23-11-08

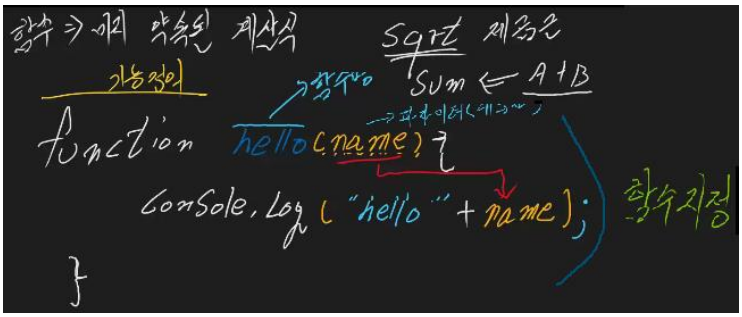
- 함수(function) 나가기

<함수>

함수 : 미리 약속된 계산식

- Sqrt : 제곱근 표현하는 함수
- Sum : A+B의 형태로 더하기로 약속되어 있다 => 더하기 함수

예제)



function : 기능(에 대해서)정의 -> 함수에 대해 지정해주는 것 **exm12.js**

```
function hello(name){
    console.log("hello"+name);
}
```

Function : 기능정의 -> 함수 지정하겠다는 것

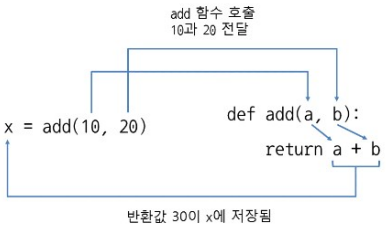
hello : 함수명 -> hello라는 함수를 지정하겠다는 것

(name) : **파라미터(대입자 : 대신입력하는 것)**

밑에 +name은 파라미터의 name을 그대로 가지고 오는 것이다.

console.log : 출력하라 hello 에 name에 해당하는 값 출력하라

⇨ 이 구조가 '함수 지정'하는 것

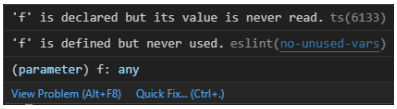



```
function add(a,b) { //함수명 add , 파라미터는 a,b로 변수 두개 받으라는 것
    return a+b; //돌려받을거야 어떻게, a+b의 형태로.
}
```

add 함수가 이런 형태인 것을 선언하는 것(덧셈을 하는 함수인 것을 선언)

덧셈을 하는 함수, 곱셈을 하는 함수가 지정이 되어 있지 않다.

여기서 지정해주는 것. 그래서 add를 덧셈함수로 지정해주었다.



파라미터(parameter)는 변수의 관련된 값으로, 위에서는 a와 b 변수 두개를 받으라고 하고 있다.

return은 돌려받는 것(반환). a+b로 a와 b의 덧셈형태로 돌려받을 거라는 것.(함수호출해준 바깥에 결과 알려주기 위해 사용)

Function 함수명(파라미터=변수에 관련된 값)

```
hello("안녕하세요");                    hello안녕하세요
console.log(add(3,4));                  7
```
함수를 정의해 놓고 난 다음에 함수명에 해당하는 값 집어넣으면 호출이 된다.

위에 함수 지정한 것을 불러준 것, add는 위에 함수안에 출력하라는 값이 없었기 때문에 console.log(add(3,4));를 해준 것

```
console.log(add(3));            NaN
```
값이 NaN으로 나온다 값을 하나만 써줬을 경우 값이 나오지 않는다.

```
console.log(add(3,));           NaN
```
값이 NaN으로 위와 같이 값이 똑같이 나온다.

```
console.log(add(,3));
```
```
SyntaxError: Unexpected token ','
    at internalCompileFunction (node:internal/vm:73:18)
    at wrapSafe (node:internal/modules/cjs/loader:1176:20)
    at Module._compile (node:internal/modules/cjs/loader:12
```
,로 인식할 수 없는 에러가 나온다.

```
console.log(add("",3));         3
```
빈칸과 3이 들어가니까 3이 출력되었다.

```
console.log(add("papa",3));     papa3
```
문자와 숫자가 더해진 값으로 나왔다, 이것은 문자던 숫자던 상관없이 작동한다는 것.

문자를 받아도 자료에 대한 지정을 하지 않는다 -> 자바스크립트의 장점이자 단점.

자바스크립트는 자료형 지정을 하지 않으므로

자료형 지정이 가능한 타입 스크립트가 추후에 나오게 된다.

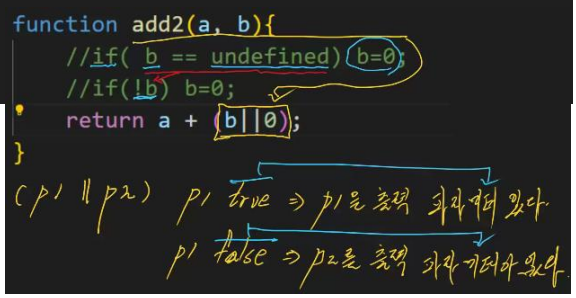
⇨ 파라미터 값이 제대로 들어오지 않으면 반환과 출력이 제대로 작동되지 않는다.

```
function add2(a,b) {
    if(b==undefined) b=0;
    return a+b;
}
console.log(add2(3,4));     7
console.log(add2(3));       3
```
b의 값이 undefined 지정되지 않으면 b=0으로 계산하라는 것

그래서 3만 넣어줬던 것에 결과 값이 3이 된다.

```
function add2(a,b) {
    // if(b==undefined) b=0;
    if(!b) b=0; //!b 의 내용은 b==undefined 와 같다, b 의 값이 비어있으면 이라는 뜻
    return a+b;
}
console.log(add2(3,4));   7
console.log(add2(3));     3
```
'if(!b)' == 'b==undefined' 와 같은 의미. B의 값이 비어있으면 b=0을 집어넣어라 라는 것

```
function add2(a,b) {
    // if(b==undefined) b=0;
    // if(!b) b=0;
    return a+(b||0); //자료값이 없으면 0으로 대체해라 라는 것
}
console.log(add2(3,4));   7
console.log(add2(3));     3
```



a+(b||0); -> 자료값이 없으면 0으로 대체하라는 것.

