[Part 02. ECMAScript 6 연산자, 제어문]

Contents.

1. Spread 연산자(스프레드 연산자, 펼침 연산자)
→ 펼침 연산자(...)는 이미 존재하고 있는 배열, 객체 등의 기존 요소에 새로운 항목을 추가하여 더 많은 요소로 확장합니다.

1) 형식 → 배열, 객체에 추가하고자 하는 요소를 아래와 같이 추가할 수 있습니다.

• 보기.
```
const ary = [1, 2, 3];
console.log(ary.toString());

const aryNew = [...ary, 4, 5, 6];   // 펼침연산자로 배열 항목 추가
console.log(aryNew.toString());
```

• 출력결과.

```
1, 2, 3
1, 2, 3, 4, 5, 6
```

2) 활용 예 ① → 펼침 연산자를 적용한 함수의 나머지 매개변수(Rest Parameter, 레스트 파라미터) 사용법
참고. 나머지 매개변수(...매개변수명)를 사용하면 함수의 인수 개수만큼 정의하지 않고 배열로 처리할 수 있습니다.

• 보기.
```
function fnSum(...args) {    // ...args → 나머지 매개변수
  let sum = 0;
  for (let i=0; i<args.length; i++) sum += args[i];
  return sum;
}

let x = fnSum(4, 9, 16, 25);
```

• 출력결과. 54

3) 활용 예 ② → 펼침 연산자를 사용하여 함수의 인수에 사용하는 변수의 개수를 효율적으로 줄여서 사용할 수 있습니다.

• 보기.
```
const numbers = [23,55,21,87,56];
let maxValue = Math.max(...numbers);
// .max(numbers);를 사용하면 배열 자체를 사용하여
// 최대값을 찾기 때문에 NaN(Not a Number)이 출력됩니다.

console.log(`최대값 : ${maxValue}`);
```

• 출력결과. 최대값 : 87

• 참고. Math.max(숫자1, 숫자2, ...)에서 max() 메서드는 인수에 있는 숫자 리스트 중에서 최대값을 반환합니다.
min() 메서드는 최소값을 반환합니다.

2. for - in 순환 제어문 → 배열의 인덱스 번호를 순환하거나 리터럴 객체의 키(key)를 순환합니다.

구조	인덱스 번호를 사용한 배열 또는 리터럴 객체 참조	

for (let index in array) {
  // 관련 code
}
또는
 	
for (let key in object) {
  // 관련 code
}
	const numbers = [45, 4, 9, 16, 25];
	
	let txt = "";
	for (let idx in numbers) {
	  txt += numbers[idx];
	}
	
	console.log("txt : " + txt);
		const person = {fname:"John", lname:"Doe", age:25};
		
		let txt = "";
		for (let key in person) {
		  txt += person[key];
		}
		
		console.log("txt : " + txt);

• 참고. key의 자료형은 string입니다.
• 주의. for-in 제어문을 사용하지 않고 배열을 참조할 경우 표기법 → 배열변수명[인덱스번호];
for-in 제어문을 사용하지 않고 리터럴 객체를 직접 참조할 경우 표기법(따옴표, 홑따옴표, 백틱 기호에 유의합니다.)
→ 객체명[`키이름`] 또는 객체명.키이름 (O) 보기. person[`fname`], person.fname
객체명[키] 또는 객체명.`키` (X)

3. for - of 순환 제어문 → 배열 또는 문자열, Map, Set, arguments 객체의 값을 순환합니다.

구조	배열 값을 참조	문자열의 값을 참조

for (let value of array) {
  // 관련 code
}
또는
 	
for (let value in object) {
  // 관련 code
}
	const cars = ["BMW", "Volvo", "Mini"];
	
	let txt = "";
	for (let x of cars) {
	  txt += x;
	}
	
	console.log("txt : " + txt);
		let language = "JavaScript";
		
		let txt = "";
		for (let x of language) {
		  txt += x;
		}
		
		console.log("txt : " + txt);

• 리터럴 객체는 for - of 제어문으로 처리할 수 없습니다.

4. forEach() 메서드 → 배열.forEach() 메서드로 배열, 컬렉션 또는 Map객체의 각 항목 값과 인덱스 번호를 참조할 수 있습니다.
리터럴 객체는 forEach() 메서드로 처리할 수 없습니다.

1) 사용법 ① → 배열 요소의 값과 인덱스 번호를 참조할 수 있습니다.

