FetchErrorTranslator - Fetch API Error Translation System

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

Enterprise-grade translation system for Fetch API errors with intelligent pattern matching, multi-language support, and configurable caching. Convert cryptic error messages into user-friendly, localized text.

Table of Contents

Reference

- Default Error Messages
- TypeScript Types
- Best Practices
- Troubleshooting

Installation

NPM / Yarn

bash

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

Import

typescript

```typescript
// Import singleton instance (recommended)
import { fetchErrorTranslator } from '@wlindabla/form_validator';

// Or import class
import { FetchErrorTranslator } from '@wlindabla/form_validator';

// Create custom instance
const translator = new FetchErrorTranslator({
    defaultLanguage: 'fr',
    debug: true
});
```

Basic Usage

Simple Translation

typescript

```typescript
import { fetchErrorTranslator } from '@wlindabla/form_validator';

try {
    const response = await fetch('/api/data');
    if (!response.ok) throw new Error('HTTP Error');
} catch (error) {
    // Translate error to user-friendly message
    const message = fetchErrorTranslator.translate(error.name, error);
    console.error(message);
    // "Network error - Check your internet connection"
}
```

With Specific Language

typescript

```typescript
try {
    await fetch('/api/data');
} catch (error) {
    // English
    const enMessage = fetchErrorTranslator.translate(error.name, error, 'en');
    console.log(enMessage); // "Network error - Check your internet connection"

    // French
    const frMessage = fetchErrorTranslator.translate(error.name, error, 'fr');
    console.log(frMessage); // "Erreur réseau - Vérifiez votre connexion internet"

    // Spanish
    const esMessage = fetchErrorTranslator.translate(error.name, error, 'es');
    console.log(esMessage); // "Error de red - Verifique su conexión a internet"
}
```

Complete Example

typescript

```typescript
import { fetchErrorTranslator } from '@wlindabla/form_validator';

async function fetchData(url: string) {
    try {
        const controller = new AbortController();
        const timeoutId = setTimeout(() => controller.abort(), 5000);

        const response = await fetch(url, {
            signal: controller.signal
        });

        clearTimeout(timeoutId);

        if (!response.ok) {
            throw new Error(`HTTP ${response.status}`);
        }

        return await response.json();

    } catch (error) {
        // Translate error
        const userMessage = fetchErrorTranslator.translate(
            error.name,
            error
        );

        // Show to user
        alert(userMessage);

        // Log original error for debugging
        console.error('Original error:', error);

        throw error;
    }
}
```

Why FetchErrorTranslator?

✖ Without FetchErrorTranslator

typescript

```
try {
  await fetch('/api/data');
} catch (error) {
  console.error(error.message);
  // "Failed to fetch" - Not helpful for users!
  // "NetworkError" - Technical jargon
  // "TypeError" - Meaningless to end users
}
```

✔ With FetchErrorTranslator

typescript

```
try {
  await fetch('/api/data');
} catch (error) {
  const message = fetchErrorTranslator.translate(error.name, error);
  console.error(message);
  // "Network error - Check your internet connection" - Clear and actionable!
}
```

✨ Key Features

✔ **User-Friendly Messages** - Convert technical errors to plain language
✔ **Multi-Language Support** - Built-in support for en, fr, es (extensible)
✔ **Intelligent Matching** - Pattern matching for unknown errors
✔ **Caching** - Configurable cache adapters for performance
✔ **Type-Safe** - Full TypeScript support
✔ **Zero Dependencies** - Only requires the cache adapter module
✔ **Extensible** - Easy to add new languages and error types
✔ **Production Ready** - Battle-tested in enterprise applications

---

How It Works

Translation Process

1. Error occurs in fetch()
   ↓
2. Call translate(error.name, error)
   ↓
3. Try exact match for error.name
   ├─ MATCH → Return translation
   │
   └─ NO MATCH → Pattern matching
      ├─ Check keywords in error.message
      ├─ Check SSL/TLS patterns
      └─ Return fallback message

