JAVASCRIPT

Event Driven Model

- 이벤트 처리
  - 브라우저에 표시된 컴포넌트에 마우스를 올리거나 클릭하거나 하는 등의 동작을 하면 브라우저에서는 이벤트(사건)라는 것이 동작되도록 만들어져 있다.
  - 이벤트 발생 시 어떤 동작을 추가하기 위해 자바스크립트를 활용하게 된다.

- 이벤트 드리븐 모델
  - 브라우저에 보여지는 페이지 내에서 발생한 여러 이벤트에 대해 대응하는 처리(이벤트 핸들러)를 위해 호출할 함수를 작성해 두어 실행되도록 하는 형태의 모델

- 이벤트 핸들러
  - 이벤트 발생 시 동작할 내용을 정의한 함수

브라우저 주요 이벤트

- 자바스크립트로 사용이 가능한 브라우저의 주요 이벤트
  - ()괄호는 이벤트의 주 대상이 되는 요소

[읽기]
abort : 이미지 로딩 중단 시 (img)
load : 페이지, 이미지의 로딩 완료 시 (body, img)
unload : 다른 페이지로 이동 시 (body)

[마우스]
click : 클릭 시
dbclick : 더블클릭 시
mousedown : 마우스로 누를 때
mousemove : 마우스 포인터가 이동할 때
mouseout : 마우스가 컴포넌트 밖으로 나갔을 때
mouseover : 컴포넌트에 마우스가 올라갔을 때
mouseup : 마우스를 놓을 때
contextmenu : context menu가 표시되기 전(body)

[키]
keydown : 키를 누를 때
keypress : 키를 누른 상태일 때
keyup : 키를 놓을 때

[폼]
change : 내용이 변경되었을 때 (input, select)
reset : 리셋될 때 (form)
submit : 서브밋 될 때 (form)

[포커스]
blur : 포커스를 잃을 때
focus : 포커스를 얻을 때

[기타]
resize : 사이즈를 변경 했을 때
scroll : 스크롤 했을 때 (body)

이벤트 처리 방법

- 이벤트와 이벤트 핸들러를 연결하기(이벤트 처리의 핵심)
  - 어떤 요소에서
  - 어떤 이벤트가 발생 할 때
  - 어떤 이벤트 핸들러 함수를 연결할 것인가

- 처리방법

```
<태그명 on이벤트명="핸들러함수호출">
```

이벤트 처리 방법

- <script>태그는 <head>태그 하위에 작성

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <script>
        function btn_onclick() {
            window.alert('버튼이 클릭되었다');
        }
    </script>
</head>
<body>
    <input type="button" value="다이얼로그 표시" onclick="btn_onclick()" />
</body>

</html>
```

이벤트 처리 방법

- 단순한 처리라면 별도의 함수를 정의하지 않고 바로 정의 가능
  - 단순한 처리가 아닌 복잡한 코드를 기술하는 것은 피해야 한다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <script>

    </script>
</head>
<body>
    <input type="button" value="다이얼로그 표시"
        onclick="window.alert('버튼이 클릭되었다');" />
</body>
</html>
```

이벤트 처리 방법

- 자바스크립트의 코드 내에서 선언하는 방법 – 프로퍼티로써 설정 –
  - HTML태그는 문서의 레이아웃을 정의하는 용도이므로 자바스크립트 코드를 섞어서 사용하는 것은 좋은 방법이 아니다
  - 이벤트와 이벤트 핸들러의 선언을 나누어서 자바스크립트 내에서 작성할 수 있다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <script>
        window.onload = function () {
            document.getElementById('btn').onclick = function () {
                window.alert('버튼이 클릭되었다.');
            };
        }
    </script>
</head>
<body>
    <input id="btn" type="button" value="다이얼로그 표시" />
</body>
</html>
```

이벤트 처리 방법

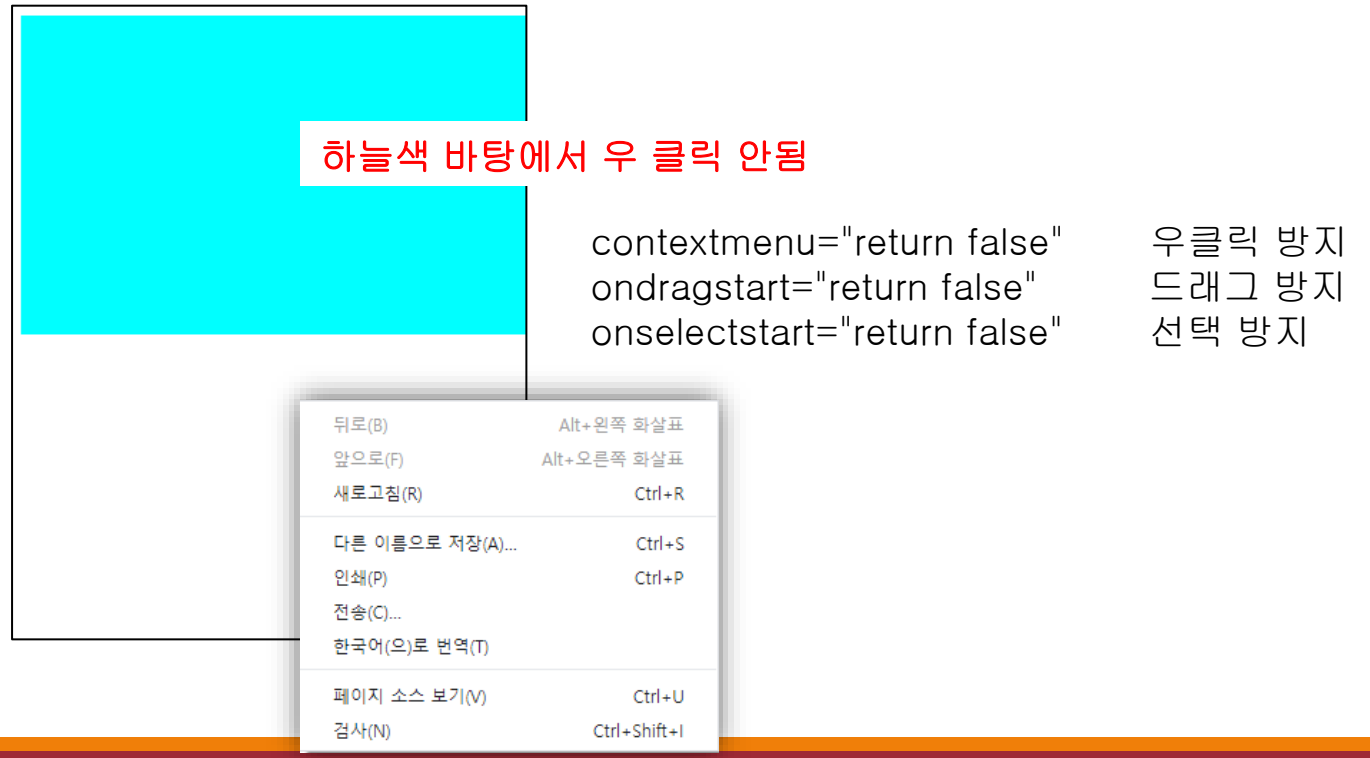
- 이벤트를 등록하는 방법
  - <body> 태그를 대상으로 적용
    - window.<on이벤트명>=function(){ …처리코드… }
    - window.<on이벤트명>=함수명
  - 특정 태그 요소에 적용
    - document.getElementById('ID값').<on이벤트명>=function(){ …처리코드… }
    - document.getElementById('ID값').<on이벤트명>=함수명
- 주의
  - 이벤트 이름은 모두 소문자로 작성할 것
  - 이벤트 핸들러로 등록하는 것은 함수 호출이 아닌 함수 객체
  - 개별 요소의 이벤트 핸들러는 onload핸들러 아래에 정의 (onload가 끝나야 id를 불러올 수 있다.)

이벤트 처리 방법

❖ 이벤트 표준 동작 취소하기(동작 시키지 않기)

- Anchor태그 같은 경우 클릭 시 링크를 요청하게 된다.
- form내에서 submit같은 경우 폼 내용을 지정된 action으로 전송하며 요청한다.
- 이러한 기본 동작을 하지 않도록 만들어야 할 경우가 있다.

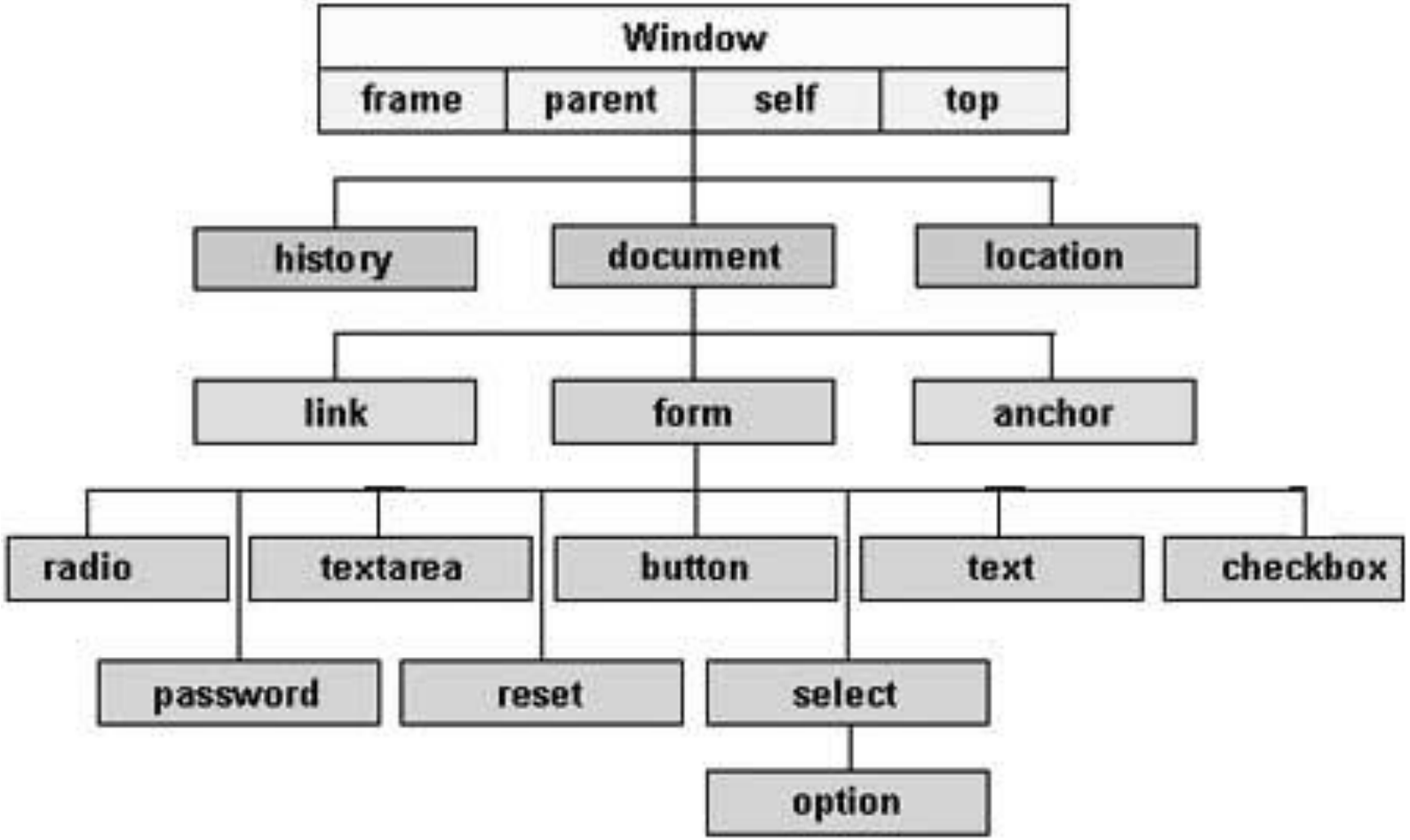
```
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <style>
        div{
            background-color: aqua;
            height: 200px;
        }
    </style>
</head>

