23.11.21

작성 일시	@2023년 11월 21일 오후 2:58
태그	

비밀번호 유효성 검사(조건 충족 추가)

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* 기본 스타일 설정 */
        ul {
            padding: 0;
        }
        li {
            margin: 0;
            padding: 5px;
            box-sizing: border-box;
            display: flex;
            flex-direction: row;
            align-items: center;
        }
        .checkbox {
            width: 18px;
            height: 18px;
            border-radius: 50px;
            background-color: lightgray;
            margin-right: 10px;
        }
        .label {
            color: lightgray;
        }
        li.active .checkbox {
            background-color: aquamarine;
        }
        li.active .label {
            color: aquamarine;
        }
        .chack{
            color: red;
        }
    </style>
</head>
<body>
<div class="signup-container">
    <!-- 비밀번호 유효성 검사 컨테이너 -->
    <div class="title"><span>비밀번호 유효성 검사</span></div>
    <div class="password">
        <!-- 비밀번호 입력란 -->
        <input type="text" class="input-password">
    </div>
    <ul>
        <!-- 각 비밀번호 유효성 항목 -->
        <li class="lowercase"><div class="checkbox"></div><span class="label">소문자</span> </li>
        <li class="uppercase"><div class="checkbox"></div><span class="label">대문자</span> </li>
        <li class="number"><div class="checkbox"></div><span class="label">숫자</span> </li>
        <li class="special"><div class="checkbox"></div><span class="label">특수 문자 (!@#$%^&*)</span> </li>
        <li class="character-length"><div class="checkbox"></div><span class="label">8자 이상</span> </li>
        <li class="chack"><div class="checkbox"></div><span class="label">조건 충족 </span> </li>
        <li class="chackNot"><div class="checkbox"></div><span class="label">조건 충족X </span> </li>
    </ul>

</div>
<script>
    // JavaScript 비밀번호 유효성 검사 로직
    const inputElem = document.querySelector('input'); // 입력란 엘리먼트 가져오기
    const lowercaseElem = document.querySelector('.lowercase'); // 소문자 요소 가져오기
    const uppercaseElem = document.querySelector('.uppercase'); // 대문자 요소 가져오기
    const numberElem = document.querySelector('.number'); // 숫자 요소 가져오기
    const specialElem = document.querySelector('.special'); // 특수문자 요소 가져오기
    const characterLengthElem = document.querySelector('.character-length'); // 길이 요소 가져오기
```

```
    const chackElem = document.querySelector(".chack");
    const chackNotElem = document.querySelector(".chackNot");

    const isValidLowercase = (password) => {
        return password.search(/[a-z]/g) >= 0; // 소문자 포함 여부 확인
    }
    const isValidUppercase = (password) => {
        return  password.search(/[A-Z]/g) >= 0; // 대문자 포함 여부 확인
    }
    const isValidNumber = (password) => {
        return  password.search(/[0-9]/g) >= 0; // 숫자 포함 여부 확인
    }
    const isValidSpecial = (password) => {
        return password.search(/[~!@#$%^&*()_+|<>?:{}`]/gi) >= 0; // 특수문자 포함 여부 확인 / 대소문자 구분X
    }
    const isValidCaracterLength = (password) => {
        return password.length >= 8 // 최소 8자 이상 여부 확인
    }


    // const isValidChackNot = (password) => {
    //     if(isValidLowercase(password) || isValidUppercase(password) || isValidNumber(password) || isValidSpecial(password) || isVali
    //         change2()

