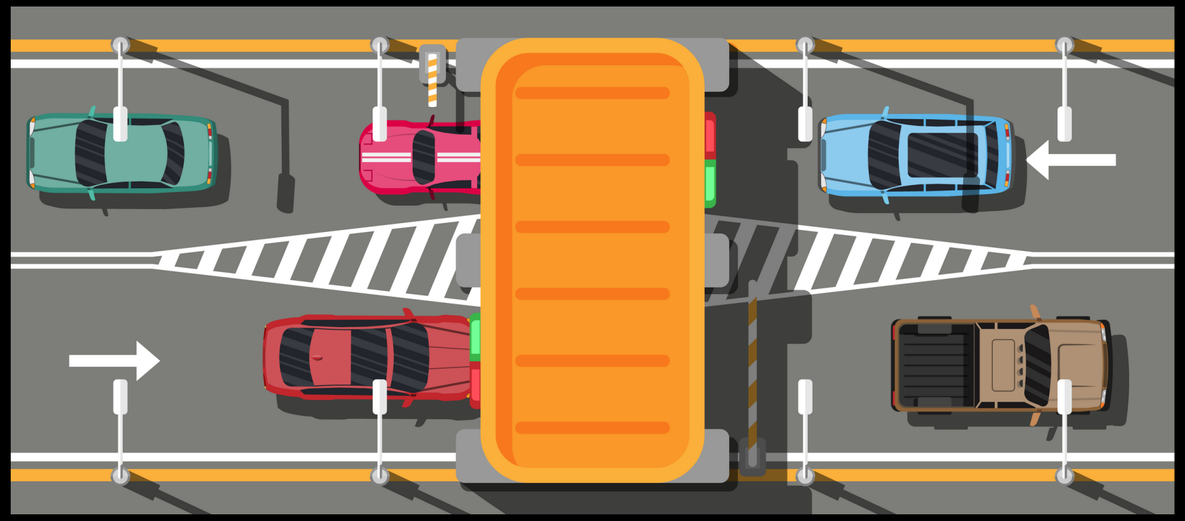
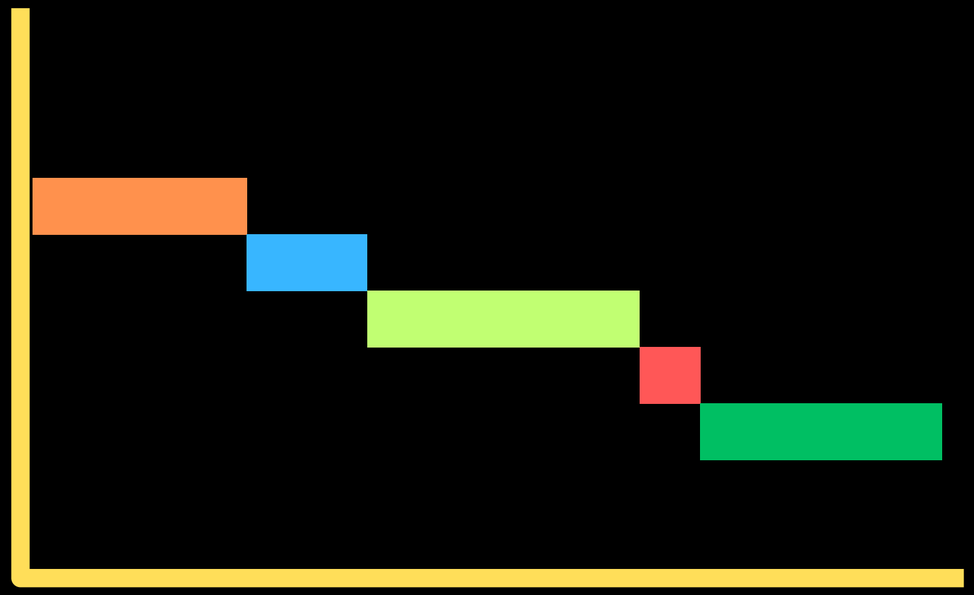
진짜! 모두를 위한
자바스크립트

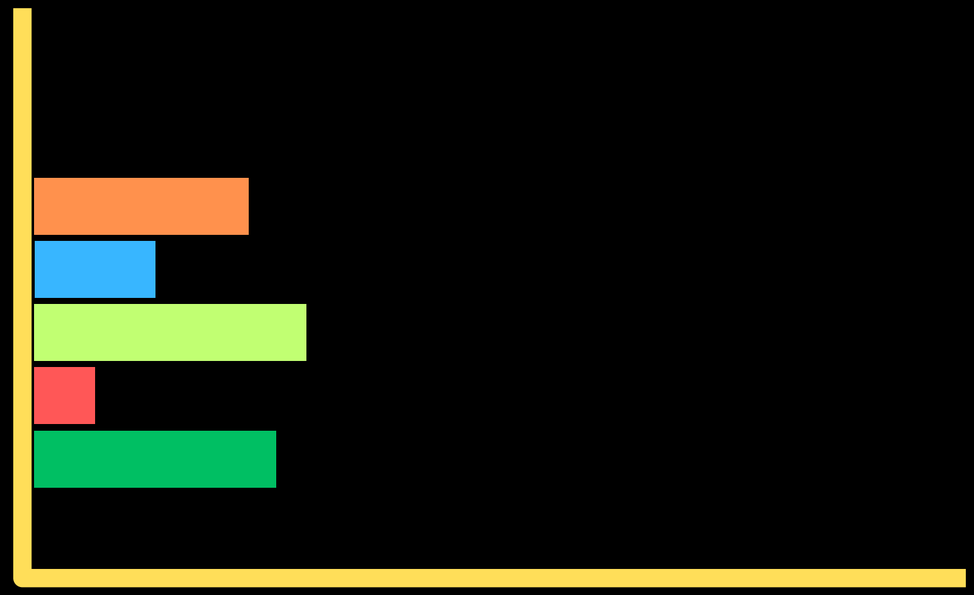
비동기 프로그래밍



동기와 비동기
(Synchronous & Asynchronous)



동기(순차적 실행)

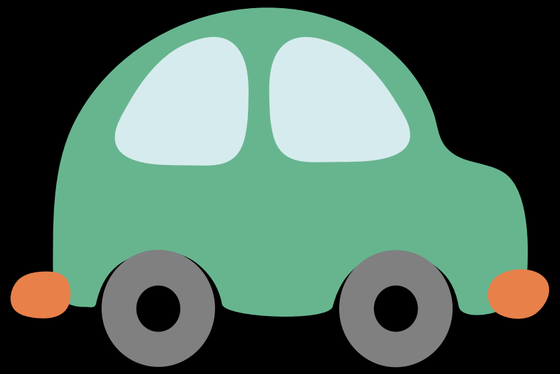


비동기(비순차적 실행)

동기
(Synchronous)

비동기
(Asynchronous)

점유와 Blocking

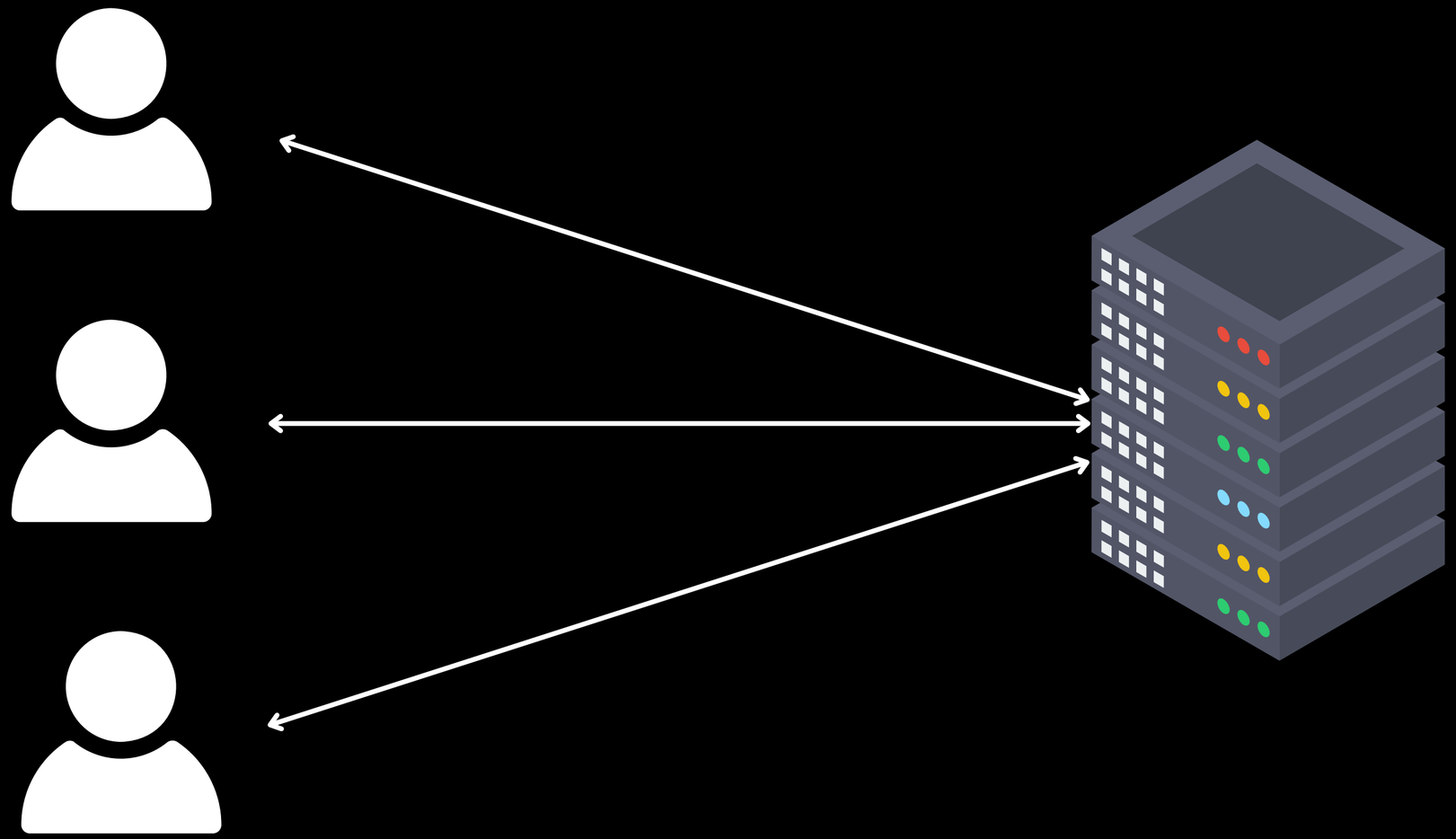


검은색 자동차가 요금 계산기를 점유하고 있다.

