Event Loop

Event Loop je mechanismus používaný v prostředích založených na událostech (event-driven systems), který zajišťuje zpracování událostí a asynchronních operací. Je klíčovou součástí technologií jako javascript a node.js.

Jeho úkolem je nepřetržitě sledovat frontu událostí a spouštět příslušné funkce ve chvíli, kdy jsou připraveny k vykonání.

Proč je Event Loop potřeba

JavaScript běží ve většině případů v jednom vlákně (single-threaded). To znamená, že dokáže v jednom okamžiku vykonávat pouze jednu operaci.

Přesto mohou aplikace současně:

načítat data ze serveru, pracovat se soubory, čekat na kliknutí uživatele, spouštět časovače, zpracovávat síťovou komunikaci.

To umožňuje právě mechanismus Event Loop.

Princip fungování

Event Loop neustále kontroluje, zda jsou ve frontě připravené úlohy.

```
 Call Stack ↓ Asynchronní operace ↓ Callback Queue ↓ Event Loop ↓ Vykonání callbacku
```

Průběh:

Program spustí asynchronní operaci. Operace běží mimo hlavní vlákno. Po dokončení je callback zařazen do fronty událostí. Event Loop zjistí, zda je hlavní zásobník volný. Pokud ano, callback se vykoná.

Call Stack

Call Stack (zásobník volání) obsahuje aktuálně vykonávané funkce.

Příklad:

```
 function first() { second(); }

function second() {
console.log("Ahoj");
```

```
}

first();
```

Při spuštění se funkce ukládají na zásobník a po dokončení se z něj odebírají.

Callback Queue

Callback Queue obsahuje funkce čekající na spuštění.

Příklad:

```
setTimeout(() => { console.log("Hotovo"); }, 1000);
```

Po uplynutí jedné sekundy je callback zařazen do fronty a čeká na zpracování Event Loopem.

Příklad fungování

```
console.log("První");

setTimeout(() => {
console.log("Druhý");
}, 0);

console.log("Třetí");
```

Výstup:

```
První Třetí Druhý
```

I když je časovač nastaven na 0 ms, callback se vykoná až po dokončení aktuálního kódu.

Event Loop v Node.js

V prostředí node.js Event Loop zajišťuje efektivní obsluhu:

HTTP požadavků, práce se soubory, databázových operací, síťové komunikace, časovačů.

Díky tomu může jediný proces obsluhovat tisíce současných připojení.

Výhody

Efektivní práce s asynchronními operacemi. Nízké nároky na systémové prostředky. Vysoká

škálovatelnost. Jednodušší správa vláken.

Nevýhody

Dlouhé synchronní operace mohou blokovat celý proces. Náročnější pochopení pro začátečníky. Chyby v asynchronním kódu se mohou obtížně ladit.

Související pojmy

javascript node.js event event handler event-driven programming callback promise async await call stack

Shrnutí

Event Loop je mechanismus, který koordinuje zpracování událostí a asynchronních operací v JavaScriptu a Node.js. Neustále kontroluje frontu událostí a přesouvá připravené úlohy do hlavního zásobníku volání. Díky tomu mohou aplikace efektivně reagovat na uživatelské akce, síťové požadavky i další události bez nutnosti vytváření velkého množství vláken.