231108

작성 일시	@2023년 11월 8일 오전 9:44
태그	

함수- 미리 약속된 계산식

ex)sum= A+B (sum=함수)

function - 기능 정의(함수 지정)

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

hello(" 변수");

PowerShell
hello 변수
```

hello - 함수명

(name) - 파라미터 (대입자)

console.log("hello"+ name) 의 name 은 파라미터의 정보값 가져온다.

hello의 name에 해당하는 파라미터를 가져온다.

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

PowerShell
7
```

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

add(124,1456)
add("papa",2)
```

```
PowerShell
1580
papa2
```

JavaScript는 자료형 지정을 하지 않음으로 자료형 지정이 가능한 타입 스크립트가 추후에 나온다.

```
function add2(a,b){
    *if(!b) b=0      /* !b b에 값이 비어있을경우(!b)=(b == undefined)  */
    return a+b;
}
*if(b == undefined) b=0;
*return a+(b||0);    /* 자료값이 없으면 0으로 대체 (b == undefined) b=0 와 동일 */

console.log(add(6,4));
console.log(add(6));

console.log(add2(6,4));
console.log(add2(6));

PowerShell
10
NaN
10
6
```

(p1 || p2) p1이 true → p1을 출력 (p1파라미터 O)

p1이 false → p2를 출력(p1파라미터 X)

Power 함수 작성

```
/* ex1 */
function power(a,b,c){
    if(b == undefined || c == undefined) b=0; c=0;
    return (a*b*c)
}
/* ex2 */
/*(!b||!c) = (b == undefined || c == undefined)*/
function power(a,b,c){
    if (!b||!c) b=0; c=0;
    console.log(a*b*c);
}
/* ex3 */
```

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

power(6,6,);

PowerShell
0
```

```
// 가변파라미터 (...a) :파라미터의 값을 정해 두지 않는다.
function power(...a){ /* 자료를 배열 형식으로 받는다. */
    let result = 0;
    for(let i=0; i<a.length; i++ )//a.length 대신 arguments.langth[power()] 배열로 받는다.
    // result = result + a[i];
    result += a[i];
    return result;
}

console.log(power(3, 3, 3, 1));
console.log(power(1,2,3,4,5,6));
console.log(power());

PowerShell
10
21
0
```

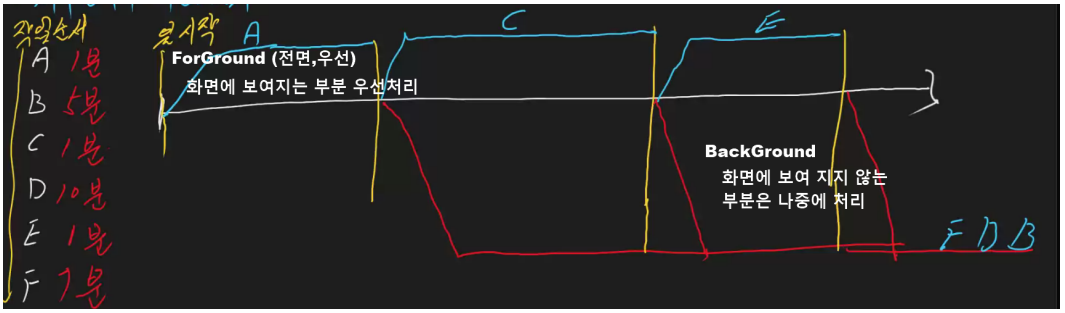
CallBack 함수 ⇒ 언제든 호출(과거의 것) 할 수 있다.

SASP : 순차 접근 [비디오TAPE]

DASD : 직접 접근 [CD]