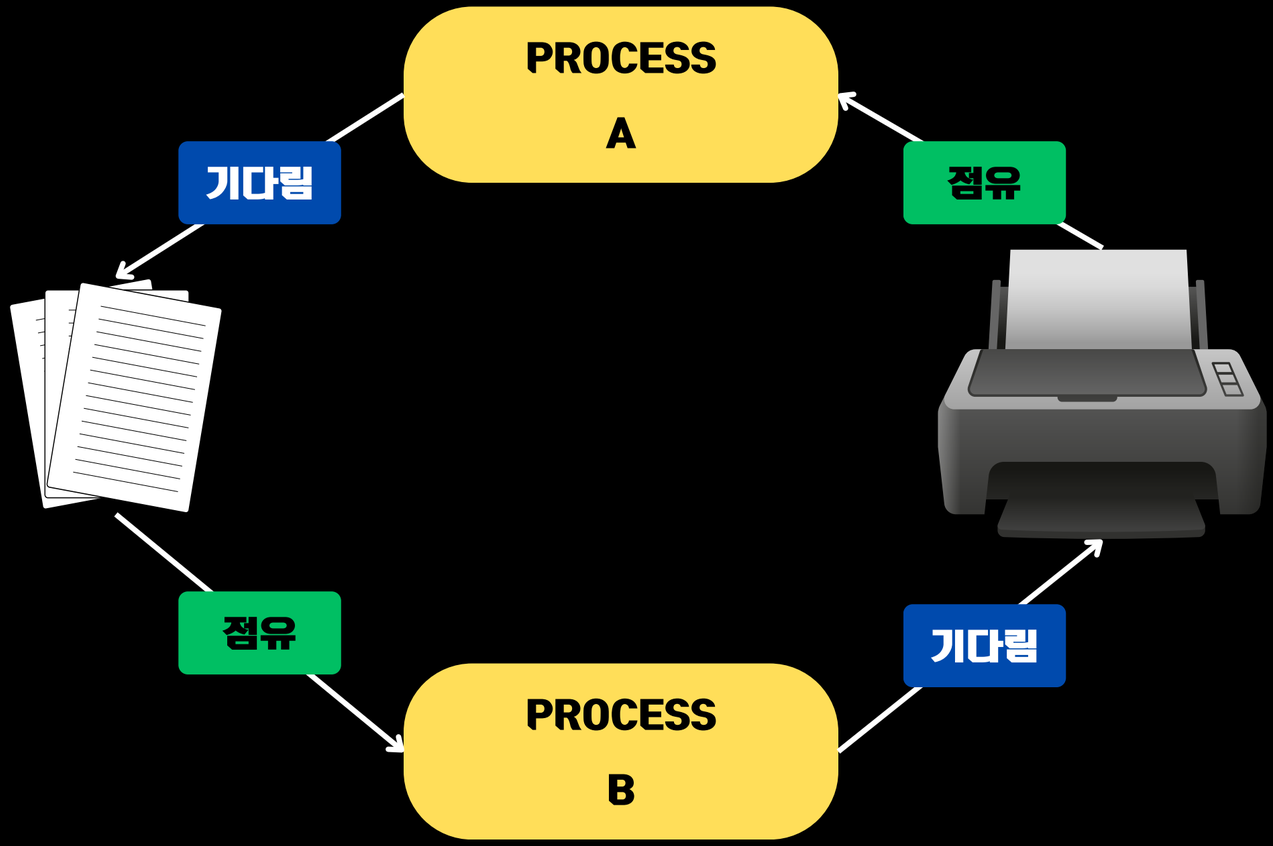
검은색 자동차가 요금 계산기을 마칠 때까지
뒷 차들은 Blocking 된다.

컴퓨터 자원의 제어권을 가진(점유한) 코드가 끝날 때까지
다른 코드는 실행이 Blocking 된다.

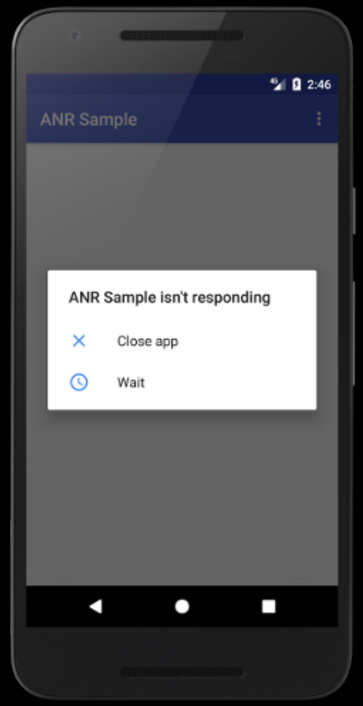
Blocking은 언제 문제가 될까?



Blocking은 언제 문제가 될까?



Blocking은 언제 문제가 될까?



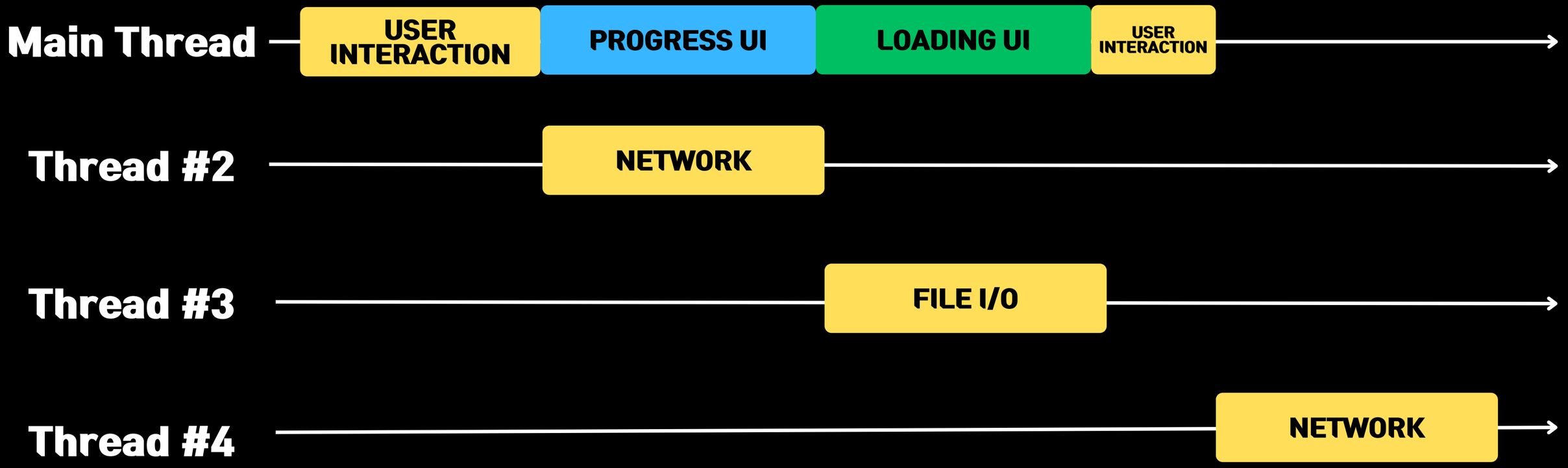
Blocking은 왜 문제가 될까?

사용자 입장
응답성 저하

시스템 입장
자원 활용도 저하

비동기 프로그래밍 

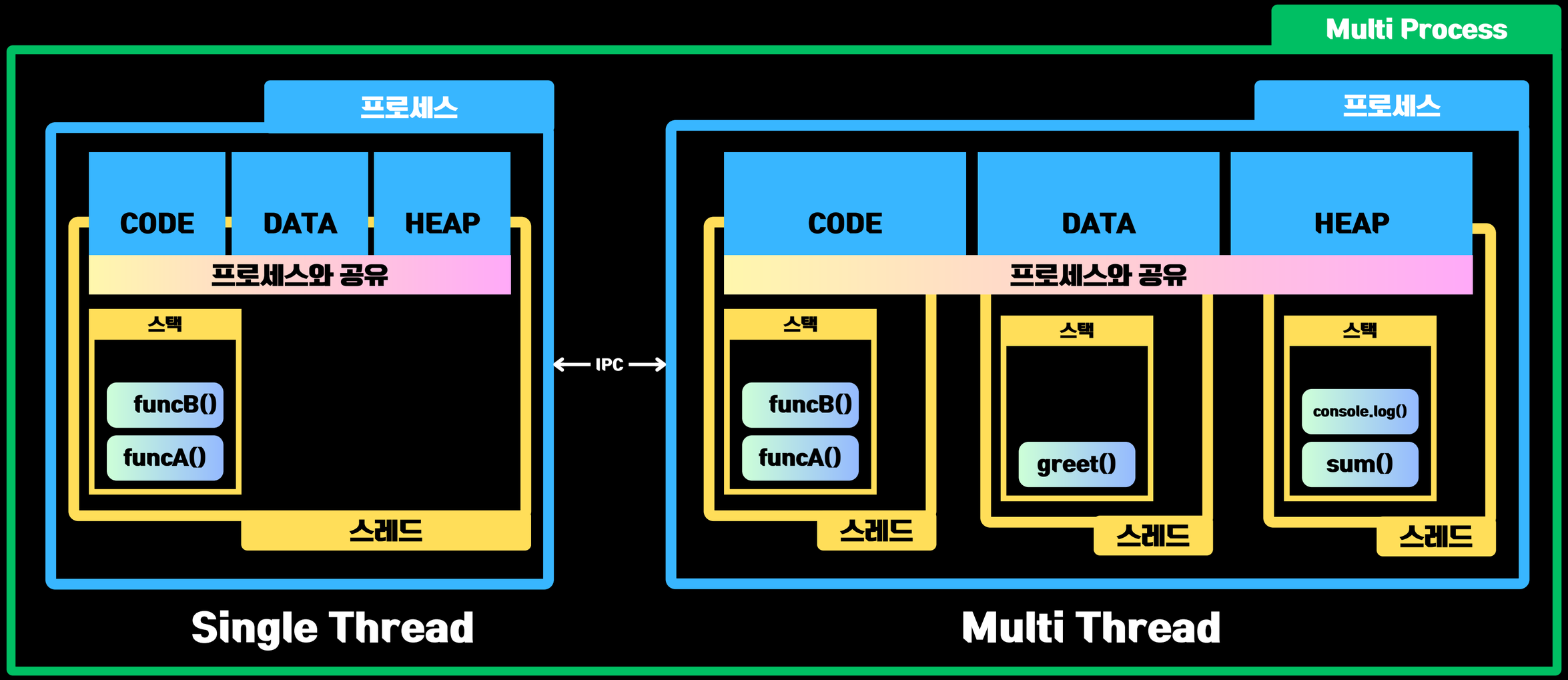
어떻게 Blocking 의 문제점을 해결할 수 있을까?



