🧠 What are Events?

An **event** is an **action or occurrence** that happens in the browser, which JavaScript can respond to.

- Examples: User clicks a button, moves the mouse, types in a textbox, submits a form, loads a page, etc.
- **Purpose:** Events allow web pages to **react dynamically** to user actions.

🔔 How Events are Triggered

Events are triggered by **user actions** or **browser actions**:

User Actions:

- Clicking a button (`click`)
- Hovering over an element (`mouseover`)
- Typing in a text box (`keypress`, `input`)
- Selecting a checkbox (`change`)

Browser Actions:

- Page loading (`load`)

- Resizing window (`resize`)
- Scrolling (`scroll`)

⚙️ Ways to Create Events in JavaScript

There are **three main ways** to create event handlers:

1️⃣ Inline Event Handler (HTML attribute)

- Directly add the event in HTML using attributes like `onclick`, `onmouseover`, etc.

```
<button onclick="alert('Button Clicked!')">Click Me</button>
```

Output: Clicking the button shows an alert.

💡 **Note:** Simple but **not recommended for large projects** because HTML and JS are mixed.

2️⃣ Using DOM Property (Assign function to element property)

- Assign a JavaScript function to the **event property** of the element

```
<button id="btn1">Click Me</button>

<script>
  let button = document.getElementById("btn1");
  button.onclick = function() {
    alert("Button Clicked using DOM property!");
  }
</script>
```

Output: Clicking the button triggers the alert.

💡 **Note:** Only **one function can be assigned** to a property like `onclick`. If assigned again, previous handler is overwritten.

3️⃣ Using `addEventListener()` (Recommended)

- Attach **one or multiple event listeners** to an element
- Syntax:

```
element.addEventListener("eventType", functionName);
```

Example:

```
<button id="btn2">Click Me</button>

<script>
  let btn = document.getElementById("btn2");

  // Add click event
  btn.addEventListener("click", function() {
    alert("Button Clicked using addEventListener!");
  });

  // Add another event listener to the same button
  btn.addEventListener("mouseover", function() {
    btn.style.backgroundColor = "lightblue";
  });
</script>
```

Output:

- Click → alert
- Hover → background turns light blue

💡 **Advantages:**

- Can attach **multiple handlers**
- Keeps **HTML and JS separate** (clean code)
- Works for modern projects

🧩 Types of Events

Events are categorized into several types:

1️⃣ Mouse Events

Event	Description
`click`	User clicks an element
`dblclick`	Double click
`mouseover`	Mouse enters an element
`mouseout`	Mouse leaves an element
`mousemove`	Mouse moves over element
`mousedown` / `mouseup`	Mouse button pressed/released

2️⃣ Keyboard Events

Event	Description
`keydown`	Key pressed down
`keyup`	Key released
`keypress`	Key pressed (deprecated in modern JS)

3️⃣ Form Events

Event	Description
`submit`	Form submitted
`change`	Input value changed
`input`	Input field changes (real-time)
`focus`	Element gains focus
`blur`	Element loses focus

4️⃣ Window / Document Events

Event	Description
`load`	Page fully loaded
`resize`	Window resized
`scroll`	Page scrolled
`unload`	Page unloaded

5️⃣ Other Events

Event	Description
`contextmenu`	Right-click menu
`error`	Error occurs in page or element
`drag` / `drop`	Drag and drop actions

📝 Event Example

```
<input type="text" id="name" placeholder="Type your name">
<p id="display"></p>

<script>
  let input = document.getElementById("name");
  let display = document.getElementById("display");

  // Event: input
  input.addEventListener("input", function() {
    display.innerText = "Hello, " + input.value;
  });

  // Event: focus
  input.addEventListener("focus", function() {
    input.style.backgroundColor = "lightyellow";
  });

  // Event: blur
  input.addEventListener("blur", function() {
    input.style.backgroundColor = "white";
  });
</script>
```

Output:

- Typing → paragraph updates in real-time
- Focus → input background light yellow
- Blur → input background returns to white

⚡ Summary

Concept	Description
Event	Action happening in browser/user interaction
Triggered by	User actions (click, hover) or browser actions (load, scroll)
Ways to attach events	1. Inline HTML 2. DOM Property 3. addEventListener()
Recommended method	`addEventListener()` (flexible, multiple handlers)
Types	Mouse, Keyboard, Form, Window, Other

📝 Practice Tasks

1. Create a button and show an alert when it is clicked (use all 3 ways).
2. Create an input box and display typed text in real-time using `input` event.
3. Change the background color of a div when the mouse hovers over it and revert when it leaves.

