Умовні розгалуження: if, '?'

Іноді нам потрібно виконувати різні дії на основі різних умов.

Для цього ми можемо використовувати інструкцію `if` та умовний оператор `?`, що також називається оператором “знак питання”.

Інструкція “if”

Інструкція `if(...)` обчислює умову в дужках і, якщо результат умови `true`, виконує блок коду.

Наприклад:

```
let year = prompt('В якому році була опублікована специфікація ECMAScript-2015?', '');

if (year == 2015) alert( 'Ви маєте рацію!' );
```

У наведеному вище прикладі умовою є проста перевірка рівності (`year == 2015`), але умова може бути набагато складнішою.

Якщо ми хочемо виконати більше однієї інструкції, ми повинні загорнути блок коду у фігурні дужки:

```
if (year == 2015) {
  alert( "Це правильно!" );
  alert( "Ви такий розумний!" );
}
```

Ми рекомендуємо огортати блок коду фігурними дужками `{}` кожного разу, коли ви використовуєте інструкцію `if`, навіть якщо виконуватиметься лише одна операція. Це покращує читабельність.

Булеве перетворення

Інструкція `if (…)` обчислює вираз у дужках і перетворює результат у логічний тип.

Нагадаємо правила перетворення з розділу Перетворення типу:

- Число `0`, порожній рядок `""`, `null`, `undefined` та `NaN` перетворюються на `false`. Через це їх називають “хибними” (“falsy”) значеннями.
- Інші значення перетворюються на `true`, тому їх називають “правдивими”.

Отже, код ніколи не виконається за такої умови:

```
if (0) { // 0 є хибним
  ...
}
```

…а всередині цієї умовної конструкції – завжди буде виконуватися:

```
if (1) { // 1 є правдивим
  ...
}
```

Ми також можемо передавати попередньо обчислене значення до `if`, наприклад:

```
let condition = (year == 2015); // рівність обчислюється як true або false

if (condition) {
  ...
}
```

Блок “else”

Вираз `if` може містити необов’язковий блок “else” (“інакше”). Він виконується, коли умова є хибною.

Наприклад:

```
let year = prompt('В якому році була опублікована специфікація ECMAScript-2015?', '');

if (year == 2015) {
  alert( 'Ви здогадалися правильно!' );
} else {
  alert( 'Як ви можете так помилятися?' ); // будь-яке значення, окрім 2015
}
```

Декілька умов: “else if”

Іноді ми хотіли б перевірити кілька варіантів умови. Блок `else if` дозволяє нам це зробити.

Наприклад:

```
let year = prompt('В якому році була опублікована специфікація ECMAScript-2015?', '');
```

```
if (year < 2015) {
  alert( 'Зарано...' );
} else if (year > 2015) {
  alert( 'Запізно' );
} else {
  alert( 'Саме так!' );
}
```

У наведеному вище коді JavaScript спочатку перевіряє `year < 2015`. Якщо це не вірно, перевіряється наступна умова `year > 2015`. Якщо це також неправда, показується останній `alert`.

Може бути більше `else if` блоків. Останній блок `else` є необов'язковим.

Умовний оператор '?'

Іноді нам необхідно присвоїти значення змінній в залежності від умови.

Наприклад:

```
let accessAllowed;
let age = prompt('Скільки вам років?', '');

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

alert(accessAllowed);
```

Так званий "умовний" оператор або оператор "знак питання" дає нам зробити це в більш короткій і простій формі.

Оператор представлений знаком питання `?`. Іноді його називають "тернарним", оскільки оператор має три операнди. Насправді це єдиний оператор у JavaScript, який має так багато операндів.

Синтаксис:

```
let result = умова ? значення1 : значення2;
```

Спочатку обчислюється `умова`: якщо вона є правдивою, тоді повертається `значення1`, інакше – `значення2`.

Наприклад:

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

З технічного погляду, ми можемо пропускати дужки навколо `age > 18`. Оператор "знак питання" має низький пріоритет, тому він виконується після порівняння `>`.

Цей приклад робить теж саме, що і попередній:

```
// оператор порівняння "age > 18" виконується першим
// (не потрібно обертати його у дужки)
let accessAllowed = age > 18 ? true : false;
```

Але дужки роблять код більш читабельним, тому ми рекомендуємо їх використовувати.

Будь ласка, зверніть увагу:

У наведеному вище прикладі можна уникнути використання оператора “знак питання”, оскільки само порівняння повертає `true/false`:

```
// те ж саме
let accessAllowed = age > 18;
```

Декілька ‘?’

Послідовність операторів знака питання `?` може повернути значення, яке залежить від більш ніж однієї умови.