Example Flow

typescript

```typescript
// 1. Network error occurs
fetch('/api/data') // → TypeError: Failed to fetch

// 2. Translate the error
const message = fetchErrorTranslator.translate('TypeError', error);

// 3. Exact match found
// Returns: "Network error - Check your internet connection"

// 4. If no exact match, pattern matching kicks in
// Checks error.message for keywords: 'network', 'failed to fetch', etc.
// Returns appropriate translation
```

Error Pattern Matching

How Pattern Matching Works

When an exact error name match isn't found, the translator analyzes the error message for keywords:

typescript

```
const patterns = [
    {
        keywords: ['network', 'failed to fetch'],
        errorKey: 'NetworkError'
    },
    {
        keywords: ['timeout', 'abort'],
        errorKey: 'AbortError'
    },
    {
        keywords: ['cors'],
        errorKey: 'SecurityError'
    },
    {
        keywords: ['dns', 'not found'],
        errorKey: 'NotFoundError'
    }
];
```

Example

typescript

```
// Error with unclear name
const error = new Error('Failed to fetch due to network issues');
error.name = 'UnknownError';

// Pattern matching finds 'network' and 'failed to fetch'
const message = fetchErrorTranslator.translate(error.name, error);
// Returns: "Network error - Check your internet connection"
```

Supported Languages

Built-in Languages

Language	Code	Example Message
English	en	"Network error - Check your internet connection"
French	fr	"Erreur réseau - Vérifiez votre connexion internet"
Spanish	es	"Error de red - Verifique su conexión a internet"

Adding More Languages

typescript

```typescript
// Add German translations
fetchErrorTranslator.addTranslations('de', {
  'AbortError': 'Anfrage hat Zeitüberschreitung überschritten',
  'TypeError': 'Netzwerkfehler - Überprüfen Sie Ihre Internetverbindung',
  'NetworkError': 'Netzwerkfehler - Verbindung unmöglich',
  'SecurityError': 'Sicherheitsfehler - Zugriff verweigert (CORS)'
});

// Add Italian translations
fetchErrorTranslator.addTranslations('it', {
  'AbortError': 'Richiesta scaduta',
  'TypeError': 'Errore di rete - Controlla la tua connessione internet',
  'NetworkError': 'Errore di rete - Connessione impossibile'
});
```

API Documentation

FetchErrorTranslator Class

Main class for error translation.

Constructor

Creates a new FetchErrorTranslator instance.

typescript

```typescript
constructor(config?: FetchErrorTranslatorConfig)
```

Configuration Options:

typescript

```typescript
interface FetchErrorTranslatorConfig {
  defaultLanguage?: string;              // Default: 'en'
  cacheAdapter?: CacheTranslationInterface;  // Default: LocalStorage
  debug?: boolean;                       // Default: false
  customPatterns?: ErrorPattern[];       // Additional patterns
  additionalTranslations?: ErrorTranslations;  // Extra translations
}
```

Examples:

typescript

```typescript
// Default configuration
const translator = new FetchErrorTranslator();

// Custom configuration
const translator = new FetchErrorTranslator({
  defaultLanguage: 'fr',
  debug: true
});

// With custom patterns
const translator = new FetchErrorTranslator({
  customPatterns: [
    {
      keywords: ['gateway', '502', '503'],
      errorKey: 'ServerError'
    }
  ],
  additionalTranslations: {
    en: {
      'ServerError': 'Server is temporarily unavailable'
    },
    fr: {
      'ServerError': 'Le serveur est temporairement indisponible'
    }
  }
});
```

---

translate()

Translates an error name to a user-friendly message.

typescript

```
  translate(
    errorName: string,
    error?: Error | null,
    language?: string
  ): string
```

Parameters:

Parameter	Type	Required	Description
errorName	string	✅ Yes	Error name (e.g., 'AbortError')
error	Error | null	❌ No	Error object for pattern matching
language	string	❌ No	Language code (uses current if not specified)

Returns: `string` - Translated error message

Examples:

typescript

```
// Simple translation
const message = fetchErrorTranslator.translate('NetworkError');
console.log(message); // "Network error - Connection impossible"

// With error object for pattern matching
try {
    await fetch('/api/data');
} catch (error) {
    const message = fetchErrorTranslator.translate(error.name, error);
    alert(message);
}

// With specific language
const frenchMessage = fetchErrorTranslator.translate('AbortError', null, 'fr');
console.log(frenchMessage); // "La requête a expiré..."

// Complete example
async function handleFetch(url: string) {
    try {
        const response = await fetch(url);
        return await response.json();
    } catch (error) {
        const userMessage = fetchErrorTranslator.translate(
            error.name,
            error,
            document.documentElement.lang
        );

        // Show user-friendly message
        showErrorToast(userMessage);

        // Log technical details for debugging
        console.error('Fetch error:', error);

        throw error;
    }
}
```

---

addTranslations()

Adds or extends translations for a language.

