

# 8. 제어문

## ▼ 제어문이란?

조건에 따라 코드 블록을 실행하거나 반복실행한다.  
식이 아니다.

## ▼ 블록문

0개 이상의 문을 중괄호로 묶은 것.  
ex)


```
{  
  var foo = 10;  
}  
  
function sum(a, b) {  
  return a + b;  
}
```

## ▼ 조건문

주어진 조건'식'의 평가 결과에 따라 코드 블록문의 실행을 결정.

if...else

switch

 Node.js 혹은 Next.js에서 서버 관련 코드 작성시 유의점.

switch문 내의 변수들은 모두 같은 블록 범위 내에 있다.

```
switch (req.method) {  
  case "GET":  
    const res = await fetch('url');  
    const data = res.json();  
    res.status(200).json(data);  
    break;  
  case "POST":
```

```

// 에러 발생. 같은 블록 범위내에 res 재선언 재할당 금지.
    const res = await fetch('url', {
      method: 'POST',
      body: req.body,
    });
    const data = res.json();
    res.status(200).json(data);
    break;
    // 생략...
  }

  if (req.method === "GET") {
    const res = await fetch('url');
    const data = res.json();
    res.status(200).json(data);
  } else if (req.method === "POST") {
    // 다른 블록 범위이므로 에러 발생 x
    const res = await fetch('url', {
      method: 'POST',
      body: req.body,
    });
    const data = res.json();
    res.status(200).json(data);
  }
}

```

#### ▼ fall through?

일치하는 case문에서 코드 블록이 실행되나, break;가 없을 경우 switch 문의 모든 case문과 default문이 실행되는 현상

해결 → 각 case문에 break; 명시

#### ▼ break 문

코드 블록을 탈출 할 때 사용.

불필요한 반복을 멈출 때 사용.

🔴 .forEach, .map같은 배열 메서드에서 break사용 불가 → 무시됨.

사용할 경우 for 문으로 변형 혹은 .every 혹은 .some 사용.

추가 → 일부러 error를 발생시켜 제어할 수 있음 그러나 잘 안씀.

```
try {  
  [1, 2, 3].forEach(a => {  
    if (a === 2) {  
      throw new Error('value');  
    }  
  });  
} catch(e) {  
  console.log(e.message) // value 출력  
}
```