REST

REST stands for Representational State Transfer and API stands for Application Program Interface.

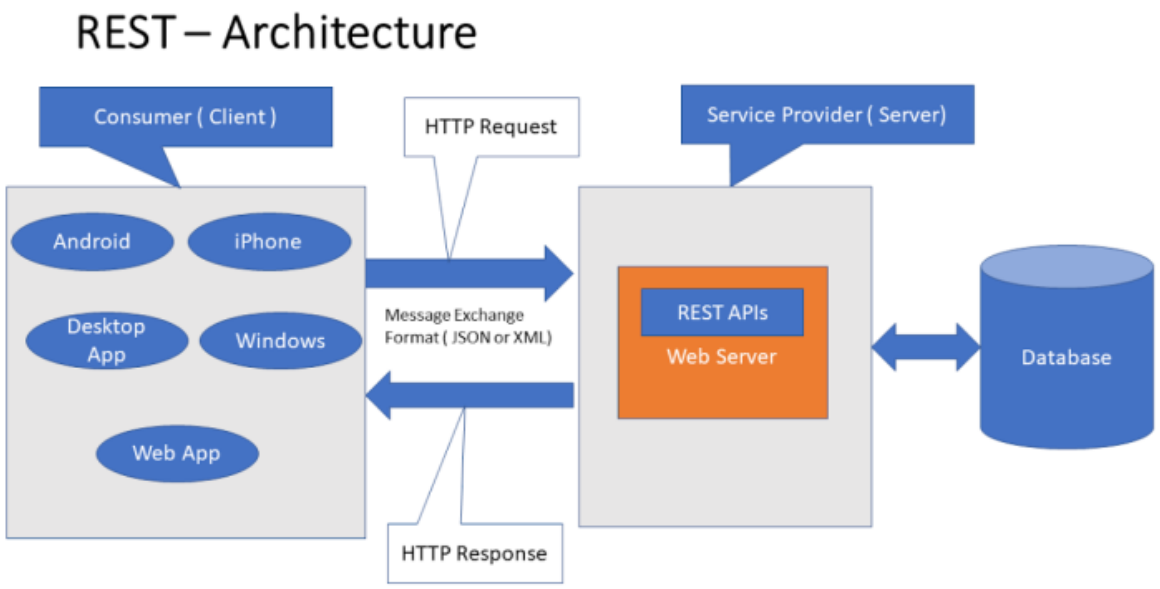
REST is a software architectural style that defines the set of rules to be used for creating web services. Web services which follow the REST architectural style are known as RESTful web services. It allows requesting systems to access and manipulate web resources by using a uniform and predefined set of rules.

Interaction in REST based systems happen through Internet's Hypertext Transfer Protocol (HTTP).

A Restful system consists of a:

- client who requests for the resources.
- server who has the resources.

Rest Architecture:



Architectural Constraints of RESTful API:

There are six architectural constraints which makes any web service are listed below:

- Uniform Interface
- Stateless

- Cacheable
- Client-Server
- Layered System

Http Verbs:

Some of the common HTTP methods/verbs are described below:

HTTP Method	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	Optional	Yes	Yes	Yes	Yes
HEAD	No	No	Yes	Yes	Yes
POST	Yes	Yes	No	No	Yes
PUT	Yes	Yes	No	Yes	No
DELETE	No	Yes	No	Yes	No
CONNECT	Yes	Yes	No	No	No
OPTIONS	Optional	Yes	Yes	Yes	No
TRACE	No	Yes	Yes	Yes	No
PATCH	Yes	Yes	No	No	No

Javascript API:

JavaScript APIs, or Application Programming Interfaces, are sets of rules and protocols that allow different software applications to communicate and interact with each other.

In the context of JavaScript, APIs provide a way to access and manipulate various functionalities and data of web browsers, operating systems, or other services.

Here are some commonly used JavaScript APIs:

1. **Document Object Model (DOM) API**: It provides methods and properties to interact with the HTML structure of a webpage, allowing you to dynamically manipulate the content, style, and structure of the page.

2. **XMLHttpRequest and Fetch API**: These APIs enable making HTTP requests from JavaScript code to retrieve data from a server and handle the responses.

3. **Geolocation API**: It allows obtaining the user's geographic location through the browser, providing information such as latitude and longitude coordinates.

4. **Canvas API**: This API provides a way to dynamically create and manipulate graphics, animations, and images using JavaScript.

5. **Web Storage API**: It enables storing and retrieving data on the client-side, persisting information even after the user navigates away from a webpage. This includes localStorage and sessionStorage.

6. **Web Audio API**: It provides a powerful interface for creating, manipulating, and playing audio within web applications.

7. **Web Speech API**: This API allows integrating speech recognition and synthesis capabilities into web applications, enabling voice-based interactions.

8. **Web Notifications API**: It allows web applications to display system notifications to the user, providing alerts, reminders, or updates even when the application is not in focus.

9. **WebRTC API**: This API enables real-time communication between web browsers, supporting features like video chat, voice calls, and data sharing.

10. **Service Worker API**: It allows developers to create scripts that run independently of the web page, providing background processes, offline caching, and push notifications.

SOAP VS REST:

SOAP	REST
1. Simple Object Access Protocol	1. Representational State Transfer
2. Function-driven (data available as services, e.g.: "getUser")	2. Data-driven (data available as resources, e.g. "user").
3. Message format supported: XML	3. Message format supported: Plain text, HTML, XML, JSON, YAML, etc.
4. Transfer Protocols supported: HTTP, SMTP, UDP, etc.	4. Transfer Protocols supported: HTTP
5. SOAP is recommended for Enterprise apps, financial services, payment gateways	5. REST is recommended for mobile applications and Social networking applications.
6. Highly secure and supports distributed environment.	6. Less secured and not suitable for distributed environment.

Fig. SOAP vs REST

Role of APIS:

1. **Integration**: APIs enable different software systems, applications, or components to seamlessly communicate and share data with each other. They provide a standardized interface and set of rules that allow developers to integrate their applications with existing platforms, services, or databases.

2. **Interoperability**: APIs promote interoperability by providing a common language and structure for different applications to interact. They ensure compatibility and enable systems built on different technologies to work together effectively.

3. **Data Access**: APIs serve as gateways for accessing and retrieving data from various sources, such as databases, web services, or cloud platforms. They provide a secure and controlled way to retrieve specific information or perform operations on the underlying data.

4. **Functionality Extension**: APIs allow developers to extend the functionality of their applications by leveraging services and features provided by other applications or platforms. By integrating with APIs, developers can add capabilities like payment processing, social media sharing, mapping services, or machine learning algorithms without having to build these functionalities from scratch.

5. **Development Efficiency**: APIs simplify software development by providing pre-built components, libraries, and services that developers can utilize. Instead of reinventing the wheel, developers can leverage APIs to access ready-made functionalities, reducing development time and effort.

6. **Ecosystem Development**: APIs facilitate the creation of developer ecosystems around platforms or services. By exposing APIs, companies encourage third-party developers to build applications, plugins, or integrations that extend the capabilities of their offerings. This helps foster innovation, expand the reach of products, and build a vibrant developer community.

7. **Mobile and Web Applications**: APIs are fundamental for mobile and web application development. They enable applications to interact with servers, access data, and perform actions such as user authentication, data synchronization, push notifications, and more.

8. **Microservices Architecture**: APIs are a central component of microservices architecture, where applications are built as a collection of small, independent services. APIs enable these services to communicate and work together, allowing for scalability, flexibility, and easier maintenance.