Daniel Santos Lara

ESTUDIANTE DEL ISTITUTO TECNOLOGICO DE TIJUANA.

Ver perfil

Acerca de Node.js

Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para construir aplicacionesen red escalables. En la siguiente aplicación de ejemplo "hola mundo", se pueden manejar muchas conexiones concurrentes. Por cada conexión el callback será ejecutado, sin embargo si no hay trabajo que hacer Node.js estará durmiendo.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola Mundo');
});

server.listen(port, hostname, () => {
  console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
});
```

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node.js están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi ninguna función en Node.js realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node.js.

Node.js tiene un diseño similar y está influenciado por sistemas como Event Machine de Ruby ó Twisted de Python. Node.js lleva el modelo de eventos un poco más allá, este presenta un bucle de eventos como un entorno en vez de una librería. En otros sistemas siempre existe una llamada que bloquea para iniciar el bucle de eventos. El comportamiento es típicamente definido a través de callbacks al inicio del script y al final se inicia el servidor mediante una llamada de bloqueo como EventMachine::run(). En Node.js no existe esta llamada. Node.js simplemente ingresa el bucle de eventos después de ejecutar el script de entrada. Node.js sale del bucle de eventos cuando no hay más callbacks que ejecutar. Se comporta de una forma similar a JavaScript en el navegador - el bucle de eventos está oculto al usuario. HTTP es ciudadano de primera clase en Node.js, diseñado con operaciones de streaming y baja latencia en mente. Esto hace a Node.js candidato para ser la base de una librería o un framework web. Solo porque Node.js esté diseñado sin hilos, no significa que usted no puede aprovechar los múltiples cores de su sistema. Procesos hijos pueden ser lanzados usando nuestra API [child_process.fork()][], la cual está diseñada para comunicarse fácilmente con el proceso principal. Construida sobre la misma interfaz está el módulo [cluster][], el cual permite compartir sockets entre procesos para activar el balanceo de cargas en sus múltiples cores.