2026.01.16.금

생성일	@2026년 1월 16일 오후 3:46
태그	

목차

▼ 1) 함수형 프로그래밍 실습

https://github.com/sanggyoon/202601-KakaoCloudAIaaS-TIL/tree/main/TestCode/20260116/함수형프로그래밍실습

```
// 비순수 함수 1
let total = 0;
function addToTotal(amount) {
  total += amount;
  return total;
}

// 비순수 함수 2
function shuffle(arr) {
  for (let i = arr.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [arr[i], arr[j]] = [arr[j], arr[i]];
  }
  return arr;
}

// 비순수 함수 3
const config = { debug: false };
function log(message) {
  if (config.debug) {
    console.log(message);
  }
}

//TODO: 순수 함수 버전 구현
// 순수 함수 1
function add(a, b) {
  return a + b;
}

// 순수 함수 2
function getShuffledArray(arr) {
  const newArr = arr.slice(); // 원본 배열 복사
  for (let i = newArr.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [newArr[i], newArr[j]] = [newArr[j], newArr[i]];
  }
  return newArr;
}

// 순수 함수 3
function createLogger(isDebug) {
  return function (message) {
    if (isDebug) {
```

```
      console.log(message);
    }
  };
}
```

```
const state = {
  user: {
    profile: {
      name: 'Kim',
      email: 'kim@test.com',
    },
    settings: {
      theme: 'dark',
      notifications: {
        email: true,
        push: false,
      },
    },
  },
  posts: [
    { id: 1, title: 'Hello', likes: 10 },
    { id: 2, title: 'World', likes: 20 },
  ],
};

//TODO: 다음 함수들을 구현하세요

function updateUserName(state, newName) {
  // user.profile.name 업데이트
  return {
    ...state,
    user: {
      ...state.user,
      profile: {
        ...state.user.profile,
        name: newName,
      },
    },
  };
}

function togglePushNotification(state) {
  // user.settings.notifications.push 토글
  return {
    ...state,
    user: {
      ...state.user,
      settings: {
        ...state.user.settings,
        notifications: {
          ...state.user.settings.notifications,
          push: state.user.settings.notifications.push ? false : true,
        },
      },
    },
  };
}

function incrementPostLikes(state, postId) {
```

```
  // 특정 post의 likes 증가
  return {
    ...state,
    posts: state.posts.map((post) => {
      if (post.id === postId) {
        return {
          ...post,
          likes: post.likes + 1,
        };
      } else {
        return {
          ...post,
        };
      }
    }),
  };
}

function addPost(state, newPost) {
  // posts 배열에 새 포스트 추가
  return {
    ...state,
    posts: [...state.posts, newPost],
  };
}

// 테스트
const s1 = updateUserName(state, 'Lee');
console.log(s1.user.profile.name); // 'Lee'
console.log(state.user.profile.name); // 'Kim' (원본 유지)

const s2 = togglePushNotification(state);
console.log(s2.user.settings.notifications.push); // true

const s3 = incrementPostLikes(state, 1);
console.log(s3.posts[0].likes); // 11

const s4 = addPost(state, { id: 3, title: 'New', likes: 0 });
console.log(s4.posts.length); // 3
```

```
//TODO: 구현하세요

function myMap(arr, fn) {
  // Array.prototype.map 구현
  const result = [];
  for (let i = 0; i < arr.length; i++) {
    result.push(fn(arr[i], i, arr));
  }
  return result;
}

function myFilter(arr, predicate) {
  // Array.prototype.filter 구현
  const result = [];
  for (let i = 0; i < arr.length; i++) {
    if (predicate(arr[i], i, arr)) {
      result.push(arr[i]);
    }
  }
```

```
  return result;
}

function myReduce(arr, reducer, initialValue) {
  // Array.prototype.reduce 구현
  let accumulator = initialValue;
  let startIndex = 0;
  if (accumulator === undefined) {
    accumulator = arr[0];
    startIndex = 1;
  }
  for (let i = startIndex; i < arr.length; i++) {
    accumulator = reducer(accumulator, arr[i], i, arr);
  }
  return accumulator;
}

function myFind(arr, predicate) {
  // Array.prototype.find 구현
  for (let i = 0; i < arr.length; i++) {
    if (predicate(arr[i], i, arr)) {
      return arr[i];
    }
  }
  return undefined;
}

function myEvery(arr, predicate) {
  // Array.prototype.every 구현
  for (let i = 0; i < arr.length; i++) {
    if (!predicate(arr[i], i, arr)) {
      return false;
    }
  }
  return true;
}

function mySome(arr, predicate) {
  // Array.prototype.some 구현
  for (let i = 0; i < arr.length; i++) {
    if (predicate(arr[i], i, arr)) {
      return true;
    }
    return false;
  }
}

// 테스트
const nums = [1, 2, 3, 4, 5];

console.log(myMap(nums, (x) => x * 2)); // [2, 4, 6, 8, 10]
console.log(myFilter(nums, (x) => x % 2 === 0)); // [2, 4]
console.log(myReduce(nums, (a, b) => a + b, 0)); // 15
console.log(myFind(nums, (x) => x > 3)); // 4
console.log(myEvery(nums, (x) => x > 0)); // true
console.log(mySome(nums, (x) => x > 4)); // true
```

▼ 2) 리액트 memo project

https://github.com/sanggyoon/202601-KakaoCloudAIaaS-TIL/tree/main/TestCode/20260116/memo-app

- CURD 구현
- localStorage에 데이터 저장
- 배열/객체 상태 업데이트 패턴