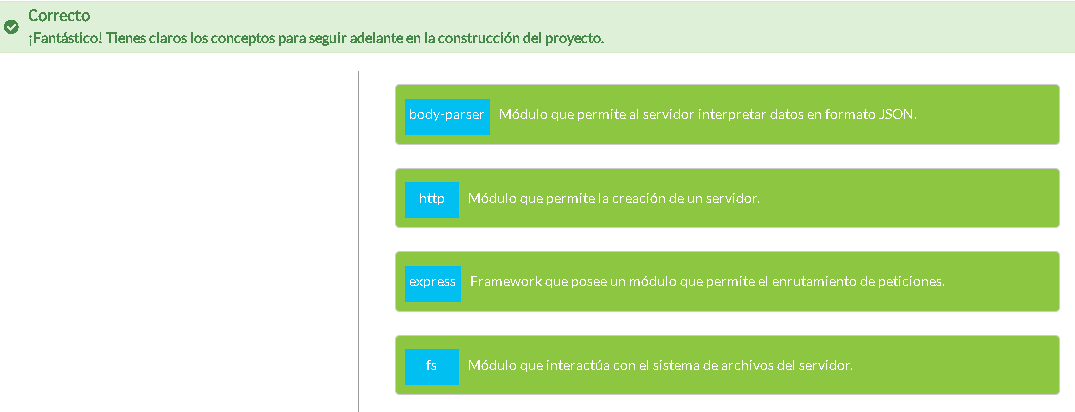
watchMessages()

renderMessages()

* 1. Integrando el front y back-end
  2. Actividad

Tiempo de pensar

En lo que llevas de esta lección has aprendido mucho. Hemos iniciado el desarrollo de un proyecto con Node y has podido conocer nuevos conceptos como las promesas y los enrutadores. Hasta ahora, hemos utilizado una serie de módulos tanto propios de Node como externos y es muy importante conocer claramente para qué sirve cada uno de ellos.  
  
Instrucciones:  
  
A continuación, verás a la izquierda una columna con un conjunto de módulos utilizados en Node para el desarrollo del proyecto que hemos iniciado y a su derecha la definición de cada módulo. Relaciónalos según corresponda:



* 1. Ejercicio de codificación

HORA DE  
**codificar**

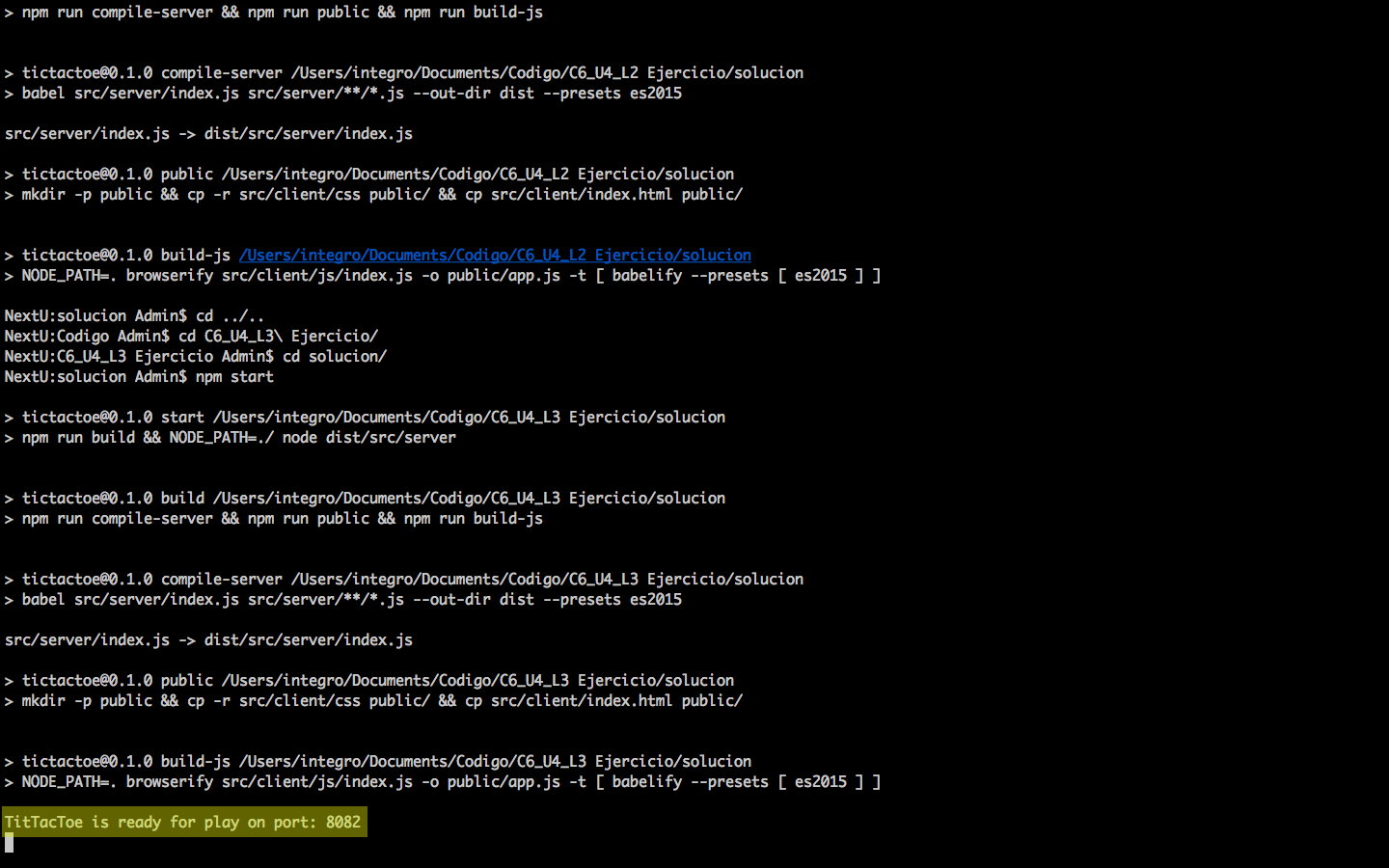
Usar Node.js como un entorno de ejecución, trae consigo muchas ventajas y permite crear aplicaciones que interactúan entre clientes y servidor de una manera muy fácil y rápida. Además es posible hacer uso de una gran cantidad de módulos propios o paquetes externos para complementar y añadir funcionalidades al proyecto. De esta manera es preciso conocer cómo se inicializan y usan los elementos de dichos módulos, para sacar su mejor provecho.  
  
