3. 함수형 프로그래밍 (실습)

Prof. Seunghyun Park (sp@hansung.ac.kr)

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 3. 함수형 프로그래밍

- 코드 분석 준비
- 함수를 매개변수로 활용
- 함수의 결과로 함수를 반환
- 불변성: 원본 객체의 값 수정여부 확인
- 순수함수

실습1: 코드분석 준비

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>What It Means to Be Functional</title>
  <script src="https://unpkg.com/babel-
standalone@6.15.0/babel.min.js"></script>
</head>
<body>
<h1>Functional JavaScript</h1>

<script>
/* ch03-01.html */

a = 5;
console.log(a);

console.log(this);

</script>
</body>
</html>
```

```
<script>
<script type="text/javascript">
<script type="module">
<script type="text/babel">
```

```
/* ch03-02.js */

a = 5;
console.log(a);

console.log(this);
```

```
'use strict'
```

실습1: 코드분석 준비

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>What It Means to Be Functional</title>
  <script src="https://unpkg.com/babel-
standalone@6.15.0/babel.min.js"></script>
</head>
<body>
<h1>Functional JavaScript</h1>

<script>
/* ch03-01.html */

a = 5;
console.log(a);

console.log(this);

</script>
</body>
</html>
```

```
<script>
<script type="text/javascript">
<script type="module">
<script type="text/babel">
```

```
/* ch03-02.js */

a = 5;
console.log(a);

console.log(this);
```

```
'use strict'
```

```
a = 5;
  ^
ReferenceError: a is not defined
```

```
5
{}
```

```
5
Window {window: Window, self: Window, document:
#document, name: '', location: Location, …}
```

```
Uncaught ReferenceError ReferenceError: a is not
defined at <anonymous>
undefined
```

실습2: 함수를 매개변수로 활용 분석

```
/* ex03-03.html from ch03-01-05-functional.html */

const insideFn = logger => logger("함수를 다른 함수에 매개변수로 전달")

insideFn(message => console.log(message))
```

```
함수를 다른 함수에 매개변수로 전달
```

1. 코드의 주요 지점에 break point를 찍고,

2. 실행 중 객체의 값 확인

```
Watch
> insideFn
> logger
> message
```

실습2: 함수를 매개변수로 활용 분석

```
/* ex03-03.html from ch03-01-05-functional.html */
```

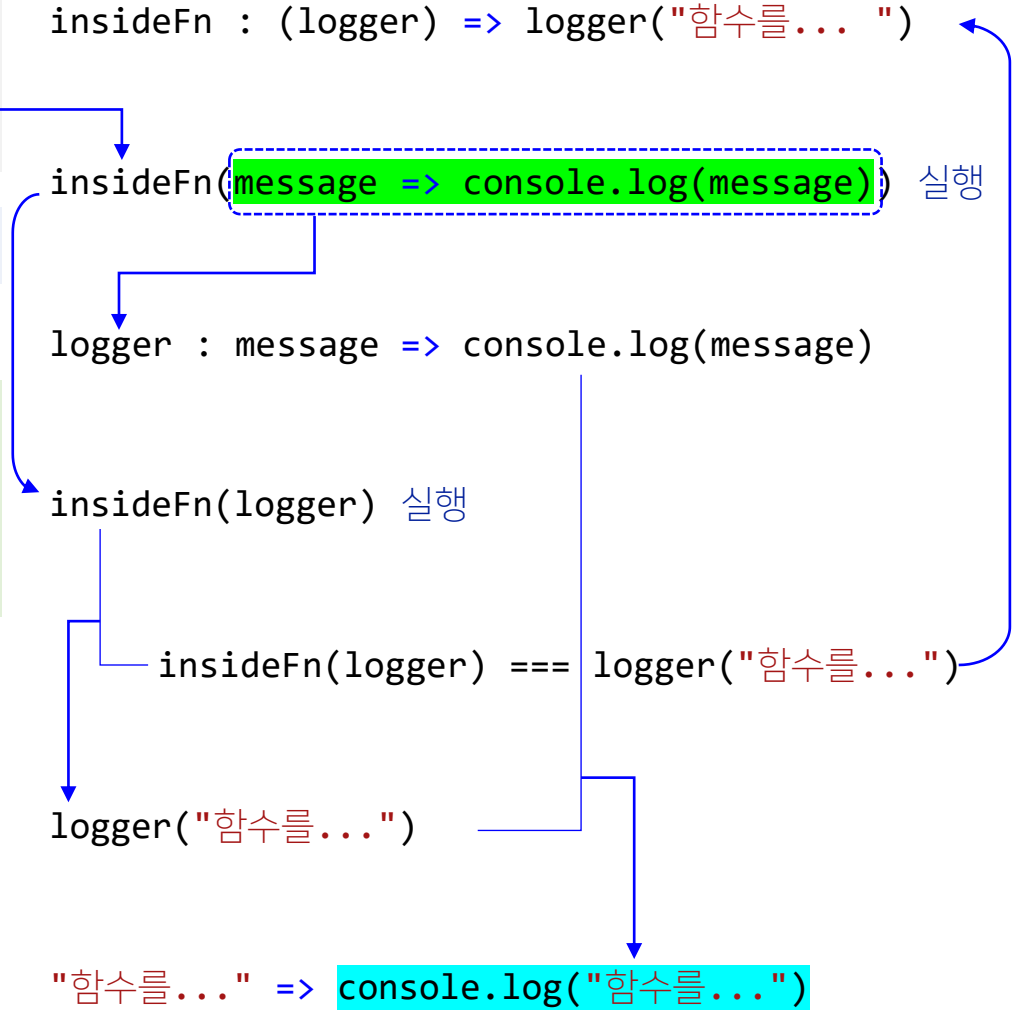
- `const insideFn = logger => logger("함수를 다른 함수에 매개변수로 전달")`
- `insideFn(message => console.log(message))`

