

# 혼자 공부하는 자바스크립트

Chapter 07 문서 객체 모델





# Contents

- CHAPTER 07: 문서 객체 모델

SECTION 7-1 문서 객체 조작하기

SECTION 7-2 이벤트 활용

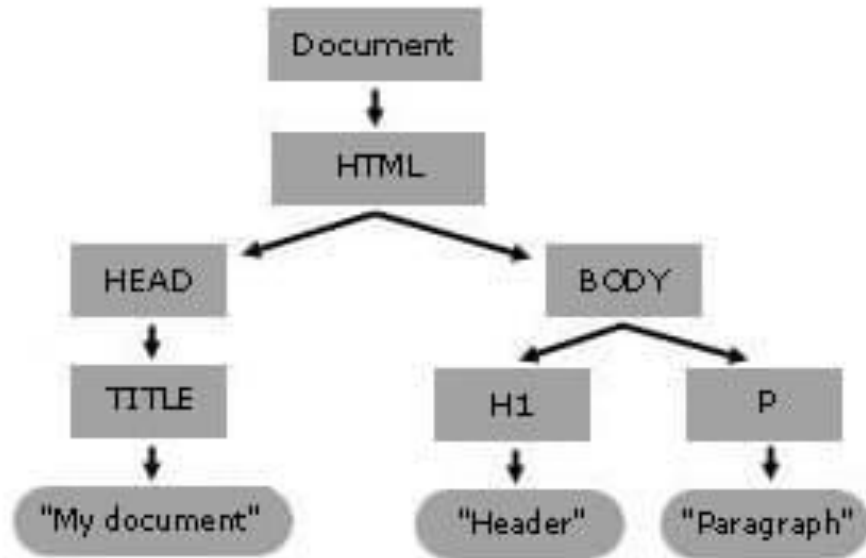


# CHAPTER 07 문서 객체 모델

DOMContentLoaded 이벤트를 사용한 문서 객체 조작과 다양한 이벤트의 사용 방법 이해

# SECTION 7-1 문서 객체 모델

- 문서 객체 모델(Document Objects Model, DOM)
  - 문서 객체 모델은 HTML, XML 문서의 프로그래밍 인터페이스
  - 이 객체 모델은 문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공
  - 웹 페이지는 일종의 문서(document)로서 웹 브라우저를 통해 내용이 해석되어 화면에 나타남



document 콘텐츠 트리

## 자바스크립트로 할 수 있는 객체 모델 작업

- 새로운 HTML 요소나 속성을 추가
- 존재하는 HTML 요소나 속성을 제거
- HTML 문서의 모든 HTML 요소, 속성 변경
- HTML 문서의 모든 CSS 스타일 변경
- HTML 문서에 새로운 HTML 이벤트 추가
- 모든 HTML 이벤트 처리

# SECTION 7-1 문서 객체 조작하기(1)

- DOMContentLoaded 이벤트
  - HTML 코드를 자바스크립트로 조작하기 (소스 코드 7-1-1.html)

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04 <title>DOMContentLoaded</title>
05 <script>
06 // HTML 태그를 쉽게 만들 수 있는 콜백 함수를 선언합니다.
07 const h1 = (text) => `<h1>${text}</h1>`
08 </script>
09 <script>
10 document.body.innerHTML += h1('1번째 script 태그')
11 </script>
12 </head>
13 <body>
14 <script>
15 document.body.innerHTML += h1('2번째 script 태그')
16 </script>
17 <h1>1번째 h1 태그</h1>
18 <script>
19 document.body.innerHTML += h1('3번째 script 태그')
20 </script>
21 <h1>2번째 h2 태그</h1>
22 </body>
23 </html>
```

body 태그가 생성되기 이전에 script 태그로 body 태그를 조작

앞에서 선언한 h1 함수를 실행

body 태그는 head 태그 다음에 생성

실행 결과

2번째 script 태그  
1번째 h1 태그  
3번째 script 태그  
2번째 h2 태그

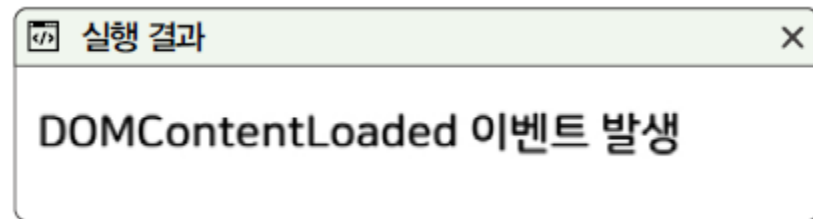
## SECTION 7-1 문서 객체 조작하기(2)

### ◦ DOMContentLoaded 이벤트

- DOMContentLoaded 이벤트는 웹 브라우저가 문서 객체를 모두 읽고 나서 실행하는 이벤트
- 다음과 같이 코드를 구성하면 DOMContentLoaded 상태가 되었을 때 콜백 함수를 호출
- DOMContentLoaded 이벤트 (소스 코드 7-1-2.html)

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04 <title>DOMContentLoaded</title>
05 <script>
06 // DOMContentLoaded 이벤트를 연결합니다.
07 document.addEventListener('DOMContentLoaded', () => {
08   const h1 = (text) => `<h1>${text}</h1>`
09   document.body.innerHTML += h1('DOMContentLoaded 이벤트 발생')
10 })
11 </script>
12 </head>
13 <body>
14
15 </body>
16 </html>
```

문서 객체를 모두 읽어들이면(DOMContentLoaded)  
이 콜백 함수가 실행



## SECTION 7-1 문서 객체 조작하기(3)

- 문서 객체 가져오기

- document.body 코드를 사용하여 문서의 body 요소 읽기

---

```
document.head  
document.body  
document.title
```

---

- head 요소와 body 요소 내부에 만든 다른 요소들은 다음과 같은 별도의 메소드를 사용

---

```
document.querySelector(선택자)  
document.querySelectorAll(선택자)
```

---

이름	선택자 형태	설명
태그 선택자	태그	특정 태그를 가진 요소를 추출
아이디 선택자	#아이디	특정 id 속성을 가진 요소를 추출
클래스 선택자	.클래스	특정 class 속성을 가진 요소를 추출
속성 선택자	[속성=값]	특정 속성 값을 갖고 있는 요소를 추출
후손 선택자	선택자_A 선택자_B	선택자_A 아래에 있는 선택자_B를 선택

## SECTION 7-1 문서 객체 조작하기(4)

- 문서 객체 가져오기

- querySelector() 메소드를 사용해서 h1 태그를 추출하고 조작하기
- querySelector() 메소드 (소스 코드 7-1-3.html)

```
01 <script>
02  document.addEventListener('DOMContentLoaded', () => {
03    // 요소를 읽어들이니다.
04    const header = document.querySelector('h1') → h1 태그 이름으로 요소를 선택
05
06    // 텍스트와 스타일을 변경합니다.
07    header.textContent = 'HEADERS'
08    header.style.color = 'white'
09    header.style.backgroundColor = 'black'
10    header.style.padding = '10px'
11  })
12 </script>
13 <body>
14  <h1></h1>
15 </body>
```





