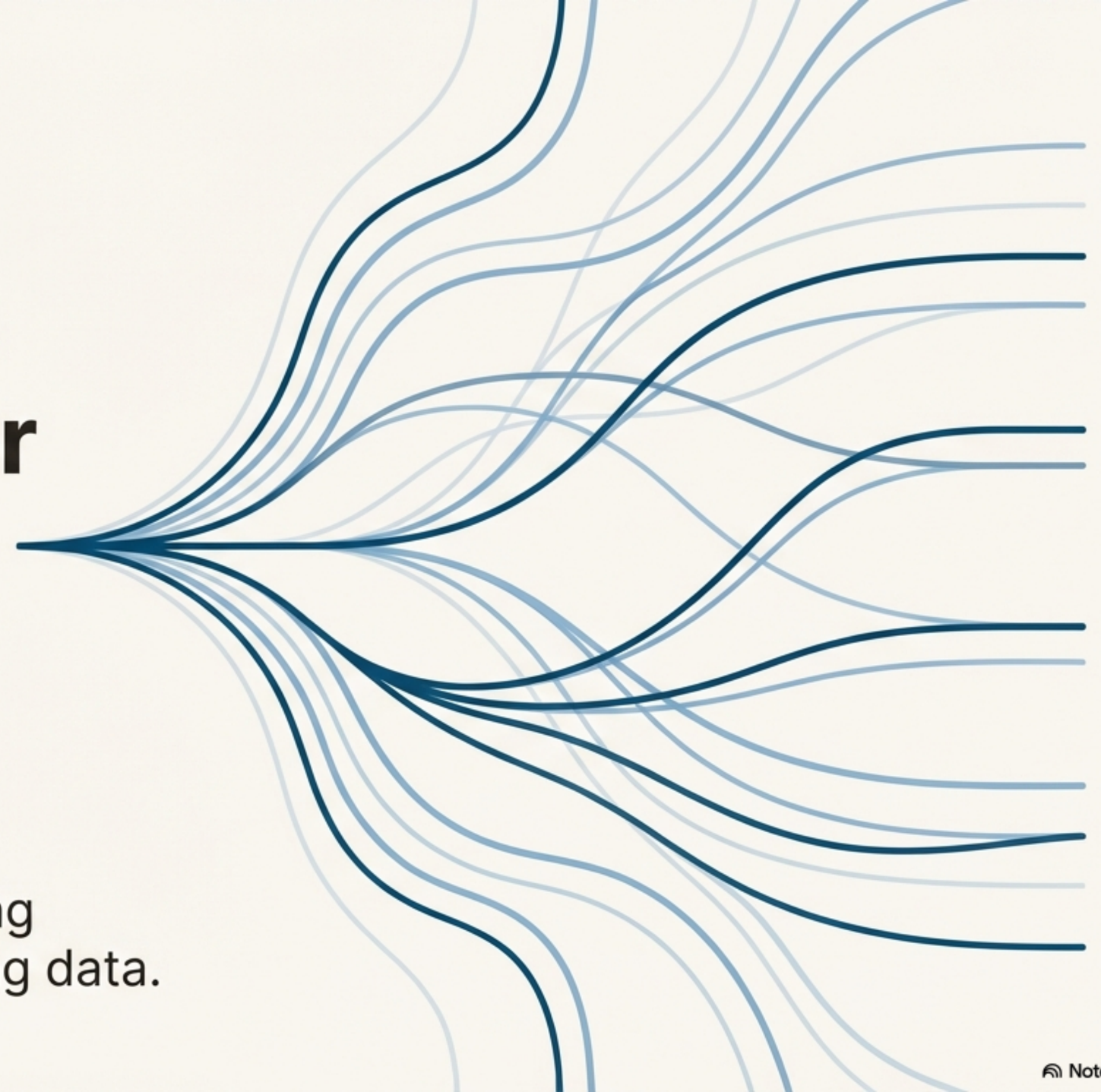
From Setup to Scripting: Your First Steps in JavaScript JavaScript

A practical guide to connecting your code, storing information, and manipulating data.



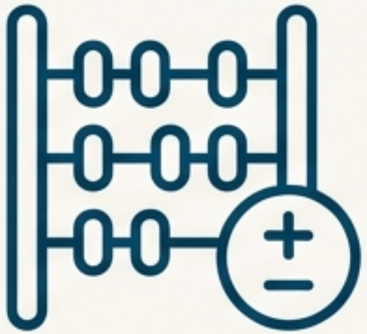
What You'll Master in This Guide

1. The Connection

- Use the `<script>` tag to add JavaScript to your HTML.
- Ensure your code is **non-blocking** for faster page loads.
- Master the use of the **JavaScript Console** for debugging.

