Toggle Confirmation Handler - Complete Documentation

Afficher l'image Afficher l'image Afficher l'image Afficher l'image

Enterprise-grade confirmation dialog system for toggle actions (activate/deactivate accounts, enable/disable features, etc.). Framework-agnostic with built-in support for React, Vue, Angular, jQuery, and Vanilla JavaScript.

Table of Contents

Reference

- TypeScript Types
- Error Classes
- Troubleshooting

Installation

NPM / Yarn

bash

```bash
npm install @wlindabla/form_validator sweetalert2
# or
yarn add @wlindabla/form_validator sweetalert2
```

Dependencies

- **jQuery** (for legacy browser support)
- **SweetAlert2** (for beautiful modals)

Import

typescript

```typescript
// ES6 Module
import {
    handleToggleConfirmation,
    extractToggleData,
    createToggleEvent
} from '@wlindabla/form_validator';

// CommonJS
const {
    handleToggleConfirmation
} = require('@wlindabla/form_validator');
```

Basic Usage

HTML Setup

html

```
<div class="user-list">
  <!-- Account toggle button -->
  <a href="/admin/users/123/toggle"
    class="btn btn-toggle-account"
    data-action-confirm="Are you sure you want to deactivate this account?"
    data-toggle-enabled="true"
    title="Deactivate Account"
    data-additional='{"userId": 123, "userName": "john.doe"}'>
     <i class="fa fa-ban"></i> Deactivate
  </a>
</div>
```

JavaScript Usage

typescript

```
import { handleToggleConfirmation } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

// Handle button click
document.querySelector('.btn-toggle-account').addEventListener('click', async (event) => {
    event.preventDefault();

    const confirmed = await handleToggleConfirmation({
        element: event.currentTarget as HTMLElement,
        eventName: 'account:toggle:confirmed',
        dialogHandler: Swal.fire
    });

    if (confirmed) {
        console.log('User confirmed the action');
    }
});

// Listen for the custom event
document.addEventListener('account:toggle:confirmed', (event: CustomEvent) => {
    console.log('Toggle confirmed:', event.detail);
    // event.detail contains: { data, url_action_confirm, sourceElement, timestamp }
});
```

Why Toggle Handler?

✨ Key Features

✔ **Framework-Agnostic** - Works with any JavaScript framework
✔ **Type-Safe** - Full TypeScript support with strict typing
✔ **Legacy Browser Support** - Uses jQuery for IE8+ compatibility
✔ **Beautiful Dialogs** - SweetAlert2 integration for modern UIs
✔ **Customizable** - Extensive configuration options
✔ **Event-Driven** - Custom events for decoupled architecture
✔ **Translation Ready** - Built-in i18n support
✔ **Error Resilient** - Comprehensive error handling
✔ **Production Ready** - Battle-tested in enterprise applications

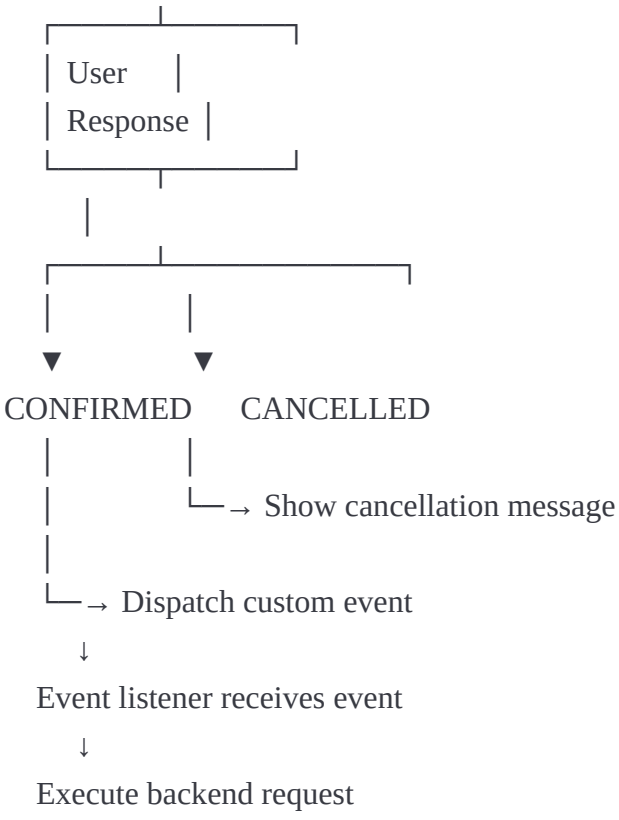
🎯 Use Cases

- ✔ Account activation/deactivation
- ✔ Feature enable/disable toggles
- ✔ Permission grants/revokes
- ✔ Status changes (active/inactive)
- ✔ Subscription enable/disable
- ✔ Any action requiring confirmation

