

# Logarithms

## Why This Matters

Logarithms are the **inverse of exponentials**. They answer the question: "What power do I need to raise this base to get this number?"

As a developer, logarithms are absolutely everywhere:

- **Big-O notation:**  $O(\log n)$  is super fast
- **Binary search:** Halving search space repeatedly
- **Compression:** JPEG, MP3 use logarithmic scales
- **Sound:** Decibels are logarithmic
- **Chemistry:** pH scale
- **Data structures:** Tree height is  $\log(n)$
- **Cryptography:** Key sizes (2048-bit =  $2^{2048}$ )

Logarithms turn multiplication into addition, division into subtraction, and exponentials into multiplication. They're one of the most powerful tools in mathematics.

---

## The Big Picture: Undoing Exponents

### The Relationship

If:  $2^3 = 8$

Then:  $\log_2(8) = 3$

Read as: "log base 2 of 8 equals 3"

Meaning: "2 to what power gives 8? Answer: 3"

### General form:

If:  $b^x = y$

Then:  $\log_b(y) = x$

b = base

x = exponent

y = result

### Mental Model: Inverse Operations

Exponent asks: "What is  $2^3$ ?" → 8

Logarithm asks: "2 to what power is 8?" → 3

Square asks: "What is  $5^2$ ?" → 25

Square root asks: "What squared is 25?" → 5

Multiply asks: "What is  $3 \times 4$ ?" → 12

Divide asks: "What times 4 is 12?" → 3

Logarithms are as fundamental as square roots or division—they're just the inverse of a different operation.

---

# 1. Basic Logarithm Notation

## Standard Form

$$\log_b(x) = y$$

b = base (subscript)

x = argument (what you're taking the log of)

y = result (the exponent)

## Examples:

$$\begin{array}{ll} \log_2(8) = 3 & \text{because } 2^3 = 8 \\ \log_{10}(100) = 2 & \text{because } 10^2 = 100 \\ \log_3(27) = 3 & \text{because } 3^3 = 27 \\ \log_5(25) = 2 & \text{because } 5^2 = 25 \end{array}$$

## Common Bases

### Base 10 (common logarithm):

$$\log_{10}(x) \quad \text{often written as} \quad \log(x)$$

$$\log(100) = 2 \quad \text{because } 10^2 = 100$$

$$\log(1000) = 3 \quad \text{because } 10^3 = 1000$$

### Base 2 (binary logarithm):

$$\log_2(x) \quad \text{often written as} \quad \lg(x) \quad \text{in CS}$$

$$\lg(8) = 3 \quad \text{because } 2^3 = 8$$

$$\lg(1024) = 10 \quad \text{because } 2^{10} = 1024$$

### Base e (natural logarithm):

$$\log_e(x) \quad \text{written as} \quad \ln(x)$$

$$e \approx 2.71828\dots \quad (\text{Euler's number})$$

$$\ln(e) = 1 \quad \text{because } e^1 = e$$

$$\ln(e^2) = 2 \quad \text{because } e^2 = e^2$$

## Programming

```
Math.log10(100); // 2 (base 10)
Math.log2(8);    // 3 (base 2)
Math.log(Math.E); // 1 (natural log, base e)

// General base
function logBase(x, base) {
```

```
    return Math.log(x) / Math.log(base);  
}  
  
logBase(8, 2); // 3
```

## 2. Why Logarithms Exist: The Huge Numbers Problem

### The Problem

**Question:** 2 to what power equals 1024?

You could try:

```
21 = 2  
22 = 4  
23 = 8  
24 = 16  
25 = 32  
...keep going...  
210 = 1024 ✓
```

But this is tedious. **Logarithms give you the answer directly:**

```
log2(1024) = 10
```

### Logarithms Make Hard Problems Easy

**Without logs:** "What power of 2 gives 4,294,967,296?"

- You'd have to compute powers forever

**With logs:**  $\log_2(4,294,967,296) = 32$

- Instant answer

### The Scale Problem

Some quantities span huge ranges:

- Sound: From whisper ( $10^{-12}$  W/m<sup>2</sup>) to jet engine (1 W/m<sup>2</sup>)
- Earthquakes: From 2.0 (barely felt) to 9.0 (catastrophic)
- Chemical acidity: pH 1 (strong acid) to pH 14 (strong base)

**Logarithmic scales compress these ranges** so they're manageable:

```
Linear scale:  1, 10, 100, 1000, 10000, ...  
Log scale:    0,  1,  2,   3,   4, ...
```

