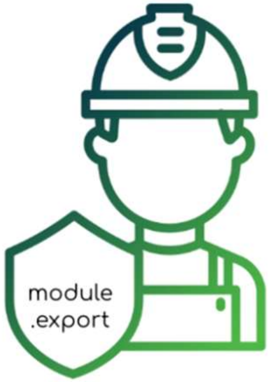
Export

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f486967687207

- Default Export

Named Export

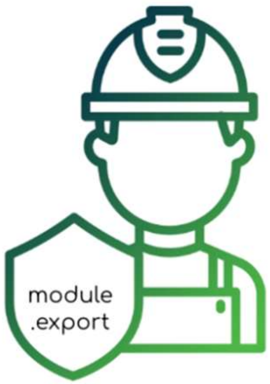


Export

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Default Export

```
module.exports = {
  getName: () => {
    return 'Jim';
  },

  getLocation: () => {
    return 'Munich';
  },

  dob: '12.01.1982',
};
```



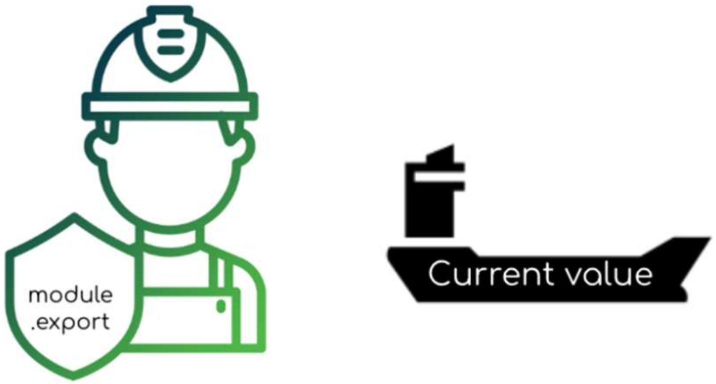
Named Export

```
exports.getName = () => {
  return 'Jim';
};

exports.getLocation = () => {
  return 'Munich';
};

exports.dob = '12.01.1982';
```

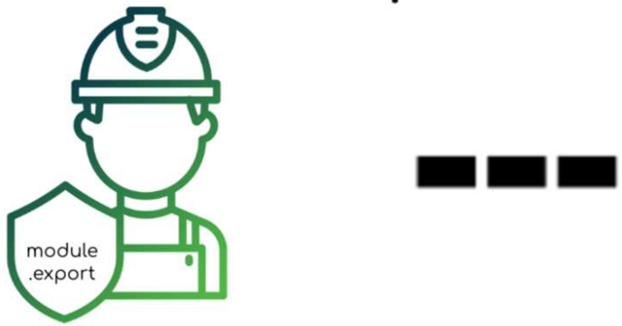
Require

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Default Export



Named Export



appUser.js;

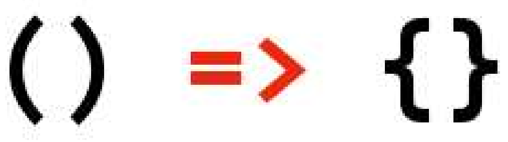
```
const { getName, dob } = require('./user');
console.log(
  `${getName()} was born on ${dob}.`
);
```

JavaScript Funções

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Função Seta

Também conhecida como função lambda



JavaScript Funções

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070

Função Seta

A criação de uma *arrow function* consistem em 3 “passos”:

1. Os parênteses, que é por onde a função recebe os argumentos (assim como na function tradicional);
2. A “seta” => - responsável pelo nome “arrow” function;
3. E as chaves: o bloco de código que representa o corpo da função

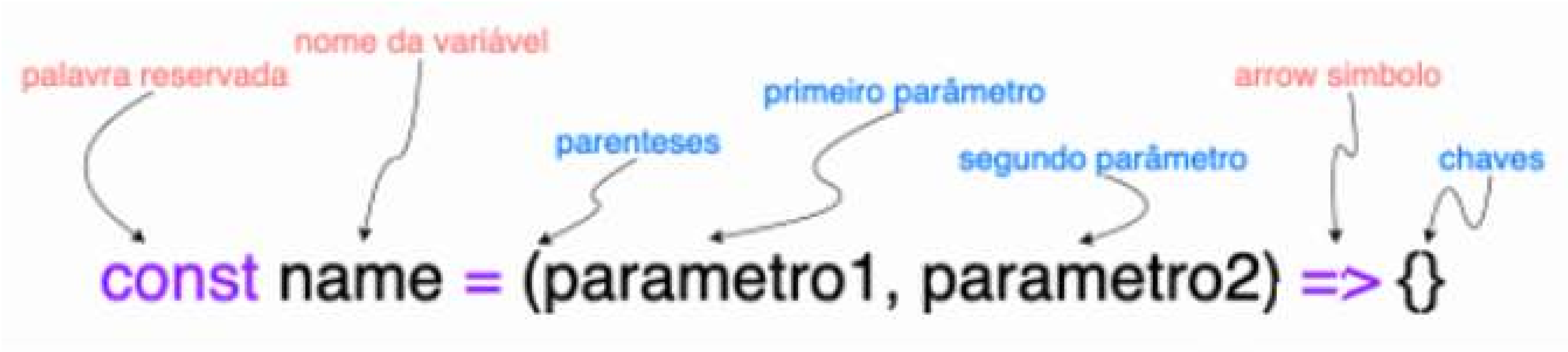
```
const sum = (a, b) => {
return a + b
}
```

Elas são sempre **anônimas**.

JavaScript Funções

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

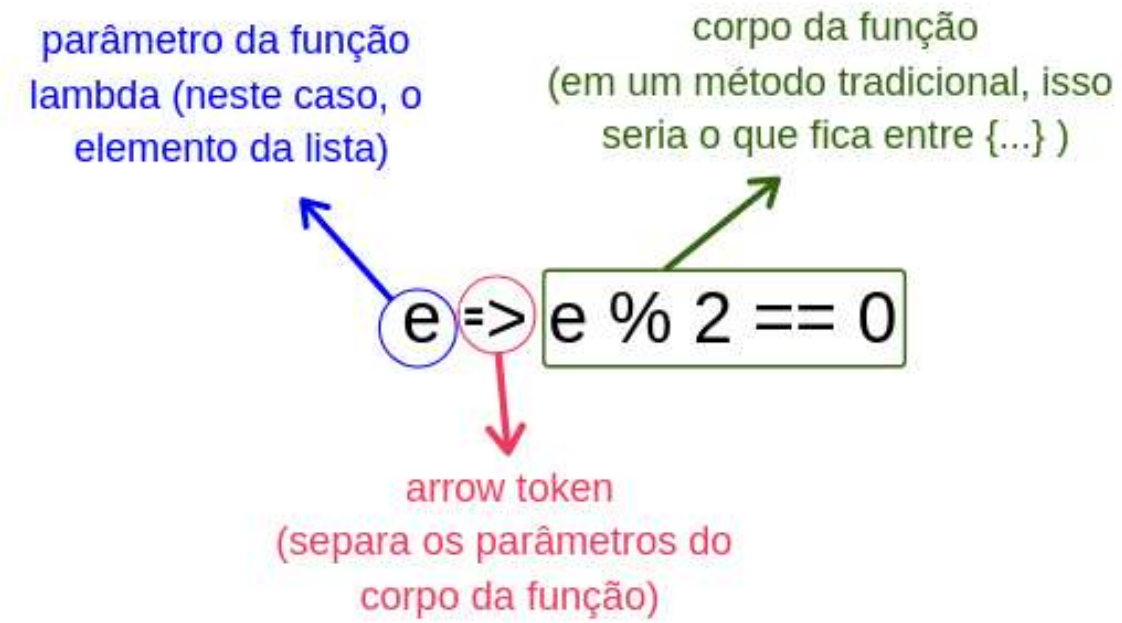
Função Seta



JavaScript Funções

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

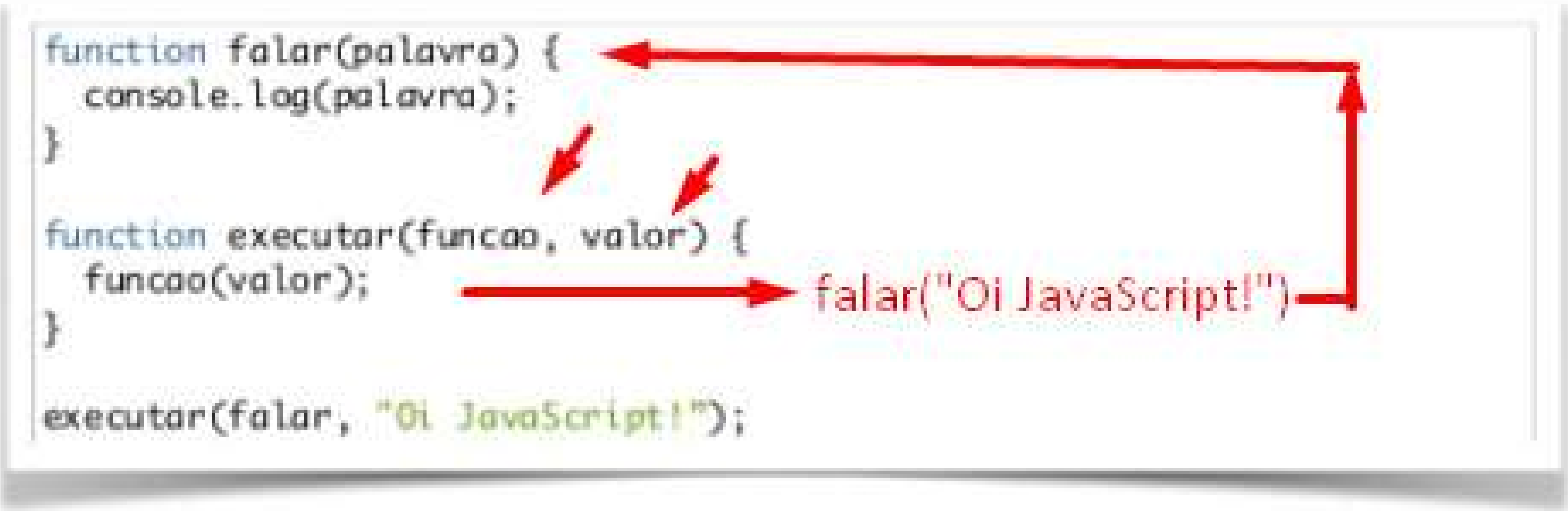
Função Seta



const soma = (a, b) => { a + b }

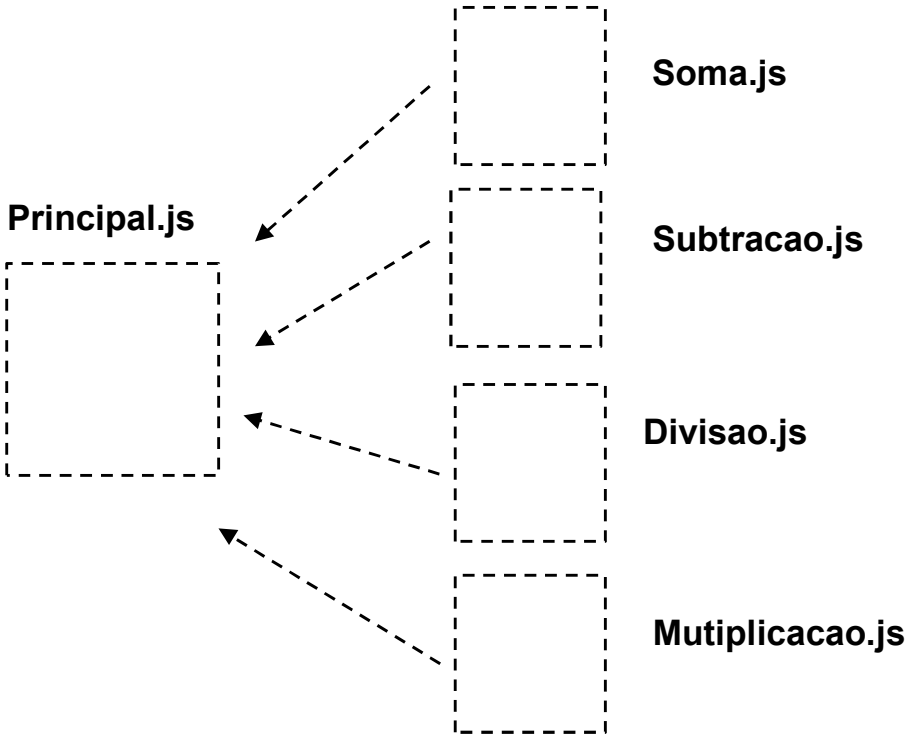
Modulo Node.JS

- Uma Função pode receber outra função como parâmetro;



Atividade

- Crie uma calculadora.
- Ela deve ter 5 arquivos.
- Sendo 4 com as operações básicas e 1 com a função principal.

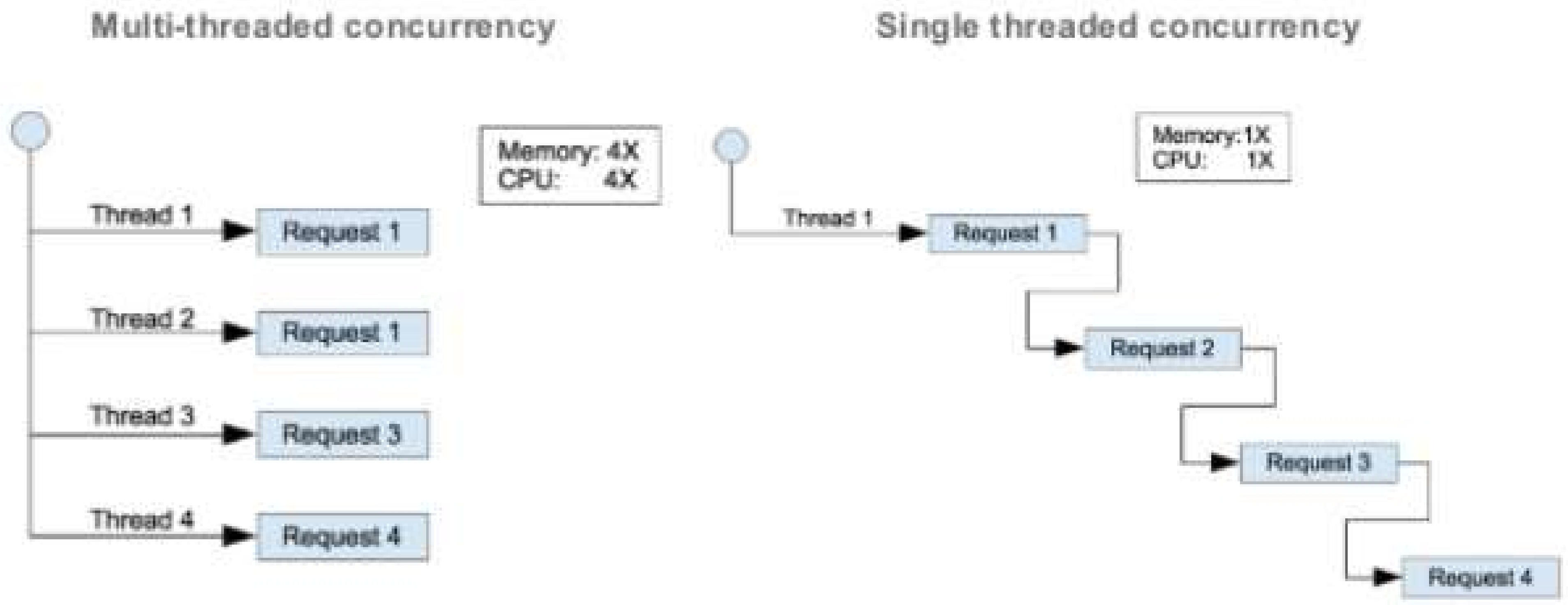


Sistema Node.js

- Node.js é usado principalmente para servidores controlados por eventos sem bloqueio, devido à sua natureza de thread único.
- É usado para sites tradicionais e serviços de API de back-end, mas foi projetado com arquiteturas em tempo real.

