# Тема 5. Условные конструкции.


хекслет колледж

# Цель занятия:

Сформировать у студентов понимание условных конструкций в JavaScript, научить принимать решения в программе с помощью if / else, switch и тернарного оператора.

# Учебные вопросы:

1. Понятие условия в программе
2. Условная конструкция if
3. Конструкция if ... else
4. Конструкция else if
5. Логические операторы в условиях
6. Конструкция switch
7. Тернарный оператор (?:)

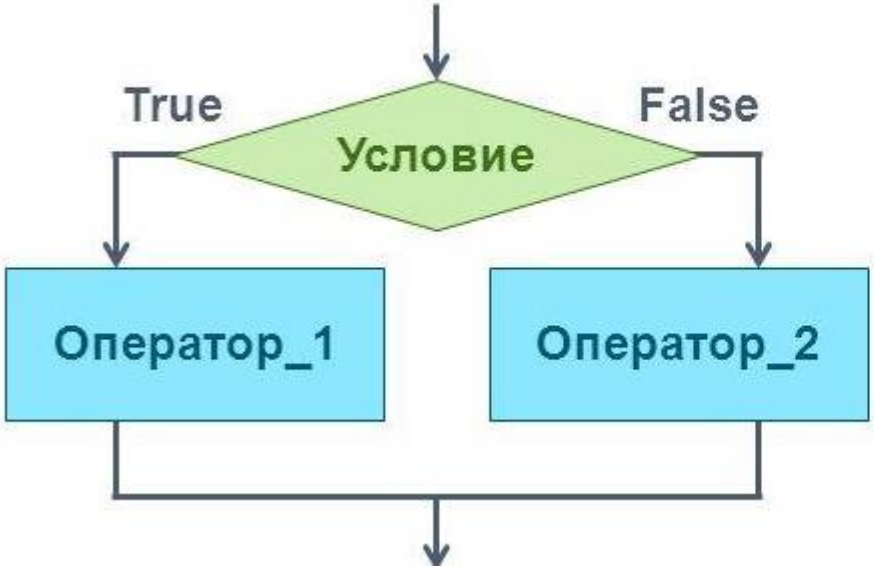
# 1. Понятие условия в программе

Что такое условие?

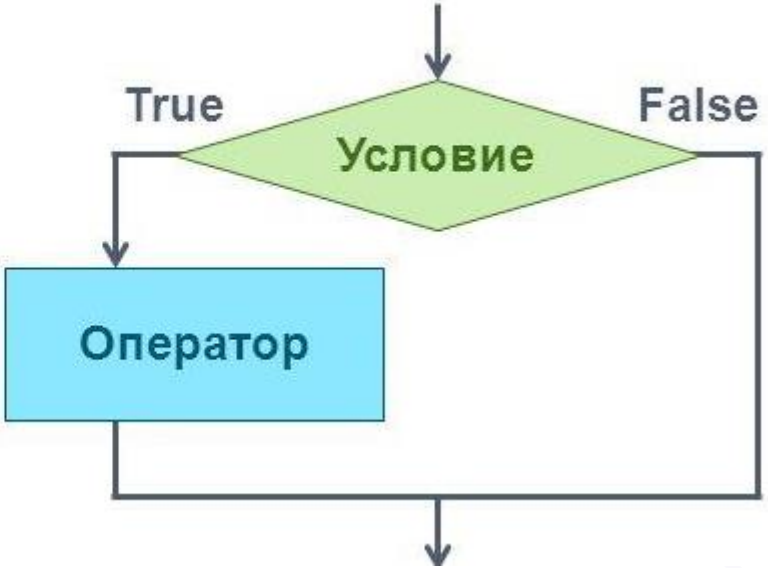
Условие — это логическое выражение, результатом которого является одно из двух значений:

- true (истина)
- false (ложь)

## Полная форма:




## Краткая форма:



На основе результата условия программа принимает решение:

- выполнять определённый участок кода
- или пропустить его

Именно условия делают программы «умными» и позволяют им по-разному реагировать на данные пользователя, состояние системы или значения переменных.

