자바스크립트 이벤트

브라우저 이벤트는 웹 페이지에서 발생하는 다양한 사용자 상호작용(예: 클릭, 스크롤, 키보드 입력 등)을 나타냅니다. 웹 페이지에서는 이러한 이벤트를 감지하고, 필요한 작업을 수행하기 위한 코드(이벤트 리스너)를 작성할 수 있습니다.

- 마우스 이벤트: click, dblclick, mousedown, mouseup, mouseover, mouseout 등
- 키보드 이벤트: keydown, keyup, keypress
- 폼 이벤트: submit, change, focus, blur
- 문서 로딩 이벤트: DOMContentLoaded, load, unload, abort, error
- UI 이벤트: resize, scroll

자바스크립트 이벤트 종류

이벤트명	설명
클릭 이벤트 (click event)	사용자가 마우스로 요소를 클릭할 때 발생합니다. 주로 버튼, 링크, 이미지 등의 요소에 사용됩니다.
마우스 이벤트 (mouse event)	이벤트는 마우스의 움직임과 관련이 있습니다. 예를 들어, 요소 위에 마우스를 가져다 놓거나 요소를 누르거나 떼는 등의 동작에 대한 이벤트가 있습니다.
키보드 이벤트 (keyboard event)	사용자가 키보드를 사용할 때 발생합니다. 특정 키를 누르거나 떼는 등의 동작에 대한 이벤트가 있습니다.
포커스 이벤트 (focus event)	요소에 포커스가 설정되거나 해제될 때 발생합니다. 주로 입력 필드나 버튼 등의 요소에 사용됩니다.
양식 이벤트 (form event)	폼 제출(submit)이나 리셋(reset)과 같은 양식과 관련된 이벤트입니다.
터치 이벤트 (touch event)	터치 기반 장치에서 발생하는 이벤트로, 터치, 드래그, 스와이프 등의 제스처에 반응합니다.
스크롤 이벤트 (scroll event)	사용자가 페이지를 스크롤할 때 발생합니다. 주로 페이지 내에서 스크롤 위치를 추적하거나 특정 요소가 보이거나 사라지는 것을 감지하는 데 사용됩니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JavaScript Event Examples</title>
<style>
  body {display: flex; flex-direction: column; gap: 24px;}
  div {margin: 8px 0; padding: 8px;border: 1px solid #ccc;border-radius:
4px;}
</style>
```

```
</head>
<body>

<div>
  <h3>Click Event</h3>
  <button id="myButton">Click Me</button>
</div>

<div>
  <h3>Mouse Event</h3>
  <div id="myDiv" style="width: 200px; height: 40px; background-color:
lightblue;">Mouse over me</div>
</div>

<div>
  <h3>Keyboard Event</h3>
  <input type="text" id="myInput" placeholder="Press Enter">
</div>

<div>
  <h3>Focus Event</h3>
  <input type="text" id="focusInput" placeholder="Focus here">
</div>

<div>
  <h3>Form Event</h3>
  <form id="myForm">
    <input type="text" name="username" placeholder="Username">
    <button type="submit">Submit</button>
  </form>
</div>

<div>
  <h3>Keyboard: keydown and keyup</h3>
  <input type="text" id="keyboardInput" placeholder="Type something">
</div>

<div>
  <h3>Touch Event</h3>
  <div id="myTouchArea" style="width: 200px; height: 40px; background-
color: lightblue;">Touch Me</div>
</div>

<div>
  <h3>Mouse & Pointer Events</h3>
  <div id="mousePointerDiv" style="width: 200px; height: 100px;
background-color: lightblue;">Mouse over or touch me</div>
</div>

<div>
  <h3>JavaScript Animations</h3>
  <div id="animationDiv" style="width: 100px; height: 100px; background-
color: lightblue; position: relative;"></div>
</div>
```

```
<div>
  <h3 id="scrollHeader" data-px="">Scroll Event</h3>
  <p id="scrollValue">Scrolled: 0px</p>
  <div id="myScrollDiv" style="width: 200px; height: 300px; background-
color: lightblue; overflow: auto;">
    <div style="height: 600px; background-color: #eee;">
      Scroll me
    </div>
  </div>
</div>

