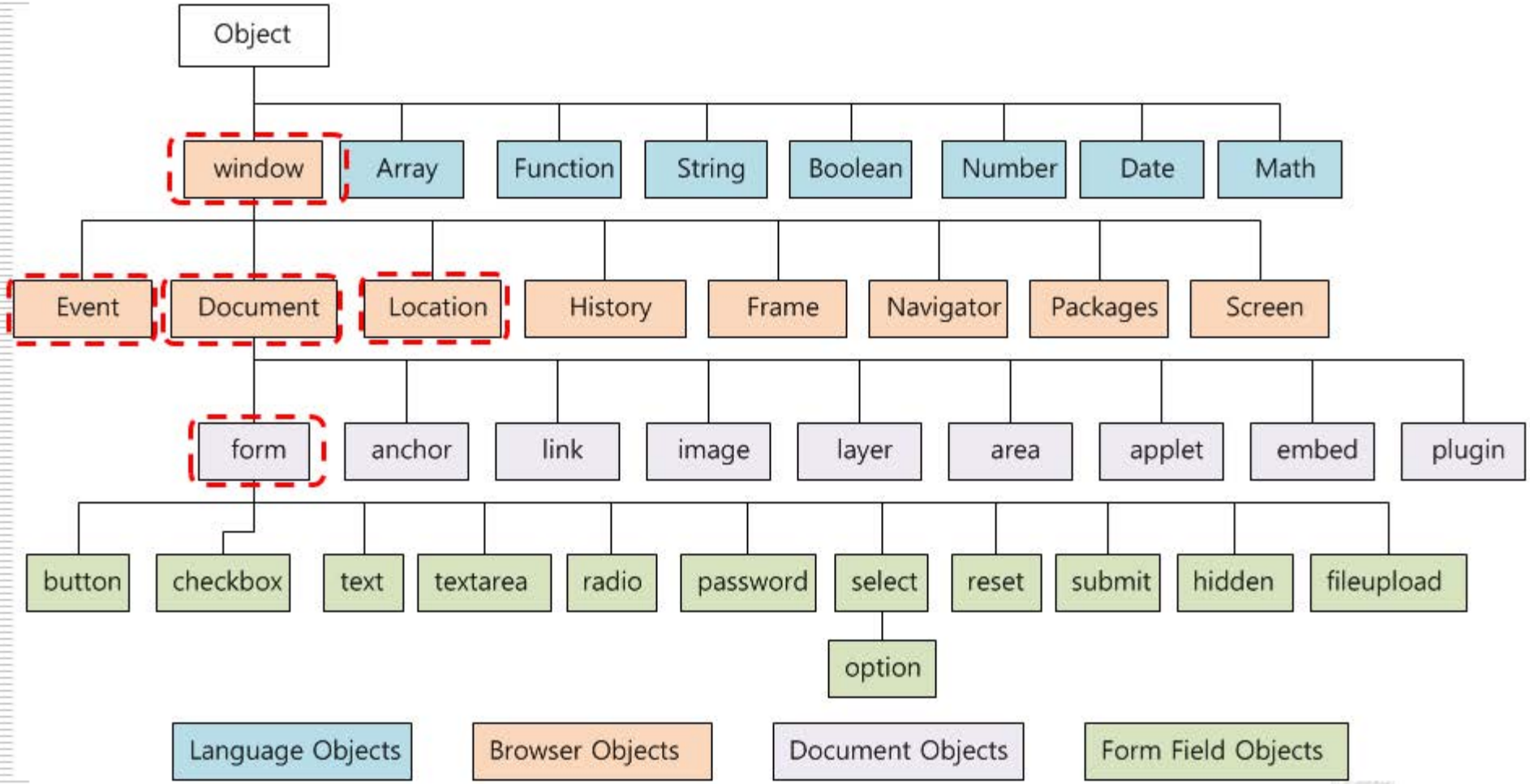
브라우저 객체

학습목표

- ✓ 브라우저 객체
- ✓ 이벤트란?
  - 이벤트 모델
  - 이벤트 종류
- ✓ 페이지 로딩 순서
- ✓ 이벤트 핸들러 설정
  - 인라인 모델 방식
  - 고전 모델 방식
  - 표준 모델 방식
- ✓ 이벤트 핸들러 제거
- ✓ 이벤트 객체
  - 이벤트 전달 - 버블링
  - 이벤트 취소 : return false;
- ✓ window 객체
  - alert()
  - confirm() / prompt() / open()
  - setInterval() / setTimeout()
- ✓ form 객체
  - form 요소
    - action / method / enctype
  - form validation
- ✓ location 객체
  - location.href
  - location.search
- ✓ document 객체
  - 쿠키

브라우저 객체(BOM==DOM)

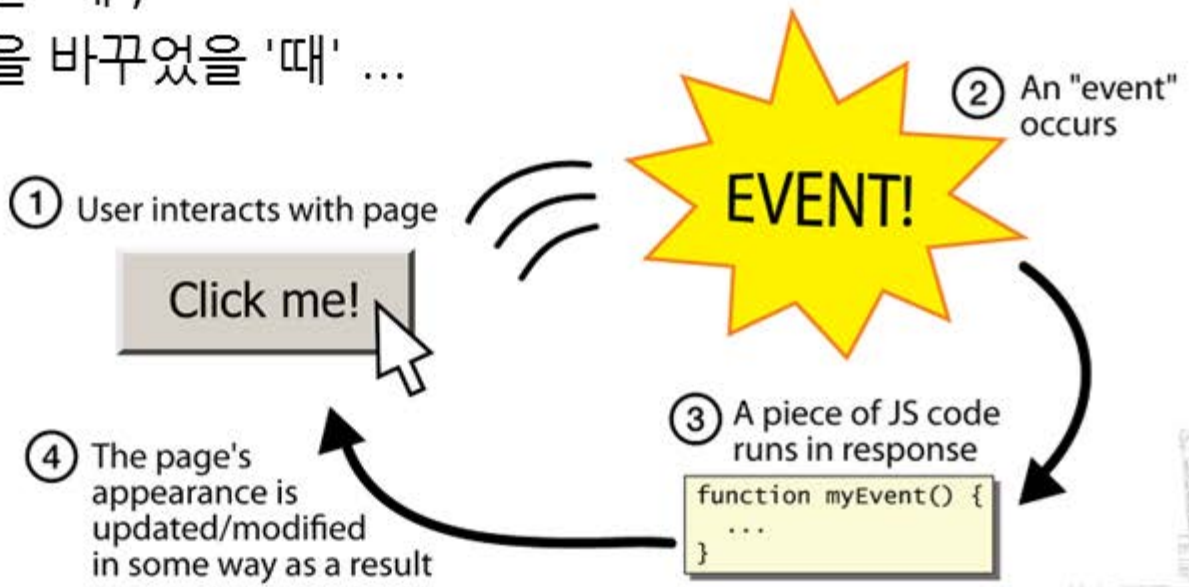


BOM : Browser Object Model
DOM : Document Obeect Model

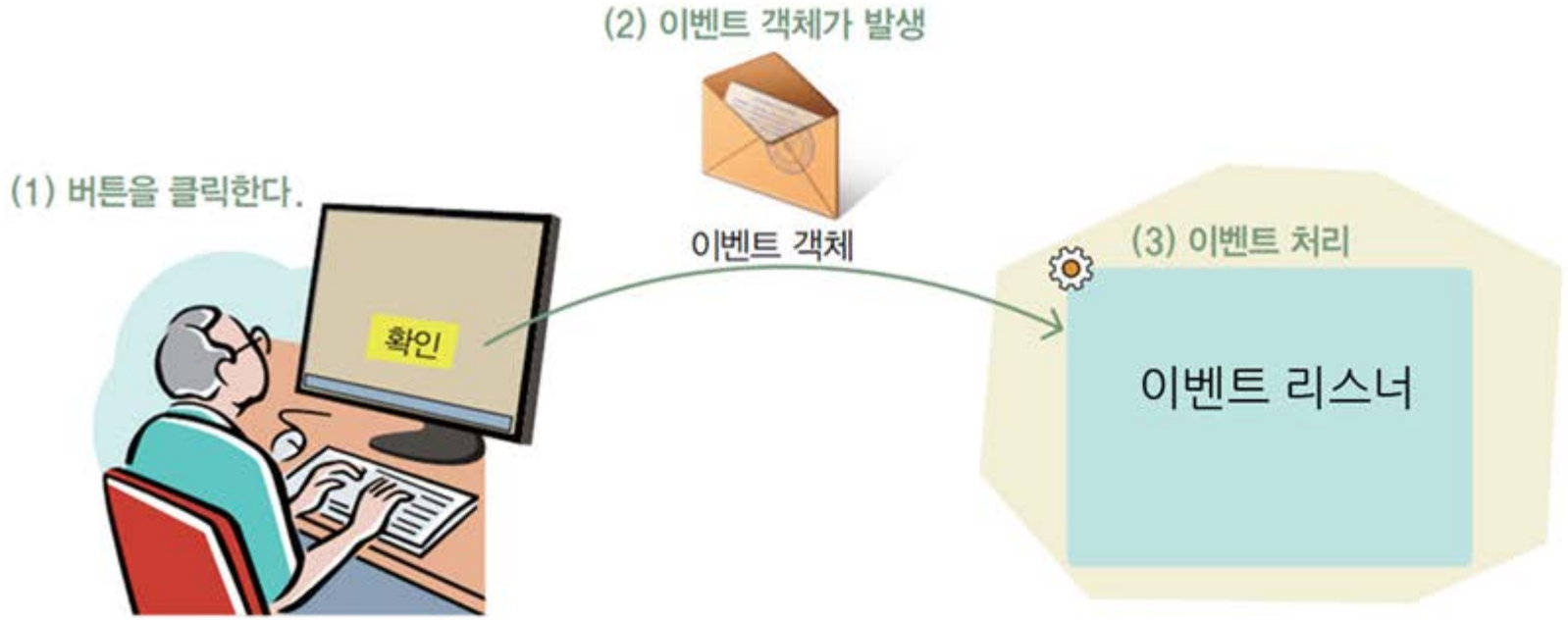
이벤트란?

