



## Building a Web-App

If you want to include the Tram-Lite HTML API on your page, include the following script tag:

```
<script src="https://unpkg.com/tram-lite@4"></script>
```

## Building a Library

If you are building a library with Tram-Lite, you can make component definitions in HTML files, and have users import them using `import-components.js`

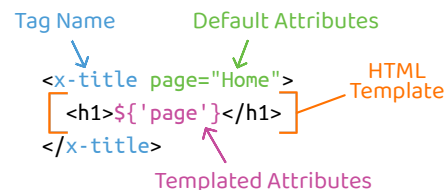
```
<script src="../../../tram-lite@4/output/import-components.js"
  tl-components="./your-component.html ...">
```

If you are building a library, and would like to serve component definitions as a single bundle, you can use the `export-components` CLI command to make a JS file.

```
$ npx tram-lite@4 export-components x-title.html ...
```

Both `import-components.js` and assets built with `export-components` do not depend on Tram-Lite being on the page, and can even work when a different version of Tram-Lite is included on the page.

## Anatomy of a Component Definition



**Tag Name** — name of the new tag, needs to be hyphenated.

**Default Attributes** — optional, attributes that can be used in the template.

**HTML Template** — HTML that will be inserted when an instance of this web-component is created. Can include style tags, and script tags with the `tl-effect` attribute.

**Templated Attributes** — optional, attributes to include in your HTML or CSS (in style tags).

If you included the Tram-Lite API on your page, you can build component definitions in your HTML using the `tl-definition` attribute on template tags.

```
<template tl-definition>
  <x-title ... </x-title>
</template>
```

## Side-Effects using tl-effect

```
<x-title page="Home">
  <script tl-effect
    tl-dependencies="page">
    this.getAttribute('page')
  </script>
</x-title>
```

**tl-dependencies** — optional, attribute list that when updated will cause the script to re-trigger.

**this** — reference to the host component in the light DOM.

`tl-effect` works with script tags, and is required for script tags to work on-mount for Tram-Lite components.

## Interfacing with the ShadowRoot

When you want to query or interface with elements inside a web-component, you can use the `shadowRoot` attribute. This can be useful for attaching event listeners or getting the state of internal elements.

```
<x-button clicked='false'>
  <button>Was Clicked: ${'clicked'}</button>
  <script tl-effect>
    const b = this.shadowRoot.querySelector('button')
    b.addEventListener('click', () => {
      this.setAttribute('clicked', 'true')
    })
  </script>
</x-button>
```

Full Documentation And More  
Go to the website [tram-one.io/tram-lite](https://unpkg.com/tram-one.io/tram-lite)

Tram-Lite is a declarative HTML-first library for building native web-components. The API provides an elegant and easy way to build modern components, that work in any framework, and on any modern platform, all in HTML.

What is Tram-Lite?

## Controlled Inputs using tl-controlled

```
<x-title page="Home">
  <input tl-controlled
    tl-hostattr="page"
    tl-trigger="input" />
</x-title>
```

**tl-hostattr** — optional, attribute to set on the host component, by default "value"

**tl-trigger** — optional, attribute list of events that should trigger updates to the host component

`tl-controlled` can be used added to any element in the shadow DOM. It specifically works well with input and select elements.

## Style Tags and Global CSS

You can use `style` tags to have scoped CSS for an instance of an element. If you'd like to reference the host element, you can use the `:host` selector.

```
<x-title>
  <style>
    :host { color: blue }
  </style>
</x-title>
```

If you have global styles that you want to apply to your component, include a `link` tag in your element.

```
<x-title>
  <link rel="stylesheet" href="./styles.css" />
</x-title>
```