자바스크립트는 Single Thread

Main Thread ⟶

자바스크립트는 Single Thread

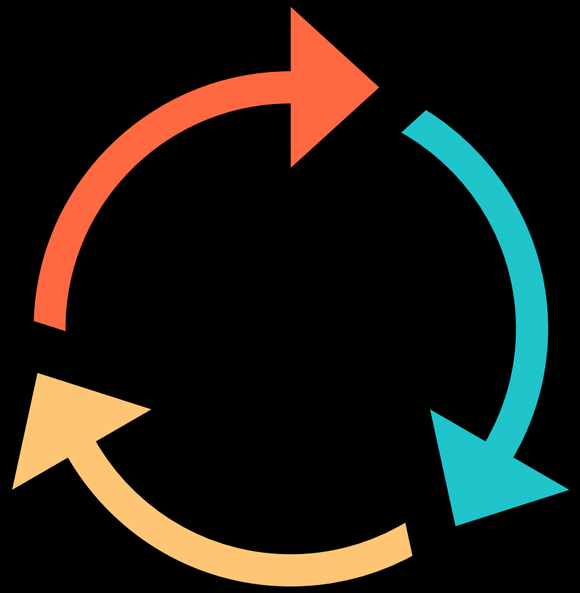


비동기 프로그래밍

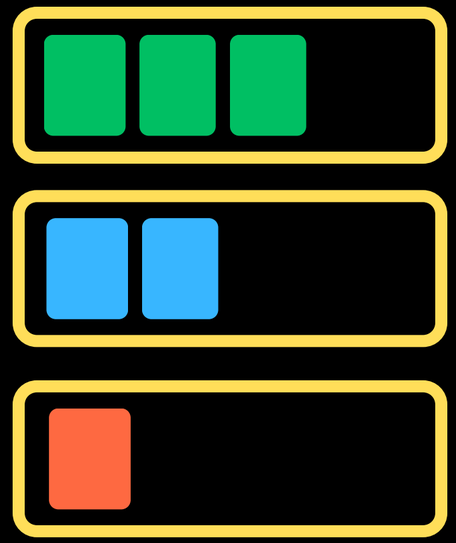
진짜! 모두를 위한
자바스크립트

자바스크립트는 Single Thread

Main Thread ⟶



Event Loop



Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

자바스크립트 Event Loop만 Single Thread

SINGLE THREAD

API
setTimeout
Promise
...

WebAPIs

Qeues

Event Loop

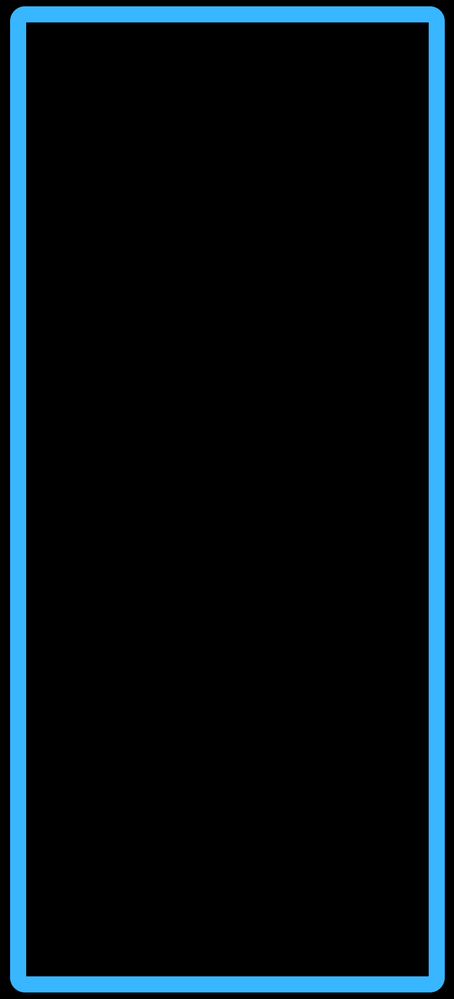
Thread Pool
T1
T2
T3
T4
NETWORK
FILESYSTEM
PROCESS
OTHER

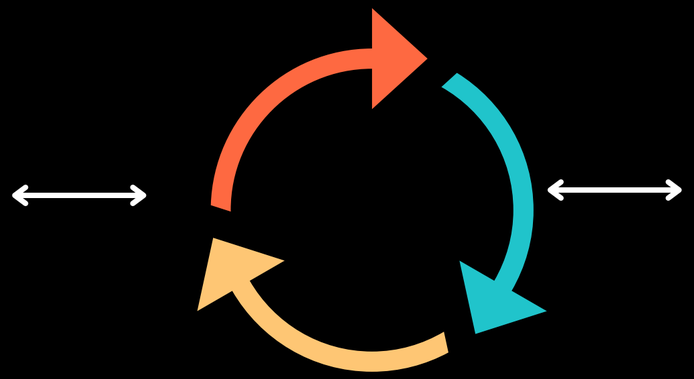
MULTI THREAD

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack





Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

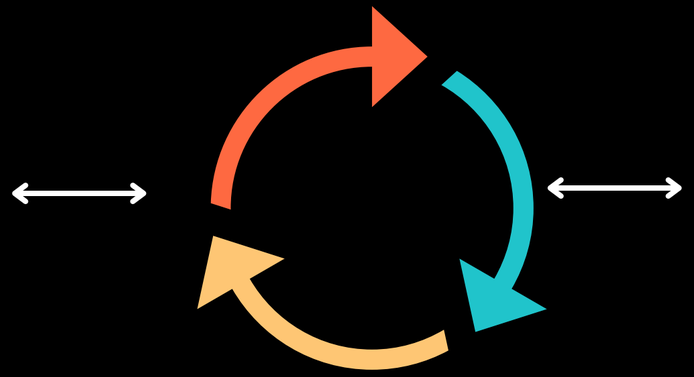
Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

```
setTimeout

console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

```
console

start
```

console.log('start')

anonymous
(글로벌 실행 컨텍스트)

Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

setTimeout()

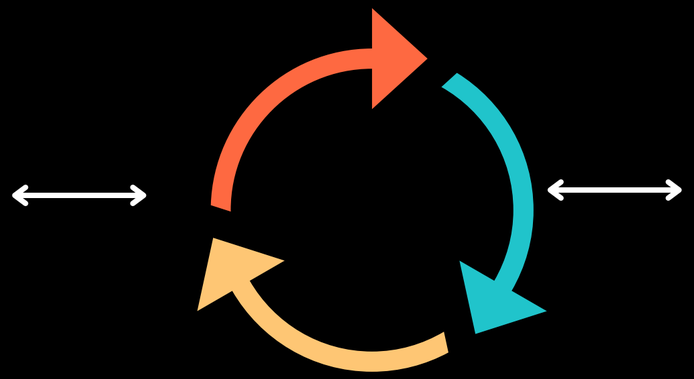
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

start



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

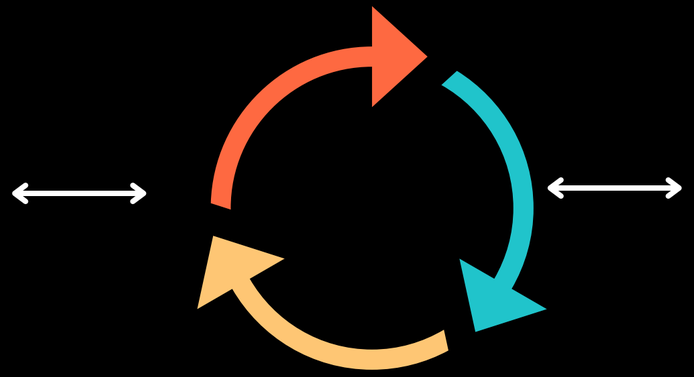
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

start



Event Loop

MACROTASK QUEUE

Call
back

MICROTASK QUEUE

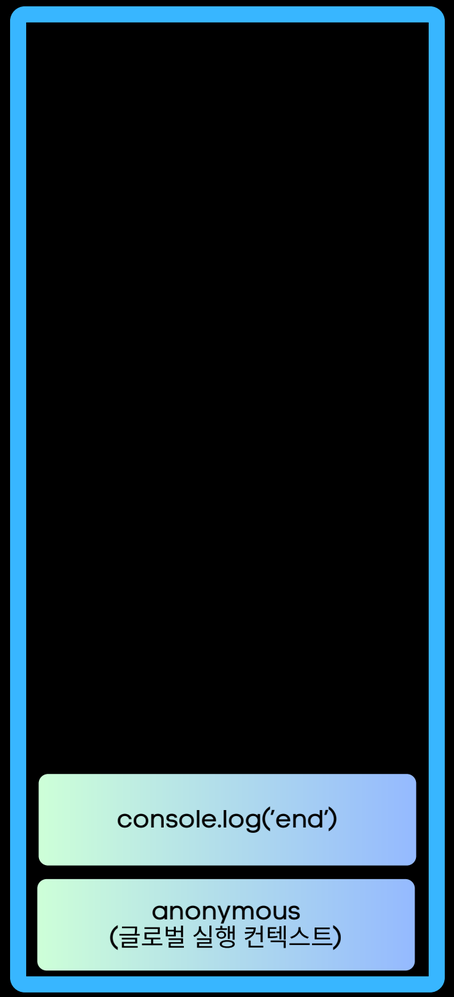
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

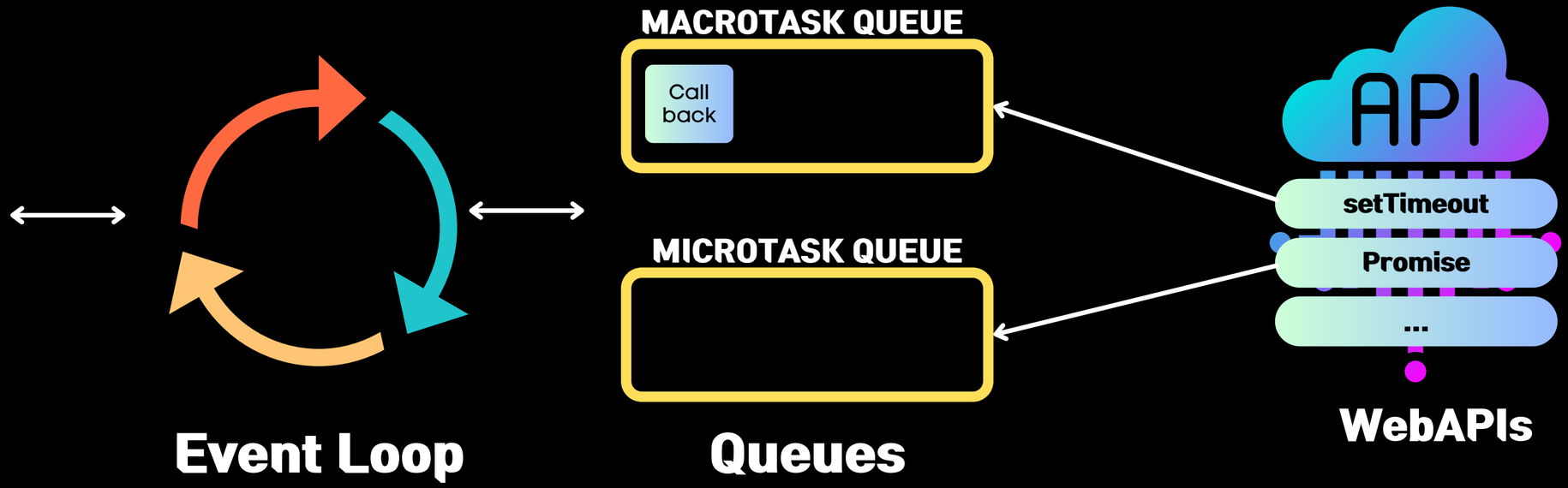


setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

