PDF 07 · SECTION 02

Understanding the if Statement

The First Decision-Making Tool in JavaScript

A deep dive into how the if statement lets JavaScript check a condition and choose exactly one path to run.

Table of Contents

01 The First Decision-Making Tool

02 What Is an if Statement?

03 How the if Statement Works

04 A Real-Life Analogy

05 Real-World Example: Online Banking

06 Real-World Example: haas.dev

07 Why if Statements Are Everywhere

08 Thinking Like a Programmer

09 Common Beginner Mistakes

10 Best Practices

11 Key Takeaways

12 Cheat Sheet

13 Checklist

14 Mini Architecture Challenge

15 Related Resources

16 Recommended Learning Path

01 The First Decision-Making Tool

Imagine you're building the login page for haas.dev.

A learner enters their email and password, then clicks the Log In button.

At this point, the application faces an important question:

> Should this user be allowed to enter the dashboard?

The answer depends on a condition.

If the entered credentials are correct, the learner should be logged in. If they're incorrect, the application should display an error message.

Notice that the application doesn't perform both actions. It chooses one path based on a condition. This is the purpose of the if statement.

The if statement is the simplest and most commonly used decision-making tool in JavaScript. It allows your program to say:

> "If this condition is true, execute this block of code."

If the condition isn't true, JavaScript simply skips that block and continues executing the rest of the program.

This simple idea forms the foundation of almost every application you'll ever build.

02 What Is an if Statement?

An if statement is a conditional statement.

A conditional statement tells JavaScript that some code should only run when a specific condition is satisfied.

Think of it as setting a rule.

For example:

— If the user is logged in, show the dashboard.

— If the shopping cart is empty, disable the checkout button.

— If the learner scores above 80%, unlock the certificate.

Without conditions, every action would happen every time, which would make applications useless.

The if statement gives your program the ability to make intelligent choices.

03 How the if Statement Works

Every if statement follows the same thought process.

Step 1 — Check a Condition

JavaScript first evaluates the condition. This condition must eventually become either:

— true

— false

Remember that comparison and logical operators are usually used to create these conditions.

Step 2 — Make a Decision

If the condition is true, JavaScript executes the code inside the if block. If the condition is false, JavaScript skips that block entirely and continues with the next instruction.

This process happens extremely quickly, often millions of times while an application is running.

04 A Real-Life Analogy

Imagine you're standing outside your house with a smart door lock.

The door follows one simple rule:

— If the fingerprint matches, unlock the door.
— Otherwise, keep the door locked.

The door doesn't unlock for everyone. It first checks a condition.

The if statement behaves in exactly the same way. It doesn't ask, "What code should I run?"

Instead, it asks,

> Should I run this code?

Only if the answer is yes does the code execute.

05 Real-World Example: Online Banking

Imagine you're transferring money through a banking app.

Before completing the transfer, the application checks:

- Does the account exist?
- Is the user authenticated?
- Does the account have enough balance?

If all required conditions are satisfied, the transfer proceeds. Otherwise, it stops and displays an appropriate message.

The banking system doesn't blindly process every request. It makes decisions based on conditions.

06 Real-World Example: haas.dev

Suppose a learner wants to access an advanced JavaScript course.

Before opening the lesson, the platform checks:

— Has the learner completed the prerequisite course?

If yes,

the lesson opens.

If not,

the learner is guided back to complete the required material first.

This creates a structured learning experience.

07 Why if Statements Are Everywhere

Once you begin noticing them, you'll realize that if statements power almost every feature in modern software.

- Authentication

  If the password is correct, log the user in.

- E-commerce

  If the product is in stock, allow checkout.

- Social Media

  If the user owns the post, show the Delete button.

- Video Streaming

  If the subscription is active, allow video playback.

- Navigation Apps

  If GPS permission is granted, display the user's location.

- Learning Platforms

  If the learner finishes a quiz, unlock the next lesson.

08 Thinking Like a Programmer

Beginners often focus on writing code.

Professional developers focus on writing rules.

Instead of thinking:

> "I need to write JavaScript."

Think:

> "What decision does my application need to make?"

Once the decision is clear, the code becomes much easier to write.

Programming is largely about translating business rules into logical conditions.

09 Common Beginner Mistakes

1. Forgetting That Every if Needs a Condition

   An if statement without a meaningful condition cannot make a decision.

   > What exactly am I checking?

2. Writing Complicated Conditions Too Early

   Beginners sometimes try to combine many conditions inside a single if statement. Start simple. Once you understand individual conditions, combining them with logical operators becomes much easier.

3. Forgetting That if Doesn't Repeat

   An if statement checks its condition once at that moment. If you need continuous checking or repetition, you'll later learn about loops. Understanding this difference prevents many beginner misunderstandings.

10 Best Practices

When writing if statements:

- Make conditions easy to read.
- Keep each condition focused on one idea.
- Use meaningful variable names.
- Avoid deeply nested conditions whenever possible.
- Think about the real-world rule before writing the code.

Readable conditions make applications easier to debug and maintain.

11 Key Takeaways

- An if statement allows JavaScript to make decisions.
- It executes code only when a condition is true.
- Conditions usually come from comparison or logical operators.
- if statements are used in almost every real-world application.
- Good developers think about business rules before writing conditions.

12 Cheat Sheet

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

if (condition) {
  // true path
} else {
  // false path
}
```

— A condition always resolves to true or false.

— The if block runs once — it does not repeat.

— Keep one condition, one idea per if statement.

— Think in business rules before writing syntax.

Checklist

- [ ] I understand what a conditional statement is.
- [ ] I can explain the two steps: check a condition, then decide.
- [ ] I can connect the if statement to a real-world rule.
- [ ] I know that if statements don't repeat on their own.
- [ ] I avoid combining too many conditions in a single if statement.
- [ ] I think about the business rule before writing the code.

Mini Architecture Challenge

You're designing the rule for unlocking an advanced lesson on haas.dev. Think through the condition before writing any code.

The scenario:

— A learner must complete the prerequisite course before this lesson unlocks.

— If they haven't, they should be guided back to the required material.

— If they have, the lesson should open immediately.

Your task:

— Write out the rule in plain English before touching syntax.

— Identify exactly one condition this if statement needs to check.

— Sketch the if / else block on paper first.

The goal is to practice translating a business rule into a condition — the code itself should feel like the easy part.

15 Related Resources

To strengthen your understanding, revisit these related haas.dev PDFs:

- → JavaScript Comparison, Logical & Modern Operators

  Learn how conditions are created.

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

  Understand where condition values come from.

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

  Learn how different values behave inside conditions.

Recommended Learning Path

PDF 06	Comparison, Logical & Modern Operators
PDF 07	JavaScript Control Flow
PDF 07B	**Understanding the if Statement ← you are here**
PDF 08	Loops: for, while & Iteration
PDF 09	Functions: Reusable Logic Blocks

haas.dev

Understand → Break → Design → Build