<body oncontextmenu="return false;">
    <div></div>
</body>
```



contextmenu="return false" 우클릭 방지
ondragstart="return false" 드래그 방지
onselectstart="return false" 선택 방지

BOM

- Browser Object Model



BOM

- 모든 브라우저 객체는 최상위에 위치하는 Window객체를 통해 접근
  - Window객체의 하위에 존재하는 document, history, location 객체를 참조
  - 실제 사용 시 자바스크립트에서는 window객체를 명시할 필요가 없다.

- 기본적으로 브라우저 객체에 접근하는 방법
  - document.write('hello!');

- Window객체를 이용하는 방법으로는 다음과 같다.
  - **대문자 W**indow.document.write('hello!'); 이 방법은 에러이다.

- Window객체에 접근하려면 window프로퍼티로 접근한다.
  - **소문자 w**indow.document.write('hello'); 이 방법으로 접근이 가능.(의미는 없다.)

BOM

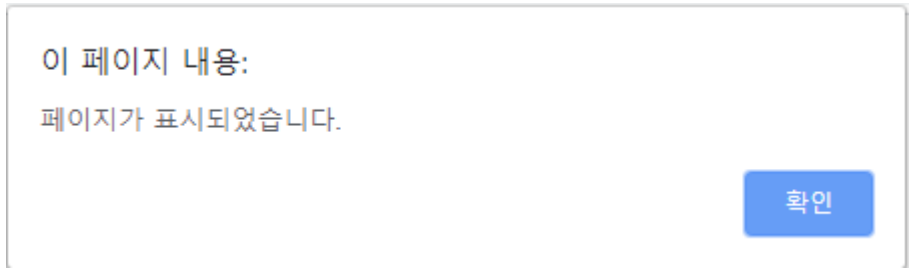
- 다음과 같은 개념은 알고 있도록 한다.
  - Window객체 접근
    - window : Window객체를 참조하는 프로퍼티
  - Document객체 접근
    - document : Document객체를 참조하는 프로퍼티
  - Location 객체 접근
    - location : Location객체를 참조하는 프로퍼티
  - History객체 접근
    - history : History객체를 참조하는 프로퍼티

BOM – Window 객체

- Window 객체는 윈도우 조작과 다이얼로그, 타이머 등을 위한 기능 제공
- Window표기는 생략이 가능함

- 다이얼로그 표시
  - alert() : 지정된 메시지를 경고 다이얼로그로 표시.
  - 평문과 escape sequence만을 포함할 수 있음(₩n, ₩t, ₩', ₩" 등)
  - HTML태그를 포함할 경우 그대로 표시됨.

```
<body onload="window.alert('페이지가 표시되었습니다.');">

</body>
```



BOM – Window 객체

❖ 다이얼로그 표시

- confirm() : 사용자에게 의사를 물어보는 창을 표시한다.

```
<form onsubmit="return window.confirm('제출하시겠습니까?');">
    <input type="submit" value="데이터" />
</form>
```

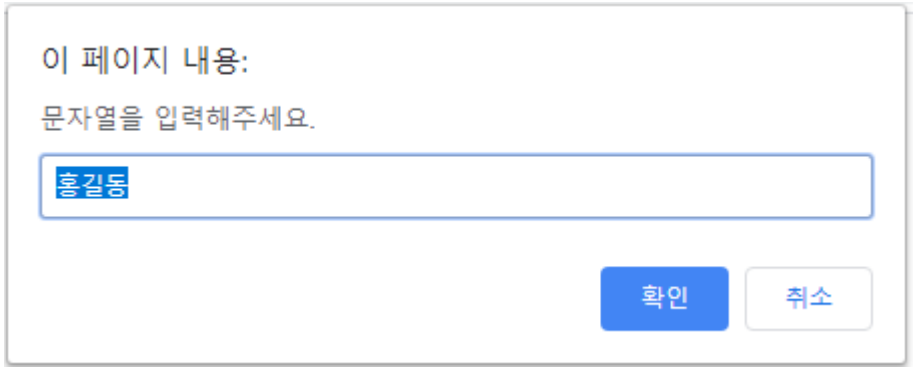
확인이 선택되면 true반환
취소가 선택되면 false반환

BOM – Window 객체

❖ 다이얼로그 표시

- prompt() : 사용자에게 입력할 수 있는 다이얼로그를 표시
- 사용자가 입력한 문자열을 반환한다.
- 브라우저에 따라 설정이 필요할 수 있음.

