자바스크립트 기본

if와 '?'를 사용한 조건 처리

- 다음 값은 모두 false로 평가 (falsy value)
- https://developer.mozilla.org/en-US/docs/Glossary/Falsy
    - 숫자 0
    - 빈 문자열 ("")
    - null
    - undefined
    - NaN
        - null과 undefined의 경우 유용하게 사용되는 케이스 있음 (유효한 값이 없을 경우 처리할 로직을 처리하는케이스)
- 반대(그 외 참 값으로 평가되는 값)는 truthy value
- 조건부 연산자는 물음표(?)로 표시
    - **피연산자가 세 개**이기 때문에 조건부 연산자를 삼항(ternary) 연산자라고도 칭함
- 평가 대상인 condition이 truthy라면 콜론 기호 기준으로 왼쪽값(value1)이, 그렇지 않으면 오른쪽값(value2)가 반환

```
let result = condition ? value1 : value2;
```

- 가독성에 따라 적절히 if 구문이나 삼항 연산자를 선택하여 사용

논리 연산자

- 단락 평가(short circuit evaluation)
-     - OR은 왼쪽부터 시작해서 오른쪽으로 평가를 진행하는데, truthy를 만나면 **나머지 값들은 건드리지 않은 채 평가를 멈춤**
        - OR는 하나만 true이면 true를 반환해도 되므로 뒤의 식을 평가할 필요가 없기 때문!
- AND 연산자(&&)도 OR 연산자와 비슷한 순서로 동작
    - 단, 평가된 값이 false인 경우 평가를 멈춤 (OR과 반대)
        - AND는 하나만 false이면 false를 반환해도 되므로 뒤의 식을 평가할 필요가 없기 때문!
- NOT을 두 개 연달아 사용(!!)하면 값을 불린형으로 변환
    - Boolean 함수를 써서 변환하는 것보다 짧게 코드 작성 가능

```
// truthy value => true
alert( !!"non-empty string" ); // true
// falsy value => false
alert( !!null ); // false
```

while과 for 반복문

while 반복문

- 기본 문법

```
while (condition) {
  // 코드
  // '반복문 본문(body)'이라 불림
}
```

- 활용 사례

```
let i = 0;
// 0, 1, 2가 출력
while (i < 3) {
  alert( i );
  i++;
}
```

- do - while 문 사용 가능

```
do {
  // 반복문 본문
} while (condition);
```

- 활용 사례

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

for 반복문

- 기본 문법

```
for (begin; condition; step) {
  // ... 반복문 본문 ...
}
```

- 활용 사례

```
// 0, 1, 2가 출력
for (let i = 0; i < 3; i++) {
  alert(i);
}
```

구성 요소

begin	**i = 0**	**반복문에 진입할 때 단 한 번 실행됩니다. (초기식)**
condition	i < 3	반복마다 해당 조건이 확인됩니다. false이면 반복문을 멈춥니다. (조건식)
body	alert(i)	condition이 truthy일 동안 계속해서 실행됩니다.
step	i++	각 반복의 body가 실행된 이후에 실행됩니다. (증감식)

- 변수를 반복문 안(초기식)에서 선언할 경우 **인라인 변수 선언**이라고 부르며 이렇게 선언한 변수는 반복문 안에서만 접근 가능, 만약 반복문 바깥에서도 변수에 접근하고 싶다면 미리 변수를 선언하면 됨

```
// 반복문 바깥에 변수 정의
let i = 0;

// 기존에 정의된 변수(i)를 사용하여 증감
for (i = 0; i < 3; i++) {
  alert(i);
}

// 3, 반복문 밖에서 선언한 변수이므로 사용할 수 있음
alert(i);
```

- 초기식, 조건식, 증감식 생략 가능

```
// 끊임 없이 본문이 실행 (무한루프)
for (;;) {
    // 본문
}
```

- **break 구문**을 이용하여 반복문 빠져나오기 가능
- **continue 구문**을 이용하여 조건을 만족할 경우 특정 코드를 생략하고 다시 반복을 진행하도록 설정 가능

switch문

- 복수의 if 조건문을 switch문으로 대체 가능

```
switch(x) {
    case 'value1':  // if (x === 'value1')
        ...
        [break]
    case 'value2':  // if (x === 'value2')
        ...
        [break]
    default:
        ...
        [break]
}
```

- 변수 x의 값과 첫 번째 case문의 값 'value1'를 일치 비교(===)한 후, 두 번째 case문의 값 'value2'와 비교하며, 이런 과정을 계속 반복함
- case문에서 변수 x의 값과 일치하는 값을 찾으면 해당 case문의 아래의 코드가 실행되며 break문을 만나면 swtich문 탈출, (만약 break문이 없다면 계속해서 인접한 case문 아래의 코드를 실행)
- **default문**이 있는 경우, 값과 일치하는 case문이 없다면 default문 아래의 코드가 실행

```
let a = 2 + 2;

switch (a) {
    case 3:
        alert( '비교하려는 값보다 작습니다.' );
        break;
    case 4:
        alert( '비교하려는 값과 일치합니다.' );
        break;
    case 5:
        alert( '비교하려는 값보다 큽니다.' );
        break;
    default:
        alert( "어떤 값인지 파악이 되지 않습니다." );
}
```

- 아래 코드의 경우 break문이 없으므로 case 4이후의 모든 case 내부 코드가 실행됨

```
let a = 2 + 2;

switch (a) {
    case 3:
        alert( '비교하려는 값보다 작습니다.' );
    // 이 시점에 a와 값이 일치하는데, break문이 없으므로 case 4, 5, default 아래 코드 모두 실행
    case 4:
        alert( '비교하려는 값과 일치합니다.' );
    case 5:
        alert( '비교하려는 값보다 큽니다.' );
    default:
        alert( "어떤 값인지 파악이 되지 않습니다." );
}
```

- switch 문 오른쪽 괄호와 case 오른쪽에 **표현식 사용 가능**

```
let a = "1";
let b = 0;

// switch 문 오른쪽 괄호에 표현식 사용 가능
switch (+a) {
    // case 오른쪽에 표현식 사용 가능
    case b + 1:
        alert("표현식 +a는 1, 표현식 b+1는 1이므로 이 코드가 실행됩니다.");
        break;
    default:
        alert("이 코드는 실행되지 않습니다.");
}
```

여러 개의 case문 묶기

- switch/case문에서 break문이 없는 경우엔 **조건에 상관없이 다음 case문이 실행**되므로 주의!

```
let a = 3;

switch (a) {
    case 4:
        alert('계산이 맞습니다!');
        break;
    // (*) 두 case문을 묶음 (즉, a가 3이거나 5이면 case 5 아래 코드 실행)
    // 즉, case 3과 case 5에서 실행하려는 코드가 같은 경우
    case 3:
    case 5:
        alert('계산이 틀립니다!');
        alert("수학 수업을 다시 들어보는걸 권유 드립니다.");
        break;
    default:
        alert('계산 결과가 이상하네요.');
}
```