

2023-11-08

함수 → 미리 약속된 계산식

sqrt(함수명) - 제곱근 표현할 때 사용

```
/* 함수지정 */  
function hello(name){  
  console.log("hello"+name);  
}  
  
function add(a,b){  
  return a+b;  
}
```

function → 기능정의(함수지정)

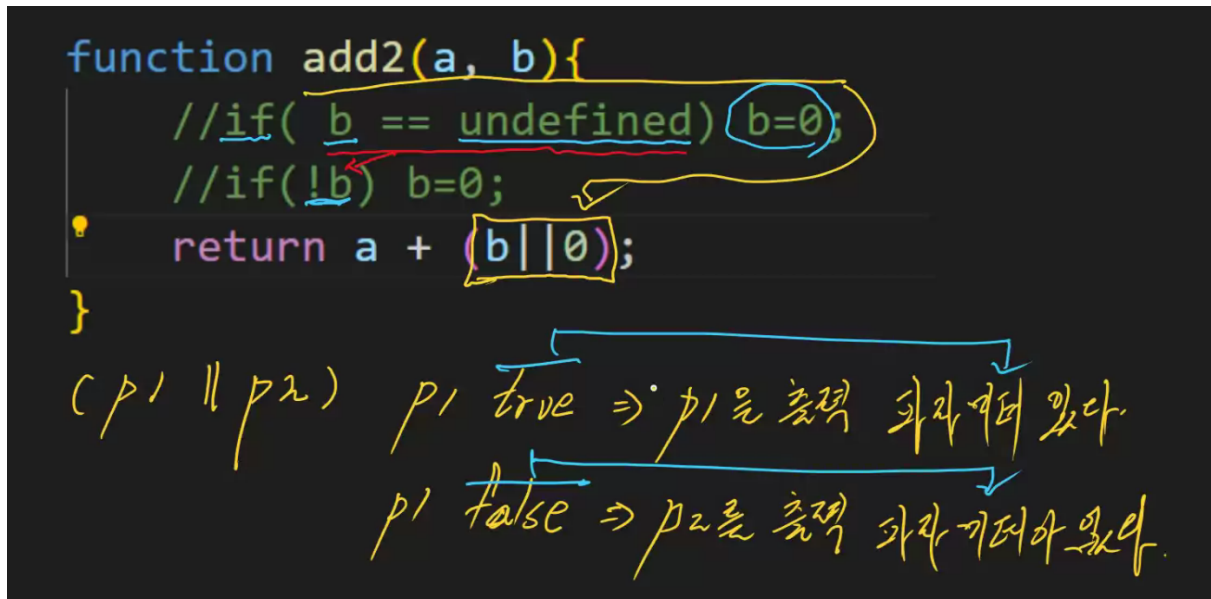
hello → 함수명

(name) → 파라미터(대입자/ 대신 입력하는 글자) 내가 입력받을 변수의 내용

console.log의 name은 파라미터의 name을 가져옴

```
function hello(name){  
  console.log("hello"+name);  
}  
function add(a,b){  
  return a+b;  
}  
  
function add2(a,b){  
  // if (b == undefined) b=0;  
  if (!b) b=0;  
  return a+b;  
}  
  
hello(" 박태석");  
console.log(add(3,4));  
console.log(add(3));  
  
console.log(add2(3,4));  
console.log(add2(3));
```

자바스크립트는 자료형 지정을 하지 않으므로 자료형 지정이 가능한 타입스크립트가 추후에 나오게된다.



```
function power (...a){  
  let result = 0;  
  for (let i=0; i<a.length; i++)  
    result = result + a[i];  
  return result;  
}  
  
console.log(power(1,2,3,4));
```

(...a) → 가변 파라미터, 개수를 정해 놓지 않음

```
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);
```

callback 함수 → 언제든지 호출 할 수 있다.

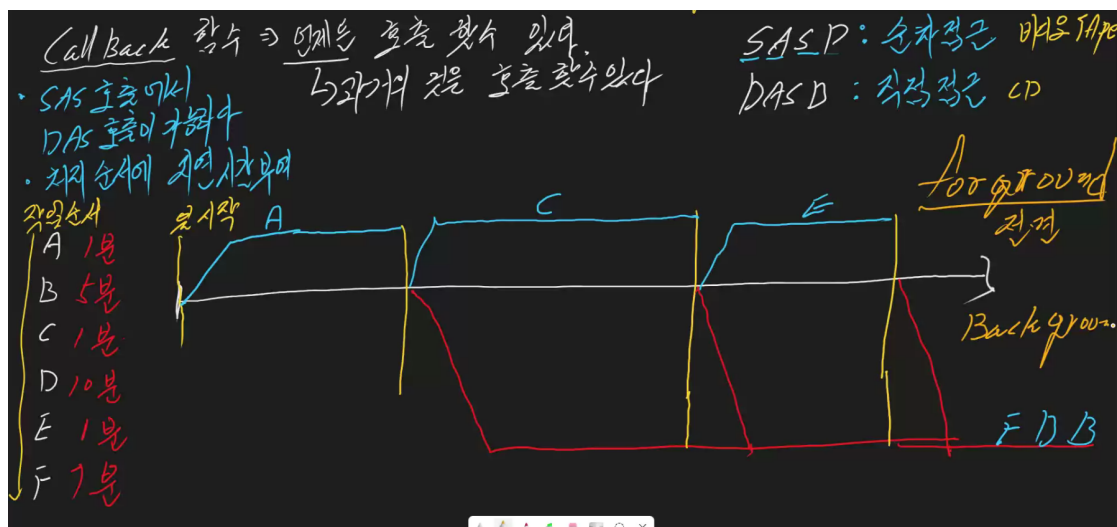
언제든 → 과거의 것을 호출 할 수 있다.

