

02

함수 : 하나의 업무 처리를 위한 기능별 코드의 묶음

02-5. Function

- 예시) 회원가입
- 유저 입력 획득
 - 유효성 검증
 - id 중복체크
 - 아이디, 패스워드를 최종 저장
 - ...

JavaScript Basic

하나의 기능만 바꾸고 싶으면 해당 함수만 바꾸면 된다.

어떤 업무를 하기 위해 줄글로 작성하는 것보다,
함수의 모임으로 작성하는 것이 프로그래밍의 기본.

함수

함수란

- 함수는 하나의 관련된 업무를 실행하기 위한 코드들을 묶기 위해서 선언되는 프로그램의 구성요소입니다.
- 각 업무가 각각의 함수로 구분되어 작성됨으로 코드를 작성하기도 편하고 나중에 필요한 부분의 코드를 식별해서 분석하거나 실행시키기에도 편해 집니다.



함수와 메서드

- 함수를 메서드라 부르기도 합니다.
- 혼용되어 사용되기는 하지만 두 용어에는 엄밀히 이야기하면 차이가 있습니다.
- 함수는 영어 단어로 function 입니다. 즉 기능이라는 뜻의 단어입니다.
- 즉 함수내에 작성된 기능에 초점을 맞춘 단어입니다.

- 그런데 클래스내에 함수가 선언될 수도 있는데 이 클래스에 선언된 함수를 메서드라고 부르기도 합니다.
- 메서드는 영어 단어로 method, 즉 수단이라는 뜻입니다.
- 즉 함수이지만 클래스내에 선언되어 있어 그 클래스를 이용하는 수단으로 본다는 측면의 용어입니다.

- 정리하자면 모두 동일한 규칙으로 작성된 함수인데 “클래스 내에 선언된 함수를 메서드라 부르기도 한다” 정도로 정리해 주면 좋을 듯 합니다.

함수

선언과 이용

- 함수를 선언하기 위해서는 function 이라는 예약어를 이용합니다.
- 이름 뒤에 () 가 추가되어야 하며 함수의 Body 부분을 { } 로 묶어 주어야 합니다.
- 선언된 함수는 어디선가 그 함수를 호출해야 실행됩니다. 함수 호출은 함수이름() 형태입니다.

함수 선언 부분

```
function myFun ( ) {  
    함수 Body  
}
```

함수 선언 {
function myFun(){
 console.log('hello world')
}
}

함수 호출 — myFun()

함수

매개변수와 반환 값

- 함수는 인수와 반환 값을 가질 수 있습니다.
- 인수(argument)는 함수를 호출하는 곳에서 그 함수에 전달되는 값이며 반환 값은 함수내용이 실행되고 함수를 호출한 곳에 전달되는 값입니다.
- 함수 입장에서 인수는 Input 이며 반환 값은 output 입니다.





Parameter vs Argument

- 함수에서 Parameter 와 Argument 가 혼용되어서 사용되기는 하지만 정확한 용어 정리를 해보면 Parameter 는 함수를 선언하는 입장에서 그 함수를 호출하는 곳에서 전달하는 값을 받기 위해 선언되는 변수를 의미합니다.
- 한국어로 흔히 매개변수라고 합니다.
- Argument 는 함수를 이용하는 곳에서 그 함수를 호출하면서 전달하는 값을 의미합니다.
- 그럼으로 "함수를 선언하면서 Parameter 를 선언하고 외부에서 함수를 호출하면서 Parameter 에 Argument 를 전달해서 실행시킵니다." 라는 표현으로 용어를 사용하면 좋습니다.

함수

매개변수와 반환 값

- 인수를 받기 위해서는 함수에 매개변수가 먼저 선언되어 있어야 합니다.
- 매개변수 선언은 함수 선언위치의 () 에 선언합니다.
- 어떤 함수를 선언하면서 () 를 생략할 수는 없습니다.
- () 부분을 비워둘 수는 있는데 이렇게 되면 이 함수는 외부로부터 인수를 전달 받지 않는 함수가 됩니다.
- 만약 함수를 호출하는 외부로부터 인수를 전달받고자 한다면 () 부분에 그 인수를 받기 위한 변수를 선언해 주어야 합니다.
- 이를 매개변수라고 합니다. 콤마(,) 를 이용해 여러개의 매개변수도 선언이 가능합니다.

```
function 함수명 (매개변수1, 매개변수2, ... ) {  
    ...  
    return 반환값  
}
```