Sistema Node.js

– **Multi-thread vs Single-thread**

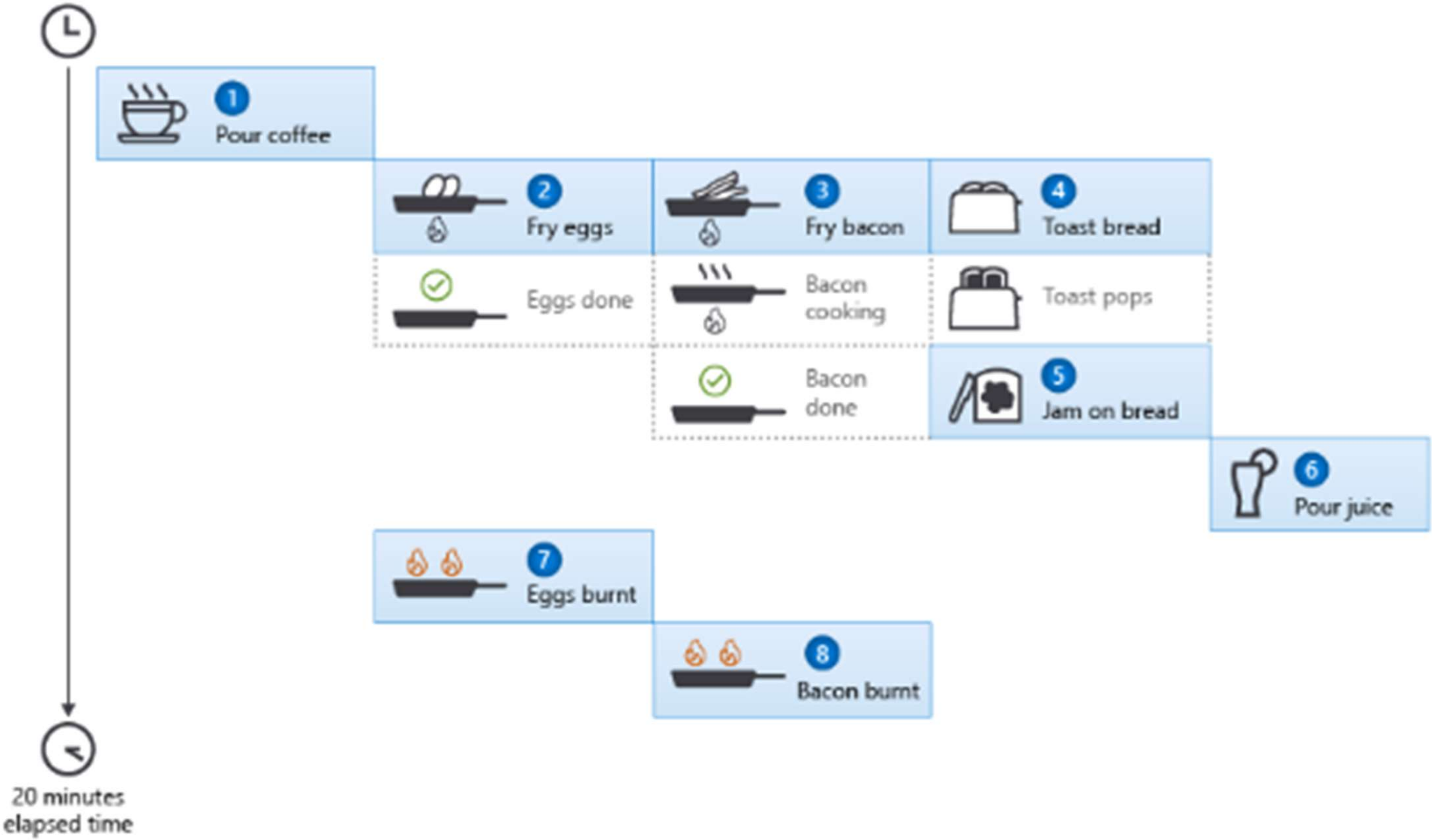


Sistema Node.js

– **Multi-thread vs Single-thread**

Sistema Node.js

– **Multi-thread vs Single-thread**



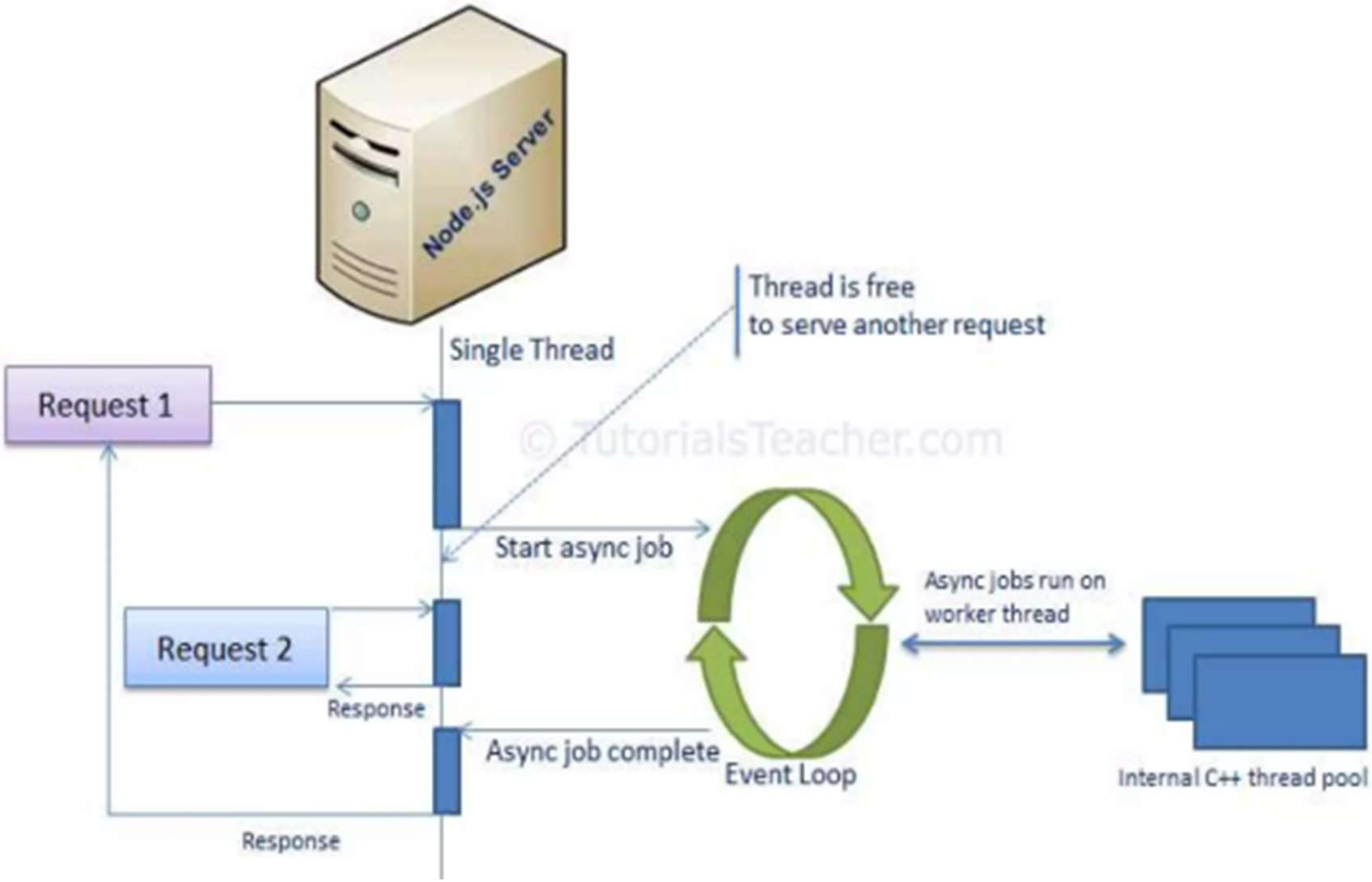
Sistema Node.js

– **Multi-thread vs Single-thread**

Sistema Node.js

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

– **Multi-thread vs Single-thread**



Sistema Node.js

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

– O Google V8

– Single thread

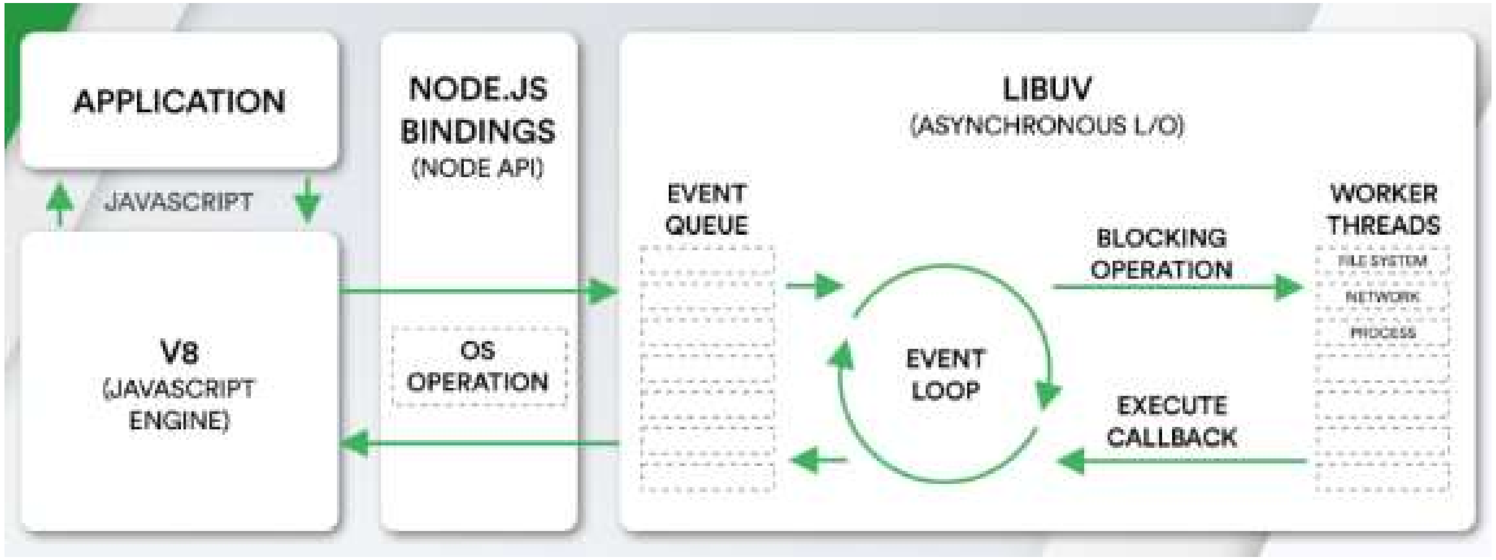
– Event Loop

– Call Stack

– Heap

– Task Queue

– Macro e Micro Tasks

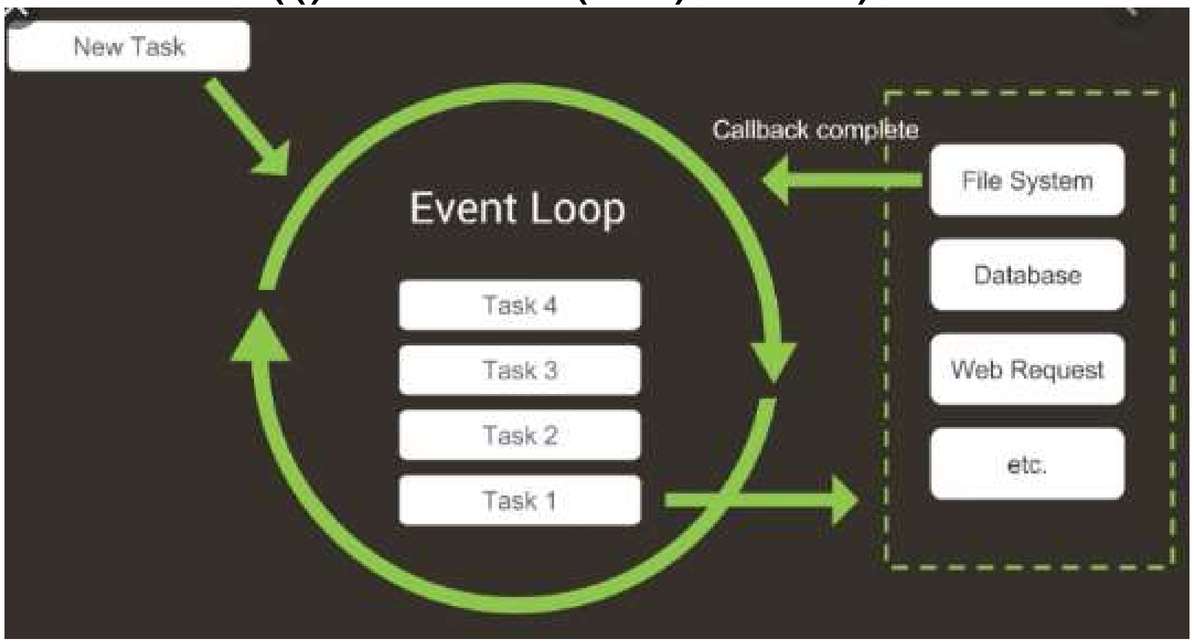


Event Loop

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- O que é Event Loop?
  - Um *core* (central) que escuta todos os eventos e chama seus respectivos *callbacks* quando eles são lançados;
- O que é Call back?
  - É uma função que te permite operar em cima do retorno de outras funções.

Ex.: setTimeout(() => alert("1"),5000);



Event Loop

- Sempre que você chama uma função síncrona (i.e. “normal”) ela vai para uma “call stack” ou pilha de chamadas de funções com o seu endereço em memória, parâmetros e variáveis locais;
- Vamos “executar” esse código e ver o que acontece:

```
console.log('Hi');
setTimeout(function cb1() {
    console.log('cb1');
}, 5000);
console.log('Bye');
```

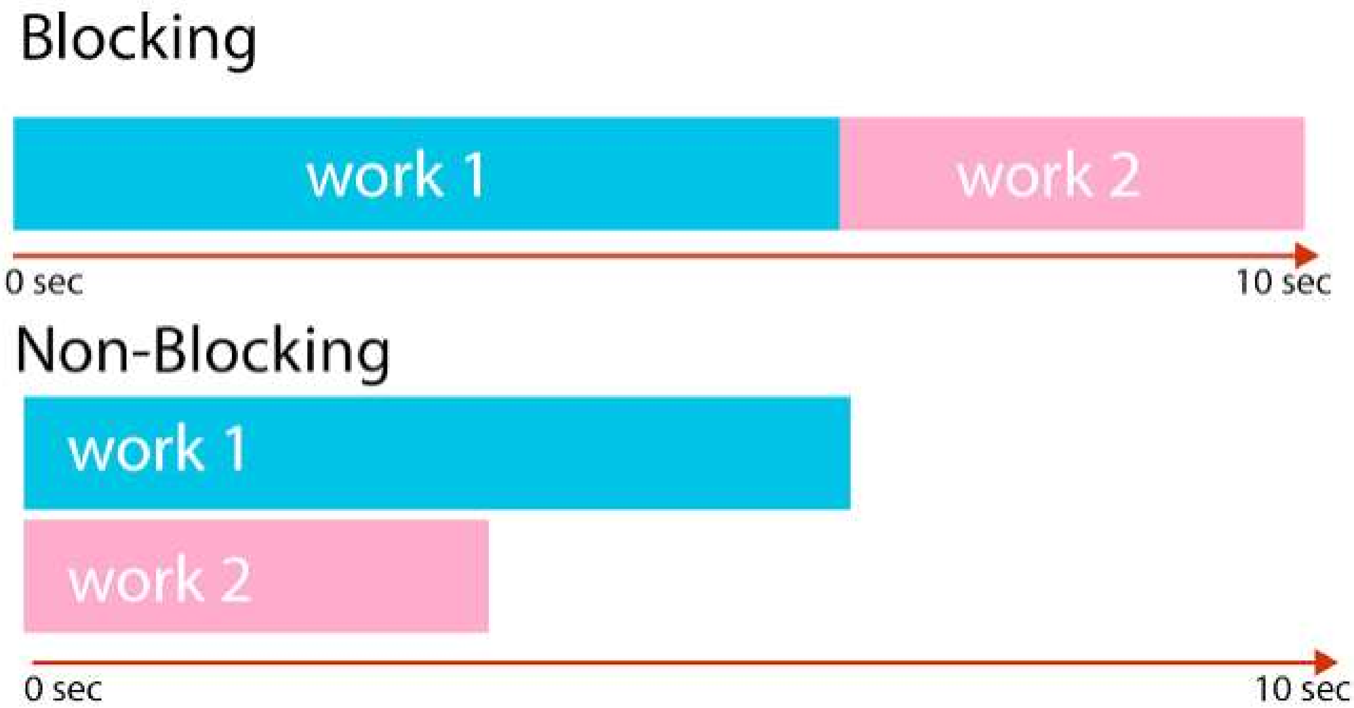
☺

Blocking vs Non-Blocking - operations

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Blocking refere-se a operações que bloqueiam a execução adicional até que a operação seja concluída;

