3. 함수형 프로그래밍 (실습)

4. 리액트의 작동 원리 (실습)

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

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 3. 함수형 프로그래밍

- 함수형 프로그래밍
  - 데이터 변환
    - Array: join(), filter(callback), map(callback), reduce(callback), indexOf()
    - Object: keys(object)
  - 고차 함수
- 리액트의 작동 원리
  - React element 생성
    - React.createElement()
  - ReactDOM 렌더링
    - ReactDOM.render()

실습 준비: 데이터 변환

- 순수 함수를 사용한 데이터 처리 → 원본의 복제본을 생성하여 처리하고, 결과를 반환

- `Array.prototype.join()`
  - 배열의 모든 요소를 연결해 하나의 문자열로 만들어 반환

- `Array.prototype.filter(callback)`
  - 주어진 함수의 조건을 통과하는 모든 요소를 모아 새로운 배열로 반환

- `Array.prototype.map(callback)`
  - 배열 내 모든 요소에 대해 주어진 함수를 호출한 결과를 새로운 배열로 반환

- `Array.prototype.reduce(callback, initValue)`
  - 배열 내 모든 요소에 대해 callback을 실행하고 하나의 결과 값을 반환

- `Array.prototype.indexOf()`
  - 배열 내 지정된 요소를 찾을 수 있는 첫 번째 인덱스 반환 (단, 존재하지 않으면 -1 반환)

- `Object.keys(object)`
  - 주어진 객체의 속성 이름을 열거하는 배열을 반환

실습1: 데이터 변환 – Array.join()

- **문제: data의 요소를 (,)으로 연결하여 문자열로 출력**

  (단, 원래의 문자열을 변경하지 않음)

```
/* ex04-01.html */

const address = [
    "Division of Computer Engineering",
    "Hansung University",
    "116",
    "Samseongyo-ro",
    "16-gil",
    "Seongbuk-gu",
    "Seoul",
    "02876",
    "South Korea"
]

console.log( address )

console.log( address )
```

- Array.prototype.join()

  - 배열의 모든 요소를 연결해 하나의 문자열로 만들어 반환

```
(9) ['Division of Computer Engineering', 'Hansung
University', '116', 'Samseongyo-ro', '16-gil',
'Seongbuk-gu', 'Seoul', '02876', 'South Korea’]
Division of Computer Engineering, Hansung University,
116, Samseongyo-ro, 16-gil, Seongbuk-gu, Seoul, 02876,
South Korea
(9) ['Division of Computer Engineering', 'Hansung
University', '116', 'Samseongyo-ro', '16-gil',
'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
```

```
 0: "Division of Computer Engineering"
 1: "Hansung University"
 2: "116"
 ...
 8: "South Korea"
 length: 9
[[Prototype]]: Array(0)
```

실습2-1: 데이터 변환 – Array.filter()

- 문제: data의 요소 중 첫 글자가 'S'인 요소만 출력

```
 0: "Division of Computer Engineering"
 1: "Hansung University"
 2: "116"
 3: "Samseongyo-ro"
 4: "16-gil"
 5: "Seongbuk-gu"
 6: "Seoul"
 7: "02876"
 8: "South Korea"
 length: 9
[[Prototype]]: Array(0)
```

```
/* ex04-02.html */

const address = [ ... ];

console.log( address )

console.log( address )
```

- `Array.prototype.filter(callback)`
  - 주어진 함수의 조건을 통과하는 모든 요소를 모아 새로운 배열로 반환

```
(9) ['Division of Computer Engineering', 'Hansung University', '116', 'Samseongyo-ro', '16-gil', 'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
(4) ['Samseongyo-ro', 'Seongbuk-gu', 'Seoul', 'South Korea']
(9) ['Division of Computer Engineering', 'Hansung University', '116', 'Samseongyo-ro', '16-gil', 'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
```

실습2-2: 데이터 변환 – Array.filter()

- 문제: data의 요소 중
  첫 글자가 'S'가 아닌 요소만 출력하되,
  주어진 기능을 수행하는 함수로 구현

- `Array.prototype.filter(callback)`
  - 주어진 함수의 조건을 통과하는 모든 요소를 모아 새로운 배열로 반환

```
/* ex04-03.html */

const address = [ ... ];

console.log( address )

const newFilter =

console.log( newFilter(address, 'S') );

console.log( address )
```

```
(9) ['Division of Computer Engineering', 'Hansung University', '116', 'Samseongyo-ro', '16-gil', 'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
(5) ['Division of Computer Engineering', 'Hansung University', '116', '16-gil', '02876']
(9) ['Division of Computer Engineering', 'Hansung University', '116', 'Samseongyo-ro', '16-gil', 'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
```

실습3: 데이터 변환 – Array.map()

- **문제: data의 요소를 객체로 변환**

  (원래의 배열은 변경하지 않으며, 객체의 속성은 'element'로 지정)

```
/* ex04-04.html */

const address = [ ... ];

console.log( address )

const addrMap1 =
console.log( addrMap1 );

const addrMap2 =
console.log( addrMap2(address) );

console.log( address )
```

