@wlindabla/form_validator - FormFormattingEvent API Documentation

Overview

`FormFormattingEvent` is a singleton class from the **@wlindabla/form_validator** library that provides real-time formatting and validation utilities for form input fields. It handles automatic transformation of user input including uppercase conversion, word capitalization, and comprehensive username formatting with locale support.

Author: AGBOKOUDJO Franck
Company: INTERNATIONALES WEB APP & SERVICES
Contact: internationaleswebservices@gmail.com | +229 0167 25 18 86

Table of Contents

Features

- **Singleton Pattern**: Single instance ensures consistent behavior across the application
- **Real-time Formatting**: Automatic transformation on `blur` events
- **Locale Support**: Customizable locale settings for text transformations
- **Event Delegation**: Efficient event handling using jQuery delegation
- **HTML Sanitization**: Built-in protection against HTML injection
- **Default Export**: Pre-instantiated formatter available for immediate use
- **Comprehensive Logging**: Detailed logging for debugging and monitoring

---

Installation

Prerequisites

- jQuery 3.0+
- TypeScript 4.0+ (or JavaScript ES6+)

Via npm

```bash
npm install @wlindabla/form_validator
```

Via yarn

```bash
yarn add @wlindabla/form_validator
```

Getting Started

Import Methods

Method 1: Using the Default Export

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

// Apply formatting directly to your form
const form = document.getElementById("myForm");
formatterEvent.init({ locales: "en-US" });
formatterEvent.lastnameToUpperCase(form);
formatterEvent.capitalizeUsername(form);
formatterEvent.usernameFormatDom(form);
```

Method 2: Using the Class

```typescript
import { FormFormattingEvent } from "@wlindabla/form_validator";

const formatter = FormFormattingEvent.getInstance();
formatter.init({ locales: "en-US" });
```

Method 3: Named Import

```typescript
import { FormFormattingEvent, formatterEvent } from "@wlindabla/form_validator";

// Use pre-instantiated formatter
formatterEvent.init({ locales: "fr-FR" });
```

Quick Setup

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

const form = document.querySelector("form");

formatterEvent
    .init({ locales: "en-US" })
    .lastnameToUpperCase(form)
    .capitalizeUsername(form)
    .usernameFormatDom(form);
```

API Reference

`getInstance(): FormFormattingEvent`

Returns the singleton instance of `FormFormattingEvent`.

Returns: `FormFormattingEvent` - The singleton instance

Example:

```typescript
import { FormFormattingEvent } from "@wlindabla/form_validator";

const formatter = FormFormattingEvent.getInstance();
```

`init(options?: OptionsFormattingEvent): this`

Initializes global formatting options for the event manager. This method supports method chaining.

Parameters:

Parameter	Type	Default	Description
`options`	`OptionsFormattingEvent`	`{}`	Configuration object for formatting events

Options Interface:

```typescript
interface OptionsFormattingEvent {
  locales?: string | string[];  // Locale code(s) for text formatting
}
```

Returns: `this` - The current instance for method chaining

Example:

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

// Single locale
formatterEvent.init({ locales: "en-US" });

// Multiple locales (fallback)
formatterEvent.init({ locales: ["fr-FR", "fr"] });

// Method chaining
formatterEvent
  .init({ locales: "en-US" })
  .lastnameToUpperCase(document);
```

`lastnameToUpperCase(subject, locales?): void`

Converts last name input field values to uppercase in real-time when users leave the field.

Parameters:

Parameter	Type	Required	Description
`subject`	`HTMLElement | Document | JQuery`	Yes	The DOM context to search within
`locales`	`string | string[]`	No	Locale settings for uppercase transformation (overrides init locale)

Target Selector: `input.lastname`

Event Trigger: `blur` event

Behavior:

- Listens for `blur` events on `input.lastname` elements

- Trims whitespace and sanitizes HTML using `escapeHtmlBalise()`
- Applies locale-specific uppercase transformation
- Skips empty fields silently
- Logs all formatting operations

Throws: `Error` if the `.lastname` input field is not found

Example:

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

const formContainer = document.getElementById("userForm");

// Use default locale from init
formatterEvent.lastnameToUpperCase(formContainer);

// Override with custom locale
formatterEvent.lastnameToUpperCase(formContainer, "tr-TR"); // Turkish

// Multiple locales
formatterEvent.lastnameToUpperCase(formContainer, ["en-US", "en"]);
```

HTML Template:

```html
<form id="userForm">
  <div class="form-group">
    <label for="lastname">Last Name</label>
    <input type="text" id="lastname" class="lastname" placeholder="Enter your last name">
  </div>
</form>
```

Transformation Examples:

Input	Output
`martin`	`MARTIN`
`garcía`	`GARCÍA`
`müller`	`MÜLLER`

---

capitalizeUsername(subject, separator_toString?, finale_separator_toString?, locales?): void

Capitalizes each word in the first name input field when users leave the field.

