7주차 JavaScript

상태	승인
설명	기본함수, 연산자,if

alert

`alert` 함수는 앞선 예제에서 살펴본 바 있다. 이 함수가 실행되면 사용자가 '확인(OK)' 버튼을 누를 때까지 메시지를 보여주는 창이 계속 떠있게 된다.

```
alert("Hello");
```

모달 ❓

메시지가 있는 작은 창을 모달 창이라고 부르며, 페이지의 나머지 부분과는 상호작용이 불가능하다는 의미가 숨겨 져 있다. 즉, 모달창을 띄우면 확인 버튼이나 취소버튼을 누르기 전까지는 웹페이지에서 다른 행동이 불가능하게 된다.

prompt

브라우저에서 제공하는 `prompt` 함수는 두 개의 인수를 받으며 함수가 실행되면 텍스트 메시지와 입력 필드(input field), 확인(OK) 및 취소(Cancel) 버튼이 있는 모달 창을 띄워준다.

```
result = prompt(title, [default]);
```

`title`

사용자에게 보여줄 문자열`default`입력 필드의 초깃값(선택값)

> 💡 `deafult` 에 대괄호 `[…]` 로 감싸져 있는 이유 ❓
>
> prompt 함수는 속성을 제어할 수 있는 값을 2개를 넣을 수가 있는데, 2번째 속성의 경우 필수적으로 넣어주지 않아도 되는 값이기 때문이다.

`prompt` 함수는 사용자가 입력 필드에 기재한 문자열을 반환합니다. 사용자가 입력을 취소한 경우는 `null`이 반환된다.

```
let age = prompt('나이를 입력해주세요.', 100);

alert(`당신의 나이는 ${age}살 입니다.`); // 당신의 나이는 100살입니다.
```

confirm

```
result = confirm(question);
```

`confirm` 함수는 매개변수로 받은 `question(질문)`과 확인 및 취소 버튼이 있는 모달 창을 보여준다.

사용자가 확인 버튼을 누르면 `true`, 그 외의 경우는 `false`를 반환한다.

```
let isBoss = confirm("당신이 주인인가요?");

alert( isBoss ); // 확인 버튼을 눌렀다면 true가 출력됩니다.
```

문제 - 간단한 페이지 만들기

사용자에게 이름을 물어보고, 입력받은 이름을 그대로 출력해주는 페이지를 만들어 보세요.

정답

```
<!DOCTYPE html>
<html>
<body>

  <script>

    let name = prompt("이름을 입력해 주세요.", "");
    alert(name);
  </script>

</body>
</html>
```

기본 연산자와 수학

덧셈 `+`, 곱셈 `*`, 뺄셈 `-`과 같은 연산은 학교에서 배워서 이미 알고 있을 것이다. 이러한 기본적인 연산자 외에도 JS만의 연산자에 대해서 알아보자.

피연산자(operand)

연산자가 연산을 수행하는 대상을 말한다. `5 *2` 에는 왼쪽 피연산자 `5`와 오른쪽 피연산자 `2`, 총 두 개의 피연산자가 있으며 피연산자는 `인수`라고 불리기도 한다.

- 피연산자를 하나만 받는 연산자는 ***단항(unary)*** *연산자* 라고 부른다.

```
let x = 1;

x = -x;alert( x ); // -1, 단항 마이너스 연산자는 부호를 뒤집습니다.
```

- 두 개의 피연산자를 받는 연산자는 ***이항(binary)*** *연산자* 라고 부른다.

```
let x = 1, y = 3;
alert( y - x ); // 2, 이항 마이너스 연산자는 뺄셈을 해줍니다.
```

수학

자바스크립트에서 지원하는 수학 연산자는 아래와 같다.

- 덧셈 연산자 `+`,
- 뺄셈 연산자 `-`,
- 곱셈 연산자 `*`,
- 나눗셈 연산자 `/`,

- 나머지 연산자 `%`,
- 거듭제곱 연산자 `*`

나머지 연산자 `%`

나머지 연산자를 사용한 표현식 `a % b`는 `a`를 `b`로 나눈 후 그 `나머지`를 정수로 반환해준다.

```
alert( 5 % 2 ); // 5를 2로 나눈 후의 나머지인 1을 출력
alert( 8 % 3 ); // 8을 3으로 나눈 후의 나머지인 2를 출력
```

거듭제곱 연산자 `**`

`a ** b`를 계산하면 `a`를 `b`번 곱한 값이 반환된다.

