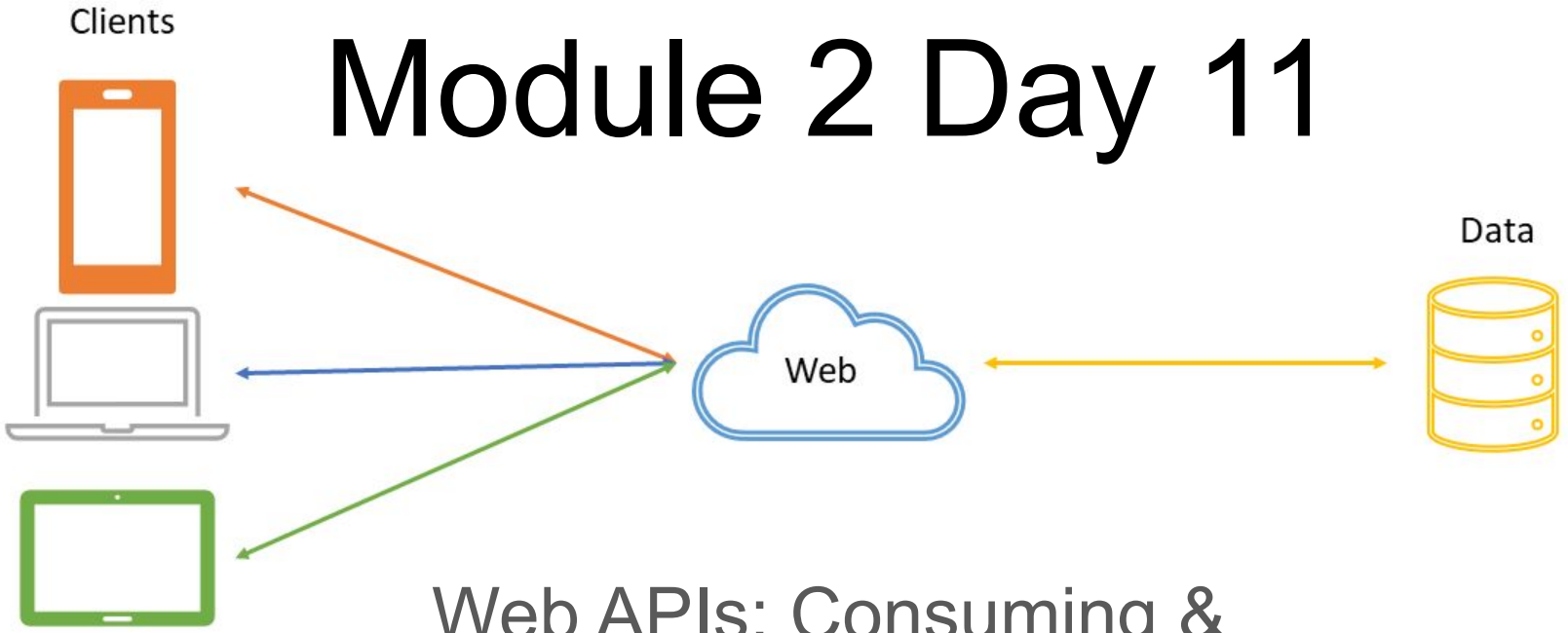
Module 2 Day 11



Web APIs: Consuming & Web Services GET

Module 2 Day 11
Can you ... ?

- Explain the purpose of: IP Addresses, DNS, Ports, HTTP, TLS
- Identify and explain the purpose of the main components of HTTP, including:
  a. Methods, Resources
  b. Headers, Request Body
  c. Status Codes, Request / Response (Stateless)

- Explain the steps of a typical HTTP request between a web browser and a server
- Explain what a GET request is used for
- Explain distinctions between 2xx, 4xx, and 5xx Status Codes
- Make an HTTP GET request using Postman and inspect the result
- Explain what JSON is and how to use it in a Java
- Make an HTTP GET request to a RESTful web service using Java and process the response

Anatomy of an IP URL

Here is a sample URL using an IP number:

https://**127.0.0.1**:**3000**

- **protocol:** others - http, ftp
- **ip address:** This is the unique address of a machine on a network.
- **port**: Number allocated for a specific type of service.
  https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Anatomy of a Named Host URL

Here is another URL that uses hostnames, this is certainly easier to remember than a bunch of numbers.

