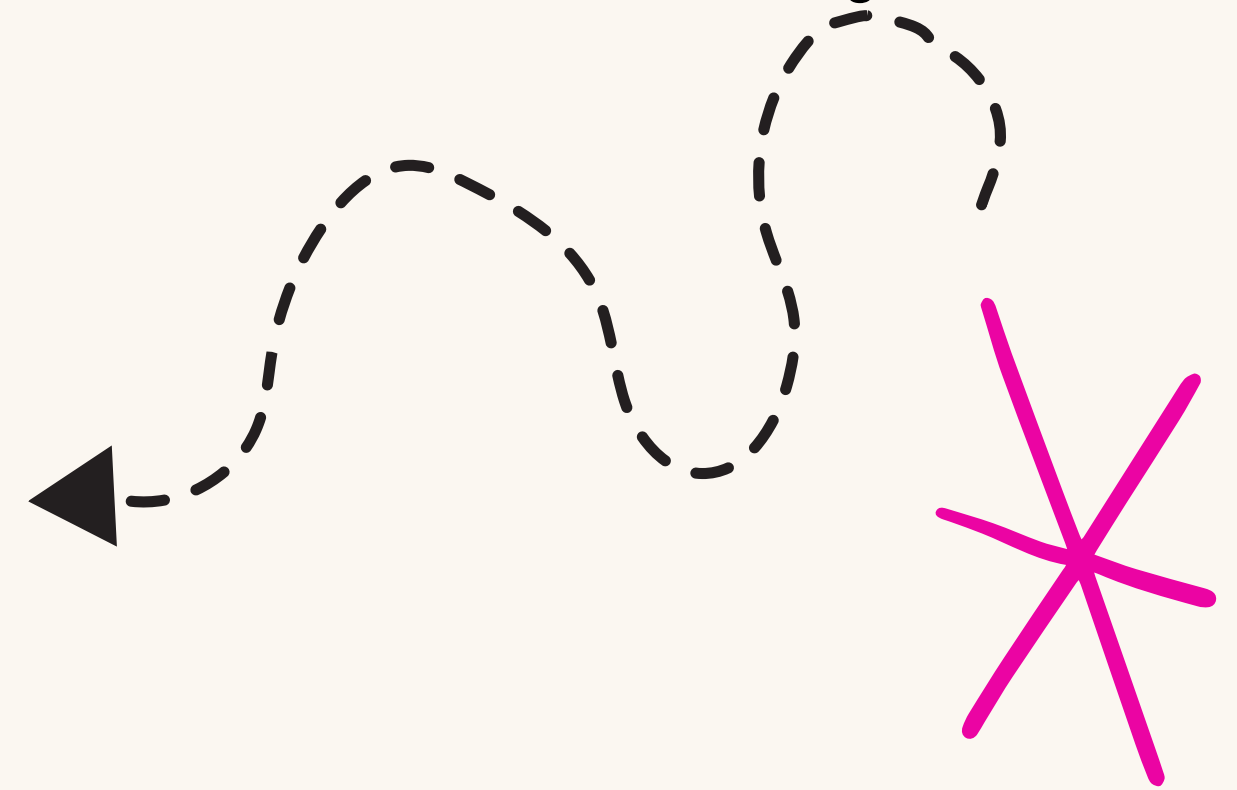



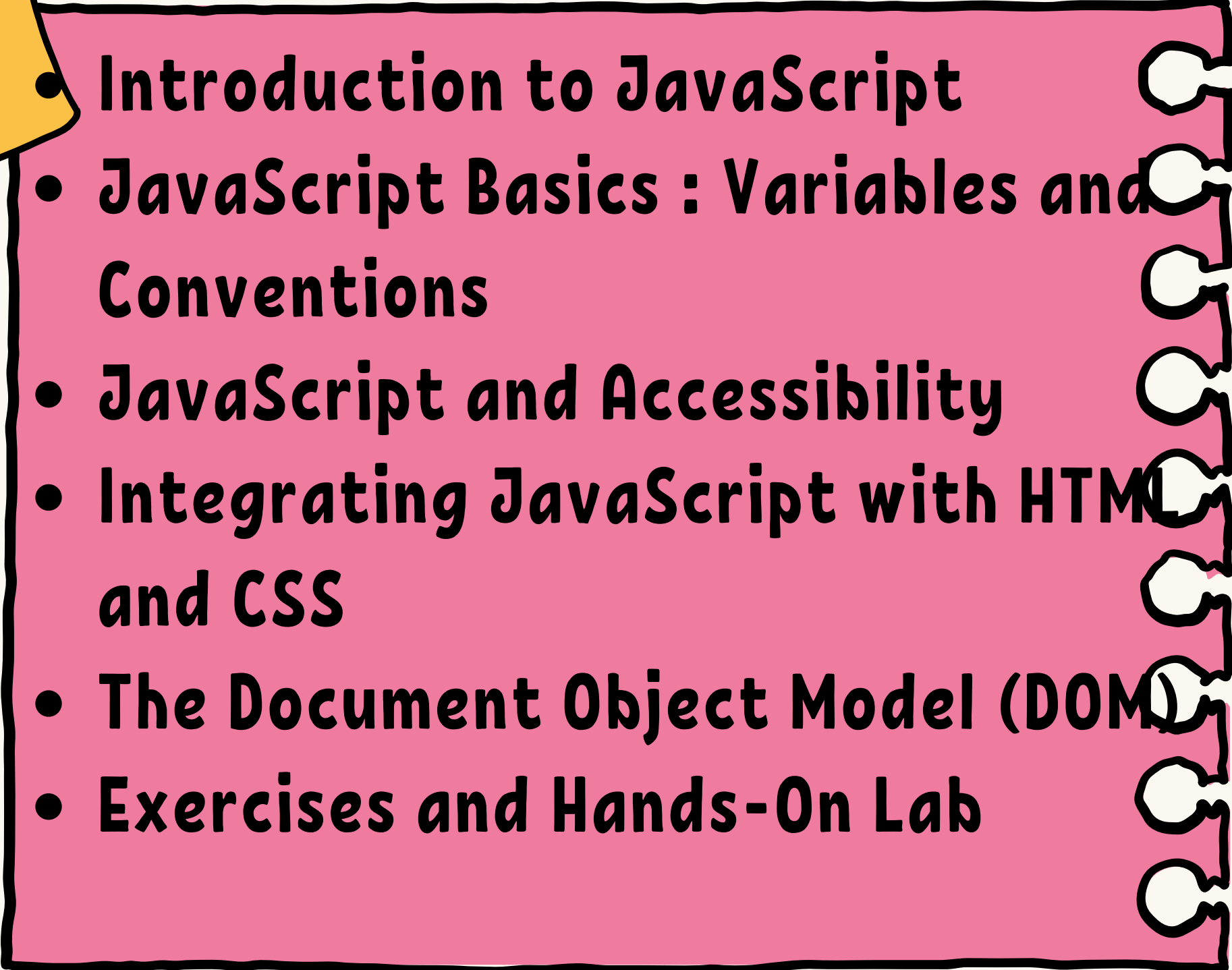
by Elena Petrova

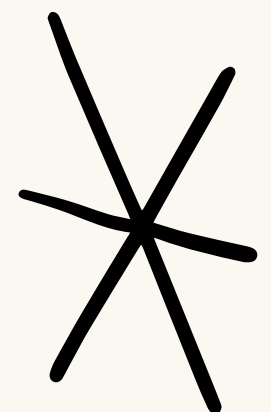
Introduction to JAVASCRIPT





Agenda

- 
- Introduction to JavaScript
 - JavaScript Basics : Variables and Conventions
 - JavaScript and Accessibility
 - Integrating JavaScript with HTML and CSS
 - The Document Object Model (DOM)
 - Exercises and Hands-On Lab



What is JavaScript?

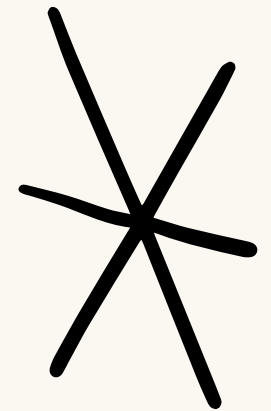
Definition: JavaScript is the language that makes web pages interactive. Imagine a static image turning into an interactive game—JavaScript does that.

