# Fundamentos de Node.js

## Prework

### Conocer qué es Node.js

#### Conocer qué es Node.js.

##### Node.js es un entorno de ejecución de JavaScript creado en el año 2010, dando la posibilidad de ejecutar código fuera del navegador. Su uso principal es para la creación de servidores (lo que conocemos hoy en día como Back-end) pero también puede utilizarse para Aplicaciones Móviles, Aplicaciones de Escritorio, Internet de las Cosas, etc.

##### Node.js es una tecnología extremadamente popular hoy en día, teniendo un aumento en el número de vacantes disponibles en el ámbito laboral. Es ideal para crear funciones a tiempo real como podrían ser un chat o un sistema de notificaciones en vivo.

#### ¿Qué es node.js?

##### Como ya lo habíamos mencionado anteriormente Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Usa un modelo de concurrencia para manejar operaciones de Entrada y Salida (E/S) sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

#### Recuerda

##### En windows los comandos tienen diversas variaciones. Si tienes problemas basta con googlear para buscar su equivalente.

### ¿Para qué sirve node.js?

#### CHATS

##### Debido a la naturaleza asíncrona y orientada a eventos de node.js, este es ideal para desarrollar chats. Si lo piensas estas son las características que necesita un chat para ser funcional.

##### Tanto el servidor como el cliente estarán siempre comunicados y en cuanto uno de los usuarios escriba un mensaje, Node.js reaccionaría al evento y de manera rápida comunicaría el mensaje al otro usuario.

#### REDES SOCIALES

##### Vivimos en un mundo dominado por las Redes Sociales, cada día hay más y más y son millones las personas las que las utilizan a diario. Nunca sabes si puedes crear la próxima Red Social exitosa, ya sea mundialmente o en su nicho y por supuesto es una muy buena idea crearla con Node.js.

##### Una red social debe funcionar a tiempo real y en ese aspecto Node.js es el rey.

#### CUALQUIER COSA QUE TE PUEDAS IMAGINAR

##### Si bien hemos mencionado la importancia de Node.js en implementaciones que conllevan tiempo real, eso no limita a que puedas implementar desde una página web hasta una tienda en línea.

### ¿Qué es un entorno de ejecución?

#### Un entorno de ejecución (runtime environment en inglés) es un estado de máquina virtual que suministra servicios para los procesos de un programa de computadora que se está ejecutando. Puede pertenecer al mismo sistema operativo, o ser creado por el software del programa en ejecución.

#### Es decir, que node.js lo que hace es proporcionar servicios extra a JavaScript mientras este se está ejecutando.

### Características de node.js

#### ¿Qué es V8?

##### V8 es un motor open-source escrito en C++ para compilar JavaScript y WebAssembly en código máquina. Esto quiere decir que traduce JavasScript a un código puramente digital capaz de ser interpretado por la CPU donde se ejecuta.

##### Este motor fue desarrollado por Google para Google Chrome y su primera versión vio la luz en 2008 junto con la primera versión del navegador.

##### Además de Chrome, el resto de navegadores basados en Chromium también usan este motor como, por ejemplo, en el nuevo Microsoft Edge. El motivo por el que este motor es tan eficiente y rápido es precisamente porque compila el código JavaScript en lugar de interpretarlo, lo cual baja drásticamente su tiempo de ejecución.

##### Además de esto, V8 se encarga de gestionar el call stack, que es la pila de ejecución; el memory heap, que es la zona de almacenamiento dinámica donde se almacenan las variables declaradas en los bloques y el garbage collector para limpiar y liberar espacio. Por último, proporciona todos los tipos de datos, operadores, objetos y funciones.

#### V8 y node.js

##### Una característica del motor V8 es que es independiente del navegador en el que está alojado. Esta característica fue clave en el rápido auge de Node.js.

##### V8 fue elegido para ser el motor que impulsó Node.js en 2009. A medida que la popularidad de Node.js creció, V8 se convirtió en un gran motor para ejecutar JavaScript del lado del servidor.

#### Tamaño y velocidad

##### Gracias a la compresión de punteros es capaz de reducir en un 40% el uso de memoria. En la práctica, esto se traduce en una carga más rápida de las webs. Estos son los números de las pruebas que ha hecho el equipo de V8 sobre webs reales:📊

###### prueba

#### JavaScript en servidor/backend

##### JavaScript siempre ha sido conocido por ser el lenguaje de programación web que actúa en el lado del cliente (Frontend), nunca ha podido interactuar por el lado del servidor (Backend). Para esta labor siempre se han encargado otros lenguajes como por ejemplo PHP, Python o Ruby.

