PDF 07 · SECTION 04

The else if Statement

Handling Multiple Conditions

Learn how JavaScript checks several conditions in order using else if, and why the sequence of your conditions matters.

Table of Contents

01 When Two Choices Aren't Enough

So far, you've learned how JavaScript uses:

— if to execute code when a condition is true.

— else to execute an alternative block when that condition is false.

This works perfectly when there are only two possible outcomes.

But real-world applications are rarely that simple.

Imagine you're building an online examination platform. A student completes the final exam.

The application doesn't simply decide whether the student passed or failed. Instead, it may assign different grades.

90-100	Grade A
80-89	Grade B
70-79	Grade C
60-69	Grade D
Below 60	Fail

Notice that there are more than two possible outcomes. An if...else structure alone isn't enough.

The application needs to evaluate several conditions one after another until it finds the correct one.

This is exactly why JavaScript provides the else if statement.

02 What Is else if?

The else if statement allows JavaScript to check another condition if the previous condition was false.

Instead of stopping after one comparison, the program continues asking additional questions.

Think of it like an interview.

Imagine applying for a scholarship. The university may evaluate applications like this:

- If the student has exceptional grades, award the highest scholarship.
- Else if the student has very good grades, award a partial scholarship.
- Else if the student meets the minimum requirements, offer standard admission.
- Otherwise, reject the application.

The university doesn't evaluate every category. It stops as soon as it finds the first condition that matches.

JavaScript behaves in exactly the same way.

03 How else if Works

An else if chain follows a simple process.

STEP 1

JavaScript checks the first if condition. If it's true, the corresponding block executes, and the remaining conditions are skipped.

STEP 2

If the first condition is false, JavaScript checks the first else if. If that condition is true, its block executes.

STEP 3

If that condition is also false, JavaScript moves to the next else if. This process continues until a matching condition is found, or no conditions match.

STEP 4

If none of the conditions are true, the optional else block executes. This final block acts as the default outcome.

04 Real-World Example: haas.dev Membership Levels

Imagine haas.dev introduces different membership plans.

Depending on the learner's subscription, the dashboard should display different features. For example:

- Premium members unlock every course.
- Pro members unlock advanced courses.
- Free members can access beginner courses only.

The application checks each membership level one after another. As soon as it finds the correct plan, it stops checking the rest.

This ensures that each learner sees only the features appropriate for their subscription.

05 Real-World Example: Traffic Signals

Traffic lights provide another excellent analogy.

Imagine approaching an intersection. The traffic control system evaluates conditions like this:

— If the light is green, continue driving.

— Else if the light is yellow, prepare to stop.

— Else if the light is red, stop completely.

Only one instruction can apply at any given moment. The system doesn't tell drivers to stop and drive simultaneously.

Similarly, JavaScript executes only the first matching condition in an if...else if chain.

06 Real-World Example: Weather Application

Imagine creating a weather app.

Depending on the current conditions, the application displays different recommendations. For example:

- If it's raining, recommend carrying an umbrella.
- Else if it's snowing, recommend wearing warm clothing.
- Else if it's sunny, recommend sunscreen.
- Otherwise, display a general weather message.

The application continues checking until it finds the first condition that matches the current weather.

07 Why Order Matters

One of the most important things to understand about else if statements is that JavaScript checks conditions from top to bottom.

The order of your conditions directly affects the result.

Imagine a teacher grading assignments.

If they first check whether a score is greater than 50, then every student with a score above 90 will also satisfy that condition.

The more specific condition would never be reached.

Professional developers therefore arrange conditions carefully, usually placing the most specific rules before more general ones.

Thinking about the order of conditions is just as important as writing the conditions themselves.

08 Only One Block Executes

A common beginner misconception is that JavaScript evaluates every condition in an else if chain.

It doesn't.

As soon as one condition evaluates to true, JavaScript executes that block and skips the remaining conditions.

This makes execution faster and avoids unnecessary work.

Think of it like searching for a person's name in an alphabetically organized list.