함수를 다른 함수에 매개변수로 전달

1. 코드의 주요 지점에 break point를 찍고,

2. 실행 중 객체의 값 확인

```
Watch
> insideFn
> logger
> message
```



실습3: 함수의 결과로 함수를 반환

```
/* ex03-04.html from ch03-01-06-functional.html */

const createScream = function(logger) {
    return function(message) {
        logger(message.toUpperCase() + "!!!")
    }
}

const scream = createScream(message => console.log(message))
scream('function can return other functions')
```

```
FUNCTION CAN RETURN OTHER FUNCTIONS!!!
```

1. 코드의 주요 지점에 break point를 찍고,

2. 실행 중 객체의 값 확인

Watch
> createScream
> logger
> message
> scream

실습3: 함수의 결과로 함수를 반환

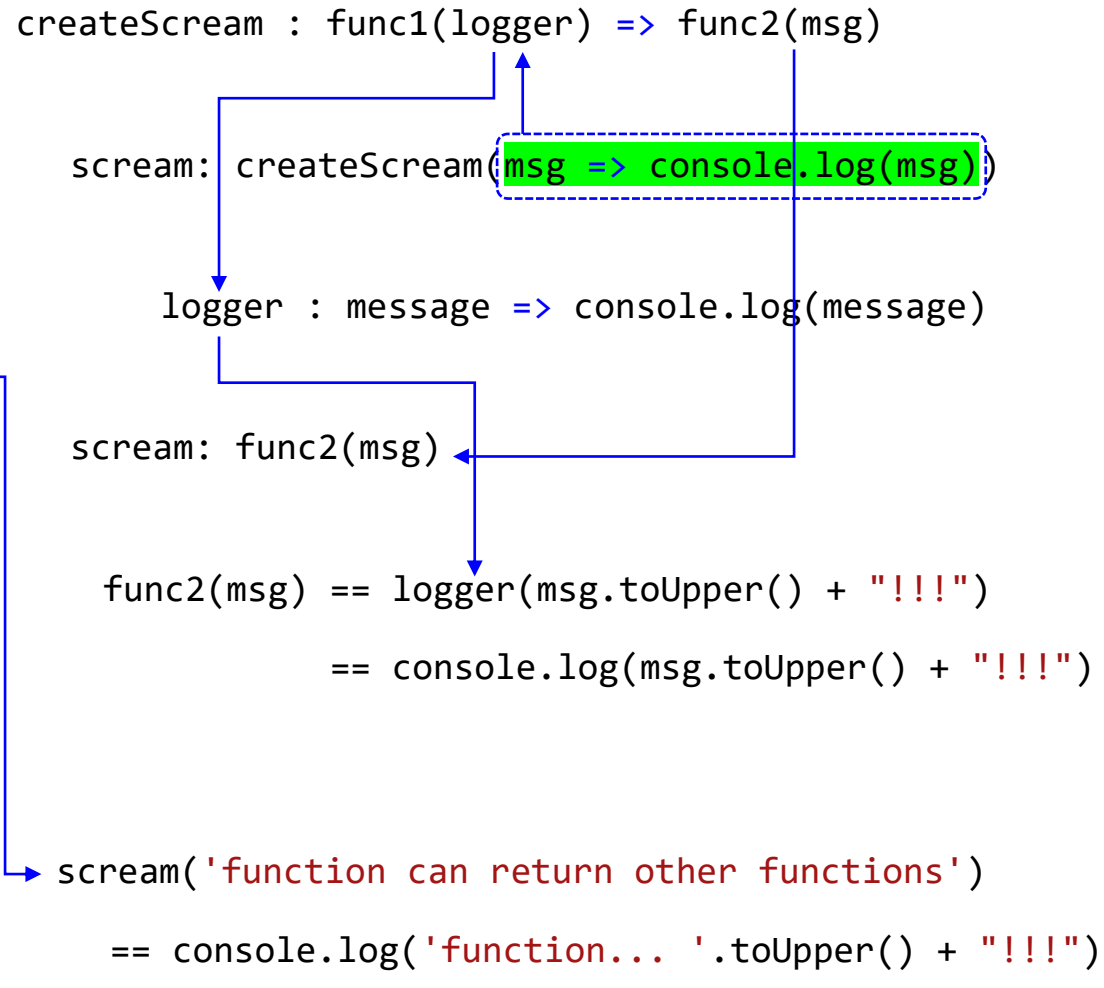
```
/* ex03-04.html from ch03-01-06-functional.html */

const createScream = function(logger) {
    return function(message) {
        logger(message.toUpperCase() + "!!!")
    }
}

const scream = createScream(message => console.log(message))
scream('function can return other functions')
```

```
FUNCTION CAN RETURN OTHER FUNCTIONS!!!
```