- `Array.prototype.map(callback)`
  - 배열 내 모든 요소에 대해 주어진 함수를 호출한 결과를 새로운 배열로 반환

```
(9) ['Division of Computer Engineering', 'Hansung
University', '116', 'Samseongyo-ro', '16-gil',
'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
(9) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
(9) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
0: {element: 'Division of Computer Engineering'}
1: {element: 'Hansung University'}
2: {element: '116'}
3: {element: 'Samseongyo-ro'}
4: {element: '16-gil'}
5: {element: 'Seongbuk-gu'}
6: {element: 'Seoul'}
7: {element: '02876'}
8: {element: 'South Korea'}
length: 9
[[Prototype]]: Array(0)
(9) ['Division of Computer Engineering', 'Hansung
University', '116', 'Samseongyo-ro', '16-gil',
'Seongbuk-gu', 'Seoul', '02876', 'South Korea']
```

실습4: 데이터 변환 – Array.reduce()

- 문제: data의 모든 요소를 탐색한 후 최대 값을 출력

```
/* ex04-05.html */

const ages = [ 20, 50, 60, 35, 70, 25, 45 ];

console.log( ages );

const maxAge =



console.log(`maxAge: ${maxAge}`);

const maxFinder1 =



console.log(`maxAge: ${maxFinder1(ages)}`);

console.log( ages );
```

- Array.prototype.reduce(callback, initValue)

- 배열 내 모든 요소에 대해 callback을 실행하고 하나의 결과 값을 반환

* *callback*:
(*accumulator*, *currentValue*, *currentIndex*, *array*) => { ... }
- *accumulator*: 콜백의 반환 값 누적 (mandatory)
- *currentValue*: 현재 처리할 요소 (mandatory)
- *currentIndex*: 현재 처리할 요소의 인덱스 (optional)
- *array*: reduce()를 호출한 배열 (optional)

```
(7) [20, 50, 60, 35, 70, 25, 45]
idx 1: 20 > 50 = false
idx 2: 50 > 60 = false
idx 3: 60 > 35 = true
idx 4: 60 > 70 = false
idx 5: 70 > 25 = true
idx 6: 70 > 45 = true
maxAge: 70
...
(7) [20, 50, 60, 35, 70, 25, 45]
```

실습5: 고차함수

- 문제: 코드 분석

```
/* ch03-07-01-higher-order-fns.html */

const invokeIf = (condition, fnTrue, fnFalse) =>
                    condition ? fnTrue() : fnFalse()

const showWelcome = () => console.log("Welcome!")

const showUnauthorized = () => console.log("Unauthorized!")

invokeIf(true, showWelcome, showUnauthorized)
invokeIf(false, showWelcome, showUnauthorized)
```

```
Watch
> invokeIf
> condition
> fnTrue
> fnFalse
> showWelcome
> showUnauthorized
```

```
invokeIf :
  (condition, fnTrue, fnFalse) =>
    condition ? fnTrue() : fnFalse()

  showWelcome : () => console.log("Welcome!")

  showUnauthorized : () => console.log(" Unauthorized!")
```

```
invokeIf(true, showWelcome, showUnauthorized)

  condition: true

  fnTrue: showWelcome

  fnFalse: showUnauthorized


  fuTrue() === () => console.log("Welcome!")
```

실습6: 고차함수

- 문제: 코드 분석

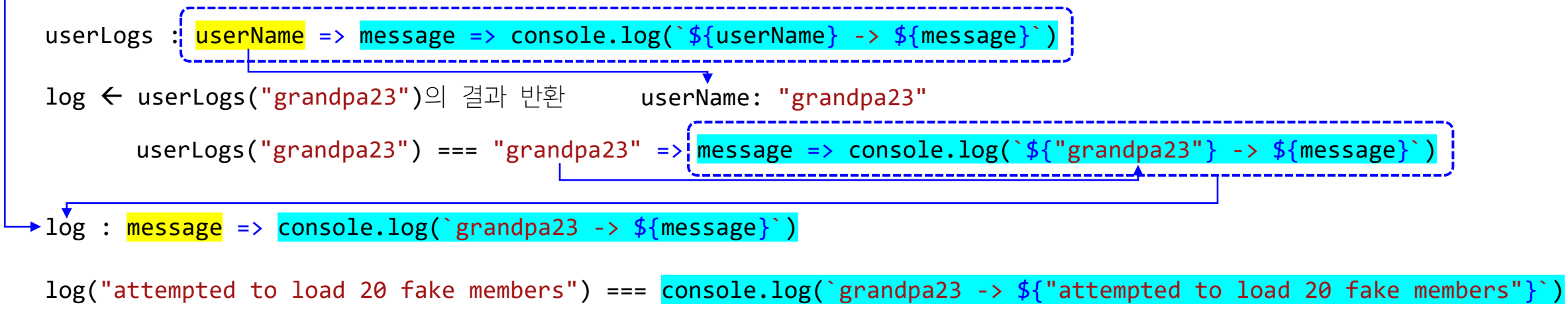
```
/* ch03-07-02-higher-order-fns.html */

const userLogs = userName => message => console.log(`${userName} -> ${message}`)

const log = userLogs("grandpa23")


log("attempted to load 20 fake members")
```

```
Watch
> userLogs
> userName
> message
> log
```