    //         }
    // }




    // 비밀번호 입력 시 유효성 검사 상태 업데이트
    inputElem.addEventListener('input', (e) => {
        console.log(e)
        const password = e.target.value;
        isValidLowercase(password) ? lowercaseElem.classList.add('active') : lowercaseElem.classList.remove('active'); // 소문자 유효성 표
        isValidUppercase(password) ? uppercaseElem.classList.add('active') : uppercaseElem.classList.remove('active'); // 대문자 유효성 표
        isValidNumber(password) ? numberElem.classList.add('active') : numberElem.classList.remove('active'); // 숫자 유효성 표시 여부 업데이
        isValidSpecial(password) ? specialElem.classList.add('active') : specialElem.classList.remove('active'); // 특수문자 유효성 표시 여
        isValidCaracterLength(password) ? characterLengthElem.classList.add('active') : characterLengthElem.classList.remove('active');
        // isValidChack(password) ? chackElem.classList.add('active') : chackElem.classList.remove('active');
        // isValidChackNot(password) ? chackNotElem.classList.remove('active') : chackNotElem.classList.add('active');

        if(isValidLowercase(password) && isValidUppercase(password) && isValidNumber(password) && isValidSpecial(password) && isValidCa
            chackElem.classList.add('active');
            chackNotElem.classList.remove('active');

        }else{
            chackElem.classList.remove('active');

        }
    }
    )
    chackNotElem.classList.add('active');


</script>

</body>
</html>
```

html Copy code

비밀번호 유효성 검사

qw1Q@qwe

- 소문자
- 대문자
- 숫자
- 특수 문자 (!@#$%^&*)
- 8자 이상
- 조건 충족
- 조건 충족X

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Event Propagation</title>
    <link rel="stylesheet" href="style.css">

</head>

<body class="container" >
<div class="open-modal-btn" onclick="openModal();">Open Modal</div>


<script>
    // HTML 문서에서 '.container' 클래스를 가진 요소를 찾아 bodyElem에 할당
    const bodyElem = document.querySelector('.container');

    // 모달 닫기 함수
    const closeModal = (event) => {
        // 모달 래퍼 엘리먼트를 찾아 제거
        const modalWrapperElem = document.querySelector('.modal-wrapper');
        // 모달을 감싸는 외부 컨테이너 엘리먼트를 찾는 것
        bodyElem.removeChild(modalWrapperElem);
        //  modalWrapperElem을 문서의 bodyElem에서 제거. 모달을 화면에서 제거 사용자에게 보이지 않도록 하는 역할
    }

    // 이벤트 전파 중지 함수
    const stopPropagation = (e) => {
        e.stopPropagation();
    }

    // 모달 열기 함수
    const openModal = () => {
        // 모달 래퍼 엘리먼트 생성
        const modalWrapperElem = document.createElement('div'); // 모달을 감싸는 외부 컨테이너 역할을
        // 모달 래퍼에 클릭 이벤트 리스너 추가하여 모달 닫기 함수 호출
        modalWrapperElem.addEventListener("click", closeModal);
        modalWrapperElem.classList.add('modal-wrapper');
        //모달 컨텐츠와 닫기 버튼을 추가하고, 문서에 삽입하여 모달을 표시하는 데 사용


        // 모달 엘리먼트 생성
        const modalElem = document.createElement('div');
        modalElem.classList.add('modal');
        // 모달에 클릭 이벤트 리스너 추가하여 이벤트 전파 중지 함수 호출
        modalElem.addEventListener('click', stopPropagation);

        // 모달 콘텐츠 엘리먼트 생성
        const modalContentElem = document.createElement('div');
        modalContentElem.classList.add('modal-content');
        // 모달 콘텐츠에 내용 추가
        modalContentElem.innerHTML = `
            <span>This is Modal</span>
        `

        // 닫기 버튼 엘리먼트 생성
        const closeBtn = document.createElement('div')
        closeBtn.classList.add('close-modal-btn')
        closeBtn.innerHTML = "닫기";
        // 닫기 버튼에 클릭 이벤트 리스너 추가하여 모달 닫기 함수 호출
        closeBtn.addEventListener("click", closeModal);

        // 생성한 엘리먼트들을 계층적으로 추가
        modalElem.appendChild(modalContentElem) // modalContentElem은 모달의 내용을 담는 엘리먼트
        modalElem.appendChild(closeBtn) // 모달 창 내에 닫기 버튼 추가. closeBtn은 모달을 닫는 동작을 수행할 버튼 엘리먼트

        modalWrapperElem.appendChild(modalElem) // 모달 창을 감싸는 외부 컨테이너에 모달 자체를 추가하
        bodyElem.appendChild(modalWrapperElem) // 전체 문서(body)에 모달 컨테이너를 추가, 모달이 문서의 어떤 부분에 표시될지 결정
    }

    // 초기화 함수
    const init = () => {
        // '.open-modal-btn' 클래스를 가진 엘리먼트를 찾아 openModal 함수를 클릭 이벤트에 연결
        const openModalBtnElem = document.querySelector('.open-modal-btn');
        //  .open-modal-btn 클래스를 가진 첫 번째 엘리먼트를 찾아 openModalBtnElem 변수 할당.
        //  .querySelector 메서드는 주어진 CSS 선택자에 해당하는 첫 번째 엘리먼트 반환
        openModalBtnElem.addEventListener('click', openModal);
        // 엘리먼트에 대해 클릭 이벤트 리스너를 추가합니다. 클릭 이벤트가 발생하면 openModal 함수가 호출
    }
```

