I. 웹 브라우저 단축키 제어

웹스퀘어를 이용한 웹 어플리케이션 개발 시 웹 브라우저에서 사용하는 단축키 기능을 중지시키고, 사용자가 원하는 기능으로 재정의해야할 경우 KeyDown 이벤트에서 preventDefault API 실행을 통해서 웹 브라우저의 기능을 중지시킬 수 있습니다. 이때 모든 웹 브라우저의 모든 단축키 동작을 KeyDown 이벤트에서 preventDefault API 실행을 통해서 중지시킬 수 있는 것은 아닙니다. 본 문서는 KeyDown 이벤트에서 preventDefault API 실행을 통해서 중지가 가능한 단축키와 중지 시킬 수 없는 단축키를 테스트하고 결과를 정리했습니다.

WRM 5.0.4.3 버전(다운로드, 릴리즈 노트)에 적용되어 있습니다.

1. 웹 브라우저 단축키 제어를 위한 KeyDown 이벤트

웹 브라우저 단축키의 동작이 실행되는 시점은 키보드의 Key가 눌려지는 시점(KeyDown 이벤트)에서 발생합니다. 따라서 웹 브라우저 화면 내에서 Javascript 이벤트로 웹 브라우저의 단축키 동작을 제어하려면 대부분 KeyDown 이벤트에서 처리해야 합니다. KeyUp이나 KeyPress 이벤트에서 처리하면 웹 브라우저의 단축키 동작이 먼저 실행되게 됩니다.

2. 웹 브라우저 단축키 제어 공통 함수

아래는 KeyDown 이벤트에 전달된 Key Event 객체(e)의 preventDefault API 실행을 통해서 웹 브라우저의 단축키 동작을 막고, 원하는 동작을 Javascript로 작성할 수 있도록 하는 코드입니다.

```
// 웹 브라우저 단축키가 동작하지 않도록 설정함 (true : 동작, false : 미동작)
// iframe 안에 포함된 컨텐츠는 Frame 영역이 분리되기 때문에 별도의 추가 작업을 해야 단축키 제어를 할 수 있다.
var gcm = {};
gcm.hkey = {
 IS_USE_BROWSER_SHORTCUT : false
};
/**
 * KeyDown 이벤트 발생 시 로그를 출력하고, 웹 브라우저의 단축 동작을 제어한다.
 *
 * @param {Object} e 키보드 이벤트
 */
gcm.hkey.keydownEvent = function(e) {
 try {
  var lastKey = e.key || e.keyCode || e.which;
  var complexKey = "";
  if (e.ctrlKey && e.altKey) {
   complexKey = "ctrlAltKey";
  } else if (e.ctrlKey && e.shiftKey) {
   complexKey = "ctrlShiftKey";
  } else if (e.altKey && e.shiftKey) {
   complexKey = "altShiftKey";
  } else {
   if (e.altKey) {
    complexKey = "altKey";
   } else if (e.ctrlKey) {
    complexKey = "ctrlKey";
   } else if (e.shiftKey) {
    complexKey = "shiftKey";
   } else {
    complexKey = "singleKey";
   }
  }
  // (Ctrl, Alt, Shift)가 아닌 lastKey가 인식될 경우
  if (lastKey != "Control" && lastKey != "Alt" && lastKey != "Shift") {

   // KeyboardEvent.key 값이 전달되지 않는 구형 브라우저에서 lastKey 재설정
   if ((typeof lastKey === "number") || (lastKey === "Unidentified")) {
    if (e.keyCode >= 112 && e.keyCode <= 123) {
     var funtionKey = [ "F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10", "F11", "F12" ];
     lastKey = funtionKey[e.keyCode - 112];
    } else if (e.keyCode == 9) {
     lastKey = "Tab";
```

```
    } else if (e.keyCode == 27) {
