https://extra-fdcp.scrimba.com/

MODAL css:

Position: fixed; ……….hides the modal

Adding top: 0

And bottom: 0;

Left: 0

Right: 0

Margin: auto;

will place it back in the plane

------------------------------------------------------------------------------------------------------------------------------------------

.setTimeout()

It takes two parameters (required) first is a function and the second is the time

Note: that the first parameter is the name of a function or an anonymous function

.setTimeout(nameOfFunction{

Code here

}, 3000)

------------------------------------------------------------------------------------------------------------------------------------------

Now we took control of the modal and use it’s id of ‘modal’ we set the modal.display.style = ‘inline’

const modal = document.getElementById('modal');

setTimeout(function () {

  modal.style.display = 'inline';

}, 1500);

This took our modal and made it reappear after 1.5seconds

------------------------------------------------------------------------------------------------------------------------------------------

Now we are going to target the close button at the top of the modal and will close it by doing the opposite of the having the modal appear using none using the click function inside the **addEventListener**

const modal = document.getElementById("modal");

const modalCloseBtn = document.getElementById("modal-close-btn");

modalCloseBtn.addEventListener("click", function () {

  modal.style.display = "none";

});

So just to be sure that you understand….first we target the “x” or the close button from the modal…now that it has been targeted use the addEventListener with the click function..and when the click has been executed take the modal itself and we will hide it by taking the display and setting it to none

------------------------------------------------------------------------------------------------------------------------------------------

Note on using “required”

<input

type="password"

id="astronautPassword"

name="astronautPassword"

placeholder="\*\*\*\*\*\*"

required

>

Required makes it compulsory so the user must fill out that section.

And we also gave the submit button the “type” of submit

<button type="submit">submit</button>

What this does is that when we apply prevent default behavior we will use the (submit) event to identify when the submit button has been clicked from the form

PREVENTING DEFAULT BEHAVIOUR

Astronaut example

//note that the preventDefault() will prevent the browser from listing the input fields entries in the browser field so your email, password, etc will not be displayed and will be hidden

loginForm.addEventListener('submit', function(e){

    e.preventDefault()

})

Modal example

const modal = document.getElementById("modal");

const modalCloseBtn = document.getElementById("modal-close-btn");

const consentForm = document.getElementById("consent-form");

consentForm.addEventListener("submit", function (e) {

  e.preventDefault();

  console.log("form submitted");

});

So what we are doing is we’ve targeted the consent form from the modal by getting the elementById

Andwe are using and addEventListener to listen for the submit event when the accept button has been clicked. Then what we will do is to add and event parameter of (e) and preventDefault(); to prevent the browser defaults behavior of refreshing the browser and displaying the data as a query string inside the browser field.

PREVENTING WEIRD BEHAVIOR OF HAVING TWO BUTTONS INSIDE THE SAME FORM NEXT TO EACH OTHER

<button class="modal-btn" type="submit">Accept</button>

      <button class="modal-btn" type="button">Decline</button>

So when you have two buttons next to each other there may be times when some unexpected behavior can happen. What you do is give the second button a type=’button’….by labeling it button inside the type field it prevents anything from happening.

consentForm.addEventListener("submit", function (e) {

  e.preventDefault();

  modalText.innerHTML = `<div class="modal-inner-loading">

    <img src="images/loading.svg" class="loading">

    <p id="uploadText">

        Uploading your data to the dark web...

    </p>

</div>`;

});

Ok here we are taking the modalText which is attached to the text that displays We have a strict policy on tracking and spamming: we'll definitely track you, and we'll definitely spam you. To use this site, enter your name and email address and accept our ridiculous terms and conditions.

Now this will change once the accept button has been clicked which will fire off the submit event then triggering off a preventDefault() and taking the modalText.innerHTML and changing the text to the loading svg image

Changing the text to ‘Making the sale’ and keeping the svg image

consentForm.addEventListener("submit", function (e) {

  e.preventDefault();

  modalText.innerHTML = `

  <div class="modal-inner-loading">

    <img src="images/loading.svg" class="loading">

    <p id="uploadText">

        Uploading your data to the dark web...

    </p>

</div>`;

  setTimeout(function () {

    document.getElementById("uploadText").innerText = `Making the sale`;

  }, 1500);

});

Here we are adding a setTimeout function below the svg image. So after 1.5seconds the text will change only to `Making the sale` but the svg will remain. We still target the ‘uploadText’ and use .innerText as we are only changing text and not html

Here we are changing the last part of the modal to change again from the svg loading image and replacing the html using .innerHTML after another 1.5seconds from the svg loading image from above and also having a pirate picture…note that we are not getting the id using getElementById…we are simply going right to the modal-inner from the html itself

setTimeout(function () {

    document.getElementById(

      "modal-inner"

    ).innerHTML = `<h2>Thanks you sucker! </h2>

    <p>We just sold the rights to your eternal soul.</p>

    <div class="idiot-gif">

        <img src="images/pirate.gif">

    </div>`;

  }, 3000);

FORMDATA………..

We use the FormData function to retrieve data from the form submitted. First we need to target the form..which ours is called consentForm….then we need to create a new instance of FormData and pass the consentForm as a parameter inside FormData(consentForm) and store it inside a variable

const modalFormData = new FormData(consentForm);

next we can use this modalFormData variable to use the .get() function to retrieve any of the input values ie: name, email, password…by using the (name=) entry from the form

here we have the modalFormData.get(“fullName”)…which fullName is from input field of fullName from the name= section.

const fullName = modalFormData.get("fullName");

ADD to CART button..

So when the add to cart button is at zero we want it to not be accessible or disabled and when the quantity is greater than 0 then we want it to be accessible or enable it for the user. It’s a matter of two fixes and nothing to actually do with the add to cart addEventListener button….

We want to add another value to the decrement button and to the increment button addEventListeners….

So if you look below in the if statement we just added the cartBtn.diabled to equal to being true so the when the quantity is at 0 then the addToCard button will no longer be accessible.

And when we increment (note that when the page loads the quantity will be set at zero so in our html we set the addToCard button to disabled by default.) then the addTo Cart button will be enabled or accessible to to the user. With cartBtn.disabled = false;

decrement.addEventListener("click", function () {

  quantity--;

  if (quantity === 0) {

    decrement.disabled = true;

    cartBtn.disabled = true;

  }

  quantityDisplay.innerText = quantity;

});

increment.addEventListener("click", function () {

  quantity++;

  decrement.disabled = false;

  cartBtn.disabled = false;

  quantityDisplay.innerText = quantity;

});