## 3. Evaluating Logarithms

### Easy Cases (Powers You Know)

$$\log_2(8) = ?$$

Think: 2 to what power is 8?

$$2^3 = 8$$

Answer: 3

$$\log_{10}(1000) = ?$$

Think: 10 to what power is 1000?

$$10^3 = 1000$$

Answer: 3

$$\log_5(1) = ?$$

Think: 5 to what power is 1?

$$5^0 = 1$$

Answer: 0

## Special Values

$$\log_b(1) = 0 \quad \text{because } b^0 = 1$$

$$\log_b(b) = 1 \quad \text{because } b^1 = b$$

$$\log_b(b^n) = n \quad \text{because } b^n = b^n$$

### Examples:

$$\log_{10}(1) = 0$$

$$\log_2(2) = 1$$

$$\log_5(5^3) = 3$$

## Fractional Results

Not all logs are whole numbers:

$$\log_2(5) \approx 2.32 \quad \text{because } 2^{2.32} \approx 5$$

$$\log_{10}(50) \approx 1.70 \quad \text{because } 10^{1.70} \approx 50$$

You need a calculator for most of these.

## Negative Results

$$\log_2(1/2) = -1 \quad \text{because } 2^{-1} = 1/2$$

$$\log_{10}(0.01) = -2 \quad \text{because } 10^{-2} = 0.01$$

**Pattern:** Fractions (values less than 1) give negative logs.

## Undefined Cases

$$\log_b(0) = \text{undefined} \quad (\text{no power of } b \text{ gives } 0)$$

$$\log_b(\text{negative}) = \text{undefined} \quad (\text{in real numbers})$$

---

## 4. Logarithm Rules (The Magic)

These rules make logarithms incredibly powerful. They turn multiplication/division into addition/subtraction.

## Rule 1: Log of a Product

**Multiplication becomes addition:**

$$\log_b(xy) = \log_b(x) + \log_b(y)$$

**Example:**

$$\begin{aligned}\log_2(8 \times 4) &= \log_2(8) + \log_2(4) \\ \log_2(32) &= 3 + 2 \\ 5 &= 5 \quad \checkmark\end{aligned}$$

**Why it works:**

If  $x = b^m$  and  $y = b^n$ , then:  
 $xy = b^m \times b^n = b^{(m+n)}$

$$\text{So: } \log_b(xy) = m + n = \log_b(x) + \log_b(y)$$

**Programming analogy:**

```
// Multiplying in linear space
const result = x * y;

// Adding in log space
const logResult = Math.log(x) + Math.log(y);
const result = Math.exp(logResult); // convert back
```

## Rule 2: Log of a Quotient

**Division becomes subtraction:**

$$\log_b(x/y) = \log_b(x) - \log_b(y)$$

**Example:**

$$\begin{aligned}\log_2(8/2) &= \log_2(8) - \log_2(2) \\ \log_2(4) &= 3 - 1 \\ 2 &= 2 \quad \checkmark\end{aligned}$$

## Rule 3: Log of a Power

**Exponents become multiplication:**

$$\log_b(x^n) = n \times \log_b(x)$$

**Example:**

$$\begin{aligned}\log_2(8^3) &= 3 \times \log_2(8) \\ \log_2(512) &= 3 \times 3\end{aligned}$$

$$9 = 9 \checkmark$$

**This is huge:** It means you can pull exponents out front.

$$\log_{10}(x^{100}) = 100 \times \log_{10}(x)$$

## Rule 4: Change of Base

**Convert between bases:**

$$\log_b(x) = \log_a(x) / \log_a(b)$$

**Example:** Convert  $\log_2(8)$  to base 10

$$\begin{aligned} \log_2(8) &= \log_{10}(8) / \log_{10}(2) \\ &= 0.903 / 0.301 \\ &\approx 3 \checkmark \end{aligned}$$

**Why it's useful:** Calculators only have  $\log_{10}$  and  $\ln$ , so you use this to compute other bases.

## Summary of Rules

Operation	Logarithm Rule	Example
Multiply	$\log(xy) = \log(x) + \log(y)$	$\log(6) = \log(2) + \log(3)$
Divide	$\log(x/y) = \log(x) - \log(y)$	$\log(4) = \log(8) - \log(2)$
Power	$\log(x^n) = n \cdot \log(x)$	$\log(8) = 3 \cdot \log(2)$
Change Base	$\log_b(x) = \log(x) / \log(b)$	$\log_2(8) = \log(8) / \log(2)$

