

K-Digital Training 웹 풀스택 과정

JavaScript (2)

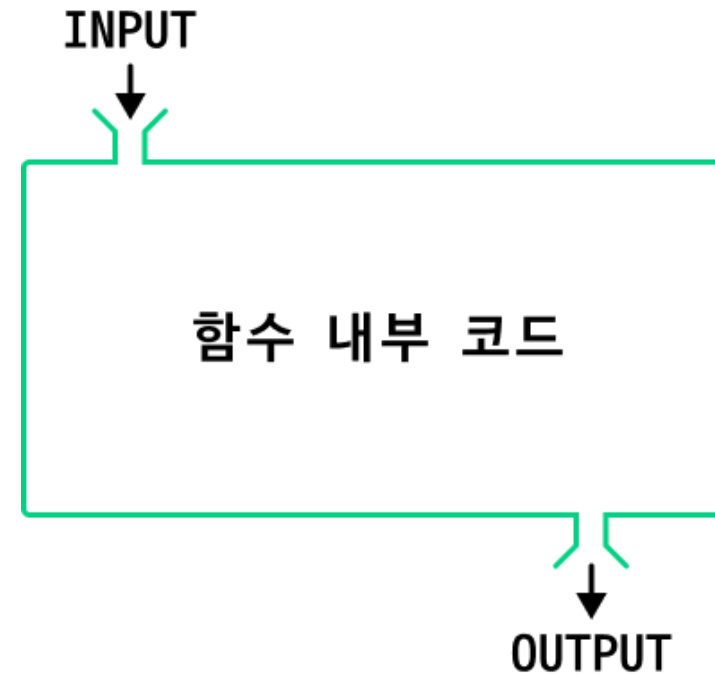
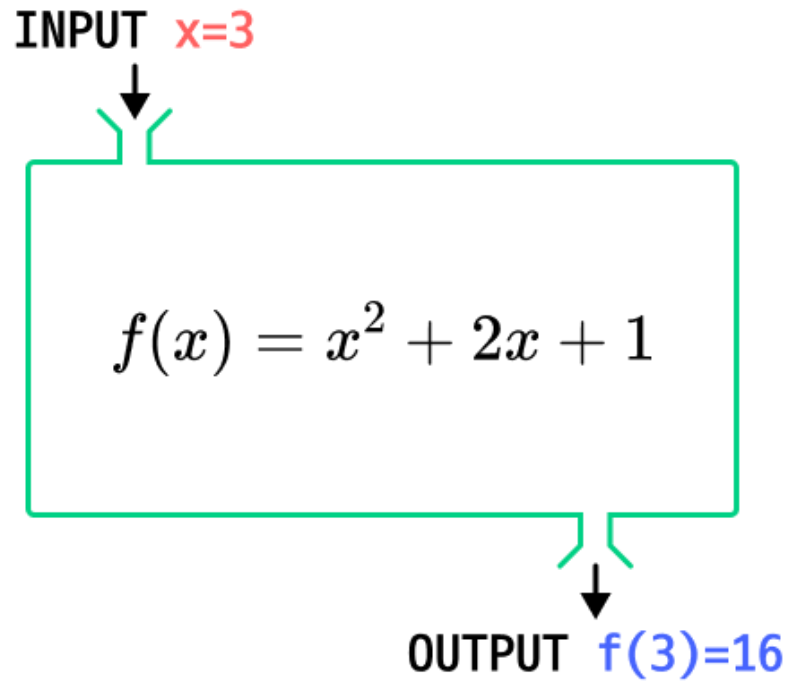
함수

Function!