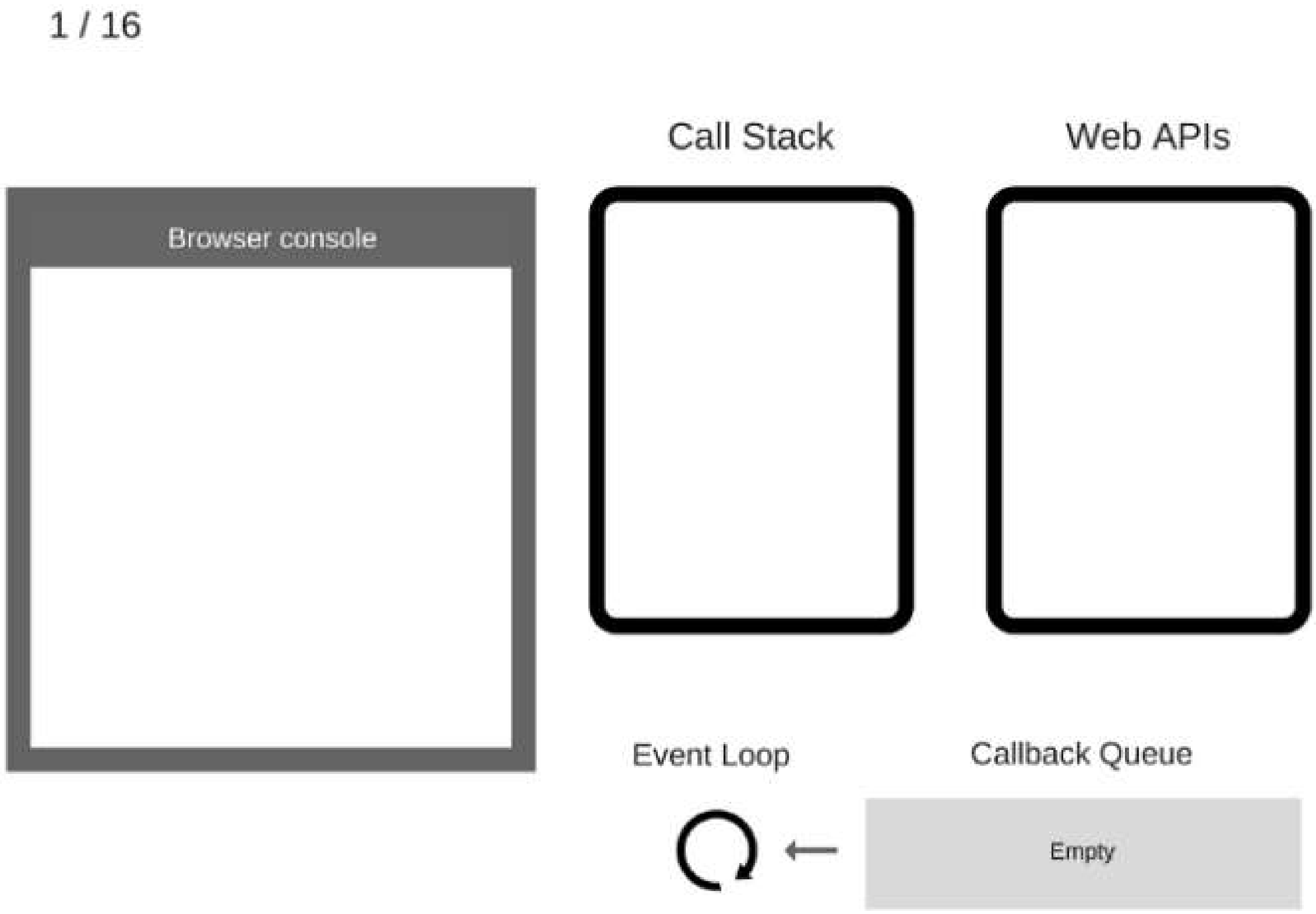
Non-Blocking refere-se ao código que não bloqueia a execução;



Event Loop

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

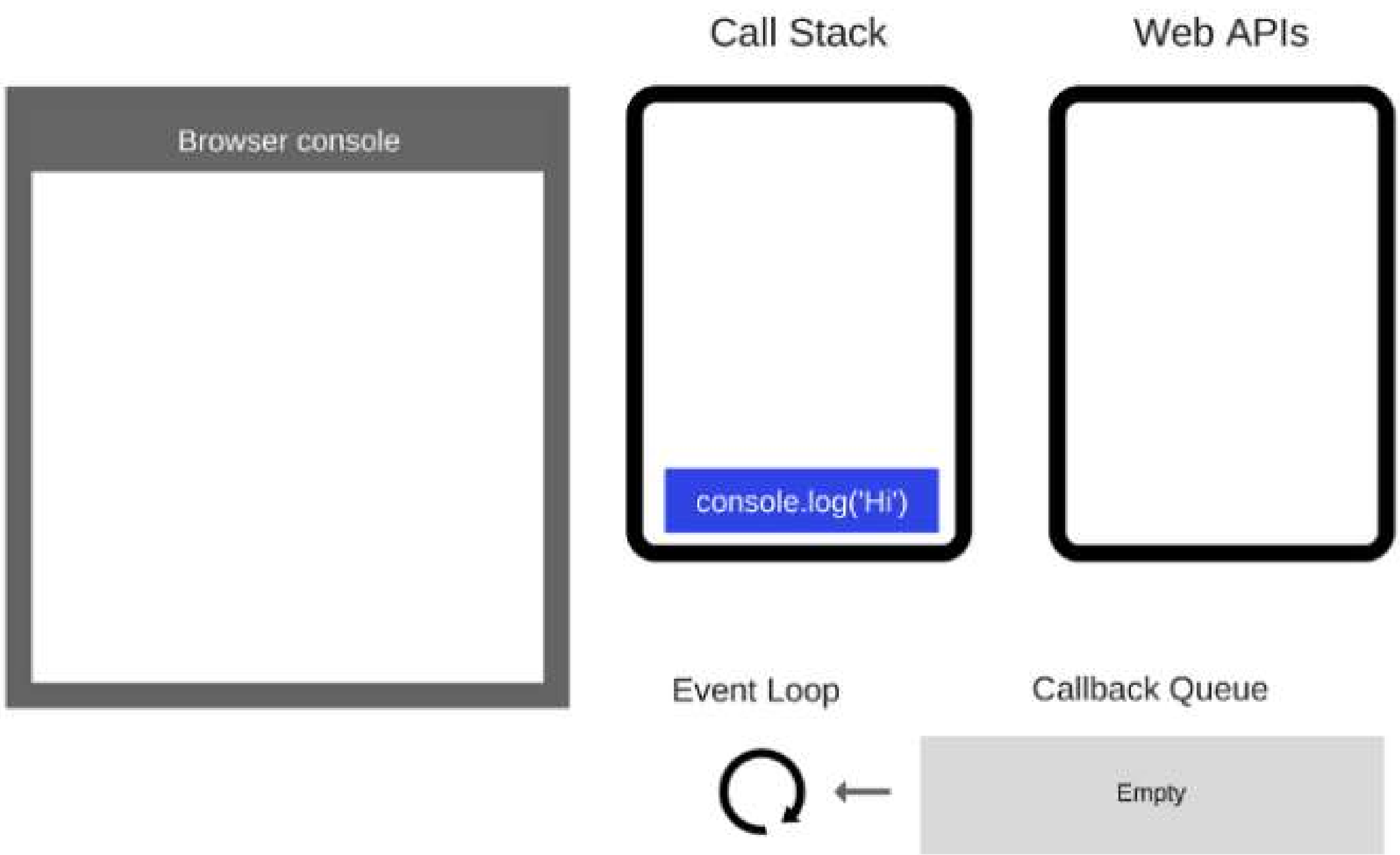
1. O estado é claro. O console do navegador é claro e a call stack está vazia.



Event Loop

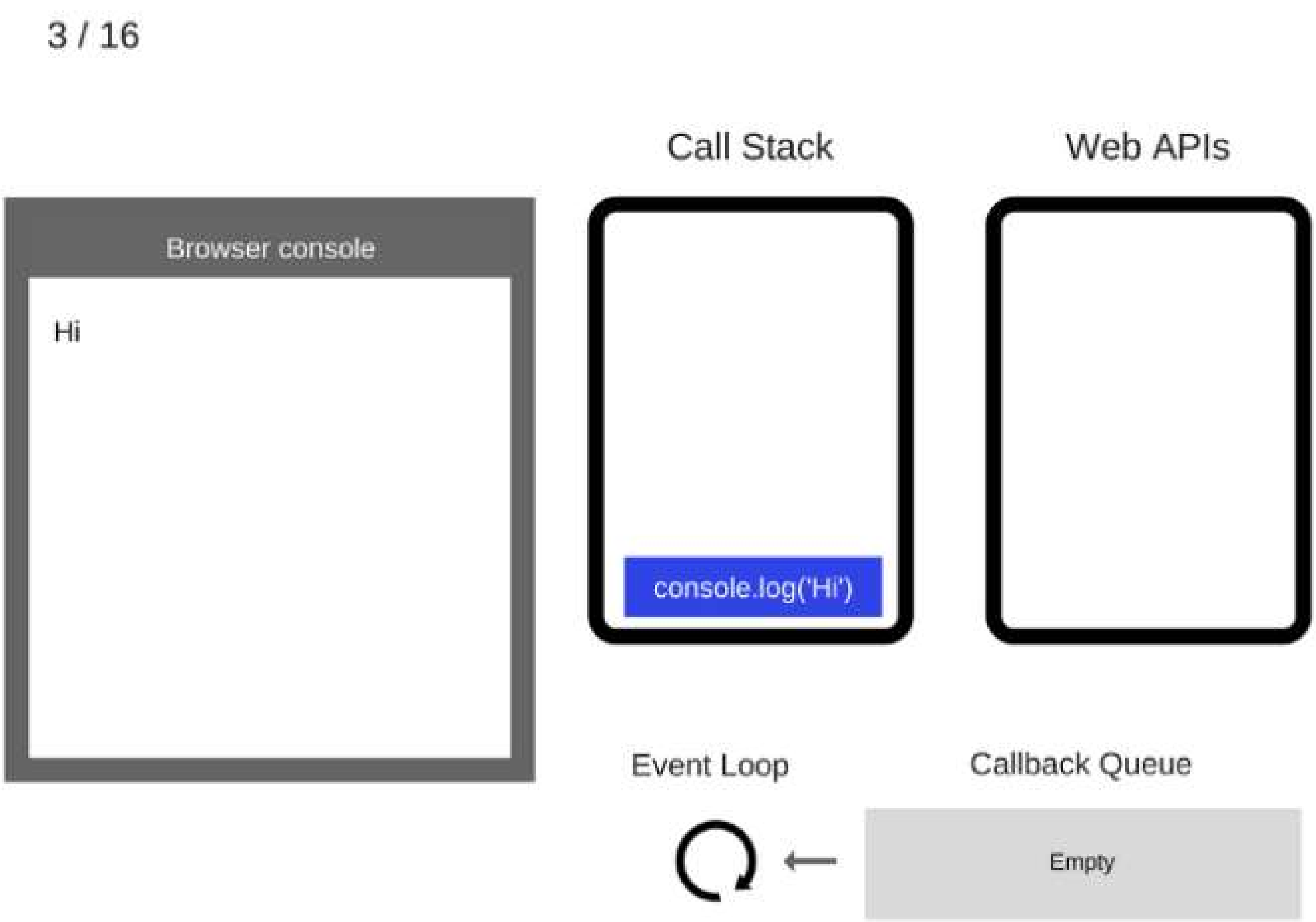
2. `console.log('Hi')` é adicionado à call stack.

2 / 16



Event Loop

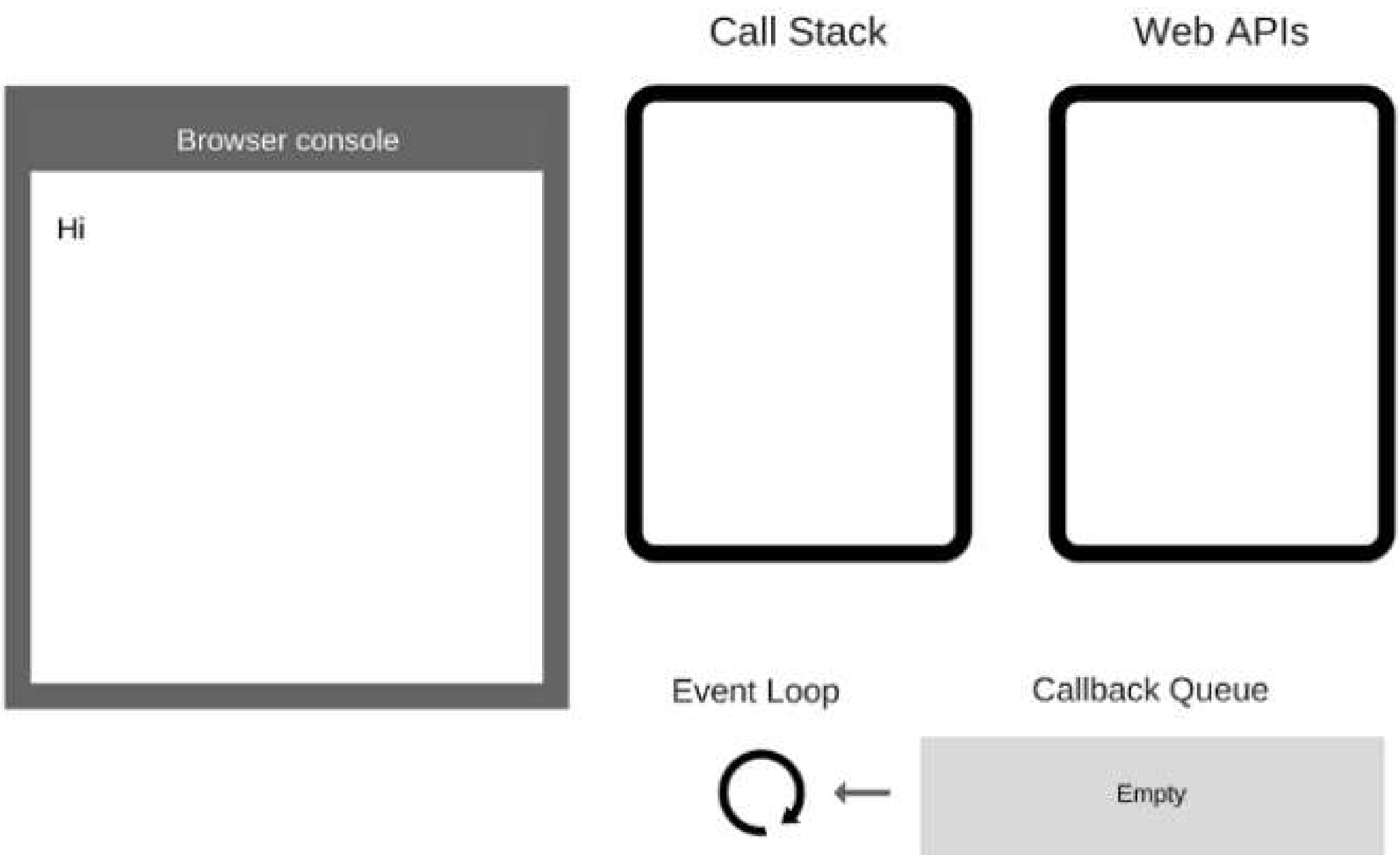
3. `console.log('Hi')` é executado.



Event Loop

4. `console.log('Hi')` é removido da pilha de chamadas.