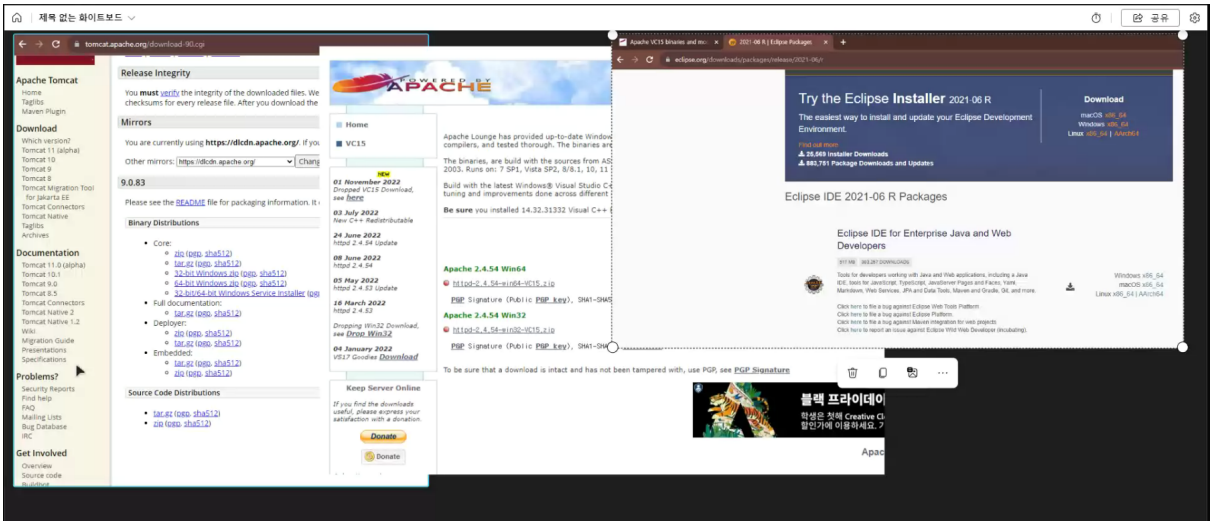
```
    // 문서 로딩 완료 시 init 함수 호출
    document.addEventListener('DOMContentLoaded', () => {
        init();
    })
    // DOMContentLoaded 이벤트에 대한 이벤트 리스너를 추가. HTML 문서의 모든 DOM이 로드되고 파싱된 후에 발생
    // 외부 리소스(images, stylesheets, 등)가 아직 로드되지 않은 상태에서도 발생
    //HTML 구조가 완전히 로드된 이후에 JavaScript 코드가 실행되도록 보장
</script>


</body>
</html>
```

