<link>: The External Resource Link element

Baseline Widely available *

The `<link>` HTML element specifies relationships between the current document and an external resource. This element is most commonly used to link to stylesheets, but is also used to establish site icons (both "favicon" style icons and icons for the home screen and apps on mobile devices) among other things.

Try it

HTML Demo: <link> RESET

HTML CSS

```
<link href="/media/examples/link-element-example.css" rel="stylesheet" />

<p>This text will be red as defined in the external stylesheet.</p>
<p style="color: blue">The <code>style</code> attribute can override it, though.</p>

```

To link an external stylesheet, you'd include a `<link>` element inside your `<head>` like this:

mdn web docs

```
<link href="main.css" rel="stylesheet" />
```

This example provides the path to the stylesheet inside an `href` attribute and a `rel` attribute with a value of `stylesheet`. The `rel` stands for "relationship", and is one of the key features of the `<link>` element — the value denotes how the item being linked to is related to the containing document.

There are a number of other common types you'll come across. For example, a link to the site's favicon:

HTML

```
<link rel="icon" href="favicon.ico" />
```

There are a number of other icon `rel` values, mainly used to indicate special icon types for use on various mobile platforms, e.g.:

HTML

```
<link
  rel="apple-touch-icon"
  sizes="114x114"
  href="apple-icon-114.png"
  type="image/png" />
```

The `sizes` attribute indicates the icon size, while the `type` contains the MIME type of the resource being linked. These provide useful hints to allow the browser to choose the most appropriate icon available.

You can also provide a media type or query inside a `media` attribute; this resource will then only be loaded if the media condition is true. For example:

HTML

```
<link href="print.css" rel="stylesheet" media="print" />
<link
  href="mobile.css"
  rel="stylesheet"
  media="screen and (max-width: 600px)" />
```

Some interesting new performance and security features have been added to the `<link>` element too. Take this example:

HTML

```
<link
  rel="preload"
  href="myFont.woff2"
  as="font"
  type="font/woff2"
  crossorigin="anonymous" />
```

A `rel` value of `preload` indicates that the browser should preload this resource (see `rel="preload"` for more details), with the `as` attribute indicating the specific class of content being fetched. The `crossorigin` attribute indicates whether the resource should be fetched with a CORS request.

Other usage notes:

- A `<link>` element can occur either in the `<head>` or `<body>` element, depending on whether it has a link type that is **body-ok**. For example, the `stylesheet` link type is body-ok, and therefore `<link rel="stylesheet">` is permitted in the body. However, this isn't a good practice to follow; it makes more sense to separate your `<link>` elements from your body content, putting them in the `<head>`.
- When using `<link>` to establish a favicon for a site, and your site uses a Content Security Policy (CSP) to enhance its security, the policy applies to the favicon. If you encounter problems with the favicon not loading, verify that the `Content-Security-Policy` header's `img-src` directive is not preventing access to it.
- The HTML and XHTML specifications define event handlers for the `<link>` element, but it is unclear how they would be used.
- Under XHTML 1.0, void elements such as `<link>` require a trailing slash: `<link />`.
- WebTV supports the use of the value `next` for `rel` to preload the next page in a document series.

Attributes

This element includes the global attributes.

`as`

This attribute is required when `rel="preload"` has been set on the `<link>` element, optional when `rel="modulepreload"` has been set, and otherwise should not be used. It specifies the type of content being loaded by the `<link>`, which is necessary for request matching, application of correct content security policy, and setting of correct `Accept` request header.

Furthermore, `rel="preload"` uses this as a signal for request prioritization. The table below lists the valid values for this attribute and the elements or resources they apply to.

Value	Applies To
audio	`<audio>` elements
document	`<iframe>` and `<frame>` elements
embed	`<embed>` elements
fetch	fetch, XHR **Note:** This value also requires `<link>` to contain the crossorigin attribute, see CORS-enabled fetches.
font	CSS @font-face **Note:** This value also requires `<link>` to contain the crossorigin attribute, see CORS-enabled fetches.
image	`<img>` and `<picture>` elements with srcset or imageset attributes, SVG `<image>` elements, CSS `*-image` rules
object	`<object>` elements
script	`<script>` elements, Worker `importScripts`
style	`<link rel=stylesheet>` elements, CSS `@import`
track	`<track>` elements
video	`<video>` elements
worker	Worker, SharedWorker

`blocking`

This attribute explicitly indicates that certain operations should be blocked on the fetching of an external resource. It must only be used when the `rel` attribute contains `expect` or `stylesheet` keywords. The operations that are to be blocked must be a space-separated list of blocking tokens listed below.

