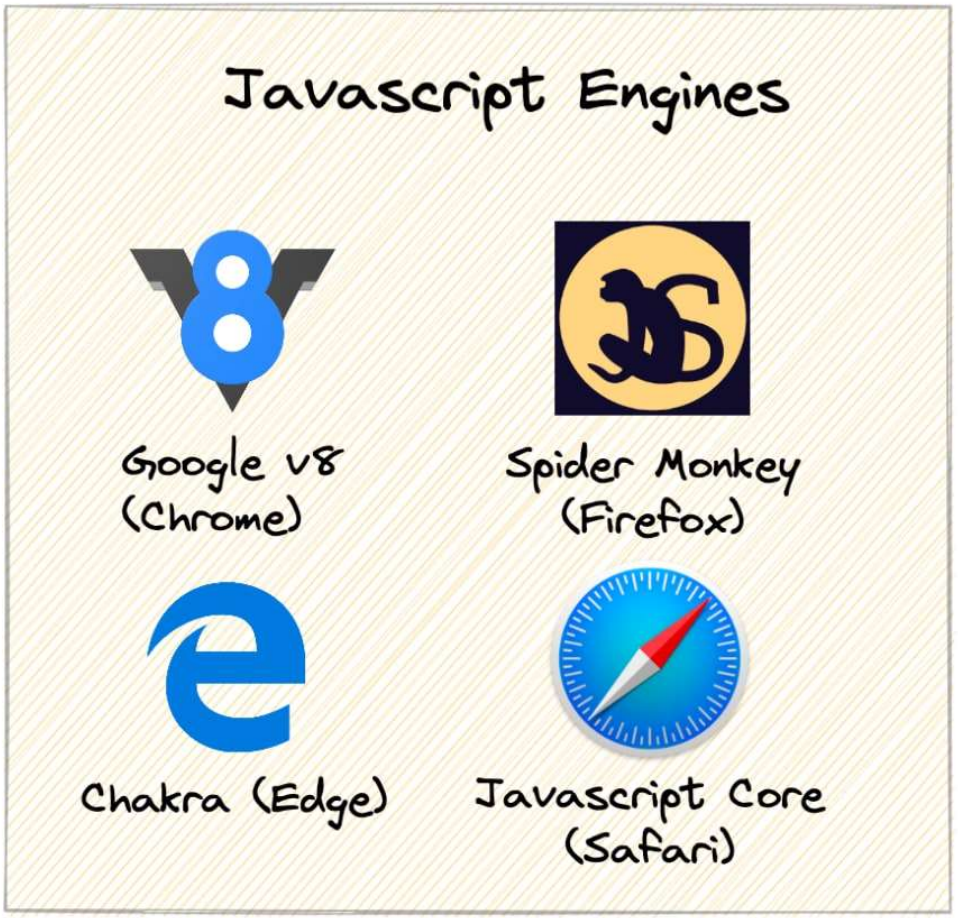
Motores JavaScript

Como o JavaScript inicialmente foi pensado em uma linguagem executada em navegadores, cada plataforma criou seu motor para interpretar e executar código JavaScript: o Mozilla tem o SpiderMonkey, o Edge tem o Chakra, o Safari tem o JavascriptCore e o Chrome tem o V8.



Ambiente de tempo de execução

Para que uma linguagem de programação seja executada fora do navegador é necessário que haja um conjunto de componentes e recursos:

Os objetos globais que o programa acessará (console, por exemplo), ferramentas como APIs, bibliotecas e a infraestrutura para indicar como o programa interage com o sistema de arquivos do computador e com a rede.

Esse conjunto é chamado de ambiente de tempo de execução ou Runtime Environment.

Cada linguagem de programação tem seu próprio ambiente de tempo de execução, adaptado às suas necessidades e características.

Node.js

É um ambiente de tempo de execução JavaScript, que possui código aberto e foi construído com base no motor V8 do Chrome.

O Node.js foi criado por Ryan Dahl e lançado em 2009, surgindo da necessidade de criar servidores web mais eficientes e escaláveis, explorando características da linguagem JavaScript, como o assincronismo.

Outras plataformas em ascensão desse formato são o Deno que também usa o motor V8 e Bun que usa o JavascriptCore do Safari.

Navegador vs Servidor

No navegador, o JavaScript é usado para criar interatividade e dinamismo nas páginas da web, manipulando a interface do usuário, interagindo com os elementos HTML e fazendo solicitações HTTP.

