2024 KWEB 준회원 평가시험 (Front-end session)

학번	2020320056	이름	강승균	분반	목

[개요]

- 일시: 2024. 06. 21. (금) / 18:00 ~ 19:00 (60분)
- 문항: 15문항, 총 100점

[안내사항]

- 총 시험지는 10장으로 구성되어 있으며, 문제 부분과 답안 작성 부분이 분리되어 있습니다.
- 답안은 8~10 페이지의 〈답안〉 란을 활용해 자유롭게 작성하시면 됩니다.
- 이 시험은 Closed-book 시험으로, 시험지 외의 다른 자료는 참고하실 수 없습니다.
- 정확한 성취 확인을 위해 시험 시간에 비해 문항의 수가 많습니다. 잘 아는 문항부터 푸시는 것을 권장합니다.

Code 1. index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>KWEB Board</title>
    <link type="text/css" rel="stylesheet" href="style.css" />
    <script type="text/javascript" src="data.js"></script>
    <script type="text/javascript" src="interface.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <div id="search-container" class="container">
      <div class="search-option">
        <input type="checkbox" value="title" />
        <a>Title</a>
      </div>
      <div class="search-option">
        <input type="checkbox" value="content" />
        <a>Content</a>
      </div>
      <div class="search-option">
        <input type="checkbox" value="id" />
        <a>Post ID</a>
      </div>
      <div id="search">
        <input type="text" id="search-input" placeholder="Search posts..." />
        <button id="search-button">Search</button>
      </div>
    </div>
    <div id="post-container" class="container">
      <!-- Posts will be loaded here -->
    </div>
  </body>
</html>
```

Code 2. style.css

```css
.container {
  padding: 10px;
}
.search-option {
  display: inline-block;
  padding: 0 10px 10px 0;
  font-size: 16px;
}
#search-input {
  width: 300px;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 8px;
  font-size: 15px;
}
button {
  padding: 12px;
  background-color: dodgerblue;
  color: white;
  font-size: 15px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
}
.post {
  padding: 15px;
  margin-bottom: 15px;
  background-color: white;
  border: 1px solid #ddd;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
.post-title, .post-content, .post-user {
  display: block;
  font-size: 16px;
}
.post-title {
  color: dodgerblue;
  font-weight: bold;
}
.post-content {
  color: #333;
}
.post-user {
  color: #666;
  text-align: right;
}
@media (min-width: 768px) {
  #search {
    display: inline-block;
  }
  .post-title, .post-content, .post-user {
    display: inline;
    padding-right: 10px;
  }
  .post-title {
    font-size: 20px;
  }
}
```

Code 3. `script.js`

```
const board = {
  posts: [],
  elements: document.querySelector(".post"),
  init: function () {
    this.posts = interface.getPosts();
    this.render();
  },
  search: function (value, option) {
    this.clear();
    switch (option) {
      case "title":
        var posts = interface.getPostsByTitle(value); break;
      case "content":
        var posts = interface.getPostsByContent(value); break;
      case "id":
        var posts = interface.getPostsById(value); break;
    }
    this.posts = posts;
    this.render();
  },
  clear: function () {
    while (this.elements.length) {
      this.elements[0].remove();
    }
  },
  render: function () {
    this.posts.forEach((post) => {
      const postDiv = document.createElement("div");
      postDiv.classList.add("post");

      const titleSpan = document.createElement("span");
      titleSpan.classList.add("post-title");
      titleSpan.innerText = post.title;
      postDiv.appendChild(titleSpan);

      const contentSpan = document.createElement("span");
      contentSpan.classList.add("post-content");
      contentSpan.innerText = post.content;
      postDiv.appendChild(contentSpan);

      const user = interface.getUserById(post.userId);
      const userSpan = document.createElement("span");
      userSpan.classList.add("post-user");
      userSpan.innerText = user.name + " (" + user.email + ")";
      postDiv.appendChild(userSpan);

      const container = document.querySelector("#post-container");
      container.appendChild(postDiv);
    });
  },
};

