@uptime.link/webwidget

the webwidget for uptime.link

- readme.md for @uptime.link/webwidget
- changelog.md for @uptime.link/webwidget

readme.md for @uptime.link/webwidget

@uptimelink/webwidget

the webwidget for public use of uptimelink

Install

To install the `@uptimelink/webwidget` package, you need to have Node.js and npm (or yarn) installed. Once you have these prerequisites, you can install the package via npm by running the following command in your terminal:

```
npm install @uptimelink/webwidget
```

Alternatively, if you use yarn, you can run:

```
yarn add @uptimelink/webwidget
```

Usage

Introduction

The `@uptimelink/webwidget` package provides a web component that can be embedded into web pages to display uptime information for a given project on the UptimeLink platform. The component is implemented using TypeScript and leverages modern web standards including Web Components and LitElement.

Basic Setup

First, you will need to import the `UptimelinkWebwidget` class and define it in your project. To do this, create an HTML file and include a script to register the web component.

```
// index.ts
import { UptimelinkWebwidget } from '@uptimelink/webwidget';

// Append the webwidget component to the document body
document.body.appendChild(UptimelinkWebwidget.demo());
```

Setting Up in HTML

To use the widget in an HTML document, you will need to include the built JavaScript file. Here is an example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Uptime Link Widget</title>
  </head>
  <body>
    <script type="module">
      import { UptimelinkWebwidget } from './path_to_your_built_index.js';
      customElements.define('uptimelink-webwidget', UptimelinkWebwidget);

      const widgetElement = document.createElement('uptimelink-webwidget');
      widgetElement.projectSlug = 'uptime.link';
      document.body.appendChild(widgetElement);
    </script>
  </body>
</html>
```

Creating a Custom Page with the Widget

You can integrate the widget into a custom page to display uptime status. First, create a new TypeScript file for the page.

```
// pages/customPage.ts
import { html } from '@design.estate/dees-element';

export const customPage = () => html`
  <style>
    .container {
      box-sizing: border-box;
      position: absolute;
      top: 20px;
      height: 120px;
      padding: 40px;
      width: 100%;
      background: rgba(0, 0, 0, 0.2);
    }
  </style>
  <div class="container">
    <uptimelink-webwidget
      projectSlug="custom-project-slug"
    ></uptimelink-webwidget>
  </div>
`;
```

Styling the Widget

The `UptimelinkWebwidget` component comes with default styles but you can override those styles to match your site's aesthetics. Below is an example of how this can be done:

```
// customStyles.ts
import { UptimelinkWebwidget } from '@uptimelink/webwidget';
import { cssManager } from '@design.estate/dees-element';

UptimelinkWebwidget.styles = [
  cssManager.defaultStyles,
  css`
    .mainbox {
      background-color: #f0f0f0;
      color: #333;
      border: 1px solid #ccc;
    }
```

```
    .statusindicator {
      background: #28a745;
    }
  `,
];

document.body.appendChild(UptimelinkWebwidget.demo());
```

Advanced Interactions

The `UptimelinkWebwidget` allows for advanced interactions such as hovering effects to show detailed information. This is handled within the component's lifecycle and event handlers. Below is a detailed code snippet demonstrating lifecycle hooks and event listeners:

```
// uppimelink-webwidget.ts
import {
  DeesElement,
  property,
  html,
  customElement,
  type TemplateResult,
  cssManager,
} from '@design.estate/dees-element';
import { DeesWindowLayer } from '@design.estate/dees-catalog';

@customElement('uptimelink-webwidget')
export class UptimelinkWebwidget extends DeesElement {
  @property({ type: Boolean }) isOnTop = false;
  @property() public projectSlug: string;
  @property() public isFocused = false;
  @property() public isElevated = false;
  @property() public showExpanded: boolean = false;

  constructor() {
    super();
    this.setupEventing();
  }

  public static styles = [cssManager.defaultStyles];
```

```
public render(): TemplateResult {
  return html`
    <style>
      /* Add your custom styles here */
    </style>
    <div class="mainbox ${this.isFocused ? 'focused' : null}">
      <div class="firstLine">
        <div class="statusindicator"></div>
        <div class="statustext">All systems are up!</div>
      </div>
      ${this.showExpanded
        ? html` <div class="expanded">/* Expanded view content */</div> `
        : null}
    </div>
  `;
}

