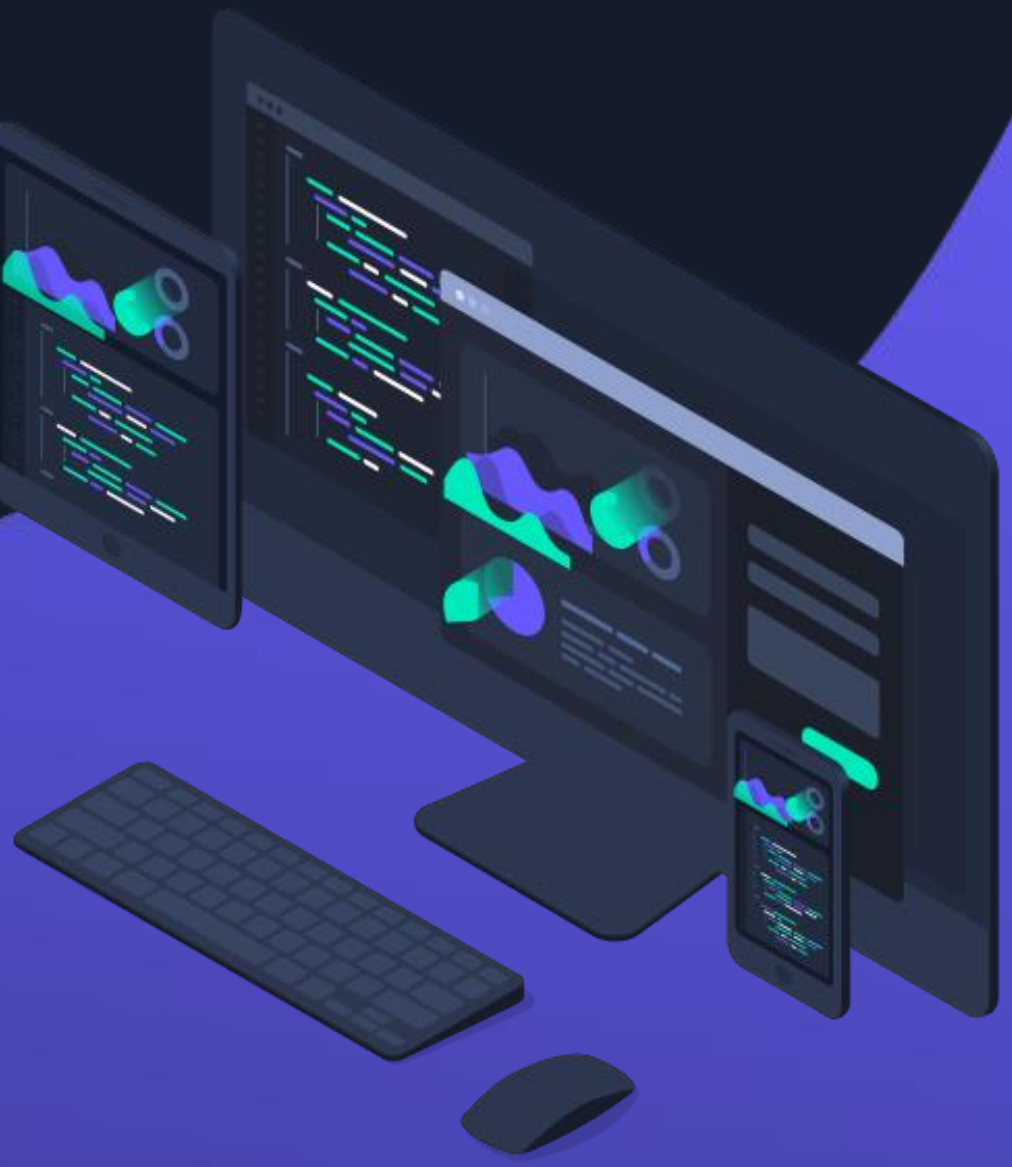
ITEA

ONLINE

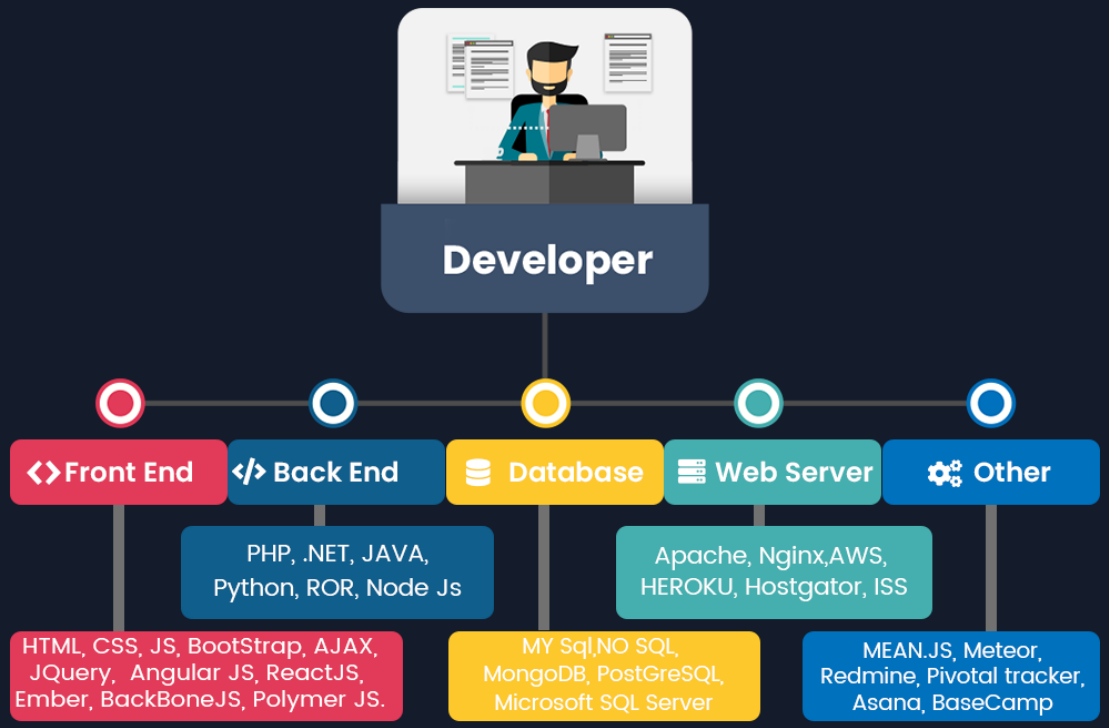


I T E A
ONLINE

На сегодняшнем занятии:

1. Условная конструкция
2. Условный оператор if
3. Условный оператор if…else
4. Условный оператор if…else if…else
5. Тернарный оператор
6. Оператор switch
7. Циклы
8. Цикл с предусловием while
