getMetaContent - Meta Tag Utility Library

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

A professional TypeScript utility library for retrieving and parsing HTML meta tag content with maximum browser compatibility (IE8+). Built with jQuery for legacy browser support.

📋 Table of Contents

Installation

NPM / Yarn

bash

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

CDN

html

```html
<!-- jQuery (Required) -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- getMetaContent Library -->
<script src="https://cdn.jsdelivr.net/npm/@wlindabla/form_validator@latest/dist/get-meta-content.min.js"></script>
```

Manual Import

typescript

```typescript
// ES6 Module
import { getMetaContent } from '@wlindabla/form_validator';

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

Basic Usage

HTML Setup

html

```html
<!DOCTYPE html>
<html>
<head>
  <meta name="csrf-token" content="abc123xyz789">
  <meta name="api-url" content="https://api.example.com">
  <meta name="app-config" content='{"theme":"dark","debug":true}'>
</head>
<body>
  <!-- Your content -->
</body>
</html>
```

JavaScript Usage

typescript

```typescript
import { getMetaContent } from '@wlindabala/form_validator';

// Simple retrieval
const csrfToken = getMetaContent('csrf-token');
console.log(csrfToken); // "abc123xyz789"

// With default value
const theme = getMetaContent('theme', { defaultValue: 'light' });
console.log(theme); // "light" (if meta doesn't exist)

// Parse as JSON
const config = getMetaContentAsJSON('app-config');
console.log(config.theme); // "dark"
```

Why Use This Library?

✅ **Type-Safe** - Full TypeScript support with generics
✅ **Browser Compatible** - Works on IE8+ using jQuery
✅ **Error Handling** - Custom error classes for precise error handling
✅ **Flexible** - Multiple parsing options (JSON, Number, Boolean)
✅ **Safe Mode** - Exception-free functions available
✅ **Well-Tested** - Comprehensive test coverage
✅ **Zero Dependencies** - Only requires jQuery (already in most projects)
✅ **Production Ready** - Used in enterprise applications

Core Functions

getMetaContent()

Main function for retrieving meta tag content with full validation.

Signature

typescript

```typescript
function getMetaContent(
  name: string,
  options?: GetMetaContentOptions
): string
```

Parameters

Parameter	Type	Required	Description
name	string	✅ Yes	The name attribute of the meta tag
options	GetMetaContentOptions	❌ No	Configuration options

Options

typescript

```typescript
interface GetMetaContentOptions {
    trim?: boolean;         // Trim whitespace (default: true)
    throwOnEmpty?: boolean; // Throw error if empty (default: true)
    defaultValue?: string;  // Return this if not found (default: undefined)
}
```

Return Value

Returns the `content` attribute value as a string.

Throws

- **`TypeError`** - If name parameter is invalid
- **`JQueryNotAvailableError`** - If jQuery is not loaded
- **`MetaTagNotFoundError`** - If meta tag doesn't exist
- **`EmptyContentError`** - If content is empty (when throwOnEmpty is true)

Examples

Basic Usage

typescript

```typescript
// HTML: <meta name="csrf-token" content="abc123">
const token = getMetaContent('csrf-token');
console.log(token); // "abc123"
```

With Whitespace Handling

typescript

```typescript
// HTML: <meta name="spacing" content="  value  ">
const trimmed = getMetaContent('spacing');
console.log(trimmed); // "value"

const raw = getMetaContent('spacing', { trim: false });
console.log(raw); // "  value  "
```

With Default Value

typescript

```typescript
// Meta tag doesn't exist
const theme = getMetaContent('user-theme', {
    defaultValue: 'light'
});
console.log(theme); // "light"
```

Allow Empty Content

typescript

```typescript
// HTML: <meta name="optional" content="">
const value = getMetaContent('optional', {
    throwOnEmpty: false,
    defaultValue: 'N/A'
});
console.log(value); // ""
```

Combined Options

typescript

```typescript
const config = getMetaContent('config', {
    trim: true,
    throwOnEmpty: false,
    defaultValue: '{}'
});
```

Use Cases

✅ Retrieving CSRF tokens for AJAX requests
✅ Getting API endpoints from configuration
✅ Reading application settings
✅ Accessing user preferences
✅ Loading translation data

---

getMetaContentSafe()

Exception-free version that returns a result object instead of throwing errors.

Signature

typescript

```typescript
function getMetaContentSafe(
    name: string,
    options?: GetMetaContentOptions
): MetaContentResult
```

Return Type

typescript

```typescript
interface MetaContentResult {
    success: boolean;  // true if meta was found
    content?: string;  // the content (if successful)
    error?: string;    // error message (if failed)
}
```

Examples

Basic Usage

typescript