How It Works

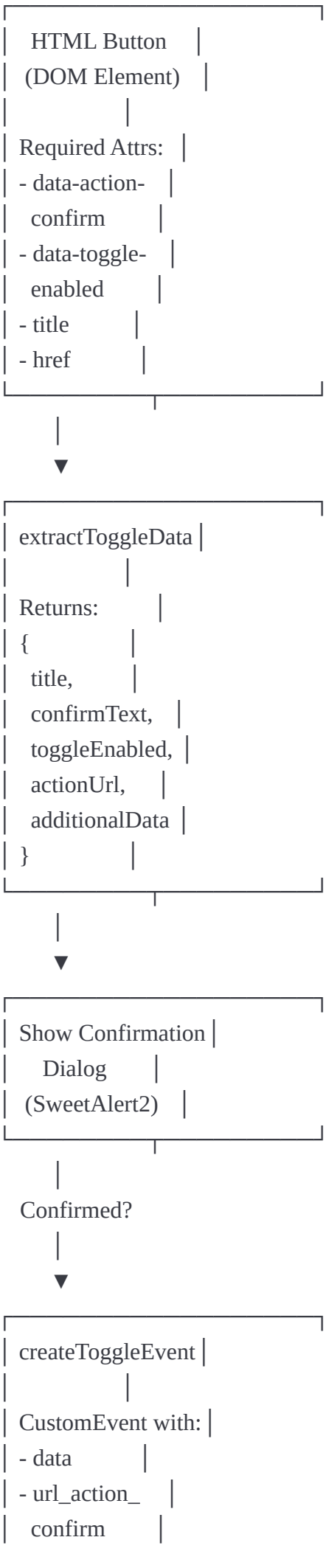
Process Flow

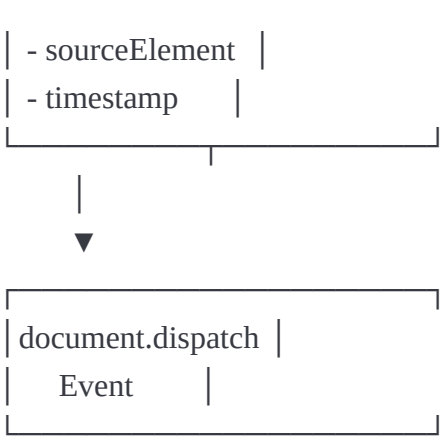
1. User clicks toggle button
   ↓
2. handleToggleConfirmation() called
   ↓
3. Extract data from DOM attributes
   ↓
4. Show confirmation dialog (SweetAlert2)
   ↓



   ↓
   Event listener receives event
   ↓
   Execute backend request

Data Flow Diagram


HTML Button
(DOM Element)
Required Attrs:
- data-action-confirm
- data-toggle-enabled
- title
- href
extractToggleData
Returns:
{
title,
confirmText,
toggleEnabled,
actionUrl,
additionalData
}
Show Confirmation Dialog
(SweetAlert2)
Confirmed?
createToggleEvent
CustomEvent with:
- data
- url_action_confirm



HTML Requirements

Required Attributes

Attribute	Type	Description	Example
data-action-confirm	string	Confirmation message	"Are you sure?"
data-toggle-enabled	boolean	Current state	"true" or "false"
title	string	Dialog title	"Deactivate Account"
href or data-url	string	Action URL	"/admin/users/123/toggle"

Optional Attributes

Attribute	Type	Description
data-additional	JSON	Additional data (must be valid JSON)
data-bs-original-title	string	Bootstrap 5 tooltip title (fallback)
data-original-title	string	Older Bootstrap tooltip title (fallback)

Complete HTML Example

html

```html
<!-- Simple example -->
<button
    class="btn btn-danger"
    data-action-confirm="Deactivate this user's account?"
    data-toggle-enabled="true"
    title="Deactivate Account"
    data-url="/api/users/123/toggle">
    Deactivate
</button>

<!-- Advanced example with additional data -->
<a href="/admin/users/456/toggle"
   class="btn-toggle-account"
   data-action-confirm="Are you sure you want to suspend this account? This action can be reversed later."
   data-toggle-enabled="true"
   title="Suspend Account"
   data-additional='{"userId": 456, "userName": "jane.smith", "userEmail": "jane@example.com"}'>
    <i class="fa fa-user-times"></i> Suspend Account
</a>

<!-- With Bootstrap tooltip -->
<button
    class="btn btn-warning"
    data-action-confirm="Enable premium features for this user?"
    data-toggle-enabled="false"
    data-bs-toggle="tooltip"
    data-bs-original-title="Enable Premium Features"
    data-url="/api/users/789/enable-premium">
    Enable Premium
</button>
```

API Documentation

extractToggleData()

Extracts and validates toggle data from a DOM element.

typescript

```typescript
function extractToggleData(element: HTMLElement): ExtractedToggleData
```

Parameters:

Parameter	Type	Description
element	HTMLElement	DOM element with toggle attributes

Returns: `ExtractedToggleData`

typescript

```
interface ExtractedToggleData {
  title: string;
  actionConfirmText: string;
  toggleEnabled: boolean;
  actionUrl: string;
  additionalData: Record<string, unknown>;
}
```

Throws:

- `TypeError` - If element is null/undefined
- `Error` - If jQuery is not available
- `MissingAttributeError` - If required attributes are missing

Example:

typescript

```
const button = document.querySelector('.btn-toggle');
const data = extractToggleData(button);

console.log(data);
// {
//   title: "Deactivate Account",
//   actionConfirmText: "Are you sure?",
//   toggleEnabled: true,
//   actionUrl: "/admin/users/123/toggle",
//   additionalData: { userId: 123 }
// }
```

---

createToggleEvent()

Creates a custom event with toggle data.

typescript

```
function createToggleEvent(
  eventName: string,
  toggleData: ExtractedToggleData,
  sourceElement: HTMLElement
): CustomEvent<ToggleEventDetail>
```

Parameters:

Parameter	Type	Description
`eventName`	`string`	Custom event name
`toggleData`	`ExtractedToggleData`	Extracted toggle data
`sourceElement`	`HTMLElement`	Source DOM element