9. Цикл с постусловием do…while
10. Цикл со счетчиком for
11. Операторы break и continue
12. Логические операторы
13. GitHub

Условная конструкция



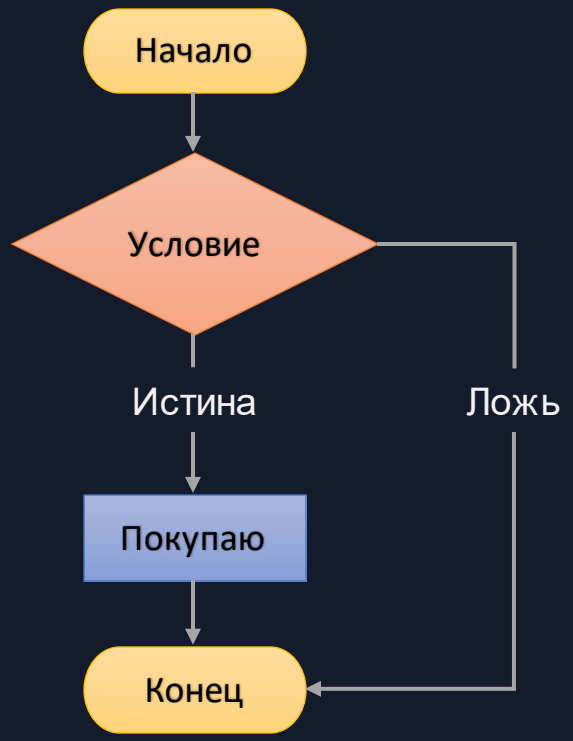
Оператор ветвления (условная инструкция, условный оператор) — оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