```typescript
const result = getMetaContentSafe('csrf-token');

if (result.success) {
    console.log('Token:', result.content);
    // Use result.content safely
} else {
    console.error('Error:', result.error);
    // Handle error gracefully
}
```

Pattern: Try Meta, Fallback to Alternative

typescript

```typescript
const tokenResult = getMetaContentSafe('csrf-token');

let token: string;
if (tokenResult.success) {
  token = tokenResult.content!;
} else {
  // Fallback: fetch token from API
  token = await fetchTokenFromAPI();
}

useToken(token);
```

Pattern: Multiple Attempts

typescript

```typescript
const sources = ['primary-config', 'backup-config', 'default-config'];

let config: string | null = null;
for (const source of sources) {
  const result = getMetaContentSafe(source);
  if (result.success) {
    config = result.content!;
    break;
  }
}

if (!config) {
  console.error('No configuration found');
}
```

Use Cases

✅ Optional meta tags that may not exist
✅ Graceful degradation scenarios
✅ User-facing applications (no uncaught errors)
✅ Configuration with fallback chains

---

hasMetaTag()

Check if a meta tag exists in the DOM without retrieving its content.

Signature

typescript

```typescript
function hasMetaTag(name: string): boolean
```

Parameters

Parameter	Type	Description
name	string	The name attribute to check

Return Value

Returns `true` if the meta tag exists, `false` otherwise.

Examples

Conditional Retrieval

typescript

```typescript
if (hasMetaTag('csrf-token')) {
  const token = getMetaContent('csrf-token');
  // Use token
} else {
  console.warn('CSRF token not found');
  generateNewToken();
}
```

Validate Required Meta Tags

typescript

```typescript
const requiredMetas = [
  'csrf-token',
  'api-url',
  'app-version'
];

const missingMetas = requiredMetas.filter(name => !hasMetaTag(name));

if (missingMetas.length > 0) {
  throw new Error(`Missing required meta tags: ${missingMetas.join(', ')}`);
}
```

Feature Detection

typescript

```typescript
const features = {
  analytics: hasMetaTag('analytics-id'),
  debugging: hasMetaTag('debug-mode'),
  betaFeatures: hasMetaTag('beta-access')
};

if (features.analytics) {
  initializeAnalytics();
}

if (features.debugging) {
  enableDebugMode();
}
```

Use Cases

✅ Feature flags detection
✅ Validation before retrieval
✅ Conditional initialization
✅ Configuration checks

---

getMultipleMetaContents()

Retrieve multiple meta tag contents in a single call.

Signature

typescript

```typescript
function getMultipleMetaContents(
  names: string[],
  options?: GetMetaContentOptions
): Record<string, string | null>
```

Parameters

Parameter	Type	Description
names	string[]	Array of meta tag names
options	GetMetaContentOptions	Options applied to all

Return Value

Returns an object mapping meta names to their content values. Non-existent metas have `null` values.

Examples

Batch Retrieval

typescript

```typescript
const metas = getMultipleMetaContents([
  'csrf-token',
  'api-url',
  'user-id',
  'session-timeout'
]);

console.log(metas['csrf-token']);      // "abc123"
console.log(metas['api-url']);         // "https://api.example.com"
console.log(metas['user-id']);         // "12345"
console.log(metas['session-timeout']); // "3600"
```

Check for Missing Values

typescript

```typescript
const metas = getMultipleMetaContents([
  'required-1',
  'required-2',
  'optional-1'
]);

const missing = Object.entries(metas)
  .filter(([name, value]) => value === null)
  .map(([name]) => name);

if (missing.length > 0) {
  console.warn('Missing meta tags:', missing);
}
```

Build Configuration Object

typescript

```
const config = getMultipleMetaContents([
  'app-name',
  'app-version',
  'environment',
  'region'
]);

const appConfig = {
  name: config['app-name'] || 'Unknown',
  version: config['app-version'] || '1.0.0',
  env: config['environment'] || 'production',
  region: config['region'] || 'us-east-1'
};

console.log(appConfig);
```

Use Cases

✅ Loading application configuration
✅ Initializing multiple services
✅ Validation of required dependencies
✅ Batch operations for performance

Type Conversion Functions

getMetaContentAsJSON()

Parse meta content as JSON with TypeScript type safety.

Signature

typescript

```typescript
function getMetaContentAsJSON<T = unknown>(
  name: string,
  options?: GetMetaContentOptions
): T
```

Type Parameter

- **`T`** - The expected type of the parsed JSON object (with intellisense!)

Throws

- **`SyntaxError`** - If content is not valid JSON
- All errors from `getMetaContent()`

Examples

Simple Object

typescript