Example: When you click a button, and the page changes without reloading, that's JavaScript in action.



JavaScript vs. Other Languages

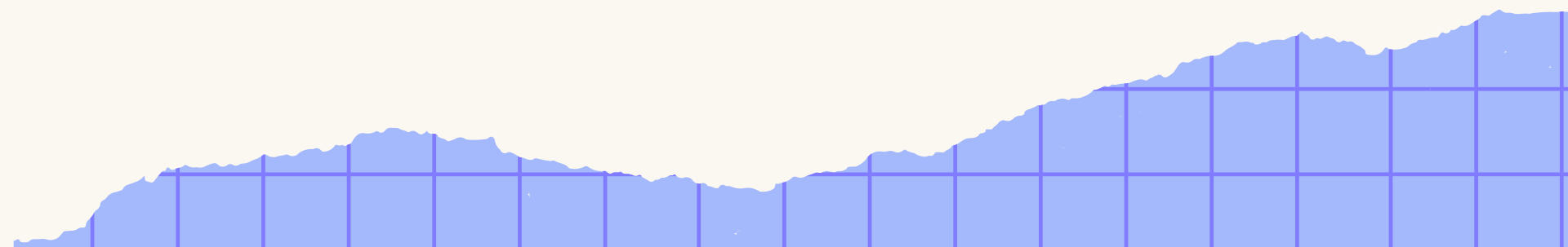
- HTML: Structures web content.
- CSS: Styles web content.
- JavaScript: Adds interactivity to web content.



JavaScript : Case sensitivity

JavaScript pays attention to whether letters are uppercase or lowercase in variable and function names.

For instance, "myvar" and "Myvar" are treated as different names.





JavaScript Variables

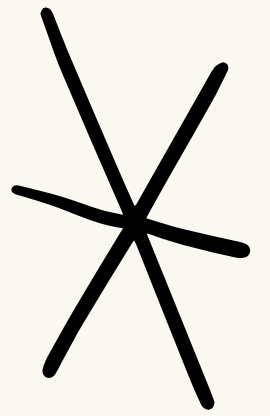
What is a Variable?

A variable is like a nickname for something you want to remember. Imagine writing a name on a sticky note and sticking it to a number or a word. You can use that name later to find or change what's written.

Rules for Naming Variables:

- A variable name must start with a letter(A-Z ,a-z), an underscore(_), or a dollar sign(\$).
- After the first character, variable names can also include numbers(0-9).

How to Declare Variables:



In JavaScript, you can declare a variable using one of three keywords: `var`, `let`, and `const`.

var

Think of it as the old-school way to declare a variable.
It works everywhere in a function but can cause confusion if not
careful.

Example:

```
javascript
```

```
var name = "Alice";  
console.log(name); // Alice
```

let

A better way to declare a variable when the value might change.

It only works within the block { } where it's declared.

Example:

javascript

```
let age = 25;  
console.log(age); // 25  
age = 30; // You can update the value  
console.log(age); // 30
```

const

Use this for variables that shouldn't change.
Once you assign a value, you can't change it.

Example:

javascript

```
const birthYear = 1990;  
console.log(birthYear); // 1990  
// birthYear = 2000; // This will give an error
```

What is a Function?

A function is like a small helper that does a specific job for you. You give it some information (input), it works on it, and then gives you a result (output).

javascript

```
function greet(name) {  
    return "Hello, " + name + "!";  
}
```

// Use the function

```
console.log(greet("Alice")); // Output: Hello, Alice!  
console.log(greet("Bob"));   // Output: Hello, Bob!
```



Scope in JavaScript

Scope is like the rules about where your variables and functions can be seen or used in your code. It's about knowing where they "live" and where you can access them.

Global Scope

Variables declared outside any block `{ }` or function are global.
They can be used anywhere in your code.

Example:

javascript

```
let globalVar = "I'm global!";

function showGlobal() {
  console.log(globalVar); // Works here
}

showGlobal();
console.log(globalVar); // Works here too
```

Function Scope

Variables declared inside a function can only be used inside that function.

Example:

javascript

 Copy code


```
function showMessage() {  
  let message = "Hello from inside the function!";  
  console.log(message); // Works here  
}  
  
showMessage();  
console.log(message); // Error! 'message' is not accessible here
```


Block Scope (For let and const)

Variables declared inside a block { } can only be used inside that block.