Разновидности условных конструкций:

- Условный оператор: `if … else`
- Тернарный оператор: `… ? … : …`
- Оператор многозначного выбора: `switch`

ITEA ONLINE

Условный оператор if



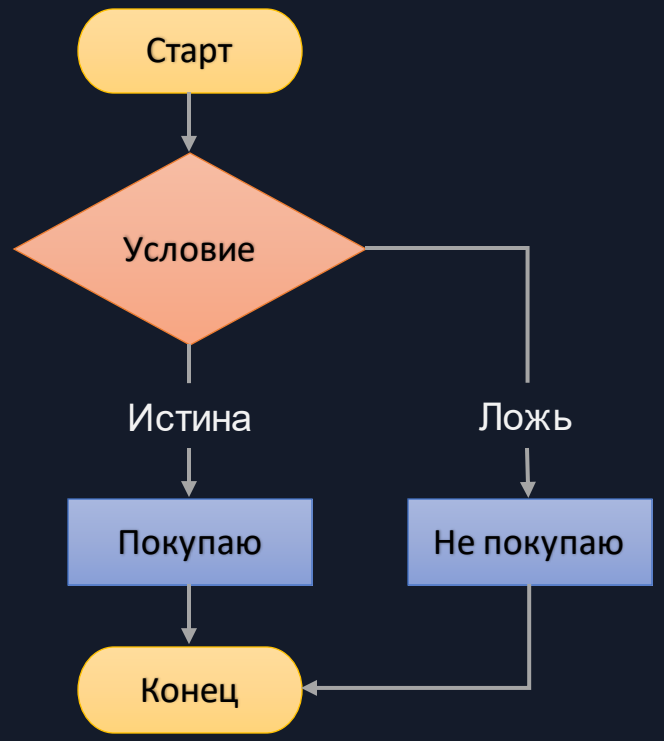
Оператор `if` проверяет переданное ему условие: если результат — true, то выполняется команда(серия команд) в операторных скобках.

```
var price = 150;
if (price < 200) {
    console.log("Покупаю");
}
```

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»).