```
start
end
```



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

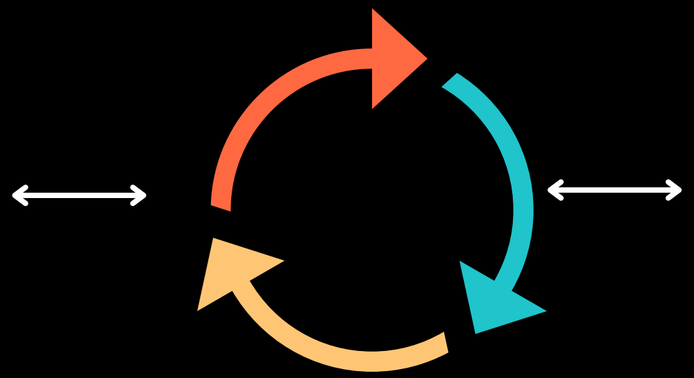
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

```
start
end
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

Callback
console.log('hello')

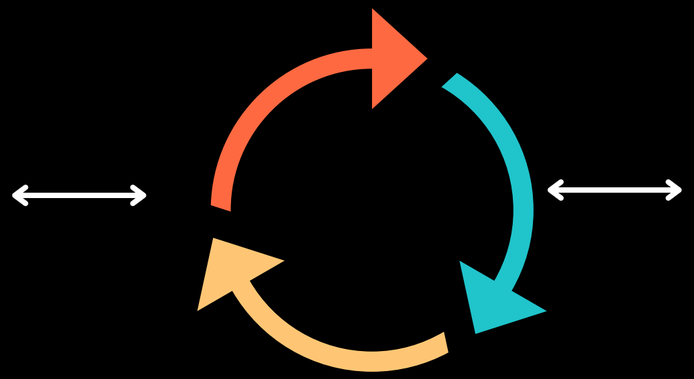
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

```
start
end
hello
```



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

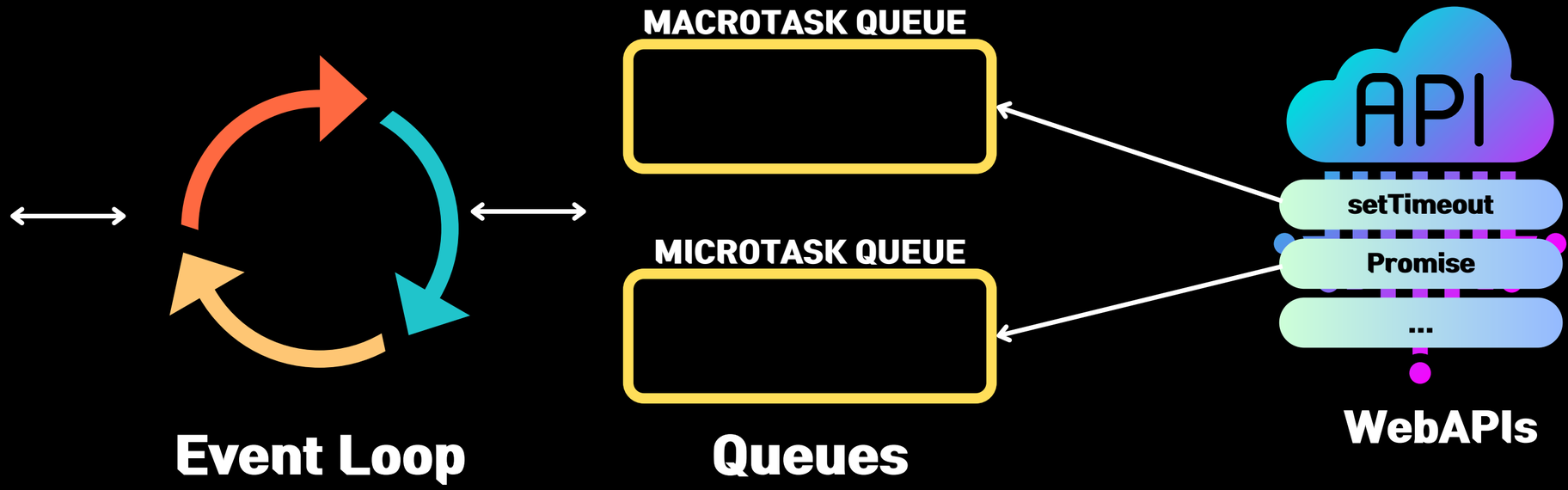
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

```
start
end
hello
```



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

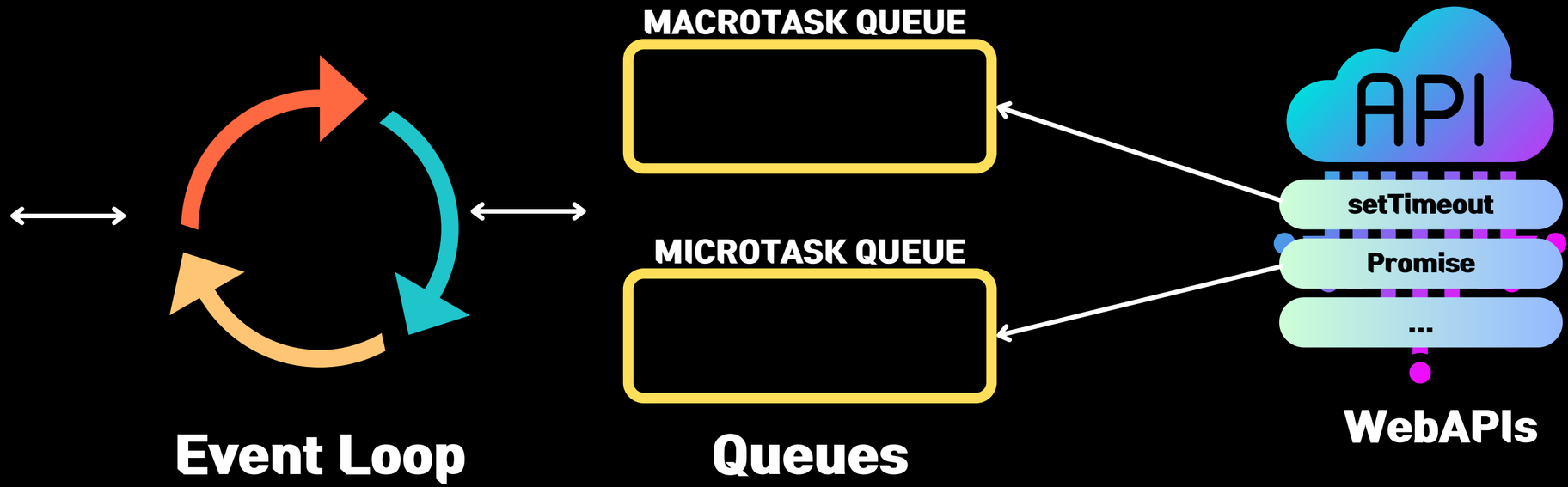
Main Thread의
Call Stack

setTimeout

```
console.log('start');
setTimeout(() => {
    console.log('Hello');
}, 0);
console.log('end');
```

console

```
start
end
hello
```



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

MACROTASK QUEUE

MICROTASK QUEUE

API

setTimeout

Promise

...

Event Loop

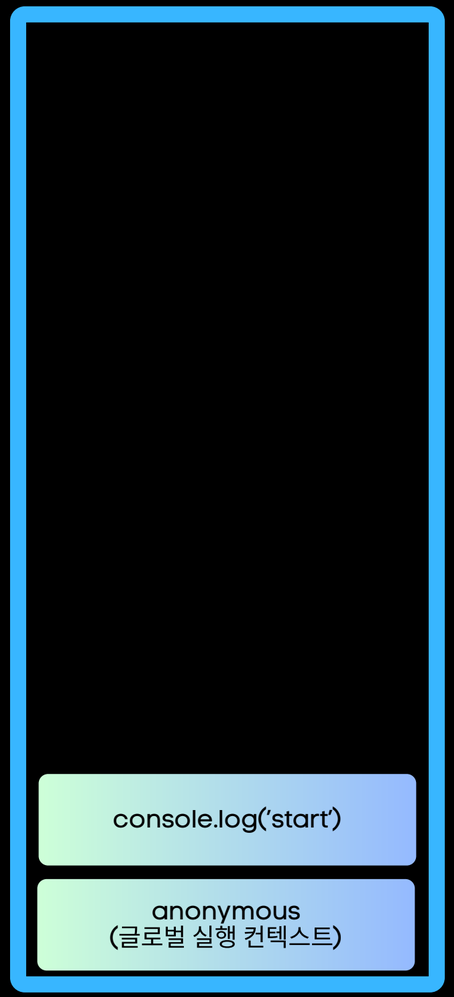
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

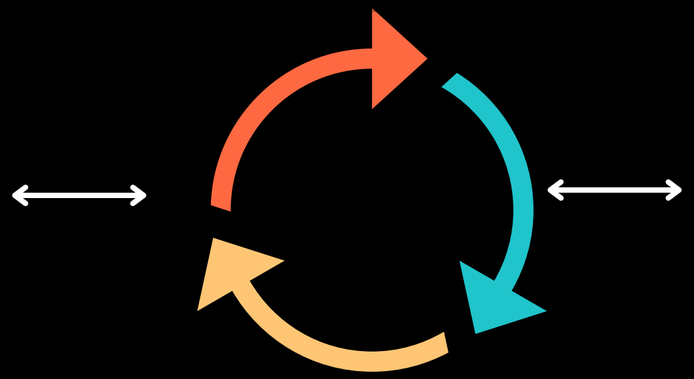


setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
```



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

setTimeout()

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
```

MACROTASK QUEUE

MICROTASK QUEUE

API

setTimeout

Promise

...

