3강

함수 & 이벤트핸들러

3강

함수

JavaScript

➢ 함수(function)

- 자주쓰는 실행코드를 블록단위로 묶어놓은 것이다.
- 기능단위로 재사용하기 위해서 사용한다.
- 함수 선언은 function키워드 사용하거나 arrow function(화살표 함수=>)를 사용한다.
- 호출 : **정의되어 있는 함수는 호출해야 비로서 기능이 실행된다.**

➢ 함수(function)의 종류

1. 선언적함수 : 이름을 붙여서 정의한 함수

1. 자바스크립트를 읽을 때 **우선적**으로 읽어준다.
2. 호출위치가 자유롭다

2. 익명함수 : 이름 없이 정의한 함수 - 콜백함수 - 이벤트 처리시 사용

1. 자체로 호출불가
2. 함수명 대신 변수에 익명함수를 대입하거나 특정 이벤트 객체에 대입해서 호출 – 대입형 함수
3. 익명 함수의 소스 코드는 변수값이므로 끝에 세미콜론 ; 을 써준다.

3. 일급함수와 고차함수

4. 즉시실행함수 : 함수가 자기자신을 정의하자마자 바로 자신 호출함

함수(function) 란 ?

JavaScript

➢ 선언적 함수(function)

JavaScript의 호이스팅hoisting)은 코드가 실행하기 전 변수선언/함수선언이 해당 스코프의 최상단으로 끌어 올려진 것 같은 현상을 말한다.

- 선언적 함수는 어디서든 호출이 가능하다.
- **호이스팅** 이슈가 있어서 선언적 함수는 일반적으로 사용을 지양한다.

```
sayHello(); //오류 없이 실행됨
function sayHello(){ console.log('Hello') }
sayHello(); //오류 없이 실행됨
```

➢ 익명(대입형 = 함수 표현식) 함수(function)

- 익명 함수는 한 줄씩 읽으면서 생성된다. 코드에 도달하면 생성된다.

```
sayHello(); //에러발생
let sayHello = function(){ console.log('Hello') }
sayHello(); //오류없이 실행됨
```

JavaScript

➢ 선언적 함수와 익명 함수 비교

- 선언적 함수와 익명함수는 실행 순서가 다르다..

```
함수()
함수=function(){ // 2. 나중에 실행됨
        console.log("익명함수 입니다")
 }
function 함수(){ // 1. 먼저 만들어짐
     console.log("선언적 함수입니다")
}
함수()
```

```
"선언적 함수입니다"
"익명함수 입니다"
```

JavaScript

➢ 일급 함수(function)

- 함수가 값으로서 사용된다. (함수 스스로 객체 취급)
  - 변수에 담을 수 있다 - **대입형 함수**
  - 인자로 전달할 수 있다 - **콜백함수**

```
function sayHello() { return "Hello,"; }

function greeting(helloMessage, name){

        console.log(helloMessage()+name);

 } // 'sayHello'를 greeting함수에 인자로 전달 함

greeting(sayHello,'javaScript');
```

콜백함수

JavaScript

➢ 화살표 함수(arrow function)

- 함수 표현식(=익명 함수)보다 단순하고 간결한 문법으로 함수를 만들 수 있는 방법으로 **화살표 함수 (arrow function)**를 사용할 수 있다.
- 화살표 함수라는 이름은 문법의 생김새를 차용해 만들어 졌다.

```
let func = (arg1, arg2, ...argN) => expression
```

```
let func = function(arg1, arg2, ...argN) {
  return expression;
};
```

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

/* 위 화살표 함수는 아래 함수의 축약 버전입니다.

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

alert( sum(1, 2) ); // 3
```

```
let sum = (a, b) => {  // 중괄호는 본문 여러 줄로 구성되어 있음을 알려줍니다.
  let result = a + b;
  return result; // 중괄호를 사용했다면, return 지시자로 결괏값을 반환해주어야 합니다.
};

alert( sum(1, 2) ); // 3
```

함수 / 매개 / ↳result는 데리고 나옴 / 함수이름 → 1+2=3

JavaScript

➢ 즉시 실행 함수(function)

- 선언과 동시에 호출되는 함수로 재사용하지 않을 경우에 사용한다. ex) 변수를 초기화하는 함수

