02

이벤트(Event) 처리

학습내용

01 이벤트(Event)의 개념

02 이벤트 리스너 등록

03 이벤트 객체와 속성

04 이벤트 위임

05 이벤트 처리 구현 방법

학습목표

- 이벤트(Event)의 개념을 이해하고 웹 페이지 상에서 발생하는 다양한 이벤트 유형을 설명할 수 있다.
- 이벤트 리스너를 적절히 등록하여 사용자 상호작용에 반응하는 웹 기능을 구현할 수 있다.
- 이벤트 객체의 구조와 주요 속성을 이해하고, 이를 활용하여 이벤트 관련 정보를 처리할 수 있다.
- 이벤트 위임(Event Delegation)의 개념과 필요성을 이해하고, 동적 요소 처리에 적용할 수 있다.
- 코드를 통해 이벤트에 대한 처리를 직접 구현할 수 있다.

01

이벤트(Event)의 개념

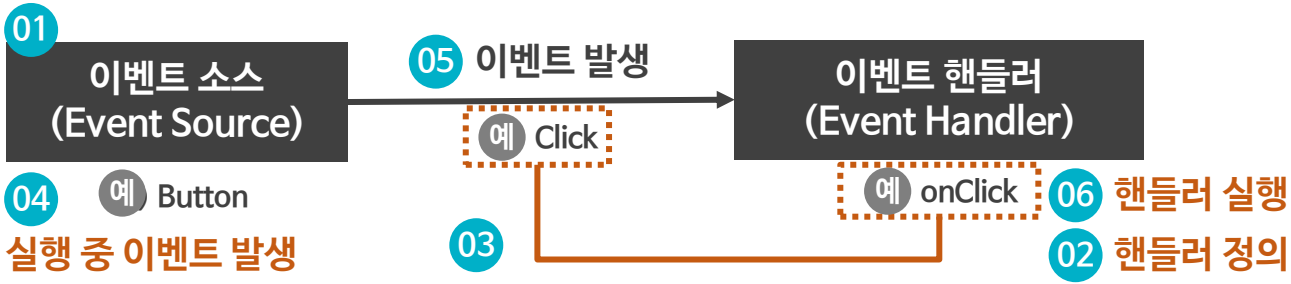
이벤트(Event)

웹 페이지에서 사용자나 브라우저가 발생시키는 사건

예 버튼 클릭, 키보드 누름, 마우스 드래그, 웹 페이지가 로드 완료되는 순간

참고 이벤트 처리를 하는 이유는 사용자의 행동에 반응하여
웹 페이지를 동적이고 상호작용 가능한 공간으로 만들기 위해서!

2) 이벤트 처리 흐름



- 리스너(Event Listener) :
  이벤트를 감지하기 위해 등록되는 함수 = Click
- 핸들러를 addEventListener()로 등록 :
  addEventListener("click", handlerFunc)

02

이벤트 리스너 등록

1) 등록 방법


```
//addEventListener()는 click 이벤트 발생 시 실행할 핸들러를
DOM 요소(element)에 등록
element.addEventListener("click", function () {
    console.log("클릭됨!"); //핸들러
});
```

방식	예시	특징
HTML 속성	〈button onclick="..."〉	비추천 (유지보수 어려움)
JS 직접 연결	addEventListener	권장 방식

2) 이벤트 리스너의 종류

2) 이벤트 리스너의 종류

이벤트 이름	설명	이벤트 핸들러 예시
click	요소를 클릭했을 때	element.addEventListener("click", () => { console.log("클릭"); });
dblclick	요소를 더블 클릭했을 때	element.addEventListener("dblclick", () => { alert("두 번 클릭!"); });
mouseover	마우스를 요소 위로 올렸을 때	element.addEventListener("mouseover", () => { console.log("오버"); });
mouseout	마우스가 요소 밖으로 나갔을 때	element.addEventListener("mouseout", () => { console.log("아웃"); });

2) 이벤트 리스너의 종류

이벤트 이름	설명	이벤트 핸들러 예시
mousedown	마우스 버튼을 누를 때	element.addEventListener(“mousedown”, () => { console.log("눌림"); });
mouseup	누른 버튼을 뗐을 때	element.addEventListener(“mouseup”, () => { console.log(“뗌”); });
mousemove	마우스를 움직일 때	element.addEventListener(“mousemove”, (e) => { console.log(e.clientX); });
contextmenu	마우스 우클릭 시 메뉴가 열릴 때	element.addEventListener(“contextmenu”, (e) => { e.preventDefault(); });

