

Referensi Cepat Socket.IO

Event, room, namespace, middleware, pola real-time

Setup

Setup Server (Node.js)

```
import { Server } from "socket.io";
const io = new Server(3000, {
  cors: { origin: "http://localhost:5173" }
});
```

Setup Client

```
import { io } from "socket.io-client";
const socket = io("http://localhost:3000");
```

Dengan Express

```
import express from "express";
import { createServer } from "http";
import { Server } from "socket.io";
const app = express();
const server = createServer(app);
const io = new Server(server);
```

Opsi Server

cors	Konfigurasi CORS untuk cross-origin
path	Path kustom (default: /socket.io)
pingInterval	Interval heartbeat (ms, default 25000)
pingTimeout	Timeout sebelum disconnect (default 20000)
maxHttpBufferSize	Ukuran pesan maksimum dalam bytes (default 1MB)

Event

Event Bawaan (Server)

connection	Client terhubung
disconnect	Client terputus
disconnecting	Client sedang terputus (masih di room)
error	Event error

Event Bawaan (Client)

connect	Terhubung ke server
disconnect	Terputus dari server
connect_error	Koneksi gagal
reconnect	Berhasil terhubung kembali
reconnect_attempt	Sedang mencoba reconnect

Siklus Hidup Koneksi

```
io.on("connection", (socket) => {
  console.log(`connected: ${socket.id}`);
  socket.on("disconnect", (reason) => {
    console.log(`disconnected: ${reason}`);
  });
});
```

Emit

Emit dari Server

```
socket.emit("hello", { msg: "world" });
socket.emit("data", arg1, arg2);
io.emit("broadcast", data);
```

Emit dari Client

```
socket.emit("chat:message", { text });
socket.emit("update", data, (res) => {
  console.log("ack:", res);
});
```

Pola Emit

socket.emit(ev, data)	Kirim hanya ke socket ini
io.emit(ev, data)	Kirim ke semua client yang terhubung
socket.broadcast.emit()	Semua client kecuali pengirim
io.to(room).emit()	Semua client dalam room
socket.to(room).emit()	Anggota room kecuali pengirim

Broadcasting

Metode Broadcast

```
io.emit("msg", data);
socket.broadcast.emit("msg", data);
io.to("room1").emit("msg", data);
io.except("room2").emit("msg", data);
```

Volatile & Compressed

socket.volatile.emit()	Lewati jika client belum siap (tanpa buffering)
socket.compress(true).emit()	Aktifkan kompresi per pesan
io.local.emit()	Broadcast ke server lokal saja (multi-node)
socket.timeout(5000).emit()	Emit dengan timeout untuk acknowledgement

Room

Operasi Room

```
socket.join("room-1");
socket.join(["room-1", "room-2"]);
socket.leave("room-1");
io.to("room-1").emit("msg", data);
```

Properti Room

socket.rooms	Set room yang diikuti socket ini
socket.id	Setiap socket otomatis bergabung ke room ID-nya sendiri
io.sockets.adapter.rooms	Map semua room dan anggotanya

Pola Room

```
socket.on("join:room", (room) => {
  socket.join(room);
  io.to(room).emit("user:joined", socket.id);
});
socket.on("disconnecting", () => {
  for (const room of socket.rooms) {
    socket.to(room).emit("user:left", socket.id);
  }
});
```

Namespace

Membuat Namespace

```
const chat = io.of("/chat");
const admin = io.of("/admin");
chat.on("connection", (socket) => {
  chat.emit("user:online", socket.id);
});
```

Client Terhubung ke Namespace

```
const chat = io("http://localhost:3000/chat");
const admin = io("http://localhost:3000/admin");
```

Namespace Dinamis

```
io.of(/^\/project-\d+$/).on("connection",
(socket) => {
  const ns = socket.nsp.name;
  console.log(`joined namespace: ${ns}`);
});
```

Middleware

Middleware Server

```
io.use((socket, next) => {
  const token = socket.handshake.auth.token;
  if (isValid(token)) return next();
  next(new Error("authentication failed"));
});
```

Middleware Namespace

```
const admin = io.of("/admin");
admin.use((socket, next) => {
  if (socket.handshake.auth.role === "admin")
    return next();
  next(new Error("not authorized"));
});
```

Properti Middleware

socket.handshake.auth	Data auth yang dikirim dari client
socket.handshake.headers	Header HTTP dari request awal
socket.handshake.query	Query parameter dari URL koneksi
socket.data	Data arbitrer yang dilampirkan di middleware

Penanganan Error

Error di Sisi Server

```
socket.on("action", (data, callback) => {
  try {
    const result = process(data);
    callback({ status: "ok", data: result });
  } catch (err) {
    callback({ status: "error", msg: err.message });
  }
});
```

Error di Sisi Client

```
socket.on("connect_error", (err) => {
  console.log("connection error:", err.message);
});
socket.io.on("reconnect_failed", () => {
  console.log("reconnection failed");
});
```

Opsi Reconnect Client

reconnection	Aktifkan auto-reconnect (default true)
reconnectionAttempts	Maksimum percobaan (default Infinity)
reconnectionDelay	Delay awal ms (default 1000)
reconnectionDelayMax	Delay maksimum ms (default 5000)

Referensi Cepat Socket.IO

Acknowledgements

Client Kirim, Server Ack

```
// client
socket.emit("save", data, (response) => {
  console.log("server ack:", response);
});
// server
socket.on("save", (data, callback) => {
  callback({ saved: true, id: 42 });
});
```

Server Kirim, Client Ack

```
// server
socket.emit("ping", (response) => {
  console.log("client ack:", response);
});
// client
socket.on("ping", (callback) => {
  callback("pong");
});
```

Dengan Timeout

```
socket.timeout(5000).emit("save", data,
  (err, response) => {
    if (err) console.log("timeout!");
    else console.log("ack:", response);
  }
);
```

Pola Umum

Chat Room

```
io.on("connection", (socket) => {
  socket.on("chat:join", (room) => {
    socket.join(room);
    socket.to(room).emit("chat:joined",
      socket.id);
  });
  socket.on("chat:message", ({ room, text }) => {
    io.to(room).emit("chat:message", {
      from: socket.id, text
    });
  });
});
```

Kehadiran Online

```
const users = new Map();
io.on("connection", (socket) => {
  users.set(socket.id, socket.handshake.auth);
  io.emit("users:list", [...users.values()]);
  socket.on("disconnect", () => {
    users.delete(socket.id);
    io.emit("users:list", [...users.values()]);
  });
});
```

Rate Limiting

```
io.use((socket, next) => {
  const ip = socket.handshake.address;
  if (rateLimiter.consume(ip)) return next();
  next(new Error("rate limit exceeded"));
});
```