document.querySelector("#search-button").addEventListener("click", function () {
  const value = document.querySelector("#search-input").value;
  const option = document.querySelector(".search-option > input:checked").value;
  board.search(value, option);
});

board.init();
```

Code 4. interface.js

```
const interface = {
  getPosts: function () {
    return data.posts;
  },
  getPostsByTitle: function (title) {
    return data.posts.filter((post) => post.title.includes(title));
  },
  getPostsByContent: function (content) {
    return data.posts.filter((post) => post.content.includes(content));
  },
  getPostsById: function (postId) {
    return data.posts.filter((post) => post.id === postId);
  },
  getUserById: function (userId) {
    return /* TODO */;
  },
};
```

Code 5. data.js

```
const data = {
  users: [
    { id: 1, name: "Alice", email: "alice@korea.ac.kr" },
    { id: 2, name: "Bob", email: "bob@korea.ac.kr" },
    { id: 3, name: "Charlie", email: "charlie@korea.ac.kr" },
    { id: 4, name: "David", email: "david@korea.ac.kr" },
    { id: 5, name: "Eve", email: "eve@korea.ac.kr" },
  ],
  posts: [
    { id: 1, userId: 1, title: "Hello World", content: "This is my first post!" },
    { id: 2, userId: 1, title: "My Second Post", content: "Hello world again!" },
    {
      id: 3,
      userId: 4,
      title: "I am too much talker",
      content:
        "Lorem ipsum dolor sit amet, consectetur adipiscing elit, \
        sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. \
        Integer quis auctor elit sed vulputate. Lorem ipsum dolor sit amet. \
        Quam nulla porttitor massa id. Metus vulputate eu \
        scelerisque felis imperdiet proin fermentum leo vel. \
        Tristique nulla aliquet enim tortor. Massa massa \
        ultricies mi quis hendrerit dolor magna eget est.",
    },
    { id: 4, userId: 2, title: "Coffee Chat", content: "Let's talk about coffee." },
    { id: 5, userId: 3, title: "Tech Trends", content: "Latest in tech July 2024." },
    { id: 6, userId: 3, title: "Morning Routines", content: "How do you start your day?" },
    { id: 7, userId: 3, title: "Book Club", content: "Discussing 'The Great Gatsby'." },
    { id: 8, userId: 2, title: "Favorite Recipes", content: "Share your best recipes." },
    { id: 9, userId: 3, title: "Fitness Goal", content: "Set fitness goals for the year." },
    { id: 10, userId: 5, title: "Music Reviews", content: "Reviewing the latest albums." },
  ],
};
```

[Part A. 웹 어플리케이션 구현]

여러분은 KWEB 회원들이 사용할 수 있는 간단한 게시판 형태의 웹 어플리케이션을 구현하고 있습니다. 제시된 코드들이 지금까지 구현된 부분입니다. 여러분은 테스트 과정에서 몇 가지의 문제점을 발견했으며, 하나씩 그 원인을 분석하고 해결해 보고자 합니다.

Problem 01. `index.html` 파일의 Line 8~10에서는 별도로 작성된 스크립트 파일들을 로드하고 있습니다. 하지만 여러분은 스크립트 파일이 제대로 작동하지 않고, 다음과 같은 오류가 발생하는 것을 발견했습니다. (7점)