Example:

javascript

 Copy code

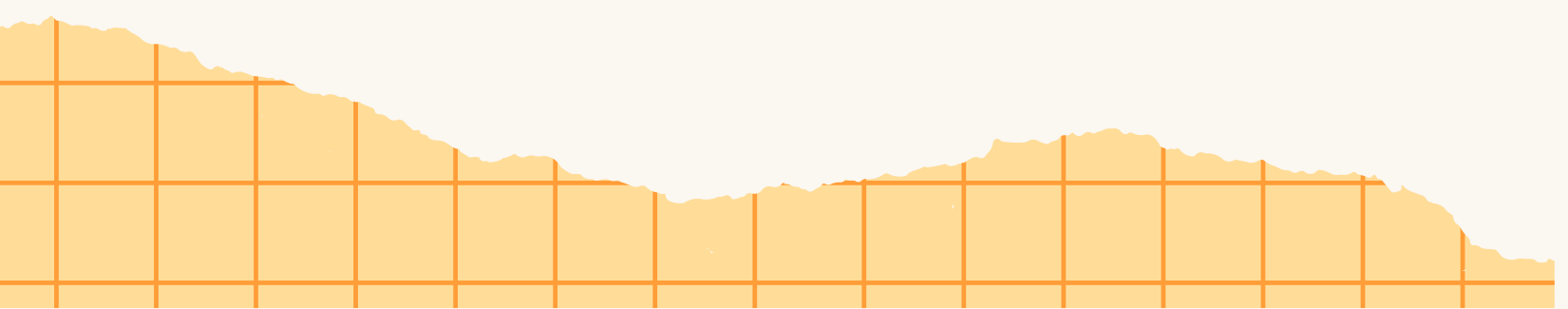
```
if (true) {  
  let blockVar = "I live inside this block";  
  console.log(blockVar); // Works here  
}
```

```
console.log(blockVar); // Error! 'blockVar' is not accessible here
```



Why is Scope Important?

It helps organize your code and avoid mistakes, like accidentally changing a variable that another part of your code is using.



Literals & Expressions

In JavaScript, literals represent fixed values that are not variables. Literals are fixed values you use directly in your code, not stored in variables.

Array Literals: Represent lists of data enclosed in square brackets [].

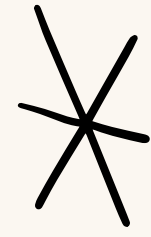
Boolean Literals: Represent true or false values: true or false.

Integer Literals: Represent whole numbers: 42, -10, 0.

Floating-Point Literals: Represent decimal numbers: 3.14, -0.01.

Object Literals: Represent key-value pairs enclosed in curly braces { }.

String Literals: Represent sequences of characters enclosed in single ' ' or double " " quotes.



Array Literals

A list of items inside square brackets [].

Example:

javascript

```
let fruits = ["apple", "banana", "cherry"];  
console.log(fruits[0]); // Output: apple
```

Boolean Literals

Values that represent either true or false.

Example:



Integer Literals

Whole numbers without decimals.

Example:

javascript

```
let count = 42;  
console.log(count); // Output: 42
```

Floating-Point Literals

Numbers with decimals.

Example:

javascript

```
let price = 19.99;  
console.log(price); // Output: 19.99
```

Object Literals

Key-value pairs inside curly braces { }.

Example:

javascript

```
let person = { name: "Alice", age: 25 };  
console.log(person.name); // Output: Alice
```


String Literals

Text enclosed in single `' '` or double `" "` quotes.

Example:

javascript

```
let greeting = "Hello, world!";  
console.log(greeting); // Output: Hello, world!
```

Expressions

An expression is any valid unit of code that resolves to a value.

For example:

javascript

```
let sum = 5 + 3;  
console.log(sum);
```

Operators in JavaScript

JavaScript includes various types of operators for performing operations on variables and values.

