**Ответы на вопросы**

1. Socket.IO — это библиотека, которая обеспечивает двунаправленную связь на основе событий с малой задержкой между клиентом и сервером.
2. WebSocket API — это передовая технология, позволяющая открыть двусторонний интерактивный сеанс связи между браузером пользователя и сервером. С помощью этого API вы можете отправлять сообщения на сервер и получать ответы на основе событий без необходимости опрашивать сервер для получения ответа.
3. Socket.IO НЕ является реализацией WebSocket. Хотя Socket.IO действительно использует WebSocket для транспортировки, когда это возможно, он добавляет дополнительные метаданные к каждому пакету. Вот почему клиент WebSocket не сможет успешно подключиться к серверу Socket.IO, а клиент Socket.IO также не сможет подключиться к простому серверу WebSocket.
4. Дуплексная связь – способ, когда передача возможна в обоих направлениях канала электросвязи.  
   Полудуплексная связь – способ симплексной связи на одном конце линии и дуплексной на другом.
5. JavaScript, Python, Ruby, Java, PHP, C#
6. Соединение вернется к long-polling HTTP, если соединение WebSocket не может быть установлено. Эта функция была причиной №1, по которой люди использовали Socket.IO, когда проект был создан более десяти лет назад (!), поскольку поддержка WebSockets браузерами все еще находилась в зачаточном состоянии.  
   Broadcasting: на стороне сервера вы можете отправить событие всем подключенным клиентам или подмножеству клиентов.  
   Multiplexing: пространства имён позволяют разделить логику вашего приложения по одному общему соединению. Это может быть полезно, например, если вы хотите создать «административный» канал, к которому смогут присоединиться только авторизованные пользователи.
7. Динамически создавать пространства имен возможно с помощью регулярного выражения io.of(/^\/dynamic-\d+$/) или с функцией:

io.of((name, auth, next) => {

next(null, true); // or false, when the creation is denied

});

Получить доступ к новому пространству имен можно в connection-событии:

io.of(/^\/dynamic-\d+$/).on("connection", (socket) => {

const namespace = socket.nsp;

});

Возвращаемое значение метода of()— это то, что мы называем родительским пространством имен, из которого вы можете:

зарегистрировать промежуточное программное обеспечение

const parentNamespace = io.of(/^\/dynamic-\d+$/);

parentNamespace.use((socket, next) => { next() });

Промежуточное программное обеспечение будет автоматически зарегистрировано в каждом дочернем пространстве имен.

const parentNamespace = io.of(/^\/dynamic-\d+$/);

parentNamespace.emit("hello"); // will be sent to users in /dynamic-1, /dynamic-2, ...

При этом существующие пространства имен имеют приоритет над динамическими пространствами имен. Например:

// register "dynamic-101" namespace

io.of("/dynamic-101");

io.of(/^\/dynamic-\d+$/).on("connection", (socket) => {

// will not be called for a connection on the "dynamic-101" namespace

});