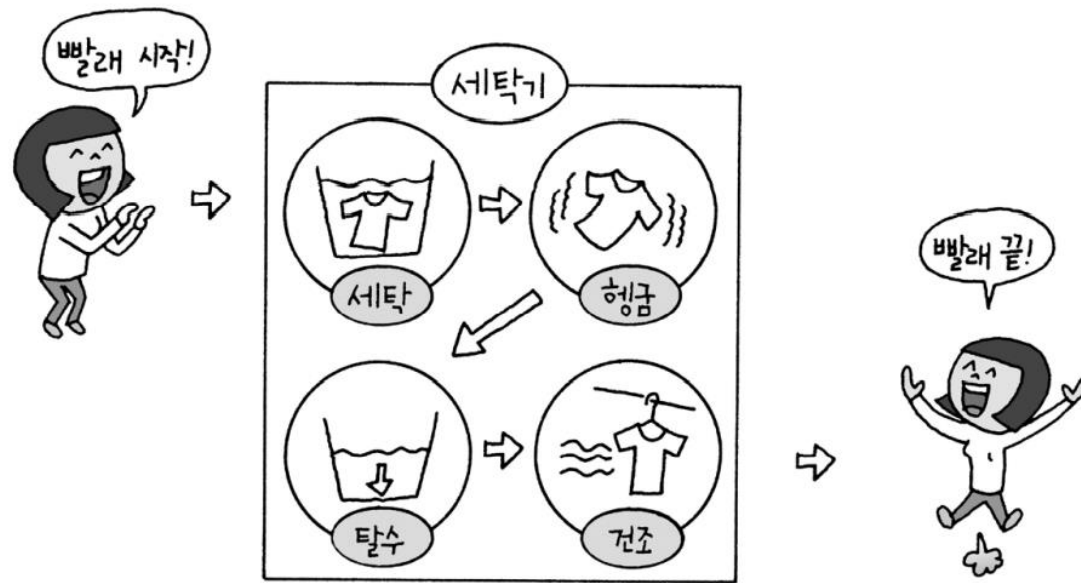


## 05. 함수와 이벤트

# 여러 동작을 묶은 덩어리, 함수

- 빨래처럼 자주 실행해야 하는 기능에 포함된 명령이 여러 가지일 때 그 명령을 한 번에 실행할 수 있게 한 덩어리로 묶을 수 있음
- 세탁, 헹굼, 탈수, 건조 과정을 한 번에 처리할 수 있게 만들어진 세탁기



# 여러 동작을 묶은 덩어리, 함수

함수(function)를 사용하면 무엇이 좋을까?

- 각 명령의 시작과 끝을 명확하게 구별할 수 있다.
- 함수에 별도의 이름을 붙이면 같은 기능이 필요할 때마다 해당 함수를 실행할 수 있다.

함수 선언 (함수 정의)

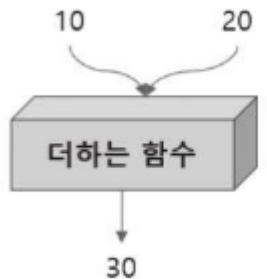
- 함수가 어떤 명령을 처리해야 할지 미리 알려주는 것
- function 예약어를 사용하고, { } 안에 실행할 명령을 작성

```
> function addNumber( ) {  
    var sum = 10 + 20;  
    console.log(sum);  
}
```

함수 호출 (함수 실행)

- 함수 이름을 사용해 함수 실행

```
> addNumber( )  
30
```



# 함수 선언은 어디에 두어야 할까

함수를 선언해 놓기만 하면, 선언한 위치와 상관없이 함수를 실행할 수 있습니다

## 함수 선언과 실행순서

- 웹 브라우저에서 자바스크립트 소스를 해석할 때 함수 선언 부분을 가장 먼저 해석합니다.
- 개발자가 원하는 어느 곳이든 함수를 선언해 놓기만 하면, 선언한 위치와 상관없이 함수를 실행할 수 있습니다.

```
addNumber( );  
  
function addNumber( ) {  
    .....  
}
```

함수 호출 소스를 선언 소스보다 앞에 작성한 예

```
function addNumber( ) {  
    .....  
}  
  
addNumber( );
```

함수 호출 소스를 선언 소스 뒤에 작성한 예

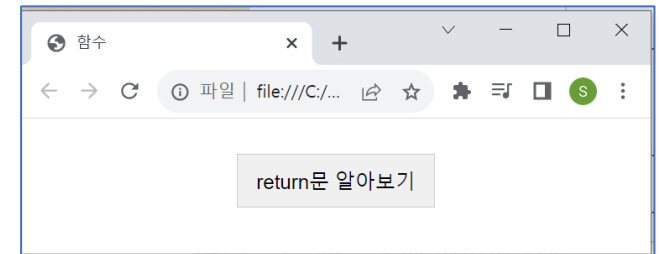
# 버튼으로 함수를 실행하는 프로그램 만들기

- 웹 문서에 있는 버튼 모양을 눌렀을 때 결과가 나타나거나, 마우스를 옮기면 다른 내용이 나타나는 웹 사이트에서 많이 보았습니다
- 사용자의 동작에 따라 함수를 실행하기 [덧셈 계산하기]

```
1  <!DOCTYPE html>
2  <html lang="ko">
3      <head>
4          <meta charset="UTF-8">
5          <title>함수</title>
6          <link rel="stylesheet" href="css/add.css">
7      </head>
8      <body>
9          <button>덧셈 계산하기</button>
10
11      </body>
12 </html>
```

add.css

```
1  button {
2      border:1px solid #ccc;
3      padding:10px 15px;
4      margin:20px auto;
5      font-size:16px;
6  }
7
8  body {
9      text-align: center;
10 }
```



# 버튼으로 함수를 실행하는 프로그램 만들기

```
1  <!DOCTYPE html>
2  <html lang="ko">
3    <head>
4      <title>함수</title>
5      <link rel="stylesheet" href="css/add.css">
6    </head>
7    <body>
8      <button onclick="addNumber()">덧셈 계산하기</button>
9
10     <script src="js/add-result.js"></script>
11   </body>
12 </html>
```

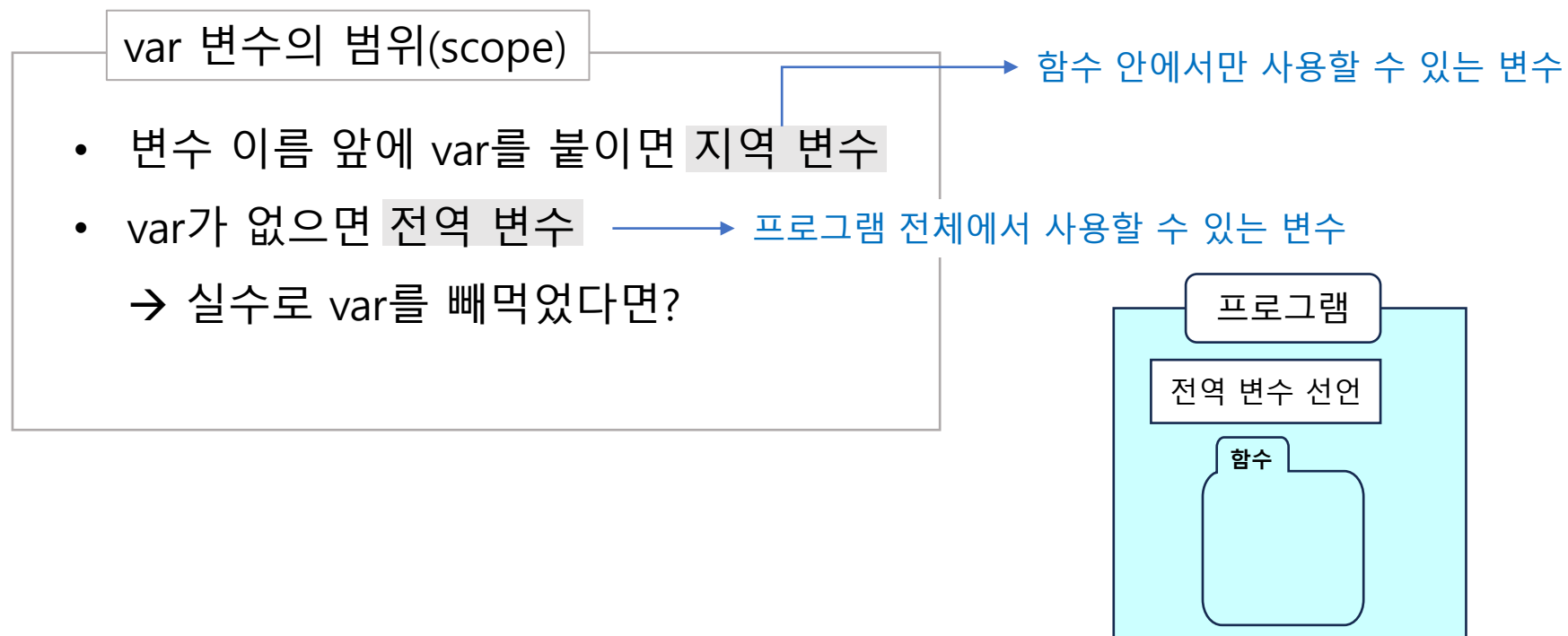
onclick은 버튼을 눌렀을 때 실행할 대상인 함수를 알려주는 예약어로, onclick 뒤에는 실행할 함수이름을 작성합니다.  
addNumber() 함수를 실행합니다.

## add-result.js

```
1  function addNumber() {
2    var sum = 10 + 20;
3    alert(sum);
4  }
```

# let과 const

- let, const – ES6 버전 이후에 변수를 선언하는 새로운 예약어
- var가 있는데 왜 let과 const라는 예약어가 필요할까?



# let과 const

var 변수의 호이스팅

변수를 선언하기 전에 변수를 사용하면?

→ 오류가 생기지 않는다

→ 변수 선언이 앞에 있는 것처럼 끌어올려(hoisting) 인식한다.

전역변수 선언

```
1 var x = 100;
2
3 test();
4
5 function test() {
6   document.write("x is " + x + ", y is " + y);
7   var y = 200;
8 }
```

지역변수 선언





# let과 const

var 변수의 재선언

이미 있는 변수를 다시 선언할 수 있다

→ 실수로 서로 다른 위치에서 같은 변수를 선언할 수 있다는 것은 문제

→ 재선언하면 이전 변수를 덮어쓰기 때문에 예상하지 못한 오류가 생길 수 있음.

