CHAPTER 04 반복문

자바스크립트 프로그래밍 입문 (2판)

Contents

학습목표

- 배열을 생성하고 사용하는 방법을 익힙니다.
- while 반복문과 for 반복문을 이해합니다.
- for in 반복문과 for of 반복문을 이해합니다.
- break 키워드와 continue 키워드를 이해합니다.

Contents

내용

- 배열
- while 반복문
- for 반복문
- 역 for 반복문
- for in 반복문과 for of 반복문
- 중첩 반복문
- break 키워드
- continue 키워드
- 조금 더 나아가기

0. 반복문

- 붙여 넣기를 사용한 반복 - 1000번 출력하는 것은 무리임

코드 4-1 복사해서 붙여 넣기를 사용한 반복

```
console.log("출력");
console.log("출력");
console.log("출력");
console.log("출력");
console.log("출력");
```

- 반복문을 사용한 반복

코드 4-2 반복문을 사용한 반복

```
for (let i = 0; i < 1000; i++) {
    console.log("출력");
}
```

1. 배열

배열 생성 방법

- 여러 개의 자료를 한꺼번에 다룰 수 있는 자료형
- 대괄호 내부의 각 자료는 쉼표로 구분
- 배열에는 여러 자료형이 섞여 있을 수 있음

let 이름 = [자료, 자료, 자료, 자료, 자료]

그림 4-1 배열 선언 형태

```
> let array = [52, 273, '아침밥', '점심밥', true, false]
undefined
> array
[ 52, 273, '아침밥', '점심밥', true, false ]
```

1. 배열

배열의 요소와 인덱스

- 요소(Element)

  배열 안에 들어 있는 각 자료
  배열의 요소에 접근할 때는 대괄호를 사용

- 인덱스(Index): 대괄호 안에 넣는 숫자

그림 4-2 배열의 요소

1. 배열

- [예제 4-1] 배열 생성하고 요소에 접근
  - 인덱스의 시작 숫자는 0 임에 주의

코드 4-3 배열 생성하고 요소에 접근 arrayBasic.js

```
// 배열을 생성합니다.
let array = [52, 273, '아침밥', '점심밥', true, false]

// 배열의 요소를 변경합니다.
array[0] = 0

// 요소를 출력합니다.
console.log(array[0]);
console.log(array[1]);
console.log(array[2]);
console.log(array[3]);
console.log(array[4]);
```

실행 결과

```
0
273
아침밥
점심밥
true
```

2. while 반복문

- 기본 형태

```
while (불_표현식) {
    // 불 표현식이 참인 동안 실행할 문장
}
```

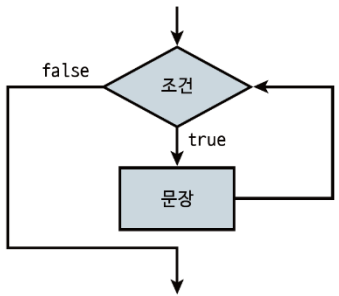

그림 4-3 while 반복문

코드 4-5 무한 반복문

```
while (true) {
    console.log("무한 반복");
}
```

2. while 반복문

- [예제 4-2] while 반복문 이용
  - 특정한 숫자를 증가시켜 불 표현식을 false로 만들어 반복문을 벗어남

코드 4-6 while 반복문 whileLoop.js

```
// 변수를 선언합니다.
let i = 0;
let array = [52, 273, 32, 65, 103];

// 반복을 수행합니다.
while (i < array.length) {
    // 출력합니다.
    console.log(i + "번째 출력:" + array[i]);

    // 반복문을 탈출하기 위해 변수를 더합니다.
    i++;
}
```

실행 결과

```
0번째 출력:52
1번째 출력:273
2번째 출력:32
3번째 출력:65
4번째 출력:103
```

3. for 반복문

- for 반복문의 각 단계
  - ❶ 초기식을 비교
  - ❷ 조건식을 비교
    조건이 false이면 반복문을
  - ❸ 문장을 실행
  - ❹ 종결식을 실행
  - ❺ 2단계로 이동

