자바스크립트의 함수 작동원리

싱글스레드와 비동기 처리

https://github.com/jio-ping

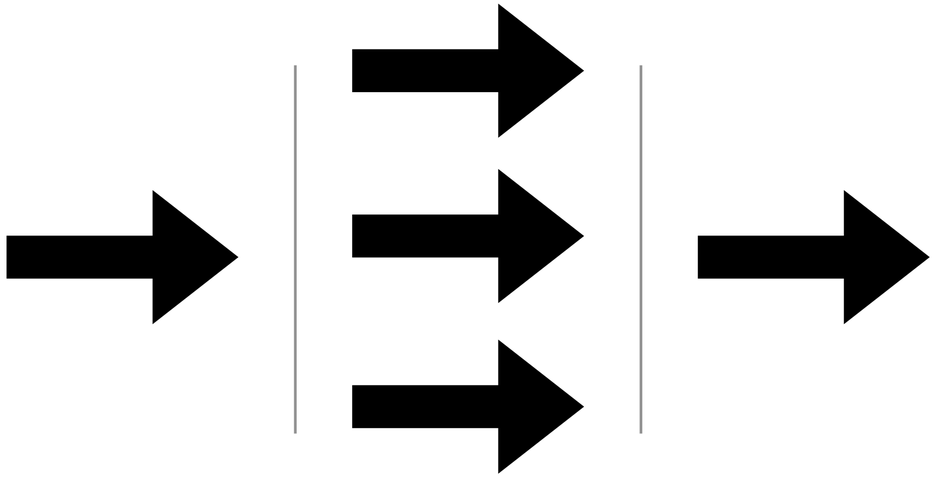
목차

싱글 스레드 vs 멀티 스레드

thread: 프로그램의 처리흐름

Single Thread

프로그램 한 개의 처리흐름으로 프로그램을 순차적으로 실행



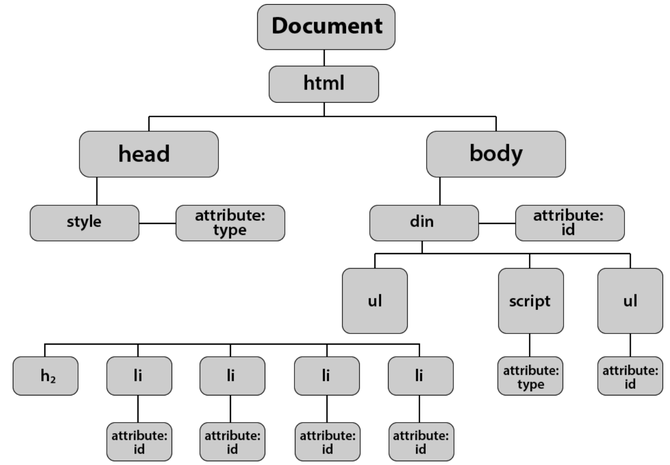
Multi Thread

프로그램 여러 개의 처리 흐름으로 동시에 작업을 여러 개 병렬로 실행

싱글스레드인 자바스크립트

왜 싱글스레드를 채택했을까?