# let과 const

- let – 프로그램 안에서 값이 변하는 변수
- const – 프로그램 안에서 값이 변하지 않는 변수

## 변수의 범위(scope)

let 변수와 const 변수는 블록 영역의 스코프

## 호이스팅 없음

변수를 선언하지 않고 사용하면 오류 발생

## 변수의 재선언 불가

같은 변수를 다시 선언하면 오류 발생

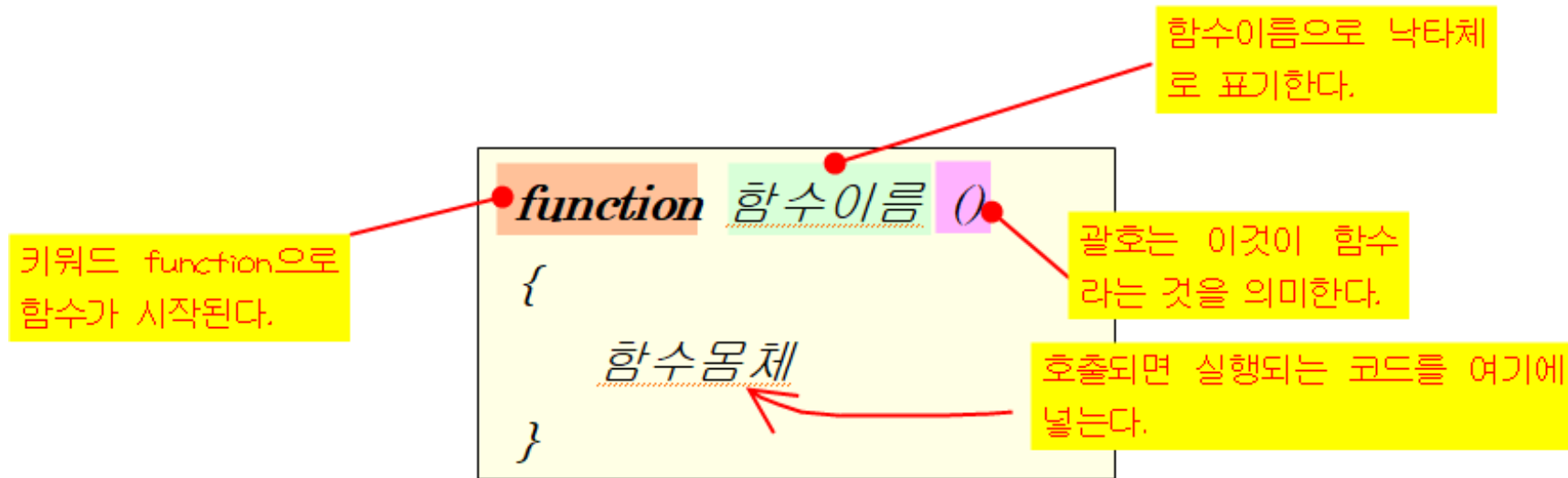
```
1  function calcSum(n) {  
2      let sum = 0;  // 블록 변수 선언  
3      for(let i = 1; i < n + 1; i++) {  
4          sum += i;  
5      }  
6      console.log(sum);  // 블록 변수 사용  
7  }  
8  
9  calcSum(10);
```

# 변수는 이렇게 사용하세요

- 전역 변수는 최소한으로 사용한다
- var 변수는 함수의 시작 부분에서 선언한다 (호이스팅 방지)
- for문의 카운터 변수는 블록 변수(let)를 사용하는게 좋다
- ES6를 사용한 프로그램이라면 var보다 let을 사용한다

# 여러 번 사용할 수 있는 함수 만들기

- 함수는 입력을 받아서 특정한 작업을 수행하여서 결과를 반환하는 블랙 박스
- 함수의 구조



# 여러 번 사용할 수 있는 함수 만들기

- 기본 함수 정의 및 호출

```
function myFnc( ) {
```

```
    document.write( "hello~", "<br>" );  
    document.write( "welcome", "<br>" );
```

```
}
```

```
myFnc( );
```

```
myFnc( );
```

함수가 2회 호출되어 코드  
내용을 2회 실행합니다.

# 여러 번 사용할 수 있는 함수 만들기

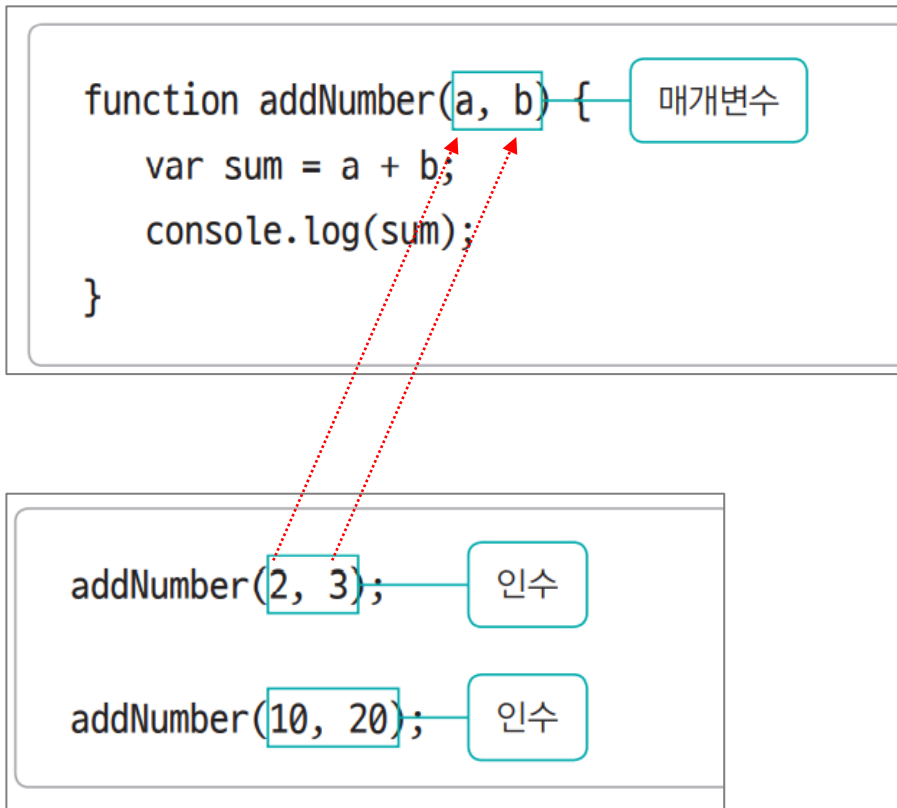
## 매개변수(parameter)

- 함수를 실행하기 위해 필요하다고 지정하는 값
- 함수를 선언할 때 함수 이름 옆의 괄호 안에 매개변수 이름을 넣음

## 인수(argument)

- 함수를 실행하기 위해 필요하다고 지정하는 값
- 함수를 실행할 때 매개변수로 넘겨주는 값

```
function addNumber(a, b) {  
    var sum = a + b;  
    console.log(sum);  
}
```



The diagram illustrates the relationship between function parameters and arguments. In the top box, the function `addNumber(a, b)` is shown. The parameters `a` and `b` are highlighted with blue boxes, and a label '매개변수' (parameter) points to them. In the bottom box, two function calls are shown: `addNumber(2, 3);` and `addNumber(10, 20);`. The arguments `(2, 3)` and `(10, 20)` are highlighted with blue boxes, and a label '인수' (argument) points to them. Red dotted arrows connect the arguments to the parameters: from `2` to `a`, from `3` to `b`, from `10` to `a`, and from `20` to `b`.

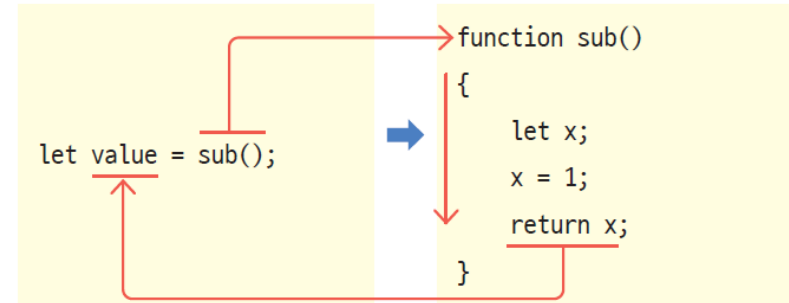
`addNumber(2, 3);`

`addNumber(10, 20);`

# 여러 번 사용할 수 있는 함수 만들기

## return 문

- 함수를 실행한 결과값을 함수 밖으로 넘기는 문
- 반환된 값은 함수를 실행한 소스 위치로 넘겨짐.

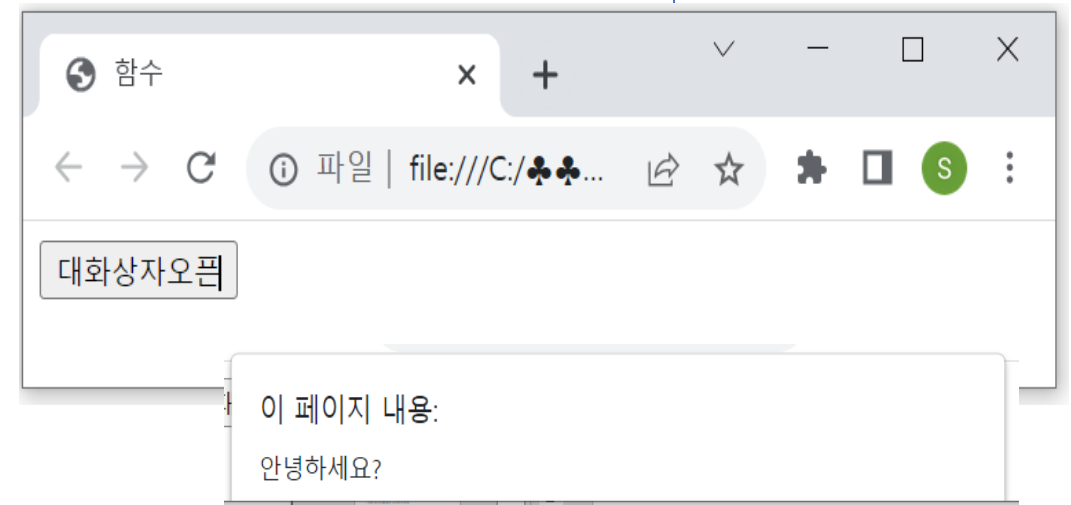
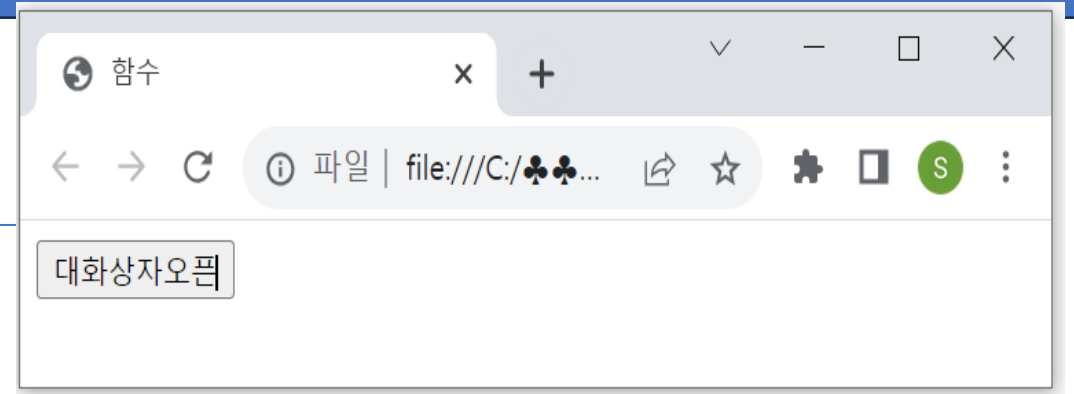


```
1  var num1 = parseInt(prompt("첫 번째 숫자는? ")); ❶
2  var num2 = parseInt(prompt("두 번째 숫자는? "));
3  var result = addNumber(num1, num2); ❷
4  alert("두 수를 더한 값은 " + result + "입니다."); ❸
5
6  ❹
7  function addNumber(a, b) { ❺
8    var sum = a + b; ❻
9    return sum; ❼
10 }
```

- ❶ num1 변수와 num2 변수에 값을 저장합니다.
- ❷ num1과 num2 값을 가지고 addNumber( ) 함수를 호출합니다.
- ❸ 함수 선언부로 넘어와 함수를 실행하는데, num1 값은 a 변수로, num2 값은 b 변수로 넘겨집니다.
- ❹ a 값과 b 값을 더해 sum 변수에 저장합니다.
- ❺ 결과값 sum을 반환합니다.
- ❻ 반환된 값을 변수 result에 저장합니다.
- ❼ result 변수값을 화면에 표시합니다.

