JavaScript: gestire l'elemento dialog

In questo articolo spiegheremo le principali differenze tra i due modi di gestire un elemento dialog con JavaScript.

Un elemento `dialog` può essere gestito come una finestra di dialogo modale a pagina intera o come una semplice finestra di dialogo. Con la modalità modale, un utente non può interagire con il resto della pagina, mentre con la modalità di dialogo semplice gli utenti possono ancora accedere ad altri elementi del documento HTML.

Nella prima modalità, dietro la finestra di dialogo principale apparirà uno sfondo sovrapposto, impedendo così qualsiasi ulteriore interazione con gli elementi della pagina sottostante. Invece, se usato come una semplice finestra di dialogo, un elemento `dialog` non mostrerà lo sfondo.

Ci sono due metodi disponibili su un elemento `dialog` che controllano il modo in cui verrà presentato all'utente finale:

1. `showModal()`: mostrerà la finestra di dialogo con lo sfondo
2. `show()`: mostrerà la finestra di dialogo senza lo sfondo.

Per chiudere qualsiasi elemento `dialog` mostrato con uno dei due metodi descritti in precedenza, dobbiamo usare il metodo `close()` dell'elemento DOM correlato.

Ad esempio:

```
'use strict';

const openModal = document.querySelector('.open-
modal');
```

```
const openDialog = document.querySelector('.open-
dialog');
const closeModal = document.querySelector('.modal-
close');
const modal = document.querySelector('.modal');

const handleOpenModal = () => {
    return modal.showModal(); // Con il backdrop
};

const handleOpenDialog = () => {
    return modal.show(); // Senza il backdrop
};

const handleCloseModal = () => {
    return modal.close(); // Chiusura
};

document.addEventListener('DOMContentLoaded', () => {
openModal.addEventListener('click', handleOpenModal,
false);
        openDialog.addEventListener('click',
handleOpenDialog, false);
        closeModal.addEventListener('click',
handleCloseModal, false);
    }, false);
```

L'elemento backdrop è in realtà gestito con lo pseudo-elemento CSS `::backdrop`, quindi per cambiarne gli stili dobbiamo selezionarlo come segue nel nostro codice CSS:

```
.modal::backdrop {
    background-color: rgba(0, 0, 0, 0.8);
```

```
}
```

L'elemento `dialog` stesso ha un bordo ed è posizionato su un lato della viewport per impostazione predefinita, quindi dobbiamo cambiare anche i suoi stili:

```
.modal {
    background-color: #fff;
    border: none;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
}
```

Con questi stili applicati, ora la finestra di dialogo è centrata orizzontalmente e verticalmente all'interno della finestra e non mostra più un bordo attorno ad essa. Presenta anche uno sfondo bianco.

La differenza principale con una finestra di dialogo personalizzata quando si tratta di CSS è che non si può posizionare il controllo di chiusura al di fuori dell'elemento `dialog` (ad esempio sul livello dello sfondo) se tale controllo è un elemento figlio dell'elemento stesso. Ciò accade a causa del contesto creato dal posizionamento CSS: una volta scelto uno schema di posizionamento su un elemento genitore (ad esempio `dialog`), questo elemento fungerà da contesto di riferimento per tutti i suoi discendenti posizionati.

Per risolvere questo problema, si dovrebbe inserire l'elemento `dialog` in un contenitore a piene dimensioni per posizionare il pulsante di chiusura al di fuori dell'elemento `dialog`. Così facendo, però, si dovrebbe anche gestire la visibilità del contenitore con JavaScript.

Dal momento che l'elemento `dialog` è stato progettato per risolvere i problemi di accessibilità creati dalle finestre di dialogo personalizzate, l'aggiunta di un elemento wrapper solo per una scelta di design grafico causerebbe diversi problemi alle persone che utilizzano tecnologie assistive.

Demo

JavaScript: dialog element