# Edge Runtime

The Next.js Edge Runtime is based on standard Web APIs, it supports the following APIs:

## Network APIs

| API | Description |
| --- | --- |
| `fetch` | Fetches a resource |
| `Request` | Represents an HTTP request |
| `Response` | Represents an HTTP response |
| `Headers` | Represents HTTP headers |
| `FetchEvent` | Represents a fetch event |
| `addEventListener` | Adds an event listener |
| `FormData` | Represents form data |
| `File` | Represents a file |
| `Blob` | Represents a blob |
| `URLSearchParams` | Represents URL search parameters |

## Encoding APIs

| API | Description |
| --- | --- |
| `TextEncoder` | Encodes a string into a Uint8Array |

| API | Description |
| --- | --- |
| `TextDecoder` | Decodes a Uint8Array into a string |
| `atob` | Decodes a base-64 encoded string |
| `btoa` | Encodes a string in base-64 |

## Stream APIs

| API | Description |
| --- | --- |
| `ReadableStream` | Represents a readable stream |
| `WritableStream` | Represents a writable stream |
| `WritableStreamDefaultWriter` | Represents a writer of a WritableStream |
| `TransformStream` | Represents a transform stream |
| `ReadableStreamDefaultReader` | Represents a reader of a ReadableStream |
| `ReadableStreamBYOBReader` | Represents a reader of a ReadableStream |

## Crypto APIs

| API | Description |
| --- | --- |
| `crypto` | Provides access to the cryptographic functionality of the platform |
| `SubtleCrypto` | Provides access to common cryptographic primitives, like hashing, signing, encryption or decryption |
| `CryptoKey` | Represents a cryptographic key |

## Web Standard APIs

| API | Description |
| --- | --- |
| `AbortController` | Allows you to abort one or more DOM requests as and when desired |

| API | Description |
| --- | --- |
| `DOMException` | Represents an error that occurs in the DOM |
| `structuredClone` | Creates a deep copy of a value |
| `URLPattern` | Represents a URL pattern |
| `Array` | Represents an array of values |
| `ArrayBuffer` | Represents a generic, fixed-length raw binary data buffer |
| `Atomics` | Provides atomic operations as static methods |
| `BigInt` | Represents a whole number with arbitrary precision |
| `BigInt64Array` | Represents a typed array of 64-bit signed integers |
| `BigUint64Array` | Represents a typed array of 64-bit unsigned integers |
| `Boolean` | Represents a logical entity and can have two values: `true` and `false` |
| `clearInterval` | Cancels a timed, repeating action which was previously established by a call to `setInterval()` |
| `clearTimeout` | Cancels a timed, repeating action which was previously established by a call to `setTimeout()` |
| `console` | Provides access to the browser's debugging console |
| `DataView` | Represents a generic view of an `ArrayBuffer` |
| `Date` | Represents a single moment in time in a platform-independent format |
| `decodeURI` | Decodes a Uniform Resource Identifier (URI) previously created by `encodeURI` or by a similar routine |
| `decodeURIComponent` | Decodes a Uniform Resource Identifier (URI) component previously created by `encodeURIComponent` or by a similar routine |
| `encodeURI` | Encodes a Uniform Resource Identifier (URI) by replacing each instance of certain characters by one, two, three, or four escape sequences representing the UTF-8 encoding of the character |
| `encodeURIComponent` | Encodes a Uniform Resource Identifier (URI) component by replacing each instance of certain characters by one, two, three, or four escape sequences representing the UTF-8 encoding of the character |
| `Error` | Represents an error when trying to execute a statement or accessing a property |
| `EvalError` | Represents an error that occurs regarding the global function `eval()` |
| `Float32Array` | Represents a typed array of 32-bit floating point numbers |
| `Float64Array` | Represents a typed array of 64-bit floating point numbers |