(b==undefined) b=0; 전체와 같은 의미

(p1 || p2) => p1 이 true 이면 p1 을 출력(True 라는 건, 파라미터(값)가 있다는 것

=> p1 이 false 이면 p2 를 출력(false 라는 건, 파라미터(값)가 없다는 것

가변파라미터 : 파라미터의 개수를 정해놓은 것이 아님

```
function power(...a) {
    let result = 0;
    for(let i=0; i<a.length; ++i){ //a.length는 위에 값을 배열처럼 받겠다는 것
        result += a[i]; //result에 값 추가하라는 것
        //result = result + a[i];
    }
        return result;
}                                   10
console.log(power(1,2,3,4));
```

power(...a)를 넣어준 것은 값이 정해져있지 않은 가변 파라미터를 넣어준 것이다

a.length 는 위에 가변 파라미터의 개수를 배열처럼 받은 그 값의 길이만큼 작동하겠다는 것

```
function power2() { //power1과 2는 동일한 기능을 하는 함수
    let result = 0;
    for(let i=0; i<arguments.length; ++i){
        result = result + arguments[i];
    }
        return result;
}
console.log(power2(1,2,3,4));       10
```

power 와 power2 는 동일한 기능을 하는 함수이다.

arguments 를 사용하면 따로 변수 사용하지 않아도 배열로 받겠다는 의미이다.

<Callback 함수>

exm13.js

```
//callback 함수
function add(a,b){
    return a+b;
}
let a=add(3,4);
console.log(a);
```

```
7
```

add 함수를 a 가 callback 했다.

a 라는 변수에 add 함수를 저장한 것이다. -> a 를 출력해도 add 가 출력되는 구조

```
let f = add; //f 에다가 add 를 콜백 불러오라는 것.
console.log(typeof f); //typeof 는 타입의 종류 알려주는 것

let b = f(3,4);
console.log(b);
```

```
function
7
```

f에 add를 불러오라는 것이다. b는 f(즉,add에 3과4를 넣는 것)에 값을 넣어줬고 그 b를 출력함

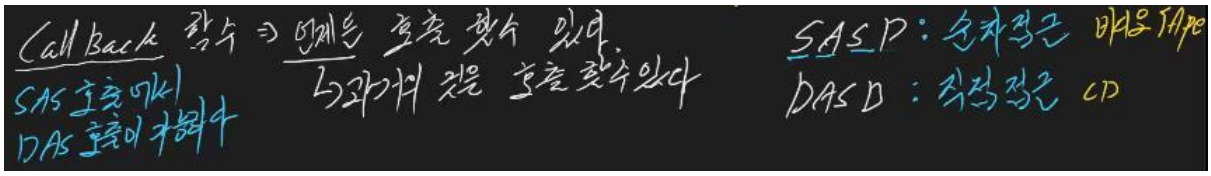
Callback 하고 불러내고 집어넣고 불러내고 이렇게 사용하는 것이 없다.

그런데 자바스크립트는 이런 Callback 을 사용한다.

함수를 재사용하고 불러내려고 할 때마다 이렇게 callback 을 사용한다

Callback 의 의미 : 빠르게 처리할 것들은 빠르게 처리하고
느리게 처리되는 것들은 느리게 처리하기 위한 것

결국 위의 b 의 값은 add 의 값이 출력되는 것이다. => 이것이 이해되면 됨

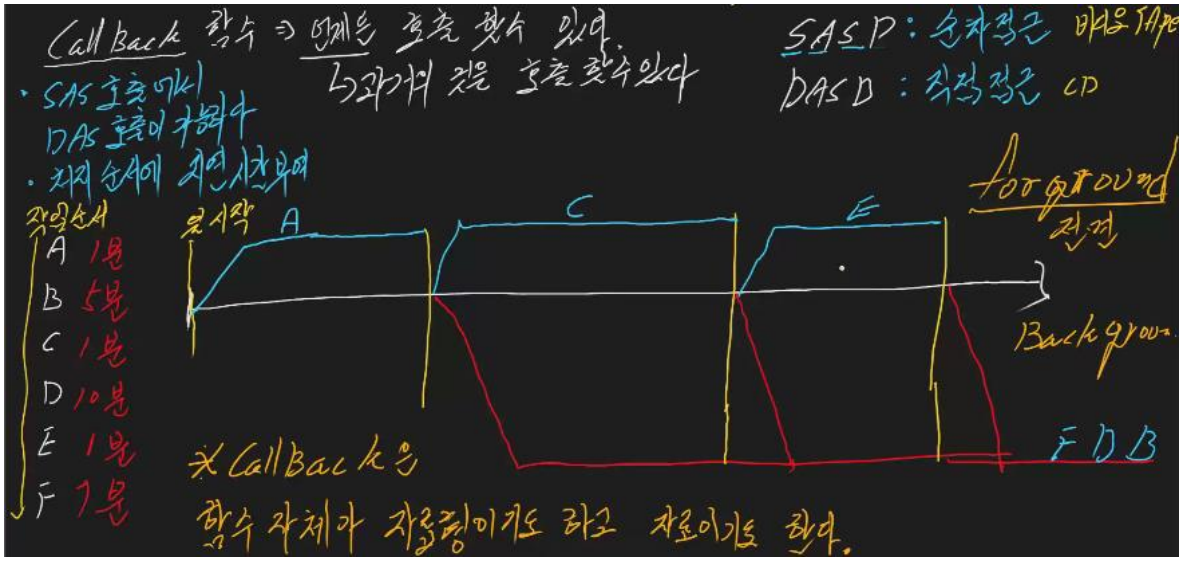


Callback 함수 => 언제든 호출할 수 있다. (**언제든** => 과거의 것을 호출할 수 있다는 것)

