JavaScript Dialog API

Summary: in this tutorial, you will learn how to create dialogs using the native `<dialog>` element and JavaScript Dialog API.

Introduction to JavaScript Dialog API

In HTML5, you can use `<dialog>` element to create a modal or modeless dialog. The native `<dialog>` element includes built-in support for a backdrop and methods for controlling its visibility.

To create a new dialog element, you use the `<dialog>` tag as follows:

```
<dialog>
</dialog>
```

Between the opening `<dialog>` and closing `</dialog>` tags, you can place the contents. Typically, it includes a button for closing the dialog:

```
<dialog>
<button type="button">Close</button>
</dialog>
```

The `<dialog>` element has a type of `HTMLDialogElement` that inherits from the `HTMLElement` interface.

The `HTMLDialogElement` provides the following methods for controlling dialog visibility:

Method	Description
`show()`	Display a modal dialog, allowing you to interact with the elements outside of the dialog.
`showModal()`	Display a modal dialog, blocking you from interacting with the elements outside of the dialog.
`close()`	Close the dialog.

JavaScript dialog API examples

Let’s take some examples of using the JavaScript Dialog API.

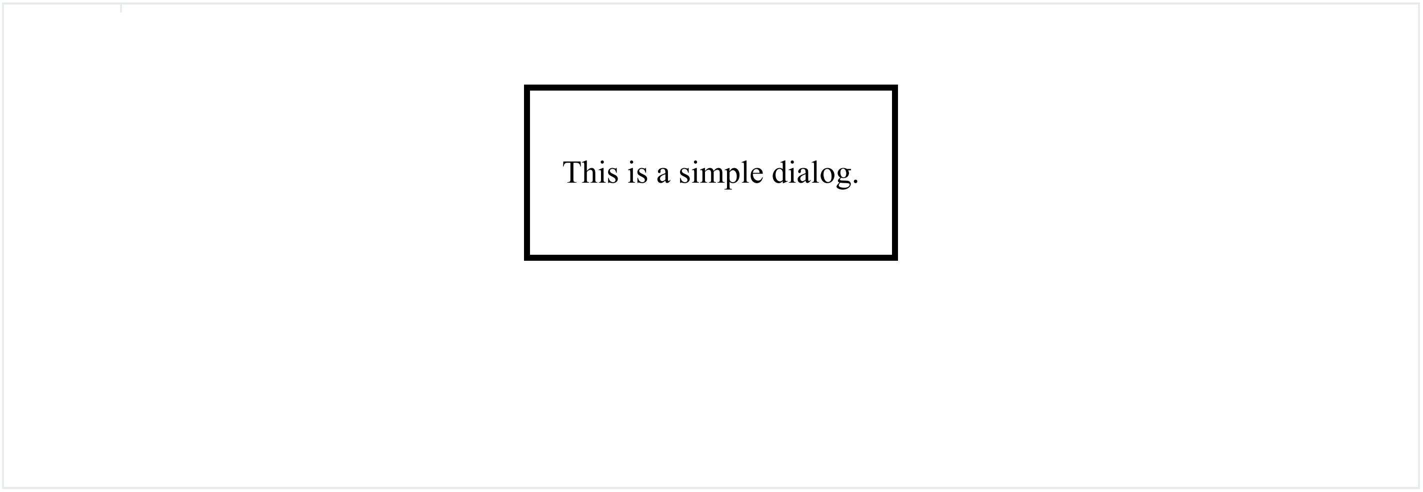
1) Creating a simple dialog example

The following page uses the `<dialog>` element to create a simple dialog with a message:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript Dialog API</title>
    </head>
    <body>
        <dialog id="myDialog" open>
            <p>This is a simple dialog.</p>
        </dialog>
    </body>
</html>
```

In this example, the `<dialog>` element includes the `open` attribute that will display automatically when the page loads.

Output

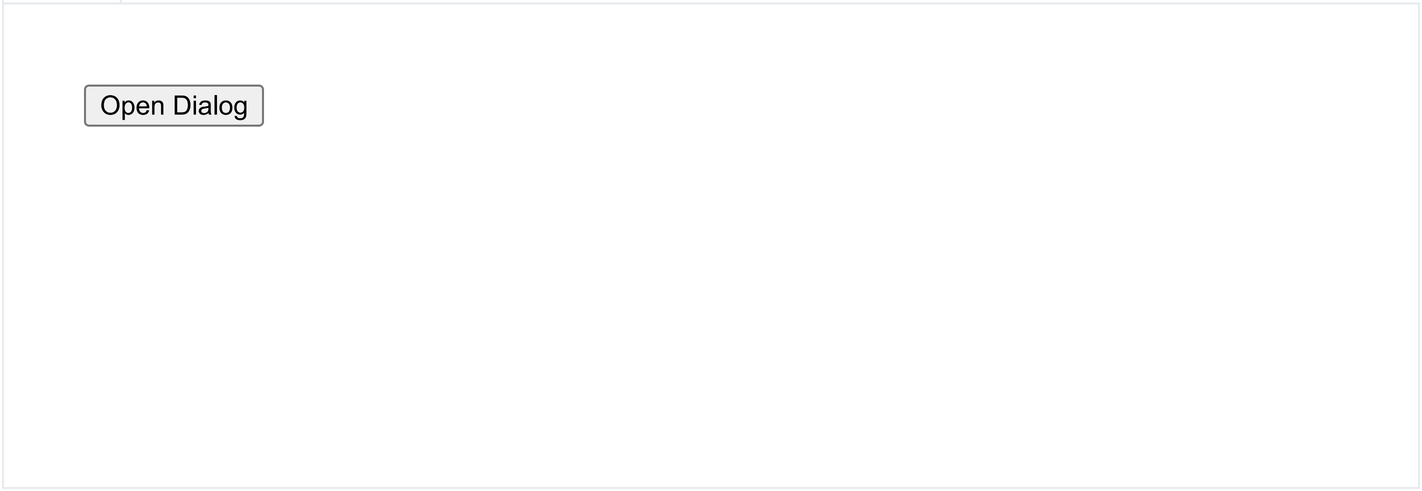

This is a simple dialog.

2) Using JavaScript dialog API to control the dialog visibility

The following example shows how to use JavaScript Dialog API to control the dialog visibility:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>JavaScript Dialog API</title>
