AhnLab SEM

2022. 3. 23 (chapter. 03)

- 자바스크립트 기본 -

인선미

3. 조건문과 반복문

1) 조건문
2) 다중 분기문
3) 배열
4) 반복문

조건문

조건문

if 조건문

- 조건문은 프로그램의 흐름 변경 시 사용(조건 분기)
- 가장 일반적인 조건문
- 표현식의 값이 true면 중괄호 안의 문장을 실행하고 false면 문장을 무시
- 비교 연산자와 논리 연산자를 활용해 조건을 만들고 이 조건에 따라 조건 분기

조건문

생각해 봅시다

- 다음의 결과는 참(true)일까요 거짓(false)일까요

```
const x = 10
x < 100
```

```
const x = 10
5 < x && x < 15
```

```
const x = 10
5 < x || x < 15
```

조건문의 형식

if문과 if~else문

```
if(조건){
  조건의 결과가 true일 때 실행할 명령;
}
```

```
if(조건){
  조건의 결과가 true일 때 실행할 명령;
}else{
  조건의 결과가 false일 때 실행할 명령;
}
```

미션> 입력된 숫자가 3의 배수인지 판별

프로그램 설계

- 사용자에게 숫자를 입력받는다
- 입력된 숫자가 3의 배수인지 확인한다
- 3의 배수일 경우 “3의 배수입니다” 출력
- 아닐 경우 “3의 배수가 아닙니다” 출력

조건문의 형식

중첩 조건문

- 조건문 안에 조건문을 중첩해서 사용하는 것

```
if(조건1){
  if(조건2){
    조건1과 조건2가 참일때 실행할 명령;
  }else{
    조건1은 참, 조건2는 거짓일때 실행할 명령;
  }
}else{
  조건의 결과가 false일 때 실행할 명령;
}
```

미션> 입력된 숫자가 3의 배수인지 판별2

프로그램 설계

- 사용자에게 숫자를 입력받는다
- 입력된 숫자가 3의 배수인지 확인한다
- 3의 배수일 경우 “3의 배수입니다” 출력
- 아닐 경우 “3의 배수가 아닙니다” 출력
- 사용자가 취소 버튼을 눌렀을 때의 경우를 처리한다.

잠깐!

삼항연산자

```
(조건)? True일때실행명령 : false일때실행명령;
```

잠깐!

삼항연산자

```
let userNumber = prompt("숫자를 입력하세요:");
console.log(userNumber);
```

조건문의 형식

if else if 조건문

- 중첩 조건문에서 중괄호를 생략한 형태

```
if(조건1){
  조건1의 결과가 참일 때 실행할 명령;
}else if(조건2){
  조건2의 결과가 참일 때 실행할 명령;
}else if(조건3){
  조건3의 결과가 참일 때 실행할 명령;
}else{
  거짓일 때 실행할 명령;
}
```

미션> 현재 시각에 따라 오늘의 일정을 알려주는 프로그램

현재 시각을 구하는 방법

- 현재 날짜와 시간을 갖는 객체 생성
- 현재 시간을 0~23사이의 값으로 출력하는 메소드 사용

```
const date = new Date()
const hour = date.getHours()
```

미션> 현재 시각에 따라 오늘의 일정을 알려주는 프로그램

프로그램 설계

- 현재 시각이 11시 미만이면 “아침 먹을 시간”
- 현재 시각이 오후 3시 미만이면 “점심 먹을 시간”
- 아니면 “저녁 먹을 시간” 등으로 출력하기

도전>

현재가 몇 월인지 확인하고 계절을 알려주는 프로그램 만들기

- const mth = Number(prompt(“몇 월인지 입력해주세요” , “”));

다중 분기문

다중 조건문

switch 조건문

- 특정 값의 조건을 비교할 때 사용

```
Switch(조건){
  case 값1: 명령1;
    break;
 case 값2: 명령2;
    break;
 case 값3: 명령3;
    break;
 default: 명령n;
}
```