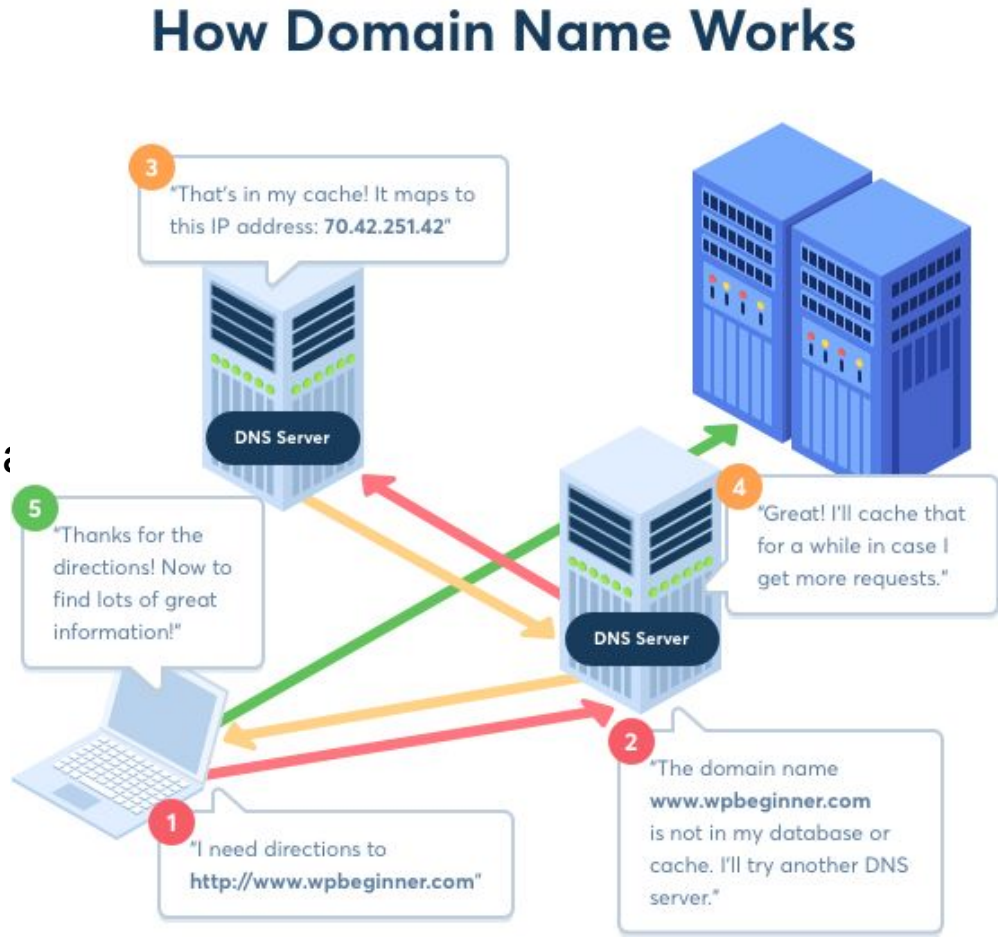
https://**skynet**.**wecomeinpeace.com**

- **host name**: A name assigned to a physical resource connected to a network
- **domain name**: Defines a specific “region of control” on the internet, also, .com is referred to as the top-level domain name.

The above URL is an example of a fully qualified domain name (FQDN).

DNS & DNS Servers

- **DNS** is an acronym for Domain Name System.
- A DNS server is responsible for converting a URL containing huma readable domain names (second example) to one containing an IP address (first example).

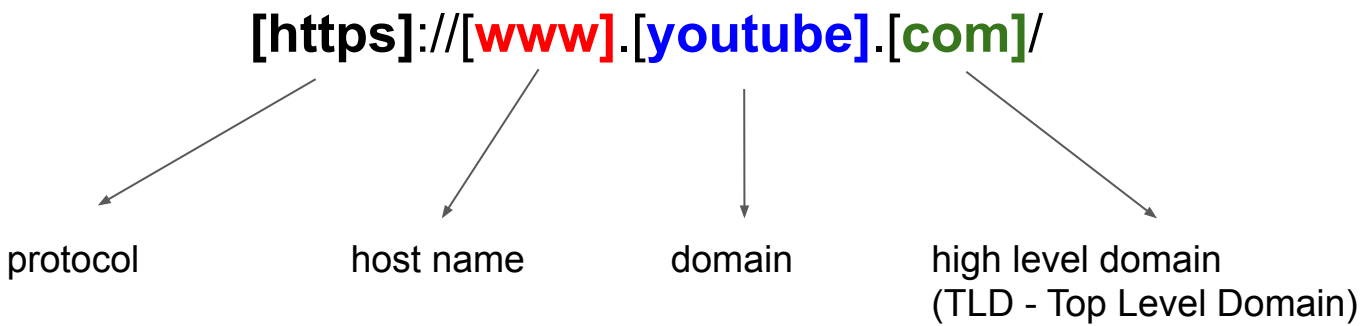


https://www.wpbeginner.com/glossary/dns/

So … www ?