# [실습-1] 함수

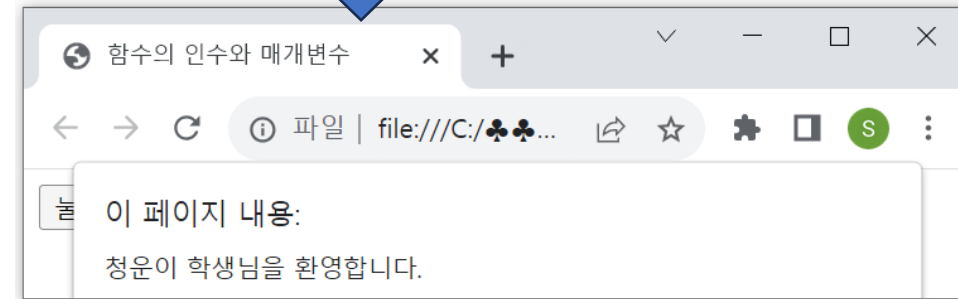
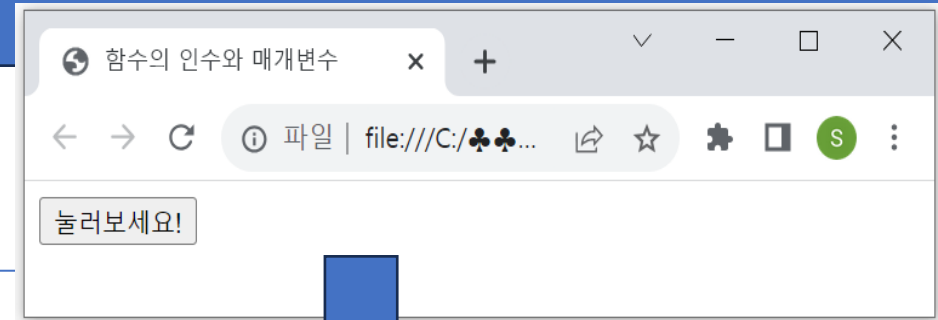
```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <title>함수</title>
6          <script>
7              function showDialog() {
8                  alert("안녕하세요?");
9              }
10         </script>
11     </head>
12     <body>
13         <button onclick="showDialog()">대화상자오픈</button>
14     </body>
15 </html>
```





## [실습-2] 인수와 매개변수

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>함수의 인수와 매개변수</title>
6      <script>
7          function greeting(name, position) {
8              alert(name + " " + position + "님을 환영합니다.");
9          }
10     </script>
11 </head>
12 <body>
13     <button onclick="greeting('청운이', '학생')">눌러보세요!</button>
14 </body>
15 </html>
```



# 함수 표현식

## 익명 함수

- 이름이 없는 함수
- 함수 자체가 '식(Expression)'이기 때문에 함수를 변수에 할당하거나 다른 함수의 매개변수로 사용할 수도 있음.

```
> var add = function(a, b) { //함수 선언 후 변수 add에 할당
    return a + b;
}
```

```
> var sum = add(10, 20); //익명 함수 실행 후 결과값을 변수 sum에 저장
> sum //변수 sum 값 확인
< 30
```

```
function showDialog() {
    alert("안녕하세요?");
}
```



```
let greeting = function()
{
    alert("안녕하세요?");
};
greeting();
```

# 함수 표현식

## 즉시 실행 함수

- 함수를 정의함과 동시에 실행하는 함수
- 즉시 실행 함수는 변수에 할당할 수 있고, 함수에서 반환하는 값을 변수에 할당할 수도 있음.

```
(function( ) {  
    .....  
} )( );
```

또는

```
(function( ) {  
    .....  
} )( ) ;
```

# 함수 표현식

## 즉시 실행 함수

매개변수와 인수를 사용한다면?

→ 매개변수는 function() 괄호 안에, 인수는 함수 끝에 있는 괄호 안에.

```
> var result = (function( ) {  
    return 10 + 20;  
})( );  
> console.log(result);  
< 30
```

```
> var result = (function(a, b) {           //매개변수 추가  
    return a + b;  
} (10, 20));    //인수 추가  
> console.log(result);  
< 30
```

# 화살표 함수

ES6 이후 버전에는 => 표기법을 사용해 함수 선언을 간단하게 작성

- 화살표 함수를 사용하면 더 짧은 함수 구문을 작성할 수 있다. 화살표 함수는 자바 언어의 람다식과 유사하다

매개변수가 없을 때

```
const hi = function() {  
  return "안녕하세요?";  
}
```



```
const hi = () => { return "안녕하세요?" };
```



```
const hi = () => "안녕하세요? " ;
```

# 화살표 함수

매개변수가 있을 때

```
let hi = function(user) {  
  document.write(user + "님, 안녕하세요?");  
}
```



```
let hi = user => document.write(user + "님, 안녕하세요?");
```

```
let sum = function(a, b) {  
  return a + b;  
}
```



```
let sum = (a, b) => { return a + b }
```



```
let sum = (a, b) => a + b;
```

# 화살표 함수 ex)

```
function sub(a, b){  
  return a + b;  
}
```



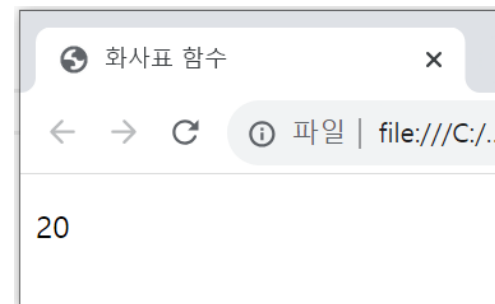
```
(a, b) => { return a+b; }  
또는  
(a, b) => return a+b;  
또는  
(a, b) => a + b;
```

```
function sub(a, b){  
  let data = 29;  
  return a + b + data;  
}
```



```
(a, b) => {  
  let data = 29;  
  return a + b + data;  
}
```

```
1  <!DOCTYPE html>  
2  <html lang="en">  
3    <head>  
4      <title> 화사표 함수</title>  
5    </head>  
6    <body>  
7      <p id="test"></p>  
8      <script>  
9        let func = (a, b) => a * b;  
10       document.getElementById("test").innerHTML = func(4, 5);  
11      </script>  
12    </body>  
13  </html>
```



# 객체 생성자 함수의 활용

- 객체 생성자 함수

- 내장 객체를 생성할 때는 이미 자바스크립트 엔진에 내장되어 있는 객체 생성자 함수를 사용하여 객체를 생성
- 객체 생성자 함수 선언과 객체를 생성하는 기본형

```
기본형 | function 함수명(매개변수1, 매개변수2, ...매개변수n) { //객체 생성자 함수
        this.속성명 = 새 값;
        this.함수명 = function() {
            자바스크립트 코드;
        }
    }

    var 참조 변수(인스턴스 네임) = new 함수명(); //객체 생성
        ↓
    var 참조 변수 = { 속성 : 새 값, 함수명 : function() { ... } }
```



# 객체 생성자 함수의 활용

- CheckWeight라는 이름으로 객체 생성자 함수를 선언하고 2개의 객체를 생성
  - 생성된 각각의 객체에는 속성(이름, 키, 몸무게)과 함수(getInfo(), getResult())를 등록

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>객체생성함수-키 몸무게</title>
6      <script>
7        // 객체 생성자 함수명은 대문자로 시작(개발자 간의 약속)
8        function CheckWeight(name, height, weight) {
9          this.userName = name;
10         this.userHeight = height;
11         this.userWeight = weight;
12         this.minWeight;
13         this.maxWeight;
14
15         this.getInfo = function() {
16           var str = "";
17           str += "이름: " + this.userName + ", ";
18           str += "키: " + this.userHeight + ", ";
19           str += "몸무게: " + this.userWeight + "<br>";
20           return str;
21         }
```

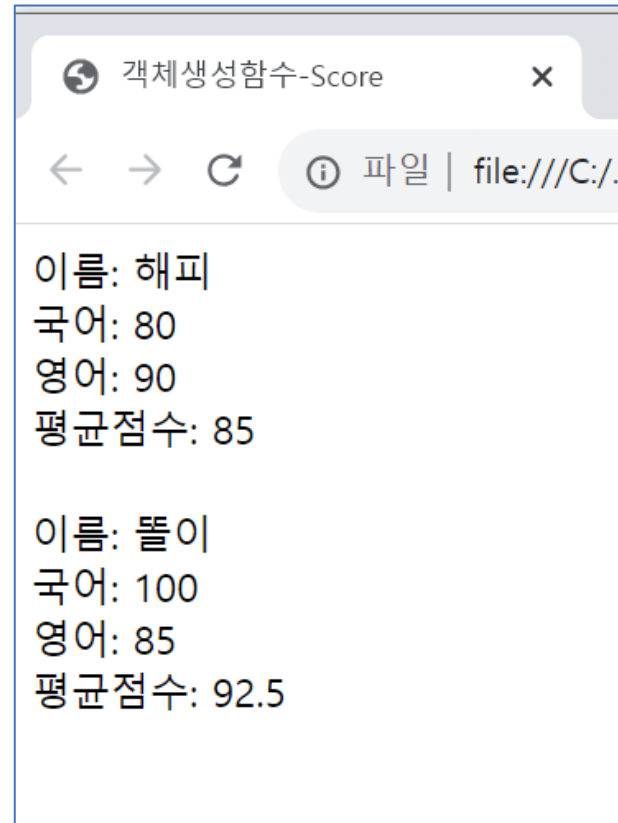
# 객체 생성자 함수의 활용

```
22
23     this.getResult = function() {
24         this.minWeight = (this.userHeight - 100) * 0.9 - 5;
25         this.maxWeight = (this.userHeight - 100) * 0.9 + 5;
26
27         if(this.userWeight >= this.minWeight && this.userWeight <= this.maxWeight) {
28             return "정상 몸무게입니다.<br>";
29         } else if(this.userWeight < this.minWeight) {
30             return "정상 몸무게보다 미달입니다.<br>";
31         } else {
32             return "정상 몸무게보다 초과입니다.<br>";
33         }
34     }
35 }
36
37 var jang = new CheckWeight("장보리", 168, 62);
38 var park = new CheckWeight("박달재", 188, 88);
39
40 console.log(jang);
41 console.log(park);
42
43 document.write(jang.getInfo());
44 document.write(jang.getResult());
45
46 document.write(park.getInfo());
47 document.write(park.getResult());
48 </script>
49 </head>
50 <body>
51 </body>
52 </html>
```

