# Carl’s Quick REST Cheat Sheet

## Resources

A resource is anything that we present through an API.

## URI, URL & URN

A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource. URIs can be just a scheme and name (URN) or access method, name and location (URL)

A Uniform Resource Locator (URL) is a unique identifier used to locate a resource.

|  |  |  |
| --- | --- | --- |
| URI, URL | | |
|  | URI | |
|  | | URI, URN |
| *Scheme* | *Host* | *Path* |
| **https://** | **visualstudio.microsoft.com** | **/downloads/vs2017.zip** |

## REST

Representational State Transfer (REST) is a name given to an architectural style for transferring a representation (think like ‘View’) of the state (think like ‘data’) of a resource. The state may be transferred by encoding in one of many transferrable formats such as XML, JSON, plain text, CSV, PDF, JPEG, MPEG, etc.

The six ‘guiding’ constraints of a RESTful system are:

### Client-Server

A client-server architecture provides a ‘separation of concerns’ and this allows the components of a system to change independently.

### Stateless

Stateless communication means that each request must contain all of the information necessary to understand and process the request. This improves scalability because no state resources are required to be stored by the server between requests.

### Cacheable

If a resource is cacheable then the system can return a cached response to improve speed and lower network and server utilisation.

### Uniform Interface

A uniform interface simplifies understanding and by using existing mechanisms such as carefully named URIs to identify resources along with representations to manipulate resources the REST style becomes somewhat predictable. The use of HTTP Verbs such as GET, PUT, POST & DELETE and MIME types to describe the transferrable formats provides further uniformity. Then there is HATEOAS – Hypermedia As The Engine Of Application State where hyperlinks are included in a REST response to provide URLs for navigation and useful relationships.

### Layered System

The client communicates with a layer beyond which there may be other layers that are hidden from the client. This layering, just like client-server, helps to separate concerns and provides opportunities for caching and load balancing behind the scenes.

### Code on Demand

This is an ‘optional’ constraint where the client’s functionality can be enhanced/altered by using code that is supplied by the server. As an example a RESTful server’s response to a client’s request could contain a mixture of resource state and behaviour such as the JavaScript code to build a form to display the state and advance the UI.

## RESTful

An system (architecture and API) are said to be RESTful is they meet the non-optional constraints.

If you want to read the famous dissertation that started it all it is here:

**Architectural Styles and the Design of Network-based Software Architectures**

<https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>