EventHandling

Event Handling

- Event Handling
  - ✓ Event: 다른 것에 영향을 미치는 사건
    - 키보드로 키를 입력하거나 마우스 클릭하는 동작 등
    - 애플리케이션 사용자가 발생할 수 있고 애플리케이션이 스스로 발생시키는 것도 있음
  - ✓ Event Handling
    - 이벤트가 발생했을 때 처리를 하도록 하는 작업
    - 이벤트 처리를 위한 객체를 Event Handler 라고 함
  - ✓ 종류
    - 마우스 이벤트
    - 키보드 이벤트
    - HTML 프레임 이벤트
    - HTML 입력 양식 이벤트
    - 유저 인터페이스 이벤트
    - 구조 변화 이벤트
    - 터치 이벤트

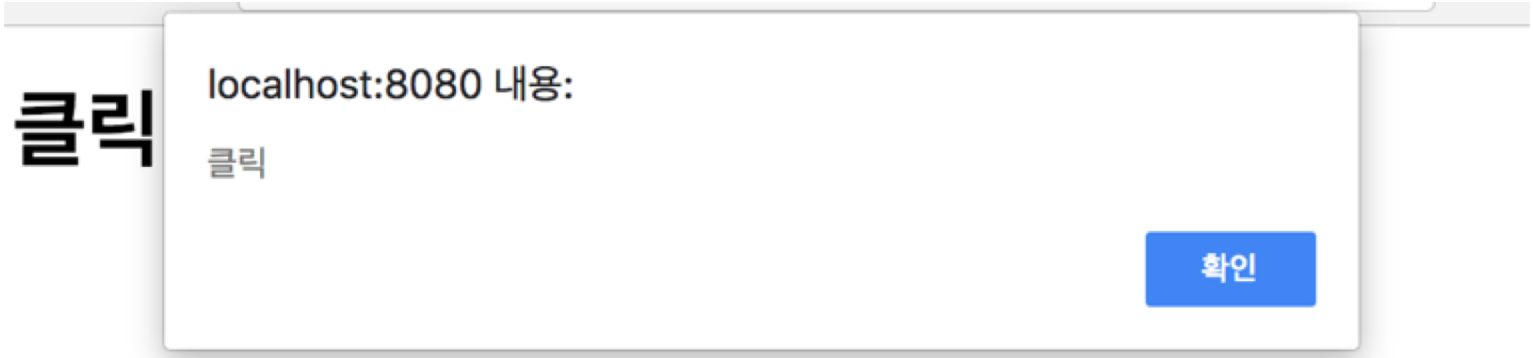
Event Handling

- Event Handling 방법
  - ✓ Inline 처리: 태그 안에 이벤트 속성에 자바스크립트 코드를 추가
  - ✓ 고전 이벤트 모델: JavaScript에서 문서 객체의 이벤트 속성에 함수를 연결하는 구조
    - DOM객체.이벤트이름 = function(){이벤트 처리 내용; }
    - 이벤트 하나에 이벤트 핸들러(함수) 하나 만 연결 가능
    - 이벤트 핸들러 제거 시 문서 객체의 이벤트 속성에 null 할당
  - ✓ 표준 이벤트 모델
    - 웹 표준 단체인 W3C에서 공식 지정한 DOM Level 2 이벤트 모델
    - 하나의 DOM 객체에 여러 개의 이벤트 핸들러 설정 가능
    - 이벤트 핸들러 추가 - addEventListener(eventName, handler)
    - 이벤트 핸들러 제거 - removeEventListener(eventName, handler)
      - eventName: 이벤트 이름을 매개 변수로 입력 – on 제외하고 입력
      - handler: 수행될 함수를 지정

