To implement a drag-and-drop functionality for images on a website that works on both desktop and mobile devices, you can use a combination of HTML, CSS, and JavaScript. Here’s a step-by-step guide to achieve this:

### HTML

1. \*\*Set up the HTML structure\*\*:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Drag and Drop Image</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="drop-zone" id="drop-zone">

<img src="path-to-your-image.jpg" alt="Draggable" id="draggable-image" class="draggable">

</div>

<script src="script.js"></script>

</body>

</html>

```

### CSS

2. \*\*Style the elements\*\*:

```css

/\* styles.css \*/

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.drop-zone {

width: 300px;

height: 300px;

border: 2px dashed #cccccc;

display: flex;

justify-content: center;

align-items: center;

position: relative;

}

.draggable {

width: 100px;

height: 100px;

cursor: move;

position: absolute;

}

```

### JavaScript

3. \*\*Implement the drag-and-drop logic for both desktop and mobile\*\*:

```javascript

// script.js

const dropZone = document.getElementById('drop-zone');

const draggableImage = document.getElementById('draggable-image');

let offsetX = 0;

let offsetY = 0;

let isDragging = false;

draggableImage.addEventListener('mousedown', (e) => {

isDragging = true;

offsetX = e.clientX - draggableImage.getBoundingClientRect().left;

offsetY = e.clientY - draggableImage.getBoundingClientRect().top;

});

document.addEventListener('mousemove', (e) => {

if (isDragging) {

draggableImage.style.left = `${e.clientX - offsetX}px`;

draggableImage.style.top = `${e.clientY - offsetY}px`;

}

});

document.addEventListener('mouseup', () => {

isDragging = false;

});

draggableImage.addEventListener('touchstart', (e) => {

isDragging = true;

const touch = e.touches[0];

offsetX = touch.clientX - draggableImage.getBoundingClientRect().left;

offsetY = touch.clientY - draggableImage.getBoundingClientRect().top;

}, { passive: true });

document.addEventListener('touchmove', (e) => {

if (isDragging) {

const touch = e.touches[0];

draggableImage.style.left = `${touch.clientX - offsetX}px`;

draggableImage.style.top = `${touch.clientY - offsetY}px`;

}

}, { passive: true });

document.addEventListener('touchend', () => {

isDragging = false;

});

```

### Explanation:

- \*\*HTML\*\*: Contains a container `div` for the drop zone and an `img` element that will be draggable.

- \*\*CSS\*\*: Provides basic styling for the drop zone and the draggable image.

- \*\*JavaScript\*\*: Handles the drag-and-drop functionality.

- For \*\*desktop\*\*: `mousedown`, `mousemove`, and `mouseup` events are used.

- For \*\*mobile\*\*: `touchstart`, `touchmove`, and `touchend` events are used.

- Event listeners update the position of the image based on cursor or touch movement.

### Notes:

- Ensure the image path (`src="path-to-your-image.jpg"`) is correctly set to point to your image.

- The `position: absolute;` style on the image allows it to be positioned anywhere inside the parent `drop-zone` container.

- The `passive: true` option in touch event listeners is used to improve scrolling performance.

By following this setup, you can create a drag-and-drop image functionality that works seamlessly on both desktop and mobile devices.