4. Capture `keydown` and `keyup` events on an input field and log the key pressed.
5. Use `submit` event on a form and prevent default submission using `event.preventDefault()`.

🖱️ Mouse Events in JavaScript

Mouse events are triggered by **user interactions with the mouse**, such as clicking, moving, or hovering over elements.

1️⃣ Syntax

Using `addEventListener()`

```
element.addEventListener("eventType", function(event) {
  // Code to execute when event occurs
});
```

Using DOM property

```
element.onclick = function(event) {
  // Code to execute
};
```

💡 **Recommended:** Always use `addEventListener()` for multiple handlers.

2️⃣ Mouse Event Types & Examples

a) `click` – User clicks an element

```
<button id="clickBtn">Click Me</button>

<script>
let btn = document.getElementById("clickBtn");

btn.addEventListener("click", function() {
  alert("Button clicked!");
});
</script>
```

Output:

- Clicking the button shows an alert: "Button clicked!"

---

b) `dblclick` – User double-clicks an element

```
<p id="dblPara">Double-click me!</p>

<script>
let para = document.getElementById("dblPara");

para.addEventListener("dblclick", function() {
  para.style.color = "red";
  para.innerText = "You double-clicked me!";
});
</script>
```

Output:

- Double-clicking the paragraph changes text and color to red.

---

c) `mouseover` – Mouse enters an element

```
<div id="box1" style="width:100px; height:100px;
background:yellow;">Hover me!</div>

<script>
let box = document.getElementById("box1");

box.addEventListener("mouseover", function() {
  box.style.backgroundColor = "orange";
});
</script>
```

Output:

- Moving the mouse over the div changes its background color to orange.

---

d) `mouseout` – Mouse leaves an element

```
<div id="box2" style="width:100px; height:100px;
background:green;">Hover and leave me!</div>

<script>
let box2 = document.getElementById("box2");

box2.addEventListener("mouseout", function() {
  box2.style.backgroundColor = "lightgreen";
});
</script>
```

Output:

- Moving the mouse out of the div changes background to light green.

---

e) `mousemove` – Mouse moves over an element

```
<p id="movePara">Move your mouse here!</p>

<script>
let movePara = document.getElementById("movePara");

movePara.addEventListener("mousemove", function(event) {
  movePara.innerText = `Mouse X: ${event.clientX}, Y:
${event.clientY}`;
});
</script>
```

Output:

- Paragraph dynamically shows the current mouse coordinates as you move over it.

f) `mousedown` / `mouseup` – Mouse button pressed / released

```
<button id="mouseBtn">Press and Release</button>

<script>
let mouseBtn = document.getElementById("mouseBtn");

mouseBtn.addEventListener("mousedown", function() {
  mouseBtn.style.backgroundColor = "pink";
  mouseBtn.innerText = "Mouse Down!";
});

mouseBtn.addEventListener("mouseup", function() {
  mouseBtn.style.backgroundColor = "lightblue";
  mouseBtn.innerText = "Mouse Up!";
});
</script>
```

Output:

- Pressing the mouse button → background pink, text “Mouse Down!”
- Releasing the button → background light blue, text “Mouse Up!”

⚡ Notes

- `click` = single click
- `dblclick` = double click
- `mouseover` and `mouseout` are triggered when entering or leaving the element
- `mousemove` fires continuously as the mouse moves over an element
- `mousedown` / `mouseup` are useful for **drag & drop, button effects, or interactive elements**

📝 Practice Tasks

1. Create a button and use `click` and `dblclick` to change its text.
2. Create a div and change its color when mouse enters (`mouseover`) and leaves (`mouseout`).
3. Track mouse coordinates inside a paragraph using `mousemove`.
4. Use `mousedown` and `mouseup` on a button to simulate a **press effect**.
5. Combine `mouseover` and `click` to create a dynamic hover-click interaction.

⌨️ Keyboard Events in JavaScript

Keyboard events are triggered when the user **presses or releases keys** on the keyboard. These events are mostly used for **input validation, shortcuts, games, or interactive applications**.

