

CODE SPITZ



ES6+

BLOCK, NONBLOCK & SYNC, ASYNC

1

2

3

4

5

6

BLOCK, NONBLOCK

FLOW IS BLOCKING

프로그램이 실행되면 도중에 멈춰지지 않고 끝까지 실행됨

FLOW IS BLOCKING

프로그램이 실행되면 도중에 멈춰지지 않고 끝까지 실행됨

```
for(const i of (function*(){  
  let i = 0;  
  while(true) yield i++;  
}))() console.log(i);
```

FLOW IS BLOCKING

프로그램이 실행되면 도중에 멈춰지지 않고 끝까지 실행됨

```
for(const i of (function*(){  
  let i = 0;  
  while(true) yield i++;  
}))() console.log(i);  
  
//script timeout
```

플랫폼의 안정성을 위해 블록되는 시간이 길면 강제 종료시킴

BLOCKING FUNCTION

점유하는 시간만큼 블록을 일으키는 함수

BLOCKING FUNCTION

점유하는 시간만큼 블록을 일으키는 함수

```
const f = v->{  
  let i = 0;  
  while(i++ < v);  
  return i;  
};  
f(10);  
f(1000000000000000);
```

BLOCKING FUNCTION

점유하는 시간만큼 블록을 일으키는 함수

```
const f = v->{  
  let i = 0;  
  while(i++ < v);  
  return i;  
};  
f(10);  
f(1000000000000000);
```

배열순회, 정렬 - 배열크기에 따라
DOM순회 - DOM의 하위구조에 따라
이미지프로세싱 - 이미지크기에 따라

BLOCKING EVASION

BLOCKING EVASION

독점적인 cpu점유로 인해 모든 동작이 정지됨

BLOCKING EVASION

독점적인 cpu점유로 인해 모든 동작이 정지됨

타임아웃체크에 의해 프로그램이 강제 중단됨

BLOCKING EVASION

독점적인 cpu점유로 인해 모든 동작이 정지됨

타임아웃체크에 의해 프로그램이 강제 중단됨

블록킹의 조합을 예측할 수 없음

BLOCKING EVASION

독점적인 cpu점유로 인해 모든 동작이 정지됨

타임아웃체크에 의해 프로그램이 강제 중단됨

블록킹의 조합을 예측할 수 없음

```
const f = v->other(some(v), v * 2);  
f(10);
```

BLOCKING EVASION

순차적인 실행

1

2

3

BLOCKING EVASION

시분할 운영체제의 동시 실행



BLOCKING EVASION

자바스크립트 쓰레드

MAIN UI THREAD 1

BACKGROUND THREAD N

WEB WORKER THREAD

BLOCKING EVASION

TIME SLICING

```
const looper = (n, f)=>{  
  for(let i = 0; i < n; i++) f(i);  
};
```

```
looper(10, console.log);  
looper(10000, console.log);
```

BLOCKING EVASION

TIME SLICING MANUAL

```
const looper = (n, f, slice = 3)=>{  
  let limit = 0, i = 0;  
  const runner =_=>{  
    while(i < n){  
      if(limit++ < slice) f(i++);  
      else{  
        limit = 0;  
        requestAnimationFrame(runner);  
        break;  
      }  
    }  
  };  
  requestAnimationFrame(runner);  
};
```

```
looper(12, console.log);
```

BLOCKING EVASION

TIME SLICING AUTO

```
const looper = (n, f, ms = 5000, i = 0)=>{  
  let old = performance.now(), curr;  
  const runner =_=>{  
    while(i < n){  
      curr = performance.now();  
      if(curr - old < ms) f(i++);  
      else{  
        old = curr;  
        requestAnimationFrame(runner);  
        break;  
      }  
    }  
  };  
  requestAnimationFrame(runner);  
};
```

BLOCKING EVASION

WEB WORKER

```
const backRun = (f, end, ...arg)=>{  
  const blob = new Blob([`  
    onmessage =e=>postMessage((${f}))(e.data));  
  `], {type:'text/javascript'});  
  const url = URL.createObjectURL(blob);  
  const worker = new Worker(url);  
  worker.onmessage =e=>end(e.data);  
  worker.onerror =e=>end(null);  
  worker.postMessage(arg);  
};
```

```
backRun(v=>v[0] + v[1], console.log, 3, 5);
```

BLOCKING EVASION

~~BLOCKING EVASION~~
NON BLOCKING

~~BLOCKING EVASION~~ NON BLOCKING

서브루틴이 즉시 플로우 제어권을 내놓는 것

~~BLOCKING EVASION~~ NON BLOCKING

서브루틴이 즉시 플로우 제어권을 내놓는 것

```
const a = 123;  
looper(12, console.log);  
backRun(v=>v[0] + v[1], console.log, 3, 5);  
console.log(a); //어쨌든 콘솔은 123부터 출력
```


