

기본문법 활용하기

조건문, 반복문을 구성하는 블록 `{ }` 안에 다른 문법 표현이 포함될 수 있다. 이러한 규칙으로 다양한 문법 중첩 패턴이 생기는데 이 중에서 자주 사용되는 패턴으로는 `if-if`, `if-for`, `for-if`, `for-for`가 있습니다.

#01. 변수의 유효성 범위 (변수의 스코프)

1) `var` 키워드로 선언된 변수의 경우

블록 `{ }` 안에서 선언된 변수는 블록의 실행 여부에 따라 블록 밖에서의 식별 여부가 결정됨.

블록 `{ }`을 갖는 부분이 실행되지 않을 경우 블록 안에서 선언된 변수를 블록 밖에서 사용할 경우 할당되지 않은 `undefined`가 됨

2) `let` 키워드로 선언된 변수와 `const` 키워드로 선언된 상수의 경우

블록 밖에서 선언된 변수는 블록 안으로 침투할 수 있지만 블록 안에서 선언된 변수는 블록을 빠져나올 수 없음.

변수의 범위가 블록안으로 한정되므로 서로 다른 블록끼리는 중복 선언 가능

3) `for`문의 초기식에 대한 유효성 범위

`var` 키워드를 사용한 경우에는 초기식에서 선언된 변수가 `for`문 밖에서 식별 가능함.

`let` 키워드를 사용한 경우에는 초기식에서 선언된 변수는 `for`문 밖에서 식별 할 수 없음.

`var` 키워드는
실행당
혹 선언이 가능.

#02. `if`문 안에 포함된 흐름제어

1) `if-if`구조

특정 조건이 참으로 판단되어 블록안에 진입했을 때 상세조건을 판별하는 구조.

```
if (조건) {  
  if (조건) {  
    ...  
  }  
} else if (조건) {  
  if (조건) {  
    ...  
  } else {  
    ...  
  }  
} else {  
  if (조건) {  
    ...  
  } else if (조건) {  
    ...  
  } else {  
    ...  
  }  
}
```

`if () { }` \Rightarrow 먼저만들고, 그안에 내용 넣기.

```

    ...
  }
}

```

2) if-for 구조

특정 조건이 참으로 판단되어 블록안에 진입했을 때, 반복을 수행하는 구조.

```

if (조건) {
    for (초기식; 조건식; 증감식) {
        ...
    }
}

```

#03. for문 안에서의 흐름제어

1) 반복문 안에서의 조건문

반복문 안에서 매 반복 수행시마다 조건을 판별한다.

주로 반복문에 사용되는 조건값(=초기식에서 생성한 변수)에 대한 조건 판별을 위해 사용된다.

2) For문 안의 For문

바깥의 반복문(부모)이 1회 수행할 때 마다 안쪽의 반복문(자식)이 매번 처음부터 새로 시작하는 이중 반복문 구조.

두 반복문간의 조건값이 서로 달라야 한다.

3) 반복범위 동적설정

자식 반복문의 조건식이 부모 반복문의 조건변수를 활용하여 구성되면 자식 반복문의 반복 범위에 변화를 줄 수 있다.

4) 반복문 안에서의 흐름 제어

무한루프

절대로 조건식이 종료되지 않는 형태의 반복문.

let y=0;

```

while (true) {           // 무조건 반복. 종료되지 않는다 (무한루프)
    ...
}

```

반복문 제어하기

반복을 몇 번 수행해야 하는지 판단할 수 없는 경우 무한루프 형태로 지정하고 특정 조건이 충족되는지에 따라 반복의 중단 여부를 결정한다.

반복문의 흐름제어 기능을 갖는 키워드

- `continue`: 조건식으로 강제 이동 (for문의 경우는 증감식으로 이동함.)
- `break`: 현재 반복문을 강제로 종료하고 블록을 빠져 나간다.