1️⃣ Keyboard Event Types

Event	Description
`keydown`	Triggered when a key is **pressed down**
`keyup`	Triggered when a key is **released**
`keypress`	Triggered when a key is pressed (deprecated, not recommended in modern JS)

2️⃣ Syntax

Using `addEventListener()`

```
element.addEventListener("eventType", function(event) {
  // Access key info: event.key, event.code, event.keyCode
});
```

Properties of the event object:

- `event.key` → The key value (e.g., "a", "Enter")
- `event.code` → Physical key location (e.g., "KeyA", "Enter")
- `event.keyCode` → Numeric code of key (deprecated)

3️⃣ Examples

a) keydown – Detect when a key is pressed

```
<input type="text" id="input1" placeholder="Press any key">
<p id="output1"></p>

<script>
let input1 = document.getElementById("input1");
let output1 = document.getElementById("output1");

input1.addEventListener("keydown", function(event) {
  output1.innerText = `Key "${event.key}" pressed down`;
});
</script>
```

Output:

- Press any key → Paragraph shows: `Key "a" pressed down` (example for pressing "a").

---

b) keyup – Detect when a key is released

```
<input type="text" id="input2" placeholder="Type here">
<p id="output2"></p>

<script>
let input2 = document.getElementById("input2");
let output2 = document.getElementById("output2");

input2.addEventListener("keyup", function(event) {
  output2.innerText = `Key "${event.key}" released`;
});
</script>
```

Output:

- Press and release a key → Paragraph shows: `Key "Enter" released`.

---

c) `keypress` – Detect key pressed (deprecated)

```
<input type="text" id="input3" placeholder="Press a key">
<p id="output3"></p>

<script>
let input3 = document.getElementById("input3");
let output3 = document.getElementById("output3");

input3.addEventListener("keypress", function(event) {
  output3.innerText = `Key "${event.key}" pressed (keypress)`;
});
</script>
```

Output:

- Pressing a key shows which key is pressed.
  Note: `keypress` is **deprecated**, use `keydown` or `keyup` instead.

---

d) Detect specific keys (e.g., Enter key)

```
<input type="text" id="input4" placeholder="Press Enter">
<p id="output4"></p>

<script>
let input4 = document.getElementById("input4");
let output4 = document.getElementById("output4");

input4.addEventListener("keydown", function(event) {
  if (event.key === "Enter") {
    output4.innerText = "You pressed the Enter key!";
  }
});
</script>
```

Output:

- Pressing **Enter** in the input field displays the message.

---

e) Real-time input display using keydown

```
<input type="text" id="input5" placeholder="Type something">
<p id="display5"></p>

<script>
let input5 = document.getElementById("input5");
let display5 = document.getElementById("display5");

input5.addEventListener("keydown", function(event) {
  display5.innerText = `You typed: ${input5.value}${event.key}`;
});
</script>
```

Output:

- Paragraph dynamically shows typed text in real-time as keys are pressed.

⚡ Notes

- `keydown` → Fires **once when key is pressed**, continues firing if held down.
- `keyup` → Fires **once when key is released**.
- `keypress` → **Deprecated**, may not work in modern browsers.
- Use **event.key** for the actual character and **event.code** for physical key detection.

📝 Practice Tasks

1. Create an input box and show which key is pressed using `keydown`.
2. Detect when the **Enter** key is pressed in an input field and show an alert.
3. Use `keyup` to display the last key released in a paragraph.
4. Highlight an input field border when typing (`keydown`) and remove highlight on release (`keyup`).
5. Build a small typing app that shows all typed characters dynamically in a div.

📝3️⃣Form Events in JavaScript

Form events are triggered when the **user interacts with forms or input elements**. They are essential for **form validation, dynamic feedback, and improving user experience**.

1️⃣`submit` – Form Submitted

- **Purpose:** Triggered when a form is submitted
- **Syntax:**

```
formElement.addEventListener("submit", function(event) {
  event.preventDefault(); // Prevent form from actual submission
  // Code to execute
});
```

Examples

Example 1 – Alert on submit

```
<form id="form1">
  <input type="text" placeholder="Enter Name" required>
  <button type="submit">Submit</button>
</form>

<script>
```

```
let form1 = document.getElementById("form1");
form1.addEventListener("submit", function(event) {
  event.preventDefault();
  alert("Form Submitted!");
});
</script>
```

Output: Alert pops up when the form is submitted.

