JavaScript Start

Занятие 2. Условные конструкции и циклы

IT Education Academy

Содержание

- Условные конструкции
- Циклические конструкции
- Дополнительно: инструменты для отладки кода

Условные конструкции

Иногда, в зависимости от условия, нужно выполнить различные действия. Для этого используется оператор **if**.

```
var year = prompt('Current year?', '');

if (year != 2016) alert( 'You are wrong!' );
...
```

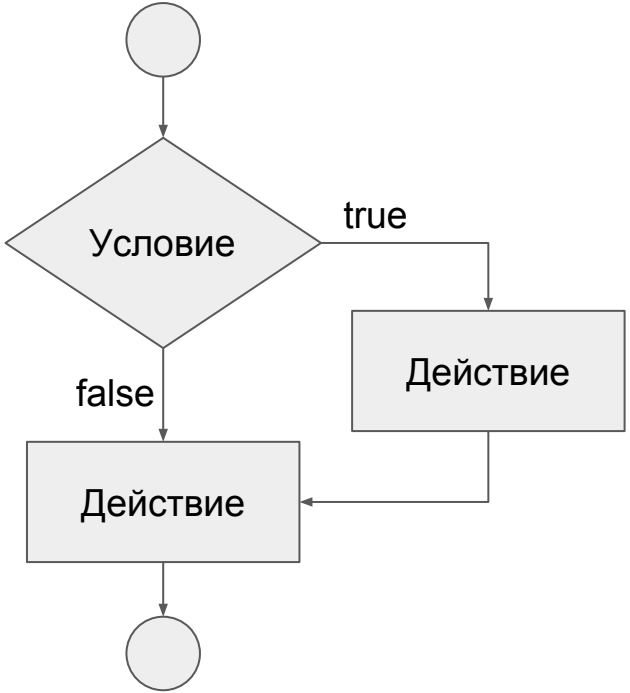
Если нужно выполнить более одной команды — они оформляются блоком кода в фигурных скобках.

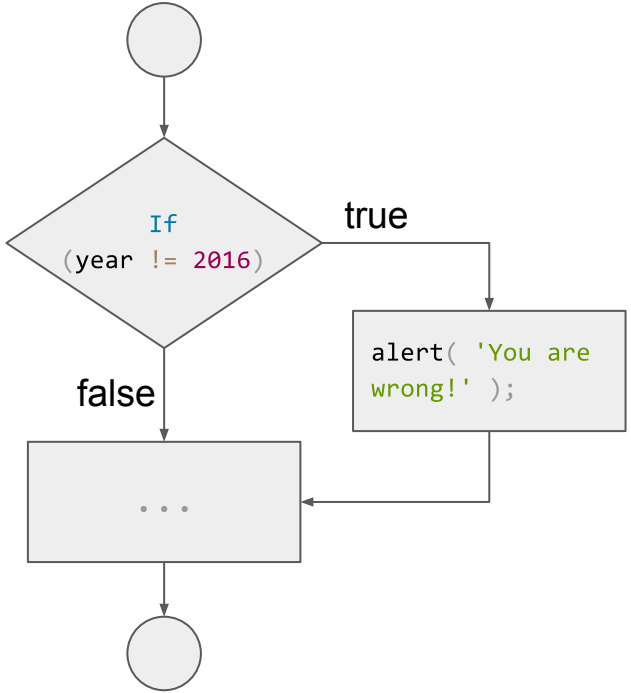
```
if (year != 2016) {
  alert( 'You are wrong!' );
  alert( 'Correct answer 2016' );
}
```

Рекомендуется использовать фигурные скобки всегда, даже когда команда одна.

Это улучшает читаемость кода.

Блок схема оператора if





Преобразование к логическому типу

Оператор `if (...)` вычисляет и преобразует выражение в скобках к true или false.

В логическом контексте:

- Число **`0`**, пустая строка **""**, **null** и **undefined**, а также **NaN** являются **false**,
- Остальные значения — **true**.