# Зачем нужны условия?

Без условий программа выполнялась бы строго последовательно, без возможности выбора.
Условия позволяют:

- реагировать на ввод пользователя;
- проверять данные перед обработкой;
- управлять логикой интерфейса;
- выполнять разные действия в зависимости от ситуации.

# Логические выражения

Условие строится на основе логического выражения — выражения, которое можно проверить на истинность. Примеры логических выражений:

```
5 > 3       // true
10 === 5    // false
age >= 18   // true или false
```

Результатом любого логического выражения в JavaScript всегда является **true** или **false**.

# Вывод:

Условие — это проверка, дающая ответ да / нет.
Оно основано на логическом выражении.
Результатом условия всегда является true или false.
Условия используются для управления логикой выполнения программы.

# 2. Условная конструкция if

Условная конструкция if используется для выполнения блока кода только при выполнении заданного условия.

- Если условие истинно (true) — код выполняется.
- Если условие ложно (false) — код пропускается.

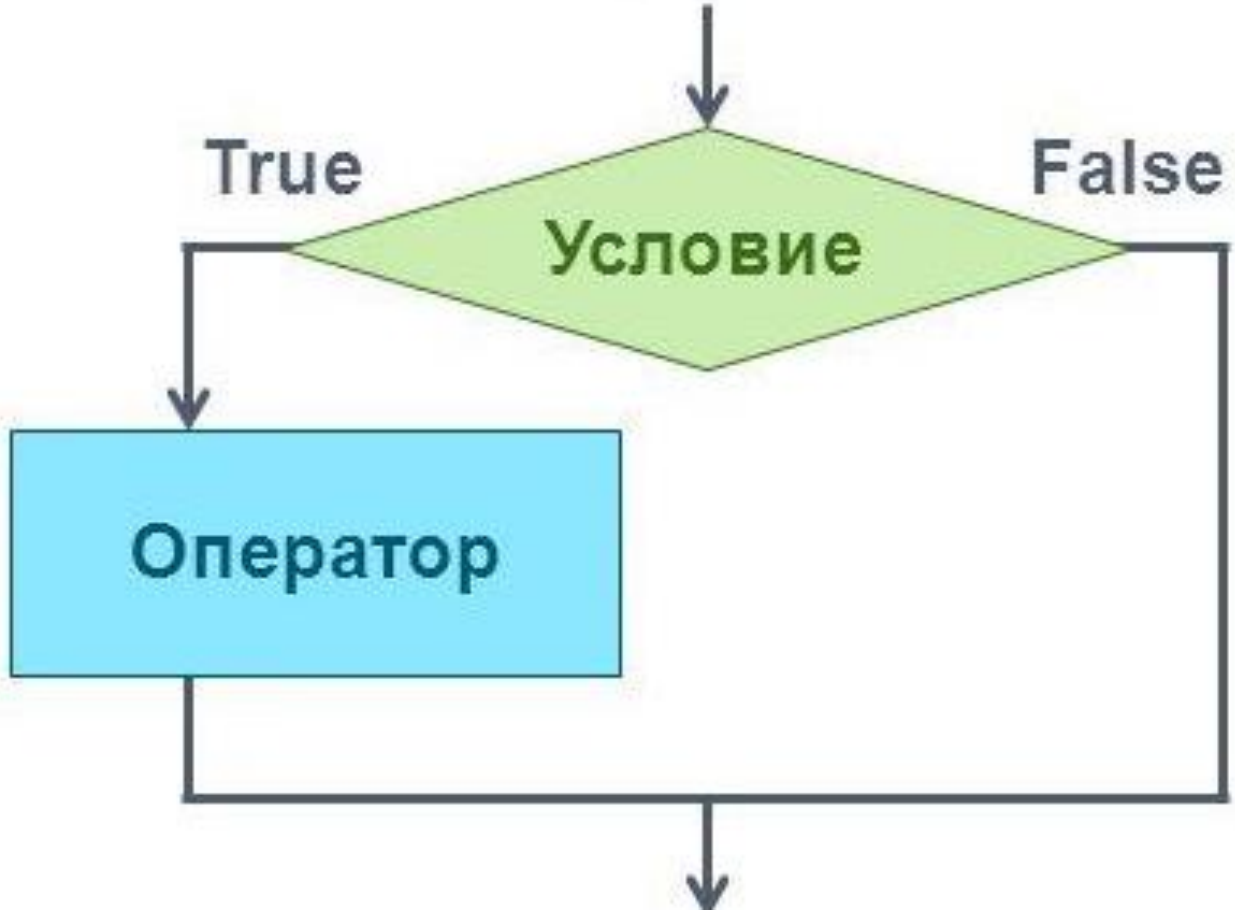
Синтаксис конструкции if:

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

- условие — логическое выражение;
- код внутри фигурных скобок называется телом условия.

# Принцип работы

JavaScript вычисляет условие в круглых скобках.
Если результат — true, выполняется код внутри { }.
Если результат — false, код внутри блока if не выполняется вовсе.



# Пример простого условия

```
let age = 18;

if (age >= 18) {
    console.log("Доступ разрешён");
}
```

Если значение **age** равно 18 или больше, сообщение появится в консоли.

# Использование переменных в условиях

Часто в условиях используются переменные:

```
let isLoggedIn = true;

if (isLoggedIn) {
    console.log("Пользователь вошёл в систему");
}
```

Если переменная имеет значение true, условие считается выполненным.

# Условия с выражениями

Внутри if может быть любое логическое выражение:

```
let temperature = 25;

if (temperature > 20) {
    console.log("Тепло");
}
```

Главное правило: результат выражения должен быть true или false.

# Фигурные скобки и их значение

Фигурные скобки {} определяют границы блока if.

Если в блоке одна инструкция, скобки можно опустить, но в учебной практике это не рекомендуется.

Пример (допустимо, но нежелательно):

```
if (age >= 18)
    console.log("Совершеннолетний");
```

Рекомендуемый вариант:

```
if (age >= 18) {
    console.log("Совершеннолетний");
}
```

# Вывод:

- if — базовая конструкция для принятия решений.
- Выполняет код только при истинном условии.
- Основана на логических выражениях.
- Является основой всей условной логики в JavaScript.

