



02



02-7. *Event Programming*



JavaScript Basic

이벤트 프로그래밍

이벤트 종류

- 브라우저 이벤트
 - 사용자 이벤트
-
- 브라우저 이벤트는 브라우저 자체에서 발생하는 이벤트입니다.
 - 사용자 이벤트란 애플리케이션 화면에서 사용자가 키보드 혹은 마우스로 발생시키는 이벤트를 의미합니다.
 - 사용자 이벤트는 이벤트 종류에 따라 3가지로 구분해 볼 수 있습니다.
-
- 마우스 이벤트
 - 키 이벤트
 - HTML FORM 관련 이벤트



유저 입력 데이터는 아주 중요하다.

이벤트 프로그래밍

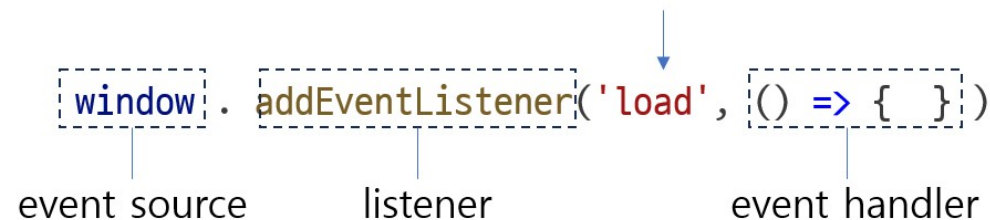
이벤트 프로그래밍 구조

- 애플리케이션에서 이벤트 처리를 하기 위해서는 이벤트 소스와 이벤트 핸들러를 리스너로 연결해야 합니다.

- 이벤트 소스 : 이벤트 발생 ^{node} 객체
- 이벤트 핸들러 : 이벤트 처리 내용
- 리스너 : 이벤트 소스와 이벤트 핸들러를 연결



이벤트 명 이미 내장된 이름을 사용해야 한다.



이벤트 프로그래밍

가장 기본은 1번. 2, 3번은 선별적으로 사용하면 된다.

방법1 - addEventListener() js 파일에 작성하는 것.

- 자바스크립트에서 이벤트를 등록하는 가장 기본적인 방법은 addEventListener() 라는 함수를 이용하는 방법입니다.

이벤트소스 . addEventListener(이벤트 명, 이벤트 핸들러)

콜백 함수(이벤트 콜백)

내가 직접 함수를 호출해서 실행되는 함수를 콜백함수라고 하지 않는다.

내가 함수를 정의했지만, 개발자가 직접 코드로 함수 호출을 하지는 않고, 특정 상황이 되었을 때 자동으로 호출되는 함수를 '콜백 함수' 라고 한다.

이벤트 프로그래밍

방법2 - DOM 노드에서 이벤트 등록

- 화면을 구성하는 각 태그에 이벤트를 등록하고 싶다면 HTML 문서의 태그에서 직접 이벤트 핸들러를 등록할 수 있습니다.

| | |
|--------|---|
| 이벤트 등록 | |
| 1 | <code><button onclick="myEventHandler()">click me</button></code> |

이벤트 프로그래밍


방법3 - 자바스크립트에서 onXXX 로 이벤트 등록

- onXXX 에서 XXX 는 각 이벤트를 식별하기 위한 이름입니다.
- 클릭 이벤트를 등록하겠다면 onclick, 브라우저 로딩 이벤트를 등록하겠다면 onload 를 사용합니다.

onXXX 로 이벤트 등록

```
1 window.onload = () => {  
2   console.log('HTML 문서 로딩이 완료 되었습니다.')  
3 }  
4 let button = document.querySelector('#button1')  
5 button.onclick = () => {
```

마우스 이벤트



| 이벤트 종류 | 설명 |
|------------|----------------------|
| click | 마우스 클릭 이벤트 |
| dblclick | 마우스 더블 클릭 이벤트 |
| mousedown | 마우스 버튼을 누르는 순간의 이벤트 |
| mouseup | 마우스 버튼을 떼는 순간 이벤트 |
| mousemove | 마우스 이동 이벤트 |
| mouseenter | 마우스 포인터가 들어오는 순간 이벤트 |
| mouseleave | 마우스 포인터가 나가는 순간 이벤트 |
| mouseover | 마우스 포인터가 들어오는 순간 이벤트 |
| mouseout | 마우스 포인터가 나가는 순간 이벤트 |

