# WebSockets - Send & Receive Messages

The **Message** event takes place usually when the server sends some data. Messages sent by the server to the client can include plain text messages, binary data, or images. Whenever data is sent, the **onmessage** function is fired.

This event acts as a client's ear to the server. Whenever the server sends data, the **onmessage** event gets fired.

The following code snippet describes opening the connection of Web Socket protocol.

```
connection.onmessage = function(e){
   var server_message = e.data;
   console.log(server_message);
}
```

It is also necessary to take into account what kinds of data can be transferred with the help of Web Sockets. Web socket protocol supports text and binary data. In terms of Javascript, **text** refers to as a string, while binary data is represented like **ArrayBuffer**.

Web sockets support only one binary format at a time. The declaration of binary data is done explicitly as follows −

```
socket.binaryType = ”arrayBuffer”;
socket.binaryType = ”blob”;
```

## Strings

Strings are considered to be useful, dealing with human readable formats such as XML and JSON. Whenever **onmessage** event is raised, client needs to check the data type and act accordingly.

The code snippet for determining the data type as String is mentioned below −

```
socket.onmessage = function(event){

   if(typeOf event.data === String ) {
      console.log(“Received data string”);
   }
}
```

## JSON (JavaScript Object Notation)

It is a lightweight format for transferring human-readable data between the computers. The structure of JSON consists of key-value pairs.

### Example

```
{
   name: “James Devilson”,
   message: “Hello World!”
}
```

The following code shows how to handle a JSON object and extract its properties −

```
socket.onmessage = function(event) {
   if(typeOf event.data === String ){
      //create a JSON object
      var jsonObject = JSON.parse(event.data);
      var username = jsonObject.name;
      var message = jsonObject.message;

      console.log(“Received data string”);
   }
}
```

## XML

Parsing in XML is not difficult, though the techniques differ from browser to browser. The best method is to parse using third party library like jQuery.

In both XML and JSON, the server responds as a string, which is being parsed at the client end.

## ArrayBuffer

It consists of a structured binary data. The enclosed bits are given in an order so that the position can be easily tracked. ArrayBuffers are handy to store the image files.

Receiving data using ArrayBuffers is fairly simple. The operator **instanceOf** is used instead of equal operator.

The following code shows how to handle and receive an ArrayBuffer object −

```
socket.onmessage = function(event) {
   if(event.data instanceof ArrayBuffer ){
      var buffer = event.data;
      console.log(“Received arraybuffer”);
   }
}
```

## Demo Application

The following program code shows how to send and receive messages using Web Sockets.

```
<!DOCTYPE html>
<html>
   <meta charset = "utf-8" />
   <title>WebSocket Test</title>

   <script language = "javascript" type = "text/javascript">
      var wsUri = "ws://echo.websocket.org/";
      var output;

      function init() {
         output = document.getElementById("output");
         testWebSocket();
      }

      function testWebSocket() {
         websocket = new WebSocket(wsUri);

         websocket.onopen = function(evt) {
            onOpen(evt)
         };

         websocket.onmessage = function(evt) {
            onMessage(evt)
         };

         websocket.onerror = function(evt) {
            onError(evt)
         };
      }

      function onOpen(evt) {
         writeToScreen("CONNECTED");
         doSend("WebSocket rocks");
      }

      function onMessage(evt) {
         writeToScreen('<span style = "color: blue;">RESPONSE: ' +
            evt.data+'</span>'); websocket.close();
      }

      function onError(evt) {
         writeToScreen('<span style="color: red;">ERROR:</span> ' + evt.data);
      }
```

```
      function doSend(message) {
         writeToScreen("SENT: " + message); websocket.send(message);
      }

      function writeToScreen(message) {
         var pre = document.createElement("p");
         pre.style.wordWrap = "break-word";
         pre.innerHTML = message; output.appendChild(pre);
      }

      window.addEventListener("load", init, false);

   </script>

   <h2>WebSocket Test</h2>
   <div id = "output"></div>

</html>
```

The output is shown below.