# WebSocket communications

This chapter explores bidirectional communications using WebSockets.

### Lesson 1: Communicating by using WebSocket

**Recap**: Long polling is where the client asks the server for information, and the server keeps the connection open until information is found.

**WebSocket**: provides a standard way for the server to send messages to the client without first being asked by the client, and vice versa

### Defining the WebSocket API

At the heart of the WebSocket API is the WebSocket Object, which is defined on the browser's window object. *You can easily test if the WebSocket Object exists to determine whether a browser supports WebSockets or not.*

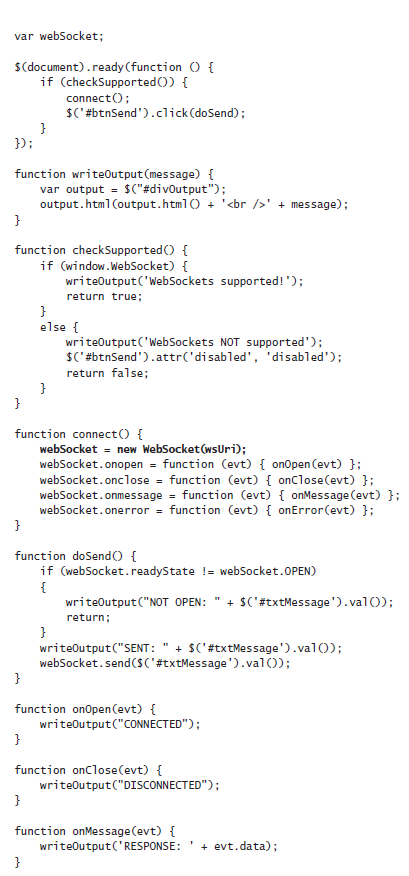
The WebSocket object contains the following members:

* **WebSocket** constructor A method that requires a URL argument and can optionally accept additional parameters to define the sub-protocol that you’ll use, such as chat or rpc. The client and the server are typically matched to use the same protocol.
* **close** A method that closes WebSocket.
* **send** A method that sends data to the server.
* **binaryType** A property that indicates the binary data format the onmessage event receives.
* **bufferedAmount** A property containing the number of data bytes queued using the send method.
* **extensions** A property that indicates the extensions the server selected.
* **onclose** An event property that’s called when the socket is closed.
* **onerror** An event property that’s called when there is an error.
* **onmessage** An event property that’s called when a message is received.
* **onopen** An event property that’s called when WebSocket establishes a connection.
* **protocol** A property that indicates the protocol that the server selected.
* **readyState** A property that indicates the state of the WebSocket connection.
* **url** A property that indicates the current URL of the WebSocket object.

### Implementing the WebSocket object

* WebSocket protocol communications typically use TCP port number 80
* The WebSocket URL begins with ws:// or wss:// for secure WebSocket protocol

The example before shows how to create a WebSocket object and configure its onopen, onerror, and onclose events, as well as calling the send method to send message, and the onmessage event triggers if there is a response.





The example above does the following:

* The connect function instantiates WebSocket.
* The constructor accepts a URI argument. Creating the WebSocket object automatically initiates communications to the URI to attempt to open the connection asynchronously.
* The connect function also subscribes to the onopen, onclose, onmessage, and onerror events.
* **NOTE**: It’s important to subscribe to these events imme-diately because the connection might open quickly, and you want to ensure that you are subscribed to the onopen event as soon as possible so you don’t miss the event.
* The doSend function checks the readyState property of the WebSocket object, then sends a message to the server if the connection is OPEN. (The readyState properties include: **CONNECTING = 0, OPEN = 1, CLOSING = 2, CLOSED = 3**)
* When a message is received from the server, the onMessage function is called with the event object passed
* When an error is received, the onError function is called with the event object passed

For an example showcasing the above, go to https://jsfiddle.net/9gxLLqr2/1/