✓

typescript

```
addTranslations(language: string, translations: TranslationMessages): void
```

Parameters:

Parameter	Type	Description
language	string	Language code (e.g., 'de', 'it')
translations	TranslationMessages	Error name to message mappings

Examples:

typescript

```
// Add German translations
fetchErrorTranslator.addTranslations('de', {
  'AbortError': 'Anfrage hat Zeitüberschreitung überschritten',
  'TypeError': 'Netzwerkfehler - Überprüfen Sie Ihre Internetverbindung',
  'NetworkError': 'Netzwerkfehler - Verbindung unmöglich',
  'SecurityError': 'Sicherheitsfehler - Zugriff verweigert (CORS)',
  'NotFoundError': 'Ressource nicht gefunden',
  'TimeoutError': 'Zeitüberschreitung',
  'InvalidStateError': 'Ungültige Anfrage - Ungültiger Status',
  'SyntaxError': 'Syntaxfehler in der Antwort',
  'ReferenceError': 'Referenzfehler',
  'RangeError': 'Wert außerhalb des Bereichs'
});

// Extend existing English translations
fetchErrorTranslator.addTranslations('en', {
  'CustomError': 'A custom error occurred',
  'ValidationError': 'Validation failed - Please check your input'
});

// Add Italian translations
fetchErrorTranslator.addTranslations('it', {
  'AbortError': 'Richiesta scaduta',
  'TypeError': 'Errore di rete',
  'NetworkError': 'Errore di rete - Connessione impossibile'
});
```

---

getSupportedLanguages()

Gets all supported language codes.

typescript

```
getSupportedLanguages(): string[]
```

Returns: `string[]` - Array of language codes

Example:

typescript

```
const languages = fetchErrorTranslator.getSupportedLanguages();
console.log(languages); // ['en', 'fr', 'es', 'de']

// Use in language selector
const languageOptions = fetchErrorTranslator.getSupportedLanguages().map(lang => ({
  value: lang,
  label: getLanguageName(lang)
}));
```

---

getSupportedErrorNames()

Gets all supported error names for a language.

typescript

```
getSupportedErrorNames(language?: string): string[]
```

Parameters:

Parameter	Type	Description
`language`	`string`	Optional language code (uses current if not specified)

Returns: `string[]` - Array of error names

Example:

typescript

```typescript
// Get error names for current language
const errorNames = fetchErrorTranslator.getSupportedErrorNames();
console.log(errorNames);
// ['AbortError', 'TypeError', 'NetworkError', 'SecurityError', ...]

// Get error names for specific language
const frenchErrors = fetchErrorTranslator.getSupportedErrorNames('fr');
console.log(frenchErrors);

// Debugging: Check if error type is supported
if (fetchErrorTranslator.getSupportedErrorNames().includes('CustomError')) {
  console.log('CustomError is supported');
}
```

---

hasTranslationFor()

Checks if a translation exists for an error name.

typescript

```typescript
hasTranslationFor(errorName: string, language?: string): boolean
```

Parameters:

Parameter	Type	Description
errorName	string	Error name to check
language	string	Optional language code

Returns: `boolean` - `true` if translation exists

Example:

typescript

```typescript
// Check if translation exists
if (fetchErrorTranslator.hasTranslationFor('AbortError')) {
  console.log('Translation available');
}

// Check for specific language
if (fetchErrorTranslator.hasTranslationFor('CustomError', 'fr')) {
  const message = fetchErrorTranslator.translate('CustomError', null, 'fr');
} else {
  console.warn('French translation not available for CustomError');
}

// Conditional translation
function getErrorMessage(errorName: string): string {
  if (fetchErrorTranslator.hasTranslationFor(errorName)) {
    return fetchErrorTranslator.translate(errorName);
  } else {
    return `An error occurred: ${errorName}`;
  }
}
```

