TypeScript Basics for Automation Testers – Day 6

Topic: Conditional Statements in TypeScript

What are Conditional Statements?

Conditional statements in TypeScript help us make **decisions** in our program.
They check whether a given condition is **true or false** and based on that, execute different sets of code.

In simple terms, they allow our script to **react differently** based on different inputs or situations.
For example:

- "If user is admin, show settings tab."
- "If score is above 90, print Grade A."

Types of Conditional Statements

1. if statement
2. if–else statement
3. nested if–else (if–else if) statement
4. switch–case statement

---

if Statement

The `if` statement checks a condition.
If the condition is **true**, it executes the code inside the block.
If it is **false**, the code block is skipped.

Why and Where to Use:

Used when you need to run a specific piece of code **only under certain conditions**.
Example: verifying if a user is logged in before proceeding to dashboard.

Syntax:

```
if (condition) {
  // code to execute if condition is true
}
```

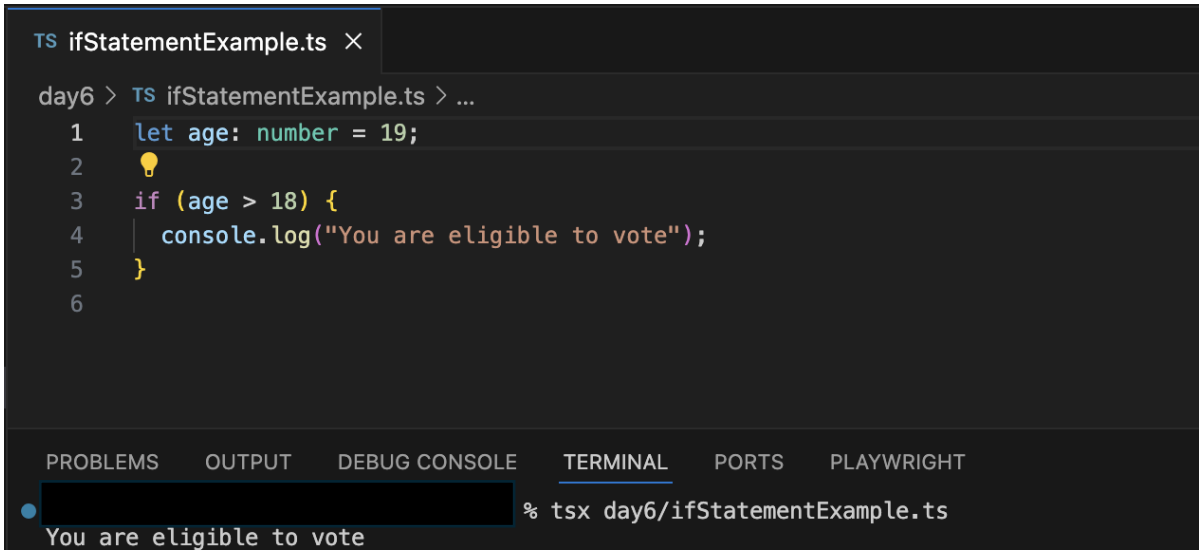
Example:

```
let age: number = 18;

if (age > 18) {
  console.log("You are eligible to vote");
}
```

Explanation:

Here, the program checks if `age` is greater than 18.
If true → prints "You are eligible to vote".
If false → nothing happens.



---

if–else Statement

The `if-else` statement provides **two possible paths** —
one block runs if the condition is true, another block runs if it's false.

Why and Where to Use:

Use this when there are **two possible outcomes**, like even/odd, pass/fail, login success/failure.

Syntax:

```
if (condition) {
  // code if condition is true
} else {
  // code if condition is false
}
```

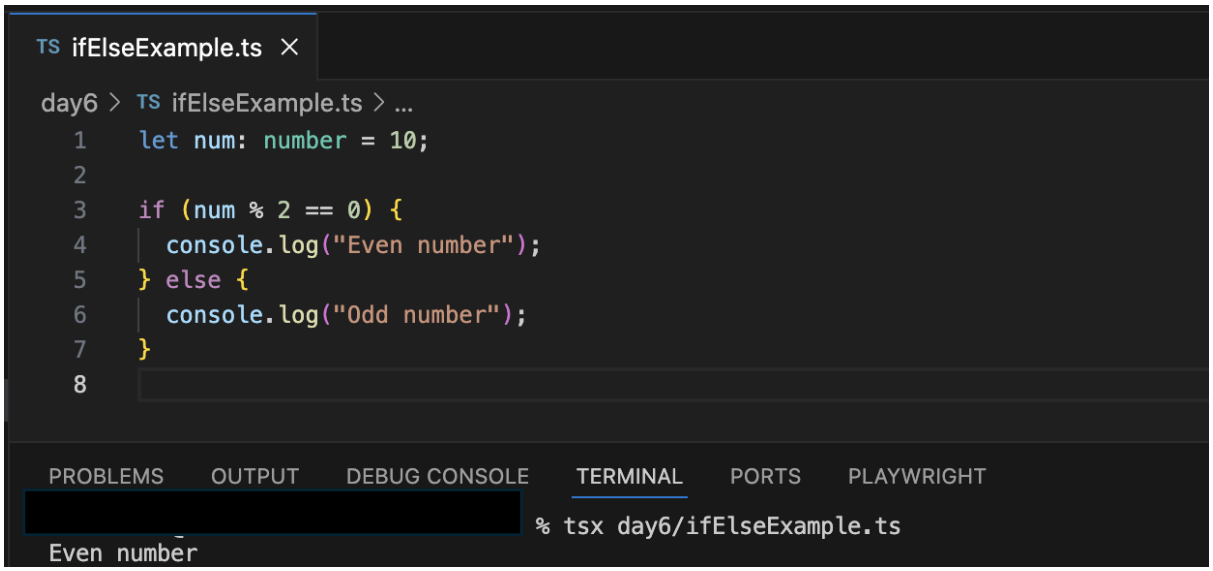
Example:

```
let num: number = 10;

if (num % 2 == 0) {
  console.log("Even number");
} else {
  console.log("Odd number");
}
```