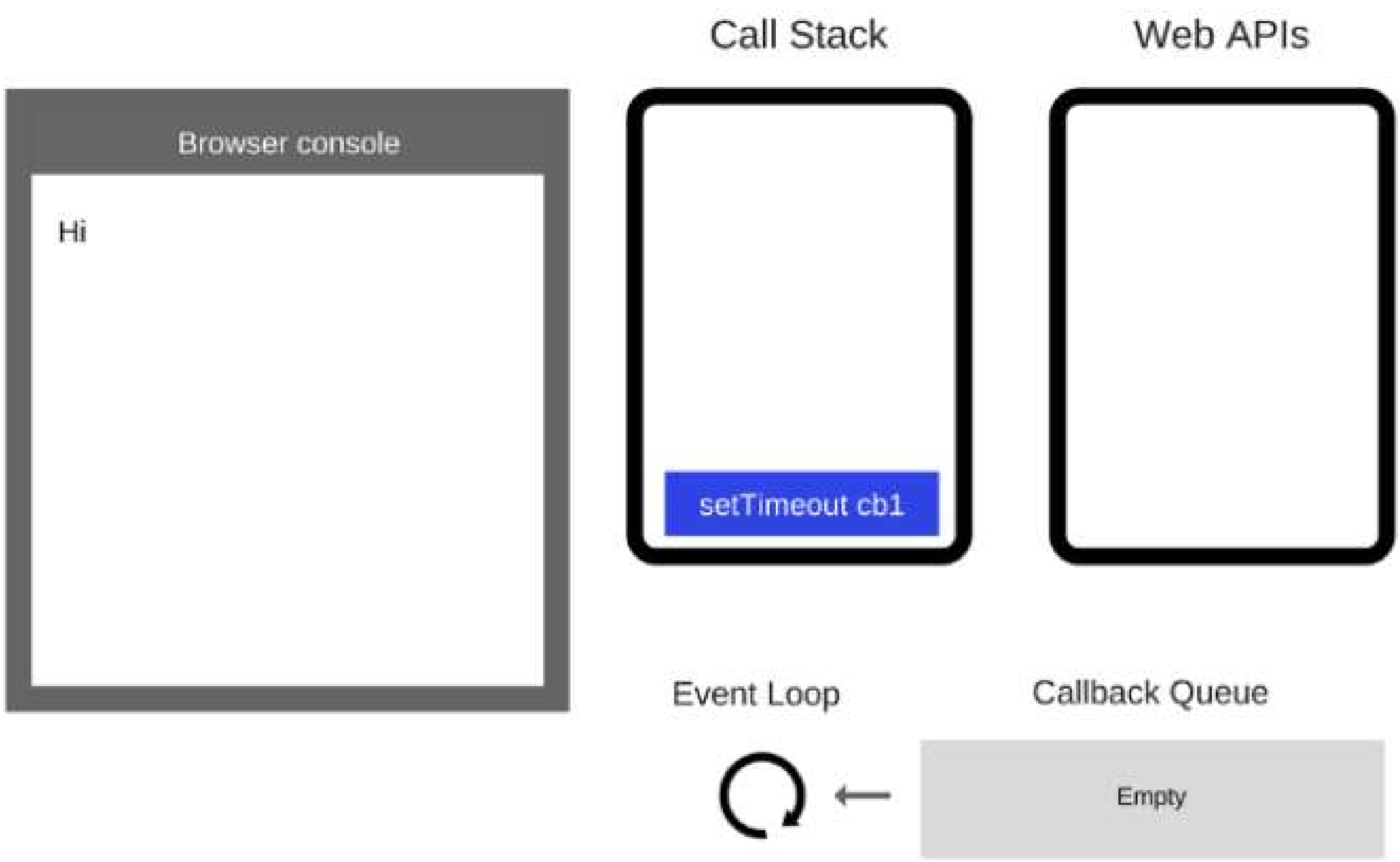
4 / 16



Event Loop

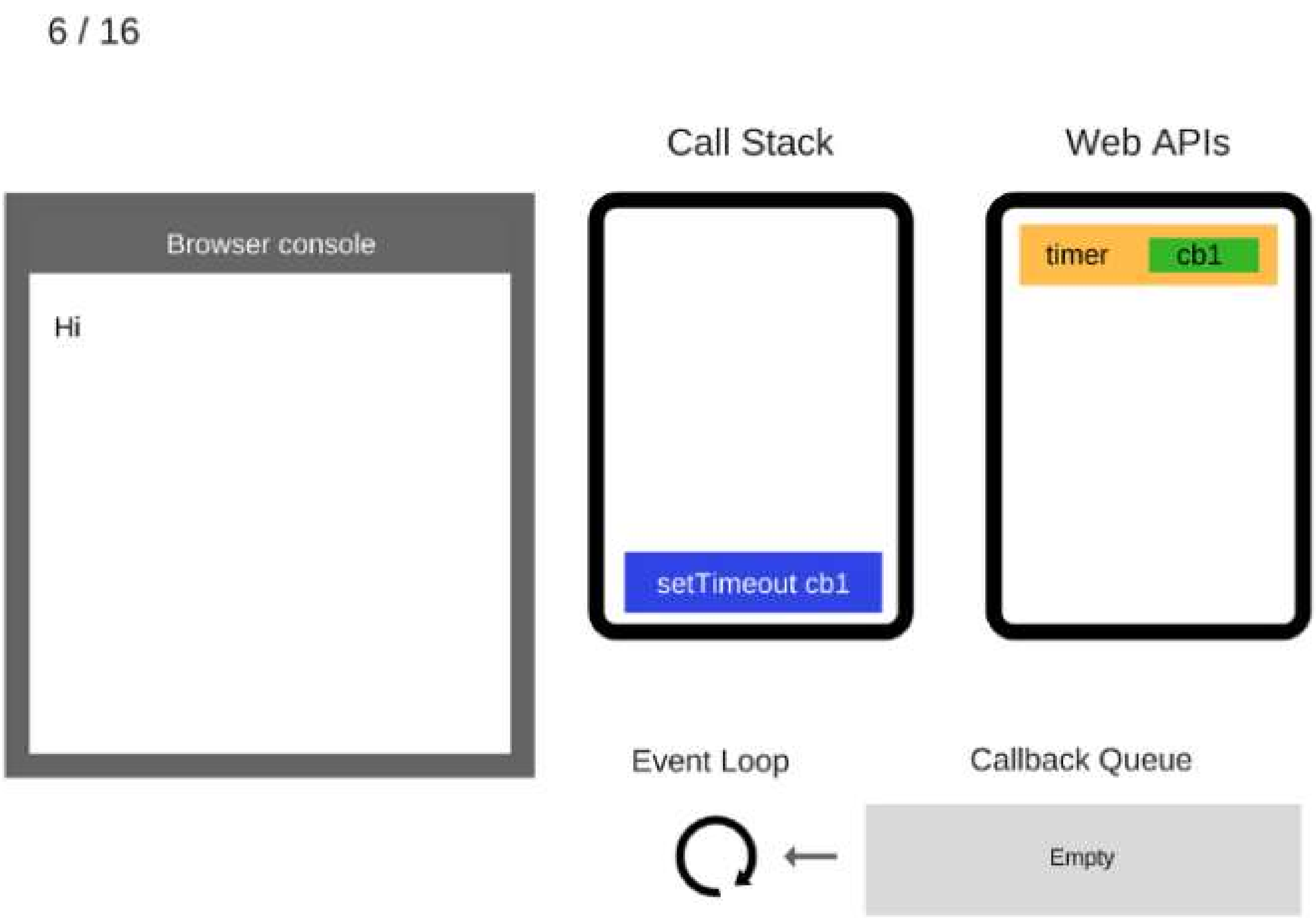
5. `setTimeout(function cb1() { ... })` é adicionado à call stack.

5 / 16



Event Loop

6. `setTimeout(function cb1() { ... })` é executado. O navegador cria um cronômetro como parte das APIs da Web. Ele vai lidar com a contagem regressiva para você.

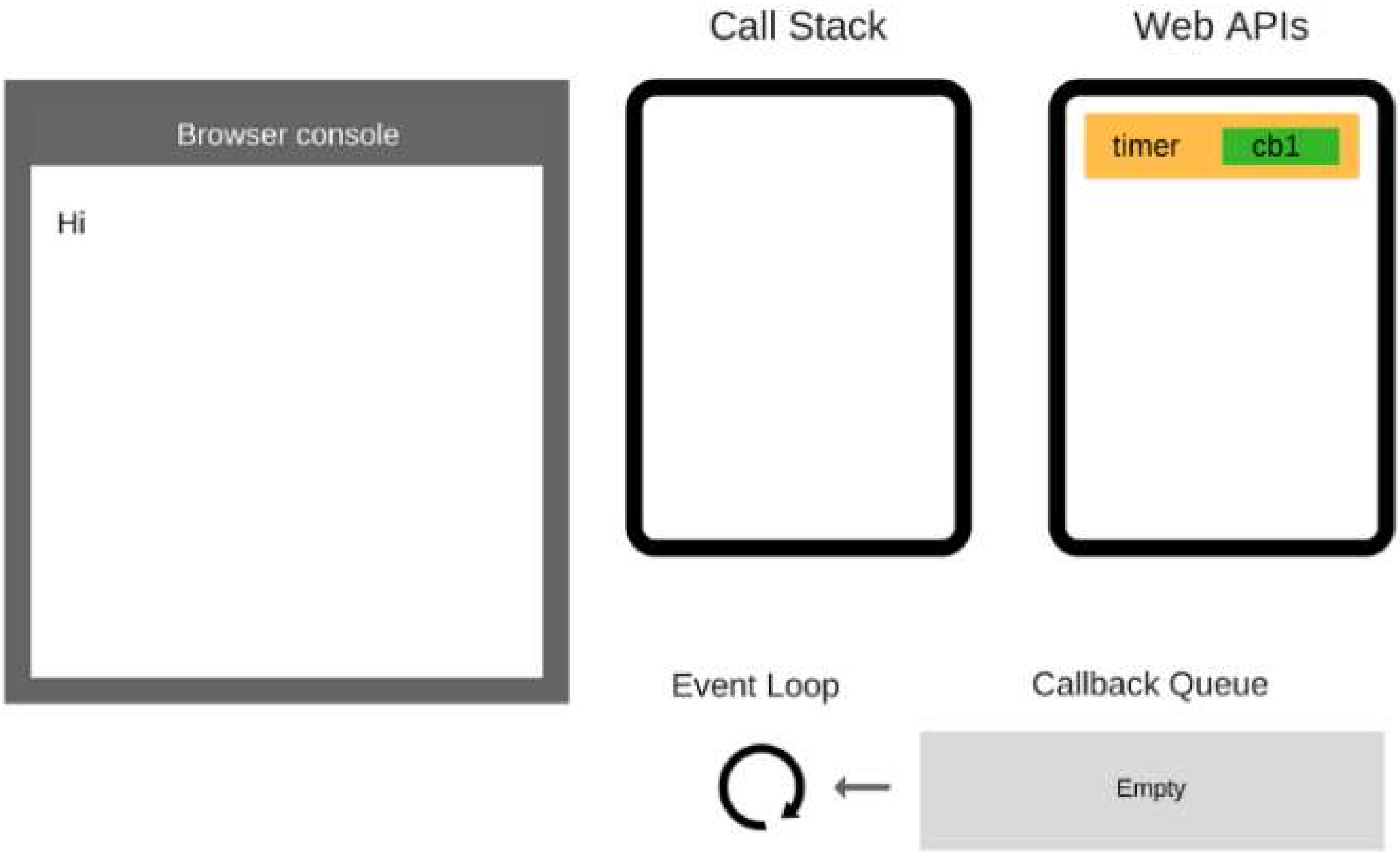


Event Loop

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

7. O `setTimeout(function cb1() { ... })` está completo e é removido da call stack.

7 / 16

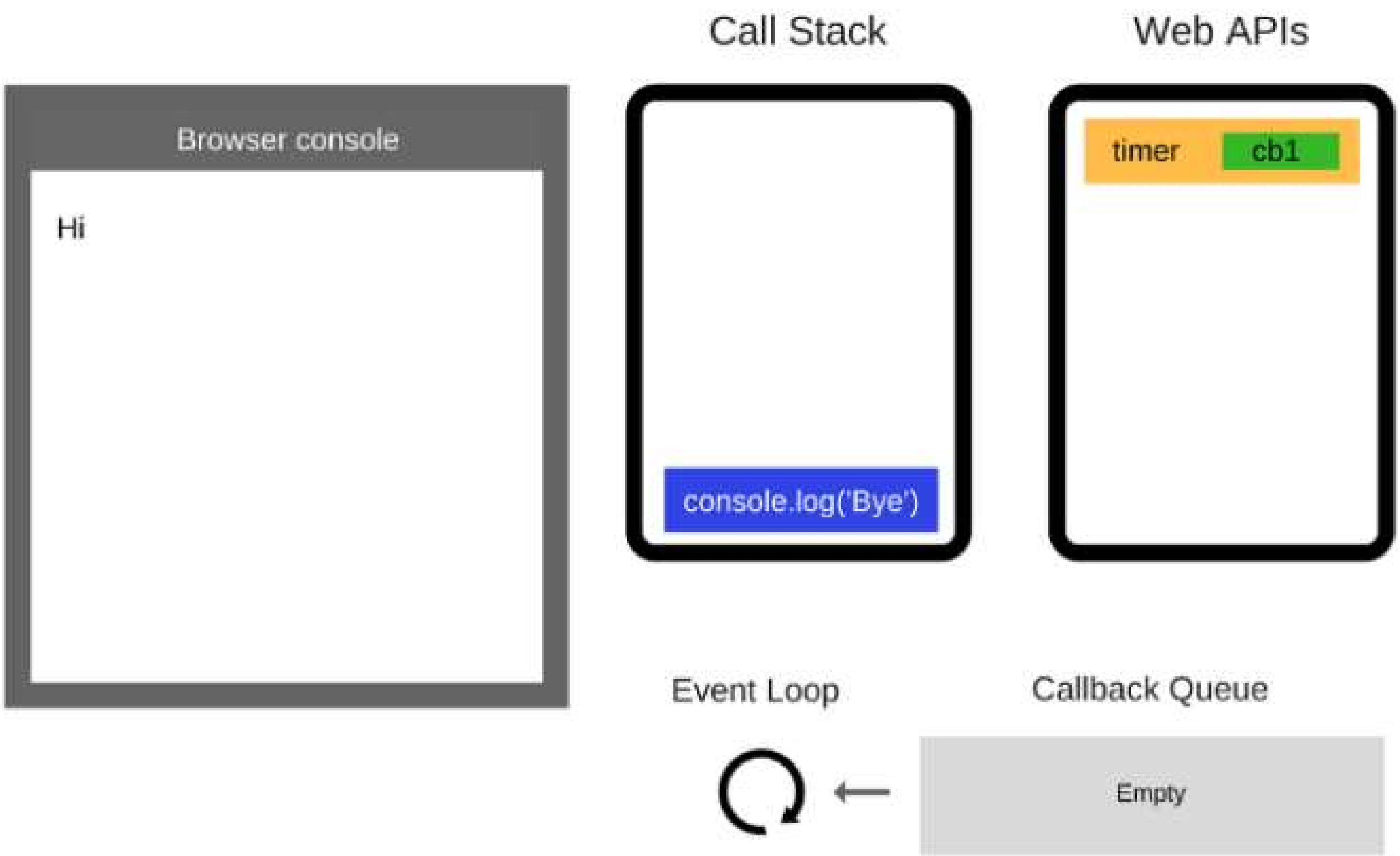


Event Loop

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

8. `console.log('Bye')` é adicionado à call stack.

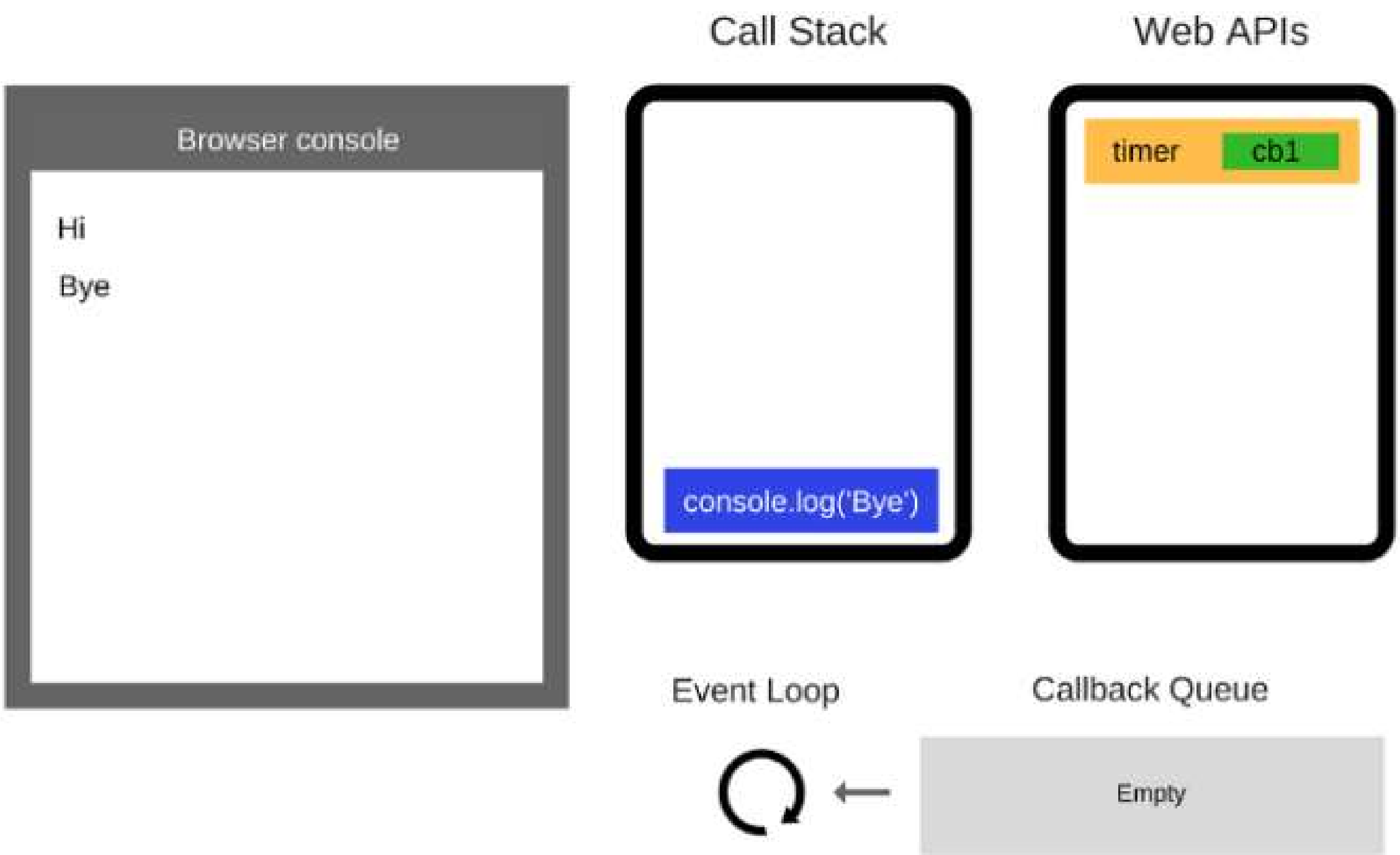
8 / 16



Event Loop

9. `console.log('Bye')` é executado.