미션> 태어난 연도를 입력 받아 띠 출력하는 프로그램

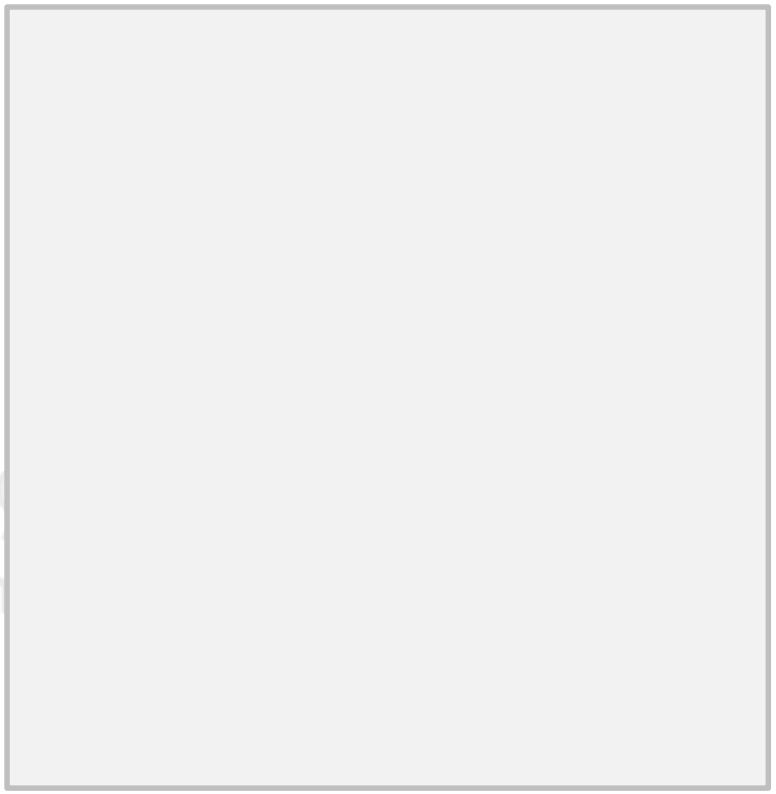
프로그램 설계

- 태어난 연도를 입력 받는다.
- 연도에 따라 띠를 출력해준다.

미션> 태어난 연도를 입력 받아 띠 출력하는 프로그램

```
const rawInput = prompt('태어난 해를 입력해주세요.', '');
const year = Number(rawInput);
const e = year % 12;

let result;
```



```
alert(`${year}년에 태어났다면 ${result} 띠입니다.`);
```

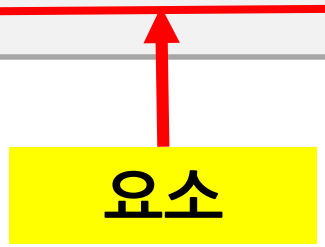
배열

배열(array)

복합 자료형 - 배열

- 여러 자료를 묶어서 활용할 수 있는 특수한 자료
- 여러 개의 자료를 저장하기 위해 변수를 여러 개 선언해야 하는 불편함 해소
- 대괄호[...] 사용해 생성, 내부의 값은 쉼표로 구분

```
let seasons = [‘봄’, ‘여름’, ‘가음’, ‘겨울’];
```



배열(array)

배열

```
> const str = '안녕하세요';
<- undefined
> str[2];
<- '하'
> str[str.length-1]
<- '요'
```

배열(array)

배열

```
> const array = [256, "문자열", true, function(){ }, {}, [273, 111]]
< undefined
> array
< ▼(6) [256, '문자열', true, ƒ, {…}, Array(2)]
    0: 256
    1: "문자열"
    2: true
  ▶ 3: ƒ ()
  ▶ 4: {}
  ▶ 5: (2) [273, 111]
    length: 6
  ▶ [[Prototype]]: Array(0)
```

배열(array)

배열 요소에 접근하기

- 배열 이름 뒤에 대괄호 사용
- 인덱스 index를 이용하여 접근
- 인덱스는 0부터 시작

```
> const fruits = ['사과', '배', '바나나', '망고', '딸기'];
<· undefined
> fruits[0]
<· '사과'
> fruits[3]
<· '망고'
```

배열(array)

배열 요소의 갯수

- 배열 내부에 들어있는 요소의 개수
- length 속성 사용

```
> const fruits = ['사과', '배', '바나나', '망고', '딸기'];
<· undefined
> fruits.length
<· 5
> fruits[fruits.length-1]
<· '딸기'
```

배열(array)

배열 뒷부분에 요소 추가하기

- push() 메소드 사용
- 배열 뒷부분에 요소를 추가
- 배열.push(요소);

```
> const fruits = ['사과', '배', '바나나', '망고', '딸기'];
<· undefined
> fruits
<· ▶(5) ['사과', '배', '바나나', '망고', '딸기']
> fruits.push('수박');
<· 6
> fruits
<· ▶(6) ['사과', '배', '바나나', '망고', '딸기', '수박']
```

