2020320086 최승민 1차과제

1.

(1) addEventListener 메서드를 호출하는 코드가 HTML 문서 끝에 있는 #search-button 요소를 찾으려 하지만, 이 시점에서 DOM이 완전히 로드되지 않아 null을 반환하게 된다. 따라서 null 객체에 addEventListener를 호출하게되어 에러가 발생한다.

(2) -1 DOM이 완전히 로드된 후에 JavaScript가 실행되도록 보장하기 위해 DOMContentLoaded를 사용한다.

ocument.addEventListener("DOMContentLoaded", function () {

document.querySelector("#search-button").addEventListener("click", function () {...

(2) – 2 스크립트 파일을 <body> 태그 끝에 배치하여, HTML 요소가 모두 로드된 후에 스크립트가 실행되도록 한다.

<!-- Posts will be loaded here -->

<script type="text/javascript" src="data.js"></script>

<script type="text/javascript" src="interface.js"></script>

<script type="text/javascript" src="script.js"></script>

</body>...

2.

(1) 각 검색 방식(제목, 내용, 게시물 ID)이 체크박스로 구현되어 있다. 그러나 체크박스는 여러 개를 동시에 선택할 수 있는 요소로, 여러 개의 체크박스를 사용 시, 동시에 여러 검색 방식을 선택할 수 있게 되어 의도와 다르게 동작할 수 있다.

(2) 하나의 검색 방식만을 선택하도록 하기 위해 체크박스 대신 radio 버튼을 사용해야 한다. 라디오 버튼은 동일한 이름을 가진 그룹 내에서 하나의 옵션만 선택할 수 있기 때문이다.

<input type="radio" name="search-option" value="title" id="search-title" />

<label for="search-title">Title</label>

...

3.

(1) 문제의 원인은 board.clear 메서드가 게시글 요소를 제대로 삭제하지 않기 때문이다. board.clear 메서드에서 this.elements는 단일 요소(document.querySelector(".post"))를 가리키고 있어 배열처럼 사용될 수 없다. 따라서, 검색 결과를 지우지 못하고 이전 검색 결과가 계속해서 누적된다.

(2) board.clear 메서드를 수정하여 #post-container 내의 모든 게시글 요소를 삭제하도록 한다. 이를 위해 querySelectorAll을 사용하여 모든 게시글 요소를 선택한 후 루프로 삭제할 수 있다.

```
clear: function () {
    const container = document.querySelector("#post-container");
    while (container.firstChild) {
        container.removeChild(container.firstChild);
    }
},
```

(3) 현재 방식은 첫 번째 .post 요소만 선택해 제대로 삭제하지 못하여 검색 결과가 누적된다. 변경한 방식은 querySelectorAll로 모든 게시글을 삭제해 이전 결과가 남지 않도록 한다.

4.

(1) 문제의 원인은 getPostsById 메서드에서 postId를 문자열로 받는 반면, 데이터에서 post.id는 숫자로 저장되어 있기 때문이다. JavaScript에서는 비교 연산자 "==="를 사용할 때 문자열과 숫자는 다르다고 간주되므로, 필터링이 제대로 이루어지지 않는다.

(2) postId를 숫자로 변환하여 비교한다. 혹은 비교 연산자 "=="를 사용하여 문자열과 숫자를 비교한다

5.

(1) 잘못된 입력은 사용자가 숫자가 아닌 값을 입력했을 때 발생한다. 따라서, 입력 값이 숫자가 아니거나, 빈 문자열일 경우를 확인한다.

(2) alert 메서드를 사용하여 브라우저에 알림을 보낸다.

```
getPostsById: function (postId) {
    if (isNaN(postId) || postId.trim() === "") {
        alert("Please enter a valid numeric Post ID value.");
```

```
        return [];
    }...
```

6.

```
(1) getUserById: function (userId) {
    return data.users.find(user => user.id === userId);
},
```

(2)

```
getUserById: function (userId) {
    for (let i = 0; i < data.users.length; i++) {
        if (data.users[i].id === userId) {
            return data.users[i];
        }
    }
},
```

7. (1) index.html은 웹 어플의 기본 골조를 이루는 html파일이고, CSS, JavaScript 파일을 불러온다. Style.css는 html 파일의 스타일을 정의하고, script.js는 웹의 주요 기능을 구현하는 파일이다. 게시글을 검색하고, 화면 렌더링 등 기능을 한다. Interface.js는 데이터를 다루는 자바스크립트파일로 데이터를 불러오는 메서드들을 포함한다.

(2) 사용자가 URL에 접속하면 브라우저는 index.html을 요청하고 로드하며, 이 파일과 같이 필요한 CSS와 JavaScript 파일들이 로드된다. script.js를 실행하여 board.init 함수로 게시글 데이터를 로드하고 화면에 렌더링한다. 사용자가 검색 옵션을 선택하고 검색어를 입력한 후 검색 버튼을 클릭하면, board.search가 호출되어 검색 결과를 가져온다. 이후 검색 결과가 화면에 표시되어 사용자가 확인할 수 있게된다.

8. (1) @media 문을 사용하여 반응형으로 구현한다.

(2) 맨 아래 pdf 페이지 답안 그림.

9.

```
(1) document.querySelector("#search-input").addEventListener("keydown", function(event) {
  });
```

(2) if (event.key === "Enter") {

const value = document.querySelector("#search-input").value;

const option = document.querySelector(".search-option > input:checked").value;

board.search(value, option);

}

(1)번 {}안에 구현.

10.