Условный оператор if…else

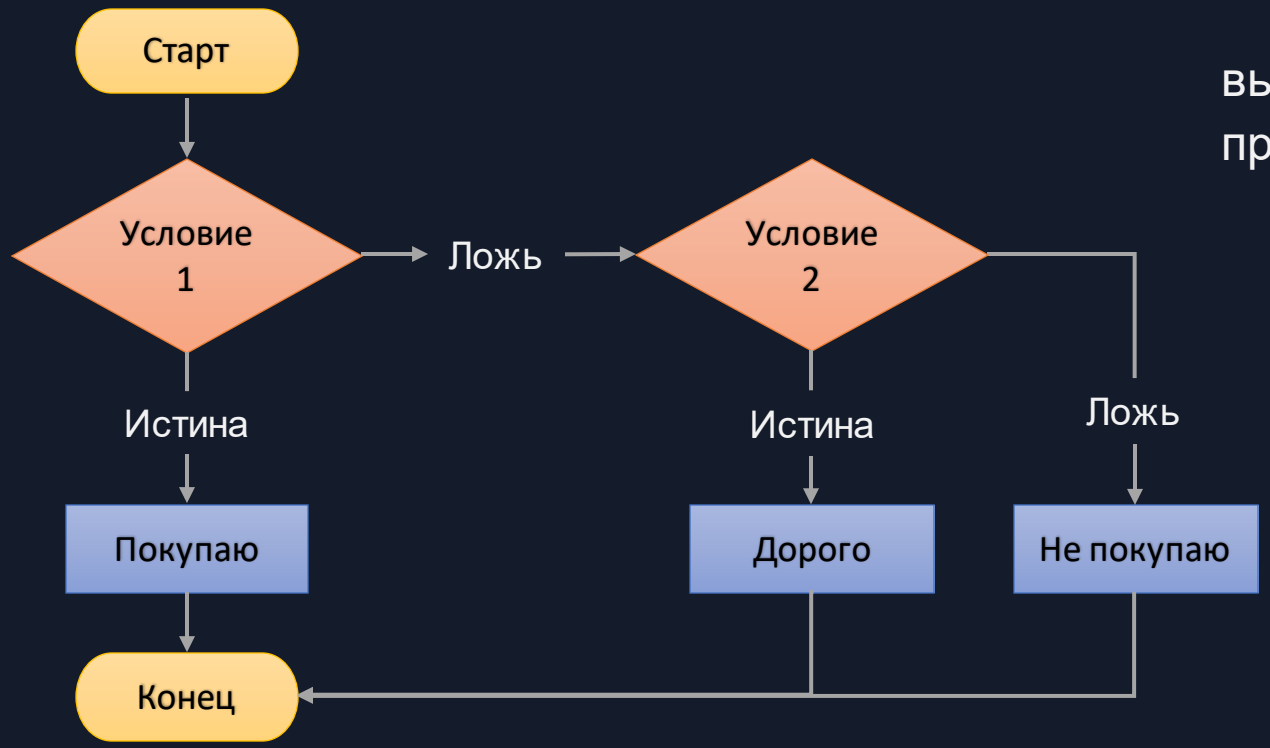


Необязательный блок else оператора if выполняется в том случае, если условие ложное (false).

```
var price = 350;
if (price < 200) {
    console.log("покупаю");
} else {
    console.log("не покупаю");
}

console.log("end");
```

Условный оператор if…else if…else



Необязательный блок else if оператора if выполняется в том случае, если условие предыдущего if было ложное (false).

```
var price = 150;
if (price < 200) {
    console.log("покупаю");
} else if (price < 300) {
    console.log("дорого");
} else {
    console.log("не покупаю");
}
```

ITEA ONLINE

Тернарный оператор

Тернарный оператор имеет следующий вид:

```
условие ? значение_true : значение_false
```

Он проверяет условие, затем, если оно истинно, возвращает «значение1» , если ложно — «значение2».

```
var result = (price < 200) ? “покупаю” : “не покупаю”
```

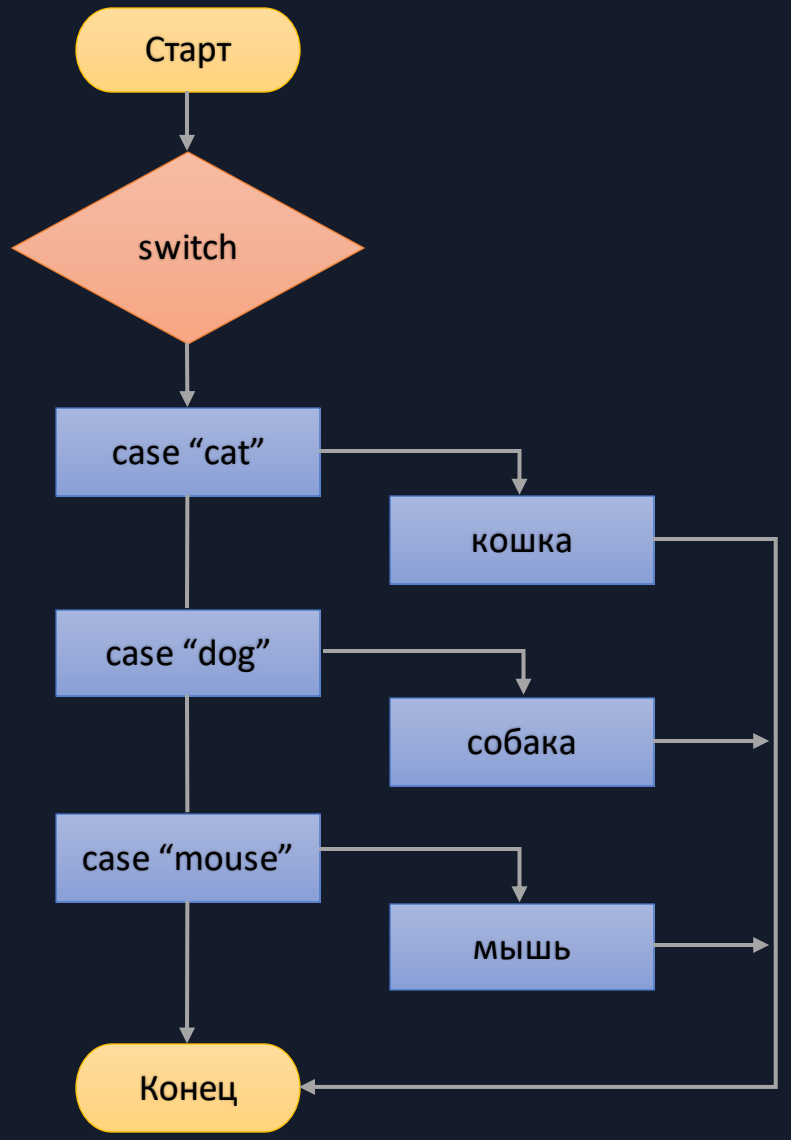
```
if (price < 200) {
    console.log(“покупаю”);
} else {
    console.log(“не покупаю”);
}
```

