

- Quickstart
- Concepts
- Choosing a Runtime
 - Node.js
 - Go
 - Python
 - Ruby
 - Edge Runtime**
- Functions API Reference
- Configuring Functions
- Streaming Functions
- OG Image Generation
- Using WebAssembly
- Logs
- Limitations
- Usage & Pricing
- Edge Middleware
- Image Optimization
- Incremental Static Regeneration
- Data Cache
- Cron Jobs

Edge Runtime

Learn about the Edge runtime, the environment in which Edge Functions run.

Table of Contents ▾	🌐 Next.js (/app) ▾
---------------------	--------------------

Functions using the Edge Runtime, are ideal for a cost-effective option that use a limited set of Web Standard APIs that make sense on the server. The Edge runtime is built on top of the **V8 engine**, allowing it to run in isolated execution environments that don't require a container or virtual machine.

Functions using the Edge runtime execute in the data center closest to the user, or in a **region near your databases**. This can result in a much lower latency and allows you to provide personalization at speed.

Edge Functions run *after* the cache and so are ideal to be used on specific, dynamic parts of your site once the page is loaded. This response can be cached on the **Edge Network** making future invocations even faster.

See the quickstart to learn how to create your first function with the Edge runtime.

Supported APIs

The Edge runtime provides a subset of Web APIs such as `fetch`, `Request`, and `Response`. This lightweight API layer is built to be performant and execute code with minimal latency.

The following tables list the APIs that are available in the Edge runtime.

Network APIs

API	Description
<code>fetch</code>	Fetches a resource
<code>Request</code>	Represents an HTTP request

On this page

Supported APIs

- Network APIs
- Encoding APIs
- Stream APIs
- Crypto APIs
- Other Web Standard APIs

Check if you're running on the Edge runtime

Compatible Node.js modules

Unsupported APIs

Environment Variables

API	Description
<code>Response</code>	Represents an HTTP response
<code>Headers</code>	Represents HTTP headers
<code>FormData</code>	Represents form data
<code>File</code>	Represents a file
<code>Blob</code>	Represents a blob
<code>URLSearchParams</code>	Represents URL search parameters
<code>Blob</code>	Represents a blob
<code>Event</code>	Represents an event
<code>EventTarget</code>	Represents an object that can handle events
<code>PromiseRejectEvent</code>	Represents an event that is sent when a JavaScript Promise is rejected

Encoding APIs

API	Description
<code>TextEncoder</code>	Encodes a string into a Uint8Array
<code>TextDecoder</code>	Decodes a Uint8Array into a string
<code>atob</code>	Decodes a base-64 encoded string
<code>btoa</code>	Encodes a string in base-64

Stream APIs

API	Description
<code>ReadableStream</code>	Represents a readable stream
<code>WritableStream</code>	Represents a writable stream
<code>WritableStreamDefaultWriter</code>	Represents a writer for a writable stream
<code>TransformStream</code>	Represents a transform stream
<code>ReadableStreamDefaultReader</code>	Represents a reader for a readable stream
<code>ReadableStreamBYOBReader</code>	Represents a reader for a readable stream that supports BYOB

Crypto APIs

Edge Runtime	
API	Description
<code>crypto</code>	Provides access to the cryptographic functions
<code>SubtleCrypto</code>	Provides access to common cryptographic algorithms for signing, encryption or decryption
<code>CryptoKey</code>	Represents a cryptographic key

Other Web Standard APIs

API	Description
<code>AbortController</code>	Allows you to abort one or more operations
<code>AbortSignal</code>	Represents a signal object that is used to abort a request with a DOM request (such as a <code>fetch</code>)
<code>DOMException</code>	Represents an error that occurs in the DOM
<code>structuredClone</code>	Creates a deep copy of a value
<code>URLPattern</code>	Represents a URL pattern
<code>Array</code>	Represents an array of values
<code>ArrayBuffer</code>	Represents a generic, fixed-length array of bytes
<code>Atomics</code>	Provides atomic operations as specified by the WebAssembly specification
<code>BigInt</code>	Represents a whole number with arbitrary precision
<code>BigInt64Array</code>	Represents a typed array of 64-bit signed integers
<code>BigUint64Array</code>	Represents a typed array of 64-bit unsigned integers
<code>Boolean</code>	Represents a logical entity and can be either <code>true</code> or <code>false</code>
<code>clearInterval</code>	Cancels a timed, repeating action established by a call to <code>setInterval</code>
<code>clearTimeout</code>	Cancels a timed, repeating action established by a call to <code>setTimeout</code>
<code>console</code>	Provides access to the browser's console
<code>DataView</code>	Represents a generic view of an ArrayBuffer
<code>Date</code>	Represents a single moment in time in UTC format
<code>decodeURI</code>	Decodes a Uniform Resource Identifier (URI) created by <code>encodeURIComponent</code> or by a similar method