- `render` : The rendering of content on the screen is blocked.

`crossorigin`

This enumerated attribute indicates whether CORS must be used when fetching the resource. CORS-enabled images can be reused in the `<canvas>` element without being *tainted*. The allowed values are:

> `anonymous`
>
> A cross-origin request (i.e. with an `Origin` HTTP header) is performed, but no credential is sent (i.e. no cookie, X.509 certificate, or HTTP Basic authentication). If the server does not give credentials to the origin site (by not setting the `Access-Control-Allow-Origin` HTTP header) the resource will be tainted and its usage restricted.
>
> `use-credentials`
>
> A cross-origin request (i.e. with an `Origin` HTTP header) is performed along with a credential sent (i.e. a cookie, certificate, and/or HTTP Basic authentication is performed). If the server does not give credentials to the origin site (through `Access-Control-Allow-Credentials` HTTP header), the resource will be *tainted* and its usage restricted.

If the attribute is not present, the resource is fetched without a CORS request (i.e. without sending the `Origin` HTTP header), preventing its non-tainted usage. If invalid, it is handled as if the enumerated keyword **anonymous** was used. See CORS settings attributes for additional information.

`disabled`

For `rel="stylesheet"` only, the `disabled` Boolean attribute indicates whether the described stylesheet should be loaded and applied to the document. If `disabled` is specified in the HTML when it is loaded, the stylesheet will not be loaded during page load. Instead, the stylesheet will be loaded on-demand, if and when the `disabled` attribute is changed to `false` or removed.

Setting the `disabled` property in the DOM causes the stylesheet to be removed from the document's `Document.styleSheets` list.

`fetchpriority`

Provides a hint of the relative priority to use when fetching a resource of a particular type. Allowed values:

> `high`
>
> Fetch the resource at a high priority relative to other resources of the same type.
>
> `low`
>
> Fetch the resource at a low priority relative to other resources of the same type.
>
> `auto`
>
> Don't set a preference for the fetch priority. This is the default. It is used if no value or an invalid value is set.

See `HTMLLinkElement.fetchPriority` for more information.

`href`

This attribute specifies the URL of the linked resource. A URL can be absolute or relative.

`hreflang`

This attribute indicates the language of the linked resource. It is purely advisory. Allowed values are specified by RFC 5646: Tags for Identifying Languages (also known as BCP 47) . Use this attribute only if the `href` attribute is present.

`imagesizes`

For `rel="preload"` and `as="image"` only, the `imagesizes` attribute has similar syntax and semantics as the `sizes` attribute that indicates to preload the appropriate resource used by an `img` element with corresponding values for its `srcset` and `sizes` attributes.

`imagesrcset`

For `rel="preload"` and `as="image"` only, the `imagesrcset` attribute has similar syntax and semantics as the `srcset` attribute that indicates to preload the appropriate resource used by an `img` element with corresponding values for its `srcset` and `sizes` attributes.

`integrity`

Contains inline metadata — a base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch. The browser can use this to verify that the fetched resource has been delivered without unexpected manipulation. The attribute must only be specified when the `rel` attribute is specified to `stylesheet`, `preload`, or `modulepreload`. See Subresource Integrity.

`media`

This attribute specifies the media that the linked resource applies to. Its value must be a media type / media query. This attribute is mainly useful when linking to external stylesheets — it allows the user agent to pick the best adapted one for the device it runs on.

`referrerpolicy`

A string indicating which referrer to use when fetching the resource:

- `no-referrer` means that the `Referer` header will not be sent.
- `no-referrer-when-downgrade` means that no `Referer` header will be sent when navigating to an origin without TLS (HTTPS). This is a user agent's default behavior, if no policy is otherwise specified.
- `origin` means that the referrer will be the origin of the page, which is roughly the scheme, the host, and the port.
- `origin-when-cross-origin` means that navigating to other origins will be limited to the scheme, the host, and the port, while navigating on the same origin will include the referrer's path.
- `unsafe-url` means that the referrer will include the origin and the path (but not the fragment, password, or username). This case is unsafe because it can leak origins and paths from TLS-protected resources to insecure origins.

`rel`

This attribute names a relationship of the linked document to the current document. The attribute must be a space-separated list of link type values.

