# 8장 제어문

제어문은 조건에 따라 코드 블록을 실행(조건문)하거나 반복 실행(반복문)할 때 사용한다.

일반적으로 코드는 위에서 아래 방향으로 순차적으로 실행된다.

하지만 코드의 실행 순서가 변경된다는 것은 단순히 위에서 아래로 순차적으로 진행하는 직관적인 코드의 흐름을 혼란스럽게 만든다.

제어문은 코드의 흐름을 이해하기 어렵게 만들어 가독성을 해치는 단점이 있다.

나중에 살펴볼 forEach, map, filter, reduce 같은 고차 함수를 사용한 함수형 프로그래밍 기법에서는 제어문의 사용을 억제하여 복잡성을 해결하려고 노력한다.

# 블록문

블록문은 0개 이상의 문을 중괄호로 묶은 것으로, 코드 블록 또는 블록이라고 부르기도 한다.

자바스크립트는 블록문을 하나의 실행 단위로 취급한다. 블록문은 단독으로 사용할 수 있으나 일반적으로 제어문이나 함수를 정의할 때 사용하는 것이 일반적이다.

# 조건문

조건문은 주어진 조건식의 평가 결과에 따라 코드 블록(블록문)의 실행을 결정한다.

조건식은 불리언 값으로 평가될 수 있는 표현식이다.

## if-else 문

if-else 문은 주어진 조건식(불리언 값으로 평가될 수 있는 표현식)의 결과, 즉 논리적 참 또는 거짓에 따라 실행할 코드 블록을 결정한다.

```
if (조건식) {
    // 참일 때
} else {
    // 거짓일 때
}
```

if문의 조건식은 **불리언 값으로 평가**되어야 한다. 만약 if 문의 조건식이 불리언 값이 아닌 값으로 평가되면 **자바스크립트 엔진에 의해 암묵적으로 불리언 값으로 강제 변환**되어 실행할 코드 블록을 결정한다.

```
if (조건식1) {
    // 참일 때
} else if (조건식2) {
    // 거짓일 때
} else {
    // 조건식1과 조건식2가 모두 거짓일 때
}
```

else if문과 else 문은 옵션이다. 즉, 사용할 수도 있고 사용하지 않을 수도 있다.

if문과 else 문은 2번 이상 사용할 수 있지만, else if 문은 여러 번 사용할 수 있다.

만약 코드 블록 내의 문이 하나 뿐이라면 중괄호를 생략할 수 있다.

```
var num = 2;
var kind;

if (num > 0) kind = "양수";
```

```
else if (num < 0) kind = "음수";
else kind = "영";
```

대부분의 if-else 문은 삼항 연산자로 바꿔 쓸 수 있다.

```
var x = 2;

var result = x % 2 ? '홀수' : '짝수';
console.log(result); // 짝수
```

위 예제는 두 가지 경우의 수('홀수' 또는 '짝수')를 갖는 경우이다. 만약 경우의 수가 세 가지라면 다음과 같이 바꿔 쓸 수 있다.

```
var num = 2;

var kind = num ? (num > 0 ? '양수' : '음수') : '영';

console.log(kind); // 양수
```

`num > 0 ? '양수' : '음수'` 는 표현식이다. 즉, 삼항 연산자는 값으로 평가되는 표현식을 만든다. 그러므로 삼항 연산자 표현식은 값처럼 사용할 수 있기 때문에 변수에 할당할 수 있다.

하지만 if-else 문은 표현식이 아닌 문이다. 따라서 if-else 문은 값처럼 사용할 수 없기 때문에 변수에 할당할 수 없다.

## switch문

switch 문은 주어진 표현식을 평가하여 그 값과 일치하는 표현식을 갖는 case 문으로 실행 흐름을 옮긴다.

```
switch(표현식) {
    case 표현식1:
        switch 문의 표현식과 표현식1이 일치하면 실행될 문;
        break;
    case 표현식2:
        switch 문의 표현식과 표현식2가 일치하면 실행될 문;
        break;
    default:
        switch 문의 표현식과 일치하는 case 문이 없을 때 실행될 문;
}
```

if-else 문의 조건식은 불리언 값으로 평가되어야 하지만 **switch 문의 표현식은 불리언 값보다는 문자열이나 숫자 값인 경우가 많다.**

```
var month = 11;
var monthName;

switch (month) {
    case 1: monthName = 'January';
    case 2: monthName = 'February';
    case 3: monthName = 'March';
    case 4: monthName = 'April';
    case 5: monthName = 'May';
    default: monthName = 'Invaild month';
}
```

switch 문의 표현식은 불리언 값보다는 문자열이나 숫자 값인 경우가 많다.

