

# JavaScript

# INDEX

---

- Event
  - Event란?
  - Event 등록하기
  - Event 전파

# Event

## | 개요

- **Event**란 HTML 요소에서 발생하는 모든 상황을 의미
  - 예를 들어 사용자가 웹 페이지의 버튼을 클릭한다면 클릭에 대해 이벤트가 발생하고 우리는 이벤트를 통해 클릭이라는 사건에 대한 결과를 받거나, 조작을 할 수 있음
- 클릭 말고도 웹에서는 각양각색의 Event가 존재
  - 키보드 키 입력, 브라우저 닫기, 데이터 제출, 텍스트 복사 등

# Event Intro

## Event object (1/2)

- 네트워크 활동이나 사용자와의 상호작용 같은 사건의 발생을 알리기 위한 객체
- 이벤트가 발생했을 때 생성되는 객체
- Event 발생
  - 마우스를 클릭하거나 키보드를 누르는 등 사용자 행동으로 발생할 수도 있고
  - 특정 메서드를 호출하여 프로그래밍적으로도 만들어 낼 수 있음

## | Event object (2/2)

- DOM 요소는 Event를 받고("수신")
- 받은 Event를 "처리"할 수 있음
  - Event 처리는 주로 **addEventListener()** 메서드를 통해  
Event 처리기(Event handler)를 다양한 html 요소에 "부착"해서 처리함

## | Event handler

- 특별한 함수가 아닌 일반적인 JavaScript Function을 정의하여 사용
- 웹 페이지에서 발생하는 Event에 대해 반응하여 동작하는 함수를 지칭
- Event handler 함수는 이벤트가 발생했을 때 호출되며,  
Event 객체를 매개 변수로 전달 받음



## | Event handler - `addEventListener()` (1/4)

“대상”에 특정 Event가 발생하면, 할 일을 등록하자”

`EventTarget.addEventListener(type, handler function)`

## | Event handler - `addEventListener()` (2/4)

- `EventTarget.addEventListener(type, handler function[, options])`
  - 지정한 Event가 대상에 전달될 때마다 호출할 함수를 설정
  - Event를 지원하는 모든 객체(Element, Document, Window 등)를 대상(EventTarget)으로 지정 가능

## | Event handler - `addEventListener()` (3/4)

- `EventTarget.addEventListener(type, handler function[, options])`
  - `type`
    - 반응 할 Event 유형을 나타내는 대소문자 구분 문자열
    - 대표 이벤트
      - `input`, `click`, `submit` ...
    - 다양한 이벤트 확인(<https://developer.mozilla.org/en-US/docs/Web/Events>)

## | Event handler - `addEventListener()` (4/4)

- `EventTarget.addEventListener(type, handler function[, options])`
  - `handler function`
    - 지정된 타입의 Event를 수신할 객체
    - JavaScript function 객체(콜백 함수)여야 함
    - 콜백 함수는 발생한 Event의 데이터를 가진 Event 객체를 유일한 매개변수로 받음

