Javascript por debaixo dos panos

Laís Lima

Designer/Front End Developer @ Cosmobots

@laislima.dev @laislima_dev @laislimadev

Por que entender?

É fundamental que você saiba os poderes da linguagem com que trabalha

Performance, quanto mais você conhece, mais você sabe os poderes da linguagem

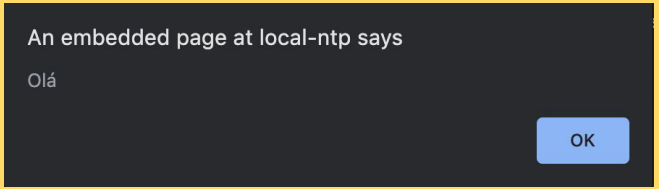
Facilitar debug, quanto antes encontrar o problema mais rápida será a reação

É essencial para o seu currículo! =P


alert("Olá")
?
An embedded page at local-ntp says
Olá
OK

```
alert("Olá")
```

Javascript
engine



O que acontece quando a engine lê seu código no Browser?

Primeiro devemos saber que:

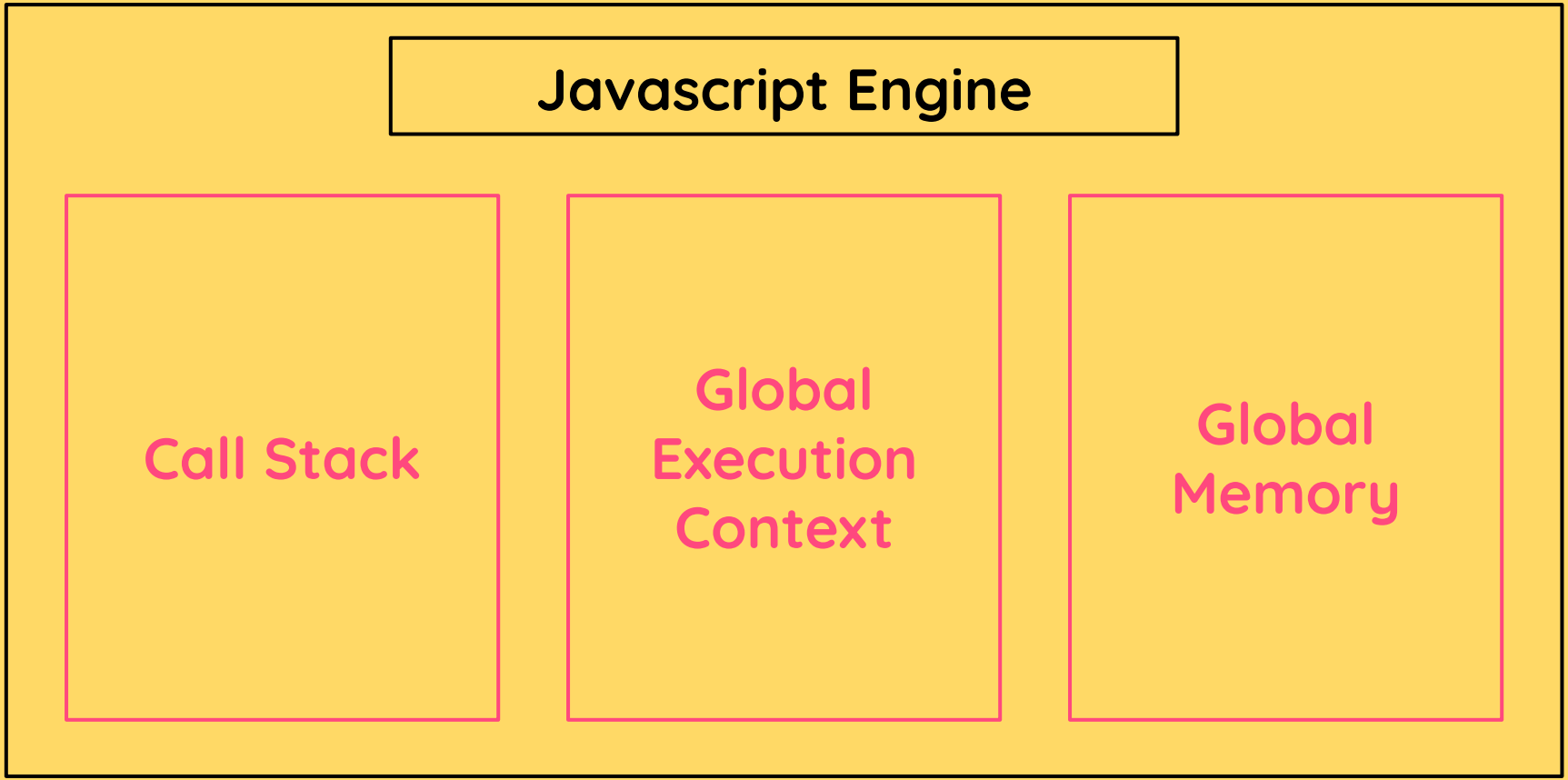
Javascript é single - threaded

Quando o seu código é executado no browser, são criados:

Global Execution Context

Global Memory

Call Stack


Javascript Engine
Call Stack
Global Execution Context
Global Memory

Como a engine lê seu código

```
// Legal, temos uma constante, vamos armazenar na Global Memory
const num = 2;

// olha só uma função, vamos armazenar ela na Global memory também
function pow(num) {
  return num * num;
}

//Pronto terminei!!
```

Javascript Engine

Call Stack

Global Execution Context

Global Memory

```
pow:function
num: 2
```

```
pow();
```

Call stack help!

Javascript Engine

Call Stack

```
pow()
```

Global Execution Context

```
pow:
Executando...
```

Global Memory

```
pow:function
num: 2
```

Mais dois contextos são criados

Estes são:

Local Execution Context

Local Memory

Javascript Engine

Call Stack

```
pow()
```

Global Execution Context

pow:

Local execution context

Local memory

Global Memory

pow:

Local execution context

Local memory

Javascript Engines

As engines são responsáveis por fazer com que a máquina entenda o nosso código Javascript