예시:

```
alert( 2 ** 2 ); // 4  (2 * 2)
alert( 2 ** 3 ); // 8  (2 * 2 * 2)
alert( 2 ** 4 ); // 16 (2 * 2 * 2 * 2)
```

거듭제곱 연산자는 정수가 아닌 숫자에 대해서도 동작한다 즉, `1/2`을 사용하면 제곱근을 구할 수 있다.

```
alert( 4 ** (1/2) ); // 2 (1/2 거듭제곱은 제곱근)
alert( 8 ** (1/3) ); // 2 (1/3 거듭제곱은 세제곱근)
```

이항 연산자 '+'와 문자열 연결

`+`의 피연산자로 문자열이 전달되면 덧셈 연산자는 덧셈이 아닌 문자열을 합치는 역할을 한다.

```
let s = "my" + "string";
alert(s); // mystring
```

만약 `정수형`과 `문자열`을 더하면 그 결과는 문자열로 나오게 된다.

```
alert( '1' + 2 ); // "12"
alert( 2 + '1' ); // "21"
```

```
alert(2 + 2 + '1' ); // '221'이 아니라 '41'이 출력됩니다.
```

연산은 왼쪽에서 오른쪽으로 순차적으로 진행되기 때문에 위와 같은 결과가 나오게 된다.

`-`과 나눗셈 `/` 연산자는 `+`와는 달리 연산자들을 모두 `정수형`으로 바꿔준다.

```
alert( 6 - '2' ); // 4, '2'를 숫자로 바꾼 후 연산이 진행됩니다.
alert( '6' / '2' ); // 3, 두 피연산자가 숫자로 바뀐 후 연산이 진행됩니다.
```

연산자 우선순위

하나의 표현식에 둘 이상의 연산자가 있는 경우, 실행 순서는 연산자의 *우선순위*에 의해 결정된다. 만약 우선순위를 바꾸고 싶다면, 수학에서와 똑같이 `()` 를 활용하면 된다.

순위	연산자 이름	기호
…	…	…
16	지수	`**`
15	곱셈	`*`
15	나눗셈	`/`
13	덧셈	`+`
13	뺄셈	`-`
…	…	…
3	할당	`=`

…	…	…

할당 연산자

무언가를 할당할 때 쓰이는 `=`도 연산자이며 우선순위가 `3`으로 아주 낮을걸 확인할 수 있다.

`x = 2 * 2 + 1`과 같은 표현식에서 계산이 먼저 이뤄지고, 그 결과가 `x`에 할당되는 이유가 바로 이 이유 때문이다.

```
let x = 2 * 2 + 1;

alert( x ); // 5
```

복합 할당 연산자

프로그램을 짜다 보면, 변수에 연산자를 적용하고 그 결과를 같은 변수에 저장해야 하는 경우가 종종 생긴

```
let n = 2;
n = n + 5;
n = n * 2;
```

이때, `+=`와 `*=`연산자를 사용하면 짧은 문법으로 동일한 연산을 수행할 수 있다.

```
let n = 2;
n += 5; // n은 7이 됩니다(n = n + 5와 동일).
n *= 2; // n은 14가 됩니다(n = n * 2와 동일).

alert( n ); // 14
```

복합연산자는 거의 모든 연산자에 사용이 가능하다.

```
let n = 2;

n *= 3 + 5;

alert( n ); // 16  (*=의 우측이 먼저 평가되므로, 위 식은 n *= 8과 동일합니다.)
```

증가/감소 연산자

숫자를 하나 늘리거나 줄이는 것은 프로그래밍에서 굉장히 자주 사용되며, 이를 위한 연산자가 존재한다.

1️⃣ 증가(increment) 연산자 `++`는 변수를 1 증가시킨다.

```
let counter = 2;
counter++;      // counter = counter + 1과 동일하게 동작합니다. 하지만 식은 더 짧습니다.
alert( counter ); // 3
```

2️⃣ 감소(decrement) 연산자 `-`는 변수를 1 감소시킨다.

```
let counter = 2;
counter--;      // counter = counter - 1과 동일하게 동작합니다. 하지만 식은 더 짧습니다.
alert( counter ); // 1
```

증가·감소 연산자는 변수에만 쓸 수 있으며 `5++`와 같이 값에 사용하려고 하면 에러가 발생하게 된다.

`++`와 `--` 연산자는 변수 앞이나 뒤에 올 수 있다.

- `counter++`와 같이 피연산자 뒤에 올 때는, `'후위형(postfix form)'`이라고 부른다.
- `++counter`와 같이 피연산자 앞에 올 때는, `'전위형(prefix form)'`이라고 부른다.

