

대학 주도형 아카데미 K-디지털 트레이닝
AI 데이터분석 풀스택 웹 개발자 양성과정

RESTful API

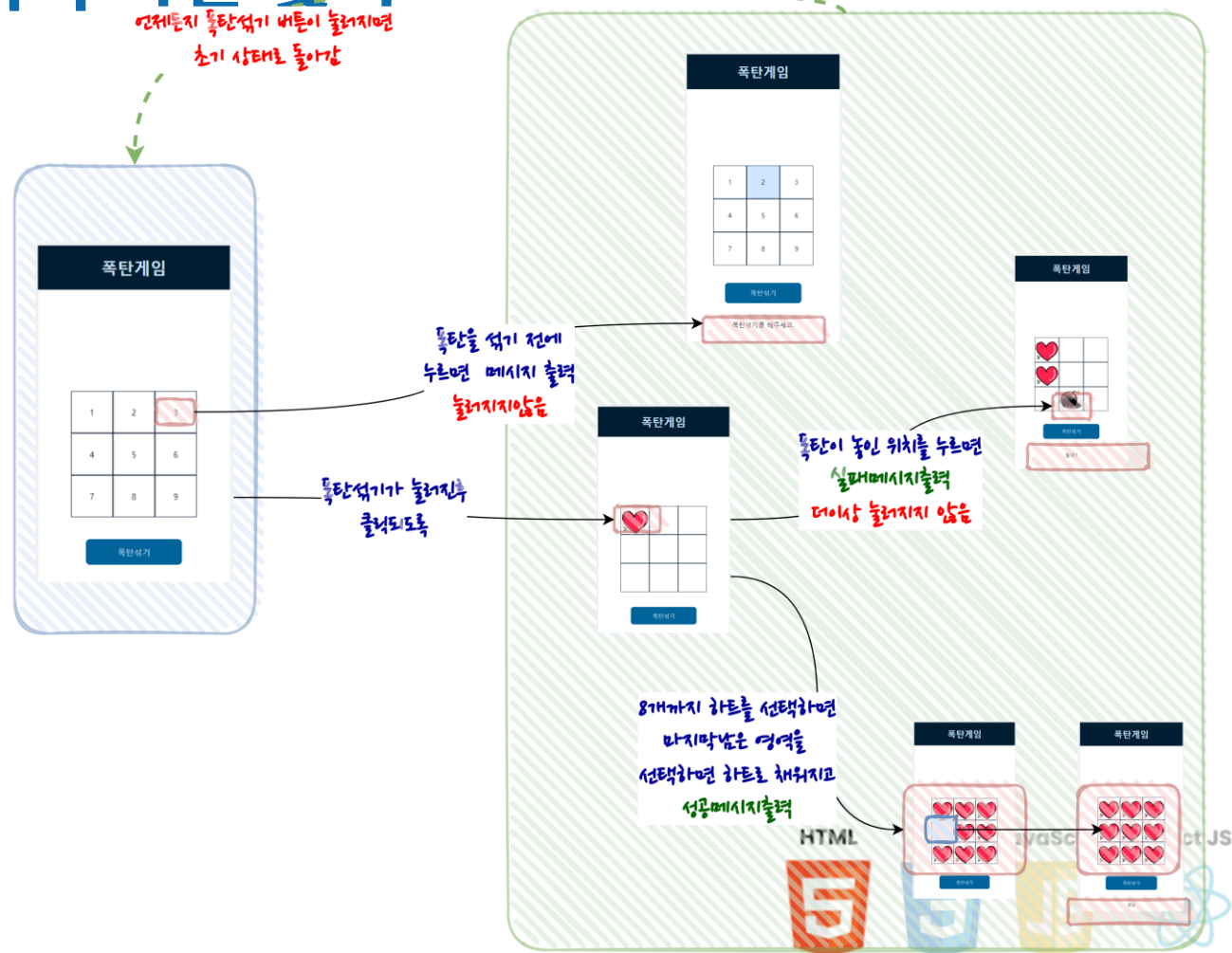
김경민

pnumin@pusan.ac.kr



해결문제

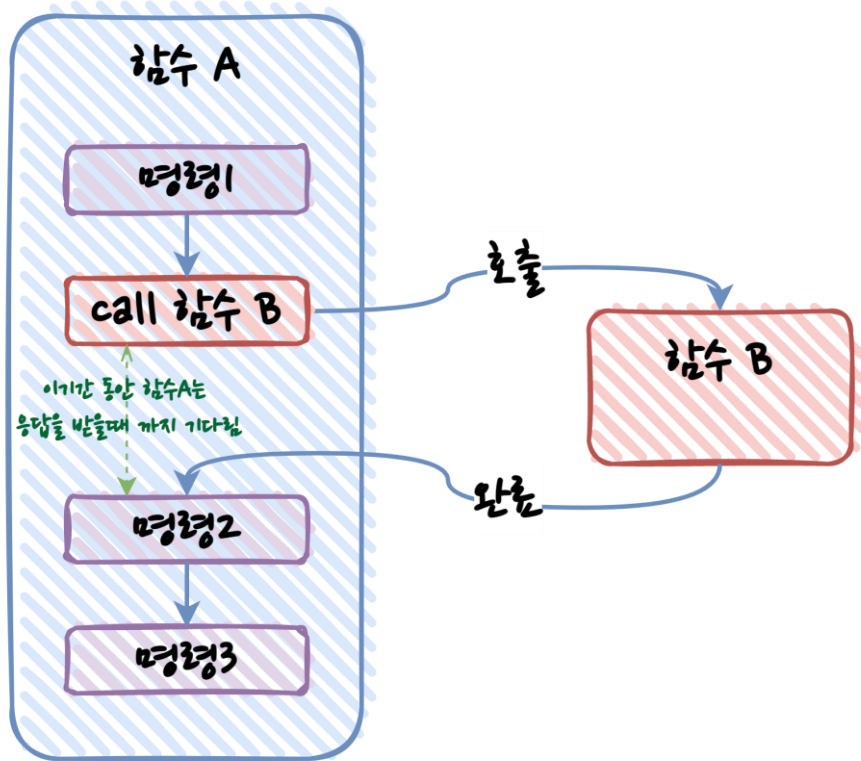
- 3 x 3 으로 이루어진 보드를 눌러서 폭탄 찾기



동기(Synchronous) 처리

• 동기처리

- 함수를 포함한 모든 코드가 위에서 아래로 순서대로 실행