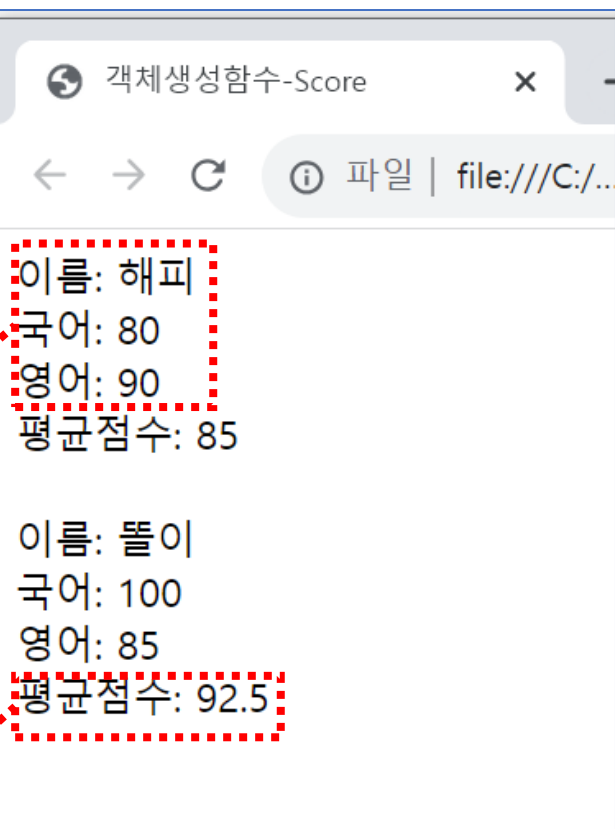
이름: 장보리, 키: 168, 몸무게: 62  
정상 몸무게입니다.  
이름: 박달재, 키: 188, 몸무게: 88  
정상 몸무게보다 초과입니다.





# [실습과제-1]

- TestScore 객체 생성자 함수 만들기
  - 두 학생(happy, ddolli)의 객체를 생성하고 속성으로 이름, 국어점수, 영어점수를 등록
  - 속성
    - userName, korNum, engNum
  - getTestInfo() getAvg() 함수



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>객체생성함수-Score</title>
6     <script>
7       function TestScore(name, kor, eng) {
8         
9
10
11
12       this.getTestInfo = function() {
13         
14
15
16       }
17
18       this.getAvg = function() {
19         
20
21       }
22     </script>
  </head>
</html>
```



```
23     var happy = new TestScore("해피", 80, 90);
24     var ddolli = new TestScore("돌이", 100, 85);
25
26     
27     document.write("평균점수: " +  "<br><br>");
28
29     
30     document.write("평균점수: " +  "<br><br>");
31 </script>
32 </head>
33 <body>
34 </body>
35 </html>
```

# 지역 변수와 전역 변수

- 함수 안에서 선언된 변수는 지역 변수(local variable)가 된다. 따라서 함수 안에서만 사용이 가능

```
function add(a, b) {  
  let sum = 0;  
  sum = a + b;  
  return sum;  
}
```

- 함수 외부에서 선언된 변수는 전역 변수(global variable) 임

```
let sum = 0;  
  
function add(a, b) {  
  sum = a + b;  
  return sum;  
}
```

# [미리보기] 버튼 클릭해서 상세 설명 열고 닫기



# 이벤트

-웹 페이지에서 상호작용이 발생하면 이벤트가 일어난다. 예를 들어 사용자가 버튼을 클릭하거나 특정 요소 위로 마우스 커서를 가져가면 이벤트가 발생한다

The image shows a web browser window with the address bar displaying `127.0.0.1:5500/eventtype.htm`. The page content includes the text "원하는 과일을 선택해주세요.", a large image of a red apple, and three radio buttons labeled "APPLE", "GRAPE", and "BANANA". Below the radio buttons is a button labeled "클릭" followed by a text input field. Red arrows point from various event descriptions to these elements:

- onmousemove 이벤트:** 마우스를 움직이면 발생한다. (Points to the text "원하는 과일을 선택해주세요.")
- onload 이벤트:** 문서 로딩 완료 시 발생한다. (Points to the browser window)
- onload 이벤트:** 이미지 로딩 완료 시 발생한다. (Points to the apple image)
- onchange 이벤트:** 라디오 버튼 선택 시에 발생한다. (Points to the "APPLE" radio button)
- onclick 이벤트:** 버튼을 누를 때 발생한다. (Points to the "클릭" button)
- onkeydown 이벤트:** 사용자가 키를 누를 때 발생한다. (Points to the text input field)



# 이벤트

## 이벤트

- 웹 브라우저나 사용자가 행하는 어떤 동작
- Ex) 웹 문서에서 키보드의 키를 누르는 것도 이벤트, 브라우저가 웹 페이지를 불러오는 것도 이벤트
- 웹 페이지를 읽어 오거나 링크를 누르는 것처럼 웹 문서 영역 안에서 이루어지는 동작만을 말함
- 사용자가 웹 문서 영역을 벗어나 하는 동작은 이벤트가 아님  
ex) 브라우저 창 맨 위의 제목 표시줄을 누르는 것은 이벤트가 아님..
- 자바스크립트 이벤트는 주로 마우스나 키보드를 사용할 때, 웹 문서를 불러올 때, 폼(Form)에 내용을 입력할 때 주로 발생

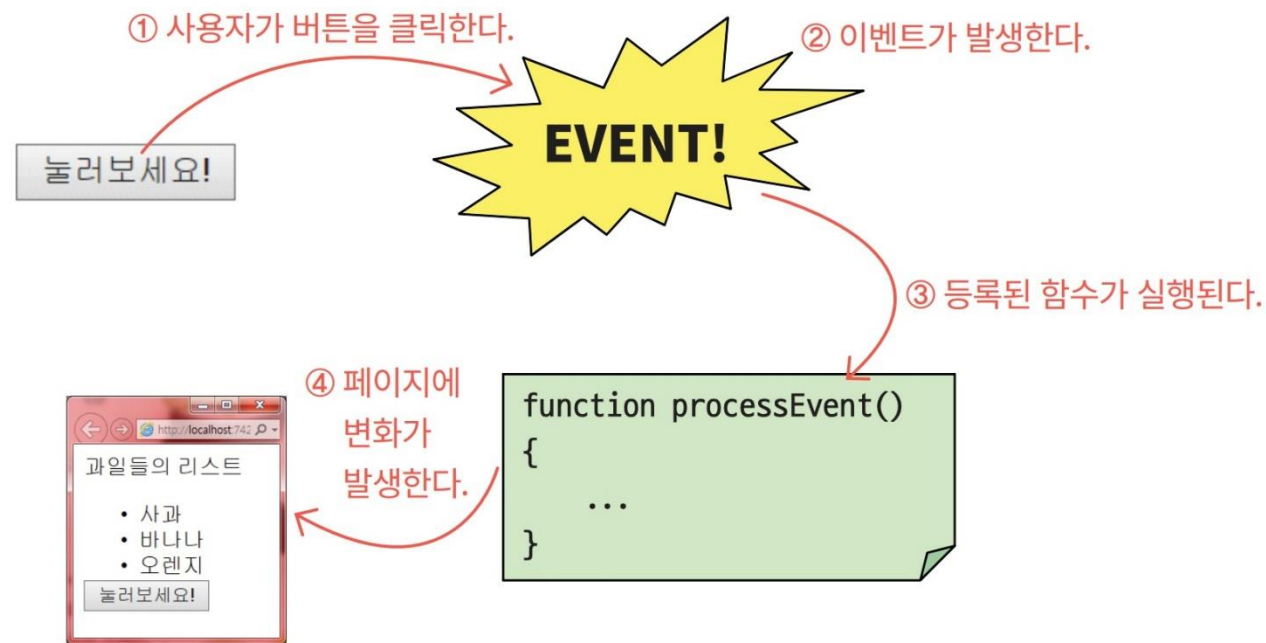
## 이벤트 처리기

- 이벤트가 발생했을 때 어떤 함수를 실행할지 알려줌
- 태그 안에서 이벤트를 처리할 때는 "on"+"이벤트명" 사용 (예, 클릭하면 onclick 사용)

Ex) 쇼핑몰 웹 페이지는 이미지를 누르면 확대된 옷 이미지를 보여주고, 상품 목록에서 항목을 선택하면 해당 상품의 페이지로 연결

# 이벤트 처리 절차

- 이벤트를 처리하는 코드를 버튼에 등록하여야 한다. 이벤트를 처리하는 코드를 이벤트 핸들러, 또는 이벤트 리스너라고 한다.



# 이벤트

## 마우스 이벤트

마우스 이벤트는 마우스에서 버튼이나 휠 버튼을 조작할 때 발생하는 이벤트입니다.

표 6-2 마우스 이벤트의 종류

| 이벤트       | 이벤트가 발생하는 순간               |
|-----------|----------------------------|
| click     | 사용자가 HTML 요소를 클릭했을 때       |
| dblclick  | 사용자가 HTML 요소를 더블클릭했을 때     |
| mousedown | 사용자가 요소에서 마우스 버튼을 눌렀을 때    |
| mousemove | 사용자가 요소에서 마우스 포인터를 움직일 때   |
| mouseover | 마우스 포인터를 요소 위로 옮길 때        |
| mouseout  | 마우스 포인터가 요소를 벗어날 때         |
| mouseup   | 요소 위에 올려놓은 마우스 버튼에서 손을 뗄 때 |

# 이벤트

- 마우스 이벤트 예제

다음은 버튼을 클릭했을 때 문서의 배경색을 바꾸는 예제입니다.

06\js\event-2.js

```
const button = document.querySelector("button");

button.onclick = function() {
  document.body.style.backgroundColor = "green";
}
```

# 이벤트

## 키보드 이벤트

키보드 이벤트는 키보드에서 특정 키를 조작할 때 발생하는 이벤트입니다.

표 6-3 키보드 이벤트의 종류

| 이벤트      | 이벤트가 발생하는 순간 |
|----------|--------------|
| keydown  | 키를 누르는 동안    |
| keypress | 키를 눌렀을 때     |
| keyup    | 키에서 손을 떼 때   |

# 이벤트

- 키보드 이벤트 예제

다음은 키보드에서 키를 눌렀을 때 어떤 키인지 알아내는 예제입니다.

06\js\keycode.js