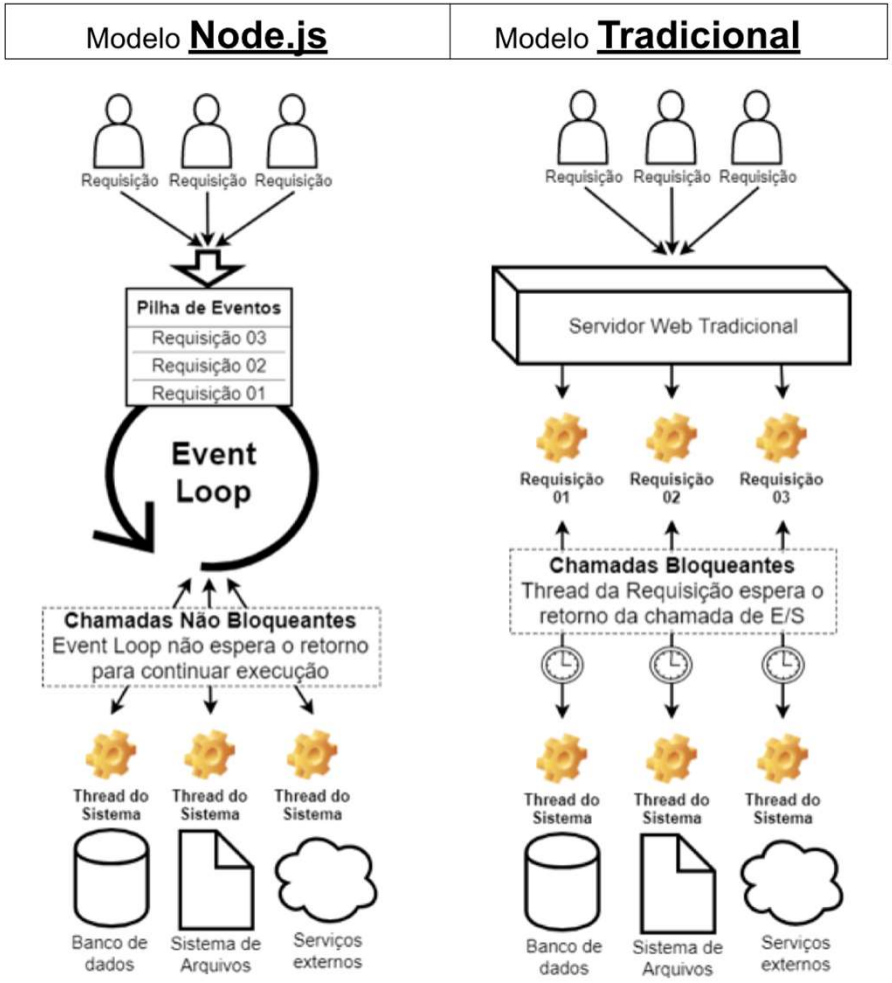
No servidor, o Node.js é capaz de realizar tarefas como manipular solicitações HTTP, acessar bancos de dados, realizar entrada e saída de arquivos e criar servidores web.

A grande vantagem é que o JavaScript no Node.js e no navegador compartilham a mesma linguagem, o que permite compartilhar código e lógica entre o lado do cliente e do servidor.

NPM

O Node Package Manager (npm) é um poderoso gerenciador de pacotes utilizado para administrar as bibliotecas e frameworks utilizados em uma aplicação. Ele permite instalar, desinstalar e atualizar dependências em uma aplicação por meio de uma simples instrução na linha de comando.

Esses gerenciador é utilizado também no desenvolvimento frontend em conjunto com o React.js



Eventos que podem bloquear o sistema, como o acesso ao banco de dados ou uso de setTimeout possuem diferentes tratamentos:

Modelo tradicional: o sistema esperaria o retorno para poder realizar outras tarefas.

Modelo Node.js: o sistema pode realizar outras tarefas enquanto aguarda o banco de dados.