En este ejercicio continuarás el desarrollo del videojuego en línea Tic Tac Toe, para lo cual necesitas el proyecto que iniciaste en el ejercicio de la lección anterior. En esta ocasión debes completar la preparación del entorno del servidor para poder iniciarlo y que esté listo para atender las peticiones.

Contenido

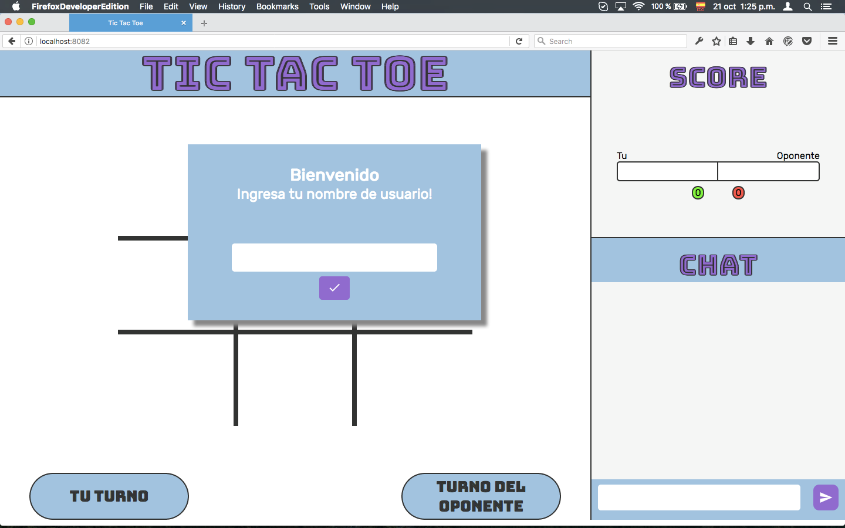
Para el desarrollo de este ejercicio debes continuar con el resultado del ejercicio de la lección anterior.

Instrucciones

1. Crea un servidor http basado en una aplicación express en el archivo **index.js** ubicado en la carpeta **server**.
2. Instancia un objeto de la librería Socket.io para que corra sobre el servidor creado en el paso anterior.
3. Ejecuta el servidor para que escuche por el puerto definido en la variable creada en el ejercicio de la lección anterior.
4. Imprime en consola, al lanzar el servidor, un mensaje que indique que éste se ha inicializado y que está escuchando por el puerto correspondiente.
5. Ejecuta el comando **npm start** sobre el directorio del proyecto y verificar que el servidor haya sido iniciado correctamente.



1. Verifica el funcionamiento correcto del servidor accediendo desde el navegador a localhost y el puerto definido. Debes acceder a la siguiente página:



##### Buenas Prácticas

* Indicar el puerto por el cual correrá la aplicación es un proceso de mucho cuidado al que se debe prestar la atención merecida. En este ejercicio hemos definido el puerto en una variable dentro del código por facilidad y mejor comprensión del concepto del puerto y el servidor. Sin embargo, es muy recomendado que el puerto sea asignado desde una variable de entorno y no desde el mismo código que ejecuta el servidor, ésto con el fin de aumentar la seguridad del sistema y evitar que posibles atacantes conozcan fácilmente el puerto por el corre la aplicación.

1. Lección 4: Realtime
2. Lección 5: ECMAScript 6
3. UNIDAD 4: Prueba