221212_2반_실습

JavaScript

Event

- 프로그래밍하고 있는 시스템에서 일어나는 사건(action) 혹은 발생(occurrence)인데, 이는 여러분이 원한다면 그것들에 어떠한 방식으로 응답할 수 있도록 시스템이 말해주는 것
- 웹페이지에서 마우스를 클릭했을 때, 키를 입력했을 때, 특정요소에 포커스가 이동되었을 때 어떤 사건을 발생시키는 것
- 이벤트의 종류
  - 포커스 이벤트(focus, blur)
  - 폼 이벤트(reset, submit)
  - 뷰 이벤트(scroll, resize)
  - 키보드 이벤트(keydown, keyup)
  - 마우스 이벤트(mouseenter, mouseover, click, dbclick, mouseleave)
  - 드래그 앤 드롭 이벤트 (dragstart, drag, dragleave, drop)

1. 마우스 이벤트

이벤트	설명
click	요소에 마우스를 클릭했을 때 이벤트가 발생
dbclick	요소에 마우스를 더블클릭했을 때 이벤트가 발생
mouseover	요소에 마우스를 오버했을 때 이벤트가 발생
mouseout	요소에 마우스를 아웃했을 때 이벤트가 발생
mousedown	요소에 마우스를 눌렀을 때 이벤트가 발생
mouseup	요소에 마우스를 떼었을 때 이벤트가 발생
mousemove	요소에 마우스를 움직였을 때 이벤트가 발생
contextmenu	context menu(마우스 오른쪽 버튼을 눌렀을 때 나오는 메뉴)가 나오기 전에 이벤트 발생

이벤트	설명
mouseenter	마우스포인터 요소안 이동

2. 키 이벤트

이벤트	설명
keydown	키를 눌렀을 때 이벤트가 발생
keyup	키를 떼었을 때 이벤트가 발생
keypress	키를 누른 상태에서 이벤트가 발생

3. 폼 이벤트

이벤트	설명
focus	요소에 포커스가 이동되었을 때 이벤트 발생
blur	요소에 포커스가 벗어났을 때 이벤트 발생
change	요소에 값이 변경 되었을 때 이벤트 발생
submit	submit 버튼을 눌렀을 때 이벤트 발생
reset	reset 버튼을 눌렀을 때 이벤트 발생
select	input이나 textarea 요소 안의 텍스트를 드래그하여 선택했을 때 이벤트 발생

4. 로드 및 기타 이벤트

이벤트	설명
load	페이지의 로딩이 완료되었을 때 이벤트 발생
abort	이미지의 로딩이 중단되었을 때 이벤트 발생
unload	페이지가 다른 곳으로 이동될 때 이벤트 발생
resize	요소에 사이즈가 변경되었을 때 이벤트 발생
scroll	스크롤바를 움직였을 때 이벤트 발생

event listener 👂🎙️

- 이벤트 리스너는 DOM 객체에서 이벤트가 발생할 경우 해당 이벤트 처리 핸들러를 추가할 수 있는 오브젝트이다.
- 이벤트 리스너를 이용하면 특정 DOM에 위에 말한 Javascirpt 이벤트가 발생할 때 특정 함수를 호출한다.
- 이벤트 리스너 등록하기

- addEventListener

```
DOM객체. addEventListener(이벤트명, 실행할 함수명, 옵션)
```

- **이벤트명** : Javascript에서 발생할 수 있는 이벤트 명을 입력한다.
- **함수명** : 해당 변수는 생략 가능하며, 해당 이벤트가 발생할 때 실행할 함수 명을 입력한다.
- **옵션**: 옵션은 생략이 가능하며, 자식과 부모 요소에서 발생하는 버블링을 제어하기 위한 옵션이다.

```
addEventListener(type, listener);
addEventListener(type, listener, options);
addEventListener(type, listener, useCapture);
```

- `type`
  - 수신할 이벤트 유형을 나타내는 대소문자 구분 문자열입니다.
- `listener`
  - 지정한 이벤트(`Event` 인터페이스를 구현한 객체)를 수신할 객체입니다. `handleEvent()` 메서드를 포함하는 객체 또는 JavaScript 함수여야 합니다. 이벤트 수신기 콜백에서 콜백 자체에 대한 정보를 더 알아보세요.
- `options` *Optional*
  - 이벤트 수신기의 특징을 지정할 수 있는 객체입니다. 가능한 옵션은 다음과 같습니다.`capture` 이벤트 대상의 DOM 트리 하위에 위치한 자손 `EventTarget`으로 이벤트가 전달되기 전에, 이 수신기가 먼저 발동돼야 함을 나타내는 불리언 값입니다. 명시하지 않을 경우 기본 값은 `false`입니다.`once` 수신기가 최대 한 번만 동작해야 함을 나타내는 불리언 값입니다. `true`를 지정할 경우, 수신기가 발동한 후에 스스로를 대상에서 제거합니다. 명시하지 않을 경우 기본 값은 `false`입니다.`passivetrue`일 경우, 이 수신기 내에서 `preventDefault()`를 절대 호출하지 않을 것임을 나타내는 불리언 값입니다. 이 값이 `true`인데 수신기가 `preventDefault()`를 호출하는 경우, 사용자 에이전트는 콘솔에 경고를 출력하는 것 외에 아무런 동작도 하지 않습니다. 명시하지 않을 경우의 기본 값은 `false`지만, Safari와 Internet Explorer를 제외한 브라우저에서 `wheel` (en-US), `mousewheel` (en-US), `touchstart`, `touchmove` (en-US) 이벤트에서의 기본 값은 `true`입니다. 패시브 수신기로 스크롤 성능 향상에서 이 값에 대해 더 알아보세요.`signal` `AbortSignal`입니다. 지정한 `AbortSignal` 객체

의 `abort()` 메서드를 호출하면 이 수신기가 제거됩니다. 명시하지 않을 경우 이벤트 수신기가 아무 `AbortSignal`에도 연결되지 않습니다.