2. The Language of Storage

- Declare variables with `let` and `const`.
- Assign **strings**, **numbers**, **booleans**, and `null` values.
- Use **template literals** to combine text and variables seamlessly.



3. The Language of Logic

- Perform **arithmetic operations**: addition, subtraction, multiplication, division.
- Understand and control **operator precedence** using **parentheses**.
- Use **increment** and **decrement operators** to modify numeric values.

4. The Language of Text

- Investigate and transform **strings** with **built-in methods**.
- Extract parts of a string using **slicing**.
- **Concatenate** (join) strings together effectively.

Chapter 1: Making the Connection

How JavaScript Breathes Life into a Web Page

JavaScript code must be linked to an HTML file to interact with it. The **<script>** tag is the bridge between your static HTML structure and your dynamic JS logic. This is the first and most crucial step.

Two Methods for Adding JavaScript

Inline Script

Code is placed directly in the HTML file, just before the closing `</body>` tag.

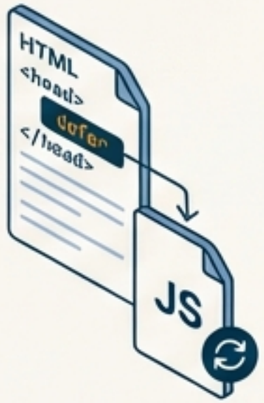
```
``html
  ...
  <script>
    console.log("Hello from web page.");
  </script>
</body>
</html>
```

Good for very small, page-specific scripts.

External File RECOMMENDED

Code is kept in a separate `.js` file and linked from the HTML `<head>`. The `defer` attribute is critical.

```
<head>
  ...
  <script defer src="script.js"></script>
</head>
```