```
TypeError: Cannot read properties of null (reading 'addEventListener')
```

(1) 문제의 원인은 무엇인가요? (3점)

(2) 해결 가능한 방법을 2가지 이상 작성하세요. (4점)

Problem 02. `index.html` 파일의 Line 14~25에서는 게시글 검색 시 검색 방식을 선택하기 위한 HTML element들의 구조를 표현하고 있습니다. 하지만 여러분은 검색 방식 선택 기능이 의도했던 것과 다르게 동작한다는 사실을 발견했습니다. (7점)

(1) 검색 방식 선택이 어떻게 원래 의도와 다르게 동작하나요? (3점)

(2) 의도했던 것처럼 기능이 동작하도록 하기 위해서 수정해야 하는 부분을 설명해 주세요. (4점)

Problem 03. `script.js` 파일의 Line 3에서는 DOM에서 제공하는 메서드를 이용해 HTML 요소를 조회하고 있습니다. 하지만 여러분은 게시글을 검색할 때 이전 검색 결과가 삭제되지 않고 계속해서 누적된다는 사실을 발견했습니다. (8점)

(1) 문제의 원인은 무엇인가요? (3점)

(2) 해결 가능한 방법을 작성하세요. (2점)

(3) 현재 방식과 (2)에서 변경한 방식 간의 또 다른 차이점에 대해 작성하세요. (3점)

Problem 04. `interface.js` 파일의 Line 11~13에서는 게시글의 ID로 게시글을 검색하는 기능을 제공하기 위해 사용되는 메서드를 정의하고 있습니다. 여러분은 다른 검색 방식은 의도한 대로 정확히 동작하지만, ID 값으로 검색할 때는 값을 정확히 입력했음에도 검색 결과가 나타나지 않는다는 문제를 발견했습니다. (7점)

(1) 문제의 원인은 무엇인가요? (3점)

(2) 해결 가능한 방법을 2가지 이상 작성하세요. (4점)

Problem 05. 여러분은 Problem 04에서 발생한 문제를 해결하면서, 게시글의 ID로 게시글을 검색하는 기능에서 사용자가 잘못된 입력을 했을 때 오류 메시지를 발생시켜야 하겠다는 생각을 하게 되었습니다. `interface.js` 파일의 Line 11~13을 수정해서 이 기능을 구현해 봅시다. (단, Problem 04에서 발생한 문제는 해결되었다고 가정합니다.) (6점)

(1) 먼저, 사용자의 잘못된 입력을 확인하는 조건을 작성하고, (3점)

(2) 조건에 해당할 때, BOM에서 제공하는 메서드를 이용해 브라우저에 알림을 보내세요. (3점)

Problem 06. `interface.js` 파일의 Line 14~16에서는 단일 사용자를 사용자 ID로 조회하는 메서드를 정의하고 있습니다. 여러분은 아직 이 메서드를 작성하지 않았습니다. 이제, 다음 사실을 이용해 메서드를 구현해 봅시다. (6점)

JavaScript에서 제공하는 Standard built-in object 중 하나인 Array Object는 다양한 instance 메서드를 제공합니다.
주요 메서드 목록: length, concat, filter, find, findIndex, forEach, includes, join, map, pop, push, reverse, slice, sort

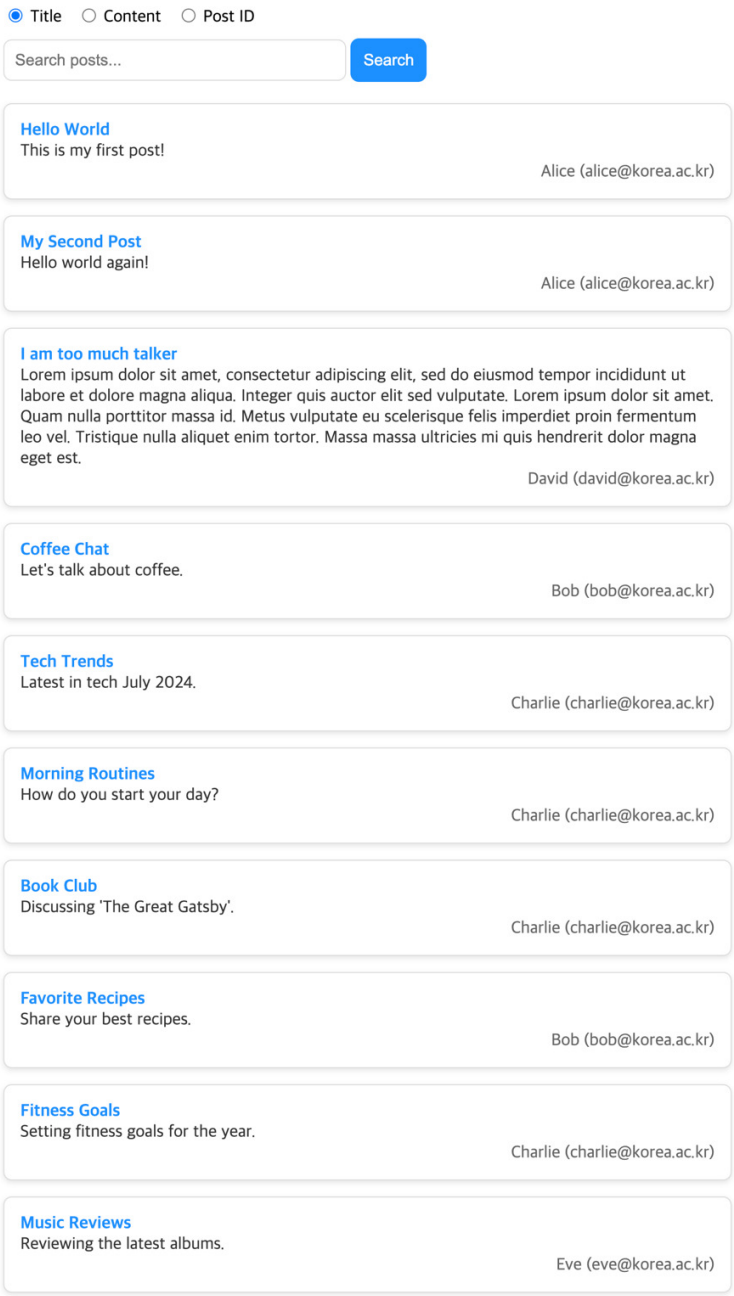
(1) Array Object의 instance 메서드를 사용해서 메서드를 구현해 보세요. (3점)

(2) Array Object의 instance 메서드를 사용하지 않고 메서드를 구현해 보세요. (3점)

[Part B. **웹 어플리케이션 분석**]

축하합니다! 여러분은 이제 대부분의 문제점을 해결했습니다. 이제 여러분이 만든 웹 어플리케이션을 확인해 봅시다.

Figure 1. 웹 어플리케이션 @ width: 720px



Problem 07. 제작된 웹 어플리케이션은 다섯 개 파일 (`index.html`, `style.css`, `script.js`, `interface.js`, `data.js`)로 구성되어 있습니다. 여러분은 이 웹 어플리케이션이 동작하는 원리에 대해 완벽히 알고자 합니다. (8점)

(1) 각 파일이 웹 어플리케이션에서 어떤 역할을 하는지 설명하고, (5점)

(2) 사용자가 이 웹 어플리케이션에 접속해 정보를 확인하기까지의 과정을 간단히 소개해 보세요. (3점)

Problem 08. Figure 1은 브라우저의 너비가 720px일 때 index.html 페이지를 조회한 사진입니다. 여러분은 브라우저의 너비가 1080px일 때의 상태도 능숙하게 추론할 수 있습니다. (10점)

(1) 웹 페이지를 반응형으로 디자인하기 위한 방법을 간단히 소개해 보세요. (2점)

(2) 브라우저의 너비가 1080px일 때 index.html 페이지를 조회한 결과를 그려 보세요. (8점)

(긴 글자나 중복되는 요소는 반드시 완벽하게 표현하지 않아도 됩니다.)

[Part C. **웹 어플리케이션 개선**]

웹 어플리케이션에서 사용자에게 보여지는 영역을 개선하는 것은 Front-end 분야 개발의 핵심입니다. 여러분은 이제 사용자 측면에서 부족한 부분을 조금 더 개선해 보려고 합니다.

Problem 09. 사용자의 편의를 위해 Text Input에서 Enter 키를 누를 때 자동으로 검색이 이루어지도록 하는 기능을 구현해 봅시다. 아래는 사용자가 브라우저에서 Enter 키를 누를 때 발생하는 이벤트입니다. (8점)