- SASP : 순차접근 (ex_비디오테이프) / 저장장치, SAS 로도 사용할 수 있다
- DASD : 직접접근 (ex_CD)

➔ SAS 호출에서 DAS 호출이 가능하다

➔ 처리 순서에 지연 시간 부여



A~F 까지 작업순서대로 진행할 때
순서대로 하면 SASP 이고, 내가 원하는 것만 작업하는 것이 DASD 이다.

- ⇨ A 를 작업시작했다. 1 분동안 간다 (노란색선이 1 분기준)
- ⇨ 그리고 B 를 작업하려고 보니 5 분이 너무 긴 것 -> background 로 보내버림
  따로 작업하는 back 으로 보내버린 것 (아래 빨간색선)
- ⇨ C 를 작업함 1 분 걸리니 작업시작
- ⇨ D 는 10 분이 걸려서 다시 background 로 보냄
- ⇨ E 는 1 분이라서 작업
- 적은 시간들은 빨리 끝내버리고 시간이 긴 것은 back 단으로 보내버리는 것
  눈에 보이는 것 처리한 다음 뒷단(화면상)보여지지 않는 것들, 일의 순서대로 처리하지 않아도 되는 것들을 처리한다.
- ⇨ F 는 7 분으로 back 단으로 보냄

여기서 전면에 처리되는 것이 Forground(전면)이고, 뒤에서 처리되는 것이 Background 이다.

기본개념은 forground 에서 빨리 진행하고, 다 처리가 되면 background 에 있는 걸 처리하는 것
(눈에보이는 것 빨리처리, 화면상 출력안되는 건 back 으로 보냄)
(우선순위를 화면출력에 관련된 forground 에 주는 것)

➔ 이때 사용하는 것이 **callback** 함수이다.

```
callback

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

let a = add(3,4);
console.log(a);

let f = add;
console.log(typeof f);

let b = f(3, 4);
console.log(b);
```

그래서 이렇게 함수 지정해놓은 것

위에 각 1 분 10 분 1 분이라고 가정하면 1 분짜리를 먼저처리하는 것 10 분은 백그라운드로 보냄

➔ 이것이 Callback

여기서 멀티프로세서와 멀티프로세싱을 기억해야 한다.

여러 개 계산할 수 있는 계산기가 여러 개 있으면 한번에 돌릴 수 있는 것이 많아지는 것

➔ 프로세서가 2 개면 각각 2 개니까 4 개. 한번에 4 개가 가능
그래서 back 으로 돌릴 수 있다

⇨ 콜백함수(callback 함수) 기본적개념

*** CallBack 은 함수 자체가 자료형이기도 하고 자료이기도 하다.**

```
callback

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

let a = add(3,4);
console.log(a);

let f = add;
console.log(typeof f);

let b = f(3, 4);
console.log(b);
```

add함수는 a의 자료가 된다

typeof f하면 function이 나온다 자료형이 된다

자료형 가지고 f인 자료형가지고
데이터가 출력된다

자료형과 자료가 얽혀있는건
나중에 콜백을 쉽게 하기 위해서이다

자바스크립트는 데이터 자료 형태 상관없이 쓰였다.

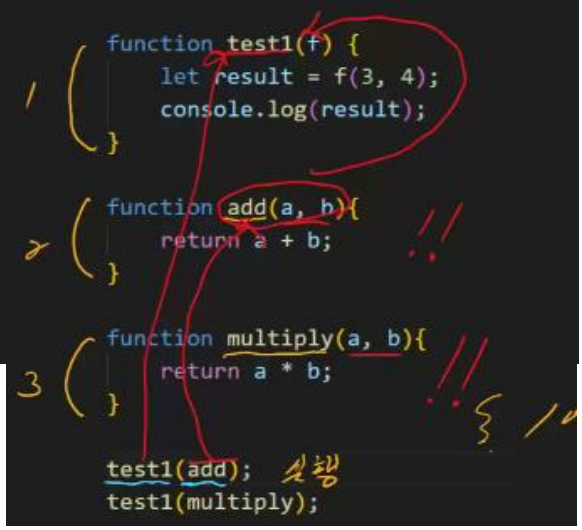
```
//콜백함수예제
function add(a, b);{
    return a + b;
}
let f = add;
```

위와 아래는 완전히 똑같은 코드이다.

```
let f = function(a,b); {//위에것을 이렇게 만들 수도 있다.
    return a + b;
}
```

```
function test1(f) {
    let result = f(3,4);
    console.log(result);
}

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

function multiply(a, b){
    return a * b;
}
test1(add);
test1(multiply);
```

```
7
12
```



test1 이 부분은 이미 구동되어 있는 것.

1,2,3 이 함수로 정의되어 있다.

Test1 이 실행되는 순간 function test1(f)으로 가서 f 에 add 가 실행된다.

