JavaScript

텍스트 편집 – 김근형 강사

편집 가능한 문서와 요소

- 웹 컨텐츠의 텍스트 편집
  - HTML5 에서는 여러 요소의 컨텐츠를 편집할 수 있다.
  - 편집 영역은 요소에 한정지을 수도 있고 문서 전체를 대상으로 할 수 있다.
  - 모든 요소에 contenteditable 속성을 넣어 요소의 컨텐츠를 편집할 수 있다.

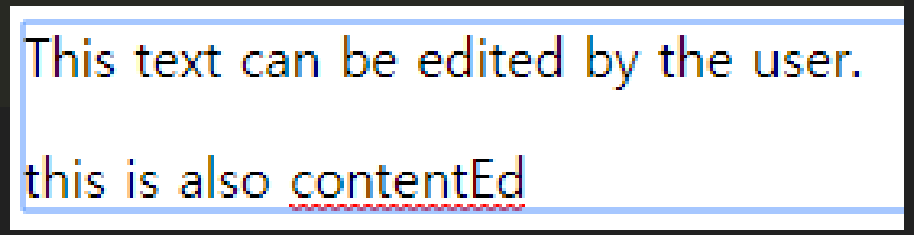
속성	의미
contenteditable="true"	편집할 수 있다
contenteditable="false"	편집할 수 없다. 기본값
contenteditable=""	공백 문자열을 지정하면 "true"가 지정된 것으로 간주하며 편집 가능한 것을 의미한다.

  - 참고 : contenteditable 속성을 지정하지 않은 요소는 모두 "inherit" 상태이다.

편집 가능한 문서와 요소

- 웹 컨텐츠의 텍스트 편집 예제 - 1

```
<body>
    <div contentEditable="true">
        This text can be edited by the user.
        <p>this is also contentEditable</p>
    </div>
</body>
```



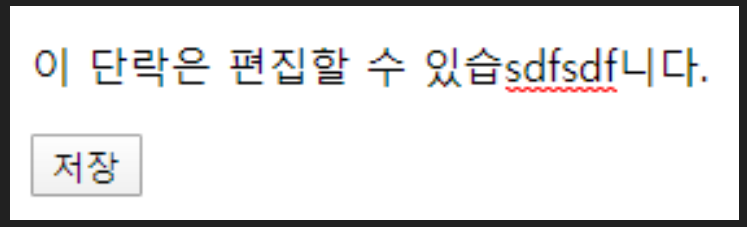
편집 가능한 문서와 요소

- 웹 컨텐츠의 텍스트 편집
  - contentEditable 속성과 관련하여 요소 객체에는 속성이 두 개 있다.

속성	의미
element.contentEditable	요소가 편집 가능한지를 문자열로 반환한다. 편집할 수 있으면 “true”를 할 수 없으면 “false”를, 부모의 상태를 상속하면 “inherit”를 반환한다. 속성값을 지정하여 편집 여부를 지정할 수 있다.
element.isContentEditable	요소가 편집 가능하다면 true를, 그렇지 않으면 false 를 반환한다. 값 지정 불가

편집 가능한 문서와 요소

- 웹 컨텐츠의 텍스트 편집 예제 - 2




```
<p id="p">이 단락은 편집할 수 있습니다.</p>
<p><button>편집</button></p>
<script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function() {
        // button 요소에 click이벤트 리스너를 지정
        button = document.querySelector('button');
        button.addEventListener("click", function() {
            // p요소
            var p = document.querySelector('#p');
            // 편집 가능여부로 조건 분기
            if( p.isContentEditable == false ) {
                // 편집 불가면 가능으로 변경
                p.contentEditable = "true";
                // 포커스 주기
                p.focus();
                // 버튼 표기 변경
                button.textContent = "저장 ";
            } else {
                // 편집 가능이면 불가능으로 변경
                p.contentEditable = "false";
                // 버튼 표기를 변경
                button.textContent = "편집 ";
            }
        }, false);
    }, false);
</script>
```

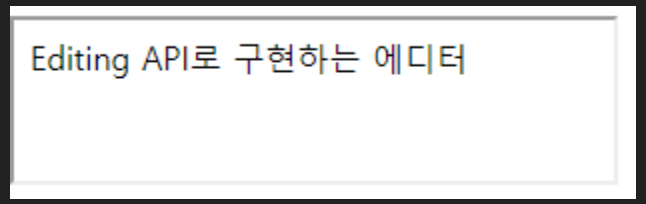
편집 가능한 문서와 요소

- 문서 전체 편집
  - 문서 전체를 편집 가능하게 하려면 document 객체의 designMode 속성을 사용한다.

