IoT MQTT

Implementing MQTT in JavaScript

by **HiveMQ Team** MAR 5, 2024 12 min read

Table of Contents

Implementing MQTT in JavaScript

How to Use an MQTT JavaScript Client?

How to Configure an MQTT Broker in JavaScript?

Code Examples of Using MQTT in JavaScript

MQTT Security Best Practices While Using JavaScript MQTT Broker

Use Case Examples for MQTT in JavaScript

Conclusion

JavaScript can be used to implement **MQTT**, enabling efficient data exchange in IoT systems. MQTT supports real-time communication with minimal resources. Its publish/subscribe model uses less code and data than other communication protocols. JavaScript is one of the most popular programming languages and is one of the technologies the internet is built on. Web browsers, along with some servers and other applications, use JavaScript engines. Users can connect to an MQTT broker, publish data, and subscribe to topics and receive data, all using a JavaScript MQTT client.

How to Use an MQTT JavaScript Client?

The most common environment for running JavaScript outside of a web browser is Node.js. Node.js is lightweight and can run on devices with limited resources, which makes it a good fit for IoT projects. It also has tools and libraries to support its use in IoT applications. Node.js can be used for building scalable, high-performance applications and can be used with MQTT.

There are two main popular JavaScript MQTT clients, **MQTT.js and Paho JavaScript**, and they are both open source.

MQTT.js MQTT client supports both Node.js and web browsers. It allows users to connect to a broker, publish messages, and subscribe to topics. The community that uses it is large and active. MQTT.js supports both CommonJS and ES6 systems and also supports MQTT over WebSockets in both the browser and in Node.js.

The Paho JavaScript MQTT client is supported by the Eclipse Foundation. It is web browser based and uses WebSockets to connect to brokers. Both the Eclipse Paho website and **HiveMQ's WebSocket Client** allow users to implement the Paho JavaScript client using a graphical user interface. This client library is popular for

its stability, and because it uses WebSockets, programmers don't need to deal with the possibility of port 1833 being blocked.

JavaScript MQTT client libraries let users write applications to connect to a broker, publish messages, and subscribe to topics to receive data. **MQTT's pub/sub architecture** means that when devices subscribe to a topic, they receive new data automatically once it comes in and they don't need to regularly poll a server to check if new data is available yet. The most popular JavaScript MQTT client library is MQTT.js and that's what we'll be focusing on in this post, but HiveMQ also supports Paho JavaScript and our blog has a great **tutorial on Paho JS client here**.

To get started with MQTT.js, first install Node.js.

Then install MQTT.js.

```
npm install mqtt
```

Then, create a client instance.

```
import mqtt from 'mqtt'; const client = mqtt.connect('mqtt://broker.hivemq.com');
```

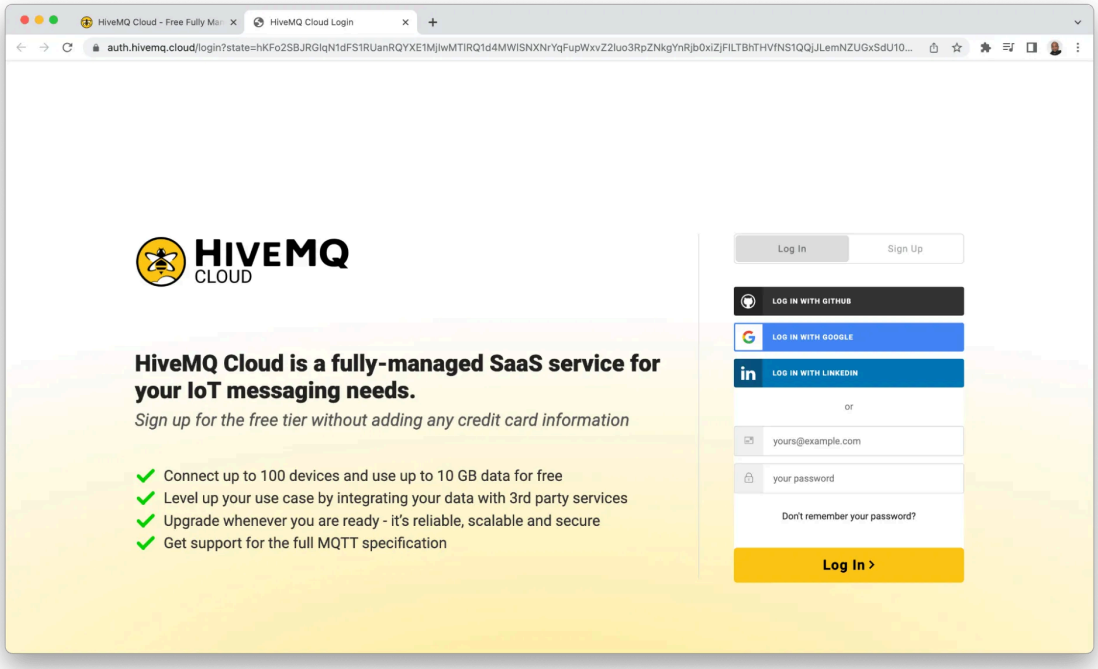
How to Configure an MQTT Broker in JavaScript?