Explanation:

Here, if `num` is divisible by 2, the condition is true and prints “Even number”.
Otherwise, “Odd number” is printed.



Nested if–else (if–else if) Statement

When you need to check **multiple conditions** in sequence, you use nested if–else.
Once one condition is true, the remaining ones are skipped.

Why and Where to Use:

Used in cases like **grading**, **eligibility checks**, or **test case categorization**.

Syntax:

```
if (condition1) {
  // code for condition1
} else if (condition2) {
  // code for condition2
} else if (condition3) {
  // code for condition3
} else {
  // code if none are true
}
```

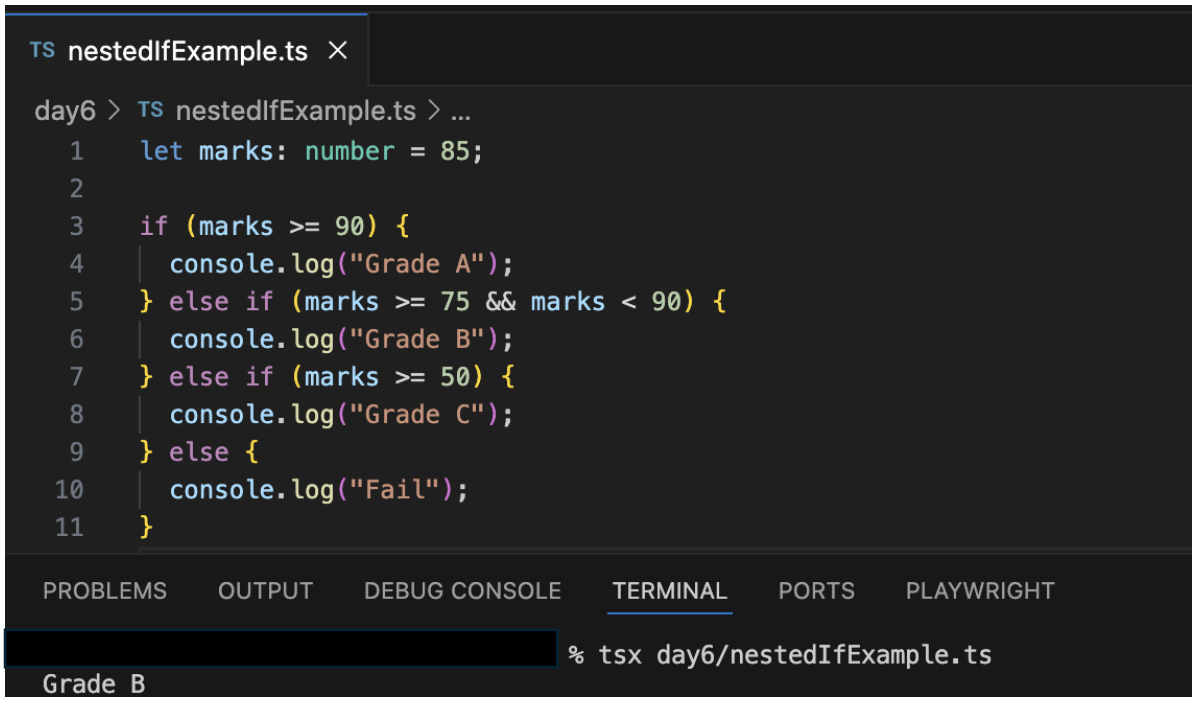
Example:

```
let marks: number = 85;

if (marks >= 90) {
  console.log("Grade A");
} else if (marks >= 75 && marks < 90) {
  console.log("Grade B");
} else if (marks >= 50) {
  console.log("Grade C");
} else {
  console.log("Fail");
}
```

Explanation:

Here, `marks = 85`, so the program prints "Grade B".
Only the **first true condition** executes, the rest are ignored.



Switch–case Statement

The `switch` statement is used to compare a single value (called **expression**) against multiple **case values**.

Why and Where to Use:

Useful when checking one variable against **many fixed options**, like days of week, menu selections, or test environments.