`sizes`

This attribute defines the sizes of the icons for visual media contained in the resource. It must be present only if the `rel` contains a value of `icon` or a non-standard type such as Apple's `apple-touch-icon`. It may have the following values:

- `any`, meaning that the icon can be scaled to any size as it is in a vector format, like `image/svg+xml`.
- a white-space separated list of sizes, each in the format `<width in pixels>x<height in pixels>` or `<width in pixels>X<height in pixels>`. Each of these sizes must be contained in the resource.

> **Note:** Most icon formats are only able to store one single icon; therefore, most of the time, the `sizes` attribute contains only one entry. Microsoft's ICO format and Apple's ICNS format can store multiple icon sizes in a single file. ICO has better browser support, so you should use this format if cross-browser support is a concern.

`title`

The `title` attribute has special semantics on the `<link>` element. When used on a `<link rel="stylesheet">` it defines a default or an alternate stylesheet.

`type`

This attribute is used to define the type of the content linked to. The value of the attribute should be a MIME type such as **text/html**, **text/css**, and so on. The common use of this attribute is to define the type of stylesheet being referenced (such as **text/css**), but given that CSS is the only stylesheet language used on the web, not only is it possible to omit the `type` attribute, but is actually now recommended practice. It is also used on `rel="preload"` link types, to make sure the browser only downloads file types that it supports.

Non-standard attributes

`target`

Defines the frame or window name that has the defined linking relationship or that will show the rendering of any linked resource.

Obsolete attributes

`charset`

This attribute defines the character encoding of the linked resource. The value is a space- and/or comma-delimited list of character sets as defined in RFC 2045 . The default value is `iso-8859-1` .

> **Note:** To produce the same effect as this obsolete attribute, use the `Content-Type` HTTP header on the linked resource.

`rev`

The value of this attribute shows the relationship of the current document to the linked document, as defined by the `href` attribute. The attribute thus defines the reverse relationship compared to the value of the `rel` attribute. Link type values for the attribute are similar to the possible values for `rel` .

> **Note:** Instead of `rev` , you should use the `rel` attribute with the opposite link type value. For example, to establish the reverse link for `made` , specify `author` . Also, this attribute doesn't stand for "revision" and must not be used with a version number, even though many sites misuse it in this way.

Examples

Including a stylesheet

To include a stylesheet in a page, use the following syntax:

HTML

```
<link href="style.css" rel="stylesheet" />
```

Providing alternative stylesheets

You can also specify alternative style sheets.

The user can choose which style sheet to use by choosing it from the **View > Page Style** menu. This provides a way for users to see multiple versions of a page.

HTML

```
<link href="default.css" rel="stylesheet" title="Default Style" />
<link href="fancy.css" rel="alternate stylesheet" title="Fancy" />
<link href="basic.css" rel="alternate stylesheet" title="Basic" />
```

Providing icons for different usage contexts

You can include links to several icons on the same page, and the browser will choose which one works best for its particular context using the `rel` and `sizes` values as hints.

HTML

```
<!-- iPad Pro with high-resolution Retina display: -->
<link
  rel="apple-touch-icon"
  sizes="167x167"
  href="/apple-touch-icon-167x167.png" />
<!-- 3x resolution iPhone: -->
```

```
<link
  rel="apple-touch-icon"
  sizes="180x180"
  href="/apple-touch-icon-180x180.png" />
<!-- non-Retina iPad, iPad mini, etc.: -->
<link
  rel="apple-touch-icon"
  sizes="152x152"
  href="/apple-touch-icon-152x152.png" />
<!-- 2x resolution iPhone and other devices: -->
<link rel="apple-touch-icon" href="/apple-touch-icon-120x120.png" />
<!-- basic favicon -->
<link rel="icon" href="/favicon.ico" />
```

For information about what `sizes` to choose for Apple icons, see Apple's documentation on configuring web applications and the referenced Apple human interface guidelines. Usually, it is sufficient to provide a large image, such as 192x192, and let the browser scale it down as needed, but you may want to provide images with different levels of detail for different sizes, as the Apple design guideline recommends. Providing smaller icons for lower resolutions also saves bandwidth.

It may not be necessary to provide `<link>` elements at all. For example, browsers automatically request `/favicon.ico` from the root of a site, and Apple also automatically requests `/apple-touch-icon-[size].png`, `/apple-touch-icon.png`, etc. However, providing explicit links protects you against changes to these conventions.

