이벤트 객체

`addEventListener`를 사용하여 이벤트 리스너를 등록할 때, 이벤트가 발생하면 해당 이벤트에 관한 이벤트 객체(Event Object)가 콜백 함수로 전달됩니다. 이 객체는 이벤트의 세부사항(예: 마우스 좌표, 누른 버튼, 키 값 등)을 포함하고 있으며, 이벤트의 종류에 따라 그 속성과 메서드가 달라집니다. 아래에서는 마우스 관련 이벤트 객체와 키보드 관련 이벤트 객체에 대해 자세히 살펴보겠습니다.

1. 마우스 이벤트 객체 (MouseEvent)

마우스 이벤트 객체는 `MouseEvent` 타입이며, 사용자가 마우스와 상호작용할 때 발생하는 이벤트(예: `click`, `dblclick`, `mousedown`, `mouseup`, `mouseover`, `mouseout`, `mousemove` 등)에 대한 정보를 담고 있습니다.

주요 속성

- 좌표 관련

  - `clientX` / `clientY`: 브라우저 뷰포트(화면) 내에서 마우스 커서의 x, y 좌표입니다.

  - `pageX` / `pageY`: 문서 전체를 기준으로 한 마우스 좌표입니다. 스크롤된 경우에도 실제 위치를 알 수 있습니다.

  - `screenX` / `screenY`: 사용자의 모니터(화면) 기준 좌표입니다.

- 버튼 관련

  - `button`: 어떤 마우스 버튼이 눌렸는지를 나타냅니다. 일반적으로 왼쪽 버튼은 `0`, 가운데 버튼은 `1`, 오른쪽 버튼은 `2`입니다.

  - `buttons`: 현재 눌려진 모든 버튼의 상태를 비트 마스크 형태로 제공합니다. 여러 버튼을 동시에 눌렀을 때 유용합니다.

- 수정 키 관련

  - `altKey`, `ctrlKey`, `shiftKey`, `metaKey`: 각각 Alt, Ctrl, Shift, Command/Windows 키가 함께 눌렸는지 여부를 `true` 또는 `false`로 제공합니다.

- 관련 타겟

  - `relatedTarget`: `mouseover`나 `mouseout` 이벤트에서 사용됩니다. 이벤트가 이동하기 전/후의 요소를 나타냅니다.

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>Mouse Event Example</title>
```

```
</head>

<body>
    <button id="myButton">클릭해 주세요!</button>

    <script>
        const btn = document.getElementById('myButton');

        btn.addEventListener('click', function (event) {
            console.log("마우스 클릭 이벤트 발생!");
            console.log("뷰포트 내 좌표 (clientX, clientY):", event.clientX, event.clientY);
            console.log("버튼 번호 (button):", event.button);

            // 수정 키 여부 체크
            if (event.altKey) {
                console.log("Alt 키가 함께 눌렸습니다.");
            }
            if (event.ctrlKey) {
                console.log("Ctrl 키가 함께 눌렸습니다.");
            }
            if (event.shiftKey) {
                console.log("Shift 키가 함께 눌렸습니다.");
            }
            if (event.metaKey) {
                console.log("Meta 키(Windows/Command)가 함께 눌렸습니다.");
            }
        });
    </script>
</body>