Event Handling

- Event Handling
  - ✓ inline_eventhandling.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>인라인 이벤트 처리</title>
</head>
<body>
        <h1 id='header' onclick="alert('클릭')">클릭</h1>
</body>
</html>
```



Event Handling

- Event Handling
  - ✓ custom_eventhandling.html


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>전통적인 이벤트 처리</title>
</head>
<body>
    <h1 id='header'>클릭</h1>
</body>
<script>
        // 변수를 선언합니다.
        var header = document.getElementById('header');
        // 이벤트를 연결합니다.
        header.onclick = function() {
            alert('클릭');
        };
</script>
</html>
```

Event Handling

- Event Handling
  - ✓ standard_eventhandling.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>표준 이벤트 모델</title>
</head>
<body>
    <h1 id="myHeader">Click</h1>
</body>
<script>
        var header = document.getElementById('myHeader');
        header.addEventListener('click', function() {
            this.innerHTML += '+';
        });
</script>
</html>
```

Event Handling

- event 객체
  - ✓ 이벤트 정보를 저장하는 객체
  - ✓ 이벤트 객체 가져오기
    - var 이벤트변수 = 이벤트 처리 함수의 매개변수 || window.event;
    - 매개변수가 존재하면 매개변수를 변수에 저장하고 매개변수가 undefined이면 window.event 속성을 변수 event에 넣음
    - 인터넷 익스플로러 8 이하의 버전에서는 이벤트가 발생시 이벤트 객체가 window.event 속성으로 전달되기 때문에 위처럼 작성
    - 다른 브라우저는 이벤트 핸들러의 매개 변수로 전달
  - ✓ 이벤트 처리 함수에는 this가 자동으로 넘어가는데 this는 이벤트가 발생한 객체

Event Handling

- event 객체
  - ✓ event_this.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>이벤트 처리</title>
</head>

<body>
    <h1 id='header'>클릭</h1>
</body>
<script>
        // 변수를 선언합니다.
        var header = document.getElementById('header');
        // 이벤트를 연결합니다.
        header.addEventListener("click",function() {
            this.style.color = "Orange"
        });
</script>
</html>
```

Event Handling

- event 객체
  - ✓ 이벤트 객체의 프로퍼티
    - data: DragDrop 발생 시 드롭된 객체들의 URL을 문자열 배열로 반환
    - height: 윈도우 또는 프레임의 높이를 반환
    - layerX, layerY: resize 이벤트가 발생한 객체의 높이 나 너비 또는 이벤트가 발생한 레이어 상의 X 좌표 나 Y 좌표를 픽셀 값으로 반환하며 x 프로퍼티와 동일
    - modifier: 같이 누른 키값을 반환
    - pageX, pageY: 페이지 상의 X 좌표 또는 Y 좌표 리턴 – 익스플러러에서 지원하지 않음
    - screenX, screenY: 스크린 상의 X 좌표 또는 Y 좌표 리턴
    - target: 이벤트가 전달된 객체를 문자열로 반환
    - type: 이벤트의 타입을 문자열로 반환
    - which: 마우스 버튼의 ASCII 코드 값을 반환
      - ◆ IE 에서는 window.event.keyCode를 사용
    - width: 윈도우 또는 프레임의 너비
    - x: layerX와 동일
    - y: layerY와 동일

Event Handling

- event 객체
  - ✓ event_object.html

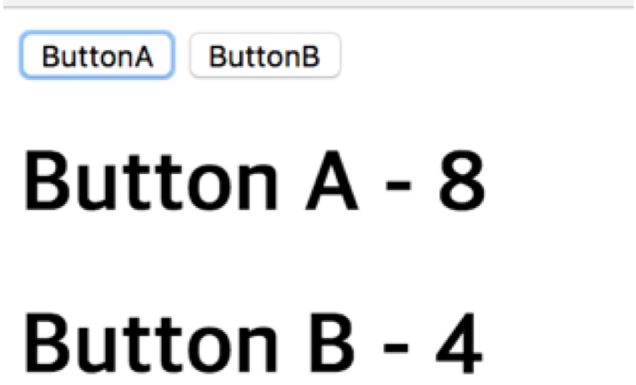
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>이벤트 객체 내용</title>
</head>

