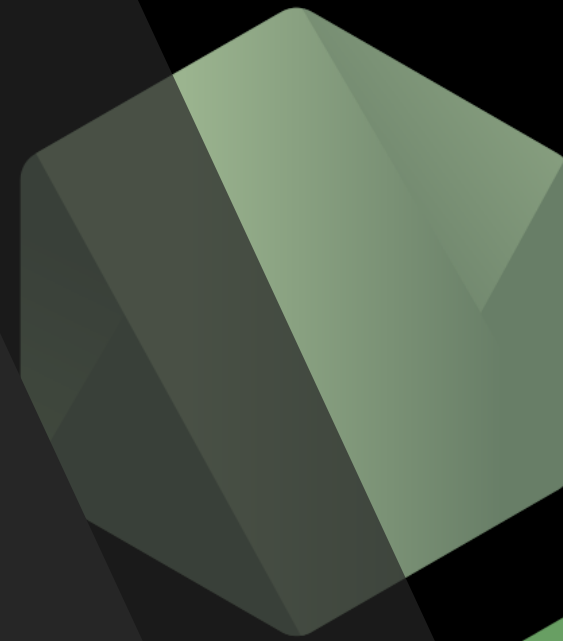


Ing. Rodrigo J. Garcia Hoyos

Node.js

Servidor en Tiempo Real



JavaScript



- JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas

Historia JavaScript



Primer versión del lenguaje

Netscape Navigator (dominante de la web en ese momento) decide agregar interactividad en las paginas HTML con un lenguaje de programación liviano. Contrata a Brendan Eich para implementarlo y termina su primer versión en 10 días con el nombre Mocha.

**1 Mayo
1995**

Nuevo nombre: LiveScript

Se modificó su nombre a uno mas del palo de Netscape: LiveScript.

**1 Sept
1995**

Netscape y Sun hacen un acuerdo de licencia para darle el nombre Javascript

El nombre JavaScript lograba el roll original del lenguaje: Java de Sun proveia la construcción de bloques de gran escala para la Web, mientras que JavaScript era el "pegamento" conectando los bloques.

**1 Mayo
1995**

Node.js



Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla



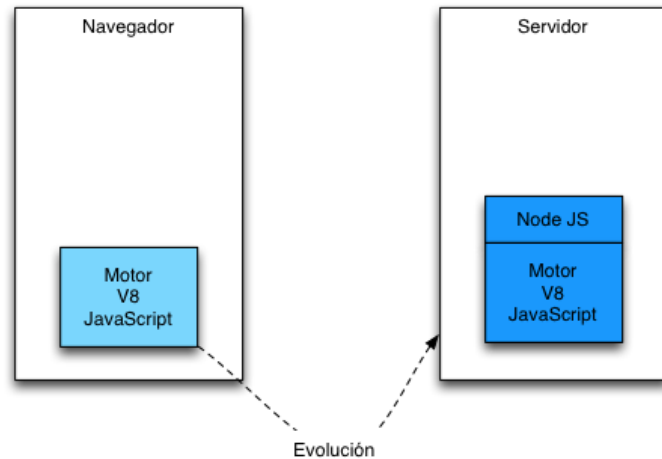
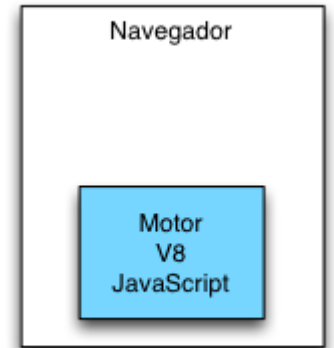
Arquitectura



libuv

El motor V8 de google

Node.js esta basado en el motor V8 de Javascript de Google. Este motor está diseñado para correr en un navegador y ejecutar el código de Javascript de una forma extremadamente rápida.