(add 가 실행되는 과정 : add -> 함수 add -> test1(F)안으로 넣는다)

```
test1(add);
```

이부분이 실행되면 => **과거 정의된 함수를 호출하여 사용한다 CallBack**

```
test1(multiply);
```

이 부분이 실행되면 => **다른 함수의 파라미터의 DATA 로 전달되게 호출되는 함수**

```
function multiply(a, b){
    return a * b;
}
```

Test1 부분은 아래 부분

multiply 부분은 위에 함수가 호출된다.

```
function test1(f) {
    let result = f(3,4);
    console.log(result);
}
```

F 에 multiply 에서 return 된 값 a*b 가 넣어지고 그 값이 출력되는 것이다.

결국 test1(multiply) => 3 번 multiply 함수로 => 1 번 test1 함수를 실행

다른 함수의 파라미터의 DATA 로 전달되게 호출되는 함수

```
test1(add);
test1(multiply);
```

⇨ ☆ **이 두개가 Callback 함수인 것**. ☆

```
//위와 똑같지만 add 와 multiply 를 function 함수로 만들지 않을 거다
function test1(f) {
    let result = f(3, 4);
    console.log(result);
}

let add = (a, b) => { //=>화살표로도 콜백함수 지정이 가능하다
    return a + b;
}
let multiply = (a, b) => {
    return a * b;
}
test1(add);
test1(multiply);
```

함수명 지정하고 리턴 값 주는 방법에서 => 를 사용하여 add 는 a,b 자료형으로 받는다고 알려줌

그것을 a+b 로 만들어서 보내라고 하는 것

방법은 위에 것과 지금 한 것 두가지 있다 function 로 사용하는 것을 선생님은 추천

(함수를 사용하면 => 보다 양이 많아지니까 처리속도 느려진다고 하지만 심각한 정도는 아님)

```
function test2(f) {
    let result = f(5, 7);
    console.log(result);
}
test2((a, b) => {return a + b;}) //한줄로 처리가능
test2((a, b) => {return a * b;})
```

```
12
35
```

test(add); 이렇게 처리했던 것을

test2 에서는 한번에 합쳐서 한줄로 처리한 것이다.

(a,b)=>{return a+b} 의 값을 위에 test2 함수안에 파라미터 f 로 보내준 것.

콜백은 여기까지

나중에 실제 자료 나오는 것으로 작업할 것.

exm14.js

```
//단어와 단어가 합해질 때는 그다음 올 때 대문자로 시작하는 것(printTime)
function printTime (msg){ //msg 파라미터를 통해서
    console.log(msg, new Date()); //생성자 새롭게 시작할 때 붙여줌
data 는 내 컴퓨터 기준의 시스템의 시간을 가지고 옴(동기화해라 라는 의미)
}
setTimeout(printTime,1000,"1 초 후"); //내 컴퓨터에 입력되어 있는 자료값 출력할거야
setTimeout(printTime,2000,"2 초 후");
setTimeout(printTime,3000,"3 초 후");
```

```
PS C:\work\javascript\11-08> node exm14.js
1초 후 2023-11-08T05:16:27.602Z
2초 후 2023-11-08T05:16:28.613Z
3초 후 2023-11-08T05:16:29.601Z
```

steTimeout : 시간에 관련된 callback 함수이다.
실제 메시지가 1 초후에 뜨게 된다.

순서가 바뀌어도 더 빠른 초가 먼저 실행된다.

ctrl + c => 터미널에서 강제종료

```
setTimeout(printTime, 5000, "5초 후");
setTimeout(printTime, 2000, "2초 후");
setTimeout(printTime, 3000, "3초 후");
```

```
PS C:\work\javascript> node exm14
2초 후 2023-11-08T05:38:38.710Z
3초 후 2023-11-08T05:38:39.713Z
5초 후 2023-11-08T05:38:41.711Z
```

callback 안에 callback 호출 => callback 이 계속 돌아가서 빠져나오지 못함(무한루프)

exm15.js

```
let person = {name:"박원일", age:17};
console.log(person);
console.log(person.name);
console.log(person.age);
```

```
{ name: '박원일', age: 17 }
박원일
17
```

console.log(person); => 여기 안에 person 이 오브젝트다.

정보가 담길 공간이 필요하다 => object , 객체로 만든다

그리고 이렇게 데이터 값을 넣어서 만들게 되는 것.

```
let person1 = {}; //빈객체 만드는 것
```

[ ] 는 빈 배열 만드는 것 { } 는 빈 객체 만드는 것

객체를 만드는 방법

```
//1
let person = {name:"박원일", age:17};
console.log(person);
console.log(person.name);
console.log(person.age);

//2
let person1 = {};
person1.name = "박원일";
person1.age = 17;
console.log(person1);

//3
let person2 = {name:"박원일"};
console.log(person2);
person2.age=53;
console.log(person2)

//4
function createPerson(s,i){
    return { name:s, age:i};
} // callback 함수와 똑같아 진다

let person1 = createPerson("박원일",17);
let person2 = createPerson("박진영",19);

console.log(person1);
console.log(person2);
```

```
PS C:\work\javascript\11-08> node exm15.
{ name: '박원일', age: 17 }
박원일
17
{ name: '박원일', age: 17 }
{ name: '박원일' }
{ name: '박원일', age: 53 }
{ name: '박원일', age: 17 }
{ name: '박진영', age: 19 }
```

여기서 비교를 해보자

```
let person1 = createPerson("박원일",17);
let person2 = createPerson("박진영",19);
let p =person1

console.log(person1);                    { name: '박원일', age: 17 }
console.log(person2);                    { name: '박진영', age: 19 }
console.log(person1 == person2);         false
console.log(person1 == p);               true
```

true와 false의 값으로 나온다 비교를 할 때 수치로 하는 것이 아니라 같은 값인지를 비교하는 것

궁금한 점 : 약타입언어일 때 반대로 나온다고 했는데 이거는 그대로 나온 이유

```
function equals(person1, person2){
    return person1.name == person2.name && person1.age == person2.age;
}
console.log(equals);                     [Function: equals]
```

자료 값이 없기 때문에 위 결과와 같이 나옴

```
console.log(equals(person1, person2));   false
```

위의 나이 값을 같게 주면 어떤 값이 나올까? False 가 나온다 => &&로 두가지 조건이 다 참일 때 True 를 출력하기 때문이다.

return 값에 ||으로 주게 되면 값이 하나만 맞아도 true 를 주기 때문에 나이가 같아서 True 가 나옴

자바스크립트는 동일한 이름이 존재하면 못 찾기 때문에 작동하지 않는다.