<body>
    <h1 id='header'>클릭</h1>
</body>
<script>
    document.body.addEventListener("click",
            (e) => {
                var event = e || window.event;
                document.body.innerHTML = '';
                for ( var key in event) {
                    document.body.innerHTML += '<p>' + key + ': ' + event[key]
                            + '</p>';
                }
            });
</script>
</html>
```

```
isTrusted: true
screenX: 833
screenY: 301
clientX: 48
clientY: 37
ctrlKey: false
shiftKey: false
altKey: false
metaKey: false
button: 0
buttons: 0
relatedTarget: null
pageX: 48
pageY: 37
```

Event Handling

- Event Trigger : 이벤트 강제 발생
  - ✓ 고전적 이벤트 모델 - 객체의 on이벤트( ) 호출
  - ✓ 표준 이벤트 모델 - 객체의 이벤트( ) 호출



Event Handling

- Event Trigger
  - ✓ event_trigger.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>이벤트 강제 발생</title>
</head>
<body>
    <button id="button_a">ButtonA</button>
    <button id="button_b">ButtonB</button>
    <h1>
        Button A - <span id="counter_a">0</span>
    </h1>
    <h1>
        Button B - <span id="counter_b">0</span>
    </h1>
</body>
```

Event Handling

- Event Trigger
  - ✓ event_trigger.html

```
<script>
    var buttonA = document.getElementById('button_a');
    var buttonB = document.getElementById('button_b');
    var counterA = document.getElementById('counter_a');
    var counterB = document.getElementById('counter_b');
    buttonA.addEventListener("click",() => {
        counterA.innerHTML = Number(counterA.innerHTML) + 1;
    });
    buttonB.addEventListener("click",() => {
        counterB.innerHTML = Number(counterB.innerHTML) + 1;
        buttonA.click();
    });
</script>
</html>
```

Event Handling

- DefaultEventHandler
  - ✓ 일부 HTML 태그는 기본적으로 이벤트 핸들러를 소유하고 있는 경우가 있음
    - a 태그는 다른 URL 이나 책갈피로 이동
    - input type이 submit이면 form의 action으로 이동
    - Input type이 reset 이면 폼 안의 내용을 clear
    - button 태그를 form 안에 사용하면 submit 기능
    - keydown 은 누른 문자를 화면에 표시
  - ✓ 디폴트 이벤트를 수행하지 않도록 하기
    - 디폴트 이벤트를 수행하지 않도록 하기 위해서는 event객체의 preventDefault()를 호출해서 기본 이벤트를 수행하지 않도록 할 수 있음

Event Handling

- DefaultEventHandler
  - ✓ default_event.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>기본 이벤트 제거</title>
</head>
<body>
        <form id="my_form" action="server.jsp">
            <label for="name">이름</label><br />
            <input type="text" name="name"
                id="name" /><br />
            <label for="pass">비밀번호</label><br />
            <input type="password" name="pass" id="pass" /><br />
            <label for="pass_check">비밀번호 확인</label><br />
            <input type="password" id="pass_check" /><br /> <input type="submit"
    value="제출" />
        </form>
</body>
```

Event Handling

- DefaultEventHandler
  - ✓ default_event.html

```
<script>
        // 이벤트를 연결합니다.
        document.getElementById('my_form').addEventListener("submit", (e) => {
            // 변수를 선언합니다.
            var pass = document.getElementById('pass').value;
            var pass_check = document.getElementById('pass_check').value;
            // 비밀번호가 같은지 확인합니다.
            if (pass == pass_check) {
                alert('성공');
            } else {
                alert('다시 입력해주세요.');
                e.preventDefault();
            }
        });
</script>
</html>
```

Event Handling

- 이벤트 버블링
  - ✓ 부모와 자식에서 동일한 이벤트를 처리하게 되면 자식에서 이벤트가 발생한 경우 부모쪽으로 연속적으로 이벤트가 발생하는데 이것이 이벤트 버블링
  - ✓ 이벤트 버블링 막기
    - 인터넷 익스플로러: 이벤트 객체의 cancelBubble 속성 true로 변경
    - 그 이외의 브라우저: 이벤트 객체의 stopPropagation() 함수 사용

Event Handling

- Event Bubbling
  - ✓ event_bubble.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>이벤트 버블링</title>
</head>
<body>
        <h1 id="header">
            <span id="paragraph">Pagagraph</span>
        </h1>
</body>
```