함수

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)
function

함수



함수 선언문 vs 함수 표현식

```
function sayHello(){  
    console.log('Hello');  
}
```

함수 선언문

함수 선언문 vs 함수 표현식

```
let sayHello = function(){  
  console.log('Hello');  
}
```

함수 표현식

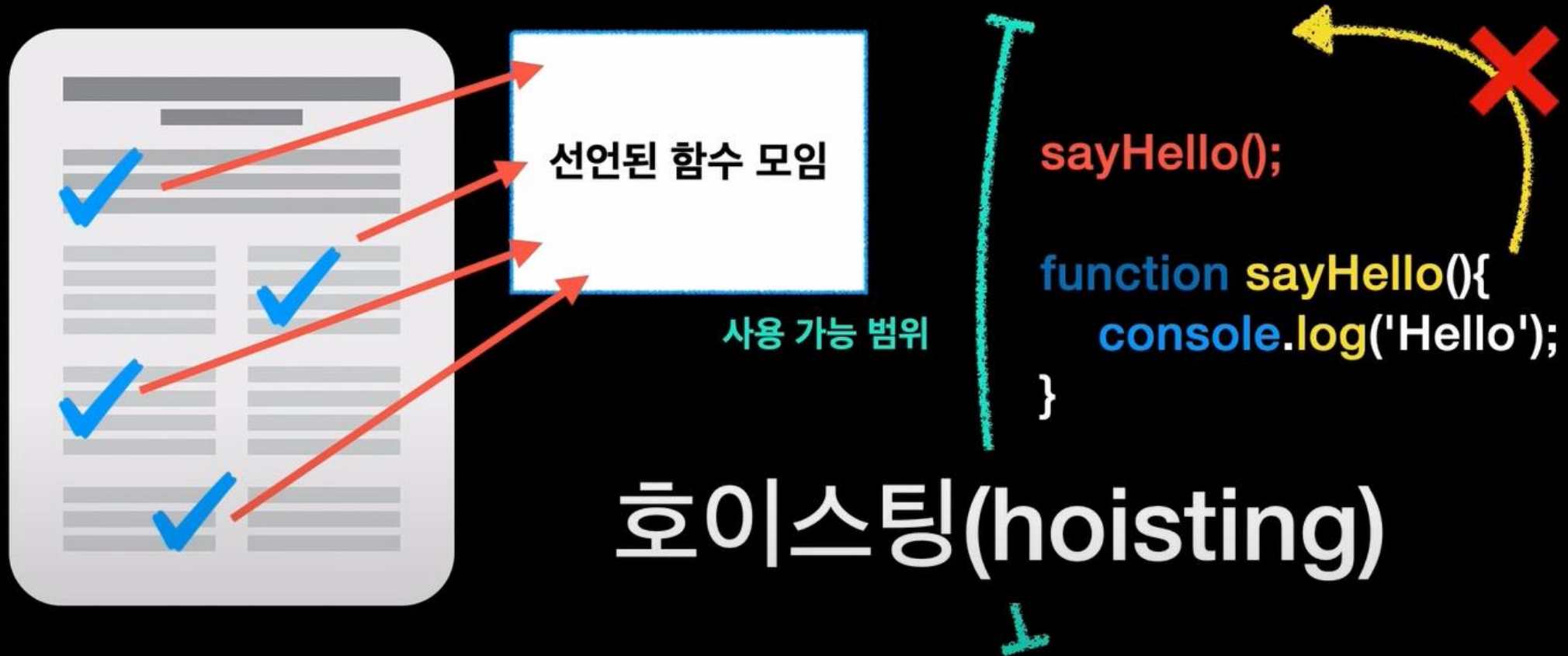
함수 선언문 : 어디서든 호출 가능

sayHello();



```
function sayHello(){  
  console.log('Hello');  
}
```

함수 선언문 : 어디서든 호출 가능



함수 표현식 : 코드에 도달하면 생성

1 ...

2 ...