⇨ **자바스크립트 object 객체명은 무조건 유일해야 한다.**

```
let ps1 = {name:"박지영", age:48};
let ps2 = {name:"박지영", age:17};
let ps3 = {name:"박태민", age:20};
let ps4 = {name:"박태석", age:26};

let person = [ps1, ps2, ps3, ps4]; //person 을 배열로 만들었다.
console.log(person);
```

```
[
  { name: '박지영', age: 48 },
  { name: '박지영', age: 17 },
  { name: '박태민', age: 20 },
  { name: '박태석', age: 26 }
]
```

객체 만든 것을 배열안에 넣을 수 있다.

```
for (let i=0; i<person.length; i++){
    console.log(person[i]);
}
```

```
{ name: '박지영', age: 48 }
{ name: '박지영', age: 17 }
{ name: '박태민', age: 20 }
{ name: '박태석', age: 26 }
```

이렇게 했을 때는 각각 객체의 값으로 나온다.

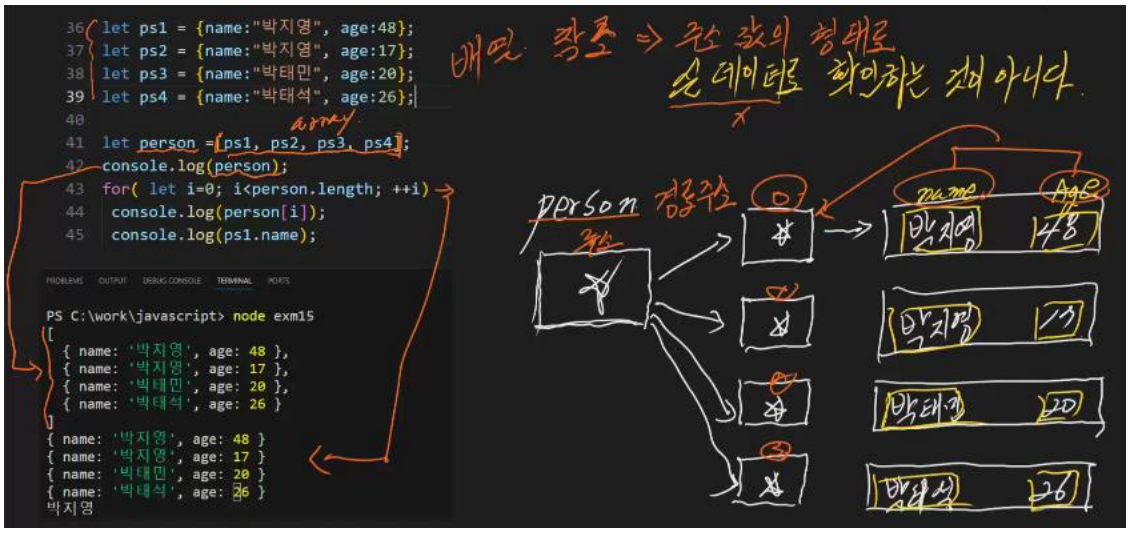
```
console.log(person.name);
```

person.name 하면 값이 나오지 않는다 이유는 각 ps1,ps2,ps3,ps4 에 name 이 있는 것이기 때문에 그렇기 때문에 ps1.name 하면 값이 나온다.

```
45  console.log(ps1.name);
```
박지영

참조 배열 => 주소 값의 형태로 저장된다. 실 데이터로 확인하는 것이 아니다.

Person 은 대표주소 하나만 기억한다. (데이터 값을 기억하는 것이 아니다.)
0,1,2,3 번 주소들을 기억하고 그 값들은 각각 그 주소와 관련된 name 과 age 만 기억한다.
person 은 경로 주소만 아는 것이다. => 그래서 person.name 했을 때 값이 나오지 않았다.

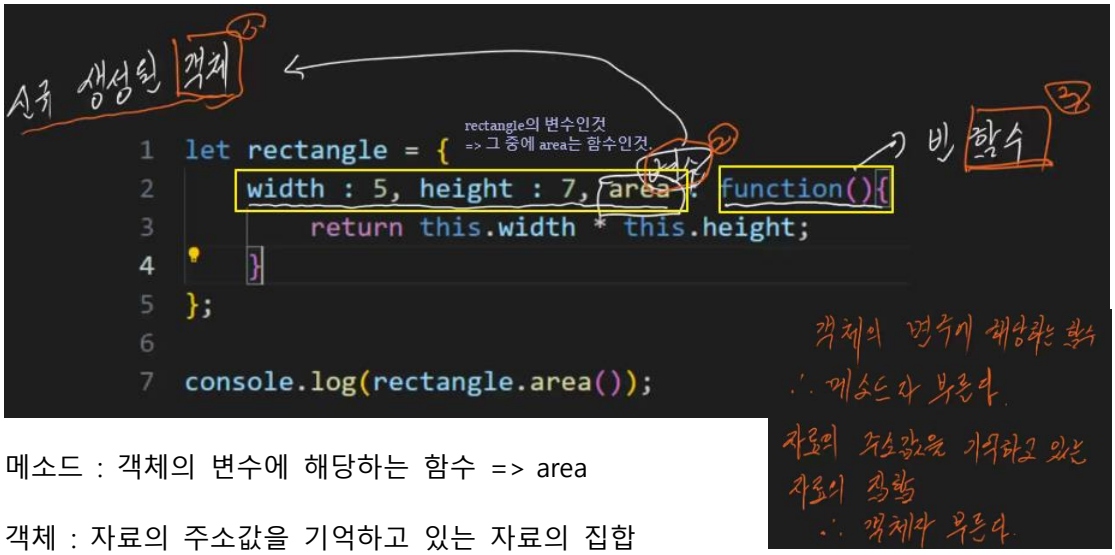


Person 에 연결관 각 0,1,2,3 에는 대응하는 실제값을 가지고 있다.