I T E A
ONLINE

Оператор switch

Конструкция switch заменяет собой сразу несколько if. Применяется в том случае, если нужно сравнить выражение сразу с несколькими вариантами.

```
switch(wordToTranslate) {
    case "cat": // строгое сравнение
        console.log("кошка"); break;
    case "dog":
        console.log("собака"); break;
    case "mouse":
        console.log("мышь"); break;
   default:
        console.log("нет слова в словаре");
}
```



ITEA ONLINE

Циклы

Цикл — разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций. Также циклом может называться любая многократно исполняемая последовательность инструкций, организованная любым способом (например, с помощью условного перехода).

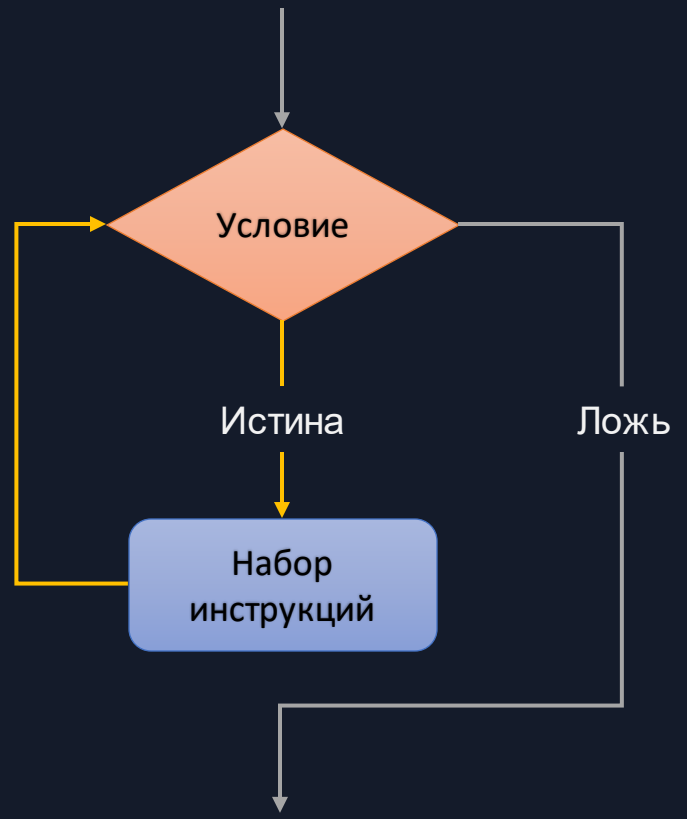
Разновидности циклических конструкций:

- Цикл с предусловием: while
- Цикл с постусловием: do…while
- Цикл со счётчиком: for

Цикл с предусловием while

Цикл while выполняет набор инструкций до тех пор, пока условие удовлетворяет истинности.
Итерация — одно повторение цикла.