```
KeyboardEvent {isTrusted: true, key: 'Enter', code: 'Enter', keyCode: 13, location: 0, …}
```

(1) 적절한 HTML 요소, 적절한 이벤트를 선택해서 Event Listener 추가하고, (4점)
(2) 이벤트 객체를 받아 누른 키가 Enter인지 확인해서 검색을 실행하면 됩니다. (4점)

Problem 10. 이번에는 각 게시글의 사용자 정보 부분에 이메일을 보내는 하이퍼링크를 추가하려고 합니다. 이메일을 전송하기 위한 리소스 주소는 `mailto:<address>`와 같이 표현합니다. 하지만 여러분이 `script.js`의 Line 45에 다음과 같은 코드를 추가해도 하이퍼링크가 동작하지 않습니다. 하이퍼링크가 제대로 동작하도록 해 봅시다. (8점)

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

(1) 위 코드가 동작하지 않는 원인이 무엇인가요? (2점)
(2) 위 코드 대신 `script.js`의 Line 45에 어떤 의미의 코드를 추가해야 하이퍼링크가 제대로 동작할까요? (2점)
(3) 실제로 코드를 작성해 봅시다. 단, `innerHTML` 메서드는 사용하지 않아야 합니다. (4점)

Problem 11. Problem 10에서 하이퍼링크를 추가하니, 브라우저 엔진에 의해 강제로 삽입된 다음 CSS가 적용된 것을 확인할 수 있었습니다. 사용자가 보는 화면이 원래와 동일해질 수 있도록 CSS 코드를 추가로 작성해 봅시다. (5점)