Once you find the correct name, you stop searching. There is no reason to continue looking through the rest of the list.

09 When Should You Use else if?

Use else if whenever your application has three or more possible outcomes.

Examples include:

- **Student Grades**
  Assign different grades based on exam scores.
- **User Roles**
  Display different dashboards for Students, Teachers, and Administrators.
- **Shipping Charges**
  Calculate delivery options based on order value.
- **Membership Plans**
  Unlock different features for different subscription levels.
- **Discount Systems**
  Apply different discount percentages depending on purchase amount.
- **Customer Support**
  Route customers to different departments based on the type of issue they select.

10 Common Beginner Mistakes

1. Writing Conditions in the Wrong Order

   If a broad condition appears before a more specific one, the specific condition may never execute. Always think carefully about the sequence of your conditions.

2. Creating Too Many else if Statements

   Long chains of else if statements can become difficult to read. When many fixed values need to be checked, another control flow statement called switch is often a better choice. You'll learn about switch in the next section.

3. Forgetting the Final else

   Although the final else block is optional, it's often helpful. It provides a default response when none of the previous conditions match. This prevents unexpected situations from being ignored.

11 Best Practices

When writing else if chains:

- Arrange conditions from most specific to most general.
- Keep each condition focused on one idea.
- Avoid unnecessary repetition.
- Include a default else block whenever appropriate.
- Test every possible outcome, not just the first one.

Well-organized conditions make your applications easier to maintain and less likely to contain hidden bugs.

12 Key Takeaways

- else if allows JavaScript to evaluate multiple conditions.
- Conditions are checked one at a time from top to bottom.
- JavaScript executes only the first condition that evaluates to true.
- The order of conditions is extremely important.
- A final else block provides a default outcome when no conditions match.

13 Cheat Sheet

```
if (scoreA) {
  // grade A
} else if (scoreB) {
  // grade B
} else if (scoreC) {
  // grade C
} else {
  // default: fail
}
```

— Conditions are checked top to bottom — order changes the outcome.

— Only the first true condition runs; the rest are skipped.

— Put the most specific condition first, general ones last.

— Too many else if branches? Consider switch instead.

Checklist

- [ ] I understand why if...else alone isn't enough for 3+ outcomes.
- [ ] I can explain how an else if chain is evaluated step by step.
- [ ] I know that only the first matching condition executes.
- [ ] I understand why the order of conditions changes the result.
- [ ] I know when a long else if chain should become a switch instead.
- [ ] I always consider whether a default else block is needed.

Mini Architecture Challenge

You're designing the grading logic for haas.dev's course quizzes. Think through the order of conditions before writing any code.

The scenario:

- Scores of 90 and above should receive an 'Excellent' badge.
- Scores of 70 to 89 should receive a 'Good' badge.
- Scores of 50 to 69 should receive a 'Needs Improvement' badge.
- Anything below 50 should prompt the learner to retake the quiz.

Your task:

- Decide which condition should be checked first, and why.
- Sketch the full if / else if / else chain in plain English.
- Double-check: could a broader condition accidentally catch a case meant for a more specific one?

The goal is to practice ordering conditions correctly before ever touching syntax.

16 Related Resources

Related haas.dev PDFs:

- → JavaScript Comparison, Logical & Modern Operators

  Review how comparison operators create the conditions used in else if statements.

- → Variables & Memory: How JavaScript Stores, Remembers, and Uses Data

  Understand where the values being compared originate.

- → JavaScript Data Types: Understanding Every Kind of Data Your Program Can Store

  Learn how different data types affect conditional logic.

UP NEXT

In the next section, you'll learn about the switch statement—a cleaner alternative to long else if chains when your program needs to compare one value against many fixed options, such as user roles, menu selections, application themes, languages, or payment methods.

Recommended Learning Path

PDF 07B	Understanding the if Statement
PDF 07C	The else Statement
PDF 07D	**The else if Statement ← you are here**
PDF 07E	The switch Statement
PDF 08	Loops: for, while & Iteration

haas.dev

Understand → Break → Design → Build