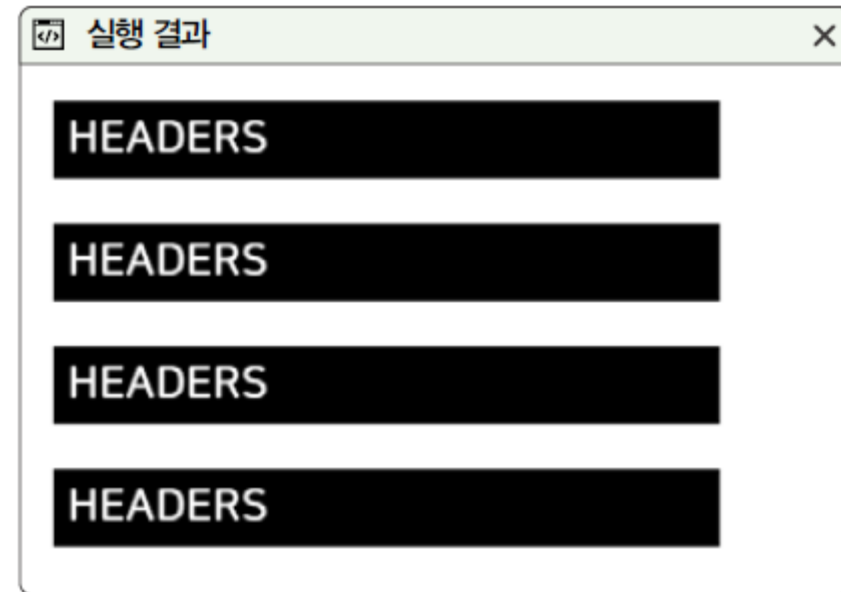
# SECTION 7-1 문서 객체 조작하기(5)

- 문서 객체 가져오기

- querySelectorAll() 메소드: 문서 객체 여러 개를 배열로 읽어들이는 함수
- querySelectorAll() 메소드 (소스 코드 7-1-4.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03     // 요소를 읽어들이니다.
04     const headers = document.querySelectorAll('h1')
05
06     // 텍스트와 스타일을 변경합니다.
07     headers.forEach((header) => {
08         header.textContent = 'HEADERS'
09         header.style.color = 'white'
10         header.style.backgroundColor = 'black'
11         header.style.padding = '10px'
12     })
13 })
14 </script>
15 <body>
16 <h1></h1>
17 <h1></h1>
18 <h1></h1>
19 <h1></h1>
20 </body>
```

→ 태그 이름으로 요소를 선택



## SECTION 7-1 문서 객체 조작하기(6)

### ◦ 글자 조작하기

속성 이름	설명
문서 객체.textContent	입력된 문자열을 그대로 기입
문서 객체.innerHTML	입력된 문자열을 HTML 형식으로 기입

#### ▪ 글자 조작하기 (소스 코드 7-1-5.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const a = document.querySelector('#a')
04   const b = document.querySelector('#b')
05
06   a.textContent = '<h1>textContent 속성</h1>'
07   b.innerHTML = '<h1>innerHTML 속성</h1>'
08 })
09 </script>
10 <body>
11 <div id="a"></div>
12 <div id="b"></div>
13 </body>
```

특정 아이디로 요소를 선택



## SECTION 7-1 문서 객체 조작하기(7)

### ◦ 속성 조작하기

메소드 이름	설명
문서 객체.setAttribute(속성 이름, 값)	특성 속성에 값을 지정
문서 객체.getAttribute(속성 이름)	특정 속성을 추출

#### ▪ 속성 조작하기 (소스 코드 7-1-6.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const rects = document.querySelectorAll('.rect')
04
05   rects.forEach((rect, index) => {
06     const width = (index + 1) * 100
07     const src = `http://placekitten.com/${width}/250`
08     rect.setAttribute('src', src)
09   })
10 })
11 </script>
12 <body>
13 <img class="rect">
14 <img class="rect">
15 <img class="rect">
16 <img class="rect">
17 </body>
```

index 값은 [0, 1, 2, 3]이 반복. 1을 더해서  
[1, 2, 3, 4]가 되게 만들고, 100을 곱해서 너비가  
[100, 200, 300, 400]이 되게 만든 것



## SECTION 7-1 문서 객체 조작하기(8)

- 스타일 조작하기

CSS 속성 이름	자바스크립트 style 속성 이름
background-color	backgroundColor
text-align	textAlign
font-size	fontSize

- 스타일 조작하기 3 가지

<div><div><div><div><div></div><div>h1.style.backgroundColor</div></div><div><div>h1.style['backgroundColor']</div><div>h1.style['background-color']</div></div></div></div></div> <div><div>→</div><div>이 형태를 가장 많이 사용</div></div>
---

## SECTION 7-1 문서 객체 조작하기(9)

### 스타일 조작하기

- 25개의 div 태그를 조작해서 검은색에서 흰색으로 변화하는 그라데이션트를 만드는 코드 만들기
- 스타일 조작하기 (소스 코드 7-1-7.html)

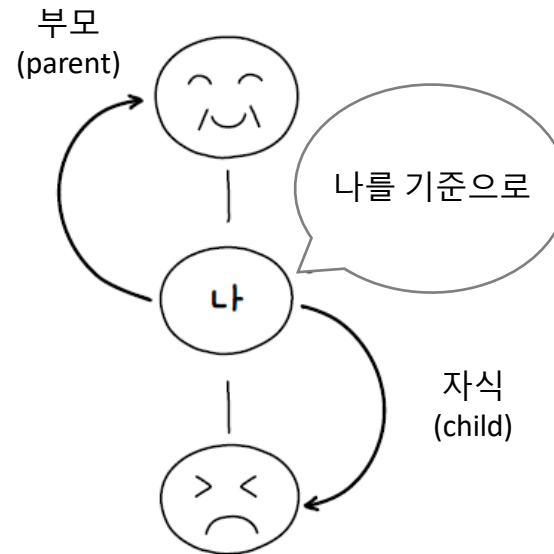
```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const divs = document.querySelectorAll('body > div') → body 태그 아래에 있는 div 태그를 선택
04
05   divs.forEach((div, index) => { → div 개수만큼 반복하여 출력
06     console.log(div, index)
07     const val = index * 10 → 인덱스는 0부터 24까지 반복
08     div.style.height = `10px` → 크기를 지정할 때는 반드시 단위를 함께 붙여줘야 함
09     div.style.backgroundColor = `rgba(${val}, ${val}, ${val})`
10   })
11 })
12 </script>
13 <body>
14 <!-- div 태그 25개 -->
15 <div></div><div></div><div></div><div></div><div></div>
16 <div></div><div></div><div></div><div></div><div></div>
17 <div></div><div></div><div></div><div></div><div></div>
18 <div></div><div></div><div></div><div></div><div></div>
19 <div></div><div></div><div></div><div></div><div></div>
20 </body>
```



## SECTION 7-1 문서 객체 조작하기(10)

### ◦ 문서 객체 생성하기

- `document.createElement()` 메소드를 사용  
`document.createElement(문서 객체 이름)`
- 문서 객체 트리(tree) 구조
  - 문서를 어떤 문서 아래에 추가할 지를 지정  
`부모 객체.appendChild(자식 객체)`

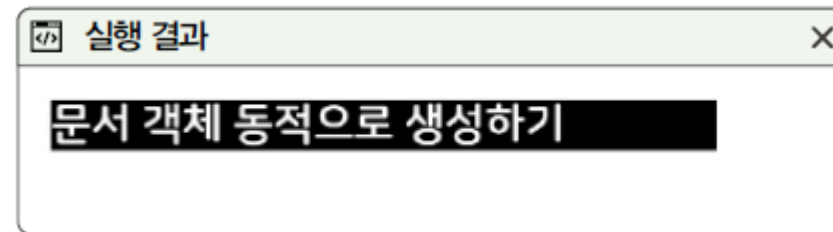


# SECTION 7-1 문서 객체 조작하기(11)

## 문서 객체 생성하기

- document.createElement() 메소드로 h1 태그를 생성하고, 이를 document.body 태그 아래에 추가하는 코드
- 문서 객체 생성하고 추가하기 소스 (코드 7-1-8.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03     // 문서 객체 생성하기
04     const header = document.createElement('h1')      —————> h1 태그 생성
05
06     // 생성한 태그 조작하기
07     header.textContent = '문서 객체 동적으로 생성하기'
08     header.setAttribute('data-custom', '사용자 정의 속성')
09     header.style.color = 'white'
10     header.style.backgroundColor = 'black'
11
12     // h1 태그를 body 태그 아래에 추가하기
13     document.body.appendChild(header)
14 })
15 </script>
16 <body>
17
18 </body>
```