3 `let sayHello = function(){
 console.log('Hello');
}` 생성
 사용가능

4 `sayHello();`

화살표 함수(arrow function)

```
let add = function(num1, num2){  
  return num1 + num2;  
}
```

화살표 함수(arrow function)

```
let add = (num1, num2) => {  
  return num1 + num2;  
}
```

화살표 함수(arrow function)

```
let showError = () => {  
    alert('error!');  
}
```

화살표 함수로 만들어 보기

```
// 함수 선언문
function sayHello(name) {
  console.log(`Hello, ${name}`);
}

// 함수 표현식
let sayHello = function (name) {
  console.log(`Hello, ${name}`);
}

// 화살표 함수
let sayHello = (name) => {
  console.log(`Hello, ${name}`);
}
```

함수 형태

// 1. 매개변수 X, 반환값 X

```
function sayHello() {  
  console.log(`Hello`);  
}
```

// 2. 매개변수 X, 반환값 O

```
let sayHello = () => {  
  return `Hello`  
}
```

// 3. 매개변수 O, 반환값 X

```
let sayHello = function (name) {  
  console.log(`Hello, ${name}`);  
}
```

// 4. 매개변수 O, 반환값 O

```
let sayHello = function (name) {  
  return `Hello, ${name}`  
}
```

Onclick!

onclick

- 각각의 HTML 요소에 속성 값으로 JS 함수를 연결

```
<body>  
  <div class="box"  
onclick="test();">click</div>  
</body>
```

```
function test() {  
  alert("TEST");  
}
```

이 페이지 내용:

TEST

확인

조건문

Javascript 조건문

특정 조건 만족 시 (조건이 참인 경우) 실행하는 명령의 집합

특정한 조건 속에서 작업을 수행하고 싶을 때 사용

if

switch

조건문

if

IF / ELSE

```
if ( 조건1 ) {  
    // 조건1이 참이라면 실행  
} else {  
    // 조건1이 거짓이라면 실행  
}
```

IF / ELSE IF / ELSE

```
if ( 조건1 ) {  
    // 조건1이 참이라면 실행  
} else if ( 조건2 ) {  
    // 조건2가 참이라면 실행  
} else {  
    // 조건 1과 2가 모두 참이 아닐 때 실행  
}
```

IF 중첩

```
if ( 조건1 ) {  
    if ( 조건2 ) {  
        //실행  
    } else {  
        //실행2  
    }  
}
```

```
let isShow = true;
let checked = false;

if (isShow) {
  console.log('Show!'); // Show!
}

if (checked) {
  console.log('Checked!');
}
```

```
let isShow = true;

if (isShow) {
    console.log('Show!');
} else {
    console.log('Hide?');
}
```


조건문

Switch

```
switch ( 변수 ) {  
    case 값1:  
        // 변수와 값1이 일치하면 실행  
        break;  
    case 값2:  
        // 변수와 값2가 일치하면 실행  
        break;  
    default:  
        //일치하는 값이 없을 때 실행  
        break;  
}
```

3항 연산자

IF 문을 간단하게 표현하는 방법

- 조건식 ? 조건이 참인 경우 : 조건이 거짓인 경우;
- 한 줄로 간단히 표현 가능!

```
// 3항 연산자
let name = "lily";

if (name === "lily") {
  console.log("맞았어요 😊");
} else {
  console.log("틀렸어요 😞");
}

name !== "lily" ? console.log("맞았어요 😊") : console.log("틀렸어요 😞");
```

