자바스크립트 반복문과 조건문

구분	설명
반복문(Loop)	같은 코드 블록을 조건에 따라 여러 번 반복해서 실행하도록 하는 구문입니다. 예를 들어 `for`, `while`, `do...while` 등이 있습니다.
조건문(Conditional Statement)	주어진 조건의 참/거짓에 따라 서로 다른 코드 블록을 실행하도록 하는 구문입니다. 예를 들어 `if`, `else`, `else if`, `switch` 등이 여기에 속합니다.

자바스크립트 반복문

- 반복문 종류

반복문 종류	설명
for	가장 기본적인 반복문으로, 특정 조건이 만족될 때까지 반복 실행
for in	객체의 모든 열거 가능한 속성을 순회
for of	반복 가능한 객체의 요소를 순회 (Array, String, Map, Set 등)
while	주어진 조건이 거짓이 될 때까지 반복해서 실행
do...while	조건을 검사하기 전에 블록 내의 코드를 최소 한 번 실행, 이후 조건이 참인 동안 반복 실행

- 반복문과 같이 사용되는 키워드

키워드	설명
break	반복문(예: `for`, `while`, `do...while`)의 실행을 즉시 중단시키거나, `switch` 문에서 사용되어 특정 케이스 실행 후 빠져나올 때 사용
continue	현재 반복의 나머지 부분을 건너뛰고 반복문의 다음 반복으로 제어를 이동시킴, 주로 `for`, `while`, `do...while` 반복문 내에서 특정 조건에서 추가적인 코드 실행을 건너뛸 때 사용
switch	다중 조건 분기를 처리하는 데 사용되며, 표현식을 평가하여 그 값에 따라 다른 코드 블록을 실행, 반복문과는 직접적인 연관이 없지만 조건에 따른 작업 수행에 유용

```
// for 반복문 예제
for (let i = 0; i < 5; i++) {
  console.log(i); // 0부터 4까지 출력
}

// for...in 반복문 예제
const object = { a: 1, b: 2, c: 3 };
for (const key in object) {
  console.log(key, object[key]); // 객체의 각 속성과 값을 출력
}
```

```
// for...of 반복문 예제
const array = [1, 2, 3, 4, 5];
for (const value of array) {
  console.log(value); // 배열의 각 요소를 출력
}

// while 반복문 예제
let i = 0;
while (i < 5) {
  console.log(i); // 0부터 4까지 출력
  i++;
}

// do...while 반복문 예제
let j = 0;
do {
  console.log(j); // 0부터 4까지 출력, 최소 한 번은 실행
  j++;
} while (j < 5);
```

```
// break 예제
for (let k = 0; k < 10; k++) {
  if (k === 5) break; // i가 5에 도달하면 반복문을 중단
  console.log(k); // 0부터 4까지만 출력
}

// continue 예제
for (let l = 0; l < 10; l++) {
  if (l % 2 === 0) continue; // l이 짝수이면 다음 반복으로 건너뜀
  console.log(l); // 1, 3, 5, 7, 9 출력
}

// switch 예제
const fruit = 'apple';
switch (fruit) {
  case 'apple':
    console.log('Apple!'); // 'Apple!' 출력
    break;
  case 'banana':
    console.log('Banana!');
    break;
  default:
    console.log('Unknown fruit');
}

// switch 를 if 문으로 변환
const fruit = 'apple';

if (fruit === 'apple') {
    console.log('Apple!'); // 'Apple!' 출력
```

```
} else if (fruit === 'banana') {
    console.log('Banana!');
} else {
    console.log('Unknown fruit');
}
```

---

for, for of, for in

1. 배열로 구성된 예제를 반복문으로 원하는 요소 추출

```
const breadArray = ['소금빵', '초코빵', '모카빵'];

for (let i = 0; i < breadArray.length; i++) { // for 반복문
  console.log('for : ' + breadArray[i]); // 각 빵 이름을 출력
}

for (const breadName of breadArray) { // for...of 반복문
  console.log('for of : ' + breadName); // 각 빵 이름을 출력
}

for (const index in breadArray) { // for...in 반복문
  console.log('for in : ' + index + ': ' + breadArray[index]); // 인덱스와 빵
이름을 출력
}
```

2. 객체로 구성된 예제를 반복문으로 원하는 요소 추출

```
const coffeeObject = { americano: '아메리카노', mochalatte: '모카라떼',
espresso: '에스프레소' };

for (const key of Object.keys(coffeeObject)) { // for 반복문 (키 사용)
  console.log('for of Object.keys() : ' + key + ': ' + coffeeObject[key]);
// 각 키와 값을 출력
}

for (const [coffee, coffeeName] of Object.entries(coffeeObject)) {  //
for...of 반복문
  console.log('for of : ' + coffee + ' is ' + coffeeName); // 각 커피와 이름을
출력
}

for (const coffee in coffeeObject) { // for...in 반복문
  console.log('for in : ' + coffee + ': ' + coffeeObject[coffee]); // 커피와
이름을 출력
}
```

객체 메서드 / 구문	설명
`for...in` 반복문	객체의 모든 열거 가능한 키를 순회합니다.
`Object.keys(obj)`	객체의 모든 키를 배열로 반환합니다.
`Object.values(obj)`	객체의 모든 값을 배열로 반환합니다.
`Object.entries(obj)`	키-값 쌍을 배열로 반환합니다.