함수

매개변수와 반환 값

- 함수가 실행되고 그 함수를 호출한 곳에 데이터를 반환하고 싶다면 return 예약어를 이용합니다.

```
function myFun(arg1, arg2){  
  ... ..  
  return 10  
}  
  
let result = myFun(10, 20)
```

The diagram illustrates the flow of data from a function to its caller. In the function definition, the value 10 is returned. This value is then passed to the first argument (10) of the function call myFun(10, 20) in the line below. A dashed circle highlights the 10 in the return statement, and another dashed circle highlights the 10 in the function call. A curved arrow points from the first 10 to the second 10, showing the data flow. Additionally, a straight arrow points from the function call to the result variable, indicating that the return value is assigned to it.

함수

매개변수와 반환 값

- 함수에 매개변수가 선언되어 있다고 하더라도 함수를 호출하면서 매개변수에 인수를 전달하지 않아도 됩니다.
- 매개변수에 인수가 전달되지 않으면 그 매개변수는 값을 가지지 못하기 때문에 undefined 가 됩니다.

함수

default parameter

- default parameter 란 함수를 선언하면서 매개변수에 기본 값을 대입할 수 있다는 의미입니다.
- 함수에 매개변수를 선언하고 함수를 호출하면서 그 매개변수에 인수를 전달하지 않으면 undefined 가 되는데 만약 default parameter 가 선언이 되었다면 인수가 전달되지 않는 경우 그 매개변수는 선언하면서 지정한 값을 가지게 됩니다.

Default Parameter 생략

```
1 function myFun(arg1, arg2 = 0){  
2   console.log(`arg1 : ${arg1}, arg2 : ${arg2}`)
```

함수

reset parameter

- Rest Parameter 란 나머지 매개변수라는 의미입니다.
- Rest parameter 는 ... 으로 선언되는 매개변수입니다.
- Rest Parameter 는 하나의 매개변수이지만 여러 개의 값을 가질 수 있는 매개변수가 됨으로 내부적으로 배열로 만들어 집니다.
- 함수 내부에서는 Rest Parameter 의 매개변수를 배열로 이용하면 됩니다.

Rest Parameter 생략

```
1 function myFun(arg1, ...arg2){  
2   console.log(arg2)  
3   if(arg2.length > 0){  
4     for(let i=0; i<arg2.length; i++){
```

함수

reset parameter

- Rest Parameter 는 ... 이 추가된 매개변수이며 하나의 함수내에서 하나의 매개변수만 Rest Parameter 로 선언할 수 있습니다.
- 그리고 Rest Parameter 로 선언한 매개변수는 그 함수의 매개변수들 중 가장 마지막에 위치해야 합니다.

함수 표현식

- 함수 표현식이란 함수가 변수처럼 선언되는 것을 의미합니다.
- 소프트웨어 개발에서 흔히 표현식(Expression)이란 값이 도출되는 구문을 의미합니다.
- 그럼으로 표현식 구문은 변수에 대입되어 사용이 가능합니다.
- 그런데 함수 표현식이란 함수를 선언하는데 이 선언된 함수를 변수에 대입할 수 있다는 의미입니다.

함수 표현식	
1	//함수 선언문
2	function myFun1() {
3	console.log('myFun1 call')
4	}
5	//함수 표현식
6	const myFun2 = function(){
7	console.log('myFun2 call')
8	}
9	
10	myFun1();//myFun1 call
11	myFun2();//myFun2 call

화살표 함수

- 화살표 함수(Arrow Function) 은 다른 소프트웨어 언어에서는 람다함수(Lambda Function) 이라고 불리우는 함수입니다.
- 자바스크립트에서는 함수 선언에 화살표(=>) 가 들어간다고 해서 화살표 함수라고 칭하고 있습니다.

```
function myFun() {
```

```
}
```



```
() => {
```

```
}
```

화살표 함수

- 화살표 함수는 함수명을 주지 않음으로 이렇게 선언된 함수를 외부에서 호출할 방법이 없습니다.
- 그럼으로 일반적으로 화살표 함수는 변수에 대입되어 함수 표현식으로 선언됩니다

화살표 함수

```
1  const myFun = () => {  
2    console.log('myFun call')  
3  }  
4  myFun()//myFun call
```

- 만약 화살표 함수의 내용이 한줄이라면 함수 내용을 묶는 {} 을 생략할 수 있습니다.