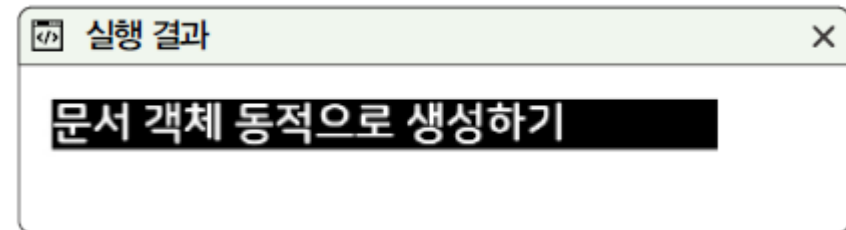


# SECTION 7-1 문서 객체 조작하기(12)

## 문서 객체 이동하기

- appendChild() 메소드는 문서 객체를 이동할 때도 사용
  - 문서 객체의 부모(parent)는 언제나 하나여야 하고, 문서 객체를 다른 문서 객체에 추가하면 문서 객체가 이동
- 문서 객체 생성하고 추가하기 (소스 코드 7-1-8.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03     // 문서 객체 생성하기
04     const header = document.createElement('h1')  —————> h1 태그 생성
05
06     // 생성한 태그 조작하기
07     header.textContent = '문서 객체 동적으로 생성하기'
08     header.setAttribute('data-custom', '사용자 정의 속성')
09     header.style.color = 'white'
10     header.style.backgroundColor = 'black'
11
12     // h1 태그를 body 태그 아래에 추가하기
13     document.body.appendChild(header)
14 })
15 </script>
16 <body>
17
18 </body>
```





## SECTION 7-1 문서 객체 조작하기(13)

### ◦ 문서 객체 이동하기

- appendChild() 메소드는 문서 객체를 이동할 때도 사용
- 문서 객체 이동하기 (소스 코드 7-1-9.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03     // 문서 객체 읽어들이고 생성하기
04     const divA = document.querySelector('#first')  —————> id 속성이 first인 div 태그를 선택
05     const divB = document.querySelector('#second') —————> id 속성이 second인 div 태그를 선택
06     const h1 = document.createElement('h1')     —————> h1 태그를 생성.
07     h1.textContent = '이동하는 h1 태그'
08
09     // 서로 번갈아가면서 실행하는 함수를 구현합니다.
10     const toFirst = () => {
11         divA.appendChild(h1)  —————> h1을 divA에 추가
12         setTimeout(toSecond, 1000) —————> 1초 뒤에 toSecond 함수를 실행.
13     }
14     const toSecond = () => {
15         divB.appendChild(h1)  —————> h1을 divB에 추가.
16         setTimeout(toFirst, 10000) —————> 10초 뒤에 toFirst 함수를 실행
17     }
18     toFirst()
19 })
20 </script>
```

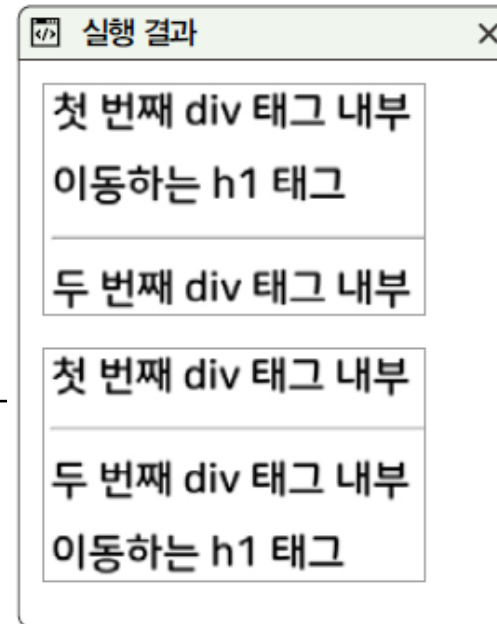
▶ 다음 쪽에 코드 이어짐

## SECTION 7-1 문서 객체 조작하기(14)

- 문서 객체 이동하기
  - appendChild() 메소드는 문서 객체를 이동할 때도 사용
  - 문서 객체 이동하기 (소스 코드 7-1-9.html)

◀ 앞쪽에 이어

```
21 <body>
22 <div id="first">
23   <h1>첫 번째 div 태그 내부</h1>
24 </div>
25 <hr>
26 <div id="second">
27   <h1>두 번째 div 태그 내부</h1>
28 </div>
29 </body>
```



## SECTION 7-1 문서 객체 조작하기(15)

- 문서 객체 제거하기

- removeChild() 메소드: 문서 객체를 제거

---

부모 객체.removeChild(자식 객체)

---

- appendChild() 메소드 등으로 부모 객체와 이미 연결이 완료된 문서 객체의 경우 parentNode 속성으로 부모 객체에 접근할 수 있으므로, 일반적으로 어떤 문서 객체를 제거할 때는 다음과 같은 형태의 코드를 사용

---

문서 객체.parentNode.removeChild(문서 객체)