##### Pero todo esto ha cambiado con Node.js. Ahora JavaScript puede actuar por los dos lados, tanto para el lado del cliente como para el del servidor, pudiendo así encargarse de todo por sí mismo

#### Modelo asíncrono orientado a eventos

##### Estamos muy acostumbrados a que cuando se realiza algún cambio en la página para visualizarlo tenemos que volver a recargar dicha página.

##### Por ejemplo; si se cambia el texto de este post mientras tu estás leyendo, hasta que tú no recargues la página no serías capaz de ver los cambios, y por supuesto, nadie te avisará de ello.

##### Node.js funciona en tiempo real y por ello, cada vez que ocurra algún evento este será modificado inmediatamente de forma que el cliente aun estando dentro de la página vea el cambio.

##### Esta es de hecho una de las características principales y uno de los motivos por los cuales Node.js se utiliza tanto hoy en día. Se podría considerar como la programación del futuro, una programación más eficiente, sólida e interactiva.

#### E/S sin bloqueos

##### JavaScript siempre ha tenido el problema de ser secuencial, pero las cosas han cambiado con Node.js.

##### Pero… ¿Qué significa esto de Entrada y Salida sin Bloqueos? Básicamente que las operaciones que deba realizar el entorno se pueden hacer en modo multi-tarea. Es decir, que si hay que realizar 3 procesos por ejemplo, estos tres se realizaran a la vez en lugar de ir de uno en uno.

##### ¿Y esto sirve para algo? ¡Pues claro! Mejorará muchísimo la velocidad de procesado, imagínate que tienes que realizar en una función 5 procesos y que cada uno tarda 1 segundo en procesarse. Si los procesos se realizarán de uno en uno, nos llevaría un total de 5 segundos en realizarlos todos, pero en su lugar tardará solo 1 segundo, ya que los cinco se procesarán simultáneamente.

##### Esto obviamente tiene sus límites, pero sin ninguna duda convierte a Node.js en un entorno muy eficiente.

#### Incluye NPM

##### Si eres desarrollador web, sobre todo en la parte del frontend, seguramente ya conozcas NPM. NPM es un gestor de paquetes para Javascript. Es una especie de Maven (si usas Java te sonará) para paquetes Javascript, es decir, sirve para instalar y gestionar versiones de paquetes y librerías js.

### ¿Qué es npm?

#### ¿Qué es npm?

##### npm es el gestor de paquetes que viene incluído en la instalación de Node.js, así que para instalarlo simplemente tendrás que instalar NodeJS.

##### Pero… ¿Qué es esto de un gestor de paquetes? Para los que hayan trabajado con el sistema de LINUX ya sabrán de sobras que es todo esto, pero para los que no… Un gestor de paquetes es un conjunto de utilidades o herramientas diseñadas para mejorar algo, en el caso de npm son mejoras para el entorno de Node.js.

#### ¿Para qué sirve npm?

##### Como hemos visto NPM es un gestor de paquetes, eso quiere decir que puede:

###### Descargar librerías js.

###### Actualizar en caso de nueva versión las librerías instaladas.

###### Descargar una versión en específico de la librería.

###### Gestionar las dependencias entre paquetes.

##### Una de las ventajas de npm es que todos los paquetes los descarga de un repositorio de paquetes llamado npmjs.

##### NPM usa un fichero especial llamado package.json en el que se declaran las librerías y sus versiones. Esto es muy útil ya que puedes tener este archivo con todas las librerías que necesites para que con un simple comando se descarguen todas y no tengas que estar buscándolas en sus respectivos repositorios.

##### Lo bueno es que al repositorio de librerías puedes subir las tuyas propias de tal forma que si actualizas la librería y la subes al repositorio, con un comando dentro del proyecto en el que se usa se actualiza a la nueva versión.

##### .

## Work

### Bienvenidos al mundo del Back-end.

#### Node.js es un entorno de ejecución que nos permite correr JavaScript del lado del servidor, en este programa podrás entender cómo funciona, sus bondades y lo poderoso que puede llegar a ser.

#### Objetivos

##### Darás el primer paso en el ecosistema NodeJS. Esta sesión comprende la instalación y uso básico de NodeJS así cómo algunos datos sobre la estructura de un nuevo proyecto.