    </head>
    <body>
        <dialog id="myDialog">
            <p>This is a dialog.</p>
            <button id="closeDialog">Close</button>
        </dialog>

        <button type="button" id="openDialog">Open Dialog</button>

        <script>
            const btnOpen = document.querySelector('#openDialog');
            const btnClose = document.querySelector('#closeDialog');
            const dialog = document.querySelector('#myDialog');

            btnOpen.addEventListener('click', () => {
                dialog.showModal();
            });
            btnClose.addEventListener('click', () => {
                dialog.close();
            });
        </script>

    </body>

</html>
```

Output



How it works.

First, declare the `<dialog>` element with the id `myDialog` :

```
<dialog id="myDialog">
   <p>This is a dialog.</p>
   <button id="closeDialog">Close</button>
</dialog>
```

Inside the `<dialog>` element, we have a button with the id `closeDialog` . When it is clicked, the dialog is closed.

Second, declare the `<button>` element with the id `openDialog` . When it is clicked, the dialog is open:

```
<button type="button" id="openDialog">Open Dialog</button>
```

Third, select the open button, dialog element, and close button using the `querySelector()` method:

```
const btnOpen = document.querySelector('#openDialog');
const dialog = document.querySelector('#myDialog');
const btnClose = document.querySelector('#closeDialog');
```

Fourth, show the dialog when the open button is clicked by calling the `showModal()` method of the dialog element inside the click handler of the open button:

```
btnOpen.addEventListener('click', () => {
    dialog.showModal();
});
```

Fifth, close the dialog when the close button is clicked by calling the `close()` method of the dialog element inside the click handler of the close button:

```
btnClose.addEventListener('click', () => {
    dialog.close();
});
```