```
(function (){
    console.log('Hello');
})();
// 화살표 함수로도 사용 가능
(()=>{
    console.log("JavaScript");
})();
```

JavaScript

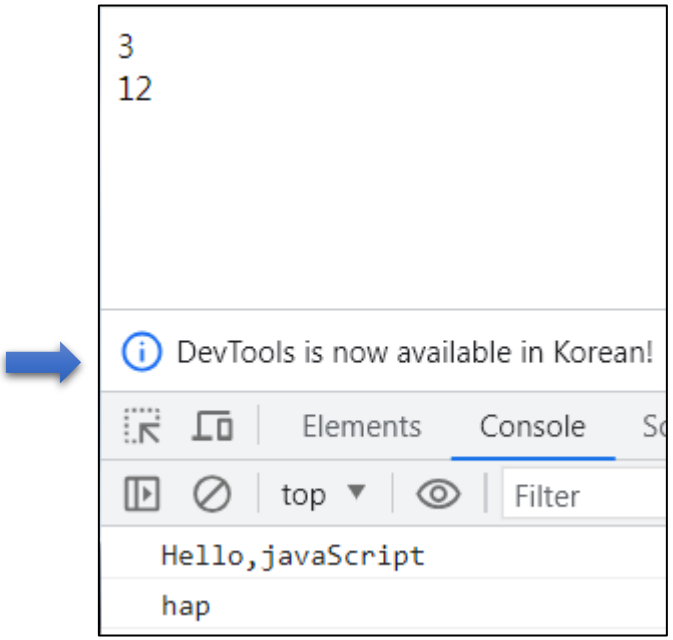
➢ javaScript에 함수(function) 작성 예

```
<body>
  <script>
    function sayHello() {return "Hello,";}
    function greeting(helloMessage, name){
      console.log(helloMessage()+name);
    } // 'sayHello()'를 greeting함수에 인자로 전달 함 => 일급함수
    greeting(sayHello,'javaScript');

   hap();
   function hap(){ //선언적 함수
     console.log("hap");
   }

   const minus = function (a=1,b=2){  //익명함수
     const num1 = 2;
     const num2 = 1;
     document.write(num1+num2);
     document.write("<br />");
     document.write(a+b);
   };
   minus(5,7);

  </script>
</body>
```



JavaScript

➢ javaScript에 함수(function) 작성 예

JavaScript의 a == b에서 ==은 a, b의 값이 같은지 비교하지만, a === b에서 ===는 a, b의 변수 유형과 값이 같은지 비교한다.

```
// 간단한 함수 예제
   let nums=[1,2,3,4,5,6,7,8,9,10];

   // 배열의 filter 함수 : callback 함수를 실행하고
   // 결과가 true인 요소들만으로 이루어진 새로운 배열을 반환

   const isOdd = (currentNum) => { return currentNum % 2 == 0 };
   console.log(nums.filter(isOdd));

   // 홀수만 추출
   nums = nums.filter( number => number % 2 == 1);
   console.log(nums);

   // 5이하의 값만 추출
   nums = nums.filter( value => value <= 5 );
   console.log(nums);

   // 3의 배수만 추출
   nums = nums.filter((value)=>( value % 3 === 0));
   console.log(nums);
```

```
▶ (5) [2, 4, 6, 8, 10]
▶ (5) [1, 3, 5, 7, 9]
▶ (3) [1, 3, 5]
▶ [3]
```

함수 호출 단계별 동작 흐름

JavaScript

➢ 함수(function) 호출 단계별 동작 흐름도

➢ 함수가 호출되면 **입력(매개변수) → 함수 내부 실행(연산/부작용) → 출력(반환값 또는 콘솔/DOM 출력)** 의 흐름으로 결과가 만들어 진다.

1. **입력**: 함수에 전달되는 값(매개변수) 혹은 이벤트 정보
2. **실행 컨텍스트**: 함수 내부에서 변수/로직이 실행되는 공간
3. **출력**: return으로 반환되거나, console.log/DOM 변경 등으로 외부에 드러나는 값
4. 부작용(side-effect): 함수가 외부 상태(DOM, 전역 변수 등)를 변경하는 경우

JavaScript

➢ 함수(function) 호출 단계별 동작 흐름도

입력 (arguments : 인수)

↓

함수 호출

```
Function add(a, b){
  //실행문들
  return a + b;
}
```

↓

출력 (return value/ console / DOM)

JavaScript

➢ 함수(function) 호출 단계별 동작 흐름도