속성	의미
document.designMode	문서가 편집 가능한지를 문자열로 반환한다. 편집 가능하다면 “on”을, 블가능하다면 “off”를 반환한다. 값을 변경하여 편집 여부를 지정할 수 있다.

```
<iframe src="edit.html" width="300" height="80"></iframe>
```

```
    <script type="text/javascript">
        window.onload = function() {
            document.designMode = "on";
        };
    </script>
</head>
<body>
    Editing API로 구현하는 에디터
</body>
```



Text Selection API

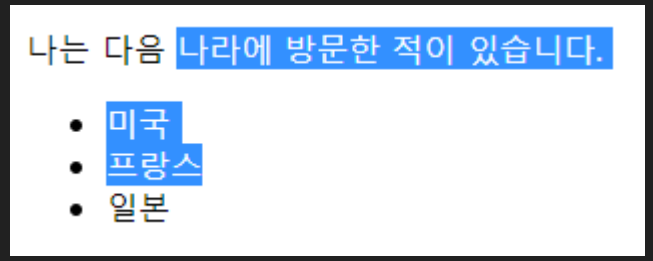
- Test Selection API
  - Test Selection API 는 선택된 텍스트의 영역을 Selection 객체로 제공한다.

메서드	의미
window.getSelection() document.getSelection()	선택한 택스트의 영역을 나타내는 Selection 객체를 반환한다. 또한 toString()으로 객체에 담긴 텍스트 영역을 연결한 문자열을 얻을 수 있다.

  - Selection 객체는 window 객체의 getSelection() 함수로 얻는다. 다만 기존 브라우저의 구현 상태를 반영하여 document 객체에도 getSelection() 함수가 있다.
  - Selection 객체에는 선택된 텍스트의 정보를 다루는 다양한 함수와 속성이 있다. toString()을 이용하여 텍스트 정보를 간단하게 얻을 수 있다.

Text Selection API

- Test Selection API 예제 - 1




```
<p>나는 다음 나라에 방문한 적이 있습니다. </p>
<ul>
    <li>미국 </li>
    <li>프랑스 </li>
    <li>일본 </li>
</ul>
<script type="text/javascript">
  // document오브젝트에 select이벤트 리스너를 지정
  document.addEventListener("mouseup", function() {
    // Selection오브젝트
    var selection = window.getSelection();
    // 선택된 텍스트
    var text = selection.toString();
    // 선택된 텍스트를 경고 창으로 표시
    if( text ) {
      alert( text );
    }
  });
</script>
```

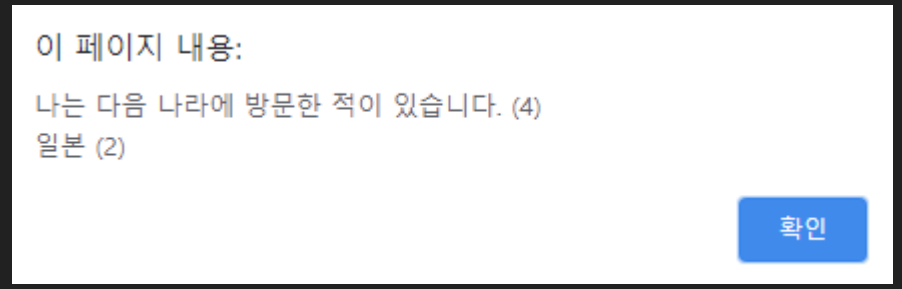
Text Selection API

- 선택 영역의 시작 위치와 종료 위치
  - 선택 영역의 시작 위치와 종료 위치에 관한 속성은 아래와 같다.