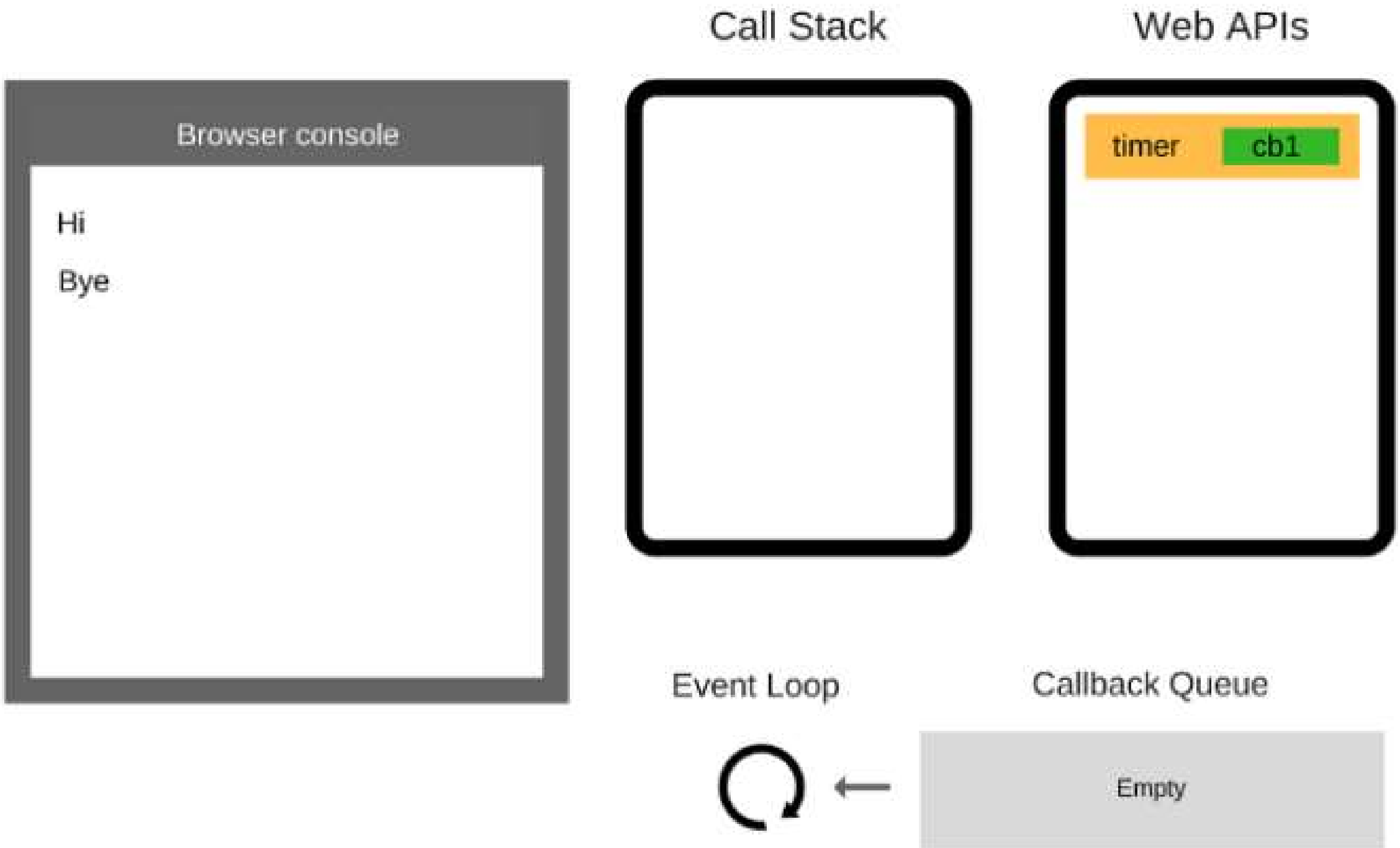
9 / 16



Event Loop

10. `console.log('Bye')` é removido da call stack.

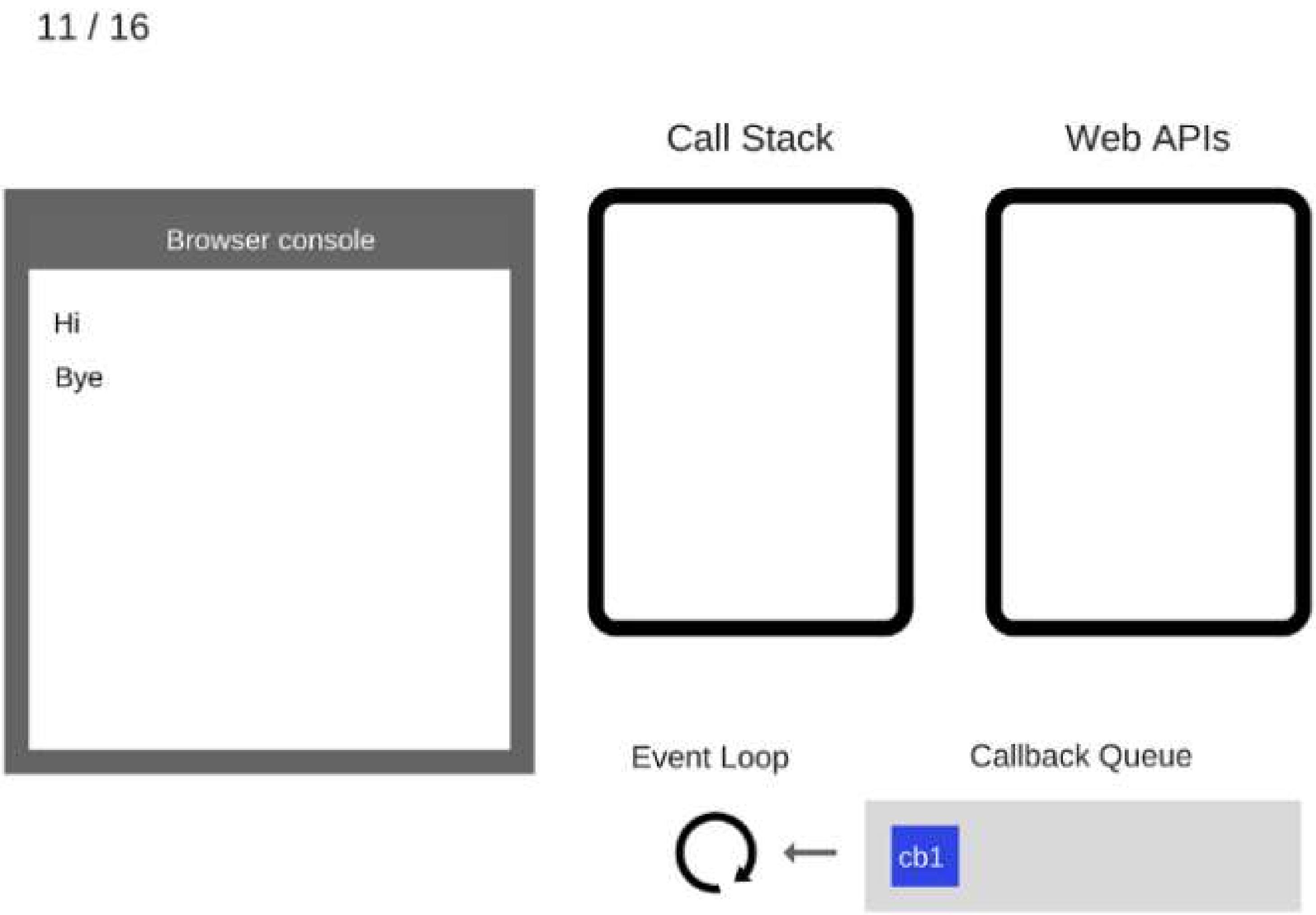
10 / 16



Event Loop

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070

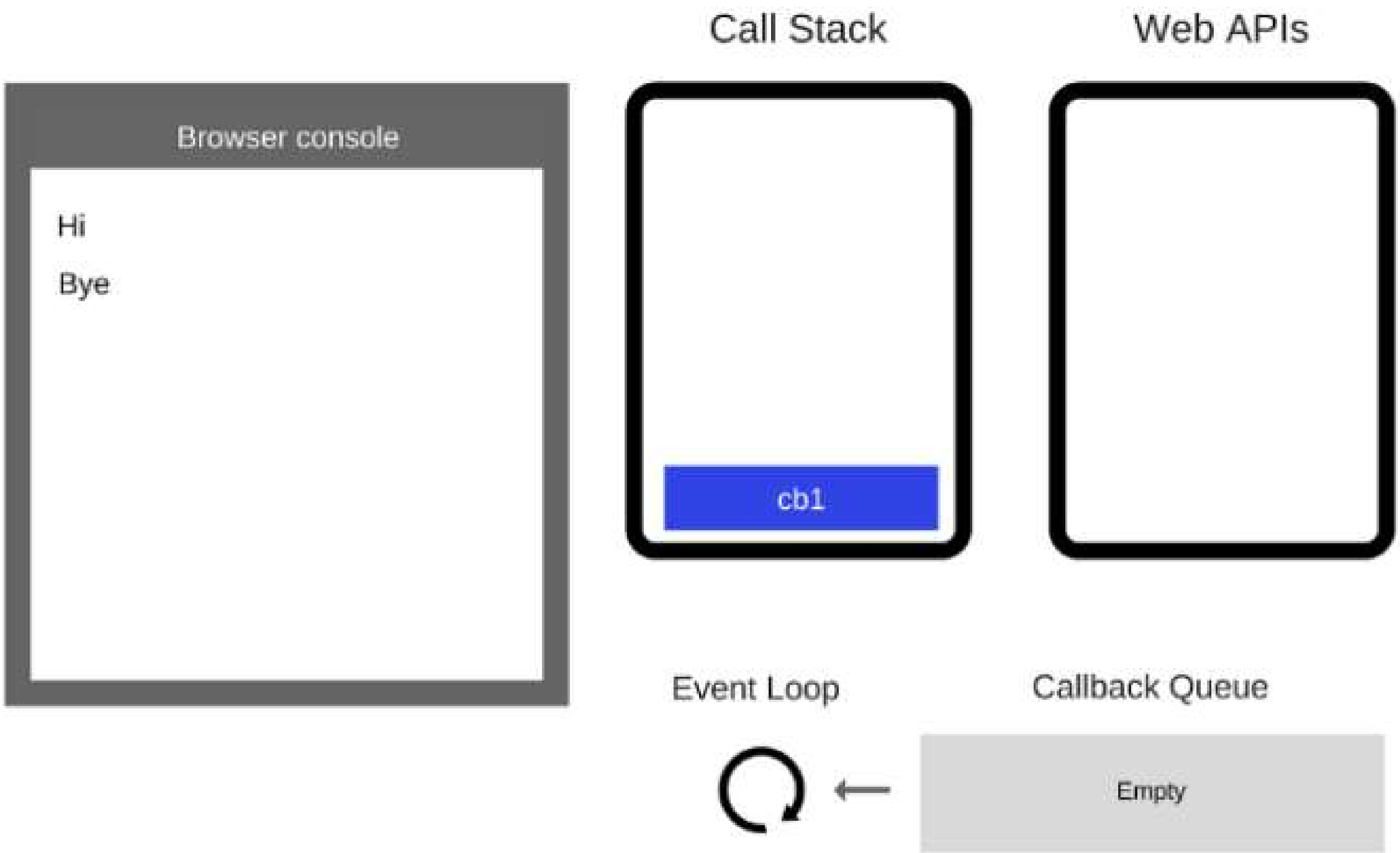
11. Após pelo menos 5000 ms, o temporizador é concluído e ele envia o retorno de chamada `cb1` para a fila de retorno de chamada.