반복문

Javascript 반복문

똑같은 명령을 일정 횟수만큼 반복해 수행하도록 하는 실행문

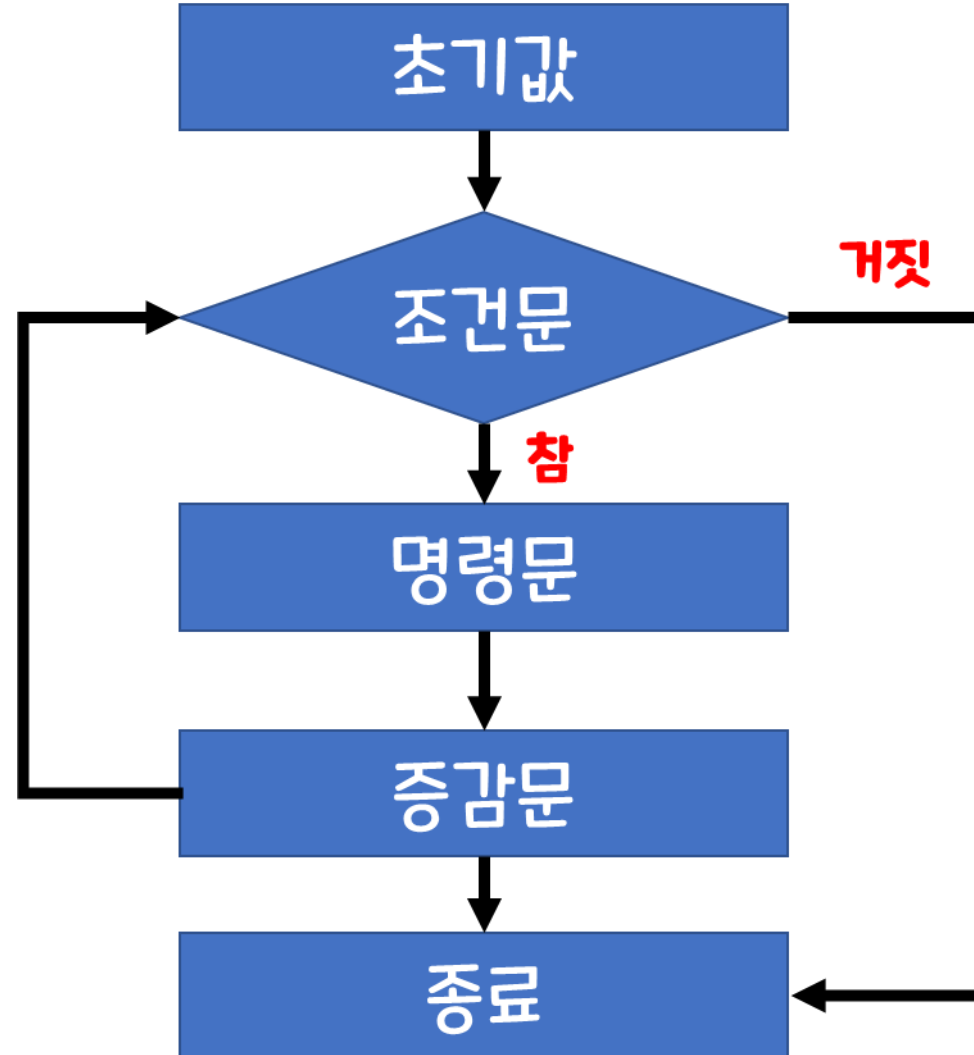
for

while

for / of

do / while





For 문



```
// for 문
for(let index = 0; index < 10; index++) {
  console.log("인사를 ", i+1, "번째 드립니다! 😊");
}
```

```
인사를 1 번째 드립니다! 😊
인사를 2 번째 드립니다! 😊
인사를 3 번째 드립니다! 😊
인사를 4 번째 드립니다! 😊
인사를 5 번째 드립니다! 😊
인사를 6 번째 드립니다! 😊
인사를 7 번째 드립니다! 😊
인사를 8 번째 드립니다! 😊
인사를 9 번째 드립니다! 😊
인사를 10 번째 드립니다! 😊
```

반복문

while

while 문

- while(조건문) {

실행할 코드(명령문)