Parameters:

Parameter	Type	Default	Description
subject	HTMLElement | Document | JQuery	—	The DOM context containing the input field
separator_toString	string	" "	Word separator in the input value
finale_separator_toString	string	" "	Word separator used after formatting
locales	string | string[]	—	Locale(s) for capitalization (overrides init locale)

Target Selector: input.firstname

Event Trigger: blur event

Behavior:

- Listens for blur events on input.firstname elements
- Capitalizes the first letter of each word
- Converts remaining letters to lowercase
- Supports custom separators (spaces, hyphens, underscores, etc.)
- Handles multi-word names with different separators

Example:

typescript

```
import { formatterEvent } from "@wlindabla/form_validator";

const formContainer = document.getElementById("userForm");

// Standard space-separated words
formatterEvent.capitalizeUsername(formContainer);

// Hyphenated names
formatterEvent.capitalizeUsername(
    formContainer,
    "-",      // Split on hyphens
    "-",      // Join with hyphens
    "en-US"
);

// Multiple word separators
formatterEvent.capitalizeUsername(
    formContainer,
    " ",      // Split on spaces
    " ",      // Join with spaces
    "fr-FR"   // French locale
);
```

Transformation Examples:

Input	Output
`john doe`	`John Doe`
`MARIE-LOUISE`	`Marie-Louise`
`pierre BERNARD`	`Pierre Bernard`
`jean-paul martin`	`Jean-Paul Martin`

HTML Template:

```html
<input type="text" class="firstname" placeholder="Enter your first name">
```

`usernameFormatDom(subject, separator_toString?, finale_separator_toString?, locales?): void`

Automatically formats username fields by applying comprehensive formatting rules for full names.

Parameters:

Parameter	Type	Default	Description
subject	HTMLElement | Document | JQuery	—	The DOM context containing the input
separator_toString	string	" "	Word separator in the input
finale_separator_toString	string	" "	Word separator after processing
locales	string | string[]	—	Locale(s) for formatting (overrides init locale)

Target Selector: `input.username`

Custom Attributes:

- `data-position-lastname`: Position of the last name (`"left"` or `"right"`, default: `"left"`)

Event Trigger: `blur` event

Behavior:

- Listens for `blur` events on `input.username` elements
- Applies comprehensive username formatting via `usernameFormat()` utility
- Respects last name position based on `data-position-lastname` attribute
- Supports custom word separators
- Returns formatted full name with proper capitalization

Example:

```typescript
```

```
import { formatterEvent } from "@wlindabla/form_validator";

const formContainer = document.getElementById("userForm");

// Standard formatting
formatterEvent.usernameFormatDom(formContainer);

// With custom separators
formatterEvent.usernameFormatDom(
  formContainer,
  " ",      // Split on spaces
  "-",      // Join with hyphens
  "en-US"
);

// With specific locale
formatterEvent.usernameFormatDom(formContainer, " ", " ", "fr-FR");
```

HTML Templates:

```html
<!-- Last name on the left -->
<input
  type="text"
  class="username"
  data-position-lastname="left"
  placeholder="Enter full name"
>

<!-- Last name on the right -->
<input
  type="text"
  class="username"
  data-position-lastname="right"
  placeholder="Enter full name"
>

<!-- Default (left) -->
<input
  type="text"
  class="username"
  placeholder="Enter full name"
>
```

Transformation Examples:

Input	Output	Position
`john martin`	`John Martin`	left
`martin john`	`John Martin`	right
`marie-louise dupont`	`Marie-Louise Dupont`	left

Usage Examples

Example 1: Basic Form Setup

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

// Initialize formatter
formatterEvent.init({ locales: "en-US" });

// Target the form
const form = document.getElementById("contactForm");

// Apply all formatters
formatterEvent.lastnameToUpperCase(form);
formatterEvent.capitalizeUsername(form);
formatterEvent.usernameFormatDom(form);
```

HTML:

```html
<form id="contactForm">
  <input type="text" class="firstname" placeholder="First Name">
  <input type="text" class="lastname" placeholder="Last Name">
  <input type="text" class="username" placeholder="Full Name">
  <button type="submit">Submit</button>
</form>
```

Example 2: Multi-Locale Support

```typescript
```

```
import { formatterEvent } from "@wlindabla/form_validator";

// Initialize with multiple locales
formatterEvent.init({ locales: ["fr-FR", "en-US"] });

const form = document.getElementById("intlForm");

// French formatting
formatterEvent.lastnameToUpperCase(form, "fr-FR");
formatterEvent.capitalizeUsername(form, " ", " ", "fr-FR");

// English formatting for username
formatterEvent.usernameFormatDom(form, " ", " ", "en-US");
```

Example 3: Dynamic Forms

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

formatterEvent.init({ locales: "en-US" });

// Use document for event delegation - works with dynamically added elements
formatterEvent.lastnameToUpperCase(document);
formatterEvent.capitalizeUsername(document);
formatterEvent.usernameFormatDom(document);

// Later, when adding new form elements dynamically
const newForm = document.createElement("form");
newForm.innerHTML = `
    <input type="text" class="firstname">
    <input type="text" class="lastname">
    <input type="text" class="username">
`;
document.body.appendChild(newForm);
// Formatting automatically applies to new elements!
```

Example 4: Selective Field Formatting

```typescript
```

```
import { formatterEvent } from "@wlindabla/form_validator";

formatterEvent.init({ locales: "en-US" });

// Format different sections separately
const profileSection = document.querySelector(".profile-section");
const registrationSection = document.querySelector(".registration-section");

// Profile uses standard formatting
formatterEvent.capitalizeUsername(profileSection);
formatterEvent.lastnameToUpperCase(profileSection);

// Registration uses custom separators
formatterEvent.usernameFormatDom(registrationSection, " ", "-", "en-US");
```

Example 5: Method Chaining

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

const form = document.getElementById("myForm");

// Chain method calls for clean code
formatterEvent
    .init({ locales: "en-US" })
    .lastnameToUpperCase(form)
    .capitalizeUsername(form)
    .usernameFormatDom(form);
```

Example 6: Complete Registration Form

```html
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
    <style>
        .form-group {
            margin-bottom: 1.5rem;
        }
        input {
            width: 100%;
            padding: 0.5rem;
            font-size: 1rem;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <form id="registrationForm">
        <div class="form-group">
            <label for="firstname">First Name *</label>
            <input
                type="text"
                id="firstname"
                class="firstname"
                required
            >
        </div>

        <div class="form-group">
            <label for="lastname">Last Name *</label>
            <input
                type="text"
                id="lastname"
                class="lastname"
                required
            >
        </div>

        <div class="form-group">
            <label for="username">Full Name</label>
            <input
                type="text"
                id="username"
                class="username"
                data-position-lastname="right"
```

```
      >
    </div>

    <button type="submit">Register</button>
  </form>

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script type="module">
    import { formatterEvent } from "@wlindabla/form_validator";

    // Initialize and apply formatting
    formatterEvent
      .init({ locales: "en-US" })
      .lastnameToUpperCase(document)
      .capitalizeUsername(document)
      .usernameFormatDom(document);
  </script>
</body>
</html>
```

Locale Support

The class supports any locale string recognized by JavaScript's `toLocaleUpperCase()` and `toLocaleLowerCase()` methods.

Common Locale Codes

Locale	Language	Region	Example
`en`	English	General	—
`en-US`	English	United States	"Müller" → "MÜLLER"
`en-GB`	English	Great Britain	—
`fr`	French	General	—
`fr-FR`	French	France	"café" → "CAFÉ"
`de-DE`	German	Germany	"größe" → "GRÖSSZE"
`es-ES`	Spanish	Spain	"niño" → "NIÑO"
`it-IT`	Italian	Italy	—
`tr-TR`	Turkish	Turkey	"ı" → "I", "i" → "İ"
`ru-RU`	Russian	Russia	—
`ja-JP`	Japanese	Japan	—
`zh-CN`	Chinese	Mainland China	—

Locale-Specific Behavior

Some locales have special character handling:

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

const form = document.getElementById("myForm");

// Turkish has special case mapping
formatterEvent.lastnameToUpperCase(form, "tr-TR");
// "ı" (lowercase dotless i) → "I"
// "i" (lowercase i with dot) → "İ"

// German uses special character rules
formatterEvent.lastnameToUpperCase(form, "de-DE");
// "ß" (sharp s) → "SS"

// French preserves accented characters
formatterEvent.lastnameToUpperCase(form, "fr-FR");
// "café" → "CAFÉ"
```