Event Loop

12. O Event Loop retira o `cb1` da Fila de Retorno de Chamada e o envia para a call stack.

12 / 16

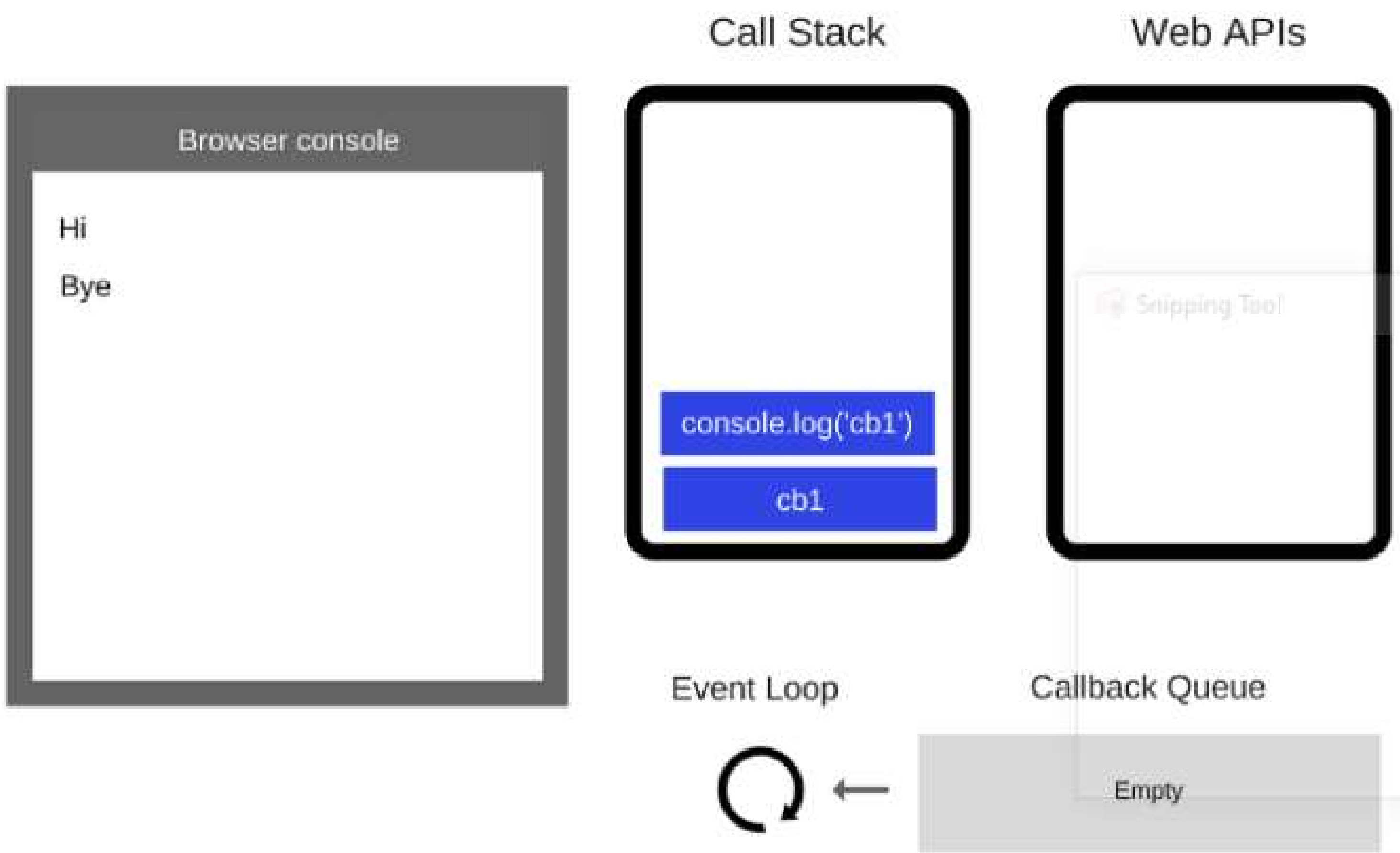


Event Loop

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

13. O `cb1` é executado e adiciona o `console.log('cb1')` à call stack.

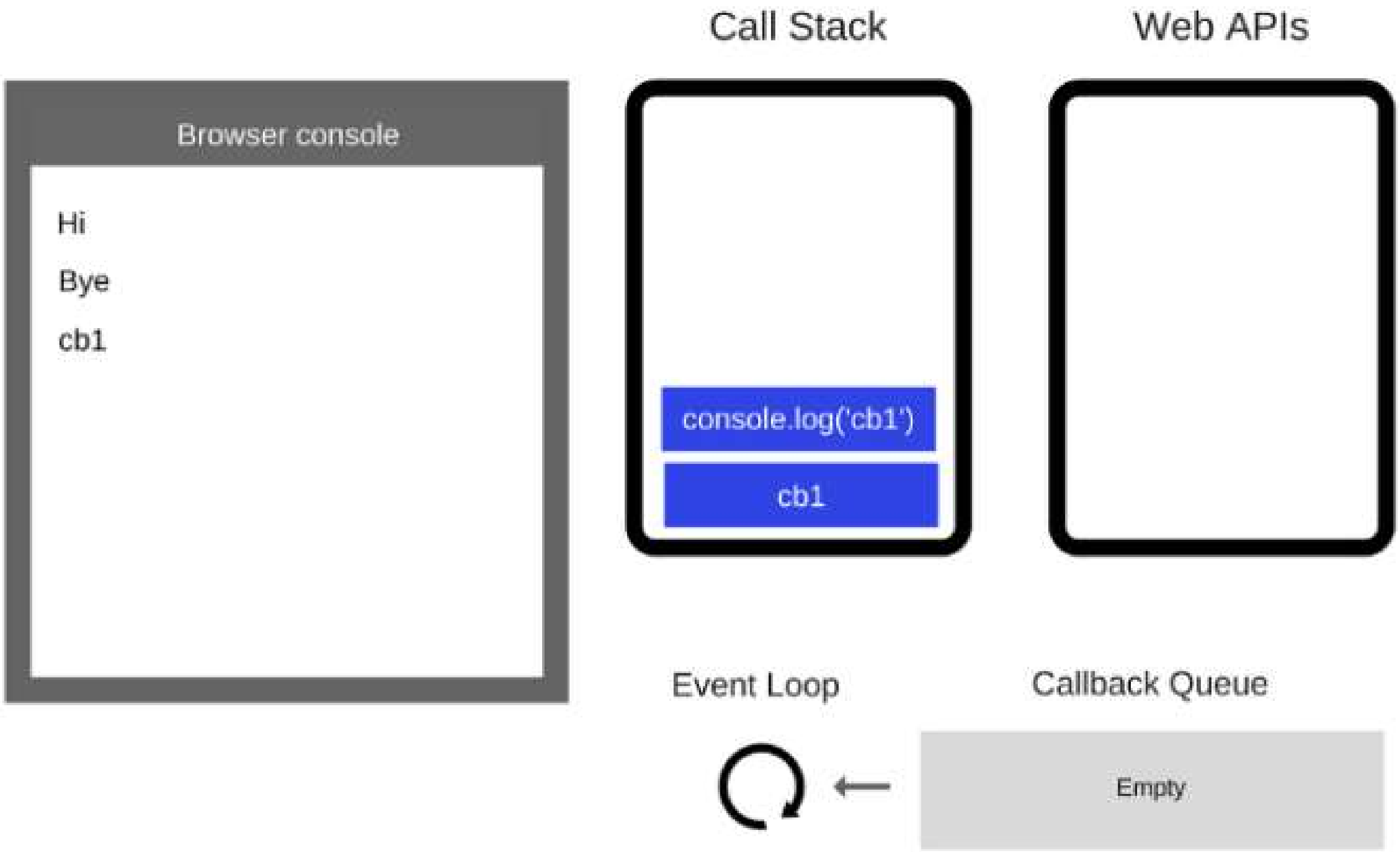
13 / 16



Event Loop

14. `console.log('cb1')` é executado.

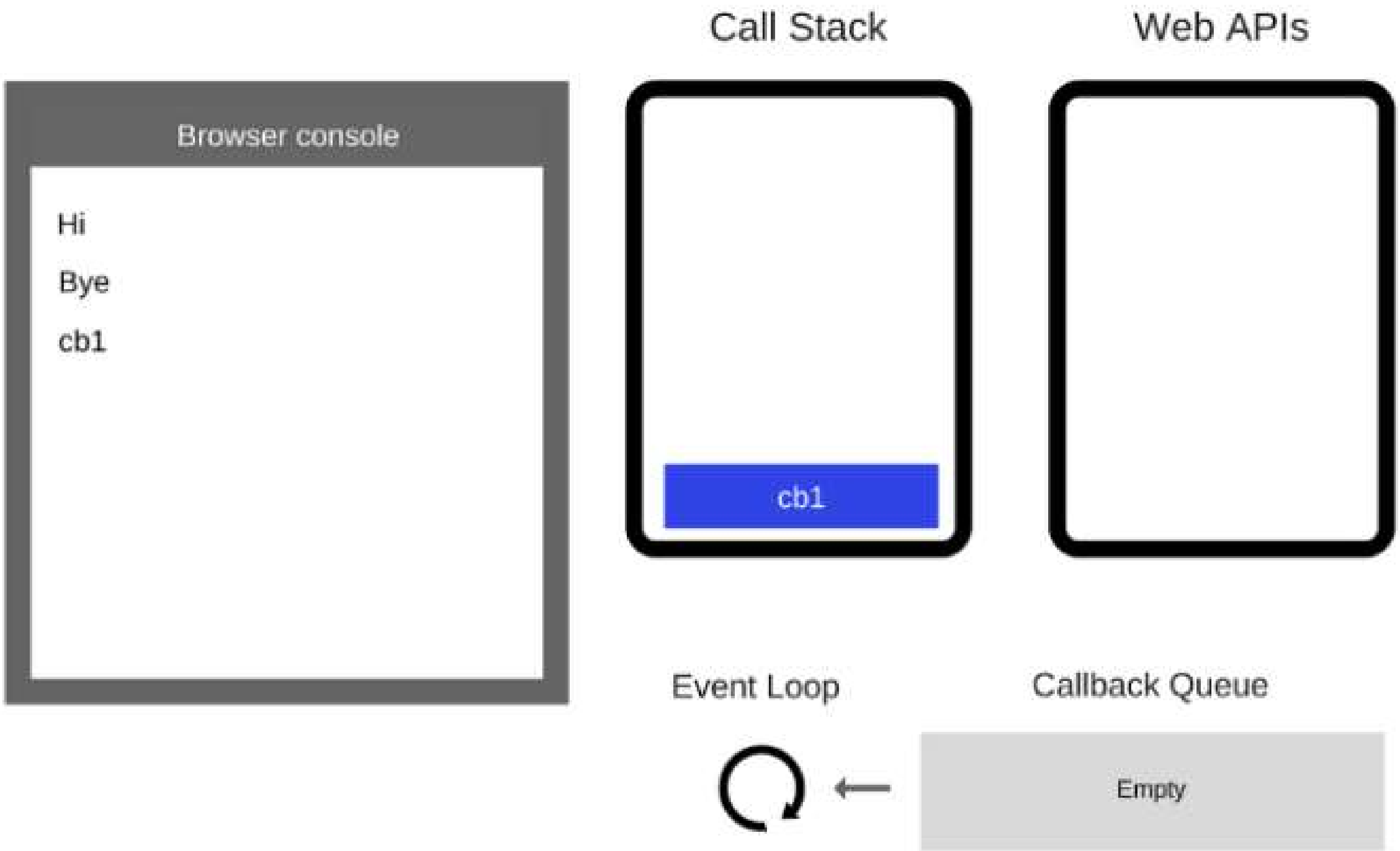
14 / 16



Event Loop

15. O `console.log('cb1')` é removido da pilha de chamadas.

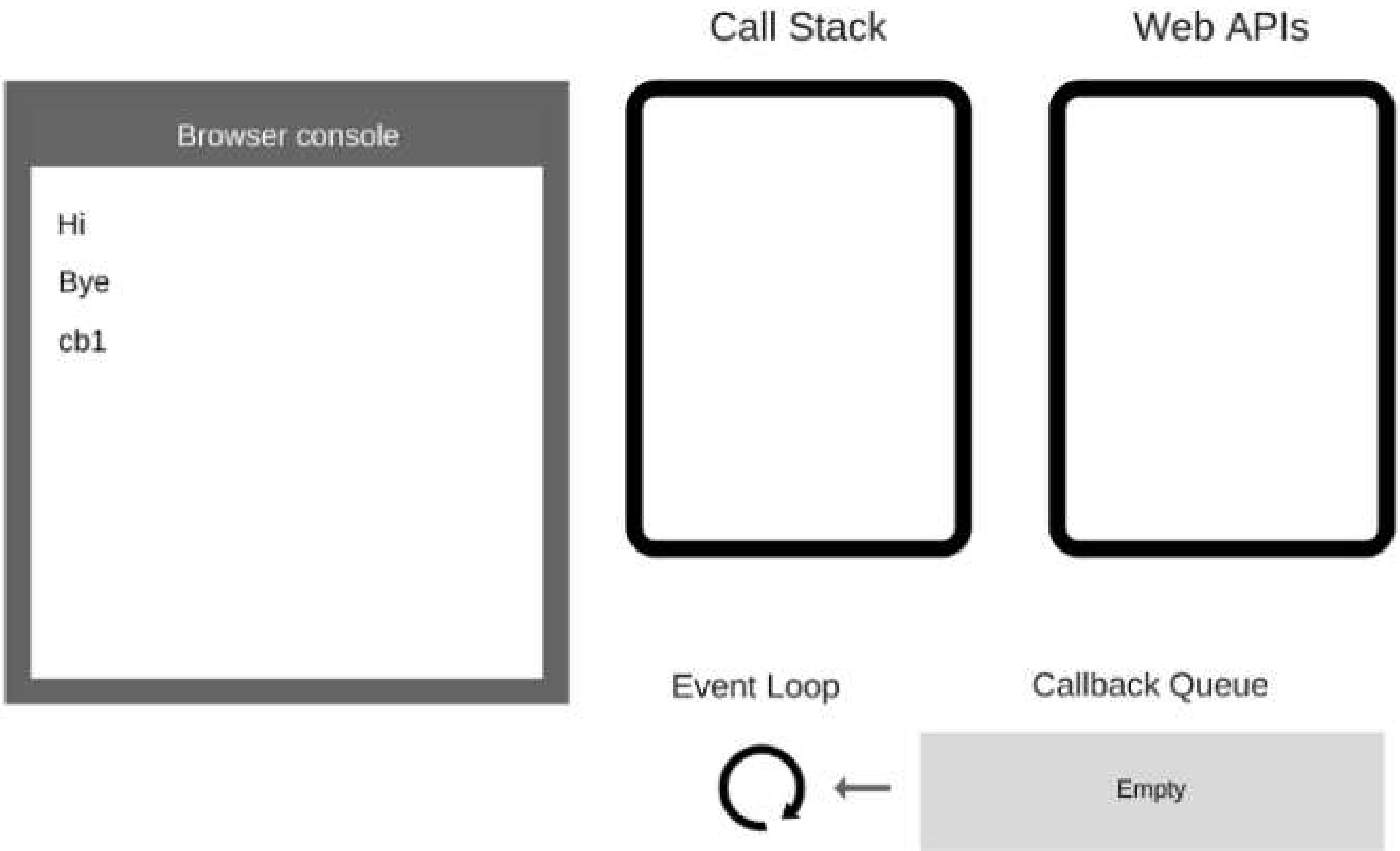
15 / 16



Event Loop

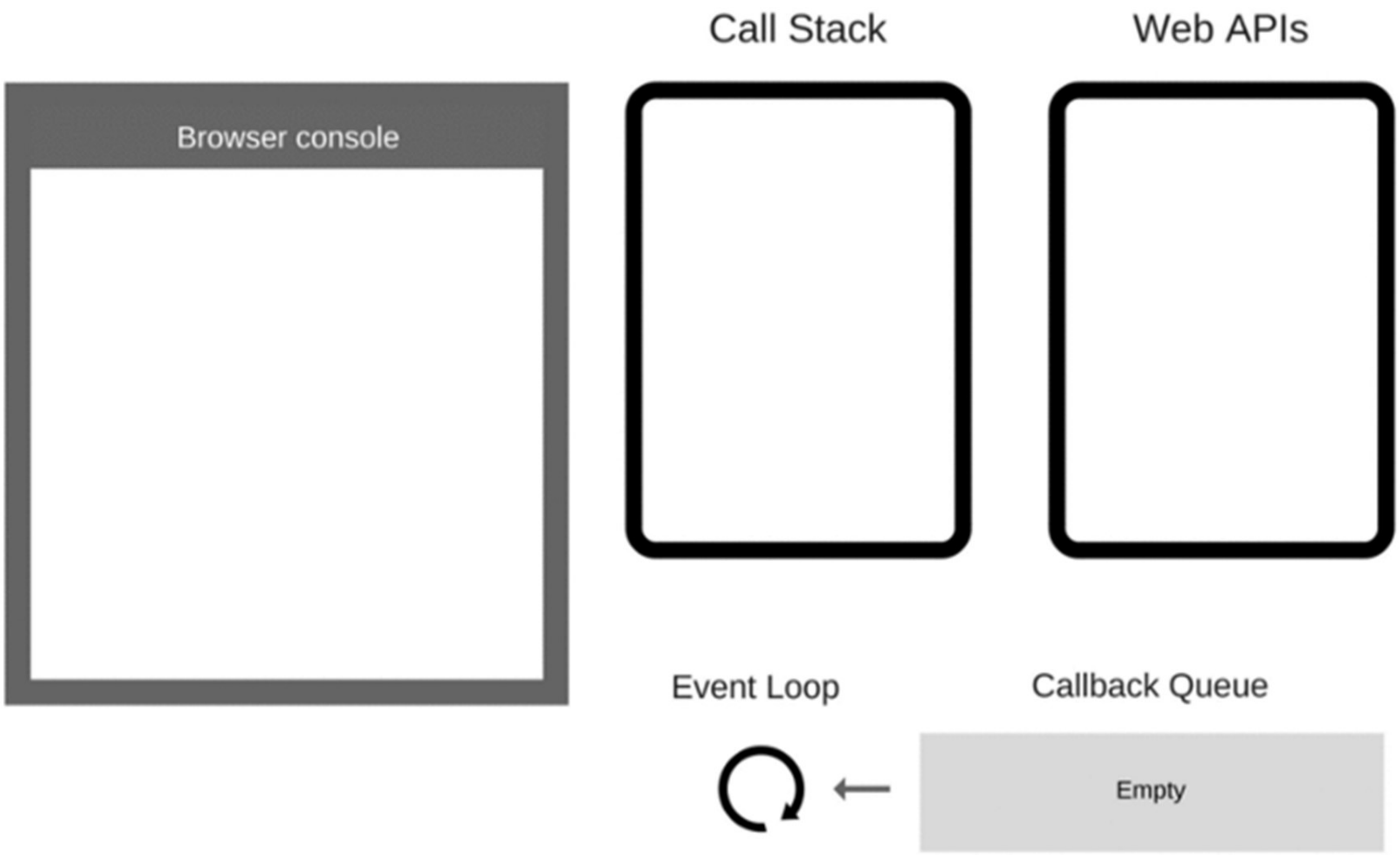
16. O `cb1` é removido da call stack.