```
function add(a, b) {
        console.log("[1] add 함수 시작");
       // 함수가 호출되면 제일 먼저 찍힘

         console.log("[2] 매개변수:", a, b);

        const result = a + b;
        console.log("[3] 계산 결과:", result);

        console.log("[4] 함수 종료 직전");
        return result;
      }

      console.log("호출 결과:", add(2, 3));
```

```
[1] add 함수 시작
[2] 매개변수: 2 3
[3] 계산 결과: 5
[4] 함수 종료 직전
호출 결과: 5
```

JavaScript

➢ 함수(function) 호출 단계별 동작 흐름도

1. add(2, 3)가 호출되며 매개변수 a=2, b=3이 만들어짐.
2. 함수 내부 첫 로그: [add] 시작 — 매개변수 2 3 출력.
3. result = a + b 계산(2 + 3 → 5).
4. 내부 로그: [add] 계산 결과 5 출력.
5. return result로 5가 반환되고, 호출 지점의 console.log('호출 결과:', ...)가 호출 결과: 5 를 출력.

JavaScript

➢ 함수(function) : 버튼 클릭 시 함수 호출

```
<button id="btn">클릭해봐</button>
  <script>
    const btn = document.getElementById('btn');

      function onClickHandler(event) {
      console.log('[handler] 이벤트 시작', event.type);
      const time = new Date().toLocaleTimeString();
      console.log('[handler] 현재 시간:', time);
      btn.textContent = '클릭됨 @ ' + time; // DOM 변경 (부작용)
      }

      btn.addEventListener('click', onClickHandler);
  </script>
```

```
[handler] 이벤트 시작 click
[handler] 현재 시간: 오전 3:32:02
```

호출

JavaScript

➢ 함수(function) : 버튼 클릭 시 함수 호출

1. 사용자가 버튼 클릭 → 브라우저가 click 이벤트 생성
2. 이벤트는 등록된 onClickHandler를 호출(입력: event 객체)
3. 핸들러 내부 로그 출력 → DOM 변경(버튼 텍스트)

JavaScript

➢ 여기서 잠깐!!

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

add(a, b) 매개 함수에 a="hello", b=3의 인수로 입력하면 결과 값은 어떻게 될까요?

```
function divide(a, b) {
  return a / b;
}
console.log(divide(10, 0));
```

0으로 나누면 수학적으로 정의되지 않음. JS에서는?

3강

이벤트핸들러

이벤트(event) 란 ?

JavaScript

➢ 이벤트(event)

- 우리가 이용하는 웹사이트에는 수많은 기능을 제공한다. 특정 메뉴 버튼을 클릭할 때 정보가 노출된다던가 반대로 정보를 입력하면 서버에 저장 기능 등, 이러한 기능들이 없다면 웹페이지는 그냥 글자만 있는 전자문서와 동일하다. 주로 우리들은 이 웹페이지에 있는 기능을 이용하기 위해서 **마우스나 키보드를 조작하여 기능을 실행**한다. 바로 이러한 행위들을 **이벤트**라고 부른다.

➢ 이벤트(event) 처리기

- 이벤트 처리기 란 말그대로 이벤트 청취자, 즉 위와 같은 **이벤트가 발생하는 것을 감지하는 역할**을 한다. 따라서 이벤트가 발생했을 때 이벤트를 받은 다음 처리기 안에 있는 코드를 실행한다. 대개 특정 기능을 하는 함수가 실행 대상이다. 이벤트 처리기는 **이벤트 핸들러**와 **이벤트 리스너**가 있다.

1. 핸들러 : 하나의 요소에 하나의 이벤트 처리 가능, HTML 태그에 설정하거나, DOM요소 객체의 이벤트 처리기 프로퍼티 설정하는 방법이다.
2. 리스너 : 하나의 요소에 복수의 이벤트 처리 가능, addEventListner 메서드

JavaScript

➢ 주요 이벤트(event) 처리기 종류

✓ **UI 이벤트 – 사용자가 웹페이지가 아닌 브라우저의 UI와 상호작용할 때 발생**

이벤트	설명
load	웹 페이지의 로드가 완료되었을 때
unload	웹 페이지가 언로드 될 때(새로운 페이지를 요청한 경우)
error	브라우저가 자바스크립트 오류를 만났거나 요청한 자원이 없는 경우
resize	브라우저의 창 크기를 조정했을 때
scroll	사용자가 페이지를 위아래로 스크롤 할 때

✓ **키보드 이벤트 – 사용자가 키보드를 이용할 때 발생한다.**

이벤트	설명
keydown	사용자가 키를 처음 눌렀을 때
keyup	키를 땔 때
keypress	사용자가 눌렀던 키의 문자가 입력되었을 때

JavaScript

➢ 주요 이벤트(event) 처리기 종류

✓ **마우스 이벤트 – 사용자가 마우스나 터치 화면을 사용할 때 발생**

이벤트	설명
click	사용자가 동일한 요소 위에서 마우스 버튼을 눌렀다 땔 때
dbclick	두 번 눌렀다 땔 때
mousedown	마우스를 누르고 있을 때
mouseup	눌렀던 마우스 버튼을 땔 때
mousemove	마우스를 움직였을 때
mouseover	요소 위로 마우스를 움직였을 때
mouseout	요소 바깥으로 마우스를 움직였을 때

✓ **포커스 이벤트 – 포커스 이벤트**

이벤트	설명
focus	요소가 포커스를 얻었을 때
blur	요소가 포커스를 잃었을 때

JavaScript

➢ 주요 이벤트(event) 처리기 종류

✓ **폼 이벤트 - 폼 이벤트**

이벤트	설명
input	<input>,<textarea>요소 값이 변경되었을 때
change	선택 상자, 체크박스, 라디오 버튼의 상태가 변경되었을 때
submit	사용자가 버튼키를 이용하여 폼을 제출할 때
reset	리셋 버튼을 클릭할 때
cut	폼 필드의 콘텐츠를 잘라내기 했을 때
copy	폼 필드의 콘텐츠를 복사했을 때
paste	폼 필드의 콘텐츠를 붙여넣을 때
select	텍스트를 선택했을 때

JavaScript

➢ HTML 태그에 설정

✓ 가장 간단한 방법이지만 HTML과 자바스크립트 코드가 섞여있기 때문에 후에 유지보수가 힘들다.