```typescript
// HTML: <meta name="config" content='{"theme":"dark","lang":"en"}'>

interface Config {
  theme: string;
  lang: string;
}

const config = getMetaContentAsJSON<Config>('config');
console.log(config.theme); // "dark" (with autocomplete!)
console.log(config.lang);  // "en"
```

Nested Objects

typescript

```typescript
// HTML: <meta name="user" content='{"name":"John","settings":{"notifications":true}}'>

interface User {
  name: string;
  settings: {
    notifications: boolean;
  };
}

const user = getMetaContentAsJSON<User>('user');
console.log(user.name);                    // "John"
console.log(user.settings.notifications);    // true
```

Array Data

typescript

```
// HTML: <meta name="colors" content='["red","green","blue"]'>

const colors = getMetaContentAsJSON<string[]>('colors');
colors.forEach(color => console.log(color));
// "red"
// "green"
// "blue"
```

Complex Configuration

typescript

```typescript
interface AppConfig {
  api: {
    baseUrl: string;
    timeout: number;
    retries: number;
  };
  features: {
    analytics: boolean;
    chatbot: boolean;
  };
  theme: {
    mode: 'light' | 'dark';
    primaryColor: string;
  };
}

const config = getMetaContentAsJSON<AppConfig>('app-config');

// Full type safety and autocomplete!
if (config.features.analytics) {
  initAnalytics(config.api.baseUrl);
}

document.body.setAttribute('data-theme', config.theme.mode);
```

Error Handling

typescript

```
try {
  const data = getMetaContentAsJSON('config');
  processConfig(data);
} catch (error) {
  if (error instanceof SyntaxError) {
    console.error('Invalid JSON in config meta tag');
    useDefaultConfig();
  }
}
```

Use Cases

✅ Application configuration
✅ Translation data
✅ Feature flags
✅ User preferences
✅ API response formats

---

getMetaContentAsNumber()

Parse meta content as a numeric value.

Signature

typescript

```typescript
function getMetaContentAsNumber(
  name: string,
  options?: GetMetaContentOptions
): number
```

Throws

- **`TypeError`** - If content cannot be parsed as a number
- All errors from `getMetaContent()`

Examples

Integer Values

typescript

```
// HTML: <meta name="max-items" content="100">
const maxItems = getMetaContentAsNumber('max-items');
console.log(maxItems); // 100
console.log(typeof maxItems); // "number"
```

Floating Point

typescript

```
// HTML: <meta name="tax-rate" content="0.075">
const taxRate = getMetaContentAsNumber('tax-rate');
const price = 100;
const total = price + (price * taxRate);
console.log(total); // 107.5
```

Negative Numbers

typescript

```
// HTML: <meta name="temperature" content="-15">
const temp = getMetaContentAsNumber('temperature');
console.log(temp); // -15
```

Scientific Notation

typescript

```
// HTML: <meta name="large-number" content="1.5e6">
const largeNum = getMetaContentAsNumber('large-number');
console.log(largeNum); // 1500000
```

Practical Example: Pagination

typescript

```
// HTML:
// <meta name="page-size" content="20">
// <meta name="total-items" content="157">

const pageSize = getMetaContentAsNumber('page-size');
const totalItems = getMetaContentAsNumber('total-items');
const totalPages = Math.ceil(totalItems / pageSize);

console.log(`Showing ${pageSize} items per page`);
console.log(`Total pages: ${totalPages}`);
```

Practical Example: Timeouts

typescript

```
// HTML:
// <meta name="session-timeout" content="3600">
// <meta name="api-timeout" content="30">

const sessionTimeout = getMetaContentAsNumber('session-timeout') * 1000; // Convert to ms
const apiTimeout = getMetaContentAsNumber('api-timeout') * 1000;

setTimeout(() => {
  console.warn('Session expiring soon');
}, sessionTimeout - 60000); // 1 minute before

fetch('/api/data', {
  signal: AbortSignal.timeout(apiTimeout)
});
```

Use Cases

✅ Pagination settings
✅ Timeout configurations
✅ Numeric limits and thresholds
✅ Dimensions and sizes
✅ Version numbers

---

getMetaContentAsBoolean()

Parse meta content as a boolean value.

Signature

typescript

```typescript
function getMetaContentAsBoolean(
    name: string,
    options?: GetMetaContentOptions
): boolean
```

Truth Values

The following values are considered `true` (case-insensitive):

- `"true"`
- `"1"`
- `"yes"`

All other values return `false`.

Examples

Basic Usage

typescript