2) 이벤트 리스너의 종류

이벤트 이름	설명	이벤트 핸들러 예시
keydown	키를 누르는 순간	document.addEventListener(“keydown”, (e) => { console.log(e.key); });
keyup	키에서 손을 뗄 때	document.addEventListener(“keyup”, () => { console.log(“해제”); });
keypress	문자 키를 누를 때 (일부 환경)	document.addEventListener(“keypress”, (e) => { console.log(e.key); });

2) 이벤트 리스너의 종류

이벤트 이름	설명	이벤트 핸들러 예시
submit	폼이 제출될 때	form.addEventListener("submit", (e) => { e.preventDefault(); });
change	값이 변경되었을 때	select.addEventListener("change", () => { console.log("변경됨"); });
input	입력 중 실시간으로 발생	input.addEventListener("input", () => { console.log(input.value); });
focus	요소가 포커스를 받을 때	input.addEventListener("focus", () => { console.log("포커스"); });
blur	요소가 포커스를 잃을 때	input.addEventListener("blur", () => { console.log("포커스 해제"); });

2) 이벤트 리스너의 종류

이벤트 이름	설명	이벤트 핸들러 예시
load	문서나 리소스 로드 완료 시	window.addEventListener(“load”, () => { console.log(“로드 완료“); });
resize	창 크기 변경 시	window.addEventListener(“resize”, () => { console.log(“크기조정”); });
scroll	스크롤할 때	window.addEventListener(“scroll”, () => { console.log(“스크롤”); });

03

이벤트 객체와 속성

1) 이벤트 객체(event)

> 이벤트 발생 시 자동으로 전달되는 정보 객체

```
element.addEventListener("click", function (event) {
  console.log(event.target); // 이벤트 발생한 요소
  event.preventDefault();   // 기본 동작 막기
});
```

2) 이벤트 객체의 속성/메소드

event.target	이벤트 발생 대상 요소
event.type	이벤트 타입 정보
event.preventDefault()	기본 동작 차단 예 링크 클릭 시 페이지 이동을 막음
event.stopPropagation()	이벤트가 부모 요소로 전파 차단 (이벤트 버블링 차단)

04

이벤트 위임

이벤트 위임(Event Delegation)

자식 요소마다 이벤트를 거는 대신,
상위 요소에 한 번만 리스너 등록하여 효율적으로 처리

```
document.getElementById("list").addEventListener("click", function(e) {
  if (e.target.tagName === "LI") {  //element.tagName은 항상 대문자로 반환
    console.log("클릭한 항목:", e.target.textContent);
  }//
});
```

사용자가 실제로 클릭한 건 〈li〉 항목이지만, 이벤트 버블링 덕분에 이벤트가 부모 〈ul〉까지 전파) ➡ 리스트 전체(부모)에 이벤트를 위임해서, 실제 클릭된 자식 〈li〉만 처리하는 방식 ➡ 이벤트 위임

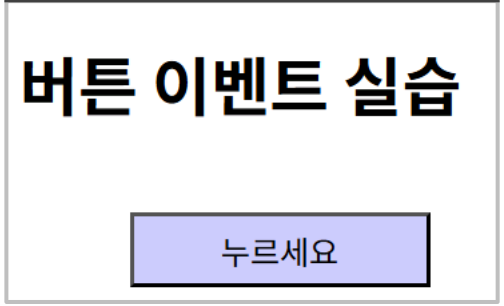
05

이벤트 처리 구현 방법

- 버튼을 누르면 메시지를 화면에 출력하는 이벤트 처리 구현하기
  - CSS

```
button {
  width:150px;
  margin:20px 55px;
  font-size:1em;
  padding:5px 10px;
  background-color:#ccccff;
}
```

실행 결과

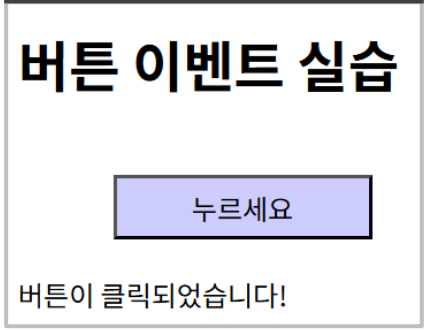


버튼을 누르면 메시지를 화면에 출력하는 이벤트 처리 구현하기

HTML

```
<button id="btn">누르세요</button>
<div id="output"></div>
```

실행 결과



JavaScript

```
document.getElementById("btn").addEventListener("click", function () {
  document.getElementById("output").textContent = "버튼이 클릭되었습니다!";
} );
```

세종사이버대학교

08주. 웹 기반 챗봇을 위한 UI 구현하기 : DOM과 이벤트 처리

03

LLM 기반의 챗봇 UI/UX 구현 실습