# 3. Конструкция if … else

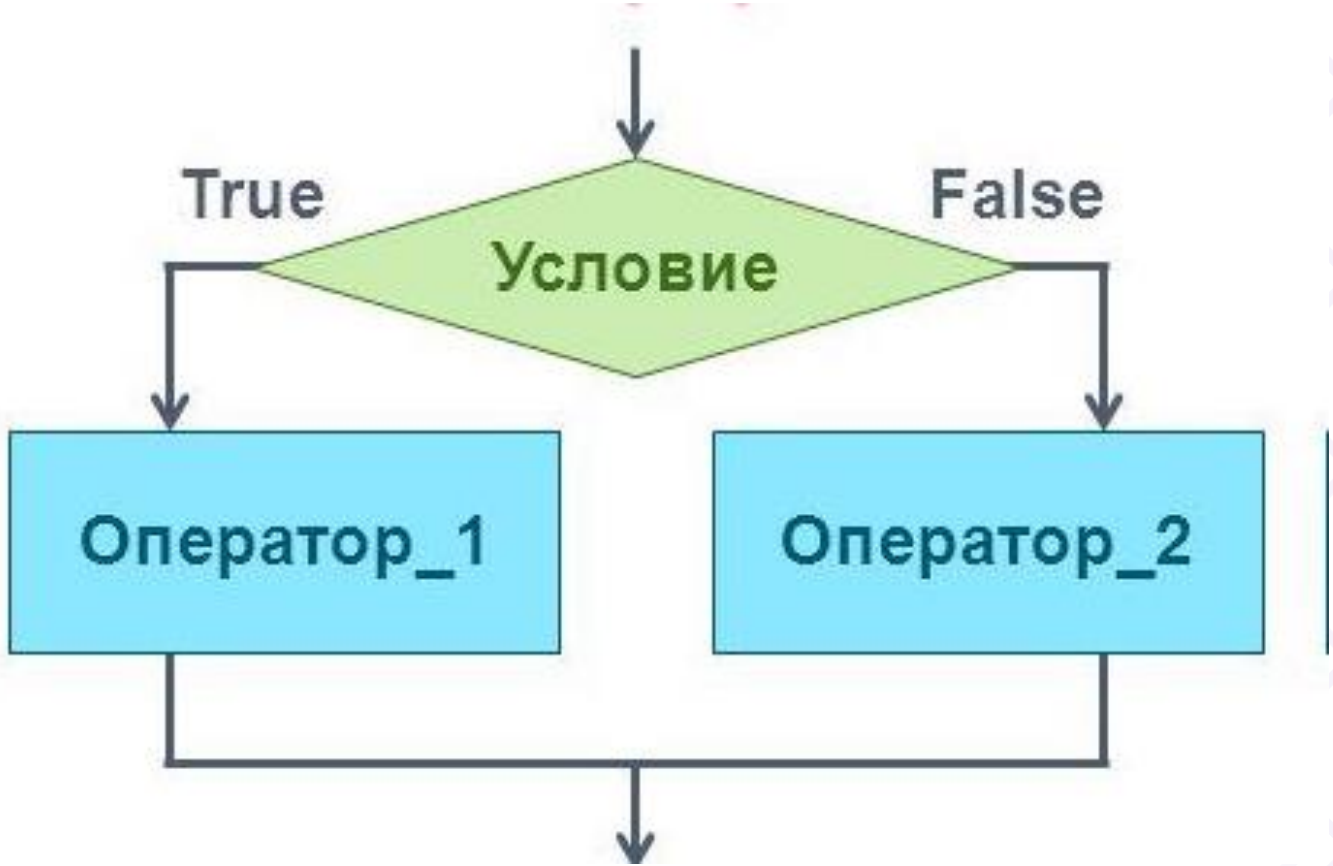
Конструкция if … else используется, когда программе нужно выбрать одно из двух действий:

- одно — если условие истинно;
- другое — если условие ложно.

Таким образом, выполнение кода происходит в любом случае, но по разному сценарию.

# Синтаксис if ... else

```
if (условие) {
    // код выполняется, если условие true
} else {
    // код выполняется, если условие false
}
```



# Как работает if ... else

- JavaScript проверяет условие.
- Если результат — true, выполняется блок if.
- Если результат — false, выполняется блок else.
- Одновременно выполняется только один из блоков.

# Пример:

```
let age = 16;

if (age >= 18) {
    console.log("Доступ разрешён");
} else {
    console.log("Доступ запрещён");
}
```

В зависимости от значения переменной age программа выведет одно из двух сообщений.

# Использование с логическими выражениями

```
let temperature = 10;

if (temperature > 20) {
    console.log("Тепло");
} else {
    console.log("Холодно");
}
```

Условие может быть любым логическим выражением.

# Вложенное условие:

```
let age = 20;
let hasTicket = true;

if (age >= 18) {
    if (hasTicket) {
        console.log("Проход разрешён");
    } else {
        console.log("Нет билета");
    }
} else {
    console.log("Возрастное ограничение");
}
```