후위형과 전위형은 피연산자인 `counter`를 `1`만큼 증가시켜 준다는 점에서 동일한 일을 하는데 순서가 다르다. 아래 예제를 보면서 이해해보자.

```
let counter = 1;
let a = ++counter; // (*)

alert(a); // 2
alert(counter); //2
```

이제 후위형을 살펴보자

```
let counter = 1;
let a = counter++; // (*) ++counter를 counter++로 바꿈

alert(a); // 1
alert(counter); //2
```

즉, 해당 줄의 코드를 실행할 때 ++를 미리 해놓고 코드를 실행하냐 코드를 실행하고 ++를 더해주냐 순서가 다른 것을 확인할 수 있다.

비교 연산자

자바스크립트에서 기본 수학 연산은 아래와 같은 문법을 사용해 표현할 수 있다.

1️⃣ 보다 큼·작음: `a > b`, `a < b`

2️⃣ 다 크거나·작거나 같음: `a >= b`, `a <= b`

3️⃣ 같음(동등): `a == b`. 등호 `=`가 두 개 연달아 오는 것에 유의하자. `a = b`와 같이 등호가 하나일 때는 변수를 넣어주는 것이다.

4️⃣ 같지 않음(부등): 같지 않음을 나타내는 수학 기호 `≠`는 자바스크립트에선 `a != b`로 나타낸다

불린형 반환

다른 연산자와 마찬가지로 비교 연산자 역시 불린형을 반환한다.

1️⃣ `true`가 반환되면, ‘긍정’, ‘참’, '사실’을 의미

2️⃣ `false`가 반환되면, ‘부정’, ‘거짓’, '사실이 아님’을 의미

```
alert( 2 > 1 );  // true
alert( 2 == 1 ); // false
alert( 2 != 1 ); // true
```

반환된 불린값은 다른 여타 값처럼 변수에 할당 할 수 있다.

```
let result = 5 > 4; // 비교 결과를 변수에 할당
alert( result ); // true
```

자료형이 다른 값들끼리의 비교

비교하려는 값의 자료형이 다르면 자바스크립트는 이 값들을 숫자형으로 바꾸게 된다.

```
alert( '2' > 1 ); // true, 문자열 '2'가 숫자 2로 변환된 후 비교가 진행됩니다.
alert( '01' == 1 ); // true, 문자열 '01'이 숫자 1로 변환된 후 비교가 진행됩니다.
```

불린값의 경우 `true`는 `1`, `false`는 `0`으로 변환된 후 비교가 이뤄진다.

```
alert( true == 1 ); // true
alert( false == 0 ); // true
```

💡 이상한 연산 ❓

동시에 일어나지 않을 법한 두 상황이 동시에 일어나는 경우도 JS에선 존재한다.

```
let a = 0;
alert( Boolean(a) ); // false

let b = "0";
alert( Boolean(b) ); // true

alert(a == b); // true!
```

위 코드를 자세히 보면 말이 안되는것처럼 느껴진다. 하지만 JS에선 이런 결과가 맞다.

동등 비교 연산자 `==`는 (예시에서 문자열 `"0"`을 숫자 `0`으로 변환시킨 것처럼) 피연산자를 숫자형으로 바꾸지만, '`Boolean`'을 사용한 변환은 숫자 `0`이 아닌 경우엔 다 `true`로 반환하기 때문이다.

일치 연산자

동등 연산자(equality operator) `==`은 `0`과 `false`를 구별하지 못한다.

```
alert( 0 == false ); // true
```

피연산자가 빈 문자열일 때 역시 구분하지 못한다.

```
alert( '' == false ); // true
```

이런 문제는 동등 연산자 `==`가 형이 다른 피연산자를 비교할 때 피연산자를 숫자형으로 바꾸기 때문에 발생한다. 빈 문자열과 `false`는 숫자형으로 변환하면 `0`이 되기 때문이다.

즉, 빈 문자열과 `false`는 `==`으로 구별할땐 `0`의 값을 가지게 된다.

일치 연산자(strict equality operator) `===`를 사용하면 형 변환 없이 값을 비교할 수 있게 된다.

```
alert( 0 === false ); // false, 피연산자의 형이 다르기 때문입니다.
```

'불일치' 연산자 `!==`는 부등 연산자 `!=`의 엄격헌 버전이다.

