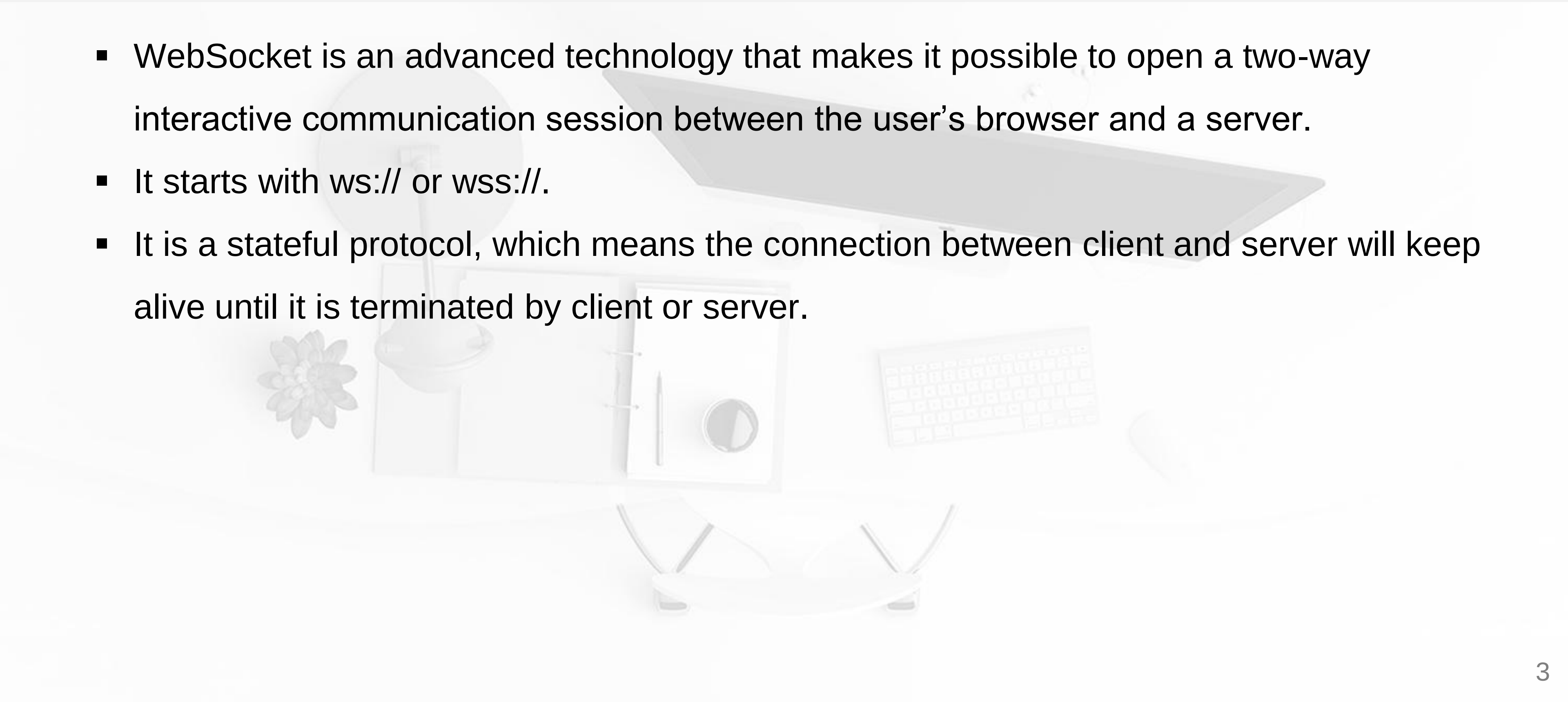
Network Programming

Ung Văn Giàu
Email: giau.ung@eiu.edu.vn

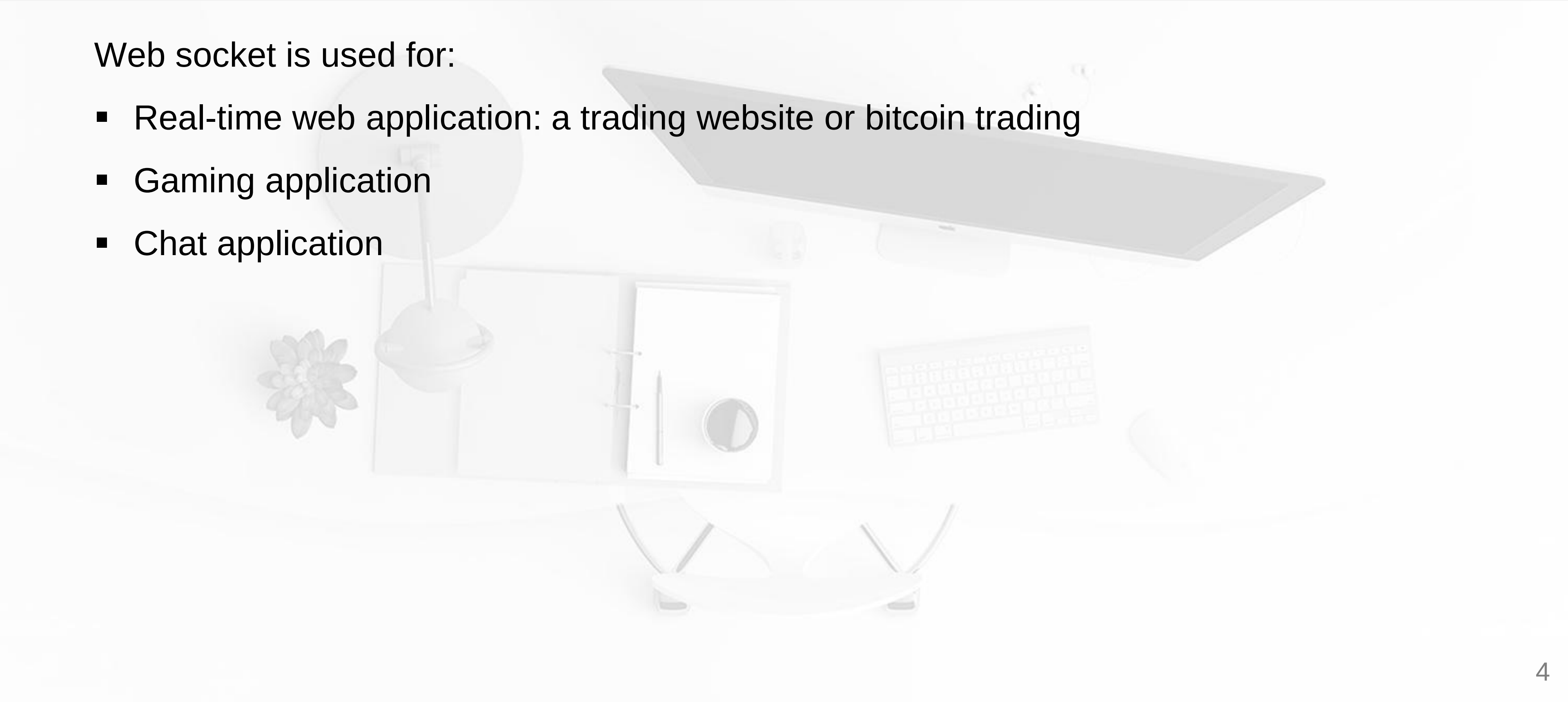
WebSocket

Introduction



- WebSocket is an advanced technology that makes it possible to open a two-way interactive communication session between the user’s browser and a server.
- It starts with ws:// or wss://.
- It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by client or server.