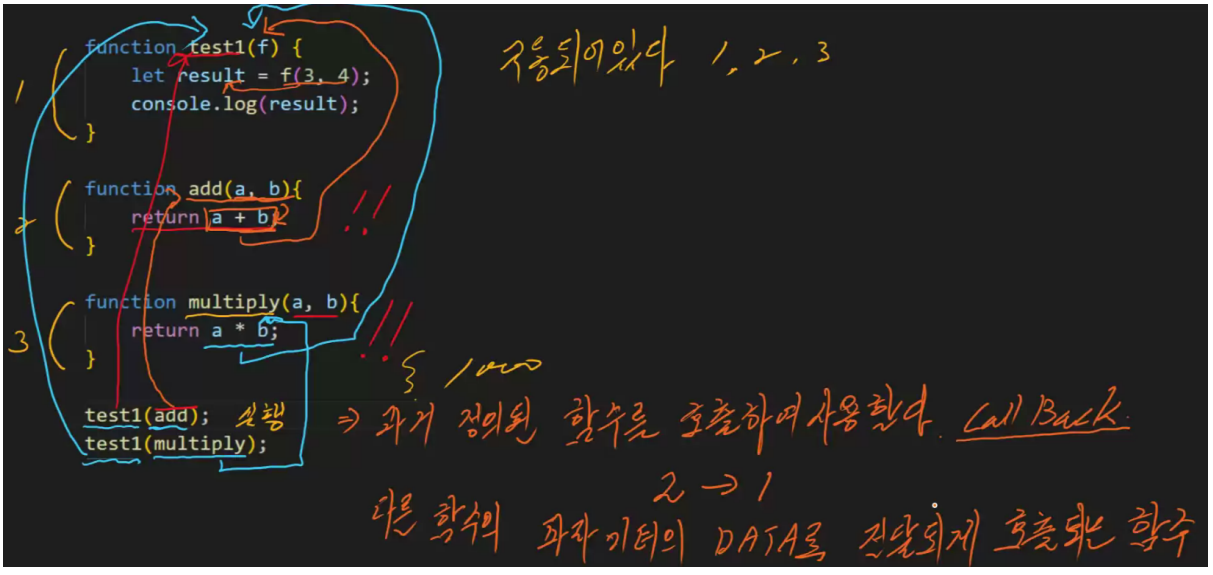
CallBack

- SASP 호출에서 DASD 호출이 가능 하다.
- 처치 순서에 지연 시간을 부여한다.



#CallBack은 함수 자체가 자료형 이기도 하고, 자료 이기도 하다.

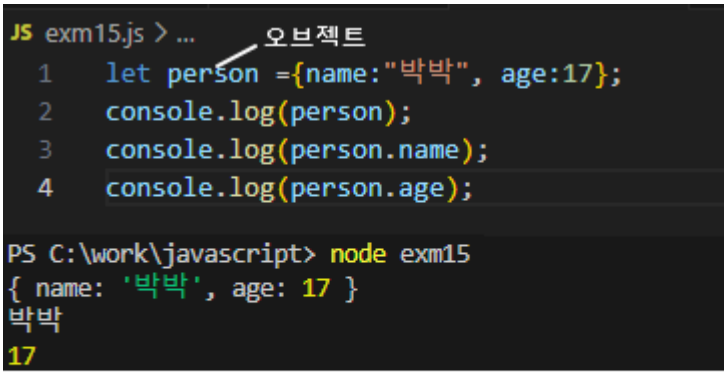
```
function add(a, b){
    return a+b;
}

let a= add(3,4); /* a가 add 함수를 callback함 */

console.log(a)

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

let b = f(3,4);
console.log(b)
PowerShell
7               //자료
function        //자료형
7
```



```
/* CallBack함수 사용 */
//ex1)
function test1(f){
    let result = f(3,4);
    console.log(result)
}

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

function multiplay(a,b){
    return a*b;
}

//ex2)
function test1(f){
    let result = f(3,4);
    console.log(result)
}

let add= (a, b) =>{
    return a + b;
}


let multiply= (a, b) =>{
    return a * b;
}

test1(add)
test1(multiplay)

//ex3)
function test1(f){
    let result = f(3,4);
    console.log(result);
}
test1((a, b) =>{return a + b;});
test1((a, b) =>{return a * b;});

PowerShell
7
12
```

```
//시간
function printTime(msg){
    console.log(msg, new Date()); //new 새로운 정보 입력시 입력
```

```
}                                   //date 내컴퓨터 기준 시간과 날짜를 가져온다

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

2초 후 2023-11-08T05:39:23.267Z
3초 후 2023-11-08T05:39:24.275Z
5초 후 2023-11-08T05:39:26.261Z
```