Example 2 – Show input value

```
<form id="form2">
  <input type="text" id="name2" placeholder="Enter Name">
  <button type="submit">Submit</button>
</form>
<p id="output2"></p>

<script>
let form2 = document.getElementById("form2");
let output2 = document.getElementById("output2");

form2.addEventListener("submit", function(event) {
  event.preventDefault();
  let name = document.getElementById("name2").value;
  output2.innerText = "Submitted Name: " + name;
});
</script>
```

Output: Displays submitted name in paragraph.

2️⃣ change – Input Value Changed

- **Purpose:** Triggered when the **value of an input, select, or textarea changes**
- **Syntax:**

```
inputElement.addEventListener("change", function() {
  // Code executes after value changes
});
```

Examples

Example 1 – Input field

```
<input type="text" id="changeInput" placeholder="Type something">
<p id="changeOutput"></p>

<script>
let changeInput = document.getElementById("changeInput");
let changeOutput = document.getElementById("changeOutput");

changeInput.addEventListener("change", function() {
  changeOutput.innerText = "Final input value: " +
changeInput.value;
});
</script>
```

Output:

- Paragraph updates **only after leaving the input field** or pressing Enter.

Example 2 – Select dropdown

```
<select id="mySelect">
  <option value="">Select a fruit</option>
  <option value="Apple">Apple</option>
  <option value="Banana">Banana</option>
</select>
<p id="selectOutput"></p>

<script>
let select = document.getElementById("mySelect");
let selectOutput = document.getElementById("selectOutput");

select.addEventListener("change", function() {
  selectOutput.innerText = "You selected: " + select.value;
});
</script>
```

Output:

- Paragraph shows selected fruit **after the selection changes**.

3️⃣ `input` – Input Field Changes (Real-time)

- **Purpose:** Triggered **every time the input value changes**
- **Syntax:**

```
inputElement.addEventListener("input", function() {
  // Code executes on every change
});
```

Examples

Example 1 – Real-time typing

```
<input type="text" id="realInput" placeholder="Type here">
<p id="realOutput"></p>

<script>
let realInput = document.getElementById("realInput");
let realOutput = document.getElementById("realOutput");

realInput.addEventListener("input", function() {
  realOutput.innerText = "You typed: " + realInput.value;
});
</script>
```

Output:

- Paragraph updates **immediately** as the user types.

Example 2 – Live character count

```
<input type="text" id="charInput" placeholder="Type something">
<p id="countOutput">Characters: 0</p>

<script>
```

```
let charInput = document.getElementById("charInput");
let countOutput = document.getElementById("countOutput");

charInput.addEventListener("input", function() {
  countOutput.innerText = "Characters: " + charInput.value.length;
});
</script>
```

Output:

- Shows live character count as user types.

4️⃣ `focus` – Element Gains Focus

- **Purpose:** Triggered when an input or textarea **gains focus**
- **Syntax:**

```
element.addEventListener("focus", function() {
  // Code executes when element is focused
});
```

Example

```
<input type="text" id="focusInput" placeholder="Click me">
<script>
let focusInput = document.getElementById("focusInput");

focusInput.addEventListener("focus", function() {
  focusInput.style.backgroundColor = "lightyellow";
});
</script>
```

Output:

- Input background turns light yellow when clicked/focused.

5️⃣ `blur` – Element Loses Focus

- **Purpose:** Triggered when an input or textarea **loses focus**
- **Syntax:**

```
element.addEventListener("blur", function() {
  // Code executes when element loses focus
});
```

Example

```
<input type="text" id="blurInput" placeholder="Click and leave">
<script>
let blurInput = document.getElementById("blurInput");

blurInput.addEventListener("blur", function() {
  blurInput.style.backgroundColor = "white";
});
</script>
```

Output:

- Input background returns to white when the user clicks outside the input field.

⚡ Notes

- `change` → Triggered **after value change is finalized** (focus leaves the field)
- `input` → Triggered **in real-time as user types**
- `focus` → Triggered **when element is clicked or tabbed into**
- `blur` → Triggered **when element loses focus**
- `submit` → Triggered when form is submitted; use `event.preventDefault()` to prevent page reload

---

📝 Practice Tasks

1. Create a form and show an alert on **submit**.
2. Use `change` to show the selected value of a dropdown.
3. Use `input` to show real-time typing in a paragraph.
4. Highlight an input field when **focused** and remove highlight on **blur**.
5. Count characters in real-time using `input` and alert if more than 20 characters are typed.