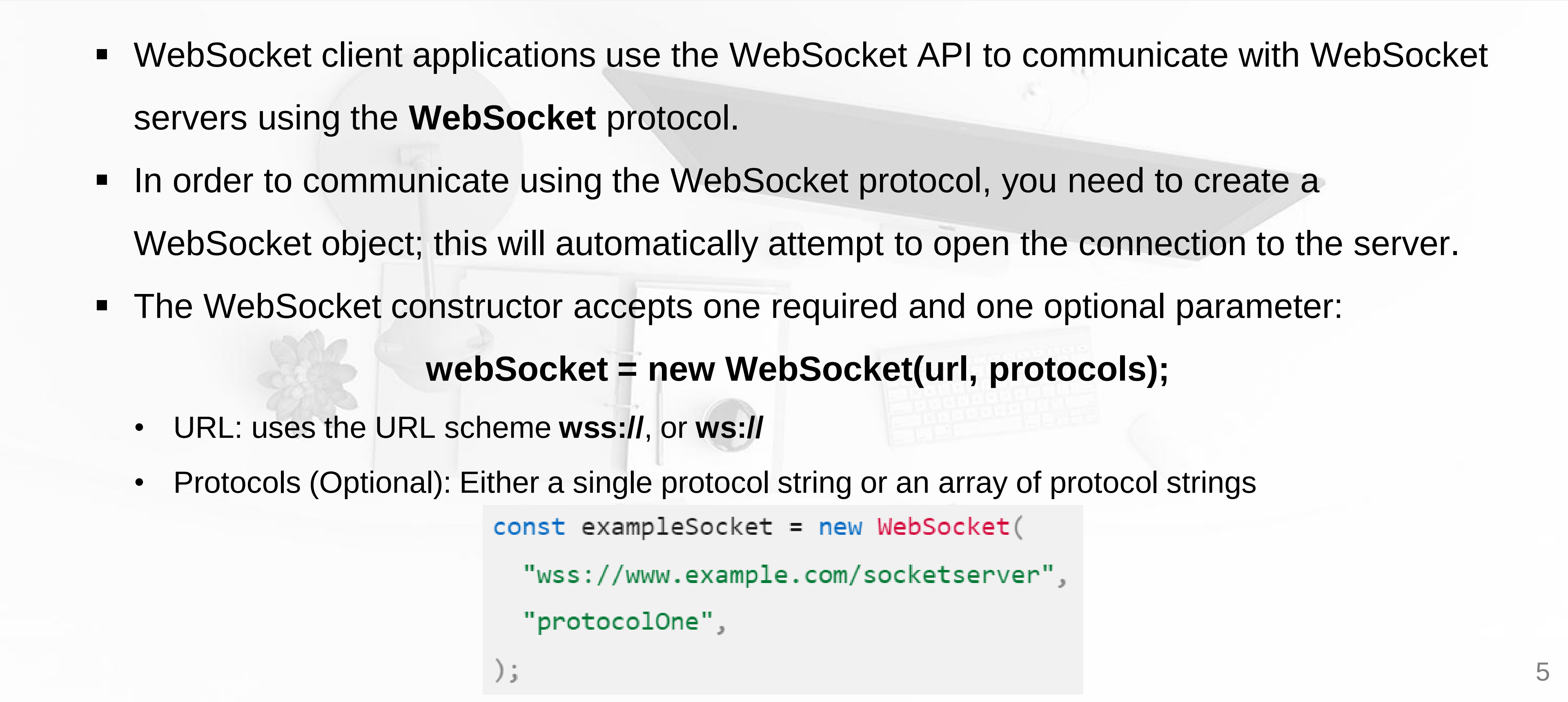
Introduction



Web socket is used for:

- Real-time web application: a trading website or bitcoin trading
- Gaming application
- Chat application