3) Using a form inside a dialog

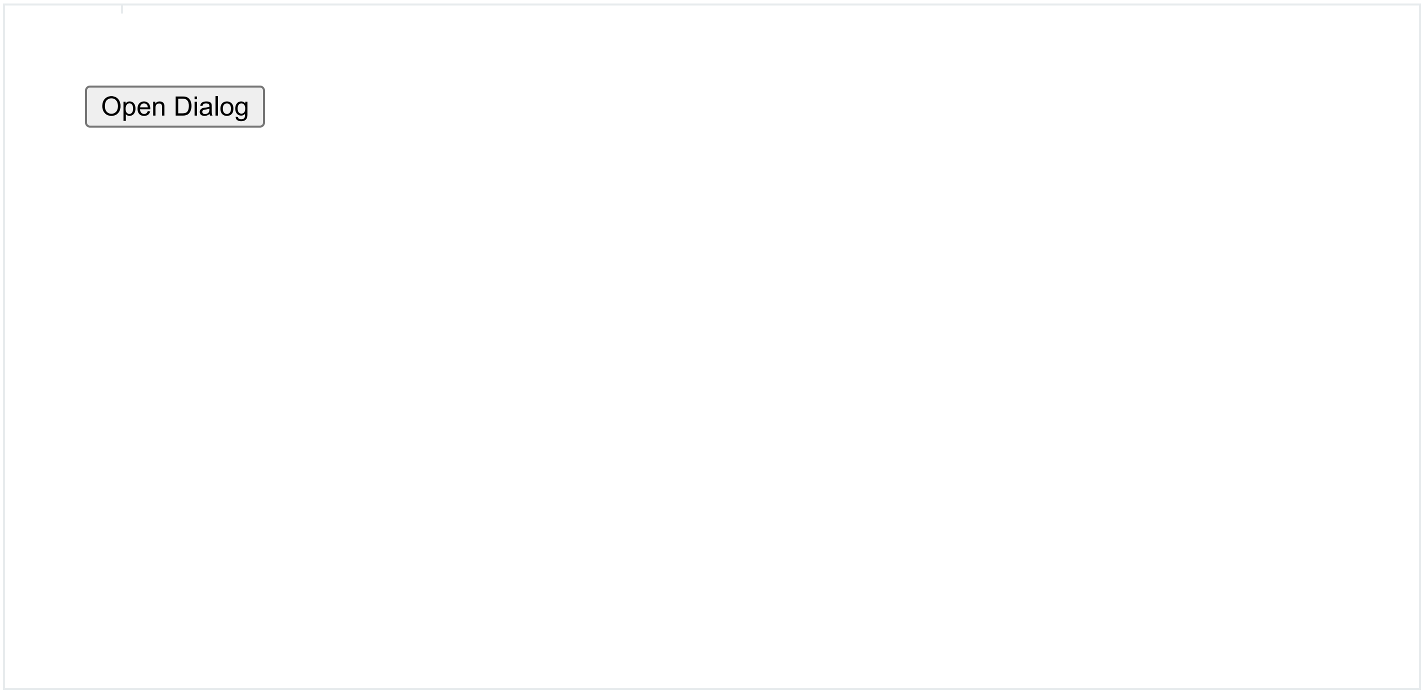
If you use a form in your dialog, you can set the `method` attribute of the form to the `dialog` . This will cause the dialog to close the form when you submit it.

However, the form data will not be submitted to the server. Instead, it is saved in the browser. If you reopen the same dialog, the form fields will be filled with the saved data.

Here's an example:

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="robots" content="noindex">
        <title>JavaScript Dialog Form</title>
    </head>

    <body>
        <dialog id="subscribeDialog">
            <form method="dialog">
                <p>
                    <label for="name">Name:</label>
                    <input type="text" name="name" id="name">
                </p>
                <p>
                    <label for="email">Email:</label>
                    <input type="email" name="email" id="email">
                </p>

                <button type="submit" id="btnOk">Subscribe</button>
            </form>
        </dialog>
        <button type="button" id="btnOpenDialog">Open Dialog</button>

        <script>
            const dialog = document.querySelector('#subscribeDialog');
            const btnOpenDialog = document.querySelector('#btnOpenDialog');
            btnOpenDialog.addEventListener('click', (event) => {
                dialog.showModal();
            });
        </script>
    </body>