```typescript
// HTML: <meta name="debug-mode" content="true">
const debugMode = getMetaContentAsBoolean('debug-mode');
console.log(debugMode); // true

if (debugMode) {
    console.log('Debug mode is enabled');
}
```

Different True Values

typescript

```typescript
// HTML:
// <meta name="feature-a" content="true">
// <meta name="feature-b" content="1">
// <meta name="feature-c" content="yes">
// <meta name="feature-d" content="YES">

console.log(getMetaContentAsBoolean('feature-a')); // true
console.log(getMetaContentAsBoolean('feature-b')); // true
console.log(getMetaContentAsBoolean('feature-c')); // true
console.log(getMetaContentAsBoolean('feature-d')); // true (case-insensitive)
```

False Values

typescript

```typescript
// HTML:
// <meta name="feature-x" content="false">
// <meta name="feature-y" content="0">
// <meta name="feature-z" content="no">
// <meta name="feature-w" content="anything">

console.log(getMetaContentAsBoolean('feature-x')); // false
console.log(getMetaContentAsBoolean('feature-y')); // false
console.log(getMetaContentAsBoolean('feature-z')); // false
console.log(getMetaContentAsBoolean('feature-w')); // false
```

Practical Example: Feature Flags

typescript

```typescript
// HTML:
// <meta name="analytics-enabled" content="true">
// <meta name="maintenance-mode" content="false">
// <meta name="beta-features" content="1">

const analyticsEnabled = getMetaContentAsBoolean('analytics-enabled');
const maintenanceMode = getMetaContentAsBoolean('maintenance-mode');
const betaFeatures = getMetaContentAsBoolean('beta-features');

if (analyticsEnabled) {
  loadAnalyticsScript();
}

if (maintenanceMode) {
  showMaintenancePage();
} else {
  initializeApp();
}

if (betaFeatures) {
  enableBetaFeatures();
}
```

Practical Example: User Preferences

typescript

```
// HTML:
// <meta name="notifications-enabled" content="true">
// <meta name="dark-mode" content="yes">
// <meta name="auto-save" content="1">

const preferences = {
    notifications: getMetaContentAsBoolean('notifications-enabled'),
    darkMode: getMetaContentAsBoolean('dark-mode'),
    autoSave: getMetaContentAsBoolean('auto-save')
};

if (preferences.darkMode) {
    document.body.classList.add('dark-theme');
}

if (preferences.autoSave) {
    startAutoSaveInterval();
}
```

Use Cases

✅ Feature flags
✅ Debug mode toggles
✅ User preferences
✅ Maintenance mode
✅ A/B testing flags

Error Handling

Custom Error Classes

The library provides three custom error classes for precise error handling:

JQueryNotAvailableError

Thrown when jQuery is not loaded or not available globally.

typescript

```typescript
try {
    const content = getMetaContent('test');
} catch (error) {
    if (error instanceof JQueryNotAvailableError) {
        console.error('Please load jQuery before using this library');
    }
}
```

MetaTagNotFoundError

Thrown when the specified meta tag doesn't exist in the DOM.

typescript

```typescript
try {
  const content = getMetaContent('missing-tag');
} catch (error) {
  if (error instanceof MetaTagNotFoundError) {
    console.error(`Meta tag "${error.metaName}" not found`);
    // error.metaName property available
  }
}
```

EmptyContentError

Thrown when the meta tag exists but has empty content.

typescript

```typescript
try {
  const content = getMetaContent('empty-tag');
} catch (error) {
  if (error instanceof EmptyContentError) {
    console.error(`Meta tag "${error.metaName}" has no content`);
    // error.metaName property available
  }
}
```

Comprehensive Error Handling Pattern

typescript

```
function safelyGetMetaContent(name: string): string | null {
  try {
    return getMetaContent(name);
  } catch (error) {
    if (error instanceof JQueryNotAvailableError) {
      console.error('jQuery not loaded');
    } else if (error instanceof MetaTagNotFoundError) {
      console.warn(`Meta "${error.metaName}" not found`);
    } else if (error instanceof EmptyContentError) {
      console.warn(`Meta "${error.metaName}" is empty`);
    } else if (error instanceof TypeError) {
      console.error('Invalid parameter:', error.message);
    } else {
      console.error('Unexpected error:', error);
    }
    return null;
  }
}
```

TypeScript Types

GetMetaContentOptions

Configuration options for meta content retrieval.

typescript

```typescript
interface GetMetaContentOptions {
  /**
   * Whether to trim whitespace from content
   * @default true
   */
  trim?: boolean;

  /**
   * Whether to throw error if content is empty
   * @default true
   */
  throwOnEmpty?: boolean;

