



# JavaScript Geolocation

**Summary:** in this tutorial, you'll learn how to use the Geolocation API to allow web applications to access your location once you agree to share.

## What is Geolocation API

The Geolocation API allows the web application to access your location if you agree to share it.

## Why Geolocation API

The Geolocation API is useful for web applications that need to work based on the user's locations such as Search Engines, eCommerce websites, maps, and weather apps.

For example, an eCommerce website can request you to share your location. Once having the information, it can show you the product availability as well as prices and discounts based on your location.

Similarly, the search engine like Google.com can return local search results based on your search term and location.

For example, if you're in San Francisco and search for Pizza, Google will show you a list of Pizza restaurants that are near to your current location.

## The geolocation object

The Geolocation API is available through the `navigator.geolocation` object.

## Check if the Geolocation API is supported

To check if a browser supports it, you can test if the `geolocation` property exists on the navigator object like this:

```
if (!navigator.geolocation) {  
    console.error(`Your browser doesn't support Geolocation`);  
}
```

## Get the current position

To get the user's current location, you call the `getCurrentPosition()` method of the `navigator.geolocation` object.

The `getCurrentPosition()` method sends an asynchronous request to detect the user's location and queries the positioning hardware (like GPS) to get the up-to-date data.

The `getCurrentPosition()` can succeed or fail. It accepts two optional callbacks that will execute in each case respectively.

If the `getCurrentPosition()` succeeds, you'll get the user's position as an `GeolocationCoordinates` object.

The `GeolocationCoordinates` object has the `latitude` and `longitude` properties that represent a location.

## JavaScript Geolocation example

The following example shows a [simple application that has one button](#). When you click the button, the browser will request you to share your location. If you agree, it'll show your current latitude and longitude.

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>JavaScript Geolocation Demo</title>  
    <link rel="stylesheet" href="css/style.css">  
</head>  
<body>  
    <div class="container">  
        <button id="show" class="btn">Show my location</button>
```

```
    <div id="message"></div>
  </div>
  <script src="js/app.js"></script>
</body>
</html>
```

And the following shows the app.js file:

```
(( ) => {
  const message = document.querySelector('#message');

  // check if the Geolocation API is supported
  if (!navigator.geolocation) {
    message.textContent = `Your browser doesn't support Geolocation`;
    message.classList.add('error');
    return;
  }

  // handle click event
  const btn = document.querySelector('#show');
  btn.addEventListener('click', function () {
    // get the current position
    navigator.geolocation.getCurrentPosition(onSuccess, onError);
  });

  // handle success case
  function onSuccess(position) {
    const {
      latitude,
      longitude
    } = position.coords;

    message.classList.add('success');
    message.textContent = `Your location: (${latitude},${longitude})`;
  }