16 / 16



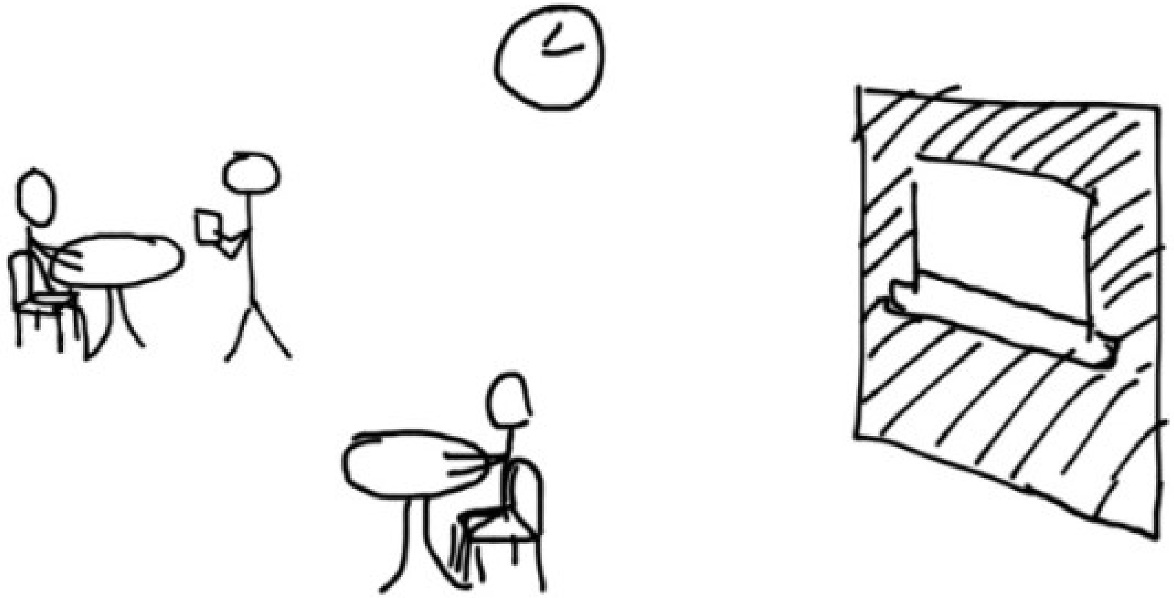
Event Loop - recapitulação

1 / 16

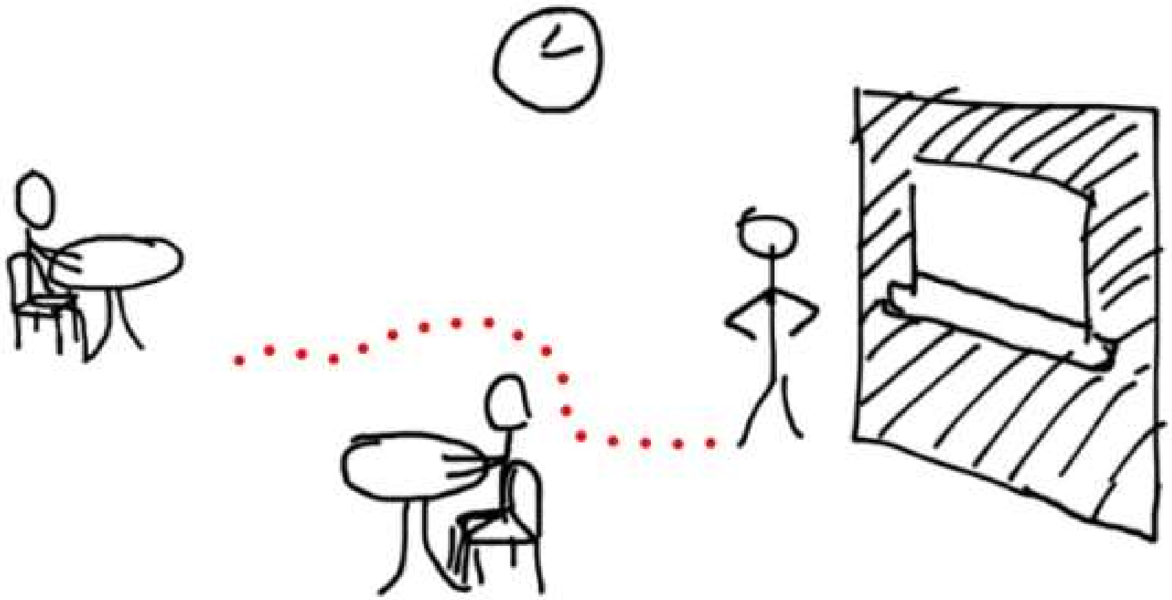


Event Loop - Assincrono

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



Event Loop - Assincrono



Event Loop - Assincrono

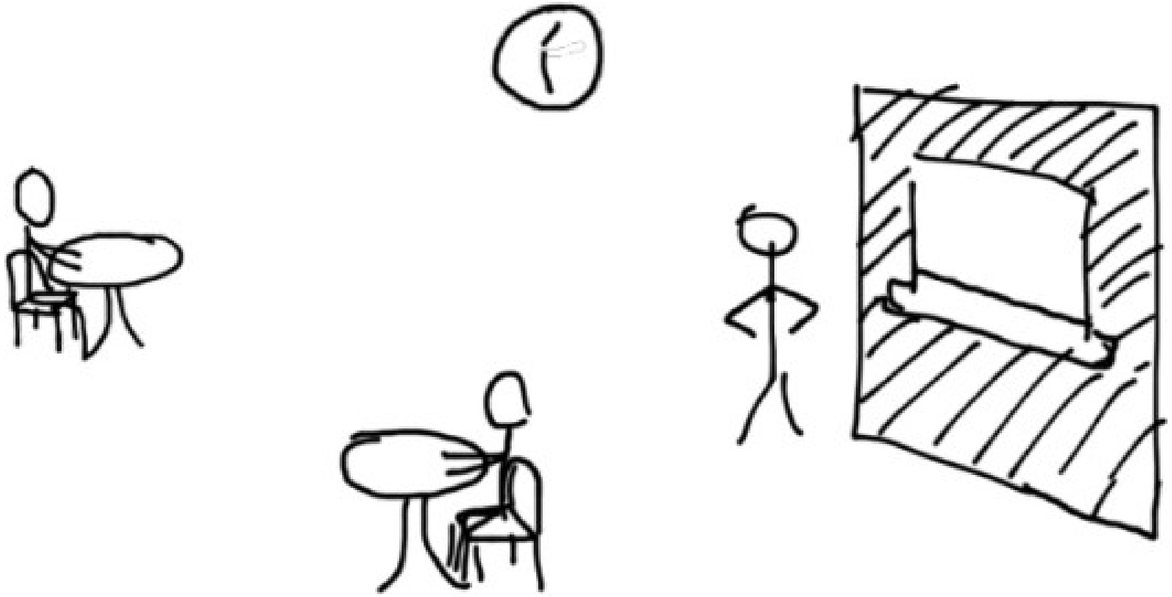
2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

Event Loop - Assincrono

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

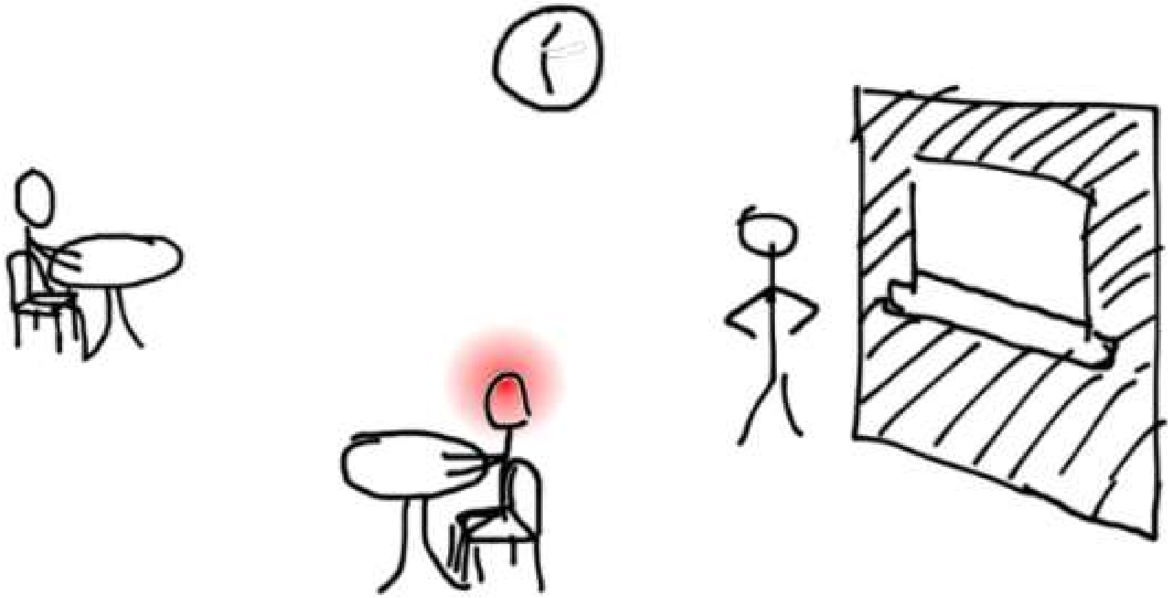
Event Loop - Assincrono

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

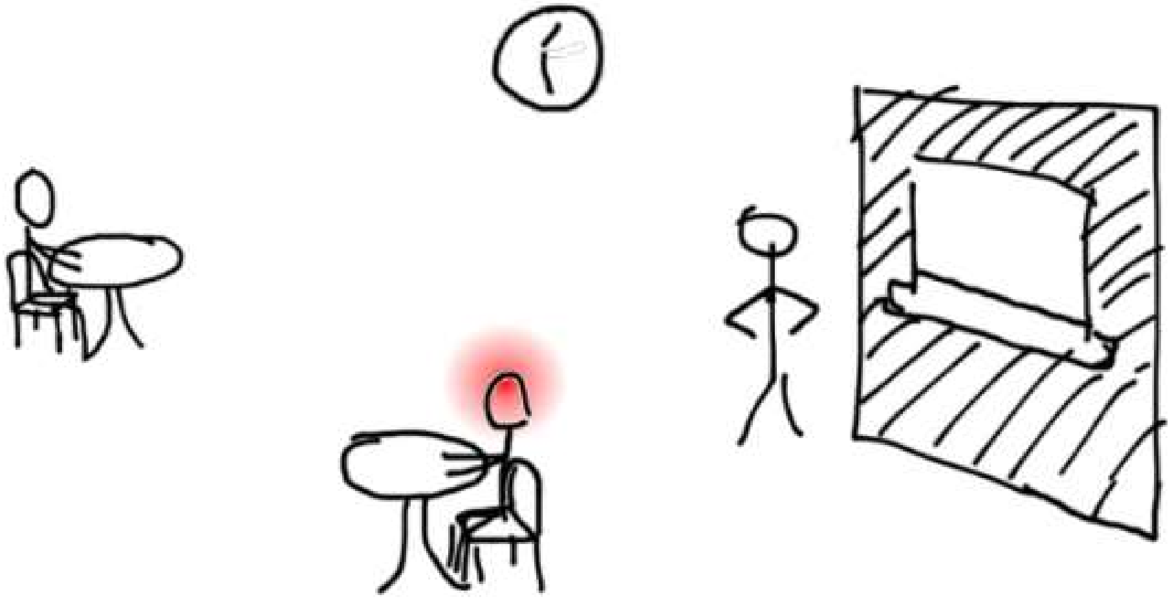


Event Loop - Assincrono

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

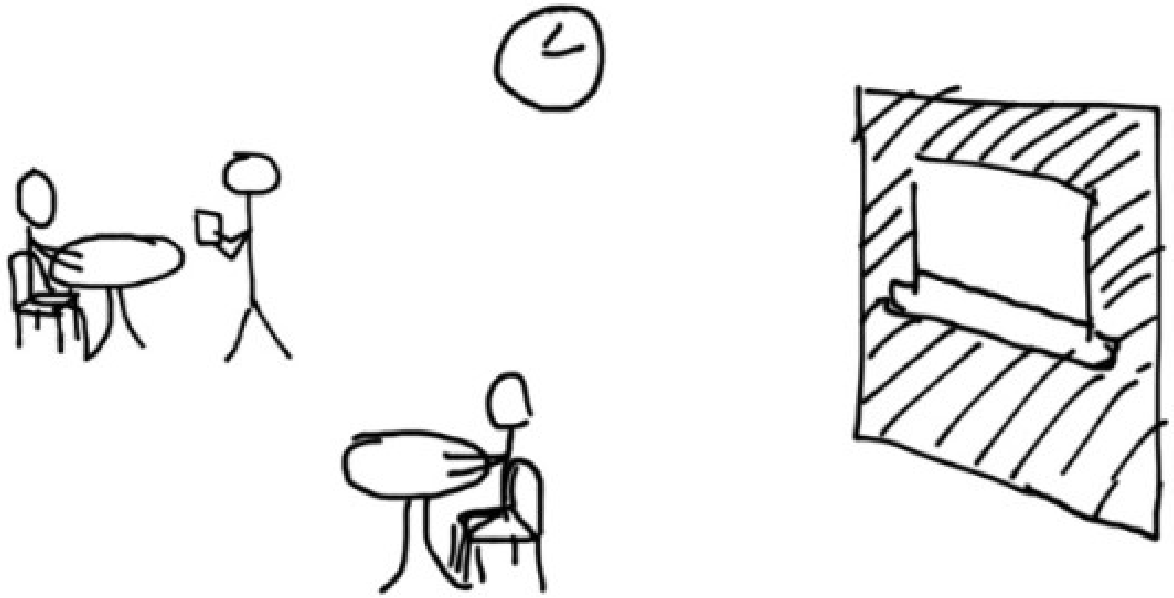


Event Loop - Assincrono

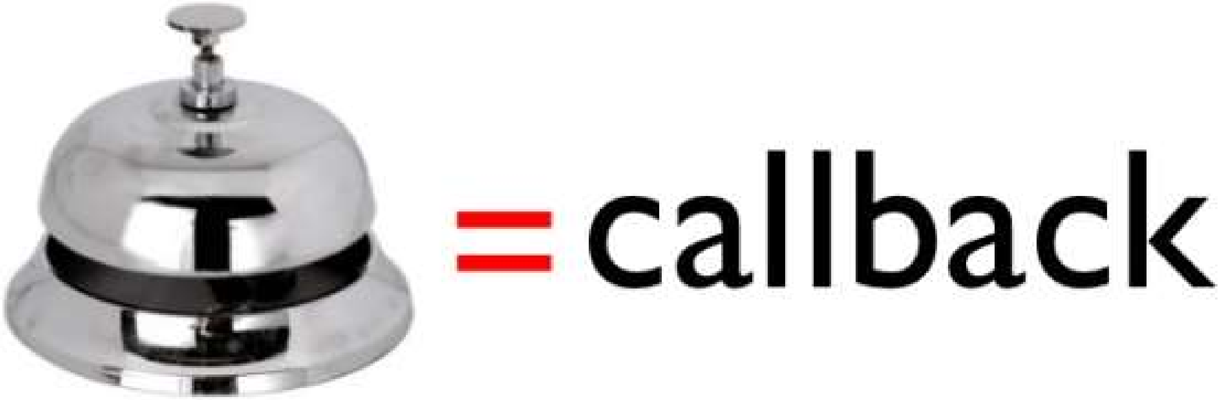


Event Loop - Assincrono

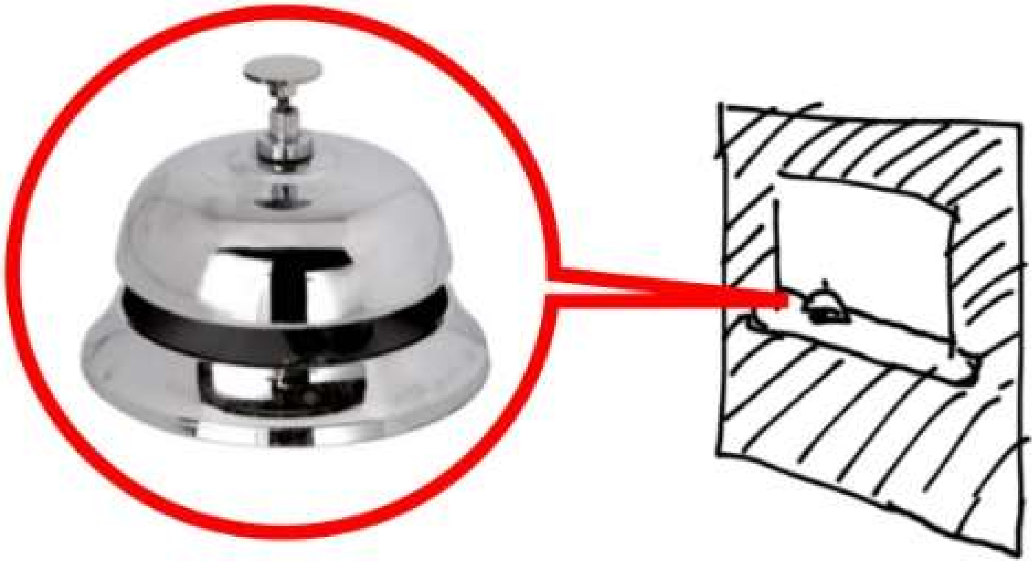
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