---

## SECTION 7-1 문서 객체 조작하기(16)

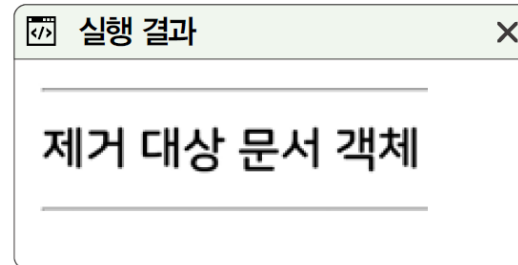
### 문서 객체 제거하기

- 특정 개체를 간단하게 실행하고 3초 후에 화면에서 h1 태그를 제거하는 코드 만들기
- 문서 객체 제거하기 (소스 코드 7-1-10.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   setTimeout(() => {
04     const h1 = document.querySelector('h1')
05
06     h1.parentNode.removeChild(h1)
07     // document.body.removeChild(h1)
08   }, 3000)
09 })
10 </script>
11 <body>
12   <hr>
13   <h1>제거 대상 문서 객체</h1>
14   <hr>
15 </body>
```

h1 태그의 부모 객체 body 태그에 접근하여 제거

h1.parentNode가 document.body이므로,  
이런 형태로도 제거할 수 있음



# SECTION 7-1 문서 객체 조작하기(17)

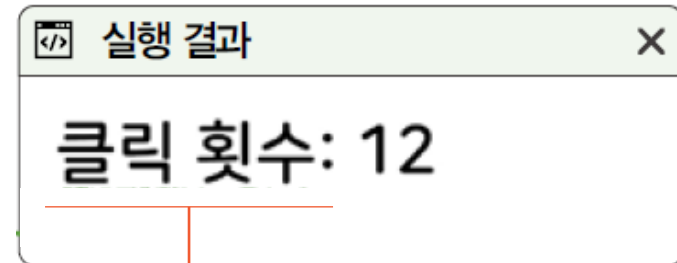
## 이벤트 설정하기

### addEventListener() 메소드

문서 객체.addEventListener(이벤트 이름, 콜백 함수) → 이벤트 리스너 (이벤트 핸들러)

### 이벤트 연결하기 (소스 코드 7-1-11.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   let counter = 0
04   const h1 = document.querySelector('h1')
05
06   h1.addEventListener('click', (event) => { → h1 태그에 이벤트가 발생할 때 실행할 함수
07     counter++
08     h1.textContent = `클릭 횟수: ${counter}`
09   })
10 })
11 </script>
12 <style>
13 h1 {
14   /* 클릭을 여러 번 했을 때
15    글자가 선택되는 것을 막기 위한 스타일 */
16   user-select: none;
17 }
18 </style>
19 <body>
20 <h1>클릭 횟수: 0</h1>
21 </body>
```



클릭할 때마다  
클릭 횟수를 출력

## SECTION 7-1 문서 객체 조작하기(18)

### 이벤트 설정하기

#### removeEventListener() 메소드

문서 객체.removeEventListener(이벤트 이름, 이벤트 리스너)

#### 이벤트 연결 제거하기 (소스 코드 7-1-12.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   let counter = 0
04   let isConnected = false
05
06   const h1 = document.querySelector('h1')
07   const p = document.querySelector('p')
08   const connectButton = document.querySelector('#connect')
09   const disconnectButton = document.querySelector('#disconnect')
10
11   const listener = (event) => {
12     h1.textContent = `클릭 횟수: ${counter++}`
13   }
14
15   connectButton.addEventListener('click', () => {
16     if (isConnect === false) {
17       h1.addEventListener('click', listener)
18       p.textContent = '이벤트 연결 상태: 연결'
19       isConnected = true
20     }
21   })
```

이벤트를 제거하려면 이벤트 리스너를  
변수 또는 상수로 가지고 있어야 함

▶ 다음 쪽에 코드 이어짐

# SECTION 7-1 문서 객체 조작하기(19)

## 이벤트 설정하기

### 이벤트 연결 제거하기 (소스 코드 7-1-12.html)

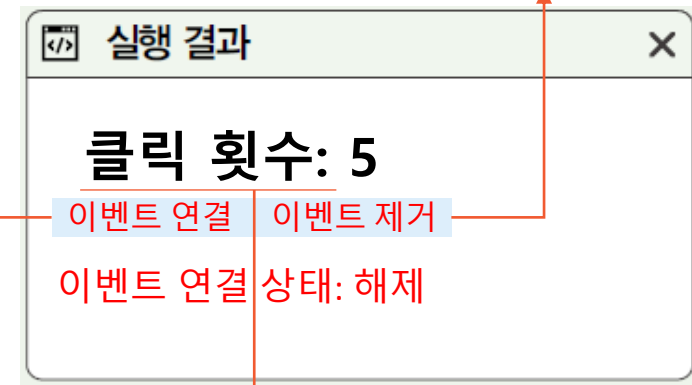
#### ◀ 앞쪽에 이어

```
22 disconnectButton.addEventListener('click', () => {
23   if (isConnect === true) {
24     h1.removeEventListener('click', listener)
25     p.textContent = '이벤트 연결 상태: 해제'
26     isConnect = false
27   }
28 })
29 })
30 </script>
31 <style>
32 h1 {
33   /* 클릭을 여러 번 했을 때
34     글자가 선택되는 것을 막기 위한 스타일 */
35   user-select: none;
36 }
37 </style>
38 <body>
39 <h1>클릭 횟수: 0</h1>
40 <button id="connect">이벤트 연결</button>
41 <button id="disconnect">이벤트 제거</button>
42 <p>이벤트 연결 상태: 해제</p>
43 </body>
```

→ 해제할 때 이벤트 리스너를 사용

클릭 시 연결 상태는  
'연결'로 나옴

클릭 시 연결 상태는  
'해제'로 나옴



이벤트 연결 상태에서  
클릭하면 클릭 횟수를 출력

## [요점 정리]

### ◦ 6가지 키워드로 정리하는 핵심 포인트

- DOMContentLoaded 이벤트는 HTML 페이지의 모든 문서 객체(요소)를 웹 브라우저가 읽어들이었을 때 발생시키는 이벤트
- querySelector() 메소드는 문서 객체를 선택할 때 사용하는 메소드
- textContent 속성과 innerHTML 속성은 문서 객체 내부의 글자를 조작할 때 사용하는 속성
- style 속성은 문서 객체의 스타일을 조작할 때 사용하는 속성
- 이벤트 리스너(이벤트 핸들러)는 이벤트가 발생할 때 실행하는 함수를 의미



## SECTION 7-2 이벤트 활용(1)

- 이벤트 모델

- 표준 이벤트 모델: `addEventListener()`


---

```
document.body.addEventListener('keyup', () => {  
  
})
```

---

- 고전 이벤트 모델: 문서 객체가 갖고 있는 `on○○`으로 시작하는 속성에 함수를 할당해서 이벤트를 연결

---

```
document.body.onkeyup = (event) => {  
     속성  
}
```

---

- 인라인 이벤트 모델: `on○○`으로 시작하는 속성을 HTML 요소에 직접 넣어서 이벤트를 연결

---

```
<script>  
  const listener = (event) => {  
  
  }  
</script>  
<body onkeyup="listener(event)">  
</body>
```

---

## SECTION 7-2 이벤트 활용(2)

- 키보드 이벤트

이벤트	설명
keydown	키가 눌릴 때 실행. 키보드를 꼭 누르고 있을 때도, 입력될 때도 실행됨
keypress	키가 입력되었을 때 실행. 하지만 웹 브라우저에 따라서 아시아권의 문자(한국어, 중국어, 일본어)를 제대로 처리하지 못하는 문제가 있음
keyup	키보드에서 키가 떨어질 때 실행

- keydown 이벤트

글자 수: 1

- keypress 이벤트

공백이 들어가기 전까지는 글자수를 세지 않습니다

- keyup 이벤트

□ □ □ □ □ □ □ □

## SECTION 7-2 이벤트 활용(3)

- 키보드 이벤트
  - 남은 글자 수 출력하기 (소스 코드 7-2-1.html)

