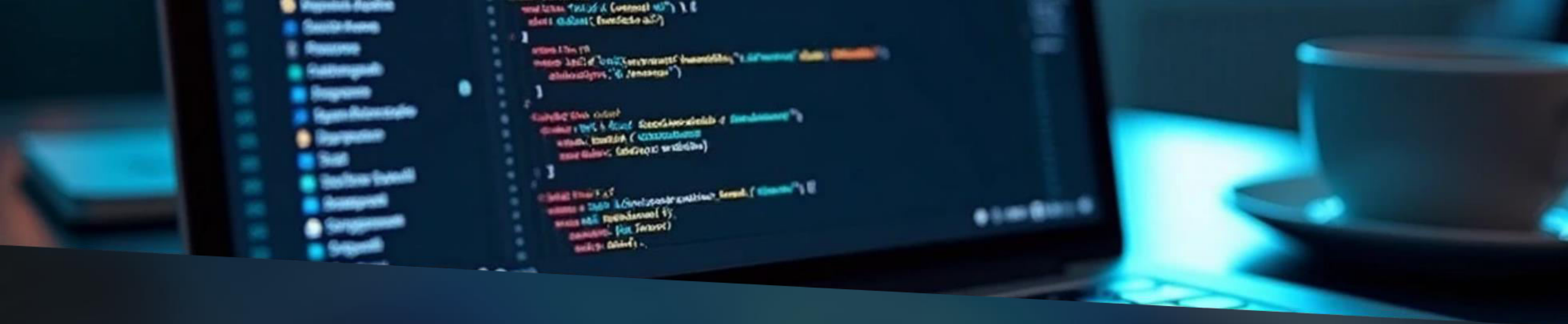


# Introduction to JavaScript Concepts

M Author: Madiha Ayaz

Created with Pi

# CONTENTS

# Slide:1 Suhaira

## 1. Loops

### While Loops

What is a While Loop? A while loop repeatedly executes a block of code as long as a specified condition is true.

while (condition) { // code to be executed }

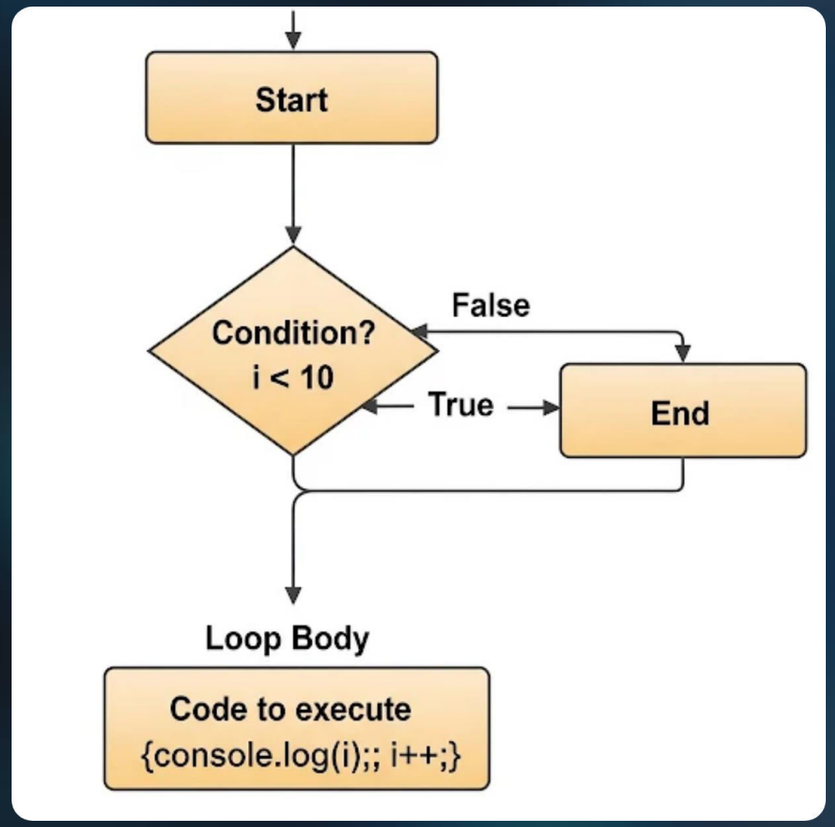
Key Point

The condition is checked before each iteration. If the condition is initially initially false, the loop's code block will never run.

Syntax:

```
let count = 1;

while (count <= 5) {
    console.log("Count is: " + count);
    count++;
}
```

**illustrating the structure of a JavaScript while loop.**




A while loop checks a condition first.

If the condition is true, it runs a block of code (the loop body)

After the code runs, it checks the condition again

This process repeats until the condition becomes false.

