**What is the difference between HTTP and REST and SOAP in web services?**

SOAP uses WSDL for communication between consumer and provider, whereas REST just uses XML or JSON to send and receive data  
   
WSDL Defines contract between client and service and is static by its nature. In case of REST contract is somewhat complicated and is defined by HTTP, URI, Media Formats and Application Specific Coordination Protocol. It’s highly dynamic unlike WSDL.

SOAP doesn’t return human readable result, while REST result is readable with is just plain XML or JSON

This is not true. Plain XML or JSON are not RESTful at all. None of them define any controls(i.e. links and link relations, method information, encoding information etc…) which is against REST as far as messages must be self contained and coordinate interaction between agent/client and service.

With links + semantic link relations clients should be able to determine what is next interaction step and follow these links and continue communication with service.

It is not necessary that messages be human readable, it’s possible to use cryptic format and build perfectly valid REST applications. It doesn’t matter whether message is human readable or not.

Thus, plain XML(application/xml) or JSON(application/json) are not sufficient formats for building REST applications. It’s always reasonable to use subset of these generic media types which have strong semantic meaning and offer enough control information(links etc…) to coordinate interactions between client and server.

REST is over only HTTP. HTTP is most widely used and when we talk about REST web services we just assume HTTP. HTTP defines interface with it’s methods(GET, POST, PUT, DELETE, PATCH etc) and various headers which can be used uniformly for interacting with resources. This uniformity can be achieved with other protocols as well.

REST permits many different data formats where as SOAP only permits XML.  
While this may seem like it adds complexity to REST because you need to handle multiple formats, in my experience it has actually been quite beneficial. JSON usually is a better fit for data and parses much faster. REST allows better support for browser clients due to it’s support for JSON.

[REST](http://en.wikipedia.org/wiki/Representational_State_Transfer) doesn’t add any specific functionality to HTTP but is an architectural style that was developed alongside HTTP and most commonly uses HTTP for its application layer protocol.

HTTP is an application protocol. REST is a set of rules, that when followed, enable you to build a distributed application that has a specific set of desirable constraints.

If you are looking for the most significant constraints of REST that distinguish a RESTful application from just any HTTP application, I would say the “self-description” constraint and the hypermedia constraint (aka Hypermedia as the Engine of Application State (HATEOAS)) are the most important.

The self-description constraint requires a RESTful request to be completely self descriptive in the users intent. This allows intermediaries (proxies and caches) to act on the message safely.

The HATEOAS constraint is about turning your application into a web of links where the client’s current state is based on its place in that web. It is a tricky concept and requires more time to explain than I have right now.