```
<button onclick="showSideNavi();"></button>
```

➢ DOM 요소 객체의 이벤트 처리기 프로퍼티에 설정

✓ HTML과 자바스크립트를 분리할 수 있지만 위와 마찬가지로 특정 DOM 요소 객체에 하나의 이벤트 핸들러만 등록시키는 단점이 있다.

```
let sideNavi = document.getElementById('sideNavi');
let btn1 = document.getElementById('btn');
 btn1.onclick = () => {
    sideNavi.style.backgroundColor='pink';
};
```

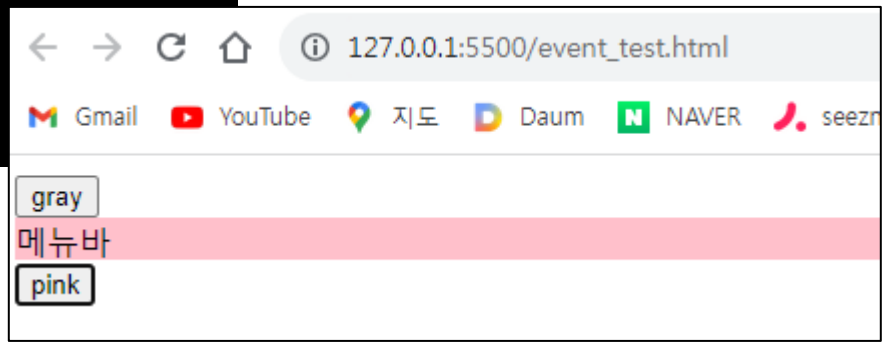
이벤트 핸들러(Event Handler) 간단한 예제

JavaScript

```
<body>
 <button type="button" onclick="showSideNavi();">gray</button>
 <div id="sideNavi">메뉴바</div>
 <button type="button" id="btn">pink</button>

 <script>
   let sideNavi = document.getElementById('sideNavi');
   let btn1 = document.getElementById('btn');
   btn1.onclick = () => {
    sideNavi.style.backgroundColor='pink';
  };