# 5. Visual Intuition: Logarithmic Scales

## Linear vs Logarithmic

**Linear scale:** Even spacing

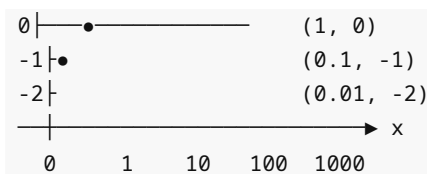
0—1—2—3—4—5—6—7—8—9—10

**Logarithmic scale:** Each step is a multiplication

1—10—100—1000—10000  
 ↑    ×10   ×10   ×10   ×10

## Graph of $y = \log(x)$



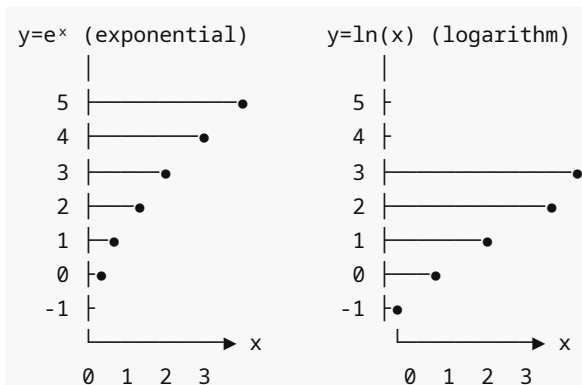


Key features:

- **Passes through (1, 0):**  $\log(1) = 0$
- **Increases slowly:** Takes large  $x$  changes for small  $y$  changes
- **Negative for  $0 < x < 1$ :** Fractions have negative logs
- **Undefined at  $x = 0$ :** Vertical asymptote
- **Never horizontal:** Always increasing (for base  $> 1$ )

## Comparison to $y = e^x$

They're **inverses** (mirror images across  $y = x$ ):



Notice:

- Exponential grows fast, log grows slow
- They're reflections across the line  $y = x$

## 6. Real-World Applications

### Big-O Notation: $O(\log n)$

**Binary search** is  $O(\log n)$ :

Array of 1,000,000 elements  
 Linear search: Up to 1,000,000 comparisons  
 Binary search:  $\log_2(1,000,000) \approx 20$  comparisons

That's 50,000× faster!

**Why?** Each comparison cuts the problem in half:

$1,000,000 \rightarrow 500,000 \rightarrow 250,000 \rightarrow 125,000 \rightarrow \dots \rightarrow 1$

Number of halvings =  $\log_2(1,000,000)$

#### Code:

```
function binarySearch(arr, target) {  
  let left = 0, right = arr.length - 1;  
  
  while (left <= right) {  
    const mid = Math.floor((left + right) / 2);  
  
    if (arr[mid] === target) return mid;  
    if (arr[mid] < target) left = mid + 1;  
    else right = mid - 1;  
  }  
  
  return -1;  
}  
  
// Time complexity: O(log n)
```

### Tree Height

**Balanced binary tree with n nodes has height  $\log_2(n)$ :**

1 node: height 0 ( $2^0 = 1$ )  
3 nodes: height 1 ( $2^1 - 1 = 1$ , plus 2 children)  
7 nodes: height 2 ( $2^2 - 1 = 3$ , plus 4 grandchildren)

Height =  $\lceil \log_2(n+1) \rceil - 1$

**Why it matters:** Operations take  $O(\text{height})$  time.

### Decibels (Sound)

Decibels (dB) =  $10 \times \log_{10}(I / I_0)$

I = intensity

$I_0$  = reference intensity (threshold of hearing)

#### Examples:

Whisper: 30 dB ( $1,000\times$  reference)  
Conversation: 60 dB ( $1,000,000\times$  reference)  
Jet engine: 140 dB ( $10^{14}\times$  reference)

Each 10 dB increase =  $10\times$  louder.

### pH Scale (Chemistry)

$\text{pH} = -\log_{10}([\text{H}^+])$

$[\text{H}^+]$  = hydrogen ion concentration

### Examples:

pH 7 (neutral):  $[H^+] = 10^{-7}$   
pH 1 (strong acid):  $[H^+] = 10^{-1}$  (1,000,000× more acidic)  
pH 14 (strong base):  $[H^+] = 10^{-14}$

### Richter Scale (Earthquakes)

$$\text{Magnitude} = \log_{10}(A / A_0)$$

A = amplitude

$A_0$  = reference amplitude

Each whole number increase = 10× more energy released.