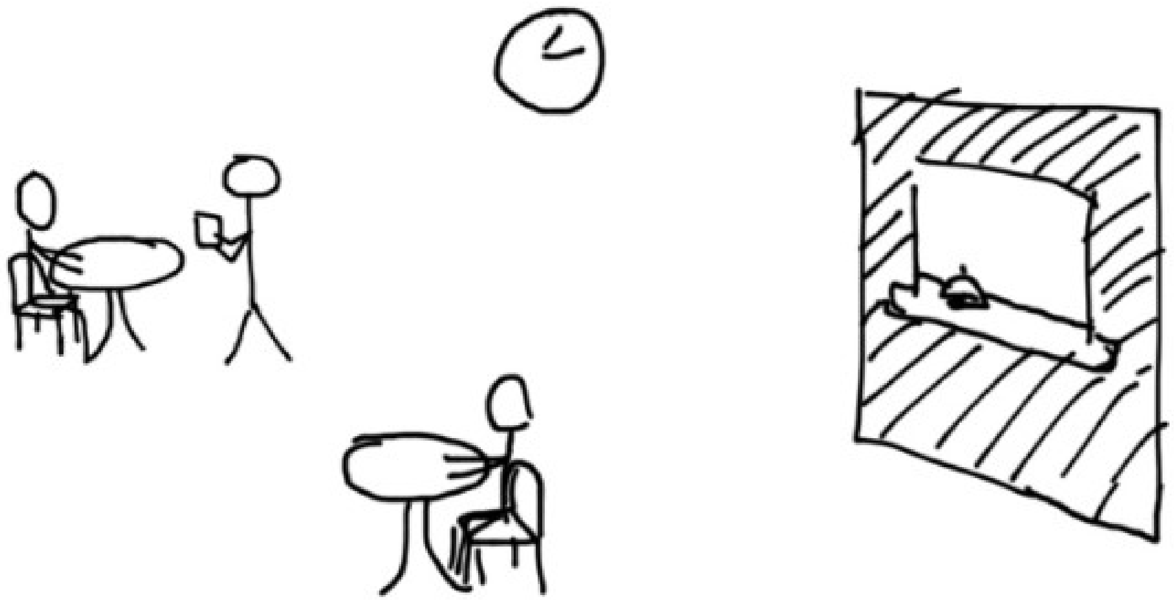
Event Loop – Call back



Event Loop – Assincrono

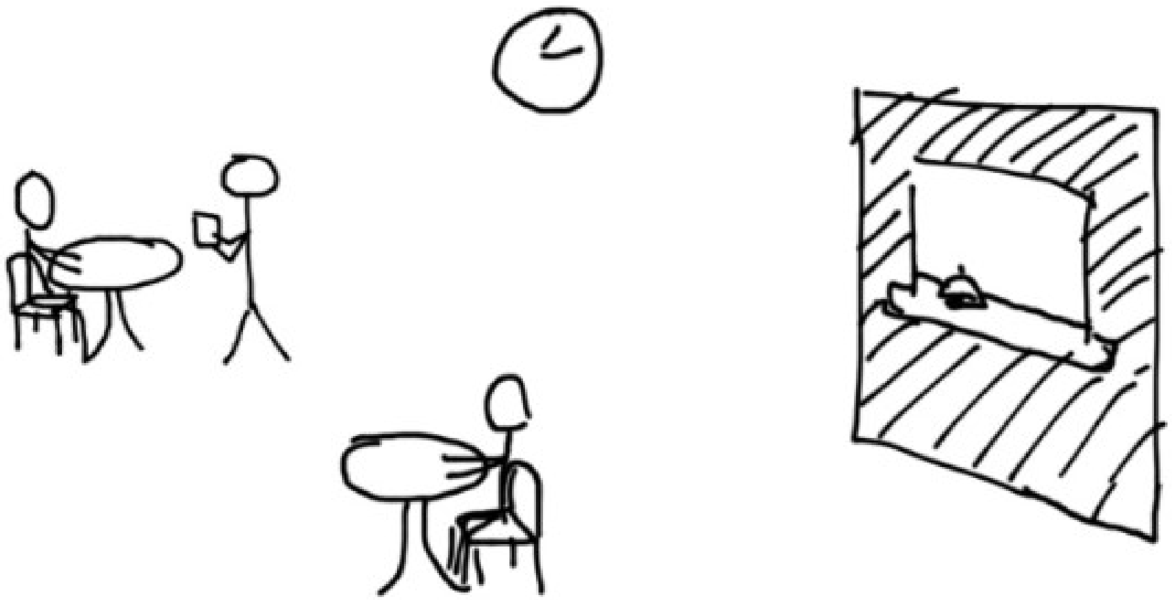


Event Loop – Assincrono



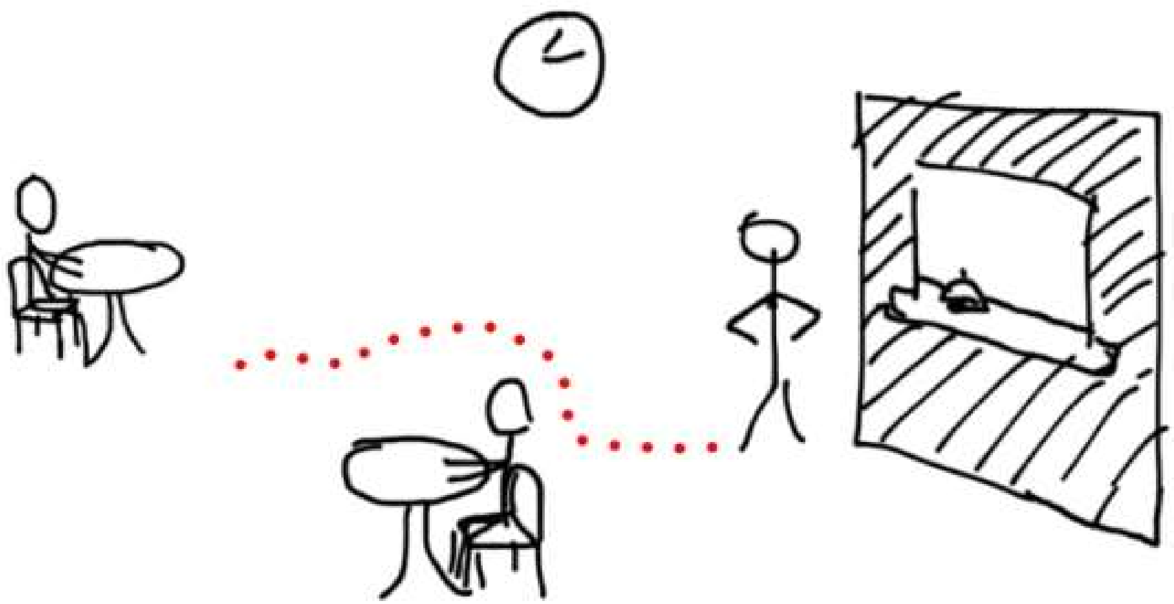
Event Loop – Assincrono

2f486967682070657266 6f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

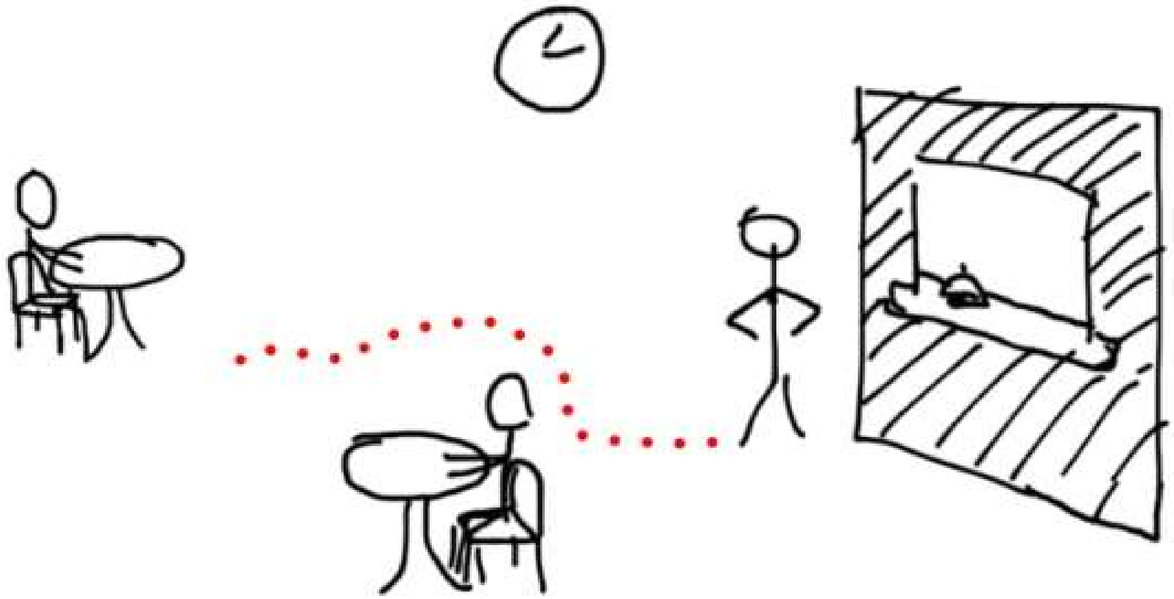


Event Loop – Assincrono

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

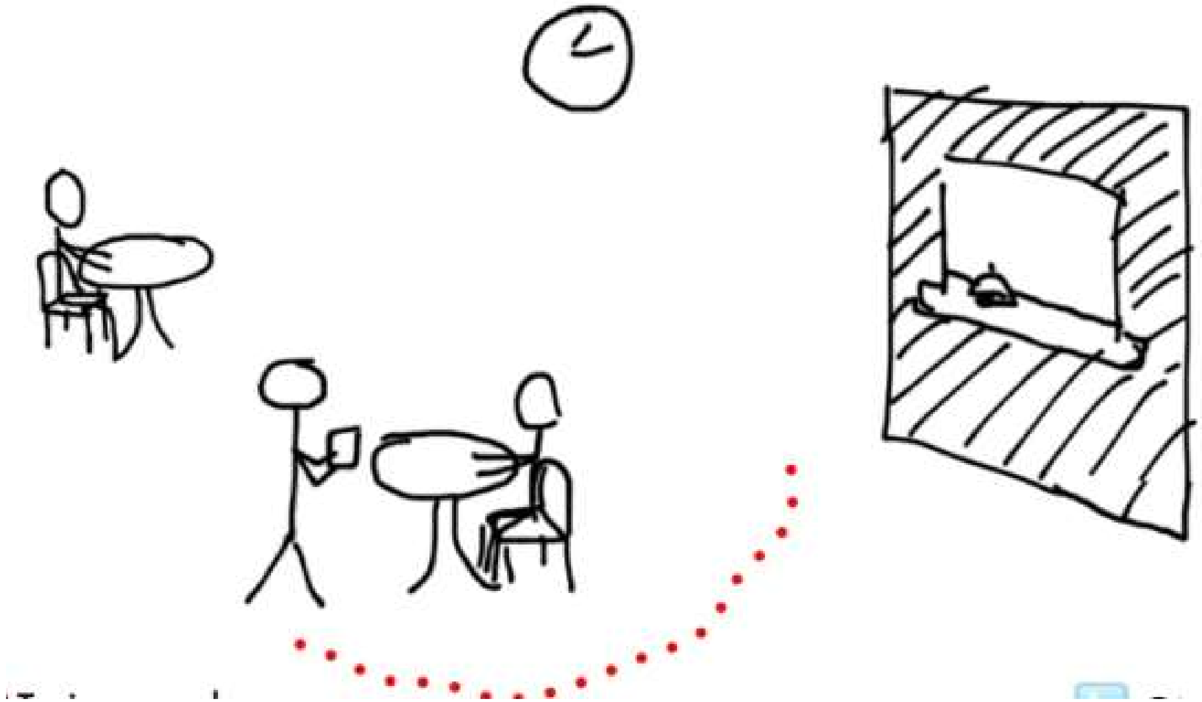


Event Loop – Assincrono

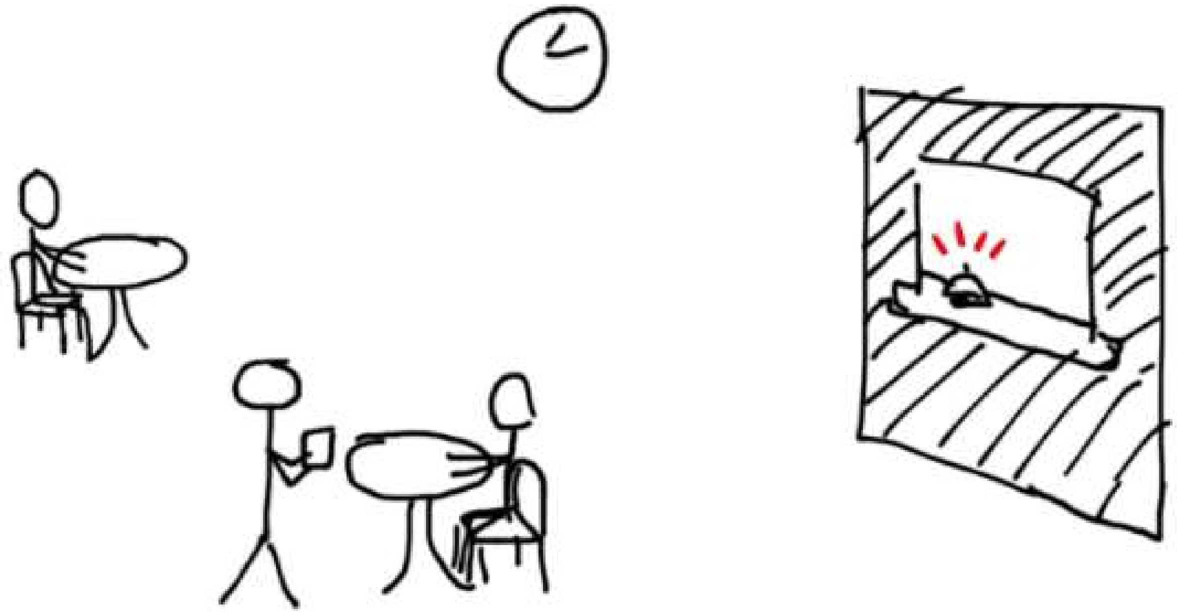


Event Loop – Assincrono

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

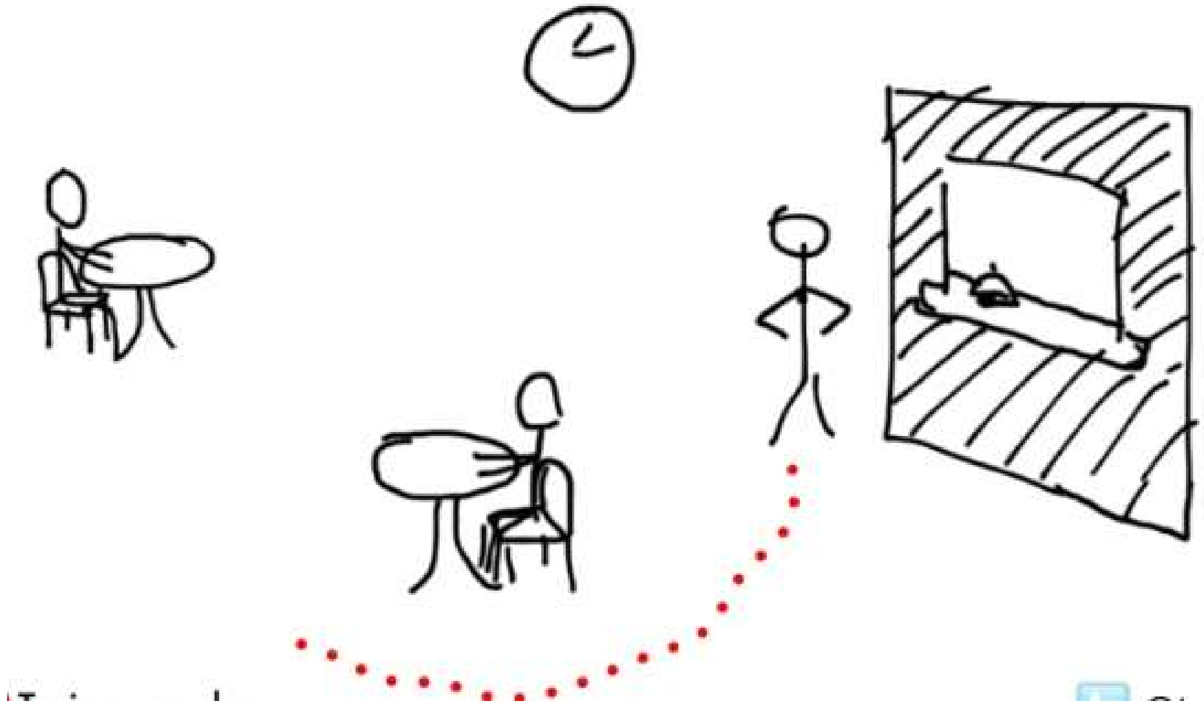


Event Loop –



Event Loop – Call back

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



Event Loop – Call back

Conclusão:

- **Callbacks** nada mais são do que funções que levam algum tempo para produzir um resultado;

- Normalmente, esses retornos de chamada assíncronos são usados para acessar valores de bancos de dados, fazer download de imagens, ler arquivos etc.

- Como isso leva tempo para terminar, não podemos prosseguir para a próxima linha, pois isso pode gerar um erro, nem podemos pausar nosso programa;

- Portanto, precisamos armazenar o resultado e retornar a chamada quando estiver concluído;

Event Loop – Call back

CALLBACK HELL:

Event Loop – Call back

CALLBACK HELL:

- É um grande problema causado pela codificação com retornos de chamada aninhados complexos;

- Cada retorno de chamada recebe um argumento que é resultado dos retornos de chamada anteriores. Dessa forma, a estrutura do código parece uma pirâmide, dificultando a leitura e a manutenção. Além disso, se houver um erro em uma função, todas as outras funções serão afetadas.