

ECMAScript 6

조건문과 반복문

조건문

조건문의 종류와 특징

□ if statement

- 조건 표현식의 결과값을 Boolean 타입으로 변환 후 참/거짓을 판단

□ switch statement

- 조건 표현식의 결과값이 어느 값(case)에 해당하는지 판별
- (참고*) 주로 특정 변수의 값에 따라 조건을 분기할 때 활용
 - 조건이 많아질 경우 if문보다 가독성이 나을 수 있음

| 조건문 (if statement)

```
if (condition) {  
    // do something  
} else if (condition) {  
    // do something  
} else {  
    // do something  
}
```

□ If, else if, else

- 조건은 소괄호(condition) 안에 작성
- 실행할 코드는 중괄호{} 안에 작성
- 블록 스코프 생성

| 조건문 (if statement) 예시

```
const nation = 'Korea'

if (nation === 'Korea') {
  console.log('안녕하세요!')
} else if (nation === 'France') {
  console.log('Bonjour!')
} else {
  console.log('Hello!')
}
```

조건문 (switch statement)

```
switch(expression) {  
  case 'first value': {  
    // do something  
    [break]  
  }  
  case 'second value': {  
    // do something  
    [break]  
  }  
  [default: {  
    // do something  
  }]  
}
```

□ switch

- 표현식(expression)의 결과값을 이용한 조건문
- 표현식의 결과값과 case문의 오른쪽 값을 비교
- break 및 default문은 [선택적]으로 사용 가능
- break문이 없는 경우 break문을 만나거나 default문을 실행할 때까지 다음 조건문 실행
- 블록 스코프 생성

switch 예시 (1) – break가 있는 경우

```
const nation = 'Korea'

switch(nation) {
  case 'Korea': {
    console.log('안녕하세요!')
    break
  }
  case 'France': {
    console.log('Bonjour!')
    break
  }
  default: {
    console.log('Hello!')
  }
}
```

switch 예시 (2) – break가 없는 경우

Fall-through

```
const nation = 'Korea'

switch(nation) {
  case 'Korea': {
    console.log('안녕하세요!')
  }
  case 'France': {
    console.log('Bonjour!')
  }
  default: {
    console.log('Hello!')
  }
}
```


(참고*) If vs. switch

```
const numOne = 5
const numTwo = 10
let operator = '+'

if (operator == '+') {
  console.log(numOne + numTwo)
} else if (operator == '-') {
  console.log(numOne - numTwo)
} else if (operator == '*') {
  console.log(numOne * numTwo)
} else if (operator == '/') {
  console.log(numOne / numTwo)
} else {
  console.log('유효하지 않은 연산자입니다.')
}
```

```
const numOne = 5
const numTwo = 10
let operator = '+'

switch(operator) {
  case '+': {
    console.log(numOne + numTwo)
    break
  }
  case '-': {
    console.log(numOne - numTwo)
    break
  }
  case '*': {
    console.log(numOne * numTwo)
    break
  }
  case '/': {
    console.log(numOne / numTwo)
    break
  }
  default: {
    console.log('유효하지 않은 연산자입니다.')
  }
}
```

| 조건문 실습

- 목표: 자바스크립트의 If 문과 switch 문 연습 (conditions.js)
- 문제: 파일에 작성된 주석 참고

반복문

| 반복문의 종류와 특징

- while
- for
- for... in
 - 주로 객체(object)의 속성들을 순회할 때 사용
 - 배열도 순회 가능하지만 인덱스 순으로 순회한다는 보장이 없으므로 권장하지 않음
- for... of
 - 반복 가능한(iterable)* 객체를 순회하며 값을 꺼낼 때 사용
 - 반복 가능한(iterable) 객체*의 종류: Array, Map, Set, String 등

| 반복문 (while)

```
while (condition) {  
    // do something  
}
```

□ while

- 조건문이 참(true)인 동안 반복 시행
- 조건은 소괄호(condition) 안에 작성
- 실행할 코드는 중괄호{} 안에 작성
- 블록 스코프 생성

| 반복문 (while) 예시

```
let i = 0

while (i < 6) {
  console.log(i) // 0, 1, 2, 3, 4, 5
  i += 1
}
```

| 반복문 (for)

```
for (initialization; condition; expression) {  
    // do something  
}
```

□ for

- 세미콜론(;)으로 구분되는 세 부분 으로 구성
- initialization
 - 최초 반복문 진입시 1회만 실행되는 부분
- condition
 - 매 반복 시행 전 평가되는 부분
- expression
 - 매 반복 시행 이후 평가되는 부분
- 블록 스코프 생성

