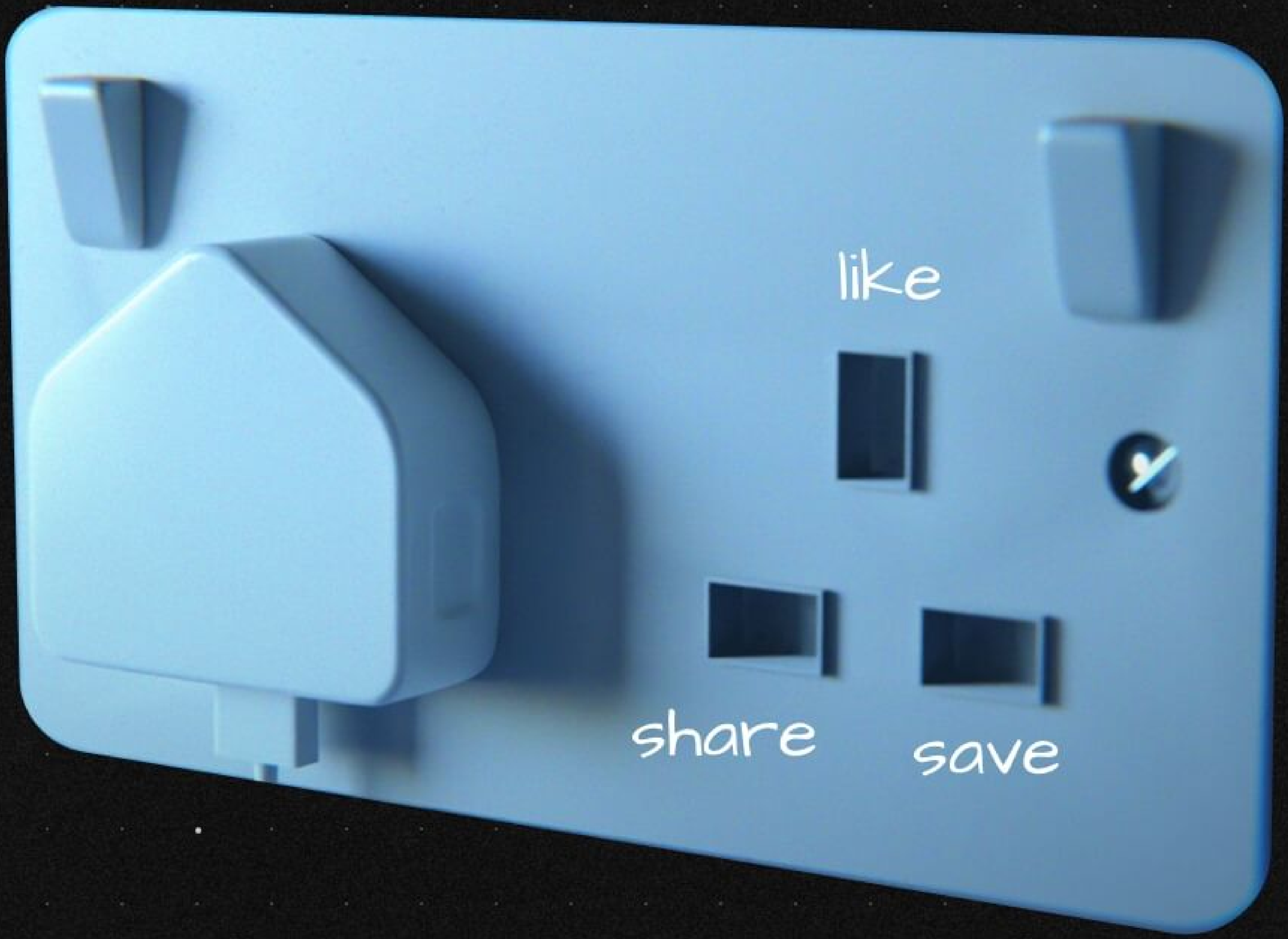
JS Most common JavaScript Interview Questions

WebSockets

in JavaScript



JS

WebSockets in JavaScript allow for real-time, bidirectional communication between a client (usually a web browser) and a server.

They are particularly useful for building interactive web applications that require instant data updates.

Swipe ⟶

Understanding WebSockets

WebSockets provide a full-duplex communication channel over a single TCP connection.

Unlike HTTP, WebSockets enable both the client and server to send messages to each other at any time without waiting for a request from the client.