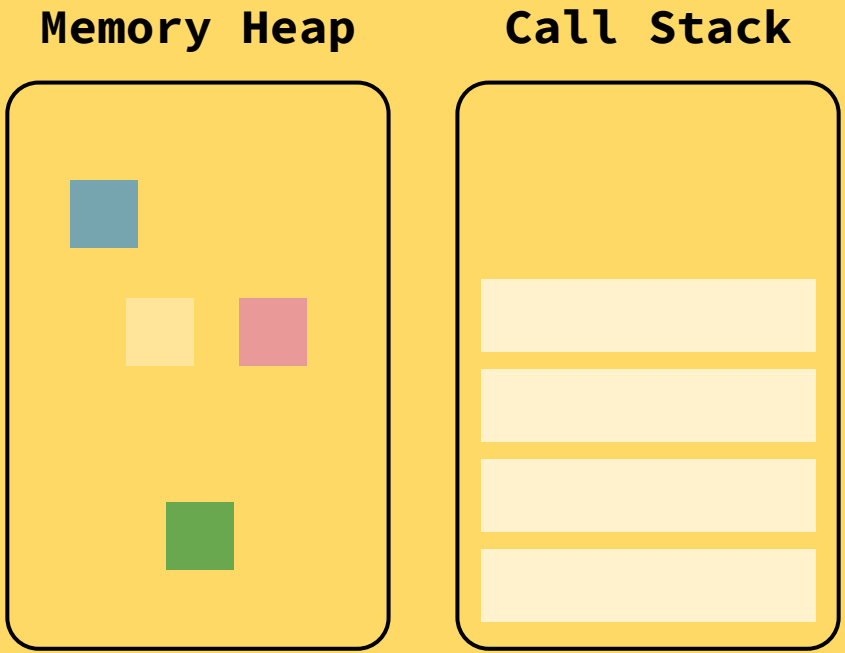
Algumas delas são

V8 - Chrome e Node JS

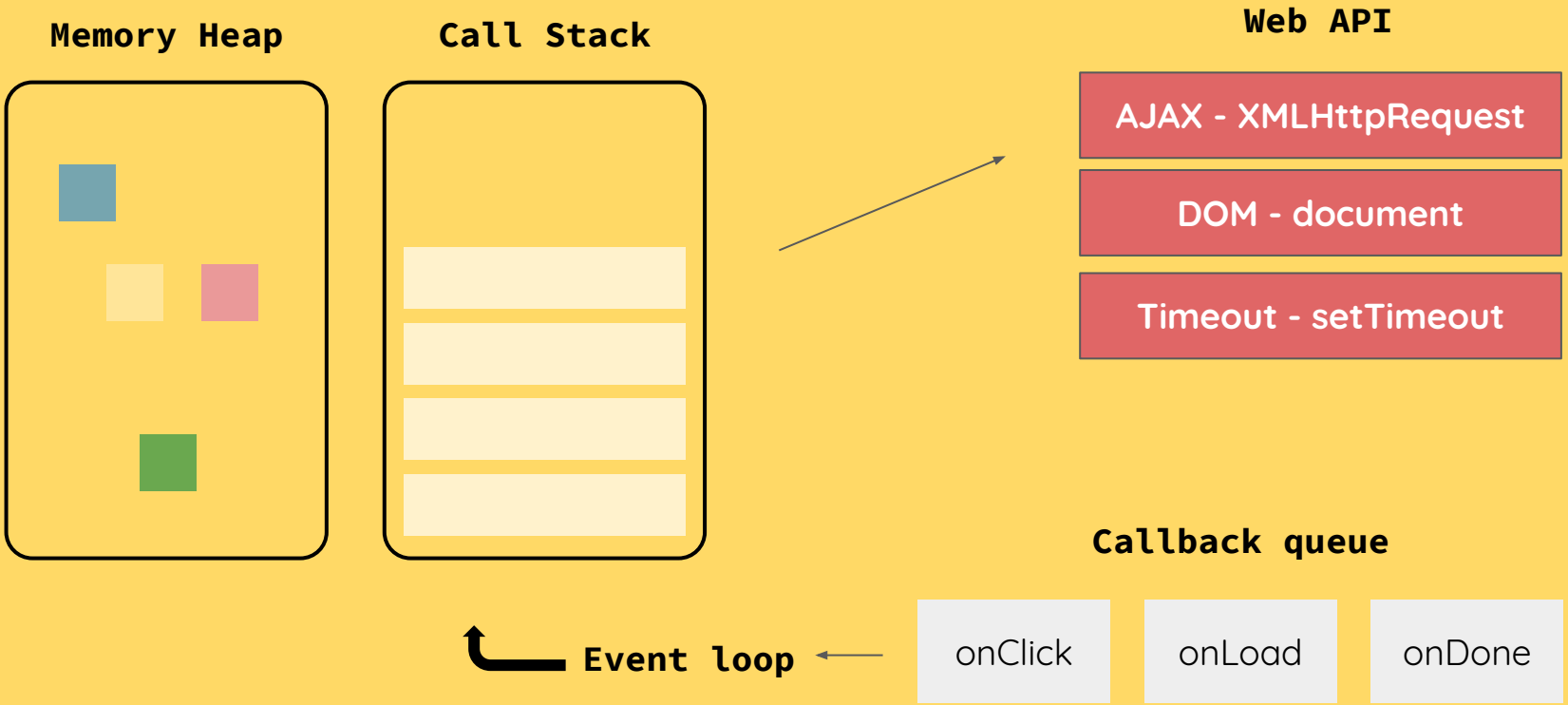
Spider Monkey - Mozilla

Chakra - Microsoft Edge

Como a engine parece



Runtime no browser



Tá,vamos ver como funciona?

