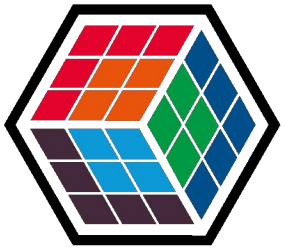
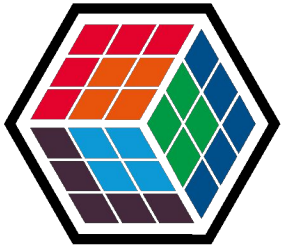


THE DEVELOPER'S CONFERENCE

Trilha – Node.Js

Kamila Santos Oliveira
Software Developer



THE DEVELOPER'S CONFERENCE

Event Loop

Entendendo o node por baixo dos panos

Kamila Santos Oliveira

21 anos,

Dev na Cognizant,

~ 3 anos na área,

Graduanda em ciência da computação

THE DEVELOPER'S CONFERENCE

Agenda

THE DEVELOPER'S CONFERENCE

- Event Loop
- Call Stack
- Multi threading
- Task Queue
- Programação Assíncrona

Agenda

THE DEVELOPER'S CONFERENCE

- Event Notification
- Event-Carried State Transfer
- Event-Sourcing
- CQRS (Command Query Responsibility Segregation)

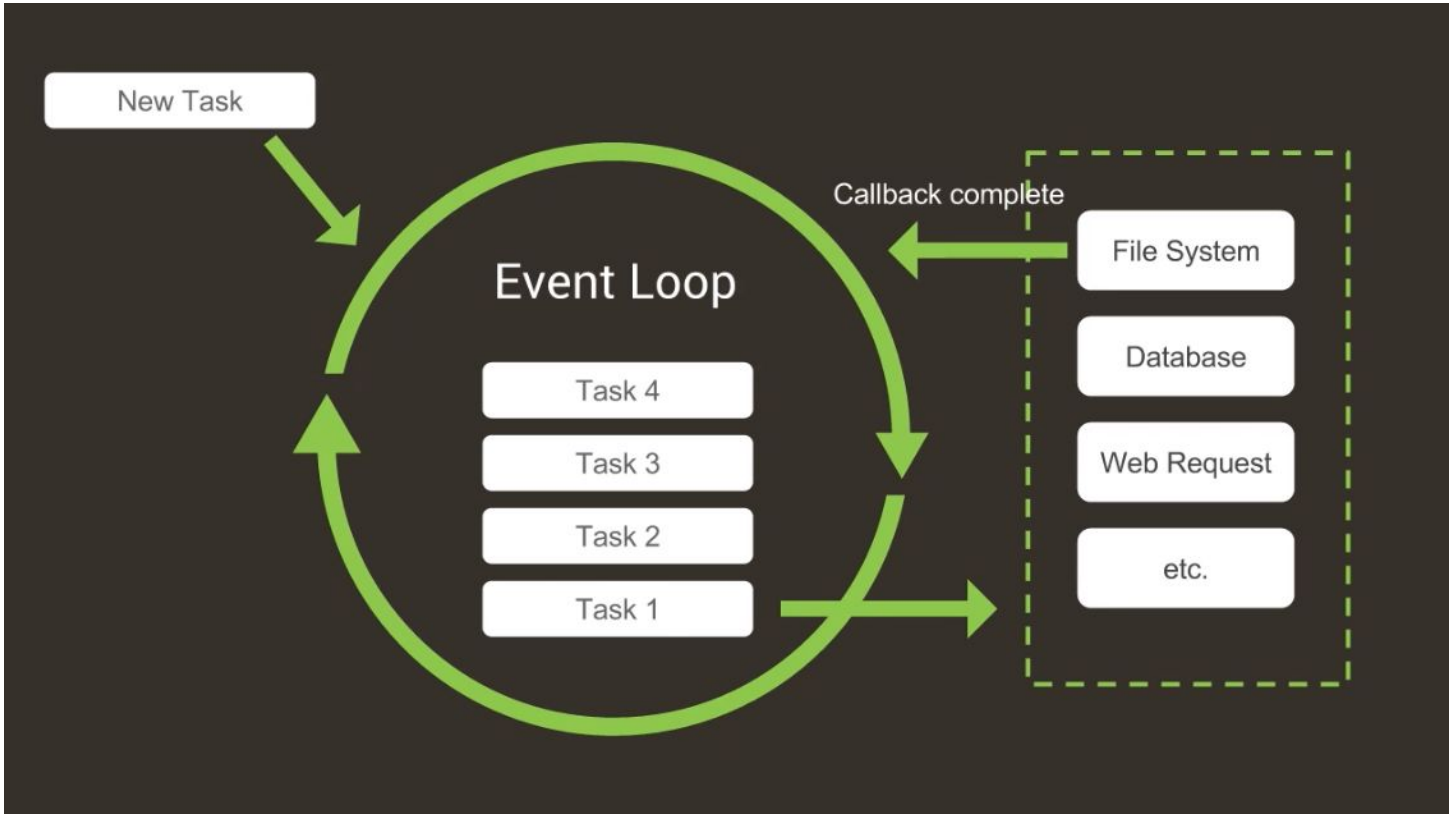
THE
DEVELOPER'S
CONFERENCE

EventLoop

THE DEVELOPER'S CONFERENCE

EventLoop

Captura e emite eventos para o sistema.
Loop infinito que a cada ação verifica na sua fila de eventos se aquele específico já foi emitido, quando isto ocorre, ele vai para a fila de executados.


New Task
Event Loop
Task 4
Task 3
Task 2
Task 1
Callback complete
File System
Database
Web Request
etc.

THE
DEVELOPER'S
CONFERENCE

Call Stack

THE DEVELOPER'S CONFERENCE

Estrutura de dados (pilha), que guarda em que parte do programa estamos.

Call Stack

THE DEVELOPER'S CONFERENCE

Ao entrar numa função vai para o topo da stack e ao retornar da função sai do topo da stack.

Call Stack

THE DEVELOPER'S CONFERENCE

Funciona como uma estrutura LIFO (Last in,Fisrt Out)

