**The Modal Window**

How to make a text box appear when we click on a button.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generatedOn the webpage there are three buttons:

Here is the html for the buttons:

<body>

<button class="show-modal">Show modal 1</button>

<button class="show-modal">Show modal 2</button>

<button class="show-modal">Show modal 3</button>

Notice that the class corresponds to the following CSS:

.show-modal {

*font-size*: 2rem;

*font-weight*: 600;

*padding*: 1.75rem 3.5rem;

*margin*: 5rem 2rem;

*border*: none;

*background-color*: #fff;

*color*: #444;

*border-radius*: 10rem;

*cursor*: pointer;

}

What this means, is that we can see the modal button with this formatting on the web page.

When we click on the button we want it to display some information.

Specifically this information:

<div class="modal hidden">

<button class="close-modal">&times;</button>

<h1>I'm a modal window 😍</h1>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim

veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

commodo consequat. Duis aute irure dolor in reprehenderit in voluptate

velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint

occaecat cupidatat non proident, sunt in culpa qui officia deserunt

mollit anim id est laborum.

</p>

It is important to note that there are two classes for this button. ‘modal’ and ‘hidden’.

Here is the CSS:

.hidden {

*display*: none;

}

.modal {

*position*: absolute;

*top*: 50%;

*left*: 50%;

*transform*: translate(-50%, -50%);

*width*: 70%;

*background-color*: white;

*padding*: 6rem;

*border-radius*: 5px;

*box-shadow*: 0 3rem 5rem rgba(0, 0, 0, 0.3);

*z-index*: 10;

}

Because the modal window has the class of hidden, we can clearly see in the CSS file that the display is set to ‘none’. This means we will not see the modal window unless this class changes. We can use JavaScript to change these properties so that the modal window appears when we click a button.

Efficient Code

Because we will be using the same code repeatedly, it would be prudent to save some commands to variables;

*const* modal = document.querySelector(".modal");

*const* overlay = document.querySelector(".overlay");

*const* btnCloseModal = document.querySelector(".close-modal");

*const* btnsOpenModal = document.querySelectorAll(".show-modal");

console.log(btnsOpenModal);

After this we can begin to add some properties that can show and hide the modal window.

For Loop

for (*let* i = 0; i < btnsOpenModal.length; i++) {

btnsOpenModal[i].addEventListener("click", *function* () {

console.log("Button clicked");

modal.classList.remove("hidden");

overlay.classList.remove("hidden");

});

}

We can use the for loop to iterate through ‘btnsOpenModal’. Graphical user interface, text, application, email

Description automatically generated When logging to the console, this is what was displayed earlier. The for loop is simply iterating through these buttons looking for changes with .addEventListener property.

The event listener used here is ‘click’. After a click has been detected we execute a nameless function. Notice that we use the ‘modal.classList’ with the ‘remove’ operator. In this instance we have removed the ‘hidden’ class property.

After the hidden class property has been removed, the information within the box will become visible.

In this example we have simply removed a class. But we could have also edited the ‘hidden’ class;

modal.getElementsByClassName.display = 'block'

Which would change this:

.hidden {

*display*: none;

}

To this:

.hidden {

*display*: block;

}

This later example is not efficient. There is only one property here. What if there were 10 properties? We would have to write code that edited them all. It is far simpler to remove the class all together. That way we do not have to change multiple properties.

To be able to close the modal window on click;

});

btnCloseModal.addEventListener("click", *function* () {

modal.classList.add("hidden");

overlay.classList.add("hidden");

});

}

Currently our modal window will only close if we click on the ‘X’ button. To change this we could add the following code;

btnsOpenModal[i].addEventListener("click", *function* () {

modal.classList.remove("hidden");

overlay.classList.remove("hidden");

});

overlay.addEventListener("click", *function* () {

modal.classList.add("hidden");

overlay.classList.add("hidden");

});

}

This time we have added a function of the overlay class. When a click is detected on the overlay, it will set it to hidden.

There is a problem, we have used the same code repeatedly. Would it be possible to simply this code block and write it more simply?

The easiest way to do this would be to create a variable ‘closeModal’.

*const* closeModal = *function* () {

modal.classList.add("hidden");

overlay.classList.add("hidden");

};

btnCloseModal.addEventListener("click", closeModal);

overlay.addEventListener("click", closeModal);

};

This way we do not have to use the same code over and over again.

After refactoring:

//Open and Close Modal

*const* closeModal = *function* () {

modal.classList.add("hidden");

overlay.classList.add("hidden");

};

*const* openModal = *function* () {

modal.classList.remove("hidden");

overlay.classList.remove("hidden");

};

//For Loop to iterate through buttons

for (*let* i = 0; i < btnsOpenModal.length; i++) {

btnsOpenModal[i].addEventListener("click", openModal);

btnCloseModal.addEventListener("click", closeModal);

overlay.addEventListener("click", closeModal);

}

**Close on Key Press**

What if the individual presses escape while on the modal window? Or any other key for that matter? It is possible to implement reactions.

document.addEventListener("keydown", *function* (*e*) {

console.log(*e*);

});

Table

Description automatically generatedThis line of code creates an event listener that listens for button presses. When it detects one, it will log it to the console with information about the button.

Graphical user interface, text

Description automatically generatedHere we can see which button was pressed.

It just so happens that JavaScript is generating an object with all of the relevant information about the key press.

In this instance, the ‘Enter’ key was hit. If it was the ‘esc’ key the object would say key: ‘Escape’.

If we wanted to just read the property form the object we could write our function like this;

document.addEventListener("keydown", *function* (*e*) {

console.log(*e*.key);

});

Now, only the button press will be logged to the console, instead of the whole object.

**Closing on Escape**

document.addEventListener("keydown", *function* (*e*) {

console.log(*e*.key);

if (*e*.key === "Escape") {

console.log(`Escape was pressed.`);

if (!modal.classList.contains("hidden")) {

closeModal();

}

}

});

We first want to listen for the ‘Escape’ key press. ‘IF’ the escape key press was made we then want the following code to execute.

If the modal.classList.contains(‘hidden’), or rather in this case because we use the ‘!’ we have said DOES NOT CONTAIN ‘hidden’. Then we want to execute our closeModal function which will close the modal window.