Best Practices

1. Initialize Once at Startup

Initialize the formatter once when your application loads:

```typescript
// app.ts or main.ts
import { formatterEvent } from "@wlindabla/form_validator";

// Global initialization
formatterEvent.init({ locales: "en-US" });
```

2. Use Semantic HTML Class Names

Maintain consistent class naming for form fields:

```html
<!-- Clear, semantic naming -->
<input class="firstname" />  <!-- For first name -->
<input class="lastname" />   <!-- For last name -->
<input class="username" />   <!-- For full name -->
```

3. Leverage Data Attributes for Configuration

Use `data-position-lastname` to control name ordering:

```html
<!-- Names in first-last order (default) -->
<input class="username" data-position-lastname="left">

<!-- Names in last-first order -->
<input class="username" data-position-lastname="right">
```

4. Use Event Delegation for Dynamic Content

Apply formatters to `document` to handle dynamically added elements:

```typescript
import { formatterEvent } from "@wlindabla/form_validator";

// Works with elements added to DOM later
formatterEvent.lastnameToUpperCase(document);
formatterEvent.capitalizeUsername(document);
formatterEvent.usernameFormatDom(document);
```

5. Take Advantage of Method Chaining

Chain methods for cleaner, more readable code:

```typescript
formatterEvent
  .init({ locales: "en-US" })
  .lastnameToUpperCase(form)
  .capitalizeUsername(form)
  .usernameFormatDom(form);
```

6. Handle Errors Gracefully

Use try-catch when working with dynamic DOM elements:

```typescript
```

```
try {
  const form = document.getElementById("dynamicForm");
  if (form) {
    formatterEvent.lastnameToUpperCase(form);
  }
} catch (error) {
  console.error("Formatting initialization failed:", error);
}
```

7. Document Custom Separators

When using custom separators, document the expected format:

```typescript
// Hyphenated first names (e.g., "Jean-Paul")
formatterEvent.capitalizeUsername(
  form,
  "-",  // input separator
  "-",  // output separator
  "fr-FR"
);
```

Troubleshooting

Issue: Formatting not being applied

Cause: Input field has incorrect class name or is not within the specified container

Solution:

```typescript
// Verify the element exists
console.log(document.querySelector("input.lastname"));

// Use document if unsure about container
formatterEvent.lastnameToUpperCase(document);
```

Issue: Locale not working as expected

Cause: Invalid locale string or unsupported browser

Solution:

```typescript
```

```
// Test locale support
console.log("é".toLocaleUpperCase("fr-FR")); // Should output "É"

// Use fallback locale
formatterEvent.init({ locales: ["fr-FR", "en"] });
```

Issue: jQuery not found

Cause: jQuery library not loaded before FormFormattingEvent

Solution:

```html
<!-- Load jQuery BEFORE your script -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script type="module">
  import { formatterEvent } from "@wlindabla/form_validator";
  // Now safe to use
</script>
```

Issue: Multiple event listeners being attached

Cause: Calling formatter method multiple times on same element

Solution:

```typescript
// The library handles this - it removes old listeners with .off()
// before attaching new ones, so multiple calls are safe
formatterEvent.lastnameToUpperCase(form);
formatterEvent.lastnameToUpperCase(form); // Safe - old listener removed first
```

Issue: Empty fields being processed

Cause: Expected behavior - empty fields are logged but not processed

Solution:

```typescript
// This is normal - empty fields are skipped with a log message:
// "The last name input field is empty, uppercase formatting ignored."

// No special handling needed
```

Package Information

- **Package Name:** @wlindabla/form_validator
- **Author:** AGBOKOUDJO Franck
- **Company:** INTERNATIONALES WEB APP & SERVICES
- **License:** See package.json

Support & Contact

For support, feature requests, or bug reports:

- **Email:** internationaleswebservices@gmail.com
- **Phone:** +229 0167 25 18 86
- **LinkedIn:** https://www.linkedin.com/in/internationales-web-apps-services-120520193/
- **Company:** INTERNATIONALES WEB APP & SERVICES

Version History

Version	Date	Changes
1.0.0	2024	Initial release with default export
0.9.0	2024	Beta release

Related Documentation

- jQuery Event Handling
- JavaScript Locale Strings
- HTML Form Elements

Last Updated: 2024
Documentation Version: 1.0.0
Status: Production Ready ✅