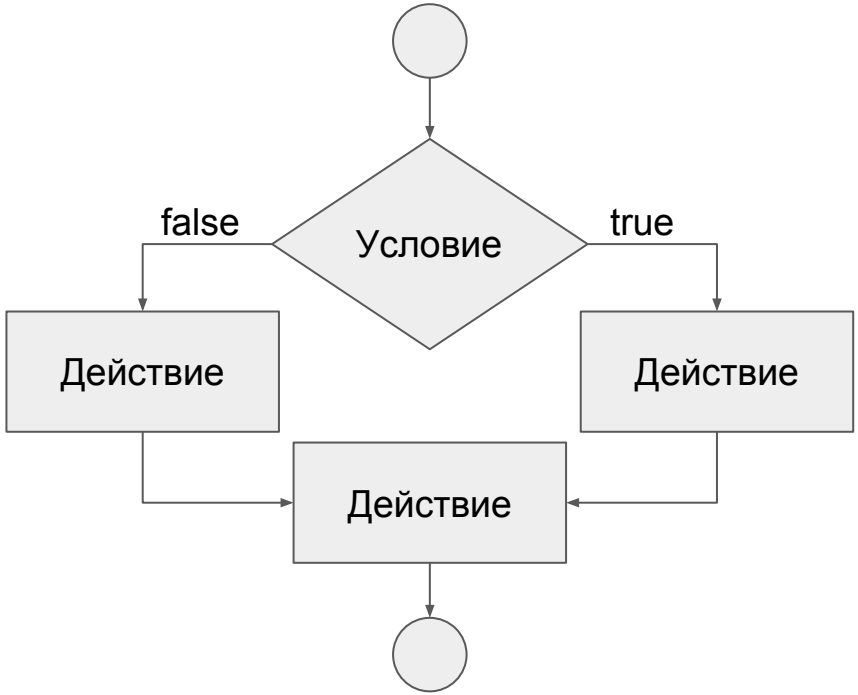
If … else

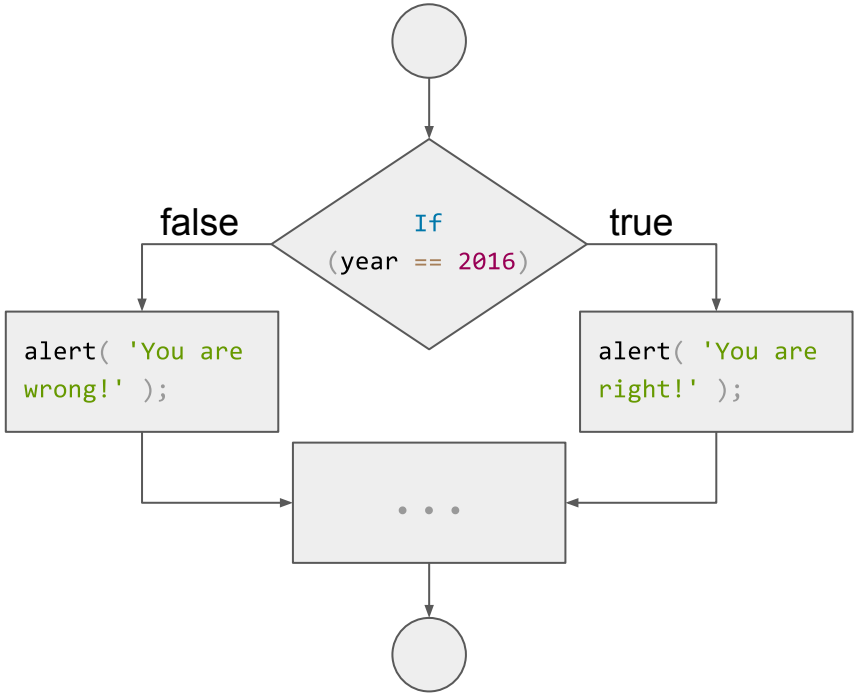
Необязательный блок **else** («**иначе**») выполняется, если условие неверно.

```
var year = prompt('Current Year?', '');

if (year == 2016) {
  alert( 'You are right!' );
} else {
  alert( 'You are wrong!' ); // everything except 2016
}
...
```

Блок схема оператора if … else





Задача 1

Написать программу, которая:

- Запросит 2 числа.
- Проверит, равно ли второе число нулю.
  - Если да, то выведет на экран фразу: «На ноль делить нельзя!»
  - Если нет, то найдет их частное (поделит одно на другое).