<script>
  // Click Event
  document.getElementById("myButton").addEventListener("click", function()
{
    console.log("Button clicked!");
  });

  // Mouse Event
  document.getElementById("myDiv").addEventListener("mouseover",
function() {
    this.style.backgroundColor = "lightgreen";
    console.log("lightgreen");
  });

  document.getElementById("myDiv").addEventListener("mouseout", function()
{
    this.style.backgroundColor = "lightblue";
    console.log("lightblue");
  });

  // Keyboard Event
  document.getElementById("myInput").addEventListener("keydown",
function(event) {
    if (event.key === "Enter") {
      console.log("Enter key pressed!");
    }
  });

  // Focus Event
  document.getElementById("focusInput").addEventListener("focus",
function() {
    this.style.backgroundColor = "#0000ff";
    this.style.color = "#fff";
  });

  document.getElementById("focusInput").addEventListener("blur",
function() {
    this.style.backgroundColor = "white";
  });

  // Form Event
```

```
  document.getElementById("myForm").addEventListener("submit",
function(event) {
    event.preventDefault();
    var username = this.elements["username"].value;
    console.log("Form submitted by: " + username);
  });

  // Keyboard: keydown and keyup
  document.getElementById("keyboardInput").addEventListener("keydown",
function(event) {
    console.log("Key down: " + event.key);
  });

  document.getElementById("keyboardInput").addEventListener("keyup",
function(event) {
    console.log("Key up: " + event.key);
  });

  // Touch Event
  document.getElementById("myTouchArea").addEventListener("touchstart",
function(event) {
    console.log("Touch started!");
  });

  document.getElementById("myTouchArea").addEventListener("touchend",
function(event) {
    console.log("Touch ended!");
  });

  // Mouse & Pointer Events

document.getElementById("mousePointerDiv").addEventListener("pointerover",
function() {
    this.style.backgroundColor = "lightgreen";
    console.log("Pointer over");
  });


document.getElementById("mousePointerDiv").addEventListener("pointerout",
function() {
    this.style.backgroundColor = "lightblue";
    console.log("Pointer out");
  });

  // JavaScript Animations
  let animationDiv = document.getElementById("animationDiv");
  let position = 0;
  let direction = 1;
  function animate() {
    position += direction * 2;
    animationDiv.style.left = position + "px";
    if (position >= 200 || position <= 0) {
      direction *= -1;
    }
```

```
    requestAnimationFrame(animate);
  }
  animate();

  // Scroll Event inside myScrollDiv
  document.getElementById("myScrollDiv").addEventListener("scroll",
function() {
    let scrolled = this.scrollTop;
    console.log("Scrolled: " + scrolled + "px");

    // Assign the scroll value to data-px attribute
    document.getElementById("scrollHeader").setAttribute("data-px",
scrolled + "px");
    // Update the text to show scrolled value
    document.getElementById("scrollValue").textContent = "Scrolled: " +
scrolled + "px";
  });
</script>