Edge Runtime	
API	Description
	routine
<code>encodeURIComponent</code>	Encodes a Uniform Resource Identifier instance of certain characters by escape sequences representing character
<code>encodeURIComponent</code>	Encodes a Uniform Resource Identifier replacing each instance of certain characters by one, two, three, or four escape sequences representing the character
<code>Error</code>	Represents an error when trying accessing a property
<code>EvalError</code>	Represents an error that occurs <code>eval()</code>
<code>Float32Array</code>	Represents a typed array of 32-bit floating point numbers
<code>Float64Array</code>	Represents a typed array of 64-bit floating point numbers
<code>Function</code>	Represents a function
<code>Infinity</code>	Represents the mathematical constant infinity
<code>Int8Array</code>	Represents a typed array of 8-bit signed integers
<code>Int16Array</code>	Represents a typed array of 16-bit signed integers
<code>Int32Array</code>	Represents a typed array of 32-bit signed integers
<code>Intl</code>	Provides access to internationalization functionality
<code>isFinite</code>	Determines whether a value is a finite number
<code>isNaN</code>	Determines whether a value is NaN
<code>JSON</code>	Provides functionality to convert JavaScript values to and from the JSON format
<code>Map</code>	Represents a collection of values where each value is only once
<code>Math</code>	Provides access to mathematical constants and functions
<code>Number</code>	Represents a numeric value
<code>Object</code>	Represents the object that is the prototype for all objects
<code>parseFloat</code>	Parses a string argument and returns a floating point value
<code>parseInt</code>	Parses a string argument and returns an integer value of the specified radix
<code>Promise</code>	Represents the eventual completion of an asynchronous operation, and its resulting value

Edge Runtime	
API	Description
<code>Proxy</code>	Represents an object that is use for fundamental operations (e.g. enumeration, function invocator
<code>RangeError</code>	Represents an error when a valu allowed values
<code>ReferenceError</code>	Represents an error when a non-referenced
<code>Reflect</code>	Provides methods for intercepta
<code>RegExp</code>	Represents a regular expression combinations of characters
<code>Set</code>	Represents a collection of value only once
<code>setInterval</code>	Repeatedly calls a function, with each call
<code>setTimeout</code>	Calls a function or evaluates an number of milliseconds
<code>SharedArrayBuffer</code>	Represents a generic, fixed-leng
<code>String</code>	Represents a sequence of chara
<code>Symbol</code>	Represents a unique and immut the key of an object property
<code>SyntaxError</code>	Represents an error when trying invalid code
<code>TypeError</code>	Represents an error when a valu
<code>Uint8Array</code>	Represents a typed array of 8-bi
<code>Uint8ClampedArray</code>	Represents a typed array of 8-bi to 0-255
<code>Uint32Array</code>	Represents a typed array of 32-
<code>URIError</code>	Represents an error when a glob used in a wrong way
<code>URL</code>	Represents an object providing s creating object URLs
<code>URLSearchParams</code>	Represents a collection of key/v
<code>WeakMap</code>	Represents a collection of key/v are weakly referenced
<code>WeakSet</code>	Represents a collection of objec occur only once
<code>WebAssembly</code>	Provides access to WebAssemb

Check if you're running on the Edge runtime

You can check if your function is running on the Edge runtime by checking the global `globalThis.EdgeRuntime` property. This can be helpful if you need to validate that your function is running on the Edge runtime in tests, or if you need to use a different API depending on the runtime.

```
1 if (typeof EdgeRuntime !== 'string') {
2   // dead-code elimination is enabled
3 }
```

Compatible Node.js modules

The following modules can be imported with and without the `node:` prefix when using the `import` statement:

Module	Description
<code>async_hooks</code>	Manage asynchronous resources lifecycle. Supports the <code>WinterCG</code> subset of APIs.
<code>events</code>	Facilitate event-driven programming with listeners. This API is fully supported.
<code>buffer</code>	Efficiently manipulate binary data using typed allocations with <code>Buffer</code> . Every primitive <code>Uint8Array</code> accepts <code>Buffer</code> too.
<code>assert</code>	Provide a set of assertion functions for <code>JavaScript</code> code.
<code>util</code>	Offer various utility functions where we <code>promisify</code> / <code>callbackify</code> and <code>types</code> .

Also, `Buffer` is globally exposed to maximize compatibility with existing Node.js modules.

Unsupported APIs

- The Edge runtime has some restrictions including:
- Some Node.js APIs other than the ones listed above **are not supported**. For example, you can't read or write to the filesystem
 - `node_modules` *can* be used, as long as they

– Calling `require` directly is **not allowed**. Use `import` instead

The following JavaScript language features are disabled, and **will not work**:

API	Description
<code>eval</code>	Evaluates JavaScript code
<code>new Function(evalString)</code>	Creates a new function with an argument
<code>WebAssembly.compile</code>	Compiles a WebAssembly module
<code>WebAssembly.instantiate</code>	Compiles and instantiates a WebAssembly module from a buffer source


While `WebAssembly.instantiate` is supported in Edge Runtime, it requires the Wasm source code to be provided using the import statement. This means you cannot use a buffer or byte array to dynamically compile the module at runtime.

Environment Variables

You can use `process.env` to access Environment Variables.


Last updated on July 16, 2024

Was this helpful?

Observability
Previews
Rendering
Security
Turbo
v0 

Guides
Help
Integrations
Pricing
Resources
Templates

Company

About
Blog
Careers
Changelog
Contact Us
Customers
Partners
Privacy Policy
Legal 

Social

 GitHub
 LinkedIn
 Twitter
 YouTube

All systems normal

