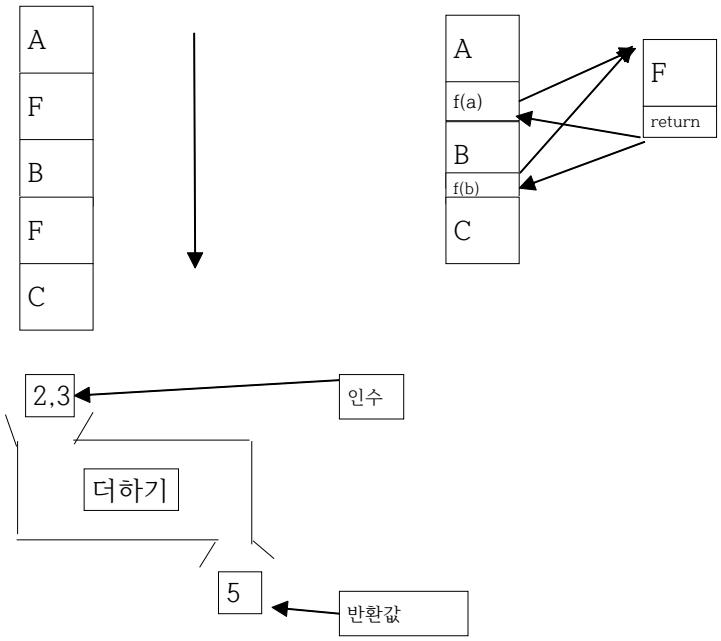
■함수
수학에서 함수는 주어진 입력값 x에 대한 출력값 y를 대응시키는 규칙

함수는 일련의 처리를 하나로 모아 언제든 호출할 수 있도록 만들어 둔 것이다
입력값을 받으면 출력값으로 함수의 결과값을 반환함
함수의 입력값은 인수 , 출력값을 반환값

코드블럭, (하나의 일처리를 위한 명령어 묶음)
양치질하기() => 양치질에 관련된 일련의 처리 과정 (수행해야할 명령의 묶음)

코드의 중복제거함, 코드의 집중화 시킴 , 유지보수에 좋은코드를 만들 수 있다.
중복되지 않더라도 함수로 만들기도 함, 코드분리목적
1000줄의 코드를 보는 것 보다 50줄의 코드를 보는 것이 작성 및 오류를 수정하는데 도움이 된다

관련된 기능으로 작게 함수로 분리하면 코드를 작성하거나 디버깅에 유리하다
코드의 가독성을 높인다.



■함수 선언문으로 함수 정의하기

```
function square(x) {
    return x*x;
}
```

■함수 선언문으로 함수 정의하기

```
function square(x) {
    return x*x;
}
```

■함수 리터럴로 함수 정의하기 (함수표현식)

```
let square = function(x) { return x *x ;} ;
```

둘 다 squares변수에 함수객체의 참조를 저장함
두 경우의 차이점
자바스크립트 엔진이 함수선언문으로 정의한 경우 끌어오림(호이스팅)이 발생함
함수표현식으로 정의한 경우 끌어올리지 않는다.

함수 호출시 여러개를 우아하게 전달하는 방법 (객체사용)
입력정보가 여러개일 때 (한 고객의 고개 정보, 이름, 나이, 주소 일 때)

```
function func1( name, age, address     ){  // // 객체로 전달받는 방법이 있다
     수행문
}
func1(  "홍길동", 25, "서울시" );


function func2( obj){
    수행문
}
let obj ={
        name: "홍길동",
        age: 25,
        address: "서울시"
 }
func2( obj) ;
```

전연변수:함수밖에서 선언, 유효범위가 전체프로그램이다 (변수 선언시 let 키워드는 반드시 사용한다)
지역변수: 함수내에서 선언, 함수내에서만 유효하다. 또한 블록의 범위를 갖는다.

■ 함수는 다름 함수의 인수로 넘겨질 수 있다.
즉 함수는 입력정보로 함수를 전달 받을 수 있다.

콜백함수 (나중에(특정이벤트가 발생했을 때, 특정시점에 도달했을 때) 호출되는 함수를 말함)
:
콜백 함수는 코드를 통해 명시적으로 호출하는 함수가 아니라,
개발자는 단지 함수를 동록하기만 하고, 어떤 이벤트가 발생했거나 특정 시점에 도달했을 때 시스템에서 호출하는 함수를 말한다. (다른함수에 인수로 넘겨지는 함수를 콜백함수라고 한다)

이벤트처리기 : 특정이벤트가 발생했을 때 실행하도록 등록하는 함수
타이머
setTimeout, setInterval에 첫 번째 인수로 넘기는 함수가 콜백함수이다.

■이벤트 처리하기

이벤트 리스너 등록하기

1. 태그내에 직접등록하기 <button onclick="show();" >버튼 </button>
2. addEventListener("click" , 이벤트핸들러함수)
   : 자바스크립트 내에서 등록할 수 있다
3. 이벤트리스너속성 사용하기
   onclick= 이벤트핸들러함수

2,3의차이

: 2번은 버튼에 여러개의 click이벤트 리스너를 처리할 수 있다.
: 3번은 버튼에 여러개의 click이벤트를 등록할 경우 마지막에 등록된 것만 처리할 수 있다.

<예제 : 이벤트리스너 매서드를 이용하여 이벤트 등록하기
addEventListener >

```
<!DOCTYPE html>
<html>
<head>
<script>
    window.addEventListener("load" , init);

    function init(){
        let btn = document.getElementById("btn");
        btn.addEventListener("click" , function(){ alert("나 클릭1") ;});
        btn.addEventListener("click" , function(){ alert("나 클릭2") ;});
    }

</script>
</head>
<body>

<button id="btn">버튼</button>
</body>
</html>
```

<예제 : 이벤트리스너 속성에 이벤트 등록하기>

```
<!DOCTYPE html>
<html>
<head>
<script>
    window.addEventListener("load" , init)
    function init(){
        let btn = document.getElementById("btn");
        btn.onclick = function(){ alert("나 클릭1");}
        btn.onclick = function(){ alert("나 클릭2");}
    }
</script>
</head>
<body>

<button id="btn">버튼</button>
</body>
</html>
```