한줄짜리 화살표 함수

```
1  const myFun = (arg1) => arg1 * 10  
2  let result = myFun(10)  
3  console.log(result)  
4  //20
```

화살표 함수

- => 왼쪽의 매개변수가 하나라면 () 를 생략할 수도 있습니다.

() 생략

```
1  const myFun1 = (arg1) => arg1 * 10  
2  const myFun2 = arg1 => arg1 * 10
```




람다함수

- 람다(Lambda) 함수, 자바스크립트에서는 람다 함수를 화살표 함수라고 부릅니다.
- 그런데 대부분의 소프트웨어에서는 람다함수라는 용어를 이용함으로 이 용어도 정리해 주면 좋습니다.
- 람다함수는 함수인데 이름이 없는 함수이며 작성규칙이 화살표를 기준으로 왼쪽이 매개변수, 오른쪽이 함수의 내용입니다.
- 소프트웨어 언어에 따라 약간의 문법 차이는 있는데 화살표를 어떤 언어는 `->` 을 사용하거나 아니면 `=>` 을 사용하기도 합니다.
- 약간의 문법차이는 있지만 화살표를 중심으로 이름없는 함수를 선언하겠다는 의미는 동일하며 이를 통칭해서 람다함수라고 부릅니다.

지역변수와 전역변수

- 변수 선언 위치에 따라 그 변수의 이용 범위가 다릅니다.
- 변수 선언이 함수 외부에 작성되어 있다면 이 변수는 전역변수(Global Variable) 이라고 부르며 어떤 함수 내부에 선언되어 있다면 이 변수는 지역변수(Local Variable) 이라고 부릅니다.
- 전역 변수는 프로그램 코드 전체 위치에서 이용이 가능합니다.
- 하지만 지역변수는 그 변수가 선언된 함수 내부에서만 사용할 수 있습니다.

```
let data1 = 10  
  
function myFun() {  
    let data2 = 20  
}  
  
}
```



스코프

- 소프트웨어 개발 혹은 IT 전반에 사용되는 용어입니다.
- 직역하자면 '범위'입니다.
- 무언가 영향을 미치는 범위를 스코프라고 합니다.
- 우리가 살펴본 지역변수는 그 지역변수를 선언한 함수내에서만 사용할 수 있습니다.
- 지역변수가 이용되는 범위가 함수내인 것이지요.
- 그래서 흔히 '지역변수는 함수 스코프에서 이용된다' 라는 표현을 많이 합니다.

익명함수

- 함수를 선언할 때 익명함수로 선언할 수 있습니다.
- 익명함수란 특별한 규칙이 있는 것이 아니라 함수를 선언하면서 함수명을 지정하지 않는 함수입니다.

익명함수	
1	<code>//일반함수</code>
2	<code>function myFun1() {</code>
3	
4	<code>}</code>
5	<code>//익명함수</code>
6	<code>const myFun2 = function(){</code>
7	
8	<code>}</code>
9	<code>////익명함수 - 화살표 함수</code>
10	<code>const myFun3 = () => {</code>
11	
12	<code>}</code>

익명함수

- 익명함수로 함수를 선언하자마자 그 함수를 변수에 대입해서 그 변수명으로 이용합니다. 함수를 변수에 대입해서 선언하는 방식을 흔히 '함수 표현식'이라고 합니다.
- 익명함수를 선언하는 대표적인 사례 중 하나가 함수를 어떤 다른 함수의 매개변수 혹은 반환값으로 이용하는 경우입니다.





High Order Function

- 함수가 매개변수 혹은 리턴 값으로 함수를 이용할 수 있는 것을 용어로 High Order Function 이라고 부르며 한국어로는 고차함수라고 합니다.
- 소프트웨어 언어 전반에서 사용되는 용어입니다.
- 함수를 매개변수 혹은 반환값으로 이용하여 함수와 함수를 유기적으로 연결해서 업무가 진행되게 하고자 하는 기법입니다.



감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어
William Shakespeare