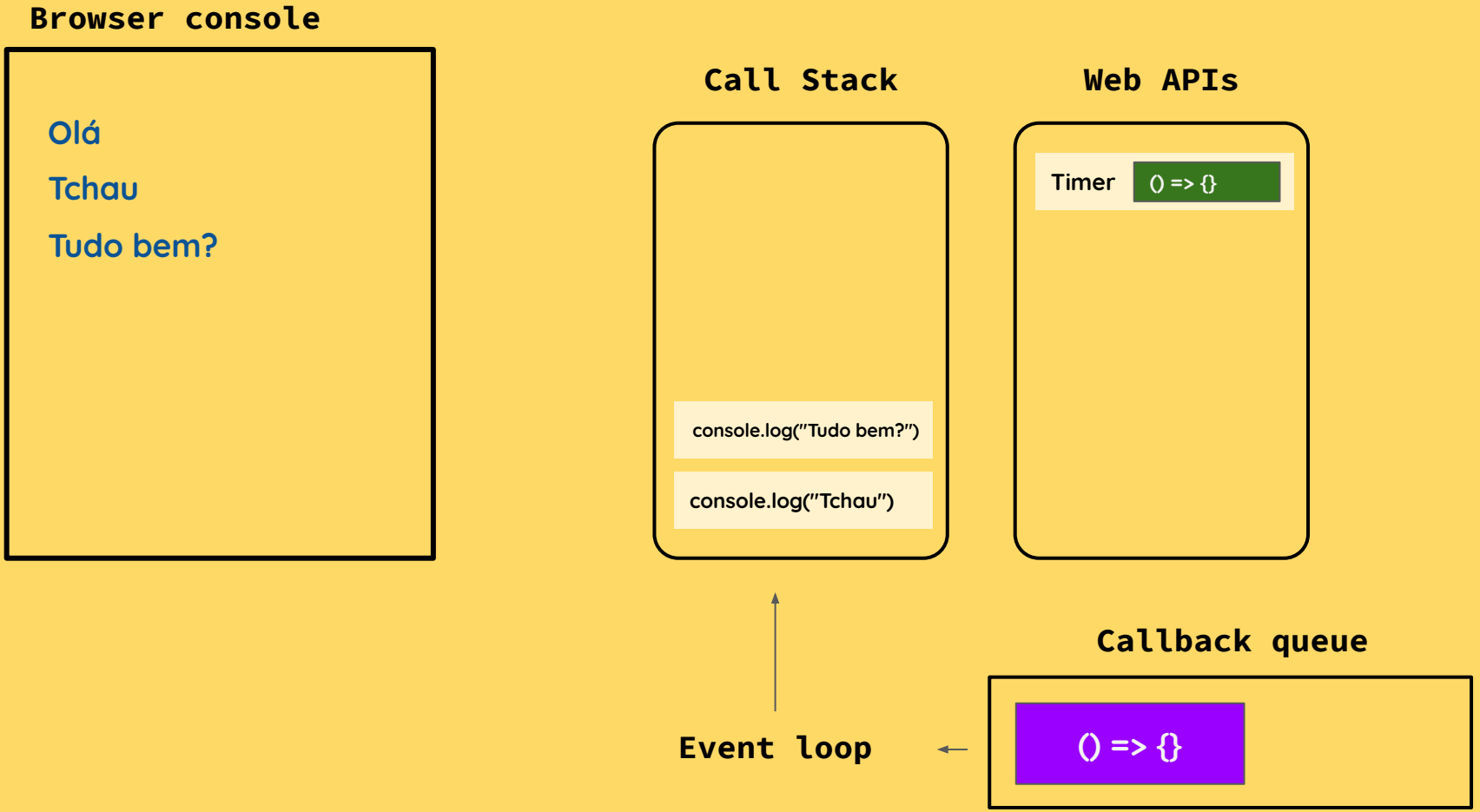
```
console.log('Olá');

setTimeout(() => {

    console.log('Tudo bem?');

}, 5000);

console.log('Tchau');
```


Browser console
Olá
Tchau
Tudo bem?
Call Stack
Web APIs
Timer
() => {}
console.log("Tudo bem?")
console.log("Tchau")
Callback queue
() => {}
Event loop

Vamos nos aprofundar mais um pouco...

