Glasshost
Programming Problems Solved

Chrome 73: Can no longer debug NodeJS with Dedicated DevTools for Node

May 20, 2023 by Glasshost

If you are using Chrome 73 or later versions, you might have noticed that you can no longer debug NodeJS applications using the Dedicated DevTools for Node. The Dedicated DevTools for Node is a built-in feature in Chrome that allows developers to debug NodeJS applications using the Chrome DevTools interface. However, with the release of Chrome 73, this feature has been removed.

So, what are your options for debugging NodeJS applications in Chrome 73 or later versions? You can still use Chrome DevTools to debug NodeJS applications, but you will need to use a different method.

Here's how you can debug a NodeJS application in Chrome 73 or later versions:

1. Start your NodeJS application with the `-inspect` flag. For example, if your NodeJS application is in a file called `app.js`, you would start it with the following command:

`node -inspect app.js`

This will start your NodeJS application and enable the Node.js debugger on port 9229.

1. Open Chrome and navigate to `chrome://inspect`. This will open the Chrome DevTools interface.

2. Under the “Remote Target” section, you should see your NodeJS application listed. Click the “inspect” button to open the DevTools interface for your NodeJS application.

3. You can now debug your NodeJS application using the DevTools interface.

Here’s an example of how you can use the Chrome DevTools interface to debug a simple NodeJS application:

```
// app.js
function add(a, b) {
  return a + b;
}

console.log(add(2, 3));
```

1. Start your NodeJS application with the `–inspect` flag:

`node –inspect app.js`

1. Open Chrome and navigate to `chrome://inspect`.

2. Under the “Remote Target” section, you should see your NodeJS application listed. Click the “inspect” button to open the DevTools interface for your NodeJS application.

3. In the DevTools interface, click on the “Sources” tab.

4. Navigate to the file that you want to debug (in this case, `app.js`).

5. Add a breakpoint by clicking on the line number where you want to pause the execution of your code.

6. Refresh your NodeJS application by executing the command `rs` in the console.

7. Once the breakpoint is hit, you can inspect the values of variables and step through your code using the DevTools interface.

In conclusion, while the Dedicated DevTools for Node has been removed in Chrome 73, you can still use Chrome DevTools to debug your NodeJS applications. Simply start your application with the `-inspect` flag and navigate to `chrome://inspect` to open the DevTools interface.

NODE.JS

Categorical bubble plot for mapping studies

Firestore to query by an array's field value

Leave a Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

Search

Search

.HTACCESS

.NET

.NET-CORE

AJAX

ALGORITHM

AMAZON-S3

AMAZON-WEB-SERVICES

ANDROID

ANDROID-FRAGMENTS

ANDROID-LAYOUT

ANDROID-STUDIO

Newsletter

Name (required)

Email (required)

BY SUBMITTING YOUR INFORMATION, YOU'RE GIVING US PERMISSION TO EMAIL YOU. YOU MAY UNSUBSCRIBE AT ANY TIME.

Subscribe

Latest Posts

Automatic resizing of the Windows Forms controls

New Flutter Project wizard not showing on Android Studio 3.0.1

Choosing the right API Level for my android application

How do I use Hamcrest with JUnit 5 when JUnit 5 doesn't have an assertThat() function?

$(document).ready(function() VS $(function(){

Links

Home

Categories

About us

Privacy Policy

Contact Us