속성	의미
selection.isCollapsed	아무것도 선택되지 않으면 true를, 무엇인가 선택되어 있다면 false를 반환한다. 값 지정 불가
selection.anchorNode	선택 영역의 시작 위치를 포함한 DOM 노드 객체를 반환한다. 아무것도 선택되어 있지 않다면 null을 반환한다. 값 지정이 불가하며 선택 영역이 여러 개면 마지막 선택 영역이 반환 대상이 된다.
selection.anchorOffset	선택 영역의 시작 위치를 포함한 DOM 노드를 기준으로 해당 선택 영역의 시작 위치의 오프셋 값을 반환한다. 아무것도 선택되지 않으면 0을 반환한다. 값 지정이 불가하며 선택 영역이 여러 개면 마지막 선택 영역이 반환 대상이 된다.
selection.focusNode	선택 영역의 종료 위치를 포함한 DOM 노드 객체를 반환한다. 아무것도 선택되지 않았으면 null을 반환한다. 값 지정이 불가능하며 선택 영역이 여러 개면 마지막 선택 영역이 반환 대상이 된다.
selection.focusOffset	선택 영역의 종료 위치를 포함한 DOM 노드를 기준으로 그 선택 영역의 종료 위치의 오프셋 값을 반환한다. 아무것도 선택되지 않았으면 0을 반환한다. 값 지정이 불가능하며 선택 영역이 여러 개면 마지막 선택 영역이 반환 대상이 된다.

Text Selection API

- 선택 영역의 시작 위치와 종료 위치 예제




```
<p>나는 다음 나라에 방문한 적이 있습니다. </p>
<ul>
    <li>미국 </li>
    <li>프랑스 </li>
    <li>일본 </li>
</ul>
<script type="text/javascript">
    // document오브젝트에 mouseup이벤트 리스너 지정
    document.addEventListener("mouseup", function() {
        // Selection오브젝트
        var selection = window.getSelection();
        // 선택된 것이 없다면 종료
        if( selection.isCollapsed == true ) { return; }
        // 선택 영역의 시작위치를 포함하는 텍스트 노드
        var anchor_node = selection.anchorNode;
        // 선택영역의 시작 위치
        var anchor_offset = selection.anchorOffset;
        // 선택 영역의 종료 위치를 포함하는 텍스트 노드
        var focus_node = selection.focusNode;
        // 선택영역의 종료 위치
        var focus_offset = selection.focusOffset;
        // 선택 영역의 결과를 표시
        var msg = anchor_node.nodeValue + "(" + anchor_offset + ")\n";
        msg += focus_node.nodeValue + "(" + focus_offset + ")";
        alert(msg);
    }, false);
</script>
```

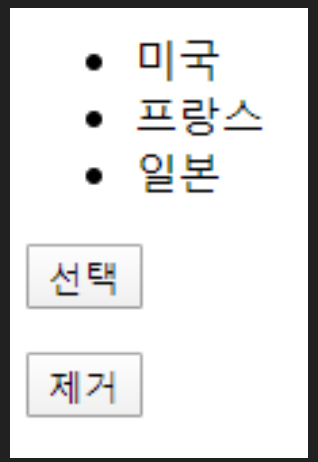
Text Selection API

- 선택 영역의 지정과 제거
  - 선택 영역의 지정과 제거에 관한 속성

메서드	의미
selection.selectAllChildren(parentNode)	현재 선택 영역을 매개변수로 지정한 요소 안에 있는 텍스트 전체로 변경한다
selection.deleteFromDocument()	현재의 선택 영역을 제거한다.