다시 말해, if-else 문은 논리적 참, 거짓으로 실행할 코드 블록을 결정한다. switch 문은 논리적 참, 거짓보다는 다양한 상황(case)에 따라 실행할 코드 블록을 결정할 때 사용한다.

switch 문의 표현식, 즉 month 변수의 평가 결과인 숫자 값 11과 일치하는 case 문으로 실행 흐름이 이동한다.

하지만 위 예제를 실행해보면 'November'가 출력되지 않고 'Invalid month'가 출력된다. 이는 switch 문의 표현식의 평가 결과와 일치하는 case 문으로 실행 흐름이 이동하여 문을 실행한 것은 맞지만 문을 실행한 후 switch 문을 탈출하지 않고 switch 문이 끝날 때까지 이후의 모든 case 문과 default 문을 실행했기 때문이다.

이를 **폴스루(fall through)**라고 한다. 다시 말해 monthName 변수에 'November'가 할당된 후 switch 문을 탈출하지 않고 연이어 'December'가 재할당되고 마지막으로 **'Invalid month'**가 재할당된 것이다.

default 문에는 break 문을 생략하는 것이 일반적이다. default 문은 switch 문의 맨 마지막에 위치하므로 default 문의 실행이 종료되면 switch 문을 빠져나간다.

따라서 별도로 break 문이 필요 없다.

```
var year = 2000;
var month = 2;
var days = 0;

switch(month){
    case1 : case 3: case 5: case 7: case 8: case 10: case 12:
        days = 31;
        break;
    case 4: case 6: case 9: case 11:
        days = 30;
        break;
    case 2:
        days = ((year % 4 === 0 && year % 100 !== 0) || (year %
```

```
        break;
    default:
        console.log('Invalid month');
}
```

switch 문은 case, default, break 등 다양한 키워드를 사용해야 하고 폴스루가 발생하는 등 문법도 복잡하다. 따라서 C 언어를 기반으로 하는 프로그래밍 언어는 대부분 switch 문을 지원하지만 파이썬과 같이 switch 문을 지원하지 않는 프로그래밍 언어도 있다.

# 반복문

반복문은 조건식의 평가 결과가 참인 경우 코드 블록을 실행한다.

그 후 조건식을 다시 평가하여 여전히 참인 경우 코드 블록을 다시 실행한다. 이는 조건식이 거짓일 때까지 반복된다.

### 반복문을 대체할 수 있는 다양한 기능

자바스크립트는 배열을 순회할 때 사용하는 forEach 메소드, 객체의 프로퍼티를 열거할 때 사용하는 for-in 문, ES6에서 도입된 이터러블을 순회할 수 있는 for-of 문가 같이 반복문을 대체할 수 있는 다양한 기능을 제공한다.

이에 대해서는 해당 장에서 자세히 살펴보자.

# for문

for문은 조건식이 거짓으로 평가될 때까지 코드 블록을 반복 실행한다. 가장 일반적으로 사용되는 for 문의 형태는 다음과 같다.

```
for (변수 선언문 또는 할당문; 조건식; 증감식) {
    조건식이 참인 경우 반복 실행될 문;
}
```

```
for (var i = 0; i<2; i++){
    console.log(i);
}
```