1. Arithmetic Operators

- Addition(+), Subtraction(-), Multiplication(*), Division(/)

2. Comparison Operators

- Equalto(==), Notequalto(!=), Strictequalto(===), Greaterthan(>), Lessthan(<)

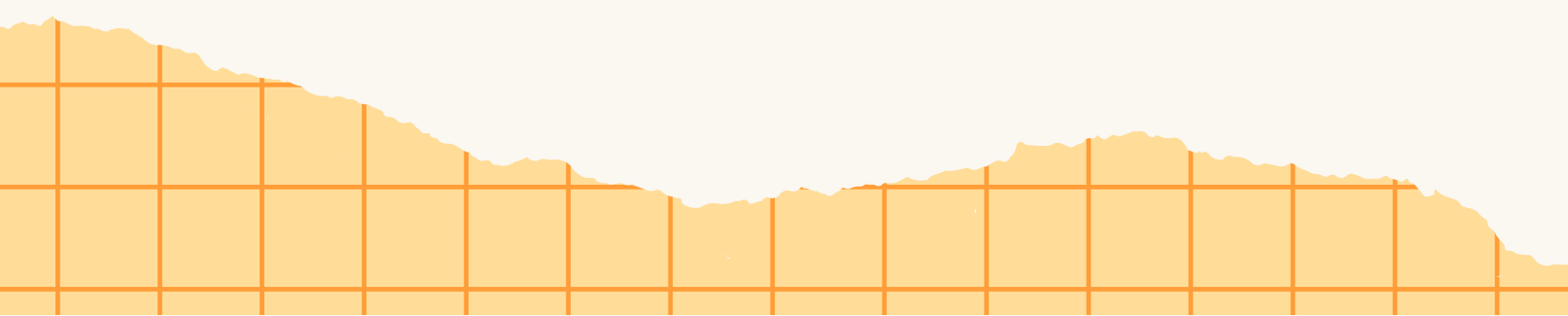
3. Logical Operators

- LogicalAND(&&), LogicalOR(||), LogicalNOT(!)



JavaScript Syntax

JavaScript syntax defines the rules for writing code in JavaScript. Understanding these rules is essential for creating functional and readable programs.



JavaScript is picky about uppercase and lowercase letters.
myVariable and myvariable are different things.

Example:

```
javascript
```

```
let myVariable = 10;  
console.log(myVariable); // Works  
console.log(myvariable); // Error!
```



Variables

Use `let`, `const`, or `var` to store values.

Example:

javascript

```
let age = 25; // Stores the number 25  
const name = "Alice"; // Stores the name Alice
```

Comments

Comments are notes in your code. They don't run.

```
javascript
```

```
// This is a comment
```

```
javascript
```

```
/*  
  This is a  
  multi-line comment  
*/
```

//

Conditional Statements

Use if to run code only if something is true.

```
if (age > 18) {  
    console.log("You are an adult!");  
} else {  
    console.log("You are not an adult yet!");  
}
```


Functions

Functions let you reuse code.

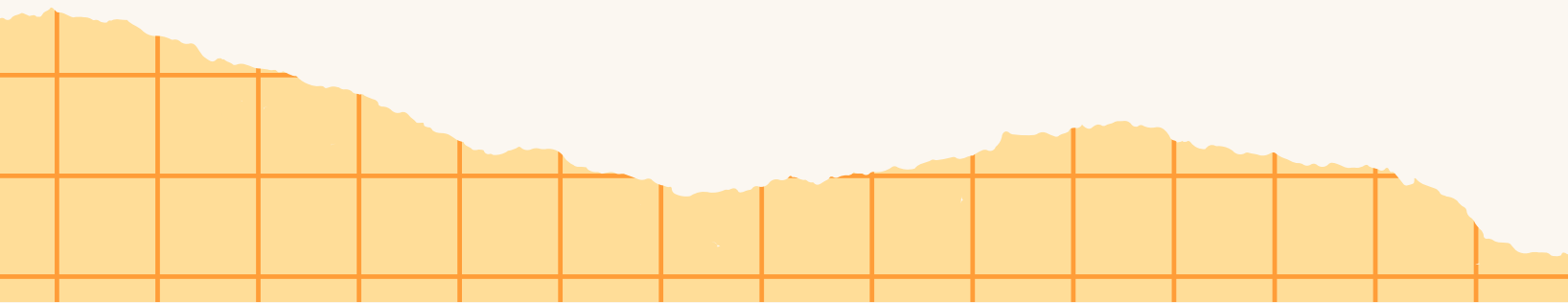
Example:

javascript

```
function sayHello() {  
    console.log("Hello, world!");  
}  
  
sayHello(); // Calls the function
```

JavaScript and Accessibility

JavaScript can make websites more inclusive and user-friendly, especially for people with disabilities. Here are two key ways it helps:



1. Keyboard Navigation


What It Means: Not everyone uses a mouse. Some users rely on keyboards to navigate, especially those with mobility challenges.