Brokers are what make MQTT so efficient. Every device only needs to open one connection with a broker to share data with numerous other devies. HiveMQ's MQTT broker can support up to 200 million connections. HiveMQ has a hosted option, HiveMQ Cloud Serverless, as well as an open-source option, HiveMQ CE.

To set up a **HiveMQ Cloud Serverless MQTT broker** to handle messages to and from your Javascript IoT application, select **'HiveMQ Cloud'** from our website mega menu under ***Platform***.

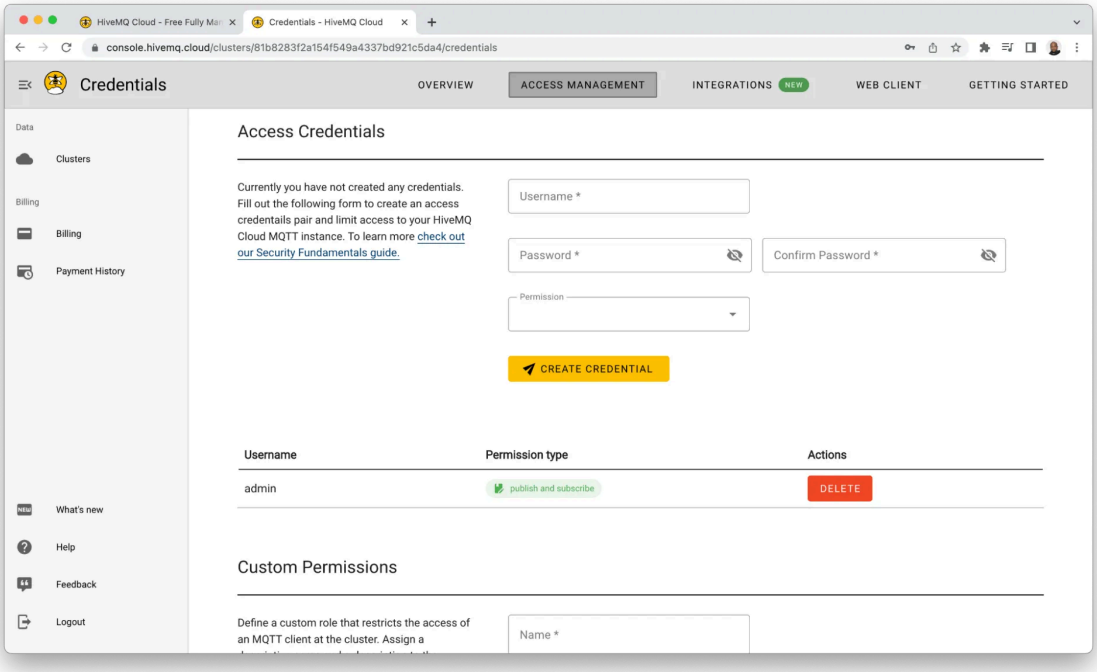
A **free HiveMQ Cloud** account allows you to create two MQTT broker clusters and connect up to 100 devices. On the HiveMQ Cloud page, click *'Try out for free'* button.

If you are accessing the HiveMQ Cloud portal for the first time, you must provide an email address and password and follow the simple steps to confirm your email and create your account.