Event Handling

- Event Bubbling
  - ✓ event_bubble.html

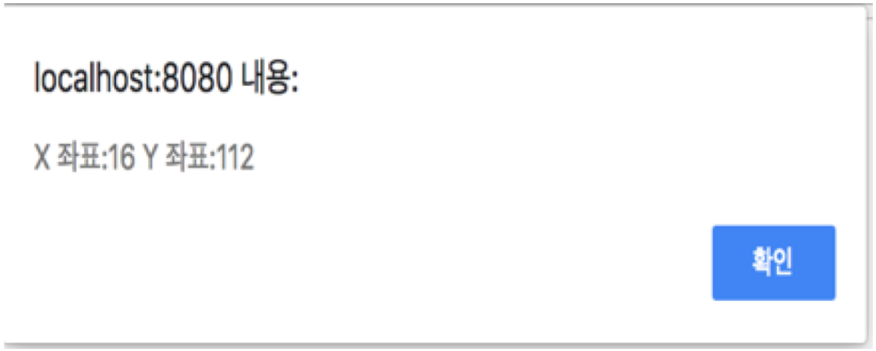
```
<script>
        document.getElementById('header').addEventListener("click", function() {
            alert('header');
        });
        document.getElementById('paragraph').addEventListener("click", function(e) {
            var event = e || window.event;
            alert('paragraph');

            event.cancelBubble = true;
            if (event.stopPropagation) {
                event.stopPropagation();
            }
        });
</script>
</html>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ click 이벤트
    - 마우스 버튼을 클릭하는 순간 발생하는 이벤트로 마우스 버튼을 누르는 mousedown 이벤트와 버튼을 놓는 순간 발생하는 mouseup 이벤트가 합쳐진 이벤트
    - click 이벤트가 발생하면 event 객체의 type 프로퍼티에는 click이 저장되고 layerX, layerY, pageX, pageY, screenX, screenY 프로퍼티에는 이벤트 발생 시 커서의 x좌표, y좌표의 값이 저장
    - 왼쪽 버튼을 누르면 which 프로퍼티에는 1의 값이 오른쪽 버튼을 누르면 3의 값이 저장
    - ALT, CTRL, SHIFT 키를 누르고 누른 경우를 알 수 있는 프로퍼티도 존재
  - ✓ dblclick 이벤트
    - 마우스 버튼을 더블 클릭하는 순간 발생하는 이벤트
    - 프로퍼티는 click과 동일하며 일반적으로 HTML 태그에서 body 태그에 기재

Event Handling



- 여러 종류의 이벤트
  - ✓ click.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>클릭이벤트</title>

</head>
<body>
    <a href="#" id="btn">여기를 클릭</a>
</body>
<script>
        var test = (e) => {
            var event = e || window.event;
            var x = event.screenX;
            var y = event.screenY;
            alert("X 좌표:" + x + " Y 좌표:" + y);
        }
        document.getElementById("btn").addEventListener("click", test);
</script>
</html>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ keydown
    - 사용자가 키를 누르는 순간 발생하는 이벤트로 keypress 이벤트 이전에 발생
    - 기본 이벤트 처리로 입력한 키보드의 값을 출력
    - event 객체의 프로퍼티
      - type: KeyDown
      - layerX, layerY, pageX, pageY, screenX, screenY: 이벤트 발생 시 커서의 X좌표 및 Y 좌표
      - which: 누른 키의 ASCII 코드 값을 저장하고 있는 프로퍼티로 이 프로퍼티가 없을 때는 keyCode에 누른 키 값이 저장
      - code: 상수
      - key: 문자
  - ✓ keypress
    - 사용자가 키를 눌렀거나 누르고 있는 동안 발생하는 이벤트로 keydown 이벤트가 true 값을 반환하는 순간 발생하는 것으로 사용자가 키를 놓는 순간 완료
  - ✓ keyup
    - 키보드에서 키를 놓는 순간 발생하는 이벤트

Event Handling

t

which	code	key
91:number	MetaLeft:string	Meta:string

Event Handling

- 여러 종류의 이벤트
  - ✓ key.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>키보드 이벤트</title>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ key.html