- 이벤트
  - 이벤트(event)는 어떤 사건을 의미합니다.
  - 사용자나 프로그램에 의해서 발생된 사건.

  - 이벤트(사건) 예시
    - 마우스가 문자열 위로 올때, 나갈때,
    - 사용자가 클릭을 했을 '때',
    - 스크롤을 했을 '때',
    - 필드의 내용을 바꾸었을 '때' …

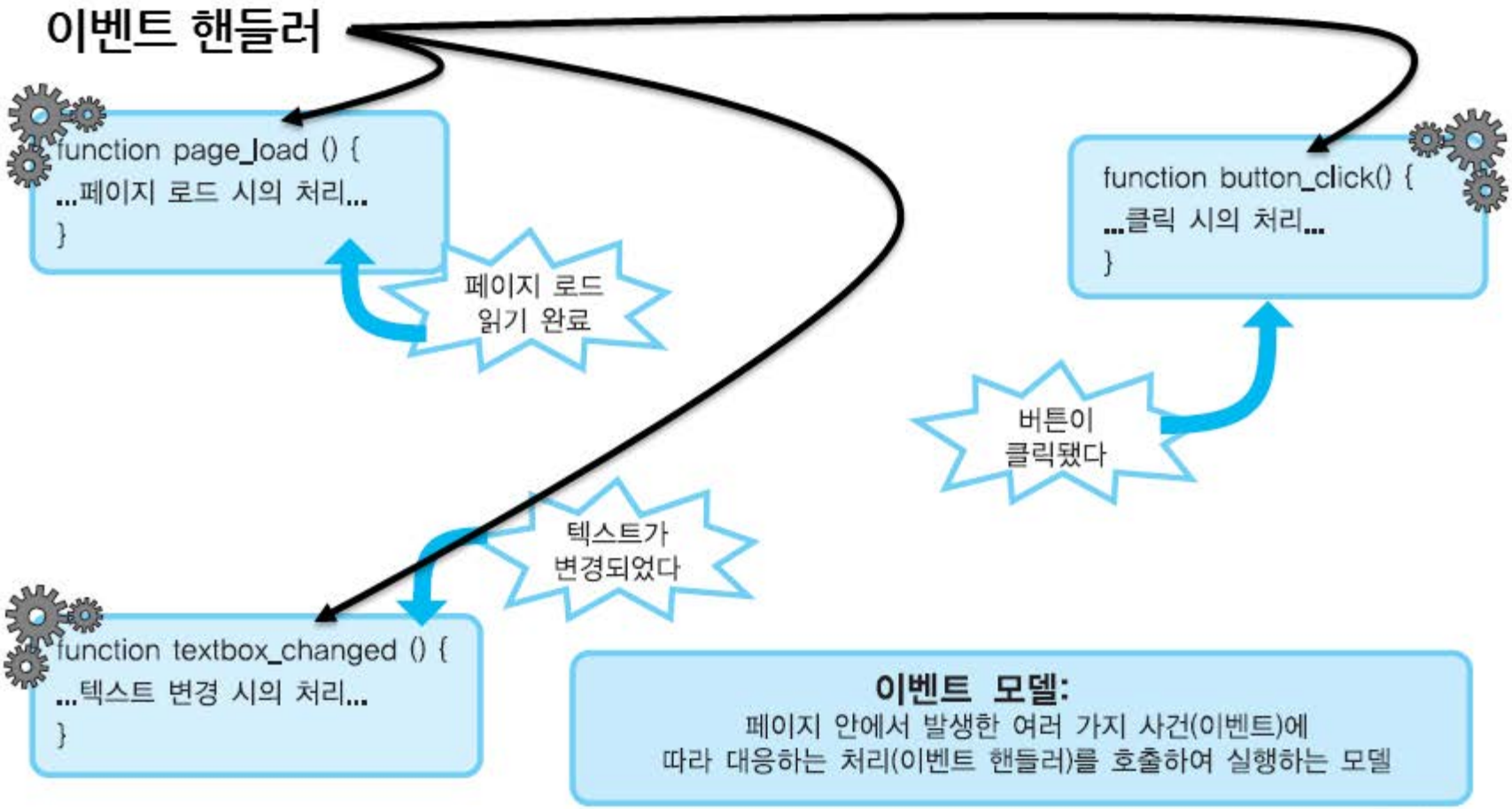


이벤트 처리 과정



- 이벤트 처러시 고려 사항
  - 어느 요소에서 발생했고
  - 어떤 이벤트 객체를
  - 누가 처리할 것인가?

이벤트 모델



이벤트 모델:
페이지 안에서 발생한 여러 가지 사건(이벤트)에
따라 대응하는 처리(이벤트 핸들러)를 호출하여 실행하는 모델

이벤트 종류

- 자바스크립트 지원하는 이벤트
  - **마우스 이벤트**
  - 키보드 이벤트
  - 유저 인터페이스 이벤트
  - **폼(입력 양식) 이벤트**
  - HTML 프레임 이벤트
  - 구조 변화 이벤트
  - 터치 이벤트

분류	이벤트명	발생 타이밍
읽기	abort	이미지의 로딩이 중단되었을 때
	load	페이지, 이미지의 로딩 완료 시
	unload	다른 페이지로 이동할 때
마우스	click	마우스 클릭 시
	dblclick	더블클릭 시
	mousedown	마우스 버튼을 눌렀을 때
	mousemove	마우스 포인터가 이동했을 때
	mouseout	요소에서 마우스 포인터가 떨어졌을 때
	mouseover	요소에 마우스 포인터가 겹쳤을 때
	mouseup	마우스 버튼을 떼어 놓았을 때
	contextmenu	context menu가 표시되기 전
키	keydown	키를 눌렀을 때
	keypress	키를 누른 상태
	keyup	키를 떼어 놓았을 때
폼	change	내용이 변경되었을 때
	reset	리셋 버튼이 눌렸을 때
	submit	서브밋 버튼이 눌렸을 때
포커스	blur	요소로부터 포커스가 벗어났을 때
	focus	요소가 포커스되었을 때

