```
io.on('connect', onConnect);

function onConnect(socket){

  // sending to the client
  socket.emit('hello', 'can you hear me?', 1, 2, 'abc');

  // sending to all clients except sender
  socket.broadcast.emit('broadcast', 'hello friends!');

  // sending to all clients in 'game' room except sender
  socket.to('game').emit('nice game', "let's play a game");

  // sending to all clients in 'game1' and/or in 'game2' room, except sender
  socket.to('game1').to('game2').emit('nice game', "let's play a game (too)");

  // sending to all clients in 'game' room, including sender
  io.in('game').emit('big-announcement', 'the game will start soon');

  // sending to all clients in namespace 'myNamespace', including sender
  io.of('myNamespace').emit('bigger-announcement', 'the tournament will start soon');

  // sending to a specific room in a specific namespace, including sender
  io.of('myNamespace').to('room').emit('event', 'message');

  // sending to individual socketid (private message)
  io.to(socketId).emit('hey', 'I just met you');

  // WARNING: `socket.to(socket.id).emit()` will NOT work, as it will send to everyone in the room
  // named `socket.id` but the sender. Please use the classic `socket.emit()` instead.

  // sending with acknowledgement
  socket.emit('question', 'do you think so?', function (answer) {});

  // sending without compression
  socket.compress(false).emit('uncompressed', "that's rough");

  // sending a message that might be dropped if the client is not ready to receive messages
  socket.volatile.emit('maybe', 'do you really need it?');

  // specifying whether the data to send has binary data
  socket.binary(false).emit('what', 'I have no binaries!');

  // sending to all clients on this node (when using multiple nodes)
  io.local.emit('hi', 'my lovely babies');

  // sending to all connected clients
  io.emit('an event sent to all connected clients');

};
```

Note: The following events are reserved and should not be used as event names by your application:

- `connect`
- `connect_error`
- `connect_timeout`
- `error`
- `disconnect`
- `disconnecting`
- `newListener`
- `reconnect_attempt`
- `reconnecting`
- `reconnect_error`
- `reconnect_failed`
- `removeListener`
- `ping`
- `pong`