### for문의 실행 순서

1. for문을 실행하면 맨 먼저 변수 선언문 var i = 0이 실행된다. 변수 선언문은 단 한 번만 실행된다.
2. 변수 선언문의 실행이 종료되면 조건식이 실행된다. 현재 i 변수의 값은 0이므로 조건식의 평가 결과는 true다.
3. 조건식의 평가 결과가 true 이므로 코드 블록이 실행된다. 증감문으로 실행 흐름이 이동하는 것이 아니라 코드 블록으로 실행 흐름이 이동하는 것에 주의하자.
4. 코드 블록의 실행이 종료되면 증감식 i++가 실행되어 i 변수의 값은 1이 된다.
5. 증감식 실행이 종료되면 다시 조건식이 실행된다. 변수 선언문이 실행되는 것이 아니라 조건식이 실행된다는 점에 주의하자 (앞에서 설명했지만 변수 선언문은 단 한 번만 실행된다.) 현재 i 변수의 값은 1이므로 조건식의 평가 결과는 true이다.
6. 조건식의 평가 결과가 true이므로 코드 블록이 다시 실행된다.
7. 코드 블록의 실행이 종료되면 증감식 i++가 실행되어 i 변수의 값은 2가 된다.
8. 증감식 실행이 종료되면 다시 조건식이 실행된다. 현재 i 변수의 값은 2이므로 조건식의 평가 결과는 false다. 조건식의 평가 결과가 false이므로 for 문의 실행이 종료된다.

## while 문

while 문은 주어진 조건식의 평가 결과가 참이면 코드 블록을 계속해서 반복 실행한다.

for 문은 반복 횟수가 명확할 때 주로 사용하고 whlie 문은 반복 횟수가 불명확할 때 주로 사용한다.

```
var count = 0;

while (count < 3){
    console.log(count); // 0 1 2
    count++;
}
```

조건식의 평가 결과가 언제나 참이면 무한루프가 된다.

```
var count = 0;

while (true) {
    console.log(count);
    if (count === 3) break;
}
```

## do-while 문

do-while 문은 코드 블록을 먼저 실행하고 조건식을 평가한다. **따라서 코드 블록은 무조건 한 번 이상 실행된다.**

# break 문

switch 문과 while 문에서 살펴보았듯이 break 문은 코드 블록을 탈출한다.

좀 더 정확히 표현하자면 코드 블록을 탈출하는 것이 아니라 레이블 문, 반복문(for, for-in, for-of, while,do-while) 또는 switch 문의 코드 블록을 탈출한다.

break로 탈출할 수 있는 문은 다음과 같다.

- 레이블문 : 식별자가 붙은 문
  - `foo : console.log('foo');`
- 반복문
- switch문

```
if (true) {
    break;
}

foo : console.log('foo');

foo : {
    console.log(1);
    break foo;
    console.log(2);
}

console.log('Done!');
```

중첩된 for 문의 내부 for 문에서 break 문을 실행하면 내부 for 문을 탈출하여 외부 for 문으로 진입한다.

이때 내부 for 문이 아닌 외부 for 문을 탈출하면 레이블 문을 사용한다.

```
outer: for (var i=0; i<3; i++){
    for (var j=0; j<3; j++){
        if(i+j === 3) break outer;
        console.log(`inner [${i} ${j}`);
    }
}
```

```
console.log('Done!');
```

레이블 문은 중첩된 for 문 외부로 탈출할 때 유용하지만 그 밖의 경우에는 일반적으로 권장하지 않는다.

break 문은 레이블 문 뿐만 아니라 반복문, switch 문에서도 사용할 수 있다. 이 경우에는 break 문에 레이블 식별자를 지정하지 않는다. break 문은 반복문을 더 이상 진행하지 않아도 될 때 불필요한 반복을 회피할 수 있어 유용하다.

```
var string = 'Hello World';
var search = 'l';
var index;

for (var i = 0; i< string.length; i++){
    if (string[i].equals(search)){
        index = i;
        break;
    }
}

console.log(index);

console.log(string.indexOf(search));
```

# continue 문

continue 문은 반복문의 코드 블록 실행을 현 지점에서 중단하고 반복문의 증감식으로 실행 흐름을 이동시킨다. break 문처럼 반복문을 탈출하지는 않는다.

```
var string = 'Hello world';
var search = 'l';
var count = 0;

for (var i=0; i<string.length; i++){
    if(string[i] !== search) continue;
    count++;
}

console.log(count);
```

위와 같은 동작을 하는 코드를 아래와 같이 간단하게 작성할 수 있다.

```
const regexp = new RegExp(search, 'g');
console.log(string.match(regexp).length);
```

**만약 if문 내에서 실행해야 할 코드가 한 줄이라면 continue 문을 사용했을 때, 보다 간편하고 가독성도 좋다. 하지만 if 문 내에서 실행해야 할 코드가 길다면 들여쓰기가 한 단계 더 깊어지므로 continue 문을 사용하는 편이 가독성이 더 좋다.**