Node.js

Iniciando com Express.js e TypeORM

Quem Sou Eu?

- Gustavo Leão Nogueira de Oliveira
- Formado em **Análise e Desenvolvimento de Sistemas** no Senac
- Atualmente trabalhando como **Desenvolvedor Full-Stack** (backend/devops)

O que é node.js

- Criado pro Ryan Dahl em 2009
- Node.js é um ambiente de servidor de código aberto.
- Node.js permite executar JavaScript no servidor.
- Como um tempo de execução JavaScript assíncrono baseado em eventos, o Node.js foi projetado para construir aplicativos de rede escalonáveis.
- É um interpretador JavaScript desvinculado do navegador

NPM

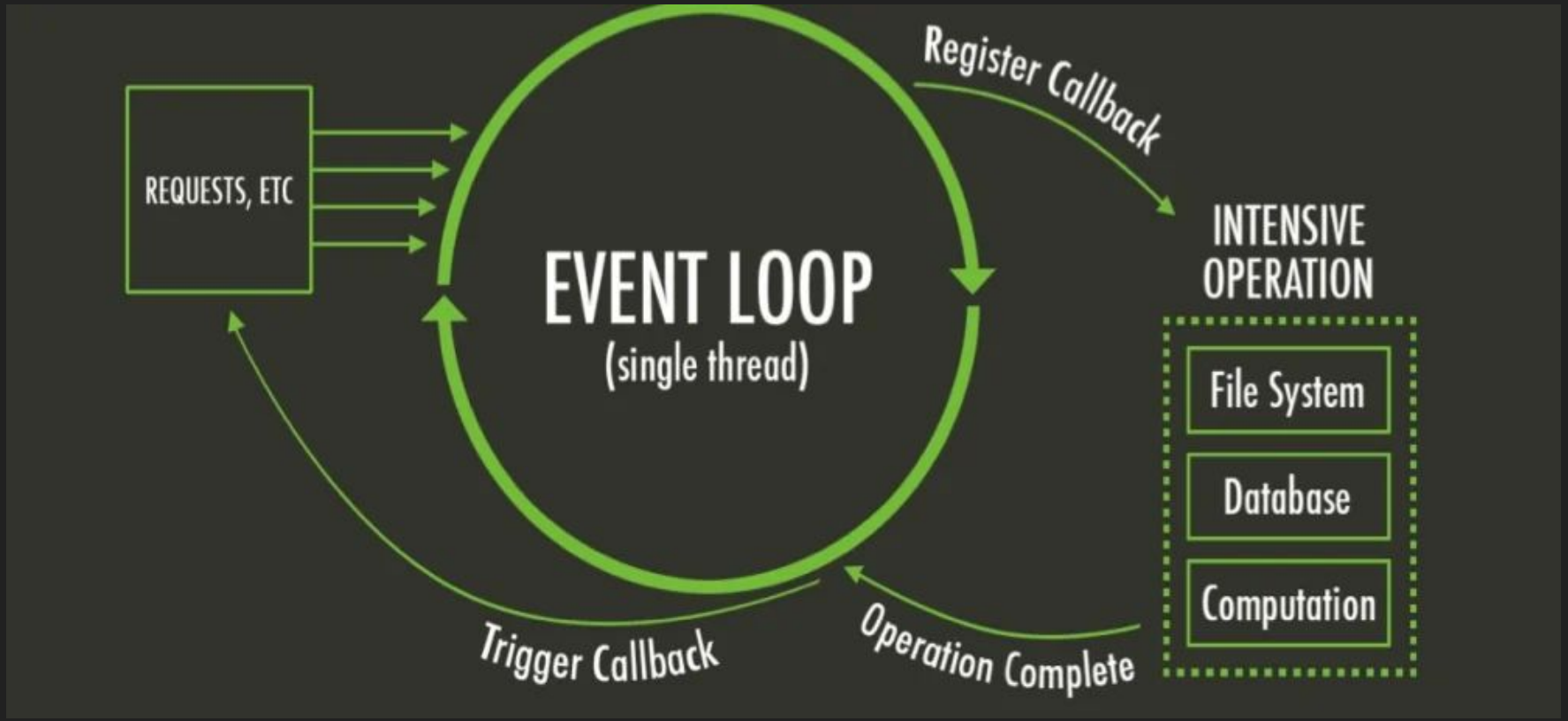
- É o Gerenciador de Pacotes do Node que é instalado juntamente com o Node, no momento de sua instalação.
- Execute o comando npm -v e verifique se é retornada a versão instalada no npm

Hello World

- Em uma pasta de sua preferência, crie um arquivo index.js e escreva dentro o comando console.log(“Oi Mundo”);
- Execute no terminal o comando node index.js
-

Primeira Api

Event Loop


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

Event Loop

- O event loop na V8 do Node.js é um componente fundamental que permite que o Node execute código de forma assíncrona e não bloqueante. Ele funciona da seguinte maneira:
- Loop de Eventos: O event loop é um loop contínuo que espera e processa eventos.
- Eventos: Eventos podem ser operações assíncronas, como leitura de arquivos, solicitações de rede, timers e interações de E/S.
- Callbacks: Cada evento é associado a um callback, uma função que será executada quando o evento for processado.
- Pilha de Chamadas: O event loop mantém uma pilha de chamadas que contém as funções em execução.
- Modelo Não Bloqueante: O Node executa operações de I/O de forma não bloqueante, permitindo que outras tarefas sejam executadas enquanto aguarda a conclusão da operação.
- Pilha de Mensagens: Quando um evento é concluído, seu callback é colocado em uma pilha de mensagens para ser processado pelo event loop.
- Microtarefas e Tarefas de Tempo: O event loop também lida com microtarefas e tarefas de tempo, permitindo a execução de código agendado e operações de alta prioridade.
- Em resumo, o event loop na V8 do Node.js permite que o servidor execute operações de forma assíncrona e não bloqueante, garantindo alta eficiência e escalabilidade em aplicações de rede e E/S intensivas.

Características

- No exemplo de servidor que acabamos de construir, muitas conexões podem ser tratadas simultaneamente. A cada conexão, uma função de callback é disparada, mas se não houver trabalho a ser realizado, o Node.js ficará inativo.
- Isso contrasta com o modelo de simultaneidade mais comum de hoje, no qual threads de sistema operacional são utilizadas.
- A rede baseada em thread é relativamente ineficiente e muito difícil de usar. Além disso, os usuários do Node.js estão livres da preocupação de travar o processo, já que não há travas.
- Quase nenhuma função no Node.js realiza I/O diretamente, então o processo nunca bloqueia. Como nada bloqueia, sistemas escaláveis são muito fáceis de desenvolver em Node.js.

Características

- O Node.js é semelhante em design e influenciado por sistemas como Ruby’s Event Machine e Python's Twisted.
- Node.js leva o modelo de evento um pouco mais longe. Ele apresenta um loop de eventos como uma construção de tempo de execução em vez de uma biblioteca.
- Em outros sistemas, há sempre uma chamada de bloqueio para iniciar o loop de eventos. Normalmente, o comportamento é definido por meio de retornos de chamada no início de um script e, no final, um servidor é iniciado por meio de uma chamada de bloqueio como EventMachine::run().
- No Node.js, não existe essa chamada start-the-event-loop. O Node.js simplesmente entra no loop de eventos após executar o script de entrada. O Node.js sai do loop de eventos quando não há mais callbacks para executar. Esse comportamento é como o JavaScript do navegador - o loop de eventos é escondido do usuário.

Obrigado!

@gusleaono1

/gusleaooliveira