13 June 2020

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 13 June 2020 | Name: | Srinidhi J C |
| Course: | Technical talk | USN: | 4al16ec078 |
| Topic: | Semiconductor as a Profession | Semester & Section: | 8th & b |
| Github Repository: | SrinidhiJC078 |  |  |

|  |
| --- |
| FORENOON SESSION DETAILS |
| Image of session |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date: | 13 June 2020 | Name: | Srinidhi J C | |
| Course: | IOT in Python with Rosberry Pi | USN: | 4al16ec078 | |
| Topic: | Security and Communication Protocols on IoT | Semester & Section: | 8th & b | |
| AFTERNOON SESSION DETAILS | | | |
| Image of session | | | |

Report:

WebSockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers. Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a **send()** method, and receive data from server to browser by an **onmessage** event handler.

Following is the API which creates a new WebSocket object.

var Socket = new WebSocket(url, [protocal] );

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

## WebSocket Attributes

Following are the attribute of WebSocket object. Assuming we created Socket object as mentioned above −

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **Socket.readyState**  The readonly attribute readyState represents the state of the connection. It can have the following values −   * A value of 0 indicates that the connection has not yet been established. * A value of 1 indicates that the connection is established and communication is possible. * A value of 2 indicates that the connection is going through the closing handshake. * A value of 3 indicates that the connection has been closed or could not be opened. |
| 2 | **Socket.bufferedAmount**  The readonly attribute bufferedAmount represents the number of bytes of UTF-8 text that have been queued using send() method. |

## WebSocket Events

Following are the events associated with WebSocket object. Assuming we created Socket object as mentioned above −

|  |  |  |
| --- | --- | --- |
| **Event** | **Event Handler** | **Description** |
| open | Socket.onopen | This event occurs when socket connection is established. |
| message | Socket.onmessage | This event occurs when client receives data from server. |
| error | Socket.onerror | This event occurs when there is any error in communication. |
| close | Socket.onclose | This event occurs when connection is closed. |

## WebSocket Methods

Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **Socket.send()**  The send(data) method transmits data using the connection. |
| 2 | **Socket.close()**  The close() method would be used to terminate any existing connection. |

## WebSocket Example

A WebSocket is a standard bidirectional TCP socket between the client and the server. The socket starts out as a HTTP connection and then "Upgrades" to a TCP socket after a HTTP handshake. After the handshake, either side can send data. **MQTT** stands for **MQ** Telemetry Transport but previously was known as Message Queuing Telemetry Transport. **MQTT** is fast becoming one of the main protocols for **IOT** (internet of things) deployments.

### MQTT Versions

There are two different variants of MQTT and several versions.

* MQTT v3.1.0 –
* MQTT v3.1.1 – In Common Use
* MQTT v5 – Currently Limited use
* **MQTT-SN** – See notes later

The original **MQT**T which was designed in 1999 and has been in use for many years and is designed for **TCP/IP networks**.

MQTTv3.1.1 is version in common use.

There is very little difference between v3.10 and 3.1.1. Here is a [Github page](https://github.com/mqtt/mqtt.github.io/wiki/Differences-between-3.1.0-and-3.1.1) detailing the main differences. Here is the actual Specification [MQTT V3.1](http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf) and here is a more detailed overview of the [MQTT protocol packet structure,.](http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/) The latest MQTT version(v5) ,[has now been approved](https://www.oasis-open.org/news/announcements/mqtt-v5-0-is-an-approved-oasis-committee-specification) (Jan 2018).. You can download the specification [here](http://docs.oasis-open.org/mqtt/mqtt/v5.0/cs01/mqtt-v5.0-cs01.pdf). If you are wondering what happened to 4 then see [here](http://www.eclipse.org/community/eclipse_newsletter/2016/september/article3.php). For More Information see [MQTT v 5.0 New Features Overview](http://www.steves-internet-guide.com/mqttv5/). Here is a [Github page](https://github.com/mqtt/mqtt.github.io/wiki/Differences-between-3.1.1-and-5.0) detailing the main differences between MQTT v3.1.1 and MQTT v5.

