

**Batch: B2 Roll No.: 16010421119 Experiment No.:6**

**Aim**: Event handling using JavaScript to explore web browser environments.

**Resources needed: Notepad++, Web Browser**

**Theory:**

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

An HTML web page has finished loading

An HTML input field was changed

An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

**What can JavaScript Do?**

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button ● Content that should be verified when a user inputs data

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled

**Syntax:**

<element event='some JavaScript'>

<element event="some JavaScript">

**Example**

In the following example, an onclick attribute (with code), is added to a <button> element:

<!DOCTYPE html>

<html>

<body>

<button onclick="document.getElementById('demo').innerHTML=Date()">The time is?</button>

<p id="demo"></p>

</body>

</html>

**JavaScript HTML DOM Events**

HTML DOM allows JavaScript to react to HTML events:

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element. To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute: onclick=JavaScript

**EXAMPLE**

<!DOCTYPE html>

<html>

<body>

<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>

</body>

</html>

**Activity:**

Apply following JS events on your web pages

Input Events

* Onblur
* onreset

Mouse Events

* Onmouseover
* Onmousedown

Click Events

* Onclick
* Ondbclick

Apply following DOM events to your webpages

* Onload
* Onchange
* onmouseover

KJSCE/IT/SY/SEM IV/WP1/2020-21

**Results: (Program printout with output)**

**import React, { useState } from 'react'**

**import './Login.css'**

**import Button from '../../components/Button/Button'**

**import Message from '../../components/Message/Message'**

**import { useNavigate } from 'react-router-dom'**

**function Login() {**

**const [username, setUsername] = useState('')**

**const [password, setPassword] = useState('')**

**const [error, setError] = useState(false)**

**const navigate = useNavigate()**

**const handleUsername = (event) => {**

**setUsername(event.target.value)**

**}**

**const handlePassword = (event) => {**

**setPassword(event.target.value)**

**}**

**const handleSubmit = (event) => {**

**if (username === '' || password === '') {**

**setError(true)**

**event.preventDefault()**

**return false**

**}**

**else if(username==='Aarya' && password==='password')**

**{**

**setError(false)**

**navigate('/')**

**}**

**else {**

**setError(true)**

**console.log(username, password)**

**}**

**}**

**return (**

**<div className='main-login'>**

**<h2 className='login-text'>Login</h2>**

**<form method='post'>**

**<div className='main-form'>**

**{error && <Message type='red' text='Please input all fields' />}**

**<input type="text" value={username} name="text" className="input" onChange={handleUsername} placeholder="Enter Your Username" />**

**<input type="password" value={password} name="text" className="input" onChange={handlePassword} placeholder="Enter Your Password" />**