HiveMQ Cloud portal

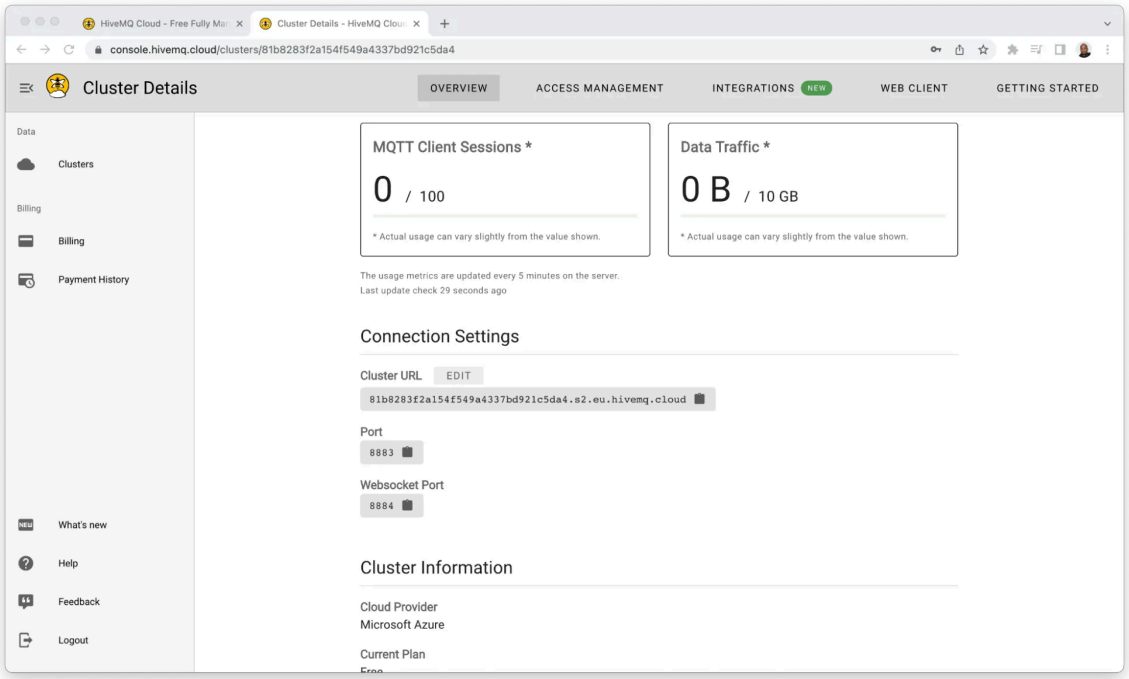
An **MQTT broker** cluster will automatically be created when you complete your account setup. Establish your credentials to allow devices to connect to your MQTT broker.



HiveMQ Cloud Broker Cluster

You can do that by specifying the username and password and clicking on *ADD*.

Next, clicking on *'Overview'* will provide the connection settings needed to connect your Javascript application to the HiveMQ Cloud Serverless MQTT broker cluster. So you need to copy the cluster URL, port number, and take note of your username and password for use in your code.



HiveMQ Cloud Broker Cluster Details

If you wish to update your connection credentials, you can always log in to your portal, click 'MANAGE CLUSTER,' and select 'ACCESS MANAGEMENT.'

To connect to HiveMQ with authentication, you can use the following code to create a client instance.

```
import mqtt from 'mqtt';
const client = mqtt.connect('mqtt://hostname', {
username: 'my-username',
password: 'my-password'
});
```

If you want to use MQTT.js to make a WebSocket connection, you can also specify that when you create a client instance with the following code.

```
const client = mqtt.connect('ws://hostname:8000');
```

After connecting to a broker, you can begin publishing messages and subscribing to topics to share data between devices.

Code Examples of Using MQTT in JavaScript

Publishing MQTT Messages in JavaScript

```
client.publish('cluster/messages/node7', 'Hello, HiveMQ!');
```

Subscribing MQTT Messages in JavaScript

```
topic.client.subscribe('ma/boston/sensors/tower1');
```

The code examples in this post come from **The Ultimate Guide on How to Use MQTT with Node.js** and there are many more examples there.

MQTT Security Best Practices While Using JavaScript MQTT Broker

It's important to follow security practices when you're implementing MQTT with JavaScript because MQTT is designed to share data between IoT devices as efficiently as possible, not as securely as possible. Considering security is vital in IoT use cases in general as well, because IoT devices are likely to have minimal computing capabilities and memory capacity. Using a secure network or VPN is one way to add security to your MQTT applications. You can also use the popular security standard SSL/TLS for transport level encryption with MQTT and JavaScript. This will encrypt your data while it is being transmitted and verify identity on both sides. To use SSL/TLS with MQTT.js, you need to import the fs module and specify your protocol as mqtts.