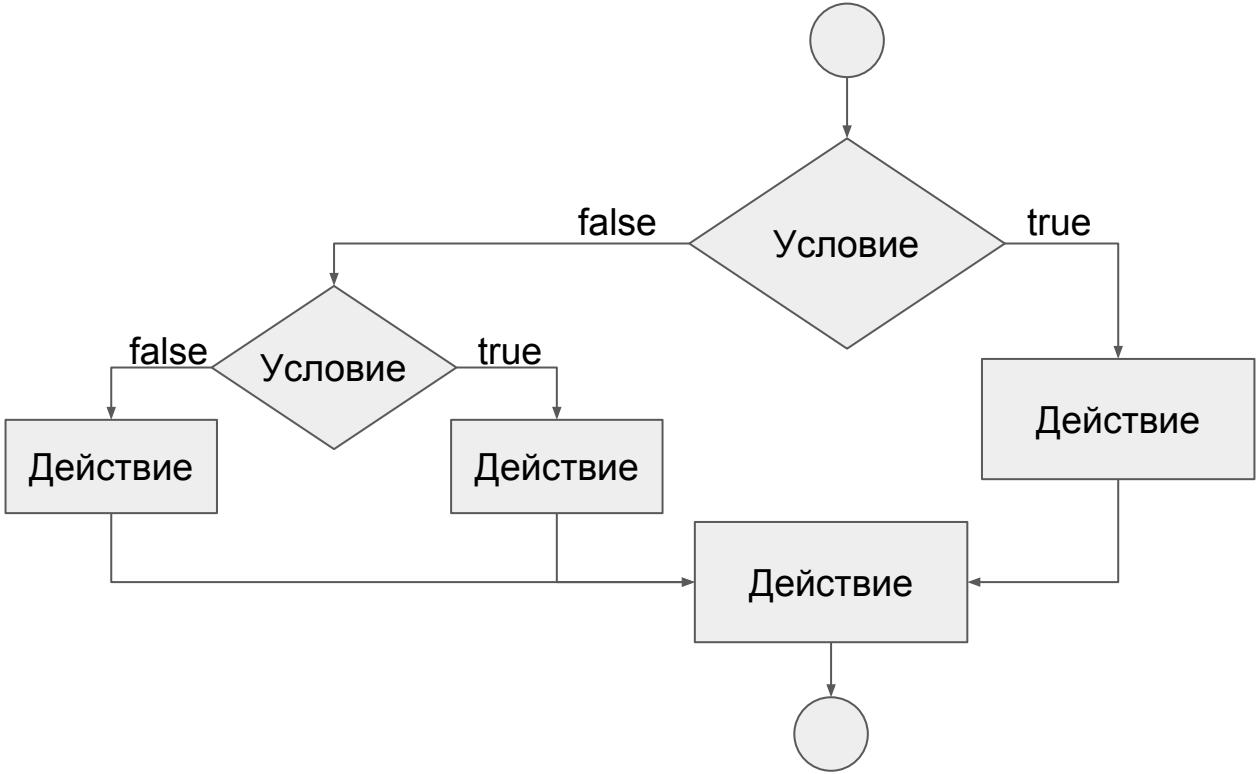
Несколько условий, else if

Бывает нужно проверить несколько вариантов условия. Для этого используется блок **else if.**

