Problem 1.

(1) Html이 로드되기 전에 자바스크립트에서 html을 참조하기 때문에, addEventListner에서 html 태그를 참조할 수 없어 null을 참조하게 되는 에러가 발생한 것이다.
(2) 먼저 html이 모두 로드된 후 자바스크립트에서 접근하도록 코드를 수정하는 방법으로, body아래에 script를 작성하는 방법이 있다. 두 번째로는 웹브라우저의 모든 구성요소가 로드된 후에 브라우저가 호출하도록 하는 함수인 window.onload 함수를 활용하여, window.onload 함수 내부에 문제가 되는 함수를 넣어주는 방법이 있다.

Problem 2.

(1) 다르게 동작한다. 선택지를 여러 개 고를 수 있다.
(2) Input 태그의 type 속성값을 checkbox가 아닌 radio로 설정한다.

Problem 3.

(1) Clear 메소드에서 this.elements를 잘못 사용하고 있다. This.element는 여러 개의 게시물을 처리해야 하는데 코드에서는 document.querySelector(".post")로 선택된 하나의 요소를 가리키고 있다.
(2) This.element 대신 document.querySelector("post-container")를 사용하여 모든 자식 요소를 제거한다.
(3) 매번 최신 상태 요소를 직접 선택하고 관리하기 때문에 정확한 DOM 조작이 이루어진다.

Problemt 4.

(1) === 연산자가 값 뿐만 아니라 데이터 형식도 비교하기 때문에 나타나는 문제이다. PostId 값이 실제로 다른 타입인데 문자열 형식으로 제공될 경우, 타입 불일치로 인해 검색결과가 나타나지 않을 수도 있다.
(2) 명시적으로 데이터 타입을 일치시키거나, 암시적 형 변환을 활용하여 postId 값을 비교하는 방법이 있다.

Problem 5.

(1) (2)

```
If ( isNaN ( postId ) ) {

    alert ( "postId는 숫자여야 합니다." )

    return [ ];

}
```

Problem 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];
        }
    }
    return null;
}
```

Problem 7.

(1) index.html : 웹 어플리케이션의 기본 구조를 정의한다.
style.css : html 요소들의 스타일을 정의하여 웹 페이지의 디자인과 레이아웃을 설정한다.
script.js : 게시글을 초기화하고 검색 기능을 처리하고, 게시글을 DOM에 렌더링하고, 검색 결과를 갱신할 때 이전 결과를 지우는 등 웹 어플리케이션의 동작을 제어하는 기능을 한다.
interface.js : 데이터를 검색하고 필터링하는 함수, 메서드를 정의한다.
data.js : users, posts 배열을 정의하여 사용자 정보와 게시글 정보를 데이터셋으로 저장한다.

(2) 사용자가 브라우저에서 웹에 접속하면 index.html 파일이 로드된다. index.html의 <head> 태그에서 style.css, data.js, interface.js, script.js 파일이 순서대로 로드된다.

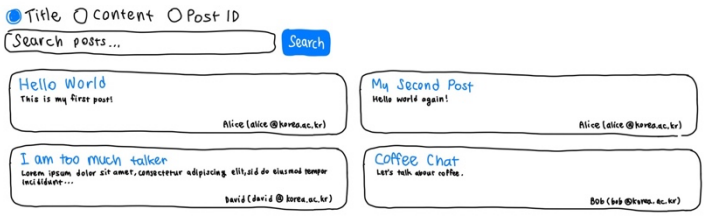
style.css 파일이 로드되면, html 요소에 해당하는 스타일이 적용되어 웹 페이지의 디자인이 완성된다.
data.js 파일이 로드되면 사용자 정보와 게시글 정보가 포함된 data 객체가 정의된다.
interface.js 파일이 로드되면 데이터를 검색하고 필터링해주는 메소드가 정의된다.
script.js 파일이 로드되면 board 객체가 초기화되고, 초기 게시글을 렌더링한다. script.js 파일에서 board.init() 메소드는 interface.getPosts() 메소드를 통해 모든 게시글을 가져와 화면에 표시한다.

이후 사용자가 검색 기능을 사용하면 script.js 에서 eventListener가 board.search()메소드를 호출하고, 이는 다시 clear() 메소드를 호출하여 이전 검색 결과를 제거한 후 검색 결과를 가져오게 한다. 검색 결과는 this.posts에 저장되고 render()를 통해 화면에 표시된다.

Problem 8.

(1) Css 파일에서 특정 화면 크기에 따라 스타일을 적용할 수 있는 @media 기능을 이용한다.

(2)



Problem 9.

(1) (2)

```
document.querySelector("#Search-input").addEventListener("keypress", function(event) {

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

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

    }

});
```

Problem 10.

(1) href 속성이 span 태그에 들어있다.
(2) span 태그 대신 a 태그를 이용하여 코드를 작성해야 한다.
(3) const userSpan = document.createElement("a");
userSpan.classList.add("post-user");
user.Span.innerText = user.name;
userSpan.setAttribute("href", mailto: + user.email);
postDiv.appendChild(userSpan)

Problem 11.

(1) (2)

```
.post-user a {

    color : #666;
    cursor : pointer;

}
```

Problem 12.

(1) (2)

```
Button : hover {

    background-color : #4CAF50;
    color : white;
    border : 2px solid #4CAF50;
    border-radius : 5px;
    box-shadow : 0 4px 8px rgba(0, 0, 0, 0.2);

}
```

Problem 13.

```
@media (max-width : 800px) {

    .post-content {

        overflow : hidden;
        white-space : nowrap;
        text-overflow: ellipsis;

    }

}
```

Problem 14.

init 메소드가 일반함수가 아닌 화살표 함수로 정의되어 있으면 메소드 내부의 this가 자신을 가리키지 않기 때문에 일반 함수로 작성되었을 때와 달리 예기치 않은 동작을 일으킬 수 있다.

Problem 15.

(1) var 은 변수를 선언했을 때 함수 스코프를 따르기 때문에 함수 내에서만 유효하고, 외부에서 접근할 수 없게된다. 반면, let은 블록 스코프를 따르기에 변수가 선언된 중괄호 내에서 유효하고, 외부에서는 접근할 수 된다. 이렇게 var과 let의 스코프 규칙이 다르기 때문에 var 키워드를 let 키워드로 변경하면 오류가 발생할 수 있다.

(2)
```
search(value, option) {
    this.clear( );
    let posts;
    switch(option) {
        case “title” :
            posts = interface.getPostsByTitile(value);
            break;
        case “content” :
            posts = interface.getPostsByTitle(value);
            break;
        case “ id” :
            posts = interface.getPostsById(value);
            break;

    }

    this.posts = posts;
    this.render( );
};
```