</body>
</html>
```

- 예제에 사용된 이벤트

이벤트 종류	요소 ID	이벤트 설명
Click Event	myButton	버튼 클릭 시 발생하는 이벤트
Mouse Event	myDiv	마우스를 요소 위에 올렸을 때 (`mouseover`)와 뺐을 때 (`mouseout`) 발생하는 이벤트
Keyboard Event	myInput	키보드 키를 눌렀을 때 (`keydown`) 발생하는 이벤트, 여기서는 Enter 키에 특별히 반응
Focus Event	focusInput	입력 필드에 포커스가 가거나 사라졌을 때 (`focus`, `blur`) 발생하는 이벤트
Form Event	myForm	폼 제출 (`submit`) 이벤트, 기본 동작을 방지하고 사용자 이름을 로그에 출력
Keyboard: keydown and keyup	keyboardInput	키보드 키를 눌렀을 때 (`keydown`)와 뗐을 때 (`keyup`) 발생하는 이벤트
Touch Event	myTouchArea	터치 시작 (`touchstart`)과 종료 (`touchend`) 시 발생하는 이벤트
Mouse & Pointer Events	mousePointerDiv	포인터(마우스 포인터 포함)가 요소 위에 올랐을 때 (`pointerover`)와 뺐을 때 (`pointerout`) 발생하는 이벤트
JavaScript Animations	animationDiv	CSS 위치 속성을 조작하여 애니메이션을 만드는 코드, 이벤트는 직접적으로 사용되지 않으나 `requestAnimationFrame`을 이용
Scroll Event	myScrollDiv	스크롤 이벤트, 스크롤 시 발생하며 스크롤된 거리를 로그에 출력

자바스크립트 이벤트 유형

이벤트 처리에서 중요한 개념인 버블링(bubbling)과 캡처링(capturing)은 이벤트가 DOM 트리를 통해 어떻게 전파되는지를 설명합니다.

구분	이벤트 캡처링 (Capturing)	이벤트 버블링 (Bubbling)
설명	이벤트가 상위 요소에서 하위 요소로 전파되는 과정입니다. 캡처링은 이벤트가 실제로 발생한 요소에 도달하기 전에 상위 요소에서 이벤트를 처리할 수 있는 기회를 제공합니다.	이벤트가 실제로 발생한 요소에서 시작하여 상위 요소로 전파되는 과정입니다. 대부분의 이벤트는 버블링 단계에서 처리되며, 이를 통해 여러 요소에 대한 이벤트를 하나의 상위 요소에서 관리할 수 있습니다.
사용하는 경우	상위 요소에서 이벤트를 먼저 처리해야 할 때. 하위 요소로 이벤트가 도달하기 전에 미리 조치를 취해야 할 때.	여러 자식 요소에 공통으로 적용되는 이벤트 리스너를 상위 요소에 한 번만 설정하고자 할 때 (이벤트 위임). 하위 요소에서 발생한 이벤트에 대한 정보를 상위 요소에서 처리하고자 할 때.
장점 및 유용성	이벤트 처리 순서를 정밀하게 제어할 수 있어, 미세한 이벤트 관리 가능.	코드 효율성: 이벤트 위임을 통해 중복 리스너를 줄임. 동적인 요소 처리: 나중에 추가되는 요소들에 대해 리스너를 설정할 필요 없음.
단점 및 주의사항	복잡한 이벤트 흐름을 이해하고 관리해야 하는 필요성.	이벤트의 불필요한 전파를 막기 위해 `event.stopPropagation()` 사용 필요.

```
<!DOCTYPE html>
<html>
<head>
  <title>Event Capturing and Bubbling</title>
  <style>
    div {width: 100px;height: 100px;background: #ddd;display: flex;flex-
direction: column;justify-content: center;align-items: center;}
    div > div {width: 60px; height: 60px; background-color: #333; display:
flex;justify-content: center;align-items: center;}
  </style>
</head>
```

```
<body>
  <div id="parent">
    <button>부모</button>
    <div id="child">
      <button>자식</button>
    </div>
  </div>

  <script>
    // 부모 요소에 대한 참조를 가져옵니다.
    let parentDiv = document.getElementById('parent');

    // 자식 요소에 대한 참조를 가져옵니다.
    let childDiv = document.getElementById('child');

    // 캡처링 단계에서 작동하는 이벤트 리스너를 부모에 추가합니다.
    // 이 리스너는 이벤트가 상위 요소에서 하위 요소로 전파될 때(캡처링) 활성화됩니다.
    parentDiv.addEventListener('click', function() {
        console.log('부모 (캡처링 단계)');
    }, true); // 세 번째 매개변수 true는 캡처링을 활성화합니다.

    // 버블링 단계에서 작동하는 이벤트 리스너를 부모에 추가합니다.
    // 이 리스너는 이벤트가 하위 요소에서 상위 요소로 전파될 때(버블링) 활성화됩니다.
    parentDiv.addEventListener('click', function() {
        console.log('부모 (버블링 단계)');
    }, false); // 세 번째 매개변수 false(기본값)는 버블링을 활성화합니다.