1. 코드의 주요 지점에 break point를 찍고,

2. 실행 중 객체의 값 확인

Watch
> createScream
> logger
> message
> scream



실습4: 원본 객체의 값 수정여부 확인

```
/* ex03-05.html from ch03-04-01-immutability.html */

let color_lawn = {
  title: "잔디",
  color: "#00FF00",
  rating: 0
}

function rateColor(obj, rating) {
  obj.rating = rating
  return obj
}

console.log(color_lawn.rating)

// rateColor는 원래의 색을 변경한다.
console.log(rateColor(color_lawn, 5).rating)
console.log(color_lawn.rating)
```

실습4: 원본 객체의 값 수정여부 확인

```
/* ex03-05.html from ch03-04-01-immutability.html */

let color_lawn = {
  title: "잔디",
  color: "#00FF00",
  rating: 0
}

function rateColor(obj, rating) {
  obj.rating = rating
  return obj
}

console.log(color_lawn.rating)

// rateColor는 원래의 색을 변경한다.
console.log(rateColor(color_lawn, 5).rating)
console.log(color_lawn.rating)
```

```
0
5
5
```

```
0
5
0
```

실습4: 원본 객체의 값 수정여부 확인

```
/* ex03-05.html from ch03-04-01-immutability.html */

let color_lawn = {
  title: "잔디",
  color: "#00FF00",
  rating: 0
}

function rateColor(obj, rating) {
  obj.rating = rating
  return obj
}

console.log(color_lawn.rating)

// rateColor는 원래의 색을 변경한다.
console.log(rateColor(color_lawn, 5).rating)
console.log(color_lawn.rating)
```

```
0
5
5
```

```
0
5
0
```

```
/* ex03-06.html from ch03-04-02-immutability.html */

const rateColor = function(obj, rating) {
  return Object.assign({}, obj, {rating:rating})

}
```

```
/* ex03-07.html from ch03-04-03-immutability.html */

const rateColor = (obj, rating) =>
    ({
        ...obj,
        rating
    })
```

실습5: 순수함수

```
/* ex03-08.html from ch03-05-01-pure-functions.html */
var frederick = {
  name: "Frederick",
  canRead: false,
  canWrite: false
}

// 순수하지 않은 함수
// 인자 없음, return 문 없음, 원본 객체를 변화시킴
function selfEducate() {
  frederick.canRead = true
  frederick.canWrite = true
}

console.log( frederick )

selfEducate()
console.log( frederick )
```

```
{name: 'Frederick', canRead: false, canWrite: false}
{name: 'Frederick', canRead: true, canWrite: true}
```

실습5: 순수함수

```
/* ex03-08.html from ch03-05-01-pure-functions.html */
var frederick = {
  name: "Frederick",
  canRead: false,
  canWrite: false
}

// 순수하지 않은 함수
// 인자 없음, return 문 없음, 원본 객체를 변화시킴
function selfEducate() {
  frederick.canRead = true
  frederick.canWrite = true
}

console.log( frederick )

selfEducate()
console.log( frederick )
```

```
{name: 'Frederick', canRead: false, canWrite: false}
{name: 'Frederick', canRead: true, canWrite: true}
```

```
/* ex03-09.html from ch03-05-02-pure-functions.html */
const selfEducate = person => {
  person.canRead = true
  person.canWrite = true
  return person
}
```

```
console.log( frederick )

console.log( selfEducate(frederick) )
console.log( frederick )
```

실습5: 순수함수

```
/* ex03-08.html from ch03-05-01-pure-functions.html */
var frederick = {
  name: "Frederick",
  canRead: false,
  canWrite: false
}

// 순수하지 않은 함수
// 인자 없음, return 문 없음, 원본 객체를 변화시킴
function selfEducate() {
  frederick.canRead = true
  frederick.canWrite = true
}

console.log( frederick )

selfEducate()
console.log( frederick )
```

```
{name: 'Frederick', canRead: false, canWrite: false}
{name: 'Frederick', canRead: true, canWrite: true}
```

```
/* ex03-09.html from ch03-05-02-pure-functions.html */
const selfEducate = person => {
  person.canRead = true
  person.canWrite = true
  return person
}
```

```
console.log( frederick )

console.log( selfEducate(frederick) )
console.log( frederick )
```

```
/* ex03-10.html from ch03-05-03-pure-functions.html */
const selfEducate = person =>
    ({
        ...person,
        canRead: true,
        canWrite: true
    })
```

순수 함수

1) 매개변수 전달

2) 결과 값 반환

3) 함수 밖 객체의 값을 변경하지 않음