```
01 <script>
02  document.addEventListener('DOMContentLoaded', () => {
03    const textarea = document.querySelector('textarea')
04    const h1 = document.querySelector('h1')
05
06    textarea.addEventListener('keyup', (event) => {
07      const length = textarea.value.length → value 속성으로 입력 양식의 글자를 읽어들이 수 있음
08      h1.textContent = `글자 수: ${length}`
09    })
10  })
11 </script>
12 <body>
13  <h1></h1>
14  <textarea></textarea>
15 </body>
```



## SECTION 7-2 이벤트 활용(4)

- 키보드 키 코드 사용하기

이벤트 속성 이름	선택자 형태
code	입력한 키
keyCode	입력한 키를 나타내는 숫자
altKey	[Alt] 키를 눌렀는지
ctrlKey	[Ctrl] 키를 눌렀는지
shiftKey	[Shift] 키를 눌렀는지

- code 속성은 입력한 키를 나타내는 문자열이 들어 있고, altKey, ctrlKey, shiftKey 속성은 해당 키를 눌렀는지 불 자료형 값이 들어 있음

## SECTION 7-2 이벤트 활용(5)

- 키보드 키 코드 사용하기

- 키보드 이벤트와 관련된 이벤트 속성 (소스 코드 7-2-2.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const h1 = document.querySelector('h1')
04   const print = (event) => {
05     let output = ''
06     output += `alt: ${event.altKey}<br>`
07     output += `ctrl: ${event.ctrlKey}<br>`
08     output += `shift: ${event.shiftKey}<br>`
09     output += `code: ${typeof(event.code) !== 'undefined' ?
10       event.code : event.keyCode}<br>`
11     h1.innerHTML = output
12   }
13
14   document.addEventListener('keydown', print)
15   document.addEventListener('keyup', print)
16 })
17 </script>
18 <body>
19 <h1></h1>
20 </body>
```

이벤트가 발생하면 불 값을 반환

event.code가 있으면 event.code를 출력하고,  
undefined라면 event.keyCode를 출력

키가 눌릴 때 출력

키가 떨어질 때 출력합니다.

## SECTION 7-2 이벤트 활용(6)

- 키보드 키 코드 사용하기
  - 키로 별 움직이기 (소스 코드 7-2-3.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   // 별의 초기 설정
04   const star = document.querySelector('h1')
05   star.style.position = 'absolute'           → style 속성을 조작하여 position 값을 설정
06
07   // 별의 이동을 출력하는 기능
08   let [x, y] = [0, 0]
09   const block = 20
10   const print = () => {
11     star.style.left = `${x * block}px`
12     star.style.top = `${y * block}px`
13   }
14   print()
15
16   // 별을 이동하는 기능
17   const [left, up, right, down] = [37, 38, 39, 40] → 방향키 keycode(키코드)를 쉽게 사용할 수 있게 변수를 사용해서 이름을 붙임
18   document.body.addEventListener('keydown', (event) => {
19     switch (event.keyCode) {
20       case left: → 키보드가 눌릴 때 실행
```

▶ 다음 쪽에 코드 이어짐

## SECTION 7-2 이벤트 활용(7)

- 키보드 키 코드 사용하기
  - 키로 별 움직이기 (소스 코드 7-2-3.html)

◀ 앞쪽에 이어

```
21     x -= 1
22     break
23     case up:
24         y -= 1
25         break
26     case right:
27         x += 1
28         break
29     case down:
30         y += 1
31         break
32 }
33 print()
34 })
35 })
36 </script>
37 <body>
38 <h1>★</h1>
39 </body>
```



## SECTION 7-2 이벤트 활용(8)

- 이벤트 발생 객체

- 이벤트 리스너 내부에서 어떤 변수에 접근할 수 없는 경우
- 다음 코드에서는 listener() 함수 내부에서 textarea 변수에 접근할 수 없어 오류가 발생
  - 이벤트 리스너를 외부로 빼낸 경우

---

```
<script>
const listener = (event) => {
  const length = textarea.value.length  —————>  현재 블록에서는 textarea 변수를 사용할 수 없음
  h1.textContent = `글자 수: ${length}`
}

document.addEventListener('DOMContentLoaded', () => {  —————>  이벤트 리스너가 외부로 분리
  const textarea = document.querySelector('textarea')
  const h1 = document.querySelector('h1')
  textarea.addEventListener('keyup', listener)
})
</script>
```

---



## SECTION 7-2 이벤트 활용(09)

- 이벤트 발생 객체

- 문제 해결

- 1) event.currentTarget 속성을 사용

---

```
<script>
const listener = (event) => {
  const length = event.currentTarget.value.length  —————> event.currentTarget가 textarea가 됨
  h1.textContent = `글자 수: ${length}`
}
document.addEventListener('DOMContentLoaded', () => {
  const textarea = document.querySelector('textarea')
  const h1 = document.querySelector('h1')
  textarea.addEventListener('keyup', listener)
})
</script>
```

---

## SECTION 7-2 이벤트 활용(10)

- 이벤트 발생 객체

- 문제 해결

- 2) this 키워드를 사용

---

```
<script>
const listener = function (event) {
  const length = this.value.length
  h1.textContent = `글자 수: ${length}`
}
document.addEventListener('DOMContentLoaded', () => {
  const textarea = document.querySelector('textarea')
  const h1 = document.querySelector('h1')
  textarea.addEventListener('keyup', listener)
})
</script>
```

---

→ this가 textarea가 됨

## SECTION 7-2 이벤트 활용(11)

### ◦ 글자 입력 양식 이벤트

- 입력 양식을 기반으로 inch를 cm 단위로 변환하는 프로그램 (소스 코드 7-2-4.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const input = document.querySelector('input')
04   const button = document.querySelector('button')
05   const p = document.querySelector('p')
06
07   button.addEventListener('click', () => {
08     // 입력을 숫자로 변환합니다.
09     const inch = Number(input.value)
10     // 숫자가 아니라면 바로 리턴합니다.
11     if (isNaN(inch)) {
12       p.textContent = '숫자를 입력해주세요'
13       return
14     }
15     // 변환해서 출력합니다.
16     const cm = inch * 2.54
17     p.textContent = `${cm} cm`
18   })
19 }
20 </script>
21 <body>
22 <input type="text"> inch<br>
23 <button>계산</button>
24 <p></p>
25 </body>
```

