```
// 객체 선언
const user = {
    name: '홍길동',
    age: 20,
    isMan: true
};

// 사용
console.log(name); // X
console.log(user.name);

// 기본적인 할당
const name = user.name;
const age = user.age;
const isMan = user.isMan;
console.log(isMan);

// 구조분해할탕
const {name,age,isMan} = user;
console.log(age);

const {age, name} = user; // 필요한 것만 구조분해할당, 순서가 바뀌어도 상관없음
console.log(name);

// 바로 할당도 가능.
const { a, b, c } = { a: '신사임당', b: 30, c: false }
console.log( a, b, c );

// 새로운 변수명에 할당
const { a: userName, b: userAge } = { a: '이순신', b: 40, c: true }
console.log( userName, userAge );
//=============================================================

// 배열 구조분해할당
// 1. 객체 구조분해 할당과 크게 다르지 않다.
// 2. 배열내의 요소를 새로운 이름으로 지정해서 사용하고 싶을 때 자주 사용
// 3. 필요없는 요소를 어떻게 처리하는지가 객체 구조분해할당과 차이점이 있다.
//    객체분해 할당=>필요없으면 skip 처리
//    배열구조분해할당=> 필요없으면 empty처리
const ar = [ 2050, 5, 31 ];
const [ year, month, day ] = ar;
console.log(year,month,day);

//  바로 구조분해 할당
const [ year_, month_, day_ ] = [ 2077, 7, 31 ];


// 필요없는 배열 요소 처리하기: empty
const ar2 = [ 1, 2, 3, 4, 5 ];
const [ one, two, five ] = ar2;
console.log( one, two, five );  // 1, 2, 5 => ??? (결과는 => 1, 2, 3)
const [ one_, two_,    ,    , five_ ] = ar2;  // empty처리
```

```
console.log( one_, two_, five_ );  // 1, 2, 5

// 4. 디폴트 값 지정하기
const ar3 = [ 180, 2010, 9 ];  // 유저키, 유저가입년도, 유저레벨
const [ userHeight, userJoinYear, userLevel ] = ar3;
console.log( userHeight, userJoinYear, userLevel );

const ar4 = [ 179, 2002];
const [ userHeight_, userJoinYear_, userLevel_ = 0 ] = ar4; // 디폴트값
console.log( userHeight_, userJoinYear_, userLevel_ );  // 179, 2002, 0

const ar5 = [ 179, 2002,5];
const [ _userHeight_, _userJoinYear_, _userLevel_ = 0 ] = ar5;  // 디폴트값
console.log( _userHeight_, _userJoinYear_, _userLevel_ );  // 179, 2002, 5

//=============================================================

// 중첩된 객체의 구조분해할당
// 객체정의
const someData = {
    a: 'KT',
    b: [
        {
            name: '홍길동',
            sns: [],
            address: '대구시 수성구 수성동 수성APT 1동 2345호',
            email: 'mr.hong@kt.com',
        },
    ],
    c: '...',
    d: '...',
};
// 필요한 데이터만 이름재정의해서 구조분해할당 => userCompany, userName, userEmail
const { a: userCompany, b: [{ name: userName_ }], b: [{ email: userEmail }] } =
someData;
// const { a: userCompany, b: [{ name: userName_, email: userEmail}] } = someData;
console.log(userCompany,  userName_, userEmail); //  KT 홍길동 mr.hong@kt.com


const someData = {
    a: 'KT',
    b: [
        {
            name: '홍길동',
            sns: [],
            address: '대구시 수성구 수성동 수성APT 1동 2345호',
            email: '',
        },
    ],
    c: '...',
    d: '...',
};

const { a: userCompany, b: [{ name: userName_ }], b: [{ email:
```

```
userEmail='test@test.com' }] } = someData;
console.log(userCompany,  userName_, userEmail); //  KT 홍길동 아무것도출력되지않음


// 객체정의
const someData = {
    a: 'KT',
    b: [
        {
            name: '홍길동',
            sns: [],
            address: '대구시 수성구 수성동 수성APT 1동 2345호',
           // email: '',
        },
    ],
    c: '...',
    d: '...',
};
const { a: userCompany, b: [{ name: userName_,  email: userEmail='test@test.com'
}]} = someData;
console.log(userCompany,  userName_, userEmail); //  KT 홍길동 test@test.com

//=================================================================

//  반복문에서의 구조분해할당 => for .. of
// 객체정의
const contest = [
    {
        team: 'skyblue',
        members: {
            member1: 'aaa',
            member2: 'bbb',
            member3: 'ccc',
            leader: 'abc'
        },
        nationality: 'KOREA',
    },
    {
        team: 'reddog',
        members: {
            member1: 'xxx',
            member2: 'yyy',
            member3: 'zzz',
            leader: 'xyz'
        },
        nationality: 'USA',
    },
    {
        team: 'whitehouse',
        members: {
            member1: 'xxx',
            member2: 'yyy',
            member3: 'zzz',
            leader: 'superman'
```

```
        },
        nationality: 'CANADA',
    },
];
// 반복문 => for .. of
