JavaScript 101 - (6) 순수함수와 배열 메소드(map, filter, reduce)

요약

- mutable은 생성된 후에도 값이 변경될 수 있는 객체를 의미합니다. 예를 들어, 객체, 배열, 함수, 클래스, 맵, 셋 등은 mutable입니다. mutable 타입은 각각 고유한 식별자를 가지고 있으며 참조에 의해 비교됩니다. 즉, 변수는 객체의 참조값을 가지고 있거나 메모리 주소를 가리킵니다.
- immutable은 생성된 후에는 값이 변경될 수 없는 원시값을 의미합니다. 예를 들어, 숫자, 문자열, null, undefined, 불리언 등은 immutable입니다. immutable 타입은 값 자체에 의해 비교됩니다. 즉, 변수는 실제 값을 가지고 있습니다.
- map은 배열의 모든 요소에 대해 주어진 함수를 적용하고 그 결과를 새로운 배열로 반환합니다. 예를 들어, 배열의 모든 요소에 2를 곱하는 함수를 map에 전달하면 새로운 배열에는 원래 배열의 요소들이 2배가 된 값들이 들어갑니다.
- filter는 배열의 모든 요소에 대해 주어진 조건을 만족하는지 검사하고 그 결과가 참인 요소들만 새로운 배열로 반환합니다. 예를 들어, 배열의 모든 요소가 짝수인지 검사하는 함수를 filter에 전달하면 새로운 배열에는 원래 배열의 짝수 요소들만 들어갑니다.
- reduce는 배열의 모든 요소에 대해 주어진 함수를 순차적으로 적용하고 하나의 최종 값을 반환합니다. 예를 들어, 배열의 모든 요소를 더하는 함수를 reduce에 전달하면 하나의 숫자 값이 반환됩니다.

순수함수

순수함수는 부작용이 없는 함수를 의미합니다. 즉, 함수의 실행 결과가 외부 상태에 영향을 주거나 받지 않는 함수입니다.

순수함수는 다음과 같은 특징을 가집니다.

- 함수는 단일 책임을 가집니다. 함수는 하나의 일만 수행합니다.
- 함수는 부작용이 없습니다. 함수는 자신의 범위 밖의 상태를 변경하지 않습니다.
- 함수는 참조 투명성을 가집니다. 함수는 동일한 입력에 대해 항상 동일한 출력을 반환합니다.

예를 들어, 다음과 같은 코드를 보세요.

```
// 순수함수 예시
function add(a, b) {
  return a + b;
}

// 비순수함수 예시
let c = 10;
function multiply(a) {
  c = a * c;
  return c;
}
```

add 함수는 순수함수입니다. 왜냐하면 add 함수는 매개변수 a와 b만으로 결과값을 반환하고 외부 상태에 영향을 주거나 받지 않기 때문입니다.

multiply 함수는 비순수함수입니다. 왜냐하면 multiply 함수는 전역 변수 c를 사용하고 변경하며 입력값 a에 따라 결과값이 달라지기 때문입니다.

JavaScript에서 배열과 객체는 mutable이기 때문에, 배열과 객체의 메소드는 대부분 순수함수로 작성되어 immutable한 값을 리턴합니다.

javascript에서 배열 메소드 중 map, filter, reduce

map

map은 배열의 모든 요소에 대해 주어진 함수를 적용하고 그 결과를 새로운 배열로 반환합니다.

문법

```
let newArray = array.map(function(element) {
  // element는 현재 처리중인 배열 요소입니다.
  // 여기서 element에 대해 원하는 작업을 수행하고 그 결과 값을 리턴합니다.
});
```

예시

```
let numbers = [1, 2, 3, 4];
// numbers의 모든 요소에 2를 곱하는 함수를 map에 전달합니다.
let doubled = numbers.map(function(element) {
  return element * 2;
});
console.log(doubled); // [2, 4, 6 ,8]
```

filter

filter는 배열의 모든 요소에 대해 주어진 조건을 만족하는지 검사하고 그 결과가 참인 요소들만 새로운 배열로 반환합니다.

문법

```
let newArray = array.filter(function(element) {
  // element는 현재 처리중인 배열 요소입니다.
  // 여기서 element가 조건을 만족하는지 판단하고 그 결과 값을 리턴합니다.
});
```

예시

```
let numbers = [1, 2 ,3 ,4];
// numbers의 모든 요소가 짝수인지 검사하는 함수를 filter에 전달합니다.
let even = numbers.filter(function(element) {
  return element % 2 ===0;
});
console.log(even); // [2 ,4]
```

reduce

reduce는 배열의 모든 요소에 대해 주어진 함수를 순차적으로 적용하고 하나의 최종 값을 반환합니다.

문법

```
let result = array.reduce(function(accumulator,currentValue) {
   // accumulator는 이전 단계에서 리턴된 값입니다. 초기값이 있다면 초기값부터 시작합니다.
   // currentValue는 현재 처리중인 배열요 소입니다.
   // 여기서 accumulator와 currentValue 사이에서 원하는 작업을 수행하고 그 결과 값을 리턴합니다.
},initialValue);
```

예시

```
let numbers = [1 ,2 ,3 ,4];
// numbers 의모 든요 소 를더 하 는함 수 를reduce 에전 달합 니다 .
let sum = numbers.reduce(function(accumulator,currentValue) {
  return accumulator + currentValue;
},0);
console.log(sum); // 10
```