**목표!**

1. **ajax**의 사용자가 됩니다.
2. promise의 사용자가 됩니다.
3. rest api를 이해합니다.

**Ajax**

Asynchronous JavaScript And XML 의 약자

서버와 비동기적으로 통신할 때 사용하는 API

과거에는 XMLHttpRequest를 사용했지만, 현재는 fetch를 사용

**browser developer tools - Network tab**

브라우저와 서버 간에 통신내용을 모니터링 할 때 사용하는 도구

**json-server**

**용도**

프로토타입 백엔드 서버

**홈페이지**

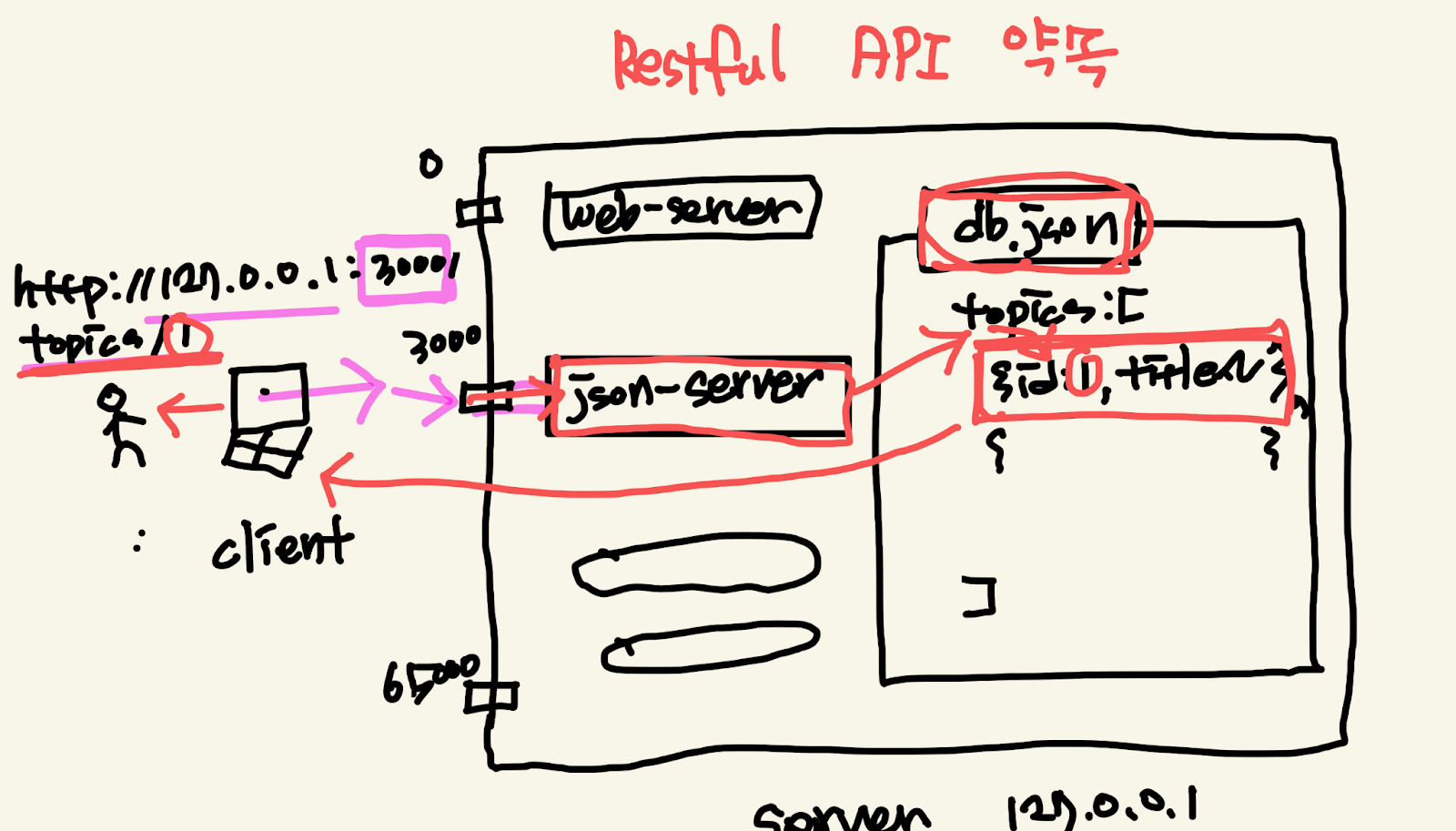
https://github.com/typicode/json-server

설치

npm install -g json-server

**실행**

json-server --watch db.json

****

**fetch**

참조 : <https://devhints.io/js-fetch>

**GET**

|  |
| --- |
| fetch('/data.json')    .then(response => response.json())    .then(data => {      console.log(data)    }) |

