readme.md for @design.estate/dees-domtools

> □□ A comprehensive TypeScript toolkit for simplifying DOM manipulation, CSS management, and web component development

Modern web development made elegant. `@design.estate/dees-domtools` provides a powerful suite of utilities for managing complex CSS structures, handling browser events, implementing smooth scrolling, and building responsive web applications with ease.

Features

- □□ **Smart DOM Management** - Singleton-based DomTools instance with race-condition-free initialization
- □□ **Responsive Breakpoints** - Built-in support for desktop, tablet, phablet, and phone viewports with container queries
- □□ **Theme Management** - Automatic dark/light mode detection and switching with RxJS observables
- ⌨ **Keyboard Shortcuts** - Elegant keyboard event handling with combo support
- □□ **Smooth Scrolling** - Native and Lenis-powered smooth scrolling with automatic detection
- □□ **State Management** - Integrated state management with smartstate
- □□ **Routing** - Client-side routing with smartrouter
- □□ **WebSetup** - Easy management of website metadata, favicons, and SEO tags
- □□ **CSS Utilities** - Grid helpers, breakpoint utilities, and base styles for web components

Installation

```
npm install @design.estate/dees-domtools
# or
pnpm add @design.estate/dees-domtools
```

Quick Start

```
import { DomTools } from '@design.estate/dees-domtools';

// Initialize DomTools (singleton pattern - safe to call multiple times)
const domtools = await DomTools.setupDomTools();

// Wait for DOM to be ready
await domtools.domReady.promise;

// Now you're ready to rock! □□
console.log('DOM is ready, head and body elements are available');
```

Core API

DomTools Instance

The `DomTools` class is the heart of the library. It provides a singleton instance that manages all the utilities.

```
import { DomTools } from '@design.estate/dees-domtools';

// Setup with options
const domtools = await DomTools.setupDomTools({
  ignoreGlobal: false // Set to true to create isolated instance
});

// Access DOM elements (available after domReady)
await domtools.domReady.promise;
const head = domtools.elements.headElement;
const body = domtools.elements.bodyElement;
```

Key Properties:

- `domtools.router` - SmartRouter instance for client-side routing
- `domtools.themeManager` - Theme management (dark/light mode)
- `domtools.scroller` - Smooth scrolling utilities
- `domtools.keyboard` - Keyboard event handling
- `domtools.websetup` - Website metadata management
- `domtools.smartstate` - State management
- `domtools.deesComms` - Communication utilities

Lifecycle Promises:

- `domtools.domToolsReady.promise` - Resolves when DomTools is initialized
- `domtools.domReady.promise` - Resolves when DOM is interactive/complete
- `domtools.globalStylesReady.promise` - Resolves when global styles are set

Responsive Breakpoints

Built-in breakpoint system with both media queries and container queries:

```
import { breakpoints, css } from '@design.estate/dees-domtools';
import { css as litCss } from 'lit';

// Breakpoint values (in pixels)
breakpoints.desktop   // 1600px
breakpoints.notebook  // 1240px
breakpoints.tablet    // 1024px
breakpoints.phablet   // 600px
breakpoints.phone     // 400px

// Use with Lit components
const myStyles = litCss`
  .container {
    padding: 20px;
  }

  ${breakpoints.cssForTablet(litCss`
    .container {
      padding: 10px;
    }
  `)}
```

```
  ${breakpoints.cssForPhone(litCss`
    .container {
      padding: 5px;
    }
  `)}
`;
```

Preset viewport helpers (emit both `@media` and `@container wccToolsViewport`):

- `cssForDesktop(css)` - Styles for 1600px and above
- `cssForNotebook(css)` - Styles for 1240px and below
- `cssForTablet(css)` - Styles for 1024px and below
- `cssForPhablet(css)` - Styles for 600px and below
- `cssForPhone(css)` - Styles for 400px and below

Low-level helpers for custom constraints and component-scoped containers:

```
import { breakpoints } from '@design.estate/dees-domtools';
import { css as litCss } from 'lit';

// Viewport-level with custom constraints (emits @media + @container wccToolsViewport)
breakpoints.cssForConstraint({ maxWidth: 800 })(litCss`.box { padding: 8px; }`)

// Component-level — targets a named container (no @media fallback)
breakpoints.cssForContainer(
  litCss`.grid { columns: 1; }`,
  '(max-width: 600px)',
  'my-component'  // CSS container-name
)

// Component-level with custom constraints (curried)
breakpoints.cssForConstraintContainer({ maxWidth: 500 }, 'my-component')(litCss`
  .grid { gap: 8px; }
`)

// Generate containment styles for :host (used by @containerResponsive decorator)
breakpoints.containerContextStyles('my-component')
// → :host { container-type: inline-size; container-name: my-component; }
```