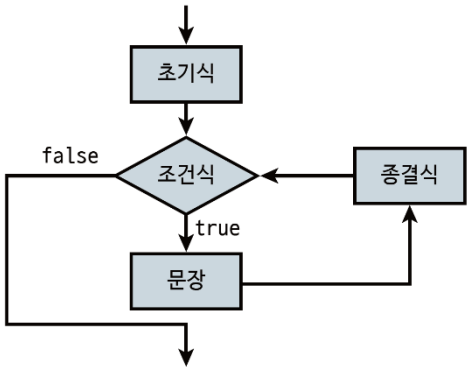

그림 4-4 for 반복문

- 기본 형태

```
for (let i = 0; i < 반복_횟수; i++) {

}
```

3. for 반복문

- [예제 4-3] for 반복문을 이용한 덧셈
  - 0부터 100까지 더하기

코드 4-7 for 반복문으로 덧셈하기 forLoopExampleA.js

```
// 변수를 선언합니다.
let output = 0;

// 반복을 수행합니다.
for (let i = 0; i <= 100; i++) {
    output += i;
}

// 출력합니다.
console.log(output);
```

실행 결과

```
5050
```

3. for 반복문

- [예제 4-4] for 반복문을 이용한 곱셈
  - 1부터 20까지 곱셈하기

코드 4-8 for 반복문으로 곱셈하기 forLoopExampleB.js

```
// 변수를 선언합니다.
let output = 1;

// 반복을 수행합니다.
for (let i = 1; i <= 20; i++) {
    output *= i;
}

// 출력합니다.
console.log(output);
```

초깃값을 0으로 놓으면 무엇을 곱해도 0이 됩니다. 따라서 이번에는 1로 설정합니다.

실행 결과

```
2432902008176640000
```

4. 역 for 반복문

```
for (let i = length - 1; i >= 0; i--) {

}
```

- [예제 4-5] 역 for 반복문 - 배열의 요소를 뒤쪽부터 출력

코드 4-10 역 for 반복문 forReverse.js

```
// 배열을 생성합니다.
let array = [1, 2, 3, 4, 5, 6];

// 요소의 길이를 출력합니다.
for (let i = array.length - 1; i >= 0; i--) {
    console.log(array[i]);
}
```

실행 결과

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

5. for in 반복문과 for of 반복문

- 객체에 쉽게 반복문을 적용함
- for in 반복문과 for of 반복문은 for 반복문 사용과 역할이 같음

```
for(let 인덱스 in 배열) {

}
```

```
for(let 요소 of 배열) {

}
```

```
for (let i = 0; i < 배열.길이; i++) {
    let 인덱스 = i;
    let 요소 = 배열[i];
}
```

5. for in 반복문과 for of 반복문

- [예제 4-6] for in 반복문과 for of 반복문

코드 4-11 for in 반복문과 for of 반복문 forInOfLoop.js

```
// 변수를 선언합니다.
let array = ["사과", "배", "포도", "딸기", "바나나"];

// 반복을 수행합니다.
for (let i in array) {
    // 출력합니다.
    console.log(`${i}번째 요소: ${array[i]}`);
}

console.log("----- 구분선 -----");

// 반복을 수행합니다.
for (let item of array) {
    // 출력합니다.
    console.log(item);
}
```

실행 결과

```
0번째 요소: 사과
1번째 요소: 배
2번째 요소: 포도
3번째 요소: 딸기
4번째 요소: 바나나
----- 구분선 -----
사과
배
포도
딸기
바나나
```

6. 중첩 반복문

- 반복문을 여러 번 중첩해서 사용
- [예제 4-7] 별 피라미드(1)

코드 4-12 별 피라미드 (1) pyramidA.js

```
let output = '';

for (let i = 0; i < 10; i++) {
    for (let j = 0; j < i + 1; j++) {
        output += '*';
    }
    output += '\n';
}

console.log(output);
```