**POST**

|  |
| --- |
| fetch(url, {          method: 'POST',          headers: {              'Content-Type': 'application/json',          },          body: JSON.stringify(data)      })      .then(response => response.json())      .then(data => {          console.log(data)       }) |

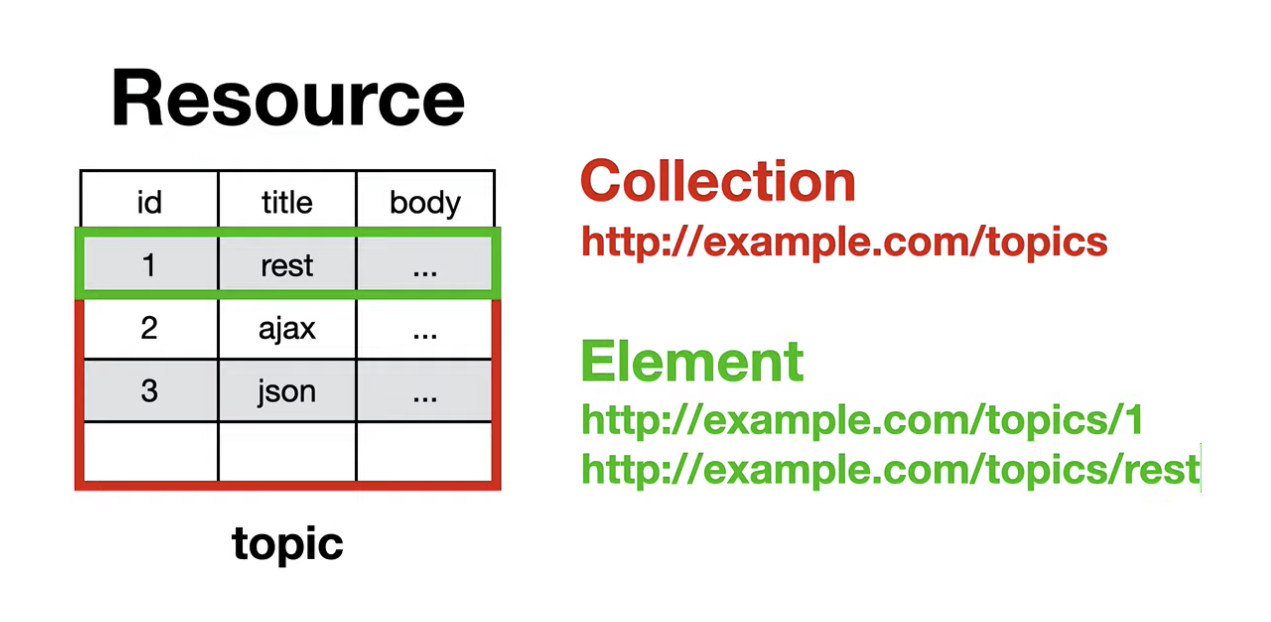