Returns: `CustomEvent<ToggleEventDetail>`

Example:

typescript

```
const data = extractToggleData(element);
const event = createToggleEvent('account:toggle', data, element);

// Event detail structure
console.log(event.detail);
// {
//   data: { status: true, userId: 123 },
//   url_action_confirm: "/admin/users/123/toggle",
//   sourceElement: <HTMLElement>,
//   timestamp: "2024-01-15T10:30:00.000Z"
// }

// Dispatch event
document.dispatchEvent(event);
```

---

handleToggleConfirmation()

Main function that handles the entire confirmation workflow.

typescript

```typescript
async function handleToggleConfirmation(
  params: ToggleConfirmationParams
): Promise<boolean>
```

Parameters:

typescript

```typescript
interface ToggleConfirmationParams {
  element: HTMLElement;                // Required
  eventName: string;                   // Required
  dialogHandler: (options) => Promise<...>;// Required (Swal.fire)
  translator?: (key: string) => string;    // Optional
  confirmDialogConfig?: Partial<SweetAlertOptions>; // Optional
  cancelDialogConfig?: Partial<SweetAlertOptions>;  // Optional
  onConfirm?: (data, event) => void | Promise<void>; // Optional
  onCancel?: (data) => void | Promise<void>;        // Optional
  onError?: (error) => void;                   // Optional
}
```

Returns: `Promise<boolean>` - `true` if confirmed, `false` if cancelled

Throws: `ToggleConfirmationError` - If handling fails and no error handler provided

Example:

typescript

```
const confirmed = await handleToggleConfirmation({
    element: buttonElement,
    eventName: 'account:toggle:confirmed',
    dialogHandler: Swal.fire,

    // Optional: Translation function
    translator: (key) => translations[key],

    // Optional: Custom dialog config
    confirmDialogConfig: {
        confirmButtonColor: '#d33',
        cancelButtonColor: '#3085d6'
    },

    // Optional: Callbacks
    onConfirm: async (data, event) => {
        console.log('Confirmed:', data);
    },

    onCancel: (data) => {
        console.log('Cancelled');
    },

    onError: (error) => {
        console.error('Error:', error);
    }
});

if (confirmed) {
    console.log('Action confirmed');
}
```

---

handleToggleConfirmationSafe()

Safe version that doesn't throw errors.

typescript

```
async function handleToggleConfirmationSafe(
    params: ToggleConfirmationParams
): Promise<{ success: boolean; confirmed?: boolean; error?: string }>
```

Example:

```typescript
const result = await handleToggleConfirmationSafe({
    element: buttonElement,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire
});

if (result.success) {
    console.log('Confirmed:', result.confirmed);
} else {
    console.error('Error:', result.error);
}
```

---

hasRequiredToggleAttributes()

Checks if an element has all required attributes.

```typescript
function hasRequiredToggleAttributes(element: HTMLElement): boolean
```

Example:

```typescript
const button = document.querySelector('.btn-toggle');

if (hasRequiredToggleAttributes(button)) {
    // Safe to extract data
    const data = extractToggleData(button);
} else {
    console.error('Missing required attributes');
}
```

Configuration Options

Default Confirmation Dialog

typescript

```typescript
const DEFAULT_CONFIRM_DIALOG_CONFIG = {
  icon: "question",
  position: "top",
  showCancelButton: true,
  animation: true,
  allowOutsideClick: false,
  allowEscapeKey: false,
  background: "#00427E",
  color: "#fff",
  confirmButtonText: "Confirm",
  cancelButtonText: "Cancel",
  showClass: {
    popup: "animate__animated animate__fadeInUp animate__faster"
  },
  hideClass: {
    popup: "animate__animated animate__fadeOutDown animate__faster"
  }
};
```

Default Cancellation Dialog

typescript

```typescript
const DEFAULT_CANCEL_DIALOG_CONFIG = {
  icon: "info",
  position: "top",
  showConfirmButton: false,
  timer: 20000,
  timerProgressBar: true,
  background: "#00427E",
  color: "#fff",
  showCloseButton: true
};
```

Custom Configuration Example

typescript

```
  await handleToggleConfirmation({
    element: buttonElement,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire,

    confirmDialogConfig: {
      icon: 'warning',
      background: '#fff',
      color: '#333',
      confirmButtonColor: '#d33',
      cancelButtonColor: '#3085d6',
      confirmButtonText: 'Yes, deactivate!',
      cancelButtonText: 'No, keep it',
      showClass: {
        popup: 'animate__animated animate__bounceIn'
      }
    },

    cancelDialogConfig: {
      icon: 'success',
      title: 'Cancelled',
      text: 'Account remains active',
      timer: 3000
    }
  });
```

Framework Integration

Vanilla JavaScript Integration

Basic Setup

javascript

```
import { handleToggleConfirmation } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

// Initialize on page load
document.addEventListener('DOMContentLoaded', () => {
    initToggleHandlers();
    initToggleListener();
});

function initToggleHandlers() {
    // Use event delegation for dynamic elements
    document.body.addEventListener('click', async (event) => {
        const target = event.target.closest('.btn-toggle-account');
        if (!target) return;

        event.preventDefault();

        await handleToggleConfirmation({
            element: target,
            eventName: 'account:toggle:confirmed',
            dialogHandler: Swal.fire
        });
    });
}

function initToggleListener() {
    document.addEventListener('account:toggle:confirmed', async (event) => {
        const { data, url_action_confirm } = event.detail;

        try {
            const response = await fetch(url_action_confirm, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'X-CSRF-Token': getCsrfToken()
                },
                body: JSON.stringify(data)
            });

            const result = await response.json();

            if (result.success) {
                Swal.fire('Success!', result.message, 'success');
                setTimeout(() => location.reload(), 2000);
```

```
            }
        } catch (error) {
            console.error('Toggle failed:', error);
            Swal.fire('Error', 'An error occurred', 'error');
        }
    });
}

function getCsrfToken() {
    return document.querySelector('meta[name="csrf-token"]')?.content || '';
}
```

