JavaScript – Functional Programming

Sung-Dong Kim,
School of Computer Engineering
Hansung University

Contents

- 함수형 프로그래밍
- 유용한 함수들: filter(), map(), reduce()

함수형 프로그래밍

함수형 프로그래밍

- 세계 = 함수의 흐름
- 입력 – 출력 하는 작은 함수를 묶어서 프로그램을 구성하는 기법
- 외부 효과(side effect)가 없는 함수(순수 함수)만의 조합으로 프로그래밍 하는 방법 = 외부의 값을 읽어오지도 않고 영향을 미치지도 않는 함수만을 만드는 것이 목표
- 병렬 처리, 비동기 처리에 유리/적합
- 함수: 단 하나의 기능만 수행하는 것이 바람직
- 변수와 for문을 사용을 억제

함수형 프로그래밍

```
let num = 1;
function f(a) {
 return a + num;
}
console.log(f(1));
num = 2;
console.log(f(1));
```

```
let num = 1;
function f(a) {
num += 1;
return a + num;
}
console.log(f(1));
console.log(f(1));
```

함수형 프로그래밍

- 단점
  - 작성이 어려움
  - 실행 속도가 느림
  - 필요 메모리 증가
- 장점
  - 검증이 쉽다.
  - 동작 파악이 쉽다.

유용한 함수들

predicate

- true / false 만을 리턴하는 함수

```
const isApple = (fruit) => {
  if (fruit === 'apple') return true;
  return false;
}

console.log(isApple('apple'));
```

filter() 함수

- array.filter(item => 조건)
- array 에서 원소 하나(item)씩 조건 검사를 하여 true인 것만 push() 하는 방식

filter() 함수

```
const ages = [11, 12, 13, 16, 21, 31];


const upper16 = ages.filter(age => age > 16);
const under13 = ages.filter(age => age < 13)
const between12And21 = ages.filter(age => age > 12 && age < 21);


console.log('upper16:', upper16);
console.log('under13:', under13);
console.log('between12And21:', between12And21);
```

```
const students = [
    { name: 'ksd', age: 31, math: 85, english: 87 },
    { name: 'kjh', age: 19, math: 95, english: 97 },
    { name: 'csh', age: 27, math: 76, english: 80 },
    { name: 'mattue', age: 40, math: 56, english: 65 },
    { name: 'kelly', age: 33, math: 49, english: 100 },
];

const mathUpper80 = students.filter(s => s.math > 80);
const mathUpper70AndEnglishUpper80 = students.filter(s => s.math > 70 && s.english > 80);

console.log(`mathUpper 80: ${mathUpper80}`);
console.log(`mathUpper 70 and EnglishUpper 80: ${mathUpper70AndEnglishUpper80}`);

console.log('mathUpper80: ', mathUpper80);
```

map() 함수

- array.map(item => action)
- array 에서 원소 하나(item)씩 action을 실행하여 push() 하는 방식

map() 함수 예

```
// define
const list = [1, 2, 3];


// process
const multipledList = list.map(item => item * 10);
multipledList.forEach(item => console.log(item));
```

map() 함수 예

```
const listEmployee = [
    { name: 'ksd', age: 35, salary: 5000 },
    { name: 'kjh', age: 21, salary: 3000 },
    { name: 'csh', age: 32, salary: 6000 },
];

const raisedSalaryList = listEmployee.map(e => e.salary * 1.1);
raisedSalaryList.forEach(s => console.log(`salary: ${s}`));
raisedSalaryList.forEach(s => console.log(s));
```

reduce() 함수

- array.reduce(action, init_value)
- array 원소 하나(item)씩 action을 실행
- 첫 원소를 이용할 때는 init_value 이용 → result 생성
- 이 후 원소 부터는 result와 item을 이용하여 action 실행
- 최종 result를 리턴 → array의 모든 원소를 하나로 줄임

reduce() 함수 예

```
const scores = [10, 20, 30, 40, 50];


const sum = scores.reduce((a, b) => (a + b));
const sumWithInitValue = scores.reduce((a, b) => (a + b), 10);


console.log('sum :', sum);
console.log('sumWithInitValue :', sumWithInitValue);
```

reduce() 함수 예

```
const students=[
    {name:'ksd', age:31, score:85 },
    {name:'kjh', age:13, score:95 },
    {name:'csh', age:35, score:76 }
];
const scores = students.map(s => s.score);
const sum = scores.reduce((a, b) => a + b, 0);
console.log(`sum: ${sum}`);
```

Examples

- 수학 80점 이상 학생들의 점수 합 구하기
  - 수학 80점 이상 학생들 고르기: filter()
  - 선택한 학생들의 점수만 추출하기: map()
  - 합 구하기: reduce()

```
const students = [
    { name: 'ksd', age: 31, math: 85, english: 87 },
    { name: 'kjh', age: 19, math: 95, english: 97 },
    { name: 'csh', age: 27, math: 76, english: 80 },
    { name: 'mattue', age: 40, math: 56, english: 65 },
    { name: 'kelly', age: 33, math: 49, english: 100 },
];
```

Examples

- 개발팀의 나이 합계 구하기
  - 개발(development) 팀원 고르기: filter()
  - 선택한 직원들의 나이만 추출하기: map()
  - 합 구하기: reduce()

```
const employee = [
    { name: 'ksd', age: 35, department: 'design' },
    { name: 'kjh', age: 21, department: 'development' },
    { name: 'csh', age: 32, department: 'test' },
    { name: 'tom', age: 32, department: 'development' },
    { name: 'july', age: 32, department: 'development' },
    { name: 'mike', age: 32, department: 'test' },
];
```

Examples

- 21세 이상 30세 미만의 점수 평균 구하기
  - 21 <= age < 30 학생 고르기: filter()
  - 선택한 학생들의 점수만 추출하기: map()
  - 평균 구하기: reduce()

```
const students = [
    { name: 'ksd', age: 31, math: 85, english: 87 },
    { name: 'kjh', age:29, math: 95, english: 97 },
    { name: 'csh', age: 27, math: 76, english: 80 },
    { name: 'mattue', age: 40, math: 56, english: 65 },
    { name: 'kelly', age: 33, math: 49, english: 100 },
];
```