#### Tabla de Contenidos

##### Ejemplo 1

###### Descargar e instalar la versión recomendada de Nodejs para tu sistema operativo.

##### Ejemplo 2

###### Conocer la línea de comandos de Node.js para ejecutar archivos con extensión .js.

###### Creamos un archivo llamado index.js en nuestra carpeta de trabajo (puedes utilizar la terminal si lo deseas). Después vamos a copiar el siguiente código que es la implementación del problema FizzBuzz:

function fizzBuzz() {

for (let i = 1; i <= 100; i++) {

if (i % 3 === 0 && i % 5 === 0) {

console.log('Fizzbuzz');

} else if (i % 3 === 0) {

console.log('Fizz');

} else if (i % 5 === 0) {

console.log('Buzz');

} else {

console.log(i);

}

}

}

fizzBuzz();

###### Ahora guarda el archivo, abre tu terminal y ejecuta el comando node index.js. Verás que en tu terminal aparecera el resultado de ejecutar el programa.

##### Ejemplo 03

###### Conocer y trabajar con módulos del core y módulos de terceros.

###### Módulos del core

Node.js tiene módulos que ofrece de manera nativa, es decir, no requiere instalación de nada mas para ser utilizados (los módulos compilados se encuentran en la instalación de Nodejs en la carpeta lib/), a estos se les conoce como modulos del core.

Node ofrece una gran cantidad de modulos, podemos encontrarlos en su documentación:

###### ENTENDAMOS LOS MODULOS

Son herramientas o grupo de funciones prefabricadas que nos facilitan la construcción de nuestro código.

###### ¿COMO SE USAN?

Imagenemos que tenemos que ir a la escuela, y nuestra mochila es nuestro documento de javascript "mochila.js", ahora imaginemos que cada materia es un modulo, y ese dia tendremos que ver 3 materias, matematicas, fisica y quimica. Entonces... yo necesito meter cada libro en la mochila para poder llevar esas materias.

Ahora pensemos que la manera de realizar la acción de meter los libros a la mochila requiere un comado, y ese comando es el siguiente :

var MATEMATICAS = require('LIBRO\_DE\_MATEMATICAS');

var FISICA = require('LIBRO\_DE\_FISICA');

var QUIMICA = require('LIBRO\_DE\_QUIMICA');

Entonces en mi archivo mochila.js ya tengo 3 variables que cada una esta cargando una materia.

Ahora si una materia es igual a un modulo, esto quiere decir que tengo 3 modulos cargados en mi archivo, si lo pasamos al mundo de node, eso quiere decir que puedo cargar los modulos mandadolos llamar con esta sintaxis:

var VARIABLE\_DONDE\_SE\_GUARDARA = require('NOMBRE\_DEL\_MODULO');

Basicamente la función "require()" es la que se encarga de cargar los modulos en nuestro archivo, a este proceso se le conoce como : INYECCION DE DEPENDENCIAS.

Existen 3 tipos de modulos que podemos utilizar en nuestros proyectos:

MODULOS INTEGRADOS DE NODE.JS

MODULOS DE TERCEROS NPM

MODULOS CREADOS

###### MODULOS INTEGRADOS DE NODE.JS

Estos modulos ya vienen integrados en node.js no requieren ningún tipo de instalación, solo los mandamos llamar directo y podremos hacer uso de ellos.

Si revisamos la lista de modulos, encontraremos el modulo OS, este modulo nos permite revisar los datos del hardware de nuestra computadora, como uso de CPU, memoria, cuantos cores tiene de procesador etc.

Si yo quiero saber cuantos CPU tiene mi computadora, con el modulo de OS, el primer paso seria cargar el modulo en mi archivo

var os = require('os');

El siguiente paso seria revisar la documentación para determinar cual función del modulo OS es la que me permite ver los CPUS (Procesadores)

podemos observar que para saber cuantos CPU tengo, se usa la funcion cpus()

lo siguiente entonces es llamar la funcion cpus() y almacenaremos en una variable que llevara el nombre de "misCpu" tmb imprimire la variable en consola para ver el resultado final:

--- EJEMPLO COMPLETO ---

var os = require('os');

var misCpu = os.cpus();

console.log(misCpu);

###### USANDO MODULOS DE TERCEROS CON NPM

Node.js tiene un numero limitado de modulos que nos ofrece de manera nativa como lo vemos en la documentación, por ello npm viene a solucionarnos la vida, ya que en sus arcas, podemos encontrar millones de modulos para hacer cualquier cosa que querramos, pero a diferencia de los modulos de node.js, los de npm requieren una serie de procesos para incorporarlos a nuestros proyectos.

Sigamos los pasos a continuación para poder lograr esto:

.

## Postwork.