```
<script>
    window.onload = function() {
        let input = window.prompt('문자열을 입력해주세요.', '홍길동');
        window.alert('입력값 : ' + input);
    };
</script>
```



BOM – Window 객체

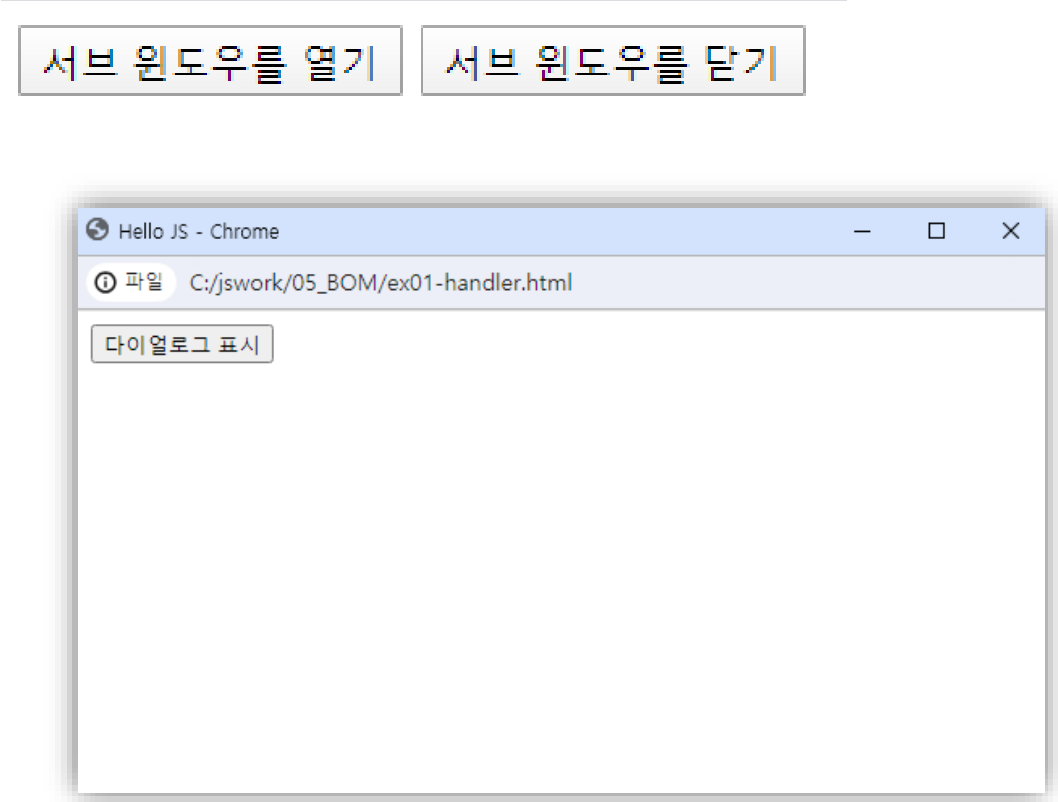
- 자식 창(window) 생성하기
  - open메서드는 alert으로 표현하기 어려운 복잡한 내용을 표시할 경우에 사용
  - 변수 = window.open(URL, 윈도우이름, 화면구성옵션)
  - 윈도우이름은 앵커의 target과 같은 속성에서 지정할 수 있는 이름
  - 옵션은 옵션=값 형식으로 지정

옵션	설명
width	윈도우 폭
height	윈도우 높이
location	주소바 표시 여부(yes | no)
scrollbars	스크롤바 표시 여부(yes | no)
resizable	창 크기 변경 가능 여부(yes | no)
toolbar	툴바 표시 여부(yes | no)
status	상태바 표시 여부(yes | no)
menubar	메뉴바 표시 여부(yes | no)

BOM – Window 객체

❖ 예제(부모창과 자식창의 URL의 도메인이 같아야 close가능)

```
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <script>
        let subwin;

        function win_open() {
            subwin = window.open('./ex01-handler.html', 'Google',
                'width=600, height=300, scrollbars=yes, location=yes');
        }

        function win_close() {
            if (subwin && !subwin.closed) {
                subwin.close();
            } else {
                alert('서브 윈도우가 열려 있지 않습니다.');
            }
        }
    </script>
</head>
<body>
    <input type="button" value="서브 윈도우를 열기" onclick="win_open()" />
    <input type="button" value="서브 윈도우를 닫기" onclick="win_close()" />
</body>
```



BOM – Window 객체

❖ 윈도우 사이즈나 스크롤 위치 조작

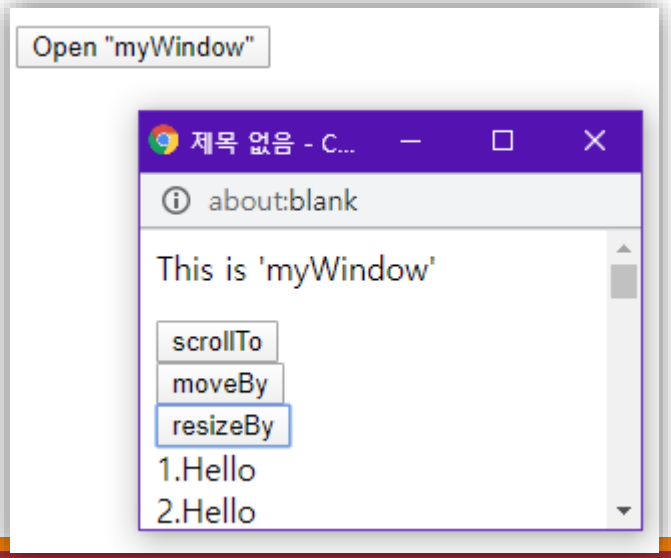
- Window객체에 윈도우 사이즈나 위치를 조작할 수 있는 프로퍼티와 메서드가 내장되어 있다.

멤버	설명
moveBy(x, y)	윈도우 위치를 이동(상대 좌표)
moveTo(x, y)	윈도우 위치를 이동(절대 좌표)
resizeBy(x, y)	윈도우 사이즈 변경(상대 사이즈)
resizeTo(x, y)	윈도우 사이즈 변경(절대 사이즈)
scrollBy(x, y)	윈도우 내용을 스크롤(상대 위치)
scrollTo(x, y)	윈도우 내용을 스크롤(절대 위치)
blur()	윈도우로부터 포커스를 벗어남
focus()	윈도우에 포커스 맞춤(액티브 상태로)
print()	윈도우 내용을 인쇄

BOM – Window 객체

❖ 예제

```
    <script>
        function openWin() {
            myWindow = window.open("", "myWindow", "width=200, height=100");
            myWindow.document.write("<p>This is 'myWindow'</p>");
            myWindow.document.write('<input type="button" value="scrollTo" onclick="window.scrollTo(0, 200)"/><br>');
            myWindow.document.write('<input type="button" value="moveBy" onclick="moveBy(100, 100)"/><br>');
            myWindow.document.write('<input type="button" value="resizeBy" onclick="window.resizeBy(50, 50)"/><br>');
            for(let i = 0; i < 1000; i++){
                myWindow.document.write( i + 1 + '.Hello <br>');
            }
        }
    </script>
</head>

<body>
    <button onclick="openWin()">Open "myWindow"</button>
</body>
```



BOM – Window 객체

❖ 타이머 기능 구현하기

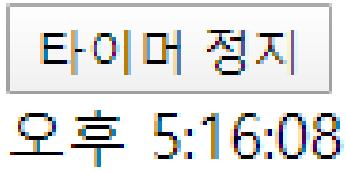
- setInterval() : 지정된 시간 간격으로 처리를 반복적으로 수행
- setTimeout() : 지정된 시간이 경과 했을 때 한 번만 처리

- 위 두 메서드는 타이머를 식별하는 고유ID를 반환함.
- 이를 이용하여 clearInterval(timer), clearTimeout(timer)메서드를 호출하여 타이머 해제가능

BOM – Window 객체

- 예제 – setInterval()

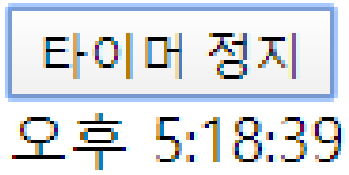
```
    <script>
      let timer;

      window.onload = function () {
        timer = window.setInterval(function () {
          let date = new Date();
          document.getElementById("result").innerHTML =
            date.toLocaleTimeString();
        }, 1000);
      };
    </script>
  </head>