  /**
   * Default value to return if meta tag not found
   * @default undefined
   */
  defaultValue?: string;
}
```

MetaContentResult

Result object for safe operations.

typescript

```typescript
interface MetaContentResult {
  /** Whether the operation succeeded */
  success: boolean;

  /** The content value (if successful) */
  content?: string;

  /** Error message (if failed) */
  error?: string;
}
```

Usage with TypeScript

typescript

```typescript
import {
  getMetaContent,
  getMetaContentAsJSON,
  GetMetaContentOptions,
  MetaContentResult
} from '@wlindabla/form_validator';

// Type-safe options
const options: GetMetaContentOptions = {
  trim: true,
  throwOnEmpty: false,
  defaultValue: 'default'
};

// Type-safe JSON parsing
interface MyConfig {
  apiUrl: string;
  timeout: number;
}

const config = getMetaContentAsJSON<MyConfig>('config');
// config.apiUrl has autocomplete!

// Type-safe result handling
const result: MetaContentResult = getMetaContentSafe('token');
if (result.success) {
  const token: string = result.content!;
}
```

Best Practices

1. Always Check jQuery Availability

typescript

```typescript
// At app initialization
if (typeof jQuery === 'undefined') {
  throw new Error('jQuery is required for this application');
}
```

2. Use Default Values for Optional Meta Tags

typescript

```typescript
// Good ✅
const theme = getMetaContent('theme', { defaultValue: 'light' });

// Less ideal ❌
let theme: string;
try {
  theme = getMetaContent('theme');
} catch {
  theme = 'light';
}
```

3. Validate Required Meta Tags Early

typescript

```typescript
// At app startup
const requiredMetas = ['csrf-token', 'api-url', 'app-version'];

requiredMetas.forEach(name => {
  if (!hasMetaTag(name)) {
    throw new Error(`Missing required meta tag: ${name}`);
  }
});
```

4. Use Type-Safe JSON Parsing

typescript

```typescript
// Good ✅
interface Config {
  apiUrl: string;
  debug: boolean;
}
const config = getMetaContentAsJSON<Config>('config');

// Less type-safe ❌
const config = JSON.parse(getMetaContent('config'));
```

5. Centralize Configuration Loading

typescript

```typescript
// config.ts
export class AppConfig {
    static load(): AppConfig {
        return {
            csrfToken: getMetaContent('csrf-token'),
            apiUrl: getMetaContent('api-url'),
            debug: getMetaContentAsBoolean('debug-mode'),
            settings: getMetaContentAsJSON('settings')
        };
    }
}

// app.ts
const config = AppConfig.load();
```

6. Use Safe Functions in User-Facing Code

typescript

```typescript
// User-facing feature
function loadUserPreferences() {
    const result = getMetaContentSafe('user-prefs');

    if (result.success) {
        applyPreferences(result.content!);
    } else {
        useDefaultPreferences();
    }
}
```

Real-World Examples

Example 1: CSRF Token for AJAX

typescript

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

// Get CSRF token
const csrfToken = getMetaContent('csrf-token');

// Use in jQuery AJAX
jQuery.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': csrfToken
    }
});

// Or with fetch
fetch('/api/data', {
    method: 'POST',
    headers: {
        'X-CSRF-TOKEN': csrfToken,
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
});
```

Example 2: Application Configuration

typescript

```typescript
// HTML:
// <meta name="app-config" content='{"apiUrl":"https://api.example.com","timeout":30000,"retries":3}'>

interface AppConfig {
  apiUrl: string;
  timeout: number;
  retries: number;
}

class ApiClient {
  private config: AppConfig;

  constructor() {
    this.config = getMetaContentAsJSON<AppConfig>('app-config');
  }

  async fetchData(endpoint: string) {
    const url = `${this.config.apiUrl}${endpoint}`;

    for (let attempt = 0; attempt < this.config.retries; attempt++) {
      try {
        const response = await fetch(url, {
          signal: AbortSignal.timeout(this.config.timeout)
        });
        return await response.json();
      } catch (error) {
        if (attempt === this.config.retries - 1) throw error;
        await this.delay(1000 * attempt);
      }
    }
  }

  private delay(ms: number) {
    return new Promise(resolve => setTimeout(resolve, ms));
  }
}

const api = new ApiClient();
```

