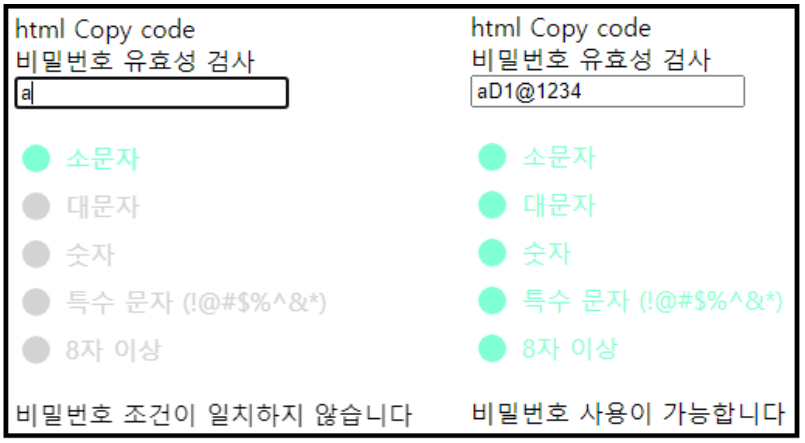
[11/21] Password validation, Event Propagation

1) Password validation

- 결과화면



Q) 모든 비밀번호 조건이 맞을시 표시할 것

1-2) if문을 통한 설정

- html

```
<div><p class="passok">비밀번호 사용이 가능합니다</p></div>
<div><p class="passno">비밀번호 조건이 일치하지 않습니다</p></div>
```

- css

```
.passok, .passno{
    display: none;
}
```

- java
  - inputElem.addEventListener('input', (e) => {
    const password = e.target.value;

```
const ok = document.querySelector('.passok');
const no = document.querySelector('.passno');
if(isValidLowercase(password)==true &&
    isValidUppercase(password)==true &&
    isValidNumber(password)==true &&
    isValidSpecial(password)==true &&
    isValidCaracterLength(password)==true){
    ok.style.display="block"
    no.style.display="none"
}
else{
    ok.style.display="none"
    no.style.display="block"
}
```

위 이미지의 코드는 inputElem.~ 내부에 작성됨

}

  - ==true 생략가능, 이미 3항 연산자를 통해 true 값들만이 (password)에서 도출됨

1-3) 3항 연산자를 통한 설정

- html

```
<li class="corrElem"><span class="label">비밀번호 충족</span> </li>
```

- java

```
const corrElem = document.querySelector('.corrElem');
```

```
isValidLowercase(password) && isValidUppercase(password) &&
isValidNumber(password) && isValidSpecial(password) &&
isValidCaracterLength(password)
=== true ? corrElem.classList.add('active') : corrElem.classList.remove('active');
```

1-4) getElementById

- html

```
<p id="checkPass">check</p>
```

- java

```
const checkPass = document.getElementById('checkPass');
```

```
if (isValidCaracterLength(password) && isValidSpecial(password) &&
 isValidNumber(password) && isValidUppercase(password) && isValidLowercase(password)) {
    console.log("T")
    checkPass.innerHTML = "유효성 검사를 충족합니다.";
} else {
    console.log("F")
    checkPass.innerHTML = "유효성 검사를 충족하지 못합니다.";
}
```

1-5) getElementsByClassName

- html

```
<p class="checkPass">check</p>
```

- java

```
const checkPass = document.getElementsByClassName('checkPass');
```

```
if (isValidCaracterLength(password) && isValidSpecial(password) &&
 isValidNumber(password) && isValidUppercase(password) && isValidLowercase(password)) {
    console.log("T")
    checkPass[0].innerHTML = "유효성 검사를 충족합니다.";
} else {
    console.log("F")
    checkPass[0].innerHTML = "유효성 검사를 충족하지 못합니다.";
}
```

- ByClassName 은 배열로 받아오기 때문에 배열을 지정 해줘야 함

1-6) 톺아보기

- inputElem.addEventListener

```
inputElem.addEventListener('input', (e) => {
    const password = e.target.value;
    isValidLowercase(password) ? lowercaseElem.classList.add('active') : lowercaseElem.classList.remove('active');
    isValidUppercase(password) ? uppercaseElem.classList.add('active') : uppercaseElem.classList.remove('active');
    isValidNumber(password) ? numberElem.classList.add('active') : numberElem.classList.remove('active');
    isValidSpecial(password) ? specialElem.classList.add('active') : specialElem.classList.remove('active');
    isValidCaracterLength(password) ? characterLengthElem.classList.add('active') : characterLengthElem.classList.remove('active');
```