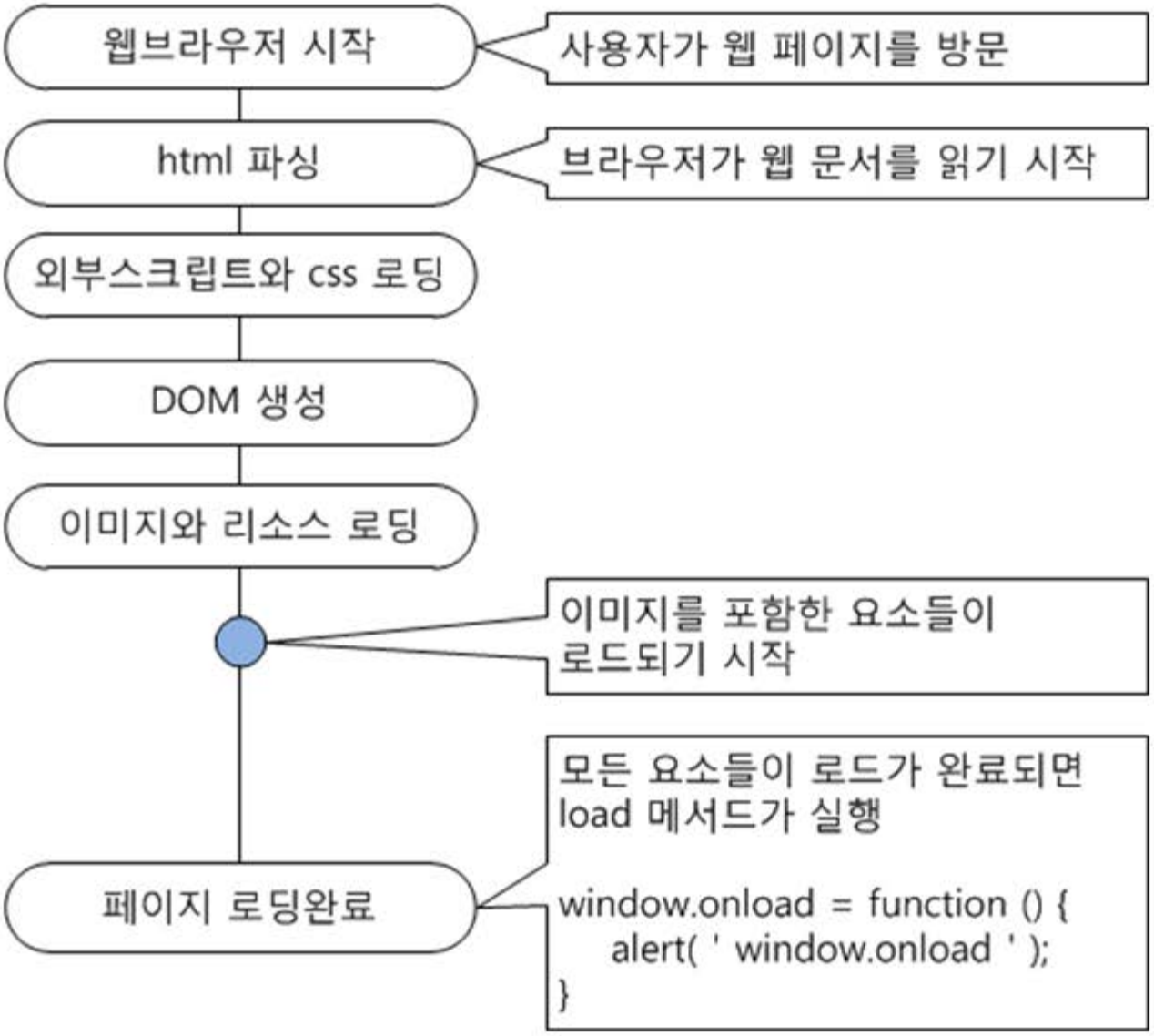
이벤트 종류

분류	이벤트명	이벤트핸들러	발생 타이밍
윈도우	abort	onabort	이미지의 로딩이 중단되었을 때
	load	onload	페이지, 이미지의 로딩 완료 시
	unload	onunload	다른 페이지로 이동 할 때
	move	onmove	윈도우나 프레임 사이즈를 움직일 때
	resize	onresize	윈도우나 프레임 사이즈를 움직일 때
마우스	click	onclick	마우스 클릭시
	dbclick	ondbclick	더블클릭시
	mousedown	onmousedown	마우스를 누를 경우
	mousemove	onmousemove	마우스를 움직일 경우
	mouseup	onmouseup	마우스를 눌렀다 놓은 경우
	mouseout	onmouseout	마우스 포인터가 떨어졌을 때
	mouseover	onmouseover	마우스가 올라온 경우
	mouseenter	onmouseenter	마우스가 영역 안으로 들어왔을 때
	mouseleave	onmouseleave	마우스가 영역 밖으로 나갈 때
	dragdrop	ondragdrop	마우스 누르고 움직일 경우
	contextmenu		context menu가 표시되기 전

이벤트 종류

분류	이벤트명	이벤트핸들러	발생 타이밍
키	keydown	onkeydown	키 입력시
	keypress	onkeypress	키 누를 때
	keyup	onkeyup	키를 누르고 놓았을 때
폼	change	onchange	요소가 변경될대 실행
	~~select~~	~~onselect~~	~~입력양식의 하나가 선택될 때~~
	submit	onsubmit	폼을 전송하는 경우
	reset	onreset	리셋 버튼이 눌렸을 때
포커스	blur	onblur	요소가 변경될대 실행
	focus	onfocus	포커스가 위치할 경우
	focusin	onfocusin	
	focusout	onfocusout	
	scroll	onscroll	
	error	onerror	문서나 이미지에서 에러를 만났을 때

페이지 로딩 순서


웹브라우저 시작
사용자가 웹 페이지를 방문
html 파싱
브라우저가 웹 문서를 읽기 시작
외부스크립트와 css 로딩
DOM 생성
이미지와 리소스 로딩
이미지를 포함한 요소들이
로드되기 시작
모든 요소들이 로드가 완료되면
load 메서드가 실행
window.onload = function () {
alert(' window.onload ');
}
페이지 로딩완료

```
<script>
    window.onload = function (  ) {
        alert( '페이지 onload 실행');
        console.log('페이지 onload 실행');
    };
    window.onunload = function (  ) {
        alert( '페이지 onunload 실행');
        console.log('페이지 onunload 실행');
    };
</script>


<body>
    <script> console.log('body 안에 p 앞에'); </script>
    <p> body 안에...</p>
    <script> console.log('/body 직전, p 뒤에'); </script>
    <a href="http://www.naver.com">naver</a>
</body>
</html>
```

Blocked alert('페이지 onunload 실행') during unload.

이벤트와 이벤트 핸들러

- window.onload = function (event) { };
  - load
    - 이벤트 이름, 이벤트 타입이라고 함
  - onload
    - on : ~ 할 때
    - 이벤트 핸들러(리스너).
    - 메서드
  - function(event) { }
    - 이벤트 속성에 넣는 함수
    - 이벤트 핸들러라고 함

이벤트 핸들러 설정

- 인라인 이벤트 모델
  - HTML 태그 내의 속성(attribute)에 핸들러 선언하는 방법
  - 사용하지 말자.

- 고전 이벤트 모델
  - JavaScript의 코드 내에서 핸들러 선언하는 방법
  - 이벤트 하나에 핸들러 한개만 연결 가능(1 : 1)

- 표준 이벤트 모델
  - JavaScript의 코드 내에서 핸들러 선언하는 방법
  - 한 번에 여러 가지 이벤트 핸들러 추가 가능(1 : N)

이벤트 핸들러 설정-인라인 모델

- 태그 내의 속성(attribute)으로 이벤트 핸들러 선언하는 방법
- 웹 페이지의 가장 기본적인 연결 방식
  - HTML 태그 내부에 자바스크립트 코드를 넣어 이벤트를 연결
- 잘 사용되지 않으나, 국내에서는 아직 많이 사용됨. 사용하지 말자.

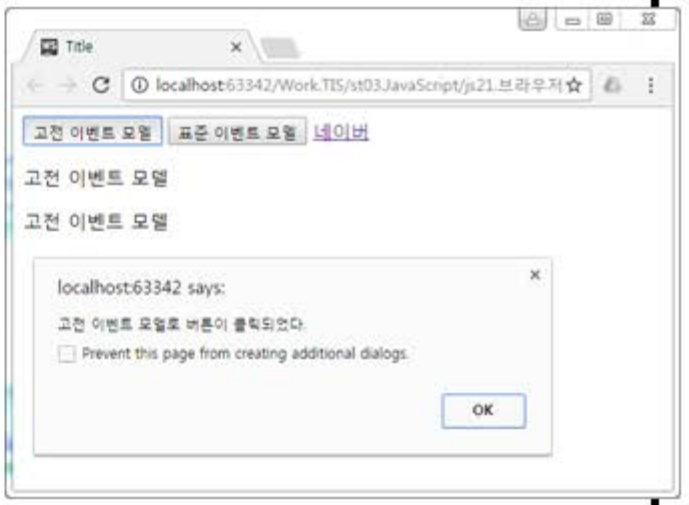
```
<script type="text/javascript">
    var btn_onclick = function (event){
        window.alert('버튼이 클릭되었다');
    }
</script>

…중략…
<input type="button" value="버튼"  onclick="btn_onclick();" />
```