Example 3: Feature Flags System

typescript

```typescript
// HTML:
// <meta name="feature-analytics" content="true">
// <meta name="feature-chatbot" content="false">
// <meta name="feature-beta" content="1">

class FeatureFlags {
  private static features = new Map<string, boolean>();

  static init() {
    const featureMetas = document.querySelectorAll('meta[name^="feature-"]');

    featureMetas.forEach((meta) => {
      const name = meta.getAttribute('name')!.replace('feature-', '');
      const enabled = getMetaContentAsBoolean(meta.getAttribute('name')!);
      this.features.set(name, enabled);
    });
  }

  static isEnabled(feature: string): boolean {
    return this.features.get(feature) ?? false;
  }
}

// Initialize
FeatureFlags.init();

// Use
if (FeatureFlags.isEnabled('analytics')) {
  loadAnalytics();
}

if (FeatureFlags.isEnabled('chatbot')) {
  initChatbot();
}
```

Example 4: Multi-Language Support

typescript

```typescript
// HTML:
// <meta name="translations" content='{"hello":"Bonjour","goodbye":"Au revoir"}'>

interface Translations {
  [key: string]: string;
}

class I18n {
  private translations: Translations;

  constructor() {
    try {
      this.translations = getMetaContentAsJSON<Translations>('translations');
    } catch {
      console.warn('Translations not found, using defaults');
      this.translations = {};
    }
  }

  t(key: string, defaultValue?: string): string {
    return this.translations[key] ?? defaultValue ?? key;
  }
}

const i18n = new I18n();
console.log(i18n.t('hello')); // "Bonjour"
```

Example 5: User Session Management

typescript

```typescript
// HTML:
// <meta name="user-id" content="12345">
// <meta name="session-timeout" content="3600">
// <meta name="is-admin" content="true">

interface UserSession {
  userId: string;
  sessionTimeout: number;
  isAdmin: boolean;
}

class SessionManager {
  private session: UserSession;

  constructor() {
    this.session = {
      userId: getMetaContent('user-id'),
      sessionTimeout: getMetaContentAsNumber('session-timeout'),
      isAdmin: getMetaContentAsBoolean('is-admin')
    };

    this.startSessionTimer();
  }

  private startSessionTimer() {
    const warningTime = (this.session.sessionTimeout - 60) * 1000;

    setTimeout(() => {
      this.showSessionWarning();
    }, warningTime);
  }

  private showSessionWarning() {
    alert('Your session will expire in 1 minute');
  }

  hasPermission(action: string): boolean {
    if (this.session.isAdmin) {
      return true; // Admins can do everything
    }

    // Check specific permissions
    return this.checkUserPermission(action);
  }

  private checkUserPermission(action: string): boolean {
    // Implementation specific to your app
```

```
        return false;
    }
}

const session = new SessionManager();
```

Example 6: API Configuration with Environment Detection

typescript

```typescript
// HTML:
// <meta name="environment" content="production">
// <meta name="api-endpoints" content='{"dev":"http://localhost:3000","staging":"https://staging-api.example.com","product

type Environment = 'dev' | 'staging' | 'production';

interface ApiEndpoints {
  dev: string;
  staging: string;
  production: string;
}

class ApiConfig {
  private environment: Environment;
  private endpoints: ApiEndpoints;

  constructor() {
    this.environment = getMetaContent('environment', {
      defaultValue: 'production'
    }) as Environment;

    this.endpoints = getMetaContentAsJSON<ApiEndpoints>('api-endpoints');
  }

  getApiUrl(): string {
    return this.endpoints[this.environment];
  }

  isDevelopment(): boolean {
    return this.environment === 'dev';
  }

  isProduction(): boolean {
    return this.environment === 'production';
  }
}

const apiConfig = new ApiConfig();
console.log('API URL:', apiConfig.getApiUrl());

if (apiConfig.isDevelopment()) {
  console.log('Development mode - verbose logging enabled');
}
```

API Reference

Core Functions

Function	Return Type	Description
getMetaContent(name, options?)	string	Retrieve meta content with validation
getMetaContentSafe(name, options?)	MetaContentResult	Exception-free retrieval
hasMetaTag(name)	boolean	Check if meta tag exists
getMultipleMetaContents(names, options?)	Record<string, string | null>	Batch retrieval

Type Conversion Functions

Function	Return Type	Description
getMetaContentAsJSON<T>(name, options?)	T	Parse as typed JSON
getMetaContentAsNumber(name, options?)	number	Parse as number
getMetaContentAsBoolean(name, options?)	boolean	Parse as boolean

Error Classes

Class	Properties	Description
JQueryNotAvailableError	name, message	jQuery not loaded
MetaTagNotFoundError	name, message, metaName	Meta tag not found
EmptyContentError	name, message, metaName	Content is empty

TypeScript Interfaces

typescript

```typescript
// Options for getMetaContent
interface GetMetaContentOptions {
  trim?: boolean;
  throwOnEmpty?: boolean;
  defaultValue?: string;
}

// Result for getMetaContentSafe
interface MetaContentResult {
  success: boolean;
  content?: string;
  error?: string;
}
```

Browser Compatibility

Supported Browsers

Browser	Version	Status
Internet Explorer	8+	✅ Fully Supported
Edge	All versions	✅ Fully Supported
Chrome	All versions	✅ Fully Supported
Firefox	All versions	✅ Fully Supported
Safari	All versions	✅ Fully Supported
Opera	All versions	✅ Fully Supported

Requirements

- **jQuery 1.7+** (recommended: jQuery 3.x)
- JavaScript enabled
- HTML5 meta tags

Why jQuery?