  - 해당 문단중 발췌

```
const inputElem = document.querySelector('input');
```

```
inputElem.addEventListener('input', (e) => {
    const password = e.target.value;
```

  - inputElem.addEventListner('input', (e) => { const password = e.target.value; …이하 중략}
    - inputElem. : HTML 의 input 태그
    - inputElem.addEventListner('input',(e) : input 태그에 input event 발생시
      - input : input 이벤트(O) , input tag (X)
    - const password = e.target.value
      - e.target : 현재의 e.target 은 password 를 의미
      - e.target.value : input한 value 값
      - password = e.target.value : input 된 value 의 값을 password에 적용

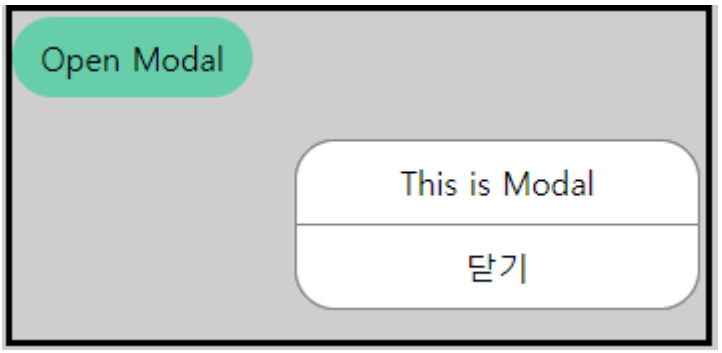
- 정규표현식[1] : /패턴/ 의 형태
  - g : 전역변수 / gi :

```
const isValidNumber = (password) => {
    return  password.search(/[0-9]/g) >= 0;
}
const isValidSpecial = (password) => {
    return password.search(/[~!@#$%^&*()_+|<>?:{}`]/gi) >= 0;
}
```

[1] Regular expressions - mdn web docs
자주 사용하는 정규 표현식 (Regular Expression) 정리

2) Event Propagation : createElement과 Modal 에 대한 이해

- 실행 화면



  - Open Modal 을 클릭시 This is Modal 이라는 새로운 Modal 창이 등장
  - 화면 바깥(회색영역)을 클릭해도 닫기 버튼이 적용, 더블 클릭시 닫아짐

2-1) createElement()

```
var a = document.querySelector('.answerBox');
var answer = document.createElement('button');
    answer.classList.add('answerList','my-3','py-3','mx-auto','fadeIn');
    a.appendChild(answer);
    answer.innerHTML = answerText;
```

전체 구조

- 요소 생성 : HTML 문서에서 지정한 태그의 요소 생성, 위치 미지정 상태이므로 보이지 않음
  - 구조 : 변수 = 요소를 생성한다 '특정 태그를'

```
var answer = document.createElement('button');
```

- 요소 추가 : 새로운 요소를 자식 요소로 추가하는 것
  - 구조 : 부모 노드의 요소명.appendChild(자식요소)

```
a.appendChild(answer);
```

  - appendchild : 특정 부모 노드의 자식 노드 리스트 중 마지막 자식으로 추가
  - a 는 answerBox를 의미, qna 섹션의 div 로 answerBox가 존재
  - 즉, <qna> 하위인 <answerBox> 에 answer 를 추가

- 자식 요소값 설정 : 자식요소의 메서드를 수행하거나 속성 값 설정
  - 구조 : 자식요소.메서드 혹은 자식요소.속성 = 속성 값

```
answer.innerHTML = answerText;
```

- classList : 자바스크립트에서 HTML 요소의 Class 추가,제거등, element의 읽기전용 속성
  - 속성 : add, remove, contains(요소에 특정 클래스가 포함되어 있는지 테스트), toggle(미속성값시 추가, 추가 제거), replace(이름변경)
  - HTML 에서 class 를 여러개 늘려도 되지만 JS 에서 늘리는 방식을 채용
  - ('my-3 , py-3 , mx-auto') : 한 줄로 작성가능