     lastKey = "Escape";
    } else if (e.keyCode == 187) {
     lastKey = "=";
    } else if (e.keyCode == 189) {
     lastKey = "-";
    } else {
     lastKey = String.fromCharCode(e.keyCode).toUpperCase();
    }
   }

   if (gcm.hkey.isPreventKey(complexKey, lastKey)) {

    // 아래의 코드는 단축키 로그를 출력하기 위한 코드로 실제 프로젝트에 적용 시 주석으로 막으시기 바랍니다.
    // 원하는 단축키 동작이 있을 경우
    // searchBox 컴포넌트에 포커스가 위치한 경우에는 scwin.writeKeyLog 함수를 찾지 못해서 로그 출력이 안됩니다.
    if (typeof $p.debug.getScope(document.activeElement).scwin.writeKeyLog === "function") {
     $p.debug.getScope(document.activeElement).scwin.writeKeyLog(complexKey, lastKey);
    }

    // 전역 단축키 실행한다.
    gcm.hkey.runGlobalEvent(complexKey, lastKey);

    // 운영 환경에서 브라우저 단축키의 동작을 정지 시킴
    if (gcm.hkey.IS_USE_BROWSER_SHORTCUT === false) {

     // IE에서 F1 Key 동작을 중지 시킴
     if ('onhelp' in window) {
      window.onhelp = function() {
       return false;
      }
     }

     if (e.preventDefault) {
      e.preventDefault();
     } else if (e.returnValue) {
      e.returnValue = false;
     }
    }
   }
  }
 } catch (ex) {
  console.error(ex);
 }
};
/**
 * 단축키 실행을 막을 대상 Key인지 검사를 수행한다.
 */
gcm.hkey.isPreventKey = function(complexKey, lastKey) {
 var exTag = ["INPUT", "TEXTAREA", "IFRAME"];
 var controlKeyList = ["F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10", "F11", "F12", "ctrlKey", "altKey",
"ctrlAltKey", "ctrlShiftKey", "altShiftKey" , "Tab", "Escape"];
 var activeTag = document.activeElement.tagName;
 var ucLastKey = lastKey.toUpperCase();
 if ((exTag.indexOf(activeTag) === -1)) {
  return true;
 } else if ((exTag.indexOf(activeTag) > -1) && (((complexKey === "ctrlKey") && (ucLastKey === "A"))
  || ((complexKey === "ctrlKey") && (ucLastKey === "C")) || ((complexKey === "ctrlKey") && (ucLastKey === "V"))
  || ((complexKey === "ctrlKey") && (ucLastKey === "X")) || ((complexKey === "ctrlKey") && (ucLastKey === "Y"))
  || ((complexKey === "ctrlKey") && (ucLastKey === "Z")) || ((complexKey === "ctrlShiftKey") && (ucLastKey ===
"Z")))) {
  return false;
 } else if ((controlKeyList.indexOf(complexKey) !== -1) || (controlKeyList.indexOf(lastKey) !== -1)) {
```