  <body>
    <input type="button" value="타이머 정지" onclick="clearInterval(timer)" />
    <div id="result"></div>
  </body>
```



BOM – Window 객체

❖ 예제 – setTimeout()

```
  <script>
    let timer;

    window.onload = function () {
      timer = window.setTimeout(function () {
        let date = new Date();
        document.getElementById("result").innerHTML =
          date.toLocaleTimeString();
      }, 3000);
    };
  </script>
</head>

<body>
  <input type="button" value="타이머 정지" onclick="clearTimeout(timer)" />
  <div id="result"></div>
</body>
```



BOM – Form 객체

- <form>
  - 사용자의 입력 값을 받는 요소를 묶는 태그
  - Form객체를 이용하면 입력 요소에 대한 처리를 할 수 있음
  - Form요소에 접근하는 방법은 DOM(Document Object Model)을 이용하는 것.
  - 단순히 Form의 값을 취하기 위한 것이라면 Form객체를 활용하는 것이 간단함

- Form 객체를 이용하여 form태그 요소에 접근하는 방법
  - form태그에 지정한 name속성에 할당된 값으로 접근
  - forms배열 객체를 이용하여 접근
    (forms는 문서에 포함된 모든 form태그를 가짐)
  - elements배열 객체를 이용하여 접근
    (elements는 폼 태그 하위에 지정된 모든 태그 요소를 가짐)

BOM – Form 객체

❖ 예제 – 주석 처리된 내용 모두 동일한 입력 요소를 지칭

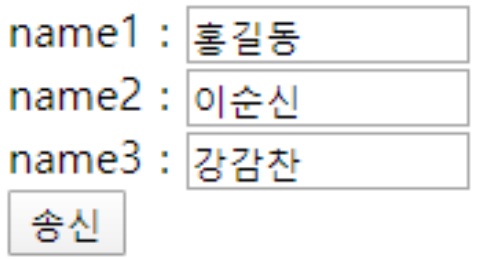
```
  <script>
    function process() {
      const name = document.fm.name.value;
      //const name = document.forms[0].elements[0].value
      //const name = document.forms['fm'].elements['name'].value
      //const name = document['fm']['name'].value
      window.alert("안녕하세요, " + name + "씨!");
      return false;
    }
  </script>
</head>

<body>
  <form name="fm" onsubmit="return process()">
    이름 :
    <input type="text" name="name" size="10" />
    <input type="submit" value="송신" />
  </form>
</body>
```

이름 : 강사 송신

BOM – Form 객체

- 폼 태그 내의 요소에 차례대로 접근하는 경우 수치를 활용하여 반복도 가능

```
<script>
  function process() {
    let result = "";
    for (let i = 0; i < document.fm.elements.length; i++) {
      result += document.fm.elements[i].value + "\n";
    }
    window.alert(result);
    return false;
  }
</script>
</head>

<body>
  <form name="fm" onsubmit="return process()">
    <label>name1:<input type="text" name="name1" size="10" /></label><br />
    <label>name2:<input type="text" name="name2" size="10" /></label><br />
    <label>name3:<input type="text" name="name3" size="10" /></label><br />
    <input type="submit" value="제출" />
  </form>
</body>
```



BOM – Form 객체

- 별도의 윈도우에 있는 요소에 접근하고자 할 경우(부모 – 자식)
  - subwin.opener.document.fm.이름 -> 자식윈도우에서 부모 윈도우의 요소를 참조
  - subwin.document.fm.이름 -> 부모윈도우에서 자식 윈도우의 요소를 참조

분류	요소이름	설명
텍스트	Text(<input type="text">)	텍스트 입력 상자(싱글 라인)
	Password(<input type="password">)	비밀번호 입력 상자(입력 문자는 * 문자로 표시)
	Textarea(<input type="textarea">)	텍스트 입력 상자(멀티 라인)
라디오/체크	Radio(<input type="radio">)	라디오 버튼(단일 선택)
	Checkbox(<input type="checkbox">)	체크박스(복수 선택)
선택	Select(<select>)	선택 박스(단일 선택
	Select(<select multiple>)	리스트 박스(복수 선택 지원)
버튼	Submit(<input type="submit">)	제출 기능(form요소의 입력 값 모두를 서버로 전송)
	Reset(<input type="reset">)	리셋(form요소에 입력 값을 모두 초기화)
	Button(<input type="button">)	기본 버튼
기타	File(<input type="file">)	파일 선택
	Hidden(<input type="hidden">)	브라우저에서 표시하지 않는 필드

BOM – Form 객체

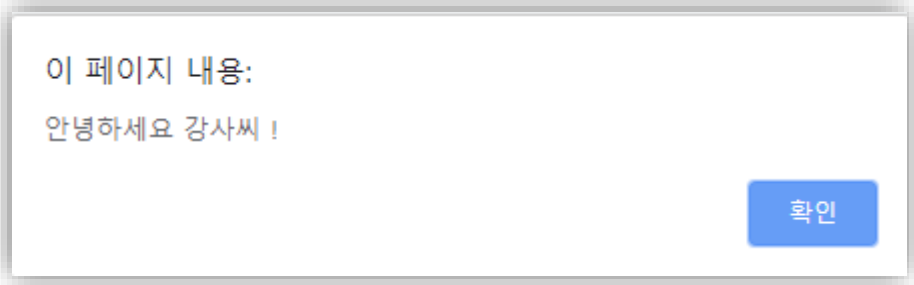
- 폼 요소들이 각각 가지는 프로퍼티나 메서드는 약간씩의 차이가 있음
- 아래 내용은 공통으로 사용 가능한 멤버이다.
  - *은 읽기 전용 값

멤버	설명
form	요소가 속하는 폼 객체
disabled	요소의 입력/선택 금지
*name	요소를 식별하는 이름
*type	요소의 종류
value	요소의 값(select에서는 이용불가)
focus()	요소에 포커스 지정
blur()	요소의 포커스 해제

BOM – Form 객체

❖ 예제

```
    <script>
      function process(f) {
        window.alert("안녕하세요 " + f.name.value + "씨 ! ");
      }
    </script>
  </head>
  <body>
    <form name="fm">
      이름 :
      <input type="text" name="name" size="10" />
      <input type="button" value="송신" onclick="process(this.form)" />
    </form>
  </body>
```



BOM – Form 객체

❖ 예제 – 비활성화 시키기

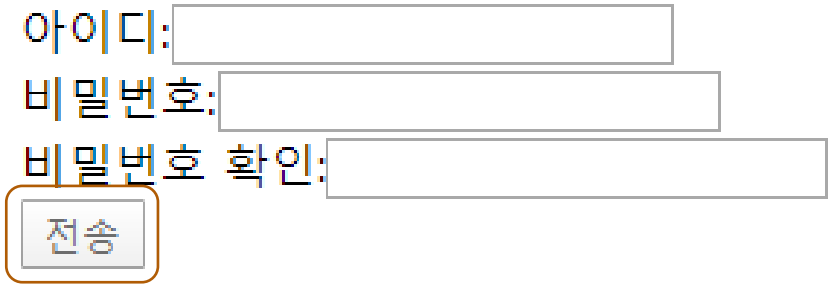
```
  <script>
    function process(f) {
      let fmt = document.fm.fmt;
      for (let i = 0; i < fmt.length; i++) {
        fmt[i].disabled = !document.fm.need.checked;
      }
    }
  </script>
</head>

<body>
  <form name="fm">
    <label>
      뉴스 이메일 구독 여부 :
      <input type="checkbox" name="need" value="1" checked="checked" onclick="process()" />
    </label>
    <br />
    <label><input type="radio" name="fmt" value="plain" />텍스트형식</label>
    <label><input type="radio" name="fmt" value="html" />HTML형식</label>
  </form>
</body>
```

- disabled는 Submit시 Submit버튼을 무효화 하여 이중 제출을 막는 경우에도 사용 가능.

Quiz

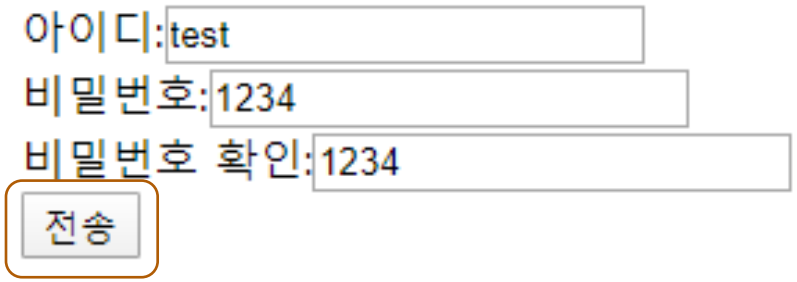
❖ 다음 HTML페이지를 이용하여 이벤트 처리를 구현하세요.