What You Can Do:


Make sure buttons, links, and form fields can be accessed using the Tab key.

Highlight the active element (focus) using CSS or JavaScript.

Example:



javascript

 Copy code

```
document.querySelector("button").addEventListener("keydown", function(event) {  
    if (event.key === "Enter") {  
        console.log("Button clicked via keyboard!");  
    }  
});
```



Form Validation Messages

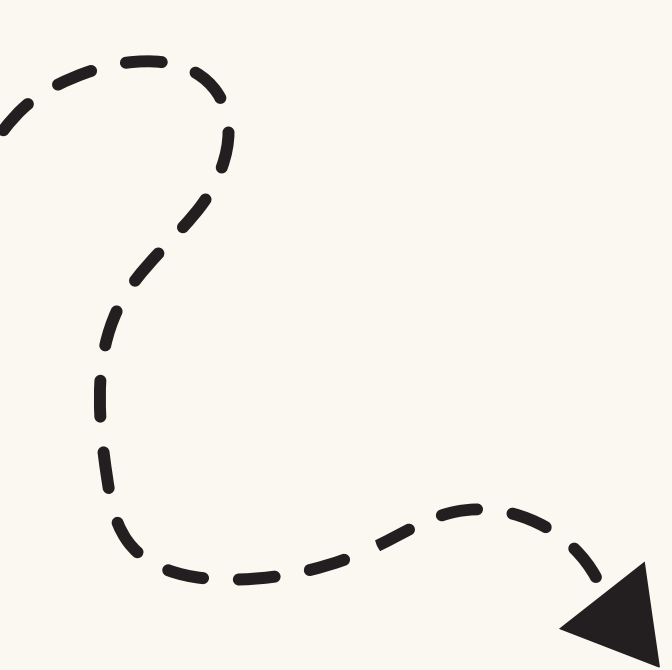
What It Means: When users fill out forms, they should get clear and understandable error messages if something is wrong.

These messages should also work with screen readers.

What You Can Do:

Use JavaScript to display error messages near the input field.
Add aria-live attributes so assistive technologies can announce the messages.

Example:



javascript

```
let form = document.querySelector("form");
form.addEventListener("submit", function(event) {
  let input = document.querySelector("#email");
  if (!input.value.includes("@")) {
    event.preventDefault(); // Stop form submission
    let error = document.querySelector("#error");
    error.innerText = "Please enter a valid email.";
    error.setAttribute("aria-live", "polite"); // Announce error
  }
});
```



Integrating JavaScript with HTML and CSS

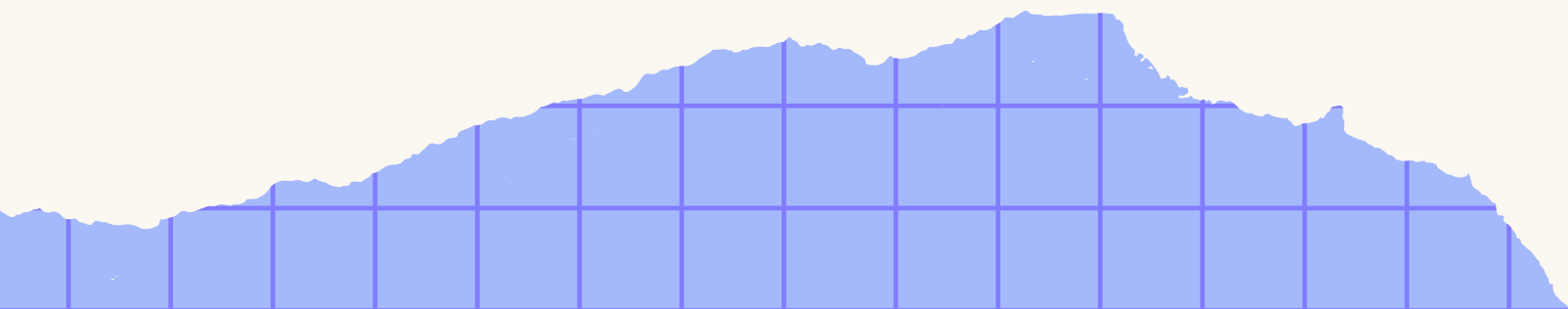
Why Integrate JavaScript?