</html>
```

```
마우스 클릭 이벤트 발생!
뷰포트 내 좌표 (clientX, clientY): 73 25
버튼 번호 (button): 0
마우스 클릭 이벤트 발생!
뷰포트 내 좌표 (clientX, clientY): 34 21
버튼 번호 (button): 0
```

2. 키보드 이벤트 객체 (KeyboardEvent)

키보드 이벤트 객체는 `KeyboardEvent` 타입이며, 사용자가 키보드를 누르거나 뗄 때 발생하는 이벤트(예: `keydown`, `keyup`, `keypress` 등)에 관한 정보를 제공합니다.

주요 속성

- 키 관련

  - `key`: 눌린 키의 값을 문자열로 전달합니다. 예를 들어, "Enter", "Escape", "a" 등 실제 키의 이름 또는 문자 값을 반환합니다.

  - `code`: 키보드의 물리적 위치를 기반으로 한 키 코드를 나타냅니다. 예를 들어, `KeyA`, `Digit1` 등이 있습니다.

  - `keyCode` 및 `which`: 과거에 사용되던 숫자 코드로, 현재는 `key`와 `code`를 사용하는 것이 권장됩니다.

- 수정 키 관련

  - `altKey`, `ctrlKey`, `shiftKey`, `metaKey`: 마우스 이벤트와 동일하게 각각 Alt, Ctrl, Shift, Command/Windows 키가 함께 눌렸는지 여부를 나타냅니다.

- 반복 여부

  - `repeat`: 키가 장시간 눌린 경우 자동으로 반복 이벤트가 발생하는데, 이 값을 통해 해당 키가 계속 눌리고 있는지 여부를 확인할 수 있습니다.

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>Keyboard Event Example</title>
</head>

<body>
    <input id="myInput" type="text" placeholder="여기에 입력하세요...">

    <script>
        const input = document.getElementById('myInput');

        input.addEventListener('keydown', function (event) {
            console.log("키다운 이벤트 발생!");
            console.log("눌린 키 (key):", event.key);
            console.log("키 코드 (code):", event.code);
```

```
            // 수정 키 여부 체크
            if (event.ctrlKey) {
                console.log("Ctrl 키가 함께 눌렸습니다.");
            }
            if (event.shiftKey) {
                console.log("Shift 키가 함께 눌렸습니다.");
            }
            if (event.altKey) {
                console.log("Alt 키가 함께 눌렸습니다.");
            }
            if (event.metaKey) {
                console.log("Meta 키(Windows/Command)가 함께 눌렸습니다.");
            }

            // 반복 입력 여부 확인
            if (event.repeat) {
                console.log("키가 장시간 눌려 반복 이벤트가 발생 중입니다.");
            }
        });
    </script>
</body>

</html>
```

```
키다운 이벤트 발생!
눌린 키 (key): Alt
키 코드 (code): AltLeft
Alt 키가 함께 눌렸습니다.
키다운 이벤트 발생!
눌린 키 (key): a
키 코드 (code): KeyA
Alt 키가 함께 눌렸습니다.
키다운 이벤트 발생!
눌린 키 (key): a
키 코드 (code): KeyA
Alt 키가 함께 눌렸습니다.
```

연습문제

문제 1: 클릭 시 좌표 확인하기

목표:

웹 페이지에 버튼을 만들고, 해당 버튼을 클릭했을 때 마우스 이벤트 객체의 `clientX`와 `clientY` 값을 이용하여 클릭한 위치의 좌표를 `alert` 창으로 보여주기.

세부 요구사항 및 단계:

1. HTML 요소 구성:

- `<button>` 요소에 `id="myButton"` 를 부여하여 버튼을 생성합니다.
- 버튼 텍스트는 예를 들어 "클릭해 주세요!"로 작성합니다.

2. JavaScript 이벤트 등록:

- JS 코드에서 `document.getElementById("myButton")`를 사용하여 버튼 요소를 가져옵니다.
- `addEventListener`를 이용해 `click` 이벤트 리스너를 등록하세요.

3. 이벤트 객체 활용:

- 이벤트 핸들러 내부에서 전달되는 이벤트 객체(`event`)의 `clientX`와 `clientY` 속성 값을 사용합니다.
- 두 값을 조합하여 `"클릭 위치: (x, y)"` 형식의 문자열을 만들고, `alert` 창으로 출력합니다.

4. 테스트:

- 브라우저에서 버튼을 클릭하면서 기대한 좌표 값이 정확하게 출력되는지 확인합니다.