Magnitude 5: Notable  
Magnitude 6: 10× stronger  
Magnitude 7: 100× stronger (destructive)  
Magnitude 8: 1,000× stronger (major earthquake)

### Information Theory

**Bits needed to represent n items:**

$$\text{bits} = \log_2(n)$$

8 items:  $\log_2(8) = 3$  bits

256 items:  $\log_2(256) = 8$  bits (1 byte)

**Entropy** (information content):

$$H = -\sum p(x) \log_2(p(x))$$

### Compound Interest (Doubling Time)

$$A = P(1 + r)^t$$

To find when money doubles:

$$2P = P(1 + r)^t$$

$$2 = (1 + r)^t$$

$$\log(2) = t \times \log(1 + r)$$

$$t = \log(2) / \log(1 + r)$$

For 5% interest:

$$t = \log(2) / \log(1.05) \approx 14 \text{ years}$$

---

## 7. Solving Logarithmic and Exponential Equations

**Type 1: Solve for x in  $\log(x) = n$**

$$\log_2(x) = 5$$

Convert to exponential form:

$$x = 2^5 = 32$$

**General:** If  $\log_b(x) = n$ , then  $x = b^n$

### Type 2: Solve for x in $b^x = n$

$$2^x = 32$$

Take log of both sides:

$$\log_2(2^x) = \log_2(32)$$

Use log rule (power comes out):

$$x \times \log_2(2) = \log_2(32)$$

$$x \times 1 = 5$$

$$x = 5$$

**Or:** Recognize that  $32 = 2^5$ , so  $x = 5$

### Type 3: Different Bases

$$3^x = 100$$

Take log of both sides (any base, use base 10):

$$\log(3^x) = \log(100)$$

$$x \times \log(3) = \log(100)$$

$$x \times \log(3) = 2$$

$$x = 2 / \log(3)$$

$$x \approx 2 / 0.477 \approx 4.19$$

### Type 4: Multiple Logs

$$\log(x) + \log(x-3) = 1$$

Use product rule:

$$\log(x(x-3)) = 1$$

Convert to exponential (assuming base 10):

$$x(x-3) = 10^1$$

$$x^2 - 3x = 10$$

$$x^2 - 3x - 10 = 0$$

Factor:

$$(x-5)(x+2) = 0$$

$$x = 5 \text{ or } x = -2$$

Check: x must be positive (can't log negative)

$$x = 5 \checkmark$$

---

## 8. Natural Logarithm (ln) and e

### Euler's Number (e)

$$e \approx 2.71828\dots$$

**Definition:** The base of natural logarithm, defined by:

$$e = \lim_{n \rightarrow \infty} (1 + 1/n)^n$$

Or:

$$e = 1 + 1/1! + 1/2! + 1/3! + 1/4! + \dots$$

### Why e Is Special

**Natural growth/decay** uses e:

Continuous compound interest:  $A = Pe^{(rt)}$

Population growth:  $P(t) = P_0 e^{(kt)}$

Radioactive decay:  $N(t) = N_0 e^{(-\lambda t)}$

**Derivative property:** The derivative of  $e^x$  is  $e^x$  (unchanged!)

### Natural Logarithm

$$\ln(x) = \log_e(x)$$

$$\ln(e) = 1$$

$$\ln(e^2) = 2$$

$$\ln(1) = 0$$

**Connection:**

If  $y = e^x$ , then  $x = \ln(y)$

They're inverses:

$$e^{(\ln(x))} = x$$

$$\ln(e^x) = x$$

### Programming

```
Math.E;           // 2.718281828459045
Math.log(Math.E); // 1 (natural log)
Math.exp(1);       // e (same as e^1)

// e^x
Math.exp(2);       // e^2 ≈ 7.389
```

```
// ln(x)
Math.log(10);    // ln(10) ≈ 2.303
```

## 9. Logarithmic Thinking

### Halving Problems

"How many times can you divide 1000 by 2 before reaching 1?"

$1000 \rightarrow 500 \rightarrow 250 \rightarrow 125 \rightarrow 62.5 \rightarrow 31.25 \rightarrow \dots$

Answer:  $\log_2(1000) \approx 10$  times

### Doubling Problems

"How many times must you double 1 to reach 1000?"