HTML: Provides the structure (e.g., buttons, text).

CSS: Adds styles (e.g., colors, fonts).

JavaScript: Brings interactivity (e.g., clicks, animations).

Together, they make websites functional, beautiful, and engaging!



The Document Object Model

How JavaScript Uses the DOM

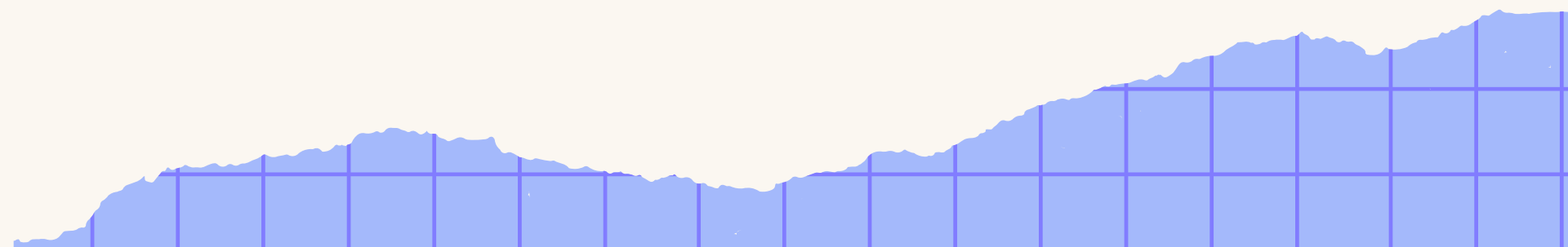
JavaScript can:

Find elements (e.g., headings, buttons).

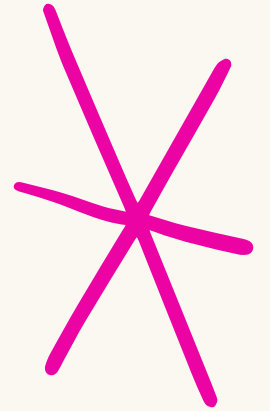
Change content (e.g., update text).

Change styles (e.g., change background color).

React to events (e.g., clicks, hover).



Why the DOM Matters

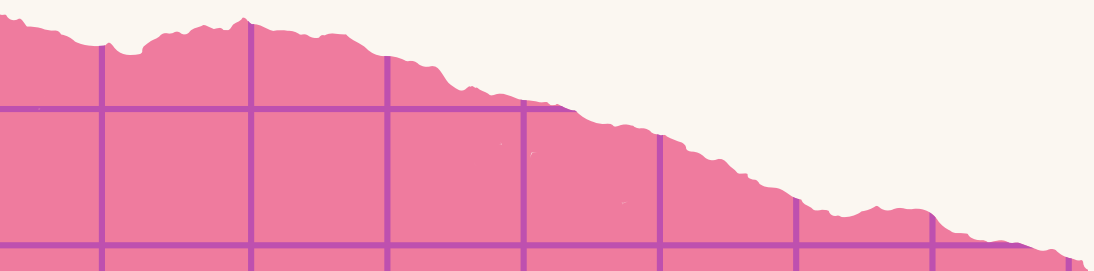


It lets JavaScript "talk" to your webpage.

You can make changes without reloading the page.

It powers dynamic and interactive websites.

Mastering the DOM is key to creating engaging web experiences!





Selecting DOM Elements

JavaScript provides several methods to select and access HTML elements for manipulation.

`getElementById`: Selects an element by its ID.

`getElementsByClassName`: Selects elements by their class name.

`getElementsByTagName`: Selects elements by their tag name.

`querySelector`: Selects the first element that matches a CSS selector.

`querySelectorAll`: Selects all elements that match a CSS selector.



THANK
YOU!