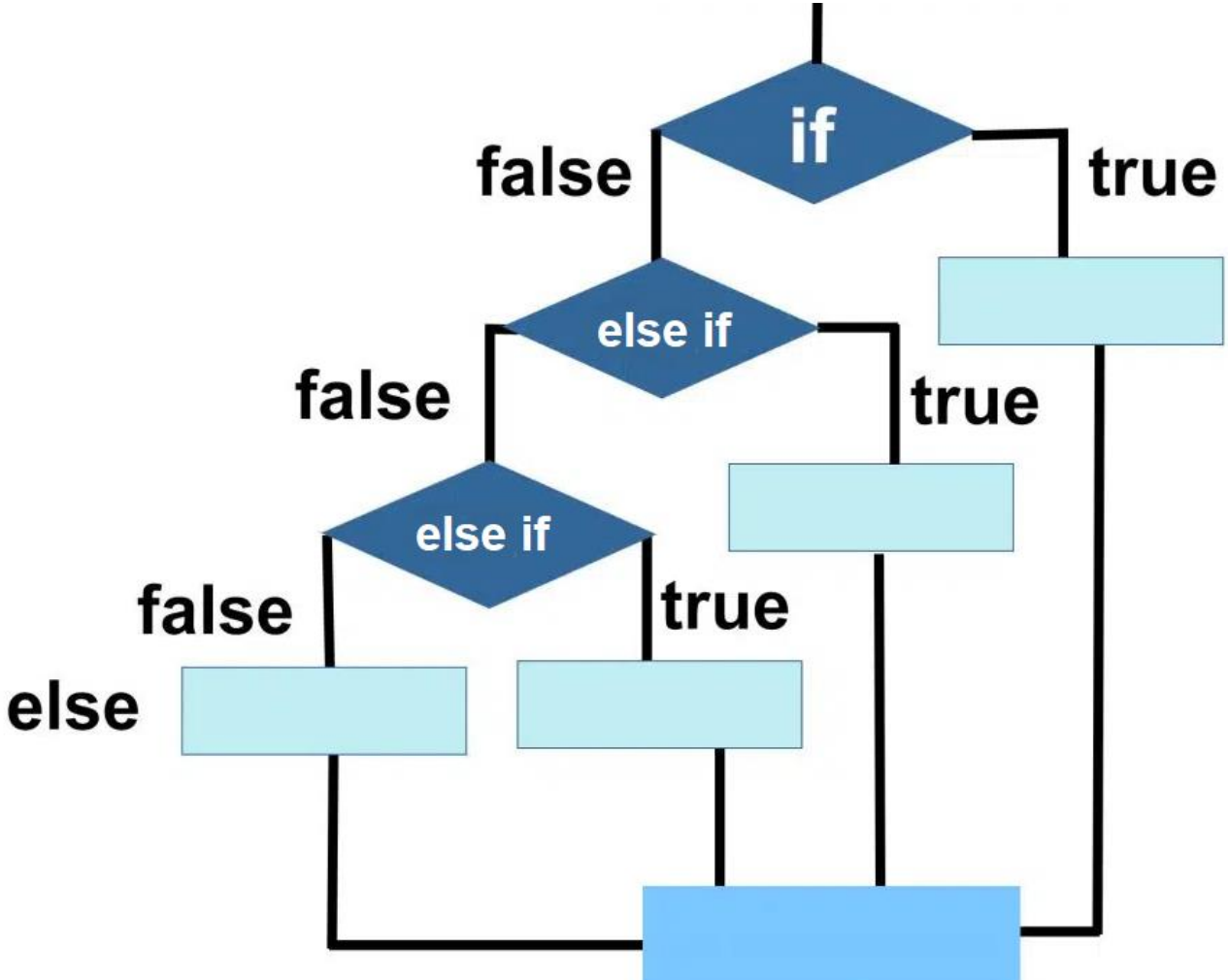
Однако при большом количестве условий такой код быстро теряет читаемость.

# Вывод:

- if ... else позволяет реализовать альтернативную логику.
- Используется, когда есть два возможных варианта действий.
- Поддерживает вложенность, но требует аккуратного использования.
- Широко применяется при проверке данных и управлении логикой интерфейса.

# 4. Конструкция else if

Конструкция else if используется, когда необходимо проверить несколько условий по очереди и выполнить код только для первого истинного условия.
Она применяется в ситуациях, где двух вариантов (if ... else) недостаточно.