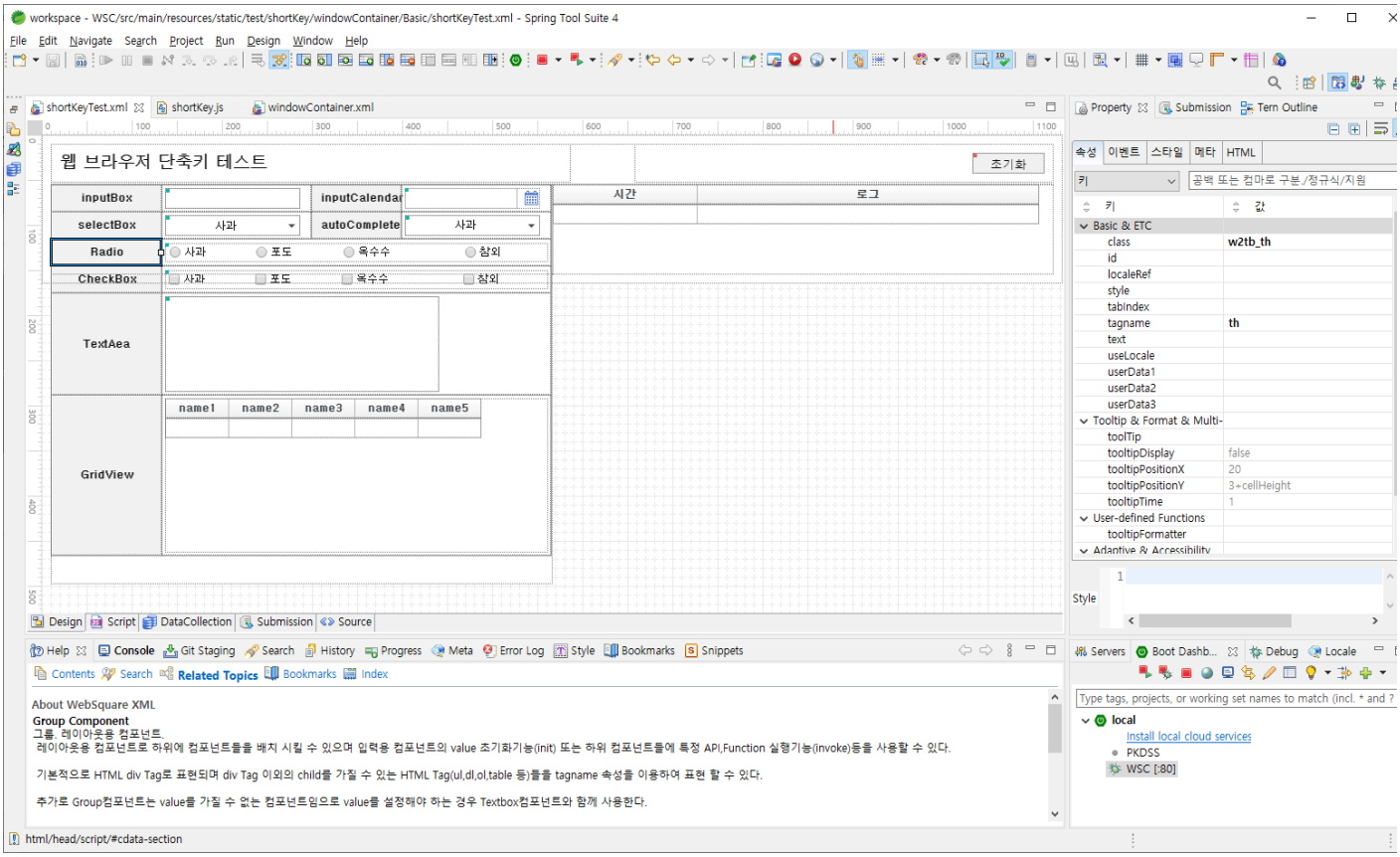
```
  return true;
 } else {
  return false;
 }
};
/**
 * 전역 단축키 실행 함수.
 * @param {String} complexKey 입력된 복합키 ("ctrlKey", "altKey", "ctrlAltKey", "ctrlShiftKey", "altShiftKey")
 * @param {String} eventKey 입력된 키 ("F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10", "F11", "F12", "Tab",
"Escape", "0~9", "a-z", "A-Z", 특수문자(!@#~) 등)
 */
gcm.hkey.runGlobalEvent = function(complexKey, eventKey) {
 try {
  // 아래의 전역 단축키 동작에 대해서 코드를 작성합니다.
  if (eventKey === "F1") {
   alert("F1 키가 입력 되었습니다.");
  } else if (eventKey === "Escape") {
   alert("Escape 키가 입력 되었습니다.");
  }
 } catch (ex) {
  console.error(ex);
 }
}
/**
 * Document 객체에 keyDown 이벤트를 바인딩 시킨다.
 */
gcm.hkey.setShortKeyDownAction = function() {
 var shortcutTargetElement = document;

 if (shortcutTargetElement.addEventListener) {
  shortcutTargetElement.addEventListener('keydown', gcm.hkey.keydownEvent);
 } else if (shortcutTargetElement.attachEvent)  {
```

```
  shortcutTargetElement.attachEvent('keydown', gcm.hkey.keydownEvent);
 }
};
```

