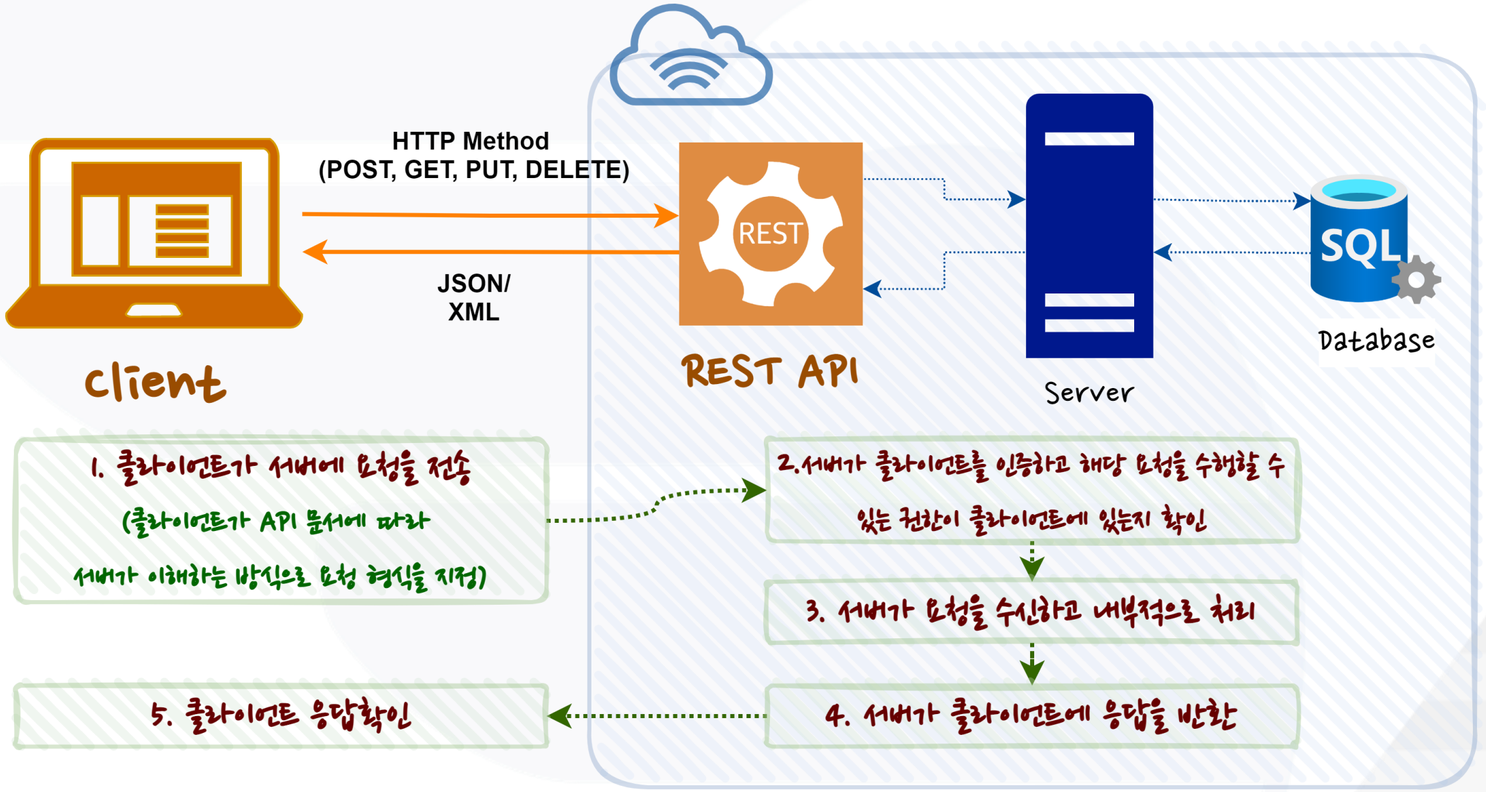
AI 활용 빅데이터분석
풀스택웹서비스 SW 개발자 양성과정

서버통신

부산대학교 소프트웨어교육센터
PUSAN NATIONAL UNIVERSITY SOFTWARE EDUCATION CENTER

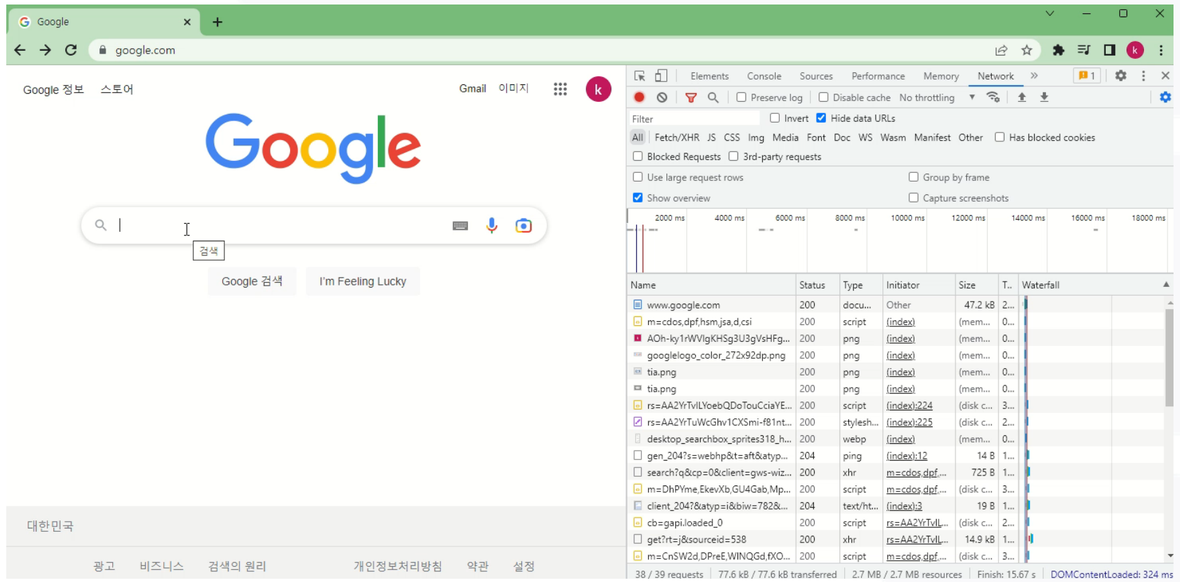
REST API

- Representational State Transfer약어로 RESTful API라고도 함
  - REST 아키텍처의 제약 조건을 준수하는 애플리케이션 프로그래밍 인터페이스