    // 자식에 클릭 이벤트 리스너를 추가합니다.
    // 이 리스너는 자식가 클릭될 때 실행됩니다.
    childDiv.addEventListener('click', function(event) {
        console.log('자식');
        // 이벤트 버블링을 중지하려면 아래의 주석을 해제하세요.
        event.stopPropagation(); // 이벤트 버블링 중지
    }, false); // 여기서도 버블링을 활성화하고 있습니다.
  </script>
</body>
</html>
```

커스텀 이벤트 디스패치

커스텀 이벤트(custom event)는 개발자가 정의하고 생성하는 사용자 정의 이벤트입니다. 이를 통해 특정 조건에서 이벤트를 발생시키거나, 애플리케이션에 특정한 상호작용을 구현할 수 있습니다.

단계	설명	메소드/속성

단계	설명	메소드/속성
커스텀 이벤트 생성	`CustomEvent` 생성자를 사용하여 새로운 커스텀 이벤트를 만듭니다. 이 생성자는 이벤트의 이름과, 선택적으로 이벤트의 상세 정보를 포함하는 `detail` 속성을 설정할 수 있는 옵션 객체를 받습니다.	`new CustomEvent(eventName, options)`
이벤트 디스패치	`dispatchEvent` 메소드를 사용하여 커스텀 이벤트를 특정 DOM 요소에 디스패치(전달)합니다. 이 메소드는 커스텀 이벤트 객체를 인자로 받으며, 이 이벤트는 해당 요소에 연결된 이벤트 리스너에 의해 캡처되고 처리됩니다.	`element.dispatchEvent(event)`
이벤트 리스너 추가	`addEventListener` 메소드를 사용하여 DOM 요소에 이벤트 리스너를 추가합니다. 이 리스너는 커스텀 이벤트가 디스패치될 때 호출됩니다. 이벤트 리스너는 이벤트의 상세 정보에 접근할 수 있으며, 이를 기반으로 특정 로직을 수행할 수 있습니다.	`element.addEventListener(eventName, handler)`
이벤트 흐름	커스텀 이벤트도 표준 DOM 이벤트처럼 버블링(bubbling) 또는 캡처(capturing) 단계를 통해 전파될 수 있습니다. 이벤트 생성 시 `bubbles`와 `cancelable` 옵션을 설정하여 이러한 동작을 제어할 수 있습니다.	`new CustomEvent(eventName, { bubbles: true, cancelable: true })`

> "디스패치(dispatch)"란 특정한 작업이나 명령을 수행하도록 요청하거나 전달하는 것을 의미합니다. 컴퓨터 과학 분야에서 이 용어는 주로 이벤트 기반 시스템에서 사용됩니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>Custom Event Example</title>
  <style>
    .box {width: 100px;height: 60px;color: white;background-color:
#333;border-radius: 8px;display: flex;justify-content: center;align-items:
center;font-size: 18px;cursor: pointer;}
  </style>
</head>
<body>
```

```
  <div class="container">
    <div class="box" id="myBox">Click me!</div>
  </div>
  <script>
    // 커스텀 이벤트 생성
    let myEvent = new CustomEvent("myEvent", { // "myEvent"라는 이름의 커스텀 이벤트를 생성합니다.

      detail: { message: "소금빵!" } // 커스텀 이벤트에는 "소금빵!"이라는 메시지를 포함하는 detail 속성이 있습니다.
    });

    // 이벤트 리스너 추가
    document.getElementById("myBox").addEventListener("click", function() { // "myBox" 요소를 클릭했을 때 실행할 함수를 등록합니다.
      // 클릭시 커스텀 이벤트 디스패치
      this.dispatchEvent(myEvent); // 현재 요소에서 커스텀 이벤트를 디스패치(발생)합니다.
    });

    // 커스텀 이벤트 핸들러
    document.getElementById("myBox").addEventListener("myEvent", function(e) { // "myEvent" 커스텀 이벤트가 발생했을 때 실행할 함수를 등록합니다.
      console.log("맛있는 빵은 : " + e.detail.message); // 발생한 커스텀 이벤트의 detail 속성에서 메시지를 가져와 출력합니다.
    });
  </script>
</body>
</html>
```