문제 2: 수정키와 함께 클릭 감지하기

목표:

버튼 클릭 이벤트 처리 시, 이벤트 객체의 `shiftKey`, `ctrlKey`, `altKey`, `metaKey` 속성을 이용하여 어떤 수정키(Shift, Ctrl, Alt, Meta)가 눌렸는지 콘솔에 출력하기.

세부 요구사항 및 단계:

1. HTML 요소:

- 하나의 `<button id="modifierButton">`를 생성합니다.

2. 이벤트핸들러 등록:

- 버튼에 대해 `click` 이벤트 리스너를 등록합니다.

3. 수정키 감지:

- 이벤트 객체의 `shiftKey`, `ctrlKey`, `altKey`, `metaKey` 속성을 조건문으로 확인합니다.
- 각 수정키가 활성화되어 있다면, `"Shift 키 눌림"`, `"Ctrl 키 눌림"` 등과 같이 콘솔에 출력하세요.

4. 실행 및 검증:

- 버튼을 클릭할 때, 수정키를 함께 누른 채로 테스트해보고 콘솔 메시지를 확인합니다.

문제 3: 마우스 이동에 따른 실시간 좌표 추적

목표:

문서 전체에서 `mousemove` 이벤트를 활용하여, 마우스 커서가 움직일 때마다 그 위치를 웹 페이지의 `<div>` 요소 내에 표시하기.

세부 요구사항 및 단계:

1. HTML 구성:

- `<div id="coordinateDisplay">`를 만들고, 해당 영역에 마우스의 좌표를 텍스트 형태로 출력할 예정입니다.
- 스타일을 적용하여 눈에 띄게 하고, 위치를 고정할 수 있도록 합니다.

2. 이벤트 리스너 등록:

- `document`에 `mousemove` 이벤트 리스너를 등록합니다.

3. 동적 업데이트:

- 이벤트 객체의 `clientX`와 `clientY` 값을 읽어내고, `<div>`의 `innerText`나 `textContent`를 해당 값으로 업데이트합니다.

- 예시 텍스트 형식: `"마우스 위치: (x, y)"`

4. 테스트:

- 브라우저에서 마우스를 움직이며, `<div>` 내에 좌표가 실시간으로 변경되는지 확인합니다.

문제 4: 키보드 입력 감지 및 Enter 키 처리

목표:

텍스트 입력 필드에 사용자가 키를 누를 때마다, 눌린 키의 `key`와 `code` 정보를 콘솔에 출력하고, 만약 누른 키가 Enter인 경우 별도의 메시지를 출력하기.

세부 요구사항 및 단계:

1. HTML 요소 구성:

- `<input id="myInput" type="text" placeholder="여기에 입력하세요...">` 요소를 생성합니다.

2. 이벤트 리스너 등록:

- 입력 필드에 `keydown` 이벤트 리스너를 추가합니다.

3. 키 값 출력:

- 이벤트 객체에서 `event.key`와 `event.code` 값을 콘솔에 로그합니다.

- 조건문을 사용하여, 만약 `event.key`가 `"Enter"`라면 `"Enter 키가 눌렸습니다"`라는 메시지를 추가로 출력합니다.

4. 실행 및 검증:

- 입력 필드에 키 입력 시 콘솔 로그를 확인하고, Enter 입력 시 구별되는 메시지가 나오는지 확인합니다.

문제 5: 여러 버튼 클릭 시 마우스 버튼 번호 확인하기

목표:

페이지에 여러 개의 버튼을 만들고, 각각의 버튼을 클릭할 때 어떤 마우스 버튼(왼쪽, 가운데, 오른쪽)이 눌렸는지를 `event.button` 값을 이용하여 콘솔에 출력하기.

세부 요구사항 및 단계:

1. HTML 요소 구성:

- 3개 이상의 `<button>` 요소를 만들고, 각 버튼에 고유의 텍스트(예: "버튼 1", "버튼 2", "버튼 3")를 부여합니다.