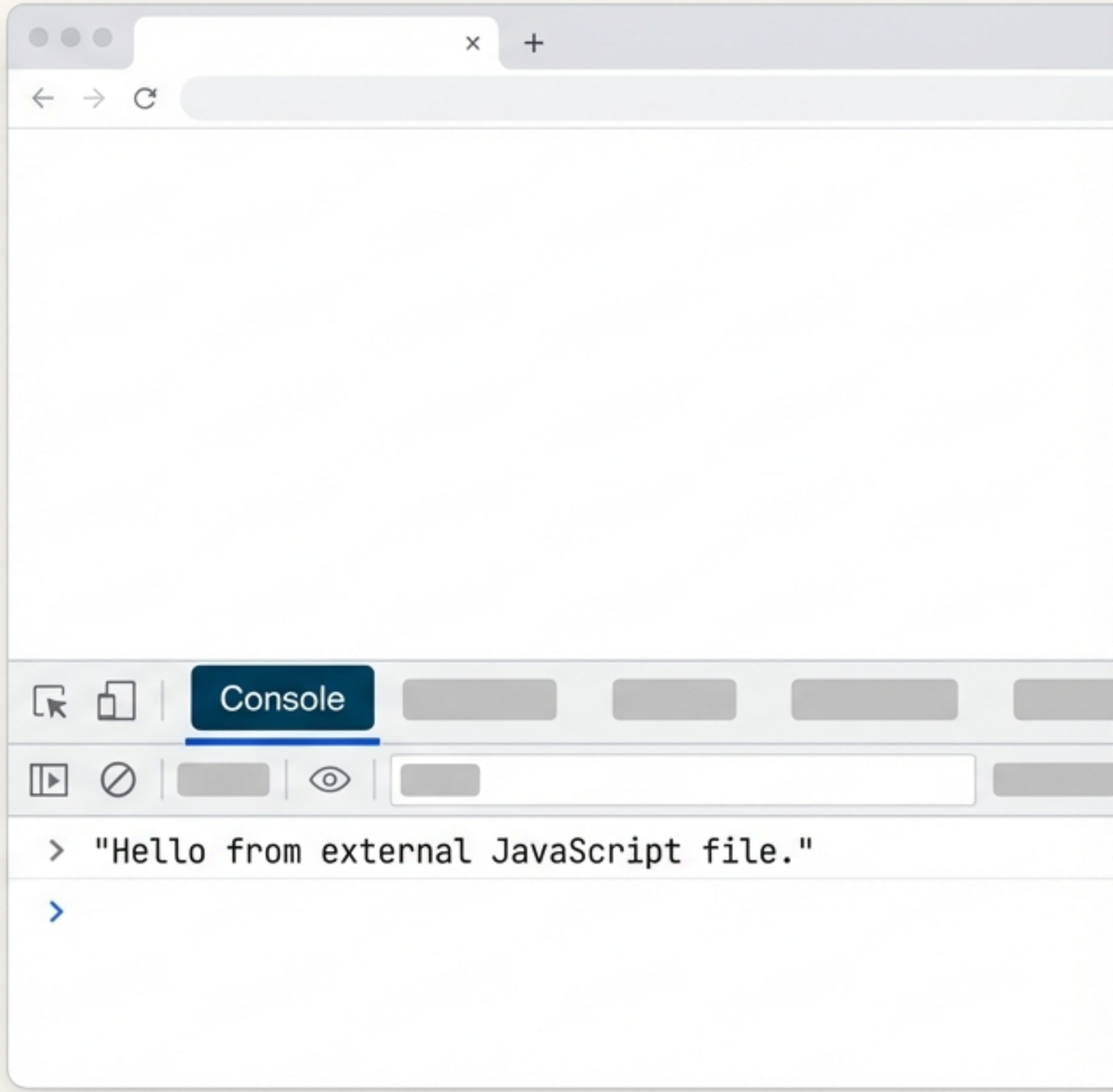
Key Benefits

- **Separation:** Keeps HTML and JavaScript clean and easier to maintain.
- **Non-Blocking:** The `defer` attribute ensures the script doesn't delay the rendering of HTML, leading to faster page loads.
- **Caching:** Browsers can cache the `.js` file, speeding up loads on multi-page sites.

Your Essential Tool: The JavaScript Console

Your web browser's Developer Tools (DevTools) includes a JavaScript Console. This is where you will see the output from your scripts, check the values of variables, and find error messages.

Use `console.log()` in your code to print information directly to the Console.



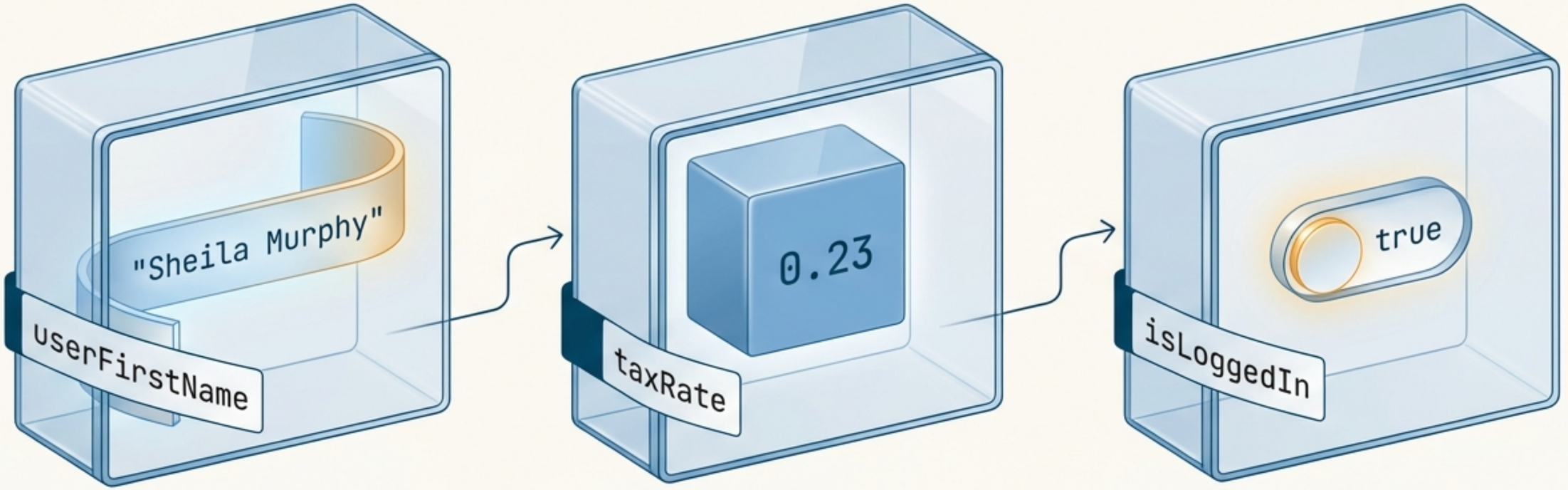
How to Open DevTools

Windows: `Ctrl + Shift + i`
Mac (Safari): `⌘ + ⌥ + i`
Alternatively: Right-click on the webpage and choose 'Inspect Element'.