웹페이지의 보조적 기능을 위한 경량적 언어로써
멀티스레드 환경의 동시성 문제 방지



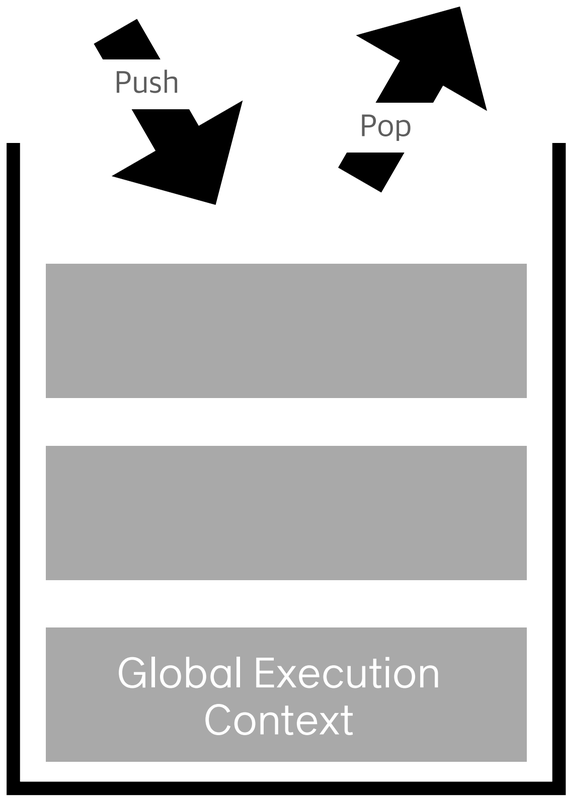
DOM 조작과 UI 업데이트의 일관성 유지

webworkers 를 사용하면 특정 작업을 백그라운드에 있는 다른 스레드에서 실행(멀티 스레드 처리) 할수 있음

https://github.com/jio-ping

함수 호출과 실행을 제어하는 콜스택

실행문맥(함수 호출)을 관리하는 구조 ➡️ stack



push : 스택의 가장 윗 부분에 데이터를 쌓는 행위
pop: 스택의 가장 윗 부분의 데이터를 빼내는 행위

stack

데이터를 아래서부터 쌓아 올려 마지막으로 추가한 데이터를 먼저 꺼내는
후입선출(LIFO, Last In First Out) 방식으로 관리

callstack

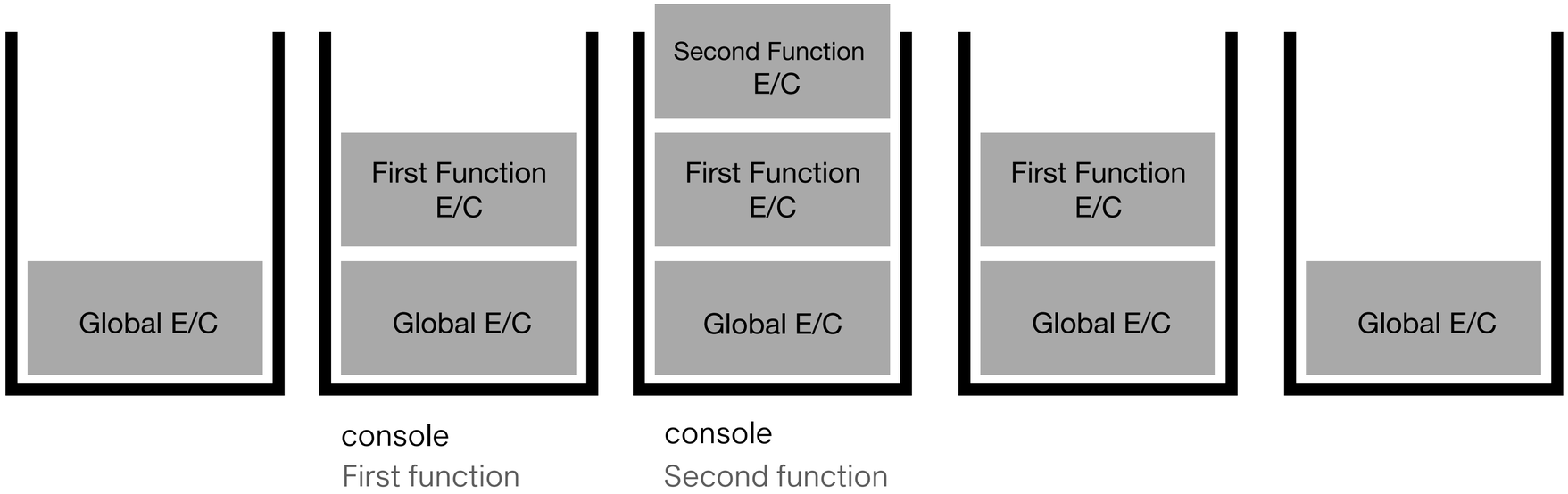
스택 자료 구조를 활용해 코드가 실행되며 생성되는 실행컨텍스트를 저장해
함수 호출을 관리하는 자바스크립트 엔진의 구성요소

