What is a REST API?

* REST is a way of thinking about how a web server or API responds to requests, and its behaviour in general
* REST APIs do not just respond with data – they respond with **resources**.

Resources?

* In some ways, resources are analogous to objects in OOP
* Think of the server as having resources, and each is able to interact with the pertinent request
* Suppose we have a web server with an endpoint of “/item/chair”. It accepts the following requests:
  + GET /item/chair
  + POST /item/chair
  + PUT /item/chair
  + DELETE /item/chair
* The “item” is a resource, and “chair” is an element of that resource
* Again, with the OOP analogy – OOP makes things easier to understand by using abstraction, and encapsulating methods and properties of that item within an object. Resources in REST APIs play a similar role to objects in OOP.
* Suppose we have another endpoint “/items” that accepts a “GET /items” request. Rather than dealing with a separate resource, intuitively we can deduce that it still deals with item resources. We can assume that it responds with something like an ItemList resource (a list of items).

Statelessness

* REST APIs are (supposed to be) stateless.
* One request does not affect other requests. The web server does not have an internal state that changes over time and affects responses to requests.
* For example:
  + “POST /item/chair” creates a chair item
  + The item now exists in the database, but the server does not know that the item exists.
  + “GET /item/chair” then goes to the database to see if the item is there
  + To get the item, you do not need to create it with a web request via the server – perhaps it was there previously. Though the data in the database affects the outcome of the GET request, the database is separate from the web server. The web server itself does NOT have state.
* Another example:
  + The user logs in to a web app. The server responds with some data.
  + The server does not know the user is logged in – it has no state, and cannot store that information.
  + So how does the server know which user it is dealing with? Well, when the user logs in, **the server must respond with enough information to identify the user in every request**. Else the server cannot associate the request with the user.