**<Button text='Submit' click={handleSubmit} type="submit"/>**

**<h5>Dont have an account??</h5>**

**<a href="/signup" className='signup-link'>Signup</a>**

**</div>**

**</form>**

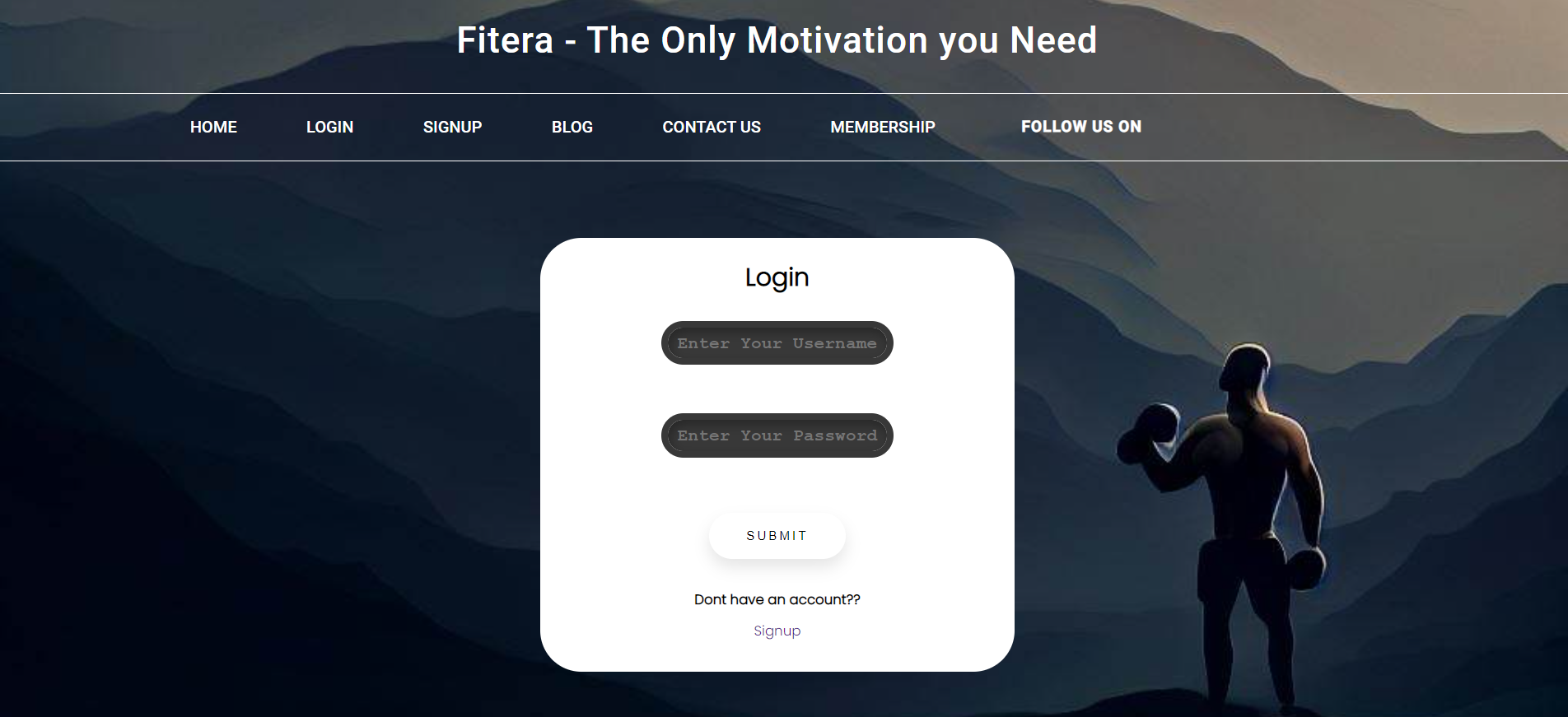
**</div>**

**)**

**}**

**export default Login**

**OUTPUT:-**

****

**Questions:**

**Q1) What are the different types of load events**

**Ans: -**

When you open a page, the following events occur in sequence:

**DOMContentLoaded –** the browser fully loaded HTML and completed building the DOM tree. However, it hasn’t loaded external resources like stylesheets and images. In this event, you can start selecting DOM nodes or initialize the interface.

**load –** the browser fully loaded the HTML and also external resources like images and stylesheets.

When you leave the page, the following events fire in sequence:

**beforeunload –** fires before the page and resources are unloaded. You can use this event to show a confirmation dialog to confirm if you really want to leave the page. By doing this, you can prevent data loss in case you are filling out a form and accidentally click a link to navigate to another page.

**unload –** fires when the page has completely unloaded. You can use this event to send the analytic data or to clean up resources.

Q2) **Explain Onkeypress , onkeyup events**

Ans: - The onkeyup event executes a JavaScript function when the user releases a key. The onkeyup method allows derived classes to handle the event without attaching the delegate. This is the preferred technique for handling the event in derived classes.

The keypress event only fires for keys that have a printable representation. The onkeypress event is not fired for all keys; for example, for ALT, SHIFT, CTRL, ESC, and BACKSPACE in all browsers, this event is fired only when the user presses a key.

**Outcomes:**

**CO3: Apply JavaScript and JSON for web application development**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

**We can Conclude that we have learnt about event handling in JS.**

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

**References:**

**Books/ Journals/ Websites:**

* “Web technologies: Black Book”, Dreamtech Publications
* [http://www.w3schools.com](http://www.w3schools.com/)