이벤트 핸들러 설정-고전 모델

- JavaScript의 코드 내에서 선언하기
- 이벤트명은 모두 소문자로 기술 - onLoad, onClick은 불가
- 이벤트 하나에 이벤트 핸들러 하나만 연결

```
<script>
    // 페이지 로드 시에 이벤트 핸들러를 등록
    window.onload = function(event) {
        // 1. 태그 선언
        var btn = null;
        // 2. 태그 찾기 : getElementById, getElementByClassName
        btn =document.getElementById( 'btn' );
        // 3. 이벤트 핸들러 설정
        btn.onclick = function( event ){
            window.alert( '버튼이 클릭되었다.' );
        };
    };
</script>
... 중략 ...
<input  id="btn"  type="button"  value="버튼" />
```

```
<script>
    window.onload = function (event) {
        var btn = document.getElementById('btn');
        btn.onclick = function (event) {
            window.alert( '버튼이 클릭되었다.' );
        }
        var btn2 = document.getElementsByClassName('btn2');
        btn2[1].onclick = function (event) {
            window.alert( 'aaaa 가 클릭되었다.' );
            this.onclick = null;  // delete onclick;
            return false;
        }
        var p = document.getElementsByName('name');
        p[0].onclick = function (event) {
            window.alert( 'name 이 클릭되었다.' );
        }
    }
</script>
<body>
    <input id="btn" type="button" value='고전 이벤트 모델'>
    <input class='btn2' type="button" value='표준 이벤트 모델'>
    <a     class='btn2' href="http://naver.com">네이버</a>
    <p name="name">고전 이벤트 모델</p>
    <p name="name">고전 이벤트 모델</p>
</body>
```



이벤트 핸들러 설정-고전 모델

- 이벤트 핸들러 연결 : 객체의 이벤트 속성을 사용해

```
window.onload = function(event) {
    var header = document.getElementById( 'btn' );
    header.onclick = function (e) {
        alert('H1 이 클릭되었어요!!!');
    };
};
```

- 이벤트 핸들러 제거 : 이벤트 속성에 null 할당

```
window.onload = function(event) {
    var header = document.getElementById( 'btn' );
    header.onclick = function (e) {
        alert('H1 이 클릭되었어요!!!');
        this.onclick = null; // 이벤트 핸들러 제거
    }
};
```

- 아래의 코드는 0번째 p 태그에만 이벤트 핸들러를 지정하였다. 이것을 for문을 이용하여 모든 p 태그에 click 이벤트 핸들러를 작성하시오.

```
<script>
    window.onload = function (event) {
        var p = document.getElementsByName('name');

        p[0].onclick = function (event) {
            window.alert( 'name 이 클릭되었다.' );
        }

    }
</script>

<body>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
    <p name='name'>고전 이벤트 모델</p>
</body>
```

이벤트 핸들러 설정-표준 모델

- 표준 이벤트 모델
  - 한 번에 여러 가지 이벤트 핸들러 추가 가능
  - addEventListener(eventName, handler);
  - removeEventListener(eventName, handler);

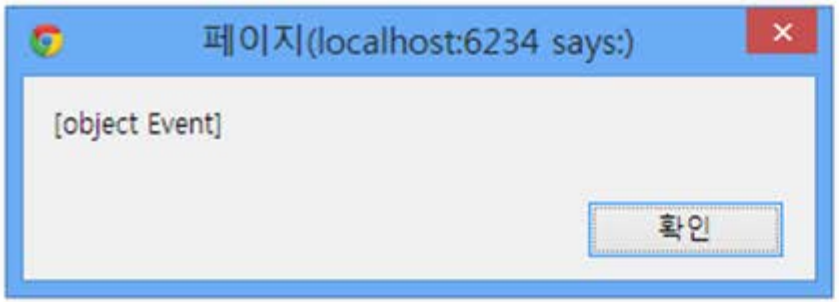
```
window.addEventListener('load', winLoad);
function winLoad(e) {
    var div = document.getElementById('outerBox');
    var anc = document.getElementById('anchor');
    var img = document.getElementById('img');

    div.addEventListener('click', outerBox  );
    img.addEventListener('click', innerImg  );
    img.addEventListener('click', stopEvent );
}
```

이벤트 객체

- 이벤트와 관련된 정보를 알려줌

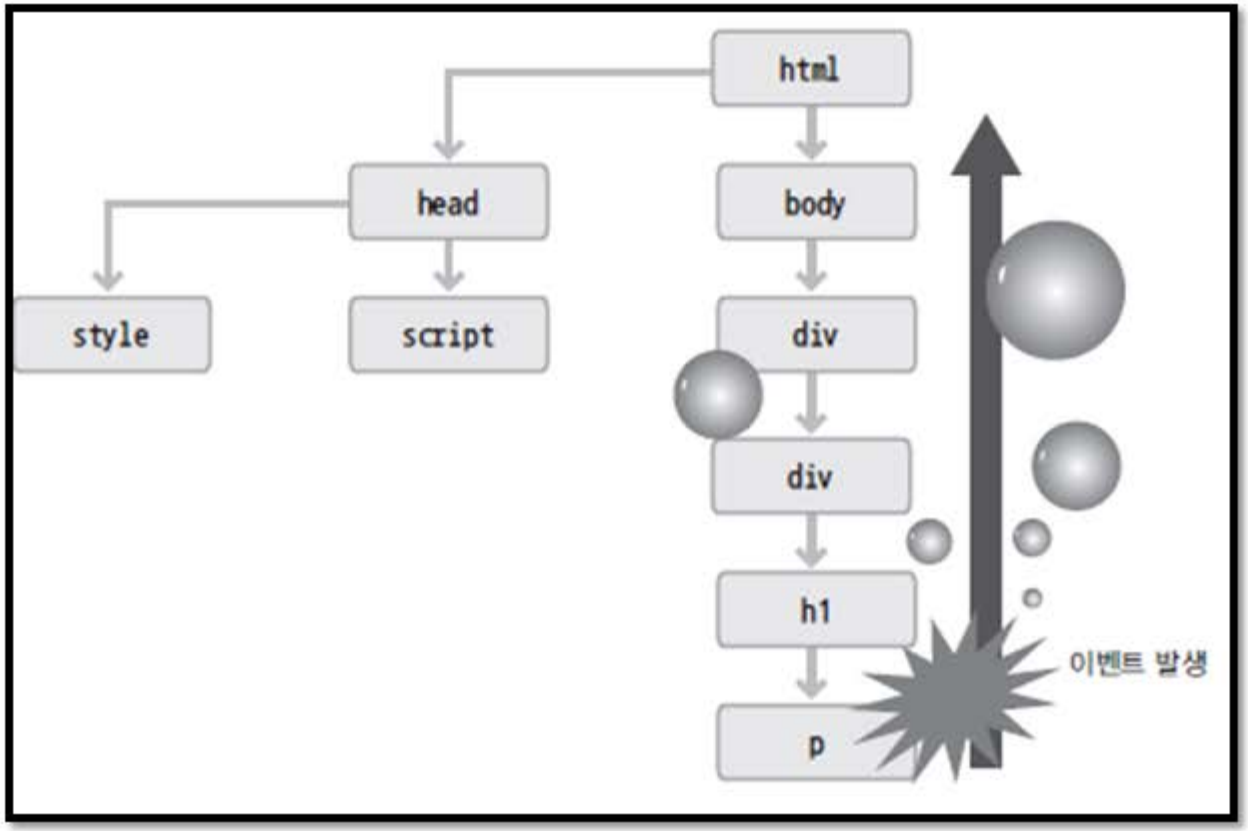
```
<!DOCTYPE html>
<html>
<head>
    <title>Event Object</title>
    <script>
        window.onload = function (event) {
            alert(event);
```

이벤트 객체



이벤트 전달 - 버블링

- 이벤트 전달
  - 어떤 순서로 발생하는가?
  - 이벤트 버블링 방식이 일반적
  - 자식 노드에서 부모 노드 순으로 이벤트 실행



이벤트 취소

- 이벤트 취소란?
  - 발생된 이벤트를 제거하고자 하는 경우
  - ex) 전송 전 유효성 검사에서 재입력이 필요한 경우에 사용됨

- 이벤트 취소 방법
  - 이벤트 핸들러 안에서 **return false;** 사용
  - 이벤트 객체의 stopPropagation() 메소드 사용
  - 이벤트 객체의 preventDefault() 메소드 사용

```
function innerImg( event ) {
    alert("image click!!");
    event.stopPropagation();
}
```

