자동 줄바꿈 ☐

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>구조분해할당</title>
    </head>
    <body>

        <h1>Destructuring Assignment</h1>

        <script>
            // ● 구조분해할당(Destructuring Assignment)

            // [1] 객체 구조분해할당
            // 객체 선언
            const user = {
                name: '홍길동',
                age: 20,
                isMan: true
            };

            // [2] 구조분해할당은 순서가 바뀌어도 상관없음 => 바로 할당도 가능.
            // const { a, b, c } = { a: '신사임당', b: 30, c: false }
            // console.log( a, b, c );


            // [3] 새로운 변수명에 할당
            const { a: userName, b: userAge } = { a: '이순신', b: 40, c: true }
            console.log( userName, userAge );


            // 배열 구조분해할당
            // const ar = [ 2050, 5, 31 ];
            const [ year, month, day ] = [ 2077, 7, 31 ];

            console.log( year );
            console.log( month );
            console.log( day );

            // 필요없는 배열 요소 처리하기
            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;
            console.log( one_, two_, five_ );  // 1, 2, 5

            // 디폴트 값 지정하기
            const ar3 = [ 180, 2010, 9 ];  // 유저키, 유저가입년도, 유저레벨
            const [ userHeight, userJoinYear, userLevel ] = ar3;

            const ar4 = [ 179, 2002, 5 ];
            const [ userHeight_, userJoinYear_, userLevel_ = 0 ] = ar4;

            console.log( userHeight_, userJoinYear_, userLevel_ );  // 179, 2002, 0


            // [4] 중첩된 객체의 구조분해할당
            const someData = {
                a: 'KT',
                b: [
                    {
                        name: '홍길동',
                        sns: [],
                        address: '경기도 성남시 분당구 서현동 서현APT 1111동 2222호',
                        email: 'mr.hong@kt.com',
                    },
                ],
                c: '...',
                d: '...',
            };

            // 필요한 데이터만 구조분해할당 => userCompany, userName, userEmail
            const { a: userCompany, b: [{ name: userName_ }], b: [{ email: userEmail }] } = someData;
```

```
76
77              console.log( userCompany );     // KT
78              console.log( userName_ );       // 홍길동
79              console.log( userEmail );       // mr.hong@kt.com
80
81
82              // [5] 반복문에서의 구조분해할당 => for .. of ★★★
83              const contest = [
84                  {
85                      team: 'skyblue',
86                      members: {
87                          member1: 'aaa',
88                          member2: 'bbb',
89                          member3: 'ccc',
90                          leader: 'abc'
91                      },
92                      nationality: 'KOREA',
93                  },
94                  {
95                      team: 'reddog',
96                      members: {
97                          member1: 'xxx',
98                          member2: 'yyy',
99                          member3: 'zzz',
100                         leader: 'xyz'
101                     },
102                     nationality: 'USA',
103                 },
104                 {
105                     team: 'whitehouse',
106                     members: {
107                         member1: 'xxx',
108                         member2: 'yyy',
109                         member3: 'zzz',
110                         leader: 'superman'
111                     },
112                     nationality: 'CANADA',
113                 },
114             ];
115
116             // 반복문 => for .. of
117             for ( let { team: t, members: { leader: zzang }, } of contest ) {
118                 // console.log( '팀명: ' + t + ' => 팀장: ' + zzang  );
119                 console.log( `팀명: ${ t } => 팀장: ${ zzang }` );
120             }
121
122
123             // [6] 파라미터 값으로 구조분해할당 ★★★ => 리액트에서 props 전달 시 이런식으로 구조분해할당
124             const member = {
125                 id: 'mr.kim',
126                 koreanName: '김유신',
127                 englishName: { firstName: 'Kim', lastName: 'YuSin' },
128             };
129
130             function userKoreanName( { englishName: { firstName } } ) {
131                 return firstName;
132             }
133
134             console.log( userKoreanName(member) );
135
136
137             // [7] 구조분해할당을 사용한 동적 변수 처리 ★★★
138             const person = {
139                 pastime: {
140                     a: '영화보기',
141                     b: '음악감상',
142                     c: '등산'
143                 }
144             };
145
146             function printPastime( obj, choice ) {
147                 let { [choice] : userPastime = 'Unknown' } = obj;
148                 console.log( `${ userPastime }` );
149             }
150
151             printPastime( person.pastime, 'c' );    // 등산
```

```
printPastime( person.pastime, '' ); // Unknown


// [8] 구조분해할당 => rest 나머지 패턴 ★★★★★

// ● 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
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
console.clear();


// ● 자바스크립트 rest vs spread 차이점 ★★★

// [1]
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]
function printSum( a, b, c ) {
    return (
        a + b + c
    )
}
```

```
228
229             const arNum = [ 100, 200, 300 ];
230             const result = printSum( ...arNum );  // printSum( 100, 200, 300 ); <-- 이렇게 호출한 것과 동일
231             console.log( result );  // 600
232
233
234             // ● 리액트에서 구조분해할당 사용 예
235             // [1]
236             import React, { Container, Navbar, ... } from 'react';
237
238             // [2]
239             const [ inputData, setInputData ] = useState( {
240                 name: "",
241                 age: "",
242                 email: "",
243             } );
244             const { name, age, email } = inputData;
245
246         </script>
247     </body>
248 </html>
249
250
251
252
253
254
255
256
257
258
259
260
```