  - REST API를 통해 요청이 수행될 때 REST API는 리소스 상태에 대한 표현을 요청자에게 전송
    - 전송자료는 **XML**이나 **JSON** 형태로 전송



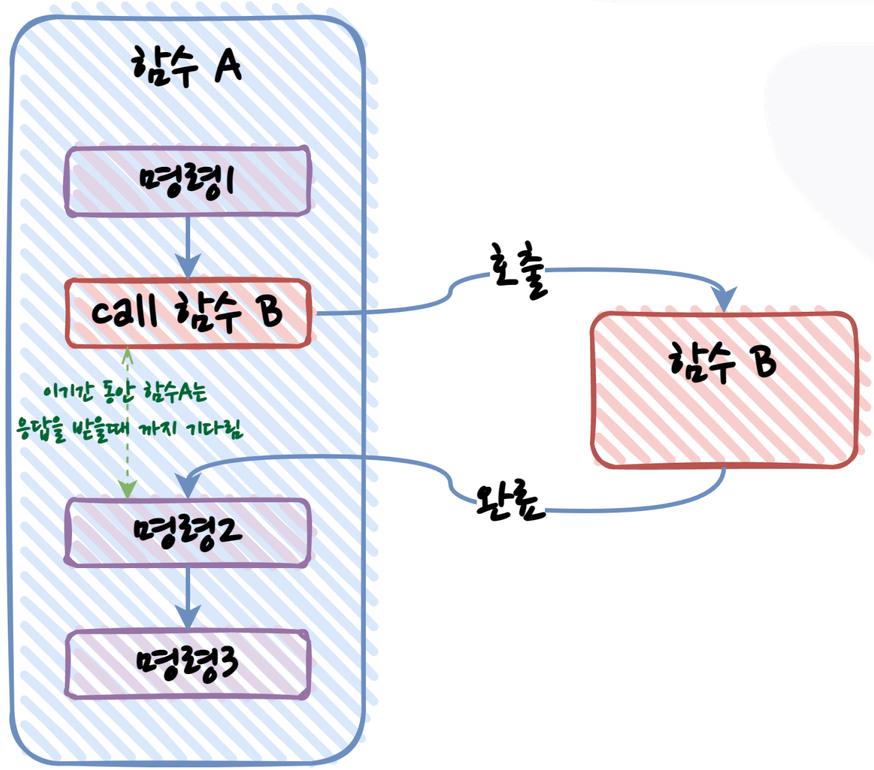
Ajax

- **Asynchronous JavaScript and XML**
  - 웹 페이지를 빠르게 갱신하면서, 서버와 브라우저 사이에서 데이터를 비동기적으로 교환하는 기술
  - 웹 페이지에서 새로운 데이터를 요청하고, 응답을 받을 때 페이지 전체를 다시 로드하지 않고, 필요한 부분만 빠르게 갱신