# Event 실습

## button 실습 (1/2)

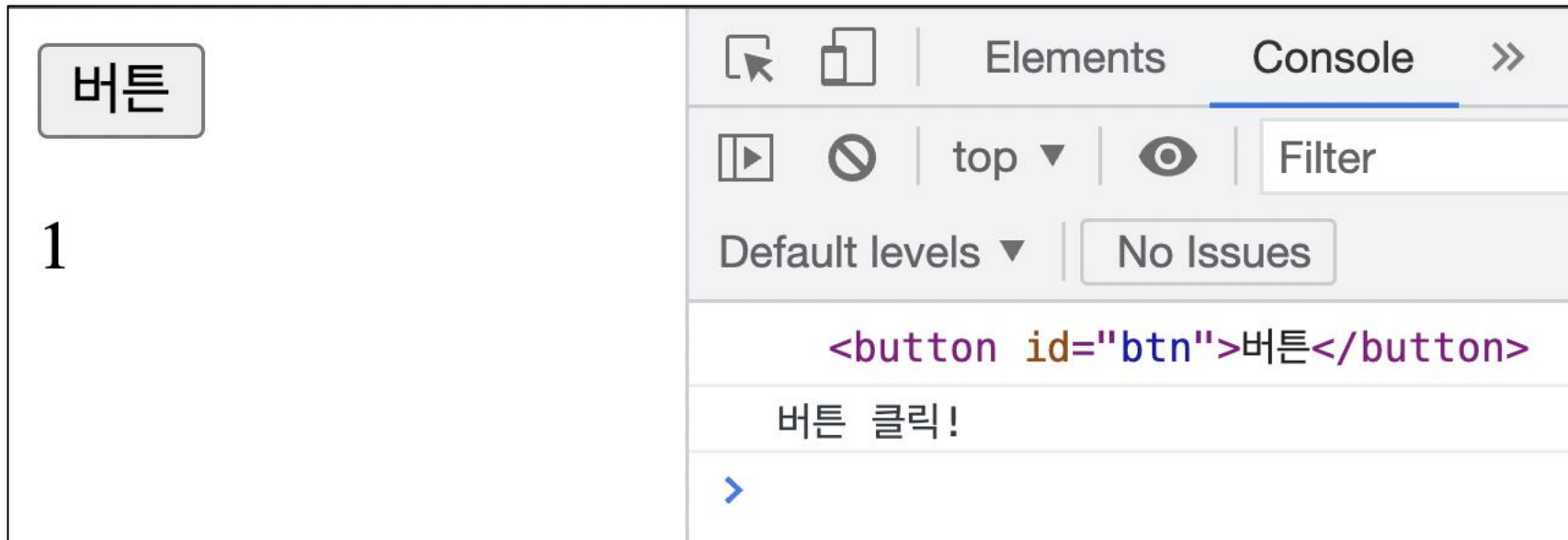
- 버튼을 클릭하면 특정 변수 값 변경하기

```
<body>
  <button id="btn">버튼</button>
  <p id="counter">0</p>
  <script>
    // 초기값
    let countNumber = 0

    // ID가 btn인 요소를 선택
    const btn = document.querySelector('#btn ')
    console.log(btn)
    // btn이 클릭 되었을 때마다 함수가 실행됨
    btn.addEventListener('click', function() {
      console.log('버튼 클릭! ')
      // countNumber를 증가시키고
      countNumber += 1
      // id가 counter인의 내용을 변경 시킨다.
      const counter = document.querySelector('#counter ')
      counter.innerText = countNumber
    })
  </script>
</body>
```

## button 실습 (2/2)

- 실행 결과