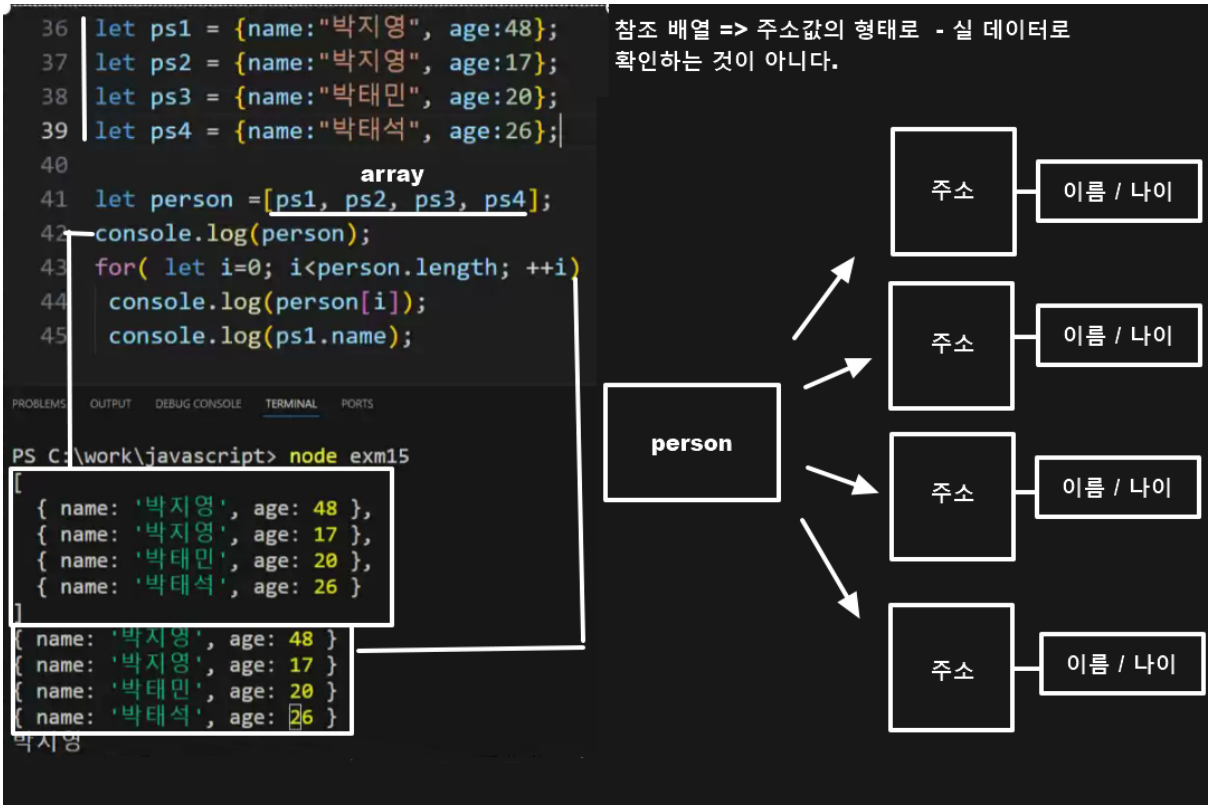


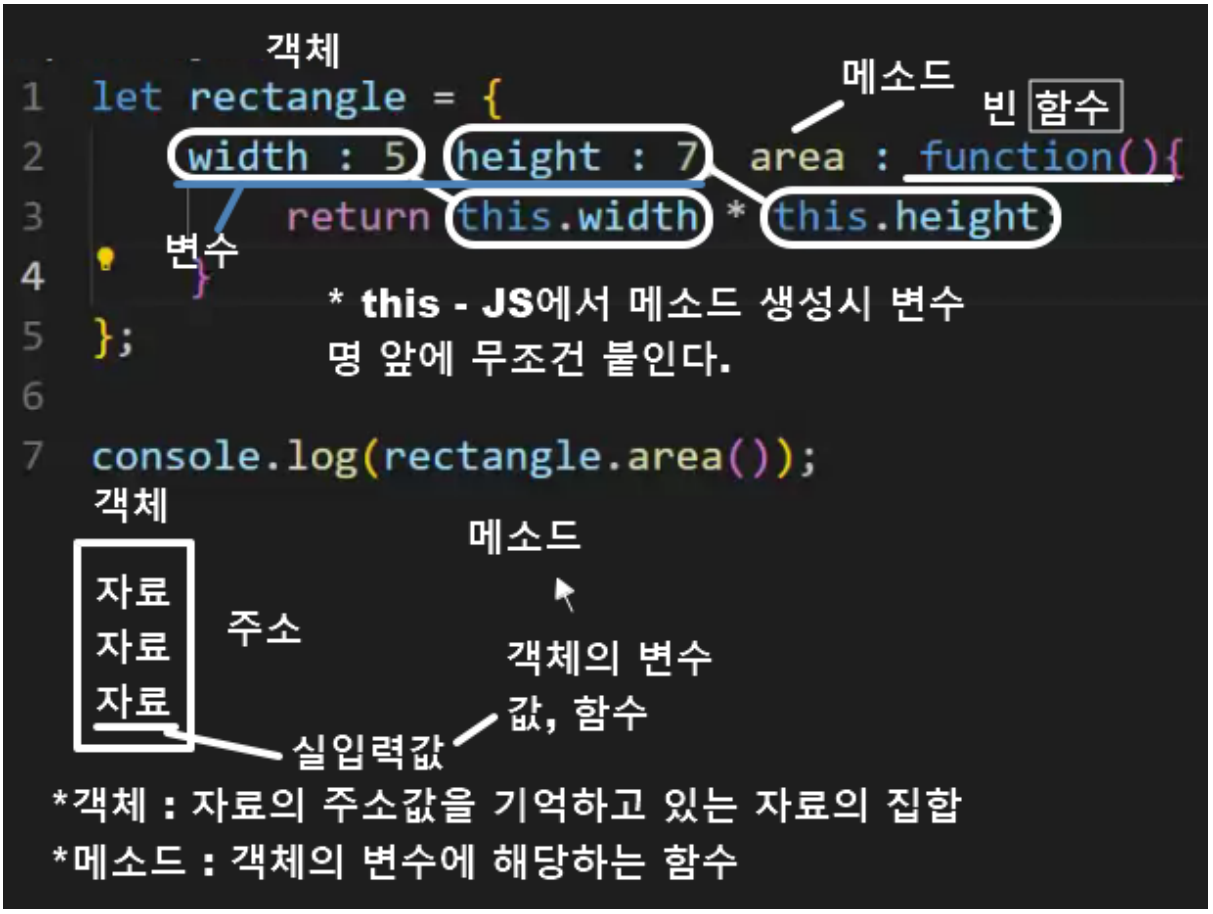
회원 정보등을 입력할 때 활용

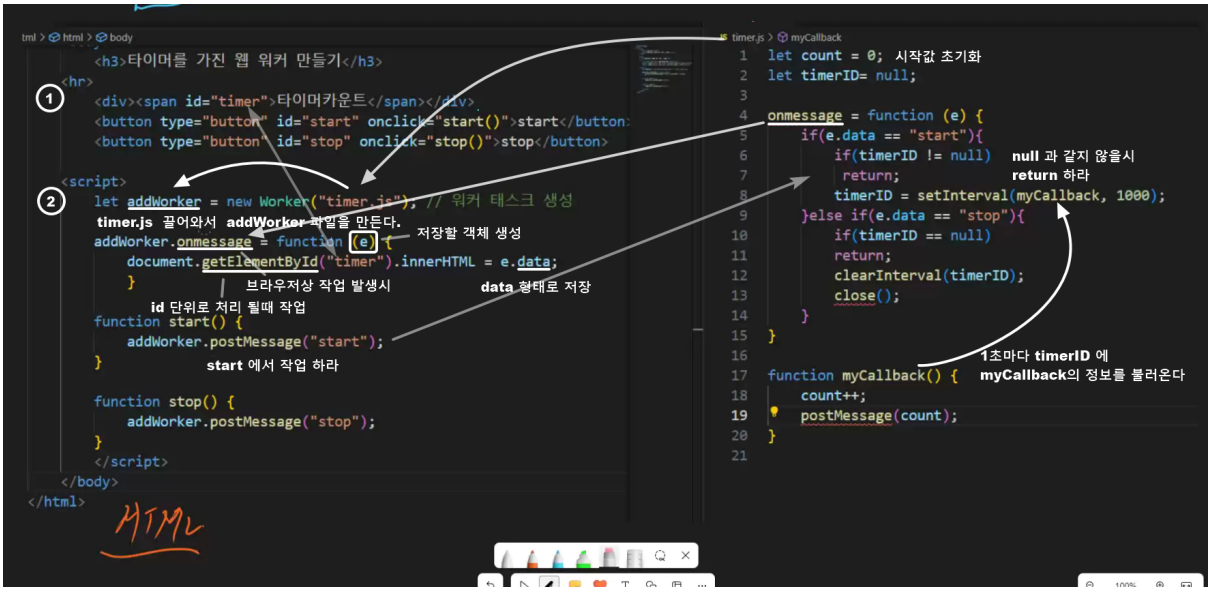
```
//오브젝트 (객체 만들기)
//ex1)
let person ={name:"박박", age:17};
console.log(person);
console.log(person.name);
console.log(person.age);

//ex2)
let person1 ={};  /* 빈 객체 */
person1.name="박박";
person1.age= 4;
console.log(person1);

//ex3)
let person2 ={name:"박박"};
person2.age = 74;
console.log(person2);
```

```
//ex4)
function createPerson(s,i){
    return { name:s, age:i};  //createPerson= CallBack함수
}

let person1 = createPerson("바보",21);
let person2 = createPerson("뚜이씨",18);
let p = person1;

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

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

PowerShell
{ name: '바보', age: 21 }
{ name: '뚜이씨', age: 18 }
{ name: '황마', age: 17 }
false
true


function equals(person1, person2){
    return person1.name == person2.name && person1.age == person2.age;
}
console.log(equals(person1, person2));

false
```