Text Selection API

- 선택 영역의 지정과 제거 예제




```
<ul>
    <li>미국 </li>
    <li>프랑스 </li>
    <li>일본 </li>
</ul>
<p><button id="button1">선택</button></p>
<p><button id="button2">제거</button></p>
<script type="text/javascript">
    // button요소에 click 이벤트 리스너 지정
    button1 = document.querySelector('#button1');
    button2 = document.querySelector('#button2');
    button1.addEventListener("click", function() {
        // ul요소
        var ul = document.querySelector('ul');
        // Selection오브젝트
        var selection = window.getSelection();
        // ul 요소의 텍스트 모두를 선택 상태로 하기
        selection.selectAllChildren(ul);
    }, false);
    // 요소 제거
    button2.addEventListener("click", function() {
        // Selection 오브젝트
        var selection = window.getSelection();
        // 선택 영역 제거
        selection.deleteFromDocument();
    }, false);
</script>
```

Text Selection API

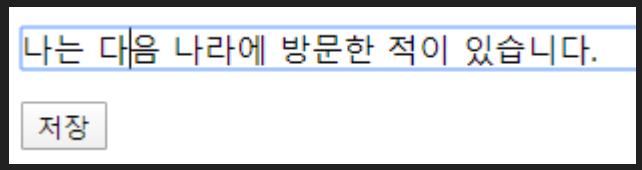
- 선택 영역 해제 및 커서의 위치 이동하기
  - 선택 영역 해제 및 위치 이동 함수

메서드	의미
selector.collapse(parentNode, offset)	현재의 선택 영역을 해제한다. 그리고 DOM 노드의 시작 위치와 종료 위치를 첫 번째 매개변수(parentNode)와 두 번째 매개변수(offset)로 지정한다. 즉 빈 선택 영역이 만들어져 커서를 지정한 위치로 옮긴다.
selector.collapseToStart()	현재 선택 영역의 종료 위치를 시작 위치와 같게 하여 선택 영역을 없앤다. 즉 커서의 위치를 선택된 영역의 시작 위치로 지정한다.
selector.collapseToEnd()	현재의 선택 영역의 시작 위치를 종료 위치와 같게 하여 선택 영역을 없앤다. 즉 커서의 위치를 선택된 영역의 종료 위치로 지정한다.

Text Selection API

- 선택 영역 해제 및 커서의 위치 이동하기 예제

```
<p id="p" contentEditable="false">나는 다음 나라에 방문한 적이 있습니다.</p>
<p><button>편집</button></p>
<script type="text/javascript">
    // button 요소에 click 이벤트 리스너를 지정
    button = document.querySelector('button');
    button.addEventListener("click", function() {
        // p요소
        var p = document.querySelector('#p');
        // 편집 가능 여부로 조건 분기
        if( p.contentEditable == "false" ) {
            // 편집 불가능 하다면 편집 가능으로 변경
            p.contentEditable = "true";
            // Selection 오브젝트
            var selection = window.getSelection();
            // p 요소의 텍스트를 모두 선택 상태로 하기
            selection.selectAllChildren(p);
            // 캐럿의 위치를 4번째 글자 다음으로 이동
            selection.collapse(p.firstChild, 4);
            // 버튼 표기 변경
            button.textContent = "저장 ";
        } else {
            // 편집 가능한 상태라면 편집 불가능상태로 변경
            p.contentEditable = "false";
            // 버튼의 표기 변경
            button.textContent = "편집 ";
        }
    }, false);
</script>
```



Text Selection API

- 여러 개의 선택 영역 다루기
  - 여러 개의 선택 영역을 다루는 속성 및 함수

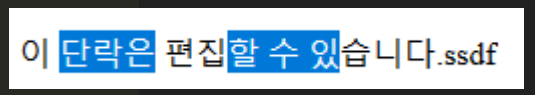
메서드	의미
selection.rangeCount	선택 영역의 수를 반환한다. 값 지정 불가
selection.getRangeAt(index)	매개변수(index) 번째에 담긴 선택 영역을 나타내는 range 객체를 반환한다. index는 0부터 시작
selection.addRange(range)	매개변수(range) 객체를 현재 선택 영역의 리스트에 추가한다.
selection.removeRange(range)	매개변수(range) 객체에 해당하는 범위의 현재 선택 영역 리스트에서 제거한다.
selection.removeAllRanges()	현재 선택 영역 리스트에 있는 모든 선택 영역을 제거한다.