- 각 버튼에 동일한 클래스나 `id`를 부여하거나, 개별적으로 이벤트 리스너를 등록할 수 있습니다.

2. 이벤트 리스너 등록:

- 각 버튼에 대해 `click` 이벤트 리스너를 등록합니다.

- 이벤트 핸들러 내에서 `event.button` 값을 가져옵니다.

  - 일반적으로 왼쪽 클릭이면 0, 가운데 클릭이면 1, 오른쪽 클릭이면 2가 출력됩니다.

  - (실습환경에 따라 우클릭 등의 동작이 브라우저에 의해 차단될 수 있으므로, 필요 시 브라우저 설정을 확인합니다.)

3. 콘솔 출력:

- 각 버튼 클릭 시 `"클릭된 버튼 번호: X"` (X는 0, 1, 2 등)를 콘솔에 출력합니다.

4. 테스트:

- 다양한 마우스 버튼(왼쪽/오른쪽/가운데)을 사용하여 각 버튼을 클릭하고 콘솔 로그를 확인합니다.

문제 6: 뷰포트 좌표와 문서 좌표의 차이 이해하기

목표:

스크롤 가능한 긴 페이지를 만들어, 버튼 클릭 시 `clientX/clientY` (뷰포트 기준 좌표)와 `pageX/pageY`

(문서 전체 기준 좌표) 값을 비교하여 출력하기.

세부 요구사항 및 단계:

1. HTML 페이지 구성:
    - 페이지에 긴 내용을 채워 스크롤이 가능하도록 합니다. (예: 여러 개의 `<p>` 요소 삽입)
    - 상단에 `<button id="coordButton">좌표 확인하기</button>`를 배치합니다.

2. 이벤트 핸들러 등록:
    - 버튼에 `click` 이벤트 리스너를 추가합니다.

3. 좌표값 읽기:
    - 이벤트 객체에서 `clientX`, `clientY`, `pageX`, `pageY` 값을 읽어옵니다.
    - 두 좌표의 차이점을 알기 쉽게 문자열을 생성합니다.
        - 예: `"뷰포트 좌표: (X, Y), 문서 좌표: (X, Y)"`
    - `alert`나 `console.log`를 통해 출력합니다.

4. 테스트:
    - 페이지를 스크롤한 후 버튼을 클릭하여 두 종류의 좌표 값이 어떻게 다른지 확인합니다.

문제 7: Ctrl+클릭 시 기본 동작 차단하기

목표:

링크(`<a>` 태그)에 대해, 사용자가 Ctrl 키를 누른 상태로 클릭하면 기본적으로 페이지가 이동하는 동작을 막고, 대신 콘솔에 관련 메세지를 출력하기.

세부 요구사항 및 단계:

1. HTML 구성:
    - `<a id="myLink" href="https://www.google.com">구글로 이동</a>` 처럼 링크를 생성합니다.

2. 이벤트 핸들러 등록:

- 해당 `<a>` 태그에 `click` 이벤트 리스너를 등록합니다.

3. 이벤트 핸들러 내용 작성:

- 이벤트 객체에서 `ctrlKey` 값을 확인합니다.

- 만약 `ctrlKey`가 `true`라면, `event.preventDefault()`를 호출하여 링크의 기본 이동 동작을 차단합니다.

- 콘솔에 `"Ctrl+클릭 감지됨: 이동하지 않습니다"`라는 메시지를 출력합니다.

- 그렇지 않은 경우는 기본 동작을 유지합니다.

4. 실행 및 검증:

- Ctrl 키를 누른 상태와 누르지 않은 상태에서 링크를 클릭하여 서로 다른 동작이 실행되는지 확인합니다.

문제 8: 더블 클릭 시 배경색 변경하기

목표:

웹 페이지에 하나의 `<div>` 요소를 만들고, 해당 요소에 더블 클릭(`dblclick`) 이벤트가 발생하면 배경색을 랜덤하게 변경하기.