| API | Description |
| --- | --- |
| `Function` | Represents a function |
| `Infinity` | Represents the mathematical Infinity value |
| `Int8Array` | Represents a typed array of 8-bit signed integers |
| `Int16Array` | Represents a typed array of 16-bit signed integers |
| `Int32Array` | Represents a typed array of 32-bit signed integers |
| `Intl` | Provides access to internationalization and localization functionality |
| `isFinite` | Determines whether a value is a finite number |
| `isNaN` | Determines whether a value is `NaN` or not |
| `JSON` | Provides functionality to convert JavaScript values to and from the JSON format |
| `Map` | Represents a collection of values, where each value may occur only once |
| `Math` | Provides access to mathematical functions and constants |
| `Number` | Represents a numeric value |
| `Object` | Represents the object that is the base of all JavaScript objects |
| `parseFloat` | Parses a string argument and returns a floating point number |
| `parseInt` | Parses a string argument and returns an integer of the specified radix |
| `Promise` | Represents the eventual completion (or failure) of an asynchronous operation, and its resulting value |
| `Proxy` | Represents an object that is used to define custom behavior for fundamental operations (e.g. property lookup, assignment, enumeration, function invocation, etc) |
| `RangeError` | Represents an error when a value is not in the set or range of allowed values |
| `ReferenceError` | Represents an error when a non-existent variable is referenced |
| `Reflect` | Provides methods for interceptable JavaScript operations |
| `RegExp` | Represents a regular expression, allowing you to match combinations of characters |
| `Set` | Represents a collection of values, where each value may occur only once |
| `setInterval` | Repeatedly calls a function, with a fixed time delay between each call |
| `setTimeout` | Calls a function or evaluates an expression after a specified number of milliseconds |

| API | Description |
| --- | --- |
| `SharedArrayBuffer` | Represents a generic, fixed-length raw binary data buffer |
| `String` | Represents a sequence of characters |
| `Symbol` | Represents a unique and immutable data type that is used as the key of an object property |
| `SyntaxError` | Represents an error when trying to interpret syntactically invalid code |
| `TypeError` | Represents an error when a value is not of the expected type |
| `Uint8Array` | Represents a typed array of 8-bit unsigned integers |
| `Uint8ClampedArray` | Represents a typed array of 8-bit unsigned integers clamped to 0-255 |
| `Uint32Array` | Represents a typed array of 32-bit unsigned integers |
| `URIError` | Represents an error when a global URI handling function was used in a wrong way |
| `URL` | Represents an object providing static methods used for creating object URLs |
| `URLSearchParams` | Represents a collection of key/value pairs |
| `WeakMap` | Represents a collection of key/value pairs in which the keys are weakly referenced |
| `WeakSet` | Represents a collection of objects in which each object may occur only once |
| `WebAssembly` | Provides access to WebAssembly |

## Next.js Specific Polyfills

- `AsyncLocalStorage`

## Environment Variables

You can use `process.env` to access Environment Variables for both `next dev` and `next build`.

## Unsupported APIs

The Edge Runtime has some restrictions including:

- Native Node.js APIs **are not supported**. For example, you can't read or write to the filesystem.
- `node_modules` *can* be used, as long as they implement ES Modules and do not use native Node.js APIs.
- Calling `require` directly is **not allowed**. Use ES Modules instead.

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

| API | Description |
| --- | --- |
| `eval` ↗ | Evaluates JavaScript code represented as a string |
| `new Function(evalString)` ↗ | Creates a new function with the code provided as an argument |
| `WebAssembly.compile` ↗ | Compiles a WebAssembly module from a buffer source |
| `WebAssembly.instantiate` ↗ | Compiles and instantiates a WebAssembly module from a buffer source |

In rare cases, your code could contain (or import) some dynamic code evaluation statements which *can not be reached at runtime* and which can not be removed by treeshaking. You can relax the check to allow specific files with your Middleware or Edge API Route exported configuration:

```
export const config = {
  runtime: 'edge', // for Edge API Routes only
  unstable_allowDynamic: [
    // allows a single file
    '/lib/utilities.js',
    // use a glob to allow anything in the function-bind 3rd party module
    '/node_modules/function-bind/**',
  ],
};
```

`unstable_allowDynamic` is a glob ↗, or an array of globs, ignoring dynamic code evaluation for specific files. The globs are relative to your application root folder.

Be warned that if these statements are executed on the Edge, *they will throw and cause a runtime error*.