Once the condition is false,

the loop ends.

# Do...While Loops

1

**What is a Do...While Loop? A do...while loop is similar to a while loop, but it guarantees that the code block will execute at least once**.

2

Syntax:

```
do {
  // code to be executed
} while (condition);
```

```
// Example: Counting from 1 to 5
let count = 1;

do {
  console.log("Count is: " + count);
  count++;
} while (count <= 5);
```

Key Point

The code block runs first, and the condition is checked after each iteration.

# 2. Script Placement

## Placing Scripts

```
document.body.style.backgroundC
olor = "lightyellow";

      console.log("JavaScript is
running after HTML is loaded!");

   <script>

   <!-- Script placed at the end -->



   <p>This text will load before the
script runs.</p>

   <h1>Hello World!</h1>
```

Option 2: At the end of the section

Why place the script here?

The HTML is fully loaded before the JavaScript runs.

No delay in showing the page's page's content.

Simple and effective for most small scripts.

# 3. Commenting in Code

## Commenting

### Clarity

Comments explain what your code does, making it easier for you and others to understand.

### Types of Comments

- Single-line Comment: Starts with //
- Multi-line Comment: Starts with /* and ends with */

# Slide:2 Alishba

# 4. Events

## Events: Link

```
</body>
</script>
```

User interaction with links. Example:

```
});

    alert("You clicked the link!");

    event.preventDefault(); // Prevents the link from navigating

document.getElementById("myLink").addEventListener("click", function(event) {

<script>
```

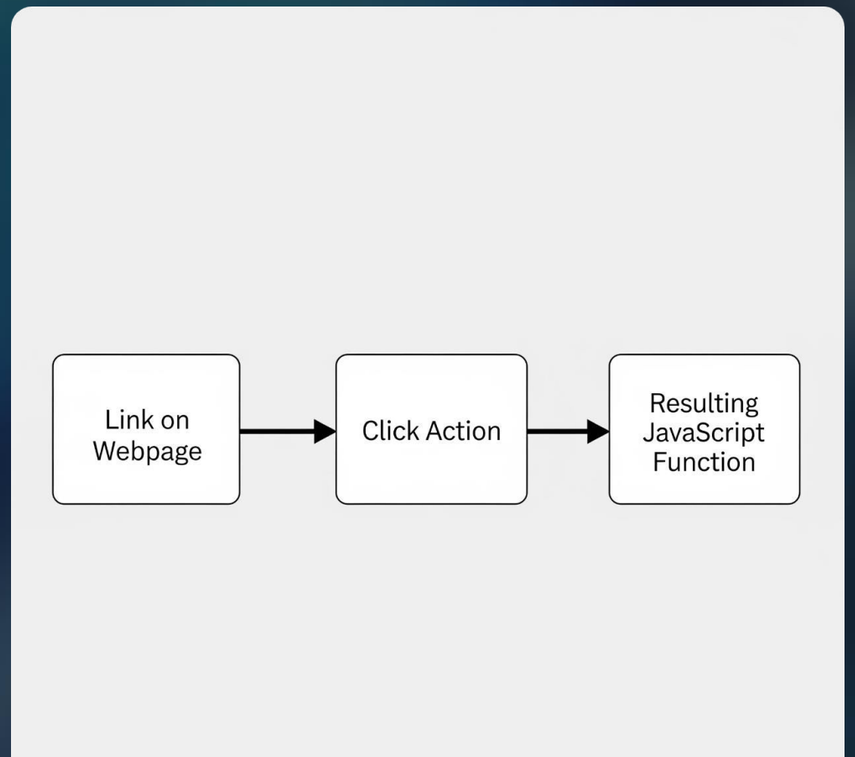
```
<a href="#" id="myLink">Click Me</a>
```

Click Me

```
<!-- Link that triggers an event -->

<body>
```

# Here is a diagram that explains how JavaScript events events work with a link.

When you click on a link, an event happens. You can tell JavaScript to listen for that specific event and run a function instead of going to a new page.



# Events: Button

1

User interaction with buttons.

Event listeners for a button wait for a specific action, like a click. When the action happens, the listener runs a function you've defined, which allows you to perform an action.

Click Me

2

Example

```
</script>
});

alert("Button was clicked!");


document.getElementById("myButton").addEventListener("click", function() {

<script>
```

# Events: Mouse

Common mouse events onmouseover (mouse enters), onmouseout (mouse leaves).