> 언제 `==`과 `===`을 사용해야 할지 헷갈린다면 `===`만 사용하는 습관을 들여도 괜찮다.
> (마치 `var`대신 `let`만 사용한 것처럼)

요약

- 비교 연산자는 불린값을 반환한다.
- 서로 다른 타입의 값을 비교할 땐 숫자형으로 형 변환이 이뤄지고 난 후 비교가 진행된다. (`==` `===` 제외)

💡 문제

아래 표현식들의 결과를 예측해보세요.

```
5 > 4
undefined == null
undefined === null
```

if와 '?'를 사용한 조건 처리

프로그래밍을 하다보면 특정 조건에 따라서 다른 코드를 실행시켜야 하는 경우가 발생한다.

이럴 땐, `if`문과 '물음표' 연산자라고도 불리는 조건부 연산자 `?`를 사용하면 된다.

IF

`if(...)`문은 괄호 안에 들어가는 조건을 평가하는데, 그 결과가 `true`이면 코드 블록이 실행되게 된다.

```
let year = prompt('올해는 몇년도일까요???', '');

if (year == 2023) alert( '정답입니다!' );
```

위 예시에선 조건(`year == 2023`)이 간단한 경우만 다뤘는데, 보다 복잡하게 구성하는것도 가능하다.

조건이 `true`일 때 복수의 문을 실행하고 싶다면 중괄호로 코드 블록을 감싸야 한다. 조건을 만족하는 경우, 중괄호 `{ }` 안의 코드를 실행하게 된다.

```
let year = prompt('올해는 몇년도일까요???', '');
if (year == 2023) {
  alert( "정답입니다!" );
  alert( "아주 똑똑하시네요!" );
}
```

`if`문을 쓸 때는 조건이 참일 경우 실행되는 구문이 단 한 줄이더라도 중괄호 `{}`를 사용해 코드를 블록으로 감싸는 것을 추천하며 그이유는 위와같이 작성하는 것이 코드 가독성을 높여주기 때문이다.

불린형으로의 반환

`if (…)` 문은 괄호 안의 표현식을 평가하고 그 결과를 불린값으로 변환한다.

이전에 배운 형 변환 규칙을 잠시복습해보자.

1️⃣ 숫자 `0`, 빈 문자열`""`, `null`, `undefined`, `NaN`은 불린형으로 변환 시 모두 `false`가 된다.

2️⃣ 이 외의 값은 불린형으로 변환시 `true`가 된다.

위규칙을 보고 아래 코드가 어떻게 작동될지 생각해보자.

```
if (0) { // 0은 falsy입니다.
  alert("동작이 될까요?");
}
```

```
if (1) { // 1은 truthy입니다.
  alert("동작이 될까요?");
}
```

아래와 같이 평가를 통해 확정된 불린값을 `if` 문에 전달할 수도 있다.

```
let year = prompt('올해는 몇년도일까요???', '');
let cond = (year == 2023); // 동등 비교를 통해 true/false 여부를 결정합니다.

if (cond) {
  alert("동작이 될까요?");
}
```

else 절

`if` 문엔 `else` 절을 붙일 수 있다. `if` 문 괄호안이 성립되면 `if {}` 안의 코드가 실행되며, 괄호안이 거짓인 경우 `else {}` 안의 코드가 실행되게 된다.

```
let year = prompt('올해는 몇년도일까요??', '');

if (year == 2023) {
  alert( '정답입니다!' );
} else {
  alert( '오답입니다!' ); // 2023 이외의 값을 입력한 경우
}
```

else if

`if` 와 `else` 를 사용하면 두가지 조건의 경우에만 판별할 수 있다. 만약 여러가지 조건을 판별하고 싶은 경우에는 `else if` 문을 사용할 수 있다.

```
let year = prompt('올해는 몇년도일까요??', '');

if (year < 2023) {
  alert( '숫자를 좀 더 올려보세요.' );
} else if (year > 2023) {
  alert( '숫자를 좀 더 내려보세요.' );
} else {
  alert( '정답입니다!' );
}
```

위 예시에서, 자바스크립트는 조건 `year < 2`023를 먼저 확인하고 이 조건이 거짓이라면 다음 조건 `year > 2023`를 확인한다. 이 조건 또한 거짓이라면 그제서야 `else` 절 내의 `alert`를 실행한다.

물론 `else if` 블록을 더 많이 붙이는 것도 가능한다. 또한 마지막에 붙는 `else`는 필수가 아닌 선택 사항이다.