```
const body = document.body;
const result = document.querySelector("#result");

body.addEventListener("keydown", (e) => {
  result.innerText = `
  code : ${e.code},
  key : ${e.key}
  `;
});
```

# 이벤트

## 문서 로딩 이벤트

서버에서 웹 문서를 가져오거나 문서를 위아래로 스크롤하는 등 웹 문서를 브라우저 창에 보여 주는 것과 관련된 이벤트

| 이벤트    | 이벤트가 발생하는 순간                   |
|--------|--------------------------------|
| abort  | 웹 문서가 완전히 로딩되기 전에 불러오기를 멈추었을 때 |
| error  | 문서가 정확히 로딩되지 않았을 때             |
| load   | 문서 로딩이 끝났을 때                   |
| resize | 문서 화면의 크기가 바뀌었을 때              |
| scroll | 문서 화면이 스크롤되었을 때                |
| unload | 문서를 벗어날 때                      |

# 이벤트

- 문서 로딩 이벤트 예제

예를 들어 문서를 불러오자마자 알림 창을 표시하려면 다음과 같이 load 이벤트와 함수를 연결합니다. 이벤트와 함수를 연결하는 다양한 방법에 대해서는 앞으로 설명할 것입니다.

06\js\event-0.js

```
window.onload = alert("안녕하세요?");
```



# 이벤트

## 폼 이벤트

폼<sup>form</sup>이란, 로그인이나 검색, 게시판, 설문 조사처럼 사용자가 데이터(정보)를 입력하는 모든 요소를 가리킵니다. 따라서 폼 이벤트는 폼 요소에 내용을 입력하면서 발생할 수 있는 다양한 이벤트입니다.

| 이벤트    | 이벤트가 발생하는 순간  |
|--------|---|
| blur   | 폼 요소에 포커스를 잃었을 때  |
| change | 목록이나 체크 상태 등이 변경되었을 때<br>( <code>&lt;input&gt;</code> , <code>&lt;select&gt;</code> , <code>&lt;textarea&gt;</code> 태그에서 사용)                          |
| focus  | 폼 요소에 포커스를 놓았을 때<br>( <code>&lt;label&gt;</code> , <code>&lt;select&gt;</code> , <code>&lt;textarea&gt;</code> , <code>&lt;button&gt;</code> 태그에서 사용) |
| reset  | 폼이 리셋되었을 때  |
| submit | [submit] 버튼을 클릭했을 때   |

# 이벤트

## • 폼 이벤트 예제

수강신청하기

수강신청

신청인

이름

학과

수강신청하기

수강신청

이 페이지 내용:  
[컴퓨터공학과]를 선택했습니다.

확인

신청인

이름

학과

다음은 선택 목록에서 옵션을 선택하면 change 이벤트가 발생하고 change 이벤트에 displaySelect() 함수를 연결한 예제입니다.

06\js\getForm.js

```
const selectMenu = document.querySelector("#major");

function displaySelect() {
    let selectedText = selectMenu.options[selectMenu.selectedIndex].innerText;
    alert(`[${selectedText}]를 선택했습니다.`);
}
```

```
selectMenu.onchange = displaySelect;
```

학과 선택시 change, 이벤트 발생하고 이벤트를  
처리하기 위해 displaySelect 함수 호출

# 이벤트 처리하기

이벤트가 발생하면 그에 따른 연결 동작이 있어야 한다.

이렇게 **이벤트를 처리하는 것을 이벤트 처리기 또는 이벤트 핸들러(event handler)**라고 한다.

## 1) HTML 태그에 연결하기

이벤트가 발생한 HTML 태그에 직접 함수를 연결한다

```
<태그 on 이벤트명 = "함수명">
```

```
<button onclick = "alert('클릭!')">Click</button>
```

HTML 태그에 스크립트를 함께 사용하기 때문에 자스크립트 소스에서 함수 이름이 바뀌거나 다른 변경 내용이 있을 경우 HTML 소스도 함께 수정해야 한다.

## 2) 웹 요소에 직접 함수 연결하기

자스크립트 소스를 변경해도 HTML 마크업에는 영향을 주지 않게 하려면  
이벤트 처리기도 스크립트 소스에서 처리하는 것이 좋다

*요소.on 이벤트명 = 함수*

```
const button = document.querySelector("button");  
  
button.onclick = function() {  
    document.body.style.backgroundColor = "green";  
}
```

클릭했을 때 (onclick)  
실행할 함수를 표현식으로 할당

함수를 미리 만들어 두었다면 그 함수를 지정해도 된다.  
이때 실행할 함수 이름 뒤에 중괄호(( ))를 사용하지 않는다.

```
function changeBackground() {  
    document.body.style.backgroundColor = "green";  
}  
  
const button = document.querySelector("button");  
button.onclick = changeBackground;
```

### 3) addEventListener() 사용하기

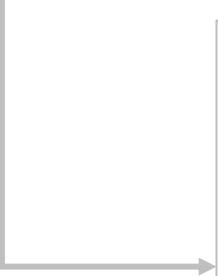
- 이벤트 리스너는 어떤 DOM 요소에서도 사용할 수 있다.
- addEventListener 함수를 사용

```
요소.addEventListener( 이벤트, 함수, 캡처 여부);
```

- 요소: 이벤트가 발생한 요소
- 이벤트: 이벤트 유형. 단, 여기에서는 이벤트 이름 앞에 on을 붙이지 않고 click이나 keypress처럼 이벤트 이름을 그대로 사용한다.
- 함수: 이벤트가 발생했을 때 실행할 함수. 기존에 있는 함수를 사용해도 되고 직접 익명 함수를 작성해도 된다. 익명 함수로 실행할 때는 event 객체를 사용해서 다양한 것들을 처리할 수 있다.
- 캡처 여부: 이벤트를 캡처링하는지의 여부. true면 캡처링을, false면 버블링을 한다는 의미. 선택 사항이며 기본값은 false

### 3) addEventListener() 사용하기

```
function changeBackground() {  
    document.body.style.backgroundColor = "green";  
}  
  
const button = document.querySelector("button");  
button.onclick = changeBackground;
```



A diagram consisting of a vertical line and a horizontal arrow pointing from the `button.onclick` property in the first code block to the `button` parameter in the second code block, illustrating the shift to using `addEventListener`.

```
function changeBackground() {  
    document.body.style.backgroundColor = "green";  
}  
  
const button = document.querySelector("button");  
button.addEventListener("click", changeBackground);
```

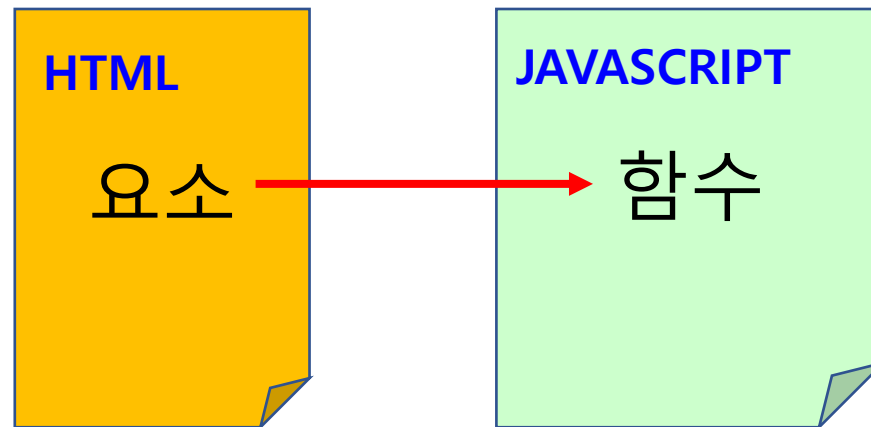
### 3) addEventListener() 사용하기

```
const button = document.querySelector("button");  
// 익명함수 사용  
button.addEventListener("click", function() {  
  document.body.style.backgroundColor = "green";  
});
```

```
const button = document.querySelector("button");  
// 화살표 함수 사용  
button.addEventListener("click", () => {  
  document.body.style.backgroundColor = "green";  
});
```

# 이벤트 리스너 만들기

- 인라인 방식: HTML 요소 안에 코드 추가
- 객체의 이벤트 리스너 속성에 함수를 대입
- 객체의 `addEventListener()` 함수를 사용
- 익명 함수로 이벤트 리스너 작성





# 이벤트 리스너 만들기

- 인라인 방식
- `<element event= " 자바스크립트 코드 " >`와 같이 HTML 요소에 코드를 작성해서 붙이는 방법이다.  
`<div onmouseover="this.style.backgroundColor='blue'"> Hello World! </div>`
- 객체의 이벤트 리스너 속성에 함수를 대입

```
<div id="special"> Hello World! </div>
<script>
  function changeColor() {
    document.getElementById("special").style.backgroundColor='blue';
  }
  document.getElementById("special").onmouseover = changeColor;
</script>
```

# 이벤트 리스너 만들기

- 객체의 `addEventListener()` 함수를 사용
  - DOM 객체가 가지고 있는 `addEventListener()` 함수를 이용하여 이벤트 리스너를 등록하는 방법이다. 객체의 이벤트 리스너 속성에 함수를 대입