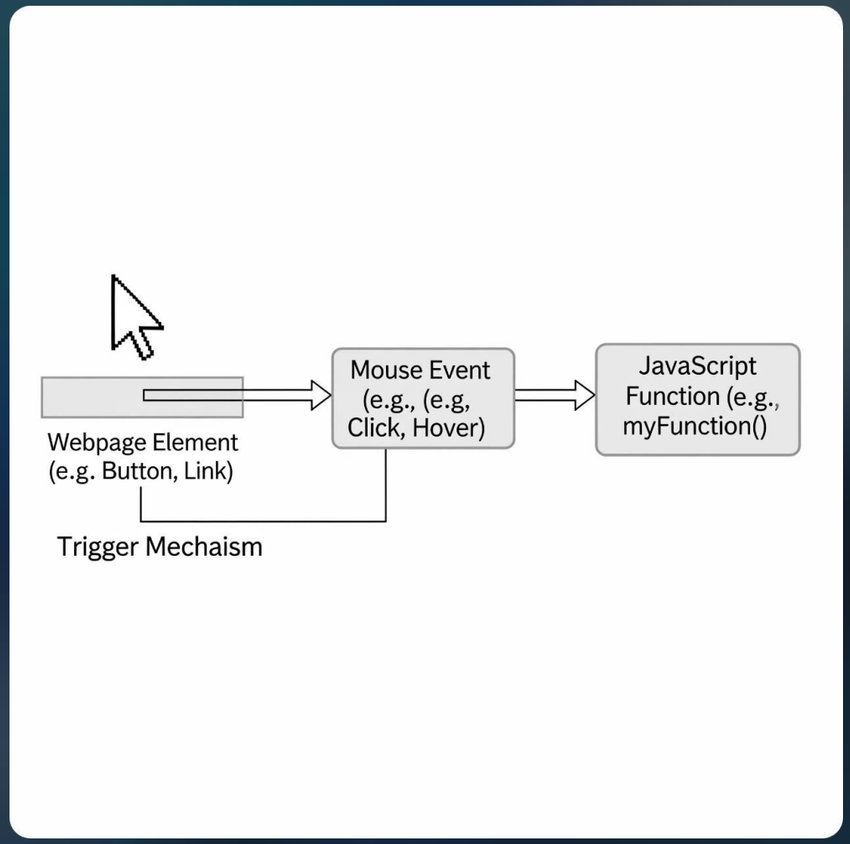
```
<div onmouseover="this.style.color='red'">Hover over me</div>
```

# Here is a simple diagram that shows how mouse mouse events work with JavaScript.

1

Event listeners are like watchers for elements on a webpage. When a user interacts with an element (like a button or link) in a certain way, the event listener detects that action and then runs a specific function. This is how you make a webpage interactive.

2



Slide:3 Mahenoor

# Events: Fields

Concept: Handling events that occur within input fields.

Common Events:

onfocus: When the field is selected.

onblur: When the field is deselected.

onchange: When the value is changed and the field is deselected.

Visual Suggestion: An input field with a border changing color when focused.

Form inputs and user data. OUTPUT

**Example Output:**

Type something...

```
<input type="text"
    onfocus="this.style.border='2px solid
```

