***DAY 2 Event, Validation, Promises***

**EVENT (Something a browser or user does)**

* Page has finished loading
* HTML input was changed
* An HTML button was clicked (submit button onclick())
* **setTimeout** delays a task to run once after a set time.
* **setInterval** keeps running a task repeatedly at regular time intervals.
* **clearInterval** stops the repeated task started by setInterval.

**Page on load event**

<body onload="alert('Page is loaded')">

**Button Clicked event**

<button

        onclick="this.innerHTML = Date()">Time is :

</button>

**Mouse Entered/left event**

    <button

        onmouseenter="this.innerHTML = 'Mouse Entered'"

        onmouseleave="this.innerHTML = 'Mouse Left'">Mouse

    </button>

**Key pressed**

    <input type="text" onkeypress="pressed()"/>

**Set Timeout, setInterval, clearInterval (There are other events other than these 3)**

<button onclick="setTimeout(buttonclick,2000)">Timeout</button>\

<button onclick="meow= setInterval(btnclick,2000)">Interval</button>

<button onclick="clearInterval(meow)">Stop Interval</button>

**Form Validation**

* Form to add vendors, users, with ID, name, rating, and products and so on
* Validation for unique IDs, name length (>=5), and products
* Dynamic product fields (add/remove)
* Display vendors sorted by rating
* Search functionality
* View detailed information in a modal
* Clear all vendor data functionality
* Store vendors as objects in **localStorage**
* Sort vendors by rating
* Validate input before saving
* Learned how to delete data from local Storage

**Promises**

* Learned about Promises, which are objects representing the eventual completion or failure of asynchronous operations.
* Can be used for Error Handling
* States: **resolve(), reject()**
* Similar to try catch finally

**Fetch an image from a UPI**

* Get image from an API
* Store it into a variable
* fetch(), then()
* async function

async function getCat()

await fetch(url).then(res => res.json()).then(data => data.url)

* await to fetch a data from API