**Rest API**

http의 기능을 최대한 활용해서 서버와 통신할 것을 제안하는 모범사례

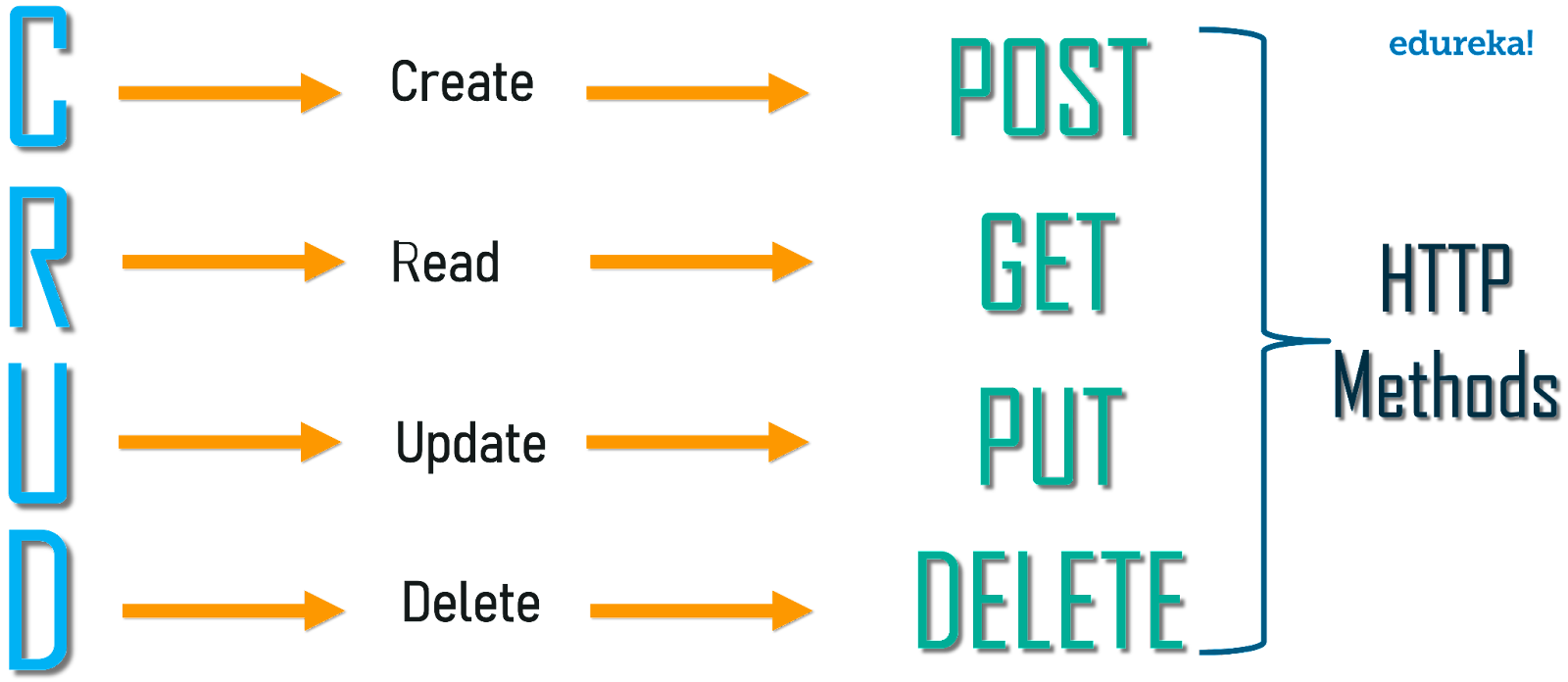
collection: 전체 반환

/id: 해당 엘리먼트만 반환

**Resource**



**Method**



**asynchronous**

**callback**

**정의**

함수의 인자로 전달된 함수

**예제1.**

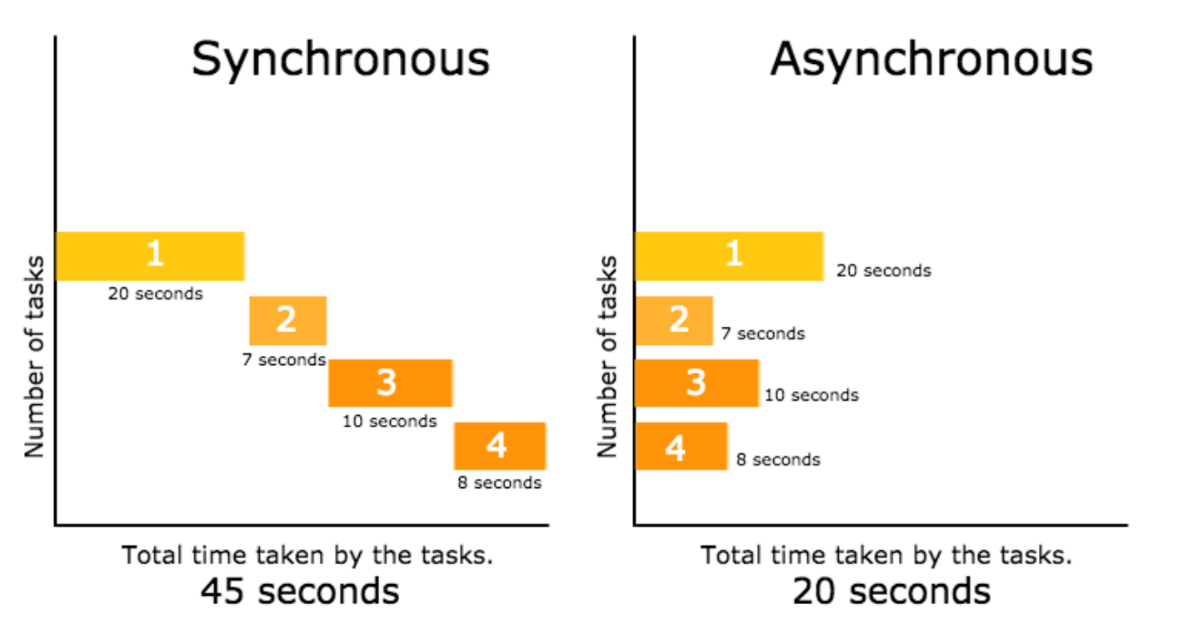
|  |
| --- |
| function callback(){       alert(‘hi’);  }  setTimeout(**callback**, 1000); |