</html>
```

Output



When you click the Open Dialog button, the dialog will be displayed. After entering your name and email and clicking the Subscribe button, the dialog will disappear.

If you click the Open Dialog button to open the same dialog again, you'll see that the name and email fields are still filled with the previously entered data.

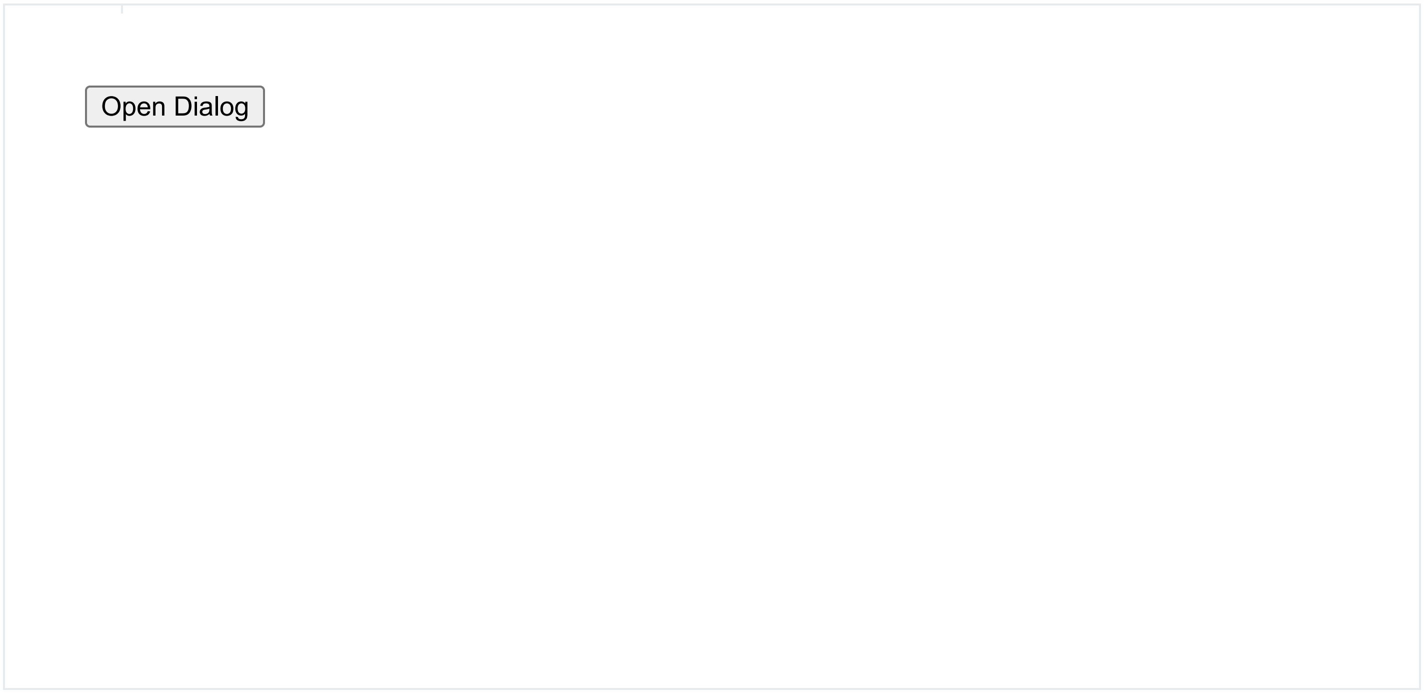
Additionally, when the dialog is displayed, the first input element (`name`) is automatically focused. This is a great accessibility feature that the dialog provides by default.

If you want to close the dialog without submitting it, you can add the `formmethod="dialog"` attribute to a submit button.

For example, you can have a cancel button in your form on a dialog, which closes the form without submitting the form data:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="robots" content="noindex">
        <title>JavaScript Dialog Form</title>
    </head>
    <body>
        <dialog id="subscribeDialog">
            <form method="dialog">
                <p>
                    <label for="name">Name:</label>
                    <input type="text" name="name" id="name">
                </p>
                <p>
                    <label for="email">Email:</label>
                    <input type="email" name="email" id="email">
                </p>

                <button type="submit" id="btnOk">Subscribe</button>
                <button type="submit" formmethod="dialog">Cancel</button>
            </form>
        </dialog>
        <button type="button" id="btnOpenDialog">Open Dialog</button>

        <script>
            const dialog = document.querySelector('#subscribeDialog');
            const btnOpenDialog = document.querySelector('#btnOpenDialog');
            btnOpenDialog.addEventListener('click', (event) => {
                dialog.showModal();
            });
        </script>
    </body>

</html>
```