  - 해당 함수들은 Gecko 기반의 브라우저가 아니면 제대로 결과가 나오지 않는다.
  - Ctrl을 통해 드래그를 해서 복수의 텍스트 드래그가 가능한 건 Gecko 브라우저 기반이다.(firefox)

Text Selection API

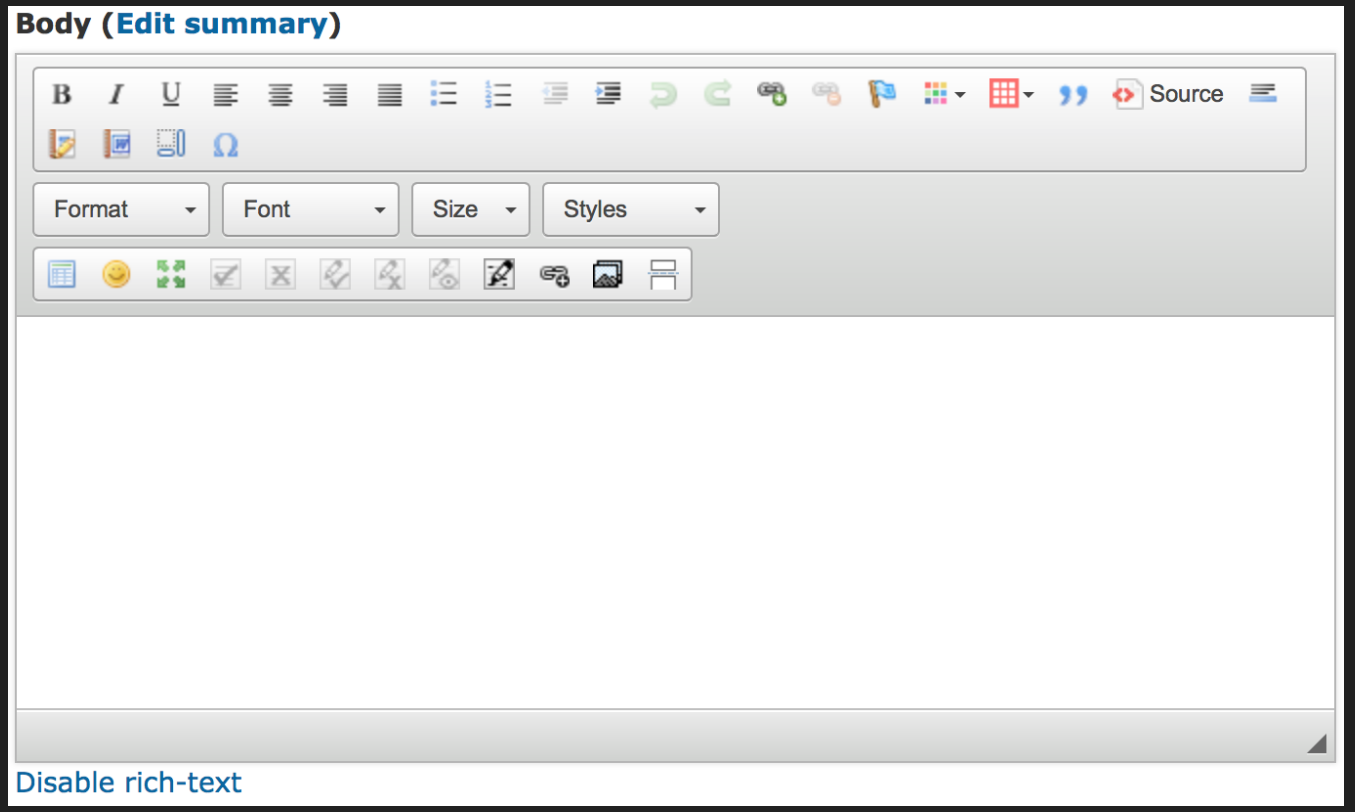
- 여러 개의 선택 영역 다루기 예제

```
<p contenteditable="true">이 단락은 편집할 수 있습니다.</p>
<script type="text/javascript">
    // p 요소에 mouseup 이벤트 리스너를 지정
    p = document.querySelector('p');
    p.addEventListener("mouseup", function() {
        // Selection 객체
        var selection = window.getSelection();
        // 선택 영역의 수
        var n = selection.rangeCount;
        // 선택 영역이 하나 이하라면 종료
        if( n <= 1 ) { return; }
        // 마지막 선택 영역
        var range = selection.getRangeAt(n-1);
        // 마지막 선택 영역을 제거
        selection.removeRange(range);
    }, false);
</script>
```



