JavaScript - Form Events

Form Events

The form events in JavaScript are events that are associated with HTML forms. These events are triggered by user actions when interacting with form elements like text fields, buttons, checkboxes, etc. Form events allow you to execute JavaScript code in response to these actions, enabling you to validate form data, perform actions on form submission or reset, and enhance the user experience.

JavaScript form events are hooked onto the elements in the Document Object Model also known as DOM where by default the bubbling propagation is used i.e. from bottom (children) to top(parent).

List of Common Form Events

Here are some common form events:

Form Event	**Description**
onsubmit	Triggered when a form is submitted. It's often used for form validation before data is sent to the server.
onreset	Triggered when the form is reset, allowing you to perform actions when the user resets the form.
onchange	Triggered when the value of a form element (input, select, textarea) changes. Commonly used for user input validation or dynamic updates.
oninput	Triggered immediately when the value of an input element changes, allowing for real-time handling of user input.
onfocus	Triggered when an element receives focus, such as when a user clicks or tabs into an input field. Useful for providing feedback or enhancing the user experience.
onblur	Triggered when an element loses focus, such as when a user clicks outside an input field or tabs away. Useful for validation or updates triggered by loss of focus.

Examples

Example: The onchange Event

The provided instance below illustrates the functionality of the onchange event. This event activates upon a user's alteration in dropdown (<select>) option selection. The function, handleChange, dynamically modifies an <h2> element to display the newly selected country; thus offering immediate feedback as user preferences evolve.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <label for="country">Select a country:</label>
   <select id="country" onchange="handleChange()">
      <option value="USA">USA</option>
      <option value="Canada">Canada</option>
      <option value="UK">UK</option>
      <option value="India">India</option>
   </select>
   <p id="txt"></p>
   <script>
      function handleChange() {
         // Perform actions when the dropdown selection changes
         var selectedCountry = document.getElementById('country').value;
         document.getElementById("txt").textContent=
         "Selected country: "+selectedCountry;
     }
   </script>
</body>
</html>
```

Example: The onsubmit Event

The following example highlights the onsubmit event's functionality upon form submission. The form features a username field and password field; both must be filled for successful validation when invoking the validateForm function. Upon passing this validation, submitting the form will trigger display of a confirmation message.

Open Compiler

```
<!DOCTYPE html>
```

```
<html>
<body>
   <form onsubmit="return validateForm()">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required>
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required>
      <br/>
      <input type="submit" value="Submit">
   </form>
   <script>
      function validateForm() {
         var username = document.getElementById('username').value;
         var password = document.getElementById('password').value;
         // Perform validation
         if (username === "" || password === "") {
            alert("Please fill in all fields");
            return false; // Prevent form submission
         }
         alert("Form submitted! Username is:"+username+",Password is:"+password
         return true; // Allow form submission
      }
   </script>
</body>
</html>
```

Example: The onreset event

In this demonstration, we observe the onreset event in action: it triggers upon the user's click of the "Reset" button within a form. The resetForm function once invoked, clears the form content filled by user and then displays an alert to confirm successful reset of said form.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <form onreset="resetForm()">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required>
```

```
      <input type="reset" value="Reset">
   </form>
   <script>
      function resetForm() {
         // Perform actions when the form is reset
         alert("Form has been reset!");
      }
   </script>
</body>
</html>
```

Example: The oninput Event

This example illustrates the oninput event: as the user types into the search input field a real-time action, indeed! The handleInput function triggers; it logs each current search input directly to screen.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <label for="search">Search:</label>
   <input type="text" id="search" oninput="handleInput()">
   <div id="message" style=" margin-top: 10px; font-weight: lighter;border: 1px

   <script>
      var messageElement = document.getElementById('message');
      function handleInput() {
         // Perform actions as the user types
         var searchInput = document.getElementById('search').value;
         messageElement.innerHTML+="Search input: " + searchInput+'<br>';
      }
   </script>
</body>
</html>
```

Example: onfocus and onblur Events

The onfocus and onblur events merge in this example. The user's focus on the input field triggers a call to the handleFocus function, which then logs a message into the

console. In contrast, when clicks outside of or tabs away from said input field – this action triggers execution of another function called handleBlur that subsequently records an alternative message within that same console log.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <label for="name">Name:</label>
   <input type="text" id="name" onfocus="handleFocus()" onblur="handleBlur()">
   <p id= "output"></p>
   <script>
      const output = document.getElementById('output');
      function handleFocus() {
         // Perform actions when the input gets focus
         output.innerHTML += "Input has focus" + "<br>";
      }
      function handleBlur() {
         // Perform actions when the input loses focus
         output.innerHTML += "Input lost focus" + "<br>";
      }
   </script>
</body>
</html>
```