---

exportTranslations()

Exports all translations for a language.

typescript

```typescript
exportTranslations(language: string): TranslationMessages
```

Parameters:

Parameter	Type	Description
`language`	`string`	`Language code to export`

Returns: `TranslationMessages` - Object with error name/message pairs

Example:

typescript

```typescript
// Export English translations
const enTranslations = fetchErrorTranslator.exportTranslations('en');
console.log(JSON.stringify(enTranslations, null, 2));

// Save to file
const json = JSON.stringify(enTranslations, null, 2);
downloadFile('error-translations-en.json', json);

// Send to server for backup
await fetch('/api/translations/backup', {
  method: 'POST',
  body: JSON.stringify({
    language: 'en',
    translations: fetchErrorTranslator.exportTranslations('en')
  })
});

// Debugging
console.table(fetchErrorTranslator.exportTranslations('fr'));
```

---

clearCache()

Clears all cached translations.

typescript

```typescript
async clearCache(): Promise<void>
```

Example:

typescript

```
// Clear cache manually
await fetchErrorTranslator.clearCache();
console.log('Cache cleared');

// Clear cache on app version change
const APP_VERSION = '2.0.0';
const cachedVersion = localStorage.getItem('app_version');

if (cachedVersion !== APP_VERSION) {
  await fetchErrorTranslator.clearCache();
  localStorage.setItem('app_version', APP_VERSION);
}

// Clear cache button
async function handleClearCache() {
  await fetchErrorTranslator.clearCache();
  alert('Translation cache cleared!');
}
```

---

getCurrentLanguage()

Gets the current language code.

typescript

```
getCurrentLanguage(): string
```

Returns: `string` - Current language code

Example:

typescript

```typescript
const lang = fetchErrorTranslator.getCurrentLanguage();
console.log(lang); // "en"

// Use in UI
document.getElementById('current-lang').textContent =
  `Language: ${fetchErrorTranslator.getCurrentLanguage()}`;

// Conditional logic
if (fetchErrorTranslator.getCurrentLanguage() === 'fr') {
  // French-specific handling
}
```

---

preload()

Preloads translations for a language.

typescript

```typescript
async preload(language: string): Promise<void>
```

Parameters:

Parameter	Type	Description
language	string	Language code to preload

Example:

typescript

```typescript
// Preload at app startup
async function initApp() {
  await fetchErrorTranslator.preload('en');
  await fetchErrorTranslator.preload('fr');
  console.log('Translations preloaded');
}

// Preload on language change
async function changeLanguage(lang: string) {
  await fetchErrorTranslator.preload(lang);
  // Now translations are instant
}
```

---

configAdapter

Gets or sets the cache adapter.

typescript

```typescript
// Getter
get configAdapter(): CacheTranslationInterface

// Setter
set configAdapter(adapter: CacheTranslationInterface)
```

Example:

typescript

```typescript
// Get current adapter
const adapter = fetchErrorTranslator.configAdapter;
console.log(adapter.constructor.name);

// Set custom adapter
import { DexieCacheAdapter } from './adapters';
fetchErrorTranslator.configAdapter = new DexieCacheAdapter();

// Conditional adapter
if (supportsIndexedDB()) {
  fetchErrorTranslator.configAdapter = new DexieCacheAdapter();
} else {
  fetchErrorTranslator.configAdapter = new LocalStorageCacheTranslationAdapter();
}
```