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

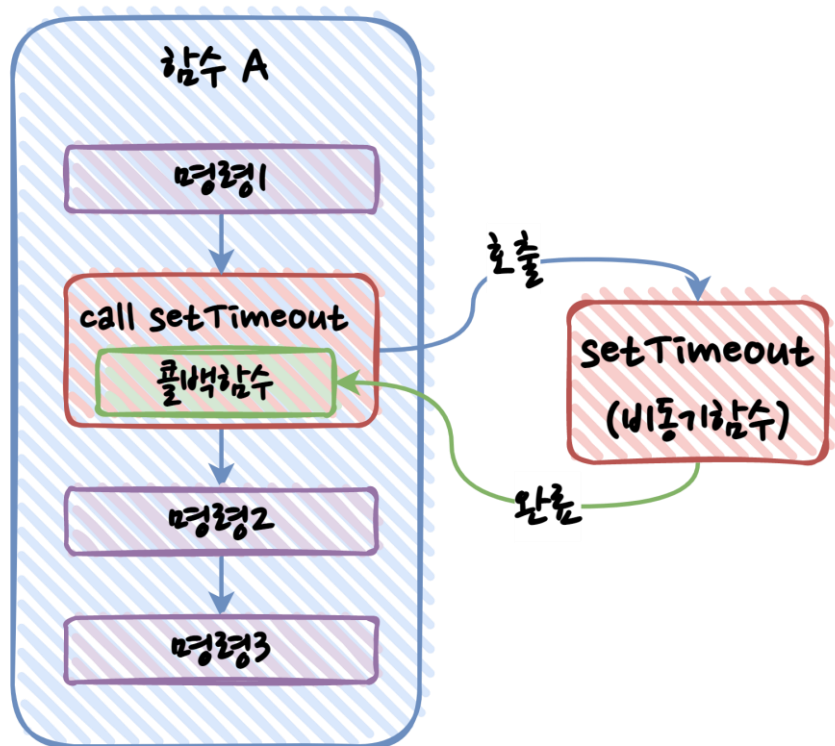
const funB = () => {
  console.log("함수B") ;
}
```

명령1	index.js:2
함수B	index.js:9
명령2	index.js:4
명령3	index.js:5

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") ;  
}
```

