URL: canParse() static method

- Syntax
  - Parameters
  - Return value
- Examples
- Specifications
- Browser compatibility
- See also

The **`URL.canParse()`** static method of the URL interface returns a boolean indicating whether or not an absolute URL, or a relative URL combined with a base URL, are parsable and valid.

This is a fast and easy alternative to constructing a `URL` within a try...catch block. It returns `true` for the same values for which the `URL()` constructor would succeed, and `false` for the values that would cause the constructor to throw.

Syntax

```
URL.canParse(url)
URL.canParse(url, base)
```

Parameters

- `url`
  - : A string or any other object with a stringifier — including, for example, an a or area element — that represents an absolute or relative URL. If `url` is a relative URL, `base` is required, and will be used as the base URL. If `url` is an absolute URL, a given

`base` will be ignored.

- `base`
  - : A string representing the base URL to use in cases where `url` is a relative URL. If not specified, it defaults to `undefined`.

> **NOTE:** The `url` and `base` arguments will each be stringified from whatever value you pass, just like with other Web APIs that accept a string. In particular, you can use an existing URL object for either argument, and it will be stringified to the object's href property.

Return value

`true` if the URL can be parsed and is valid; `false` otherwise.

Examples

This live example demonstrates how to use the `URL.canParse()` static method for a few different absolute and relative URL values.

The first part of the example defines an HTML `<pre>` element to log to, along with a logging method `log()`.

```
<pre id="log"></pre>
```

```
const logElement = document.getElementById("log");
function log(text) {
  logElement.innerText += `${text}\n`;
}
```

Next we check that the `URL.canParse()` method is supported using the condition `"canParse" in URL`. If the method is supported we log the result of checking an absolute URL, a relative URL with no base URL, and a relative URL with a valid base URL. We also log the case when `URL.canParse()` is not supported.

```
if ("canParse" in URL) {
  log("Test valid absolute URL");
  let url = "https://developer.mozilla.org/";
  let result = URL.canParse(url);
```

```
  log(` URL.canParse("${url}"): ${result}`);

  log("\nTest relative URL with no base URL");
  url = "/en-US/docs";
  result = URL.canParse(url);
  log(` URL.canParse("${url}"): ${result}`);

  log("\nTest relative URL with valid base URL");
  let baseUrl = "https://developer.mozilla.org/";
  result = URL.canParse(url, baseUrl);
  log(` URL.canParse("${url}","${baseUrl}"): ${result}`);
} else {
  log("URL.canParse() not supported");
}
```

Last of all, the code below shows that the `baseUrl` doesn't have to be a string. Here we have passed a `URL` object.

```
if ("canParse" in URL) {
  log("\nTest relative URL with base URL supplied as a URL object");
  let baseUrl = new URL("https://developer.mozilla.org/");
  let url = "/en-US/docs";
  result = URL.canParse(url, baseUrl);
  log(` URL.canParse("${url}","${baseUrl}"): ${result}`);
}
```

The results of each of the checks are shown below.

Specifications

Browser compatibility

See also

- URL()
- A polyfill of `URL.canParse()` is available in `core-js`

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:00 UTC