WebSockets use a protocol that starts with `ws://` (unencrypted) or `wss://` (encrypted) instead of http:// or https://

Swipe ⟶

JS

Creating a WebSocket Connection

To establish a WebSocket connection in JavaScript, create a new WebSocket object, passing the WebSocket server **URL as a parameter**.

```
const socket = new WebSocket('ws://example.com/socket');
```

JS

WebSocket Events

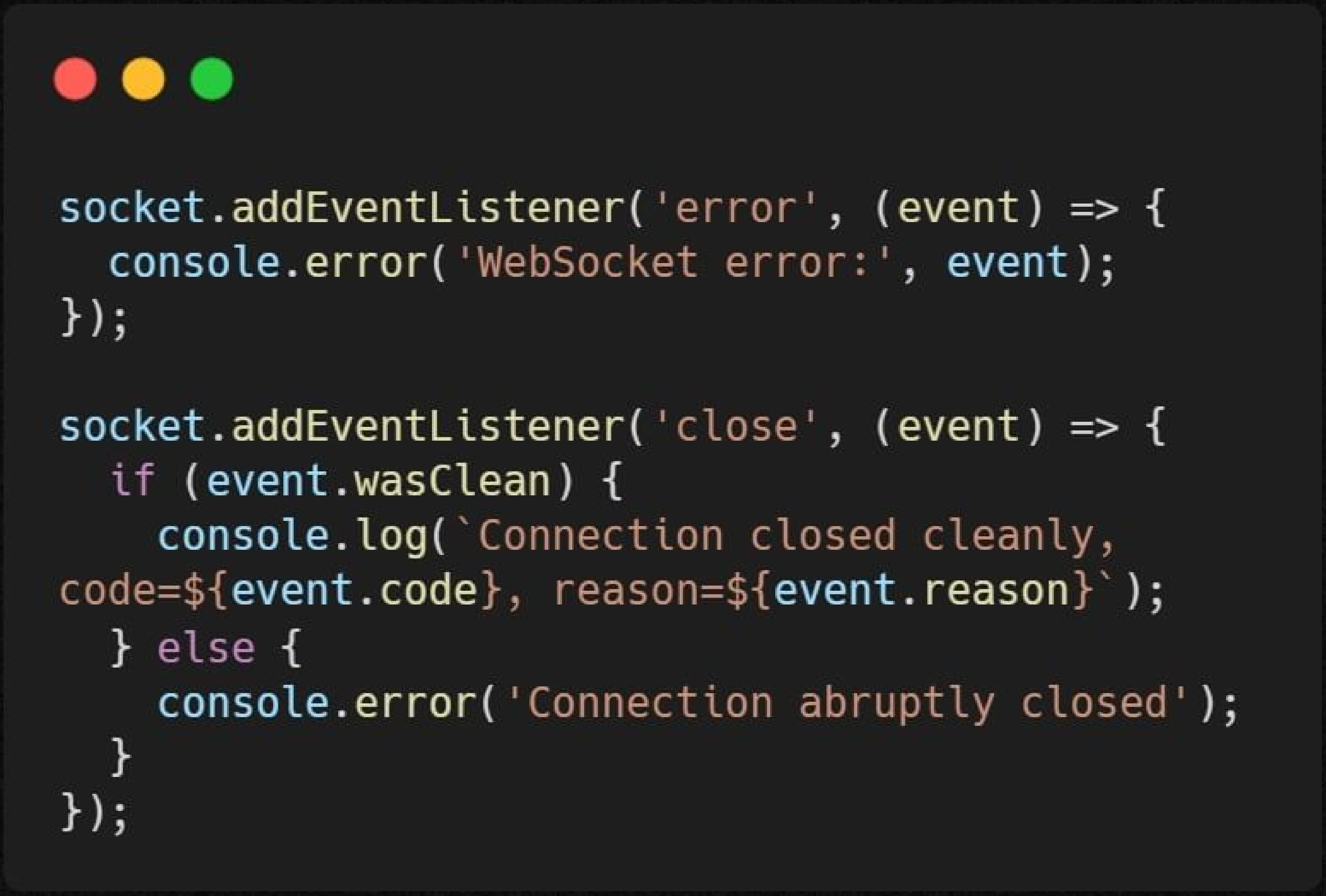
WebSocket objects emit various events to handle different stages of the connection.

open: Triggered when the connection is successfully established

```
socket.addEventListener('open', (event) => {
  // Connection is open.
});

socket.addEventListener('message', (event) => {
  const message = event.data;
  // Handle incoming message.
});
```

message: Fired when the server sends a message error: Triggered when an error occurs



```javascript
socket.addEventListener('error', (event) => {
  console.error('WebSocket error:', event);
});

socket.addEventListener('close', (event) => {
  if (event.wasClean) {
    console.log(`Connection closed cleanly,
code=${event.code}, reason=${event.reason}`);
  } else {
    console.error('Connection abruptly closed');
  }
});
```

close: Fired when the connection is closed

Swipe ⟶

JS

Sending Data

To send data to the server, use the **send** method of the WebSocket object. Data can be a string, ArrayBuffer, or Blob.

```
socket.send('Hello, server!');
```

Closing the Connection

To close the WebSocket connection, call the **close** method on the WebSocket object.

```
socket.close();
```

Server-Side WebSocket Implementation

On the server side, you need to implement a WebSocket server that listens for WebSocket connections and handles messages from clients. Popular libraries for this purpose include ws for Node.js and libraries for various languages like Python, Ruby, and Java.

Error Handling

Be sure to handle errors gracefully by listening for the error event and providing appropriate feedback to the user.

JS

Security Considerations

When using WebSockets, consider security measures like encrypting the connection using `wss://`, implementing authentication, and validating incoming messages to prevent vulnerabilities like WebSocket-based attacks.

Use Cases

WebSockets are suitable for real-time chat applications, online gaming, live notifications, collaborative tools, and any scenario where immediate data updates are required.