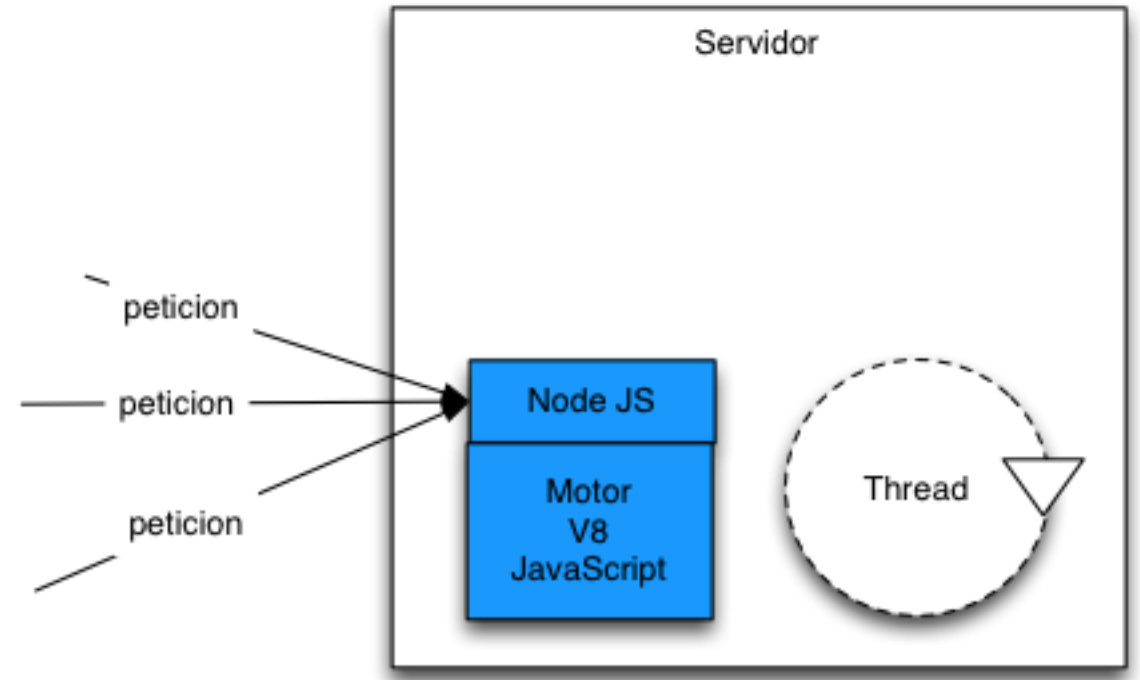


La tecnología que está detrás de Node.js permite ejecutar este motor en el lado del servidor abriendo un nuevo abanico de posibilidades en cuanto al mundo de desarrollo se refiere.

Para ello el entorno de Node.js ha desarrollado un conjunto amplio de librerías que no tienen nada que envidiar a otras plataformas. Por otro lado se han eliminado algunas funcionalidades que en el entorno de servidor no tenían sentido como por ejemplo el uso de Document Object Model.