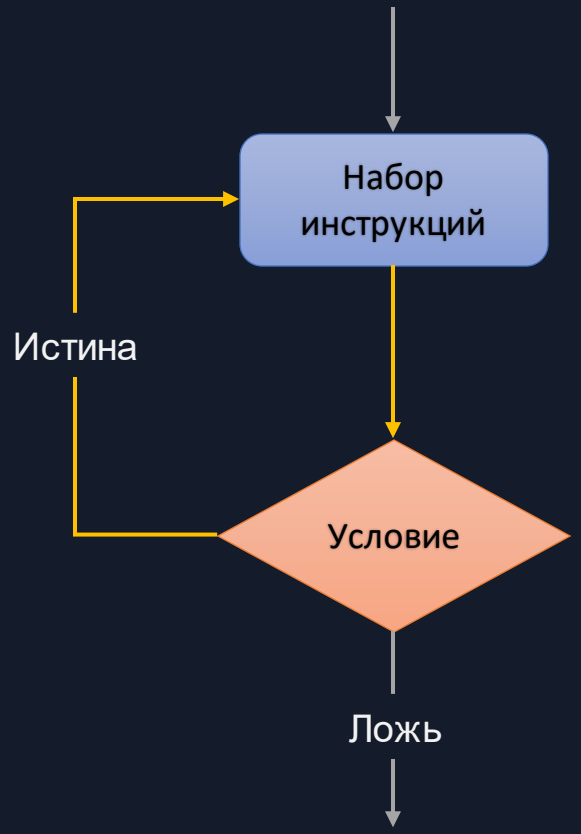
```
var a = 0;
while (a > 10) {
    console.log(a);
    a++;
}
```



Цикл с постусловием do…while

Цикл do…while вначале выполняет набор инструкций, а затем проверяет истинность условия.

```
var a = 6;
do {
    console.log(a); // a: 6, 7, 8, 9
    a++;
} while (a < 10)
```



Цикл со счетчиком for

Цикл for имеет вид: `for (счетчик; условие; шаг) { тело цикла }`, и выполняется с определенным шагом до тех пор, пока счетчик удовлетворяет условию.
Любая часть for может быть пропущена, однако сами точки с запятой обязательно должны присутствовать, иначе будет ошибка синтаксиса.

```
for (var i = 0; i < 10; i++) {
    console.log(i);
}
// here
```

Операторы break и continue

Операторы `break` и `continue` используются для более гибкого управления циклом:

- `break` — выход из цикла при необходимых условиях.
- `continue` — прекращение выполнения текущей итерации цикла и переход к следующей.

```
16      for (var i = 0; i < iteration_count; i++) {
17          if (i == exit) {
18              console.log("Работа цикла завершена принудительно");
19              console.log("***");
20              break;
21          }
22          else {
23              console.log(i);
24          }
25      }
```

```
16      for (var i = 0; i < count; i++) {
17          if (i == exit) {
18              console.log("[]");
19              continue;
20          }
21          else {
22              console.log(i);
23          }
24      }   console.log("test: " + i)
25
```

count = 10;
exit = 5

ITEA ONLINE

Логические операторы

В JavaScript поддерживаются операторы || (ИЛИ), && (И) и ! (НЕ).