   function showSideNavi(){
    sideNavi.style.backgroundColor='gray';
   }
 </script>
</body>
```



JavaScript

➢ addEventListener( ) 함수

- ✓ window 객체나 DOM 객체의 addEventListener() 메소드에 이벤트 핸들러를 바인딩(연결)하는 방법
- ✓ HTML과 자바스크립트 코드를 분리시킬 수 있다는 이점이 있다.
- ✓ 여러 개의 이벤트를 중복하여 등록시킬 수 있다. (두 개의 클릭 이벤트 핸들러)
- ✓ addEventListener() 함수는 이벤트가 발생한 요소에 이벤트 처리기를 연결해주는 함수로, 웹 문서에서 이미지나 텍스트 등 특정 요소뿐만 아니라 Document 객체나 window 객체 어디서든 사용할 수 있다.
- ✓ addEventListener()함수는 처리할 이벤트 유형을 지정한다. 단, 유형을 지정할때는 on은 붙이지 않고 'click'이나 'mouseover' 처럼 이름만 사용한다.
- ✓ 이벤트가 발생하였을 때 명령을 나열하거나 따로 함수를 만들었다면 함수 이름을 지정한다.
- ✓ addEventListener("이벤트 유형", "함수 이름")

JavaScript

➢ addEventListener( ) 함수 형식

1. addEventListener(type, listener);
2. addEventListener(type, listener, options);
3. addEventListener(type, listener, useCapture);

- ➢ Type : 이벤트 유형을 뜻하는 문자열("click", "mouseUp" 등)
- ➢ Listener : 이벤트가 발생했을 때 처리를 담당하는 콜백 함수의 참조 (자바스크립트 함수 이름)
- ➢ useCapture : 이벤트 단계 (true: 캡처링 단계 / false : 버블링 단계(생략했을 때의 기본값)

```
<a>클릭</a>
  <script>
    let a = document.querySelector('a');
    a.addEventListener('click', showConsole);

    function showConsole() {
      console.log('콘솔로그 실행');
    }
```

이벤트 리스너(EventListener) 간단한 예제

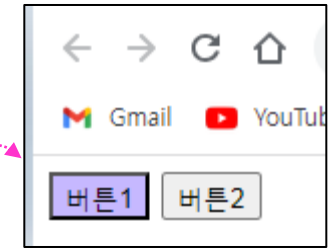
JavaScript

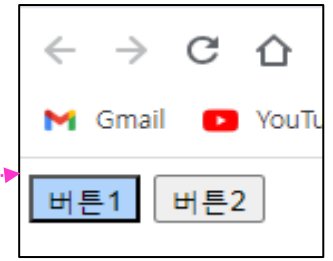
➢ addEventListener( ) 함수

```
<body>
    <button id="btn1">버튼1</button>
    <button id="btn2">버튼2</button>
    <script>
      let btn1 = document.getElementById('btn1');
      let btn2 = document.getElementById('btn2');

      btn1.addEventListener('mouseover',changColor1);
      btn1.addEventListener('mouseout',function(e){
        e.currentTarget.style.backgroundColor='#afd3fa';
      });

      function changColor1(e){
        e.currentTarget.style.backgroundColor='#c6b8ff';
      }
    </script>
 </body>
```





기초 문제

함수(function) 연습문제

예제1] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 사용자의 이름을 받아 "안녕하세요, OOO님!" 형태의 문자열을 반환하는 함수를 작성하세요.
② 실행 예시 : console.log(greetUser("철수"));
③ 저장할 파일명 Ex_01.html 로 작성하시오.

예제2] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 상품 가격과 할인율을 받아 최종 결제 금액을 반환하는 getDiscountPrice(price, rate) 함수를 작성하세요.
② 실행 예시 : console.log(getDiscountPrice(10000, 0.1));
③ 저장할 파일명 Ex_02.html 로 작성하시오.

함수(function) 연습문제

예제3] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 버튼을 클릭할 때마다 클릭 횟수를 콘솔에 출력하는 코드를 작성하세요.

② 버튼 3번 클릭 → 콘솔에 1, 2, 3 순서대로 출력

③ 저장할 파일명 Ex_03.html 로 작성하시오.

예제4] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 배열로 전달된 장바구니 가격 목록의 합계를 반환하는 getTotal(cart) 함수를 작성하세요.

② 실행 예시 : console.log(getTotal([1000, 2000, 3000]))

③ 저장할 파일명 Ex_04.html 로 작성하시오.

함수(function) 연습문제

예제5] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 사용자가 입력한 비밀번호가 **8자 이상**인지 검사하는 함수를 작성하세요.

② 실행 예시 : console.log(validatePassword("1234")); // false
console.log(validatePassword("12345678")); // true

③ 저장할 파일명 Ex_05.html 로 작성하시오.

예제6] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 버튼을 누르면 현재 시간을 콘솔에 출력하는 코드를 작성하세요.

② 버튼 클릭 → 콘솔에 현재 시간: 2025-09-30 10:23:11

③ 저장할 파일명 Ex_06.html 로 작성하시오.

함수(function) 연습문제

예제7] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 버튼 클릭 시 이미지를 보였다가 숨기 함수를 addEventListener()를 이용해 작성하시오.
② 아래 주어진 <HTML> 이용해 작성하시오.
③ classList.toggle() 말고 if문을 이용하여 작성하시오.
④ 저장할 파일명 Ex_07.html 로 작성하시오.

HTML