# 5. Reading and Setting Field Values

## Reading Field Values

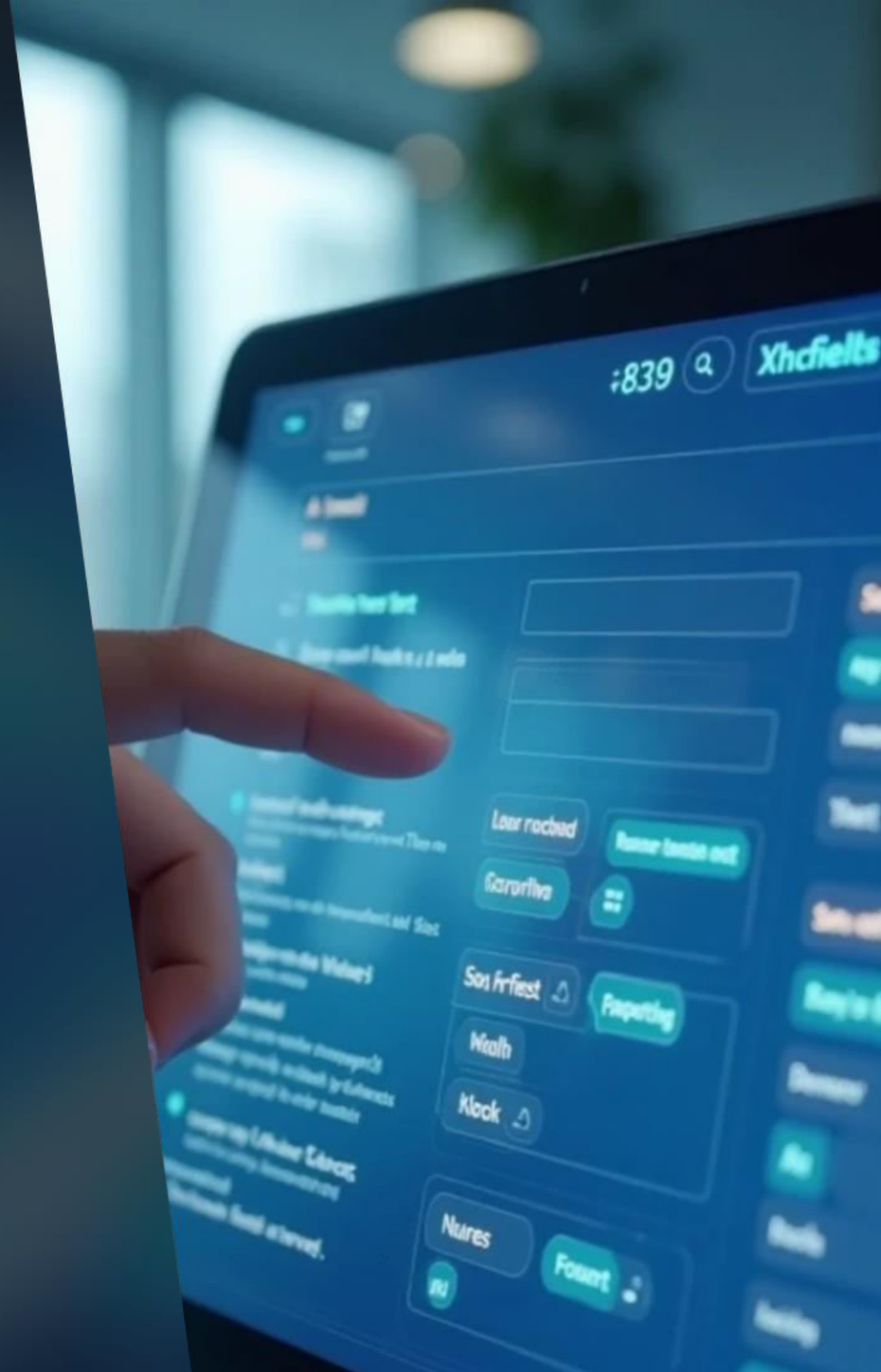
• The .value Property: This is the most common way to get the content of an input field.

## Setting Field Values

• You can assign a new value to the .value property.

```
//html
<input type="text" id="usernameInput" value="JohnDoe">
```

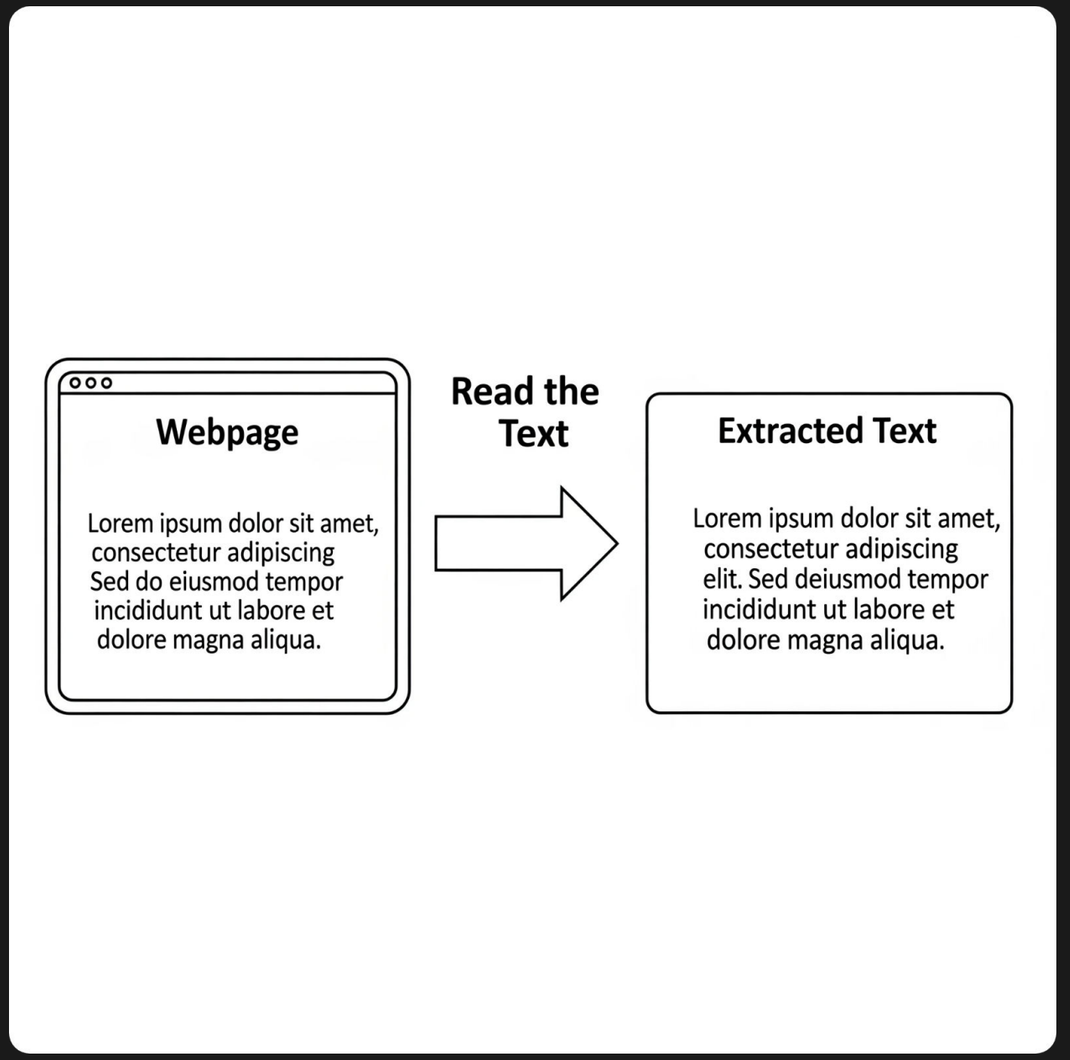
```
//javascript code
// Get the input element by its ID
const usernameField = document.getElementById("usernameInput");
document.getElementById("usernameInput");

// Read the value of the input field

const username = usernameField.value;
```