콜스택의 작동 과정

Global Execution Context 는 가장 기본적인 실행환경으로 전역객체와 연결됨

브라우저 환경: Window / Node.js: global

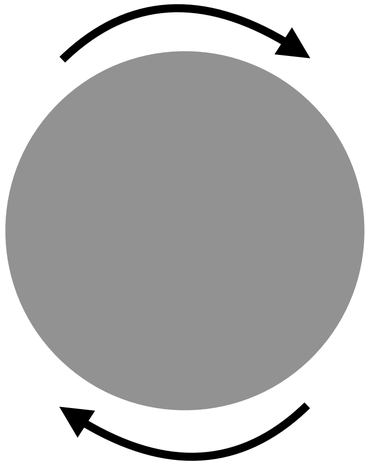
```
function first() {
  console.log("First function");
  second(); // second 함수 호출
}

function second() {
  console.log("Second function");
}

first(); // first 함수 호출
```



https://github.com/jio-ping

이벤트 루프와 태스크 큐

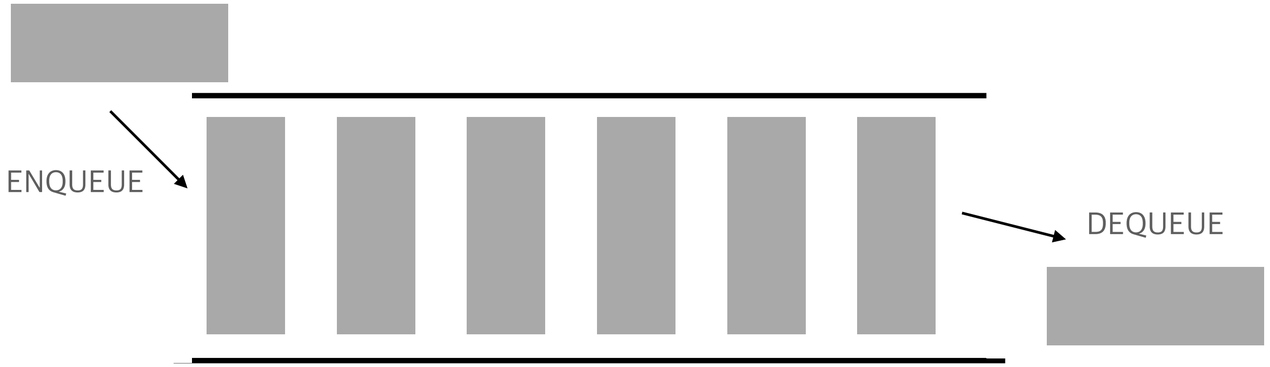
싱글 스레드임에도 비동기 작업을 실행할 수 있는 이유



Event Loop

콜스택과 태스크큐를 관리해 비동기 작업 처리

콜스택이 비어있는지 확인 후,
태스크 큐에서 대기중인 콜백을 가져와 실행



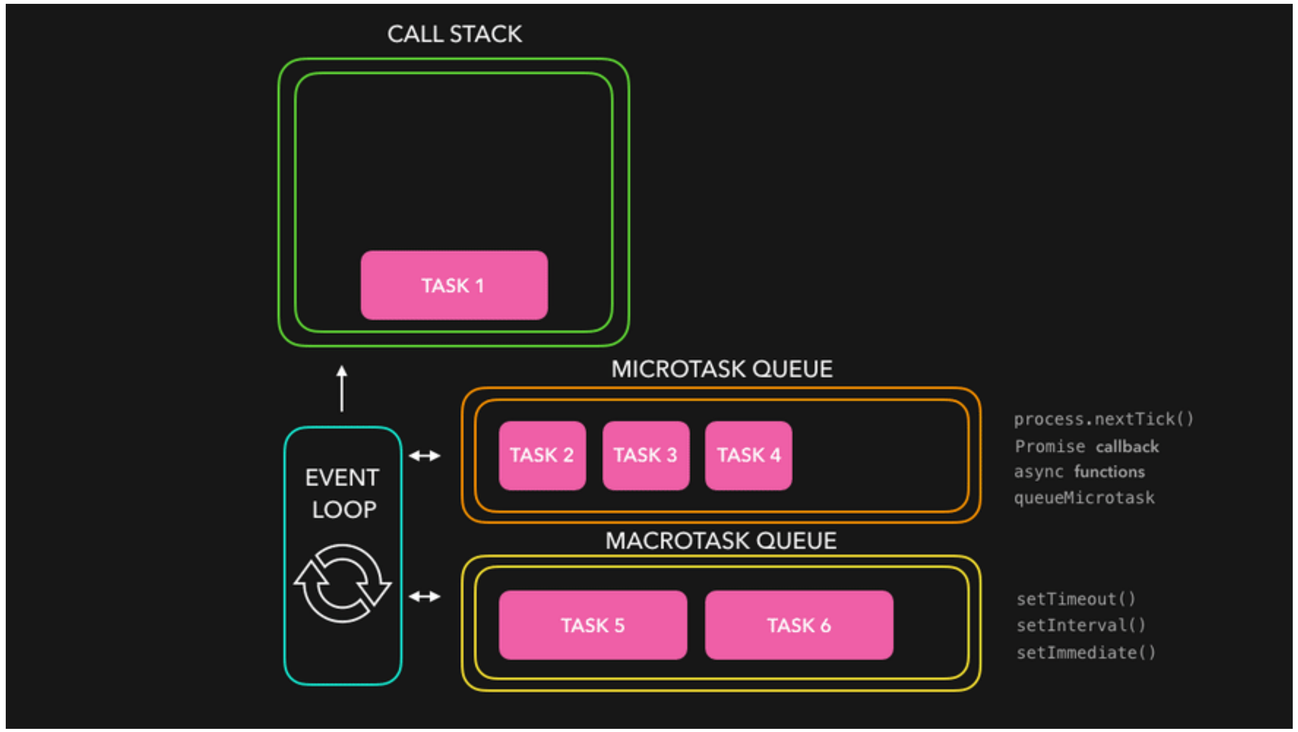
Task Queue

실행 대기 중인 콜백함수가 보관되는 큐

MicroTaskQueue와 MacroTaskQueue로 분류

MicroTaskQueue vs MacroTaskQueue

콜백 함수 보관소 TaskQueue



Micro Task Queue (=Job Queue)

현재 작업이 끝나면 (콜스택이 비면) 바로 실행되어야 하는 작업

Promise.then, MutationObserver, queueMicroTask

Macro Task Queue (=event/callback Queue)

비교적 큰 단위의 작업으로 MicroTaskQueue 작업이 모두 끝난 후 실행

setTimeout, setInterval, UI렌더링 작업, 이벤트 핸들러, setImmediate(node.js)

https://github.com/jio-ping

브라우저에서의 비동기 함수 작동과정

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

// 메모리 할당 (Heap)
const user = { name: "Alice", age: 25 };

// Web API에 등록되는 Timer (Macrotask)
setTimeout(() => {
  console.log("Macrotask: setTimeout");
}, 0);

// Promise (Microtask)
Promise.resolve().then(() => {
  console.log("Microtask: Promise.then");
});

// Web API에 등록되는 Event Listener (Macrotask)
document.addEventListener("click", () => {
  console.log("Macrotask: Event Listener");
});

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

Output)
Start
End
Microtask: Promise.then
Macrotask: setTimeout

1, console.log(“Start”)
callstack에 올라간 뒤 “Start” 출력

2. const user = { name: "Alice", age: 25 }
heap에 저장, user변수는 저장된 객체를 참조

3. console.log(“End”)
callstack에 올라간 뒤 “End” 출력

4. setTimeout
setTimeout( ) callstack에 올라간 뒤 Web API로 넘겨짐
지정 시간이 지나면 Macro TaskQueue에 콜백함수 등록