명령1	index.js:2
명령2	index.js:4
명령3	index.js:5
함수B	index.js:3

JavaScript React JS



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

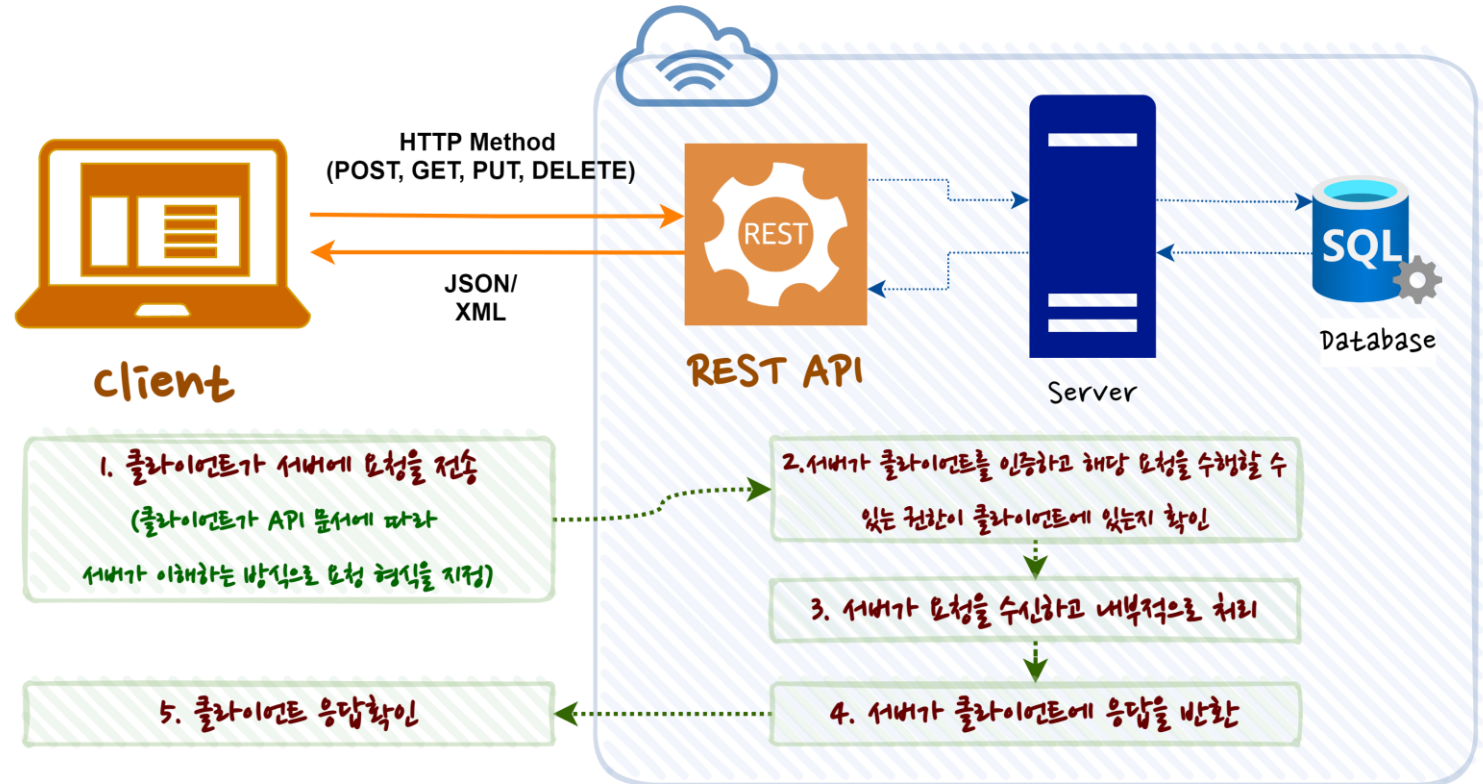
timer 실행2

timer 실행3

REST API

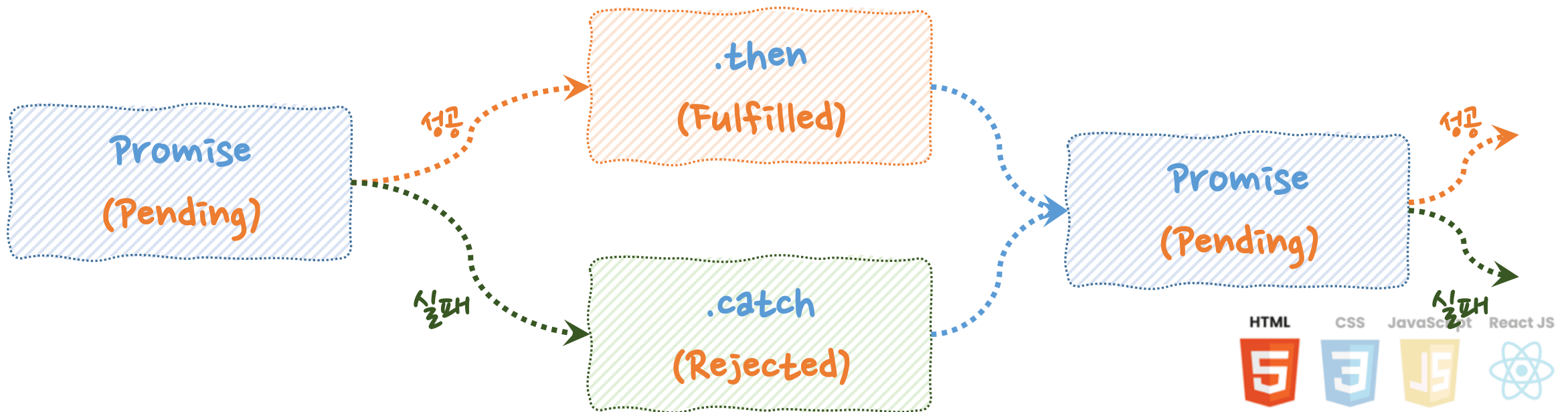
- Representational State Transfer
약어로 RESTful API라고도 함

- REST 아키텍처의 제약 조건을 준수하는 애플리케이션 프로그래밍 인터페이스
- REST API를 통해 요청이 수행될 때 REST API는 리소스 상태에 대한 표현을 요청자에게 전송
 - 전송자료는 XML이나 JSON 형태로 전송



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

- 비동기적으로 서버와 데이터를 주고받을 수 있도록 설계된 최신 웹 API
 - Promise 기반
 - fetch()는 Promise를 반환하므로, .then()과 .catch()를 사용하여 비동기 처리
 - 다양한 데이터 형식 지원
 - response.json(), response.text() 등의 메서드를 사용하여 다양한 형식의 데이터를 처리
 - 옵션 설정
 - HTTP 메서드, 헤더, 본문 등을 옵션 객체를 통해 설정

```
fetch(url, options)
  .then(response => response.json()) // 응답을 JSON으로 변환
  .then(data => console.log(data)) // 데이터 출력
  .catch(error => console.error("Error:", error));
```