# Slide:4 Sadaf

## Reading Paragraph Text

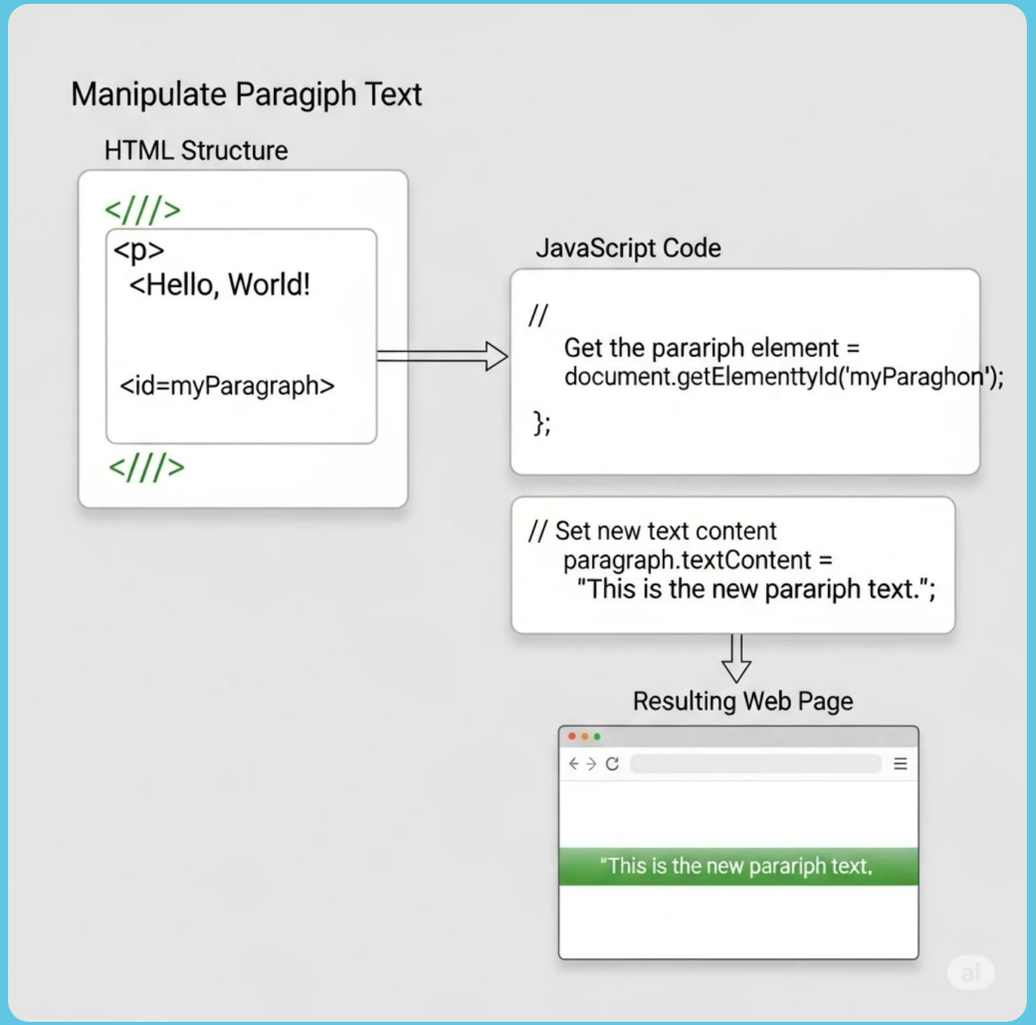
```
// Get the paragraph element with the ID "myParagraph"

const myParagraph = document.getElementById('myParagraph');

// Read the text content

const paragraphText = myParagraph.textContent;

// Log the text to the console

console.log(paragraphText);
```



