# Node installation

* Node.js runtime
  + Performance counters
  + Event tracing (ETW)
* npm package manager
* Online documentation shortcuts
* Add to PATH
  + Node.js and npm
  + npm modules

# Node Lite-Server

## Install Lite-Server

- Launch the Bash command line

- Install lite-server:

**$ npm install -g lite-server**

## New Project

* Create a directory
* Create a Node project:

**$ npm init –y**

package.json:

{

"name": "NodeJS",  
"version": "1.0.0",  
"description": "",  
"main": "index.js",  
"scripts":

{  
"test": "echo \"Error: no test specified\" && exit 1"  
},

"keywords": [],  
"author": "",  
"license": "ISC"

}

* Create an index.html file with the text "Hola Mundo" in the project
* Modify the package.json so in the "start" the lite-server will be launched:

"scripts":  
{  
"start": "lite-server"  
},

## Start and close the server

* Launch the Project:

**$ npm start**

The server is launched and listenning at <http://localhost:3000/>

All plugins introduced at the dependencies JSON of the package.JSON file, are taken from de Node Package Manager (npm). [www.npmjs.com](http://www.npmjs.com)

**$ npm install** 🡪 executes package.JSON and install all plugins writen in the sub-JSON dependencies

[**https://docs.npmjs.com/getting-started/semantic-versioning**](https://docs.npmjs.com/getting-started/semantic-versioning)

<https://blog.risingstack.com/your-first-node-js-http-server/>

<https://jesuslc.com/>

<https://geekytheory.com/node-js-primeros-pasos-y-hola-mundo>

YEOMAN 🡪 scaffolding

Comando: **yo**

Comprar un dominio, configurar las DNS para que apunten al router de la máquina servidor y permitir la entrada por el puerto 80, además de establecer un IP fija.

# NodeProjectExample

Creamos la carpeta del nuevo proyecto, entramos en ella y ejecutamos el primer comando:

1. **$ mkdir newProject**
2. **$ cd newProject/**
3. **$ npm init**
4. **$ npm install bootstrap**
5. **$ npm install jquery**
6. **$ npm install Popper.js**

The package.json file has a key called: “dependencies” with the required modules for the project. We can copy the package.json file in a new Project and then execute the command line **$ npm install** 🡪 automatically, the resources will be updated to the last version according to the semantic-versioning description of the dependencies sub-json.

<https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP>

<https://www.joezimjs.com/javascript/3-ways-to-parse-a-query-string-in-a-url/>

Node JS

Entorno de programación con numerosas librerías JAXER, RingoJS, EJScript, AppengineJS

Aplicaciones NodeJS:

Entradas en cola: Si recibe una gran diversidad de datos concurrentes se debe evitar el bottleneck de la operación query de bloqueo DB. La solución es reconocer el comportamiento del cliente antes de que los datos se escriban realmente en la base de datos.

**Streaming de datos:** En las plataformas web más tradicionales, las solicitudes HTTP y las respuestas se tratan como un evento aislado; De hecho en realidad son arroyos Esta observación se puede utilizar en Node.js para crear algunas características interesantes. Por ejemplo, es posible procesar archivos mientras se siguen cargando, ya que los datos entran a través de un flujo y podemos procesarlo de una manera en línea. Esto podría hacerse para la codificación de audio o vídeo en tiempo real, y para el proxy entre diferentes fuentes de datos.

**Proxy**: Node.js se emplea fácilmente como un proxy del lado del servidor donde puede manejar una gran cantidad de conexiones simultáneas de una manera no bloqueante. Es especialmente útil para la representación de diferentes servicios con diferentes tiempos de respuesta, o la recopilación de datos de múltiples puntos de origen.

**Monitoreo de aplicaciones:** Otro caso de uso común en el que Node.js encaja perfectamente es en el seguimiento de los visitantes de un sitio web y la visualización de sus interacciones en tiempo real. Un ejemplo sería: poder reunir estadísticas en tiempo real de su usuario o incluso moverlo al siguiente nivel introduciendo interacciones específicas con sus visitantes.

**Asíncrono y controlado por eventos:**

Todas las API de la biblioteca Node.js son asíncronas, es decir, sin bloqueo. Significa esencialmente que un servidor basado en Node.js nunca espera a que una API devuelva datos. El servidor se mueve a la siguiente API después de llamarla y un mecanismo de notificaciones de Eventos de Node.js ayuda al servidor a obtener una respuesta API llamada anteriormente.

**Único subproceso pero altamente escalable:** Node.js utiliza un único modelo de roscado con el evento de bucle.

**Sin búfer:** las aplicaciones de Node.js nunca almacenan cualquier dato. Estas aplicaciones simplemente dan salida a los datos en fragmentos.

* JavaScript del lado del servidor; basado en eventos. Node ejecuta javaScript utilizando el motor V8 de Google. Este motor permite a Node proporcionar un entorno de ejecución del lado del servidor que compila y ejecuta javaScript, a velocidades increíbles. El aumento de velocidad es importante debido a que V8 compila JavaScript en código de máquina nativa en lugar de interpretarlo o ejecutarlo como bytecode.
* NodeJS trabaja en tiempo real, recibiendo datos instantáneamente sin pasar de una página a otra; es multiusuario, se comunica con JSON de manera muy eficiente sin realizar tantas conversiones de un lenguaje a otro. Puede trabajar con conexiones simultáneas , ayudar en el tráfico de la base de datos.

Una cosa a tener en cuenta sobre los módulos es que no comparten el alcance, por lo que si desea compartir una variable entre los diferentes módulos, deben incluirlo en un módulo separado que es requerido por los otros módulos. Otra cosa interesante que debe recordar es que los módulos se cargan sólo una vez, y después se almacenan en memoria caché por nodo.

A diferencia del navegador, Node.js no tiene un objeto global de ventana, sino que tiene otros dos: “globals” y “process”. Sin embargo, se debe evitar seriamente agregar propiedades en los dos.

Una función de Callback se llama al finalizar una tarea determinada. Todas las API de Node se escriben de tal manera que admiten Callbacks.

Por ejemplo, una función para leer un archivo puede comenzar a leer el archivo y devolver el control al entorno de ejecución inmediatamente para que la siguiente instrucción se pueda ejecutar. Una vez que la E/S del archivo esté completa, llamará a la función Callbacks y ésta pasará el contenido del archivo como un parámetro.