5. Promise.resolve( ).then( )
Promise.resolve( ).then( )이 callstack에 올라감
즉시 MicroTaskQueue에 then의 콜백 등록

6. document.addEventListener( )
addEventListener( ) callstack에 올라감
WebAPI로 전달돼 리스너가 등록 됨

7. Micro Task Queue 실행
Promise.then의 콜백함수 callstack에 올라감 “Microtask: Promise.then”출력

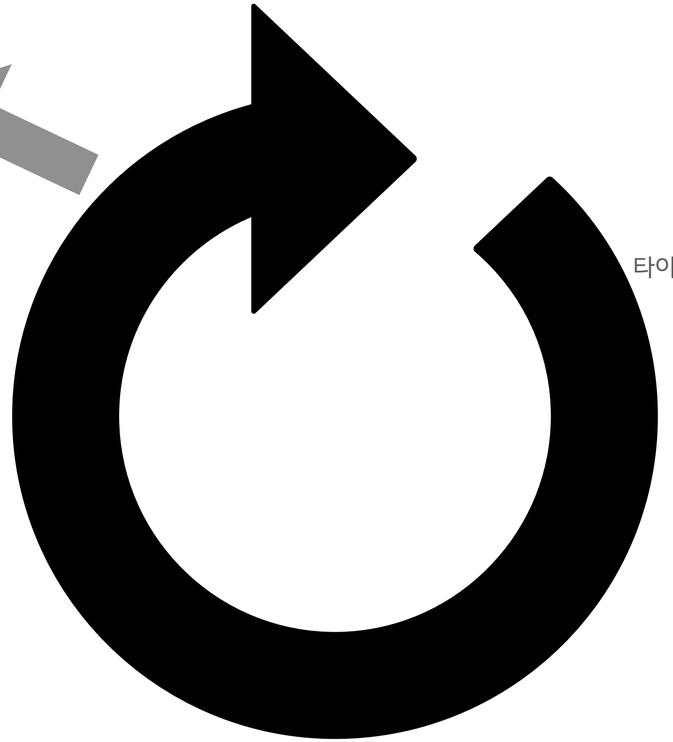
8.Macro Task Queue 실행
setTimeout의 콜백함수 callstack에 올라감 “Macrotask: setTimeout” 출력

9. 클릭이벤트 발생시
브라우저의 EventListener(WebAPI)에서 이벤트 감지
이벤트 콜백함수를 MacroTaskQueue에 등록 후 callstack이 비어있는 시점에 실행

https://github.com/jio-ping

Node.js의 이벤트 루프

브라우저와 달리 libuv라는 라이브러리에 의해 구현



refs ==0

Process.exit

내부적으로 열린 이벤트 리스너를 추적해 Reference counting함
이벤트 리스너가 필요없어지거나 콜백이 완료되면 줄어듦

Timer

setTimeout,setInterval의 콜백 처리

타이머와 연관된 작업이 있으면
Timer phase로 이동

Pending callbacks

I/O 작업에서 발생한 콜백을 처리

실행 못하면
pending callbacks등록

Poll phase

새로운 I/O 작업과 준비된 I/O작업의 콜백 처리

PollPhase의 실행할 콜백이 없는 상태에서 타이머가 대기 시간이 만료되었을 때,
해당 타이머의 콜백이 즉시 실행될 수 있음

Check

setImmediate 콜백 처리

setImmediate는 반드시 열린 콜백이 모두 실행된 다음 실행

Close Callbacks

Close 이벤트 처리 (ex. socket.on(“close”)

https://github.com/jio-ping

Reference

마이크로 태스크 https://ko.javascript.info/microtask-queue
Inpa - 자바스크립트 이벤트 루프 동작구조 원리
Understading the Javascript EventLoop
MDN Javascript의 queueMicrotask()와 함께 마이크로태스크 사용하기
모던자바스크립트 입문 - 길벗

https://github.com/jio-ping