Configuration Options

Complete Configuration Example

typescript

```
import { FetchErrorTranslator } from '@wlindabla/form_validator';

const translator = new FetchErrorTranslator({
    // Default language
    defaultLanguage: 'fr',

    // Enable debug logging
    debug: process.env.NODE_ENV === 'development',

    // Custom cache adapter
    cacheAdapter: new CustomCacheAdapter(),

    // Custom error patterns
    customPatterns: [
        {
            keywords: ['gateway', '502', '503'],
            errorKey: 'ServerError'
        },
        {
            keywords: ['rate limit', '429'],
            errorKey: 'RateLimitError'
        }
    ],

    // Additional translations
    additionalTranslations: {
        en: {
            'ServerError': 'Server is temporarily unavailable',
            'RateLimitError': 'Too many requests - Please wait'
        },
        fr: {
            'ServerError': 'Le serveur est temporairement indisponible',
            'RateLimitError': 'Trop de requêtes - Veuillez patienter'
        }
    }
});
```

Custom Error Patterns

Adding Custom Patterns

typescript

```typescript
const translator = new FetchErrorTranslator({
  customPatterns: [
    // API-specific errors
    {
      keywords: ['api key', 'unauthorized', '401'],
      errorKey: 'AuthError'
    },
    // Server errors
    {
      keywords: ['internal server', '500'],
      errorKey: 'ServerError'
    },
    // Database errors
    {
      keywords: ['database', 'connection refused'],
      errorKey: 'DatabaseError'
    }
  ],
  additionalTranslations: {
    en: {
      'AuthError': 'Authentication failed - Please log in again',
      'ServerError': 'Server error - Please try again later',
      'DatabaseError': 'Database connection error'
    }
  }
});
```

Custom Cache Adapters

Creating a Custom Adapter

typescript

```typescript
import { CacheTranslationInterface, TranslationMessages } from '@wlindabla/form_validator';

class CustomCacheAdapter implements CacheTranslationInterface {
  async getItem(key: string): Promise<TranslationMessages | null> {
    // Your implementation
    return null;
  }

  async setItem(key: string, messages: TranslationMessages): Promise<void> {
    // Your implementation
  }

  async clear(): Promise<void> {
    // Your implementation
  }
}

// Use custom adapter
fetchErrorTranslator.configAdapter = new CustomCacheAdapter();
```

Adding New Languages

Complete Language Addition

typescript

```
// Add Portuguese
fetchErrorTranslator.addTranslations('pt', {
    'AbortError': 'A solicitação expirou',
    'TypeError': 'Erro de rede - Verifique sua conexão com a internet',
    'NetworkError': 'Erro de rede - Conexão impossível',
    'SecurityError': 'Erro de segurança - Acesso negado (CORS)',
    'NotFoundError': 'Recurso não encontrado',
    'TimeoutError': 'Tempo limite excedido',
    'InvalidStateError': 'Solicitação inválida',
    'SyntaxError': 'Erro de sintaxe na resposta',
    'ReferenceError': 'Erro de referência',
    'RangeError': 'Valor fora do intervalo'
});

// Verify
console.log(fetchErrorTranslator.getSupportedLanguages());
// ['en', 'fr', 'es', 'pt']
```

Integration Guides

Basic Fetch Integration

typescript

```typescript
import { fetchErrorTranslator } from '@wlindabla/form_validator';

async function apiCall(url: string) {
    try {
        const response = await fetch(url);

        if (!response.ok) {
            throw new Error(`HTTP ${response.status}`);
        }

        return await response.json();

    } catch (error) {
        // Translate error
        const userMessage = fetchErrorTranslator.translate(error.name, error);

        // Show to user
        showNotification(userMessage, 'error');

        // Rethrow for caller handling
        throw error;
    }
}
```

Axios Integration

typescript

```typescript
import axios from 'axios';
import { fetchErrorTranslator } from '@wlindabla/form_validator';

// Add response interceptor
axios.interceptors.response.use(
    response => response,
    error => {
        // Translate error
        const message = fetchErrorTranslator.translate(
            error.name || 'NetworkError',
            error
        );

        // Show to user
        toast.error(message);

        return Promise.reject(error);
    }
);
```

React Integration

typescript

```
import React, { useState } from 'react';
import { fetchErrorTranslator } from '@wlindabla/form_validator';

function DataFetcher() {
    const [error, setError] = useState<string | null>(null);
    const [data, setData] = useState(null);

    const fetchData = async () => {
        try {
            const response = await fetch('/api/data');
            const json = await response.json();
            setData(json);
            setError(null);
        } catch (err) {
            const errorMessage = fetchErrorTranslator.translate(
                err.name,
                err
            );
            setError(errorMessage);
        }
    };

    return (
        <div>
            {error && (
                <div className="alert alert-danger">
                    {error}
                </div>
            )}
            <button onClick={fetchData}>Fetch Data</button>
        </div>
    );
}
```