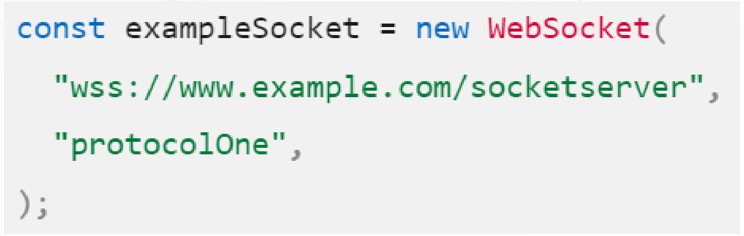
Writing WebSocket client applications



- WebSocket client applications use the WebSocket API to communicate with WebSocket servers using the **WebSocket** protocol.
- In order to communicate using the WebSocket protocol, you need to create a WebSocket object; this will automatically attempt to open the connection to the server.
- The WebSocket constructor accepts one required and one optional parameter:

  webSocket = new WebSocket(url, protocols);

  - URL: uses the URL scheme **wss://**, or **ws://**
  - Protocols (Optional): Either a single protocol string or an array of protocol strings



```
const exampleSocket = new WebSocket(
  "wss://www.example.com/socketserver",
  "protocolOne",
);
```

Sending data to the server

- Once you’ve opened your connection, you can begin transmitting data to the server. To do this, call the WebSocket object’s **send()** method for each message you want to send:

```
exampleSocket.send("Here's some text that the server is
urgently awaiting!");
```

- To send data only takes place once a connection is established, we can define an onopen event handler to do the work:

```
exampleSocket.onopen = (event) => {
  exampleSocket.send("Here's some text that the server is urgently
awaiting!");
};
```

