8장 비동기 프로그래밍, 동시성과 병렬성

8.1 JS 이벤트 루프

```
setTimeout(()=>{console.log('a')},0)
setTimeout(()=>{console.log('b')},0)
console.log('c')

// c a b
```

JS runtime

콜스택 (LIFO)

task queue

8.2 콜백 사용하기

인자로 함수 전달 (함수 실행의 책임을 위임)

비동기 함수의 에러 발생시에 대한타입을 처리해주지않는다.

```
export function readFile(
        path: PathOrFileDescriptor,
        options:
            | ({
                  encoding?: null | undefined;
                  flag?: string | undefined;
              } & Abortable)
            | undefined
            | null,
        callback: (err: NodeJS.ErrnoException | null, data: Buffer) => void
    ): void;


function __promisify__(
            path: PathOrFileDescriptor,
            options?: {
                encoding?: null | undefined;
                flag?: string | undefined;
            } | null
        ): Promise<Buffer>;

readFile.__promisify__();
```

8.3 Promise

```
/**
 * Represents the completion of an asynchronous operation
 */
interface Promise<T> {
    /**
     * Attaches callbacks for the resolution and/or rejection of the Promise.
     * @param onfulfilled The callback to execute when the Promise is resolved.
     * @param onrejected The callback to execute when the Promise is rejected.
     * @returns A Promise for the completion of which ever callback is executed.
     */
    then<TResult1 = T, TResult2 = never>(onfulfilled?: ((value: T) => TResult1 | PromiseLike<TResu

    /**
     * Attaches a callback for only the rejection of the Promise.
     * @param onrejected The callback to execute when the Promise is rejected.
     * @returns A Promise for the completion of the callback.
     */
    catch<TResult = never>(onrejected?: ((reason: any) => TResult | PromiseLike<TResult>) | undefi
}
```

8.4 Async Await

```
try {
    const data = await readFile.__promisify__('/');
  } catch (error) {
    console.error(error);
  }
```

8.5 비동기 스트림

```
interface WindowEventHandlersEventMap {
    "afterprint": Event;
    "beforeprint": Event;
    "beforeunload": BeforeUnloadEvent;
    "gamepadconnected": GamepadEvent;
    "gamepaddisconnected": GamepadEvent;
    "hashchange": HashChangeEvent;
    "languagechange": Event;
    "message": MessageEvent;
    "messageerror": MessageEvent;
    "offline": Event;
    "online": Event;
    "pagehide": PageTransitionEvent;
    "pageshow": PageTransitionEvent;
    "popstate": PopStateEvent;
    "rejectionhandled": PromiseRejectionEvent;
    "storage": StorageEvent;
    "unhandledrejection": PromiseRejectionEvent;
    "unload": Event;
}

interface Window ... {
        addEventListener<K extends keyof WindowEventHandlersEventMap>(type: K, listener: (this: Wi
}
```

8.6 타입 안전 멀티 스레딩

워커, 메인 스레드간 통신

```
interface Worker ... {
        postMessage(message: any, transfer: Transferable[]): void;
}

type Commands = {
        sendMessageToThrea: [ThreadId, Message]
}

type Events = {
        receiveMessage: [ThreadId, UserId, Message]
}

class SafeEmiter<T> {

        emit<K extends keyof T>(...)

        on<K extends keyof T>(...)

}
```