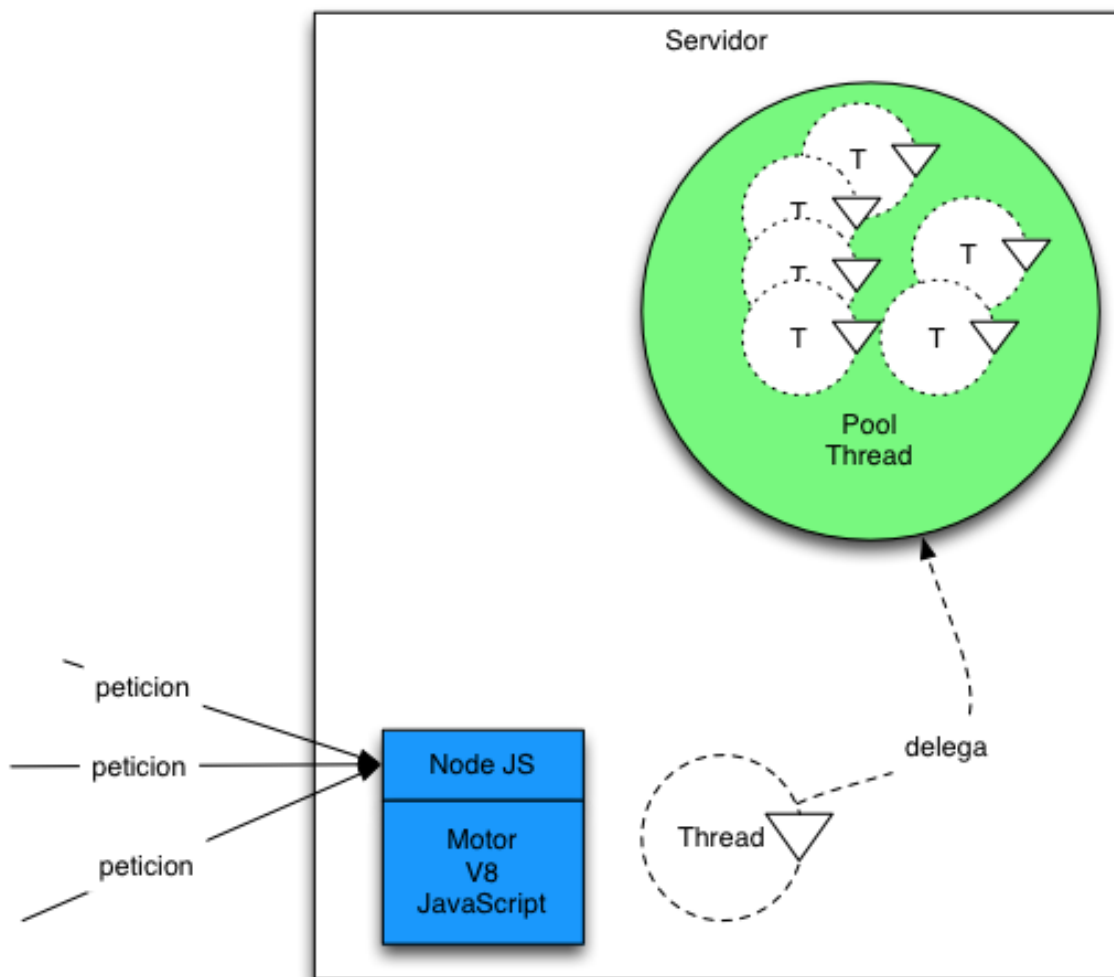
Node.js y Threads

Node.js trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar.



La primera duda que nos surge es: ¿Qué sucede si ese hilo se pone a realizar una tarea que lleva X tiempo y queda bloqueado?. Esa es una buena pregunta y la respuesta es que Node.js gestiona todas sus tareas de una forma asincrónica. Esto nos puede sonar mucho a AJAX. Ahora bien ¿Cómo funcionan esas peticiones asíncronas exactamente?.

