1. Name attributes of HTTP protocol that makes it difficult to use for real time systems.

2. Explain polling and long-polling strategies, their pros and cons.

Short polling.

A lot of request that are processed as they come on server. Creates a lot of traffic (uses resources, but frees them as soon as response is send back):

00:00:00 C-> Is the cake ready?
00:00:01 S-> No, wait.
00:00:01 C-> Is the cake ready?
00:00:02 S-> No, wait.
00:00:02 C-> Is the cake ready?
00:00:03 S-> Yeah. Have some lad.
00:00:03 C-> Is the other cake ready? ..

Long polling

One request goes to the server and the client is waiting for the response to come (its unresolved). If you use Node (or any other async approach), you can potentially minimize the resource usage a lot (store response object for http request and use it when the work is ready)

12:00 00:00:00 C-> Is the cake ready?
12:00 00:00:03 S-> Yeah have some lad.
12:00 00:00:03 C-> Is the cake ready?

If you compare that to short polling, you will see that potentially in short poll you used more transfer, but during those 3s you actually take 1,5s of processing time (means something could execute in between your calls). In case for long poll the same resources were used all the time. Now usually php with all libs starts with 4MB memory - then you have a framework 4-20MB. Assume you have 1024MB RAM available (free). Say let's be pessimistic and assume that you will use 25 MB per one php instance. It means you can get only as much as 40 long polled connection scripts.

It's precisely the reason why you could serve potentially a lot more with Node, as node would not spawn its instances (unless you want to use workers etc), so with same memory you could probably get easily to 10k connections hanging. You would get a spike in the CPU as they will come, and when they will potentially be released, but when they are idle it's like they are not there (you pay only for the memory structures you would keep in node/c++).

Websocket

Now if you want to send few things, whenever they are in or out of the client, go for the websockets (ws protocol). First call is size of http request, but later you send just the messages, from the client to server (new questions) and server to client (answers or pushes - can even do broadcast for all connected clients). Some libs, like socket.io have a hierarchy of its own, so when websocket fails, it goes back to long or short polling.

When to use

Short polling - well, never.
Long polling - potentially when you are exchanging single call with the server, and the server is doing some work in background. Also when you won't query the server on the same page anymore. Also when you are not using php as layer to handle the long polled connection (node/c++ can be a simple middle layer). Note long polling can be really beneficial, but only when you make it so.
Websocket - you potentially will exchange more than one or two calls with server, or something might come from server you did not expected / asked, like notification of email or something. You should plan different "rooms", depending on functionalities.

Source: http://stackoverflow.com/questions/4642598/short-polling-vs-long-polling-for-real-time-web-applications

3. What is HTTP streaming, SSE (Server sent events).

Server-sent events (SSE) is a technology where a browser receives automatic updates from a server via HTTP connection. The Server-Sent Events EventSource API is standardized as part of HTML5

Server-sent events is a standard describing how servers can initiate data transmission towards clients once an initial client connection has been established. They are commonly used to send message updates or continuous data streams to a browser client and designed to enhance native, cross-browser streaming through a JavaScript API called EventSource, through which a client requests a particular URL in order to receive an event stream.

Source: https://en.wikipedia.org/wiki/Server-sent_events

4. What is Websocket protocol, how is it different from http communication, what advantages it has over HTTP?

WebSocket is a protocol providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocketAPI in Web IDL is being standardized by the W3C.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. The communications are done over TCP port number 80, which is of benefit for those environments which block non-web Internet connections using a firewall. Similar two-way browser-server communications have been achieved in non-standardized ways using stopgap technologies such as Comet.

The WebSocket protocol is currently supported in most major browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it.

Unlike HTTP, WebSocket provides full-duplex communication. Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. Before WebSocket, port 80 full-duplex communication was attainable using Comet channels; however, Comet implementation is nontrivial, and due to the TCP handshake and HTTP header overhead, it is inefficient for small messages. WebSocket protocol aims to solve these problems without compromising security assumptions of the web.
The WebSocket protocol specification defines `ws` and `wss` as two new uniform resource identifier (URI) schemes that are used for unencrypted and encrypted connections, respectively. Apart from the scheme name and fragment (`#` is not supported), the rest of the URI components are defined to use URI generic syntax.
Using the Google Chrome Developer Tools, developers can inspect the WebSocket handshake as well as the WebSocket frames.

Source: https://en.wikipedia.org/wiki/WebSocket

5. Explain what the Websocket Protocol brings to the web-world.
   See answer in question 4.

6. Explain and demonstrate the process of Websocket communication – from connecting to client, through sending messages, to closing connection.

In order to create a WebSocket protocol in the initial request form the client the server will respond with a "handshake", this keeps a connection ongoing between the server and the client. The server will then respond and receive another request in which it does not have to respond to until something changes within the server or the client. In order to break the connection the client will send a close control frame and the server will respond with the close control frame.

7. What's the advantage of using libraries like Socket.IO, Sock.JS, WS, over pure WebSocket libraries in the backend and standard APIs on frontend? Which problems do they solve?

The advantage of using these libraries is that they simplify the process of running your application via WebSockets, it does this by handling failovers. Meaning that if something does not work in the environment of the application these libraries will provide your application with an automatic way of handling the connection alternatively.

8. What is Backend as a Service, Database as a Service, why would you consider using Firebase in your projects?

BaaS
Backend as a Service, or BaaS (sometimes referred to as mBaaS) is a cloud computing category that consists of companies that make it easier for developers to setup, use and operate a cloud backend for their mobile, tablet and web apps.
Backend as a Service provides web and mobile app developers with a way to link their applications to backend cloud storage while also providing features such as user management, push notifications, and

integration with social networking services. These services are provided through customized Software Development Kits (SDKs) and Application Programming Interfaces (APIs).
Cloud backend as a service platforms enable app developers to store, manage and access their data through such cloud. Moreover, they are able to develop backend features for their apps in minutes, rather than weeks.

Source: http://www.baasbox.com/what-is-backend-as-a-service/

DBaaS
Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data.
DBaaS delivers database functionality similar to what is found in relational database management systems (RDBMSes) such as SQL Server, MySQL and Oracle. Being cloud-based, on the other hand, DBaaS provides a flexible, scalable, on-demand platform that's oriented toward self-service and easy management, particularly in terms of provisioning a business' own environment. DBaaS products typically provide enough monitoring capabilities to track performance and usage and to alert users to potential issues. The products can also generate at least some degree of data analytics.
Disadvantages to the DBaaS model include a lack of control over network performance issues, such as unacceptable latency and application failures. Furthermore, some DBaaS products don't support capabilities of the typical RDBMS, such as data compression and table partitions. Before committing to DBaaS, it's essential to assess your specific requirements and ensure they are satisfactorily addressed.
DBaaS is one of a growing number of examples of cloud-based services including Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) and Software as a Service (SaaS). Databases that are administrated through this model are sometimes referred to as cloud databases.

Source: http://whatis.techtarget.com/definition/Database-as-a-Service-DBaaS

9. Explain the pros & cons of using a Backend as a Service Provider like Firebase.

10. Explain and demonstrate “three-way data binding” using Firebase and Angular.

11. Explain and demonstrate the difference between the simple chat system in your own WebSocket + Node.js backend vs. Firebase.