세부 요구사항 및 단계:

1. HTML 요소 구성:

- `<div id="colorBox">더블 클릭해 배경색 변경</div>` 요소를 생성합니다.

- 스타일(CSS)을 추가하여 크기, 경계(border), 중앙 정렬 등 시각적으로 확인하기 쉽게 구성합니다.

2. 이벤트 리스너 등록:

- 해당 `<div>`에 `dblclick` 이벤트 리스너를 등록합니다.

3. 배경색 랜덤 변경:

- JavaScript로 랜덤 색상을 생성하는 함수를 작성합니다.

  - 예를 들어, `Math.floor(Math.random() * 256)`를 이용해 R, G, B 값 생성 후 `"rgb(r, g, b)"` 형태의 문자열 만들기.

- 이벤트 핸들러 내에서 `<div>`의 `style.backgroundColor` 속성을 새롭게 생성한 색상 문자열로 업데이트합니다.

4. 실행 및 검증:

- `<div>`를 더블 클릭하여 배경색이 변경되는지 확인합니다.

문제 9: 이벤트 전파 차단 연습

목표:

부모 요소와 자식 요소에 각각 클릭 이벤트 리스너를 등록합니다. 자식 요소의 이벤트 핸들러에서 `event.stopPropagation()`을 호출하여 부모 요소로 이벤트가 전파되는 것을 막고, 각 이벤트가 발생할 때 콘솔에 별도의 메시지를 출력하기.

세부 요구사항 및 단계:

1. HTML 구조 구성:

- `<div id="parent">` 요소 안에 `<div id="child">` 요소를 배치합니다.

- 두 요소에는 서로 다른 배경색이나 테두리 스타일을 주어 구분할 수 있게 합니다.

2. 이벤트 리스너 등록:

- 부모 요소에 클릭 이벤트 리스너를 등록하고, `"부모 클릭됨"` 메시지를 콘솔에 출력하도록 합니다.

- 자식 요소에도 클릭 이벤트 리스너를 등록하는데, 이 핸들러 내에서 `event.stopPropagation()`을 호출한 후 `"자식 클릭됨 (전파 중단)"` 메시지를 출력합니다.

3. 실행 및 검증:

- 자식 요소를 클릭했을 때 부모 요소의 이벤트가 실행되지 않는지, 반대로 부모를 클릭하면 부모 이벤트만 발생하는지 확인합니다.

문제 10: 키보드 장기 누름(repeat) 감지 및 표시하기

목표:

입력 필드 또는 전체 문서에서 `keydown` 이벤트를 감지하고, 이벤트 객체의 `repeat` 속성을 확인하여 키가 장시간 눌린 경우 콘솔에 반복 여부를 표시하기.

세부 요구사항 및 단계:

1. HTML 요소 구성:

    - `<input id="repeatInput" type="text" placeholder="키를 길게 눌러보세요...">` 요소를 생성합니다.

    - 또는 `document` 전체에 이벤트를 적용할 수도 있습니다.

2. 이벤트 리스너 등록:

    - 입력 필드 (또는 문서)에 `keydown` 이벤트 리스너를 추가합니다.

3. 반복 여부 감지:

    - 이벤트 객체의 `repeat` 속성을 검사하여, 만약 `true`이면 `"키가 반복 입력되고 있음"` 메시지를 콘솔에 출력합니다.

    - 그렇지 않으면 정상적인 키 입력 정보를 출력할 수 있도록 합니다.

4. 실행 및 테스트:

    - 해당 입력 필드에서 하나의 키를 길게 눌러, 정상 입력과 반복 입력 메시지의 차이를 확인합니다.

