|| && 논리연산자

- 여러가지 조건을 결합할 때 쓴다.

&& 는 둘 다 일치할때만 실행된다. (AND)

```
const myName = "로미오";
const yourName = "줄리엣";
if (myName === "로미오" && yourName === "줄리엣") {
  console.log("&& 실행");
}
```

|| 는 둘 중 하나라도 일치할때 실행된다. (OR)

```
const myName = "로미오";
const yourName = "앨리스";
if (myName === "로미오" || yourName === "줄리엣") {
  console.log("|| 실행");
}
```

? : 삼항연산자

- if else 조건문을 짧게 쓴 것

```
const myName = "로미오";
const yourName = "줄리엣";

myName === "로미오" || yourName === "줄리엣" ? console.log("||실행") : console.log("error!");
```

조건 / true일 경우 / false일 경우

만약 조건에 따른 결과가 짧을 경우 삼항연산자를,
길 경우 if else 를 쓰면 된다.

배열(array)

배열

- 여러 개의 data를 대괄호 [ ] 를 사용해 하나로 묶은 것

배열을 쓰지 않을 경우

```
const mon = "월";
const twe = "화";
const wed = "수";
const thu = "목";
const fri = "금";
const sat = "토";
const sun = "일";

console.log(mon, twe, wed, thu, fri, sat, sun);
```

이렇게 일일히 적는건 비효율적.
왜? 7가지 모두 "요일" 이라는 공통된 특징이 있기 때문

배열을 사용하면?

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

console.log(daysOfWeek);
```

- 여러 개의 data를 대괄호 [ ] 를 사용해 하나로 묶은 것

요소(element) 와 인덱스(index)

- 요소(element) 배열을 이루는 각각의 data
- 인덱스(index) 각 요소에 부여된 번호

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

console.log(daysOfWeek[0]); // 월
console.log(daysOfWeek[1]); // 화
console.log(daysOfWeek[6]); // 일
console.log(daysOfWeek[7]); // undefined
```

정의되지 않았다는 뜻
우리는 8번째 데이터를 정의한 적 없다.

undefined vs null

- 둘 다 조건문에서 false로 인식함
- undefined는 고의가 아님
- null은 고의

왜 일부러 null을 쓸까?
개발자가 의도적으로 비워두고 싶은 경우가 있기 때문

length

- 배열의 길이 구할 때 사용

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

console.log(daysOfWeek.length); // 7
```

- 문자열의 길이도 구할 수 있다

```
const name1 = "조교행님";
console.log(name1.length); // 4
```

배열의 요소는 const로 선언되었더라도 변경 가능하다

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

daysOfWeek[2] = "ㅋㅋㅋ";

console.log(daysOfWeek);
```

push( )

- 배열에 요소를 추가할 때 사용

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

daysOfWeek.push("야호");

console.log(daysOfWeek); // 맨 뒤에 "야호" 추가된 배열 출력됨
```

그러나, const로 선언된 배열을 다시 선언할 순 없다.

```
const daysOfWeek = ["월", "화", "수", "목", "금", "토", "일"];

daysOfWeek = ["짜장면", "짬뽕", "탕수육"];  // error
```

왜? const 이기 때문.

이차원 배열

- 배열 안에 배열도 가능하다.

```
const matrix = ["김치찌개", "햄버거", ["짜장면", "짬뽕", "탕수육"]];

console.log(matrix[2][1]);  // 짬뽕
```

객체(object)

JavaScript에서 가장 중요한 두 가지

- 객체(object)
- 함수(function)

- 이 둘은 JavaScript 의 정체성

주의

- JavaScript 의 객체(object)는
- C, C++ C#, Java, Python 의 객체와 완전히 다르다.

- JavaScript 에선 클래스를 만들지 않고도 객체를 만든다.

객체(object)

- 키(key) 와 값(value) 로 이루어진 프로퍼티(property) 모음
- 가장 비슷한 개념은 Python 의 딕셔너리

내 개인정보를 배열로 저장했다고 가정하자.

```
const myInfo = [
    "조교행님",
    28,
    true,
    [
        "아빠",
        "엄마",
        "멍멍이"
    ]
];
```

참고로, JavaScript는 독특하게도
서로 다른 타입끼리도 배열을 만들 수 있다.

이 경우, 두가지 점에서 불편함
1. 뭐가 뭔지 "예상" 을 해야.
2. 인덱스를 전부 다 기억해야.

해결책: 각각의 요소에 이름을 달자!

```
const myInfo = {
    name: "조교행님",
    age: 28,
    isGrilfriend: true,
    family: [
        "아빠",
        "엄마",
        "멍멍이"
    ]
};
```

1. 뭐가 뭔지 "예상" 을 해야.
   => 뭐가 뭔지 알려줌
2. 인덱스를 전부 다 기억해야.
   => 그럴 필요 없음

여기서,

앞에 있는게 key (따옴표 안 씀. 영어만 허용)
뒤에 있는게 value
둘을 합쳐서 property (프로퍼티)
여러 개면 , (콤마) 로 구분

객체에서의 데이터 접근

- 객체는 인덱스가 없고, key 를 통해 접근한다.

```
const myInfo = {
    name: "조교행님",
    age: 28,
    isGrilfriend: true,
    family: [
        "아빠",
        "엄마",
        "멍멍이"
    ]
};

console.log(myInfo.family[0]); // 아빠
```

마침표 . 사용함

인덱스가 없다는 건, 순서가 따로 없다는 뜻

물론 잘못된 key 를 입력하면 undefined

console.log( )

console 객체 안에 있는 log 함수

객체의 프로퍼티는 const로 선언되었더라도 변경 가능하다

```
const myInfo = {
    name: "조교행님",
    age: 28,
    isGrilfriend: true,
    family: [
        "아빠",
        "엄마",
        "멍멍이"
    ]
};

myInfo.isGrilfriend = false;
console.log(myInfo);
```

그러나, const로 선언된 객체를 다시 선언할 순 없다.

```
const myInfo = {
    name: "조교행님",
    age: 28,
    isGrilfriend: true,
    family: [
        "아빠",
        "엄마",
        "멍멍이"
    ]
};

myInfo = {
    name: "Jony",
    age: 16
}
```

객체가 왜 중요한가?

- 프론트엔드(클라이언트)와 백엔드(서버)는
- 객체를 주고받아 통신한다.
- JSON

객체가 배열보다 나은가?

- 이것은 자료구조(data structure) 일 뿐이다.
- 개발자는 상황에 따라, 둘 중 알맞은 타입을 선택해야

빈 배열 [ ] 과 빈 객체 { } 는 false 아님

```
const a = {};  // 또는 []
if(a) {
    console.log(true);
} else {
    console.log(false);
}
```

결과: true

같아보이는 배열과 객체는 비교 불가

```
const a = [1, 2, 3];
const b = [1, 2, 3];


if (a === b) {
    console.log("같다");
} else {
    console.log("다르다");
}
```

```
const c = {
    myName: "조교행님",
    age: 28
}
const d = {
    myName: "조교행님",
    age: 28
}

if (c === d) {
    console.log("같다");
} else {
    console.log("다르다");
}
```

배열, 객체의 비교연산은
JavaScript 에서 제공하지 않는다.

만약 이런 기능을 원한다면
lodash 를 배워보길 추천함

직접 해보기

- 나의 정보를 객체로 만들라.
  - 이름, 나이, 직업, 학과, 사는곳
  - 좋아하는 것 (3개 이상의 배열)
- 객체와 객체 안의 배열 요소에 접근해 원하는대로 변경해보라.