```
<body>
    <form>
        아이디:<input type="text"/><br>
        비밀번호:<input type="text"/><br>
        비밀번호 확인:<input type="text"/><br>
        <input type="submit" value="전송"/>
    </form>
</body>
```

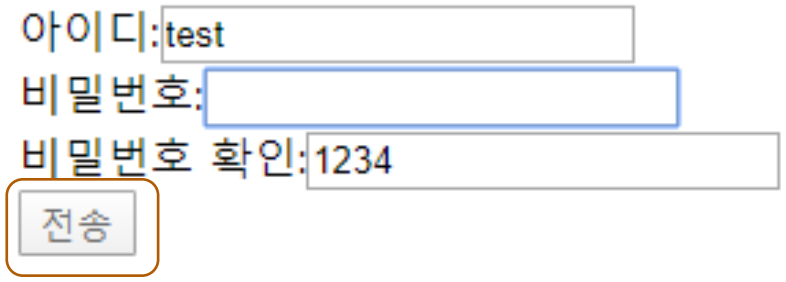


기본 전송버튼 비활성화

텍스트 요소에 모든 입력이 되면 활성화 됨



텍스트 요소에 빈 값이 있으면 다시 비활성화



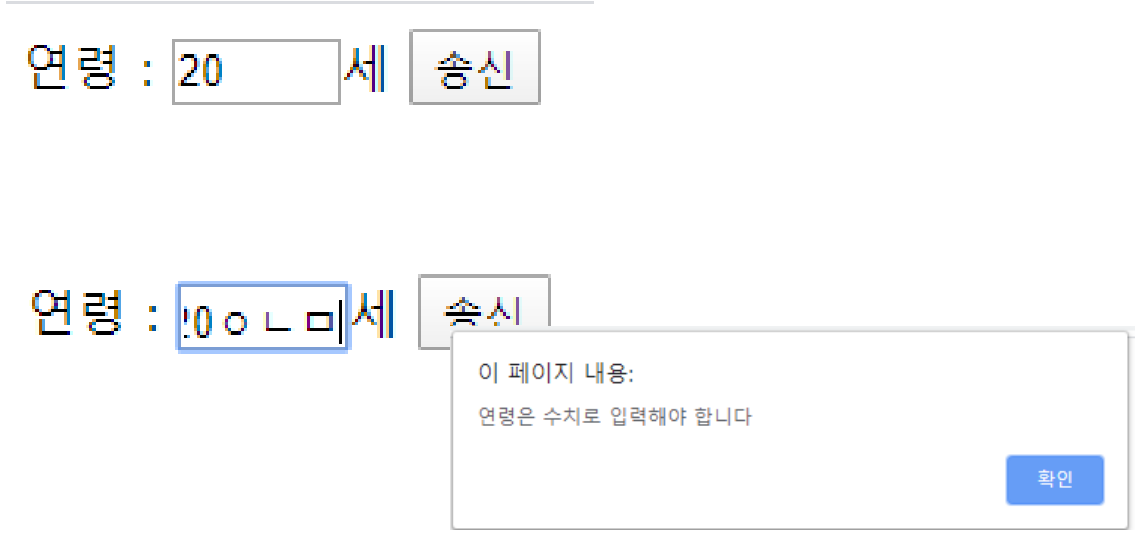
BOM – Form 객체

- 텍스트 요소 조작하기
  - text, textarea, password에 대한 조작이다.
  - 해당 요소들은 다음과 같은 멤버를 갖고 있다.
  - defaultValue : 기본 값
  - select() : 텍스트 선택 상태

BOM – Form 객체

❖ 예제

```
<script>
  function chk() {
    let q = document.fm.old;
    if (q.value != null && q.value != "") {
      if (isNaN(q.value)) {
        window.alert("연령은 수치로 입력해야 합니다");
        q.value = q.defaultValue;
        q.focus();
        q.select();
        return false;
      }
    }
  </script>
</head>
<body>
  <form name="fm" method="POST" action="" onsubmit="return chk()">
    <label>나이<input type="text" name="old" value="20" size="3" />세</label>
    <input type="submit" value="제출" />
  </form>
</body>
```



이 예시의 기능은 범용적인 개념이라 외부 라이브러리 형태로 따로 정의하는 것이 일반적
여기서는 개념 확인을 위해 작성한 예시

BOM – Form 객체

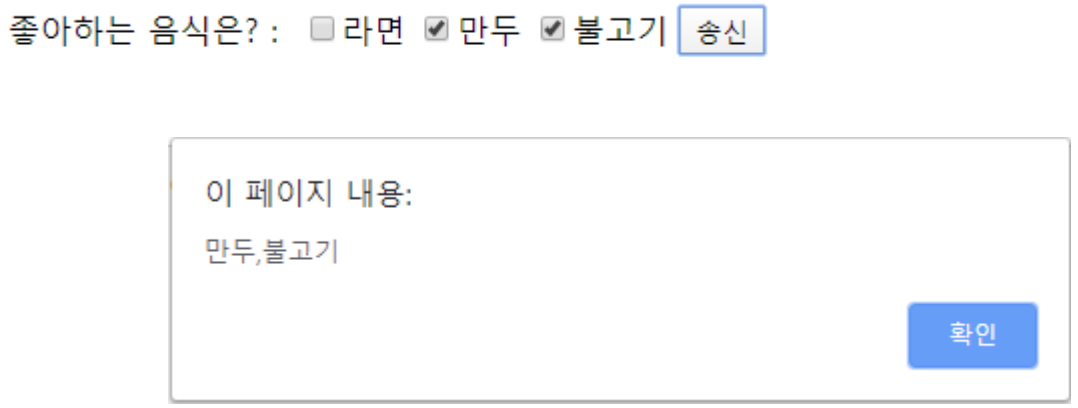
- 라디오 버튼/체크 박스 다루기
  - 체크 박스 -> 복수 선택 가능(선택한 항목에 대한 단일 해제 가능)
  - 라디오 버튼 -> 단일 선택만 가능(한번 선택 시 다른 선택으로 변경은 가능하나 해제는 불가)

- 중요
  - 같은 이름의 요소는 배열로 처리된다.
  - 체크 상태를 나타내는 것은 checked 프로퍼티이다.

BOM – Form 객체

❖ 예 제

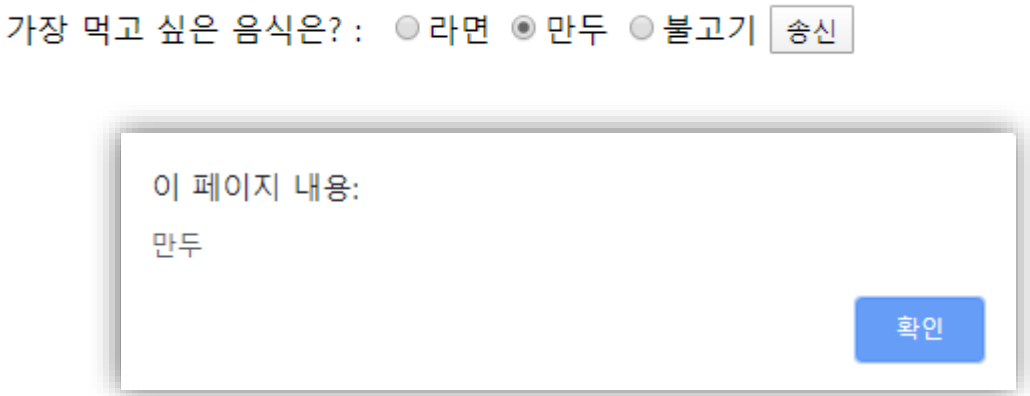
```
  <script>
    function show() {
      let result = [];
      let f = document.fm.food;
      for (let i = 0; i < f.length; i++) {
        if (f[i].checked) {
          result.push(f[i].value);
        }
      }
      window.alert(result.toString());
      return false;
    }
  </script>
</head>