Event Loop

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

setTimeout()

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
```

MACROTASK QUEUE

MICROTASK QUEUE

API

setTimeout

Promise

...

Event Loop

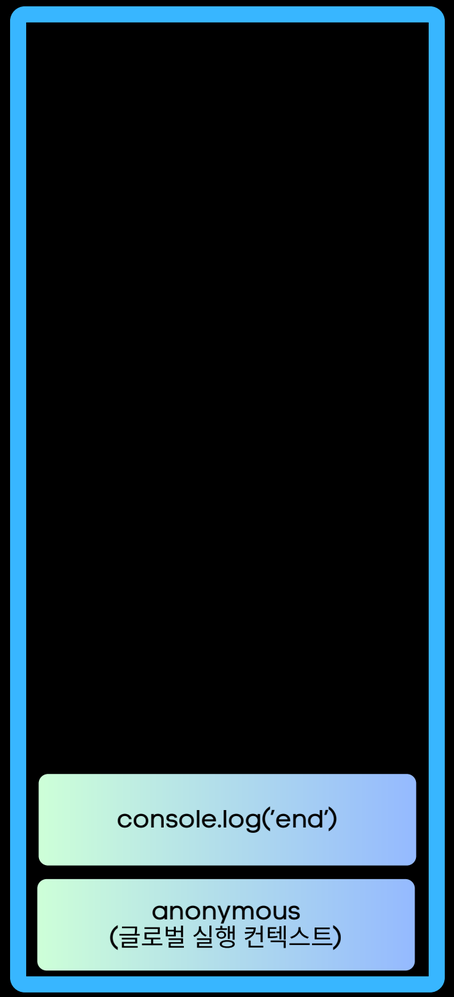
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

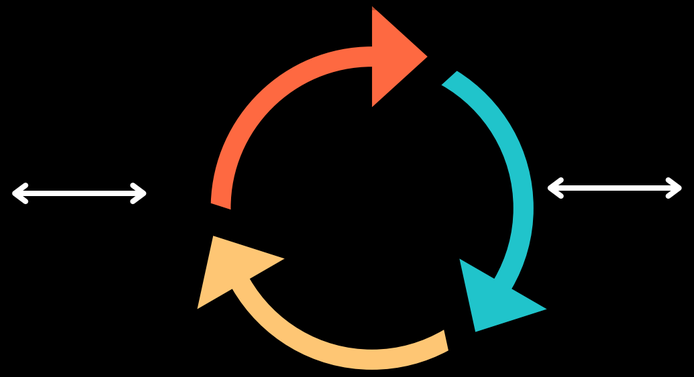


setTimeout

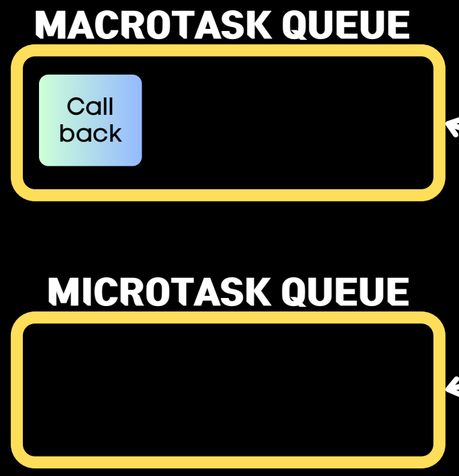
```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
```



Event Loop



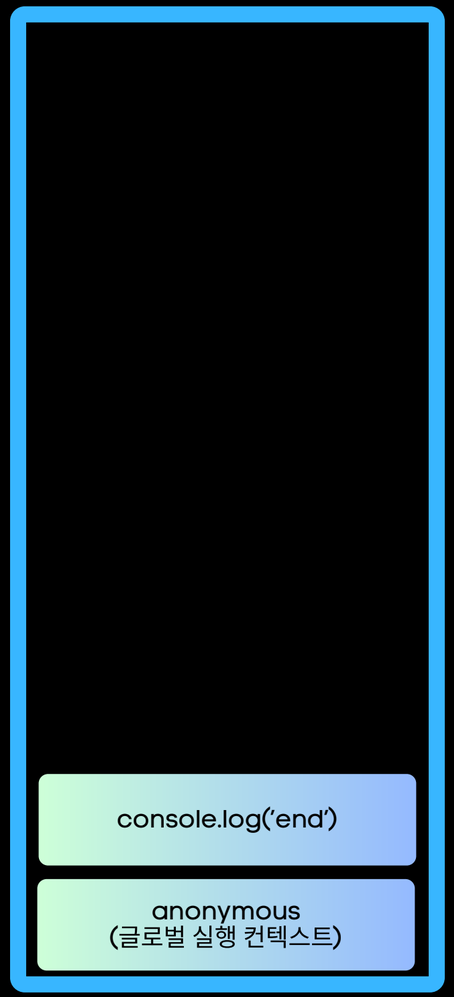
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

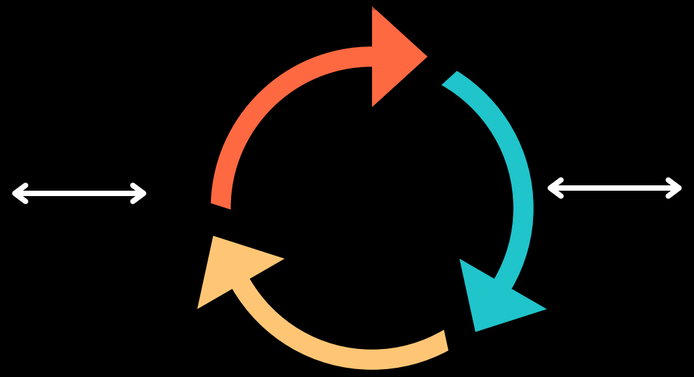


setTimeout

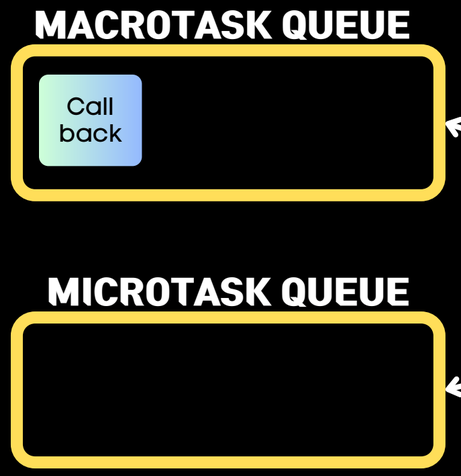
```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
01 end
```



Event Loop



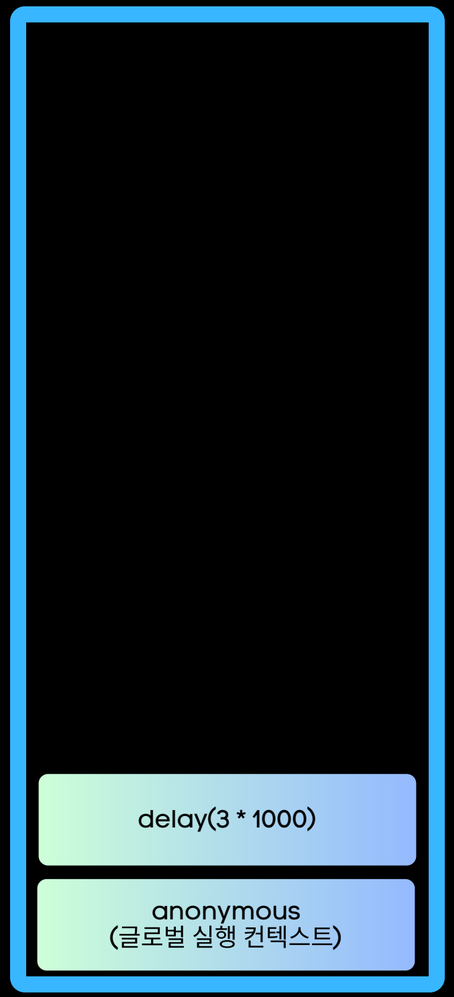
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

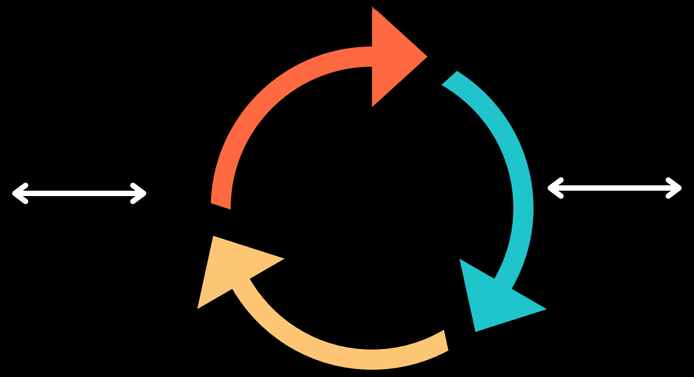


setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
01 end
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
01 end
3초 동안 delay했습니다.
```

Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

Callback
console.log('hello')

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
01 end
3초 동안 delay했습니다.
04 hello
```

Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

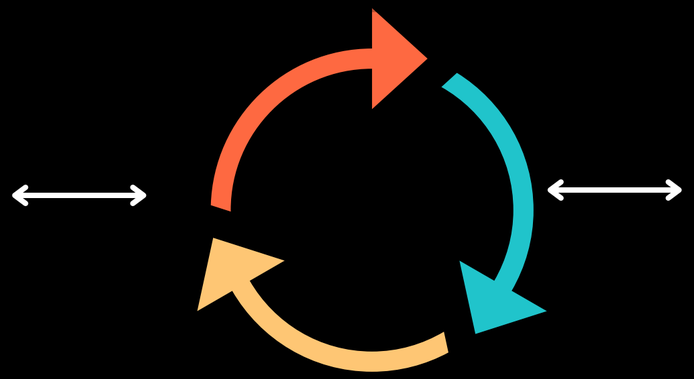
anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}동안 delay했습니다`);
}

console.log(`${new Date()} start`);
setTimeout(() => {
    console.log(`${new Date()} hello`);
}, 0);
console.log(`${new Date()} end`);
delay(3 * 1000);
```

console

```
01 start
01 end
3초 동안 delay했습니다.
04 hello
```



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

anonymous
(글로벌 실행 컨텍스트)

setTimeout

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}ms 동안 delay했습니다`);
}

setTimeout(() => {
    delay(1000);
}, 0);
console.log('Hello');

setTimeout(() => {
    delay(2000);
}, 0);
console.log('World');

setTimeout(() => {
    delay(3000);
}, 0);
console.log('This is Non Blocking');
```

console

MACROTASK QUEUE

MICROTASK QUEUE

API
setTimeout
Promise
...

Event Loop

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

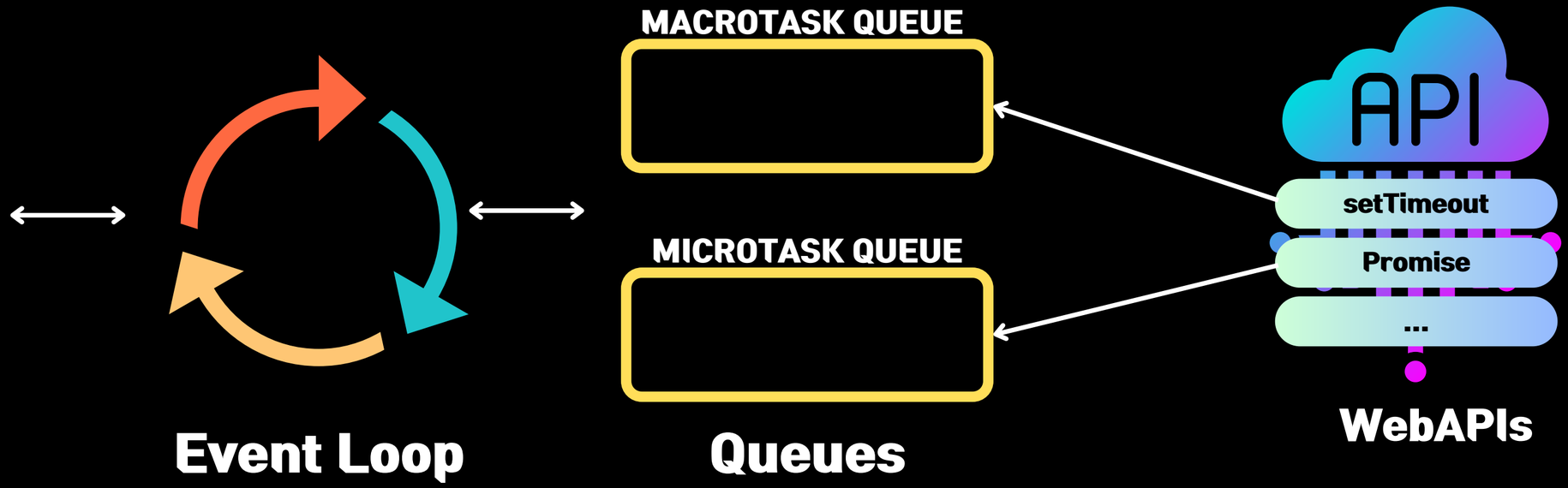
Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}ms 동안 delay했습니다`);
}

const blocking = new Promise((resolve, reject) => {
    delay(10 * 60 * 1000);
    resolve('finished');
});
blocking.then((value) => console.log(value));
```

console



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

delay(10분)

