

# WebAssembly.instantiateStreaming()


 **Baseline** Widely available



The `WebAssembly.instantiateStreaming()` static method compiles and instantiates a WebAssembly module directly from a streamed underlying source. This is the most efficient, optimized way to load Wasm code.

**Note:** Webpages that have strict [Content Security Policy \(CSP\)](#) might block WebAssembly from compiling and executing modules. For more information on allowing WebAssembly compilation and execution, see the [script-src CSP](#).

## Syntax

JS 

```
WebAssembly.instantiateStreaming(source, importObject)
```

## Parameters

`source`

A [Response](#) object or a promise that will fulfill with one, representing the underlying source of a Wasm module you want to stream, compile, and instantiate.

`importObject` Optional

An object containing the values to be imported into the newly-created `Instance`, such as functions or [WebAssembly.Memory](#) objects. There must be one matching property for each declared import of the compiled module or else a [WebAssembly.LinkError](#) is thrown.

## Return value

A `Promise` that resolves to a `ResultObject` which contains two fields:

- `module`: A [WebAssembly.Module](#) object representing the compiled WebAssembly module. This `Module` can be instantiated again or shared via [postMessage\(\)](#).
- `instance`: A [WebAssembly.Instance](#) object that contains all the [Exported WebAssembly functions](#).

## Exceptions

- If either of the parameters are not of the correct type or structure, a [TypeError](#) is thrown.
- If the operation fails, the promise rejects with a [WebAssembly.CompileError](#), [WebAssembly.LinkError](#), or [WebAssembly.RuntimeError](#), depending on the cause of the failure.

## Examples

### Instantiating streaming

The following example (see our [instantiate-streaming.html](#) demo on GitHub, and [view it live](#) also) directly streams a Wasm module from an underlying source then compiles and instantiates it, the promise fulfilling with a `ResultObject`. Because the `instantiateStreaming()` function accepts a promise for a `Response` object, you can directly pass it a `fetch()` call, and it will pass the response into the function when it fulfills.

```
JS
const importObject = {
  my_namespace: { imported_func: (arg) => console.log(arg) },
};

WebAssembly.instantiateStreaming(fetch("simple.wasm"), importObject).then(
  (obj) => obj.instance.exports.exported_func(),
);
```

The `ResultObject`'s instance member is then accessed, and the contained exported function invoked.


















**Note:** For this to work, `.wasm` files should be returned with an `application/wasm` MIME type by the server.

## Specifications

Specification
<a href="#">WebAssembly Web API</a>
<a href="#"># dom-webassembly-instantiatestreaming</a>

## 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	 WebView on iOS	 Deno	 Node.js
<code>instantiateStreaming()</code> static method	✓ 61	✓ 16	✓ 58	✓ 47	✓ 15	✓ 61	✓ 58	✓ 45	✓ 15	✓ 8.0	✓ 61	✓ 15	✓ 1.12	✓ 18.1.0

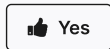
Tip: you can click/tap on a cell for more information.

✓ Full support

## See also

- [WebAssembly](#) overview page
- [WebAssembly concepts](#)
- [Using the WebAssembly JavaScript API](#)

Was this page helpful to you?



[Learn how to contribute.](#)

This page was last modified on Jul 26, 2024 by [MDN contributors](#).

