

Язык JavaScript (основы:
структурное программирование)

JS

УСЛОВНЫЕ ОПЕРАТОРЫ

Условные операторы используются, когда в зависимости от условия необходимо выполнить разные действия.

Оператор `if` вычисляет и преобразует условие к логическому типу, если указанное **условие** истинно (`true`), выполняет **код** в фигурных скобках.

Синтаксис:

```
if (условие) {  
    код, который выполнится, если условие истинно  
}
```

Например,

```
if (answer === 'added') { /* если значение переменной answer  
будет строго равно 'added' в консоль будет выведено:  
Пользователь добавлен */  
    console.log("Пользователь добавлен");  
}
```

НЕОБЯЗАТЕЛЬНЫЙ БЛОК ELSE

Необязательный блок else выполняется, если условие ложно (false).

Синтаксис:

```
if (условие) {  
    код, который выполнится, если условие истинно  
} else {  
    код, который выполнится, если условие ложно  
}
```

Например,

```
if (answer === 'added') { /* если значение переменной answer будет  
    строго равно 'added' в консоль будет выведено: Пользователь  
    добавлен */  
    console.log("Пользователь добавлен");  
} else { /* если значение переменной answer не будет строго равно  
    'added' в консоль будет выведено: Пользователь не добавлен */  
    console.log("Пользователь не добавлен");  
}
```

НЕСКОЛЬКО УСЛОВИЙ ELSE IF

Несколько условий else if используются, если необходимо добавить новые варианты условий. Каждое новое условие будет проверяться только, если предыдущие условия ложны.

Синтаксис:

```
if (условие) {  
    код, который выполнится, если условие истинно  
} else if (условие 2){  
    код, который выполнится, если условие 1 ложно и  
    условие 2 истинно  
} else {  
    код, который выполнится, если условие 1 ложно и  
    условие 2 ложно  
}
```

НЕСКОЛЬКО УСЛОВИЙ ELSE IF

Например:

```
if (answer === 'added') { /* если значение переменной answer
    будет строго равно 'added' в консоль будет выведено:
    Пользователь добавлен */
    console.log("Пользователь добавлен");
} else if (answer === 'not added') { /* если значение
    переменной answer будет строго равно 'not added' в
    консоль будет выведено: Пользователь не добавлен */
    console.log("Пользователь не добавлен");
} else { /* если значение переменной answer не будет строго
    равно 'added' или 'not added' в консоль будет выведено:
    Ошибка соединения */
    console.log("Ошибка соединения");
}
```

КОНСТРУКЦИЯ SWITCH

Конструкция switch заменяет сразу несколько if. Сравнивает выражение сразу с несколькими вариантами и может выполнять один или несколько блоков кода.

Синтаксис,

```
switch(выражение) {  
    case значение1 : // if (выражение === значение1)  
        ... КОД  
        [break]  
    case значение2 : // if (выражение === значение1)  
        ... КОД  
        [break]  
    case значение3 : // if (выражение === значение3)  
    case значение4 : // if (выражение === значение4)  
        ... КОД  
        [break]  
    ...  
    default :  
        ... код, который необходимо выполнить, если ни один case не совпал  
}
```

Выражение проверяется на строгое равенство 1му значению **значение1**, затем 2му **значение2** и так далее. Если **соответствие установлено** – switch **начинает выполняться** от соответствующего case и далее, **до ближайшего break или до конца switch**. Если **ни один case не совпал** – выполняется вариант **default** (если он описан).

Конструкция switch (пример)

```
var item = какое-то значение;  
switch (item) {  
  case "Oranges": // если item === "Oranges",  
    // отработает данный case и в консоле мы увидим "Oranges - $0.59 a pound."  
    console.log("Oranges - $0.59 a pound.");  
    // так как break не указан следующий case отработает без проверки условия  
    // в консоле мы увидим также "Apples - $0.32 a pound."  
  case "Apples": // если item === "Apples",  
    // отработает данный case и в консоле мы увидим "Apples - $0.32 a pound."  
    console.log("Apples - $0.32 a pound.");  
    break;  
  case "Mangoes": // если item === "Mangoes" или / и  
  case "Papayas": // если item === "Papayas"  
    // в консоле мы увидим "Mangoes and papayas are $2.79 a pound."  
    console.log("Mangoes and papayas are $2.79 a pound.");  
    break;  
  default: // если item не найдет совпадений,  
    // в консоле мы увидим "Sorry, we are out of " + item + "."  
    console.log("Sorry, we are out of " + item + ".");  
}
```

ЦИКЛЫ - ЦИКЛ С ПРЕДУСЛОВИЕМ WHILE

Циклы позволяют выполнять однотипное действие несколько раз, например, вывести фотографии (статьи, товары и тд). Каждое повторение цикла называется **итерацией**.