🌐4️⃣Window / Document Events in JavaScript

Window events are triggered by the **browser or the document**, such as when the page loads, is resized, scrolled, or unloaded.

1️⃣`load` – Page Fully Loaded

- **Purpose:** Triggered when the **entire page, including images, scripts, and CSS, is fully loaded**
- **Syntax:**

```
window.addEventListener("load", function() {
  // Code executes when page is fully loaded
});
```

Examples

Example 1 – Alert on page load

```
<script>
window.addEventListener("load", function() {
  alert("Page has fully loaded!");
});
```

```
</script>
```

Output:

- Alert appears once the page and all resources are fully loaded.

Example 2 – Display message

```
<p id="msg"></p>

<script>
window.addEventListener("load", function() {
  document.getElementById("msg").innerText = "Welcome! Page Loaded 
Successfully.";
});
</script>
```

Output:

- Paragraph displays a welcome message when page loads.

2️⃣ `resize` – Window Resized

- **Purpose:** Triggered when the **browser window is resized**
- **Syntax:**

```
window.addEventListener("resize", function() {
  // Code executes when window is resized
});
```

Examples

Example 1 – Show window size

```
<p id="size"></p>

<script>
```

```
window.addEventListener("resize", function() {
  document.getElementById("size").innerText =
    `Width: ${window.innerWidth}, Height: ${window.innerHeight}`;
});
</script>
```

Output:

- Paragraph updates dynamically as you resize the browser window.

Example 2 – Change background on small window

```
<script>
window.addEventListener("resize", function() {
  if (window.innerWidth < 500) {
    document.body.style.backgroundColor = "lightpink";
  } else {
    document.body.style.backgroundColor = "white";
  }
});
</script>
```

Output:

- Background becomes light pink when window width < 500px, else white.

3️⃣ `scroll` – Page Scrolled

- **Purpose:** Triggered when the **user scrolls the page**
- **Syntax:**

```
window.addEventListener("scroll", function() {
  // Code executes when page scrolls
});
```

Examples

Example 1 – Show scroll position

```
<p id="scrollPos">Scroll to see position</p>

<script>
window.addEventListener("scroll", function() {
  document.getElementById("scrollPos").innerText =
    `Scroll Y: ${window.scrollY}px`;
});
</script>
```

Output:

- Paragraph dynamically shows vertical scroll position in pixels.

Example 2 – Sticky header effect

```
<header id="header" style="background:lightblue; padding:10px;">My
Header</header>
<div style="height:1500px;"></div>

<script>
window.addEventListener("scroll", function() {
  let header = document.getElementById("header");
  if (window.scrollY > 50) {
    header.style.backgroundColor = "orange";
  } else {
    header.style.backgroundColor = "lightblue";
  }
});
</script>
```

Output:

- Header changes color when scrolled past 50px.

---

4 `unload` – Page Unloaded

- **Purpose:** Triggered when the user **leaves the page or closes the browser tab**
- **Syntax:**

```
window.addEventListener("unload", function() {
  // Code executes before page unloads
});
```

Example

Example 1 – Log message

```
<script>
window.addEventListener("unload", function() {
  console.log("User left the page.");
});
</script>
```

Output:

- Message logged in console when page is closed or navigated away.

Example 2 – Alert on unload (works in some browsers)

```
<script>
window.addEventListener("beforeunload", function(event) {
  event.preventDefault();
  event.returnValue = ''; // Some browsers require this
});
</script>
```

Output:

- Shows a confirmation dialog before leaving the page.

⚡ Notes

- `load` → Fires **once when page and all resources are fully loaded**
- `resize` → Fires **every time window size changes**
- `scroll` → Fires **continuously while scrolling**
- `unload` → Fires **when leaving the page**, use cautiously (cannot reliably show alerts in modern browsers)

📝 Practice Tasks

1. Show an alert **when the page fully loads**.
2. Display **window width and height** dynamically on resize.
3. Show **scroll position** in a paragraph as the page scrolls.
4. Change a **header's background color** after scrolling 100px.
5. Log a message to console when the user **closes or navigates away** from the page.

5️⃣ Other Events in JavaScript

Other events include **context menu, error handling, and drag & drop actions**. These events are used for **advanced user interactions and error handling**.

