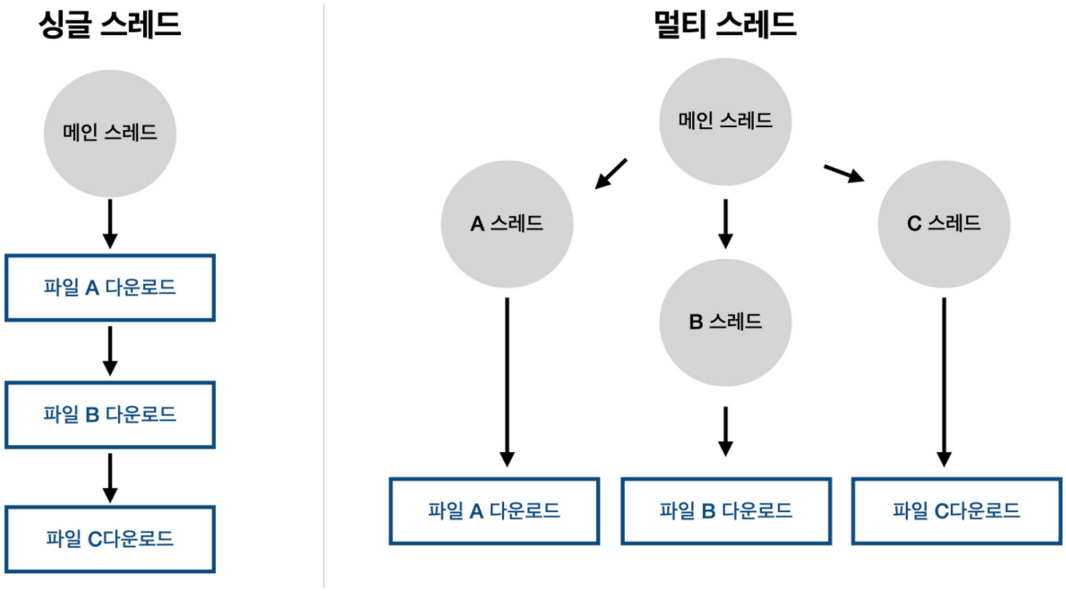
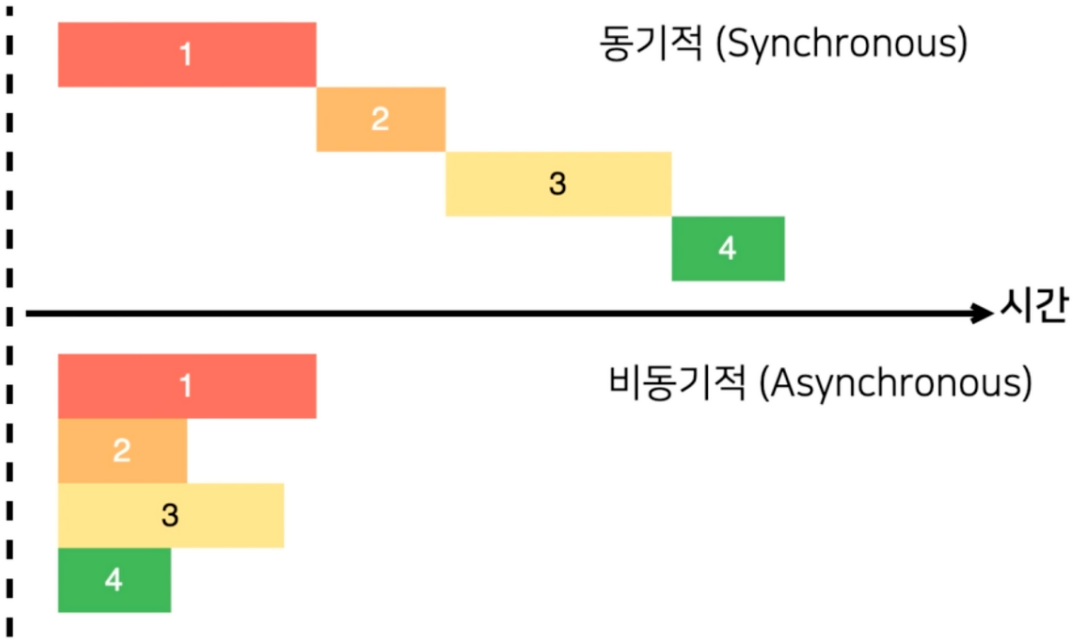
비동기, 동기 처리에 대한 이해와 특징 알아보기

- 자바스크립트는 스크립트 언어, 인터프리터 언어이기에
  - 싱글 스레드 이기 때문에 코드가 작성된 순서대로 작업이 처리된다





- 동기
  - 순차적으로 태스크를 수행 하며 하나의 일이 끝나기 전 까지 다른 작업이 불가능하도록 막음(블로킹)
- 비동기
  - 코드가 끝날때 까지 기다리는것이 아닌 다른 코드가 실행되는 순간에도 다른 일을 처리할 수 있다.

1. 비동기 처리하는 방법
   a. Callback

```
const printString = (string, callback) => {
    setTimeout(
        () => {
            console.log(string);
            callback();
        },
        Math.floor(Math.random() * 100) + 1
    )
}

const printAll = () => {
    printString("a", () => {
        printString("b", () => {
            printString("c", () => {
                printString("d", () => {
```

```
                })
            })
        })
    })
}
printAll();
```

- 다른 함수의 인자로 넘겨지는 함수, 콜백 수신함수에 의해 특정 시점에 실행된다. 동기 콜백의 경우 호출 시 즉시 실행되고 비동기 콜백의 경우 일정 시간 이후 실행된다.

b. Promise

   i. 프로미스는 객체이기 때문에 생성자 함수를 호출하여 인스턴스화할 수 있습니다.

      1. pending, fulfilled, rejected 가 있습니다.

      프로미스의 생성자 함수는 **resovle**와 **reject**함수를 인자로 전달받는 콜백 함수를 인자로 전달받습니다.

      프로미스는 인자로 전달받은 콜백 함수를 내부에서 비동기 처리 합니다.

   ii. 아래와 같이 사용할 수 있습니다.

```
const promise = () => new Promise((resolve, reject) => {
    let a = 1 + 1

    if(a == 2) {
        resolve('success')
    } else {
        reject('failed')
    }
})

promise().then((message) => {
    console.log('This is in the then ' + message)
}).catch((message) => {
    console.log('This is in the catch' + message)
})
```

   iii. **Promise.all** 메소드를 이용해 한번에 병렬처리 할 수 있습니다.
   즉 모든 프로미스가 fullfilled 상태일 때 반환됩니다.

   iv. Promise.race 메소드는 가장먼저 fullfilled되는 것을 반환합니다.

   v. Promise.allSettled는 rejected가 되어도 죽지않고 수행 상태와 결과값을 반환합니다.

c. async await

   i. async 키워드는 **함수를 선언할 때** 붙여줄 수 있습니다.

   ii. async는 최상위 함수나 변수에 쓸수있으며 항상 promise를 반환하고 await로 promise가 처리 될때까지 대기합니다

   iii. async를 붙인 함수는 반환값이 자동으로 Promise가 된다