Editing API

- WISYWIG 편집기를 구현하는 Editing API



Editing API

- WISYWIG 편집기를 구현하는 Editing API
  - WISYWIG 편집기는 웹 페이지에서 텍스트 편집을 할 수 있어 마치 워드프로세서와 같이 폰트를 굵게 하거나 이텔릭체, 리스트 만들기 등과 같은 기능을 제공한다.
  - 관련 자바스크립트 라이브러리가 많이 있고 누구나 자신의 사이트에 WISYWIG 편집기를 넣어 textarea 요소를 UI를 가진 텍스트 편집 영역으로 만들 수 있다.
  - HTML5 는 이러한 WISYWIG 편집기를 웹 표준API로 만들 수 있도록 Editing API를 규정했다.

Editing API

- WISYWIG 편집기를 구현하는 Editing API
  - Editing API에 규정된 함수는 다음과 같다.

메서드	의미
document.execCommand(commandID[, showUI[, value]])	첫 번째 매개변수(commandID)에 지정한 명령을 현재 텍스트 선택 영역에 실행한다. 두 번째 매개변수와 세 번째 매개변수는 첫 번째 매개변수에 지정된 commandID 에 의존한다. 일반적으로 두 번째 매개변수는 명령에 대한 UI를 표시할지 여부를 나타내는 BOOL(true 또는 false)을 지정한다. 세 번째 매개변수는 명령이 필요로 하는 추가 정보를 지정한다.
document.queryCommandSupported(commandID)	매개변수(commandID)에 해당하는 작업을 브라우저가 지원하는 경우 true 를, 그렇지 않다면 false 를 반환한다.
document.queryCommandEnabled(commandID)	매개변수(commandID)에 해당하는 처리가 이 함수를 호출할 때 실행 가능하다면 true를 그렇지 않다면 false를 반환한다.
document.queryCommandIndeterm(commandID)	매개변수(commandID)에 해당하는 작업의 실행 가능 여부를 결정할 수 없다면 true를 그렇지 않다면 false를 반환한다. 기본적으로 HTML5 명세에 규정되어 있는 명령은 미확정 상태가 되지 않으므로 항상 false를 반환한다. commandID에 독자적인 키워드를 지정하면 다른 값을 반환한다.

Editing API

- WISYWIG 편집기를 구현하는 Editing API
  - Editing API에 규정된 함수는 다음과 같다.

메서드	의미
document.queryCommandState(commandID)	매개변수(commandID)에 해당 작업을 실행함에 있어 어떠한 상태를 true 또는 false 값으로 반환한다. 이것은 지정한 commandID 에 의존한다. 대부분의 명령에 이 함수는 의미가 없고, 항상 false 를 반환하게 규정되어 있지만 commandID 가 bold, italic, subscript, superscript 일 때 특정 상태에 따라 BOOL(true 또는 false)을 반환하도록 규정되어 있다.
document.queryCommandValue(commandID)	매개변수(commandID)에 해당 작업을 실행함에 있어 어떠한 값을 반환한다. 이것은 commandID 에 의존한다. 대부분의 명령에 이 함수는 의미가 없고, 항상 문자열 "false"를 반환하게 되어 있지만 commandID 가 bold, italic, subscript, superscript 일 때 queryCommandState() 함수의 값으로 연동한 문자열 "true" 또는 "false" 를 반환하게 되어있다.

  - commandID 에 지정할 수 있는 키워드는 HTML5 명세에 정해져 있다.