Entenda sobre o V8, uma engine javascript

v8.dev

É open source e mantido pelo google

É escrito em C++ (estaticamente tipado enquanto javascript é dinamicamente tipado)

Primeira engine projetada para aumentar a performance da execução do javascript no navegador

https://medium.com/dailyjs/understanding-v8s-bytecode-317d46c94775

Algumas threads do V8

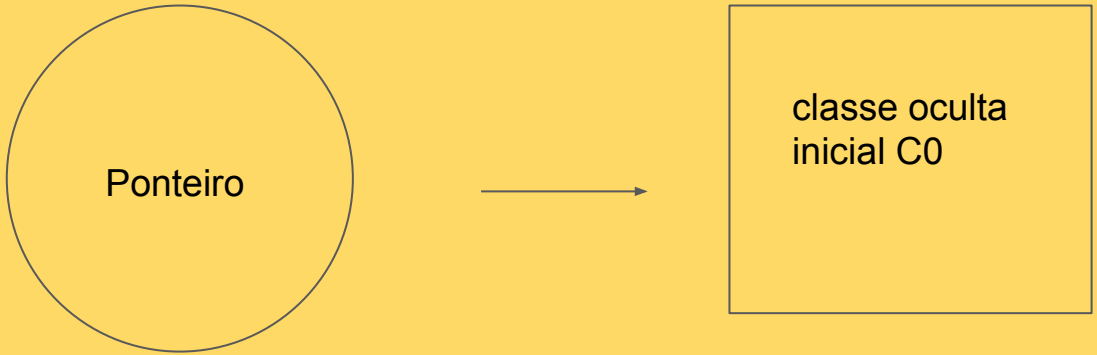
Existe uma thread principal que busca seu código, compila e então executa

Uma thread para otimizar o código que está sendo executado na principal

threads para lidar com varreduras e garbage collector

Classe oculta

São layouts de objetos que são criados em tempo de execução se são apontados pelo ponteiro do objeto na memória