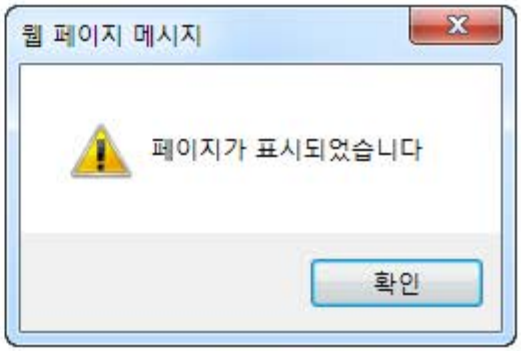
```
function innerImg( event ) {
    alert("image click!!");
    event.preventDefault();
}
```

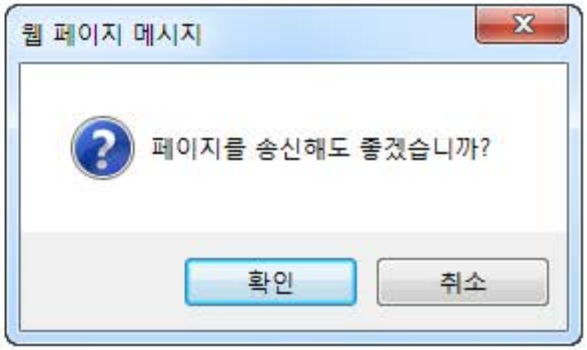
이벤트 취소

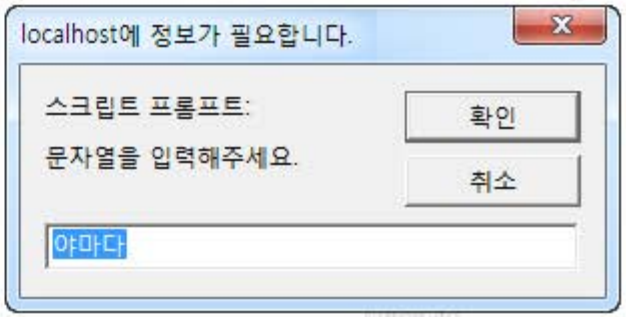
- 이벤트 핸들러의 반환값에 false를 입력
- [버튼] 링크를 클릭해도 연결된 웹 페이지로 자동으로 이동하지 않음

Window 객체

- alert 메소드(경고 다이얼로그)
  - 지정된 메시지를 출력하는 다이얼로그
- confirm 메소드(확인 다이얼로그)
  - 사용자에게 확인을 요구하는 다이얼로그
  - [확인] 버튼 → true, [취소] 버튼 → false
- prompt 메소드(입력 다이얼로그)
  - 사용자에 대해서 입력 다이얼로그를 표시
- open 메서드(새창 띄우기)
  - 새 윈도우를 생성하는 경우에 사용
  - 변수 = window.open(URL, 윈도우명, 옵션);

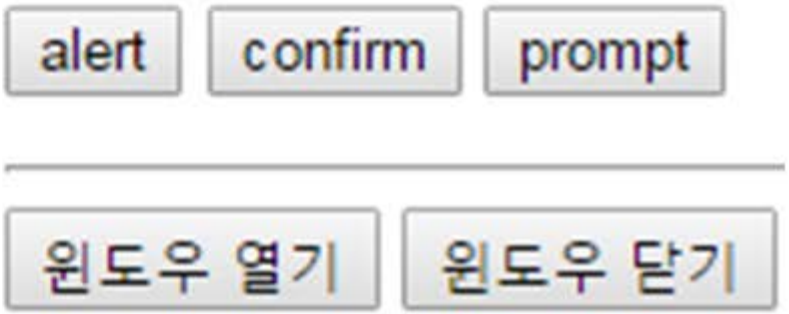






- 아래의 버튼에 click 이벤트 핸들러를 작성하시오.
  단, 고전 이벤트 모델을 이용.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    </script><script>
</head>
<body>
        <input id="alert"   type="button" value="alert"/>
        <input id="confirm" type="button" value="confirm"/>
        <input id="prompt"  type="button" value="prompt"/>
        <hr />
        <input id="win_open"  type="button" value="열기" />
        <input id="win_close" type="button" value="닫기" disabled="disabled" />
</body>
</html>
```

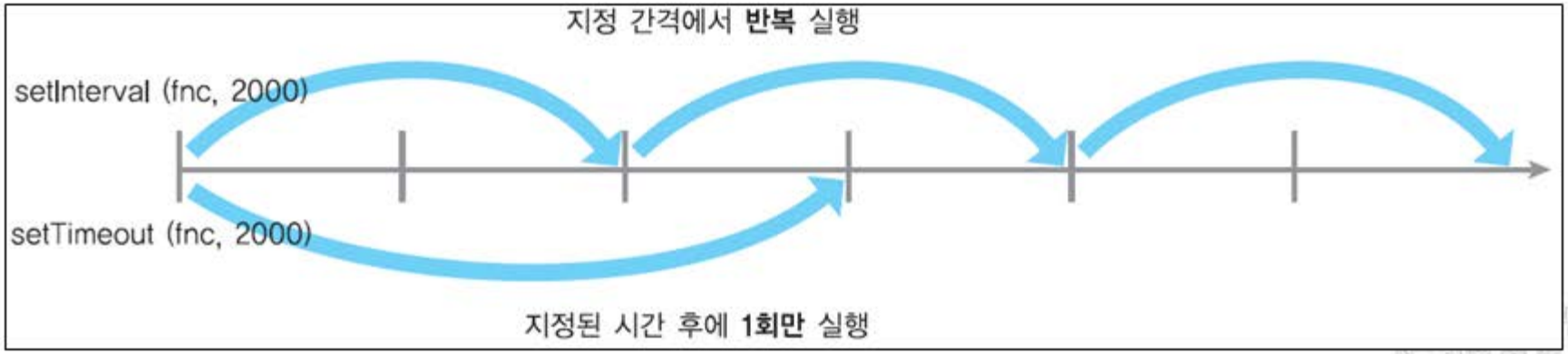


```
<script>
    var MYAPP = {};

    window.onload = function () {
        document.getElementById('alert').onclick = function (event) {
            window.alert('페이지가 표시되었습니다');
        };
        document.getElementById('confirm').onclick = function (event) {
            var input = window.confirm('페이지를 송신해도 좋겠습니까?');
            window.alert('입력값 : ' + input);
        };
        document.getElementById('prompt').onclick = function (event) {
            var input = window.prompt('문자열을 입력해주세요.', '야마다' );
            window.alert('입력값 : ' + input);
        };
        document.getElementById('win_open').onclick = function (event) {
            MYAPP.subwin = window.open('http://www.naver.com', '새창열기', 'location=yes');
            MYAPP.win_close.disabled = false;
        };
        document.getElementById('win_close').onclick = function (event) {
            // 변수 subwin이 비어 있지 않고, 서브 윈도우가 닫혀 있지 않을 때에만 클로즈
            if( MYAPP.subwin !== null && MYAPP.subwin.closed == false ) {
                MYAPP.subwin.close();// 새창을 닫는 함수.
                MYAPP.win_close.disabled = true; // 버튼 비활성화
            }
        };
    };
</script>
```

Window 객체 - 타이머 기능

- 「임의의 시간」에는 각각 처리가 실행되는 시간 간격(setInterval), 몇 밀리세컨드 후에 처리가 실행되는지(setTimeout)를 각각 지정한다.

함수 이름	설명
setInterval(함수, 시간)	정해진 시간 간격으로 **반복 처리**.
setTimeout(함수, 시간)	지정된 시간이 경과했을 때 **1회만 처리**.
clearInterval(식별번호)	setInterval() 함수로 설정한 타이머 제거
clearTimeout(식별번호)	setTimeout() 함수로 설정한 타이머 제거

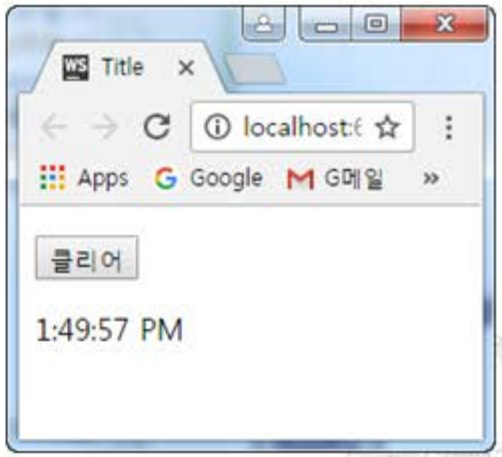


타이머 기능 - setInterval 메서드

```
<script>
    var MYAPP = { timer: null /* 타이머 ID를 대입하기 위한 프로퍼티 */ };