1️⃣ `contextmenu` – Right-click Menu

- **Purpose:** Triggered when the user **right-clicks** on an element
- **Syntax:**

```
element.addEventListener("contextmenu", function(event) {
  event.preventDefault(); // Prevent default browser context menu
  // Code to execute
});
```

Examples

Example 1 – Custom alert

```
<p id="text1">Right-click on this text</p>

<script>
let text1 = document.getElementById("text1");

text1.addEventListener("contextmenu", function(event) {
  event.preventDefault();
  alert("Custom context menu triggered!");
});
</script>
```

Output:

- Right-clicking the paragraph shows an alert instead of the browser context menu.

Example 2 – Change background

```
<div id="box1" style="width:100px; height:100px;
background:lightgreen;">Right-click me</div>

<script>
let box1 = document.getElementById("box1");

box1.addEventListener("contextmenu", function(event) {
  event.preventDefault();
  box1.style.backgroundColor = "orange";
});
</script>
```

Output:

- Right-click changes the div background color to orange.

2️⃣ `error` – Error Occurs in Page or Element

- **Purpose:** Triggered when **an error occurs** while loading a resource (like an image) or in scripts
- **Syntax:**

```
window.addEventListener("error", function(event) {
  console.log("Error occurred: ", event.message);
});
```

Examples

Example 1 – Log script error

```
<script>
window.addEventListener("error", function(event) {
  console.log("Error detected: " + event.message);
});

// Intentionally cause an error
nonExistentFunction();
</script>
```

Output:

- Console logs: `Error detected: nonExistentFunction is not defined`

Example 2 – Handle image load error

```
<img id="img1" src="wrongpath.jpg" alt="Image" width="100">

<script>
let img1 = document.getElementById("img1");

img1.addEventListener("error", function() {
  img1.alt = "Image failed to load!";
  img1.style.border = "2px solid red";
});
</script>
```

Output:

- Image fails to load → alt text changes and red border added.

3️⃣ `drag` / `drop` – Drag and Drop Actions

- **Purpose:** Triggered when an element is **dragged and dropped**
- **Syntax:**

```
draggableElement.addEventListener("dragstart", function(event) {
  // Code executes when dragging starts
});

dropZone.addEventListener("dragover", function(event) {
  event.preventDefault(); // Allow dropping
});

dropZone.addEventListener("drop", function(event) {
  event.preventDefault();
  // Code executes on drop
});
```

Example 1 – Simple drag and drop

```
<div id="drag1" draggable="true" style="width:100px; height:100px;
background:blue;">Drag me</div>
<div id="drop1" style="width:150px; height:150px;
background:lightgray; margin-top:10px;">Drop here</div>

<script>
let drag1 = document.getElementById("drag1");
let drop1 = document.getElementById("drop1");

drag1.addEventListener("dragstart", function(event) {
  event.dataTransfer.setData("text", event.target.id);
});

drop1.addEventListener("dragover", function(event) {
  event.preventDefault();
});
```

```
drop1.addEventListener("drop", function(event) {
  event.preventDefault();
  let data = event.dataTransfer.getData("text");
  let draggedElement = document.getElementById(data);
  drop1.appendChild(draggedElement);
});
</script>
```

Output:

- Drag the blue box → drop it inside the gray box.

Example 2 – Change color on drag over

```
<div id="drop2" style="width:150px; height:150px;
background:lightgray; margin-top:10px;">Drop here</div>

<script>
let drop2 = document.getElementById("drop2");

drop2.addEventListener("dragover", function(event) {
  event.preventDefault();
  drop2.style.backgroundColor = "lightgreen";
});

drop2.addEventListener("dragleave", function() {
  drop2.style.backgroundColor = "lightgray";
});
</script>
```

Output:

- Dragging over the drop zone → background turns light green
- Leaving without dropping → background returns to light gray

---

⚡ Notes

- `contextmenu` → Useful for creating **custom right-click menus**
- `error` → Helps **handle broken images or script errors**
- `drag` / `drop` → Used for **drag & drop interfaces** like file uploads, games, or interactive apps

---

📝 Practice Tasks

1. Prevent the default right-click menu on a paragraph and show an alert instead.
2. Load a non-existent image and handle the error by showing a message or border.
3. Create a simple drag-and-drop box that changes color when dragging over it.
4. Make a small game: drag an element into a drop zone and show a success message.