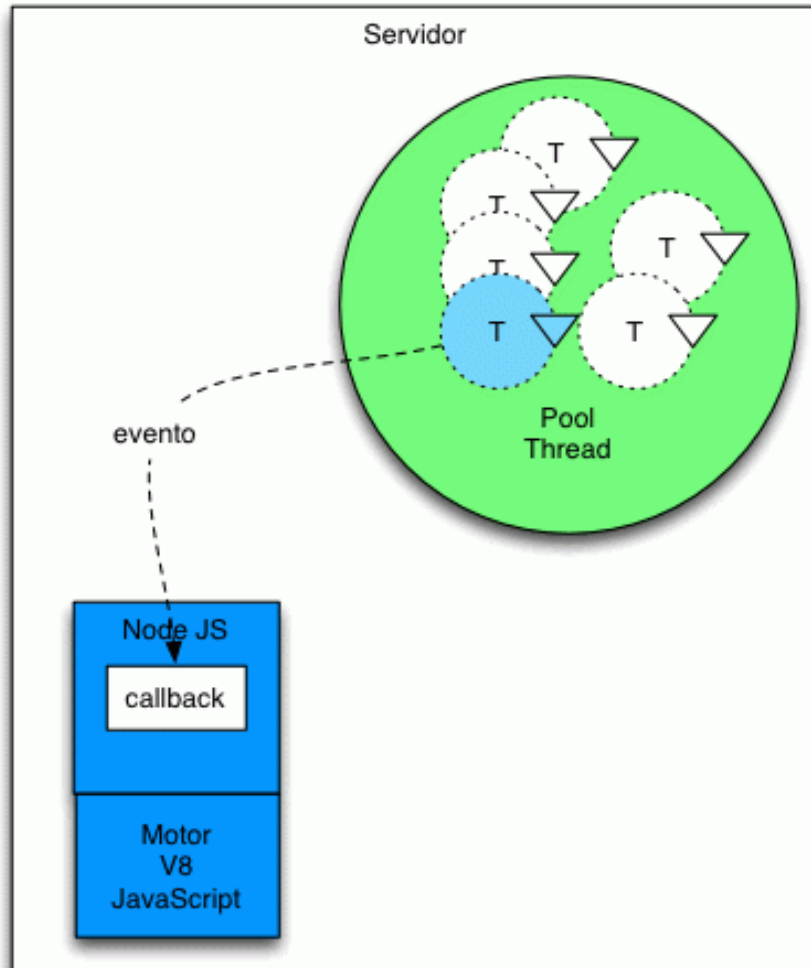
Node y tareas asíncronas



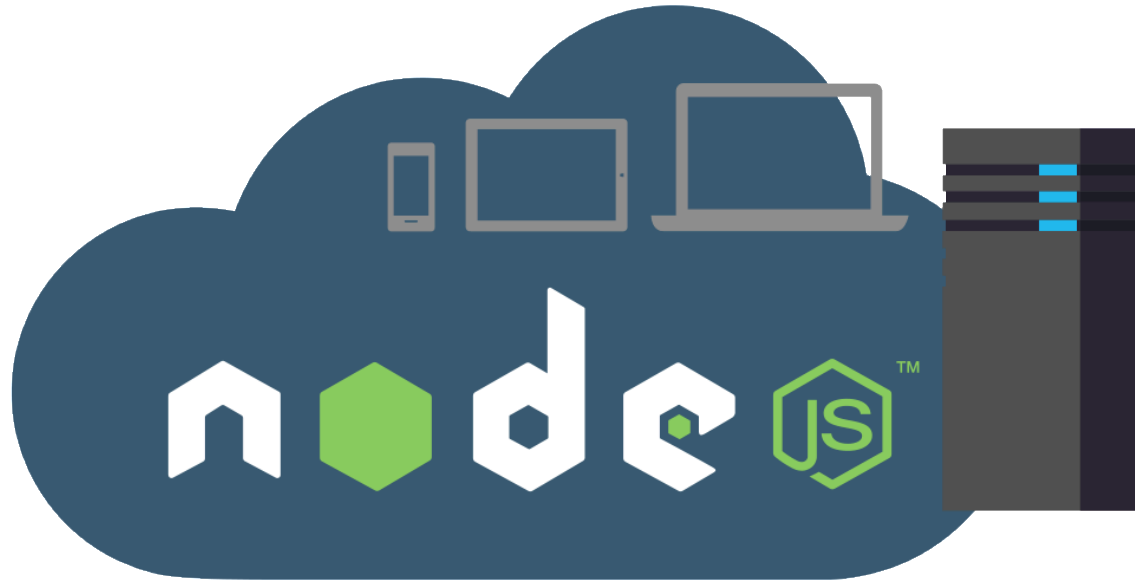
Para poder trabajar de una forma óptima Node.js delega todo el trabajo en un pool de threads. Este pool de threads esta construido con la librería libuv. Esta librería dispone de su propio entorno multithread asíncrono. Node.js envía el trabajo que hay que realizar al pool.

Libuv realizará a través de alguno de sus threads el trabajo encomentado. Una vez que el trabajo haya sido completado libuv emitirá un evento que será recibido por Node.js.

Node y tareas asíncronas



Recibido el evento una función de callback se encargará de terminar de procesarlo. Por eso cuando trabajamos con Node.js prácticamente toda la programación es asíncrona y se parece tanto a las clásicas llamadas AJAX.



Hola mundo

- Node - > `console.log('Hola, Mundo!')`

```
var http = require('http');
var server = http.createServer();

function control(petic, resp) {
    resp.writeHead(200, {'content-type': 'text/plain'});
    resp.write('Hola, Mundo!');
    resp.end();
}

server.on('request', control);console.log('Server
running');server.listen(8080);
```

```
var http = require('http');
http.createServer(function(peticion, respuesta){
  var codigo_html = '<html> <head> <title> Ejemplo de hola mundo </title>
</head> <body> Hola mundo </body> </html>';
  respuesta.writeHead(200, 'text/html');  respuesta.end(codigo_html);
}).listen(3000, '127.0.0.1');
console.log('El servidor esta funcionando correctamente en
http://localhost:3000/');
```

```
var http = require('http');
var url = require('url');
var fs = require('fs');
var mime_types = {
  'js' : 'text/javascript',
  'html' : 'text/html',
  'css' : 'text/css',
  'jpg' : 'image/jpg',
  'gif' : 'image/gif',
  'png' : 'image/png'
};
http.createServer(function(peticion, respuesta){
  var path_nombre = (url.parse(peticion.url).pathname == '/') ? '/index.html' : url.parse(peticion.url).pathname; var
  ruta_a_archivo = 'contenido' + path_nombre; fs.exists(ruta_a_archivo, function(existe){ if(existe){
  fs.readFile(ruta_a_archivo, function(error, contenido_archivo){ if(error){ respuesta.writeHead(500, 'text/plain');
  respuesta.end('Error interno.')} else{ var extension = ruta_a_archivo.split('.').pop(); var mime_type =
  mime_types[extension]; respuesta.writeHead(200, {'Content-Type': mime_type});
  respuesta.end(contenido_archivo); } }); } else{ respuesta.writeHead(404, 'text/plain');
  respuesta.end('Error 404. El enlace no existe o ha dejado de existir.')} } })).listen(3000, '127.0.0.1');console.log('El
servidor esta funcionando correctamente en http://localhost:3000/');
```

- `<!DOCTYPE><html><head> <title> Ejemplo de Node.js </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
></head><body> <h1> Info </h1> Index Info
 <p> Esta es la página info </p></body></html>`

- `<!DOCTYPE><html><head> <title> Ejemplo de Node.js </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
></head><body> <h1> Index </h1> Index Info
 <p> Bienvenido/a a la página principal
</p></body></html>`