    // 페이지 로드 시에 타이머 처리를 등록
    window.onload = function () {
        MYAPP.timer = window.setInterval(
                // 현재 시각을 <div id="result"> 태그에 표시(1000밀리초마다 갱신)
                function () {
                    var dat = new Date();
                    var result = document.getElementById("result");
                    result.innerHTML = dat.toLocaleTimeString();
                },
                1000
        );
    };
</script>... 중략 ...
<!--버튼 클릭 시에 타이머 처리를 중지 -->
<input id='start' type='button' value='시작' />
<input id='stop'  type='button' value='정지' />
<div id='result'>


</div>
```

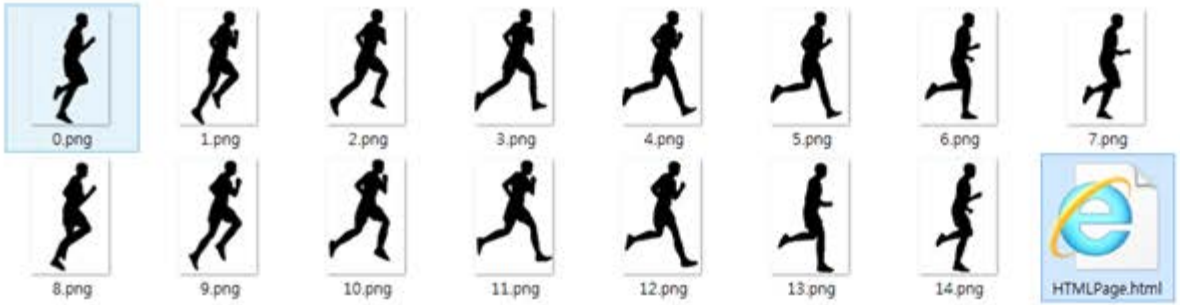
타이머 기능 - setTimeout 메소드

```
<script>
    var MYAPP = { timer: null /* 타이머 ID를 대입하기 위한 프로퍼티 */ };

    // 페이지 로드 시에 타이머 처리를 등록
    window.onload = function () {
        MYAPP.timer = window.setTimeout(
                // 현재 시각을 <div id='result'> 태그에 표시
                // (3000밀리 초 후 1회만 표시)
                function () {
                    var dat = new Date();
                    document.getElementById('result').innerHTML =
dat.toLocaleTimeString();
                },
                3000
        );
    };
</script>
... 중략 ...
<input type="button" value="클리어" />
<div id="result">
</div>
```



프레임 애니메이션 만들기

- setInterval() 을 사용해 애니메이션 만들기



window 객체 프로퍼티

status	브라우저의 상태바에 문자열을 출력하는 경우에 사용
defaultStatus	브라우저의 상태바에 초기 문자열을 설정
length	창안의 프레임 수
name	창 이름
self	현재 창 자신, window와 같음
window	현재 창 자신, self와 같음
parent	프레임에서 현재프레임의 상위프레임
top	현재프레임의 최상위프레임
opener	open()으로 열린 창에서 볼 때 자기를 연 창
document	document 오브젝트
frames	창안의 모든 프레임에 대한 배열정보
history	history 오브젝트 및 배열
location	location 오브젝트
closed	창이 닫혀 있는 상태
locationbar	location 바
menubar	창 메뉴 바
innerHeight	창 표시 영역의 높이(픽셀), 익스플로러 지원되지 않음
innerWidth	창 표시 영역의 너비(픽셀), 익스플로러 지원되지 않음
outerHeight	창 바깥쪽 둘레의 높이, 익스플로러 지원되지 않음
outerWidth	창 바깥쪽 둘레의 너비, 익스플로러 지원되지 않음
pageXOffset	현재 나타나는 페이지의 X위치, 익스플로러 지원되지 않음
pageYOffset	현재 나타나는 페이지의 Y위치, 익스플로러 지원되지 않음
personalbar	창의 퍼스널 바
scrollbar	창의 스크롤 바
statusbar	창의 상태 바
toolbar	창의 툴 바

window 객체 메서드

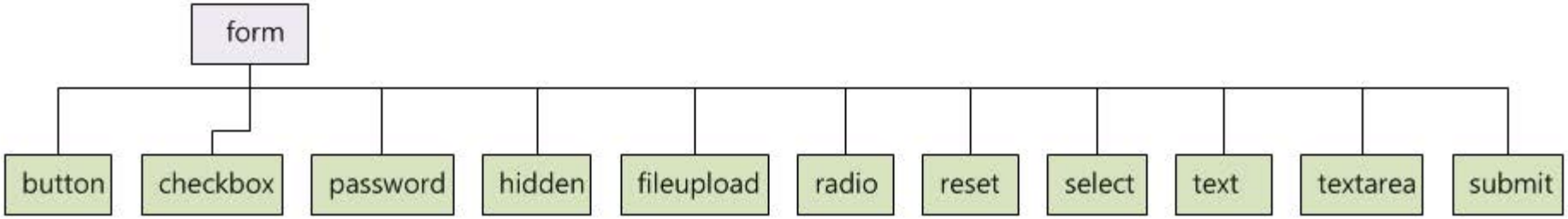
alert()	경고용 대화상자를 보여줌
clearTimeout()	setTimeout 메소드를 정지
confirm()	확인, 취소를 선택할 수 있는 대화상자를 보여줌
open()	새로운 창을 오픈
prompt()	입력창이 있는 대화상자를 보여줌
setTimeout()	일정 간격으로 함수를 호출하여 수행, millisecond 단위로 지정
eval()	문자열을 숫자로 바꿈
toString()	오브젝트를 문자열로 바꿈
blur()	focus를 이동
focus()	focus를 줌
scroll()	창을 스크롤 함
valueOf()	오브젝트 값을 반환
back()	한 단계 전 URL(이전화면)로 돌아감. 익스플로러 지원 안함
find()	창안에 지정된 문자열이 있는지 확인, 있다면 true 없으면 false. 익스플러러 지원 안함
forward()	한 단계 뒤의 URL(다음화면)로 이동. 익스플로러 지원 안함
home()	초기화 홈페이지로 이동. 익스플로러 지원 안함
moveby()	창을 상대적인 좌표로 이동. 수평방향과 수직방향의 이동량을 픽셀로 지정
moveto()	창을 절대적인 좌표로 이동. 창의 왼쪽 상단 모서리를 기준으로 픽셀을 지정
resizeby()	창의 크기를 상대적인 좌표로 재설정. 밑변의 모서리를 기준으로 수평방향, 수직방향을 픽셀로 지정
resizeto()	창의 크기를 절대적인 좌표로 재설정. 창 크기를 픽셀로 지정
scrollby()	창을 상대적인 좌표로 스크롤. 창의 표시영역의 수평방향과 수직방향에 대해 픽셀로 지정
scrollto()	창을 절대적인 좌표를 스크롤. 창의 왼쪽 상단 모서리를 기준으로 픽셀로 지정
stop()	불러오기를 중지. 익스플로러는 지원 안함
captureEvents()	모든 타입의 이벤트를 판단
setInterval()	일정시간마다 지정된 처리를 반복
clearInterval()	setInterval 메소드의 정지
handleEvent()	이벤트 취급자를 정함
print()	화면에 있는 내용을 프린터로 출력
releaseEvent()	다른 계층의 이벤트로 이벤트를 넘김
routeEvent()	판단한 이벤트와 같은 계층의 이벤트
toSource()	오브젝트값을 문자열로 반환

window 객체 이벤트 핸들러

blur	onBlur	브라우저가 포커스를 잃을 때 발생
dragdrop	onDragDrop	사용자가 다른곳에서 객체를 브라우저 안에 넣으려고 할 때 발생. 익스플로러는 지원 안함
error	onError	문서를 읽는 중에 에러가 생길 때 발생
focus	onFocus	브라우저에 포커스를 얻을 때 발생
load	onLoad	문서를 읽을 때 발생
unload	onUnload	현재 문서를 지울려고 할 때 발생
move	onMove	브라우저의 위치를 변경했을 때 발생. 익스플로러는 지원 안함
resize	onResize	창의 크기를 변경했을 때 발생. 익스플로러는 지원 안함

Form 객체

- 텍스트 박스와 같은 다양한 입력 요소를 한 묶음으로 관리하기 위한 객체



- **폼에서 입력된 정보는 통상 서버에 송신되어 처리에 이용**

멤버	개요
*name	폼의 이름
method	송신 방법(GET/POST)
enctype	인코딩 방법
action	폼 입력 데이터 처리 url
submit()	폼의 내용을 submit

- 폼은 어떻게 JavaScript로 다룰 수 있을까?
  - DOM 이용

Form 의 유효성 검사