Advanced Example with Multiple Toggle Types

javascript

```
// Multiple toggle types
const TOGGLE_EVENTS = {
    ACCOUNT: 'account:toggle:confirmed',
    FEATURE: 'feature:toggle:confirmed',
    SUBSCRIPTION: 'subscription:toggle:confirmed'
};

// Generic toggle handler
async function handleToggle(element, eventName) {
    return await handleToggleConfirmation({
        element,
        eventName,
        dialogHandler: Swal.fire,
        translator: (key) => window.translations[key] || key,
        onConfirm: async (data) => {
            console.log(`${eventName} confirmed:`, data);
        }
    });
}

// Initialize handlers for different toggle types
document.body.addEventListener('click', async (event) => {
    const accountToggle = event.target.closest('.btn-toggle-account');
    const featureToggle = event.target.closest('.btn-toggle-feature');
    const subscriptionToggle = event.target.closest('.btn-toggle-subscription');

    if (accountToggle) {
        event.preventDefault();
        await handleToggle(accountToggle, TOGGLE_EVENTS.ACCOUNT);
    } else if (featureToggle) {
        event.preventDefault();
        await handleToggle(featureToggle, TOGGLE_EVENTS.FEATURE);
    } else if (subscriptionToggle) {
        event.preventDefault();
        await handleToggle(subscriptionToggle, TOGGLE_EVENTS.SUBSCRIPTION);
    }
});

// Listen to all toggle events
Object.values(TOGGLE_EVENTS).forEach(eventName => {
    document.addEventListener(eventName, handleToggleEvent);
});

async function handleToggleEvent(event) {
```

```
    const { data, url_action_confirm, sourceElement } = event.detail;

    try {
        const response = await fetch(url_action_confirm, {
            method: 'PATCH',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(data)
        });

        if (response.ok) {
            const result = await response.json();
            updateUI(sourceElement, data.status);
            showNotification('success', result.message);
        }
    } catch (error) {
        showNotification('error', 'Operation failed');
    }
}

function updateUI(element, newStatus) {
    element.setAttribute('data-toggle-enabled', !newStatus);
    element.querySelector('i').classList.toggle('fa-check');
    element.querySelector('i').classList.toggle('fa-ban');
}

function showNotification(type, message) {
    Swal.fire({
        icon: type,
        title: type === 'success' ? 'Success' : 'Error',
        text: message,
        toast: true,
        position: 'top-end',
        showConfirmButton: false,
        timer: 3000
    });
}
```

---

jQuery Integration

Basic Setup

javascript

```
import { handleToggleConfirmation } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

jQuery(document).ready(function($) {
    initToggleHandlers();
    initToggleListener();
});

function initToggleHandlers() {
    // Use jQuery event delegation
    jQuery(document).on('click', '.btn-toggle-account', async function(event) {
        event.preventDefault();

        const $button = jQuery(this);

        await handleToggleConfirmation({
            element: this,
            eventName: 'account:toggle:confirmed',
            dialogHandler: Swal.fire,
            translator: (key) => window.SonataTranslator?.trans(key) || key
        });
    });
}

function initToggleListener() {
    jQuery(document).on('account:toggle:confirmed', async function(event) {
        const detail = event.originalEvent.detail;

        try {
            const response = await jQuery.ajax({
                url: detail.url_action_confirm,
                method: 'POST',
                data: JSON.stringify(detail.data),
                contentType: 'application/json',
                dataType: 'json'
            });

            if (response.success) {
                Swal.fire('Success!', response.message, 'success');

                // Update button state
                const $button = jQuery(detail.sourceElement);
                $button.attr('data-toggle-enabled', !detail.data.status);
```

```
            setTimeout(() => location.reload(), 2000);
        }
    } catch (xhr) {
        console.error('Toggle failed:', xhr);
        Swal.fire('Error', xhr.responseJSON?.message || 'An error occurred', 'error');
    }
  });
}
```

jQuery Plugin Wrapper

javascript

```
// Create jQuery plugin for easy use
(function($) {
    $.fn.toggleConfirmation = function(options) {
        const settings = $.extend({
            eventName: 'toggle:confirmed',
            dialogHandler: Swal.fire,
            translator: null,
            onConfirm: null,
            onCancel: null,
            onError: null
        }, options);

        return this.each(function() {
            $(this).on('click', async function(event) {
                event.preventDefault();

                await handleToggleConfirmation({
                    element: this,
                    eventName: settings.eventName,
                    dialogHandler: settings.dialogHandler,
                    translator: settings.translator,
                    onConfirm: settings.onConfirm,
                    onCancel: settings.onCancel,
                    onError: settings.onError
                });
            });
        });
    };
})(jQuery);

// Usage
jQuery(document).ready(function($) {
    $('.btn-toggle-account').toggleConfirmation({
        eventName: 'account:toggle:confirmed',
        translator: (key) => window.SonataTranslator.trans(key),
        onConfirm: (data) => {
            console.log('Confirmed:', data);
        }
    });

    // Listen for event
    $(document).on('account:toggle:confirmed', async function(event) {
        const detail = event.originalEvent.detail;
        // Handle the toggle...
```

```
    });
  });
```