```
<button id="toggleBtn">이미지 토글</button>
<img id="myImg" src="images/pic-2.jpg" style="display: block" />
```

함수(function) 연습문제

예제8] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 버튼 클릭 시 텍스트 색상을 빨강 → 파랑 → 초록 순으로 변경하세요.
② 아래 주어진 <HTML> 이용해 작성하시오.
③ const colors = ['red', 'blue', 'green']; 이용해 작성 하시오.
④ 저장할 파일명 Ex_08.html 로 작성하시오.

HTML

```
<p id="text">색상이 바뀝니다</p>
<button id="colorBtn">색상 변경</button>
```

함수(function) 연습문제

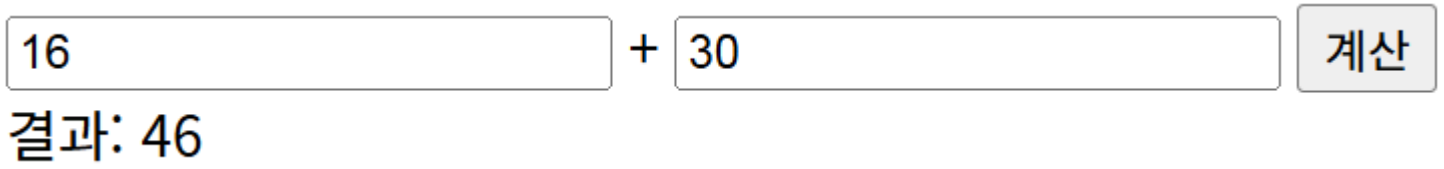
예제9] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 두 개의 입력값을 더해 결과를 출력하세요..
② 아래 주어진 <HTML> 이용해 작성하시오.
③ 저장할 파일명 Ex_09.html 로 작성하시오.

HTML

```
<input id="num1" type="number"> +
<input id="num2" type="number">
<button id="sumBtn">계산</button>
<div id="result"></div>
```



함수(function) 연습문제

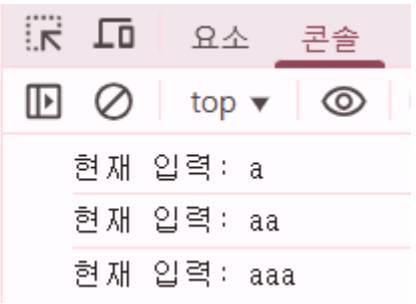
예제10] 아래 주어진 조건에 만족하도록 함수를 작성하시오.

조건

① 입력창에 글자를 입력할 때마다 현재 입력된 내용을 콘솔에 출력하세요.
② 아래 주어진 <HTML> 이용해 작성하시오.
③ addEventListener() , keyup 이벤트 이용해 작성하시오.
④ 저장할 파일명 Ex_010.html 로 작성하시오.

HTML

```
<input id="inputBox" type="text" placeholder="입력해보세요">
```



연습문제

함수(function) 연습문제 2

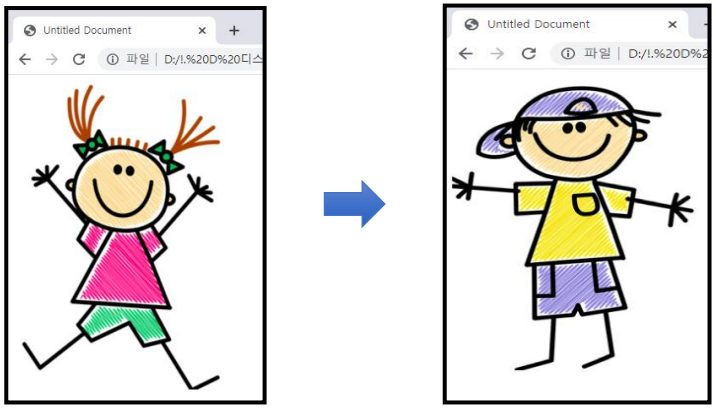
예제2] 아래 주어진 조건에 만족하도록 html 문서를 작성하시오.

조건

① 주어진 boy.png, girl.png 이미지를 이용하여 마우스가 오버되면 이미지가 변경되도록 하시오.
② 주어진 <html>을 이용하여 script를 작성하시오.
③ 저장할 파일명 Ex_changeImg.html 로 작성하시오.

HTML

