Modals

Overview

A modal is a user interface element in HTML that is used to display qualifying or prompting information. It often is used outside of the normal flow of a page or application, thus providing a critical tool to manage information. Modals are used to display content in a layer on top of the visible window.

In this module you discover how to use, program, and style an HTML modal.

Prepare

A modal dialog is built using the **`dialog`** element. The **`dialog`** element is an element that provides a way to create modal dialogs. It is a block-level element that can be styled with CSS and can be controlled with JavaScript.

To create a modal dialog, you need to use the **`dialog`** element and set its **`open`** attribute to true. This will make the dialog visible on the page. You can also use the **`show()`** and **`close()`** methods to control the visibility of the dialog.

The following is an example of how to create a modal dialog using the **`dialog`** element:

```html
<dialog id="myModal">
  <h2>Modal Dialog</h2>
  <p>This is a modal dialog.</p>
  <button id="closeModal">Close</button>
</dialog>
```

In this example, the **`dialog`** element is used to create a modal dialog with a title and a close button. The **`id`** attribute is used to identify the dialog so that it can be controlled with JavaScript.

> This dialog element will not be visible until the **`show()`** method is called in JavaScript so do not worry if you are building a test page and start with this HTML and nothing shows up.

To style the modal dialog, you can use CSS to set the width, height, background color, and other properties of the `dialog` element. You can also use the `::backdrop` pseudo-element to style the background of the modal dialog.

The following is an example of how to style a modal dialog using CSS:

```
dialog {
  width: 400px;
  height: 300px;
  background-color: white;
  border: 1px solid black;
  border-radius: 5px;
  box-shadow: 0 0 10px rgb(0 0 0 / 50%);
  padding: 20px;
}
::backdrop {
  background-color: rgb(0 0 0 / 50%);
}
```

In this example, the `dialog` element is styled with a width and height, a white background color, a black border, and a box shadow. The `::backdrop` pseudo-element is used to style the background of the modal dialog with a semi-transparent black color.

To control the modal dialog with JavaScript, you can use the `show()` and `close()` methods to display and hide the modal dialog. You can also use the `addEventListener()` method to add event listeners to the modal dialog and its elements.

The following is an example of how to control a modal dialog using JavaScript:

```
const modal = document.querySelector('#myModal');
const closeModal = document.querySelector('#closeModal');
modal.showModal(); // display the modal dialog right away.
// Usually you will want to wait for a user action to show the modal dialog
closeModal.addEventListener('click', () => {
  modal.close();
});
```

In this example, the `showModal()` method is used to display the modal dialog, and the `close()` method is used to hide the modal dialog when the close button is clicked. The `addEventListener()` method is used to add a click event listener to the close button.

The following CodePen project demonstrates two ways to employ modal dialogs

1. The first example uses the `<dialog>` element with accompanying JavaScript support as demonstrated in the code examples above.
2. The second example uses the `<dialog popover` feature as an alternative way to show overlays or 'popups' on a page. No JavaScript is needed on the second example by using HTML attributes of `popovertarget` on the "Sign Up" button to show the dialog and `popovertargetaction="hide"` on the ❌ button to close the dialog.

Code Example: Modals using dialog and ::backdrop

Example using an HTML dialog, :modal pseudo-class, and ::backdrop pseudo-element

Activity Instructions

Using your course home page, enable modal behavior by responding to users clicking on a course card/button and displaying details about the course.

The following animation is an example of the action and display effect.

0:10 / 0:10

Step 1: HTML

1. Open your course home page.
2. Add an empty `dialog` element. In added it just before the closing main tag.

▼ **Check Your Understanding – Example**

```
<dialog id="course-details"></dialog>
```

Step 2: CSS

Use existing style patterns that you have developed and note that any additional CSS should support your design scheme overall.

1. Use `dialog` to style the modal container.

   The CSS selector `:modal` will not work because `:modal` is not a supported CSS pseudo-class.
   It is recommended that you style the `dialog` or you can assign an `id` or `class` to the `dialog` and use that selector.

2. Use `::backdrop` on the modal to affect the background.
3. Style the `dialog` element's default close `button`.

▼ Check Your Understanding – Example

```
dialog {
  border: 1px solid rgb(0 0 0 / 10%);
  border-radius: .25rem;
  padding: 1rem;
  background-color: #fff;
  box-shadow: 0 0 3rem #777;
  width: 90%;
  max-width: 600px;
}

::backdrop {
  background-color: rgb(0 0 0 / 50%);
}

dialog button {
  position: absolute;
  top: 23px;
  right: 23px;
  padding: .5rem 1rem;
  border: 1px solid rgb(0 0 0 / 10%);
}
```

JavaScript

1. Write a function to display the modal.

2. Add the following content to the modal display:

   button that will close the modal.

   event listener to close the modal when the user clicks outside of the modal.

   subject and number

   title

   credits

   description

   certificate

   technology stack

▼ Check Your Understanding – Example

This example assumes that you have declared and initialized the `courseDetails` variable to reference the HTML `detail` element.

```
function displayCourseDetails(course) {
  courseDetails.innerHTML = '';
  courseDetails.innerHTML = `
    <button id="closeModal">❌</button>
    <h2>${course.subject} ${course.number}</h2>
    <h3>${course.title}</h3>
    <p><strong>Credits</strong>: ${course.credits}</p>
    <p><strong>Certificate</strong>: ${course.certificate}</p>
    <p>${course.description}</p>
    <p><strong>Technologies</strong>: ${course.technology.join(',
  `;
  courseDetails.showModal();

  closeModal.addEventListener("click", () => {
    courseDetails.close();
  });
}
```

3. Add a 'click' event listener to the course container build (inside the loop building each course container). This trigger will call the new display function, passing the current course's information.

▼ Check Your Understanding

```
courseDiv.addEventListener('click', () => {
  displayCourseDetails(course);
});
```

Coding Demonstration Videos

For those who prefer to watch and learn, you have a video lab on modal dialogs. The first video covers building a single modal. The second video builds on the first and shows you how to reuse a dialog for multiple popups. The third video shows how to build modals from a JSON file and also included JavaScript Modules with import / export. This series should help you complete the Modal Dialog assignment for this week.

These labs are designed so that you can follow along and learn. You will need to pause the video instruction as you code your own project.

Download a start lab folder for the first video

Watch Modal Dialog Part 1

Single Dialog Example

Watch Modal Dialog Part 2

Reusable Dialog

Download a start lab folder with the temple data

Watch Modal Dialog Part 3

Scripted Dialog with JSON Temple Data A

Watch Modal Dialog Part 4

Scripted Dialog with JSON Temple Data B

Optional Resources

HTML `dialog` element – MDN

`:modal` CSS Pseudo-class

Back