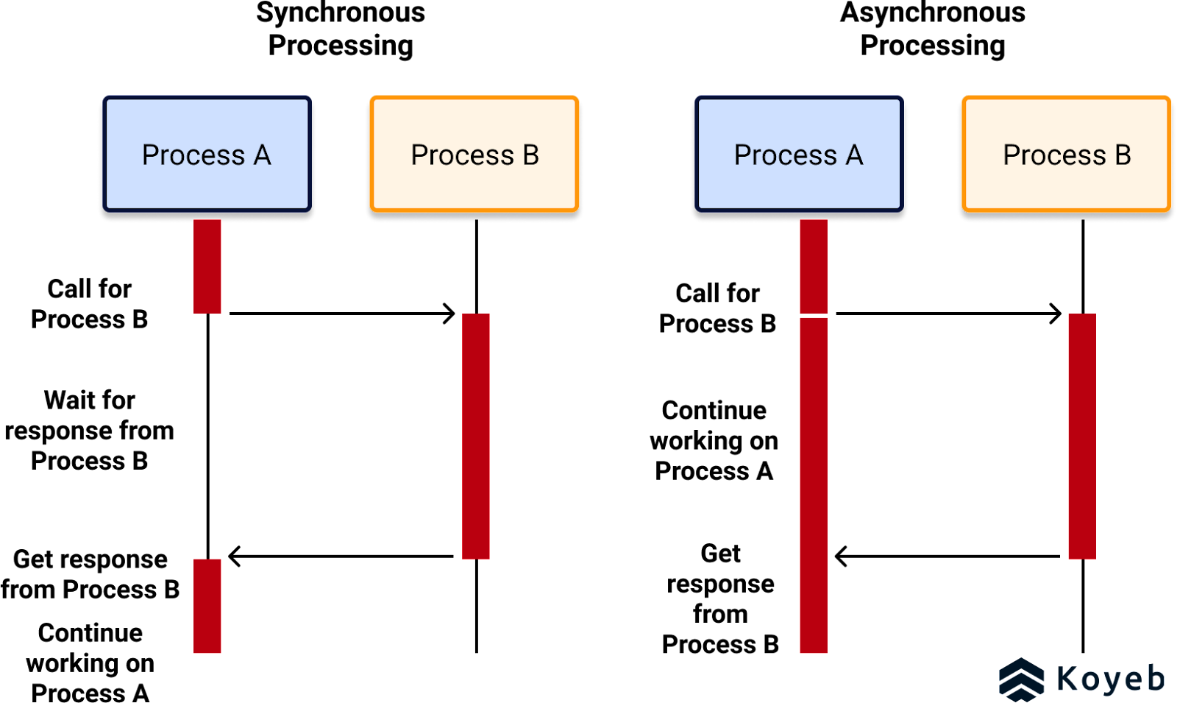
**예제2.**

|  |
| --- |
| let words = [1,2,3,4,5,6,7,8,9];  let newWords = words.filter(**element=>element>5**);  console.log('newWords', newWords); |

**synchronous(동기) VS asynchronous(비동기)**

|  |
| --- |
| console.log(1);  console.log(2);  setTimeout(function(){console.log(3);}, 2000);  console.log(4); |





**promise**

**정의**

비동기방식을 잘 처리하기 위해서 고안된 도구입니다.

fetch api가 promise를 이용합니다.

아래와 같은 리턴값을 가진 함수를 만나면 promise 방식으로 처리해야 합니다.

|  |
| --- |
| **Return value**  A **Promise** that resolves to a **Response object**. |

<https://developer.mozilla.org/en-US/docs/Web/API/fetch#return_value>

**then & catch**

1. promise는 성공했을 때 then의 콜백함수를 호출하고, 실패했을 때 catch의 콜백함수를 호출합니다.
2. then의 콜백함수가 promise를 리턴하면 그 다음 then이 이를 처리할 수 있습니다.

|  |
| --- |
| fetch('/data.json')    .then(function(response) {         return response.json()    })    .then(function(data){      console.log(data)    }) |