

Operators & Conditions

Arithmetic (+, ~/ , %), comparison and logical operators — then decisions: if / else, switch and the ternary shortcut.

Introduction

Data alone does nothing — apps must *calculate* and *decide*. Operators are Dart's calculation symbols (+, *, ~/ , % and friends), and conditions are how a program chooses a path: if marks are high, print "Grade A"; otherwise, print something else. This lesson covers the arithmetic set, the true/false operators, and the three decision tools — if / else, switch and the one-line ternary. (Roman Urdu: Operator hisaab karta hai, condition faisla — dono mil kar program ko dimagh dete hain.)

Key points

- **Arithmetic operators:** + add, - subtract, * multiply, / divide (always gives a double: 17 / 5 = 3.4), ~/ divide and keep only the whole part (17 ~/ 5 = 3), % remainder (17 % 5 = 2).
(Roman Urdu: ~/ pooro hissa deta hai, % baqi bacha hua — dono roz kaam aate hain.)
- **Comparison operators** answer with true or false: == equal, != not equal, > greater, < smaller, >= and <=. One = puts a value in a box; two == ask whether two values match. Mixing them up is the classic bug.
- **Logical operators** combine answers: && (AND — both must be true), || (OR — any one true is enough), ! (NOT — flips true to false).
- **if / else if / else** runs the first branch whose condition is true, and skips all the rest. Conditions go in (), the branch's work goes in { }.
- **switch** is cleaner than a long if chain when one value has many fixed choices (day numbers, menu options). Each case ends with break;, and default catches everything else.
- The **ternary** condition ? valueIfTrue : valueIfFalse is a one-line if/else that gives a value — perfect for small choices like age >= 18 ? 'Adult' : 'Minor'.

Code examples

Example 1 — Arithmetic: the full set on two numbers

```
void main() {  
    int a = 17;  
    int b = 5;  
  
    print(a + b); // 22  
    print(a - b); // 12  
    print(a * b); // 85  
    print(a / b); // 3.4 – hamesha double  
    print(a ~/ b); // 3 – sirf poora hissa  
    print(a % b); // 2 – baqi (remainder)  
}
```

What it does: Six prints show the six operators on the same two numbers. Comments on the right show the expected answer — always check your program against answers you already know, like a calculator you trust.

What you will see: six lines — 22, 12, 85, 3.4, 3, 2. The pair ~/ and % together split 17 into "3 groups of 5, remainder 2" — exactly how you would split 17 rotis among 5 people.

Example 2 — The grade program (if / else if / else)

```
void main() {  
    int marks = 82;  
  
    if (marks >= 90) {  
        print('Grade A');  
    } else if (marks >= 75) {  
        print('Grade B');  
    } else if (marks >= 50) {  
        print('Pass');  
    } else {  
        print('Fail');  
    }  
}
```

What it does: Dart checks the conditions **from the top**. 82 is not ≥ 90 , so the first branch is skipped; 82 is ≥ 75 , so "Grade B" prints and every branch below is ignored — even though 82 is also ≥ 50 . Order is logic: always write the biggest condition first.

What you will see: one line — Grade B. Change marks to 95 and only Grade A prints; change to 30 and only Fail prints. One run, one branch — never two.

Example 3 — switch for fixed choices + the ternary shortcut

```
void main() {
  int day = 3;
  String dayName;

  switch (day) {
    case 1:
      dayName = 'Monday';
      break;
    case 3:
      dayName = 'Wednesday';
      break;
    default:
      dayName = 'Some other day';
  }
  print(dayName);

  int age = 20;
  String result = age >= 18 ? 'Adult' : 'Minor';
  print(result);
}
```

What it does: The `switch` matches `day` against fixed cases — 3 lands on the `case 3` branch and stores 'Wednesday'. The ternary then makes a two-way choice in a single line and stores the *value* it picked in `result`.

What you will see: two lines — `Wednesday` and `Adult`. Remove any `break;` and Dart warns you — in Dart's classic switch, `break` is what stops the program from falling into the next case.

Quick recap

- `/` gives a double, `~/` gives the whole part, `%` gives the remainder.
- `=` assigns; `==` compares. Comparisons answer `true` or `false`.
- `&&` both true, `||` any true, `!` flips the answer.
- `if / else if / else` — first true branch wins; write conditions biggest first.
- `switch` for many fixed choices on one value; ternary for a one-line two-way choice.

Common mistakes

1. **Using `=` inside a condition** — `if (marks = 50)` tries to *assign* and Dart errors. A question always uses `==`. Read your condition aloud: "is marks equal to 50?" — if it sounds like a question, it needs `==`.
2. **Wrong branch order in grade-style chains** — if you check `>= 50` *before* `>= 90`, a student with 95 gets "Pass" instead of "Grade A", because the first true branch wins. Biggest and strictest conditions go on top.

3. **Forgetting break in switch** — without `break;`, Dart complains and the logic can fall through into the next case. Every case ends with `break;` — make it a habit while typing the case itself.

Practice

1. Write a program that stores a temperature and prints "Garam", "Normal" or "Thanda" using if / else if / else (above 35, 20–35, below 20).
2. Take a number in a variable and print whether it is even or odd — hint: `number % 2 == 0` means even. Use a ternary to print the answer in one line.

Watch the video lesson: <https://youtu.be/AjFMiOcuHMU>

Read online: <https://toolscope.netlify.app/notes/dart-operators-conditions> · More free tools and notes at <https://toolscope.netlify.app>