جاوااسکریپت چجوری کار میکنه؟

JS Execution Model

قسمت دوم :

Web api
Macrotasks & Microtasks
Callback queue
Event Loop

توی قسمت قبل با Call Stack آشنا
شدیم و فهمیدیم که توابع در
جاوااسکریپت برای اجرای درون پشته ای
(LIFO) قرار میگیرن و هنگام اجرا به
ترتیب از بالای پشته خارج می‌شن(تابعی
که اول از همه وارد شده آخر خارج می‌شه)

حالا اگه تابع ما async بود یعنی نیاز به زمان بیشتری
داشت برای اجرا چی؟(مثل promiseها یا
setTimeout)

اول بیاین با انواع تابع async آشنا بشیم
خب به طور کلی ما دو دسته تابع
ناهمزمان یا async داریم:

- دسته اول : Micro Tasks

که شامل
-Promise.then() / catch() / finally()
-queueMicrotask()
-MutationObserver

- دسته دوم : Macro Tasks

که شامل
-setTimeout
-setInterval
-setImmediate (در Node.js)
-DOM (click, load, scroll, ...) رویدادهای
-requestAnimationFrame

خب بیاین تصور کنیم ما یه کد از چند تابع sync و یه microtask و یه macrotask داریم

```
1  console.log("sync1") // synchronous code
2  setTimeout(() => console.log("timeout"), 0); //macro task
3  Promise.resolve().then(() => console.log("promise")); // micro task
4  console.log("sync2"); // synchronous code
```

خروجی ما به شکل زیر هست:

```
1  Output:
2  sync1
3  sync2
4  promise
5  timeout
```

بیاین بیشتر بررسیش کنیم

اول از همه `console.log("sync1")` وارد call stack می‌شه و بعد از اجرا از اون خارج میشه .

بعد از اون `setTimeout(() => console.log("timeout"), 1000)` وارد کال استک شده و چون یه تابع async و macrotask هست و نیاز به زمان داره به web api سپرده می‌شه و از callstack خارج می‌شه و بعد از ۱۰۰۰ میلی ثانیه(عددی که به timer داده شده) به macrotask queue (که بهش Callback Queue هم می‌گن) میره

```
Promise.resolve().then(() => console.log("promise"));
```

تابع وارد کال استک شده و چون یه تابع async و ایندفعه microtask هست و برای اجرا نیاز به زمان داره مثل تابع قبلی از callstack خارج می‌شه وبعد از resolve به microtask queue میره

```
console.log("sync2");
```

در آخر هم وارد callstack شده و بلافاصله خارج می‌شه و چاپ می‌شه

حالا ما توی macrotask queue (یا همون callback queue) یک تسک داریم و توی microtask queue هم یک تسک دیگه.
اینجاست که event loop وارد عمل می‌شه: به محض اینکه call stack خالی بشه، event loop اول می‌ره سراغ microtask queue و اولین تسک اون رو به call stack اضافه می‌کنه.
بعد از اینکه تمام تسک‌های موجود در microtask queue اجرا شدند، event loop به سراغ macrotask queue می‌ره و تسک‌های اون رو یکی‌یکی به call stack اضافه می‌کنه.

اولویت microtaskها از macrotaskها بیشتره، و تا زمانی که microtask queue کاملاً خالی نشه، event loop سراغ macrotaskها نمی‌ره.

امیدوارم مفید بوده باشه برات 🙂

مرسی که تا اینجا اومدی ❤️

Mohammad Amrollahi

JavaScript | ReactJs | NextJs