React Integration

Hook-Based Approach

typescript

```tsx
// useToggleConfirmation.ts
import { useState, useCallback } from 'react';
import { handleToggleConfirmation, ToggleConfirmationParams } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

export function useToggleConfirmation(eventName: string, options?: Partial<ToggleConfirmationParams>) {
  const [isProcessing, setIsProcessing] = useState(false);

  const confirmToggle = useCallback(async (element: HTMLElement) => {
    setIsProcessing(true);

    try {
      const confirmed = await handleToggleConfirmation({
        element,
        eventName,
        dialogHandler: Swal.fire,
        ...options
      });

      return confirmed;
    } finally {
      setIsProcessing(false);
    }
  }, [eventName, options]);

  return { confirmToggle, isProcessing };
}

// Component usage
import React, { useEffect } from 'react';
import { useToggleConfirmation } from './hooks/useToggleConfirmation';

interface User {
  id: number;
  name: string;
  enabled: boolean;
}

const UserListItem: React.FC<{ user: User; onToggle: () => void }> = ({ user, onToggle }) => {
  const { confirmToggle, isProcessing } = useToggleConfirmation('account:toggle:confirmed', {
    translator: (key) => t(key) // react-i18next
  });

  const handleClick = async (event: React.MouseEvent<HTMLAnchorElement>) => {
```

```
    event.preventDefault();

    const confirmed = await confirmToggle(event.currentTarget);

    if (confirmed) {
      onToggle();
    }
  };

  return (
    <div className="user-item">
      <span>{user.name}</span>
      <a
        href={`/api/users/${user.id}/toggle`}
        className="btn-toggle-account"
        data-action-confirm={`Toggle account for ${user.name}?`}
        data-toggle-enabled={user.enabled}
        title={user.enabled ? 'Deactivate Account' : 'Activate Account'}
        data-additional={JSON.stringify({ userId: user.id, userName: user.name })}
        onClick={handleClick}
        disabled={isProcessing}
      >
        {user.enabled ? 'Deactivate' : 'Activate'}
      </a>
    </div>
  );
};

export default UserListItem;
```

Context Provider Approach

typescript

```tsx
// ToggleContext.tsx
import React, { createContext, useContext, useEffect, useCallback } from 'react';

interface ToggleContextValue {
  listenToToggles: () => void;
}

const ToggleContext = createContext<ToggleContextValue | null>(null);

export const ToggleProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const listenToToggles = useCallback(() => {
    const handleToggleEvent = async (event: Event) => {
      const customEvent = event as CustomEvent;
      const { data, url_action_confirm } = customEvent.detail;

      try {
        const response = await fetch(url_action_confirm, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(data)
        });

        const result = await response.json();

        if (result.success) {
          // Refresh data or update state
          window.location.reload();
        }
      } catch (error) {
        console.error('Toggle failed:', error);
      }
    };

    document.addEventListener('account:toggle:confirmed', handleToggleEvent);

    return () => {
      document.removeEventListener('account:toggle:confirmed', handleToggleEvent);
    };
  }, []);

  useEffect(() => {
    const cleanup = listenToToggles();
    return cleanup;
  }, [listenToToggles]);
```

```
  return (
    <ToggleContext.Provider value={{ listenToToggles }}>
      {children}
    </ToggleContext.Provider>
  );
};

export const useToggle = () => {
  const context = useContext(ToggleContext);
  if (!context) {
    throw new Error('useToggle must be used within ToggleProvider');
  }
  return context;
};

// App.tsx
import { ToggleProvider } from './contexts/ToggleContext';

function App() {
  return (
    <ToggleProvider>
      <UserList />
    </ToggleProvider>
  );
}
```

Vue.js Integration

Composition API

vue

```
<!-- useToggleConfirmation.ts -->
<script setup lang="ts">
import { ref } from 'vue';
import { handleToggleConfirmation } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

export function useToggleConfirmation(eventName: string) {
    const isProcessing = ref(false);

    const confirmToggle = async (element: HTMLElement) => {
        isProcessing.value = true;

        try {
            const confirmed = await handleToggleConfirmation({
                element,
                eventName,
                dialogHandler: Swal.fire
            });

            return confirmed;
        } finally {
            isProcessing.value = false;
        }
    };

    return {
        confirmToggle,
        isProcessing
    };
}
</script>

<!-- UserListItem.vue -->
<template>
  <div class="user-item">
    <span>{{ user.name }}</span>
    <a
      :href="`/api/users/${user.id}/toggle`"
      class="btn-toggle-account"
      :data-action-confirm="`Toggle account for ${user.name}?`"
      :data-toggle-enabled="user.enabled"
      :title="user.enabled ? 'Deactivate Account' : 'Activate Account'"
      :data-additional="JSON.stringify({ userId: user.id })"
      @click.prevent="handleToggle"
```

```
      :disabled="isProcessing"
    >
      {{ user.enabled ? 'Deactivate' : 'Activate' }}
    </a>
  </div>
</template>

<script setup lang="ts">
import { useToggleConfirmation } from '@/composables/useToggleConfirmation';

interface User {
    id: number;
    name: string;
    enabled: boolean;
}

const props = defineProps<{
    user: User;
}>();

const emit = defineEmits<{
    toggle: [];
}>();

const { confirmToggle, isProcessing } = useToggleConfirmation('account:toggle:confirmed');

const handleToggle = async (event: Event) => {
    const confirmed = await confirmToggle(event.currentTarget as HTMLElement);

    if (confirmed) {
        emit('toggle');
    }
};
</script>

<style scoped>
.user-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    border-bottom: 1px solid #eee;
```

```
}
</style>
```