✅ **Maximum Compatibility** - Works on IE8+
✅ **Reliable** - Battle-tested selector engine
✅ **Universal** - Already in most web projects
✅ **Consistent** - Same behavior across browsers

Performance Considerations

Caching Strategy

typescript

```typescript
// Cache meta values for repeated access
class MetaCache {
  private cache = new Map<string, string>();

  get(name: string): string {
    if (!this.cache.has(name)) {
      const value = getMetaContent(name);
      this.cache.set(name, value);
    }
    return this.cache.get(name)!;
  }

  clear() {
    this.cache.clear();
  }
}

const metaCache = new MetaCache();

// First call - reads from DOM
const token1 = metaCache.get('csrf-token');

// Subsequent calls - reads from cache
const token2 = metaCache.get('csrf-token'); // Faster!
```

Batch Operations

typescript

```typescript
// Good ✅ - Single batch operation
const metas = getMultipleMetaContents([
  'csrf-token',
  'api-url',
  'user-id'
]);

// Less efficient ❌ - Multiple DOM queries
const token = getMetaContent('csrf-token');
const apiUrl = getMetaContent('api-url');
const userId = getMetaContent('user-id');
```

Migration Guide

From Vanilla JavaScript

Before:

javascript

```javascript
const meta = document.querySelector('meta[name="csrf-token"]');
const token = meta ? meta.getAttribute('content') : null;

if (!token) {
  throw new Error('Token not found');
}
```

After:

typescript

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

const token = getMetaContent('csrf-token');
// Automatic validation and error handling!
```

From jQuery

Before:

javascript

```javascript
const token = $('meta[name="csrf-token"]').attr('content');

if (!token) {
  console.error('Token not found');
}
```

After:

typescript

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

const token = getMetaContent('csrf-token');
// Better error messages and type safety!
```

From Manual JSON Parsing

Before:

javascript

```javascript
const configStr = $('meta[name="config"]').attr('content');
let config;
try {
  config = JSON.parse(configStr);
} catch (error) {
  console.error('Invalid JSON');
  config = {};
}
```

After:

typescript

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

const config = getMetaContentAsJSON('config');
// Type-safe with better error messages!
```

Troubleshooting

Error: jQuery must be globally available

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

Solution:

html

```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>
```

Alternative: Check if jQuery is loaded:

typescript

```typescript
if (typeof jQuery === 'undefined') {
    console.error('Please load jQuery');
}
```

Error: The meta tag does not exist

Problem: The meta tag is not in the DOM.

Solution: Check your HTML:

html

```html
<!-- Ensure meta tag is present -->
<meta name="csrf-token" content="your-token-here">
```

Defensive coding:

typescript

```typescript
if (hasMetaTag('csrf-token')) {
    const token = getMetaContent('csrf-token');
} else {
    console.warn('Token not found, using alternative method');
}
```

Error: Content attribute is empty

Problem: Meta tag exists but has no content.

Solution: Provide content or use `throwOnEmpty: false`:

typescript

```typescript
const value = getMetaContent('optional-meta', {
    throwOnEmpty: false,
    defaultValue: 'default-value'
});
```

Error: Cannot parse as JSON

Problem: Content is not valid JSON.

Solution: Validate your JSON:

html

```html
<!-- Invalid ✗ -->
<meta name="config" content="{key: value}">

<!-- Valid ✅ -->
<meta name="config" content='{"key":"value"}'>
```

Use safe parsing:

typescript

```typescript
try {
  const config = getMetaContentAsJSON('config');
} catch (error) {
  console.error('Invalid JSON, using defaults');
  const config = getDefaultConfig();
}
```

FAQ

Q: Why use jQuery instead of native DOM methods?

A: jQuery provides maximum browser compatibility (IE8+) and consistent behavior across all browsers. For modern projects targeting only newer browsers, you could create a version using native `querySelector`.

Q: Can I use this with React/Vue/Angular?

A: Yes! The library is framework-agnostic. Just ensure jQuery is loaded:

typescript