3. 단축키 제어 공통 함수를 적용한 테스트 화면

1) shortKeyTest.xml

단축키가 입력되면 Key 정보를 GridView에 로그로 출력되도록 하는 테스트 프로그램의 화면입니다.

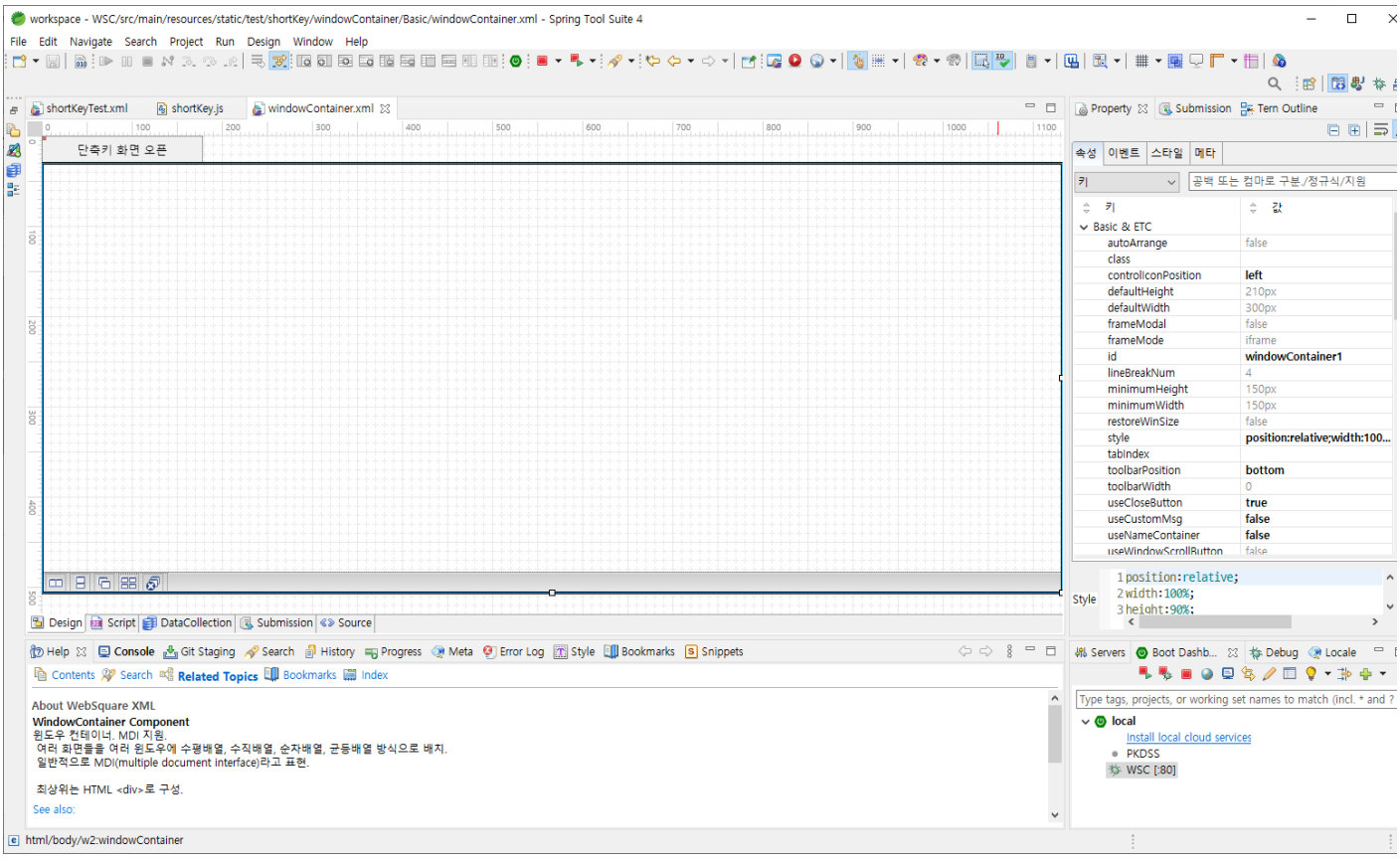


위 화면의 스크립트는 다음과 같습니다.

```
/**
 * 단축키 동작 정보를 출력한다.
 *
 * @param {String} complexKey 복합키 입력 정보 (Ctrl, Alt, Shift)
 * @param {String} lastKey 키 입력 정보
 */
scwin.writeKeyLog = function(complexKey, lastKey) {
 var logData = {
  datetime : (new Date()).toLocaleString('kr-KO'),
  log : "complexKey : " + complexKey + ", lastKey : " + lastKey
 };
 dlt_log.insertJSON(0, [logData]);
};
/**
 * 로그 내역을 모두 삭제한다.
 */
scwin.btn_refresh_onclick = function(e) {
 dlt_log.removeAll();
};
```

2) windowContainer.xml

shortKeyTest.xml 화면을 WindowContainer 컴포넌트에 여러 개 로딩해서 단축키 제어 동작이 정상적으로 동작하는지 테스트하는 프로그램입니다.