Classe oculta

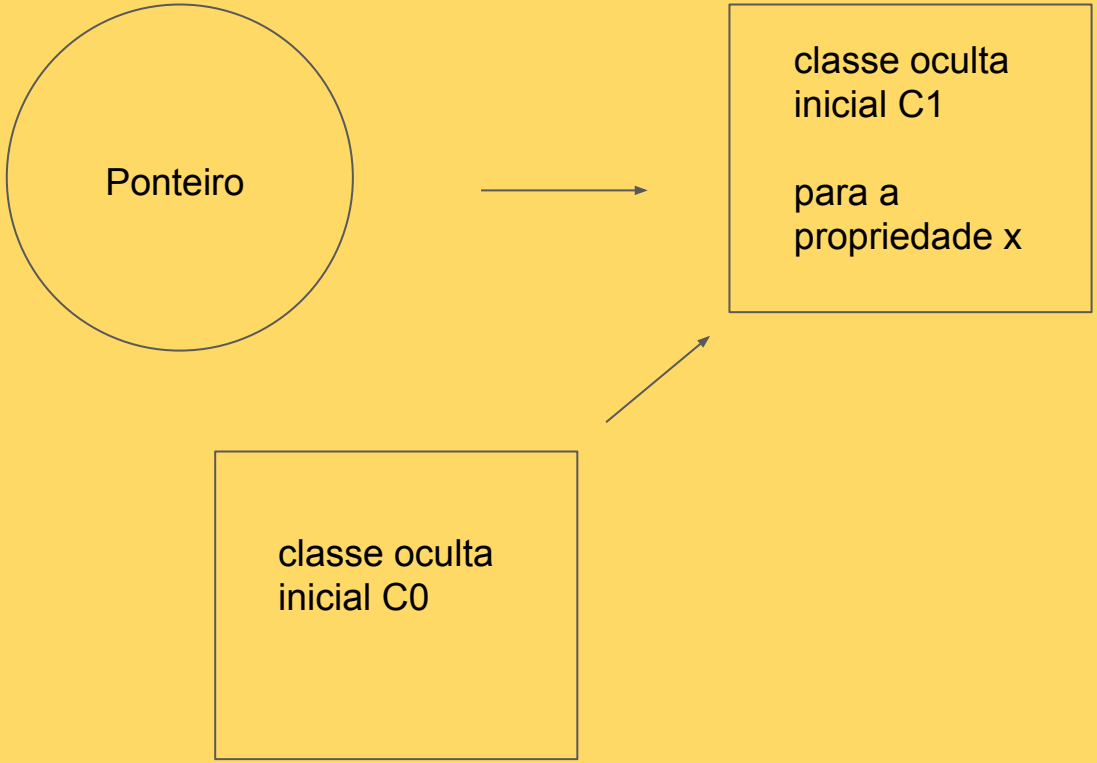
```
function Point(x) {

    this.x = x;

}


var p1 = new Point(1);
```

Classe oculta



Inlining

