mdn_

HTML

HTML: Markup language

HTML REFERENCE
Elements
Global attributes
Attributes
See all…
HTML GUIDES
Responsive images
HTML cheatsheet
Date & time formats
See all…
MARKUP LANGUAGES
SVG
MathML
XML
CSS

CSS: Styling language

CSS REFERENCE
Properties
Selectors
At-rules
Values
See all…
CSS GUIDES
Box model
Animations
Flexbox
Colors
See all…
LAYOUT COOKBOOK
Column layouts
Centering an element
Card component
See all…
JavaScriptJS

JavaScript: Scripting language

JS REFERENCE
Standard built-in objects
Expressions & operators
Statements & declarations
Functions
See all…
JS GUIDES
Control flow & error handing
Loops and iteration
Working with objects
Using classes
See all…
Web APIs

Web APIs: Programming interfaces

WEB API REFERENCE
File system API
Fetch API
Geolocation API
HTML DOM API
Push API
Service worker API
See all…
WEB API GUIDES
Using the Web animation API
Using the Fetch API
Working with the History API
Using the Web speech API
Using web workers
All

All web technology

TECHNOLOGIES
Accessibility
HTTP
URI
Web extensions
WebAssembly
WebDriver
See all…
TOPICS
Media
Performance
Privacy

- Security
- Progressive web apps

Learn

- Learn web development

- FRONTEND DEVELOPER COURSE
- Getting started modules
- Core modules
- MDN Curriculum
- LEARN HTML
- Structuring content with HTML module
- LEARN CSS
- CSS styling basics module
- CSS layout module
- LEARN JAVASCRIPT
- Dynamic scripting with JavaScript module

Tools

- Discover our tools

- Playground
- HTTP Observatory
- Border-image generator
- Border-radius generator
- Box-shadow generator
- Color format converter
- Color mixer
- Shape generator

About

- Get to know MDN better

- About MDN
- Advertise with us
- Community
- MDN on GitHub

Blog

Log in

Exit search

- Web
- HTML
- How to
- Use data attributes

Use data attributes

HTML is designed with extensibility in mind for data that should be associated with a particular element but need not have any defined meaning. `data-*` attributes allow us to store extra information on standard, semantic HTML elements without other hacks such as non-standard attributes, or extra properties on DOM.

In this article

HTML syntax

The syntax is simple. Any attribute on any element whose attribute name starts with `data-` is a data attribute. Say you have some articles and you want to store some extra information that doesn't have any visual representation. Just use `data` attributes for that:

html Copy

```html
<main>
  <article
    id="electric-cars"
    data-columns="3"
    data-index-number="12314"
    data-parent="cars">
    <!-- Electric car content -->
  </article>

  <article
    id="solar-cars"
    data-columns="3"
    data-index-number="12315"
    data-parent="cars">
    <!-- Solar car content -->
  </article>

  <article
    id="flying-cars"
    data-columns="4"
    data-index-number="12316"
    data-parent="cars">
    <!-- Flying car content -->
  </article>
</main>
```

JavaScript access

Reading the values of these attributes out in JavaScript is also very simple. You could use `getAttribute()` with their full HTML name to read them, but the standard defines a simpler way: a `DOMStringMap` you can read out via a `dataset` property.

To get a `data` attribute through the `dataset` object, get the property by the part of the attribute name after `data-` (note that dashes are converted to camel case).

js Copy

```js
const article = document.querySelector("#electric-cars");
// The following would also work:
// const article = document.getElementById("electric-cars")

article.dataset.columns; // "3"
article.dataset.indexNumber; // "12314"
article.dataset.parent; // "cars"
```

Each property is a string and can be read and written. In the above case setting `article.dataset.columns = 5` would change that attribute to `"5"`.

