**DAY - 2**

**1. Event Handling in JavaScript**

* **Description:** JavaScript allows you to execute specific code in response to events that occur in the browser, such as user interactions or the page loading.
* **Simple Code Example:**

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

<button onclick="this.innerHTML = Date()">

The time is:

</button>

<button

onmouseenter="this.style.color = 'blue'"

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

Mouse

</button>

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

</body>

</html>

**2. setInterval() Function**

* **Description:** The setInterval() method repeatedly calls a function or executes a code snippet, with a fixed time delay between each call. 1
* **Simple Code Example:**

<script>

function btnclick() {

console.log('printing after 2 sec')

}</script>

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

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

</body>

**3. setTimeout() Function**

* **Description:** The setTimeout() method sets a timer which executes a function or specified piece of code once the timer expires.
* **Simple Code Example:**

<script>

function btnclick() {

alert('btn is clicked')

}</script>

<button onclick="meow = setTimeout(btnclick, 3000)">Timeout</button>

<button onclick="clearTimeout(meow)">Stop</button>

**4. Promises in JavaScript (Commented Out)**

* **Description:** Promises are used for asynchronous operations in JavaScript. They represent a value that might be available now, or in the future, or never. The provided code snippet shows a basic structure of a Promise.
* **Simple Code Example :**

let myPromise = new Promise((resolve, reject) => {

resolve();

reject();

});

myPromise.then(

function(value) {

// let res = 0/ 0;

console.log("promise got resolved");

},

function(error) {

console.log("promise got rejected")

}

);  
  
**5. Async/Await with Fetch API**

* **Description:** The async and await keywords are used to simplify asynchronous JavaScript code, making it look and behave a bit more like synchronous code. The fetch API is used to make network requests. This example fetches a cat image URL from an API and displays it.
* **Simple Code Example:**

<script>

async function getCat() {

const url = "https://cataas.com/cat?width=200;height=200;json=true";

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

console.log(caturl);

document.getElementById("catimg").src = caturl;

// return caturl;

}</script>

<body>

<img style="width: 200px; height: 200px;" id="catimg" src="" alt="cat"/>

<button onclick="getCat()">Get Cat</button>

</body>