js 연결 확인

```
const todoInputElem = document.querySelector('.todo-input');
const todoListElem = document.querySelector('.todo-list');
const completeAllBtnElem = document.querySelector('.complete-all-btn');
const leftItemsElem = document.querySelector('.left-items')
const showAllBtnElem = document.querySelector('.show-all-btn');
const showActiveBtnElem = document.querySelector('.show-active-btn');
const showCompletedBtnElem = document.querySelector('.show-completed-btn');
const clearCompletedBtnElem = document.querySelector('.clear-completed-btn');

//items left
let id = 0;
const setId = (newId) => {id = newId};

let isAllCompleted = false; // 전체 todos 체크 여부
const setIsAllCompleted = (bool) => { isAllCompleted = bool};

let currentShowType = 'all'; // all  | active | complete
const setCurrentShowType = (newShowType) => currentShowType = newShowType

let todos = [];
const setTodos = (newTodos) => {
    todos = newTodos;
}

const getAllTodos = () => {
    return todos;
}
const getCompletedTodos = () => {
    return todos.filter(todo => todo.isCompleted === true );
}
const getActiveTodos = () => {
    return todos.filter(todo => todo.isCompleted === false);
}

const setLeftItems = () => {
    const leftTodos = getActiveTodos()
    leftItemsElem.innerHTML = `${leftTodos.length} items left`
}

const completeAll = () => {
    completeAllBtnElem.classList.add('checked');
    const newTodos = getAllTodos().map(todo => ({...todo, isCompleted: true }) )
    setTodos(newTodos)
}

const incompleteAll = () => {
    completeAllBtnElem.classList.remove('checked');
    const newTodos =  getAllTodos().map(todo => ({...todo, isCompleted: false }) );
    setTodos(newTodos)
}

// 전체 todos의 check 여부 (isCompleted)
const checkIsAllCompleted = () => {
    if(getAllTodos().length === getCompletedTodos().length ){
        setIsAllCompleted(true);
        completeAllBtnElem.classList.add('checked');
    }else {
        setIsAllCompleted(false);
        completeAllBtnElem.classList.remove('checked');
    }
}
```

```
//누를ㄸ마다 상태 변하
const onClickCompleteAll = () => {
    if(!getAllTodos().length) return; // todos배열의 길이가 0이면 return; !부정

    if(isAllCompleted) incompleteAll(); // isAllCompleted가 true이면 todos를 전체 미완료 처리 (토글처리)
    else completeAll(); // isAllCompleted가 false이면 todos를 전체 완료 처리
    setIsAllCompleted(!isAllCompleted); // isAllCompleted 토글
    paintTodos(); // 새로운 todos를 렌더링
    setLeftItems()
}

const appendTodos = (text) => {
    const newId = id + 1; // 기존에 i++ 로 작성했던 부분을 setId()를 통해 id값을 갱신하였다.
    setId(newId)
    const newTodos = getAllTodos().concat({id: newId, isCompleted: false, content: text })
    // const newTodos = [...getAllTodos(), {id: newId, isCompleted: false, content: text }]
    setTodos(newTodos)
    setLeftItems()
    checkIsAllCompleted();
    paintTodos();
}

const deleteTodo = (todoId) => {
    const newTodos = getAllTodos().filter(todo => todo.id !== todoId );
    setTodos(newTodos);
    setLeftItems()
    paintTodos()
}

const completeTodo = (todoId) => {
    const newTodos = getAllTodos().map(todo => todo.id === todoId ? {...todo,  isCompleted: !todo.isCompleted} : todo )
    setTodos(newTodos);
    paintTodos();
    setLeftItems()
    checkIsAllCompleted();
}

const updateTodo = (text, todoId) => {
    const currentTodos = getAllTodos();
    const newTodos = currentTodos.map(todo => todo.id === todoId ? ({...todo, content: text}) : todo);
    setTodos(newTodos);
    paintTodos();
}

const onDbclickTodo = (e, todoId) => {
    const todoElem = e.target;
    const inputText = e.target.innerText;