Conditionally loading resources with media queries

You can provide a media type or query inside a `media` attribute; this resource will then only be loaded if the media condition is true. For example:

HTML

```
<link href="print.css" rel="stylesheet" media="print" />
<link href="mobile.css" rel="stylesheet" media="all" />
<link
  href="desktop.css"
  rel="stylesheet"
  media="screen and (min-width: 600px)" />
<link
  href="highres.css"
  rel="stylesheet"
  media="screen and (min-resolution: 300dpi)" />
```

Stylesheet load events

You can determine when a style sheet has been loaded by watching for a `load` event to fire on it; similarly, you can detect if an error has occurred while processing a style sheet by watching for an `error` event:

HTML

```
<link rel="stylesheet" href="mystylesheet.css" id="my-stylesheet" />

<script>
  const stylesheet = document.getElementById("my-stylesheet");

  stylesheet.onload = () => {
    // Do something interesting; the sheet has been loaded
  };

  stylesheet.onerror = () => {
    console.log("An error occurred loading the stylesheet!");
```

```
  };
</script>
```

> **Note:** The `load` event fires once the stylesheet and all of its imported content has been loaded and parsed, and immediately before the styles start being applied to the content.

Preload examples

You can find a number of `<link rel="preload">` examples in Preloading content with `rel="preload"`.

Blocking rendering till a resource is fetched

You can include `render` token inside a `blocking` attribute; the rendering of the page will be blocked till the resource is fetched. For example:

HTML

```
<link blocking="render" rel="stylesheet" href="example.css" crossorigin />
```

Technical summary

Content categories	Metadata content. If `itemprop` is present: Flow content and phrasing content.
Permitted content	None; it is a void element.
Tag omission	Must have a start tag and must not have an end tag.
Permitted parents	Any element that accepts metadata elements. If itemprop is present: any element that accepts phrasing content.
Implicit ARIA role	`link` with `href` attribute
Permitted ARIA roles	No `role` permitted
DOM interface	`HTMLLinkElement`

Specifications

Specification
HTML Standard # the-link-element

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android
`link`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`as`	50	17	56	37	10	50	56	37	10	5.0	50
`blocking`	105	105	No	91	18.2	105	No	72	18.2	20.0	105
`charset`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`crossorigin`	34	17	18	21	10	34	18	21	10	2.0	37
`disabled`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`fetchpriority`	101	101	132	87	17.2	101	132	70	17.2	19.0	101
`href`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`hreflang`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`imagesizes`	73	79	78	60	17.2	73	79	52	17.2	11.0	73
`imagesrcset`	73	79	78	60	17.2	73	79	52	17.2	11.0	73
`integrity`	45	17	43	32	11.1	45	43	32	11.3	5.0	45
`media`	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
`referrerpolicy`	51	79	50	38	14	51	50	41	14	7.2	51
no-referrer-when-downgrade	51	79	50–91	38	No	51	50–91	41	No	7.2	51
origin-when-cross-origin	51	79	50–91	38	No	51	50–91	41	No	7.2	51
unsafe-url	51	79	50–91	38	No	51	50–91	41	No	7.2	51
`rel`	1	12	1	9	1	18	4	10.1	1	1.0	4.4
rel="alternate stylesheet"	1–47	No	3	15–34	No	18–47	4	14–34	No	1.0–4.2	4.4
rel=dns-prefetch	46	79	3	33	5	46	4	33	4.2	5.0	46
rel=expect	124	124	No	110	No	124	No	82	No	27.0	124
rel=icon	4	12	2	9	3.1	18	4	No	No	4.0	38
rel=manifest	39	79	No	26	17	39	58	26	11.3	4.0	39
rel=modulepreload	66	79	115	53	17	66	115	47	17	9.0	66

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android
rel=preconnect	46	79	39	33	11.1	46	39	33	11.3	4.0	46
rel=prefetch	8	12	2	15	13.1	18	4	14	13.4	1.5	4.4
rel=preload	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=fetch	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=font	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=image	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=script	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=style	50	79	85	37	11.1	50	85	37	11.3	5.0	50
as=track	50	79	No	37	11.1	50	No	37	11.3	5.0	50
rel=prerender	13	79	No	15	No	18	No	14	No	1.5	4.4
rev	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
sizes	80	80	72	67	6	80	79	57	6	13.0	80
target	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4
type	1	12	1	12.1	4	18	4	12.1	3.2	1.0	4.4

Tip: you can click/tap on a cell for more information.

Full support Partial support No support Experimental. Expect behavior to change in the future.

Non-standard. Check cross-browser support before using. Deprecated. Not for use in new websites. See implementation notes.

User must explicitly enable this feature. Has more compatibility info.

See also

- `Link` HTTP header

Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute.

This page was last modified on Nov 21, 2024 by MDN contributors.