Call Stack

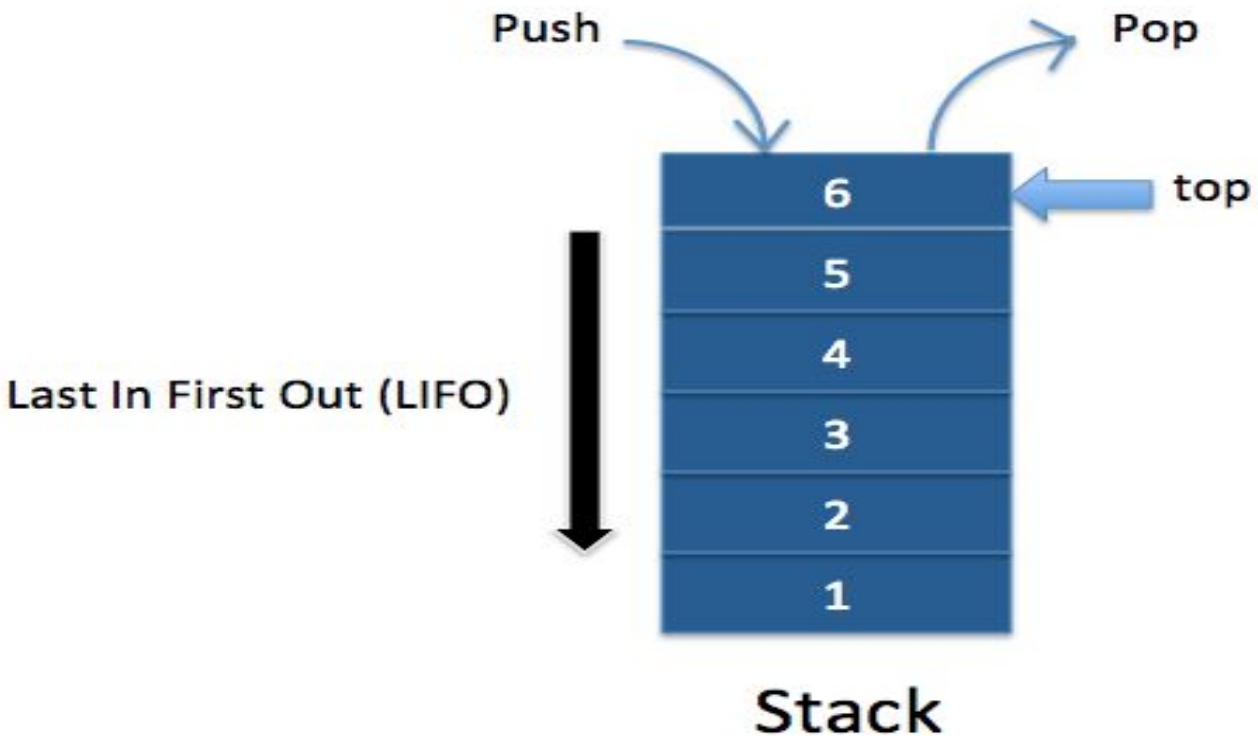
THE
DEVELOPER'S
CONFERENCE

Armazenamento temporário

THE DEVELOPER'S CONFERENCE

Assim que se invoca a função, seus parâmetros e variáveis vão para a pilha de chamadas, ocupando um quadro da pilha, este quadro é um local na memória da pilha, quando esta é retornada, sai da pilha e a função é apagada da memória.

Call Stack


Push
Pop
6
top
5
4
3
2
1
Last In First Out (LIFO)
Stack

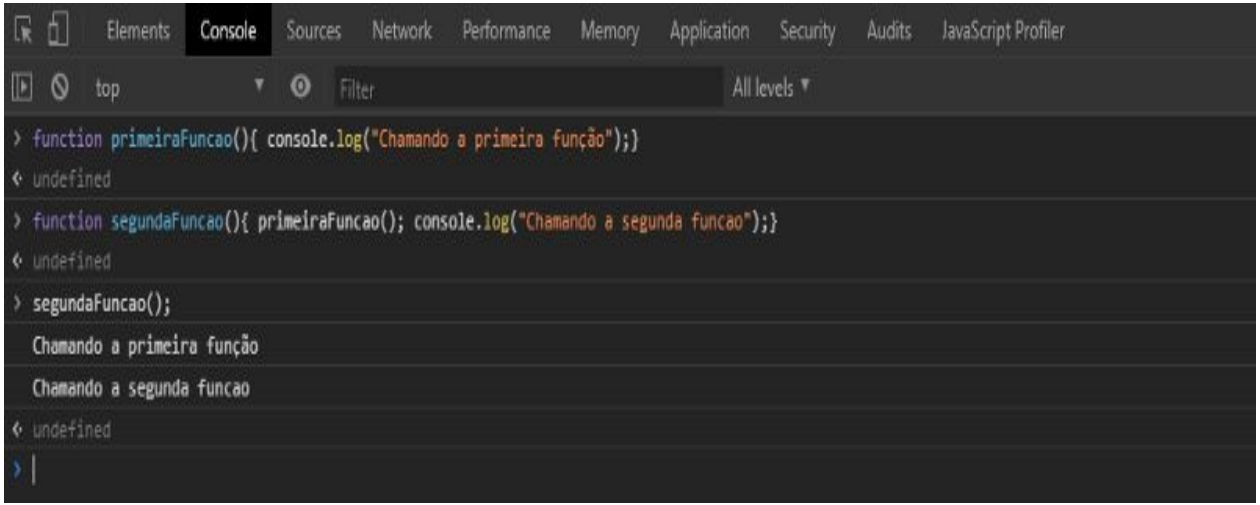
THE
DEVELOPER'S
CONFERENCE

Gerenciando a invocação das funções

THE DEVELOPER'S CONFERENCE

A pilha armazena um registro da posição de cada quadro da pilha , é conhecida a próxima função a ser executada e a mesma será removida após a execução.