You can also use `document.querySelector()` or `document.querySelectorAll()` with data attribute selectors to find one element or all elements that match:

js Copy

```
// Find all elements with a data-columns attribute
const articles = document.querySelectorAll("[data-columns]");

// Find all elements with data-columns="3"
const threeColumnArticles = document.querySelectorAll('[data-columns="3"]');
// You can then iterate over the results
threeColumnArticles.forEach((article) => {
  console.log(article.dataset.indexNumber);
});
```

CSS access

Note that, as data attributes are plain HTML attributes, you can even access them from CSS. For example to show the parent data on the article you can use generated content in CSS with the `attr()` function:

css Copy

```
article::before {
  content: attr(data-parent);
}
```

You can also use the attribute selectors in CSS to change styles according to the data:

css Copy

```
article[data-columns="3"] {
  width: 400px;
}
article[data-columns="4"] {
  width: 600px;
}
```

Data values are strings. Number values must be quoted in the selector for the styling to take effect.

Examples

Style variants

Imagine you have a `callout` class. Now you want to implement different variants, such as "note" and "warning". Traditionally, people just use different class names.

html Copy

```
<div class="callout callout--note">...</div>
<div class="callout callout--warning">...</div>
```

css Copy

```
.callout {
  margin: 0.5em 0;
  padding: 0.5em;
  border-radius: 4px;
  border-width: 2px;
  border-style: solid;
}

.callout--note {
  border-color: rgb(15 15 235);
  background-color: rgb(15 15 235 / 0.2);
}
.callout--warning {
  border-color: rgb(235 15 15);
  background-color: rgb(235 15 15 / 0.2);
}
```

With data attributes, here's an alternative you can consider:

html Copy Play

```
<div class="callout">...</div>
<div class="callout" data-variant="note">...</div>
<div class="callout" data-variant="warning">...</div>
```

css Copy Play

```
.callout {
  margin: 0.5em 0;
  padding: 0.5em;
  border-radius: 4px;
  border-width: 2px;
  border-style: solid;
}

/* Default style */
.callout:not([data-variant]) {
  border-color: rgb(15 15 15);
  background-color: rgb(15 15 15 / 0.2);
}
.callout[data-variant="note"] {
  border-color: rgb(15 15 235);
  background-color: rgb(15 15 235 / 0.2);
}
.callout[data-variant="warning"] {
  border-color: rgb(235 15 15);
  background-color: rgb(235 15 15 / 0.2);
}
```

Play

There are multiple benefits of this:

- It eliminates a lot of invalid states, such as applying `callout--note` without also adding `callout`, or applying multiple variants simultaneously.
- A separate `data-variant` attribute allows static analysis for valid values via linting or type checking.
- Toggling the variant is more intuitive: you can use `div.dataset.variant = "warning";` instead of manipulating the `classList` which requires multiple steps.

Associating arbitrary data with DOM elements

Many web apps have JavaScript data as the source-of-truth for their UI state. In these cases, you only add HTML attributes necessary for rendering. Data attributes are useful in the cases where everything is present in the markup, and JavaScript is only needed for handling events, syncing state, etc.

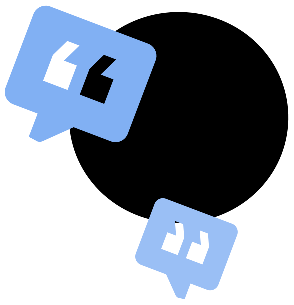
For example, in our carousel with scroll margin example, we have an HTML page already populated with many `<img>` elements. The image's source is initially stored in `data-src` to prevent any request being fired, and the real `src` is only added when the `<img>` scrolls into view. The data (image source) is colocated with the element, and the JavaScript is only responsible for defining behavior.

Issues

Do not store content that should be visible and accessible in data attributes, because assistive technology may not access them. In addition, search crawlers may not index data attributes' values. Often, if you only intend for the data attribute to be displayed, you can directly manipulate `textContent`.