Syntax:

```
switch (expression) {
  case value1:
    // code for value1
    break;
  case value2:
    // code for value2
    break;
  default:
    // code when no case matches
}
```

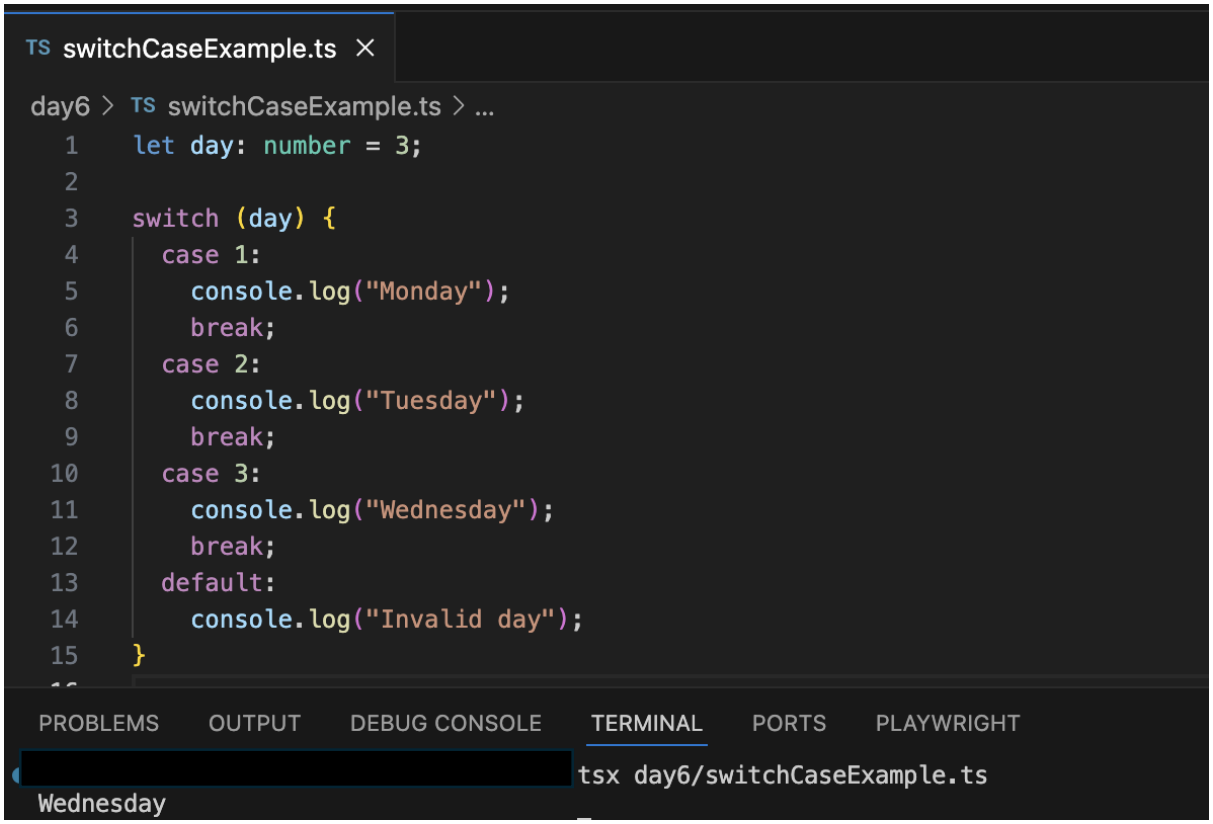
Example:

```
let day: number = 3;

switch (day) {
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  case 3:
    console.log("Wednesday");
    break;
  default:
    console.log("Invalid day");
}
```

Explanation:

Here, `day = 3`, so the program matches case 3 and prints “Wednesday”.
If no case matches, `default` is executed.



Summary

- Conditional statements help control the **program flow** based on conditions.
- `if` → checks one condition.
- `if-else` → handles true or false outcomes.
- `nested if-else` → checks multiple conditions in sequence.
- `switch` → handles multiple fixed values efficiently.

Use if/else when conditions are based on comparisons.
Use switch when you compare one variable with several possible constant values.

Questions

1. What are conditional statements in TypeScript?

2.When do we use an if statement?

3.What is the difference between if and if–else?

4.Why do we use nested if–else?

5.What happens if none of the conditions in nested if–else are true?

6.What is a switch statement used for?

7.What does the `break` keyword do in switch?

8.What is the difference between condition and expression?

Answers

1.Conditional statements allow the program to make decisions and execute specific code blocks based on conditions.

2.We use `if` when a block of code should only run if a particular condition is true.

`3.if` checks one condition and runs the code only if true,
while `if-else` provides an alternate path when the condition is false.

4.It is used when there are multiple conditions to check one after another.

5.The final `else` block executes if all conditions are false.

6.It's used when you need to compare one expression against multiple fixed values.

7.It stops the execution of the current case and prevents the next case from running accidentally.

8.A **condition** always returns true or false,
but an **expression** can return any value such as number, string, or boolean.