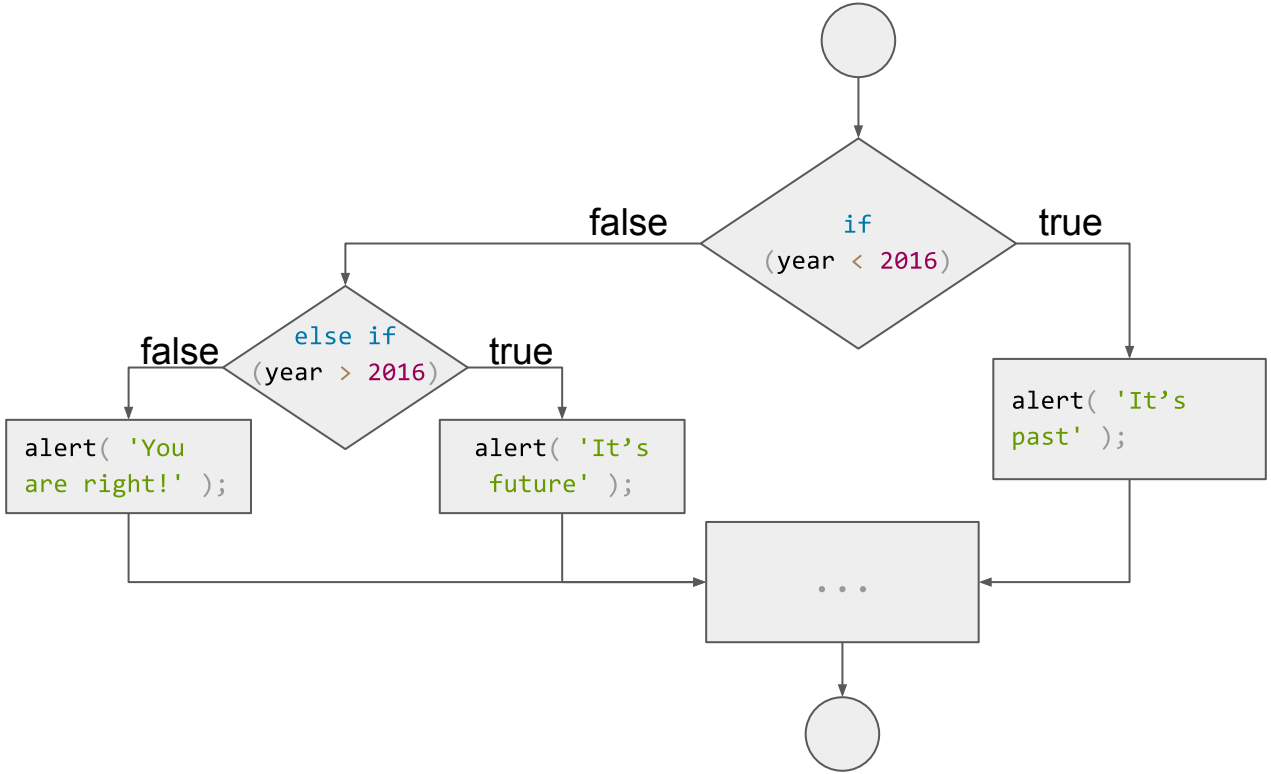
```
var year = prompt('Current year?', '');

if (year < 2016) {
  alert( 'It’s past' );
} else if (year > 2016) {
  alert( 'It’s future' );
} else {
  alert( 'You are right!' );
}
...
```

Блок схема else if



Блок схема else if



Задача 2

Написать программу, которая:

- Запрашивает возраст.
- Если меньше 18 выдает: «Слишком мал».
- Если старше 25 лет, выдаёт ответ: «Стороват будешь»
- В противном случае — «Подходишь».

Логическое ИЛИ (||)

Логическое ИЛИ в классическом программировании работает следующим образом: **«если хотя бы один из аргументов true, то возвращает true, иначе — false»**.

Оператор ИЛИ выглядит как двойной символ вертикальной черты.

```
result = a || b;
```

Обычно ИЛИ используется в if

```
var hour = 12,
  isWeekend = true;

if (hour < 10 || hour > 18 ||
isWeekend) {
  alert( 'Office is closed. Working
hours: 10:00 – 18:00, monday – friday'
);
}
```

Важно понимать 2 вещи

1. **JavaScript** вычисляет несколько **ИЛИ** слева направо. При этом, чтобы экономить ресурсы, используется так называемый **«короткий цикл вычисления»**.

Допустим, вычисляются несколько **ИЛИ** подряд: `a || b || c || ....` Если первый аргумент — **true**, то результат заведомо будет **true** (хотя бы одно из значений — **true**), и остальные значения игнорируются.

