**REST**

* Stands for Representational State Transfer
* Is an architectural style to develop services using HTTP or HTTPS
* Transfers, accesses and manipulates textual data representations in a stateless manner
* Following textual data representations are available in REST
  + JSON
  + XML
  + HTML
  + Plain text
* Two main building blocks of RESTful web services
  + Resource URI (a RESTful API service is exposed through it)
  + HTTP request method (methods/actions available to interact a resource on server)
* HTTP has a mechanism by which you can ask for information in a different form than that of its default representation. This is by specifying an ‘Accept’ header in the request if the server supports that representation. This is known as content negotiation
* The architectural benefits of REST are the following
  + Performance
  + Generality
  + Scalability
  + Simplicity
  + Modifiability
  + Extensibility

* REST verbs
  + GET
    - Transfers representations of the named resource from server to client
    - Does not modify anything on the server side
    - Idempotent – issuing a request more than once will have same consequence

* + POST
    - Should be used to create a subordinate. Location of the new resource is provided by the server
    - Not idempotent

* + PUT
    - Completely replaces whatever currently exists at the target URI
    - Can create a resource as well
    - Idempotent

* + DELETE
    - Removes a resource at the URI
    - Idempotent
  + HEAD
    - Issue a request for resource without retrieving it
    - Used to check for existence of the resource and discover metadata on it

* + OPTIONS
    - To ask a server about a resource on what other verbs are applicable on it

* + PATCH
    - Standardised way to express partial updates
    - Better to use for partial updates than POST

* HTTP response codes provide us a rich dialog between client and servers about the status of a request. There are five sets as:
  + 1xx Informational
  + 2xx Success
  + 3xx Redirection
  + 4xx Client error
  + 5xx Server error

* Success codes:

| **CODE** | **DESCRIPTION** |
| --- | --- |
| 200 | OK. The request has successfully executed. Response depends upon the verb invoked. |
| 201 | Created. The request has successfully executed and a new resource has been created in the process. The response body is either empty or contains a representation containing URIs for the resource created. The Location header in the response should point to the URI as well. |
| 202 | Accepted. The request was valid and has been accepted but has not yet been processed. The response should include a URI to poll for status updates on the request. This allows asynchronous REST requests |
| 204 | No Content. The request was successfully processed but the server did not have any response. The client should not update its display. |

* Redirection

| **CODE** | **DESCRIPTION** |
| --- | --- |
| 301 | Moved Permanently. The requested resource is no longer located at the specified URL. The new Location should be returned in the response header. Only GET or HEAD requests should redirect to the new location. The client should update its bookmark if possible. |
| 302 | Found. The requested resource has temporarily been found somewhere else. The temporary Location should be returned in the response header. Only GET or HEAD requests should redirect to the new location. The client need not update its bookmark as the resource may return to this URL. |
| 303 | See Other. This response code has been reinterpreted by the W3C Technical Architecture Group (TAG) as a way of responding to a valid request for a non-network addressable resource. This is an important concept in the Semantic Web when we give URIs to people, concepts, organizations, etc. There is a distinction between resources that can be found on the Web and those that cannot. Clients can tell this difference if they get a 303 instead of 200. The redirected location will be reflected in the Location header of the response. This header will contain a reference to a document about the resource or perhaps some metadata about it. |

* Client error

| **CODE** | **DESCRIPTION** |
| --- | --- |
| 405 | Method Not Allowed. |
| 406 | Not Acceptable. |
| 410 | Gone. |
| 411 | Length Required. |
| 412 | Precondition Failed. |
| 413 | Entity Too Large. |
| 414 | URI Too Long. |
| 415 | Unsupported Media Type. |
| 417 | Expectation Failed. |

* Server error

| **CODE** | **DESCRIPTION** |
| --- | --- |
| 500 | Internal Server Error. |
| 501 | Not Implemented. |
| 503 | Service Unavailable. |