2.1 es2015

- this. 바인딩

var relationship1 = {

name: 'zero',

friends: ['nero', 'hero', 'xero'],

logFriends: function () {

var that = this; // relationship1을 가리키는 this를 that에 저장

this.friends.forEach(function (friend) {

console.log(that.name, friend);

});

},

};

relationship1.logFriends();

const relationship2 = {

name: 'zero',

friends: ['nero', 'hero', 'xero'],

logFriends() {

this.friends.forEach(friend => {

console.log(this.name, friend);

});

},

};

relationship2.logFriends();

* 화살표 함수를 사용시, 바깥스코프의 this를 그대로 사용할 수 있음.
* 만약 현재 함수의 this를 사용하고 싶으면 예전방식으로 함수 선언

2.2 프론트엔드 자바스크립트

* AJAX : 비동기적 웹 서비스를 개발할 때 사용하는 기법(asynchronous javascript and XML)
  + 쉽게말해 페이지 이동없이 서버에 요청/응답할 수 잇는 기법
  + 이름에 xml이 들어있지만, 꼭 XML을 사용해야하는 것은 아님. 요즘엔 JSON도 많이 슴
  + 웹사이트 중에서 페이지전환없이 새로운 데이터를 불러오는 기술은 대부분 AJAX 기술을 사용중
  + AJAX 요청은 보통 Jquery나 axios 같은 라이브러리를 이용해서 보냄
* FormData
  + form 태그의 데이터를 동적으로 제어할 수 잇음
  + 주로 AJAX와 함께 사용됨
  + FormData 생성자로 객체를 만듬
    - * const formData = new FormData();
      * formData.append('name', 'zerocho');
        + 키-값 형식의 데이터를 저장할 수 있음.
      * formData.append('item', 'orange');
      * formData.append('item', 'melon');
        + 여러 번 사용하여, 하나의 키에 여러 값을 추가해도 됨.
      * formData.has('item'); // true
      * formData.has('money'); // false;
        + 주어진 키에 해당하는 값이 있는지 여부를 알립니다.
      * formData.get('item'); // orange
        + 주어진 키에 해당하는 값 하나를 가져옴
      * formData.getAll('item'); // ['orange', 'melon'];
        + 해당하는 값 모두를 가져옴.
      * formData.append('test', ['hi', 'zero']);
      * formData.get('test'); // hi, zero
      * formData.delete('test');
      * formData.get('test'); // null
      * formData.set('item', 'apple');
      * formData.getAll('item'); // ['apple'];
  + 위로 만든 formData는 axios로 보낼 수 있다.(약간 JSON 만드는 방법 같음)
    - * (async () => {
      * try {
      * const formData = new FormData();
      * formData.append('name', 'zerocho');
      * formData.append('birth', 1994);
      * const result = await axios.post('https://www.zerocho.com/api/post/formdata', formData);
      * console.log(result);
      * console.log(result.data);
      * } catch (error) {
      * console.error(error);
      * }
      * })();
* encodeURIComponent, decodeURLComponent
  + AJAX 요청을 보낼 때, URL 주소에 한글이 들어가는 경우, 서버가 해석을 못할 수도 있음.
  + 이때는 window 객체의 메서드인 encodeURIComponent를 사용
    - const result = await axios.get(`https://www.zerocho.com/api/search/${encodeURIComponent('노드')})
  + 받는 쪽에서는 decodeURIComponent를 사용
* 데이터속성과 데이터셋
  + 프론트엔드에 데이터를 내려보낼 때 첫번째로 고려할 점은 보안. -> 클라이언트에 민감한 정보를 보내지마라(비밀번호 같은거)
  + 보안과 무관한 데이터들은 자유롭게 내려보냄.
  + 이때 HTML5에도 HTML과 관련된 데이터를 저장하는 공식적인 방법이 존재. -> data attribute
    - * <ul>
      * <li data-id="1" data-user-job="programmer">Zero</li>
      * <li data-id="2" data-user-job="designer">Nero</li>
      * <li data-id="3" data-user-job="programmer">Hero</li>
      * <li data-id="4" data-user-job="ceo">Kero</li>
      * </ul>
      * <script>
      * console.log(document.querySelector('li').dataset);
      * // { id: '1', userJob: 'programmer' }
      * </script>
  + 위와 같이 HTML 태그의 속성으로 data- 로 시작하는 것들을 넣으면 이것이 데이터 속성
  + 화면에는 나타나지 않지만, 웹 어플을 구동하는데 필요한 자료들
  + 이 데이터들을 사용해 서버에 요청을 보내게 됨.
  + 장점 : 자바스크립트로 쉽게 접근 가능.( .dataset) dataset에 데이터를 넣어도 HTML 태그에 반영됨.
  + 이때 자바스크립트에서는 첫 – 을 제외한 – 뒤에 있는 단어들의 첫문자는 대문자로 변환됨