Exported types:

- `ICssForConstraints` — `{ maxWidth?: number; minWidth?: number }`
- `TViewport` — `'native' | 'desktop' | 'tablet' | 'phablet' | 'phone'`

Theme Management

Automatic theme detection with system preference support:

```
const domtools = await DomTools.setupDomTools();
const { themeManager } = domtools;

// Toggle between dark and light
themeManager.toggleDarkBright();

// Set specific theme
themeManager.goDark();
themeManager.goBright();

// Enable automatic global background changes
await themeManager.enableAutomaticGlobalThemeChange();

// Subscribe to theme changes
themeManager.themeObservable.subscribe((isBright) => {
  console.log(`Theme is now: ${isBright ? 'light' : 'dark'}`);
});

// Check current theme
if (themeManager.goBrightBoolean) {
  console.log('Light mode active');
}
```

Keyboard Shortcuts

Handle keyboard events with ease, including complex combinations:

```
import { Keyboard, Key } from '@design.estate/dees-domtools';

const domtools = await DomTools.setupDomTools();
await domtools.domReady.promise;
```

```
// Access the keyboard instance
const { keyboard } = domtools;

// Listen for Ctrl+S
keyboard.on([Key.Ctrl, Key.S]).subscribe((event) => {
  event.preventDefault();
  console.log('Save triggered!');
});

// Listen for Ctrl+Shift+P
keyboard.on([Key.Ctrl, Key.Shift, Key.P]).subscribe(() => {
  console.log('Command palette opened!');
});

// Programmatically trigger key presses
keyboard.triggerKeyPress([Key.Ctrl, Key.S]);

// Clean up when done
keyboard.stopListening();
```

Available Keys:

All standard keyboard keys are available in the `Key` enum, including:

- Modifiers: `Ctrl`, `Shift`, `Alt`
- Letters: `A` through `Z`
- Numbers: `Zero` through `Nine`
- Function keys: `F1` through `F12`
- Navigation: `Home`, `End`, `PageUp`, `PageDown`, arrows
- And many more...

Smooth Scrolling

Powerful scrolling utilities with Lenis integration:

```
const domtools = await DomTools.setupDomTools();
const { scroller } = domtools;

// Scroll to an element smoothly
const targetElement = document.querySelector('#section-2');
```

```
await scroller.toElement(targetElement, {
  duration: 1000,
  easing: 'easeInOutQuad'
});

// Enable Lenis smooth scrolling
await scroller.enableLenisScroll({
  disableOnNativeSmoothScroll: true // Auto-disable if browser has native smooth scroll
});

// Register scroll callbacks
scroller.onScroll(() => {
  console.log('Page scrolled!');
});

// Detect if native smooth scrolling is enabled
const hasNativeSmooth = await scroller.detectNativeSmoothScroll();
```

CSS Utilities

Helper functions for common CSS patterns:

```
import { css } from '@design.estate/dees-domtools';

// Create responsive grid columns
const gridTemplate = css.cssGridColumns(4, 16);
// Returns: calc((100%/4) - (48px/4)) calc((100%/4) - (48px/4)) ...

// Use in your styles
const styles = `
  .grid {
    display: grid;
    grid-template-columns: ${gridTemplate};
    gap: 16px;
  }
`;
```

Global Styles & External Resources

```
const domtools = await DomTools.setupDomTools();

// Add global CSS
await domtools.setGlobalStyles(`
  body {
    margin: 0;
    font-family: 'Inter', sans-serif;
  }
`);

// Load external CSS
await
domtools.setExternalCss('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&displ
ay=swap');

// Load external JavaScript
await domtools.setExternalScript('https://cdn.example.com/analytics.js');
```

Website Metadata

Manage your website's metadata easily:

```
const domtools = await DomTools.setupDomTools();

await domtools.setWebsiteInfo({
  metaObject: {
    title: 'My Awesome App',
    description: 'The best app ever created',
    keywords: ['awesome', 'app', 'web'],
    author: 'Your Name'
  },
  faviconUrl: '/favicon.ico',
  appleTouchIconUrl: '/apple-touch-icon.png'
});
```

Web Component Base Styles

Kickstart your Lit elements with pre-configured styles:

```
import { LitElement } from 'lit';
import { elementBasic } from '@design.estate/dees-domtools';

class MyElement extends LitElement {
  static styles = [elementBasic.staticStyles];

  async connectedCallback() {
    super.connectedCallback();
    await elementBasic.setup(this);
  }
}
```