private async setupEventing() {
  const domtools = await this.domtoolsPromise;
  await this.updateComplete;
  const mainbox = this.shadowRoot.querySelector('.mainbox') as HTMLDivElement;

  mainbox.onmouseenter = async () => {
    if (!this.isOnTop) {
      const rect = mainbox.getBoundingClientRect();
      const uptimelinkWidget = new UptimelinkWebwidget();
      uptimelinkWidget.isOnTop = true;
      uptimelinkWidget.style.position = 'fixed';
      uptimelinkWidget.style.top = `${rect.top}px`;
      uptimelinkWidget.style.left = `${rect.left}px`;
      document.body.append(uptimelinkWidget);
      return;
    }
    this.isElevated = true;
    this.isFocused = true;
    this.windowLayer = await DeesWindowLayer.createAndShow({ blur: true });
    await domtools.convenience.smartdelay.delayFor(200);
    if (!this.isFocused) return;
```

```
      this.showExpanded = true;
      await this.performUpdate();
      await domtools.convenience.smartdelay.delayFor(50);
      const expandedDiv = this.shadowRoot.querySelector(
        '.expanded',
      ) as HTMLElement;
      expandedDiv.style.opacity = '1';
    };

    mainbox.onmouseleave = async () => {
      if (!this.isOnTop) return;
      this.windowLayer.destroy();
      domtools.convenience.smartdelay.delayFor(200).then(() => {
        if (!this.isFocused) {
          this.isElevated = false;
          this.remove();
        }
      });
      if (!this.showExpanded) {
        this.isFocused = false;
        return;
      }
      this.showExpanded = false;
      await domtools.convenience.smartdelay.delayFor(50);
      this.isFocused = false;
    };
  }
}
```

Utilizing Multiple Widgets

You can also utilize multiple instances of the web widget on a single page. Below is an example of how you could do this:

```
// multipleWidgets.ts
import { UptimelinkWebwidget } from '@uptimelink/webwidget';

const widget1 = document.createElement('uptimelink-webwidget');
const widget2 = document.createElement('uptimelink-webwidget');
```

```
widget1.projectSlug = 'project1';
widget2.projectSlug = 'project2';

document.body.appendChild(widget1);
document.body.appendChild(widget2);
```

Building and Serving

To build the project, use the following npm scripts:

```
npm run build
```

To watch for changes and rebuild automatically:

```
npm run watch
```

Testing

Currently, the `test` script is identical to the `build` script. Running tests requires building the project:

```
npm run test
```

Contributing

Contributions to the `@uptimelink/webwidget` project are welcome! Please follow the guidelines provided in the repository. Reach out on the project issues page for discussions, questions, or follow the established process for submitting pull requests.

Conclusion

This guide covered the steps needed to install, set up, and use the `@uptimelink/webwidget` package in your TypeScript projects. We also explored various ways to customize and extend the widget, ensuring seamless integration into your existing web applications. For more detailed information, refer to the documentation.

Feel free to check the links provided in the initial sections for more context and updates about the project's status and availability. undefined

changelog.md for @uptime.link/webwidget

2025-12-15 - 1.2.6 - fix(ci/deps)

Update npmextra configuration keys and upgrade dependencies/devDependencies

- Renamed npmextra.json keys: 'gitzone' -> '@git.zone/cli' and 'npmci' -> '@ship.zone/szci'; added 'release' block with registries and accessLevel
- Updated runtime dependencies: @design.estate/dees-catalog ^2.0.3 -> ^3.3.1, @design.estate/dees-wcctools ^1.2.1 -> ^2.0.1
- Updated devDependencies: @git.zone/tsbuild ^3.1.2 -> ^4.0.2, @git.zone/tsrun 2.0.0 -> 2.0.1, @git.zone/tswatch 2.2.3 -> 2.3.13, @types/node ^24.10.1 -> ^25.0.2

2025-12-04 - 1.2.5 - fix(dependencies)

Update dependency versions

- Bumped @design.estate/dees-catalog from ^2.0.0 to ^2.0.3
- Bumped @git.zone/tsbundle from ^2.6.1 to ^2.6.3
- Bumped @git.zone/tswatch from ^2.2.1 to ^2.2.3

2025-11-30 - 1.2.4 - fix(uptimelink-webwidget)

Modernize element props (use accessor), bump deps, add CI/workflows, export pages and update package metadata/docs

- Converted Lit/DeesElement @property fields to use TypeScript 'accessor' for uptimelink-webwidget properties (isOnTop, projectSlug, isFocused, isElevated, showExpanded)
- Bumped runtime and dev dependencies (@design.estate/* and @git.zone/*) and added @types/node
- Added buildDocs script (tsdoc) and adjusted npm scripts
- Added CI workflows (.gitea/workflows) for tagged and non-tagged runs
- Exported pages (ts_web/pages) and added example page modules (page1, page2)
- Updated package.json metadata (repository, bugs, homepage, pnpm.overrides)
- Adjusted tsconfig (target/module settings, baseUrl/paths, exclude) to align with modern toolchain
- Updated README and changelog entries and cleaned html/index.html comments
- Added .gitignore entries for AI tooling directories (.claude, .serena)

2025-01-09 - 1.2.3 - fix(webwidget)

Fix style issues for UptimelinkWebwidget element.

- Resolved indentation and formatting issues in UptimelinkWebwidget component.
- Ensured proper use of cssManager for theming.
- Fixed event handling for expanded view toggle.

2025-01-09 - 1.2.2 - fix(component)

Fix sizing of status indicator in uptimelink-webwidget

- Corrected the status indicator dimensions for consistency.

2025-01-09 - 1.2.1 - fix(UptimelinkWebwidget)

Remove incorrect scale transformation in focused state.

- Fixed the CSS transform property for the .mainbox.focused class in the UptimelinkWebwidget component by removing an unintended scale transformation.

2025-01-09 - 1.2.0 - feat(webwidget)

Enhanced widget animations and styling

- Enabled reflect option for 'isOnTop' and 'showExpanded' properties.
- Improved the animation and styling of the 'mainbox' when it gains focus.
- Mainbox now has smoother opacity transitions and better background handling.
- Enhanced functionality of widget positioning for better rendering.

2024-06-28 - 1.1.2 - fix(elements)

Fix blur effect in UptimelinkWebwidget component

- Changed blur effect from true to false in the DeesWindowLayer setup

2024-06-27 - 1.1.1 - fix(build)

Fix build script to use correct tsbuild command

- Corrected the build script in package.json to use 'tsbuild tsfolders --allowimplicitany && tsbundle element --production' instead of 'tsbuild element --allowimplicitany && tsbundle element --production'.

2024-06-27 - 1.1.0 - feat(elements)

Add performance improvements to uptimelink-webwidget

- Added 'will-change: transform' to improve performance of the transform property
- Increased transition speed for element opacity from 0.2s to 0.1s for faster visual feedback
- Added scale transformation effect to the focused state of the widget

2024-06-27 - 1.0.82 - fix(core)

No code changes detected. Preparing for version increment based on package state.

2024-06-26 - 1.0.81 - fix(core)

Updated dependencies and improved code documentation.

- Updated internal dependencies to newer versions.
- Improved code documentation within TypeScript files.
- Adjusted scripts in package.json for better build performance.

2024-06-26 - 1.0.80 - fix(documentation)

Update package description and README for clarity

- Updated the README to provide better installation and usage instructions.
- Ensured that all sections contain relevant TypeScript and HTML examples.
- Added styling and advanced interaction examples to README.

2024-06-26 - 1.0.79 - fix(core)

Fixed various package metadata issues and improved component interactions

- Corrected package name and description in package.json
- Updated dependencies and devDependencies to latest versions
- Enhanced installation and usage documentation in readme.md
- Replaced internal 'UptimelinkWindowLayer' with '@design.estate/dees-catalog' dependency

2023-10-06 - 1.0.78 - Core

Multiple core updates

- Applied fixes and updates to core functionality repeatedly from version 1.0.55 to 1.0.78.

2022-03-25 - 1.0.69 - Core

Core updates and fixes

- Incremental core fixes and updates from version 1.0.55 to 1.0.69.

2021-03-09 - 1.0.67 - Core

Series of core fixes

- Continued fixes and updates to core from version 1.0.55 to 1.0.67.

2020-11-29 - 1.0.61 - Core

Regular core fixes

- Ongoing core fixes and updates from version 1.0.55 to 1.0.61.

2020-09-19 - 1.0.58 - Core

Core updates

- Core has been updated and fixed multiple times from version 1.0.55 to 1.0.58.