KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Vue.js를 위한 ES6(심화1)

[KB] IT's Your Life

KB 국민은행

- 전개연산자를 사용하여 obj1의 기본 값을 obj2로 복사한 후, email의 값을 mspark@gmail.com 으로 변경하세요.
  - 02-23.js

```
let obj1 = { name:"박문수", age:29 };
let obj2 =  _____________________________;

console.log(obj2);
```

```
{ name: '박문수', age: 29, email: 'mspark@gmail.com' }
```

02-23.js

```
let obj1 = { name:"박문수", age:29 };
let obj2 = { ...obj1, email:"mspark@gmail.com" };  //새로운 속성 추가

console.log(obj2);
```

```
{ name: '박문수', age: 29, email: 'mspark@gmail.com' }
```

전개 연산자를 사용하여 아래 결과가 나오도록 배열을 정의하세요.

```
let arr1 = [ 100, 200, 300 ];
let arr2 = _____________________________________;
console.log(arr1);
console.log(arr2);
```

```
[ 100, 200, 300 ]
[ 'hello', 100, 200, 300, 'world' ]
```

전개 연산자를 사용하여 아래 결과가 나오도록 배열을 정의하세요.

```
let arr1 = [ 100, 200, 300 ];
let arr2 = [ 'hello', ...arr1, 'world'];
console.log(arr1);
console.log(arr2);
```

```
[ 100, 200, 300 ]
[ 'hello', 100, 200, 300, 'world' ]
```

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Vue.js를 위한 ES6(심화2)

[KB] IT's Your Life

KB 국민은행

다음과 같이 출력되도록 아래 코드에서 Promise 객체를 완성하세요.

- 02-21.js

```
const p = new Promise((_____________________________) => {
    setTimeout(()=> {
        let num = Math.random();        //0~1사이의 난수 발생
        if (num >= 0.8) {
            _____________________________;
        }
        _____________________________;
    }, 2000)
})

p.___________((result)=> {
    console.log("처리 결과 : ", result)
})
.___________((error)=>{
    console.log("오류 : ", error)
})

console.log("## Promise 객체 생성!");
```

```
## Promise 객체 생성!
처리 결과 :  0.32435778048671526

## Promise 객체 생성!
오류 :  생성된 숫자가 0.8이상임 - 0.811963599046789
```

02-21.js Promise 객체

```
const p = new Promise((resolve, reject) => {
    setTimeout(()=> {
        let num = Math.random();        //0~1사이의 난수 발생
        if (num >= 0.8) {
            reject("생성된 숫자가 0.8이상임 - " + num);
        }
        resolve(num);
    }, 2000)
})

p.then((result)=> {
    console.log("처리 결과 : ", result)
})
.catch((error)=>{
    console.log("오류 : ", error)
})

console.log("## Promise 객체 생성!");
```

```
## Promise 객체 생성!
처리 결과 :  0.32435778048671526

## Promise 객체 생성!
오류 :  생성된 숫자가 0.8이상임 - 0.811963599046789
```

Proimise 체인을 사용하여 다음과 같은 결과가 나오도록 아래 코드를 완성하세요.

- 02-22.js

```
let p = new Promise((resolve, reject)=> {
    _____________________;
})

p.then((msg)=> {
    _____________________;
    _____________________;
})
._____((msg)=>{
    _____________________;
    _____________________;
})
.catch((error)=> {
    console.log("오류 발생 ==>  " + error)
})
```

```
first!
second
third
```

02-22.js Promise 체인 – 비동기 작업의 순차 실행

```
let p = new Promise((resolve, reject)=> {
    resolve("first!")
})

p.then((msg)=> {
    console.log(msg);
    // throw new Error("## 에러!!")
    return "second";
})
.then((msg)=>{
    console.log(msg);
    return "third";
})
.then((msg)=>{
    console.log(msg);
})
.catch((error)=> {
    console.log("오류 발생 ==>  " + error)
})
```

```
first!
second
third


first!
오류 발생 ==>  Error: ## 에러!!
```