## 값 입력 실습 (1/2)

- input에 입력하면 입력 값을 실시간으로 출력하기

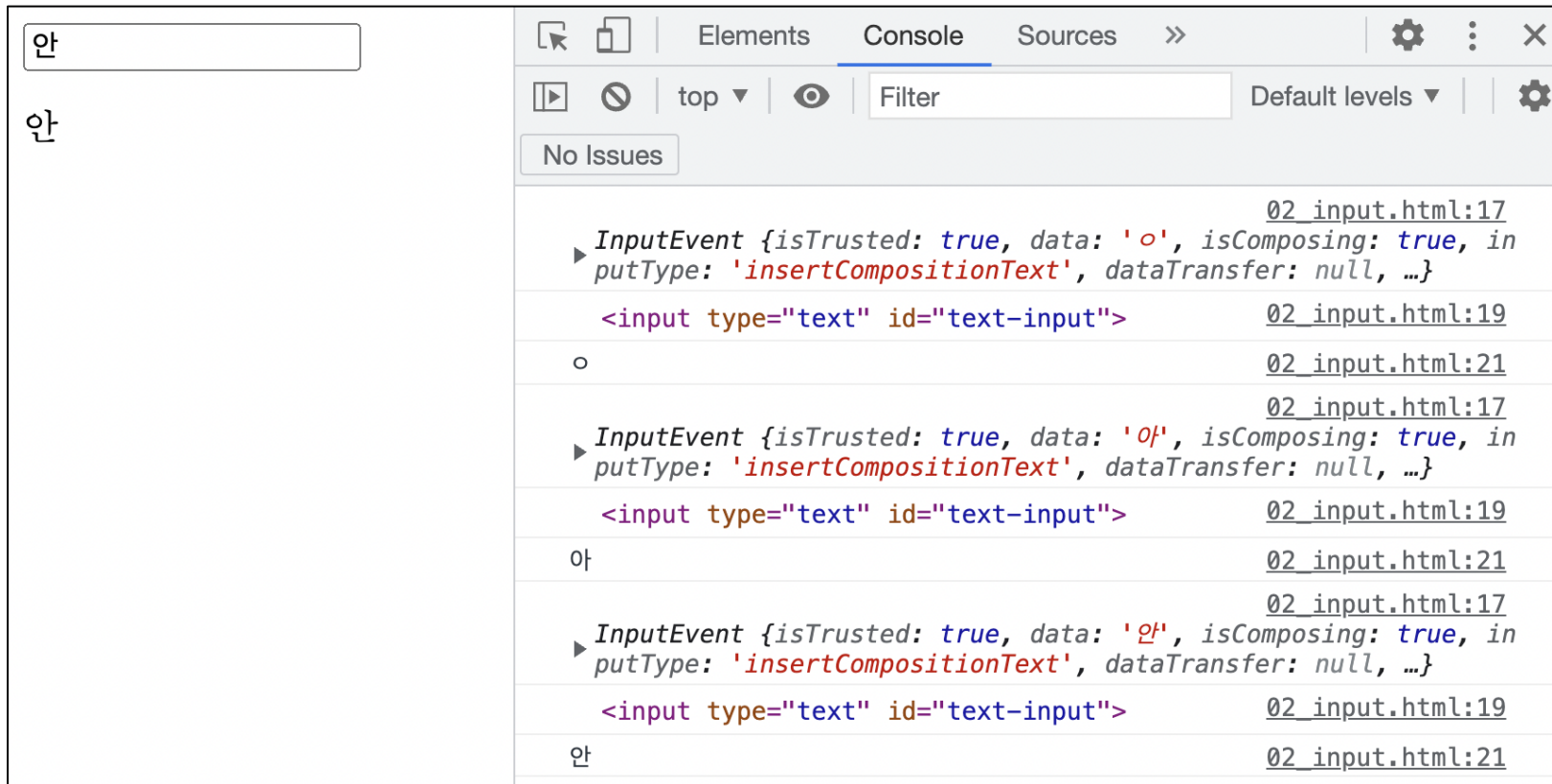
```
<body>
  <input type="text" id="text-input">
  <p></p>
  <script>
    // 1. input 선택
    const textInput = document.querySelector('#text-input ')
    // 2. input 이벤트 등록
    textInput.addEventListener('input', function (event) {
      console.log(event)
      // input은 이벤트의 대상
      console.log(event.target)
      // input의 value를 받아오기
      console.log(event.target.value)

      // 3. input에 작성한 값을 p 태그에 출력하기
      const pTag = document.querySelector('p')
      pTag.innerText = event.target.value
    })
  </script>
</body>
```