Global Event Listener Plugin

typescript

```
// plugins/toggleListener.ts
import type { App } from 'vue';

export default {
  install(app: App) {
    // Global toggle event listener
    const handleToggleEvent = async (event: Event) => {
      const customEvent = event as CustomEvent;
      const { data, url_action_confirm } = customEvent.detail;

      try {
        const response = await fetch(url_action_confirm, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(data)
        });

        const result = await response.json();

        if (result.success) {
          // Emit global event for components to react
          app.config.globalProperties.$emitter?.emit('toggle:success', result);
        }
      } catch (error) {
        console.error('Toggle failed:', error);
        app.config.globalProperties.$emitter?.emit('toggle:error', error);
      }
    };

    // Register listener on mount
    document.addEventListener('account:toggle:confirmed', handleToggleEvent);

    // Provide cleanup method
    app.provide('cleanupToggleListener', () => {
      document.removeEventListener('account:toggle:confirmed', handleToggleEvent);
    });
  }
};

// main.ts
import { createApp } from 'vue';
import App from './App.vue';
import toggleListener from './plugins/toggleListener';
```

```
const app = createApp(App);
app.use(toggleListener);
app.mount('#app');
```

Angular Integration

Service-Based Approach

typescript

```ts
// toggle-confirmation.service.ts
import { Injectable } from '@angular/core';
import { handleToggleConfirmation, ToggleConfirmationParams } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';
import { BehaviorSubject, Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ToggleConfirmationService {
  private processingSubject = new BehaviorSubject<boolean>(false);
  public isProcessing$: Observable<boolean> = this.processingSubject.asObservable();

  async confirmToggle(
    element: HTMLElement,
    eventName: string,
    options?: Partial<ToggleConfirmationParams>
  ): Promise<boolean> {
    this.processingSubject.next(true);

    try {
      const confirmed = await handleToggleConfirmation({
        element,
        eventName,
        dialogHandler: Swal.fire,
        ...options
      });

      return confirmed;
    } finally {
      this.processingSubject.next(false);
    }
  }
}

// toggle-listener.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ToggleListenerService {
```

```
  private toggleSuccessSubject = new Subject<any>();
  private toggleErrorSubject = new Subject<any>();

  public toggleSuccess$ = this.toggleSuccessSubject.asObservable();
  public toggleError$ = this.toggleErrorSubject.asObservable();

  constructor(private http: HttpClient) {
    this.initListener();
  }

  private initListener(): void {
    document.addEventListener('account:toggle:confirmed', async (event: Event) => {
      const customEvent = event as CustomEvent;
      const { data, url_action_confirm } = customEvent.detail;

      try {
        const result = await this.http.post(url_action_confirm, data).toPromise();
        this.toggleSuccessSubject.next(result);
      } catch (error) {
        this.toggleErrorSubject.next(error);
      }
    });
  }
}

// user-list.component.ts
import { Component, OnInit, OnDestroy } from '@angular/core';
import { ToggleConfirmationService } from './services/toggle-confirmation.service';
import { ToggleListenerService } from './services/toggle-listener.service';
import { Subject } from 'rxjs';
import { takeUntil } from 'rxjs/operators';

interface User {
  id: number;
  name: string;
  enabled: boolean;
}

@Component({
  selector: 'app-user-list',
  template: `
    <div class="user-list">
      <div *ngFor="let user of users" class="user-item">
        <span>{{ user.name }}</span>
```

```
            <a
              [href]="'/api/users/' + user.id + '/toggle'"
              class="btn-toggle-account"
              [attr.data-action-confirm]="'Toggle account for ' + user.name + '?'"
              [attr.data-toggle-enabled]="user.enabled"
              [attr.title]="user.enabled ? 'Deactivate Account' : 'Activate Account'"
              [attr.data-additional]="getAdditionalData(user)"
              (click)="handleToggle($event, user)"
              [disabled]="isProcessing$ | async"
            >
              {{ user.enabled ? 'Deactivate' : 'Activate' }}
            </a>
          </div>
        </div>
    `
})
export class UserListComponent implements OnInit, OnDestroy {
  users: User[] = [];
  isProcessing$ = this.toggleConfirmationService.isProcessing$;
  private destroy$ = new Subject<void>();

  constructor(
    private toggleConfirmationService: ToggleConfirmationService,
    private toggleListenerService: ToggleListenerService
  ) {}

  ngOnInit(): void {
    // Listen for toggle success
    this.toggleListenerService.toggleSuccess$
      .pipe(takeUntil(this.destroy$))
      .subscribe(result => {
        console.log('Toggle success:', result);
        this.loadUsers(); // Refresh list
      });

    // Listen for toggle error
    this.toggleListenerService.toggleError$
      .pipe(takeUntil(this.destroy$))
      .subscribe(error => {
        console.error('Toggle error:', error);
      });
  }

  async handleToggle(event: Event, user: User): Promise<void> {
```

```
    event.preventDefault();

    const confirmed = await this.toggleConfirmationService.confirmToggle(
      event.currentTarget as HTMLElement,
      'account:toggle:confirmed'
    );

    if (confirmed) {
      console.log('User confirmed toggle for:', user.name);
    }
  }

  getAdditionalData(user: User): string {
    return JSON.stringify({ userId: user.id, userName: user.name });
  }

  loadUsers(): void {
    // Load users from API
  }

  ngOnDestroy(): void {
    this.destroy$.next();
    this.destroy$.complete();
  }
}
```