Vue.js Integration

vue

```
<template>
  <div>
    <div v-if="error" class="alert alert-danger">
      {{ error }}
    </div>
    <button @click="fetchData">Fetch Data</button>
  </div>
</template>

<script setup lang="ts">
import { ref } from 'vue';
import { fetchErrorTranslator } from '@wlindabla/form_validator';

const error = ref<string | null>(null);

const fetchData = async () => {
  try {
    const response = await fetch('/api/data');
    const data = await response.json();
    error.value = null;
  } catch (err) {
    error.value = fetchErrorTranslator.translate(err.name, err);
  }
};
</script>
```

Default Error Messages

English (en)

Error Name	Message
AbortError	"Request timed out because the server did not respond within the specified time."
TypeError	"Network error - Check your internet connection"
NetworkError	"Network error - Connection impossible"
SecurityError	"Security error - Access denied (CORS)"
NotFoundError	"Resource not found"
TimeoutError	"Request timeout"
InvalidStateError	"Invalid request - Invalid state"
SyntaxError	"Syntax error in response"
ReferenceError	"Reference error"
RangeError	"Value out of range"

French (fr)

Error Name	Message
AbortError	"La requête a expiré car le serveur n'a pas répondu dans le délai imparti."
TypeError	"Erreur réseau - Vérifiez votre connexion internet"
NetworkError	"Erreur réseau - Connexion impossible"
SecurityError	"Erreur de sécurité - Accès non autorisé (CORS)"

Spanish (es)

Error Name	Message
AbortError	"Tiempo de espera agotado - El servidor no respondió a tiempo"
TypeError	"Error de red - Verifique su conexión a internet"
NetworkError	"Error de red - Conexión imposible"
SecurityError	"Error de seguridad - Acceso denegado (CORS)"

TypeScript Types

typescript

```typescript
// Error translations
type ErrorTranslations = Record<string, TranslationMessages>;
type TranslationMessages = Record<string, string>;

// Error pattern
interface ErrorPattern {
  keywords: string[];
  errorKey: string;
}

// Configuration
interface FetchErrorTranslatorConfig {
  defaultLanguage?: string;
  cacheAdapter?: CacheTranslationInterface;
  debug?: boolean;
  customPatterns?: ErrorPattern[];
  additionalTranslations?: ErrorTranslations;
}
```

Best Practices

1. Always Pass Error Object

typescript

```typescript
// ✅ Good - Enables pattern matching
const message = fetchErrorTranslator.translate(error.name, error);

// ❌ Less ideal - No pattern matching
const message = fetchErrorTranslator.translate(error.name);
```

2. Show User-Friendly Messages, Log Technical Details

typescript

```typescript
try {
  await fetch('/api/data');
} catch (error) {
  // For users
  const userMessage = fetchErrorTranslator.translate(error.name, error);
  toast.error(userMessage);

  // For developers
  console.error('Technical details:', {
    name: error.name,
    message: error.message,
    stack: error.stack
  });
}
```

3. Use Language from Document

typescript

```
const message = fetchErrorTranslator.translate(
  error.name,
  error,
  document.documentElement.lang // Use HTML lang attribute
);
```

4. Preload Translations

typescript

```
// Preload at app startup
await fetchErrorTranslator.preload('en');
await fetchErrorTranslator.preload('fr');
```

Troubleshooting

Issue: Translation returns fallback message

Problem: Generic "An unknown error occurred" message

Solution:

typescript

```
// Check if error name is supported
const errorNames = fetchErrorTranslator.getSupportedErrorNames();
console.log('Supported errors:', errorNames);

// Add translation if missing
if (!errorNames.includes('CustomError')) {
  fetchErrorTranslator.addTranslations('en', {
    'CustomError': 'Custom error message'
  });
}
```

License

Credits

Author: AGBOKOUDJO Franck
**