형식1	형식2

배열.forEach(함수명);
 	
function 함수명(v, i) {
 	
  // 매개변수 v는 배열의 각 항목의 값
  // 매개변수 i는 배열의 각 인덱스
}
	배열.forEach((v, i) => {
	
	  // 매개변수 v는 배열의 각 항목의 값
	  // 매개변수 i는 배열의 각 인덱스
	
	});

• 참고. 화살표 함수(Arrow Function) →
함수 표현식을 사용하여 함수를 단순하게 사용한 식을 의미합니다.
화살표 함수
(v, i) => { ... } 와
일반적인 함수
function(v, i) { ... } 는 같습니다.

• 보기.
```
const arr = [5, 12, 33];

arr.forEach((v, i) => {
  console.log(`(val, idx) : (${v}, ${i})`);
});
```

• 출력결과.
```
(val, idx) : (5, 0)
(val, idx) : (12, 1)
(val, idx) : (33, 2)
```

2) 사용법 ② → DOM의 HTML 요소를 참조할 수 있습니다.

형식1	형식2

배열.forEach(함수명);
 	
function 함수명(e, i) {
 	
  // 매개변수 e는 HTML 각각의 요소
  // 매개변수 i는 해당 요소의 인덱스
 	
}
	배열.forEach((e, i) => {
	
	  // 매개변수 e는 HTML 각각의 요소
	  // 매개변수 i는 해당 요소의 인덱스
	
	});

• 참고.
JS에서 선택한 HTML 요소를
DOM Collection(컬렉션)이라고 합니다.

• 보기.

[Markup]

```
<body>

  <span>span1</span>
  <span>span2</span>
  <a>a요소</a>
  <span>span3</span>
  <button>클릭</button>

</body>
```

[Script]

```
document.querySelector("button").addEventListener("click", () => {
        const spanDom = document.querySelectorAll("span");
                        // span 컬렉션
        let val, idx;
        spanDom.forEach((e, i) => {
            val = e.innerText;
            idx = i;
            console.log(`(val, idx) : (${val}, ${idx})`);
        });
});
```

[출력결과]

```
(val, idx) : (span1, 0)
(val, idx) : (span2, 1)
(val, idx) : (span3, 2)
```

5. Quiz. 두 숫자 사이의 누적을 구하는 프로그램을 작성하세요.

초기화면	결과1	결과2
[숫자 2개 입력] 숫자1 [ ] 숫자2 [ ] **결과보기**	[숫자 2개 입력] 숫자1 **5** 숫자2 **10** **결과보기** 5부터 10까지의 누적 합 : 45	[숫자 2개 입력] 숫자1 **7** 숫자2 **3** **결과보기** 3부터 7까지의 누적 합 : 25

6. Quiz. 단위변환 프로그램을 구현하세요. 기재되지 않은 관련 내용은 작업자가 임의 지정합니다.

결과1	결과2
[단위 변환] ◉ cm → inch ○ inch → cm 센티(cm)미터 입력 100 결과보기 100 cm = 39.4 inch	[단위 변환] ○ cm → inch ◉ inch → cm 인치(inch) 입력 10 결과보기 10 inch = 25.4 cm

• 1 인치 = 2.54 센티미터

7. Quiz Drill. 아래의 작업결과를 구현합니다.

- 홈디렉토리 → numberSquare
- HTML 파일명 → /numSquare.html
- JS, CSS 코드 : 내부코드

[작업 내용]
- 3이상 9이하의 숫자를 입력받은 후, 이 숫자를 사용하여 가로 10줄, 세로 10줄을 출력하는 프로그램을 작성하세요.
- [결과보기] 버튼을 클릭했을 때, 공백 유효성 검사를 실시합니다.
- 결과3, 결과4에서 출력되는 [재실행] 버튼을 클릭했을 때, [결과1 - 초기화면]이 출력되어야 합니다.
- 그 외 기재되지 않은 내용은 작업자가 임의 판단하여 구현합니다.

결과1 - 초기화면	결과2 - 입력화면	결과3 - 정상 출력화면	결과4 - 오류메시지
[입력영역] 숫자입력(3~9 사이 값 입력) [ ] 결과보기 [출력영역]	[입력영역] 숫자입력(3~9 사이 값 입력) [6] 결과보기 [출력영역]	[입력영역] 숫자입력(3~9 사이 값 입력) [6] 결과보기 [출력영역] 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 Print OK! 재실행	[입력영역] 숫자입력(3~9 사이 값 입력) [2] 결과보기 [출력영역] 입력 오류! 다시 입력해주세요. (3~9 사이 값 입력) 재실행

- 내용 끝.