Call Stack


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
JavaScript Profiler
top
Filter
All levels
function primeiraFuncao(){ console.log("Chamando a primeira função");}
undefined
function segundaFuncao(){ primeiraFuncao(); console.log("Chamando a segunda funcao");}
undefined
segundaFuncao();
Chamando a primeira função
Chamando a segunda funcao
undefined

THE
DEVELOPER'S
CONFERENCE

Multi threading

THE DEVELOPER'S CONFERENCE

Multi threading

Por origem, o Node.js é single-thread, para usá-lo em processamento paralelo, uma solução é o uso de clusters.

THE DEVELOPER'S CONFERENCE

server.js

```
const cluster = require('cluster');
const http = require('http');
const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  console.log(`Processo Master ${process.pid} está em execução`);

  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }

  cluster.on('Sair', (worker, code, signal) => {
    console.log(`worker ${worker.process.pid} morreu`);
  });
} else {

  http.createServer((req, res) => {
    res.writeHead(200);
    res.end('Olá cluster\n');
  }).listen(3000);

  console.log(`Worker ${process.pid} iniciado`);
}
```

THE DEVELOPER'S CONFERENCE

Multi threading

Padrão em algumas plataformas, que é a metodologia round-robin , na qual o processo mestre atende uma determinada porta, aceita novas conexões e as distribui pelos workers (contém todas as informações e métodos públicos) seguindo a lógica do round-robin para evitar sobrecarregar um processo de trabalho.

THE DEVELOPER'S CONFERENCE

Multi threading

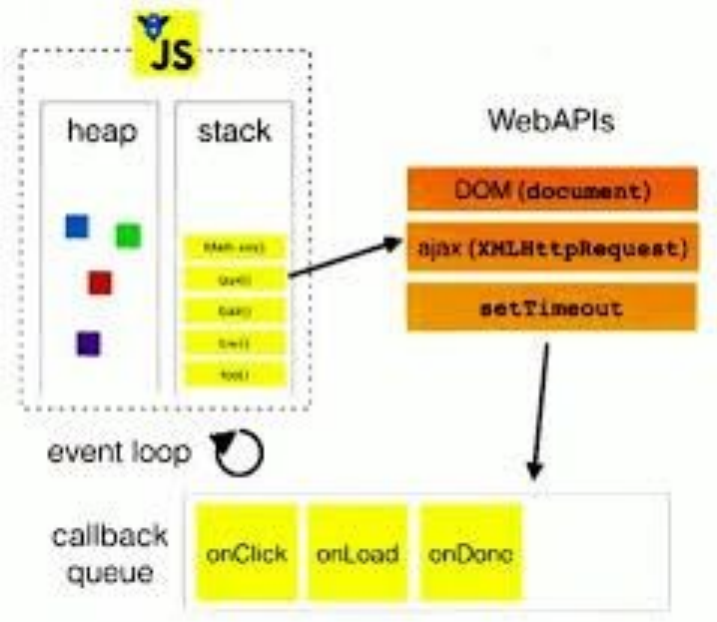
Como os workers são processos separados, podem ser mortos e iniciados dependendo da necessidade do aplicativo, sem afetar outros workers que estiverem em funcionamento. Enquanto houver um worker vivo, o master continuará aceitando conexões , quando nenhum estiver ativo, as conexões existentes serão descartadas e novas serão recusadas pelo Master, pois o mesmo não suportará toda carga de trabalho.

THE
DEVELOPER'S
CONFERENCE

```
$ node server.js
Processo Master 57384 está em execução
Worker 47492 iniciado
Worker 61208 iniciado
Worker 53108 iniciado
Worker 55132 iniciado
Worker 56468 iniciado
Worker 61392 iniciado
Worker 49096 iniciado
Worker 58896 iniciado
```

THE
DEVELOPER'S
CONFERENCE

Task Queue


JS
heap
stack
WebAPIs
DOM (document)
ajax (XMLHttpRequest)
setTimeout
event loop
callback queue
onClick
onLoad
onDone

THE DEVELOPER'S CONFERENCE

Task Queue

Em node.js tem um próprio gerenciamento dos jobs (por padrão ele trabalha com LIFO conforme dito na seção CallStack) mas também temos alguns módulos que implementam a estrutura FIFO (First-In-First-Out) que não é tão utilizado.