SASP: 순차접근 (비디오 tape)

DASD: 직접접근 (cd)

sas호출에서 das호출이 가능하다.

처리 순서에 지연시간 부여



화면출력에 필요한 foreground 를 빨리 처리하고 background를 천천히 처리해도 될 때 callback

callback은 함수 자체가 자료형이기도 하고 자료이기도 하다.

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

let f = function(a, b){
  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);

사용되어있다 1, 2, 3

⇒ 과거 정의된 함수를 호출하여 사용한다. Call Back

Call Back 함수

다른 함수의 파라미터의 DATA를 전달되게 호출되는 함수

```
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);
//////////////////////결과는 같다.
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(multiply);

function test2(f){
  let result = f(5, 7);
  console.log(result);
}
```

```

}

test2((a, b) => {return a+b;})

test2((a, b) => {return a*b;})

```

```

function printtime(msg){
    console.log(msg, new Date());
}
setTimeout(printtime, 1000, "1초 후");
setTimeout(printtime, 2000, "2초 후");
setTimeout(printtime, 3000, "3초 후");

```

```

let person={name:"박태석", age:26};
console.log(person);
console.log(person.name);
console.log(person.age);

let person1 = {};

person1.name="박태석";
person1.age=26;
console.log(person1);

let person2={name:"박태석"};
person2.age=26;
console.log(person2);

function createPerson(s,i){
    return{name:s, age:i};
}
let person1 = createPerson("박태석",26);
let person2 = createPerson("박태석2",26);
let p=person1;

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

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

```

콜백함수나 객체를 만들때 자바에선 동일한 이름을 가진 클래스나 오브젝트를 만들수있다.

자바스크립트는 동일한 이름이 있으면 못찾는다. (무조건 유일해야 한다.)

```

1 let rectangle = {
2   width : 5, height : 7, area: function(){
3     return this.width * this.height;
4   }
5 };
6 console.log(rectangle.area());

```

비 함수

자바스크립트 메소드 생성시
변수명 앞에 this는 무조건 붙인다. 객체의 변수에 해당하는 함수
∴ 메소드라 부른다.

객체
자료
자료
자료
주소
메소드
객체의 변수
주소
함수
자료의 주소값을 기억하고 있는
자료의 집합
∴ 객체라 부른다.

```

let rectangle = {
  width :5, height :7, area: function(){
    return this.width * this.height;
  } // -> 파라미터와 함수명을 밖에 두고 함수의 기능정리를 이렇게 한다.
};

console.log(rectangle.area());

```

자바스크립트에서 메소드 생성 시 변수명 앞에 this는 무조건 붙인다.

HTML

```

<h3>타이머를 가진 웹 워커 만들기</h3>
<hr>
<div><span id="timer">타이머카운트</span></div>
<button type="button" id="start" onclick="start()">start</button>
<button type="button" id="stop" onclick="stop()">stop</button>
</div>
<script>
let addWorker = new Worker("timer.js"); // 워커 생성
addWorker.onmessage = function (e) {
  document.getElementById("timer").innerHTML = e.data;
};
function start() {
  addWorker.postMessage("start");
}
function stop() {
  addWorker.postMessage("stop");
}
</script>
</body>
</html>

```

timer.js

```

1 let count = 0;
2 let timerID = null;
3
4 onmessage = function (e) {
5   if(e.data == "start"){
6     if(timerID != null)
7       return;
8     timerID = setInterval(myCallback, 1000);
9   }else if(e.data == "stop"){
10    if(timerID == null)
11      return;
12    clearInterval(timerID);
13    close();
14  }
15 }
16
17 function myCallback() {
18   count++;
19   postMessage(count);
20 }
21

```

setinterval → 간격

```
addWorker.onmessage = function (e) {
```

on.message → html 화면상에서 새로운 일이 발생이 되면 addworker에 새로운 메세지 불러온다. (브라우저에서 작동되면)

function (e) 기능 함수 만듦