Chapter 2: The Language of Storage

Using Variables to Hold Information

Think of a variable as a labeled container or 'storage box' for a value. The value inside the box might change (e.g., a user's name, a game score), but the label on the box (the variable name) remains the same.



`let` vs. `const`: The Two Keywords for Declaring Variables

Some data is meant to change, some is not. JavaScript gives you two ways to declare variables based on this principle.

`let` (The Value Can Change)

For values that you expect to reassign or update later.

```
// The score in a computer game
let playerScore = 100;
playerScore = 150; // Value is updated
```

`const` (The Value is Constant)

For values that should never change once assigned.

```
// A person's date of birth
const dateOfBirth = "1990-01-01";
// Attempting to change it will cause an error.
```

Older JavaScript used the keyword `var`. This is no longer recommended.

The Art of Naming Variables

Clear variable names make your code easier to read and understand. Follow these standard conventions.

Use camelCase

The first word is lowercase, and each subsequent word starts with a capital letter.

```
let postCode;
const localCurrency;
```

No Spaces or Dashes

These will cause errors.

```
let SubTotal; // Invalid
const Product-Name; // Invalid
```

Names are Case-Sensitive

`firstName` and `FirstName` are two different variables.

```
const firstName; // This is one variable
let FirstName;   // This is a DIFFERENT
```

Be Unique

Don't use the same name for two different variables in the same scope.

Editorial Tech

An Introduction to JavaScript's Core Data Types

`string`

Text within single (` '), double (" "), or backtick (```) quotes.

JetBrains Mono
```
"Hello, World", '42 Green Avenue'
```

`number`

Any number without quotes, including integers and decimals (floats).

JetBrains Mono
```
12, 9.99, -6.3456
```

`boolean`

Represents a logical entity and can have only two values: `true` or `false`.

JetBrains Mono
```
isLoggedIn = true;
```

`null`

Represents the intentional absence of any value. It signifies 'this deliberately has no value.'

JetBrains Mono
```
testScore = null;
```

JavaScript is a ***loosely-typed*** language. You don't specify the data type when creating a variable; JavaScript infers it from the value you assign.

Editorial Tech

Chapter 3: The Language of Logic

Performing Arithmetic with Numbers

Operator		Key	Description	Example (firstNum=12, secondNum=8)	Result
+	`+`	Plus	Addition	`firstNum + secondNum`	20
−	`-`	Dash	Subtraction	`firstNum - secondNum`	4
×	`*`	Asterisk	Multiplication	`firstNum * secondNum`	96
÷	`/`	Slash	Division	`firstNum / secondNum`	1.5

Controlling the Order of Operations

JavaScript follows standard mathematical rules for calculations: division and multiplication are performed before addition and subtraction.

Example 1: Default Order

```
console.log(5 + 2 * 3);
```

JavaScript first multiplies `2 * 3` (equals 6), then adds 5.

11

Example 2: Using Parentheses `()`

```
console.log((5 + 2) * 3);
```

You can force a different order using parentheses. Here, `5 + 2` is calculated first (equals 7), then multiplied by 3.