THE
DEVELOPER'S
CONFERENCE

Programação Assíncrona

Orientada a eventos

THE DEVELOPER'S CONFERENCE

Orientado por eventos específicos, requisições, cliques, teclas.
Programação Assíncrona: eventos são executados independentemente do fluxo do programa principal.

Programação Assíncrona

THE DEVELOPER'S CONFERENCE

Event Notification

THE DEVELOPER'S CONFERENCE

Um sistema envia mensagens dos eventos para sinalizar outros sistemas da mudança eu seu domínio.
O sistema de origem não espera a resposta do destino.
Tem como característica baixo nível de acoplamento e fácil configuração.

Event Notification

THE DEVELOPER'S CONFERENCE

Event-Carried State Transfer

THE DEVELOPER'S CONFERENCE

É o caso em que se quer atualizar uma parte do sistema sem precisar passar pelo sistema de origem, é disparado o evento com os detalhes para essa atualização com os eventos que contém os detalhes dos dados alterados.
O destinatário atualiza a sua cópia desses dados sem ter que se comunicar com o sistema principal.

Event-Carried State Transfer

THE
DEVELOPER'S
CONFERENCE

Event-Sourcing

THE DEVELOPER'S CONFERENCE

Event-Sourcing

Sempre que realizamos uma alteração no estado de um sistema, registramos essa mudança como um evento e assim podemos reconstruir este estado processando estes eventos a qualquer momento.

THE DEVELOPER'S CONFERENCE

CQRS (Command Query Responsibility Segregation)

THE DEVELOPER'S CONFERENCE

Temos estruturas de dados separadas de leitura e escrita , facilita gerenciar diferentes padrões de acesso, como por exemplo, muitas leituras e poucas gravações.

CQRS (Command Query Responsibility Segregation)

Referências:

THE DEVELOPER'S CONFERENCE

https://www.casadocodigo.com.br/products/livro-nodejs

https://medium.com/reactbrasil/como-o-javascript-funciona-uma-vis%C3%A3o-geral-da-engine-runtime-e-da-call-stack-471dd5e1aa30

https://www.freecodecamp.org/news/understanding-the-javascript-call-stack-861e41ae61d4/

https://blog.sessionstack.com/how-does-javascript-actually-work-part-1-b0bacc073cf

https://github.com/charlesfreeborn/JS-CallStack-CodeSamples/blob/master/codesamples.md

https://imasters.com.br/desenvolvimento/node-js-processando-em-paralelo

https://www.npmjs.com/package/queue-fifo

Referências:

THE DEVELOPER'S CONFERENCE

http://deinfo.uepg.br/~alunoso/2016/ROUNDROBIN/

https://www.infoq.com/br/articles/nodejs-utilizando-modulo-de-cluster/

https://nodejs.org/api/cluster.html#cluster_cluster

https://ilovecoding.org/lessons/whats-special-about-nodejs?playlist=learn-node-js-in-a-week

http://codingwithalex.com/data-structures-everything-need-know-stacks/

https://desenvolvedor.expert/o-que-eh-nodejs-ca9012914c7d

https://martinfowler.com/articles/201701-event-driven.html

Obrigada <3

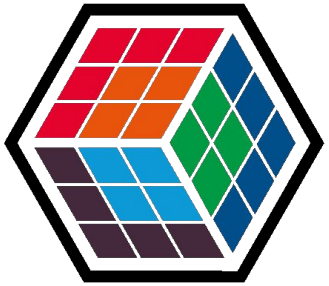
THE DEVELOPER'S CONFERENCE

@kamilah_santos

in/kamila-santos-oliveira/

@kamila_code

@kamilahsantos



THE DEVELOPER'S CONFERENCE