## 값 입력 실습 (2/2)

- 실행 결과



The screenshot displays a web browser interface with a text input field containing the Korean character '안'. Below the input field, the text '안' is also visible. To the right, the browser's developer console is open, showing the 'Console' tab. The console displays three log entries, each representing an 'InputEvent' triggered by a user input. Each entry includes the event object and the corresponding HTML element (

Log Entry	File
<code>InputEvent {isTrusted: true, data: '안', isComposing: true, inputType: 'insertCompositionText', dataTransfer: null, ...}</code>	02_input.html:17
<code>&lt;input type="text" id="text-input"&gt;</code>	02_input.html:19
안	02_input.html:21
<code>InputEvent {isTrusted: true, data: '안', isComposing: true, inputType: 'insertCompositionText', dataTransfer: null, ...}</code>	02_input.html:17
<code>&lt;input type="text" id="text-input"&gt;</code>	02_input.html:19
안	02_input.html:21
<code>InputEvent {isTrusted: true, data: '안', isComposing: true, inputType: 'insertCompositionText', dataTransfer: null, ...}</code>	02_input.html:17
<code>&lt;input type="text" id="text-input"&gt;</code>	02_input.html:19
안	02_input.html:21

## 복합 실습 (1/2)

- input에 입력하면 입력 값을 실시간으로 출력하고 버튼을 클릭하면 출력된 값의 클래스를 토글하기

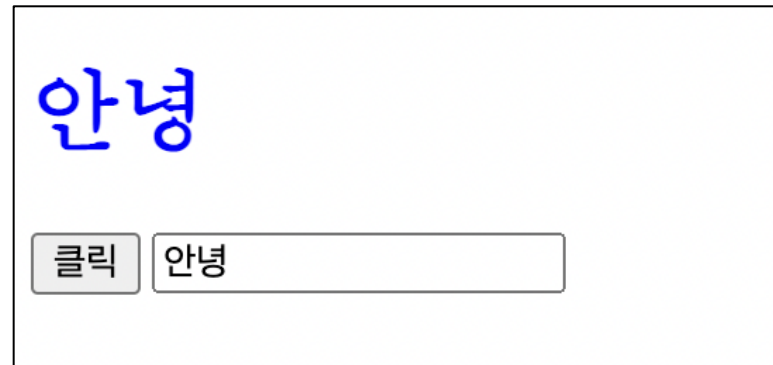
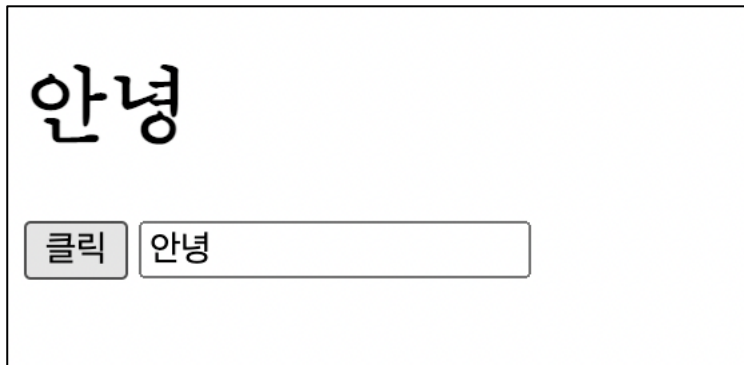
```
...
<style>
  .blue {
    color: blue;
  }
</style>
</head>
<body>
  <h1></h1>
  <button id="btn">클릭</button>
  <input type="text">

  <script>
    const btn = document.querySelector('#btn ')
    // btn이 클릭되면 함수 실행
    btn.addEventListener('click', function() {
      // h1 태그를 선택해서
      const h1 = document.querySelector('h1 ')
      // 클래스 blue를 토글하기
      h1.classList.toggle('blue ')
    })

    // input
    const input = document.querySelector('input ')
    // input에 값이 입력되면 함수 실행
    input.addEventListener('input', function(event) {
      // h1 태그를 선택해서
      const h1Tag = document.querySelector('h1 ')
      // input값을 태그의 콘텐츠로 채우기
      h1Tag.innerText = event.target.value
    })
  </script>
```

## | 복합 실습 (2/2)

- 실행 결과



## | addEventListener 정리

- “~ 하면 ~ 한다.”
  - “클릭하면, 경고창을 띄운다.”
  - “특정 Event가 발생하면, 할 일(콜백 함수)을 등록한다.”

# Event 전파와 취소

## | Event 전파란?

- DOM 요소에서 발생한 이벤트가 상위 노드에서 하위 노드  
혹은, 하위 노드에서 상위 노드로 전파되는 현상을 의미
- `addEventListener` 메서드를 사용하여 전파 방식을 제어할 수 있음  
기본 값은 하위 노드에서 상위 노드로 전파되는 방식을 사용 - Event Bubbling
- 또한, 이러한 이벤트 전파 상황을 필요에 따라 제어 할 수도 있음

## | event.preventDefault()

- 현재 Event의 기본 동작을 중단
- HTML 요소의 기본 동작을 작동하지 않게 막음
- HTML 요소의 기본 동작 예시
  - a 태그 : 클릭 시 특정 주소로 이동
  - form 태그 : form 데이터 전송

# Event 취소 실습



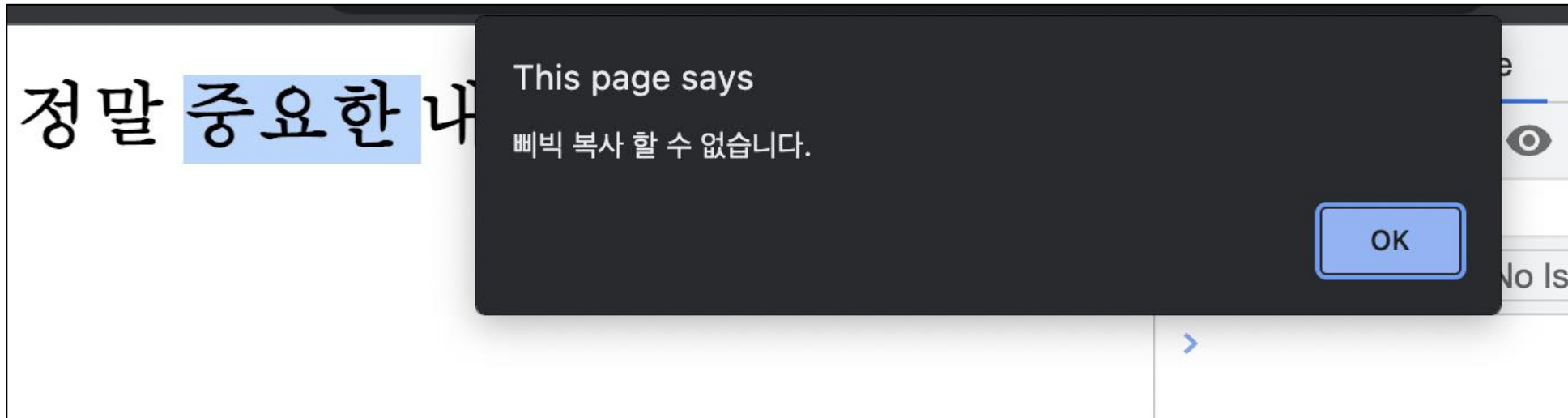
## 이벤트 취소 실습 (1/2)