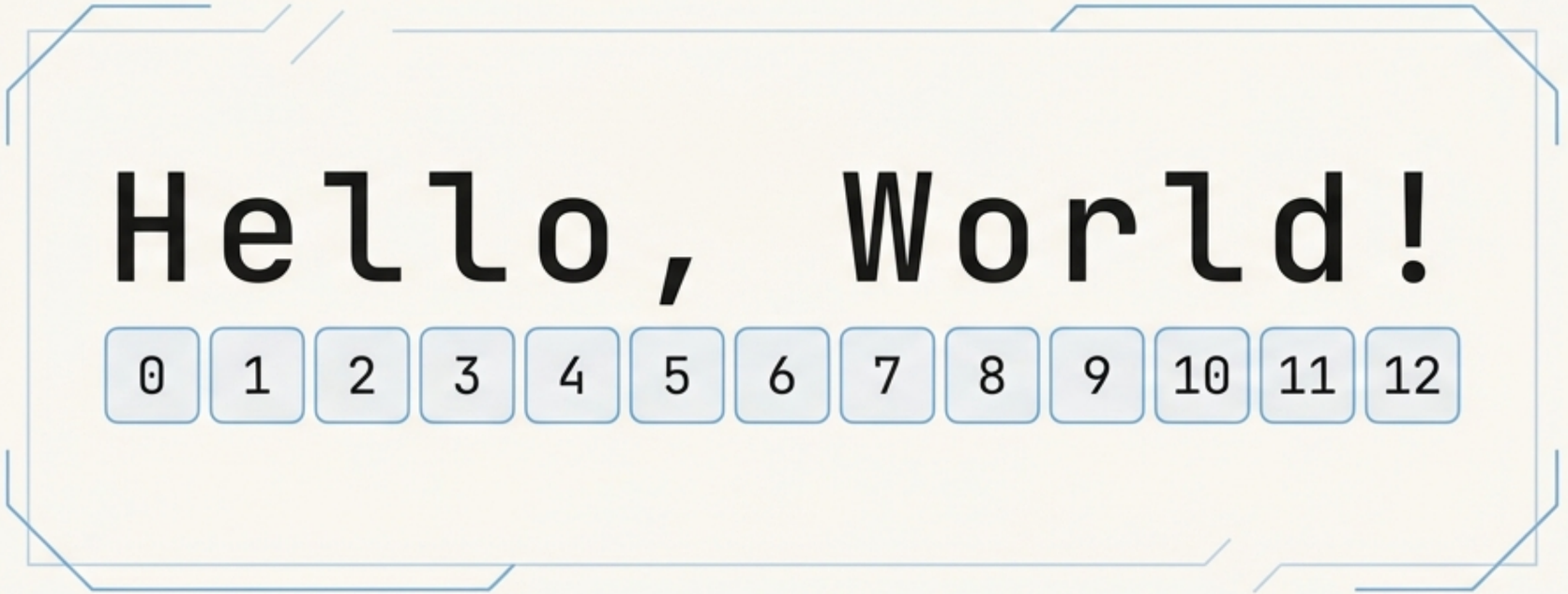
21

Use parentheses to make your calculations **explicit** and **avoid ambiguity**.

Chapter 4: The Language of Text

The Anatomy of a String

In programming, text is called a *string*. JavaScript assigns an index number to each character in a string, starting from 0. This is called *zero-based indexing*.



Key Properties

- Accessing a character:

  `myStr6[0]` returns `'H'`.
  `myStr6[7]` returns `'W'`.

- Finding the length:

  `myStr6.length` returns `` `13` ``.
  (Note: The length is always one greater than the highest index).

Your String Manipulation Toolkit

1. Investigating Strings

```
let myStr = "Hello, World!";
myStr.includes("World");    // returns true
myStr.startsWith("Hello");  // returns true
myStr.indexOf("World");     // returns 7
```

3. Chaining Methods

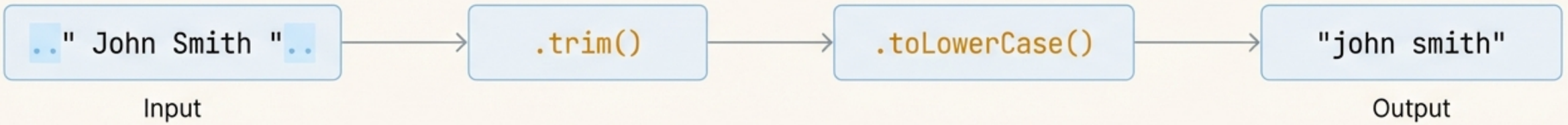
You can combine methods together in a single expression for powerful, concise transformations.

```
// Trim whitespace AND convert to lowercase in one go
let cleanUserName = strUserName.trim().toLowerCase();
strUserName.toUpperCase(); // "  JOHN SMITH "
```

2. Transforming Strings

```
let strUserName = "• John Smith •";
```

```
strUserName.toUpperCase(); // " JOHN SMITH "
```



