


[🏠](#) → [Язык JavaScript](#) → [Основы JavaScript](#) 11 декабря 2022 г.

# Условное ветвление: if, '?'

Иногда нам нужно выполнить различные действия в зависимости от условий.

Для этого мы можем использовать инструкцию `if` и условный оператор `?`, который также называют оператором «вопросительный знак».

## Инструкция «if»

Инструкция `if(...)` вычисляет условие в скобках и, если результат `true`, то выполняет блок кода.

Например:

```
1 let year = prompt('В каком году была опубликована спецификация ECMAScript-2015');
2
3 if (year == 2015) alert('Вы правы!');
```

В примере выше, условие – это простая проверка на равенство (`year == 2015`), но оно может быть и гораздо более сложным.

Если мы хотим выполнить более одной инструкции, то нужно заключить блок кода в фигурные скобки:

```
1 if (year == 2015) {
2   alert("Правильно!");
3   alert("Вы такой умный!");
4 }
```

Мы рекомендуем использовать фигурные скобки `{}` всегда, когда вы используете инструкцию `if`, даже если выполняется только одна команда. Это улучшает читаемость кода.

## Преобразование к логическому типу

Инструкция `if (...)` вычисляет выражение в скобках и преобразует результат к логическому типу.

Давайте вспомним правила преобразования типов из главы [Преобразование типов](#):

- Число `0`, пустая строка `""`, `null`, `undefined` и `NaN` становятся `false`. Из-за этого их называют «ложными» («falsy») значениями.
- Остальные значения становятся `true`, поэтому их называют «правдивыми» («truthy»).

Таким образом, код при таком условии никогда не выполнится:

```
1 if (0) { // 0 is falsy
2   ...
3 }
```

...а при таком – выполнится всегда:

```
1 if (1) { // 1 is truthy
2   ...
3 }
```

Мы также можем передать заранее вычисленное в переменной логическое значение в `if`, например так:

```
1 let condition = (year == 2015); // преобразуется к true или false
2
3 if (condition) {
4   ...
5 }
```

## Блок «else»

Инструкция `if` может содержать необязательный блок «else» («иначе»). Он выполняется, когда условие ложно.

Например:

```
1 let year = prompt('В каком году была опубликована спецификация ECMAScript-2015');
2
3 if (year == 2015) {
4   alert( 'Да вы знаток!' );
5 } else {
6   alert( 'А вот и неправильно!' ); // любое значение, кроме 2015
7 }
```

## Несколько условий: «else if»

Иногда нужно проверить несколько вариантов условия. Для этого используется блок `else if`.

Например:

```
1 let year = prompt('В каком году была опубликована спецификация ECMAScript-2015');
2
3 if (year < 2015) {
4   alert( 'Это слишком рано...' );
5 } else if (year > 2015) {
6   alert( 'Это поздновато' );
7 } else {
8   alert( 'Верно!' );
9 }
```

В приведённом выше коде JavaScript сначала проверит `year < 2015` . Если это неверно, он переходит к следующему условию `year > 2015` . Если оно тоже ложно, тогда сработает последний `alert` .

Блоков `else if` может быть и больше. Присутствие блока `else` не является обязательным.

## Условный оператор „?“

Иногда нам нужно определить переменную в зависимости от условия.

Например:

```
1 let accessAllowed;
2 let age = prompt('Сколько вам лет?', '');
3
4 if (age > 18) {
5     accessAllowed = true;
6 } else {
7     accessAllowed = false;
8 }
9
10 alert(accessAllowed);
```



Так называемый «условный» оператор «вопросительный знак» позволяет нам сделать это более коротким и простым способом.

Оператор представлен знаком вопроса `?` . Его также называют «тернарный», так как этот оператор, единственный в своём роде, имеет три аргумента.

Синтаксис:

```
1 let result = условие ? значение1 : значение2;
```

Сначала вычисляется `условие` : если оно истинно, тогда возвращается `значение1` , в противном случае – `значение2` .

Например:

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

Технически, мы можем опустить круглые скобки вокруг `age > 18` . Оператор вопросительного знака имеет низкий приоритет, поэтому он выполняется после сравнения `>` .

Этот пример будет делать то же самое, что и предыдущий:

```
1 // оператор сравнения "age > 18" выполняется первым в любом случае
2 // (нет необходимости заключать его в скобки)
3 let accessAllowed = age > 18 ? true : false;
```

Но скобки делают код более простым для восприятия, поэтому мы рекомендуем их использовать.

### На заметку:

В примере выше вы можете избежать использования оператора вопросительного знака `?`, т.к. сравнение само по себе уже возвращает `true/false`:

```
1 // то же самое
2 let accessAllowed = age > 18;
```

## Несколько операторов „?“

Последовательность операторов вопросительного знака `?` позволяет вернуть значение, которое зависит от более чем одного условия.

Например:

```
1 let age = prompt('Возраст?', 18);
2
3 let message = (age < 3) ? 'Здравствуй, малыш!' :
4   (age < 18) ? 'Привет!' :
5   (age < 100) ? 'Здравствуйте!' :
6   'Какой необычный возраст!';
7
8 alert( message );
```



Поначалу может быть сложно понять, что происходит. Но при ближайшем рассмотрении мы видим, что это обычная последовательная проверка:

1. Первый знак вопроса проверяет `age < 3`.
2. Если верно – возвращает `'Здравствуй, малыш!'`. В противном случае, проверяет выражение после двоеточия „:“, вычисляет `age < 18`.
3. Если это верно – возвращает `'Привет!'`. В противном случае, проверяет выражение после следующего двоеточия „:“, вычисляет `age < 100`.
4. Если это верно – возвращает `'Здравствуйте!'`. В противном случае, возвращает выражение после последнего двоеточия – `'Какой необычный возраст!'`.