해결문제

일일박스오피스

로그인

연도 - 월 - 일



해결문제 - 어제 날짜 구하기

일	월	화	수	목	금	토
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

어제 날짜가 선택되도록

```
const dt = document.querySelector('input[type=date]');
```

```
const mvList = document.querySelector('div')  
// 어제 날짜 구하기
```

- 현재 날짜와 시간을 가진 Date 객체를 생성
- 오늘날짜에서 1일을 빼서 어제 날짜 구함

```
const yesterday = new Date();  
yesterday.setDate(yesterday.getDate() - 1);
```

```
// 어제 년월일 가져오기
```

```
const year = yesterday.getFullYear();
```

```
// 월은 0부터 시작하므로 1을 더함
```

```
const month = String(yesterday.getMonth() + 1).padStart(2, '0');
```

```
const day = String(yesterday.getDate()).padStart(2, '0');
```

```
const maxDate = `${year}-${month}-${day}`;
```

```
// input 요소의 max 속성에 어제 날짜 설정
```

```
dt.max = maxDate;
```

- 문자열의 길이를 원하는 만큼 늘리고, 필요에 따라 문자열의 앞쪽을 특정 문자로 채워주는 메서드

해결문제 - 데이터 불러오기

1 - 에이리언: 로물루스

- 2 - 파일럿
- 3 - 늑장가든
- 4 - 사랑의 하츠피
- 5 - 트위스터스
- 6 - 행복의 나라
- 7 - 빅토리
- 8 - 필사의 추적
- 9 - 극장판 블루 록 -에피소드 나기-
- 10 - 슈퍼배드 4

• 날짜가 변경될 때 감지

```
dt.addEventListener('change', ()=>{  
  let testAPIKey = '82ca741a2844c5c180a208137bb92bd7' ;  
  let url = 'https://kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/sear  
  url = `${url}&key=${testAPIKey}&targetDt=${dt.value.replaceAll('-', '')}`
```

• 선택된 날짜에 해당하는 자료 가져오기

```
  fetch(url)  
    .then(resp => resp.json())  
    .then(data => {  
      let mv = data.boxOfficeResult.dailyBoxOfficeList ;  
      let mvLi = '' ;  
      for (let item of mv){  
        mvLi = mvLi + `<li>${item.rank} - ${item.movieNm}</li>`  
      }  
      mvList.innerHTML = mvLi;  
    })  
    .catch(err => console.error('fetch 오류:', err))  
  });
```

• 10개 데이터 목록 만들기

• 목록화면에 표시

Vercel Hosting

The collage illustrates the Vercel hosting process through five sequential screenshots:

- Login to Vercel:** The first screenshot shows the Vercel login page with the text "Log in to Vercel" and "GitHub 로그인" (GitHub login) highlighted. It includes buttons for "Continue with GitHub", "Continue with SAML SSO", "Login with Passkey", and "Continue with Email".
- New Project:** The second screenshot shows the "New Project" page with the text "Let's build something new." and "To deploy a new Project, import an existing Git Repository or...".
- Import Git Repository:** The third screenshot shows the "Import Git Repository" section where a repository named "kd01_html" is selected. The text "GitHub 레포지토리 import" (GitHub repository import) is highlighted.
- Deploy:** The fourth screenshot shows the "Deploy" button being clicked. The text "Deploy" is highlighted.
- Project Page:** The fifth screenshot shows the deployed project page titled "KDT 1기 : 프론트엔드" (KDT 1st Batch: Frontend). It lists a table of contents for the course and includes links for "HTML 인프런 참고 사이트" (Inflearn reference site for HTML), "W3school", and "MDN".