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 1024$

Answer:  $\log_2(1000) \approx 10$  times

### Order of Magnitude

How big is this number?

$\log_{10}(5000) \approx 3.7$

Interpretation: Between  $10^3$  and  $10^4$  (thousands)

### Scaling Intuition

If you double the input to a log function, how much does the output change?

$\log_2(10) \approx 3.32$

$\log_2(20) \approx 4.32$

Difference: 1 (always, regardless of starting value)

Doubling adds 1 to the log.

## Common Mistakes & Misconceptions

**✗ "log(a + b) = log(a) + log(b)"**

**No!** Logs don't distribute over addition.

$\log(a + b) \neq \log(a) + \log(b)$

$\log(10 + 10) = \log(20) \approx 1.30$

$\log(10) + \log(10) = 1 + 1 = 2 \neq 1.30$

**Correct:**

$$\log(a \times b) = \log(a) + \log(b) \quad (\text{multiplication} \rightarrow \text{addition})$$

**✗ "log(x)/log(y) = log(x/y)"**

**No!**

$$\begin{aligned}\log(x)/\log(y) &= \log_y(x) \quad (\text{change of base}) \\ \log(x/y) &= \log(x) - \log(y) \quad (\text{quotient rule})\end{aligned}$$

**✗ "log<sub>2</sub>(8) = log<sub>10</sub>(8)"**

**No!** Different bases give different results:

$$\begin{aligned}\log_2(8) &= 3 \\ \log_{10}(8) &\approx 0.903\end{aligned}$$

**✗ "Logs can take negative inputs"**

Not in real numbers:

$$\log(-5) = \text{undefined} \quad (\text{no real answer})$$

**✗ "log(0) = 0"**

**No!**

$$\begin{aligned}\log(1) &= 0 \\ \log(0) &= \text{undefined} \quad (\text{negative infinity})\end{aligned}$$

---

## Tiny Practice

Evaluate:

1.  $\log_2(16)$
2.  $\log_{10}(1000)$
3.  $\log_5(25)$
4.  $\log_3(1)$
5.  $\log_2(1/4)$

Simplify using log rules:

6.  $\log(5) + \log(2)$
7.  $\log(100) - \log(10)$
8.  $3 \times \log(2)$
9.  $\log(x^3)$
10.  $\log_2(8) + \log_2(4) - \log_2(2)$

Solve:

11.  $\log_2(x) = 4$
12.  $2^x = 64$

13. If  $\log(x) = 3$ , what is  $x$ ? (assume base 10)
14. How many times can you divide 512 by 2 before reaching 1?

► Answers

---

## Summary Cheat Sheet

### Definition

If  $b^x = y$ , then  $\log_b(y) = x$

$\log_b(y)$  asks: "b to what power gives y?"

### Common Bases

$\log(x) = \log_{10}(x)$  (common logarithm)  
 $\lg(x) = \log_2(x)$  (binary logarithm)  
 $\ln(x) = \log_e(x)$  (natural logarithm)

### Key Values

$\log_b(1) = 0$   
 $\log_b(b) = 1$   
 $\log_b(b^n) = n$

### Logarithm Rules

Rule	Formula	Intuition
Product	$\log(xy) = \log(x) + \log(y)$	Multiply → Add
Quotient	$\log(x/y) = \log(x) - \log(y)$	Divide → Subtract
Power	$\log(x^n) = n \cdot \log(x)$	Exponent → Multiply
Change Base	$\log_b(x) = \log(x)/\log(b)$	Convert bases

### Inverse Relationship

$b^{\log_b(x)} = x$   
 $\log_b(b^x) = x$   
 $e^{\ln(x)} = x$   
 $\ln(e^x) = x$

### Applications

- **Algorithm analysis:**  $O(\log n)$
- **Data structures:** Tree height
- **Sound:** Decibels
- **Chemistry:** pH

- **Earthquakes:** Richter scale
- **Information:** Bits needed

## Programming

```
Math.log10(x);    // base 10
Math.log2(x);     // base 2
Math.log(x);      // natural (base e)

// Change of base
Math.log(x) / Math.log(base);
```

---

## Next Steps

You now understand logarithms—one of the most powerful tools in mathematics. They let you work with massive ranges of values, analyze algorithms, and understand growth/decay.

Next, we'll explore **Coordinate Geometry**—how to represent points, lines, and shapes on a grid.

**Continue to:** [05-coordinate-geometry.md](#)