```
let obj = document.getElementById("special");  
obj.addEventListener("mouseover", changeColor);
```

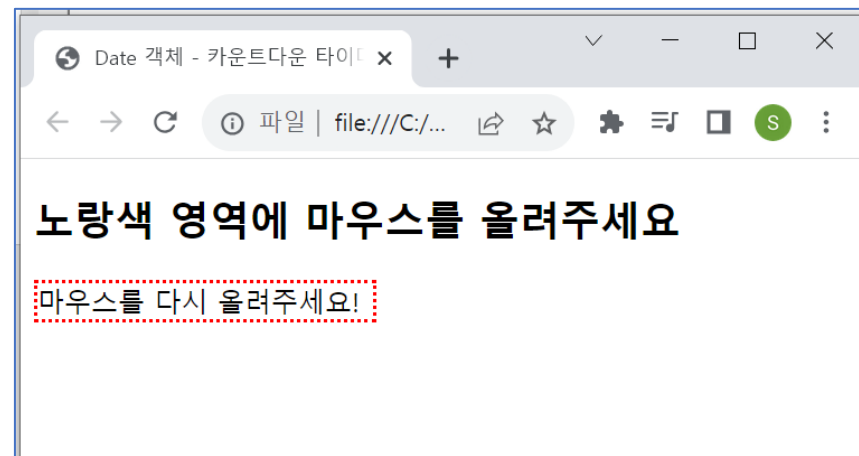
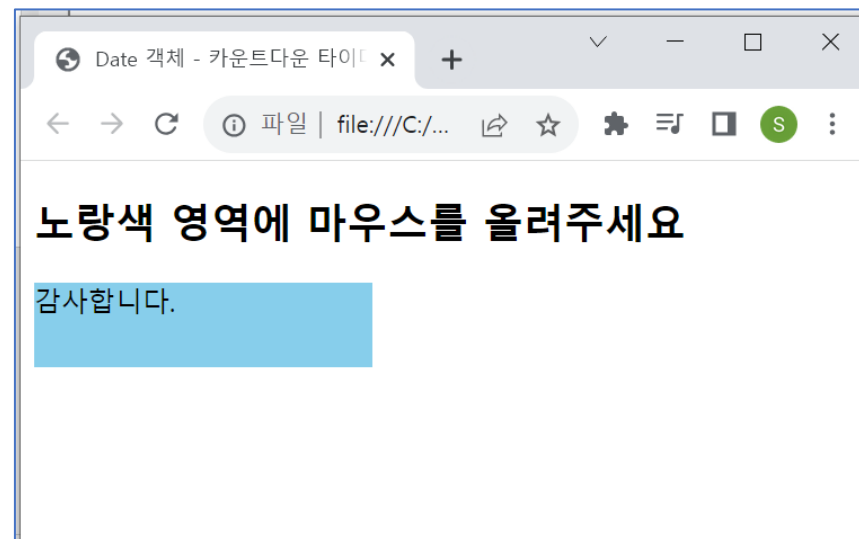
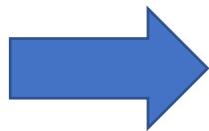
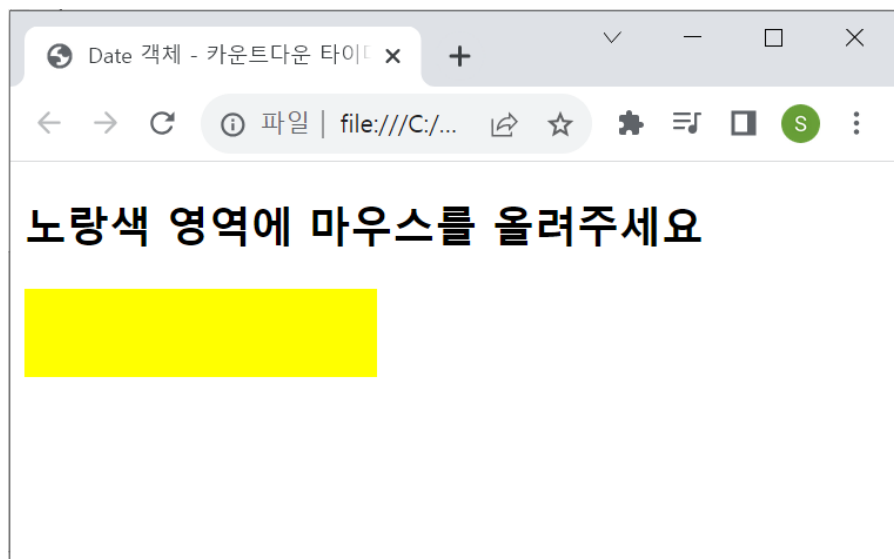
```
obj.addEventListener("mouseover", firstFunction);  
obj.addEventListener("mouseover", secondFunction);
```

# 이벤트 리스너 만들기

- 익명 함수로 이벤트 리스너 작성
  - 익명 함수(이름없는 함수)나 화살표 함수를 사용하여 이벤트 리스너를 작성할 수도 있다.

```
let obj = document.getElementById("special");  
obj.addEventListener("mouseover", function() {  
    document.getElementById("special").style.backgroundColor='blue';  
});
```

# [실습] 요소에 마우스가 올라오면 요소의 내용을 변경하는 코드를 작성해보자



# 이벤트 객체

- 이벤트가 발생하면 자동으로 만들어지는 객체

## event 객체의 메서드

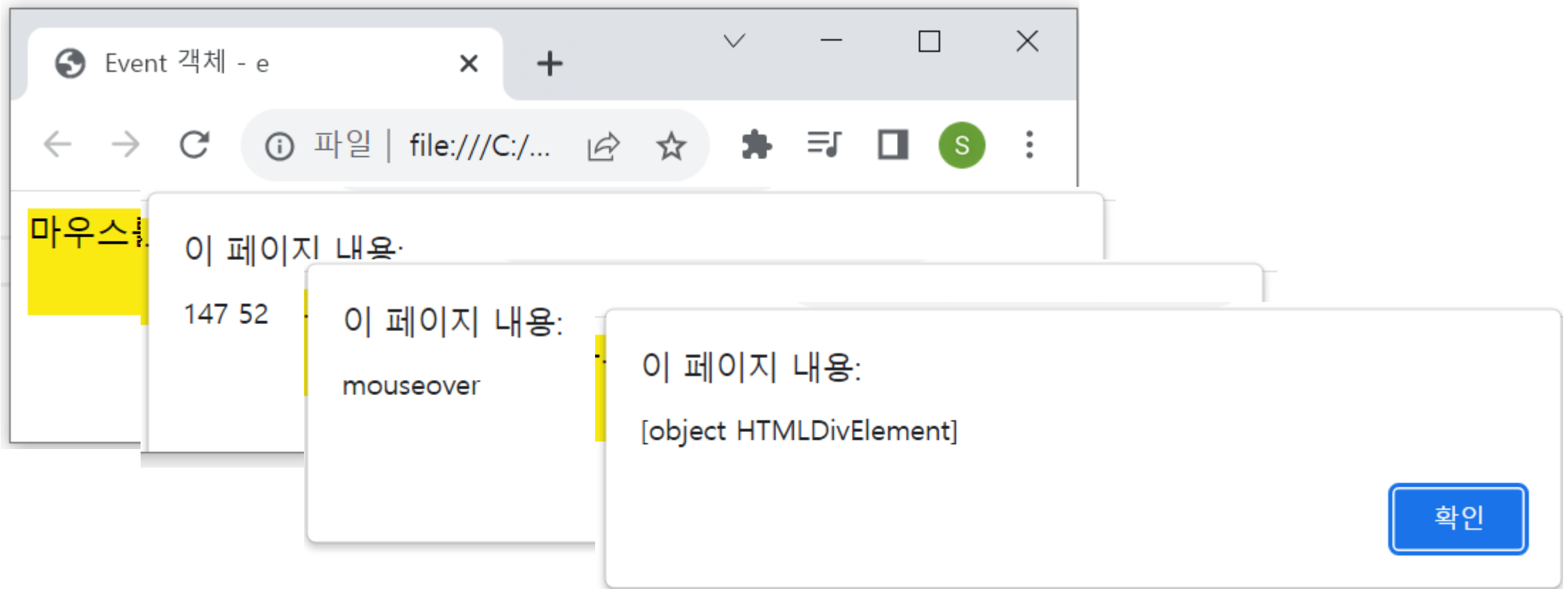
| 메서드            | 기능                          |
|----------------|-----------------------------|
| preventDefault | (취소할 수 있을 경우) 기본 동작을 취소합니다. |

## event 객체의 프로퍼티

| 프로퍼티      | 기능  |
|-----------|---|
| altKey    | 이벤트가 발생했을 때 <b>[Alt]</b> 를 누르고 있었는지의 여부를 확인하고 Boolean 값을 반환합니다.   |
| button    | 마우스 키를 반환합니다.   |
| charCode  | keypress 이벤트가 발생했을 때 어떤 키가 눌렸는지 유니코드값으로 반환합니다.                    |
| clientX   | 이벤트가 발생한 가로 위치를 반환합니다.  |
| clientY   | 이벤트 발생한 세로 위치를 반환합니다.   |
| ctrlKey   | 이벤트가 발생했을 때 <b>[Ctrl]</b> 을 누르고 있었는지의 여부를 확인하고 Boolean 값을 반환합니다.  |
| pageX     | 현재 문서를 기준으로 이벤트가 발생한 가로 위치를 반환합니다.                                |
| pageY     | 현재 문서를 기준으로 이벤트가 발생한 세로 위치를 반환합니다.                                |
| screenX   | 현재 화면을 기준으로 이벤트가 발생한 가로 위치를 반환합니다.                                |
| screenY   | 현재 화면을 기준으로 이벤트가 발생한 세로 위치를 반환합니다.                                |
| shiftKey  | 이벤트가 발생했을 때 <b>[Shift]</b> 를 누르고 있었는지의 여부를 확인하고 Boolean 값을 반환합니다. |
| target    | 이벤트가 발생한 대상을 반환합니다.   |
| timeStamp | 이벤트가 발생한 시간을 밀리초 단위로 반환합니다.                                       |
| type      | 발생한 이벤트 이름을 반환합니다.  |
| which     | 키보드와 관련된 이벤트가 발생했을 때 키의 유니코드값을 반환합니다.                             |

# 이벤트 객체

- 이벤트 리스너는 이벤트 객체(event object)를 전달받는다. 이벤트 객체는 이벤트마다 달라진다. 예를 들어서 mouseover 이벤트의 경우, 마우스 좌표와 마우스 버튼 정보 등이 포함된다.

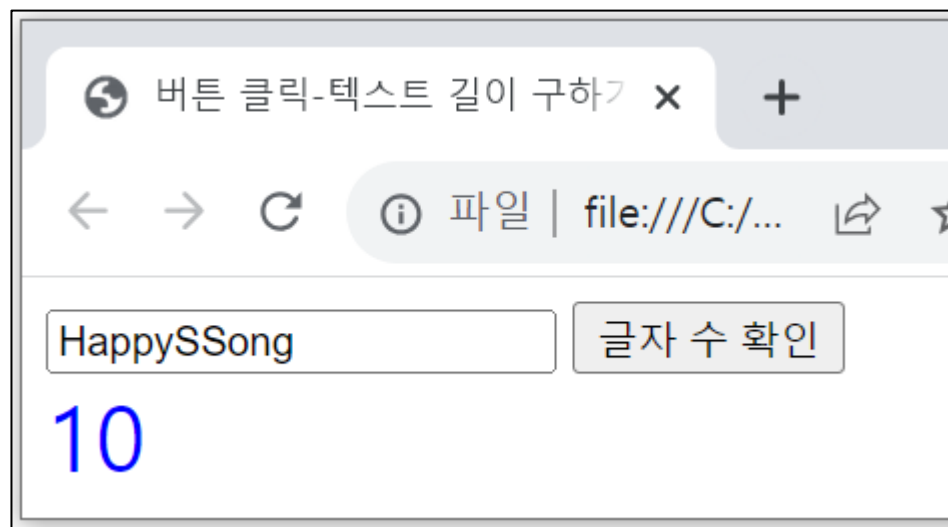
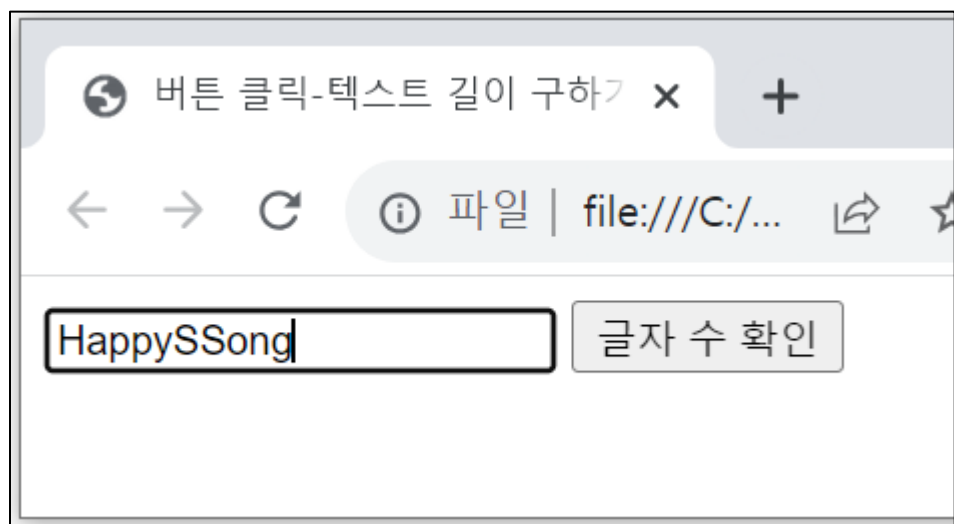


# 이벤트 객체

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title> Event 객체 - e </title>
6      </head>
7      <body>
8          <div id="special" style="background-color: ■ #faea11;width:200px;height:50px;">
9              마우스를 올려주세요!</div>
10         <script>
11             let obj = document.getElementById("special");
12             obj.onmouseover= function (e) {
13                 x=e.clientX;
14                 y=e.clientY;
15                 alert(x+" "+y);
16                 alert(e.type);
17                 alert(e.target);
18             };
19         </script>
20     </body>
21 </html>
```

