

디지털 컨버전스 기반 UI/UX Front 전문 개발자 양성 과정(BITCamp)

강사 : 이상훈

수강생: 오진욱

- JavaScript -

Equality

Loose
equality

`==`

Type conversion
데이터 타입 상관 X

Ex) '5' == 5

True

Strict
equality

`====`

No Type conversion
데이터 타입 까지 상관

Ex) '5' === 5

False

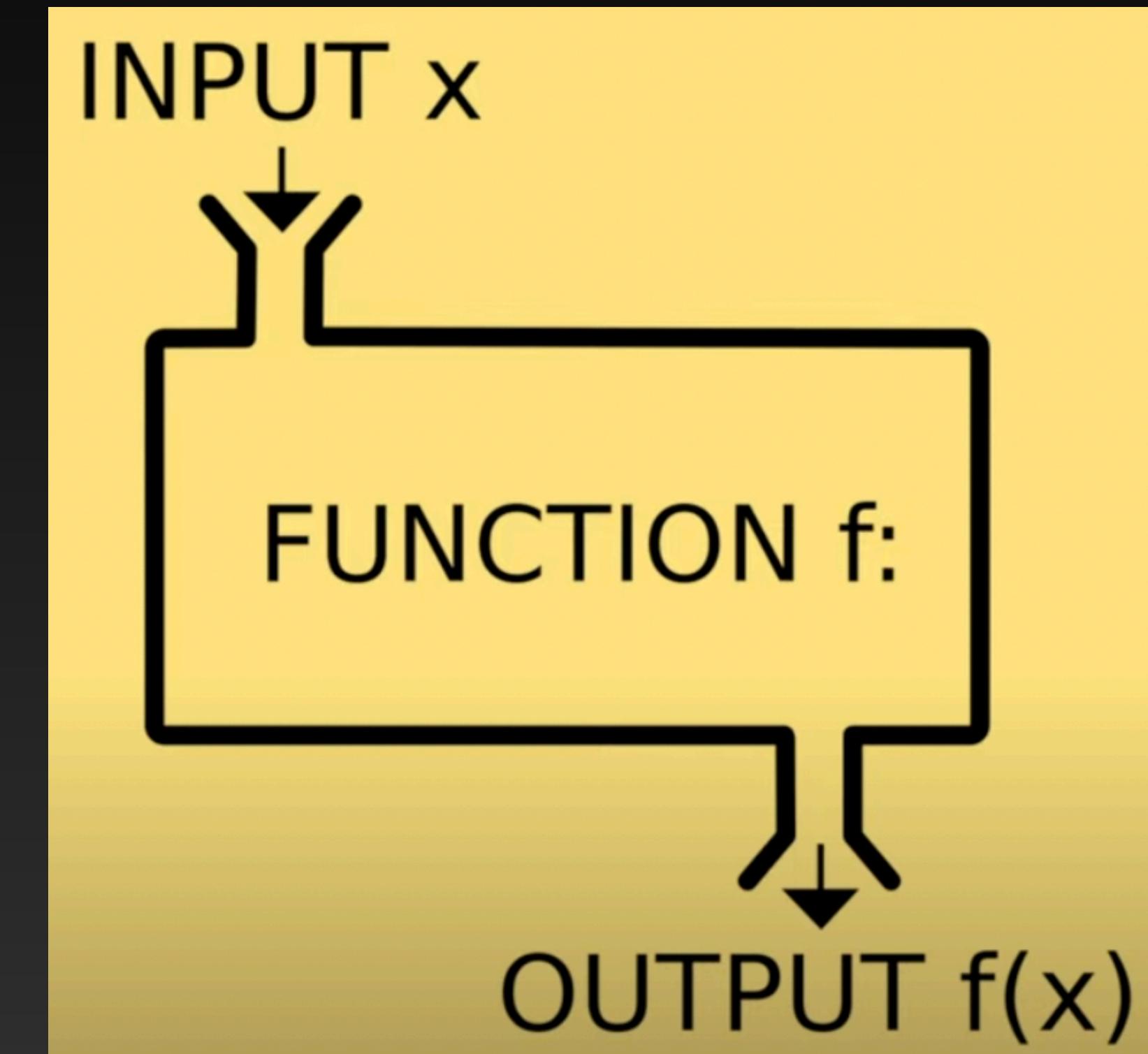
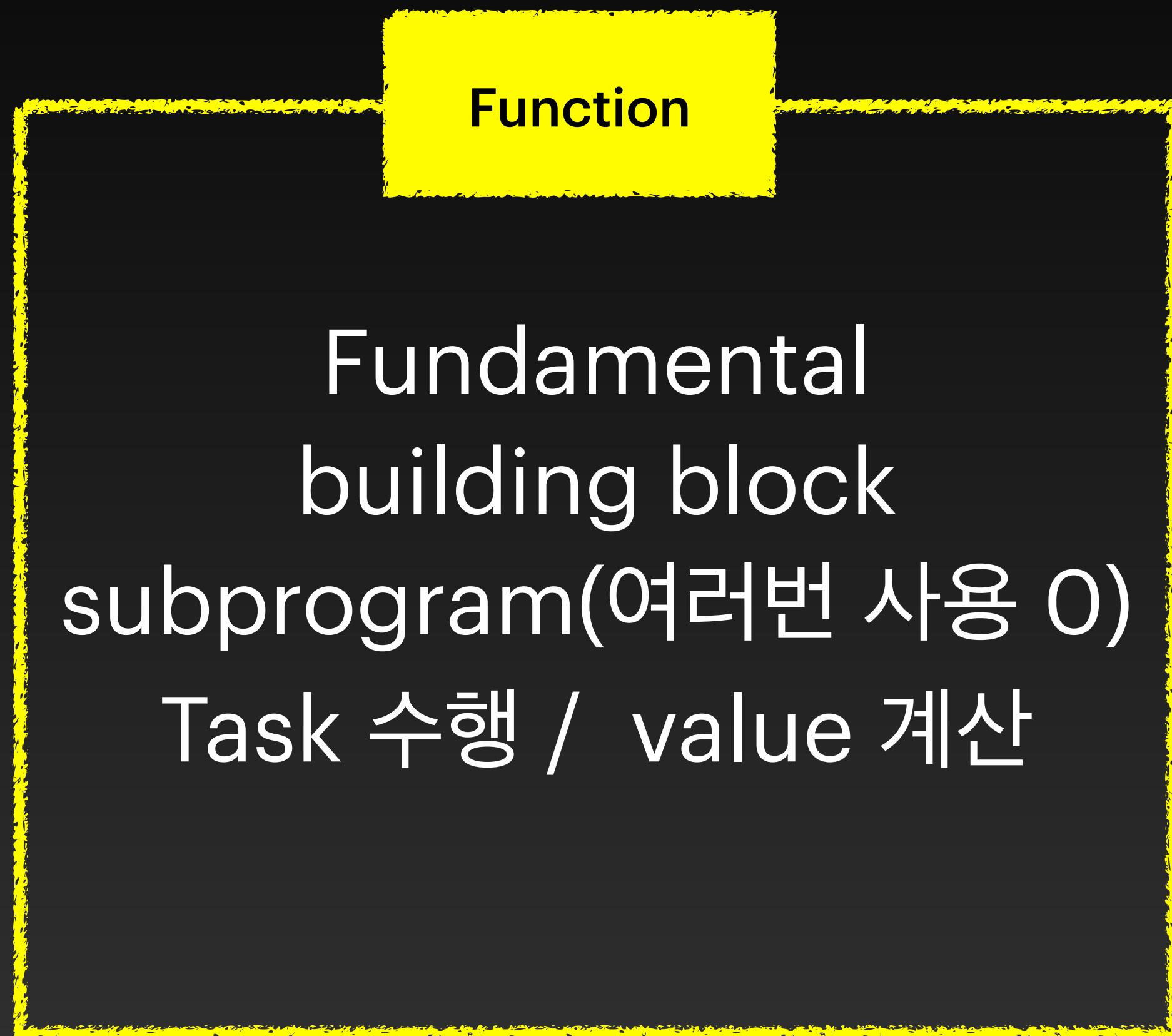
Equality