Вот как это выглядит при использовании `if..else`:

```
1 if (age < 3) {
2   message = 'Здравствуй, малыш!';
3 } else if (age < 18) {
4   message = 'Привет!';
5 } else if (age < 100) {
6   message = 'Здравствуйте!';
7 } else {
8   message = 'Какой необычный возраст!';
9 }
```

# Нетрадиционное использование „?“

Иногда оператор «вопросительный знак» ? используется в качестве замены `if`:

```
1 let company = prompt('Какая компания создала JavaScript?', '');
2
3 (company == 'Netscape') ?
4   alert('Верно!') : alert('Неправильно.');
```



В зависимости от условия `company == 'Netscape'`, будет выполнена либо первая, либо вторая часть после ?.

Здесь мы не присваиваем результат переменной. Вместо этого мы выполняем различный код в зависимости от условия.

**Не рекомендуется использовать оператор вопросительного знака таким образом.**

Несмотря на то, что такая запись короче, чем эквивалентная инструкция `if`, она хуже читается.

Вот, для сравнения, тот же код, использующий `if`:

```
1 let company = prompt('Какая компания создала JavaScript?', '');
2
3 if (company == 'Netscape') {
4   alert('Верно!');
5 } else {
6   alert('Неправильно. ');
7 }
```



При чтении глаза сканируют код по вертикали. Блоки кода, занимающие несколько строк, воспринимаются гораздо легче, чем длинный горизонтальный набор инструкций.

Смысл оператора «вопросительный знак» ? – вернуть то или иное значение, в зависимости от условия. Пожалуйста, используйте его именно для этого. Когда вам нужно выполнить разные ветви кода – используйте `if`.

## ✓ Задачи

### if (строка с нулём)

важность: 5

Выведется ли `alert`?

```
1 if ("0") {
2   alert( 'Привет' );
3 }
```

решение

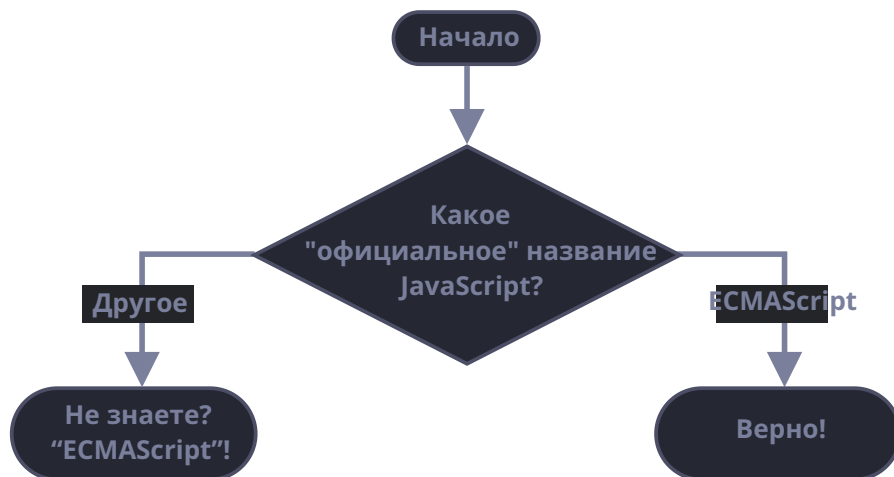
---

## Название JavaScript

важность: 2

Используя конструкцию `if..else`, напишите код, который будет спрашивать: „Какое «официальное» название JavaScript?»

Если пользователь вводит «ECMAScript», то показать: «Верно!», в противном случае – отобразить: «Не знаете? ECMAScript!»



[Демо в новом окне](#)

решение

---

## Покажите знак числа

важность: 2

Используя конструкцию `if..else`, напишите код, который получает число через `prompt`, а затем выводит в `alert`:

- 1, если значение больше нуля,
- -1, если значение меньше нуля,
- 0, если значение равно нулю.

Предполагается, что пользователь вводит только числа.

[Демо в новом окне](#)

решение

---

## Перепишите 'if' в '?'

важность: 5

Перепишите конструкцию `if` с использованием условного оператора `'?'`:

```
1 let result;  
2
```

```
3  if (a + b < 4) {
4    result = 'Мало';
5  } else {
6    result = 'Много';
7  }
```

решение

## Перепишите 'if..else' в '?'

важность: 5

Перепишите `if..else` с использованием нескольких операторов `'?'`.

Для читаемости рекомендуется разбить код на несколько строк.

```
1  let message;
2
3  if (login == 'Сотрудник') {
4    message = 'Привет';
5  } else if (login == 'Директор') {
6    message = 'Здравствуйте';
7  } else if (login == '') {
8    message = 'Нет логина';
9  } else {
10   message = '';
11 }
```

решение



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

Проводим курсы по JavaScript и фреймворкам.



## Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>`, для нескольких строк кода — тег `<pre>`, если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.