2. Оператор **ИЛИ** возвращает то значение, на котором остановились вычисления. Причём, **не преобразованное** к логическому типу.

```
var undef; // variable is undefined
var zero = 0;
var emptyStr = "";
var msg = "Hallo!";

var result = undef || zero || emptyStr || msg
|| 0;

alert( result ); // "Hallo!" – first value
that is true
```

Задача 3

Написать программу с использованием оператора **ИЛИ** (||), которая:

- Запрашивает возраст.
- Если меньше 7 и старше 10 лет, выдаёт ответ: «Подходишь».
- Если нет, то «Не подходишь».

Логическое И (&&)

В классическом программировании **И** возвращает **true**, если оба аргумента истинны, а иначе — **false.**

```
var hour = 12,
  minute = 30;

if (hour == 19 && minute == 40) {
  alert( 'Now is 19:40' );
}
```

Если левый аргумент — **false**, оператор **И** возвращает его и заканчивает вычисления. Иначе — вычисляет и возвращает правый аргумент.

Можно передать и несколько значений подряд, при этом возвратится первое «ложное» (на котором остановились вычисления), а если его нет — то последнее.

```
alert( 1 && 2 && null && 3 ); // null
```

Задача 4

Написать программу, которая:

- Запрашивает возраст.
- Если **не** меньше 7 и **не** старше 10 лет, выдаёт ответ: «Подходишь».
- Если нет, то «Не подходишь».

Логическое НЕ (!)

1. Логическое **НЕ** сначала приводит аргумент к логическому типу **true/false**.
2. Затем возвращает противоположное значение.

```
var result = !value;
```

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

Оператор «?»

```
условие ? значение1 : значение2
```

Иногда нужно в зависимости от условия присвоить переменную. Тогда можно использовать **if**.

```
var access;
var age = prompt('How old are you?', '');

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

alert(access);
```

Такой код будет компактнее если его переписать с помошью оператора **?**.

```
var access;
var age = prompt('How old are you?', '');

access = (age > 18) ? true : false;

alert(access);
```

Конкретно в этом случае даже так:

```
access = age > 18;
```

Задача 5

Переписать предыдущую программу используя оператор **?** (знак вопроса)

Конструкция switch

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

  case 'value2': // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```

Работает это так:

- Переменная x проверяется на строгое равенство первому значению **value1**, затем второму **value2** и так далее.
- Если соответствие установлено — **switch** начинает выполняться от соответствующей директивы case и далее, до ближайшего **break** (или до конца **switch**).
- Если ни один case не совпал — выполняется (если есть) вариант **default**.

Конструкция switch

```
var arg = prompt("Введите arg?", "");
switch (arg) {
  case '0':
  case '1':
    alert( 'Один или ноль' );

  case '2':
    alert( 'Два' );
    break;

  case 3:
    alert( 'Никогда не выполнится' );

  default:
    alert('Неизвестное значение: ' + arg);
}
```

Этот switch будет работать так:

- При вводе **`0`** выполнится первый **alert**, далее выполнение продолжится вниз до первого break и выведет второй `alert( 'Два' )`. Итого, два вывода **alert**.
- При вводе **`1`** произойдёт то же самое.
- При вводе **`2`**, switch перейдет к case **`'2'`**, и сработает единственный `alert( 'Два' )`.
- При вводе **`3`**, **switch** перейдет на **default**. Это потому, что prompt возвращает строку **`'3'`**, а не число. Типы разные. Оператор **switch** предполагает строгое равенство **`===`**, так что совпадения не будет.

Циклы

При написании скриптов зачастую встает задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода — предусмотрены циклы.

Цикл while и do while

```
while (условие) {
  // код, тело цикла
}
```

Пока условие верно — выполняется код из тела цикла.

```
var i = 0;

while (i < 3) {
  alert( i );
  i++;
}
...
```

Если бы `i++` в коде слева не было, то цикл выполнялся бы (в теории) вечно. На практике, браузер выведет сообщение о **«зависшем»** скрипте и посетитель его остановит.