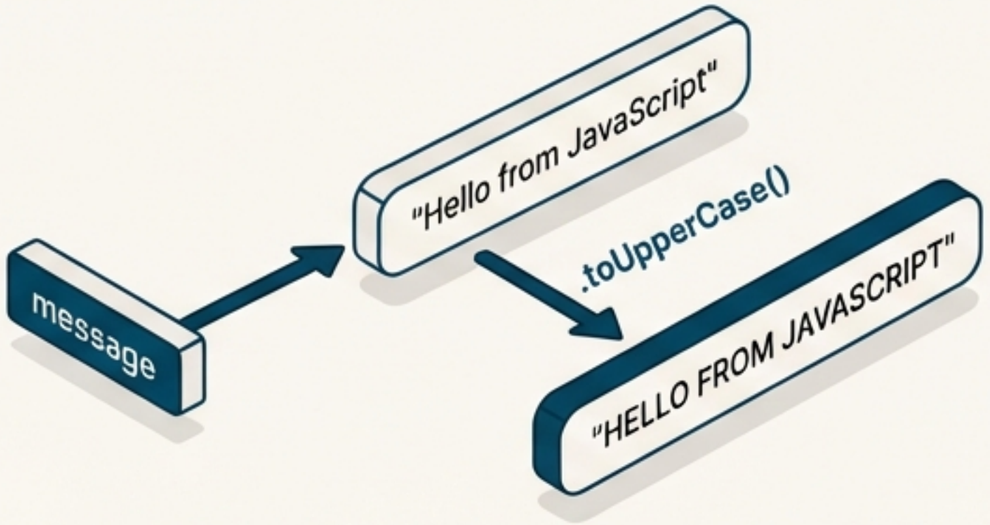
The Golden Rule of Strings: They Are Immutable

String methods do **not** change the original string. Instead, they *create and return a new string* with the changes applied.

The Common Mistake

Inter SemiBold

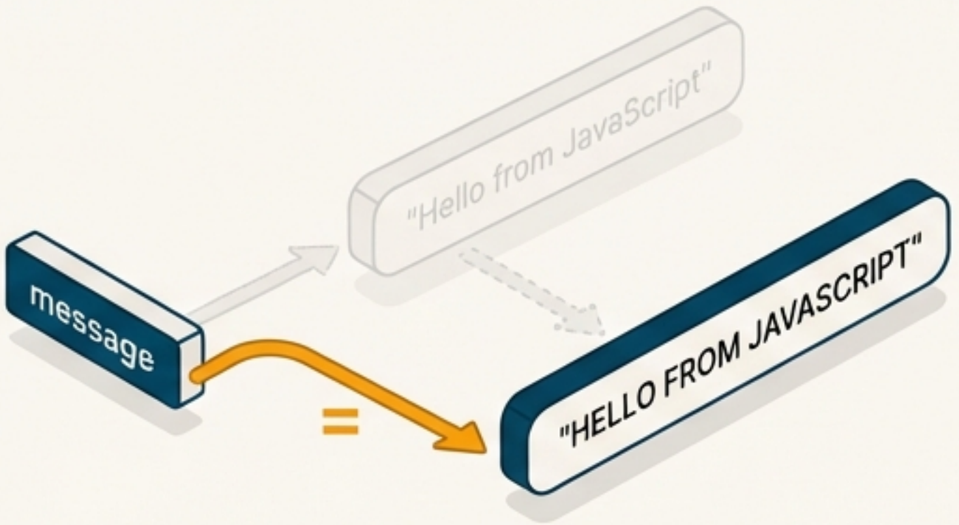
```
let message = "Hello from JavaScript";

message.toUpperCase(); // This does NOTHING to the original 'message'
console.log(message);  // Outputs: "Hello from JavaScript"
```



The Correct Approach

Inter SemiBold

```
let message = "Hello from JavaScript";
// Assign the NEW, transformed string back to the original variable
message = message.toUpperCase();
console.log(message); // Outputs: "HELLO FROM JAVASCRIPT"
```



The Modern Way to Combine Text and Data

The Old Way

Joining strings with `+` can be clumsy, especially with multiple variables and spaces.

```
txtFullName = txtFirstName + " " + txtLastName;
```

The New Way RECOMMENDED

A far more readable and powerful method. Enclose the entire string in backticks (`` `` ``) and embed variables directly using the `${...}` syntax.

```
const customerFirstName = "Marie";
const accBalance = 10;

console.log(`Hi ${customerFirstName}. Your
balance is €${accBalance}.`);
// Outputs: Hi Marie. Your balance is €10.
```

Template literals are the preferred method for building strings in modern JavaScript.

Your Journey Continues

Summary of Your New Skills

- You can now successfully **connect** JavaScript to a web page.
- You understand how to **store** information using `let` and `const` variables.
- You can perform **calculations** with numbers.
- You can inspect, transform, and build complex **strings** with ease.

What's Next?

These foundational skills are the building blocks for more advanced concepts. Your next steps in the JavaScript journey will likely involve:

- **Functions**: Creating reusable blocks of code.
- **Decision Making**: Using `if`/`else` to make your code react to different conditions.
- **Arrays & Objects**: Storing collections of related data.