조기 리턴 부분



## SECTION 7-2 이벤트 활용(12)

### ◦ 글자 입력 양식 이벤트

- 인터넷에서 특정 사이트에 가입할 때 이메일과 전화번호 유효성 등을 검사
- 일반적으로 이런 유효성 검사를 할 때에는 정규 표현식regular expression을 사용
- 이메일 형식 확인하기 (소스 코드 7-2-5.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const input = document.querySelector('input')
04   const p = document.querySelector('p')
05   const isEmail = (value) => {
06     // 골뱅이를 갖고 있고 && 골뱅이 뒤에 점이 있다면
07     return (value.indexOf('@') > 1)
08     && (value.split('@')[1].indexOf('.') > 1)
09   }
10
11   input.addEventListener('keyup', (event) => {
12     const value = event.currentTarget.value
13     if (isEmail(value)) {
14       p.style.color = 'green'
15       p.textContent = `이메일 형식입니다: ${value}`
```

→ 이메일인지 검사하는 함수

▶ 다음 쪽에 코드 이어짐

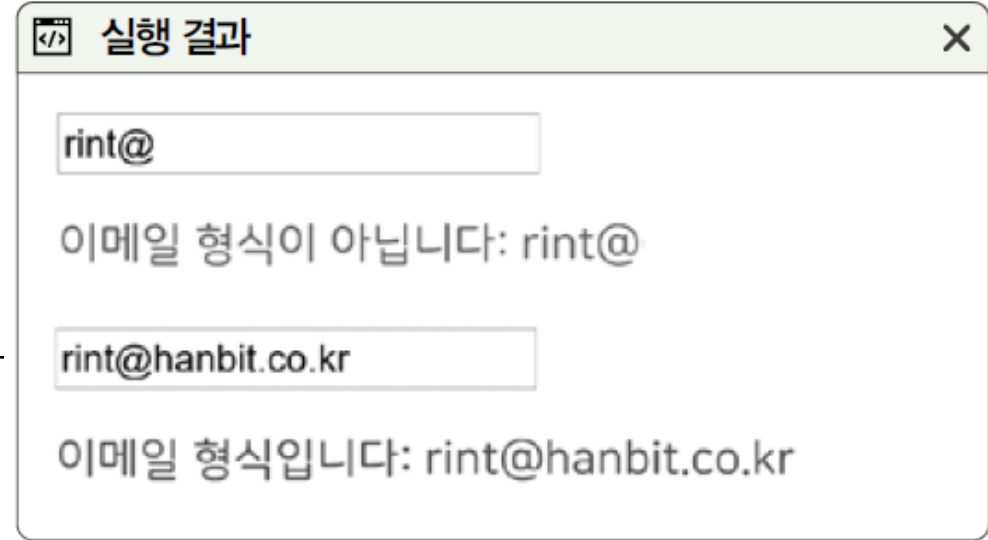
## SECTION 7-2 이벤트 활용(13)

### ◦ 글자 입력 양식 이벤트

- 인터넷에서 특정 사이트에 가입할 때 이메일과 전화번호 유효성 등을 검사
- 이메일 형식 확인하기 (소스 코드 7-2-5.html)

◀ 앞쪽에 이어

```
16   } else {  
17     p.style.color = 'red'  
18     p.textContent = `이메일 형식이 아닙니다: ${value}`  
19   }  
20 }  
21 }  
22 </script>  
23 <body>  
24   <input type="text">  
25   <p></p>  
26 </body>
```



## SECTION 7-2 이벤트 활용(14)

### ◦ 글자 입력 양식 이벤트

- 드롭다운 목록 활용하기: select 태그
- 기본 select 태그 (소스 코드 7-2-6.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const select = document.querySelector('select')
04   const p = document.querySelector('p')
05
06   select.addEventListener('change', (event) => {
07     const options = event.currentTarget.options
08     const index = event.currentTarget.options.selectedIndex
09
10     p.textContent = `선택: ${options[index].textContent}`
11   })
12 })
13 </script>
14 <body>
15 <select>
16   <option>떡볶이</option>
17   <option>순대</option>
18   <option>오뎅</option>
19   <option>튀김</option>
20 </select>
21 <p>선택: 떡볶이</p> → 처음에 떡볶이가 선택되어 있도록 초깃값을 지정
22 </body>
```

선택한 option 태그를 추출



## SECTION 7-2 이벤트 활용(15)

- 글자 입력 양식 이벤트

- multiple select 태그 소스 코드 7-2-7.html

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const select = document.querySelector('select')
04   const p = document.querySelector('p')
05
06   select.addEventListener('change', (event) => {
07     const options = event.currentTarget.options
08     const list = []
09     for (const option of options) {
10       if (option.selected) {
11         list.push(option.textContent)
12       }
13     }
14     p.textContent = `선택: ${list.join(',')}`
15   })
16 })
17 </script>
```

options 속성에는 forEach() 메소드가 없음  
따라서 이렇게 반복문으로 돌려야 함

selected 속성을 확인

▶ 다음 쪽에 코드 이어짐

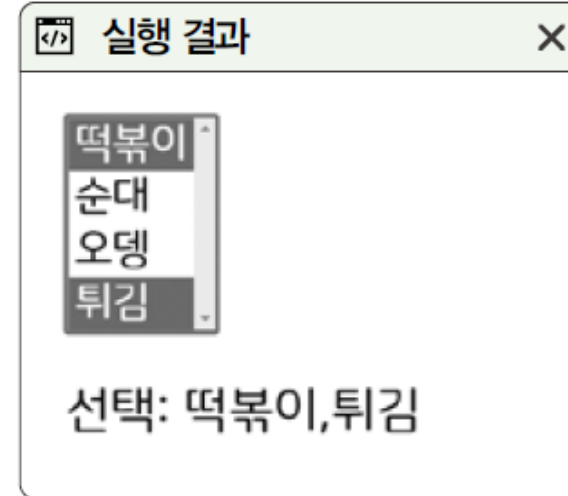
## SECTION 7-2 이벤트 활용(16)

- 글자 입력 양식 이벤트

- multiple select 태그 소스 코드 7-2-7.html

◀ 앞쪽에 이어

```
18 <body>
19 <select multiple>
20 <option>떡볶이</option>
21 <option>순대</option>
22 <option>오뎅</option>
23 <option>튀김</option>
24 </select>
25 <p></p>
26 </body>
```





## SECTION 7-2 이벤트 활용(17)

- 글자 입력 양식 이벤트

- cm 단위를 여러 단위로 변환하는 프로그램 (소스 코드 7-2-8.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   let 현재값
04   let 변환상수 = 10
05
06   const select = document.querySelector('select')
07   const input = document.querySelector('input')
08   const span = document.querySelector('span')
09
10   const calculate = () => {
11     span.textContent = (현재값 * 변환상수).toFixed(2)
12   }
13
14   select.addEventListener('change', (event) => {
15     const options = event.currentTarget.options
16     const index = event.currentTarget.options.selectedIndex
17     변환상수 = Number(options[index].value)
18     calculate()
19   })
```

↑ 소수점 2번째 자리까지 출력

→ 항목을 선택하면 항목의 value 속성을 추출

▶ 다음 쪽에 코드 이어짐

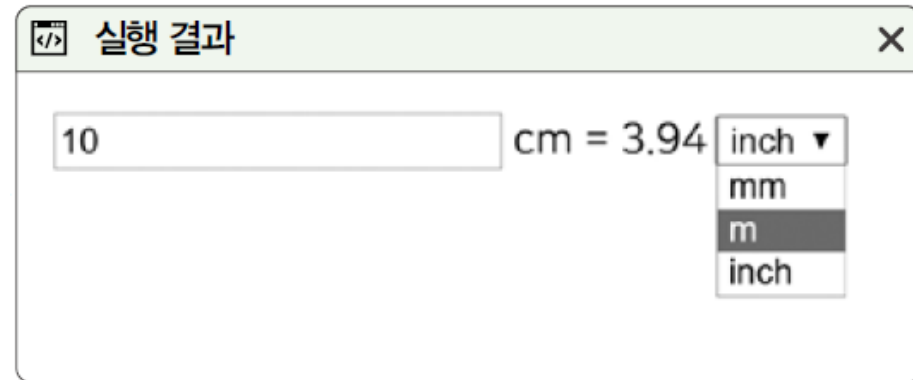
## SECTION 7-2 이벤트 활용(18)

- 글자 입력 양식 이벤트

- cm 단위를 여러 단위로 변환하는 프로그램 (소스 코드 7-2-8.html)

◀ 앞쪽에 이어

```
20
21 input.addEventListener('keyup', (event) => {
22     현재값 = Number(event.currentTarget.value) → 값을 입력하면 현재 값을 추출
23     calculate()
24 })
25 })
26 </script>
27 <body>
28 <input type="text"> cm =
29 <span></span>
30 <select>
31   <option value="10">mm</option>
32   <option value="0.01">m</option>
33   <option value="0.393701">inch</option>
34 </select>
35 </body>
```