# Общий синтаксис:

```
if (условие1) {
    // код выполняется, если условие1 true
} else if (условие2) {
    // код выполняется, если условие2 true
} else if (условие3) {
    // код выполняется, если условие3 true
} else {
    // код выполняется, если все условия false
}
```

# Принцип работы:

- Проверяется условие в if.
- Если оно ложно, проверяется первое else if.
- Проверки продолжаются сверху вниз.
- Как только найдено истинное условие — выполняется соответствующий блок.
- Остальные условия не проверяются.

# Пример:

```
let score = 75;

if (score >= 90) {
    console.log("Отлично");
} else if (score >= 70) {
    console.log("Хорошо");
} else if (score >= 50) {
    console.log("Удовлетворительно");
} else {
    console.log("Не сдал");
}
```

# Важность порядка условий

Порядок условий имеет критическое значение:

```
// Неправильно
if (score >= 50) {
    console.log("Удовлетворительно");
} else if (score >= 90) {
    console.log("Отлично");
}
```

Более общее условие не должно идти раньше более строгого.

# Использование без else

Конструкция **else** является необязательной:

```
if (temp > 30) {
    console.log("Жарко");
} else if (temp < 0) {
    console.log("Мороз");
}
```

Если ни одно условие не выполнится — код просто не будет выполнен.

# Когда использовать **else if**?

- при выборе диапазонов значений;
- при проверке вариантов (оценки, роли, статусы);
- когда условий больше двух, но они взаимоисключающие.

# Вывод:

- **else if** позволяет реализовать множественный выбор.
- Проверки выполняются последовательно сверху вниз.
- Выполняется только один блок кода.
- Важно правильно располагать условия.

# 5. Логические операторы в условиях

Логические операторы используются для:

- объединения нескольких условий;
- отрицания условий;
- построения сложной логики принятия решений.

Логические операторы позволяют объединять несколько условий в одно выражение.

Это помогает:

- создавать более компактный и читаемый код,
- уменьшать количество вложенных if,
- управлять сложной логикой с меньшим количеством блоков кода.

# Основные логические операторы в JavaScript

**&&** - логическое И (AND),
**||** - логическое ИЛИ (OR),
**!** - логическое НЕ (NOT).

| A | B | A&&B | A\|\|B | !A |
|---|---|---|---|---|
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

# Пример: упрощение кода

Без логического оператора (вложенные условия):

```
let age = 20;
let hasTicket = true;

if (age >= 18) {
    if (hasTicket) {
        console.log("Проход разрешён");
    }
}
```

С логическим оператором && (код проще и без вложенности):

```
if (age >= 18 && hasTicket) {
    console.log("Проход разрешён");
}
```

# Пример: Оператор ИЛИ (||):

```
let isAdmin = false;
let isModerator = true;

if (isAdmin || isModerator) {
    console.log("Доступ разрешён");
}
```

# Пример: Оператор НЕ (!):

```
let isLoggedIn = false;

if (!isLoggedIn) {
    console.log("Пользователь не авторизован");
}
```

# Приоритет логических операторов

**!** — высокий приоритет
**&&** — средний
**||** — низкий

Для ясности используйте скобки:

```
if ((age >= 18 && hasTicket) || isAdmin) {
    console.log("Доступ разрешён");
}
```

# Вывод:

- Логические операторы позволяют объединять условия, уменьшать вложенность if, делать код проще и читаемее.
- Они являются основой сложной логики в программах.
- Всегда следите за приоритетом операторов и используйте скобки для ясности.

# 6. Конструкция switch

**switch** — это конструкция для множественного выбора, когда необходимо проверить одну переменную или выражение на несколько возможных значений.
Она упрощает код по сравнению с цепочкой **if ... else if**.

Синтаксис:

```
switch (выражение) {
    case значение1:
        // код, если выражение === значение1
        break;
    case значение2:
        // код, если выражение === значение2
        break;
    ...
    default:
        // код, если ни одно значение не подошло
}
```

выражение — значение, которое проверяется.
case — отдельный вариант.
break — завершает выполнение после совпадения, предотвращая “провал” к следующему case.
default — выполняется, если ни одно значение не совпало (необязательный).