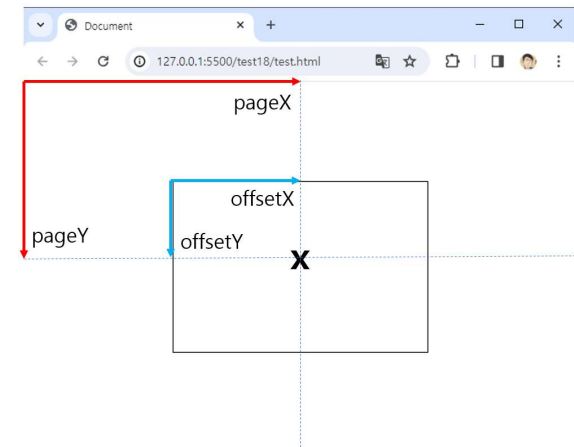
마우스 이벤트

이 객체를 활용하지 않을 것이라면 매개변수 등록 안 해도 된다.

- 모든 마우스 이벤트들은 이벤트 함수에 MouseEvent 타입의 객체가 전달되며 이 객체의 정보를 이용해 이벤트가 발생한 지점의 좌표값을 얻을 수 있습니다
- offsetX, offsetY는 이벤트가 발생한 DOM 노드 내에서의 좌표이며, pageX, pageY는 브라우저 창의 좌표입니다.

마우스 이벤트

```
1 let area = document.querySelector('#area')
2
3 area.addEventListener('mousemove', (e) => {
4   console.log(`offset(${e.offsetX}, ${e.offsetY}), page(${e.pageX},
5     ${e.pageY})`)
6 })
```



마우스 이벤트

- `mouseenter, mouseleave` 는 이벤트가 발생한 DOM 노드에서만 이벤트 처리가 가능하며 버블링이 발생하지 않습니다.
- `mouseover, mouseout` 은 버블링이 발생하여 이벤트가 발생한 DOM 노드 이외에 그 상위 노드에도 이벤트가 전파됩니다.



html이 중첩 구조로 작성되기 때문에 발생하는 문제

window 이벤트

- 브라우저에서 HTML 문서가 출력되는 브라우저 창을 지칭하는 객체는 Window 입니다.

매우 빈번하게 사용



| 이벤트 | 설명 |
|--------|----------------------------|
| copy | 브라우저에서 복사했을 때의 이벤트 |
| cut | 브라우저에서 잘라내기 했을 때의 이벤트 |
| paste | 브라우저에서 붙여넣기 했을 때의 이벤트 |
| load | 브라우저에 문서 로딩이 완료 되었을 때의 이벤트 |
| error | 브라우저에서 문서 로딩에 실패했을 때의 이벤트 |
| resize | 브라우저 창의 크기가 변경될 때의 이벤트 |

-

window 이벤트

- resize 이벤트는 브라우저 창의 크기가 변경되는 순간의 이벤트이며 이벤트 처리 함수에서 `innerWidth`, `innerHeight` 로 브라우저 창의 크기를 획득할 수 있습니다.

resize 이벤트

```
1  addEventListener('resize', () => {  
2    console.log(`width: ${innerWidth}, height: ${innerHeight}`)  
3  })
```

-

Form 관련 이벤트

- HTML 문서에서 사용자 데이터를 입력 받기 위해서 `<input>` `<textarea>` `<select>` 등의 태그를 이용하며 이 입력 태그를 묶어서 처리하기 위한 태그가 `<form>` 입니다.
- 사용자 입력과 관련된 이벤트는 `<form>` 태그에 거는 이벤트와 입력요소에 거는 이벤트로 구분해서 정리해 주어야 합니다.
- `<form>` 관련 이벤트는 submit 이벤트와 reset 이벤트가 있습니다.

| 이벤트 | 설명 |
|--------|-----------------------------|
| submit | form 데이터가 submit 되는 순간의 이벤트 |
| reset | form 데이터가 reset 되는 순간의 이벤트 |

form에서 하는

- 유효성 검증 (유저 입력 데이터는 비신뢰적, 서버로 넘어가기 전에 js가 해줘야 한다. 불필요한 네트워킹 발생 X)
- js -> ajax

Form 관련 이벤트

- <form> 내에는 <input type='text' /> 등 다양한 입력 태그들이 추가 될 수 있으며 이 입력 태그에 포커스가 추가되는 순간, 데이터가 변경되는 순간 등의 이벤트를 처리할 수 있습니다.

많이 쓴다

| 이벤트 | 설명 |
|--------|---------------------------|
| change | 입력 데이터가 변경되는 순간의 이벤트 |
| focus | 입력 요소가 포커스를 가지는 순간의 이벤트 |
| blur | 입력 요소가 포커스를 잃어버리는 순간의 이벤트 |



감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어
William Shakespeare