<body>
  <form name="fm" method="POST" action="" onsubmit="return show()">
    좋아하는 음식은?:
    <label><input type="checkbox" name="food" value="라면" />라면</label>
    <label><input type="checkbox" name="food" value="만두" />만두</label>
    <label><input type="checkbox" name="food" value="불고기" />불고기</label>
    <input type="submit" value="제출" />
  </form>
</body>
```



BOM – Form 객체

❖ 예제

```
<script>
  function getRadio(elem) {
    let result = "";
    for (let i = 0; i < elem.length; i++) {
      if (elem[i].checked) {
        result = elem[i].value;
        break;
      }
    }
    return result;
  }

  function show() {
    window.alert(getRadio(document.fm.food));
    return false;
  }
</script>
</head>

<body>
  <form name="fm" method="POST" action="" onsubmit="return show()">
    가장 먹고 싶은 음식은?:
    <label><input type="radio" name="food" value="라면" />라면</label>
    <label><input type="radio" name="food" value="만두" />만두</label>
    <label><input type="radio" name="food" value="불고기" />불고기</label>
    <input type="submit" value="제출" />
  </form>
</body>
```



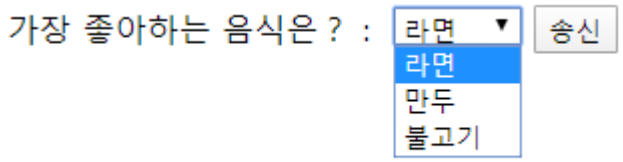
BOM – Form 객체

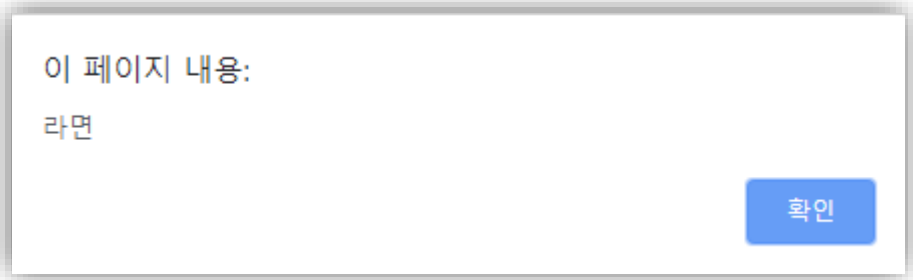
- 선택박스와 리스트박스 다루기
  - <select> 태그의 기본 기능이 선택 박스의 기능(단일 선택)
  - <select> 태그 사용 시 multiple="multiple"을 사용하면 리스트 박스(다중 선택 가능)
  - 선택박스에서 선택 여부 값을 가지는 프로퍼티는 selected이다.

BOM – Form 객체

❖ 예 제

```
  <script>
    function show() {
      let f = document.fm.food;
      window.alert(f.options[f.selectedIndex].value);
      window.alert(document.fm.food.value);
      window.alert(f.options[f.selectedIndex].text);
      return false;
    }
  </script>
</head>

<body>
  <form name="fm" onsubmit="return show()">
    가장 좋아하는 음식은 ? :
    <select name="food">
      <option value="라면">라면</option>
      <option value="만두">만두</option>
      <option value="불고기">불고기</option>
    </select>
    <input type="submit" value="제출" />
  </form>
</body>
```



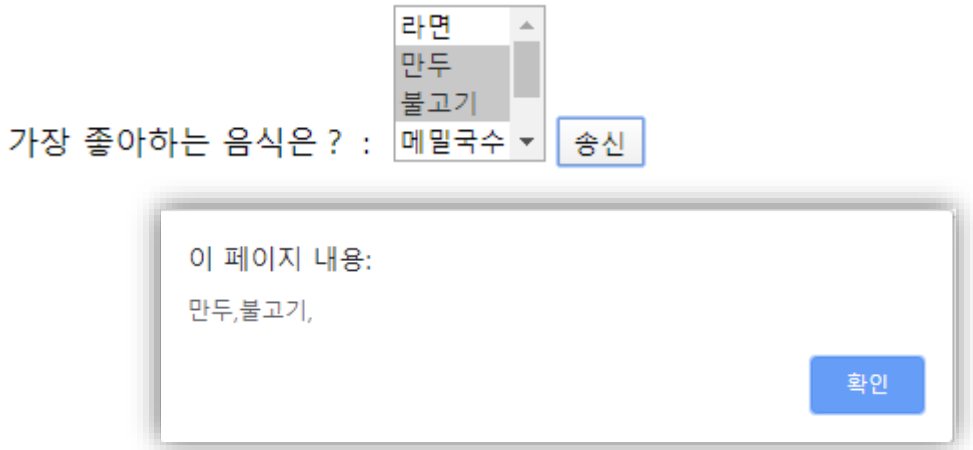


BOM – Form 객체

❖ 예 제

```
<script>
  function getList(elem) {
    let result = "";
    for (let i = 0; i < elem.options.length; i++) {
      if (elem.options[i].selected) {
        result += elem.options[i].value + ",";
      }
    }
    return result;
  }

  function show() {
    window.alert(getList(document.fm.food));
    return false;
  }
</script>
```

```
<body>
  <form name="fm" onsubmit="return show()">
    가장 좋아하는 음식은 ? :
    <select name="food" multiple="multiple" size="4">
      <option value="라면">라면</option>
      <option value="만두">만두</option>
      <option value="불고기">불고기</option>
      <option value="메밀국수">메밀국수</option>
      <option value="피자">피자</option>
      <option value="우동">우동</option>
    </select>
    <input type="submit" value="제출" />
  </form>
</body>
```



BOM – Form 객체

- <form>태그의 내부 속성 변경 가능
  - 다음과 같은 프로퍼티와 메서드를 사용한다.(*멤버는 읽기 전용)

멤버	설명
*name	폼 이름
method	전송 방법(GET/POST)
enctype	인코딩 방법
scrollIntoView(flag)	폼이 표시되도록 콘텐츠를 스크롤한다. (true: 윈도우 상단에 맞춤, false: 윈도우 하단에 맞춤)
submit()	폼의 내용을 submit(전송) 한다.
reset()	폼의 내용을 초기화한다.

BOM – Location 객체

- 페이지 이동이나 현재 페이지 리로드하고자 할 경우 사용할 수 있는 객체
  - 주요 프로퍼티와 메서드(*은 읽기 전용)
    - 프로퍼티의 샘플 값은 다음과 같은 URL을 요청한 경우의 값이다.
    - https://www.google.co.kr:8080/search#test?id=hello

멤버	설명	샘플 값
hash	앵커명(#~)	#test?id=hello
host	호스트(호스트명 + 포트번호. 80번일 경우 생략)	www.google.co.kr:8080
hostName	호스트명	www.google.co.kr
href	링크 장소(요청 경로)	https://www.google.co.kr:8080/search#test?id=hello
*pathname	경로 이름	/search#test?id=hello
*port	포트 번호	8080
*protocol	프로토콜 이름	http:
search	쿼리 정보	?id=hello
reload()	현재 페이지를 새로고침	-
replace(url)	지정한 페이지로 이동	-

BOM – Location 객체

❖ 예제 – 현재 페이지 이동

```
  <script>
    function jump(elem) {
      let isbn = elem.value;
      //location.href = 'http://www.kyobobook.co.kr/product/detailViewKor.laf?barcode=' + isbn;
      location.replace("http://www.kyobobook.co.kr/product/detailViewKor.laf?barcode=" + isbn);
    }
  </script>
</head>

