WebSocket Client

Wakanda Server implements the **WebSocket client API**, allowing you to open a WebSocket connection from the Wakanda SSJS to a WebSocket server. WebSockets enable applications to use the WebSocket protocol (defined by the IETF) for two-way communication with a remote host.

The Wakanda SSJS WebSocket client object implements the WebSocket W3C specification with certain limitations regarding binary data types (see **send( )**). This specification is based on HTML5 and was originally designed for Web applications.

Please keep in mind that the WebSocket specification is still under discussion and should neither be considered as frozen nor as finished.

WebSocket Constructor

WebSocket( )

void **WebSocket**(String *url* [, String | Array *protocols*])

Parameter	Type	Description
url	String	URL of the Web Socket server to connect to
protocols	String, Array	Sub-protocol(s) to use

Description

The **WebSocket( )** method is the constructor of the client *WebSocket* type class objects. It allows you to create a new Websocket connection to the Web Socket server whose address you passed in the *url* parameter.

The *protocols* parameter can be a string or an array of strings. If present, it specifies one or several subprotocol(s). The server must support at least one of these subprotocols for the connection to be successful. After the connection is established, the protocol actually selected by the server is returned in the **protocol** property.

Implementation Note: *In the current Wakanda implementation, the* **protocol** *property is not used, thus the* ***protocols*** *parameter can be left empty or omitted.*

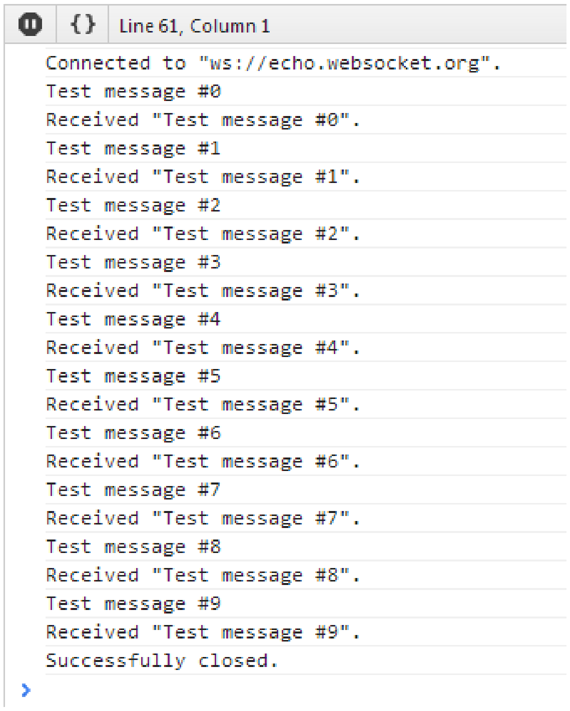
Once created, a *WebSocket* object has properties and methods that you can use to handle the connection and the exchanged data. These are described in the **WebSocket Instances** section.

Example

This simple example connects to the "echo" server of www.websocket.org, then sends 10 strings which will be echoed back by the server.

```
var ws = new WebSocket("ws://echo.websocket.org");

var i = 0;

function send () {
    var string = "Test message #" + i;
    ws.send(string);
    i++;
    log(string + "\n");
}

ws.onopen= function () {
    log("Connected to \"" + ws.url+ "\".\n");
    send();
}

ws.onmessage = function (message) {
    log("Received \"" + message.data.toString() + "\".\n");
    if (i == 10)
        ws.close();
    else
        send();
}

ws.onerror = function () {
    log("An error happened!\n");
    exitWait();
}

ws.onclose = function (closeEvent) {
    log("Successfully closed.\n");
    exitWait();
}

function log (string) {
    console.log(string);
}
wait();
```

The following information are written in the solution's log file:



```
Connected to "ws://echo.websocket.org".
Test message #0
Received "Test message #0".
Test message #1
Received "Test message #1".
Test message #2
Received "Test message #2".
Test message #3
Received "Test message #3".
Test message #4
Received "Test message #4".
Test message #5
Received "Test message #5".
Test message #6
Received "Test message #6".
Test message #7
Received "Test message #7".
Test message #8
Received "Test message #8".
Test message #9
Received "Test message #9".
Successfully closed.
```

WebSocket Instances

WebSocket objects are instantiated by the **WebSocket( )** constructor method.

onopen

Description

The **onopen** property contains the event handler function to call when an **open** event is received by the *WebSocket* object.

An **open** event is fired by Wakanda Server when the Web Socket connection is established.

onmessage

Description

The **onmessage** property contains the event handler function to call when a **message** event is received by the *WebSocket* object.

A **message** event is fired by Wakanda Server when a WebSocket message has been received with text data over the WebSocket connection.

The **onmessage** function accepts a single parameter, whose *data* attribute is set to receive data.

By default, *data* is of the *Buffer* type. You can set the data type using the **binaryType** property.