- `useCapture` *Optional*
  - 이벤트 대상의 DOM 트리 하위에 위치한 자손 `EventTarget`으로 이벤트가 전달되기 전에, 이 수신기가 먼저 발동돼야 함을 나타내는 불리언 값입니다. 캡처 모드인 수신기는 DOM 트리의 위쪽으로 버블링 중인 이벤트에 의해선 발동하지 않습니다. 이벤트 버블링과 캡처링은 조상-자손 관계를 가진 두 개의 요소가 동일한 이벤트 유형에 대한 수신기를 가지고 있을 때, 두 요소에 이벤트가 전파되는 방법을 말합니다. 이벤트 전파 모드에 따라 두 요소 중 이벤트를 먼저 수신하는 쪽이 달라집니다. DOM Level 3 Events와 JavaScript Event 순서에서 자세한 설명을 확인하세요. 기본 값은 `false`입니다.
- 예제

```
<html>
 <a>클릭</a>
</html>
<script>
 const a = document.querySelector('a');
  a.addEventListener('click', showConsole);
  function showConsole() {
    console.log("콘솔로그 실행");
  }
</script>
```

```
var t = document.getElementById('target');
if(t.addEventListener){
    t.addEventListener('click', function(event){
        alert('Hello world, '+event.target.value);
    });
} else if(t.attachEvent){
    t.attachEvent('onclick', function(event){
        alert('Hello world, '+event.target.value);
    })
}
```

```
<input type="button" id="target" value="button" />
<script>
    var t = document.getElementById('target');
    t.addEventListener('click', function(event){
        alert(1);
    });
    t.addEventListener('click', function(event){
        alert(2);
```

```
    });
</script>
```

```
<input type="button" id="target1" value="button1" />
<input type="button" id="target2" value="button2" />
<script>
    var t1 = document.getElementById('target1');
    var t2 = document.getElementById('target2');
    function btn_listener(event){
        switch(event.target.id){
            case 'target1':
                alert(1);
                break;
            case 'target2':
                alert(2);
                break;
        }
    }
    t1.addEventListener('click', btn_listener);
    t2.addEventListener('click', btn_listener);
</script>
```

- removeEventListener
  - 이벤트 리스너의 경우 웹 애플리케이션 메모리 누수의 원인이 될 수 있다.
  - 더 이상 해당 이벤트 리스너가 필요 없다고 하면 반드시 추가된 이벤트 리스너는 반드시 삭제해주어야 한다.
  - 특정 페이지에서만 사용하는 이벤트 리스너라면 해당 페이지를 떠날 때 이벤트 리스너를 삭제해준다.

```
DOM객체. removeEventListener(이벤트명, 실행했던 함수명);
```

  - 예제

```
//VUE Project EventListener 삭제 예시 코드
<script>
export default {
  ...
        mounted() {
            a.addEventListener('click', this.showConsole);
        },
        beforeDestroy() {
            a.removeEventListener('click', this.showConsole)
        },

  ...
```

```
}
</script>
```

- 해당 예제의 경우 Vue 코드에서 eventListener를 이용했다.
- 특정 vue페이지에서 addEventListener를 추가했다면 해당 페이지를 떠날 때
- 반드시 beforeDestroy() 메서드에서 추가한 이벤트 리스너를 removeEventListener로 삭제했다.
- Vue의 mounted는 해당 페이지가 렌더링 되었을 때 실행되고,
- beforeDestroy는 페이지가 떠나기 전 해당 코드가 실행된다.
- 즉, 더 이상 이벤트 리스너가 필요 없다면 꼭 제거해주어야 한다.

event handler 🖐🕹🎮🖱트

- 특정 요소에서 발생하는 이벤트를 처리하기 위해 사용하는 함수
- Event(특정 대상에 가하는 어떠한 행동)에 대한 다음 처리과정을 Event handler라고합니다.
  - 예를 들면, "버튼을 눌렀다"라는 Event가 발생하면, "이름을 알아내서 환영 메시지를 띄운다"
  - 여기서 "이름을 알아내서 환영 메시지를 띄운다"라는 일련의 다음 과정을 정의해 놓은게 Event handler 입니다

```
- Ex) 항아리를 깬다. 깰때는 망치로 깬다. 라고 할 때

      - 항아리                 : 대상(객체)

      - 깬다                    : Event

      - 망치로 항아리를 깬다 : Event Hadler
```

항아리
(객체)

깬다
(Event)

망치로 항아리를 깬다
(Event Handler)

Element 추가

.createElement()

- 요소를 만듭니다.
- 예제

```
document.body.createElement( 'h1' )
```

.createTextNode()

- 선택한 요소에 텍스트를 추가

```
document.createTextNode( 'My Text' )
```

.appendChild()

- 선택한 요소 안에 자식 요소를 추가

```
var jbBtn = document.createElement( 'button' );
//button 요소를 만들고 jbBtn에 저장
var jbBtnText = document.createTextNode( 'Click' );
//Click이라는 텍스트를 만들고 jbBtnText에 저장
jbBtn.appendChild( jbBtnText );
//jbBtn에 jbBtnText를 삽입
document.body.appendChild( jbBtn );
//jbBtn을 body의 자식 요소로 삽입
```