```
<style>
        div{
            text-align:center;
        }
        span{
            display:block;
            float:left;
            width:30%;
            border:1px solid black;
            text-align:center;
            font:1.5em serif;
        }
        .titlespan{
            margin-top:1em;
        }
        .valuespan{
            height:5em;
            font:3em serif;
        }
</style>
</head>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ key.html

```
<body>
        <input size="40" placeholder="문자를 입력하세요" id="keyinput" />
        <div>
            <span class="titlespan">which</span>
            <span class="titlespan">code</span>
            <span class="titlespan">key</span>
        </div>
        <div>
            <span class="valuespan" id="whichspan"></span>
            <span class="valuespan" id="codespan"></span>
            <span class="valuespan" id="keyspan"></span>
        </div>
</body>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ key.html

```
<script>
        var keyinput = document.getElementById("keyinput")
        keyinput.addEventListener("keydown", (e) => {
            var event = e || window.event;
            var keyCode = ('which' in event) ? event.which : event.keyCode;
            document.getElementById("whichspan").innerHTML = keyCode + ":" + typeo
    f(keyCode);
            document.getElementById("codespan").innerHTML = event.code + ":" + type
    of(event.code);
            document.getElementById("keyspan").innerHTML = event.key + ":" + typeof(
    event.key);
        });
</script>
</html>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ mousedown
    - 사용자가 마우스 버튼을 누르는 순간 발생하는 이벤트로 이 이벤트가 false를 반환하면 선택 또는 링크 등의 동작들이 중지
    - event 객체의 프로퍼티
      - type: MouseDown
      - layerX, layerY, pageX, pageY, screenX, screenY: 이벤트 발생 시 커서의 X좌표 및 Y 좌표
      - which: 왼쪽 버튼이면 1 오른쪽 버튼이면 3이 설정
      - modifiers: 함께 누른 키 값
  - ✓ mousemove(Area, Layer, Link)
    - 사용자가 마우스의 커서를 이동하는 순간 발생하는 이벤트로 captureEvents() 함수에 이 이벤트를 얻어내도록 설정하면 작업을 수행

Event Handling

- 여러 종류의 이벤트
  - ✓ mouseout
    - 이벤트가 발생한 객체에서 마우스 커서가 빠져 나가는 순간 발생하는 이벤트
  - ✓ mouseover
    - 이벤트가 발생한 객체에 마우스 커서가 들어오는 순간 발생하는 이벤트
  - ✓ mouseup(button, document, Link)
    - 사용자가 마우스 버튼을 놓는 순간 발생하는 이벤트로 이 이벤트가 false를 반환하면 선택 또는 링크 등의 동작들이 중지

Event Handling

- 여러 종류의 이벤트
  - ✓ move 이벤트(window, frame)
    - 윈도우 또는 프레임이 움직이는 순간 발생하는 이벤트
    - event 객체의 프로퍼티
      - type: Move
      - screenX, screenY: 이벤트 발생 시 커서의 X 좌표 및 Y 좌표
  - ✓ resize 이벤트
    - 윈도우 또는 프레임이 크기가 변경되는 순간 발생하는 이벤트
    - event 객체의 프로퍼티
      - type: Resize
      - screenX, screenY: 이벤트 발생 시 커서의 X 좌표 및 Y 좌표