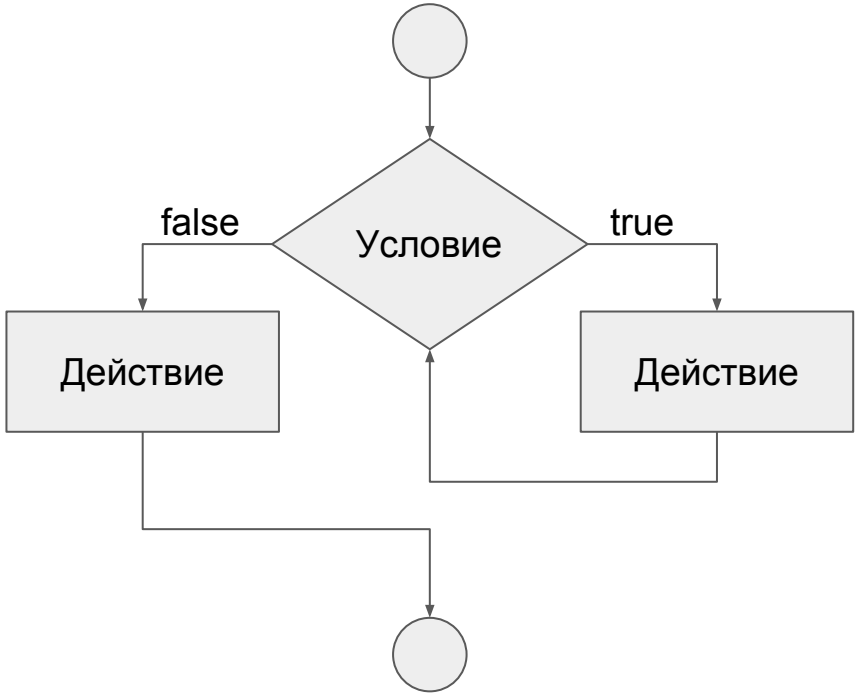
Повторение цикла называется **«итерация»**.

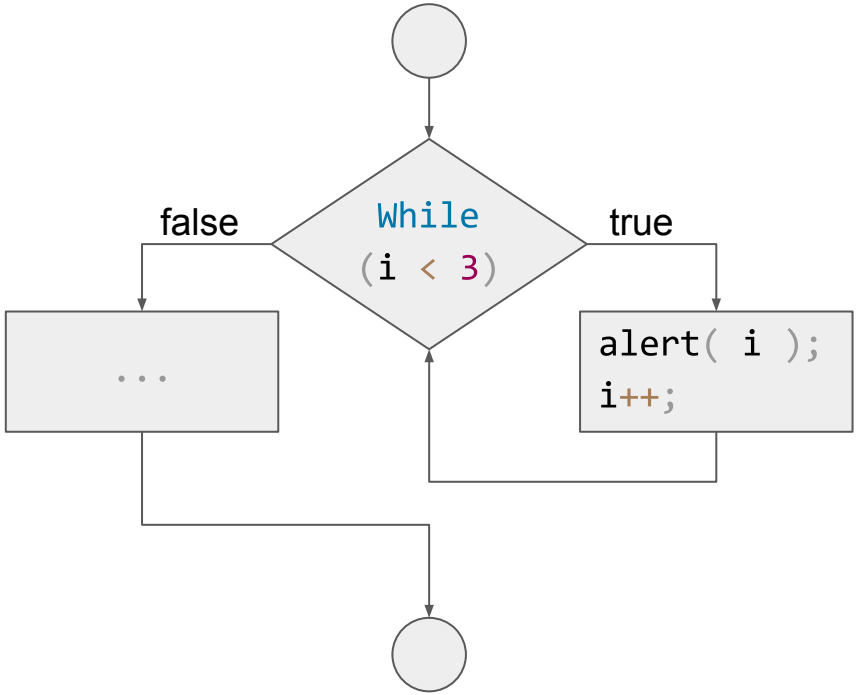
```
do {
  // тело цикла
} while (условие);
```

Цикл, описанный, таким образом, сначала выполняет тело, а затем проверяет условие.