실습준비 – 빈 페이지 생성

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Page setup: ex04-08 from ch04-01-01-page-setup.html</title>
</head>
<body>

<!-- 타겟 컨테이너 -->
<div id="react-container"></div>

<!--React와 ReactDOM 라이브러리React Library & React DOM-->
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

<script>

/* ex04-08 */
// 순수 리액트와 자바스크립트 코드

</script>

</body>
</html>
```

실습7: 리액트 엘리먼트 생성하여 렌더링

- 문제: 엘리먼트 하위에 자식 노드로 <h1> 엘리먼트 생성

https://ko.reactjs.org/docs/react-api.html#createelement
https://ko.reactjs.org/docs/react-dom.html#render

```
<!-- Target Container -->
<div id="react-container"></div>

/* ex04-09 */
```

```
ReactDOM.render(
    addr,
    document.getElementById('react-container')
)

console.log('addr', addr)
```

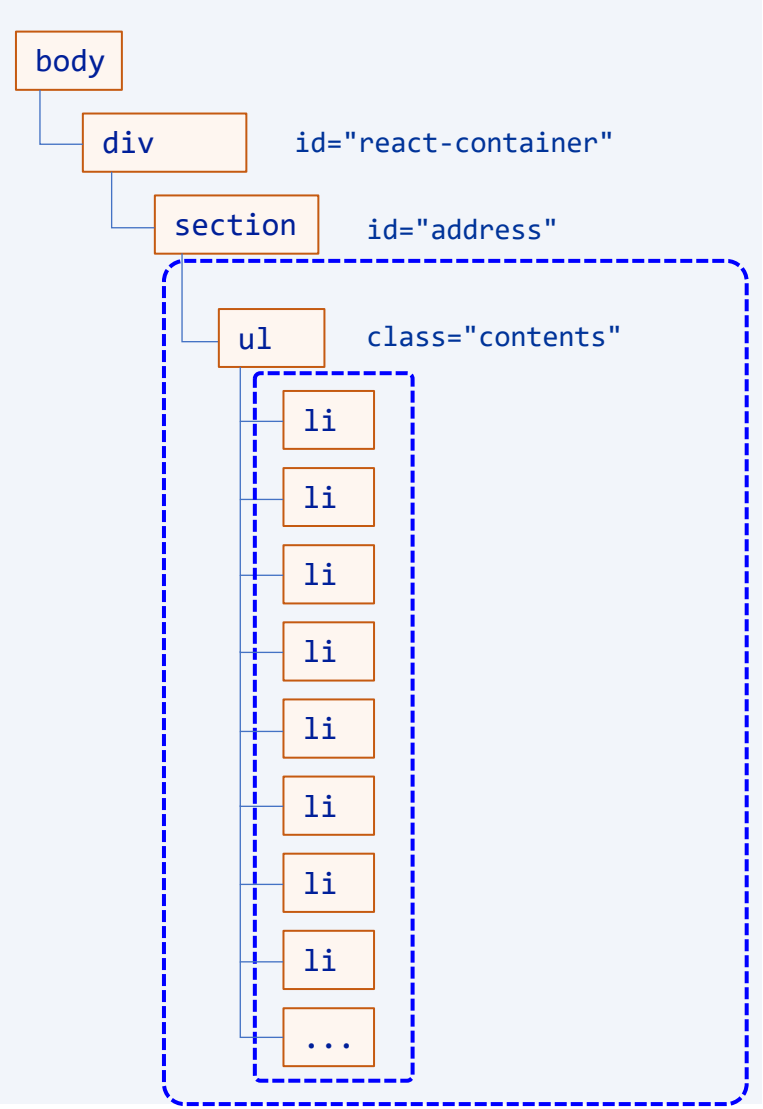
```
React.createElement(
  type, [props], [...children]
)
```

- 인자로 주어지는 타입에 따라 새로운 **리액트 엘리먼트를 생성하여 반환**

```
ReactDOM.render(
  element, container[, ...callback]
)
```

- 인자로 주어지는 렌더링 할 **리액트 엘리먼트**를
  제공된 컨테이너의 **DOM** (렌더링이 일어날 대상 DOM)**에 렌더링**,
- 구성요소에 대한 참조를 반환

실습8: 리액트 엘리먼트 생성하여 렌더링




```
<!-- Target Container -->
<div id="react-container"></div>

/* ex04-10 */
const address = [ ... ];

const addr =


ReactDOM.render(
    addr,
    document.getElementById('react-container')
)

console.log('addr', addr)
```

```
const address = [
    "Division of Computer Engineering",
    "Hansung University",
    "116",
    "Samseongyo-ro",
    "16-gil",
    "Seongbuk-gu",
    "Seoul",
    "02876",
    "South Korea"
];
```