Output



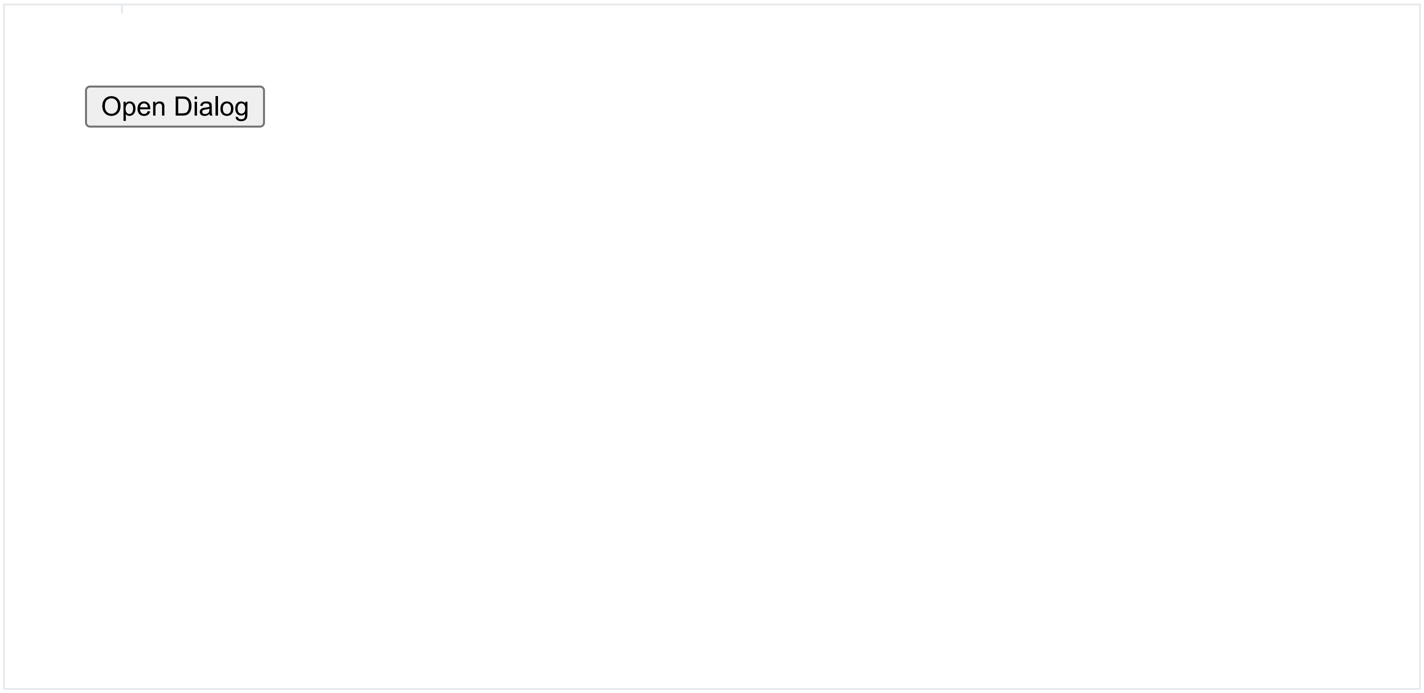
Closing the dialog when clicking outside

When you click outside a dialog, it is not closed. To close it, you can add listen to the click event of the dialog element and close it if it is clicked outside of the dialog.

```
dialog.addEventListener("click", (event) => {
    const rect = dialog.getBoundingClientRect()
    if (event.clientX < rect.left || event.clientX > rect.right || event.clientY < rect.top
        dialog.close()
    }
})
```

For example:

Output



Summary

- Use the native HTML5 `<dialog>` element to create a modal dialog.
- A dialog can be modal or modalless. Modal dialogs prevent interaction with the rest of the page while modalless dialogs do not.
- Use the `show()` method to display a modeless dialog.
- Use the `showModal()` method to display a modal dialog.
- Use the `close()` method to close the dialog.
- The dialog element fires a `close` event when it is closed or a `cancel` event when the dialog is canceled for example by pressing the Escape key.