> - 객체에는 일반적인 for 문을 직접 사용할 수 없습니다.
> 객체의 키, 값, 또는 키-값 쌍을 순회하기 위해서는 for...in 반복문이나 Object.keys(), Object.values(), Object.entries()와 같은 메서드를 사용하여 객체를 배열 형태로 변환한 후 for 반복문을 사용할 수 있습니다.
> 객체에 대한 일반적인 for 반복문을 직접 사용할 수 없는 이유는, for 반복문이 기본적으로 순차적인 순회를 위해 설계되었기 때문입니다. 객체는 순서가 없는 키-값 쌍의 집합으로, 배열과 같이 인덱스 기반의 순차적 접근이 불가능합니다.
> for 반복문은 주로 배열과 같은 인덱스 기반의 반복 가능한 데이터 구조에서 인덱스(0, 1, 2, ...)를 통해 순차적으로 요소에 접근할 때 사용됩니다. 반면, 객체는 키(Key)를 사용해 값(Value)에 접근하는 구조이므로, for 반복문으로는 직접 순회할 수 없습니다.

자바스크립트 조건문

구분	설명
if 문	주어진 조건이 참일 때 특정 코드 블록을 실행하는 조건문입니다.
else 문	if 조건이 거짓일 때 실행할 코드 블록을 지정하는 추가적인 조건문입니다.
else if 문	이전 if 문의 조건이 거짓이면, 새로운 조건을 검사하여 해당하는 경우 특정 코드 블록을 실행하는 조건문입니다.
switch 문	주어진 표현식의 값에 따라 여러 경우 중 하나를 선택하여 실행할 코드 블록을 지정하는 조건문입니다.

- if 문

  자바스크립트에서 if 문은 특정 조건을 검사하여 그 조건이 참(true)인 경우에만 지정된 코드 블록을 실행하는 조건문입니다. if 문은 프로그램의 흐름을 제어하는 데 매우 중요하며, 다양한 상황에 따라 코드를 조건적으로 실행할 수 있게 해줍니다.

  ```
  // 1. 조건이 참일 경우에 실행할 코드 블록을 중괄호 {} 안에 넣고 else 블록을 추가하여 조건이 거짓(false)일 때 실행할 코드를 지정할 수 있습니다.

  if (조건) {
    // 조건이 참일 때 실행할 코드
  } else {
  ```

```
  // 조건이 거짓일 때 실행할 코드
}

// 2. if 문은 더 복잡한 조건을 검사하기 위해 else if 블록을 여러 개 추가할 수도 있습니다.

if (조건1) {
  // 조건1이 참일 때 실행할 코드
} else if (조건2) {
  // 조건1이 거짓이고, 조건2가 참일 때 실행할 코드
} else {
  // 모든 조건이 거짓일 때 실행할 코드
}
```

- if, else, else if, wsitch 예제

```
// if 문 예제
let fruits = ['apple', 'banana', 'orange'];
if (fruits.includes('banana')) {
  console.log("바나나가 포함되어 있습니다.");
}

// else 문 예제
let weather = {
  temperature: 28,
  isSunny: true
};
if (weather.isSunny) {
  console.log("오늘 날씨는 맑습니다.");
} else {
  console.log("오늘은 흐린 날씨입니다.");
}

// else if 문 예제
let age = 15;
if (age >= 19) {
  console.log("당신은 성인입니다.");
} else if (age >= 13) {
  console.log("당신은 청소년입니다.");
} else {
  console.log("당신은 어린이입니다.");
}

// switch 문 예제
let dayOfWeek = 'Monday';
switch (dayOfWeek) {
  case 'Monday':
  case 'Tuesday':
  case 'Wednesday':
  case 'Thursday':
  case 'Friday':
    console.log("평일입니다.");
```

```
    break;
  case 'Saturday':
    console.log("토요일입니다.");
    break;
  case 'Sunday':
    console.log("일요일입니다.");
    break;
  default:
    console.log("유효하지 않은 요일입니다.");
}
```