조건부 연산자 `?`

조건에 따라 다른 값을 변수에 할당해줘야 하는 경우가 있다. 만약 `if`문을 활용한다면 아래와 같이 작성할 수 있을 것이다.

```
let accessAllowed;
let age = prompt('나이를 입력해 주세요.', '');

if (age > 18) {
  accessAllowed = true;
} else {
  accessAllowed = false;
}alert(accessAllowed);
```

'`물음표(question mark) 연산자`'라고도 불리는 '`조건부(conditional) 연산자`'를 사용하면 위 예시를 더 짧고 간결하게 나타낼 수 있다.

조건부 연산자는 물음표`?`로 표시하며 `삼항연산자` 라고도 부른다.

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

평가 대상인 `condition`이 `true`라면 `value1`이, 그렇지 않으면 `value2`가 `result`에 들어가게 된다.

```
let accessAllowed = (age > 18) ? true : false;
```

물음표 연산자 `?`를 여러 개 연결하면 복수의 조건을 처리하는 것도 가능하다.

```
let age = prompt('나이를 입력해주세요.', 18);

let message = (age < 3) ? '아기야 안녕?' :
  (age < 18) ? '안녕!' :
  (age < 100) ? '환영합니다!' :
  '나이가 아주 많으시거나, 나이가 아닌 값을 입력 하셨군요!';

alert( message );
```

처음 위 코드를 만나게 되면 굉장히 이해하기 어려울 수 있다. 한번 아래의 문장을 읽고 천천히 이해해보자.

1️⃣ 첫 번째 물음표에선 조건문 `age < 3`을 검사한다.

2️⃣ 그 결과가 참이면 `'아기야 안녕?'`를 반환하며 그렇지 않다면 첫 번째 콜론 `":"`에 이어지는 조건문 `age < 18`을 검사한다.

3️⃣ 그 결과가 참이면 `'안녕!'`를 반환하며 그렇지 않다면 다음 콜론 `":"`에 이어지는 조건문 `age < 100`을 검사합니다.

4️⃣ 그 결과가 참이면 `'환영합니다!'`를 반환하며 그렇지 않다면 마지막 콜론 `":"` 이후의 표현식인 `'나이가 아주 많으시거나, 나이가 아닌 값을 입력 하셨군요!'`를 반환합니다.

`if..else`를 사용하면 위 예시를 아래와 같이 변형할 수 있다.

```
if (age < 3) {
  message = '아기야 안녕?';
} else if (age < 18) {
  message = '안녕!';
} else if (age < 100) {
  message = '환영합니다!';
} else {
  message = '나이가 아주 많으시거나, 나이가 아닌 값을 입력 하셨군요!';
}
```

사실 모든 `?` 연산자는 `if`문으로 대체가능하며 과도한 `?` 사용은 코드 가독성을 떨어뜨리기에 적절히 사용하거나, ? 연산자 사용이 어색하다면 `if`문을 위주로 코드를 작성해도 괜찮다.

문제 1

if 문자열 0

아래 코드에서 `alert`는 실행될까요?

```
if ("0") {
  alert( 'Hello' );
}
```

문제 2

`if..else` 구조를 이용해 "자바스크립트의 ‘공식’ 이름은 무엇일까요?"라는 질문을 하는 코드를 작성해 보세요.

사용자가 'ECMAScript'를 입력했다면 ‘정답입니다!’, 아니라면 '모르셨나요? 정답은 ECMAScript입니다!'라는 메시지를 보여주세요

문제 3

`if..else`와 `프롬프트 대화상자`를 사용해 사용자로부터 숫자 하나를 입력받고, 아래 조건에 따라 그 결과를 `alert` 창에 출력해 보세요.

- 입력받은 숫자가 0보다 큰 경우 `1`을 출력
- 입력받은 숫자가 0보다 작은 경우 `1`을 출력
- 입력받은 숫자가 0인 경우 `0`을 출력

(사용자는 항상 숫자를 입력한다고 가정)

문제 4

조건부 연산자 `'?'`를 이용해 `if`문이 사용된 아래 코드를 변형해보세요. 동작 결과는 동일해야 합니다.

```
let result;

if (a + b < 4) {
  result = '미만';
} else {
  result = '이상';
}

alert(result);
```

문제 5

조건부 연산자 `'?'`를 사용해 `if..else`문이 사용된 아래 코드를 변형해보세요. 동작 결과는 동일해야 합니다.

가독성을 위해 표현식을 여러 줄로 분할해 작성해 보시길 바랍니다.

```
let message;

if (login == '직원') {
  message = '안녕하세요.';
} else if (login == '임원') {
  message = '환영합니다.';
} else if (login == '') {
  message = '로그인이 필요합니다.';
} else {
  message = '';
}
```