exm16.js

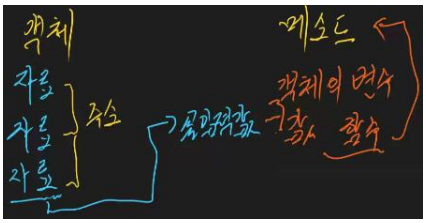
```
let rectangle = { //면적구하는 함수
    width : 5, height : 7, area : function(){
        return this.width * this.height; //this 라는 생성자 만들기
    }
}

console.log(rectangle.area());
```
35

rectangle 이라는 객체를 하나 만들거다 라는 것. 그리고 그 내용이 { } 안에 들어있다.
변수는 width 와 height 를 잡아서 area 인 함수에 넣는 것
파라미터(width,height) 바깥에 넣어놓고, 함수명(area)도 바깥에 넣어놓고 **기능정의**를 이렇게 한것
this 는 위에 width 와 height 를 각각 대응하기 위한 것/ 에 width 의 5 와 height 의 7 을 가지고 오는 것



메소드 : 객체의 변수에 해당하는 함수 => area

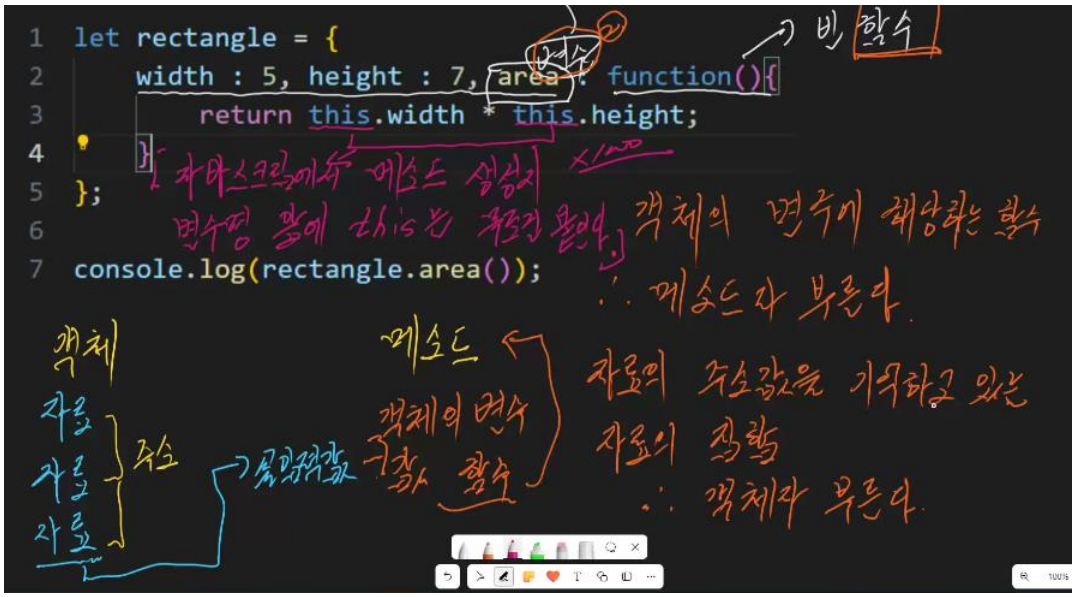
객체 : 자료의 주소값을 기억하고 있는 자료의 집합



객체 안에는 자료들이 존재한다 그 자료들을 주소로 기억한다.

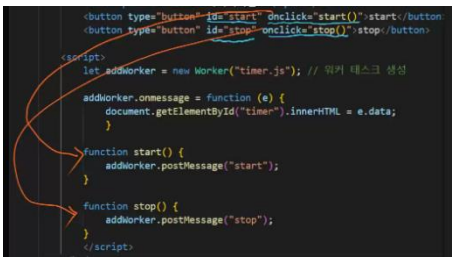
해당하는 자료의 실 입력값 => 객체의 변수 값이고 그것을 함수로 표현하는 것이 메소드이다.

☆☆☆ This => 자바스크립트에서 메소드 생성시 변수명 앞에 this 는 무조건 붙인다 (필수)



(기능적으로 많이 들어가 있으면 함수이고,

그 안에 이름 자료들이 들어가면 객체)



예제)

<html부분>

main.html

```
<button type="button" id="start" onclick="start()">start</button>
```

Onclick->Start로 가라는 것, id 지정해주면 어느 부분인지 알 수 있다.

```
<button type="button" id="stop" onclick="stop()">stop</button
```

Onclick->stop으로 가라는 것

```
let addWorker = new Worker("timer.js"); // 워커 태스크 생성
```

Timer.js를 가지고 새로운 worker을 만들어서 새로운 객체 addworker를 생성할 거라는 것

```
addWorker.onmessage = function (e) { /start를 누르면 변화가 생긴다
```

onmessage : 화면상에서 새로운 일이 발생이 되면 addworker에 새로운 메시지 불러올 거야

-> onmessage자체는 브라우저에서 무슨일이 발생이 되면 이라는 뜻!!