Event Handling

- 여러 종류의 이벤트
  - ✓ focus: 포커스를 받았을 때 발생하는 이벤트 - focusin
  - ✓ blur: 포커스를 잃어 버렸을 때 발생하는 이벤트(input 객체) - focusout
  - ✓ load: 메모리에 적재 되었을 때 발생하는 이벤트(window, file, image 객체 등)
    - script를 head에 작성하는 경우 body의 객체를 사용하고자 하는 경우 window에 load 이벤트가 발생한 이후에 사용
  - ✓ change: 값이 변경되었을 때 호출되는 이벤트(input 객체)
  - ✓ submit: 폼의 데이터가 전송될 때(form 객체-기본 기능(서버로 전송)이 있음)
  - ✓ reset: 폼의 데이터가 클리어 될 때(form 객체 – 기본 기능(폼의 내용 삭제)이 있음)
  - ✓ beforeunload: 브라우저의 내용이 사라지기 직전
  - ✓ scroll: 스크롤이 발생한 경우
  - ✓ 모바일에서는 화면 터치와 전환에 관련된 이벤트가 존재

Event Handling

파일 선택 car06.jpg

Event Handling

- 여러 종류의 이벤트
  - ✓ imagepreview.html – 이미지 미리보기

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>파일 미리보기</title>
</head>
<body>
    <form id="form1">
        <input type='file' id="imginp" accept="image/*"/>
        <img id="img" width="200" height="200" border="1" />
    </form>
</body>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ imagepreview.html – 이미지 미리보기

```
<script>
    window.addEventListener("load", function(event) {
        document.getElementById("imginp").addEventListener("change",
            (e) => {
                readURL(document.getElementById("imginp"));
            })
    });
    function readURL(input) {
        if (input.files && input.files[0]) {
            var filename = input.files[0].name;

            var reader = new FileReader();

            reader.addEventListener("load", function(e) {
                document.getElementById('img').src = e.target.result;
            });

            reader.readAsDataURL(input.files[0]);
        }
    }
</script>
</html>
```

Event Handling

Infinity Scroll

Infinity Scroll

Infinity Scroll

Infinity Scroll

Infinity Scroll

Infinity Scroll

Infinity Scroll

Infinity Scroll

Event Handling

- 여러 종류의 이벤트
  - ✓ infinitescroll.html – 무한 스크롤

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>무한 스크롤</title>
</head>

<body id="body">
</body>
```

Event Handling

- 여러 종류의 이벤트
  - ✓ infinitescroll.html – 무한 스크롤

```
<script>
    window.addEventListener("load", (e) => {
        for (var i = 0; i < 20; i++) {
            document.getElementById('body').innerHTML += '<h1>Infinity Scroll</h1>';
        }
```

Event Handling

- 여러 종류의 이벤트
  - ✓ infinitescroll.html – 무한 스크롤

```
        window.addEventListener("scroll", (e) => {
            var supportPageOffset = window.pageXOffset !== undefined;
            var isCSS1Compat = ((document.compatMode || "") === "CSS1Compat");
            var scrollTop = supportPageOffset ? window.pageYOffset : isCSS1Compat ?
document.documentElement.scrollTop : document.body.scrollTop;

            var scrollHeight = scrollTop + window.innerHeight;
            var documentHeight = document.body.clientHeight;
            //alert(scrollHeight)
            //alert(documentHeight)
            // 스크롤의 높이와 문서의 높이가 같을 때
            if (scrollHeight >= documentHeight) {
                for (var i = 0; i < 10; i++) {
                    document.getElementById('body').innerHTML += '<h1>Infinity Scr
oll</h1>';
                }
            }
        });
    });
</script>
</html>
```