정답소스

문제 1: 클릭 시 좌표 확인하기

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 1: 클릭 좌표 확인하기 - 정답 소스</title>
</head>

<body>
    <button id="myButton">클릭해 주세요!</button>

    <script>
        // 버튼 요소 선택 후 클릭 이벤트 등록
        const btn = document.getElementById('myButton');
        btn.addEventListener('click', function (event) {
            const x = event.clientX;
            const y = event.clientY;
            alert("클릭 위치: (" + x + ", " + y + ")");
        });
    </script>
</body>

</html>
```

문제 2: 수정키와 함께 클릭 감지하기

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 2: 수정키 감지하기 - 정답 소스</title>
</head>

<body>
    <button id="modifierButton">수정키와 함께 클릭</button>

    <script>
        const btn = document.getElementById('modifierButton');
        btn.addEventListener('click', function (event) {
            // 각 수정키의 활성 여부를 확인하여 콘솔에 출력
            if (event.shiftKey) {
                console.log("Shift 키 눌림");
            }
            if (event.ctrlKey) {
                console.log("Ctrl 키 눌림");
            }
            if (event.altKey) {
                console.log("Alt 키 눌림");
```

```
            }
            if (event.metaKey) {
                console.log("Meta 키(Windows/Command) 눌림");
            }
            // 어떤 수정키도 누르지 않은 경우
            if (!event.shiftKey && !event.ctrlKey && !event.altKey && !event.metaKey) {
                console.log("수정키 없이 클릭");
            }
        });
    </script>
</body>

</html>
```

문제 3: 마우스 이동에 따른 실시간 좌표 추적

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 3: 마우스 실시간 좌표 추적 - 정답 소스</title>
    <style>
        /* 좌표를 표시할 div 스타일 */
        #coordinateDisplay {
            border: 1px solid #ccc;
            padding: 10px;
            width: 250px;
            position: fixed;
            top: 10px;
            right: 10px;
            background-color: #f9f9f9;
        }
    </style>
</head>

<body>
    <div id="coordinateDisplay">마우스 위치: ( , )</div>

    <script>
        const coordDiv = document.getElementById('coordinateDisplay');
        // 문서 전체에서 마우스 이동 이벤트 감지 후 좌표 업데이트
        document.addEventListener('mousemove', function (event) {
            const x = event.clientX;
            const y = event.clientY;
            coordDiv.textContent = "마우스 위치: (" + x + ", " + y + ")";
        });
    </script>
</body>

</html>
```

문제 4: 키보드 입력 감지 및 Enter 키 처리

```html
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 4: 키보드 입력 감지하기 - 정답 소스</title>
</head>

<body>
    <input id="myInput" type="text" placeholder="여기에 입력하세요...">

    <script>
        const inputField = document.getElementById('myInput');
        // 입력 필드에 키 다운 이벤트 등록
        inputField.addEventListener('keydown', function (event) {
            console.log("누른 키: " + event.key);
            console.log("키 코드: " + event.code);
            if (event.key === "Enter") {
                console.log("Enter 키가 눌렸습니다.");
            }
        });
    </script>
</body>

</html>
```

문제 5: 여러 버튼 클릭 시 마우스 버튼 번호 확인하기

```html
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 5: 마우스 버튼 번호 확인하기 - 정답 소스</title>
</head>

<body>
    <button class="btn">버튼 1</button>
    <button class="btn">버튼 2</button>
    <button class="btn">버튼 3</button>

    <script>
        // 클래스가 btn인 모든 버튼 선택
        const buttons = document.querySelectorAll('.btn');
        buttons.forEach(function (btn) {
            btn.addEventListener('click', function (event) {
                console.log("클릭된 버튼 번호: " + event.button);
            });
        });
    </script>
</body>
```