- 웹 페이지 내용을 복사하지 못하도록 하기

```
<body>
  <div>
    <h1>정말 중요한 내용</h1>
  </div>
  <script>
    const h1 = document.querySelector('h1 ')
    h1.addEventListener('copy', function(event) {
      // copy event의 기본 동작을 막기
      event.preventDefault()
      alert('빠빅 복사 할 수 없습니다. ')
    })
  </script>
</body>
```

## | 이벤트 취소 실습 (1/2)

- 실행 결과



# Event 종합 실습

## 종합 실습 1 (1/2)

- 버튼을 클릭하면 랜덤 로또 번호 6개를 출력하기

로또 추천 번호

행운 번호 받기

33

4

14

37

9

29

# Event 종합 실습

## 종합 실습 1 (2/2)

```
<h1>로또 추천 번호</h1>
<button id="lotto-btn">행운 번호 받기</button>
<div id="result"></div>
```

```
<script src="https://cdn.jsdelivr.net/npm/lodash@4.17.21/lodash.min.js"></script>
<script>
  const button = document.querySelector('#lotto-btn')
  button.addEventListener('click', function() {
    // 컨테이너를 만들고
    const ballContainer = document.createElement('div')
    ballContainer.classList.add('ball-container')

    // 랜덤 숫자 6개 만들기
    const numbers = _.sampleSize(_.range(1, 46), 6)
    // console.log(numbers)

    // 공 만들기
    numbers.forEach((number) => {
      const ball = document.createElement('div')
      ball.classList.add('ball')
      ball.innerText = number
      ball.style.backgroundColor = 'crimson'
      // 공을 컨테이너의 자식으로 추가
      ballContainer.appendChild(ball)
    })

    // 컨테이너를 결과 영역의 자식으로 추가
    const result = document.querySelector('#result')
    result.appendChild(ballContainer)
  })
</script>
```

## [참고] lodash

- 모듈성, 성능 및 추가 기능을 제공하는 JavaScript 유틸리티 라이브러리
- array, object 등 자료구조를 다룰 때 사용하는 유용하고 간편한 유틸리티 함수들을 제공
- 함수 예시
  - reverse, sortBy, range, random ...
- <https://lodash.com/>

## | 종합 실습 2 (1/2)

- CREATE, READ 기능을 충족하는 todo app 만들기

Add

- 저녁 식사 후 알고리즘 연습
- django 실습

# Event 종합 실습

## 종합 실습 2 (2/2)

```
<form action="#">
  <input type="text" class="inputData">
  <input type="submit" value="Add">
</form>
<ul></ul>
```

```
<script>
  const formTag = document.querySelector('form ')

  const addTodo = function (event) {
    event.preventDefault()

    const inputTag = document.querySelector('.inputData ')
    const data = inputTag.value

    if (data.trim()) {
      const liTag = document.createElement('li ')
      liTag.innerText = data

      const ulTag = document.querySelector('ul ')
      ulTag.appendChild(liTag)
      event.target.reset()
    } else {
      alert('할 일을 입력하세요. ')
    }
  }

  formTag.addEventListener('submit', addTodo)
</script>
```



## [참고] this와 addEventListener (1/2)

- addEventListener에서의 콜백 함수는 특별하게 function 키워드의 경우 addEventListener를 호출한 대상을( event.target ) 뜻함
- 반면 화살표 함수의 경우 상위 스코프를 지칭하기 때문에 window 객체가 바인딩 됨
- 결론
  - “addEventListener 의 콜백 함수는 function 키워드를 사용하기”

## this와 addEventListener (2/2)

```
<body>
  <button id="function">function</button>
  <button id="arrow">arrow function</button>

  <script>
    const functionButton = document.querySelector('#function')
    const arrowButton = document.querySelector('#arrow')

    functionButton.addEventListener('click', function(event) {
      console.log(this) // <button id="function">function</button>
    })

    arrowButton.addEventListener('click', event => {
      console.log(this) // window
    })
  </script>
</body>
```

# 다음 방송에서 만나요!

삼성 청년 SW 아카데미