PDF 07 · SECTION 03

The else Statement

Handling the Alternative Path

Learn how the else statement lets your program respond intelligently when a condition turns out to be false.

Table of Contents

01 What Happens When a Condition Is False?

In the previous section, you learned that an if statement executes a block of code only when its condition is true.

But real-world applications need to handle both possibilities.

Think about logging into a website.

If the password is correct, the user should enter their account. But what happens if the password is incorrect?

The application cannot simply do nothing. It needs to inform the user that the login failed.

This is where the else statement becomes useful.

The else statement tells JavaScript:

> If the previous if condition was not true, execute this block instead.

Together, if and else allow your program to respond intelligently regardless of whether a condition succeeds or fails.

02 Understanding the else Statement

Imagine a security guard standing at the entrance of an office building.

The guard follows one simple rule:

— If the employee has a valid ID card, allow them to enter.

— Otherwise, politely deny entry.

Notice that there are only two possible outcomes. Either the person enters, or they don't. There is no third option.

The if...else structure works exactly the same way. It creates two separate execution paths.

One path runs when the condition is true. The other path runs when the condition is false.

JavaScript always chooses one of these paths—never both.

03 Real-World Example: Login System

Imagine you're building the authentication system for haas.dev.

A learner enters their credentials. The application checks:

> Is the email and password correct?

Two outcomes are possible.

OUTCOME 1

The credentials are correct. The learner is logged in and redirected to the dashboard.

OUTCOME 2

The credentials are incorrect. The application displays: "Invalid email or password."

Without an else statement, users would receive no feedback when login fails. Good software always handles both success and failure.

04 Real-World Example: Online Shopping

Suppose a customer clicks Buy Now.

Before completing the purchase, the application checks:

> Is the product currently in stock?

`IF YES`

The order proceeds.

`OTHERWISE`

The application displays: "This product is currently unavailable."

Again, there are only two possible outcomes. The customer either purchases the item or receives an explanation.

05 Real-World Example: Exam Results

Imagine an online examination platform. The passing score is 80%.

After the exam, the application checks:

> Has the learner achieved the required score?

```
IF YES
Display: "Congratulations! You passed."
```

```
OTHERWISE
Display: "Keep practicing and try again."
```

Notice that both outcomes are valuable. Applications shouldn't only celebrate success. They should also guide users when something doesn't go as planned.

06 Why else Improves User Experience

One characteristic of professional software is that it always communicates clearly.

Imagine clicking a button and nothing happens. You wouldn't know whether:

- The internet stopped working.
- The application crashed.
- The request is still processing.
- Your action was successful.

Good applications avoid this confusion. Whenever something cannot happen, they explain why.

The else statement often provides those explanations.

07 Thinking About Two Possible Worlds

A useful way to understand if...else is to imagine two parallel worlds.

In one world, the condition is true. In the other, the condition is false.

JavaScript looks at the condition, chooses one world, executes that code, and ignores the other.

This makes your application predictable and organized.

08 When Should You Use else?

Use an else statement when every possible situation should produce a response.

Examples include:

- Authentication

  Either log the user in or display an error.

- Payments

  Either complete the payment or explain why it failed.

- Form Validation

  Either submit the form or highlight missing information.

- Course Progress

  Either unlock the next lesson or continue showing the current lesson.

- Age Verification

  Either allow access or deny access.

09 When You Don't Need else

Not every decision requires an alternative action.

Imagine a news website.

If a visitor subscribes to the newsletter, the website displays a thank-you message. If they don't subscribe, nothing special needs to happen.

In situations like this, a simple if statement may be enough.

Use else only when the application should perform a specific action if the condition is false.

10 Common Beginner Mistakes

1. Assuming else Has Its Own Condition

   Beginners sometimes think the else statement checks another condition. It doesn't. The else block simply runs when the previous if condition evaluates to false. There is no separate comparison.

2. Forgetting to Handle Failure

   Some beginners only write code for successful situations. Professional developers always ask:

   > What should happen if this doesn't work?

3. Writing Duplicate Logic

   Sometimes beginners place similar code inside both the if and else blocks. If both branches perform nearly the same task, reconsider your design. Try to keep each branch focused on what makes it unique.

11 Best Practices

When using if...else statements:

- Think about both success and failure.
- Provide meaningful feedback to users.
- Keep each branch simple and easy to read.
- Avoid unnecessary duplication.
- Make sure every possible outcome is handled gracefully.

Readable decision-making logic is easier to maintain as your projects grow.

12 Key Takeaways

- The else statement executes when the preceding if condition is false.
- Together, if and else create two possible execution paths.
- Only one path is executed during each evaluation.
- else helps applications respond to unsuccessful situations instead of failing silently.
- Professional software always considers both successful and unsuccessful outcomes.

13 Cheat Sheet

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

— else never has its own condition — it catches whatever if didn't.

— Exactly one of the two blocks runs, never both.

— Always plan the failure path, not just the success path.

— Skip else when there's genuinely nothing to do on failure.

Checklist

- [ ] I understand that else runs only when the if condition is false.
- [ ] I know else has no condition of its own.
- [ ] I can identify both outcomes in a real-world example.
- [ ] I always plan for the failure path, not just success.
- [ ] I know when a simple if is enough and else isn't needed.
- [ ] I avoid duplicating logic across if and else blocks.

Mini Architecture Challenge

You're designing the feedback logic for haas.dev's quiz submission flow. Think through both outcomes before writing any code.

The scenario:

— A learner submits a quiz with a passing score of 70%.

— If they pass, they should see a congratulations message and unlock the next lesson.

— If they don't pass, they should see encouragement and a retry option.

Your task:

— Write the plain-English rule for both outcomes first.

— Decide what message belongs in the if block versus the else block.

— Check: does either branch duplicate the other's logic?

The goal is to practice designing for both success and failure before touching syntax.

16 Related Resources

Related haas.dev PDFs:

- JavaScript Comparison, Logical & Modern Operators

  Learn how conditions are evaluated before reaching an if...else statement.

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

  Understand how decision-making values are stored.

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

  Learn how different values influence conditional logic.

Recommended Learning Path

PDF 07	JavaScript Control Flow
PDF 07B	Understanding the if Statement
PDF 07C	**The else Statement ← you are here**
PDF 08	Loops: for, while & Iteration
PDF 09	Functions: Reusable Logic Blocks

haas.dev

Understand → Break → Design → Build