- Логическое ИЛИ: если хотя бы один из аргументов true, то возвращает true, иначе — false.
- Логическое И: возвращает true, если оба аргумента истинны, а иначе — false.
- Логическое НЕ: приводит аргумент к логическому типу и возвращает противоположное значение.

```
if (1 || 0) { // работает как if( true || false )
   alert('truthy!');
}

if (price > 100 && price < 200) { // вычисляется как true && false
  alert("не сработает, так как результат ложный");
}
```

I T E A
ONLINE

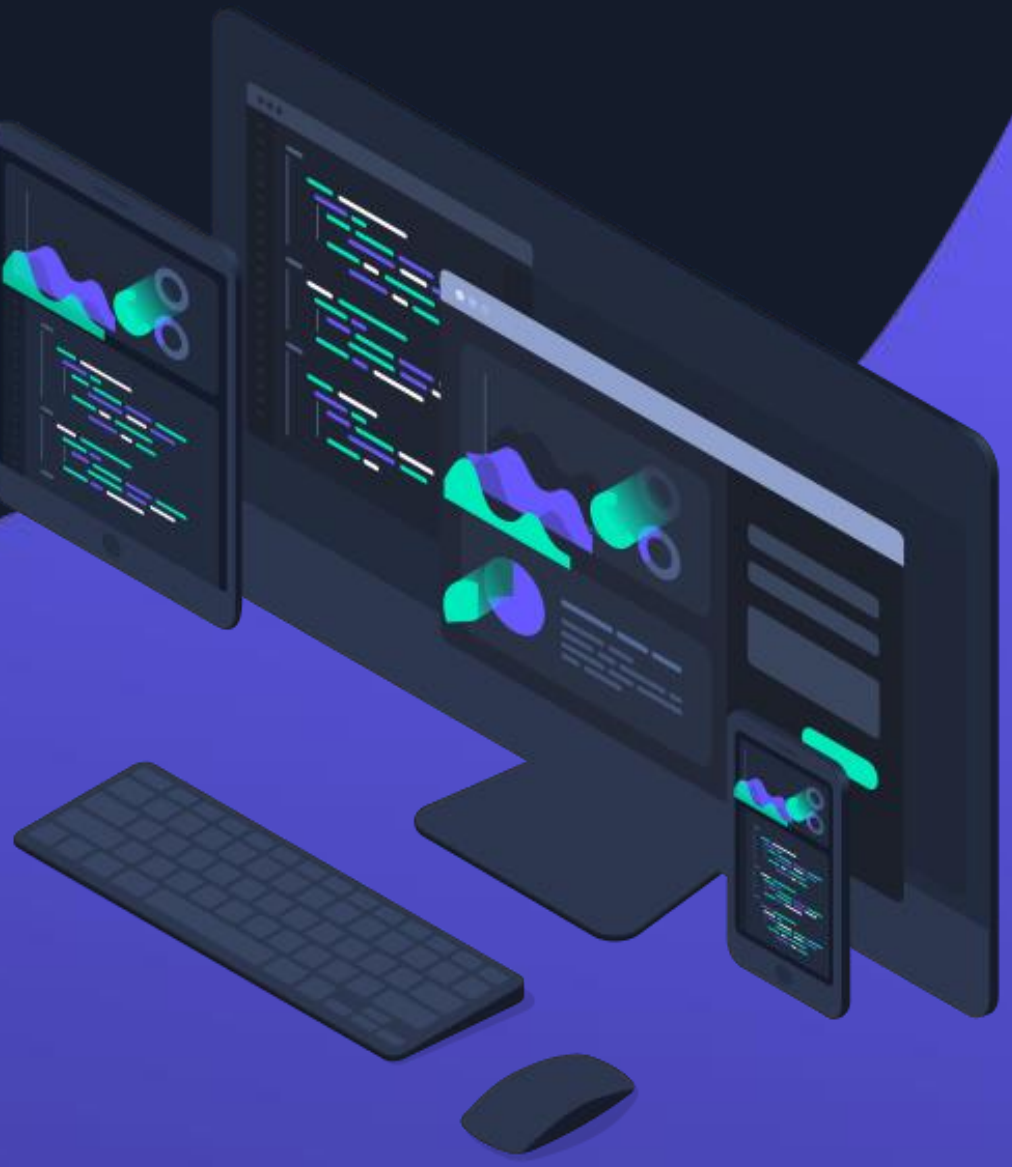
Система контроля версий. GitHub

Основные команды

- git init
- git clone
- git checkout
- git add
- git status
- git commit
- git push
- git pull
- git revert
- git merge

ITEA

ONLINE



ITEA ONLINE

Q&A