삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

2-1. Event (이벤트)

2-1-1. event handler attribute

2-1-2. event handler property

2-1-3. addEventListener

2-1-4. event handler remove

2. Event

브라우저는 클릭, 키보드 입력, 마우스 이동 등이 일어나면 이를 감지하여 특정한 타입의 이벤트를 발생시킨다. 이벤트가 발생했을 때 호출 될 함수를 **이벤트 핸들러(Event Handler)**라고 하고, 이벤트가 발생했을 때 브라우저에게 이벤트 핸들러의 호출을 위임하는 것을 이벤트 핸들러 등록이라고 한다. 이벤트와 그에 대응하는 함수(이벤트 핸들러)를 통해 사용자와 애플리케이션은 상호작용 할 수 있다.

이벤트 타입

- 이벤트 종류를 나타내는 문자열인 이벤트 타입은 약 200여가지가 있다.
  이벤트 타입에 대한 상세 목록은 MDN의 Event reference에서 확인할 수 있다.
- https://developer.mozilla.org/ko/docs/Web/Events

이벤트 핸들러 등록

- 이벤트가 발생 했을 때 브라우저에게 이벤트 핸들러의 호출을 위임하는 것을
  이벤트 핸들러 등록이라고 하며, 이벤트 핸들러를 등록하는 방법은 3가지다.

2. Event

2-1. Event (이벤트)

2-1-1. event handler attribute (이벤트 핸들러 어트리뷰트 방식)

- HTML 이벤트 핸들러 어트리뷰트 (on 접두사 + 이벤트 타입) 값으로 함수 호출문을 할당하여 이벤트 핸들러를 등록하는 방식이다.
- 주의할 점은 함수 참조가 아닌 함수 호출문을 할당한다는 것이다.

```
<button onclick="alert('클릭했네?'); console.log('클릭했네?');">클릭해보세요</button>
<button onmouseover="hello('수강생');">마우스를  올려보세요</button>
```

2. Event

2-1. Event (이벤트)

2-1-1. event handler attribute (이벤트 핸들러 어트리뷰트 방식)

```
function hello(name){
    alert(`${name}씨, 마우스 올리지마세요!`);
}

/*
핸들러 어트리뷰트 값은 사실 암묵적으로 생성 될 이벤트 핸들러의 함수 몸체를 의미한다.
이는 이벤트 핸들러에 인수를 전달하기 위해서이다.

function onclick(event) {
    alert('클릭했네?');
    console.log('클릭했네?');   // 여러 개의 문을 할당할 수 있다.
}
function onmouseover(event) {
    hello('수강생');
}
*/
```

2. Event

2-1. Event (이벤트)

2-1-2. event handler property (이벤트 핸들러 프로퍼티)

- window 객체와 Document, HTMLElement 타입의 DOM 노드 객체는 이벤트에 대응하는 이벤트 핸들러 프로퍼티를 가지고 있다.
- 이벤트 핸들러 프로퍼티의 키는 이벤트 핸들러 어트리뷰트와 동일하며 (on 접두사 + 이벤트 타입) 이벤트 핸들러 프로퍼티에 함수를 바인딩하면 이벤트 핸들러가 등록된다.
- 이벤트 핸들러 어트리뷰트 방식과 비교했을 때 HTML과 자바스크립트가 뒤섞이는 문제는 해결할 수 있지만 이벤트 핸들러 프로퍼티에 하나의 이벤트 핸들러만 바인딩 할 수 있다는 단점이 있다.

```
const $button = document.querySelector('#btn');

$button.onclick = function() {
    alert('DOM 프로퍼티 방식으로 이벤트 핸들러 등록 완료!');
};

// 이전 이벤트 핸들러는 무시되어 하나의 이벤트 밖에 연결 할 수 없다.
$button.onclick = () => alert('이벤트 덮어쓰기!');
```

```
<button id="btn">클릭해보세요</button>
```

2. Event

2-1. Event (이벤트)

2-1-3. addEventListener

- EventTarget.prototype.addEventListener 메서드를 사용하여 이벤트를 등록할 수 있다.
- EventTarget.addEventListener('eventType', functionName [, useCapture])
  - 첫 번째 매개변수에는 이벤트 타입, 두 번째 매개변수에는 이벤트 핸들러를 전달하고 마지막 매개변수에는 이벤트 전파 단계(캡처링 또는 버블링)을 지정한다. (추후 이벤트 전파에서 다룬다)
- addEventListener 메서드 방식은 이벤트 핸들러 프로퍼티에 바인딩 된 이벤트 핸들러에 아무런 영향을 주지 않는다.
- 동일한 HTML 요소에서 발생한 동일한 이벤트에 대해 addEventListener 메서드 방식으로는 하나 이상의 이벤트 핸들러를 등록 할 수 있으며 이 때 이벤트 핸들러는 등록 된 순서대로 호출된다.

2. Event

2-1. Event (이벤트)

2-1-3. addEventListener

```
<button id="btn">클릭해보세요</button>
```

```
const $button = document.getElementById('btn');

$button.addEventListener('click', function () {
    alert('클릭했네?');
});

// 이벤트 핸들러 프로퍼티 방식 추가
$button.onclick = function () {
    console.log('이벤트 핸들러 프로퍼티 방식으로 이벤트 핸들러 등록!');
}

// addEventListener 메소드 방식 하나 더 추가
$button.addEventListener('click', function () {
    console.log('addEventListener 메소드 방식으로 이벤트 핸들러 등록!');
});

// 참조가 동일한 이벤트 핸들러를 중복 등록하면 하나의 핸들러만 등록된다.
const handleMouseover = () => console.log('button mouseover');
$button.addEventListener('mouseover', handleMouseover);
$button.addEventListener('mouseover', handleMouseover);
```

2. Event

2-1. Event (이벤트)

2-1-4. event handler remove (이벤트 핸들러 제거)

- EventTarget.prototype.removeEventListener 메서드를 통해 addEventListener 메서드로 등록한 이벤트 핸들러를 제거한다.
- removeEventListener 메서드에 전달할 인수는 addEventListener 메서드와 동일하며, 전달한 인수가 일치하지 않을 경우 이벤트 핸들러는 제거되지 않는다.

```
<button id="btn">클릭해보세요</button>
```

```
const $button = document.getElementById('btn');

const handleClick = () => alert('클릭했대요~');

// 이벤트 핸들러 등록
$button.addEventListener('click', handleClick);

// 이벤트 핸들러 제거
$button.removeEventListener('click', handleClick);

// 이벤트 핸들러 등록
// 등록한 이벤트 핸들러를 참조할 수 없으므로 제거할 수 없다.
$button.addEventListener('mouseover', () => alert('마우스 올렸대요~'));
```