Цикл с предусловием while

Условие будет проверяться перед каждым выполнением тела цикла, если оно истинно, тело цикла будет выполняться. Так будет происходить, пока условие истинно, когда условие станет ложным, программы выйдет из цикла.

Если условие изначально ложно, тела цикла не будет выполнено ни разу.

Если условие всегда истинно, цикл будет продолжаться бесконечно.

Синтаксис,

```
while (условие) { // проверка условия
```

```
    тело цикла выполняется, если условие истинно
```

```
} // если условие ложно, программа выходит из цикла и продолжает работу
```

```
код после цикла;
```


ЦИКЛ С ПРЕДУСЛОВИЕМ WHILE (ПРИМЕР)

```
var itemCount = 3;
while (itemCount) {
  console.log('Item № ' + itemCount);
  itemCount--;
}
```

Первая проверка условия: goodsCount = 3, значит условие истинно и тело цикла выполнится.

Итерация 1 (первое выполнение тела цикла):

1. в консоль будет выведено: Item № 3
2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 2

Вторая проверка условия: goodsCount = 2 (после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится

Итерация 2 (второе выполнение тела цикла):

1. в консоль будет выведено: Item № 2
2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 1

Третья проверка условия: goodsCount = 1 (после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится

Итерация 3 (второе выполнение тела цикла):

1. в консоль будет выведено: Item № 1
2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 0

Четвертая проверка условия: goodsCount = 0 (после уменьшения в теле цикла), значит условие ложно (т.к. 0 преобразуется к false) и тело цикла не будет выполнено, программа выйдет из цикла.

ЦИКЛ С ПОСТУСЛОВИЕМ DO...WHILE

Сначала будет выполняться тело цикла, а затем проверяться условие, если оно истинно, тело цикла будет выполняться еще раз. Так будет происходить, пока условие истинно, когда условие станет ложным, программы выйдет из цикла.

Если условие изначально ложно, тела цикла выполняется один раз, т.к. условие проверяется после выполнения тела цикла.

Если условие всегда истинно, цикл будет продолжаться бесконечно. Синтаксис,

do {

тело цикла выполнится первый раз в любом случае, далее будет выполняться, если условие истинно

} **while** (**условие**); /* проверка условия, если условие ложно, программа выходит из цикла и продолжает работу */

код после цикла;

ЦИКЛ С ПОСТУСЛОВИЕМ DO...WHILE (ПРИМЕР)

```
var goodsCount = 2;  
do { // тело цикла: вывод в консоль: Item № значение goodsCount  
    console.log('Item №' + goodsCount);  
    goodsCount--; // уменьшение значения goodsCount на 1  
} while (goodsCount); /* проверка условия: условие истинно, пока значение  
goodsCount больше 0, т.к. 0 преобразуется к false, условие станет ложным,  
когда значение goodsCount будет равно 0 */
```

Итерация 1 (первое повторение тела цикла):

1. в консоль будет выведено: Item № 2
2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 1

Первая проверка условия: goodsCount = 1, значит условие истинно и тело цикла выполнится.

Итерация 2 (второе повторение тела цикла):

1. в консоль будет выведено: Item № 1
2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 0

Вторая проверка условия: goodsCount = 0, значит условие ложно и тело цикла выполняться не будет, программа выйдет из цикла

ЦИКЛ FOR

Цикл for состоит из выполнения трех операций.

Синтаксис,

```
for ( [ начало-инициализация ]; [ условие ]; [ шаг ] ) {  
    тело цикла выполняется, если условие истинно  
}
```

[] - квадратные
скобки в описании
используются для
обозначения
необязательных
параметров

1. **начало-инициализация** - выполняется один раз, при заходе в цикл. Обычно это выражение инициализирует один или несколько счётчиков. Также используется для объявления переменных.
2. **условие** - проверяется при входе в цикл и далее перед каждой итерацией, если оно истинно, тело цикла выполняется, если ложно, программа выходит из цикла. Если условие пропущено, оно считается истинным.
3. **шаг** - выполняется после выполнения тела цикла на каждой итерации, но перед проверкой условия. Обычно, обновление счетчика (уменьшение/увеличение).