# Setting Paragraph Text

.textContent property.

```
const myParagraph = document.getElementById('myParagraph');

// Set new text for the paragraph

myParagraph.textContent = 'This is the new paragraph text.';
```



# 6. Text and Image Manipulation

1 Reading and Setting Paragraph Text

- Use .textContent for safe text manipulation.

2 Manipulating Images and Text

- Changing an image's source: Use the .src property.

```
<p id="myParagraph">This is the original text.</p>
```

```
/ Get the paragraph element                                    //javascript
const myParagraph = document.getElementById("myParagraph");

// Read the text content

const currentText = myParagraph.textContent;

console.log(currentText); // Output: "This is the original text."

// Set new text content

myParagraph.textContent = "This is the new text.";
```

# Slide:5 Madiha

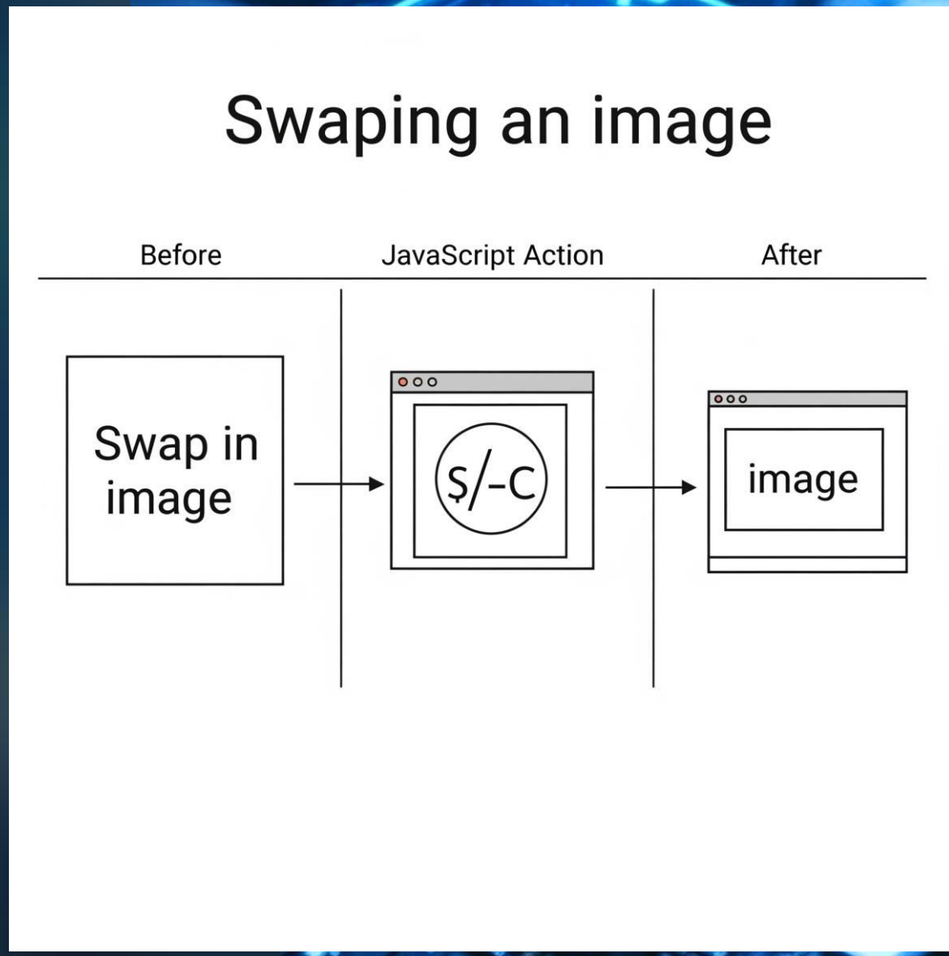
# 7. Swapping Images

## Swapping Images

- Basic image swap: Change the .src attribute.
- Example:

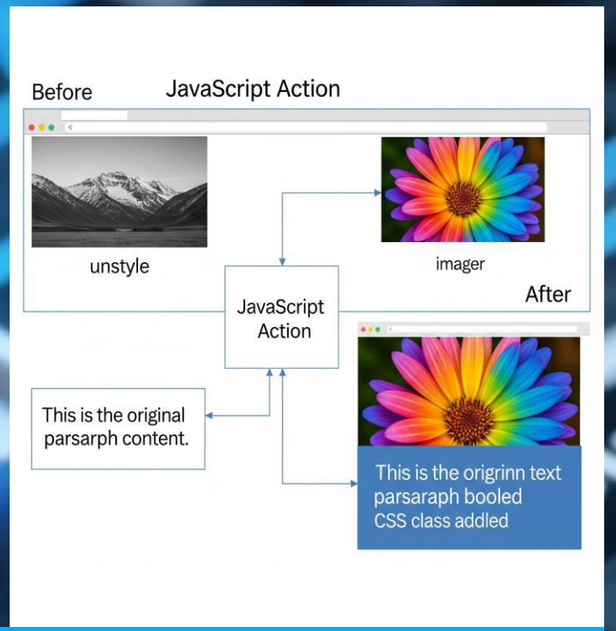
```
<button onclick="swapImage()">Swap Image</button>   //html
<img id="myImage" src="images/pic1.jpg" />
```

```
    document.getElementById("swapBtn").addEventListener("click", function() {

      const imageElement = document.getElementById("myImage");

      imageElement.src = "images/pic2.jpg";
}   });
```



# 2. Swapping Images and Setting Classes

This involves both changing the image and applying a new CSS class to a different element on the page. By adding or removing a class, you can instantly apply a set of predefined styles (like color, font size, or background) to an element.




```
#myText {
  font-size: 18px;      color: black}

/* Highlight class styling */
.highlight {
  color: white;
  background-color: #ff8f9c;    font-size: 22px;
  padding: 5px;     border-radius: 5px;
}
<img id="myImage" src="images/pic1.jpg" alt="My Image" width="300">
<p id="myText">This is some text.</p>
<button id="changeBtn">Change Image & Highlight Text</button>
```

```
document.getElementById('changeBtn').addEventListener('click', function () {
  const myImage = document.getElementById('myImage');
  const myText = document.getElementById('myText');

  // Swap the image
  myImage.src = 'images/pic2.jpg';

  // Add the highlight class to the text
  myText.classList.add('highlight');
});
```

# Slide:6 Liaba

## 9. DOM Manipulation

### The DOM

• DOM stands for Document Object Model. It represents the page so that programs can change the document structure, style, and content.

### Target All Elements by Tag Name

•The getElementsByTagName() method returns a collection of all elements with a specified tag name (e.g., <p>, <h1>, ). This method returns a live HTMLCollection, which can be treated like an arrays

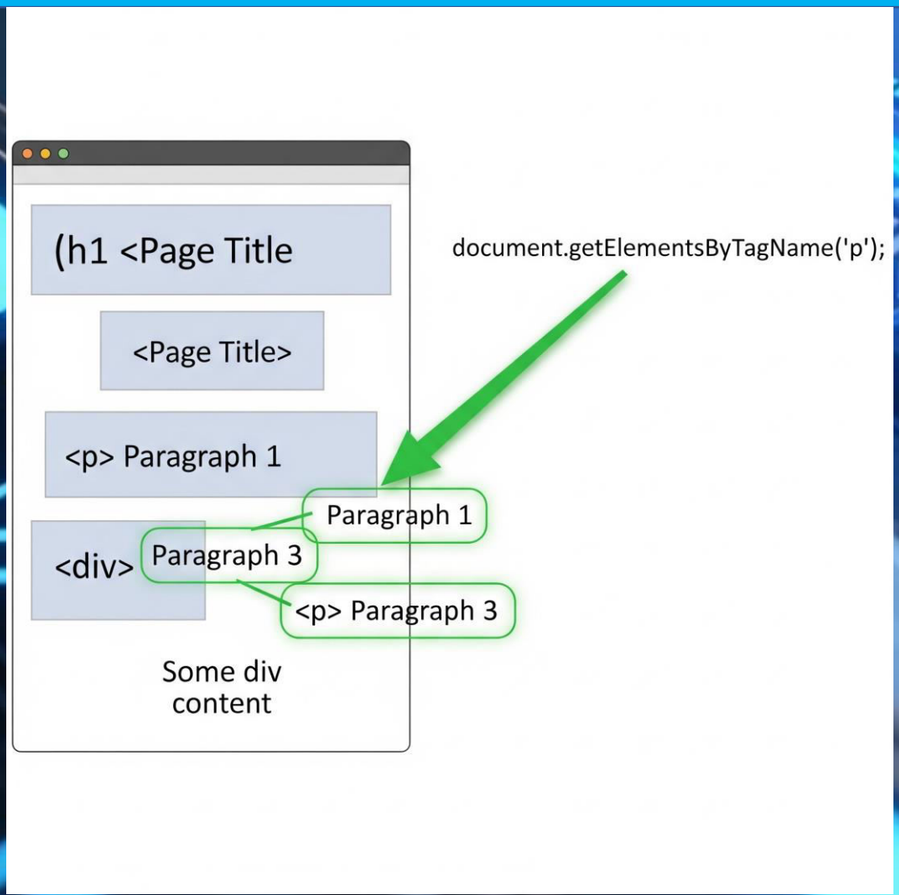
•

```
// This will get all <p> elements on the page
const allParagraphs =
document.getElementsByTagName('p');

// You can loop through them like an array
for (let i = 0; i < allParagraphs.length; i++) {
  // Change the text of each paragraph
  allParagraphs[i].textContent = "I am a paragraph!";
}
```