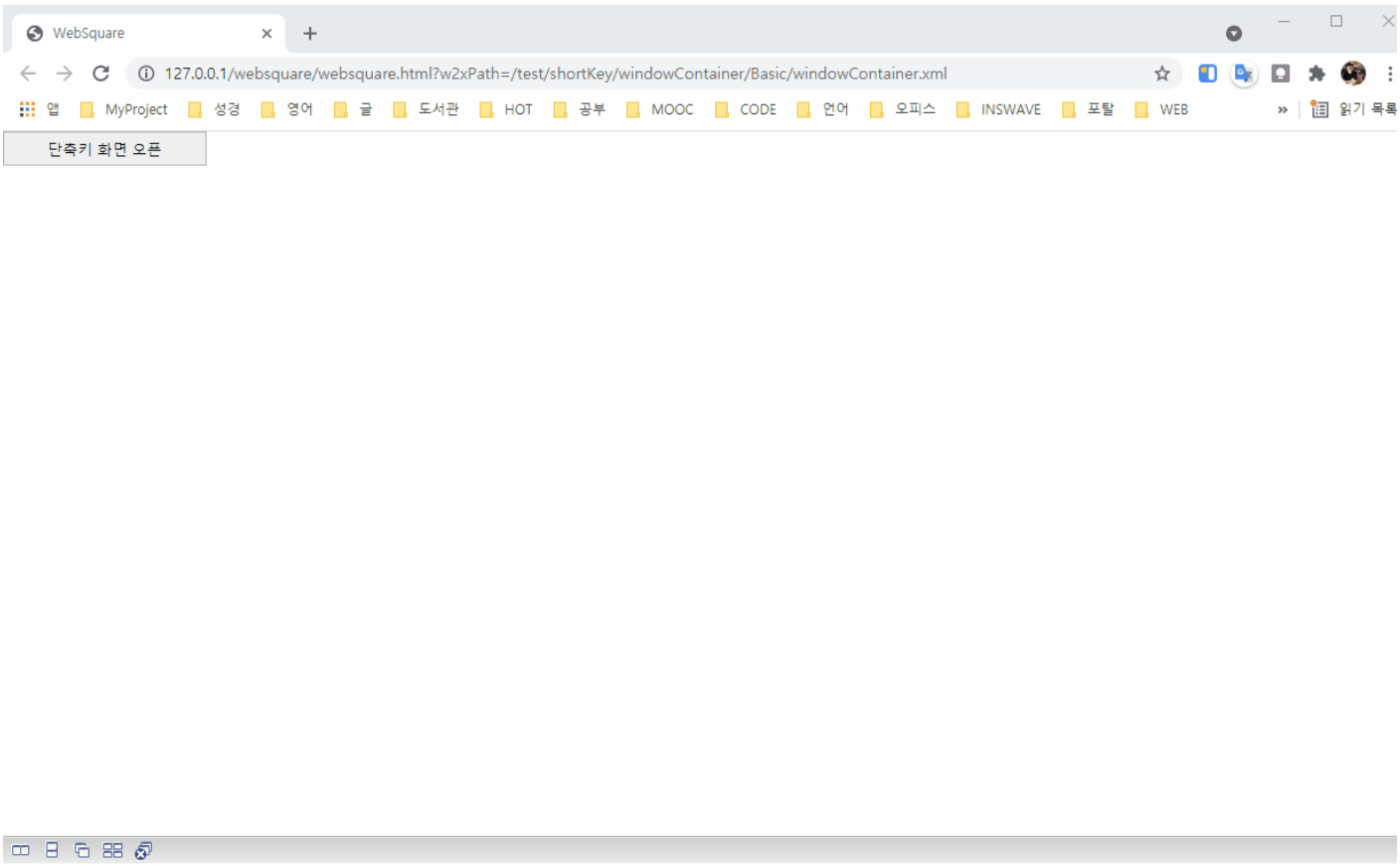


위 화면의 스크립트는 다음과 같습니다.

```
scwin.onpageload = function() {
 // 단축키 제어 동작 함수를 실행합니다.
 gcm.hkey.setShortKeyDownAction();
};
scwin.windowIdx = 0;
scwin.btn_createWin_onclick = function(e) {
 var windowId = "window" + scwin.windowIdx++;
 var option = {
  title : windowId,
  src : "shortKeyTest.xml",
  windowTitle : windowId,
  windowId : windowId,
  openAction : "existWindow",
  frameMode : "wframe"
 };
 windowContainer1.createWindow(option);
};
```

4. 단축키 테스트 프로그램 시연 영상

windowContainer.xml 화면을 실행해서 단축키 제어 동작을 시연한 영상입니다.



5. 웹 브라우저별 단축키 제어 테스트 결과

아래는 비입력 모드에서 주요 단축키 별로 웹 브라우저의 단축키 동작을 중지 시키고 화면 내에서 정의된 Script 동작으로 대체할 수 있는지 테스트한 결과입니다. 다양한 환경에서 Ctrl, Shift, Alt 키를 조합한 복합키 테스트 결과는 [웹브라우저별단축키제어테스트결과_v13_20210929.xlsx] 문서를 참고하시기 바랍니다.