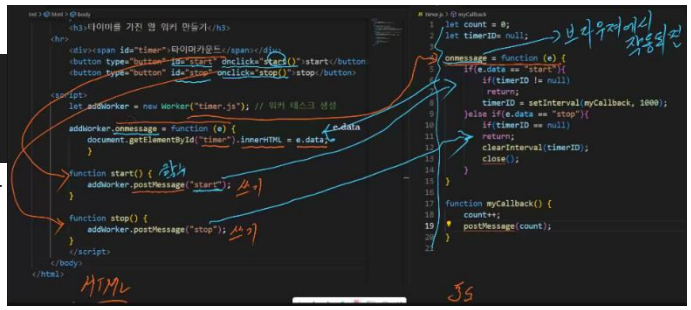
그렇게 되면 function이라는 기능함수 만들겠다는 것이다.

```
    document.getElementById("timer").innerHTML = e.data;}
```

getElementByID 엘리멘트아이디를 가지고 올거야 자바스크립트 timer를 가지고 와서

html에 e에 데이터로 만들어서 작업을 시작할거야 (e.data)

e.data는 가지고 올거라고 하는 것



```
function start() {
    addWorker.postMessage("start"); }
// start라고 쓰라는 것
```

새로운 메시지를 쓸 거라고 하는 것이postMessage이다

js부분에서 사용했다.

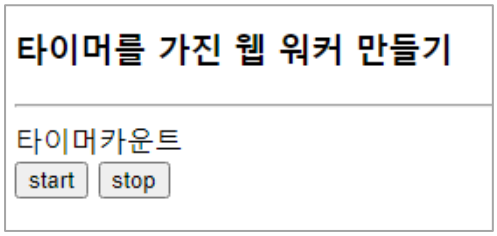
post가 쓰기다

addworker는 스크립트 파일 안에다가 "start"라고 메시지를 쓰라는 것 => 작업하라는 것

⇨ Start를 누르면 새로운 작업 발생 -> onmessage

⇨ onmesaage에서 function e로 가고 그것은

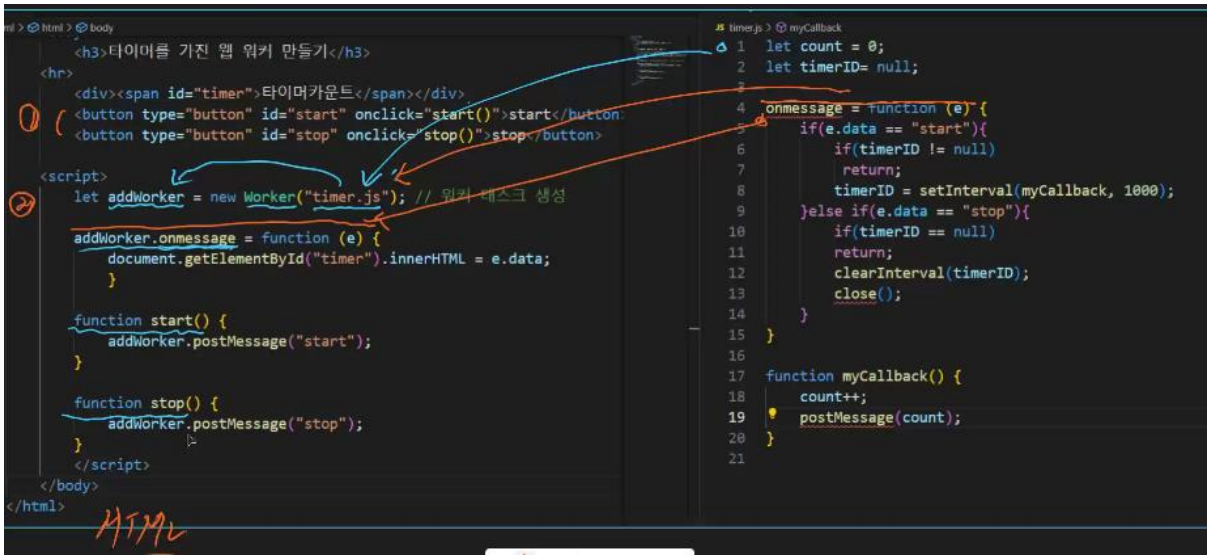
⇨ timer.js와 연결되는 것 (브라우저에서 작동되면 onmessage 작동)



<js부분>

Timer.js

```
let count =0;
let timerID = null;

onmessage = function(e) {
    if(e.data == "start"){
        if(timerID != null)  //비어있지 않으면 작동하고 있다는 이야기
            return; //작동중이면 처음으로 돌아와라
        timerID = setInterval(mycallback, 1000) //timerID 에 간격설정해라
(mycallback 내가 다시 부를 거야 1000 값 받아) 1 초간격으로 세팅하라는 것, 콜백 함수
    }
    else if(e.data == "stop"){ //onmessage 가 stop 으로 오면
//else 줘도 되지만 혹시 모르니까 조건을 준 것이다
//다른 클릭해도 이거 작동될 수도 있으니까 조건 준 것(else 안쓰고 else if 쓴 것)
        if(timerID == null)
            return; //다시 처음으로 되돌려라
        clearInterval(timerID) //깨끗하게 지우기, timerID 초기화 시킬거야
        close() //그러고 나서 화면 닫을거야
    }
}
function myCallback() {
    count++;  // 타이머 카운트 주는 것 시작하면 숫자 올라간다
    postMessage(count);
}
```



1. 아이디 지정 (timer, start, stop)
   클릭되었는지 알려줘야 하는데 그 때 필요한 것이 ID이다.