<body>
  <form name="fm">
    <select name="book" onchange="jump(this)">
      <option value="">---도서명 선택---</option>
      <option value="9791163031291(1163031291)">Do it! HTML5+CSS3 웹 표준의 정석</option>
      <option value="9788956748245(8956748241)">초보자를 위한 Javascript 200제</option>
      <option value="9788960773431(8960773433)">토비의 스프링 3.1 세트</option>
      <option value="9788979146639(8979146639)">HEAD FIRST SERVLETS & JSP(개정판)</option>
    </select>
  </form>
</body>
```

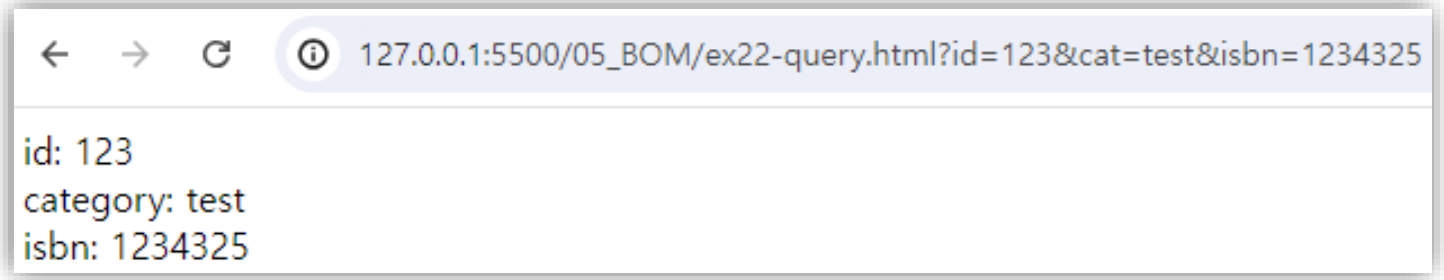
페이지 이동 시 히스토리를 남기고 싶지 않다면 replace메서드를 사용

BOM – Location 객체

❖ 예제 – 쿼리 정보 가져오기

```
  <script>
    function $q() {
      let result = {};
      let str = location.search.substring(1);
      let params = str.split("&");
      for (let i = 0; i < params.length; i++) {
        let kv = params[i].split("=");
        result[kv[0]] = decodeURIComponent(kv[1]);
      }
      return result;
    }
  </script>
</head>

<body>
  <script>
    var q = $q();
    document.write("id: " + q["id"] + "<br>");
    document.write("category: " + q["cat"] + "<br>");
    document.write("isbn: " + q["isbn"] + "<br>");
  </script>
</body>
```

- 쿼리 정보 테스트는 서버에 올라가 있는 문서를 요청해야 바르게 동작
- 파일시스템에 저장된 파일은 요청이나 아니라 그냥 내용을 열어만 보는 것



BOM – Document 객체

- 윈도우 내에 표시된 문서를 조작하는 것은 Document객체의 역할
  - Document객체의 대부분의 내용은 DOM(Document Object Model)에 대응(DOM을 사용)
  - Document객체 내용 중 DOM에 대응되지 않거나 어려운 기능만 확인해본다.

- 쿠키 저장 및 확인

- 스크립트에서 문자열 출력

BOM – Document 객체

- 쿠키 저장/확인 가능
  - 쿠키는 클라이언트에 저장되는 데이터.
  - 스크립트로부터 필요한 정보를 유지하기 위해 사용
  - 아이디 기억하기, 비회원 장바구니 등

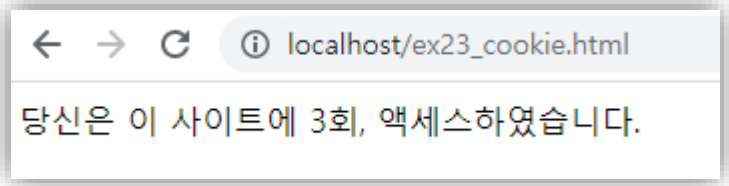
파라미터	설명
쿠키명	쿠키를 식별하기 위한 이름. 영어, 숫자 이외에는 encodeURIComponent함수로 인코딩 필요
expires	쿠키의 유효시간(그리니치 표준시). 시간이 지난 쿠키는 브라우저를 닫는 시점에 삭제
domain	쿠키가 유효한 도메인. 지정된 도메인에 요청 할 때에만 해당 쿠키를 전송함.
path	쿠키가 유효한 경로. 지정된 경로 하위에 대해서만 요청에 쿠키를 전송
secure	SSL통신에서만 쿠키를 전송할 것인지 지정

BOM – Document 객체

❖ 예제

```
<script>
  function setCookie(name, value, expires, domain, path, secure) {
    let c = '';
    c += name + '=' + encodeURIComponent(value);
    if (expires) {
      let exp = new Date();
      exp.setDate(exp.getDate() + expires);
      c += '; expires=' + exp.toGMTString();
    }
    if (domain) { c += '; domain=' + domain; }
    if (path) { c += '; path=' + path; }
    if (secure) { c += '; secure'; }
    document.cookie = c;
}

function getCookie(name) {
  let cs = document.cookie.split(';');
    for (let i = 0; i < cs.length; i++) {
      let kv = cs[i].split('=');
      if (kv[0] == name) { return decodeURIComponent(kv[1]); }
    }
    return null;
}
</script>
```

```
<body>
  <script>
    let cnt = getCookie('js_cnt');
    cnt = cnt ? ++cnt : 1;
    setCookie('js_cnt', cnt, 90);
    document.write('당신은 이 사이트에 ' + cnt + '회, 액세스하였습니다.');
  </script>
</body>
```



BOM – Document 객체

- 스크립트에서 문자열 출력하기
  - document.writeln(), write() 메서드를 이벤트 핸들러 내에서 사용하는 경우 주의

- 예제 – 정상–

```
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
</head>

<body>
    <script>
        document.writeln('안녕하세요, JavaScript!');
    </script>
</body>
```

BOM – Document 객체

❖ 예제 – 문제 있음 –

```
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
    <script>
        window.onload = function () {
            document.writeln('안녕하세요, JavaScript!');
        }
    </script>
</head>

<body>
    안녕하세요~
</body>
```

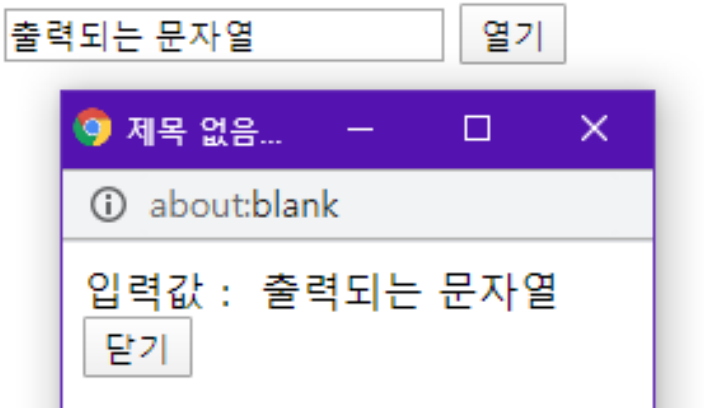
중요)

- 페이지 출력(load)가 완료된 후에 writeln을 호출하면 새로운 문서를 open
- 즉 이벤트 핸들러 내에서는 writeln을 사용하지 말 것!

BOM – Document 객체

- document.writeln()의 용도
  - 디버그 용도의 변수 출력
  - 자식 윈도우에 대해 컨텐츠 출력

```
    <script>
        function winOpen() {
            const subwin = window.open('', 'Child', 'width=200, height=200');
            subwin.document.open('text/html');
            subwin.document.writeln('입력값 : ');
            subwin.document.writeln(subwin.opener.document.fm.data.value);
            subwin.document.writeln('<br /><input type="button" value="닫기" ');
            subwin.document.writeln('onclick="self.close();" />');
            subwin.document.close();
        }
    </script>
</head>

<body>
    <form name="fm">
        <input type="text" name="data" value="출력되는 문자열" />
        <input type="button" value="열기" onclick="winOpen()" />
    </form>
</body>
```



BOM – History 객체

- 페이지 이동 기록 활용