```
{ color: -webkit-link; cursor: pointer; text-decoration: underline; }
```

(1) 적절한 선택자를 사용해서, (2점)
(2) 적절한 CSS 속성들을 적용해 보세요. (3점)

Problem 12. 버튼 위에 커서를 올렸을 때, 커서 모양이나 버튼의 색이 변경되는 디자인은 사용자 경험을 개선합니다. 커서 모양이 변경되는 디자인은 이미 적용되어 있습니다. 이제 버튼의 색이 변경되는 디자인을 적용해 봅시다. (5점)
(1) 적절한 가상 클래스 선택자를 사용해서, (2점)
(2) 여러분이 선호하는 CSS 속성을 5가지 이상 적용해 보세요. (3점)

Problem 13. 게시글 본문과 같은 긴 텍스트의 경우에는, 내용 전체를 보여주지 않고 넘치는 부분을 줄임표로 대체하기 위해 다음과 같은 속성을 적용하기도 합니다. 브라우저의 크기가 800px 미만일 때만 이 방식을 적용해 봅시다. (4점)

```
{ overflow: hidden; white-space: nowrap; text-overflow: ellipsis; }
```

Problem 14. `script.js` 파일의 Line 4~7에서는 게시판을 초기화하기 위한 메서드가 정의되어 있습니다. 이 메서드를 익명 함수로 다음과 같이 작성하였을 때, 발생할 수 있는 문제점을 설명해 보세요. (4점)

```
init: () => { this.posts = interface.getPosts(); this.render(); },
```

Problem 15. `script.js` 파일의 Line 8~25에서는 검색 결과에 따라 게시판을 갱신하기 위한 메서드가 정의되어 있습니다. 여러분은 ES5 표준에 맞게 작성된 이 메서드에서 `var` 키워드를 제거해 리팩터링 하려고 합니다. (7점)
(1) 단순히 `var` 키워드를 `let` 키워드로 변경하면 오류가 발생하게 됩니다. 이유를 설명해 보세요. (3점)
(2) 이 메서드를 `var` 키워드를 사용하지 않고 다시 작성해 보세요. (4점)

학번	2020320056	이름	강승균	분반	목

〈 답안 〉

문제 1

(1) 스크립트 파일이 로드되기 전에 DOM 요소가 아직 생성되지 않았기 때문에 발생하는 오류이다.

(2)

1. script 파일을 <body> 태그 아래에 배치하여 DOM이 완전히 로드된 후에 스크립트가 실행되도록 한다.
2. 문제가 되는 함수를 window.onload 함수 내부에 넣어 DOM이 완전히 로드된 후에 스크립트가 실행되도록 한다.

문제 2

(1) 체크박스이기 때문에 옵션을 여러 개 선택할 가능성이 존재한다.

(2) 검색 방식 선택 체크박스를 라디오 버튼으로 변경하고 name을 통일하여 하나의 옵션만 선택할 수 있게 만든다.

문제 3

(1) 검색 결과를 지우는 메서드 clear는 단일 html 요소를 반환하기에 리스트 전체를 대상으로 삭제하는 코드가 필요하다.

(2)