# Принцип работы:

- JavaScript вычисляет выражение в **switch**.
- Сравнивает с каждым **case** с помощью строгого сравнения (**===**).
- Выполняется код первого совпавшего **case**.
- Если нет совпадений — выполняется блок **default** (если он есть).

# Пример:

```
switch (day) {
    case 1:
        console.log("Понедельник");
        break;
    case 2:
        console.log("Вторник");
        break;
    case 3:
        console.log("Среда");
        break;
    default:
        console.log("Другой день");
}
```

Результат: "Среда"

# Особенности

- **switch** удобен для конкретных значений, а не диапазонов (для диапазонов лучше **if ... else if**).
- Если забыть **break**, выполнение перейдёт к следующему **case** (“провал” или fall-through).

# Пример провала:

```
let color = "red";

switch (color) {
    case "red":
        console.log("Красный");
    case "blue":
        console.log("Синий");
    default:
        console.log("Другой цвет");
}
```

Вывод:
Красный
Синий
Другой цвет

# Итог:

- switch облегчает множественный выбор по значению.
- case сравнивает строго (===).
- break предотвращает выполнение следующих блоков.
- default — обработка “неизвестного” варианта.

# 7. Тернарный оператор (?:)

Тернарный оператор используется для краткой записи условного выбора из двух вариантов.
Он является сокращённой формой конструкции **if ... else**.

Тернарный оператор особенно удобен:

- для простых проверок;
- при присваивании значений;
- когда важно сохранить компактность и читаемость кода.

# Синтаксис тернарного оператора

**условие ? выраж_если_true : выраж_если_false**

Если:

- условие истинно (**true**) — выполняется выражение после **?**
- условие ложно (**false**) — выполняется выражение после **:**

# Пример:

```
let age = 20;

let message = age >= 18 ? "Совершеннолетний" : "Несовершеннолетний";

console.log(message);
```

# Аналог с if ... else:

```
let age = 20;
let message;

if (age >= 18) {
    message = "Совершеннолетний";
} else {
    message = "Несовершеннолетний";
}
```

# Когда использовать тернарный оператор:

- выбор текста, класса, значения;
- простая проверка условия;
- компактная логика интерфейса.

# Вывод:

- Тернарный оператор — краткая форма if ... else.
- Повышает компактность кода.
- Эффективен для простых условий.
- Не подходит для сложных и вложенных проверок.

# Выводы по лекции:

- Условия являются основой логики любой программы и позволяют выполнять код выборочно.
- Условие в JavaScript строится на логических выражениях, результатом которых являются значения true или false.
- Конструкция if используется для выполнения кода при выполнении одного условия.
- Конструкция if ... else позволяет реализовать альтернативный выбор между двумя вариантами.
- Конструкция else if применяется для проверки нескольких условий по очереди и выполняет код только для первого истинного условия.
- Логические операторы (&&, ||, !) позволяют объединять условия и упрощать код, уменьшая вложенность.
- Конструкция switch удобна при проверке одного значения на несколько фиксированных вариантов.
- Использование break в switch предотвращает выполнение последующих веток условий.
- Тернарный оператор является сокращённой формой if ... else и применяется для простых условий.
- Для сложной и многоуровневой логики предпочтительно использовать if ... else, а не тернарный оператор.

# Контрольные вопросы:

- Что называется условием в программе?
- Какие значения может принимать результат логического выражения?
- Для чего используется конструкция if?
- В чём отличие конструкции if от if … else?
- Когда применяется конструкция else if?
- Какую задачу решают логические операторы в условиях?
- В чём назначение конструкции switch?
- Что произойдёт, если забыть break в блоке case?
- Что такое тернарный оператор?
- В каких случаях тернарный оператор предпочтительнее if … else?

# Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