```
<div id="container">
    <img  id="img" src="images/girl.png" >
</div>
```



함수(function) 연습문제 3

예제3] 아래 주어진 조건에 만족하도록 html 문서를 작성하시오.

조건

① 주어진 coffee-blue, coffee-gray, coffee-pink 이미지를 이용하여 아래 <출력 화면>처럼 작은 이미지를 클릭하면 큰 이미지로 변경되도록 작성하시오.

② 상세 설명 보기 클릭하면 아래 보이지 않았던 <상세설명>이 보이도록 작성하시오.

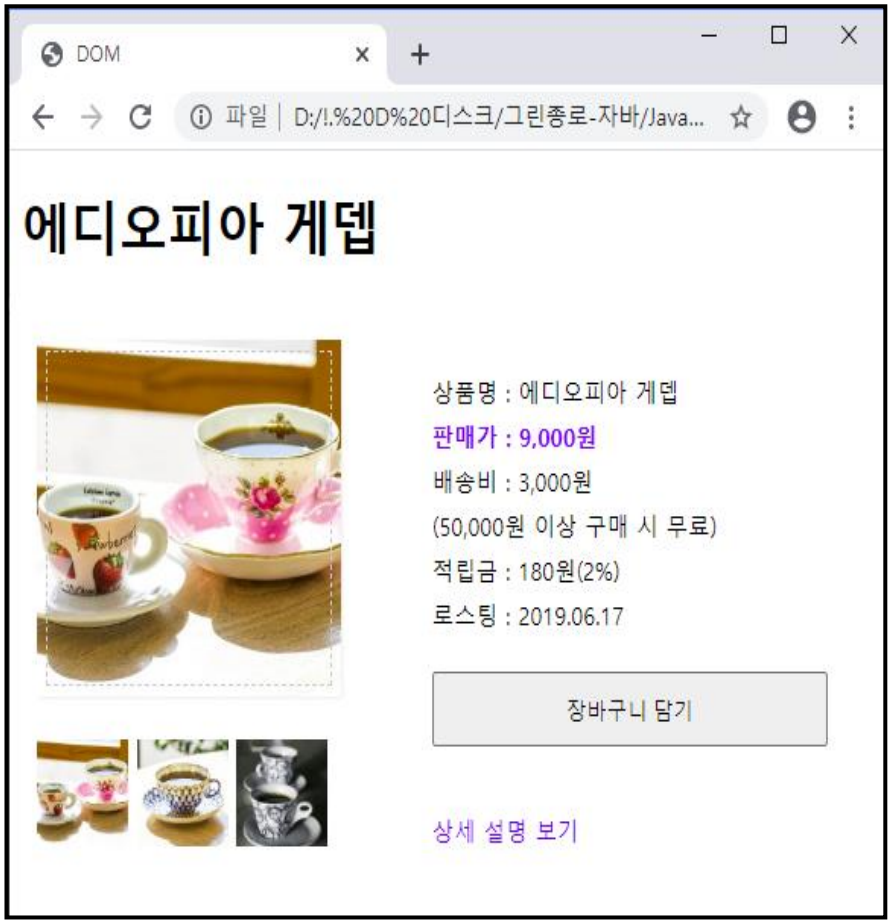
③ <html>을 아래 <출력 화면>처럼 작성하고, script를 작성하시오.

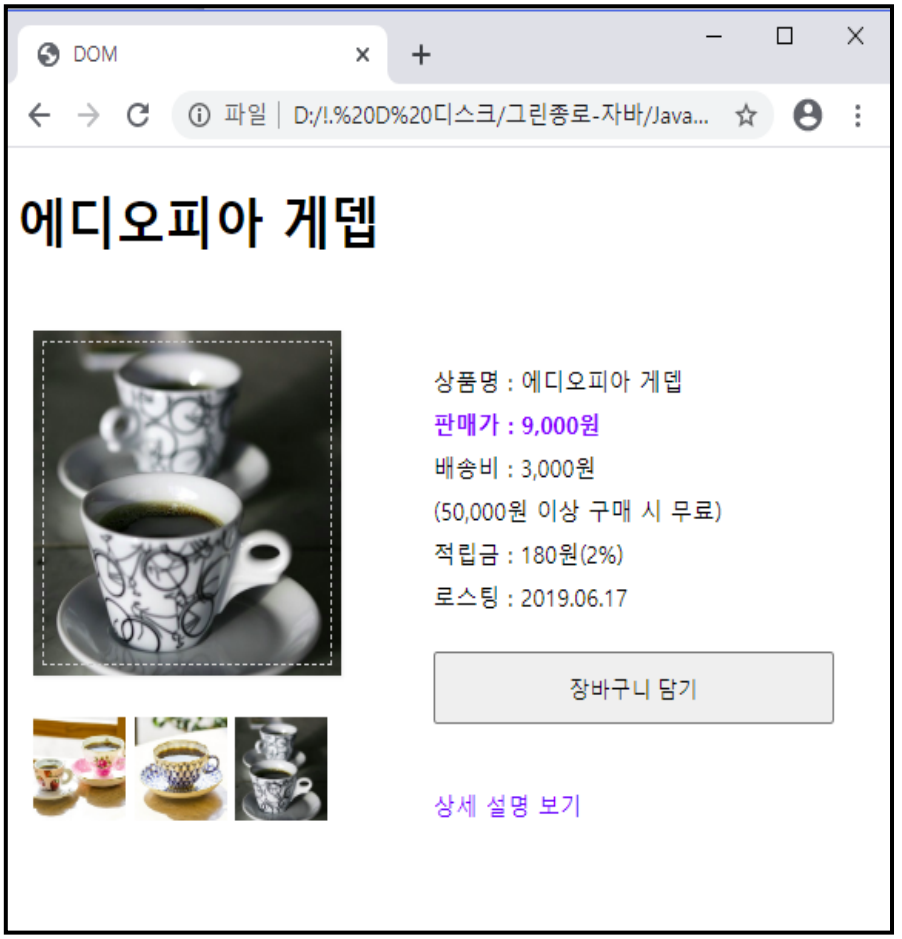
④ addEventListener( )을 이용하여 작성하시오.