```
<head>
    <meta charset="utf-8" />
    <title>Hello JS</title>
</head>

<body>
    <a href="JavaScript:history.back()">되돌아가기</a>|
    <a href="JavaScript:history.forward()">앞으로가기</a>
</body>
```

history.go(n) 메서드도 있음
history에서 페이지 수 만큼 앞으로 이동 (음수는 뒤로)

BOM – Navigator 객체

- 브라우저의 종류, 버전 등 기능의 대응 상황을 알고 싶은 경우 이용

- 주요 멤버

멤버	설명
appCodeName	브라우저의 코드명
appName	브라우저명
appVersion	브라우저 버전
platform	플랫폼명
userAgent	사용자 에이전트명
javaEnabled()	Java가 이용 가능한지 판단

```
<body>
  <script>
    document.writeln(navigator.appCodeName);
    document.writeln(navigator.appName);
    document.writeln(navigator.appVersion);
    document.writeln(navigator.platform);
    document.writeln(navigator.userAgent);
    document.writeln(navigator.javaEnabled());
  </script>
</body>
```

BOM – Navigator 객체

❖ 예제 결과

Chrome

```
Mozilla
Netscape
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36
Win32
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36
false
```

Naver Whale

```
Mozilla
Netscape
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.146 Whale/2.6.90.16 Safari/537.36
Win32
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.146 Whale/2.6.90.16 Safari/537.36
false
```

MS Edge

```
Mozilla
Netscape
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.18362
Win32
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.18362
true
```

BOM – Navigator 객체

❖ Navigator 활용

- 예전에는 navigation 객체를 이용해서 브라우저의 종류나 버전에 따라 분기 처리를 했음
- 새로운 브라우저나 버전이 업데이트 될 때마다 분기를 또 추가해야 하는 불편함 존재

- 현재는 기능 테스트 방법을 활용한다.
  - ➢ 기능테스트란 미리 호출해보고 가능한지 판단하여 동작을 결정하는 것.

```
if(elem.attachEvent){
  //attatch메서드를 이용할 수 있다면 동작할 코드
} else if(elem.addEventListener) {
  //addEventListener 메서드를 이용할 수 있는 경우의 코드
} else {
  //어떤 것도 이용할 수 없는 경우의 코드
}
```

- 일반적으로 navigator객체는 특정 브라우저나 버전에 있는 버그에 대한 대응 코드를 기술
- appCode나 appCodeName으로는 브라우저 종류를 특정하기 어려우므로 userAgent값 사용

BOM – Screen 객체

- 모니터 화면의 사이즈나 색상수(비트) 등의 정보를 사용하기 위한 객체
  - 모니터 사이즈에 따라 동적인 표시 가능

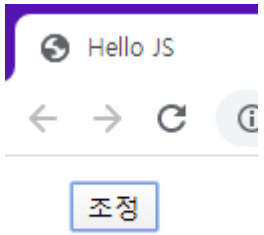
- 주요 프로퍼티

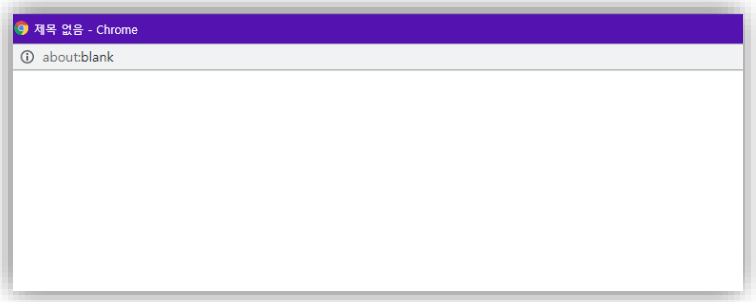
프로퍼티	설명	반환 값(샘플)
height	모니터의 높이	768
width	모니터의 폭	1024
availHeight	사용 가능한 높이(태스크 바 등을 제외한 사이즈)	738
availWidth	사용 가능한 폭(태스크 바 등을 제외한 사이즈)	1024
colorDepth	모니터의 색상 수(비트 수)	32

BOM – Screen 객체

- 예제 – (브라우저 마다 동작 허용 조건이 다르므로 확인 필요)

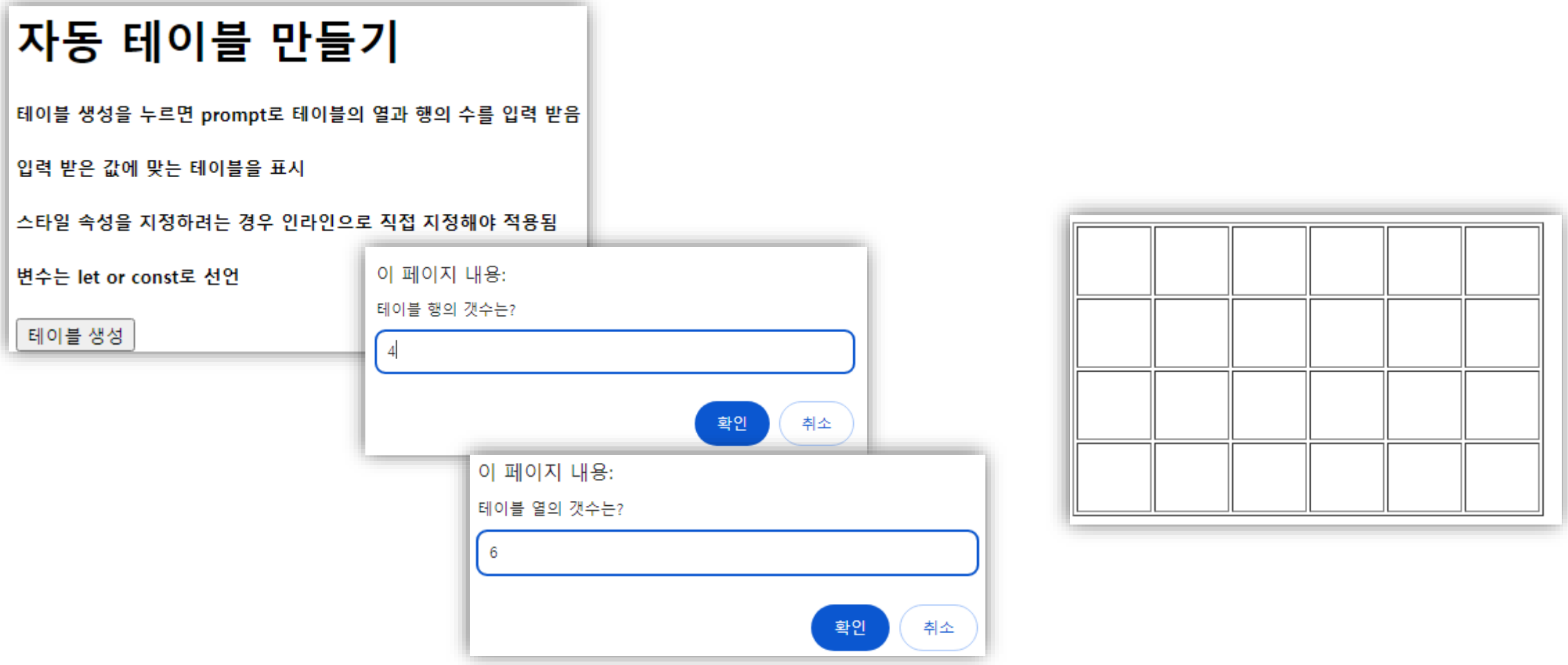
```
    <script>
      function adjust() {
        let subwin = window.open("", "Child", "width=200, height=200");
        subwin.moveTo(0, 0);
        subwin.resizeTo(screen.width, screen.height);
      }
    </script>
  </head>

  <body>
<pre>
  <input type="button" value="조정" onclick="adjust()" />
</pre>
  </body>
```





Quiz

❖ 다음과 같이 동작하도록 HTML과 Javascript를 작성하세요.



Quiz

❖ 다음과 같이 동작하도록 HTML과 Javascript를 작성하세요.

Quiz

- 다음과 같이 동작하도록 HTML과 Javascript를 작성하세요.