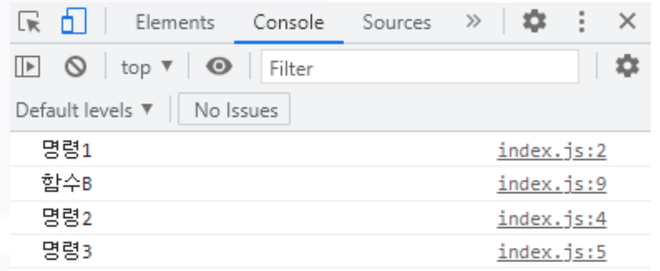
동기(Synchronous) 처리

- 동기처리
  - 함수를 포함한 모든 코드가 위에서 아래로 순서대로 실행



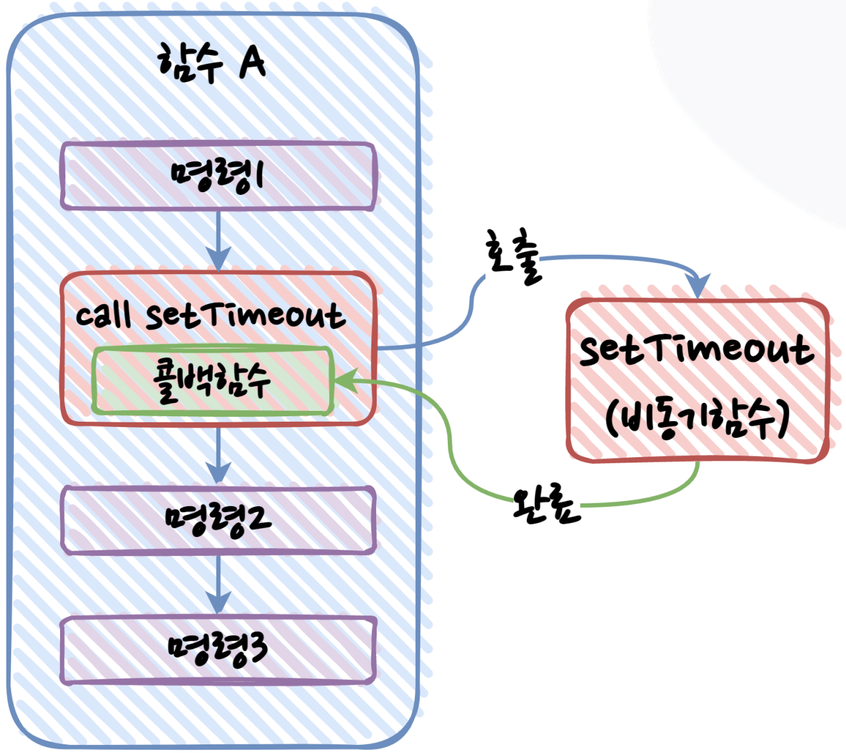

```
const funA = () => {
  console.log("명령1") ;
  funB();
  console.log("명령2") ;
  console.log("명령3") ;
}

const funB = () => {
  console.log("함수B") ;
}
```

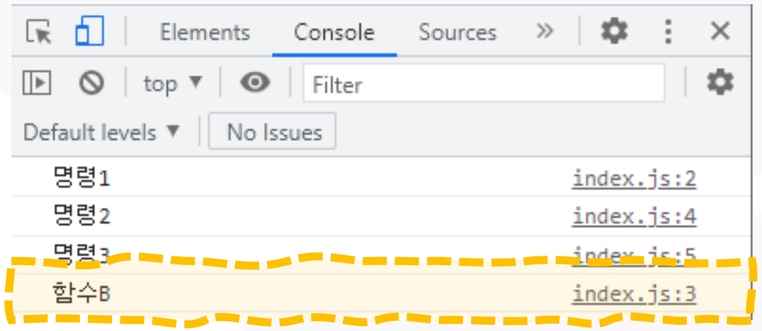


비동기 (Asynchronous) 처리

- 비동기처리
  - 명령을 요청한 결과가 나올때까지 계속 기다리는 것이 아니라 다음 작업을 계속 수행하고 요청 응답이 오면 필요한 작업을 마무리




```
const funA = () => {
  console.log("명령1") ;
  setTimeout(()=>console.log("함수B"), 1000);
  console.log("명령2") ;
  console.log("명령3") ;
}
```