```
clear: function () {
 const postElements = document.querySelectorAll(".post");
 postElements.forEach(element => element.remove());
}
```

(3) 모든 요소를 삭제하여 게시글을 검색하기 전 초기 상태에는 게시글이 보이지 않는다.

문제 4

(1) getPostsById 메서드에서 postId는 문자열이고 게시글의 id는 숫자인데 '==='로 둘을 비교하고 있다.

(2)

1. '=='를 사용하여 비교한다.
2. postid를 숫자로 변경한다. Number(postId)

문제 5

(1) 사용자가 숫자가 아닌 값 혹은 아무 값도 입력하지 않을 때 오류 메시지를 발생시킨다.

(2)

```
getPostsById: function (postId) {
 if (!postId || isNaN(postId)) {
 alert("alert");
 return [];
 }
}
```

문제 6

(1)

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

(2)

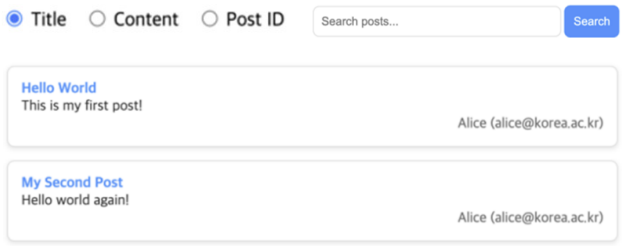
```
getUserById: function (userId) {
 for (let i = 0; i < data.users.length; i++) {
  if (data.users[i].id === userId) {
   return data.users[i];
  }
 }
return null;
}
```

학번	2020320056	이름	강승균	분반	목

〈 답안 〉

문제 7
(1) index.html: 기본적인 페이지 골격으로 css, js 파일을 불러온다.
Style.css: 페이지의 스타일(폰트, 색상 등)을 정의한다.
Script.js: search 버튼을 클릭할 때 검색 결과를 보여주기 위한 기능이다.
Interface.js: search 기능을 위해 필요한 데이터를 가져온다.
Data.js: post, user 정보가 담겨있다.
(2) 사용자가 웹에 접속하면 브라우저는 index.html와 함께 css, js 파일들을 로드한다. 사용자가 검색 옵션과 키워드를 입력한 후 검색 버튼을 클릭하면 검색 결과가 화면에 표시된다.

문제 8
(1) @media를 사용한다.
(2)



문제 9
```
document.querySelector("#search-input").addEventListener("keydown", 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);
 }
});
```

문제 10
(1) span을 사용했기 때문이다.
(2) span 대신 <a> 태그를 추가하고 href 속성을 설정한다.
(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);
```

문제 11
```
a.post-user {
  color: #666;
  cursor: default;
  text-decoration: none;
}
```

학번	2020320056	이름	강승균	분반	목

〈 답안 〉

문제 12

```
button:hover {
  background-color: red;
color: black;
font-size: 20px;
border: 1px solid;
border-radius: 10px;
}
```

문제 13

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

문제 14

=>를 사용하면 함수 내부에서 사용된 this는 init 메서드를 호출하는 객체가 아닌, 화살표 함수가 정의된 상위 스코프의 this를 참조하게 된다.

문제 15

(1) let은 block-scope이기 때문에 선언된 switch 블록 안에서만 접근이 가능하다.

(2)

```
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;
  }
```

로 수정