ЦИКЛ FOR (ПРИМЕР)

```
let goodsCount = 3;  
for (let i = 0; i < goodsCount; i++) {  
  console.log('Item №' + goodsCount);  
  goodsCount--;  
}
```

Итерация 1 (первое выполнение тела цикла): 1. в консоль будет выведено: Item № 3 2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 2

Первое обновление счетчика (шаг): i++;
- значение i станет равно 1

Вторая проверка условия: i < goodsCount, значение i равно 1 (после первого обновления счетчика), а goodsCount равно 2 (после уменьшения в теле цикла), получаем 1 < 2, значит условие истинно и тело цикла выполнится.

Начало цикла: let i = 0 - инициализация счетчика

Первая проверка условия: i < goodsCount, значение i равно 0, а значение goodsCount равно 3, получаем 0 < 3, значит условие истинно и тело цикла выполнится

Итерация 2 (второе выполнение тела цикла): 1. в консоль будет выведено: Item № 2 2. уменьшение значения goodsCount на 1, значит goodsCount будет равен 1

Второе обновление счетчика (шаг): i++;
значение i станет равно 2

Третья проверка условия: i < goodsCount, значение i равно 2 (после второго обновления счетчика), а значение goodsCount равно 1 (после уменьшения в теле цикла), получаем 2 < 1, значит условие ложно и тело не выполнится, программа выйдет из цикла

ДИРЕКТИВЫ BREAK И CONTINUE

Прерывание цикла break позволяет выйти из цикла в любой момент и продолжить выполнение кода после цикла.

Синтаксис,

```
while (условие) { // проверка условия
    тело цикла выполняется, если условие истинно
    if (условие2) {
        break; /* если условие2 истинно, программа
                выходит из цикла благодаря директиве break */
    }
}
код после цикла;
```

ДИРЕКТИВЫ BREAK И CONTINUE

Следующая итерация `continue` прекращает выполнение текущей итерации цикла.

Синтаксис и пример,

```
for (let i = 0; i < 10; i++) {  
    if (i % 2 === 0) continue; /* если i - четное число,  
    завершает текущую итерацию, цикл переходит к  
    проверки условия i < 10 */  
    console.log(i); // выводит значение i  
}
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Логические операторы используются для операций над логическими значениями. В JavaScript могут применяться к значениям любого типа и возвращают также значения любого типа. Если значение не логического типа – то оно к нему приводится в целях вычислений.

Оператор || (или) - вычисляет ровно столько значений, сколько необходимо – до первого true.

Пример 1,

```
let a = null;
let b = "";
let c = 0;
let d = "Еще какое-то значение";
let res = a || b || c || d;
console.log(res); // Еще какое-то значение
```

Пример 2,

```
let day = какой-то день недели;
if (day === 'суббота' ||
    day === 'воскресенье') {
    console.log("Сейчас выходные");
}
```


ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Оператор && (и) - вычисляет операнды слева направо до первого «ложного» и возвращает его, а если все истинные – то последнее значение.

Приоритет оператора И && больше, чем ИЛИ ||, он выполняется раньше.

Например,

```
let mon = 0;
```

```
let fr = 4;
```

```
let day = какой-то день недели в виде числа от 0 до 6;
```

```
if (day > mon && day < fr) {
```

```
    console.log('Сейчас рабочая неделя');
```

```
}
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Оператор ! (не)

1. Сначала приводит аргумент к логическому типу true/false.
2. Затем возвращает противоположное значение.

Например,

```
let isActive = true; // или false
if ( !isActive ){
  console.log('Элемент не активен');
}
```

ЧТО ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО

<https://learn.javascript.ru/ifelse> - Условные операторы: if, '?’

<https://learn.javascript.ru/logical-operators> - Логические операторы

<https://learn.javascript.ru/while-for> - Циклы while и for

<https://learn.javascript.ru/switch> - Конструкция "switch«

<https://learn.javascript.ru/coding-style> - Советы по стилю кода

<https://learn.javascript.ru/debugging-chrome> - Отладка в браузере Chrome