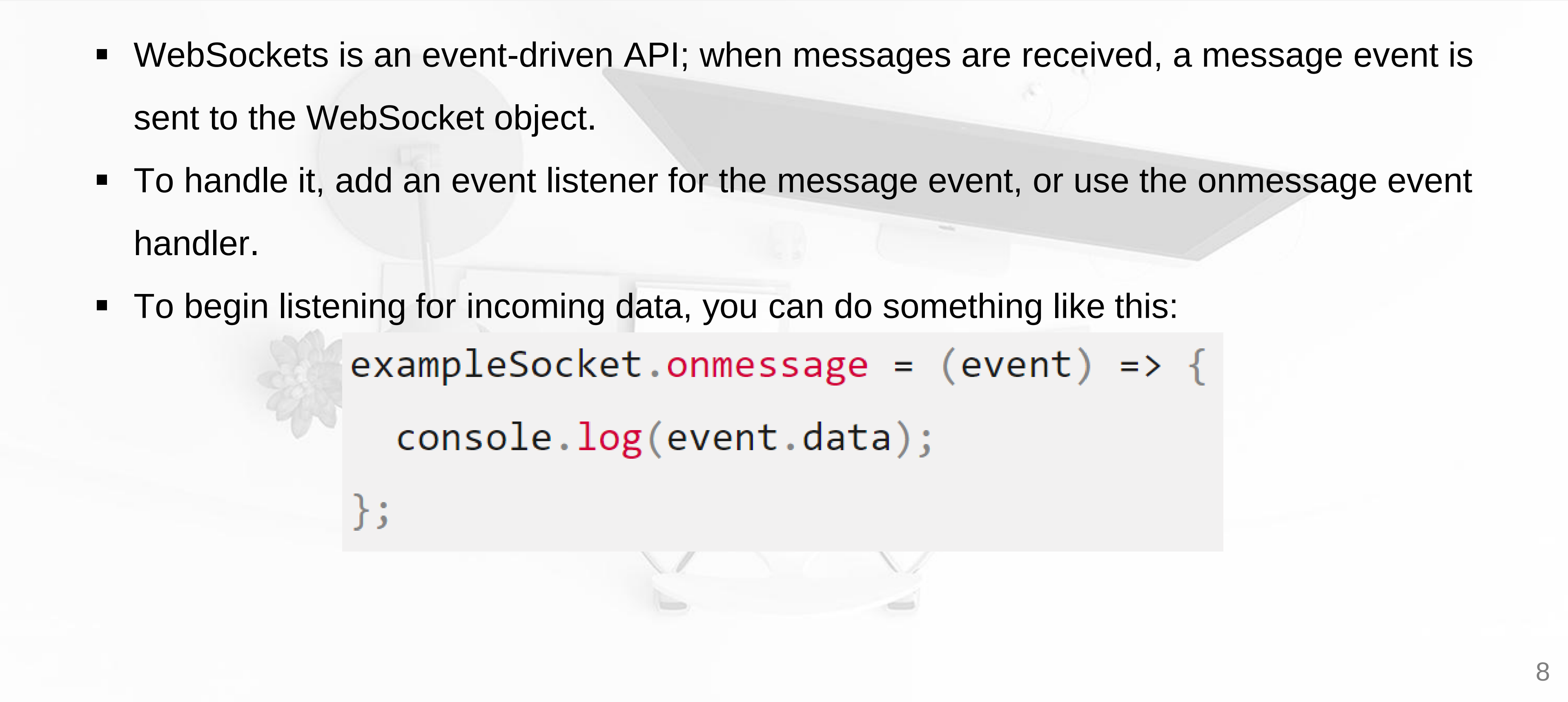
Sending data to the server

- Using JSON to transmit objects:

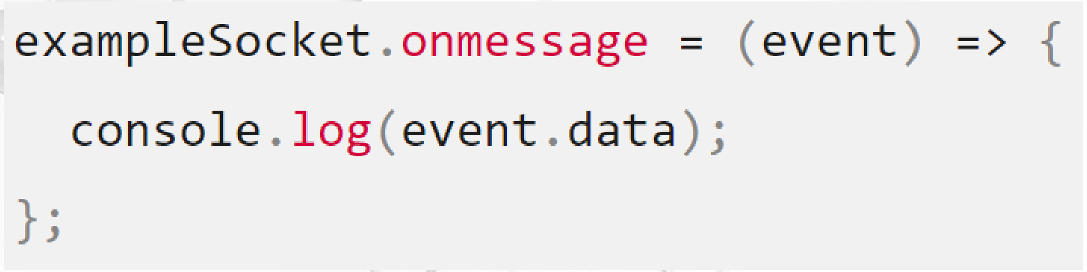
```
const msg = {
  type: "message",
  text: document.getElementById("text").value,
  id: clientID,
  date: Date.now(),
};

// Send the msg object as a JSON-formatted string.
exampleSocket.send(JSON.stringify(msg));
```

Receiving messages from the server



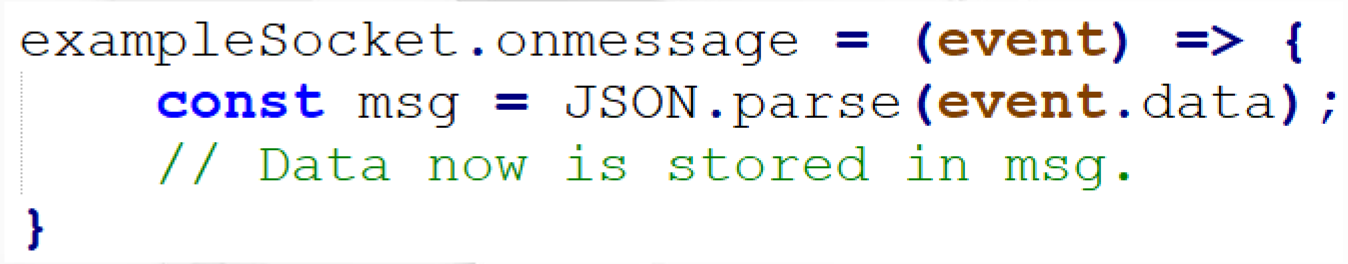
- WebSockets is an event-driven API; when messages are received, a message event is sent to the WebSocket object.
- To handle it, add an event listener for the message event, or use the onmessage event handler.
- To begin listening for incoming data, you can do something like this:



```
exampleSocket.onmessage = (event) => {
  console.log(event.data);
};
```