### MQTT Version 5 Support Notes:

As of **release 1.6** the mosquitto broker supports MQTT v5 in addition to MQTT v3.11. You can continue to use older version 3.11 client with the latest broker. Currently there is limited client support for v5.

**MQTT-SN** which was specified in around 2013, and designed to work over **UDP**, ZigBee and other transports.

**MQTT-SN** doesn’t currently appear to be very popular. and the specification hasn’t changed for several years, but I expect that to change as **IOT** deployments start. See [MQTT-SN working Notes](http://www.steves-internet-guide.com/mqtt-sn/). for more details on MQTT-SN.

### MQTT Clients

Because MQTT clients don’t have addresses like email addresses, phone numbers etc. you don’t need to assign addresses to clients like you do with most messaging systems. For MQTTv3.1.1 there is client software available in almost all programming languages and for the main operating systems Linux, Windows, Mac from the [Eclipse Paho project.](https://eclipse.org/paho/downloads.php)

* [Paho Python client.](http://www.steves-internet-guide.com/into-mqtt-python-client/)
* [Node.js MQTT Client-Starting Guide](http://www.steves-internet-guide.com/using-node-mqtt-client/)

For MQTTv5.0 there is limited support for client software from Eclipse, Currently there is only a C client available. Here is a [link](https://www.eclipse.org/paho/downloads.php) to the client comparison chart and download page. Currently I’m using a Python client from this [github page](https://github.com/wialon/gmqtt) that support v5.

### MQTT Brokers or Servers

There are many MQTT brokers available that you can use for testing and for real applications. There are free self hosted brokers , the most popular being [Mosquitto](https://mosquitto.org/) and commercial ones like [HiveMQ.](http://www.hivemq.com/)

**Mosquitto** is a free open source MQTT broker that runs on Windows and Linux.

If you don’t want to install and manage your own broker you can use a cloud based broker. [Eclipse](http://iot.eclipse.org/getting-started) has a free public MQTT broker and COAP server that you can also use for testing.

See the [MQTT Brokers and Servers](http://www.steves-internet-guide.com/mqtt-hosting-brokers-and-servers/) article for a list of hosting options. and the [Mosquitto page](http://www.steves-internet-guide.com/mosquitto-broker/) for more details on the mosquitto broker

### MQTT Over WebSockets

**Websockets** allows you to receive MQTT data directly into a web browser. This is important as the web browser may become the DE-facto interface for displaying MQTT data. MQTT websocket support for web browsers is provided by the **Javascript MQTT Client**.

### MQTT Security

MQTT supports various authentications and data security mechanisms. It is important to note that these security mechanisms are configured on the MQTT broker, and it is up to the client to comply with the mechanisms in place.

SSL stands for Secure Sockets Layer and, in short, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals from reading and modifying any information transferred, including potential personal details. The two systems can be a server and a client (for example, a shopping website and browser) or server to server (for example, an application with personal identifiable information or with payroll information).

It does this by making sure that any data transferred between users and sites, or between two systems remain impossible to read. It uses encryption algorithms to scramble data in transit, preventing hackers from reading it as it is sent over the connection. This information could be anything sensitive or personal which can include credit card numbers and other financial information, names and addresses.

TLS (Transport Layer Security) is just an updated, more secure, version of SSL. We still refer to our security certificates as SSL because it is a more commonly used term, but when you are [buying SSL](https://www.websecurity.symantec.com/ssl-certificate?inid=infoctr_buylink_sslhome) from DigiCert you are actually buying the most up to date TLS certificates with the option of [ECC, RSA or DSA encryption](https://www.websecurity.symantec.com/security-topics/how-ssl-works).

HTTPS (Hyper Text Transfer Protocol Secure) appears in the URL when a website is secured by an SSL certificate. The details of the certificate, including the issuing authority and the corporate name of the website owner, can be viewed by clicking on the lock symbol on the browser bar.

PubNub is a Realtime Communication Platform and realtime infrastructure-as-a-service company based in San Francisco, California. The company makes products for software and hardware developers to build realtime web, mobile, and Internet of Things applications.