Directive Approach

typescript

```
// toggle-confirmation.directive.ts
import { Directive, ElementRef, HostListener, Input, Output, EventEmitter } from '@angular/core';
import { ToggleConfirmationService } from './services/toggle-confirmation.service';

@Directive({
  selector: '[appToggleConfirmation]'
})
export class ToggleConfirmationDirective {
  @Input() eventName = 'account:toggle:confirmed';
  @Output() confirmed = new EventEmitter<void>();
  @Output() cancelled = new EventEmitter<void>();

  constructor(
    private el: ElementRef<HTMLElement>,
    private toggleConfirmationService: ToggleConfirmationService
  ) {}

  @HostListener('click', ['$event'])
  async onClick(event: Event): Promise<void> {
    event.preventDefault();

    const confirmed = await this.toggleConfirmationService.confirmToggle(
      this.el.nativeElement,
      this.eventName
    );

    if (confirmed) {
      this.confirmed.emit();
    } else {
      this.cancelled.emit();
    }
  }
}

// Usage in template
/*
<a appToggleConfirmation
  [eventName]="'account:toggle:confirmed'"
  (confirmed)="onToggleConfirmed()"
  (cancelled)="onToggleCancelled()"
  href="/api/users/123/toggle"
  data-action-confirm="Toggle this account?"
  data-toggle-enabled="true"
  title="Toggle Account">
```

```
    Toggle
</a>
*/
```

Custom Events

Event Detail Structure

typescript

```typescript
interface ToggleEventDetail {
    data: {
        status: boolean;        // New toggle state
        [key: string]: unknown; // Additional data from data-additional attribute
    };
    url_action_confirm: string; // Action URL
    sourceElement: HTMLElement; // Source DOM element
    timestamp: string;          // ISO 8601 timestamp
}
```

Listening to Events

typescript

```typescript
// Basic listener
document.addEventListener('account:toggle:confirmed', (event: CustomEvent) => {
    console.log('Event detail:', event.detail);
    // {
    //     data: { status: false, userId: 123, userName: "john.doe" },
    //     url_action_confirm: "/admin/users/123/toggle",
    //     sourceElement: <HTMLElement>,
    //     timestamp: "2024-01-15T10:30:00.000Z"
    // }
});

// With TypeScript typing
document.addEventListener('account:toggle:confirmed', (event: Event) => {
    const customEvent = event as CustomEvent<ToggleEventDetail>;
    const { data, url_action_confirm, sourceElement, timestamp } = customEvent.detail;

    console.log('Status:', data.status);
    console.log('URL:', url_action_confirm);
    console.log('Timestamp:', timestamp);
});
```

Multiple Event Types

typescript

```typescript
// Define event constants
const TOGGLE_EVENTS = {
  ACCOUNT: 'account:toggle:confirmed',
  FEATURE: 'feature:toggle:confirmed',
  PERMISSION: 'permission:toggle:confirmed'
} as const;

// Generic listener
function createToggleListener(eventName: string, handler: (detail: ToggleEventDetail) => void) {
  document.addEventListener(eventName, (event: Event) => {
    const customEvent = event as CustomEvent<ToggleEventDetail>;
    handler(customEvent.detail);
  });
}

// Register listeners
createToggleListener(TOGGLE_EVENTS.ACCOUNT, async (detail) => {
  await handleAccountToggle(detail);
});

createToggleListener(TOGGLE_EVENTS.FEATURE, async (detail) => {
  await handleFeatureToggle(detail);
});

createToggleListener(TOGGLE_EVENTS.PERMISSION, async (detail) => {
  await handlePermissionToggle(detail);
});
```

Error Handling

Error Classes

MissingAttributeError

Thrown when required DOM attributes are missing.

typescript

```
try {
  const data = extractToggleData(element);
} catch (error) {
  if (error instanceof MissingAttributeError) {
    console.error(`Missing attribute: ${error.attribute}`);
    console.error(`Element HTML: ${error.elementHTML}`);

    // Show user-friendly message
    alert('This button is not properly configured. Please contact support.');
  }
}
```

ToggleConfirmationError

Thrown when confirmation handling fails.

typescript

```
try {
  await handleToggleConfirmation({
    element: buttonElement,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire
  });
} catch (error) {
  if (error instanceof ToggleConfirmationError) {
    console.error('Confirmation failed:', error.message);
    if (error.cause) {
      console.error('Caused by:', error.cause);
    }
  }
}
```

Comprehensive Error Handling

typescript

```
async function safeToggleConfirmation(element: HTMLElement, eventName: string) {
  try {
    // Validate element first
    if (!hasRequiredToggleAttributes(element)) {
      throw new Error('Element missing required attributes');
    }

    // Attempt confirmation
    const confirmed = await handleToggleConfirmation({
      element,
      eventName,
      dialogHandler: Swal.fire,

      onError: (error) => {
        // Log error for debugging
        console.error('Toggle confirmation error:', error);

        // Show user-friendly message
        Swal.fire({
          icon: 'error',
          title: 'Configuration Error',
          text: 'This action cannot be performed. Please contact support.',
          footer: `Error: ${error.message}`
        });
      }
    });

    return confirmed;

  } catch (error) {
    if (error instanceof MissingAttributeError) {
      console.error('Missing attribute:', error.attribute);
      alert(`Button configuration error: Missing ${error.attribute} attribute`);
    } else if (error instanceof ToggleConfirmationError) {
      console.error('Toggle confirmation failed:', error.message);
      alert('Failed to show confirmation dialog. Please try again.');
    } else {
      console.error('Unexpected error:', error);
      alert('An unexpected error occurred. Please refresh the page.');
    }

    return false;
```

```
    }
}
```