Object Equality
By reference

Object는 메모리에 올라갈때
Reference 형태로 저장
똑같은 object를 저장해도
메모리에서는 다르다고 인식

```
let equality1 = { name: 'jinwook' };
let equality2 = { name: 'jinwook' };
let equality3 = equality1
console.log(equality1 == equality2) // false
console.log(equality1 === equality2) // false
console.log(equality1 === equality3) // true
```

Function



Function

Function

Function name(입력값){

 수행 내용}

*한가지 일만 수행

*동사형으로 naming하기

function은 object

Object

변수로 할당 가능
파라미터로 전달
리턴이 가능

Function

Parameter

Primitive
->value로 전달
Object
->reference로 전달

Default Parameter

Parameter에 값을
지정 할 수 있는 것을
말함

```
function showMessage(message, from = 'unknown') {  
  console.log(`#${message} by ${from}`);  
}
```

Rest Parameter

... 형태 흘뿌리기
배열 형태로 출력

Function Expression

Function Declaration

함수는 define 되기 전에
호출이 가능하다(Hoisted)

```
sum(2, 3);  
// 6. Return a value  
function sum(a, b) {  
    return a + b;  
}
```

Function Expression

할당된 다음부터 호출이 가능
Anonymous function
Named function

```
const print = function () {  
    // anonymous function  
    console.log('print');  
};  
print();  
const printAgain = print;  
printAgain();  
const sumAgain = sum;  
console.log(sumAgain(1, 3));
```

Function Expression

Call back

전달된 함수들 중 원하는 함수
를 불러오는 것을 말함

```
function bitcamp(classNum, printYes, printNo){  
    if(classNum === 502){  
        printYes();  
    }else{  
        printNo();  
    }  
  
    const printYes = function () {  
        console.log('yes!');  
    };  
  
    const printNo = function print() {  
        console.log('no!');  
    };  
  
    bitcamp(502,printYes,printNo) // yes  
    bitcamp(501,printYes,printNo) // no
```

Class & Object

Class

Fields, Methods 의 뮤음

Template

Declare once

No data in

Encapsulation (캡슐화) 가능

상속과 다형성 가능

Syntactical sugar (JS class)

Object

Instance of a class

Created many times

Data in

소프트웨어에서 구현해야 할 대상

Class & Object

Class Checking

instanceof
클래스안의 인스턴스
인지 알 수 있음
상속 받은 클래스 이면
상속한 클래스 또한
true로 나온다

```
console.log(rectangle instanceof Rectangle);
console.log(triangle instanceof Rectangle);
console.log(triangle instanceof Triangle);
console.log(triangle instanceof Shape);
console.log(triangle instanceof Object);
```

Array.methods

indexOf()

배열에서 지정된 요소를 찾을 수 있는 첫번째 인덱스를 반환하고
존재하지 않으면 -1를 반환

.length

배열의 길이를 반환
배열의 최대 인덱스보다 항상 크다

Array.methods

.shift

배열에서 첫번째 요소를 제거하고
그 요소를 반환 한다

```
var myFish = ['angel', 'clown', 'mandarin', 'surgeon'];

console.log('myFish before: ' + myFish);
// "제거전 myFish 배열: angel,clown,mandarin,surgeon"
```

.pop

배열에서 마지막 요소를 제거하고
그 요소를 반환 한다

```
const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];

console.log(plants.pop());
// expected output: "tomato"
```

Array.methods

.toString()

지정된 배열 및 그 요소를 나타내는 문자열을 반환

```
const array1 = [1, 2, 'a', '1a'];
console.log(array1.toString());
// expected output: "1,2,a,1a"
```

.join()

배열의 모든 요소를 연결해 하나의 문자열로 만듬

```
const elements = ['Fire', 'Air', 'Water'];

console.log(elements.join());
// expected output: "Fire,Air,Water"

console.log(elements.join(''));
// expected output: "FireAirWater"

console.log(elements.join('-'));
// expected output: "Fire-Air-Water"
```

Array.methods

.slice()

배열의 시작부터 끝까지에 대한 복사본을 새로운 배열 객체로 반환

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

console.log(animals.slice(2));
// expected output: Array ["camel", "duck", "elephant"]

console.log(animals.slice(2, 4));
// expected output: Array ["camel", "duck"]

console.log(animals.slice(1, 5));
// expected output: Array ["bison", "camel", "duck", "elephant"]
```

.splice()

배열의 기존 요소를 삭제 또는 교체하거나 새로운 요소를 추가하여 배열의 내용을 변경

```
const months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');
// inserts at index 1
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');
// replaces 1 element at index 4
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "May"]
```

Array.methods

.push()

배열의 끝에 하나 이상의 요소를
추가하고, 배열의 새로운 길이를
반환

```
var sports = ['축구', '야구'];
var total = sports.push('미식축구', '수영');

console.log(sports); // ['축구', '야구', '미식축구', '수영']
console.log(total); // 4
```

.value()

배열의 각 인덱스에 대한 값을 가
지는 새로운 ArrayIterator 객체
를 반환

```
const array1 = ['a', 'b', 'c'];
const iterator = array1.values();

for (const value of iterator) {
  console.log(value);
}

// expected output: "a"
// expected output: "b"
// expected output: "c"
```