The following code is an example of implementing MQTT.js with security measures.

```
import fs from 'fs';
import mqtt from 'mqtt';

const options = {
protocol: 'mqtts',
host: 'broker.hivemq.com',
port: 8883,
ca: [fs.readFileSync('/path/to/ca.crt')],
cert: fs.readFileSync('/path/to/client.crt'),
key: fs.readFileSync('/path/to/client.key'),
};

const client = mqtt.connect(options);
```

What's your UNS maturity level? Get a custom report: **Take the UNS Maturity Assessment**

Use Case Examples for MQTT in JavaScript

MQTT can be used with JavaScript for IoT use cases. JavaScript is often used to build web-based applications and may be used with MQTT for remote monitoring, **home automation**, telemetry, and sensor networks. Any application that requires a lightweight and efficient messaging system is a good candidate for MQTT. JavaScript can be used with MQTT to share data between devices in real-time with limited bandwidth. For example, you could write a JavaScript application to subscribe to temperature data published to an MQTT broker and automate a smart home's air conditioning system to turn on if the temperature in a room rises above a certain threshold.

Conclusion

Developers can use MQTT with JavaScript to create IoT applications and efficiently share data in real time. When deciding how to build an application, consider your specific requirements, constraints, and project objectives to make an informed choice about using MQTT in your JavaScript projects. Click here to learn more about **MQTT Client libraries**.

FAQs on Implementing MQTT in JavaScript

Can JavaScript be used for real-time applications with MQTT?

What are the advantages of using MQTT over HTTP in JavaScript?

Can you use MQTT with JavaScript without extensions?

How do you install MQTT.js?

You can install MQTT.js in Node.js with the code

```
npm install mqtt
```

How do you install Paho JavaScript?

You can download Paho JavaScript from the Eclipse Foundation and include it in your HTML files with the script tags

```
<script src="mqttws31.js"> </script>
```

Read our blog **Implementing MQTT in Java** to learn how to set up an MQTT client and a broker connection in Java

Read our blog **Implementing MQTT in C** to learn how to set up an MQTT client and a broker connection in C.

Read our blog **Implementing MQTT in C#** to learn how to set up an MQTT client and a broker connection in C#.

Read our blog **Implementing MQTT in Python** to learn how to set up an MQTT client and a broker connection in Python.

HiveMQ Team

The HiveMQ team loves writing about **MQTT**, **Sparkplug**, **Unified Namespace (UNS)**, Industrial IoT protocols, IoT Data Streaming, how to deploy our platform, and more. We focus on industries ranging from energy, to transportation and logistics, to automotive manufacturing. Our experts are here to help, **contact us** with any questions.

Related content:

Why Embracing Cloud-Native IoT is a Business Imperative for Enterprise Architecture

Discover why cloud native IoT is essential for enterprise architecture to boost agility scalability and innovation in your digital transformation journey.

Blog

A Step-by-Step Guide to Connecting Ignition to MQTT and HiveMQ

Learn how to connect Ignition to MQTT and HiveMQ to enable secure, real-time industrial data streaming and bridge OT and IT systems with ease.

Blog

A Guide to Distributed Tracing for IoT Systems Using HiveMQ and OpenTelemetry

Gain deep IoT insights by implementing distributed tracing for MQTT workloads using HiveMQ, OpenTelemetry, and Datadog to boost observability.

Blog

Newsletter sign up

Enter Your Email-Address*

Submit

By clicking the subscribe button you give your consent to the use of your data according to our **Privacy Policy**. You can withdraw your consent at any time with future effect.

Company

About us

Careers

Partners

Security & Trust

Support

Services

Learn

Blog

Resources

Community

Certification

Documentation

Case Studies

Get Started

Free HiveMQ Cloud

Products

HiveMQ Cloud

Try on Docker

Try on AWS

Download Trial

Open Source

HiveMQ Enterprise

HiveMQ Platform

HiveMQ Edge

HiveMQ Data Hub

HiveMQ Extensions

MQTT

MQTT Essentials

MQTT FAQs

MQTT 5

MQTT Sparkplug

UNS Essentials

Public MQTT Broker

© 2025 HiveMQ

Legal note

Legal

Privacy Policy

PGP