# 2. Target Some Elements by Tag Name

A more modern and flexible way to select elements is using querySelectorAll(). This method returns a NodeList of all elements that match a specified CSS selector. You can still loop through this list like an array.

Code:




```
// Get all <div> elements on the page
const allDivs =
document.querySelectorAll('div');

// Get all elements with the class 'important'
const importantElements =
document.querySelectorAll('.important');

// Get all <a> tags (links) that are inside an
element with the ID 'navigation'
const navLinks =
document.querySelectorAll('#navigation a');
```

# Here’s a clear, structured example covering all the points you mentioned for DOM Manipulation:

```
<body>
    <div id="container">
        <h2 class="title">First Title</h2>
        <h2 class="title">Second Title</h2>
        <p>This is a paragraph.</p>
    </div>
    <button id="tagButton">Target by Tag Name</button>
Name</button>
    <button id="classButton">Target by Class Name</button>
Name</button>
    <button id="childButton">Find Children</button>
```

```
        // Target all elements by tag name

document.getElementById("tagButton").addEventListener("c
lick", function() {
            let headings =
document.getElementsByTagName("h2");
    for (let i = 0; i < headings.length; i++) {
headings[i].style.color = "blue";
            }
        });

        // Target some elements by class name

document.getElementById("classButton").addEventListener(
```

# Slide 7 Rida

## The DOM: Parents and Children

**The DOM (Document Object Model) is a tree-like representation of a web page. Parents and children refers to the hierarchical relationship between the elements in this tree.**

**A parent is an element that contains other elements.**

**A child is an element that is directly inside another element.**

**In JavaScript, you can use properties like parentNode, children, firstElementChild, and lastElementChild to navigate this structure**

```
<div id="parent-div">
  <p id="child-p1">This is the first child.</p>
  <p id="child-p2">This is the second child.</p>
</div>
```

```
// Get a reference to a child element
const childElement = document.getElementById('child-p1');

// Use parentNode to get the parent element
const parentElement = childElement.parentNode;

console.log(parentElement.id);
// Output: "parent-container"
// This code defines "parentElement" as the parent of "childElement"."
```

# 2. Defining Children

**You can define a collection of child elements from a parent using the children property. This returns an HTMLCollection of all the direct child elements.**

```
// Get a reference to the parent element
const parentElement =
document.getElementById('parent-container');

// Use children to get all direct child elements
const childrenCollection =
parentElement.children;

console.log(childrenCollection.length);
// Output: 2

console.log(childrenCollection[0].id);
// Output: "child-p1"

// This code defines "childrenCollection" as the
children of "parentElement".
```

# Finding children

```
<ul id="my-list">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

**children: This property returns an array-like list of all a parent's direct child elements. You can access individual children using an index.**

```
const parent = document.getElementById('my-list');
const allChildren = parent.children;
console.log(allChildren.length); // 3

console.log(allChildren[0].textContent); // "Item 1"
console.log(allChildren[2].textContent); // "Item 3"
```

**firstElementChild**: This property returns only the very first child element.

**lastElementChild**: This property returns only the very last child element.

```
const parent = document.getElementById('my-list');
const firstChild = parent.firstElementChild;
console.log(firstChild.textContent); // "Item 1"
```

```
const parent = document.getElementById('my-list');
const lastChild = parent.lastElementChild;
console.log(lastChild.textContent); // "Item 3"
```

# Thank You

**Thank you so much for being such a great teacher. I've really learned a lot from you**