Example

```
ws.onmessage = function (message) {

    log("Received \"" + message.data.toString() + "\".\n");  //logs received message as text
}
```

onerror

Description

The **onerror** property contains the event handler function to call when an error event is received by the *WebSocket* object.

onclose

Description

The **onclose** property contains the event handler function to call when a **close** event is received by the *WebSocket* object.

A **close** event is fired by Wakanda Server when the *WebSocket* connection is closed, regardless of the reason it is closed.

The event handler function receives a single parameter which is a **CloseEvent** object. This object has the following read-only properties:

Property	Type	Description
wasClean	Boolean	Represents whether the connection closed cleanly (true) or not (false)
code	Number	WebSocket connection close code provided by the server
reason	String	WebSocket connection close reason provided by the server

Example

You want to check whether the close was successful:

```
ws.onclose = function (closeEvent) {
    if(closeEvent.wasClean = true)
        log("Successfully closed.\n");
    else
        log("Closed with an issue.\n");
    exitWait();

}
```

url

Description

The **url** property returns the result from resolving the URL that was passed to the **WebSocket( )** constructor.

readyState

Description

The **readyState** property represents the current state of the connection. It can have the following (constant) values:

Constant	Numeric value	State
CONNECTING	0	The connection has not yet been established.
OPEN	1	The WebSocket connection is established and communication is possible.
CLOSING	2	The connection is going through the closing handshake, or the **close( )** method has been invoked.
CLOSED	3	The connection has been closed or could not be opened.

bufferedAmount

Description

The **bufferedAmount** property returns the number of bytes that have been queued using **send( )** but not yet sent. If the connection is closed, this property's value will only increase with each call to the **send( )** method (the number does not reset to zero once the connection closes).

extensions

Description

The **extensions** property returns, after the *WebSocket* connection is established, the extensions selected by the server, if any.

Otherwise, it returns an empty string.

protocol

Description

Implementation Note*: This property is not used in the current Wakanda implementation.*

The **protocol** property returns the subprotocol selected by the server, if any. It can be used in conjunction with the array form of the constructor's second argument to perform subprotocol negotiation.

binaryType

Description

The **binaryType** read/write property allows you to set the type of data exchanged through the *WebSocket*, i.e., the **message.data** object received in the **onmessage** event handler function.

The following data types are supported in Wakanda:

- "buffer" (default)
- "string"

addEventListener( )

void **addEventListener**(String *type*, Function *listener*)

Parameter	Type	Description
type	String	Event type
listener	Function	Callback function

Description

Note*: This method is inherited from the EventTarget interface (DOM Level 2) that is partially implemented in Wakanda.*

The **addEventListener( )** method allows the registration of an event listener in the *WebSocket* instance.

In the *type* parameter, pass the name of the event to register. In the current Wakanda implementation, the following events are supported:

- "open", corresponding to the **onopen** property,
- "message", corresponding to the **onmessage** property,
- "error", corresponding to the **onerror** property,
- "close", corresponding to the **onclose** property.

In the *listener* parameter, pass the function to be called when the event occurs.

close( )

void **close**([Number *code* [, String *reason*]])

Parameter	Type	Description
code	Number	Code for closing
reason	String	Closing reason

Description

The **close( )** method closes the *WebSocket* connection or connection attempt, if any.

This method also change the **readyState** attribute's value to CLOSING (2). If the connection is already closed, the method does nothing.

Optionally, you can pass a *code* and a *reason* parameter to provide information about the closing operation:

- *code* is the status code to use in the WebSocket Close message. It must be a value of 1000 or be in the range from 3000 to 4999;
- *reason* is provided in the WebSocket Close message after the status code.

removeEventListener( )

void **removeEventListener**(String *type*, Function *listener*)

Parameter	Type	Description
type	String	Event type
listener	Function	Callback function

Description

Note: *This method is inherited from the* EventTarget *interface (DOM Level 2) that is partially implemented in Wakanda.*

The **removeEventListener( )** method allows the removal of event listeners from the *WebSocket* instance.

In the *type* parameter, pass the name of the event to be removed. In the current Wakanda implementation, the following events are supported:

- "open", corresponding to the **onopen** property,
- "message", corresponding to the **onmessage** property,
- "error", corresponding to the **onerror** property,
- "close", corresponding to the **onclose** property.

In the *listener* parameter, pass the callback function to be removed.

The method does nothing if it is called with arguments which do not identify any currently registered EventListener.

send( )

Boolean **send**(String | Buffer *data*)

Parameter	Type	Description
data	String, Buffer	Data to transmit
Returns	Boolean	True if data was sent successfully, or false if the connection is closed

Description

The **send( )** method sends text *data* using the WebSocket connection.

If the **readyState** attribute is not OPEN, an error is raised. If the *data* cannot be sent, the WebSocket connection is closed.

The method returns **true** if the connection is established and the data was sent successfully, or **false** if the method failed to send the data and the connection is closed.

Implementation Note: *The Wakanda WebSocket Client currenty does not support* ***data*** *of the ArrayBuffer and BLOB types.*