substituir invocação da função pelo corpo da mesma

Inline Caching

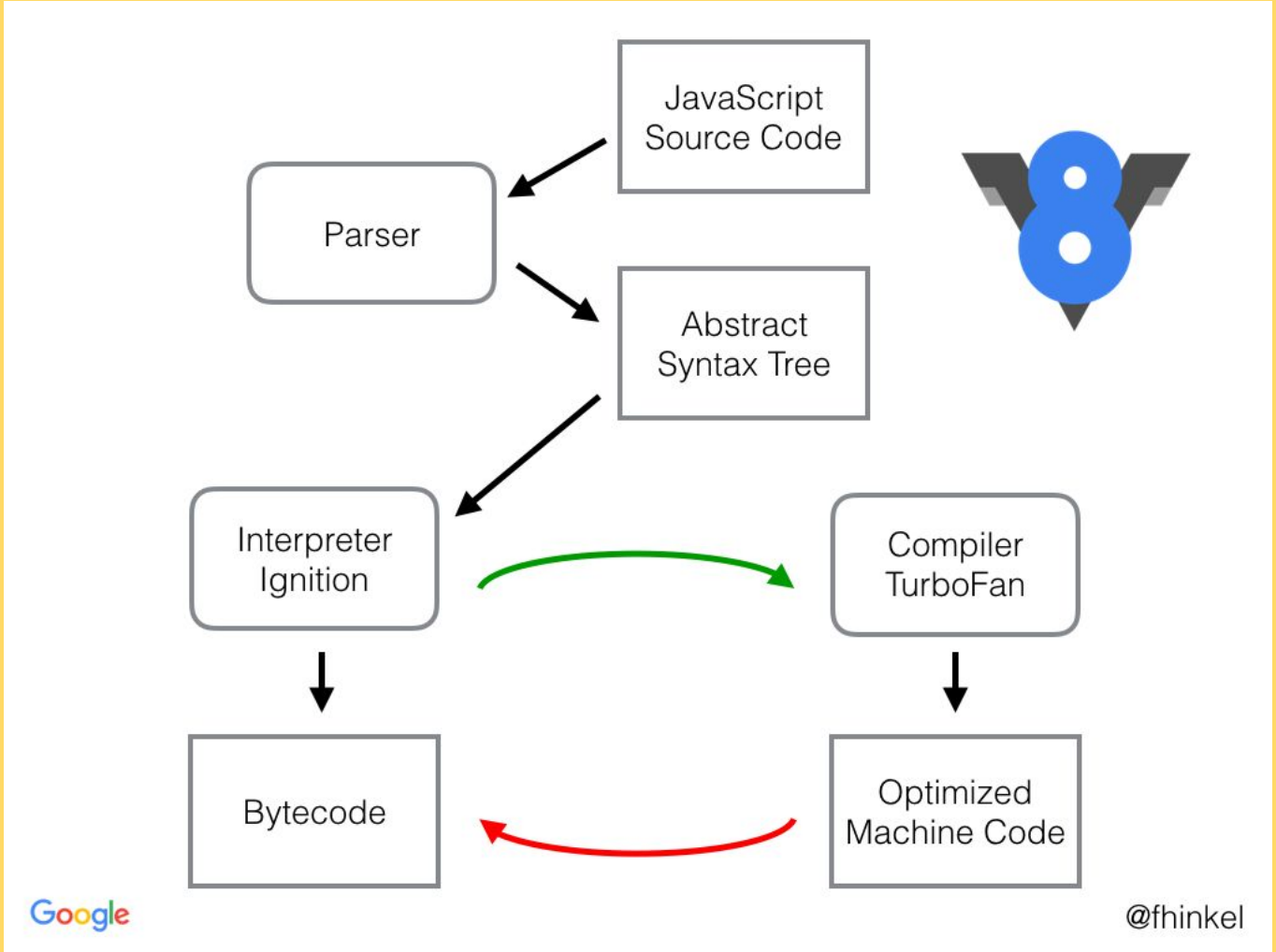
Muitas chamadas para o mesmo método tendem a ocorrer para o mesmo tipo de objeto.

