2020320054 박민욱 화요일 분반

1-(1): 자바스크립트 파일이 html문서의 dom요소들이 모두 로드되기 전에 실행되기 때문. addEventListener 메소드를 호출하려는 요소가 없어서.

1-(2): script 태그를 body태그의 마지막 부분에 위치시켜 html 문서가 모두 로드된 후에 스크립트가 실행되도록 하여 해결 가능 또는, script 태그에 async나 defer 속성을 활용할 수도 있다. Async 속성을 값 없이 명시하면 js코드가 비동기적으로 실행되고, defer는 속성이 없다면 문서로딩이 완전히 끝난 뒤에 실행됨.

2-(1): 검색 방식은 셋 중 하나로 결정되어야하는데 체크박스를 활용하면 중복 선택이 가능함.

2-(2): 같은 name을 가지게 하여 중복을 허락하지 않을 수 있다.

3-(1): clear 메소드가 this.element 참조를 제거하지 않아서 이전 검색결과가 삭제되지 않기 때문에.

3-(2): this elements를 업데이트해서 새로 검색된 게시글만을 포함하도록 하기

```
clear: function () {
   const container = document.querySelector("#post-container");
   container.innerHTML = '';
   this.elements = [];
},
```

3-(3): 현재방식은 DOM요소를 하나씩 제거하지만, 변경된 방식은 inner HTML을 사용해서 모든 자식 요소를 한 번에 제거함.

4-(1): getPostById 메서드에서 postId는 숫자인데, 입력 값이 문자열이라 검색이 실패함.

4-(2): getPostByID 메서드 내에서 postId를 숫자로 변환 또는 입력값을 받을 때 숫자로 변환하기.

5-(1)

```
if (isNaN(postId) || postId < 1) {
```

```
    alert("Invalid post ID");
    return [];
}
```

5-(2):

```
getPostsById: function (postId) {
    if (isNaN(postId) || postId < 1) {
        alert("Invalid post ID");
        return [];
    }
    return data.posts.filter((post) => post.id === Number(postId));
},
```

6-(1):

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

6-(2):

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

7-(1)

Index.html은 웹 어플리케이션의 구조, 컨텐츠를 정의하고 style.css는 웹 디자인, 스타일을 더 효과적으로 표현하기 위한 파일이다. Script.js는 웹 어플리케이션의 동작을 정의하고 interface.js는 데이터 조작 및 검색기능, data.js는 게시글 데이터를 제공한다.

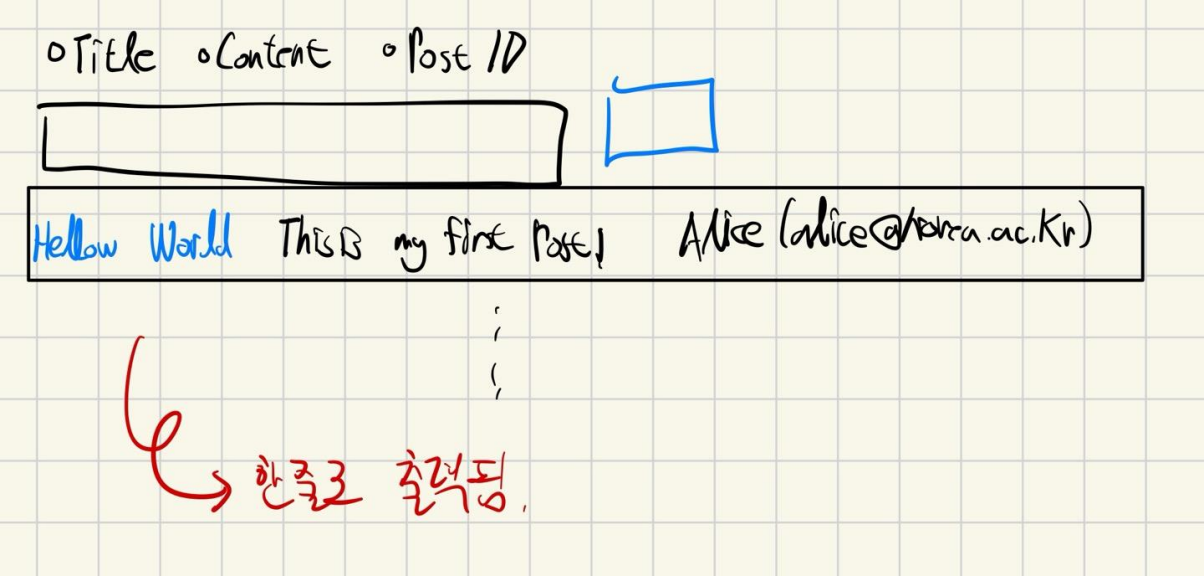
7-(2)

사용자가 브라우저를 통해 접속 -> index.html이 로드되고 CSS와 JS가 로드되는데, script.js의 init이 호출되어 게시글이 화면에 렌더링됨. 이후부터는 사용자가 검색을 통해 게시글 필터링

8-(1)

@media(max-width:768px){}, @media(min-width:769px){}를 사용하여 적용가능

8-(2)



9-(1)

```
document.querySelector("#search-input").addEventListener("keyup", function(event) {
    if (event.key === "Enter") {
        document.querySelector("#search-button").click();
    }
});
```

9-(2)

9-1의 코드에서 enter키를 확인 후 search-button의 클릭 이벤트 발생.

10-(1)

setAttribute는 href 속성을 추가하는데, 앵커 태그에서만 가능하고, span요소에는 href가 적용되지 않기 때문.

10-(2)

span대신 a를 사용해야함.

10-(3)

```
const userLink = document.createElement("a");
userLink.href = "mailto:" + user.email;
userLink.classList.add("post-user");
userLink.innerText = user.name + " (" + user.email + ")";
postDiv.appendChild(userLink);
```

11

```
.post-user {
    color: inherit;
    cursor: default;
    text-decoration: none;
}
```

12

```
button:hover {
    background-color: #0066cc;
    color: #fff;
```

```
    border: 1px solid #005bb5;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    transform: scale(1.05);
}
```

13

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

14

Arrow function은 this 바인딩을 못가지기 때문에 init메소드 내에서 this가 board 객체를 가리키지 못함

15-(1)

Let 키워드는 블록 스코프를 가져서 case 내에서 선언된 변수가 밖에서 접근할 수 없음.

15-(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
```