Receiving and interpreting JSON objects

- The code that interprets these incoming messages might look like this:



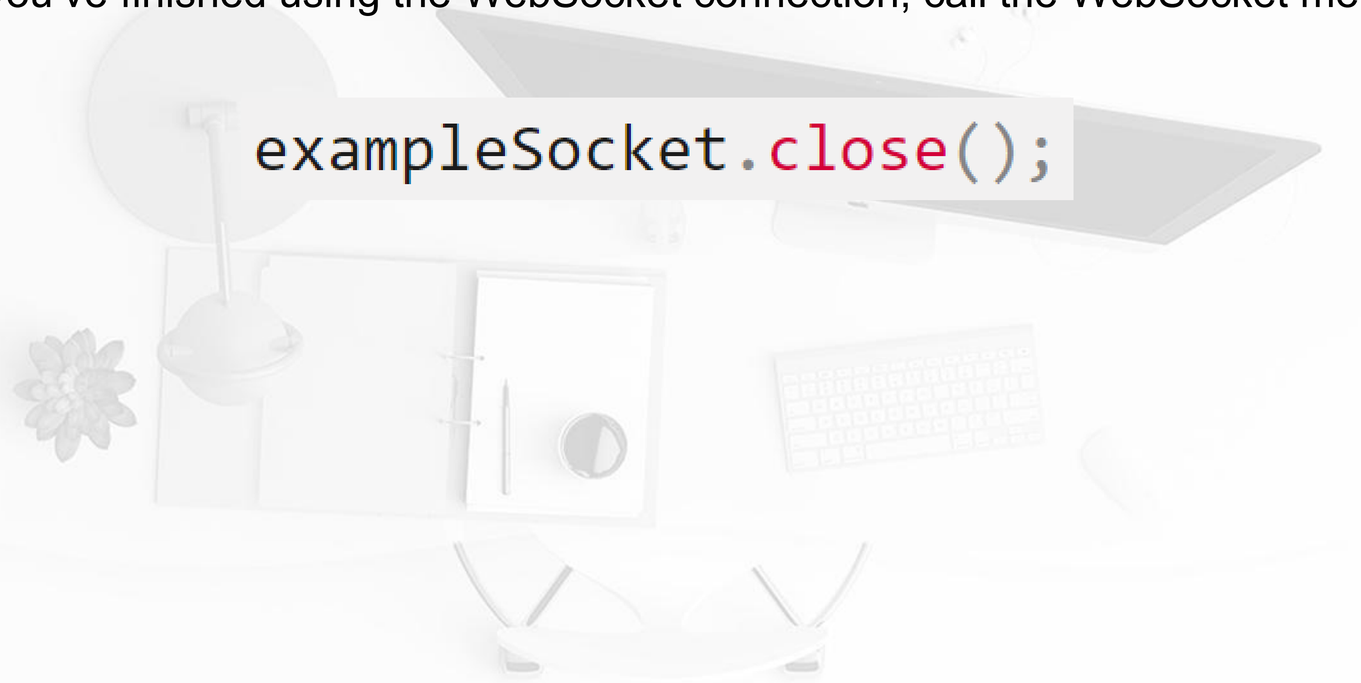
```
exampleSocket.onmessage = (event) => {
    const msg = JSON.parse(event.data);
    // Data now is stored in msg.
}
```

- JSON.parse() is used to convert the JSON object back into the original object, then examine and act upon its contents.
- **Note:** Text received over a WebSocket connection is in **UTF-8 format**.

Closing the connection

- When you’ve finished using the WebSocket connection, call the WebSocket method close():

```
exampleSocket.close();
```



Q&A