for ( let { team: t, members: { leader: zzang }, } of contest ) {
    // console.log( '팀명: ' + t + ' => 팀장: ' + zzang  );
    console.log( `팀명: ${ t } => 팀장: ${ zzang }` );
}


//=============================================================

// 파라미터 값으로 구조분해할당 => 리액트에서 props 전달 시 이런식으로 구조분해할당
// 객체정의
const member = {
    id: 'mr.kim',
    koreanName: '김유신',
    englishName: { firstName: 'Kim', lastName: 'YuSin' },
};
function userKoreanName( {id, koreanName } ) {
    //return koreanName;
    return [id,koreanName]
}
console.log( userKoreanName(member) );

const member1= {
    id: 'dr.hong',
    koreanName: '홍박사',
    englishName: { firstName: 'Hong', lastName: 'BakSa' },
};
function userKoreanName( {englishName:  { firstName } } ) {
    return firstName;
}
console.log( userKoreanName(member1) );

//=============================================================

// 구조분해할당을 사용한 동적 변수 처리
const person = {
    pastime: { // 취미, 심심풀이
        a: '영화보기',
        b: '음악감상',
        c: '등산'
    }
};
function printPastime( obj, choice ) {
    // let { ['a'] : userPastime = 'Unknown' } = obj; //영화보기
    let { [choice] : userPastime = 'Unknown' } = obj;
    // 매칭되는 key가 없을 때 디폴트값 지정
    // 디폴트 값지정하지 않으면 undefined로 반환
    console.log( `${ userPastime }` );

}
```

```
printPastime( person.pastime, 'c' );    // 등산
printPastime( person.pastime, '' ); // Unknown, userPastime = 'Unknown' 부분을
userPastime으로 수정하면 undefined 출력


//================================================================

// 구조분해할당 => rest 나머지 패턴
//rest는 spread와 비슷, 차이점도 있음
// rest는 객체와 배열 그리고 함수의 파라미터에서 넘겨받은 값을 묶어줄 때 사용
// rest라는 이름을 꼭 쓸 필요는 없음. 관례적으로 rest 사용
//{...변수명}   형태 사용
//[...변수명] 형태 사용도 가능
// ● 첫 번째 예제
const animals = [
    { name: '하마', age: 10 },
    { name: '호랑이', age: 5 },
    { name: '사자', age: 7 },
    { name: '독수리', age: 4 },
];
const { ...rest } = animals;
console.log( rest );
// [참고] 한 마리 추가 => push() => 리액트에서는 이렇게 사용하지 않음.
animals.push( { name: '코끼리', age: 11 } );
console.log( animals );

// ● 두 번째 예제
const someObj = {
    name1: 'mr.hong',
    age1: 28,
    greeting: 'Hi~',
};
const { name1, ...rest1 } = someObj;
console.log( name1 );   // mr.hong
console.log( rest1 );       // { age1: 28, greeting: 'Hi~' }

// ● 세 번째 예제 ★★★★★ => 이 방식이 리액트에서 많이 사용되는 방식
// 리엑트 엔진은 상태전용 함수 훅(Hooks)를 사용해서 변수의 변화를 감사하지 않으면 새로운
값을 push()하면 초기화 시켜버림
const family = {
    father: '철수'
};
const family2 = {
    ...family,
    mother: '영희'
};
const family3 = {
    ...family2,
    child: '옥동자'
};
console.log( family );      // 철수
console.log( family2 );     // 철수, 영희
console.log( family3 ); // 철수, 영희, 옥동자
// 보충 예제
```

```
const zooAnimals = [ 'hippo', 'elephant', 'dove' ];
const restAnimals = [ ...zooAnimals, 'rhino' ];
console.log( zooAnimals );  // hippo, elephant, dove
console.log( restAnimals ); // hippo, elephant, dove, rhino


//================================================================

// ● 자바스크립트 rest vs spread 차이점 ★★★
//  1. 함수 파라미터에서 ==> ...사용하면 =>rest 나머지 패턴이며 => 나머지 모든 요소를 배열 또는 객체로 묶어 처리
//  2. 함수호출시 ==> ... 사용하면 => spread연산자이며 =>묶여있는 덩어리를 각각의 요소로 분해(해체)하여 전달

// [1] rest 패턴: 함수 정의시
function printNumber( a, b, c, ...rest ) {
    console.log( a );       // 1
    console.log( b );       // 2
    console.log( c );       // 3
    console.log( rest );    // 4, 5, 6, 7, 8, 9, 10 => 배열로 하나에 묶어서 출력
}
printNumber( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 );

// [2] spread(펼침)연산: 함수 호출시
function printSum( a, b, c ) {
    return (
        a + b + c
    )
}
const arNum = [ 100, 200, 300 ];
const result = printSum( ...arNum );  // printSum( 100, 200, 300 ); <-- 이렇게 호출한 것과 동일
console.log( result );  // 600



//================================================================


// ● 리액트에서 구조분해할당 사용 예
// [1]
import React, { Container, Navbar, ... } from 'react';
// [2]
const [ inputData, setInputData ] = useState( {
    name: "",
    age: "",
    email: "",
} );
//[3]
const { name, age, email } = inputData;

//================================================================
```