SYNC, ASYNC

SYNC

SYNC

서브루틴이 즉시 값을 반환함

SYNC

서브루틴이 즉시 값을 반환함

ASYNC

SYNC

서브루틴이 즉시 값을 반환함

ASYNC

서브루틴이 콜백을 통해 값을 반환함

SYNC

서브루틴이 즉시 값을 반환함

```
const double = v=>v*2;  
console.log(double(2)); //4
```

ASYNC

서브루틴이 콜백을 통해 값을 반환함

SYNC

서브루틴이 즉시 값을 반환함

```
const double = v=>v*2;  
console.log(double(2)); //4
```

ASYNC

서브루틴이 콜백을 통해 값을 반환함

```
const double = (v, f)=>f(v*2);  
double(2, console.log); //4
```

SYNC

서브루틴이 즉시 값을 반환함

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

```
const sum = n=>{  
  let sum = 0;  
  for(let i = 1; i <= n; i++) sum += i;  
  return sum;  
};  
sum(100);
```

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

```
const sum = n=>{  
  let sum = 0;  
  for(let i = 1; i <= n; i++) sum += i;  
  return sum;  
};  
sum(100);
```

NON BLOCK 즉시 플로우제어권을 반환함

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

```
const sum = n=>{  
  let sum = 0;  
  for(let i = 1; i <= n; i++) sum += i;  
  return sum;  
};  
sum(100);
```

NON BLOCK 즉시 플로우제어권을 반환함

```
const sum = n=>{  
  const result = {isComplete:false};  
  requestAnimationFrame(_=>{  
    let sum = 0;  
    for(let i = 1; i <= n; i++) sum += i;  
    result.isComplete = true;  
    result.value = sum;  
  });  
};  
const result = sum(100);  
while(!result.isComplete);  
console.log(result.value);
```

ASync

서브루틴이 콜백을 통해 값을 반환함

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

```
const sum = (n, f)=>{  
  let sum = 0;  
  for(let i = 1; i <= n; i++) sum += i;  
  return f(sum);  
};  
sum(10, console.log);  
console.log(123);  
//55 → 123
```

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

```
const sum = (n, f)=>{  
  let sum = 0;  
  for(let i = 1; i <= n; i++) sum += i;  
  return f(sum);  
};  
sum(10, console.log);  
console.log(123);  
//55 → 123
```

NON BLOCK 즉시 플로우제어권을 반환함

```
const sum = (n, f)=>{  
  requestAnimationFrame(_=>{  
    let sum = 0;  
    for(let i = 1; i <= n; i++) sum += i;  
    f(sum);  
  });  
};  
sum(10, console.log);  
console.log(123);  
//123 → 55
```

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음
normalAPI, legacyAPI

NON BLOCK 즉시 플로우제어권을 반환함

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음
normal API, legacy API

NON BLOCK 즉시 플로우제어권을 반환함

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함
modern API

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음
normal API, legacy API

NON BLOCK 즉시 플로우제어권을 반환함
old API
IOCP, Future, img.complete..

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함
modern API

SYNC

서브루틴이 즉시 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음
normal API, legacy API

NON BLOCK 즉시 플로우제어권을 반환함
old API
IOCP, Future, img.complete..

ASYNC

서브루틴이 콜백을 통해 값을 반환함

BLOCK 즉시 플로우제어권을 반환하지 않음

NON BLOCK 즉시 플로우제어권을 반환함
modern API

TRAP

SIMILAR ASYNC-BLOCK

```
const sum = (n, f)=>{  
  requestAnimationFrame(_=>{  
    let sum = 0;  
    for(let i = 1; i <= n; i++) sum += i;  
    f(sum);  
  });  
};  
sum(100000000, console.log);  
console.log(123);
```

SIMILAR ASYNC-BLOCK

```
const backRun = (f, end, ...arg)=>{
  const blob = new Blob([`
    onmessage =e=>postMessage((${f}))(e.data));
  `], {type:'text/javascript'});
  const url = URL.createObjectURL(blob);
  const worker = new Worker(url);
  worker.onmessage =e=>end(e.data);
  worker.onerror =e=>end(null);
  worker.postMessage(arg);
};
```

```
const f = v=>{
  for(let i = 1, sum = 0; i <= v[0]; i++){
    sum += i;
  }
  return sum;
};
let i = 1000;
while(i-->0) backRun(f, console.log, 100000);
```