⑤ 저장할 파일명 Ex_changeImg02.html 로 작성하시오.

함수(function) 연습문제 3의 출력 결과물

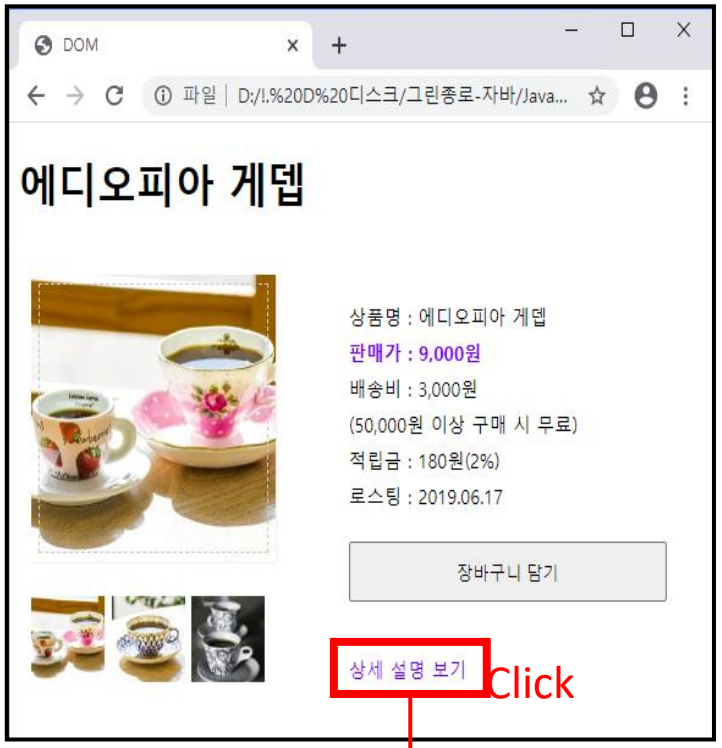
<출력 화면>





함수(function) 연습문제 3의 출력 결과물

<출력 화면>



상품 상세 정보

- 원산지 : 에디오피아
- 지 역 : 이르가체프 코체레
- 농 장 : 게뎁
- 고 도 : 1,950 ~ 2,000 m
- 품 종 : 지역 토착종
- 가공법 : 워시드

Information

2차 세계대전 이후 설립된 게뎁 농장은 유기농 인증 농장으로 여성의 고용 창출과 지역사회 발전에 기여하며 3대째 이어져 내려오는 오랜 역사를 가진 농장입니다. 게뎁 농장은 SCAA 인증을 받은 커피 품질관리 실험실을 갖추고 있어 철처한 관리를 통해 스페셜티 커피를 생산합니다.

Flavor Note

은은하고 다채로운 꽃향, 망고, 다크 체리, 달달함이 입안 가득.