Editing API

- Editing API 예제 - 1

```
<body onkeydown="myFunction(event)">
    <h2>Select Text and Press Shift</h2>
    <p>Select some text in this page, and press the SHIFT button to make the selected text toggle between
    bold and normal.</p>

    <script type="text/javascript">
        document.designMode = "on";

        console.log("queryCommandSupported : "+document.queryCommandSupported("bold"));

        function myFunction(event) {
            if (event.keyCode == 16) {
                // Execute command if user presses the SHIFT button:
                document.execCommand("bold");
            }
        }
    </script>
```

Editing API

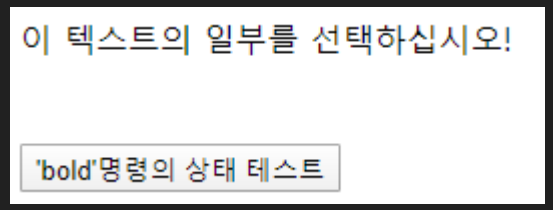
- Editing API 예제 - 2

```
<h2>all select this content</h2>

<script type="text/javascript">
    function myfunction(){
        var flg = document.queryCommandEnabled("SelectAll");

        if(flg) {
            document.execCommand("SelectAll", false, null);
        }
    }
</script>
```

Editing API

- Editing API 예제 - 3

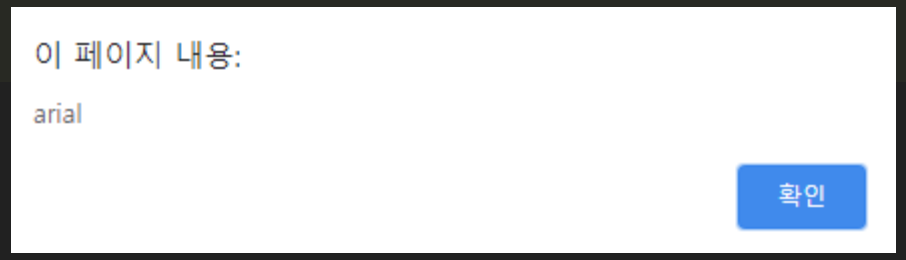
```
<div contenteditable="true">이 텍스트의 일부를 선택하십시오!</div>
<br /><br />
<button onclick="SetToBold ();">'bold'명령의 상태 테스트</button>
<script type="text/javascript">
    function SetToBold () {
        if (document.queryCommandIndeterm ("bold")) {
            alert ("'bold'명령이 미정의 상태입니다.");
        }
        else {
            if (document.queryCommandState ("bold")) {
                alert ("bold 글씨체는 선택된 텍스트에서 제거됩니다.");
            }
            else {
                alert ("선택한 텍스트는 굵게 표시됩니다.");
            }
        }
        document.execCommand ('bold', false, null);
    }
</script>
```

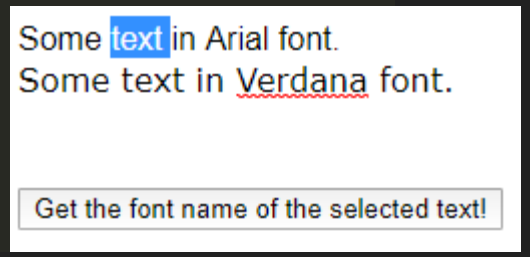


Editing API

- Editing API 예제 - 3

```
<div contenteditable="true">
    <span style="font-family:arial;">Some text in Arial font.</span>
    <br/>
    <span style="font-family:verdana;">Some text in Verdana font.</span>
</div>
<br/><br/>
<button onclick="GetFont();">Get the font name of the selected text!</button>
<script type="text/javascript">
    function GetFont(){
        alert (document.queryCommandValue ('fontname'));
    }
</script>
```