## SECTION 7-2 이벤트 활용(19)

- 글자 입력 양식 이벤트

- 체크 박스 활용하기 소스 코드 7-2-9.html

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   let [timer, timerId] = [0, 0]
04   const h1 = document.querySelector('h1')
05   const checkbox = document.querySelector('input')
06
07   checkbox.addEventListener('change', (event) => {
08     if (event.currentTarget.checked) {
09       // 체크 상태 → checked 속성을 사용
10       timerId = setInterval(() => {
11         timer += 1
12         h1.textContent = `${timer}초`
13       }, 1000)
14     } else {
15       // 체크 해제 상태
16       clearInterval(timerId)
17     }
18   })
19 })
20 </script>
21 <body>
22   <input type="checkbox">
23   <span>타이머 활성화</span>
24   <h1></h1>
25 </body>
```



## SECTION 7-2 이벤트 활용(20)

- 글자 입력 양식 이벤트
  - 라디오 버튼 사용해보기 (소스 코드 7-2-10.html)

---

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03     // 문서 객체 추출하기
04     const output = document.querySelector('#output')
05     const radios = document.querySelectorAll('[name=pet]')
06
07     // 모든 라디오 버튼에
08     radios.forEach((radio) => {
09         // 이벤트 연결
10         radio.addEventListener('change', (event) => {
11             const current = event.currentTarget
12             if (current.checked) {
13                 output.textContent = `좋아하는 애완동물은 ${current.value}이시군요!`
14             }
15         })
16     })
17 })
18 </script>
```

---

▶ 다음 쪽에 코드 이어짐

## SECTION 7-2 이벤트 활용(21)

### ◦ 글자 입력 양식 이벤트

#### ▪ 라디오 버튼 사용해보기 (소스 코드 7-2-10.html)

##### ◀ 앞쪽에 이어

```
19 <body>
20 <h3># 좋아하는 애완동물을 선택해주세요</h3>
21 <input type="radio" name="pet" value="강아지">
22 <span>강아지</span>
23 <input type="radio" name="pet" value="고양이">
24 <span>고양이</span>
25 <input type="radio" name="pet" value="햄스터">
26 <span>햄스터</span>
27 <input type="radio" name="pet" value="기타">
28 <span>기타</span>
29 <hr>
30 <h3 id="output"></h3>
31 </body>
```

라디오 버튼을 하나씩만 선택하려면  
name 속성을 동일하게 입력해 그룹으로 만들

## SECTION 7-2 이벤트 활용(22)

- 기본 이벤트 막기

- 기본 이벤트: 어떤 이벤트가 발생했을 때 웹 브라우저가 기본적으로 처리해주는 것
- 이미지 마우스 오른쪽 버튼 클릭 막기 (소스 코드 7-2-11.html)

---

```
01 <script>
02  document.addEventListener('DOMContentLoaded', () => {
03    const imgs = document.querySelectorAll('img')
04
05    imgs.forEach((img) => {
06      img.addEventListener('contextmenu', (event) => {
07        event.preventDefault()  —————>  컨텍스트 메뉴를 출력하는 기본 이벤트를 제거
08      })
09    })
10  })
11 </script>
12 <body>
13  
14 </body>
```

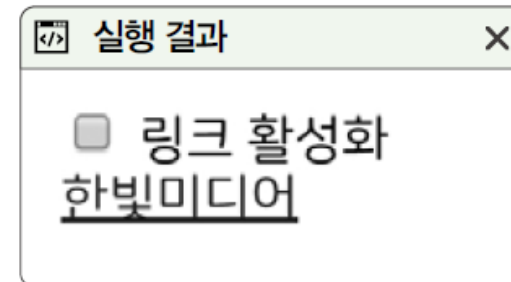
---

## SECTION 7-2 이벤트 활용(23)

### ◦ 기본 이벤트 막기

#### ▪ 체크 때만 링크 활성화하기 (소스 코드 7-2-12.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   let status = false
04
05   const checkbox = document.querySelector('input')
06   checkbox.addEventListener('change', (event) => {
07     status = event.currentTarget.checked → checked 속성을 사용
08   })
09
10   const link = document.querySelector('a')
11   link.addEventListener('click', (event) => {
12     if (!status) {
13       event.preventDefault → () status가 false가 아니면 링크의 기본 이벤트를 제거
14     }
15   })
16 })
17 </script>
18 <body>
19 <input type="checkbox">
20 <span>링크 활성화</span>
21 <br>
22 <a href="http://hanbit.co.kr">한빛미디어</a>
23 </body>
```



## SECTION 7-2 이벤트 활용(24)

- 할 일 목록 만들기[누적 예제]

- 할 일 목록 만들기 (소스 코드 7-2-13.html)

```
01 <body>
02 <h1>할 일 목록</h1>
03 <input id="todo">
04 <button id="add-button">추가하기</button>
05 <div id="todo-list">
06
07 </div>
08 </body>
09 <script>
10 document.addEventListener('DOMContentLoaded', () => {
11   // 문서 객체를 가져옵니다.
12   const input = document.querySelector('#todo')
13   const todoList = document.querySelector('#todo-list')
14   const addButton = document.querySelector('#add-button')
15
16   // 변수를 선언합니다.
17   let keyCount = 0
18
19   // 함수를 선언합니다.
20   const addTodo = () => {
```

이후에 removeTodo() 함수에서 문서 객체를 쉽게 제거하기 위한 용도로 만든 변수

▶ 다음 쪽에 코드 이어짐



## SECTION 7-2 이벤트 활용(25)

- 할 일 목록 만들기[누적 예제]
  - 할 일 목록 만들기 (소스 코드 7-2-13.html)

◀ 앞쪽에 이어

```
21 // 입력 양식에 내용이 없으면 추가하지 않습니다.
22 if (input.value.trim() === '') {
23     alert('할 일을 입력해주세요.')
24     return
25 }
26
27 // 문서 객체를 설정합니다.
28 const item = document.createElement('div')
29 const checkbox = document.createElement('input')
30 const text = document.createElement('span')
31 const button = document.createElement('button')
32
33 // 문서 객체를 식별할 키를 생성합니다.
34 const key = keyCount
35 keyCount += 1
36
```

이후에 `removeTodo()` 함수에서 문서 객체를 쉽게 제거하기 위한 용도로 만든 변수

## SECTION 7-2 이벤트 활용(26)

### ◦ 할 일 목록 만들기[누적 예제]

#### ▪ 할 일 목록 만들기 (소스 코드 7-2-13.html)

◀ 앞쪽에 이어