(1) span을 사용했기 때문이다. href 속성으로 하이퍼링크를 만들려면 a 태그를 사용해야 한다.

(2) span 요소 대신 a 요소를 생성하고, href 속성을 설정한 후, 이를 부모 요소에 추가한다.

(3) const userLink = document.createElement("a");

userLink.setAttribute("href", "mailto:" + user.email);

userLink.classList.add("post-user");

userLink.innerText = user.name + " (" + user.email + ")";

postDiv.appendChild(userLink);

11. 작성한 코드

(1) a.post-user 선택자를 사용한다.

(2) a.post-user {

color: skyblue;

cursor: default;

text-decoration: none;

}

12. 작성한 코드

button:hover {

background-color: deepskyblue;

color: white;

border: 1px solid;

```
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    font-weight: bold;
}
```

13.

```
@media (max-width: 800px) {
    .post-content {
        overflow: hidden;
        white-space: nowrap;
        text-overflow: ellipsis;
    }
```

14.

this의 스코프 문제로 인해 this.posts와 this.render()에서 this가 예상과 다를 수 있다.

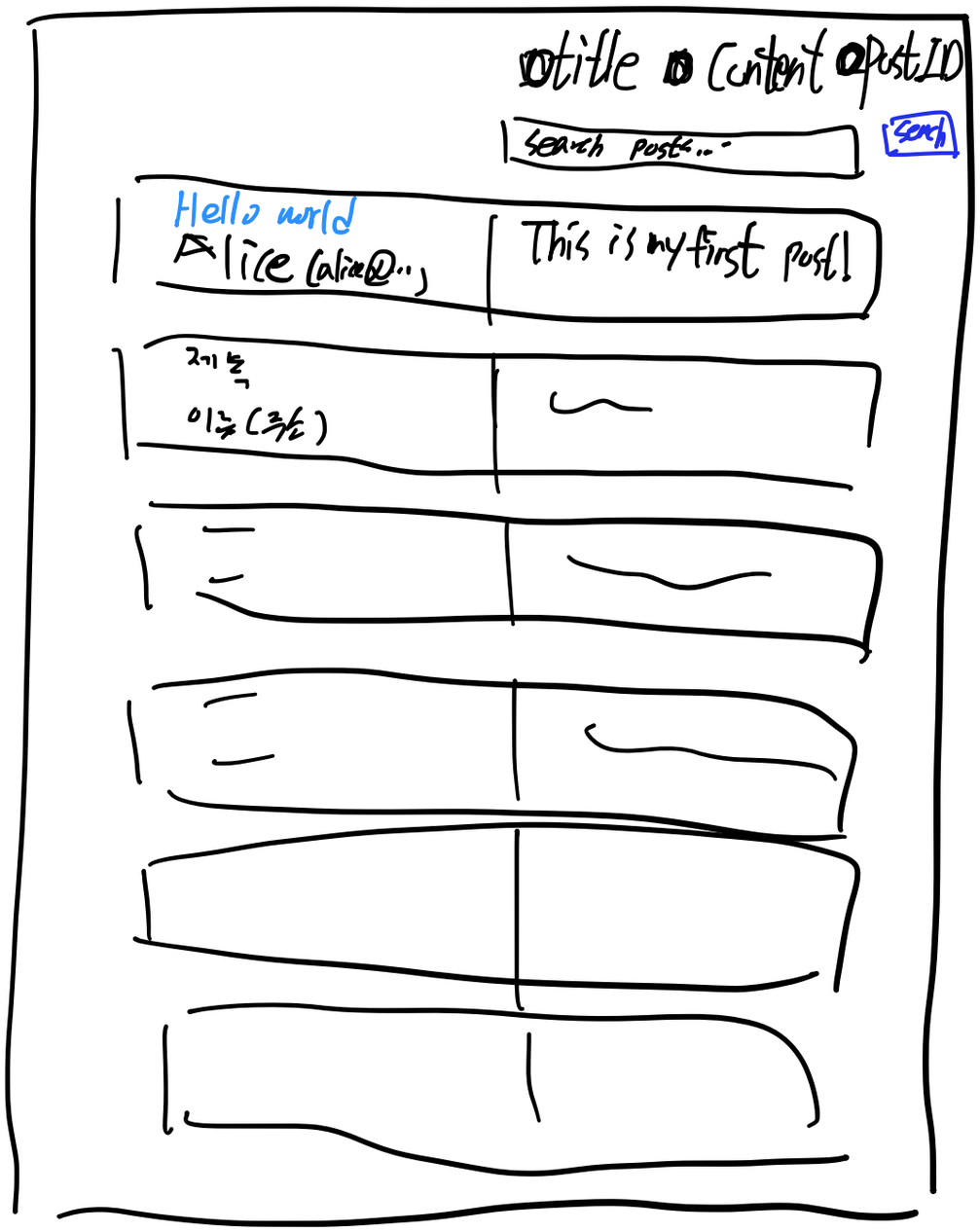
15.

(1) var로 선언한 변수는 재선언, 재할당이 가능하지만, let은 재할당은 가능하지만 재선언이 불가하기에 두 키워드의 차이 때문에 오류가 발생할 수 있다.

(2)

```
search: function (value, option) {
this.clear();
        let posts = [];
        switch (option) {
            case "title": posts = interface.getPostsByTitle(value); break;
            case "content": posts = interface.getPostsByContent(value); break;
            case "id": posts = interface.getPostsById(value); break;
        }this.posts = posts;
        this.render();}
```

와 같이 수정할 수 있다.


title
Content
postID
Search posts...
Search
Hello world
Alice (alice@...)
This is my first post!