Editing API

- 명령

commanded	설명	document.execCommand() 적용 방법
bold	선택한 텍스트를 굵게 하거나, 원래대로 돌린다.	execCommand("bold",false,null) ※
italic	선택한 텍스트 영역을 이텔릭으로 하거나 해제한다.	execCommand("italic",false,null) ※
subscript	선택된 텍스트 영역을 아래 첨자로 하거나 해제한다.	execCommand("subscript",false,null) ※
superscript	선택된 텍스트 영역을 위 첨자로 하거나 해제한다.	execCommand("superscript",false,null) ※
delete	선택한 텍스트 영역을 삭제한다. 만약 선택 영역이 없고 커서만 있으면 커서 앞의 한 문자를 삭제한다. 윈도우 PC의 키보드라면 백스페이스 키를 누른 것과 같은 효과이다.	execCommand("delete",false,null) ※
forwardDelete	선택한 텍스트 영역을 삭제한다. 만약 선택 영역이 없고 커서 뒤의 한 문자를 삭제한다. 윈도우 PC의 키보드라면 delete키를 누른 것과 같은 효과이다.	execCommand("forwardDelete",false,null) ※
insertImage	이미지를 새로 생성하고 텍스트 선택 영역을 해당 이미지로 바꾼다. 커서가 있으면 그 위치에 이미지가 삽입된다.	이미지 URL을 사용자에게 물어보고 이미지를 삽입하기 execCommand("insertImage",true,null) ※ 이미지 URL을 지정하여 이미지를 삽입하기 execCommand("insertImage",false,null) ※
insertHTML	선택한 텍스트 영역을 지정한 HTML 코드로 바꾼다. 만약 선택 영역이 없고 커서만 있으면 해당 위치에 지정된 HTML 코드를 삽입한다.	execCommand("insertHTML",false,HTMLcode) 두 번째 매개변수는 사용하지 않지만 세 번째 매개변수는 필수이므로 두 번째 매개변수에 false를 지정해야 한다. 세 번째 매개변수에는 삽입할 HTML 코드를 문자열로 지정한다.

Editing API

- 명령

commanded	설명	document.execCommand() 적용 방법
insertText	커서 위치에서 블록으로 분할한다.	execCommand("insertText",false,text) 두 번째 매개변수는 사용하지 않지만 세번째 매개변수는 필수이므로 두 번째 매개변수에 false를 지정해야 한다. 세 번째 매개변수에는 삽입할 텍스트를 지정한다.
insertLineBreak	커서 위치에서 줄바꿈한다.	execCommand("insertLineBreak",false,null)
insertOrderedList	선택된 텍스트 영역을 ol요소, li 요소를 사용하여 순서 목록으로 구성한다.	execCommand("insertOrderedList",false,null)
insertUnorderedList	선택된 텍스트 영역을 ul요소, li 요소를 사용하여 순서 목록으로 구성한다.	execCommand("insertUnorderedList",false,null)
formatBlock	편집 가능한 콘텐츠 중에서 선택한 텍스트의 부모 요소를 지정된 형식 블록 후보 요소로 바꾼다.	execCommand("formatBook",false,tagname) 두 번째 매개변수는 사용하지 않지만 세 번째 매개변수가 필수이므로 두 번째는 false를 지정한다. 세 번째 매개변수로는 태그 이름을 문자열로 지정한다. 세 번째 매개변수로 지정 가능한 태그 이름은 포멧 블록 후보로 규정된 요소 이름에 한한다. tagname 예 : section, nav, article, aside, h1 ~ h6, hgroup, header, footer, address, p, pre, blockquote, div
insertParagraph	커서의 위치에서 블록으로 분할한다.	execCommand("insertParagraph",false,null)

Editing API

- 명령

commanded	설명	document.execCommand() 적용 방법
createLink	선택한 텍스트 영역에 링크를 연결하거나 해제한다.	- 링크 URL을 사용자에게 물어보고 링크 생성하기 execCommand("createLink", true, null) - 링크 URL을 직접 지정하여 링크 생성하기 execCommand("createLink", false, url)
unlink	선택한 텍스트가 a 요소의 일부 또는 a 요소를 완전히 포함하면 해당 a 요소의 연결을 해제(a 요소의 시작 태그와 종료 태그를 제거)한다.	execCommand("unlink", false, null)
selectAll	포커스가 닿는 편집 가능한 영역의 콘텐츠 모두를 선택 상태로 한다.	execCommand("selectAll",false,null)
unselect	포커스가 닿는 편집 가능한 영역의 콘텐츠의 선택 상태를 해제한다.	execCommand("unselect",false,null)
undo	이전 명령의 실행을 취소한다.	execCommand("undo",false,null)
redo	이전 명령의 실행 취소(undo)를 되돌린다.	execCommand("redo",false,null)

Editing API

- sun editor
  - https://github.com/JiHong88/SunEditor