```
37 // item 객체를 조작하고 추가합니다.
38 item.setAttribute('data-key', key)
39 item.appendChild(checkbox)
40 item.appendChild(text)
41 item.appendChild(button)
42 todoList.appendChild(item)
43
44 // checkbox 객체를 조작합니다.
45 checkbox.type = 'checkbox'
46 checkbox.addEventListener('change', (event) => {
47   item.style.textDecoration
48     = event.target.checked ? 'line-through' : ''
49 })
50
51 // text 객체를 조작합니다.
52 text.textContent = input.value
53
54 // button 객체를 조작합니다.
55 button.textContent = '제거하기'
56 button.addEventListener('click', () => {
57   removeTodo(key)
58 })
```

→ `<div data-key="숫자">  
<input>  
<span></span>  
<button></button>  
</div>  
형태를 구성`

→ `<input type="checkbox">  
형태를 구성`

→ 체크 박스를 클릭하면 선을 그어줌

→ `<span>글자</span>  
형태를 구성`

→ `<button>제거하기</button>  
형태를 구성`

## SECTION 7-2 이벤트 활용(27)

### ◦ 할 일 목록 만들기[누적 예제]

#### ▪ 할 일 목록 만들기 (소스 코드 7-2-13.html)

◀ 앞쪽에 이어

```
59
60 // 입력 양식의 내용을 비웁니다.
61 input.value = ""
62 }
63
64 const removeTodo = (key) => {
65   // 식별 키로 문서 객체를 제거합니다.
66   const item = document.querySelector(`[data-key="${key}"]`)
67   todoList.removeChild(item)
68 }
69
70 // 이벤트 연결
71 addButton.addEventListener('click', addTodo)
72 input.addEventListener('keyup', (event) => {
73   // 입력 양식에서 Enter 키를 누르면 바로 addTodo() 함수를 호출합니다.
74   const ENTER = 13
75   if (event.keyCode === ENTER) {
76     addTodo()
77   }
78 })
79 })
80 </script>
```

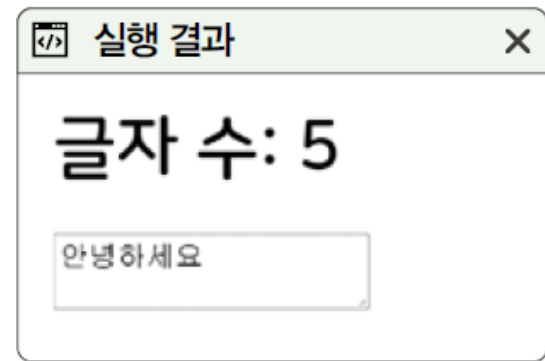
위에서 지정한 <div data-key="숫자">를  
기반으로 요소를 찾고 제거



# [좀 더 알아보기①] 타이머로 구현한 남은 글자 수 세기

- 글자 수 출력하기 소스 코드 7-2-14.html

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const textarea = document.querySelector('textarea')
04   const h1 = document.querySelector('h1')
05   let timerId
06
07   textarea.addEventListener('focus', (event) => { → 입력 양식 활성화
08     timerId = setInterval(() => {
09       const length = textarea.value.length
10       h1.textContent = `글자 수: ${length}`
11     }, 50)
12   })
13   textarea.addEventListener('blur', (event) => { → 입력 양식 비활성화
14     clearInterval(timerId)
15   })
16 })
17 </script>
18 <body>
19 <h1></h1>
20 <textarea></textarea>
21 </body>
```



## [좀 더 알아보기②] localStorage 객체

- 웹 브라우저에 데이터를 저장하는 localStorage 객체와 활용
  - localStorage.getItem(키): 저장된 값을 추출. 없으면 undefined가 나옴.  
객체의 속성을 추출하는 일반적인 형태로 localStorage.키 또는 localStorage[키] 형태로 사용 할 수도 있음
  - localStorage.setItem(키, 값): 값을 저장  
이전과 마찬가지로 객체에 속성을 지정하는 일반적인 형태를 사용할 수도 있음
  - localStorage.removeItem(키): 특정 키의 값을 제거
  - localStorage.clear(): 저장된 모든 값을 제거

## [좀 더 알아보기②] localStorage 객체

- 웹 브라우저에 데이터를 저장하는 localStorage 객체와 활용하기 (소스 코드 7-2-15.html)

```
01 <script>
02 document.addEventListener('DOMContentLoaded', () => {
03   const p = document.querySelector('p')
04   const input = document.querySelector('input')
05   const button = document.querySelector('button')
06
07   const savedValue = localStorage.getItem('input')
08   // localStorage.input도 가능합니다.
09   if (savedValue) {
10     input.value = savedValue
11     p.textContent = `이전 실행 때의 마지막 값: ${savedValue}`
12   }
13
14   input.addEventListener('keyup', (event) => {
15     const value = event.currentTarget.value
```

→ 값을 읽을 때는 getItem() 메소드를 사용

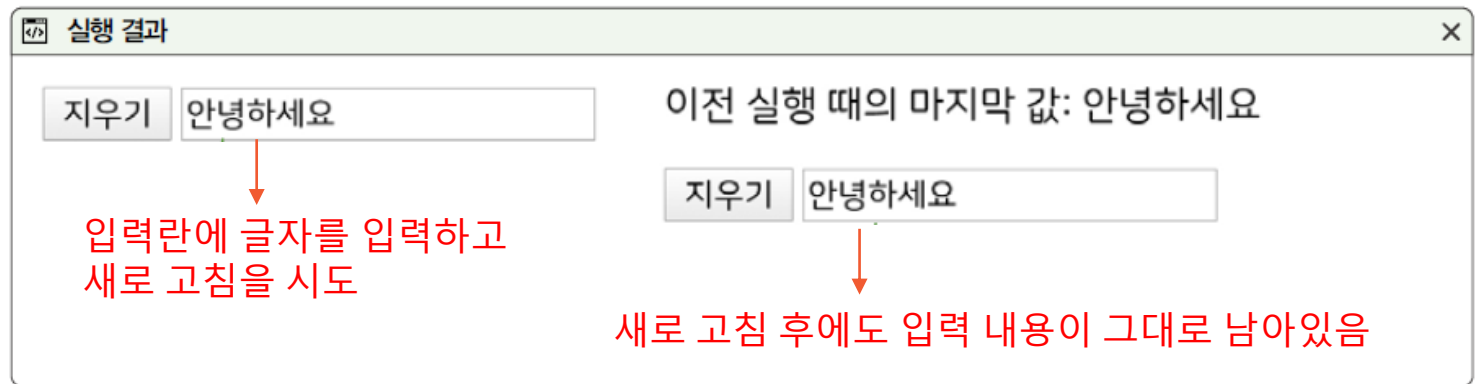
▶ 다음 쪽에 코드 이어짐

## [좀 더 알아보기②] localStorage 객체

- 웹 브라우저에 데이터를 저장하는 localStorage 객체와 활용하기 (소스 코드 7-2-15.html)

◀ 앞쪽에 이어

```
16  localStorage.setItem('input', value)  —————> 값을 저장할 때는 setItem() 메소드를 사용
17  // localStorage.input = value도 가능합니다.
18  })
19
20  button.addEventListener('click', (event) => {
21    localStorage.clear()  —————> 값을 모두 제거할 때는 clear() 메소드를 사용
22    input.value = ""
23  })
24  })
25 </script>
26 <body>
27 <p></p>
28 <button>지우기</button>
29 <input type="text">
30 </body>
```



## [요점 정리]

- 3가지 키워드로 정리하는 핵심 포인트
  - 이벤트 모델은 이벤트를 연결하는 방법을 의미
  - 이벤트 객체는 이벤트 리스너의 첫 번째 매개변수로 이벤트와 관련된 정보가 들어 있음
  - 이벤트 발생 객체는 이벤트를 발생시킨 객체를 의미  
이벤트 객체의 `currentTarget` 속성을 사용해서 확인할 수 있음