The `elementBasic.staticStyles` includes:

- Box-sizing reset
- Smooth transitions for background and color
- Custom scrollbar styles
- Default font family (Geist Sans, Inter fallback)

State Management

Integrated state management with smartstate:

```
const domtools = await DomTools.setupDomTools();

// Access the state part
const state = domtools.domToolsStatePart;

// Get current state
const currentState = state.getState();
console.log(currentState.virtualViewport); // 'native'
console.log(currentState.jwt); // ''

// Update state
state.setState({
  virtualViewport: 'tablet',
  jwt: 'your-token-here'
});

// Subscribe to state changes
```

```
state.subscribe((newState) => {
  console.log('State updated:', newState);
});
```

Run Once Pattern

Execute expensive operations only once, even if called multiple times:

```
const domtools = await DomTools.setupDomTools();

// This will only execute once, even if called multiple times
const result = await domtools.runOnce('myExpensiveOperation', async () => {
  console.log('Running expensive operation...');
  await someExpensiveAsyncOperation();
  return 'result';
});

// Subsequent calls return the same result without re-executing
const sameResult = await domtools.runOnce('myExpensiveOperation', async () => {
  console.log('This will never run!');
  return 'different result';
});

console.log(result === sameResult); // true
```

Error handling is built-in - if the function throws, all waiting callers receive the same error.

Advanced Usage

Combining Features

Here's a real-world example combining multiple features:

```
import { DomTools, breakpoints, elementBasic, Key } from '@design.estate/dees-domtools';
import { LitElement, html, css as litCss } from 'lit';
import { customElement } from 'lit/decorators.js';
```

```
@customElement('my-app')
class MyApp extends LitElement {
  static styles = [
    elementBasic.staticStyles,
    litCss`
      :host {
        display: block;
        padding: 2rem;
      }

      ${breakpoints.cssForTablet(litCss`
        :host {
          padding: 1rem;
        }
      `)}
    `
  ];

  private domtools?: DomTools;

  async connectedCallback() {
    super.connectedCallback();

    // Setup DomTools
    this.domtools = await elementBasic.setup(this);
    await this.domtools.domReady.promise;

    // Setup keyboard shortcuts
    this.domtools.keyboard.on([Key.Ctrl, Key.K]).subscribe(() => {
      this.openCommandPalette();
    });

    // Subscribe to theme changes
    this.domtools.themeManager.themeObservable.subscribe((isBright) => {
      this.requestUpdate();
    });

    // Enable smooth scrolling
```

```
    await this.domtools.scroller.enableLenisScroll({
      disableOnNativeSmoothScroll: true
    });
  }

  private openCommandPalette() {
    console.log('Command palette opened!');
  }

  render() {
    const isDark = !this.domtools?.themeManager.goBrightBoolean;

    return html`
      <div class="app" style="background: ${isDark ? '#1a1a1a' : '#ffffff'}">
        <h1>My Awesome App</h1>
        <button @click=${() => this.domtools?.themeManager.toggleDarkBright()}>
          Toggle Theme
        </button>
      </div>
    `;
  }
}
```

TypeScript Support

This package is written in TypeScript and provides full type definitions:

```
import type {
  IDomToolsState,
  IDomToolsContructorOptions,
  TViewport
} from '@design.estate/dees-domtools';

// Custom state interface
interface MyState extends IDomToolsState {
  customProperty: string;
}
```

```
// Type-safe viewport handling
const viewport: TViewport = 'tablet';
```

Browser Support

Targets the latest version of Chrome. For other browsers, you may need to include polyfills.

Why @design.estate/dees-domtools?

- **Race-condition free** - Carefully designed initialization prevents common timing issues
- **TypeScript first** - Full type safety and IntelliSense support
- **Modern APIs** - Built on Lit, RxJS, and other modern web standards
- **Batteries included** - Everything you need for sophisticated web apps
- **Production ready** - Used in real-world applications at design.estate
- **Well maintained** - Active development and support

Related Packages

This library integrates with the design.estate ecosystem:

- `@design.estate/dees-comms` - Communication utilities
- `@push.rocks/websetup` - Website setup and meta management
- `@push.rocks/smartrouter` - Client-side routing
- `@push.rocks/smartstate` - State management

License and Legal Information

use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #6
Created 2026-03-28 10:49:09 UTC by foss.global Team
Updated 2026-03-28 12:14:31 UTC by foss.global Team