실행 결과

```
*
**
***
****
*****
******
*******
********
*********
**********
```

6. 중첩 반복문

- [예제 4-8] 별 피라미드(2)

코드 4-13 별 피라미드 (2) pyramidB.js

```
let output = '';

for (let i = 0; i < 10; i++) {
    for (let j = 0; j < 10 - i; j++) {
        output += ' ';
    }
    for (let j = 0; j < i + 1; j++) {
        output += '*';
    }
    output += '\n';
}

console.log(output);
```

실행 결과

```
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

7. break 키워드

- 반복문을 벗어날 때 사용
  - 무한 반복문은 내부에서 break 키워드를 사용해야 벗어날 수 있음

```
while (true) {

}
```

7. break 키워드

- [예제 4-9] break 키워드
  - 짝수를 찾으면 break 키워드로 반복문을 벗어남

코드 4-14 break 키워드 break.js

```
let i = 0;
let array = [1, 31, 273, 57, 8, 11, 32];
let output;

while (true) {
    if (array[i] % 2 == 0) {
        output = array[i];
        break;
    }

    i = i + 1;
}

console.log(`처음 발견한 짝수는 ${output}입니다.`)
```

실행 결과

```
처음 발견한 짝수는 8입니다.
```

8. continue 키워드

- 반복문 내부에서 현재 반복을 멈추고 다음 반복을 진행함
- [예제 4-10] continue 키워드
  - 변수 i가 짝수일 때 continue 키워드로 현재 반복을 멈추고 다음 반복을 진행함. 따라서 코드를 실행하면 홀수만 출력

코드 4-15 continue 키워드 continue.js

```
for (let i = 1; i < 10; i++) {
    if (i % 2 == 0) {
        continue;
    }

    console.log(i)
}
```

짝수라면 다음 반복으로 바로 넘어갑니다.
따라서 이 다음 코드는 실행되지 않습니다.

실행 결과

```
1
3
5
7
9
```

8. continue 키워드

- [예제 4-10]을 간략하게 코드 변경

코드 4-16 [예제 4-10]의 변경 형태

```
for (let i = 1; i < 10; i++) {
    if (i % 2 != 0) {
        console.log(i);
    }
}
```

9. 조금 더 나아가기

스코프 Scope

- 변수를 사용할 수 있는 범위
- 스코프 == 블록

블록

- 중괄호로 둘러싸는 부분

```
if (표현식) {
}
```
블록

```
for(let i = 0; i < 10; i++) {
}
```
블록

```
for(let item of array) {
}
```
블록

9. 조금 더 나아가기

- 블록 내부에 선언된 변수는 해당 변수 내부에서만 사용가능

코드 4-17 괄호를 사용한 스코프

```
{
    let a = 10;
}

console.log(a);
```

- 반복문에 활용된 변수는 해당 블록에 있으므로 외부에서 활용할 수 없음

코드 4-18 반복문으로 만든 스코프

```
for (let i = 0; i < 3; i++) {
    console.log(i);
}

console.log(i);
```

변수 i는 해당 블록 외부에서 사용할 수 없습니다.

9. 조금 더 나아가기

- 스코프 내부에서 이름 중복

코드 4-19 스코프 내부에서 이름 중복

```
let a = 1;
let b = 1;
{
    let a = 2;
    {
        let a = 3;
        console.log(a);
    }
    console.log(a);
    console.log(b);
}
```

상위 블록에 있는 변수를 사용합니다.

같은 블록에 있는 변수를 사용합니다.

같은 블록에 있는 변수를 사용합니다.

같은 블록에 있는 변수를 사용합니다.

9. 조금 더 나아가기

호이스팅Hoisting

- 해당 블록에서 사용할 변수를 미리 확인해서 정리하는 작업
  - 아래 코드는 에러 발생

코드 4-20 호이스팅 문제

```
let a = 1;
{
    console.log(a);
    let a = 2;
}
```


THANK
YOU!