```
</html>
```

문제 6: 뷰포트 좌표와 문서 좌표의 차이 이해하기

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 6: 좌표 비교하기 - 정답 소스</title>
    <style>
        /* 스크롤 가능한 긴 페이지를 위해 */
        body {
            height: 2000px;
            padding: 20px;
        }
    </style>
</head>

<body>
    <button id="coordButton">좌표 확인하기</button>

    <script>
        const btn = document.getElementById('coordButton');
        btn.addEventListener('click', function (event) {
            const clientCoords = "(" + event.clientX + ", " + event.clientY + ")";
            const pageCoords = "(" + event.pageX + ", " + event.pageY + ")";
            alert("뷰포트 좌표: " + clientCoords + "\n문서 좌표: " + pageCoords);
        });
    </script>
</body>

</html>
```

문제 7: Ctrl+클릭 시 기본 동작 차단하기

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 7: Ctrl+클릭 기본 동작 차단하기 - 정답 소스</title>
</head>

<body>
    <a id="myLink" href="https://www.google.com" target="_blank">구글로 이동</a>

    <script>
        const link = document.getElementById('myLink');
        // 클릭하면 ctrlKey 여부 검사 후 기본 동작 차단
        link.addEventListener('click', function (event) {
```

```
            if (event.ctrlKey) {
                event.preventDefault();
                console.log("Ctrl+클릭 감지됨: 이동하지 않습니다.");
            }
        });
    </script>
</body>

</html>
```

문제 8: 더블 클릭 시 배경색 변경하기

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 8: 더블 클릭 배경색 변경하기 - 정답 소스</title>
    <style>
        #colorBox {
            width: 300px;
            height: 150px;
            border: 2px solid #000;
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 16px;
            margin: 50px auto;
            background-color: #eee;
            cursor: pointer;
        }
    </style>
</head>

<body>
    <div id="colorBox">더블 클릭해 배경색 변경</div>

    <script>
        const box = document.getElementById('colorBox');

        // 랜덤 색상을 생성하는 함수
        function getRandomColor() {
            const r = Math.floor(Math.random() * 256);
            const g = Math.floor(Math.random() * 256);
            const b = Math.floor(Math.random() * 256);
            return "rgb(" + r + ", " + g + ", " + b + ")";
        }

        // 더블 클릭 시 배경색 변경 이벤트 등록
        box.addEventListener('dblclick', function () {
            box.style.backgroundColor = getRandomColor();
        });
    </script>
```

```
</body>

</html>
```

문제 9: 이벤트 전파 차단 연습

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>문제 9: 이벤트 전파 차단하기 - 정답 소스</title>
  <style>
    #parent {
      width: 400px;
      height: 200px;
      background-color: #d0eaff;
      padding: 20px;
      text-align: center;
    }
    #child {
      width: 200px;
      height: 100px;
      background-color: #ffecb3;
      margin: 0 auto;
      line-height: 100px;
    }
  </style>
</head>
<body>
  <div id="parent">
    부모 요소
    <div id="child">자식 요소</div>
  </div>

  <script>
    const parent = document.getElementById('parent');
    const child = document.getElementById('child');

    // 부모 요소 클릭 이벤트
    parent.addEventListener('click', function() {
      console.log("부모 클릭됨");
    });

    // 자식 요소 클릭 시 전파 차단
    child.addEventListener('click', function(event) {
      event.stopPropagation();
      console.log("자식 클릭됨 (전파 중단)");
    });
  </script>
</body>
</html>
```

문제 10: 키보드 장기 누름(repeat) 감지 및 표시하기

```html
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <title>문제 10: 키보드 장기 누름 감지하기 - 정답 소스</title>
</head>

<body>
    <input id="repeatInput" type="text" placeholder="키를 길게 눌러보세요...">

    <script>
        const input = document.getElementById('repeatInput');
        // 키 입력 이벤트 등록, repeat 속성 확인
        input.addEventListener('keydown', function (event) {
            if (event.repeat) {
                console.log("키가 반복 입력되고 있음: " + event.key);
            } else {
                console.log("키 입력: " + event.key);
            }
        });
    </script>
</body>

</html>
```