```
// React
import { getMetaContent } from '@wlindabla/form_validator';

function App() {
  const csrfToken = getMetaContent('csrf-token');
  // Use token...
}

// Vue
import { getMetaContent } from '@wlindabla/form_validator';

export default {
  created() {
    this.token = getMetaContent('csrf-token');
  }
}
```

Q: How do I handle optional meta tags?

A: Use the `defaultValue` option or the safe version:

typescript

```typescript
// Option 1: Default value
const theme = getMetaContent('theme', { defaultValue: 'light' });

// Option 2: Safe version
const result = getMetaContentSafe('theme');
const theme = result.success ? result.content : 'light';
```

Q: Can I retrieve multiple meta tags efficiently?

A: Yes, use `getMultipleMetaContents()`:

typescript

```typescript
const metas = getMultipleMetaContents(['token', 'api-url', 'user-id']);
```

Q: Is this library suitable for production?

A: Yes! It's:

- ✅ Fully tested with comprehensive test suite
- ✅ Type-safe with TypeScript support
- ✅ Used in enterprise applications

- ✅ Well-documented with examples
- ✅ Compatible with legacy browsers

Q: How do I contribute or report issues?

A: Visit our GitHub repository or contact the author:

- **Email:** internationaleswebservices@gmail.com
- **GitHub:** https://github.com/Agbokoudjo/

Examples Repository

Complete Working Examples

Find complete, runnable examples in our GitHub repository:

bash

```bash
git clone https://github.com/Agbokoudjo/getMetaContent-examples
cd getMetaContent-examples
npm install
npm start
```

Examples included:

- Basic usage with vanilla JavaScript
- React application
- Vue.js application
- Angular application
- Symfony integration
- API configuration management
- Feature flags system
- Multi-language support

Changelog

Version 1.0.0 (Current)

Features:

- ✅ Core functions: `getMetaContent`, `getMetaContentSafe`, `hasMetaTag`
- ✅ Type conversion: JSON, Number, Boolean
- ✅ Batch operations with `getMultipleMetaContents`
- ✅ Custom error classes
- ✅ Full TypeScript support
- ✅ Comprehensive test suite
- ✅ Complete documentation

Browser Support:

- ✅ IE8+
- ✅ All modern browsers

Contributing

We welcome contributions! Here's how you can help:

Reporting Bugs

1. Check if the bug has already been reported
2. Create a detailed bug report with:
    - Browser and version
    - jQuery version
    - Code to reproduce the issue
    - Expected vs actual behavior

Suggesting Features

1. Open an issue with the `enhancement` label
2. Describe the feature and use case
3. Provide examples if possible

Pull Requests

1. Fork the repository
2. Create a feature branch
3. Write tests for your changes
4. Ensure all tests pass
5. Submit a pull request

Code Style

- Use TypeScript
- Follow existing code style
- Add JSDoc comments
- Write unit tests
- Update documentation

License

Credits

Author: AGBOKOUDJO Franck
Email: internationaleswebservices@gmail.com
GitHub: https://github.com/Agbokoudjo/
Package: @wlindabla/form_validator

Special Thanks

- jQuery team for the excellent library
- TypeScript team for type system
- All contributors and users

Support

Getting Help

- **Documentation:** You're reading it! 📖
- **Email Support:** internationaleswebservices@gmail.com
- **GitHub Issues:** https://github.com/Agbokoudjo/issues
- **Stack Overflow:** Tag your questions with `getmetacontent`

Professional Support

For enterprise support, custom integrations, or consulting:

- **Email:** internationaleswebservices@gmail.com

Quick Reference Card

Most Common Usage Patterns

typescript

```
// 1. Simple retrieval
const token = getMetaContent('csrf-token');

// 2. With default value
const theme = getMetaContent('theme', { defaultValue: 'light' });

// 3. Safe retrieval (no throw)
const result = getMetaContentSafe('optional-meta');
if (result.success) {
  // Use result.content
}

// 4. Check existence
if (hasMetaTag('debug-mode')) {
  // Meta exists
}

// 5. Parse JSON (type-safe)
const config = getMetaContentAsJSON<ConfigType>('config');

// 6. Parse number
const timeout = getMetaContentAsNumber('timeout');

// 7. Parse boolean
const debugMode = getMetaContentAsBoolean('debug-mode');

// 8. Batch retrieval
const metas = getMultipleMetaContents(['token', 'url', 'id']);

// 9. Error handling
try {
  const value = getMetaContent('meta-name');
} catch (error) {
  if (error instanceof MetaTagNotFoundError) {
    // Handle missing meta
  }
}
```

Made with ❤ by AGBOKOUDJO Franck

Last Updated: 2024

Back to Top

⬆ Back to Table of Contents