반복문 (for) 예시

```
for (let i = 0; i < 6; i++) {  
  console.log(i) // 0, 1, 2, 3, 4, 5  
}
```

// 1. 반복문 진입 및 변수 i 선언

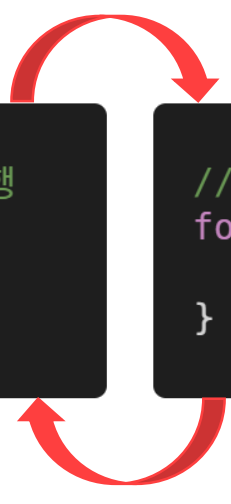
```
for (let i = 0; i < 6; i++) {  
  console.log(i)  
}
```

// 2. 조건문 평가 후 코드 블록 실행

```
for (let i = 0; i < 6; i++) {  
  console.log(i) // 0  
}
```

// 3. 코드 블록 실행 이후 i 값 증가

```
for (let i = 0; i < 6; i++) {  
  console.log(i) // 0  
}
```



| 반복문 실습

- 목표: 자바스크립트 반복문 while, for 연습 (03-loops.js)
- 문제: 파일에 작성된 주석 참고

| 반복문 (for... in)

```
for (variable in object) {  
    // do something  
}
```

□ for... in

- 객체(object)의 속성들을 순회할 때 사용
- 배열도 순회 가능하지만 **권장하지 않음**
- 실행할 코드는 중괄호 안에 작성
- 블록 스코프 생성

| 반복문 (for... in) 예시

```
const capitals = {  
  Korea: '서울',  
  France: '파리',  
  USA: '워싱턴 D.C.'  
}  
  
for (let capital in capitals) {  
  console.log(capital) // Korea, France, USA  
}
```

| 반복문 (for... of)

```
for (variable of iterables) {  
    // do something  
}
```

□ for... of

- 반복 가능한(iterable) 객체를 순회하며 값을 꺼낼 때 사용
- 실행할 코드는 중괄호 안에 작성
- 블록 스코프 생성

| 반복문 (for... of) 예시

```
const fruits = ['딸기', '바나나', '메론']  
  
for (let fruit of fruits) {  
    console.log(fruit) // 딸기, 바나나, 메론  
}
```

(참고) for... in vs. for... of

for ... in (객체 순회 적합)

```
// array
const fruits = ['딸기', '바나나', '메론']

for (let fruit in fruits) {
  console.log(fruit) // 0, 1, 2
}

// object
const capitals = {
  Korea: '서울',
  France: '파리',
  USA: '워싱턴 D.C.'
}

for (let capital in capitals) {
  console.log(capital) // Korea, France, USA
}
```

for ... of (배열 원소 순회)

```
// array
const fruits = ['딸기', '바나나', '메론']

for (let fruit of fruits) {
  console.log(fruit) // 딸기, 바나나, 메론
}

// object
const capitals = {
  Korea: '서울',
  France: '파리',
  USA: '워싱턴 D.C.'
}

for (let capital of capitals) {
  console.log(capital)
  // Uncaught TypeError: capitals is not iterable
}
```

| 반복문 실습

- 목표: 자바스크립트 반복문 for... in, for... of 연습 (03-loops.js)
- 문제: 파일에 작성된 주석 참고

조건문과 반복문 Quiz

Q1. while문은 break문을 필수적으로 작성해야 한다. T/F

Q2. for... of 구문은 객체의 속성값 순회에 유용하다. T/F

Q3. for... in 구문은 반복 가능한 객체의 순회에 사용된다. T/F

조건문과 반복문 Quiz

Q1. while문은 break문을 필수적으로 작성해야 한다.

F

A1. while문에서 break문은 선택적으로 작성할 수 있다.

Q2. for... of 구문은 객체의 속성값 순회에 유용하다.

F

A2. 객체의 속성값 순회에 유용한 구문은 for... in에 해당한다.

Q3. for... in 구문은 반복 가능한 객체의 순회에 사용된다.

F

A3. 객체의 순회에 사용되는 구문은 for... of 에 해당한다.

조건문과 반복문 정리

키워드	종류	연관 키워드	스cope
if	조건문	-	블록 스코프
switch	조건문	case, break, default	블록 스코프
while	반복문	break, continue	블록 스코프
for	반복문	break, continue	블록 스코프
for... in	반복문	객체 순회	블록 스코프
for... of	반복문	배열 등 Iterable 순회	블록 스코프