Best Practices

1. Always Use Event Delegation

typescript

```typescript
// ✅ Good - Handles dynamic elements
document.body.addEventListener('click', async (event) => {
    const target = event.target.closest('.btn-toggle-account');
    if (target) {
        event.preventDefault();
        await handleToggleConfirmation({...});
    }
});

// ❌ Bad - Won't work with dynamic elements
document.querySelectorAll('.btn-toggle-account').forEach(button => {
    button.addEventListener('click', async (event) => {
        // This won't work for dynamically added buttons
    });
});
```

2. Validate Attributes Before Use

typescript

```typescript
// ✅ Good - Check first
if (hasRequiredToggleAttributes(element)) {
    await handleToggleConfirmation({...});
} else {
    console.error('Invalid toggle button configuration');
}

// ❌ Bad - Let it throw
await handleToggleConfirmation({...}); // May throw MissingAttributeError
```

3. Use Typed Event Listeners

typescript

```typescript
// ✅ Good - Typed event detail
document.addEventListener('account:toggle:confirmed', (event: Event) => {
    const customEvent = event as CustomEvent<ToggleEventDetail>;
    const { data, url_action_confirm } = customEvent.detail;
    // TypeScript knows the structure
});

// ❌ Bad - Untyped
document.addEventListener('account:toggle:confirmed', (event: any) => {
    const detail = event.detail; // No type safety
});
```

4. Provide Translation Function

typescript

```typescript
// ✅ Good - User sees localized messages
await handleToggleConfirmation({
    element,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire,
    translator: (key) => translations[key] || key
});

// ❌ Bad - English only
await handleToggleConfirmation({
    element,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire
    // No translator - buttons show "Confirm"/"Cancel" in English only
});
```

5. Handle Callbacks Appropriately

typescript

```
// ✅ Good - Use callbacks for side effects
await handleToggleConfirmation({
    element,
    eventName: 'toggle:confirmed',
    dialogHandler: Swal.fire,

    onConfirm: async (data) => {
        // Track analytics
        analytics.track('account_toggle_confirmed', data);

        // Log for debugging
        console.log('User confirmed:', data);
    },

    onCancel: () => {
        analytics.track('account_toggle_cancelled');
    },

    onError: (error) => {
        // Send to error tracking service
        errorTracker.captureException(error);
    }
});
```

TypeScript Types

Core Types

typescript

```typescript
// Extracted toggle data
interface ExtractedToggleData {
    title: string;
    actionConfirmText: string;
    toggleEnabled: boolean;
    actionUrl: string;
    additionalData: Record<string, unknown>;
}

// Toggle confirmation parameters
interface ToggleConfirmationParams {
    element: HTMLElement;
    eventName: string;
    dialogHandler: (options: SweetAlertOptions) => Promise<SweetAlertResult>;
    translator?: ((key: string) => string) | null;
    confirmDialogConfig?: Partial<SweetAlertOptions>;
    cancelDialogConfig?: Partial<SweetAlertOptions>;
    onConfirm?: ((data: ExtractedToggleData, event: CustomEvent) => void | Promise<void>) | null;
    onCancel?: ((data: ExtractedToggleData) => void | Promise<void>) | null;
    onError?: ((error: Error) => void) | null;
}

// Custom event detail
interface ToggleEventDetail {
    data: {
        status: boolean;
        [key: string]: unknown;
    };
    url_action_confirm: string;
    sourceElement: HTMLElement;
    timestamp: string;
}
```

Troubleshooting

Issue: "jQuery must be globally available"

Problem: jQuery is not loaded or not available as `window.jQuery`

Solution:

```
html
```

```
<!-- Load jQuery before your script -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="your-app.js"></script>
```

Issue: MissingAttributeError thrown

Problem: Required attributes missing from DOM element

Solution:


html

```
<!-- Ensure all required attributes are present -->
<button
  data-action-confirm="Confirmation message"  ✔
  data-toggle-enabled="true"                  ✔
  title="Button title"                        ✔
  data-url="/api/toggle"                      ✔
>
  Toggle
</button>
```

Issue: Event not firing

Problem: Event listener not registered or wrong event name

Solution:


typescript

```
// Make sure event name matches
await handleToggleConfirmation({
  eventName: 'account:toggle:confirmed'  // Must match
});

document.addEventListener('account:toggle:confirmed', ...); // Same name
```

Issue: SweetAlert2 not styled

Problem: SweetAlert2 CSS not loaded

Solution:

html

```
<!-- Load SweetAlert2 CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/sweetalert2@11/dist/sweetalert2.min.css">
```

License

Credits

Author: AGBOKOUDJO Franck
Email: internationaleswebservices@gmail.com
Company: INTERNATIONALES WEB APPS & SERVICES
Phone: +229 0167 25 18 86
LinkedIn: View Profile
GitHub: https://github.com/Agbokoudjo/
Package: @wlindabla/form_validator

Quick Reference

typescript

```
// Import
import { handleToggleConfirmation } from '@wlindabla/form_validator';
import Swal from 'sweetalert2';

// HTML
<button
    data-action-confirm="Confirmation message"
    data-toggle-enabled="true"
    title="Button Title"
    data-url="/api/toggle">
    Toggle
</button>

// Handle click
button.addEventListener('click', async (e) => {
    e.preventDefault();
    await handleToggleConfirmation({
        element: e.currentTarget,
        eventName: 'toggle:confirmed',
        dialogHandler: Swal.fire
    });
});

// Listen to event
document.addEventListener('toggle:confirmed', (e) => {
    const { data, url_action_confirm } = e.detail;
    // Execute HTTP request
});
```

Made with ❤ by AGBOKOUDJO Franck

Last Updated: 2024

⬆ Back to Top