Código de máquina

V8 compila direto para código de máquina (bytecode), consegue otimizar o mesmo e trocar em tempo de execução isto é uma tarefa muito complexa.

Garbage Collector

ao invés de percorrer todo o heap, tentando marcar cada possível objeto, ele apenas marca parte do heap, e então volta para a execução normal. A próxima parada do GC ai continuar de onde o passo anterior do Heap parou. isso permite muitas pausas curtas durante a execução normal.


JavaScript Source Code
Parser
Abstract Syntax Tree
Interpreter Ignition
Compiler TurboFan
Bytecode
Optimized Machine Code
Google
@fhinkel

Dicas de performance

Métodos: Por conta do inline caching chamar o mesmo método várias vezes...

Propriedades de objetos: Sempre instancie propriedades de objetos na mesma ordem

Propriedades dinâmicas: *adicionando propriedades para um objeto depois da instanciação vai forçar uma classe oculta a mudar e desacelerar qualquer método que estava otimizado pela classe oculta anterior. Ao invés disso, atribua todas as propriedades de um objeto no seu construtor.*

Fontes:

https://medium.com/reactbrasil/como-o-javascript-funciona-o-event-loop-e-o-surgimento-da-programa%C3%A7%C3%A3o-ass%C3%ADncrona-5-maneiras-de-18d0b8d6849a

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

https://medium.com/reactbrasil/como-o-javascript-funciona-dentro-da-engine-v8-5-dicas-sobre-como-escrever-c%C3%B3digo-otimizado-e05af6088fd5

https://www.youtube.com/watch?v=p-iiEDtpy6I

https://www.youtube.com/watch?v=8aGhZQkoFbQ&t=370s

https://v8.dev/

Agradecimentos

Pessoas que contribuíram para este conteúdo:

Kamila Santos, https://www.linkedin.com/in/kamila-santos-oliveira

Daniel Martins, https://www.linkedin.com/in/daniel-martins-251b12127

Julio Silva, https://www.linkedin.com/in/silva-julio

Lucas Santos, https://www.linkedin.com/in/lhs-santos/

Obrigada ^^

@laislima.dev

@laislima_dev

@laislimadev