Наприклад:

```
let age = prompt('Вік?', 18);

let message = (age < 3) ? 'Привіт, крихітко!' :
  (age < 18) ? 'Вітаю!' :
  (age < 100) ? 'Моє шанування!' :
  'Який незвичайний вік!';

alert( message );
```

Спочатку може бути важко зрозуміти, що відбувається. Але, придивившись ближче, ми можемо побачити, що це звичайна послідовність перевірок:

1. Перший “знак питання” перевіряє, чи `age < 3`.
2. Якщо правда – то повертає `'Привіт, крихітко!'`. У іншому випадку переходить до виразу після двокрапки “':'”, перевіряючи `age < 18`.
3. Якщо це правда – то повертає `'Вітаю!'`. В іншому випадку переходить до виразу після наступної двокрапки “':'”, перевіряючи `age < 100`.
4. Якщо це правда – то повертає `'Моє шанування!'`. У іншому випадку переходить до виразу після останньої двокрапки “':'”, повертаючи `'Який незвичайний вік!'`.

Ось як це буде виглядати у випадку з використанням `if..else`:

```
if (age < 3) {
  message = 'Привіт, крихітко!';
} else if (age < 18) {
```

```
  message = 'Вітаю!';
} else if (age < 100) {
  message = 'Моє шанування!';
} else {
  message = 'Який незвичайний вік!';
}
```

Нетрадиційне використання ‘?’

Іноді оператор “знак питання” `?` використовується як заміна `if`:

```
let company = prompt('Яка компанія створила JavaScript?', '');

(company == 'Netscape') ?
   alert('Правильно!') : alert('Неправильно.');
```

Залежно від умови `company == 'Netscape'`, буде виконано або перший, або другий вираз після `?` і показано повідомлення.

Тут ми не присвоюємо результат змінній. Замість цього ми виконуємо різний код залежно від умови.

Не рекомендується використовувати оператор “знак питання” таким чином.

Запис є коротшим, ніж еквівалентна інструкція `if`, що приваблює деяких програмістів. Але він менш читабельний.

Ось такий самий код, що використовує `if` для порівняння:

```
let company = prompt('Яка компанія створила JavaScript?', '');

if (company == 'Netscape') {
  alert('Правильно!');
} else {
  alert('Неправильно.');
}
```

Наші очі сканують код по вертикалі. Блоки коду, що охоплюють кілька рядків, легше зрозуміти, ніж довгий горизонтальний набір інструкцій.

Мета оператора “знак питання” `?` полягає в поверненні одного або іншого значення в залежності від умови. Будь ласка, використовуйте його саме для цього. Використовуйте `if`, коли вам потрібно виконати різні гілки коду.

Завдання

Challenge 1 **if (рядок з нулем)**

Чи буде показано `alert` ?

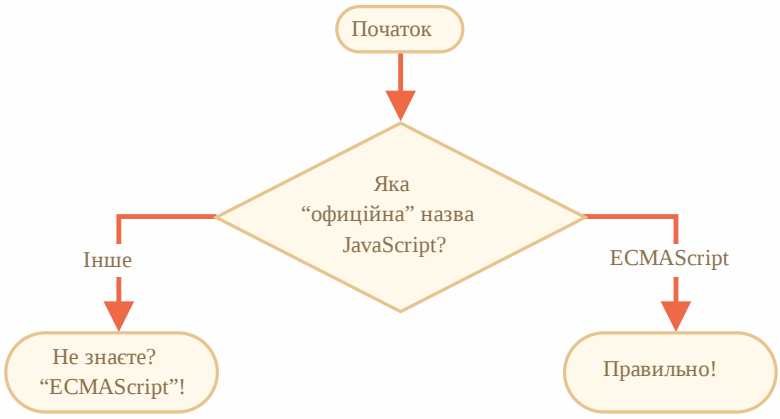
```

```

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

Challenge 2 **Назва JavaScript**

Використовуючи конструкцію `if..else`, напишіть код, що запитує: ‘Яка “офіційна” назва JavaScript?’

Якщо відвідувач вводить “ECMAScript”, виведіть “Правильно!”, в іншому випадку — виведіть: “Ви не знаєте? ECMAScript!”



Challenge 3 **Покажіть знак**

Використовуючи `if..else`, напишіть код, що отримує число за допомогою `prompt` і потім виводить повідомлення `alert`:

- `1`, якщо значення більше нуля,
- `-1`, якщо меньше нуля,
- `0`, якщо дорівнює нулю.

У цьому завданні ми припускаємо, що введенне значення завжди є числом.

Challenge 4 **Перепишіть 'if' на '?'**

Перепишіть конструкцію `if`, використовуючи умовний оператор `'?'`:

```
let result;

if (a + b < 4) {
  result = 'Нижче';
} else {
  result = 'Вище';
}
```

Challenge 5 **Перепишіть 'if..else' на '?'**

Перепишіть `if..else`, використовуючи декілька тернарних операторів `'?'`.

Для зручності читання рекомендується розділити код на кілька рядків.

```
let message;

if (login == 'Працівник') {
  message = 'Привіт';
} else if (login == 'Директор') {
  message = 'Вітаю';
} else if (login == '') {
  message = 'Немає логіну';
} else {
  message = '';
}
```