학번 2021320025	이름 신태현	분반 화

〈 답안 〉

01.

(1) DOM이 파싱되기 전에 script.js가 먼저 로드돼서 #search-button을 script.js에서 찾을 수 없음.

(2) i)script.js의 시작에서 Document.addEventListener('DOMContentLoaded',(event)=>{});추가

ii) index.html의 8-10 line의 src="" 바로 뒤에 defer 추가

02.

(1) checkbox이기 때문에 3개 다 고르는 경우가 생긴다.

(2) type="checkbox"를 "radio"로 바꾸고, 각각 name="(같은 이름)"을 추가해준다.

03.

(1) script.js의 clear의 내부가 작동하지 않는다.

(2) this.elements=document.querySelectorAll(".post");

```
while (this.elements.length){
this.elements[0].remove();
this.elements=document.querySelectorAll(".post");
}
```

(3)기존 방식과 다르게 querySelectorAll로 모든 post를 선택해서 삭제할 수 있게 된다. 이렇게 되면 검색 결과가 누적되지 않는다.

04.

(1)postId는 문자열로 받고, data의 post.id는 정수로 저장되어 있어 '==='이 성립하지 않는다.

(2)'=='로 바꾼다. 또는 postId를 Number(postId)로 변환해준다.

학번 2021320025	이름 신태현	분반 화

〈 답안 〉

05.

(1) 자연수가 아닌 값을 입력하면 오류 메시지를 발생시킨다.

(2) getPostsById: function (postId) {
if(isNaN(postId)||Number(postId)<=0)alert("invalid ID");
return data.posts.filter((post) => post.id == postId);
}

06.

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

07.

(1)index.html: 기초적인 뼈대를 담당하는 파일로써 간단한 텍스트가 적혀 있거나 어느 정도 프레임을 잡아준다. Css, js 파일과 호환도 담당한다.

style.css: 디테일한 디자인을 담당한다.

Script.js: 데이터들의 search, clear, render 등을 정의한다. Search 버튼을 클릭할 때의 반응도 구현한다.

Interface.js: script.js에서 search할 때의 값을 return해준다.

Data.js: User, Post 등의 주요 데이터가 저장되어있다.

(2)사용자가 브라우저를 통해 웹에 접속하면 브라우저는 index.html과 css, js 파일들을 호출한다. 이 과정에서 script.js를 통해 render와 search, clear등이 이루어지는데 script.js는 필요할 때마다 interface.js를 호출하고, interface.js는 필요할 때마다 data.js를 적절히 호출한다.

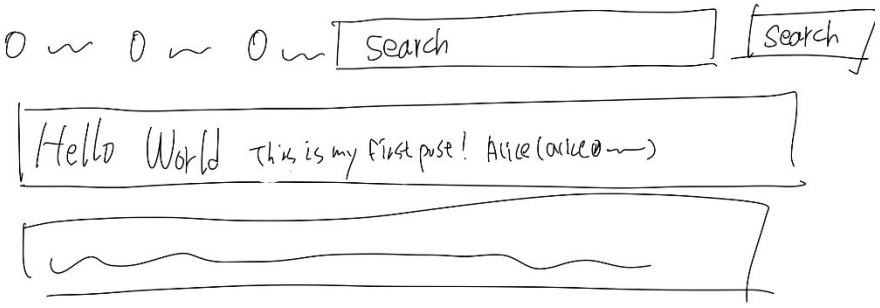
학번 2021320025	이름 신태현	분반 화

〈 답안 〉

08.

(1) css파일에서 @media를 이용하여 여러 조건을 구현한다.

(2)



제목 폰트크기, Search칸 위치
, post내의 개행여부만 바뀜.

학번 2021320025	이름 신태현	분반 화

〈 답안 〉

09.

(1)document.querySelector(“#search-button”).addEventListener(“keypress”,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);

}

10.

(1) span을 써서 동작하지 않는다.

(2) span 대신 a를 이용한다. 그리고 mailLink.callssList.add를 통해 css와 연동해주고, mailLink.innerText를 통해 시각화해준다.

(3)

Const mailLink=document.createElement(“a”);

mailLink.setAttribute(“href”,”mailto:”+user.email);

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

mailLink.innerText=`${user.name}($user.email))`;

postDiv.appendChild(mailLink);

11.

(1)a.post-user

(2) a.post-user{

Color:#666;

Cusor:default;

text-decoration:none;

학번 2021320025	이름 신태현	분반 화

〈 답안 〉

12.

(1)button:hover

(2) button:hover{

Background-color:666;

Color:dodgeblue;

Font-size:16px;

Font-weight:bold;

Border: 1px solid #ddd;

}

13.

```
@media (max-width:799px){
  .post-content{
    Overflow:hidden;
    White-space:nowrap;
    Text-overflow:ellipsis;
    }
```

14. =>를 이용하면 this가 board를 가리키지 않는다.

15.

(1) let은 block-scope여서 switch 밖에서는 접근할 수 없다.

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