```
<script>
    var MYAPP = {} ;

    window.onload = function (event) {
        MYAPP.fm = document.getElementsByName('fm');
        MYAPP.fm[0].onsubmit = function (event) {
            if( this.age.value == '' ) {
                alert ("나이를 입력하세요.");
                this.age.focus();
                return false;
            }
            if( this.name.value == '' ){
                alert ("이름을 입력하세요.");
                this.name.focus();
                return false;
            }
        };
    }
</script>...
<form name='fm' action="http://www.naver.com" method="get"
      enctype="multipart/form-data" >
    <label>나이: <input type="text" name="age" value=""  size="3"  />세 </label>
    <label>이름: <input type="text" name="name" value="" size="20" /> </label>
    <input type="submit" value="전송" />
</form>
```

나이: 세 이름: 이름을 입력하세요 전송

Form 입력 요소

분류	요소명	개요
텍스트	Text(<input type="text">)	텍스트 박스
	Password(<input type="password">)	패스워드 입력 박스
	Textarea(<textarea>)	텍스트 영역(복수행 텍스트 입력 박스)
라디오/체크	Radio(<input type="radio">)	라디오 버튼(단일 선택)
	Checkbox(<input type="checkbox">)	체크 박스(복수 선택)
선택	Select(<select>)	선택 박스(단일 선택)
	Select(<select multiple>)	리스트 박스(복수 선택도 가능)
버튼	Submit(<input type="submit">)	확정(송신) 버튼
	Reset(<input type="reset">)	리셋(취소) 버튼
	Button(<input type="button">)	일반적인 버튼
그 외	File(<input type="file">)	파일 선택 박스
	Hidden(<input type="hidden">)	은폐 필드

text

password

textarea

radio 사과 귤 딸기

checkbox 사과 귤 딸기

select 딸기

multiple select WIN MAC UNIX

file Choose File No file chosen

button 버튼

submit 송신

reset 취소

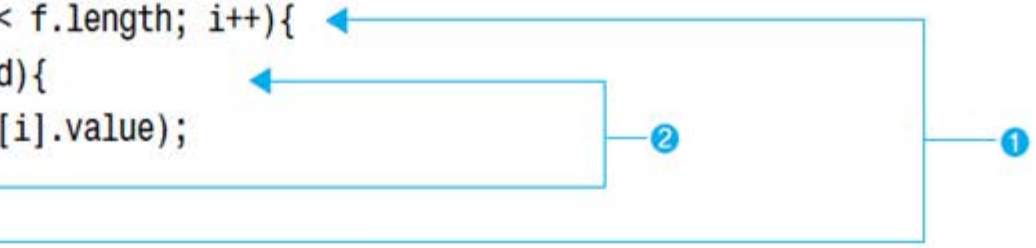
```
rlencoded" id="fm" >

>
""/><br/>
</textarea><br/>
>사과
e" checked>귤
e">딸기<br/>
"PC">사과
"SP">귤
"RD" checked>딸기

>

X

=" /><br/>
 /><br/>
```

폼 요소 - 어떻게 JavaScript로 다룰 수 있을까?

```
<script>
    window.onload =  function () {

        document.getElementById('fm').onsubmit = function(event){

            var fmt = document.fm;
            //var name = document.forms[0]
            //var name = document.forms['fm']

            if(!fmt.text.value) {
                window.alert('text 가 비었음');
                fmt.text.focus();
                return false;
            }

            ...
            var checkboxs = document.querySelectorAll('input[name="checkbox"]:checked');
            if(checkboxs.length ==0) {
                window.alert('checkbox 가 비었음');
                fmt.checkbox[0].focus();
                return false;
            }
            return true;
        };
    }
</script>
```

Form 입력 요소 - checkbox

```
<script type="text/javascript">
<!--
function show(){
  var result = [];
  var f = document.fm.food;
  for(var i = 0; i < f.length; i++){
    if (f[i].checked){
      result.push(f[i].value);
    }
  }
  window.alert(result.toString());
  return false;
}
//-->
</script>
  ... 중략 ...
<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="송신" />
</form>
```



Form 입력 요소 - radio

```
function getRadio(elem){ // 주어진 Element 배열(라디오 버튼)에서 선택값을 추출
    var result = '';
    for(var i = 0; i < elem.length; i++){ // 배열을 순서대로 조사하여 체크 상태 확인
        if (elem[i].checked){ // 체크 상태인 것을 발견하면 변수 result에 그 값을 세트
          result = elem[i].value;
          break;
        }
    }
    return result;
}
function show(){
    window.alert(getRadio(document.fm.food));
    return false;
}
... 중략 ...
<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='송신' />
</form>
```

location 객체

- JavaScript로 **페이지 이동**을 제어하는 객체

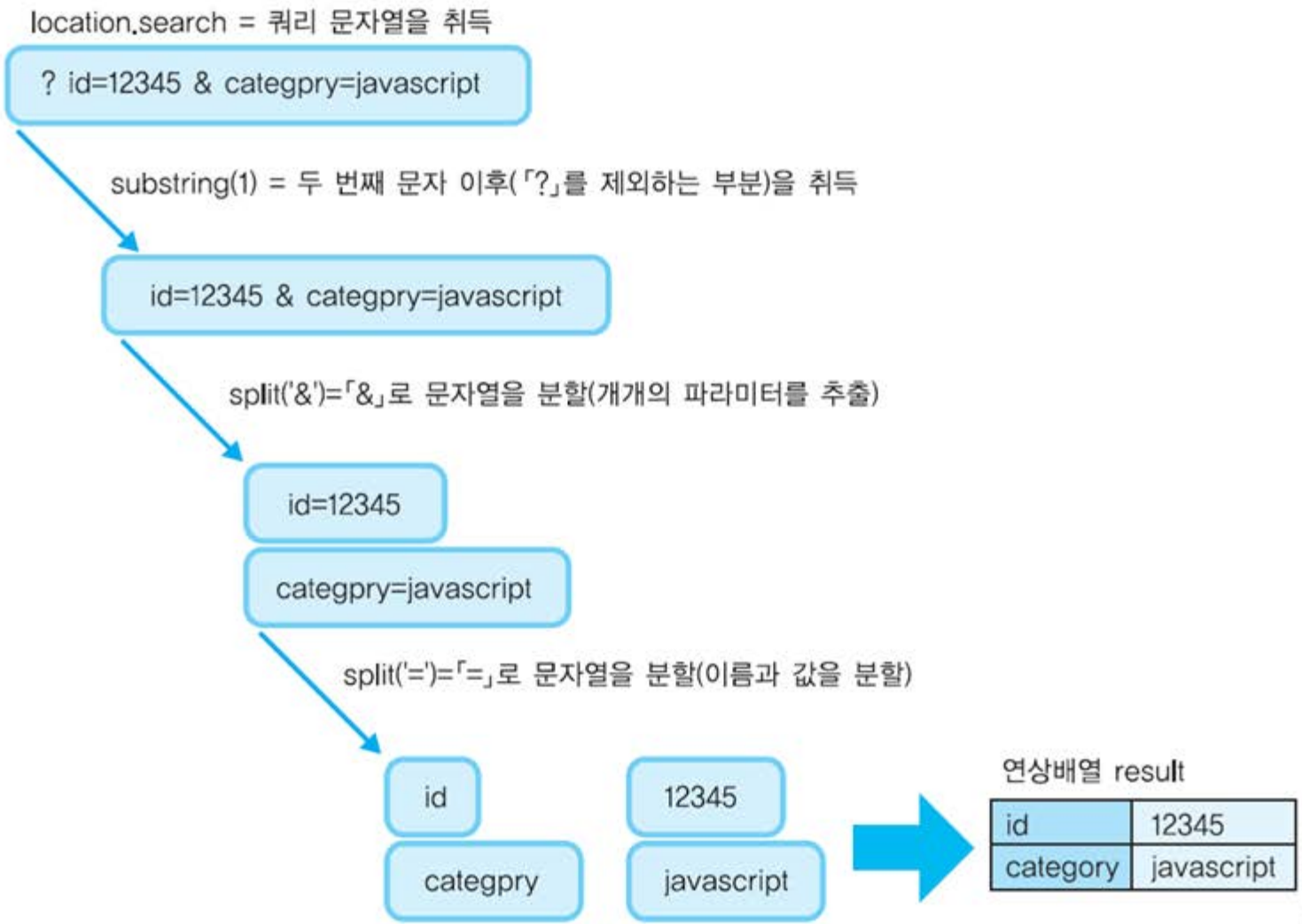
```
location.href  = 'http://www.aaa.co.kr';
location.search = '?id=12345'
```

멤버	개요	반환값의 예
hash	앵커명(#~)	#gihyo?id=12345
host	호스트(호스트명 + 포트 번호. 단, 80의 경우는 포트 번호 생략)	www.wings.msn.to:8080
hostName	호스트명	www.wings.msn.to
href	링크하는 장소	http://www.wings.msn.to:8080/js/sample.html#gihyo?id=12345
*pathname	패스명	js/sample.html
*port	포트 번호	8080
*protocol	프로토콜명	http:
search	쿼리 정보	?id=12345
reload()	현재의 페이지를 리로드함	–
replace(url)	지정 페이지에 이동	–

JavaScript로부터 자동적으로 그 페이지로 이동하는 예제