Observando a diferença de ações assíncronas

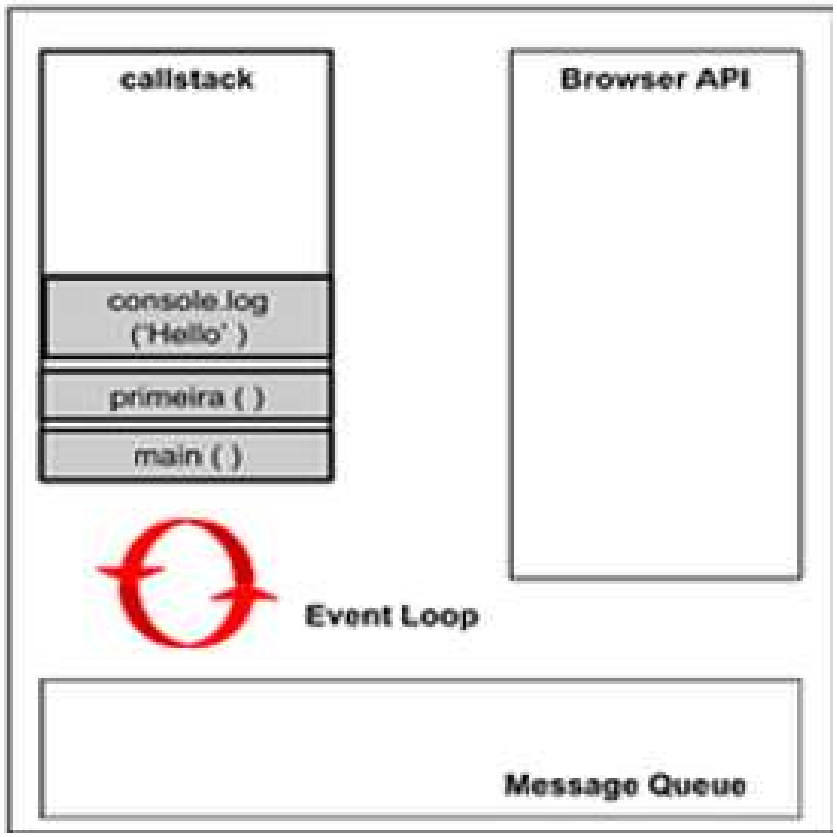
Código síncrono

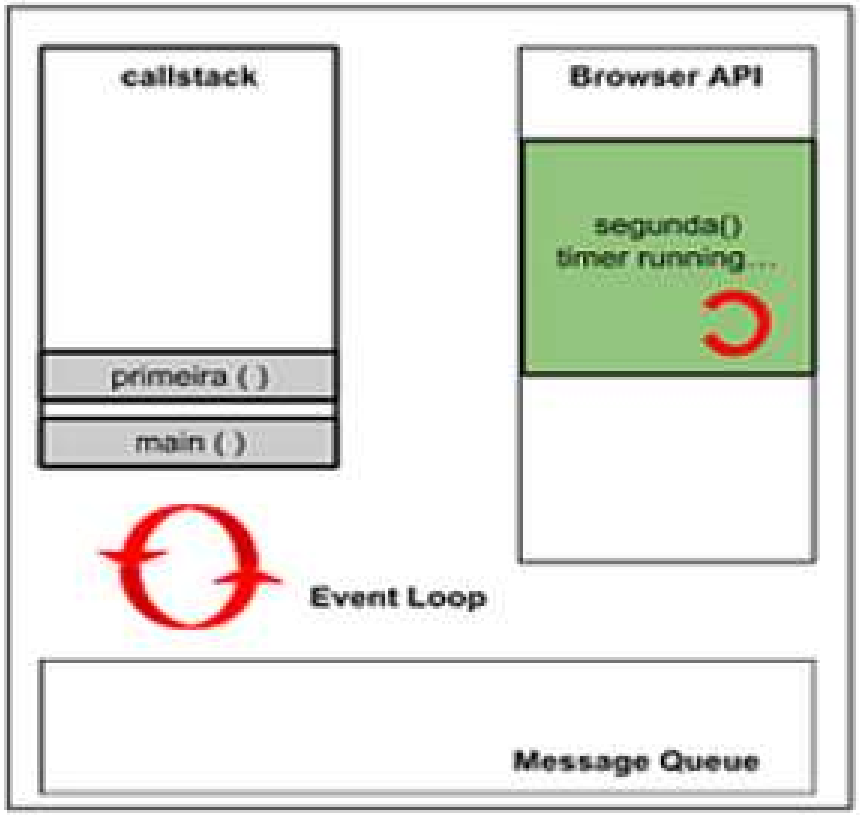
```
function segunda() {
  console.log('World')
}

function primeira() {
  console.log('Hello')
  segunda()
  console.log('Fim')
}
primeira()
```

Código assíncrono

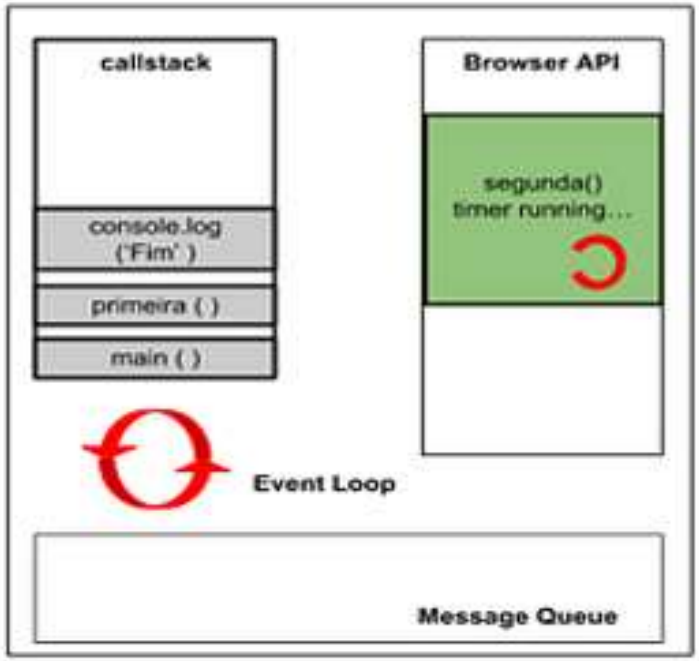
```
function primeira() {
  console.log('Hello')
  setTimeout(() => console.log('World'), 0)
  console.log('Fim')
}
```