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}ms 동안 delay했습니다`);
}

const blocking = new Promise((resolve, reject) => {
    delay(10 * 60 * 1000);
    resolve('finished');
});
blocking.then((value) => console.log(value));
```

console

Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Promise's Executor

```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}ms 동안 delay했습니다`);
}

const blocking = new Promise((resolve, reject) => {
    delay(10 * 60 * 1000);
    resolve('finished');
});
blocking.then((value) => console.log(value));
```

console

Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Call back

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

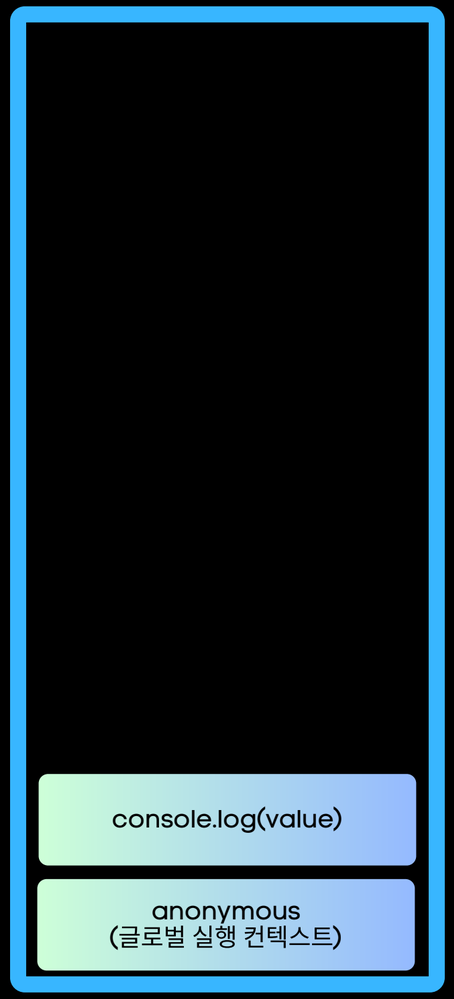
Promise's Executor

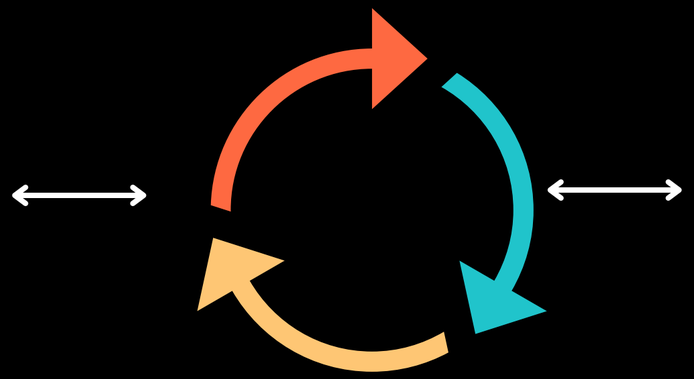
```
function delay(duration_ms) {
    const delayUntil = Date.now() + duration_ms;
    while(Date.now() < delayUntil) {
        ;
    }
    console.log(`${duration_ms}ms 동안 delay했습니다`);
}

const blocking = new Promise((resolve, reject) => {
    delay(10 * 60 * 1000);
    resolve('finished');
});
blocking.then((value) => console.log(value));
```

console

```
finished
```

Main Thread의
Call Stack





Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

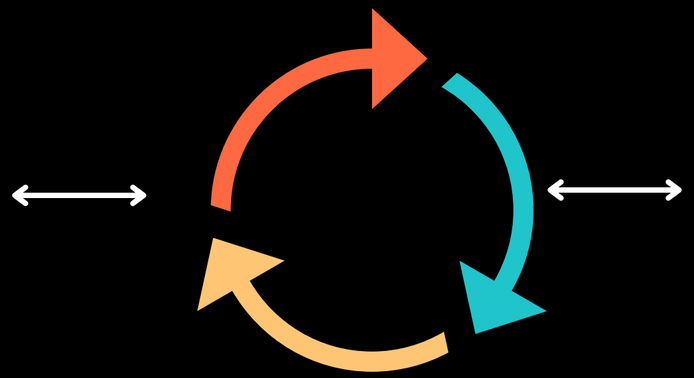
Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

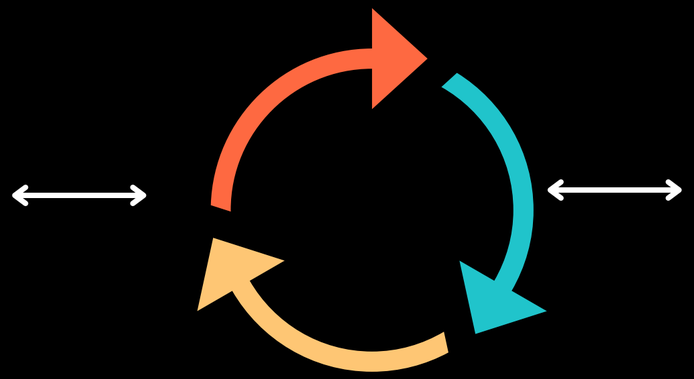
setTimeout()

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

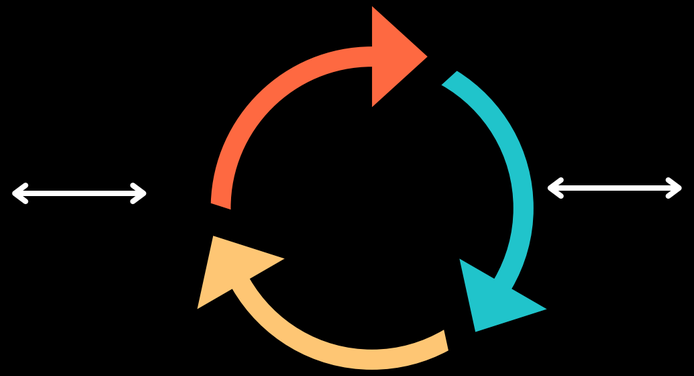
Main Thread의 Call Stack

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



MACROTASK QUEUE

MICROTASK QUEUE

T1
10분 대기

Event Loop

Queues

WebAPIs

EventLoop는 Blocking 되지 않았다.
UI 인터렉션이 가능하다.

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

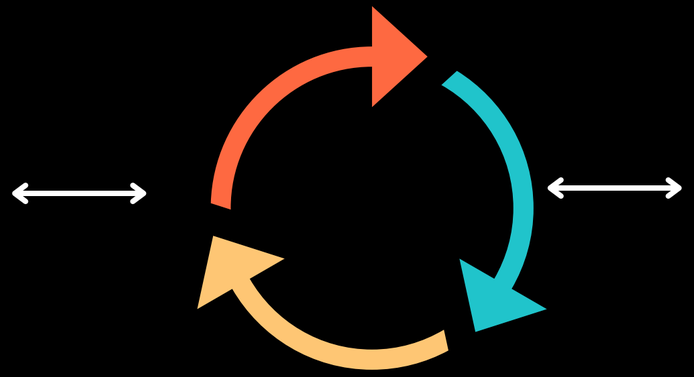
Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



Event Loop

MACROTASK QUEUE

Call
back

MICROTASK QUEUE

Queues

WebAPIs

EventLoop는 Blocking 되지 않았다.
UI 인터렉션이 가능하다.

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

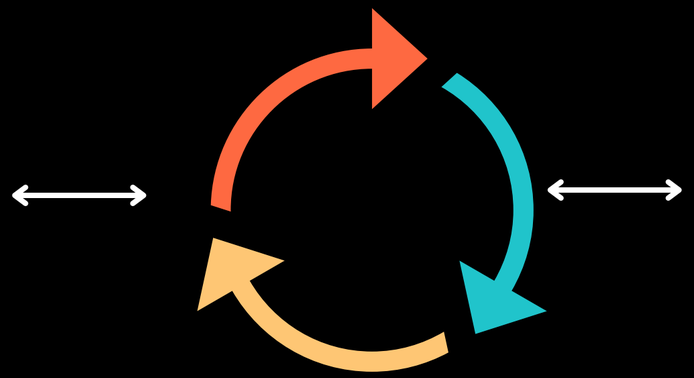
resolve('finished')

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



MACROTASK QUEUE

MICROTASK QUEUE

Event Loop

Queues

WebAPIs

EventLoop는 Blocking 되지 않았다.
UI 인터렉션이 가능하다.

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

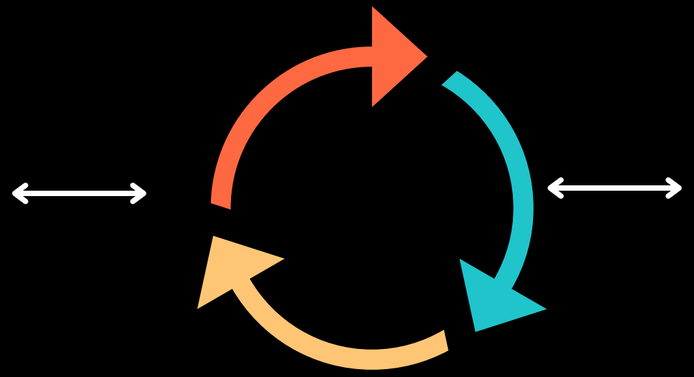
Main Thread의
Call Stack

anonymous
(글로벌 실행 컨텍스트)

Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Call
back

Queues

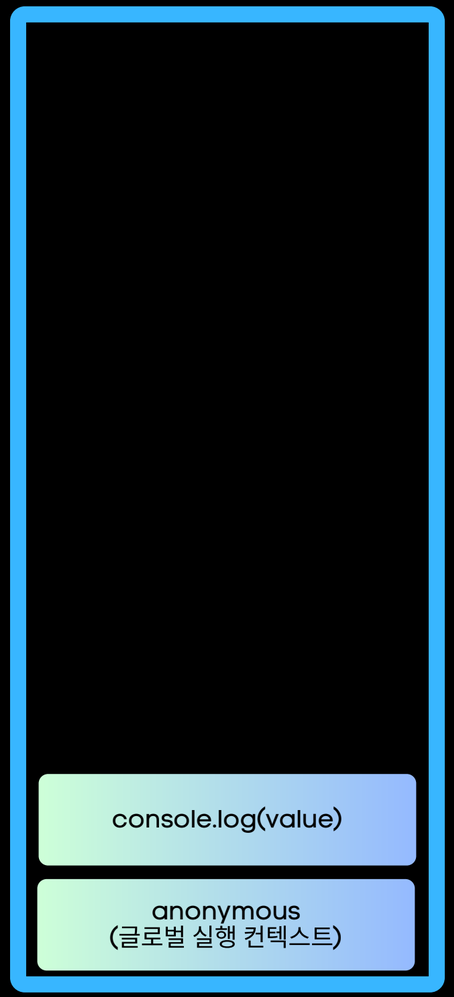
WebAPIs

EventLoop는 Blocking 되지 않았다.
UI 인터렉션이 가능하다.

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

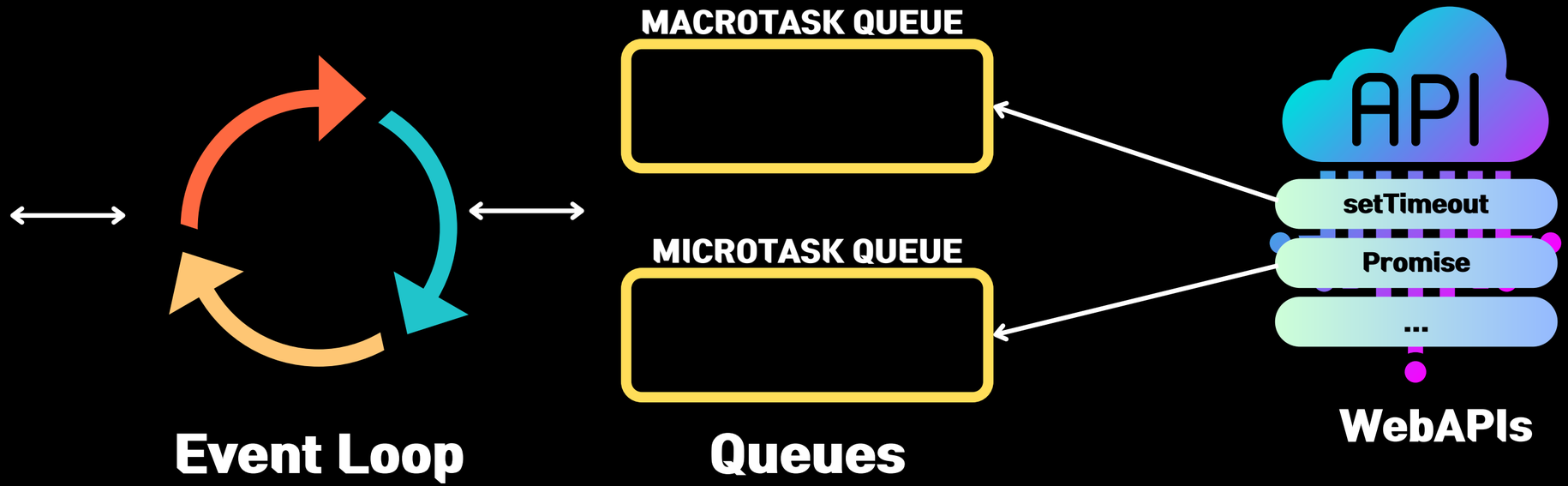


Promise's Executor

