Experiment No. 06

Title: Event handling using JavaScript to explore web browser environment.

Batch: A1 **Roll No.: 16010422013** **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

Results: (Program printout with output)

index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title> Interactive Form</title>
  <link rel="stylesheet" href="style.css">
</head>
<body onload="showOnLoadMessage()">
  <header>
    <h1>Event Handling using JS</h1>
  </header>

  <main>
    <section class="form-section">
      <h2>Enter Your Information</h2>
      <form id="userInfoForm" onreset="resetForm()">
        <label for="nameInput">Name:</label>
        <input type="text" id="nameInput" onblur="showBlurMessage('name')"
onchange="validateField('name')" placeholder="Enter your name" required>
        <br>
        <label for="emailInput">Email:</label>
        <input type="email" id="emailInput" onblur="showBlurMessage('email')"
onchange="validateField('email')" placeholder="youremail@example.com" required>
        <br>
        <label for="phoneInput">Phone Number (Optional):</label>
        <input type="tel" id="phoneInput" onblur="showBlurMessage('phone')"
onchange="validateField('phone')" placeholder="+91" required>
        <br>
        <button type="button" id="submitButton"
onclick="submitForm()">Submit</button>
        <button type="reset">Reset</button>
      </form>
      <p id="formMessage"></p>
    </section>
  </main>

  <script src="script.js"></script>
</body>
</html>
```

style.css:

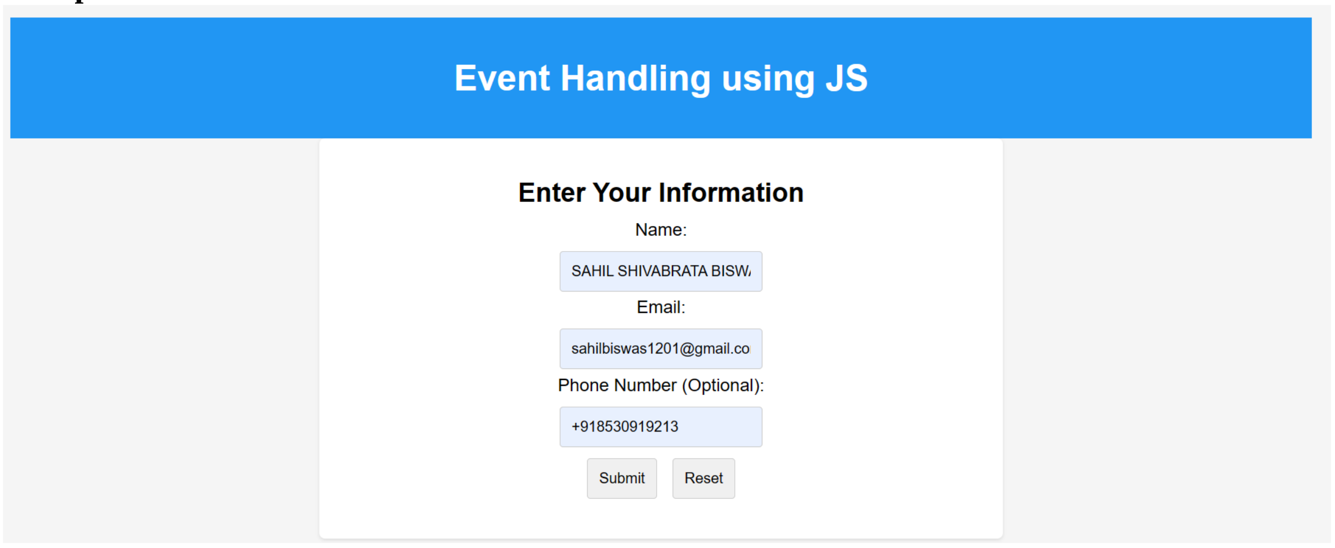
```css
body {
    font-family: sans-serif;
    margin: 0;
    padding: 20px;
    display: flex;
    flex-direction: column;
```

```css
  min-height: 100vh;
  background-color: #f5f5f5;
}

header {
  text-align: center;
  background-color: #2196f3;
  color: white;
  padding: 15px;
}

main {
  flex: 1;
  display: flex;
  flex-direction: column;
  align-items: center;
}

section {
  margin-bottom: 20px;
  padding: 15px;
  border-radius: 5px;
  background-color: white;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
  width: 50%;
  text-align: center;
}

h2 {
  margin-bottom: 10px;
}

label {
  display: block;
  margin-bottom: 5px;
}

input, button {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 3px;
  margin: 5px;
}

button:hover {
  background-color: #eee;
  cursor: pointer;
}

#formMessage {
  font-style: italic;
  color: #666;
}
```

script.js:

```
function showOnLoadMessage() {
    const formMessage = document.getElementById("formMessage");
    formMessage.textContent = "Welcome! Please fill out the form.";
  }

  function showBlurMessage(fieldId) {
    const inputField = document.getElementById(fieldId + "Input");
    const formMessage = document.getElementById("formMessage");
    if (inputField.value !== "") {
      formMessage.textContent = "You entered " + fieldId + ": " + inputField.value;
    } else {
      formMessage.textContent = "";
    }
  }

  function validateField(fieldId) {
    const inputField = document.getElementById(fieldId + "Input");
    const formMessage = document.getElementById("formMessage");

    if (fieldId === "email") {
      const emailRegex = /^[^@]+@[^@]+\.[a-z]{2,}$/i;
      if (!emailRegex.test(inputField.value)) {
        formMessage.textContent = "Please enter a valid email address.";
        inputField.focus();
      } else {
        formMessage.textContent = "";
      }
    }

    if (fieldId === "phone") {
      const phoneRegex = /^\d{3}-\d{3}-\d{4}$|^\(\d{3}\) \d{3}-\d{4}$/;
      if (!phoneRegex.test(inputField.value)) {
        formMessage.textContent = "Please enter a valid phone number XXXXXXXXXXX.";
        inputField.focus();
      } else {
        formMessage.textContent = "";
      }
    }
  }

  function resetForm() {
    const form = document.getElementById("userInfoForm");
    form.reset();
    const formMessage = document.getElementById("formMessage");
    formMessage.textContent = "";
  }

  function submitForm() {
    const name = document.getElementById("nameInput").value;
    const email = document.getElementById("emailInput").value;
    const phone = document.getElementById("phoneInput").value;

    const formMessage = document.getElementById("formMessage");
    formMessage.textContent = "Thank you, " + name + "! Your information has been
submitted.";
  }

    window.onload = showOnLoadMessage;
```

Output:



Event Handling using JS

Enter Your Information

Name:

SAHIL SHIVABRATA BISW.

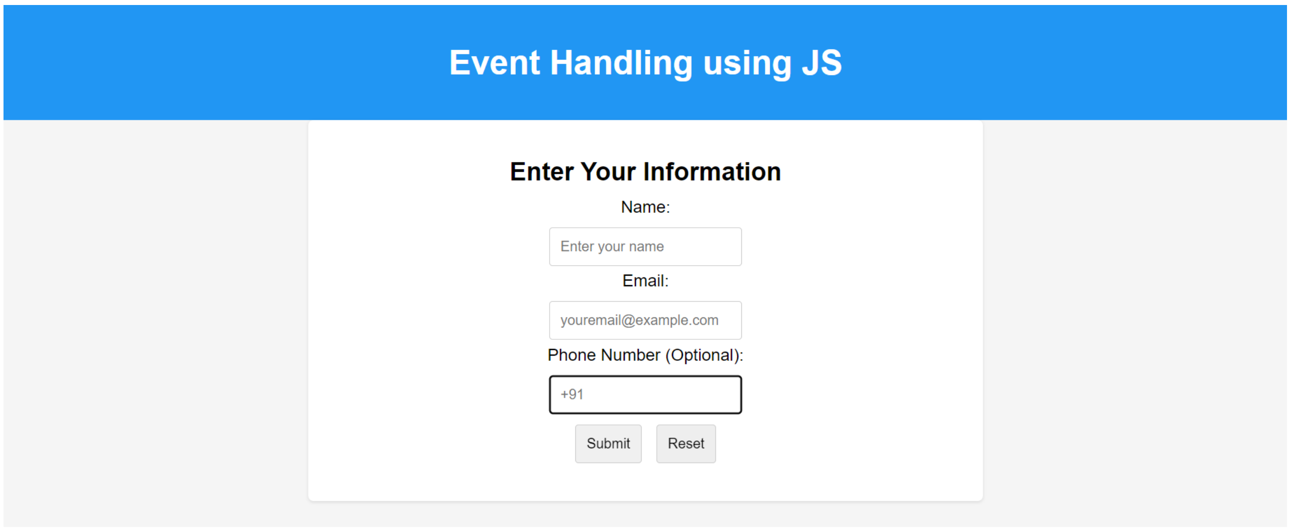
Email:

sahilbiswas1201@gmail.co

Phone Number (Optional):

+918530919213

After clicking on “Reset” button:



Event Handling using JS

Enter Your Information

Name:

Enter your name

Email:

youremail@example.com

Phone Number (Optional):

+91

Questions:

Q1) What are the different types of load events

Ans. There are actually two main load events you'll encounter in JavaScript event handling:

load: This event fires when the entire page, including all its resources (images, scripts, stylesheets, etc.), has finished loading. This is the most common load event used in JavaScript. You can use it to ensure your code executes only after everything is loaded and ready.

DOMContentLoaded: This event fires specifically when the HTML document has been completely parsed and the Document Object Model (DOM) is ready. This means the HTML structure is available for manipulation, but external resources like images and stylesheets might still be loading.

Q2) Explain Onkeypress , onkeyup events
Ans. JavaScript provides various event listeners to capture user interactions with webpages. Among these, onkeypress and onkeyup are specifically designed to handle keyboard events. But they target different stages of a keypress:

1. onkeypress:

Imagine you're typing a message. onkeypress fires when you press a key and then release it, but only for keys that generate characters you can type (letters, numbers, symbols). In essence, it activates when a character is fully entered.
It's primarily concerned with keys that produce characters, not special keys like Shift, Ctrl, Alt, or function keys.

2. onkeyup:

This event focuses solely on when you release a key. It doesn't matter if the key generates a character or not; onkeyup fires for all keys on the keyboard. It activates the moment you release a key, regardless of whether it produces a character. This event is ideal for scenarios where you want to detect any key press and release, offering broader functionality.

For general key detection, regardless of character output, use onkeyup as it's more comprehensive. Due to its limitations, avoid onkeypress. If character detection is crucial, consider using onkeydown (fires when a key is pressed) along with onkeyup for a more robust solution.

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

---

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

Learnt Event handling using JavaScript to explore web browser environments.

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