  // handle error case
  function onError() {
```

```
message.classList.add('error');
message.textContent = `Failed to get your location!`;
}
})();
```

How it works.

First, select the `#message` element by using the `querySelector()` method.

Next, show an error message if the browser doesn't support the Geolocation API.

Then, select the button and attach a click event handler. The click event handler calls the `getCurrentPosition()` method of the `navigator.geolocation` object to get the current position of the user.

The `onSuccess` and `onError` callback functions are for handling the success and error cases.

After that, define the `onSuccess()` function that gets the latitude and longitude from the `position.coords` and shows them on the `<div>` element with id `message`.

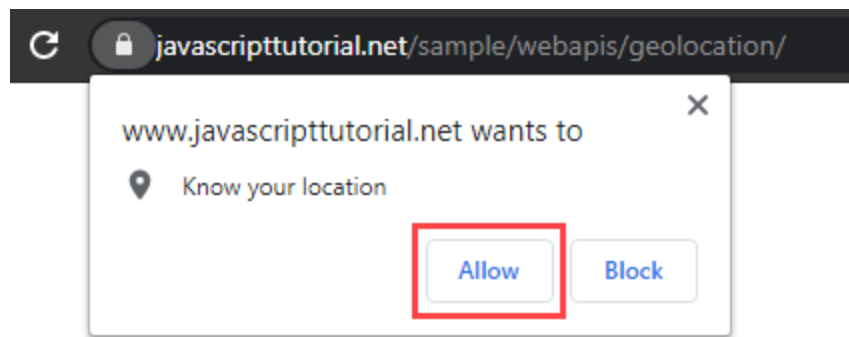
Finally, define the `onError()` function that shows an error message in case the `getCurrentPosition()` method fails.

Note that all the code in the `app.js` are wrapped in an [IIFE](#).

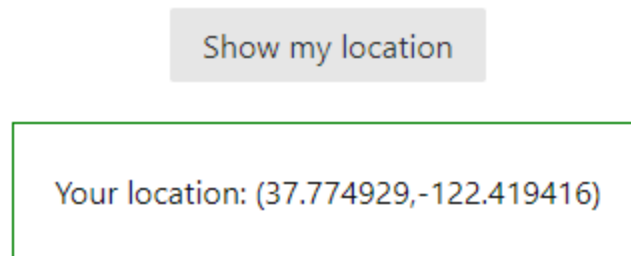
## Test the app

The following illustrates how to run the web application using Google Chrome. If you're using a different browser like Firefox, Edge, or Safari, you should be able to follow the steps.

When you click the **Show my location** button, the browser will request you to share your location like this:

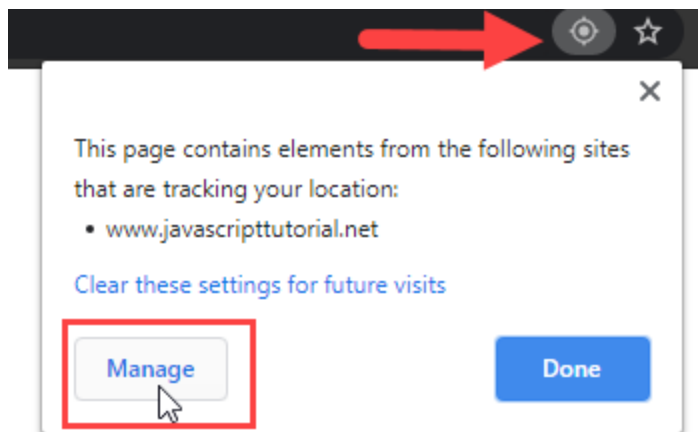


If you click the **Allow** button, you'll see your location as shown in the following picture:



Notice that your location may be different.

To change the browser setting, you click the **location** icon and the **Manage** button:



It'll show a screen like this:


Ask before accessing (recommended)



Block

No sites added

Allow

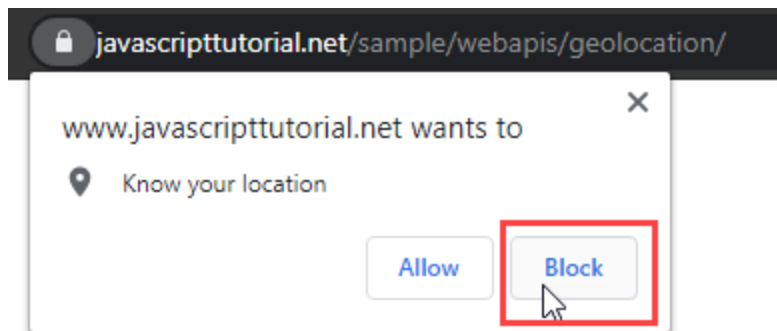
 <https://www.javascripttutorial.net:443>  
embedded on <https://www.javascripttutorial.net>



If you click the **Trash** icon, the allowed site will be removed.

And if you [refresh the app](#) and click the **Show my location** button, it'll request you to share your location again.

This time if you click the **Block** button:



...the app will show an error message:

Show my location

Failed to get your location!

# Summary

- The Geolocation API allows web applications to access the user's current position.
- Use the `navigator.geolocation.getCurrentPosition()` method to get the current user's location.