배열(array)

배열 뒷부분에 요소 추가하기

- 인덱스를 사용해 비어있는 뒷부분에 요소 추가하기

```
> const fruits = ['사과', '배', '바나나', '망고'];
<- undefined
> fruits[7] = '두리안';
<- '두리안'
> fruits
<- ▶ (8) ['사과', '배', '바나나', '망고', empty × 3, '두리안']
```

배열(array)

배열 요소 제거하기

- 인덱스를 이용하여 제거하기 splice() 메소드
- 값을 이용하여 제거하기

배열(array)

배열 요소 제거하기

- 인덱스를 이용하여 제거하기 splice() 메소드
- 배열.splice(인덱스, 제거할 요소의 개수);

```
> const fruits = ['사과', '배', '바나나', '망고', '딸기'];
< undefined
> fruits.splice(2, 2);
< ▶ (2) ['바나나', '망고']
> fruits
< ▶ (3) ['사과', '배', '딸기']
```

배열(array)

배열 요소 제거하기

- 값으로 요소 제거하기
- indexOf()메소드 사용해서 값의 위치를 추출 후 splice() 메소드 사용
- const 인덱스 = 배열.indexOf(요소);
- 배열.splice(인덱스, 1);
- indexOf() 메소드는 배열 내부에 요소가 있을 경우 인덱스를 리턴
- 배열 내부에 요소가 없을 때는 -1을 리턴

배열(array)

배열 요소 제거하기

- 값으로 요소 제거하기

```
> const fruits = ['사과', '배', '바나나', '망고', '딸기'];
< undefined
> const index = fruits.indexOf('바나나');
< undefined
> index
< 2
> fruits.splice(index, 1);
< ▶ ['바나나']
> fruits
< ▶ (4) ['사과', '배', '망고', '딸기']
> fruits.indexOf('바나나');
< -1
```

배열(array)

배열의 특정 위치에 요소 추가하기

- splice()메소드의 2번째 매개변수에 0을 입력하면 아무것도 제거하지 않음
- 배열.splice(인덱스, 0, 요소);

```
> fruits
◀ ▶ (4) ['사과', '배', '망고', '딸기']
> fruits.splice(1, 0, "포도");
◀ ▶ []
> fruits
◀ ▶ (5) ['사과', '포도', '배', '망고', '딸기']
```

반복문

반복문

반복문의 의미

- 명령어를 반복적으로 수행하고자 할 때
- for in 반복문, for of 반복문, for 반복문, while 반복문, break, continue

미션> 반복문 없이 1부터 5까지 더하기

반복문 없이 더해보기

- 숫자 1부터 5까지 더해서
- 더한 결과를 화면에 출력해 보세요

반복문

for 문 사용하기

- 가장 많이 사용하는 반복문
- **값이 일정하게 커지면서 명령을 반복하여 실행할 때 사용**

```
for(초기값; 조건; 증가식){
   실행할 명령;
}
```

반복문

for 문 사용하기

```
for(초기값; 조건; 증가식){
   실행할 명령;
}
```

- 초기값; 카운터 변수 초기화. 초기값은 0이나 1부터 시작
- 조건; 명령을 반복하기 위한 조건을 확인. 조건을 만족해야 그 다음에 오는 명령 실행
- 증가식; 명령을 반복한 후 실행. 보통 카운터 변수를 1만큼 증가

미션> 1부터 5까지 더하기

for 반복문 이용하여 더하기

- 숫자 1부터 5까지 더해서
- 더한 결과를 화면에 출력해 보세요

미션> 1부터 1000까지 더하기

for 반복문 이용하여 더하기

- 숫자 1부터 1000까지 더해서
- 더한 결과를 화면에 출력해 보세요

for반복문

for 반복문과 배열

- 배열의 length 속성만큼 반복을 돌리는 형태로 사용

```
<script>
  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']

  for (let i = 0; i < todos.length; i++) {
    console.log(`${i}번째 할 일: ${todos[i]}`)
  }
</script>
```

for반복문

for 반복문과 배열

- 반대로 출력해보기

```
<script>
  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']

  for (let i = todos.length - 1; i >= 0; i--) {
    console.log(`${i}번째 할 일: ${todos[i]}`)
  }
</script>
```

미션> 구구단 출력하기

for 반복문의 활용

- 1단부터 9단까지 구구단을 화면에 출력하는 코드를 작성해 주세요

감사합니다.