2-2) createElement : 삽입 메서드

- 삽입 메서드의 종류
  - node.append(노드나 문자열) : 노드나 문자열을 node 끝에 삽입
  - node.prepend(노드나 문자열) : 노드나 문자열을 node 맨 앞에 삽입
  - node.before(노드나 문자열) : 노드나 문자열을 node 이전에 삽입
  - node.after(노드나 문자열) : 노드나 문자열을 node 다음에 삽입
  - node.replaceWith(노드나 문자열) : node를 새로운 노드나 문자열로 대체

- 삽입 메서드의 예시(https://ko.javascript.info/modifying-document)

```
let div = document.createElement('div');
div.className = "alert";
div.innerHTML = "<strong>안녕하세요!</strong>";
document.body.append(div);
```

1. 요소 생성 : 요소를 만듬
2. 클래스 이름 지정 : 'alert'로 설정
3. statement : HTML에 내용 작성
4. 위치 지정 : div가 삽입될 곳

- 실습을 통해 확인된 예시 (하도 많이 봐서...)

```
var a = document.querySelector('.answerBox');
var answer = document.createElement('button');
    answer.classList.add('answerList');
    a.appendChild(answer);
    answer.innerHTML = answerText;
```

1. button 에 요소 생성
2. 읽기전용으로 answerList 를 추가
3. a 의 위치에 answer를 넣음
4. statement 로 행동할 기능

- 실습을 통해 확인된 예시2

```
const modalElem = document.createElement('div');
modalElem.classList.add('modal');
modalElem.appendChild(modalContentElem)
modalElem.appendChild(closeBtn)
modalElem.addEventListener('click', stopPropagation);
```

1. 요소생성
2. 읽기전용 속성 추가
3. 위치지정(1)
4. 위치지정(2)
5. 행동할 기능, 함수도 가능

- 정리하자면,
  - let answer = document.createElement('button')
    - who, where : 요소를 생성, 해당 요소가 어디를 지칭하는지 설정한다.
  - answer.classList.add('answerList')
    - what : 요소의 속성값을 부여한다.
    - 주로 읽기용도로 classList.add를 통해 찾기 용이하도록 설정한다.
  - a.appendChild(cl)
    - where : a 라는 곳에 (아래에) cl 을 넣는다.
    - 속성값인 answerList도 따라가는 것이다. 이를 통해 찾기 용이해진다.
    - 생성된 요소가 어디로 갈지를 의미한다.
  - answer.innerHTML = answerText
    - how : 변수 answer 가 어떤 행동을 할지, statement 로써 기능을 정의한다.

- classList.Methods
  - .add(string) : 지정한 클래스 값을 추가, 추가하려는 클래스가 존재시 무시
  - .remove(string) : 지정한 클래스 값을 제거, 존재하지 않는 클래스면 에러 발생
  - .contains(string) : 지정한 클래스 값이 존재하는지 확인, True or False 반환
  - .replace(old, new) : old class 를 new class 로 대체
  - .item(number) : 인덱스 값을 활용하여 클래스 값을 반환

3) todos : 모든 기능을 함수로 정의하는 예시

- 실행 화면

- ✔ : 전체선택, 모든 버튼이 완료 변경, 미완료시 회색처리
- placeholder : <input> 태그의 속성, 입력 필드에 사용자가 적절한 값을 입력할 수 있도록 도와주는 짧은 도움말을 명시

```
<input type="text" class="todo-input" placeholder="해야 할 일을 입력해주세요.">
```

- 초반 let id 가 2 item left 와 관련?

- const paintTodos

```
switch (currentShowType) {
    case 'all':
        const allTodos = getAllTodos();
        allTodos.forEach(todo => { paintTodo(todo);});
        break;
    case 'active':
        const activeTodos = getActiveTodos();
        activeTodos.forEach(todo => { paintTodo(todo);});
        break;
    case 'completed':
        const completedTodos = getCompletedTodos();
        completedTodos.forEach(todo => { paintTodo(todo);});
        break;
    default:
        break;
}
```

  - switch(조건){각 case 별 기능} : 요일, 요일별 휴무차량같은 경우에 주로 사용
  - break : case 에서 미작성 무한 루프에 주의
  - forEach : 내일 다시