```
const nonBlocking = new Promise((resolve, reject) => {
    setTimeout(() => resolve('finished'), 10 * 60 * 1000);
});
nonBlocking.then((value) => console.log(value));
```

console

```
finished
```



EventLoop는 Blocking 되지 않았다.
UI 인터렉션이 가능하다.

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

MACROTASK QUEUE

MICROTASK QUEUE

API

setTimeout

Promise

...

Event Loop

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

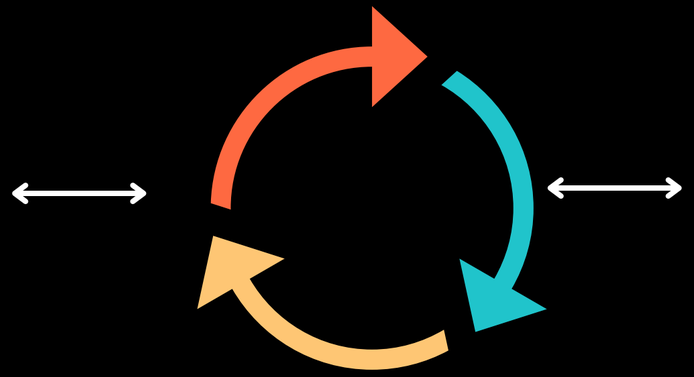
Main Thread의 Call Stack

anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

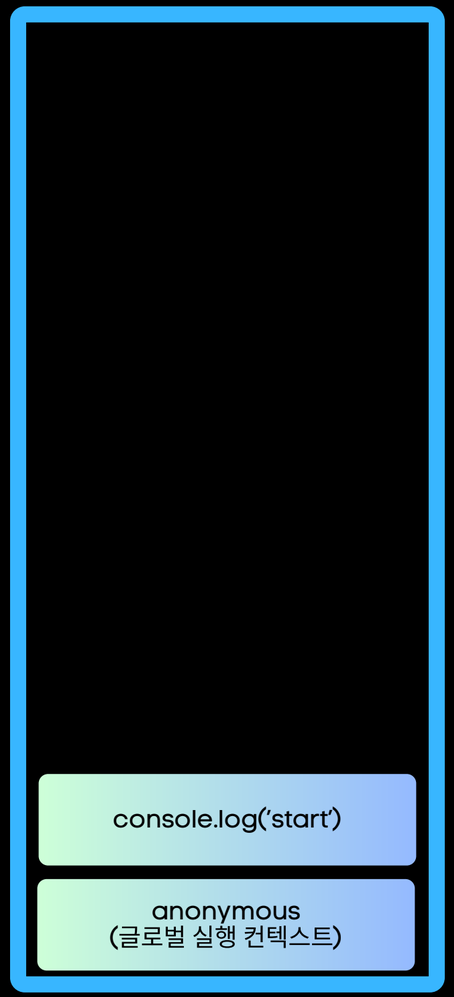
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

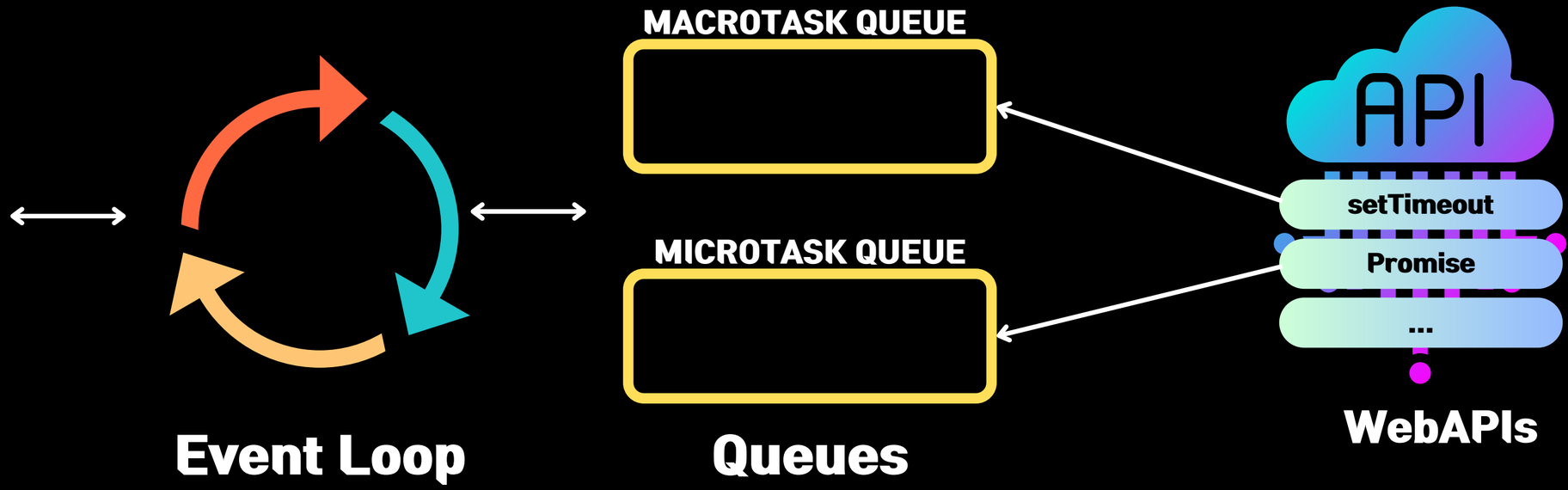


우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

start



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

setTimeout()

anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
```