}

- For 문과는 달리 조건을 변경하는 구문이 기본적으로 포함이 되어 있지 않기 때문에 무한 루프 가능
- 주의하여 사용 필요

```
// while 문

// 1번 타입, 조건문을 사용
let index = 0;

while (index < 10) {
  console.log("인사를 ", index + 1, "번째 드립니다! 😊");
  index++;
}

// 2번 타입, 조건문을 사용하지 않고 if 문 + break 사용
let index2 = 0;

while (true) {
  console.log("절을 ", index2 + 1, "번째 드립니다! 😊");
  index2++;
  if (index2 == 10) {
    break;
  }
}
```

인사를	1	번째	드립니다!	😊
인사를	2	번째	드립니다!	😊
인사를	3	번째	드립니다!	😊
인사를	4	번째	드립니다!	😊
인사를	5	번째	드립니다!	😊
인사를	6	번째	드립니다!	😊
인사를	7	번째	드립니다!	😊
인사를	8	번째	드립니다!	😊
인사를	9	번째	드립니다!	😊
인사를	10	번째	드립니다!	😊
절을	1	번째	드립니다!	😊
절을	2	번째	드립니다!	😊
절을	3	번째	드립니다!	😊
절을	4	번째	드립니다!	😊
절을	5	번째	드립니다!	😊
절을	6	번째	드립니다!	😊
절을	7	번째	드립니다!	😊
절을	8	번째	드립니다!	😊
절을	9	번째	드립니다!	😊
절을	10	번째	드립니다!	😊

while

```
let i = 0;
```

```
while (i < 10) {
```

```
// 코드
```

```
}
```

while

```
let i = 0;
```

```
while (i < 10) {  
    // 코드  
    i++;  
}
```

```
// 구구단 while 버전
let i = 2, j = 1;

while(i < 10) {
    while(j<10) {
        console.log(i, "x", j, "=", i*j);
        j++;
    }
    i++;
    j = 1;
}
```

반복문

제어

break, continue

break

: 멈추고 빠져나옴

continue

: 멈추고 다음 반복으로 진행

break

- 반복문을 멈추고 밖으로 빠져 나감

```
// break  
  
for(let i = 0; i < 100; i++) {  
    if(i==10) {  
        console.log("멈춰!");  
        break;  
    }  
    console.log(i);  
}
```

0
1
2
3
4
5
6
7
8
9
멈춰!


continue

- 반복문을 한 번만 멈추고 다음으로 진행

```
// continue
let sum = 0;

for(let i = 0; i < 100; i++) {
  if(i%2 == 0) {
    continue;
  }
  sum += i;
}

console.log(sum);
```



2500

배열에서의 For

배열, 기본 for 문 사용하기

```
let numbers = [1, 2, 3, 4, 5, 6];  
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];  
  
for (let i = 0; i < numbers.length; i++) {  
  console.log(numbers[i]);  
}  
  
for (let i = 0; i < fruits.length; i++) {  
  console.log(fruits[i]);  
}
```

배열, for of 반복문

```
let numbers = [1, 2, 3, 4, 5, 6];  
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];  
  
let numbersLength = numbers.length;  
let fruitsLength = fruits.length;  
  
for (let number of numbers) {  
  console.log(number);  
}  
  
for (let fruit of fruits) {  
  console.log(fruit);  
}
```

배열, [].forEach

```
let numbers = [1, 2, 3, 4, 5, 6];
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];

numbers.forEach(function (number, index, array) {
  console.log(number, index, array);
});

numbers.forEach((number, index, array) => {
  console.log(number, index, array);
});

fruits.forEach(function (fruit, i, arr) {
  console.log(fruit, i, arr);
});

fruits.forEach((fruit, i, arr) => {
  console.log(fruit, i, arr);
});
```

배열의 합

```
let numbers = [1, 2, 3, 4, 5, 6];
var sum1 = 0;
var sum2 = 0;
var sum3 = 0;

for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
  sum1 = sum1 + numbers[i];
}

for (let num of numbers) {
  sum2 = sum2 + num;
}

numbers.forEach((num) => {
  sum3 = sum3 + num;
});

console.log(sum1, sum2, sum3);
```