**Web Sockets vs Socket.io**

| **Feature** | **WebSocket (ws / wss)** | **Socket.IO** |
| --- | --- | --- |
| **Protocol** | Pure **WebSocket protocol** (RFC 6455). | Built on top of **WebSocket + HTTP long polling fallback**. |
| **Browser Support** | Supported in all modern browsers (no polyfills). | Uses WebSocket when available, falls back to polling for compatibility. |
| **CORS Handling** | CORS **not required** (after upgrade). | Requires explicit **CORS config** (handshake is HTTP). |
| **Ease of Use** | Low-level API (send/receive text/binary messages). You manage events yourself. | High-level API (named events: socket.emit("msg"), socket.on("msg")). More developer-friendly. |
| **Reconnection** | No built-in reconnection. You must handle retries manually. | Automatic reconnection with exponential backoff. |
| **Message Acknowledgements** | Not built-in — you must implement your own ack mechanism. | Built-in acknowledgements (socket.emit("event", data, callback)). |
| **Rooms / Namespaces** | Not supported natively. You need to implement custom grouping. | Supports **rooms** (group broadcasting) and **namespaces** (scoped connections). |
| **Scalability** | Must handle clustering + pub/sub (e.g., Redis) yourself. | Built-in adapters (Redis, Kafka, etc.) for scaling across nodes. |
| **Binary Support** | Full support for binary messages (ArrayBuffer, Blob). | Supports both text & binary, plus built-in serialization. |
| **Security** | Relies on wss:// for TLS encryption. | Relies on HTTPS + WSS; also has built-in support for middleware (auth, JWT). |
| **Use Cases** | Lightweight real-time comms, streaming, games, IoT. | Complex real-time apps (chat apps, collaboration tools, dashboards). |