    const todoItemElem = todoElem.parentNode;
    const inputElem = document.createElement('input');
    inputElem.value = inputText;
    inputElem.classList.add('edit-input');
    inputElem.addEventListener('keypress', (e)=>{
        //Enter 입력했을깨
        if(e.key === 'Enter') {
            updateTodo(e.target.value, todoId);
            document.body.removeEventListener('click', onClickBody );
        }
    })

    const onClickBody = (e) => {
        if(e.target !== inputElem)  {
            todoItemElem.removeChild(inputElem);
            document.body.removeEventListener('click', onClickBody );
        }
    }

    document.body.addEventListener('click', onClickBody)
    todoItemElem.appendChild(inputElem);
}

const clearCompletedTodos = () => {
    const newTodos = getActiveTodos()
    setTodos(newTodos)
    paintTodos();
}

//작업 실행시 화면 변화 내용
const paintTodo = (todo) => {
    const todoItemElem = document.createElement('li');
    todoItemElem.classList.add('todo-item');

    todoItemElem.setAttribute('data-id', todo.id );

    const checkboxElem = document.createElement('div');
```

```
    checkboxElem.classList.add('checkbox');
    checkboxElem.addEventListener('click', () => completeTodo(todo.id))

    const todoElem = document.createElement('div');
    todoElem.classList.add('todo');
    todoElem.addEventListener('dblclick', (event) => onDbclickTodo(event, todo.id))
    todoElem.innerText = todo.content;

    const delBtnElem = document.createElement('button');
    delBtnElem.classList.add('delBtn');
    delBtnElem.addEventListener('click', () =>  deleteTodo(todo.id))
    delBtnElem.innerHTML = 'X';

    if(todo.isCompleted) {
        todoItemElem.classList.add('checked');
        checkboxElem.innerText = '✔';
    }

    todoItemElem.appendChild(checkboxElem);
    todoItemElem.appendChild(todoElem);
    todoItemElem.appendChild(delBtnElem);

    todoListElem.appendChild(todoItemElem);
}

const paintTodos = () => {
    todoListElem.innerHTML = '';
//휴무표등에 사용
    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;
    }
}

const onClickShowTodosType = (e) => {
    const currentBtnElem = e.target;
    const newShowType = currentBtnElem.dataset.type;

    if ( currentShowType === newShowType ) return;

    const preBtnElem = document.querySelector(`.show-${currentShowType}-btn`);
    preBtnElem.classList.remove('selected');

    currentBtnElem.classList.add('selected')
    setCurrentShowType(newShowType)
    paintTodos();
}

const init = () => {
    todoInputElem.addEventListener('keypress', (e) =>{
        //Enter 눌렀을때
        if( e.key === 'Enter' ){
            appendTodos(e.target.value); todoInputElem.value ='';
        }
    })
    completeAllBtnElem.addEventListener('click',  onClickCompleteAll);
    showAllBtnElem.addEventListener('click', onClickShowTodosType);
    showActiveBtnElem.addEventListener('click',onClickShowTodosType);
    showCompletedBtnElem.addEventListener('click',onClickShowTodosType);
    clearCompletedBtnElem.addEventListener('click', clearCompletedTodos);
    setLeftItems()
}

init()
```


제목 없는 화이트보드
공유
tomcat.apache.org/download-90.cgi
Apache Tomcat
Release Integrity
Mirrors
9.0.83
Binary Distributions
Source Code Distributions
Documentation
Problems?
Get Involved
Keep Server Online
Donate
Apache Lounge
Home
VC15
Apache 2.4.54 Win64
Apache 2.4.54 Win32
eclipse.org/downloads/packages/release/2021-06/r
Try the Eclipse Installer 2021-06 R
The easiest way to install and update your Eclipse Development Environment.
Download
Eclipse IDE 2021-06 R Packages
Eclipse IDE for Enterprise Java and Web Developers