2. Addworker => callback함수 주는 것, 타이머 자바스크립트 파일을 끌고와서 새로운 Worker를 만들건데 그것이 addworker이다!!

   Addworker.Onmessage = function (e) => 브라우저 화면에서 작동이 되면
   id로 지정해준 timer를 불러와서 내꺼 HTML파일 내에서 작동할거야
   e.data : e로 만들어진 데이터 형태로 만들어진 것으로 작업할거야
   (함수부분은 HTML과 JS 부분이 형태가 같다. Onmessage 부분! 즉, 자료구조가 같다)

1. 자바스크립트의 함수 부분이 실행되는데
   해당 자료는 스크립트 파일에 만드는 것이다.
   script부분이 하나도 작업되지 않으면 -> JS에서 작업되지 않는다
   즉, 자바스크립트가 실행되지 않으며, 자바스크립트로 넘어갈 일 없다.

   Data가 start가 들어오면 1초마다 mycallback함수에 세팅을 해주고
   Data가 stop이 들어오면 깨끗하게 지우고 화면을 닫으라고 코딩해주고 있다.

   Function myCallback()함수가 작동이 된다는 것은 start가 되었다는 것이고
   초기설정해준 count=0에 1초마다 1씩 증가한 값을 postMessage로 보여준다.

문제

1. Console.log를 밖에 써주지 않아도 함수안에 출력하는 값 쓸 수 있는지?

```
function add(a,b) {
    return console.log(a+b);
}
add(3,4);
```

7

함수 값 안에 출력하는 console.log를 추가하여 만들었다.

그렇게 되면 add함수(파라미터포함)만 써줘도 값이 출력된다.

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

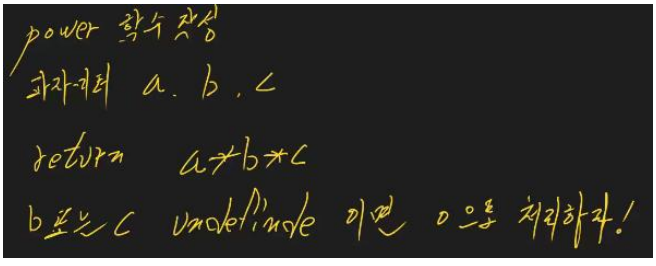
이렇게 쓰면 아래와 같이 오류가 나옴

```
PS C:\work\javascript> node exm12
C:\work\javascript\exm12.js:6
   console.log( return a+b);
                ^^^^^^

SyntaxError: Unexpected token 'return'
    at internalCompileFunction (node:internal/vm:73:18)
    at wrapSafe (node:internal/modules/cjs/loader:1176:20)
    at Module._compile (node:internal/modules/cjs/loader:1218:27)
    at Module._extensions..js (node:internal/modules/cjs/loader:1308:10)
    at Module.load (node:internal/modules/cjs/loader:1117:32)
    at Module._load (node:internal/modules/cjs/loader:958:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:23:47

Node.js v18.14.1
PS C:\work\javascript>
```

console.log 안에 return값을 사용할 수 없다고 에러가 나옴

Return을 쓰는 이유는 해당하는 값을 가지고 있어라 라는 것

2.



Power 함수 작성

파자미터 a,b,c

Return a*b*c 형태로 받을 것

B 또는 c 가 undefined 이면, 0 으로 처리하라

```
//문제 2
function power(a,b,c) {
    // if(b==undefined) b=0;
    // if(c==undefined) c=0;
    // return a*b*c;
    return a*(b||0)*(c||0);
}
console.log(power(2,3,4));
console.log(power(3));
```

두가지 방법으로 풀었다.

if문으로 각 b와 c를 지정해주는 것과 return에서 바로 b와 c가 값이 false이면0 처리하는것으로함

헷갈리는 부분 :

> *위에 c if를 할 때 처음에 else if를 써줬다. 그랬을 때 NaN의 값이 나왔는데*
> *else if라는 것은 위에 b==undefined일 때 c도 undefined이면 실행하라는 것이었다.*
> *그래서 if로 바꿔주었더니 b와 c에 값이 없으면 0으로 잘 나온다.*

여러가지 방법이 나온다 그러나 추천하는 방법은 아래와 같이 if를 사용하는 것
문법을 배우는 것이고 문법을 먼저 잘 아는 것이 중요하기 때문이다.

```
    if(b==undefined) b=0;
    if(c==undefined) c=0;
    return a*b*c;
```

문법을 잘 알고 사용할 줄 알면 간단한 방법 코드 줄 적게 나오는 아래방법 사용해도 됨

```
function power(a,b,c){
    return a*(b||0)*(c||0);
}

console.log(power(3, 3, 3));
```

<질문사항>

```
let person1 = createPerson("박원일",17);
let person2 = createPerson("박진영",19);
let p =person1

console.log(person1);
console.log(person2);
console.log(person1 == person2);
console.log(person1 == p);
```

createPerson은 함수인가?
여기서 person1은 객체인가?

```
{ name: '박원일', age: 17 }
{ name: '박진영', age: 19 }
false
true
```

true와 false의 값으로 나온다 비교를 할 때 수치로 하는 것이 아니라 같은 값인지를 비교하는 것

궁금한 점 : 약타입언어일 때 반대로 나온다고 했는데 이거는 그대로 나온 이유

1. createPerson은 함수인가? / person1은 객체인가?

2. 약타입언어일 때 True이면 False가 나오고 False이면 True가 나온다고 하는데 여기에서는 값이 정상적으로 나왔다 왜그런것인가??

```
let rectangle = { //면적구하는 함수
    width : 5, height : 7, area : function(){
        return this.width * this.height; //this 라는 생성자 만들기
    }
}

console.log(rectangle.area());
```

객체인것인가?

```
35
```

3. rectangle은 객체인가?