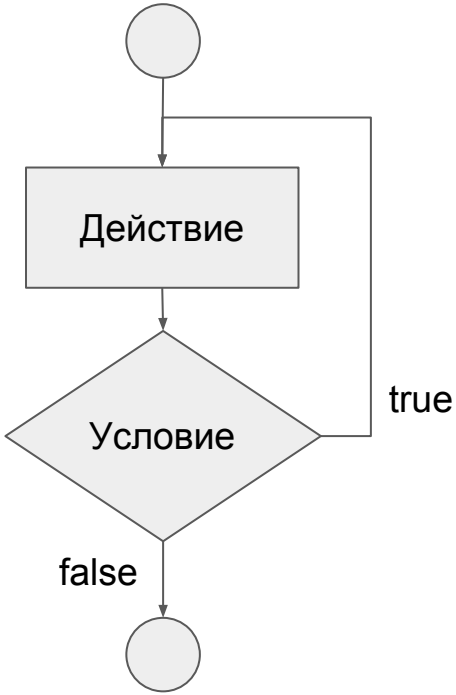
```
var i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Блок схема цикла while





Блок схема do while



Задача 6

Используя циклы, вывести в консоль столбик чисел: 1, 1.1, 1.2, 1.3, …, 2

Цикл for

Наиболее распространен цикл **for**.

```
for (начало; условие; шаг) {
  // ... тело цикла ...
}
```

Пример:

```
for (var i = 0; i < 3; i++) {
  alert( i );
}
```

Любая часть for может быть пропущена

```
var i = 0;

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

```
var i = 0;

for (; i < 3;) {
  alert( i );
  // цикл превратился в аналог while (i<3)
}
```

```
for (;;) {
  // будет выполняться вечно
}
```

Задача 7

Составить программу, которая выведет в консоль таблицу умножения на 5.

```
5 * 1 = 5
```

```
...
```

Break и continue

Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент. Эту возможность обеспечивает директива **break**.

```
var sum = 0;
while (true) {
  var value = +prompt("Enter number", '');

  if (!value) break;

  sum += value;
}
alert('Sum: ' + sum );
```

Директива **continue** прекращает выполнение **текущей итерации цикла**.

```
for (var i = 0; i < 10; i++) {

  if (i % 2 == 0) continue;

  alert(i);
}
```

Инструменты для отладки кода

Отладчик позволяет:

- Останавливаться на отмеченном месте (**breakpoint**) или по команде **debugger**.
- Выполнять код по одной строке или до определённого места.
- Смотреть переменные, выполнять команды в консоли и т.п.

Консоль

- Позволяет запускать комманды.
- Выводит ошибки.

https://learn.javascript.ru/debugging-chrome

Отладчик (панелька Sources)

Зона исходных файлов. В ней находятся все подключённые к странице файлы, включая JS/CSS.

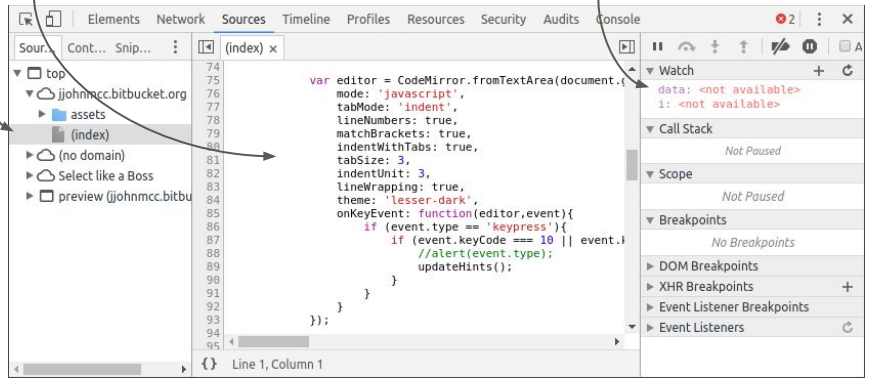
Зона текста. В ней находится текст файлов.

