Node JS

Fundamentos

Arturo Rivas Rojas

25 de enero de 2019

Índice general

CAPÍTULO 1

¿Qué es Node.js?

Es un ambiente de ejecución de código de JavaScript. Cambió el paradigma con el que se solía trabajar a JavaScript, pues este estaba aprisionado a los navegadores.

Caracteristicas:

- Libre
- Real Time
- Orientado a eventos
- Asíncrono
- Multi-plataforma
- Server Side
- Robusto
- Escalable
- Expandible
- No bloqueante

Es un aplicativo que hace uso de librerías ára procesar y ejecutar código JavaScript.

Node.js utiliza el motor de JavaScript de Google Chrome, V8.

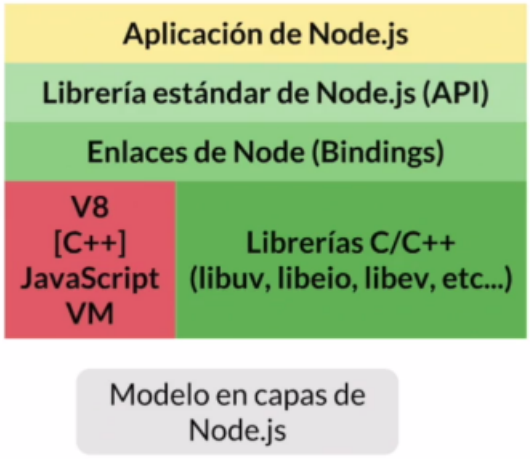

Figura 1.1: Modelo de capas de Node.js

1.1. Event Loop

Es un ciclo infinito que constantemente toma eventos de la cola y los comunica a otros procesos. Y cuando se genera la respuesta lo retorna al motor de JavaScript.

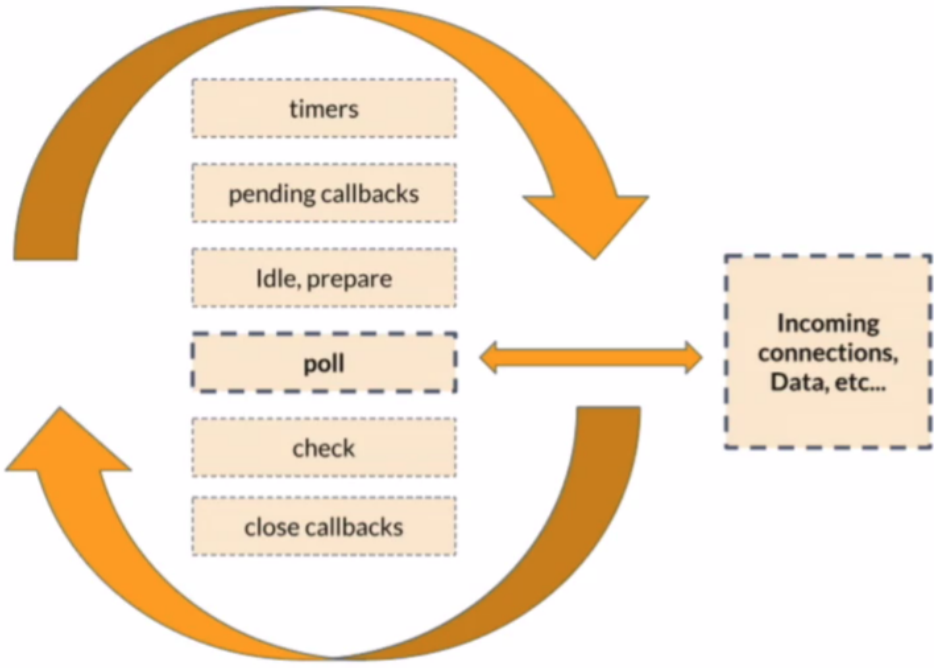

Figura 1.2: Composición del Event loop

Poll es el componente mas importante pues es mediante el cual se comunican las librerias de node.

1.2. Non Real-Time

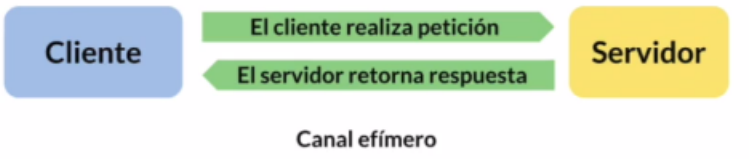

Figura 1.3: aplicación que no es de tiempo real

Ejemplos:

- Servidor API
- Servidor de Sitio web
- Servidor de Servicios de red
- Servidor de Proxy

1.3. Real-Time

Figura 1.4: Comunicación en aplicaciones de tiempo real

La comunicación se vuelve bidireccional, haciendo que el canal de comunicación deje de ser efimero y se vuelva persistente.

Ejemplos:

- Comunicación en tiempo real
- Monitoreo de datos
- Juegos en Linea

CAPÍTULO 2

Instalación

Para istalar la version mas reciente de node.js se utilisa **nvm** (node version manager). Y una vez istalado el nvm basta con ejecutar el siguiente comando

```
sudo nvm install --lts
```

Este comando instalara la version mas reciente con soporte a futuro.

CAPÍTULO 3

Node Package Manger NPM

Es un componente se node que nos permite administrar las dependencias de node, además de inicializar proyectos.

El comando **npm init** nos presentará un formulario para la creación de un nuevo poryeto. Cuuando se termine de ejecutar el comando se habra creado un archivo package.json que contine la configuración de nuesto proyecto.

Al estar trabajando con npm el archivo package.json se altera constantemente sin que nos demos cuenta.

npm ls listara las dependencias de nuestro proyecto.

npm install [dependencia] nos permite instalar nuevas dependencias.

npm uninstall remueve dependencias.

CAPÍTULO 4

Modulos

Un modulo es un conjunto de funciones que se desean reutilizar en otros programas.

4.1. Locales

El formato para declarar un modulo local es el siguiente.

```
//hello.js
module.exports = {
    // declaración de un atributo que contiene una función
    sayHello: name => `Hello to ${name}`;
    }
}
```

Para importar un modulo local se hace lo siguiente:

```
// alias del modulo = ubicación del modulo la extensión solo es necesaria si existe un archivo con el m
const hello = require("./hello.js")
// función declarada en el modulo.
hello.sayHello('Arturo')
```

Si require solo recibe la dirección de un directorio y no de un archivo, este busca dentro de ese directorio un archivo que se llame index con cualquir extensión.

4.2. Externos

Una vez que se hayan instalado las dependencias mediante **npm**.

npm install nos permite instalar todas las dependencias que esten declaradas en el package.json

El formato para importar modulos externos que ya se instalaron media npm.

```
const cowsay = require("cowsay")
```

La forma en la que se llaman a las funciones es la misma tanto para los modulos locales como externos.

CAPÍTULO 5

Modulos Nativos

5.1. File System

Es el modulo que nos permite manejar los archivos del sistema.

```
const fs = require("js")
```

5.1.1. fs.readFile(path[, option], callback)

Es una función que nos permite leer elarchivo con la ruta **path** y con larespuesta obtenida ejecutar la función**callback** que tiene la forma:

(error, data) => {…}

```
fs.readFile("./src/ejemplo.txt", "utf8", (err, data)=>{
    if(err) throw err;
    console.log(data)
})
```

El código anterior imprime todo el contenido del archivo ejemplo.txt

5.1.2. fs.writeFile()