```
<script>
    MYAPP = {};
    window.onload = function (e) {
        MYAPP.site = document.getElementById('site'); // 선택이 변경되면 실행
        MYAPP.site.onchange = function ( event ){
            if( this.value ) {
                // 선택 상자에서 취득한 값을 기초로 링크할 곳의 URL을 생성
                location.href = 'http://' + this.value + '.naver.com/';
            }
        };
    };
</script>
--중략--
<form name="fm">
    <select id="site">
        <option value="">---선택해주세요---</option>
        <option value="mail">mail.naver.com</option>
        <option value="cafe">cafe.naver.com</option>
        <option value="blog">blog.naver.com</option>
        <option value="news">news.naver.com</option>
    </select>
</form>
```

location 객체 - 쿼리 정보 취득하기



쿼리 정보 취득하기 (1/2) - 함수는 객체다.

```
<script>
    var q = function (){
        var result = {};
        var str = location.search.substring(1, location.search.lenght);
        // var str = location.search.substring(1);

        var params = str.split('&');

        for(var i = 0; i < params.length; i++){
            var kv = params[i].split('=');
            result[kv[0]] = decodeURIComponent(kv[1]);
        }
        return result;
    };
    var result = q();
    for ( var name in result ) {
        console.log(name+':'+ result[name] );
    }
    ... 뒤장에 이어서...
```

연상배열 result

id	12345
category	javascript

```
    ... 앞장에 이어서 ...
    window.onload = function (e){
        var button = document.getElementById('qmake');
        button.onclick = function (e) {
            var datetime = new Date();
            var year = datetime.getFullYear();
            var month = datetime.getMonth();

            var url = '?year='+ year + '&month=' + month ;

            location.href = url;
        };
    };
</script>
... 중략 ...
<form name="fm" action="???" method="???"  enctype="???" >
    <input type="button" id="qmake" name='text' value="url query">
</form>
```

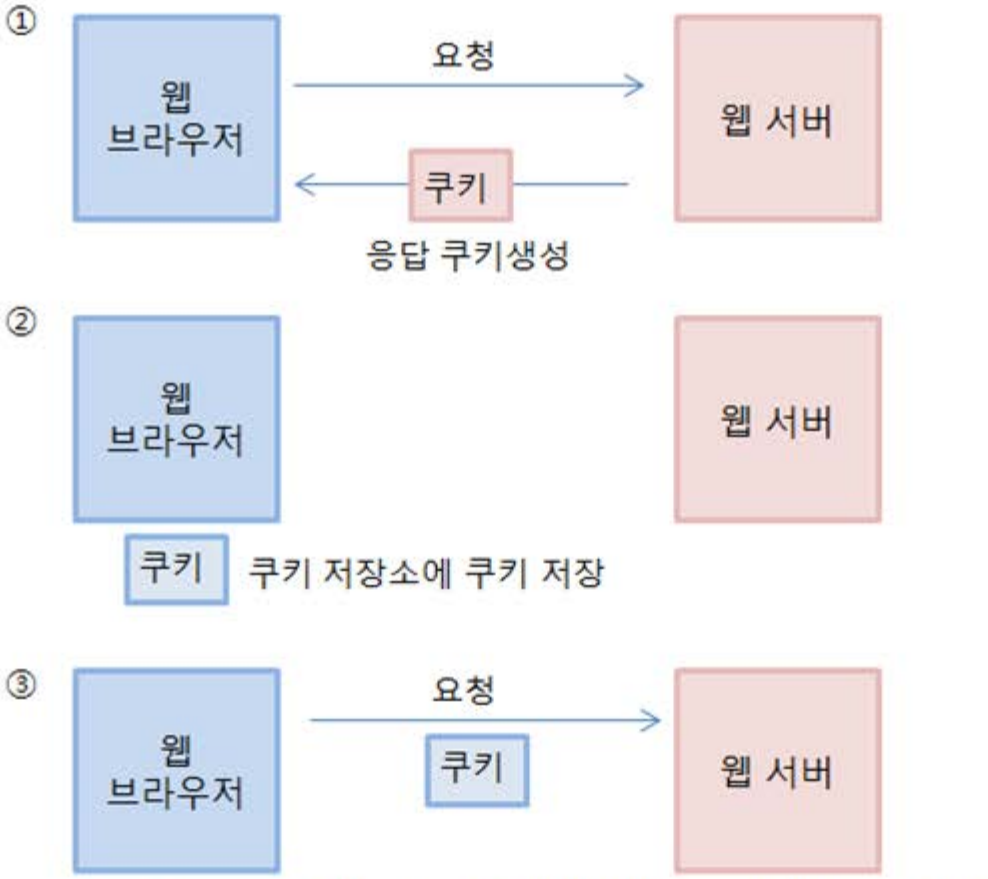
document 객체

- 가능한 한 표준적인 DOM을 이용하도록 하자
  - 윈도우 내에 표시된 문서를 조작하는 것은 document 객체의 역할
  - 객체는 많은 프로퍼티/메소드를 공개하고 있으나, 대부분이 나중에 설명할 DOM(Document Object Model)으로 대체 가능

- 쿠키를 기록/취득한다
  - 자바스크립트로부터 클라이언트에 데이터를 쓸 수 있는 유일한 것: 쿠키
  - 쿠키(Cookie)
    - 클라이언트 측에 보존되는 작은 텍스트
    - 스크립트로부터 무언가 원하는 정보를 보관/유지

쿠키

- 쿠키는 클라이언트 데이터 저장 방법이다.



쿠키 vs 세션

- **쿠키** : 클라이언트에 저장되는 정보
- **세션** : 서버에 저장되는 정보

구분	쿠키	세션
저장 위치	클라이언트	서버
저장 형식	Text로 저장	Object 로 저장
종료 시점	expire data가 지났거나 expire data 설정하지 않았으면 브라우저 종료 시나	브라우저를 닫거나, expire data가 지난 경우.
자 원	클라이언트의 자원을 사용	서버의 자원을 사용
용도	사이트 재 방문시 사용자 정보를 기억하기 위해 사용 (ID, PW, 팝업창 제한등)	서버를 이용하는 동안에 사용자 정보를 유지하기 위해 사용
용량 제한	한 도메인 당 20개, 쿠키 하나 당 4KB, 총 300개	서버가 허용하는 한 용량에 제한이 없음

쿠키 생성 및 사용

(1) 생성

```
Cookie cooke = new Cookie(String name, String value);
```

(2) 쿠키 추가(생성후에 반드시 추가)

```
response.addCookie(cookie);
```

(3) 값을 수정

```
cookie.setValue(newValue);
```

(4) 읽기

- 쿠키를 읽어 올 때

```
Cookie[] cookies = request.getCookies();
```

- 쿠키 이름 읽기

```
String cookies[i].getName();
```

- 쿠키 값읽기

```
String cookies[i].getValue();
```

(5) 쿠키의 수명(지속시간)

```
cookie.setMaxAge(int expiry);
```

Document 객체 - 쿠키

- document.cookie 프로퍼티
  - 쿠키를 읽고 쓸 수 있다.

- document.cookie는 어디에 저장되는가?
  - 클라이언트에..

팝업 창 만들기(1/2)

```
<script src="MYAPP.Variable.js"></script>
<script src="MYAPP.Cookie.js"></script>

<script>
    window.onload = function (e) {

        // 쿠키 존재 여부 확인
        if( MYAPP.Cookie.read('naps') ) {
            // 쿠키가 있는 경우 : 창 닫기.
            window.self.close();   // 자기 자신을 닫는다.
        }
        else {
            // 쿠키가 없는 경우 : 패스
        }

        ... 뒷장에 이어서...
```

팝업 창 만들기(2/2)

```
        ... 앞장에 이어서...
        // <input type="checkbox" name="chkClose"> 찾으시오
        var chkClose = document.getElementsByName('chkClose');
        chkClose[0].onclick = function (e) {
            // check 여부를 확인

            if( chkClose[0].checked === true ){
                // 쿠키 생성
                MYAPP.Cookie.create('naps', 'aaa', 1 );
            }

            // 창 닫기
            window.self.close();
        };
    };
</script>
```