O : 단축키 사용 가능, X : 단축키 사용 불가

단축키	키 설명	IE(9~11)	Edge	Chrome	FireFox	Opera	Whale
F1	IE, Edge, Opera : 도움말 열기 Chrome : 새 탭에서 크롬 고객센터 열기	O	O	O	O	O	O
F2	미사용	O	O	O	O	O	O
F3	IE, Edge, Chrome, Opera : 현재 탭에서 찾기 Firefox : 다시 검색	O	O	O	O	X	O
F4	IE : 입력한 주소 목록 표시 Edge : 주소표시줄의 URL 선택	O	O	O	O	O	O
F5	페이지 새로 고침	O	O	O	O	O	O
F6	IE : 웹페이지에서 항목 이동 Edge : 다음 창으로 포커스 전환 Chrome : 현재탭,주소표시줄,첫번째 북마크 포커스 전환 Firefox : 다음 프레임으로 이동, 팝업	O	O	O	O	O	O
F7	IE, Edge, Firefox : 커서 브라우징 시작 Chrome : 캐럿 브라우징 사용 설정	O	O	O	O	O	O
F8	Opera : 주소표시줄 포커스	O	O	O	O	O	O
F9	Edge : 몰입형 리더 시작 또는 종료 Firefox : 읽기 모드 실행하기	O	O	O	O	O	O
F10	Edge : 설정 등 "..."단추에 포커스 설정 Chrome : 크롬 툴바의 가장 오른쪽에 있는 항목에 포커스 설정 Firefox : 메뉴 표시줄 열고 닫기	O	O	O	O	O	O
F11	전체화면 및 일반 보기 전환	O	O	O	O	X	O
F12	IE, Edge, Chrome, Firefox : 개발자 도구 열기	O	O	O	O	O	O
Tab	IE, Edge : 다음 컴포넌트로 이동 Chrome : 클릭할수 있는 항목을 앞으로 이동하면서 탐색 Firefox : 다음 링크나 입력 필드 포커스 Opera : 페이지 요소 앞으로 순환	O	O	O	O	O	O
Esc	IE, Edge, Chrome, Firefox : 페이지 로딩 중지/ 대화상자 또는 팝업 닫기	O	O	O	O	O	O
spacebar	웹 페이지를 한 번에 한 화면씩 아래로 스크롤	O	O	O	O	O	O
PageDown	웹 페이지를 한 번에 한 화면씩 아래로 스크롤	O	O	O	O	O	O
PageUp	웹 페이지를 한 번에 한 화면씩 위로 스크롤	O	O	O	O	O	O
Delete	검색창에서 내용 삭제	O	O	O	O	O	O
Home	페이지 맨위로 이동	O	O	O	O	O	O
End	페이지 맨 아래로 이동	O	O	O	O	O	O

6. 관련 파일

- shortKey.js
- shortKeyTest.xml
- windowContainer.xml
- 웹브라우저별단축키제어테스트결과_v13_20210929.pdf
- 웹브라우저별단축키제어테스트결과_v13_20210929.xlsx

7. 참고 웹 사이트

- PreventDefault API : https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault
- Keydown Event : https://developer.mozilla.org/en-US/docs/Web/API/Document/keydown_event
- KeyboardEvent.key : https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key
- KeyboardEvent.keyCode : https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/keyCode
- Javascript Keycode List : https://www.freecodecamp.org/news/javascript-keycode-list-keypress-event-key-codes/
- iE11 Shortcut : https://support.microsoft.com/ko-kr/windows/internet-explorer-11-%EB%B0%94%EB%A1%9C-%EA%B0%80%EA%B8%B0-%ED%82%A4-bf5218a2-7626-b834-db7f-0633120e5620
- Edge Shortcut : https://support.microsoft.com/ko-kr/microsoft-edge/microsoft-edge%EC%9D%98-%EB%B0%94%EB%A1%9C-%EA%B0%80%EA%B8%B0-%ED%82%A4-50d3edab-30d9-c7e4-21ce-37fe2713cfad
- Chrome Shortcut : https://support.google.com/chrome/answer/157179?hl=ko&co=GENIE.Platform%3DDesktop
- FireFox Shortcut : https://support.mozilla.org/ko/kb/keyboard-shortcut-perform-firefox-tasks-quickly
- Opera Shortcut : https://help.opera.com/en/latest/shortcuts/
- Whale Shortcut : whale://settings/shortcuts