Web Serial API

Limited availability

Secure context: This feature is available only in secure contexts (HTTPS), in some or all supporting browsers.

Experimental: This is an experimental technology
Check the Browser compatibility table carefully before using this in production.

Note: This feature is available in Dedicated Web Workers.

The **Web Serial API** provides a way for websites to read from and write to serial devices. These devices may be connected via a serial port, or be USB or Bluetooth devices that emulate a serial port.

Concepts and Usage

The Web Serial API is one of a set of APIs that allow websites to communicate with peripherals connected to a user's computer. It provides the ability to connect to devices that are required by the operating system to communicate via the serial API, rather than USB which can be accessed via the WebUSB API, or input devices that can be accessed via WebHID API.

Examples of serial devices include 3D printers, and microcontrollers such as the BBC micro:bit board .

Interfaces

`Serial`

Provides attributes and methods for finding and connecting to serial ports from a web page.

`SerialPort`

Provides access to a serial port on the host device.

Extensions to other interfaces

`Navigator.serial` Read only

Returns a `Serial` object, which represents the entry point into the Web Serial API to enable the control of serial ports.

`WorkerNavigator.serial` Read only

Returns a `Serial` object, which represents the entry point into the Web Serial API to enable the control of serial ports.

HTTP headers

`Permissions-Policy` `serial` directive

Controls whether the current document is allowed to use the Web Serial API to communicate with serial devices, either directly connected via a serial port, or via USB or Bluetooth devices emulating a serial port.

Examples

The following examples demonstrate some of the functionality provided by the Web Serial API.

Checking for available ports

The following example shows how to check for available ports and allows the user to grant it permission to access additional ports.

The `connect` and `disconnect` events let sites react when a device is connected or disconnected from the system. The `getPorts()` method is then called to see connected ports that the site already has access to.

If the site doesn't have access to any connected ports it has to wait until it has user activation to proceed. In this example we use a `click` event handler on a button for this task. A filter is passed to `requestPort()` with a USB vendor ID in order to limit the set of devices shown to the user to only USB devices built by a particular manufacturer.

JS

```js
navigator.serial.addEventListener("connect", (e) => {
  // Connect to `e.target` or add it to a list of available ports.
});

navigator.serial.addEventListener("disconnect", (e) => {
  // Remove `e.target` from the list of available ports.
});

navigator.serial.getPorts().then((ports) => {
  // Initialize the list of available ports with `ports` on page load.
});

button.addEventListener("click", () => {
  const usbVendorId = 0xabcd;
  navigator.serial
    .requestPort({ filters: [{ usbVendorId }] })
    .then((port) => {
      // Connect to `port` or add it to the list of available ports.
    })
    .catch((e) => {
      // The user didn't select a port.
    });
});
```

Reading data from a port

The following example shows how to read data from a port. The outer loop handles non-fatal errors, creating a new reader until a fatal error is encountered and `SerialPort.readable` becomes `null`.

JS

```js
while (port.readable) {
  const reader = port.readable.getReader();
  try {
    while (true) {
      const { value, done } = await reader.read();
      if (done) {
        // |reader| has been canceled.
        break;
      }
      // Do something with |value|...
    }
  } catch (error) {
    // Handle |error|...
  } finally {
    reader.releaseLock();
  }
}
```

Specifications

Specification
Web Serial API # serial-interface

Browser compatibility

[Report problems with this compatibility data on GitHub](#)

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android
`Serial`	89	89	No	75	No	No	No
[`getPorts`](#)	89	89	No	75	No	No	No
[`requestPort`](#)	89	89	No	75	No	No	No
`allowedBluetoothServiceClassIds` option	117	117	No	103	No	No	No
`filters` `bluetoothServiceClassId` property	117	117	No	103	No	No	No

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

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

See implementation notes.

See also

- [Read from and write to a serial port](#)
- [Getting started with the Web Serial API](#)