MACROTASK QUEUE

MICROTASK QUEUE

Event Loop

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

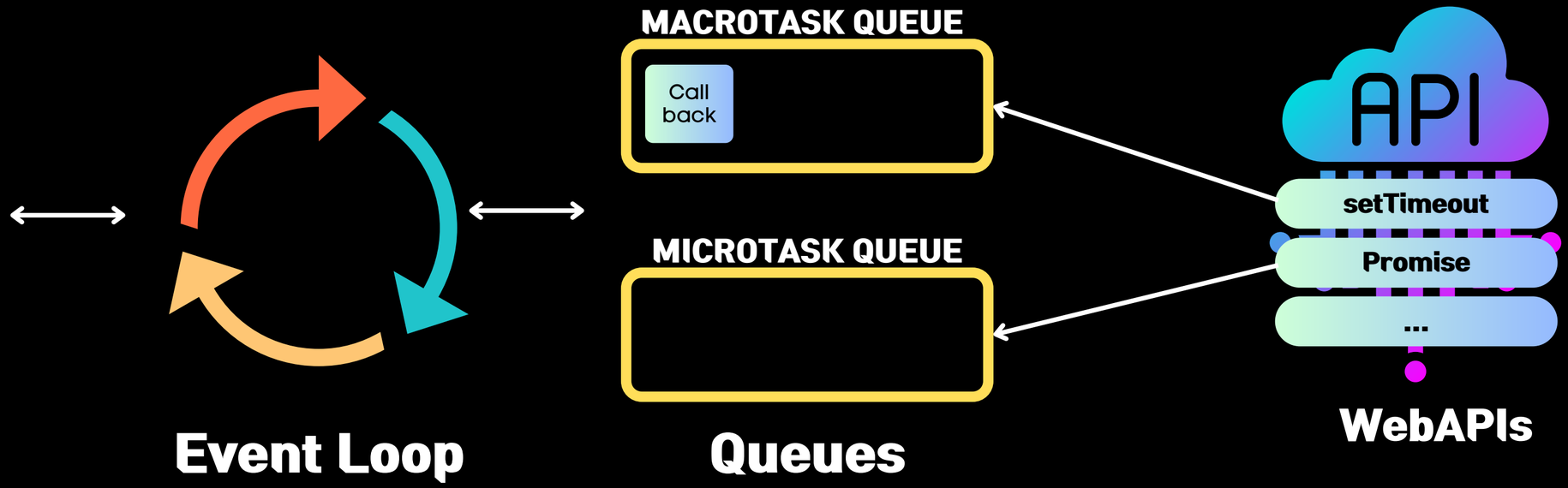
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
```



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

Promise.resolve()

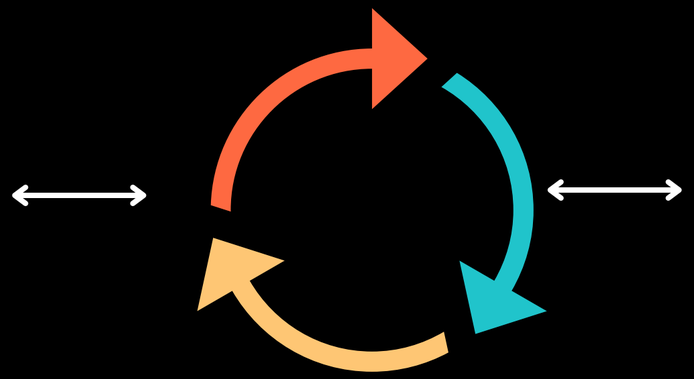
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

Promise.then()

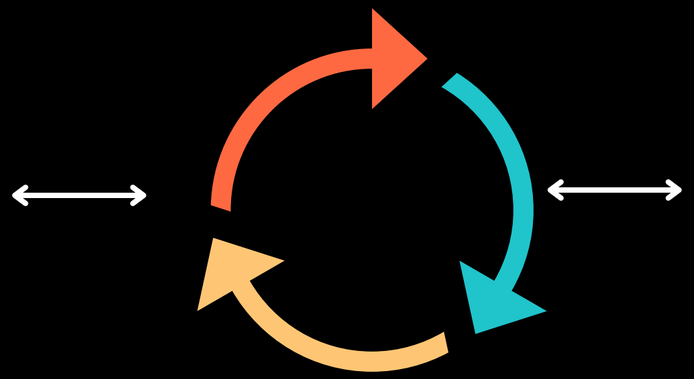
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

start



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

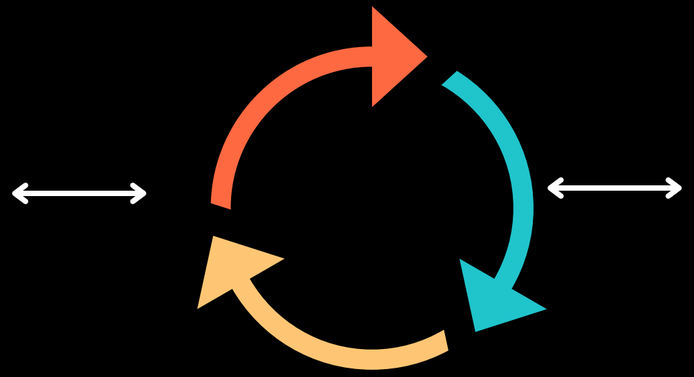
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Call back

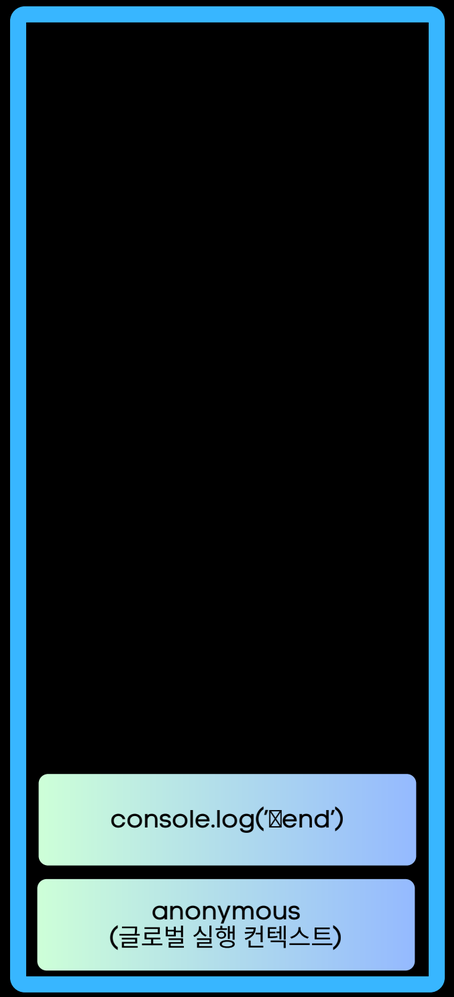
Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

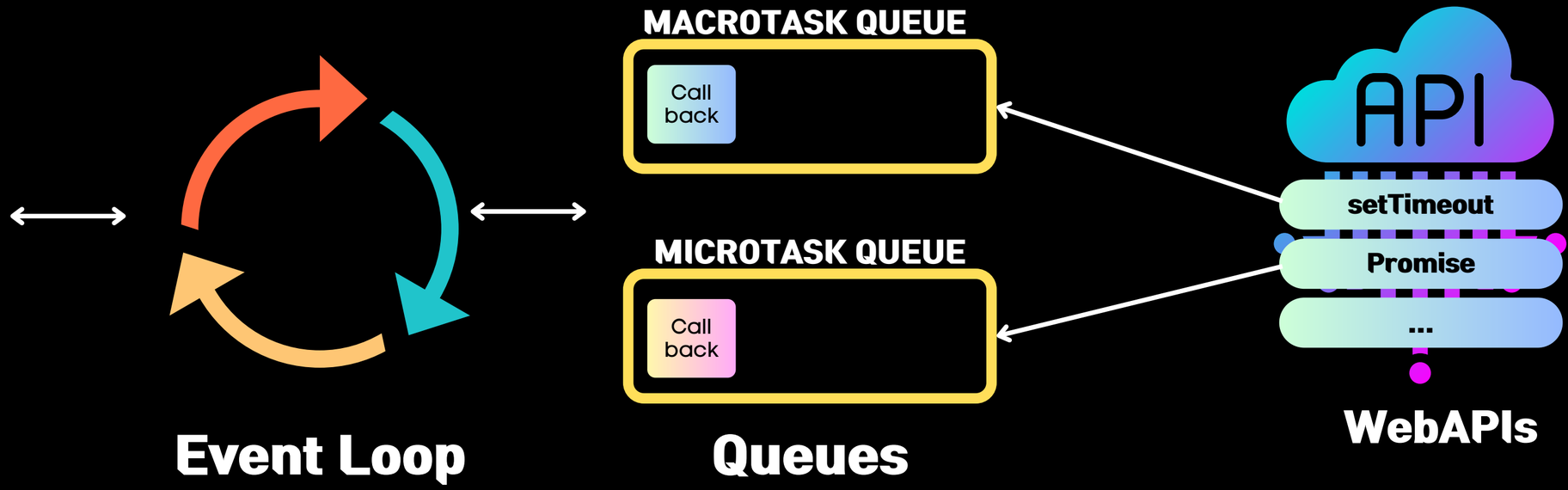


우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
  end
```



비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

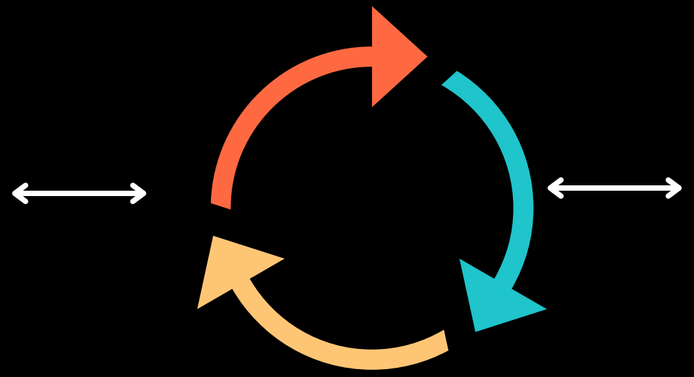
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
end
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Call back

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
end
Promise Resolved
```

Promise.then()
callback

anonymous
(글로벌 실행 컨텍스트)

Event Loop

MACROTASK QUEUE

Call
back

MICROTASK QUEUE

Queues

API

setTimeout

Promise

...

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

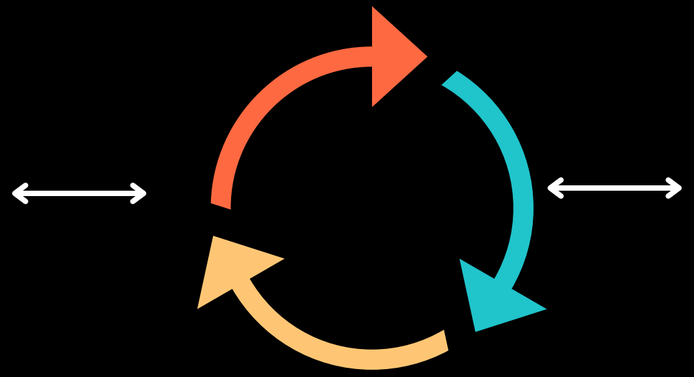
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
end
Promise Resolved
```



Event Loop

MACROTASK QUEUE

Call back

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의 Call Stack

setTimeout()
callback

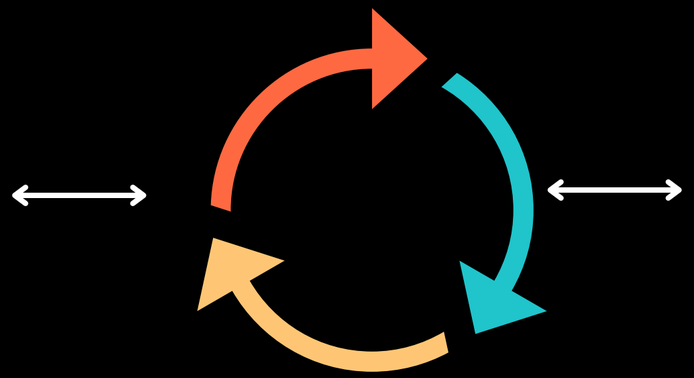
anonymous
(글로벌 실행 컨텍스트)

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
end
Promise Resolved
Timeout
```



Event Loop

MACROTASK QUEUE

MICROTASK QUEUE

Queues

WebAPIs

비동기 프로그래밍

진짜! 모두를 위한
자바스크립트

Main Thread의
Call Stack

우선순위: Macro Task Queue > Micro Task Queue

```
console.log('start');

setTimeout(() => {
    console.log('Timeout');
}, 0);

Promise.resolve('Promise Resolved')
    .then((value) => console.log(value));

console.log('end');
```

console

```
start
end
Promise Resolved
Timeout
```