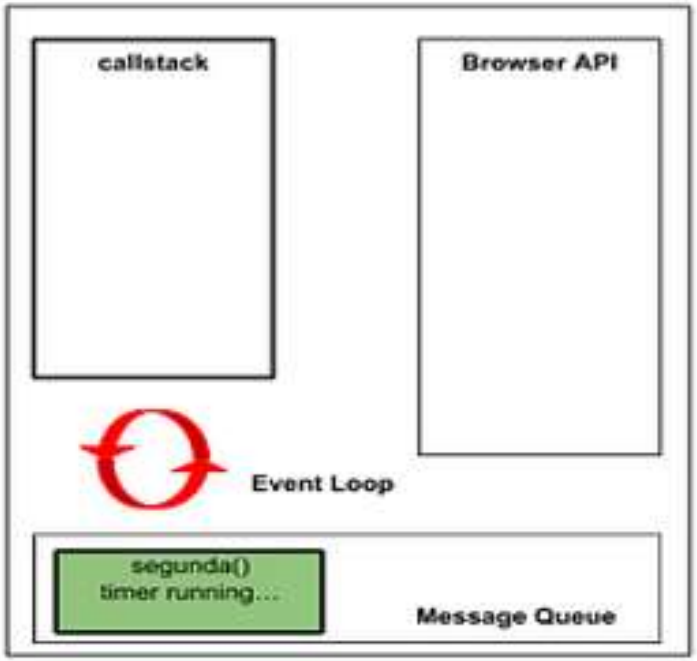
Tudo que é assíncrono, por padrão, terá a execução concluída posteriormente.
OBS: É possível manipular essas execuções com o uso de Promises e async await, para definir outra ordem.

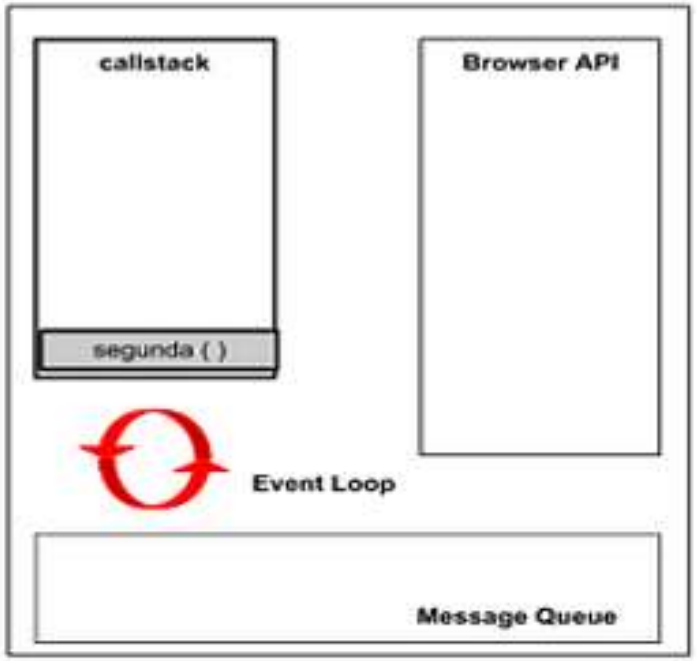





```
function primeira() {
  console.log('Hello')
  setTimeout(() => console.log('Word'), 0)
  console.log('Fim')
}
```








```
function primeira() {
  console.log('Hello')
  setTimeout(() => console.log('Word'), 0)
  console.log('Fim')
}
```

Instalação do Node + NPM

Em nodejs.org é possível fazer download do instalador do Node.js. Uma vez instalado, abra o terminal do seu sistema operacional de cheque a versão do Node e NPM:

Criar um projeto

Para criar um projeto novo em Node.js utilize o comando npm init

```
{
  "name": "projeto-novo",
  "version": "1.0.0",
  "description": "Projeto para o curso de React Entra21",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Ruan Lopes",
  "license": "ISC"
}
```

Executar um arquivo node

Para realizar a execução de um arquivo node, podemos rodar o seguinte comando:

```
node nomedoarquivo.js
```

Recentemente foi adicionada uma opção que reinicia a execução a cada alteração no arquivo, por meio do comando:

```
node --watch nomedoarquivo.js
```