See also

- This article is adapted from Using data attributes in JavaScript and CSS on hacks.mozilla.org.
- Custom attributes are also supported in SVG 2; see `HTMLElement.dataset` and `data-*` for more information.
- How to use HTML data attributes (Sitepoint)



Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute

Filter sidebar

HTML
Guides
Cheatsheet
Comments
Constraint validation
Content categories
Date and time formats
Microdata
Microformats
Quirks and standards modes
Responsive images
How to
Define terms with HTML
Use data attributes
Use cross-origin images
Add a hitmap on top of an image
Author fast-loading HTML pages
Add JavaScript
Reference
Elements
`<a>`
`<abbr>`
`<acronym>` Deprecated
`<address>`
`<area>`
`<article>`
`<aside>`
`<audio>`
`<b>`
`<base>`
`<bdi>`
`<bdo>`
`<big>` Deprecated
`<blockquote>`

`<body>`
`<br>`
`<button>`
`<canvas>`
`<caption>`
`<center>` Deprecated
`<cite>`
`<code>`
`<col>`
`<colgroup>`
`<data>`
`<datalist>`
`<dd>`
`<del>`
`<details>`
`<dfn>`
`<dialog>`
`<dir>` Deprecated
`<div>`
`<dl>`
`<dt>`
`<em>`
`<embed>`
`<fencedframe>` Experimental
`<fieldset>`
`<figcaption>`
`<figure>`
`<font>` Deprecated
`<footer>`
`<form>`
`<frame>` Deprecated
`<frameset>` Deprecated
`<h1>`
`<head>`
`<header>`
`<hgroup>`
`<hr>`
`<html>`
`<i>`
`<iframe>`
`<img>`
`<input>`
`<ins>`
`<kbd>`
`<label>`
`<legend>`
`<li>`
`<link>`
`<main>`
`<map>`
`<mark>`
`<marquee>` Deprecated
`<menu>`
`<meta>`
`<meter>`

`<nav>`
`<nobr>` Deprecated
`<noembed>` Deprecated
`<noframes>` Deprecated
`<noscript>`
`<object>`
`<ol>`
`<optgroup>`
`<option>`
`<output>`
`<p>`
`<param>` Deprecated
`<picture>`
`<plaintext>` Deprecated
`<pre>`
`<progress>`
`<q>`
`<rb>` Deprecated
`<rp>`
`<rt>`
`<rtc>` Deprecated
`<ruby>`
`<s>`
`<samp>`
`<script>`
`<search>`
`<section>`
`<select>`
`<selectedcontent>` Experimental
`<slot>`
`<small>`
`<source>`
`<span>`
`<strike>` Deprecated
`<strong>`
`<style>`
`<sub>`
`<summary>`
`<sup>`
`<table>`
`<tbody>`
`<td>`
`<template>`
`<textarea>`
`<tfoot>`
`<th>`
`<thead>`
`<time>`
`<title>`
`<tr>`
`<track>`
`<tt>` Deprecated
`<u>`
`<ul>`

`spellcheck`
`style`
`tabindex`
`title`
`translate`
`virtualkeyboardpolicy` Experimental
`writingsuggestions`
Attributes by element
`<input>` type
`<input type="button">`
`<input type="checkbox">`
`<input type="color">`
`<input type="date">`
`<input type="datetime-local">`
`<input type="email">`
`<input type="file">`
`<input type="hidden">`
`<input type="image">`
`<input type="month">`
`<input type="number">`
`<input type="password">`
`<input type="radio">`
`<input type="range">`
`<input type="reset">`
`<input type="search">`
`<input type="submit">`
`<input type="tel">`
`<input type="text">`
`<input type="time">`
`<input type="url">`
`<input type="week">`
`<script>` type
`importmap`
`speculationrules` Experimental
<meta> name
`color-scheme`
`referrer`
`robots`
`theme-color`
`viewport`
<meta> http-equiv
Attribute values
`rel` keywords
`rel="alternate stylesheet"`
`rel="compression-dictionary"` Experimental
`rel="dns-prefetch"`
`rel="manifest"`
`rel="me"`
`rel="modulepreload"`
`rel="noopener"`
`rel="noreferrer"`
`rel="preconnect"`
`rel=prefetch`
`rel=preload`

`rel=prerender` Non-standard Deprecated

Build, secure, and ship software faster with one platform Start your free trial GitLab

Ad

mdn_

Your blueprint for a better internet.