비동기 (Asynchronous) 처리

- 콜백 지옥(Callback Hell)

```
console.log('명령1')
setTimeout(() => {
    console.log('timer 실행1')
}, 700);

setTimeout(() => {
    console.log('timer 실행2')
}, 100);

setTimeout(() => {
    console.log('timer 실행3')
}, 500);

console.log('명령2')
```

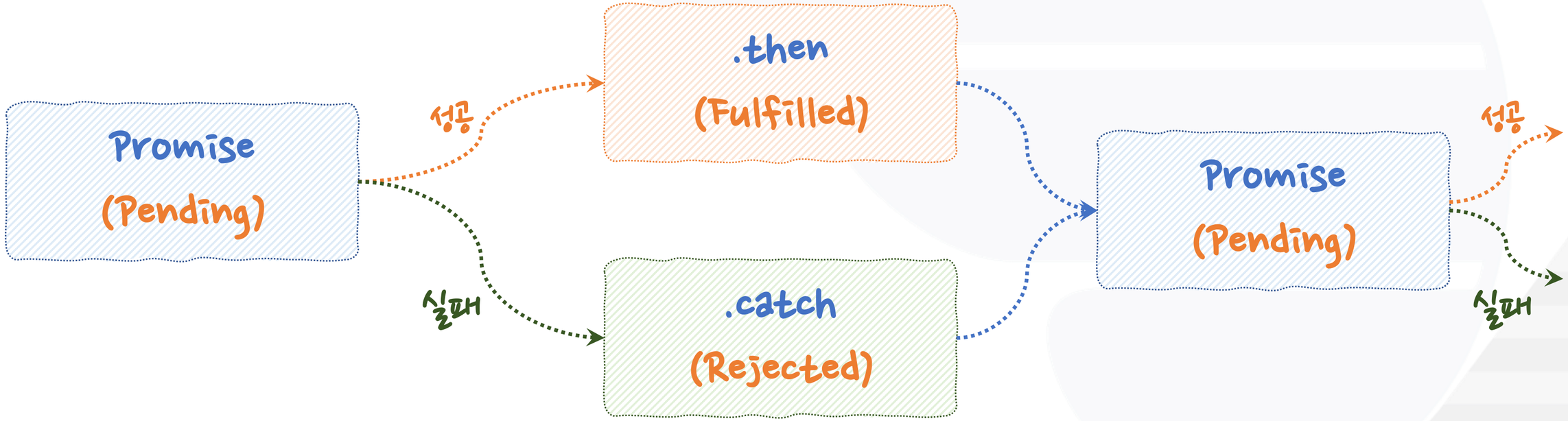
```
명령1
명령2
timer 실행2
timer 실행3
timer 실행1
```

```
console.log('명령1')
setTimeout(() => {
    console.log('timer 실행1')
    setTimeout(() => {
        console.log('timer 실행2')
        setTimeout(() => {
            console.log('timer 실행3')
        }, 500);
    }, 100);
}, 700);
console.log('명령2')
```

```
명령1
명령2
timer 실행1
timer 실행2
timer 실행3
```

Promise

- JavaScript에서 비동기 작업을 처리하는 방법
- Promise 객체는 어떤 작업의 완료 결과
  - Pending(대기): 비동기 처리 로직이 아직 미완료인 상태
  - Fulfilled(이행): 비동기 처리가 완료되어 promise가 결과 값을 반환해준 상태
  - Rejected(실패): 비동기 처리가 실패하거나 오류가 발생한 상태



Fetch API

- JavaScript에서 HTTP 요청을 보내고 응답을 받는 기능을 제공하는 API
  - Promise 기반으로 동작
  - fetch 함수를 사용하여 HTTP 요청을 보내고, 응답이 완료되면 Promise 객체를 반환
  - 응답은 Response 객체로 반환되며, 이 객체를 통해 응답의 상태와 데이터를 처리

```
let promise = fetch(url, [options])
```

- url – 접근하고자 하는 URL
- options – 선택 매개변수, method나 header 등을 지정할 수 있음
  options에 아무것도 넘기지 않으면 요청은 GET 메서드로 진행

Fetch API

```
fetch(url)
  .then((resp) => resp.json())
  .then((data) => {
      const dailyBoxOfficeList = data.boxOfficeResult.dailyBoxOfficeList
      console.log(dailyBoxOfficeList)
  })
  .catch((err) => console.log(err));
```

성공

성공

실패

해결문제

- 영화진흥위원회 박스오피스 목록 가져오기
  - https://www.kobis.or.kr/kobisopenapi/homepg/apiservice/searchServiceInfo.do