COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

"FUNCTIONS IN JAVASCRIPT"

A **function** in JavaScript is a block of code designed to perform a specific task. It can be defined once and then reused whenever needed. Functions allow you to group together a series of statements that perform a task or calculate a value. Here's why functions are important and commonly used:

Role of Functions in JavaScript:

1. **Code Reusability**: You can define a function once and call it multiple times, reducing code duplication and making your code easier to maintain.
2. **Modularity**: Functions help break a large program into smaller, manageable pieces. Each function can handle a specific task, making the code more organized.
3. **Abstraction**: Functions allow you to hide the complexity of a task. You can use a function without knowing how it works internally, focusing instead on what it does.
4. **Maintainability**: Since functions organize code into logical blocks, maintaining and debugging the program becomes easier.

SYNTAX OF A FUNCTION

```
function functionName(parameters) {
  // Block of code
  return result;  // Optional
}
```

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

```
// Function to add two numbers
function addNumbers(a, b) {
  return a + b;
}

// Calling the function
let result = addNumbers(5, 10);  // result will be 15
console.log(result);
```

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

Imagine you're developing a web application for an event registration platform. You want to gather feedback from users after they register for an event.

User Flow:

User Registration: After successfully registering for an event, users are presented with a confirmation page.

Prompt for Feedback: On the confirmation page, there's a button labeled "Open Modal" that invites users to provide feedback about their registration experience.

Opening the Modal: When the user clicks the "Open Modal" button, a modal dialog appears on the screen. The modal has a title ("Modal Title") and a short message ("This is a modal window.") that can provide additional context or instructions for feedback.

User Interaction: Inside the modal, there might be a form (not shown in the provided code) for users to fill out their feedback. The modal also includes a close button (the "×" icon) in the upper right corner.

Closing the Modal: Users can close the modal by clicking the close button or by clicking outside the modal area. This interaction is seamless and ensures users can easily return to the main page.

Feedback Submission: If the feedback form were implemented, after filling it out, users could submit their responses, helping the platform improve its services.

Key Features:

- **Responsive Design:** The modal appears centered on the screen, with a semi-transparent background that dims the rest of the content, drawing the user's focus to the feedback form.

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

- **User-Friendly:** The design allows users to easily open and close the modal, making it a simple and effective way to gather feedback without navigating away from the confirmation page.

EXPLANTION OF HTML CODE:

- **Modal Structure:**
  - `<div id="modal" class="modal">`
    - This `div` represents the modal itself. It's initially hidden and will be shown when the user interacts with the button.
  - `<div class="modal-content">`
    - This inner `div` contains the content of the modal, including the title and message.
  - `<span id="closeModalBtn" class="close">&times;</span>`
    - A span element that acts as a close button for the modal. The `&times;` entity represents the multiplication sign (×), visually indicating to users they can close the modal.
  - `<h2>Modal Title</h2>`
    - A heading that serves as the title for the modal, making it clear to users what the modal is about.

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

- `<p>This is a modal window.</p>`
    - A paragraph providing additional information or context about the modal.

- `<script src="e3.js"></script>`
    - This line links an external JavaScript file (`e3.js`) that contains the functionality for opening and closing the modal.

EXPLANATION OF CSS STYLING

Breakdown of Each Style Rule

```
body { font-family: Arial, sans-serif; }
```

1. **Purpose:** Sets the font for the entire body of the webpage.
2. `font-family`: Specifies the typeface. `Arial` is a sans-serif font, and if it's unavailable, the browser will use the default sans-serif font.

```
.modal { ... }
```

1. `position: fixed;`
    1. Positions the modal relative to the viewport, meaning it will stay in the same place even if the page is scrolled.
2. `z-index: 1;`
    1. Sets the stacking order of the modal. A higher z-index ensures it appears above other content.
3. `left: 0;` **and** `top: 0;`
    1. Positions the modal at the top-left corner of the viewport.

4. `width: 100%;` **and** `height: 100%;`
    1. Makes the modal cover the entire screen, allowing it to act as a backdrop for the content.
5. `overflow: auto;`
    1. Adds scrollbars if the content exceeds the viewport size, ensuring accessibility to all content.
6. `background-color: rgba(0, 0, 0, 0.5);`
    1. Sets a semi-transparent black background for the modal, dimming the rest of the page and focusing the user’ s attention on the modal content.

```
.modal-content { ... }
```

7. `background-color: #fefefe;`
    1. Sets a light background color for the modal's content area, providing contrast against the dark backdrop.
8. `margin: 15% auto;`
    1. Centers the modal content with an automatic left and right margin, and adds a 15% margin from the top of the viewport.
9. `padding: 20px;`
    1. Adds space inside the modal content area, preventing text from touching the edges.
10. `border: 1px solid #888;`

1. Adds a subtle gray border around the modal content, helping to define its edges.

```
.width: 80%;
```

1. Sets the modal content width to 80% of the viewport, allowing it to be responsive.

```
.close { ... }
```

11. `color: #aaa;`

    1. Sets the initial color of the close button to light gray.

12. `float: right;`

    1. Positions the close button to the right of the modal content.

13. `font-size: 28px;`

    1. Increases the size of the close button, making it more prominent and easier to click.

14. `font-weight: bold;`

    1. Makes the close button text bold for better visibility.

```
.close:hover, .close:focus { ... }
```

15. `color: black;`

    1. Changes the close button color to black when hovered over or focused (using keyboard navigation).

16. `text-decoration: none;`

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

1. Removes any underline that might appear (typically for links).

17. `cursor: pointer;`

1. Changes the mouse cursor to a pointer when hovering over the close button, indicating that it' s clickable.

EXPLANATION OF JS-CODE

```
var modal = document.getElementById("modal");
var openModalBtn = document.getElementById("openModalBtn");
var closeModalBtn = document.getElementById("closeModalBtn");
```

- `document.getElementById("modal")`: Selects the modal dialog element by its ID. This allows you to manipulate the modal' s visibility.
- `document.getElementById("openModalBtn")`: Selects the button that opens the modal.
- `document.getElementById("closeModalBtn")`: Selects the close button inside the modal.

FUNCTION TO OPEN THE MODEL

```
function openModal() {
    modal.style.display = "block";
}
```

openModal Function: This function changes the display style of the modal from none to block, making it visible on the screen.

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

```
function closeModal() {
    modal.style.display = "none";
}
```

- `closeModal` **Function**: This function sets the modal's display style back to `none`, hiding it from view.

Event Listeners for Buttons:

```
openModalBtn.onclick = openModal;
closeModalBtn.onclick = closeModal;
```

- `openModalBtn.onclick = openModal;`: Attaches the `openModal` function to the click event of the "Open Modal" button. When the button is clicked, the modal will open.
- `closeModalBtn.onclick = closeModal;`: Attaches the `closeModal` function to the click event of the close button. When this button is clicked, the modal will close.

Close the Modal When Clicking Outside:

COURSE_NAME: WEB DEVELOPMENT
COURSE_INSTRUCTOR: ABDULLAH MAHMOOD

```
window.onclick = function(event) {
    if (event.target == modal) {
        closeModal();
    }
}
```

- `window.onclick`: Sets up an event listener for clicks anywhere on the window.
- `function(event)`: Defines a function that takes the click event as an argument.
- `if (event.target == modal)`: Checks if the click target is the modal itself (meaning the user clicked outside the modal content).
- `closeModal();`: If the user clicked outside the modal content, the modal will close by calling the `closeModal` function.