JS에서 callback함수나 객체를 만들 때 이름은 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};
array
let person =[ps1, ps2, ps3, ps4];
console.log(person);
for(let i=0; i<person.length; ++i)
console.log(person[i]);
console.log(ps1.name);
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\work\javascript> node exm15
{ name: '박지영', age: 48 },
{ name: '박지영', age: 17 },
{ name: '박태민', age: 20 },
{ name: '박태석', age: 26 }
{ name: '박지영', age: 48 }
{ name: '박지영', age: 17 }
{ name: '박태민', age: 20 }
{ name: '박태석', age: 26 }
박지영
참조 배열 => 주소값의 형태로 - 실 데이터로
확인하는 것이 아니다.
person
주소
이름 / 나이
주소
이름 / 나이
주소
이름 / 나이
주소
이름 / 나이


객체
메소드
빈 함수
let rectangle = {
width : 5
height : 7
area : function(){
return this.width * this.height;
}
};
변수
* this - JS에서 메소드 생성시 변수
명 앞에 무조건 붙인다.
console.log(rectangle.area());
객체
메소드
자료
자료
자료
주소
객체의 변수
값, 함수
실입력값
*객체 : 자료의 주소값을 기억하고 있는 자료의 집합
*메소드 : 객체의 변수에 해당하는 함수


tml > html > body
<h3>타이머를 가진 웹 워커 만들기</h3>
<hr>
타이머카운트
<button type="button" id="start" onclick="start()">start</button>
<button type="button" id="stop" onclick="stop()">stop</button>
<script>
let addWorker = new Worker("timer.js"); // 워커 테스크 생성
addWorker.onmessage = function (e) {
document.getElementById("timer").innerHTML = e.data;
}
function start() {
addWorker.postMessage("start");
}
function stop() {
addWorker.postMessage("stop");
}
</script>
</body>
</html>
1
2
timer.js 끌어와서 addWorker 파일을 만든다.
저장할 객체 생성
브라우저상 작업 발생시
data 형태로 저장
id 단위로 처리 될때 작업
start 에서 작업 하라
HTML
timer.js > myCallback
let count = 0; 시작값 초기화
let timerID= null;
onmessage = function (e) {
if(e.data == "start"){
if(timerID != null)
return;
timerID = setInterval(myCallback, 1000);
}else if(e.data == "stop"){
if(timerID == null)
return;
clearInterval(timerID);
close();
}
}
function myCallback() {
count++;
postMessage(count);
}
null 과 같지 않을시
return 하라
1초마다 timerID 에
myCallback의 정보를 불러온다