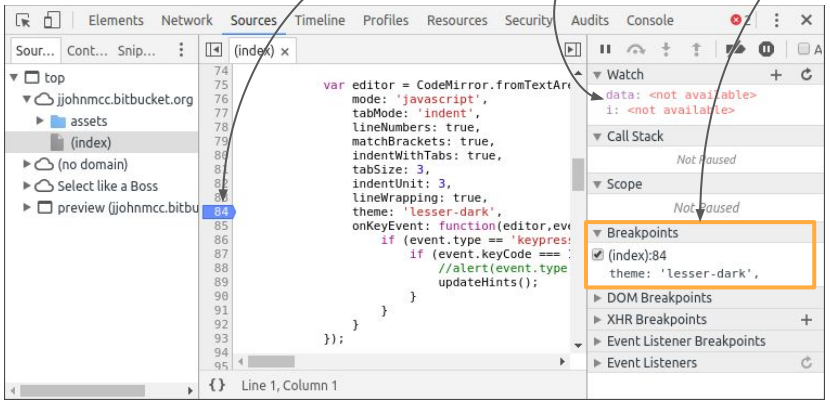
Зона информации и контроля. Мы поговорим о ней позже.

Точки останова или брейкпоинты (**breakpoint**)

Просмотр значений переменных

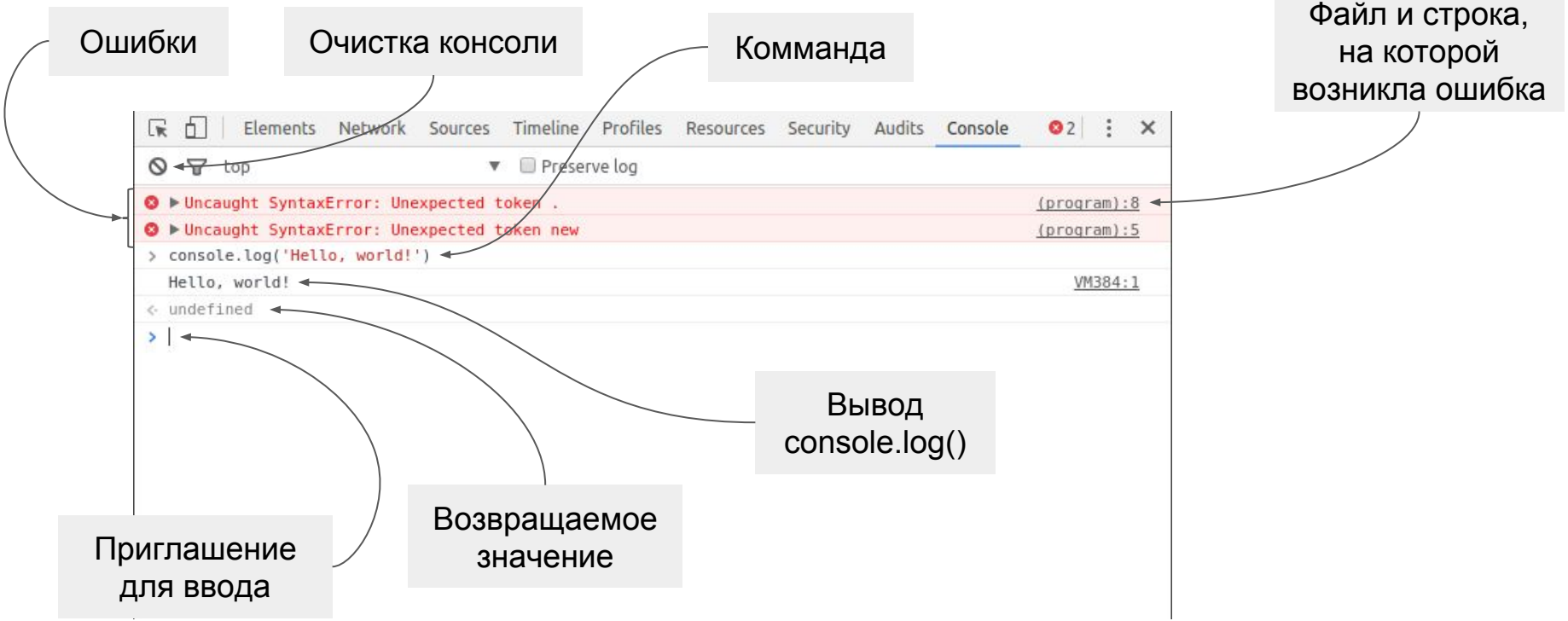
Клик





Education Academy

Консоль



Education Academy

Вопросы?