## (예) 텍스트 필드에 입력한 글자 수 체크

텍스트 필드에 단어를 입력했을 때 단어의 길이를 화면에 표시하기.



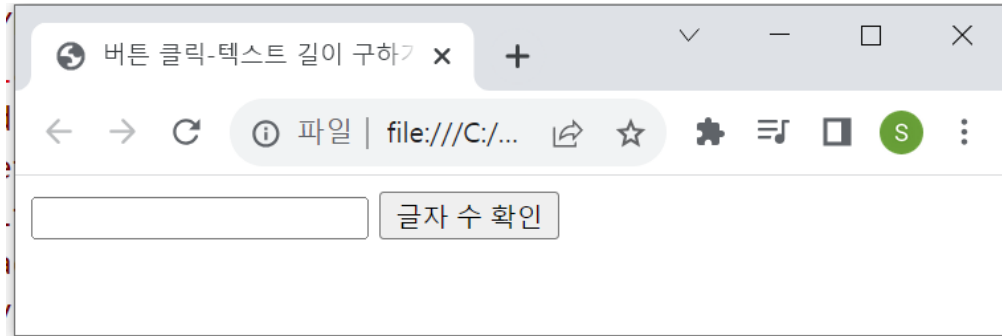
### <참고> 문자열 길이는?

문자열의 길이는 `length` 프로퍼티에 들어 있다.

```
> str = "안녕하세요?"  
< '안녕하세요?'  
> str.length  
< 6  
>
```



## (예) 텍스트 필드에 입력한 글자 수 체크



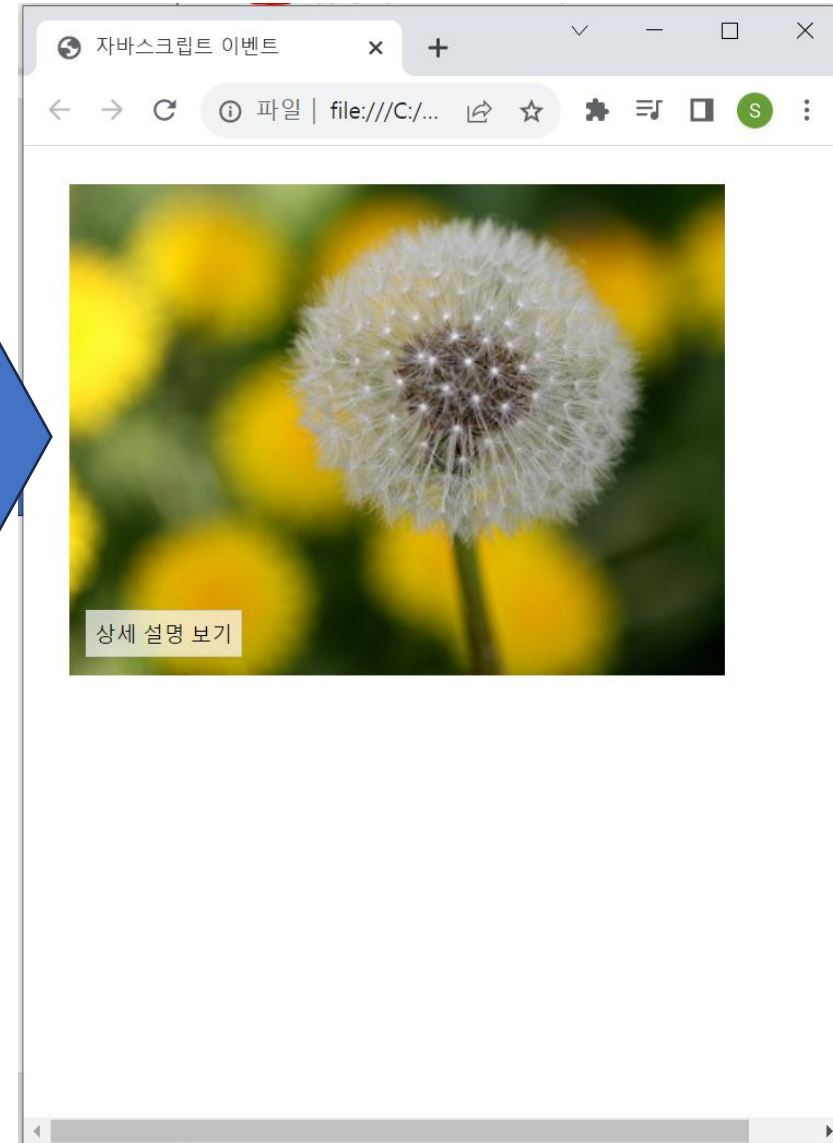
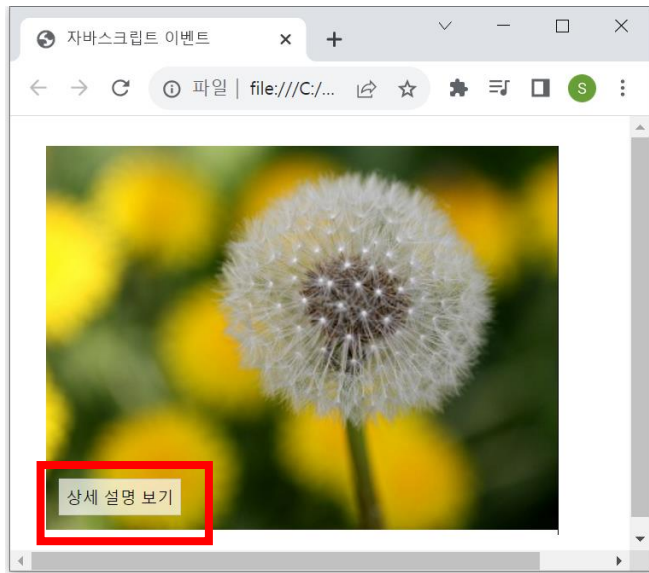
- 1) 버튼 클릭했을 때 실행
- 2) 텍스트 필드에 있는 내용을 가져와서, 그 내용의 length 확인
- 3) length값을 결과 영역에 표시

```
1  <!DOCTYPE html>
2  <html lang="ko">
3    <head>
4      <meta charset="UTF-8">
5      <title>버튼 클릭-텍스트 길이 구하기</title>
6    </head>
7    <body>
8      <div id="contents">
9        <input type="text" id="word">
10       <button id="btn">글자 수 확인</button>
11     </div>
12     <div id="result" style="color: ■ blue; font-size: 30px"></div>
```

## (예) 텍스트 필드에 입력한 글자 수 체크

```
13
14     <script>
15         const button = document.querySelector("#btn");
16         button.addEventListener("click", () => {
17             const word = document.querySelector("#word").value;    // 텍스트 상자의 내용
18             const result = document.querySelector("#result");      // 결과값 표시할 영역
19             let count = word.length;    // 문자열의 길이
20
21             result.innerText = `${count}`;    // 결과값 표시
22         });
23     </script>
24 </body>
25 </html>
```

# 버튼 클릭해서 상세 설명 열고 닫기



# 버튼 클릭해서 상세 설명 열고 닫기

상세 설명 여는 함수 & 닫는 함수 (event.js)

```
1  function showDetail() {  
2      document.querySelector('#desc').style.display = "block"; //상세 설명 내용을 화면에 표시  
3      document.querySelector('#open').style.display = "none"; //[상세 설명 보기] 버튼 감춤  
4  }  
5  
6  function hideDetail() {  
7      document.querySelector('#desc').style.display = "none"; //상세 설명 내용을 화면에서 감춤  
8      document.querySelector('#open').style.display = "block";  //[상세 설명 보기] 버튼 표시  
9  }
```

# 버튼 클릭해서 상세 설명 열고 닫기

버튼 클릭했을 때 함수 실행하기

```
11 <div id="item">
12   
13   <button class="over" id="open" onclick="showDetail()">상세 설명 보기</button>
14   <div id="desc" class="detail">
15     <h4>민들레</h4>
16     <p>어디서나 매우 흔하게 보이는 ..... 널리 퍼진다.</p>
17     <button id="close" onclick="hideDetail()">상세 설명 닫기</button>
18   </div>
```

③ 함수 실행



① 버튼에서 click 이벤트가 발생하면

② click 이벤트 처리를 찾아

# 버튼 클릭해서 상세 설명 열고 닫기(전체코드)

```
1  <!DOCTYPE html>
2  <html lang="ko">
3  <head>
4      <meta charset="UTF-8">
5      <title>자바스크립트 이벤트</title>
6      <link rel="stylesheet" href="css/event.css">
7  </head>
8  <body>
9      <div id="item">
10         
11         <button class="over" id="open" onclick="showDetail()">상세 설명 보기</button>
12         <div id="desc" class="detail">
13             <h4>민들레</h4>
14             <p>어디서나 매우 흔하게 보이는 잡초로서 바닥에
15                 딱 붙어서 꽃봉오리 하나가 솟 올라온다.
16                 톱니 모양의 잎새와 눈에 확 띄는 노란 꽃이 인상적이다.
17                 특히 꽃이 지고나면 숨털모양의 깃을 가진 씨앗들이
18                 나오는데 바람을 타고 날아가서 널리 퍼진다.</p>
19             <button id="close" onclick="hideDetail()">상세 설명 닫기</button>
20         </div>
21     </div>
22
23     <script src="js/event.js"></script>
24 </body>
25 </html>
```

# 버튼 클릭해서 상세 설명 열고 닫기(전체코드)

```
1  #item {  
2      position: relative;           event.css  
3      width: 500px;  
4      height: auto;  
5      padding: 15px 20px;  
6      margin: auto;  
7  }  
8  button {  
9      background-color:  rgba(255, 255, 255, 0.7);  
10     padding: 5px;  
11     border: 1px solid  #ccc;  
12     font-size: 0.8em;  
13 }  
14 .over {  
15     position: absolute;  
16     left: 30px;  
17     bottom: 30px;  
18 }  
19 .detail {  
20     width: 400px;  
21     text-align: left;  
22     line-height: 1.8;  
23     display: none;  
24 }  
25 #cover { border: 5px solid transparent; }
```

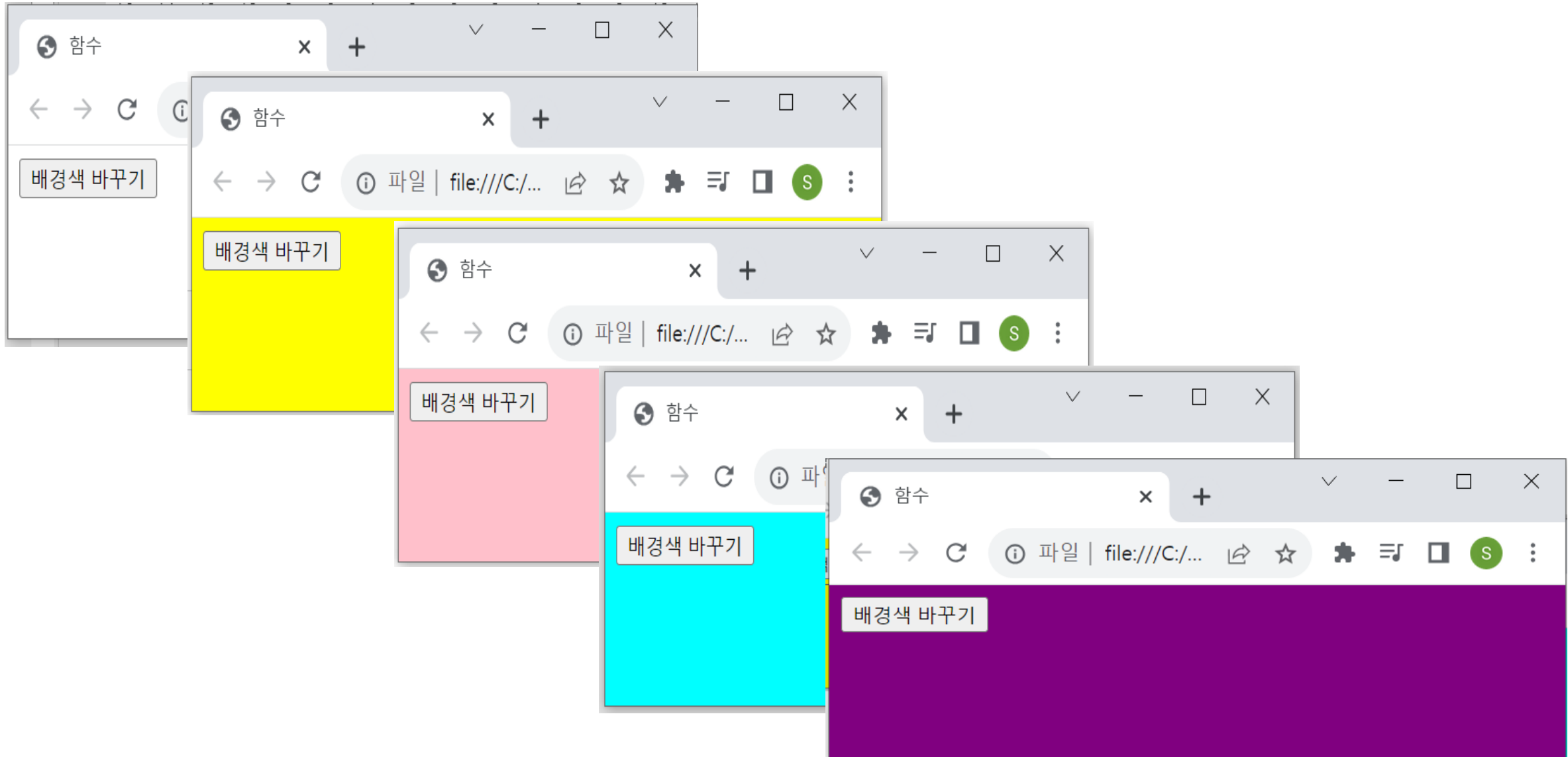
# 버튼 클릭해서 상세 설명 열고 닫기(전체코드)

## event.js

```
1 function showDetail() { // '상세 설명 보기'를 클릭했을 때 상세 설명을 보여주는 함수
2     document.querySelector('#desc').style.display = "block"; // 상세 설명 부분을 화면에 표시
3     document.querySelector('#open').style.display = "none"; // '상세 설명 보기' 단추를 화면에서 감춤
4 }
5
6 function hideDetail() { // '상세 설명 닫기'를 클릭했을 때 상세 설명을 감추는 함수
7     document.querySelector('#desc').style.display = "none"; // 상세 설명 부분을 화면에서 감춤
8     document.querySelector('#open').style.display = "block"; // '상세 설명 보기' 단추를 화면에 표시
9 }
```



# [실습-3] 배경색 바꾸기 버튼 클릭시 배경색 변경



## [실습-4]

- 총 8개의 이미지(pic\_1.jpg~pic\_8.jpg)를 사용해 갤러리를 만드는 프로그램을 완성 하시오.
  - [다음] 버튼을 누르면 pic\_2.jpg, pic\_3.jpg가 순서대로 이미지 표시
    - 즉, 이미지 파일명의 숫자가 1씩 증가되며 그림이 나타남
  - 이미지는 pic\_8.jpg까지만 있기 때문에 9가 나오면 이미지를 불러올 수 없어서 오류가 발생
  - 8보다 큰 값이 나오는 경우는 함수를 종료



```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset=utf-8">
5      <title> 사진 갤러리 </title>
6      <style type="text/css">
7        * { margin:0;padding:0; }
8        li { list-style:none; }
9        a { text-decoration:none;color:■#000; }
10       #gallery {
11         width:350px;margin:30px auto;
12         border:1px solid ■#999;
13         box-shadow:0 5px 5px ■rgba(0, 0, 0, 0.3);
14       }
15       #gallery h1 { text-align:center; }
16       #gallery img { border:1px solid ■#666; }
17       #photo { text-align:center; }

```

```

18     #gallery ul {
19       width:800px;margin:0 auto;
20       overflow:hidden;
21     }
22     #photo_list ul li {
23       width:100px;
24       float:left;
25       text-align:center;
26     }
27     #photo_list {
28       width:300px;
29       margin:10px auto;
30       overflow:hidden;
31     }
32     .btn_wrap { text-align:center; }
33     .btn_wrap a { font-weight:bold; }
34 </style>

```

```

36 <script type="text/javascript">
37     window.onload=function() {
38         /*① 포토 목록에 작은 이미지를 클릭했을 때 큰 이미지를 바꿈 */
39         /* 클릭 이벤트를 등록 하기위해서 포토 목록에 8개의 <a>를 모두 선택 */
40         var list_zone=document.getElementById("inner_list");
41         var list_a=list_zone.getElementsByTagName("a");
42
43         /* 포토 리스트의 모든 <a>에 클릭 이벤트를 등록하기 위해서
44         반복문을 이용하여 8개의 <a>에 모두 이벤트를 등록 */
45         for(var i=0; i<list_a.length; i++){
46             list_a[i].onclick = function() {
47                 var ph=document.getElementById("photo").children[0];
48                 ph.src=this.href;
49                 return false; // <a>를 클릭했을 때 링크가 되지 않도록 함
50             }
51         }
52
53         // ② 이전(<), 다음 버튼을 클릭할 때 마다 <ul>이 100px만큼 증가 또는 감소되어 좌·우측으로 이동
54         // 다음(>) 버튼을 클릭했을 때 마다 <ul>이 100px만큼 좌측으로 이동
55         var b_btn=document.getElementById("next_btn");
56         var m_num=0;
57         b_btn.onclick = function() {
58             if(m_num>=list_a.length-3) return false;
59             m_num++;
60             list_zone.style.marginLeft =- 100 * m_num + "px";
61
62             return false; // <a>를 클릭했을 때 링크가 되지 않도록 함
63         }

```

```

65 // 이전(<) 버튼을 클릭했을 때 마다 <ul>이 100px만큼 우측으로 이동
66 var n_btn=document.getElementById("before_btn");
67 n_btn.onclick = function() {
68     if(m_num<=0) return false;
69     m_num--;
70     list_zone.style.marginLeft =- 100 * m_num + "px";
71
72     return false; // <a>를 클릭했을 때 링크가 되지 않도록 함
73 }
74 }
75 </script>
76 </head>
77 <body>
78 <div id="gallery">
79     <h1>동해안 갤러리</h1>
80     <p id="photo">
81         
82     </p>
83     <div id="photo_list">
84         <ul id="inner_list">
85             <li> <a href="images/pic_1.jpg">  </a> </li>
86             <li> <a href="images/pic_2.jpg">  </a> </li>
87             <li> <a href="images/pic_3.jpg">  </a> </li>
88             <li> <a href="images/pic_4.jpg">  </a> </li>
89             <li> <a href="images/pic_5.jpg">  </a> </li>
90             <li> <a href="images/pic_6.jpg">  </a> </li>
91             <li> <a href="images/pic_7.jpg">  </a> </li>
92             <li> <a href="images/pic_8.jpg">  </a> </li>
93         </ul>
94     </div>
95     <p class="btn_wrap">
96         <a href="#" id="before_btn">&lt;</a>
97         <a href="#" id="next_btn">&gt;</a>
98     </p>
99 </div>
100 </body>
101 </html>

```

## [실습과제-2]

- 질의응답 창(prompt)을 통해 방문자의 아이디와 비밀번호를 입력 받는 프로그램 작성
  - 만일 잘못된 아이디가 입력된 경우는 "존재하지 않는 아이디" 라고 경고창으로 표시
  - 잘못된 비밀번호가 입력된 경우는 "잘못된 비밀번호" 라는 경고 창을 표시
  - 아이디와 비밀번호가 일치하면 "환영 문구" 표시
  - 함수 login(id, pw) 작성
  - 변수 rightid에 "korea" 할당
  - 변수 rightpw에 "1234" 할당

이 페이지 내용:

존재하지 않는 아이디입니다.

확인