Because I’m sure you’ve wondered…

- On a URL the appearance of www has no bearing on the means by which we are communicating with another machine on the network (the protocol is still http or https).
- www is simply a hostname:



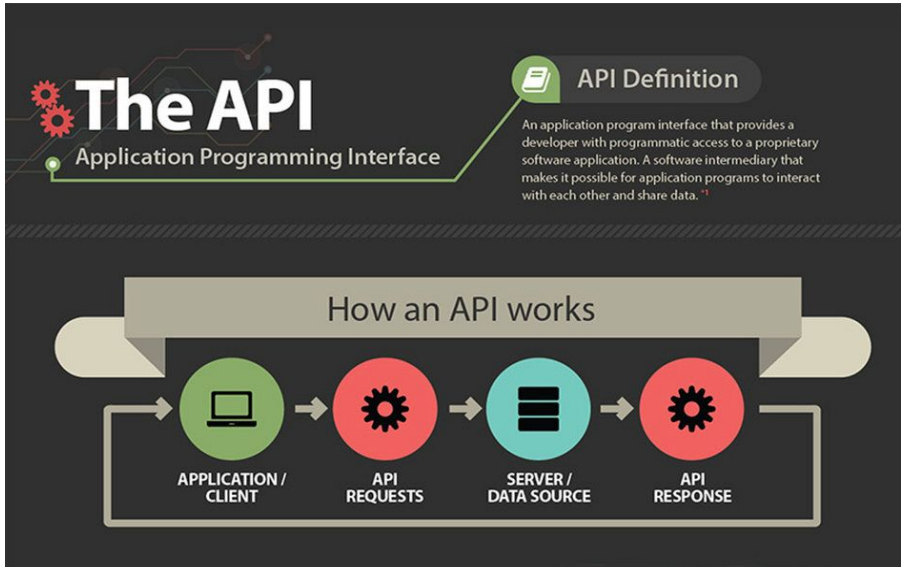
What is an API ? (Application Programming Interface)

- A set of functions and/or procedures designed to interact with an external system.
- Modern cloud architecture relies ***heavily*** on API's.
- *Consuming an API* means interacting with an API's code to produce a desired result.

More information and reading is available here: Introduction to REST based APIs

API’s as a source of data

- In Module 1, we learned about consuming data starting with reading a text file
- In the first half of Module 2, we learned how to building a relational database, like PostgreSQL, and use that to supply an application with data.
- API’s are another source of data for applications to consume and will be the focus of the second half of Module 2.



Request Types to an API

We will see that a REST controller can be configured to handle various types of requests:

- **GET**: Intended to ***retrieve*** the predefined scope of data from a REST endpoint.
- GET (with path variable): path variables (i.e. puppy/1) allow client to retrieve a single record or a desired set of data.
- **POST**: Used for ***inserting*** a new record into the data source.
- **PUT**: Used for ***updating*** an existing record within a data source.
- **DELETE**: Used for ***removing*** an existing record from the data source.

Note: when speaking of APIs we refer to the data as ***Resources*** rather than records or data, as we typically do with database CRUD operations

The Focus of Mod 2 Day 11(Lecture 9) is on GET requests and how they are consumed using Java.

Possible Responses from API

Once a request is made, the REST server can respond with specific status codes:

- **200 or 2xx**: All's well, the request was successful.
- **4XX**: The client (you or your application) has not structured the request correctly. Common examples of these are 400 Bad Request and 401 Unauthorized Request.
- **5XX**: The server has encountered some kind of error. The most common of these is the 500 Internal Server Error message.

Here is the full list: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Making a GET Request through Java

The RestTemplate class provides the means with which we can make a request to an API. Here is an example call:

```
private static final String API_BASE_URL = "http://helpful-site/v1/api/data";
private static RestTemplate restTemplate = new RestTemplate();
MyObj myobj = restTemplate.getForObject(API_BASE_URL, MyObj.class);
```

Note that we can specify the return type of the API call with the second parameter (MyObj.class). Alternatively, if you are getting an array of objects back, we can write the following:

```
MyObj [ ] myobj = restTemplate.getForObject(API_BASE_URL, MyObj[ ].class);
```

Serialize vs Deserialize

- Converting data from a JSON string into an object is called **deserializing**.

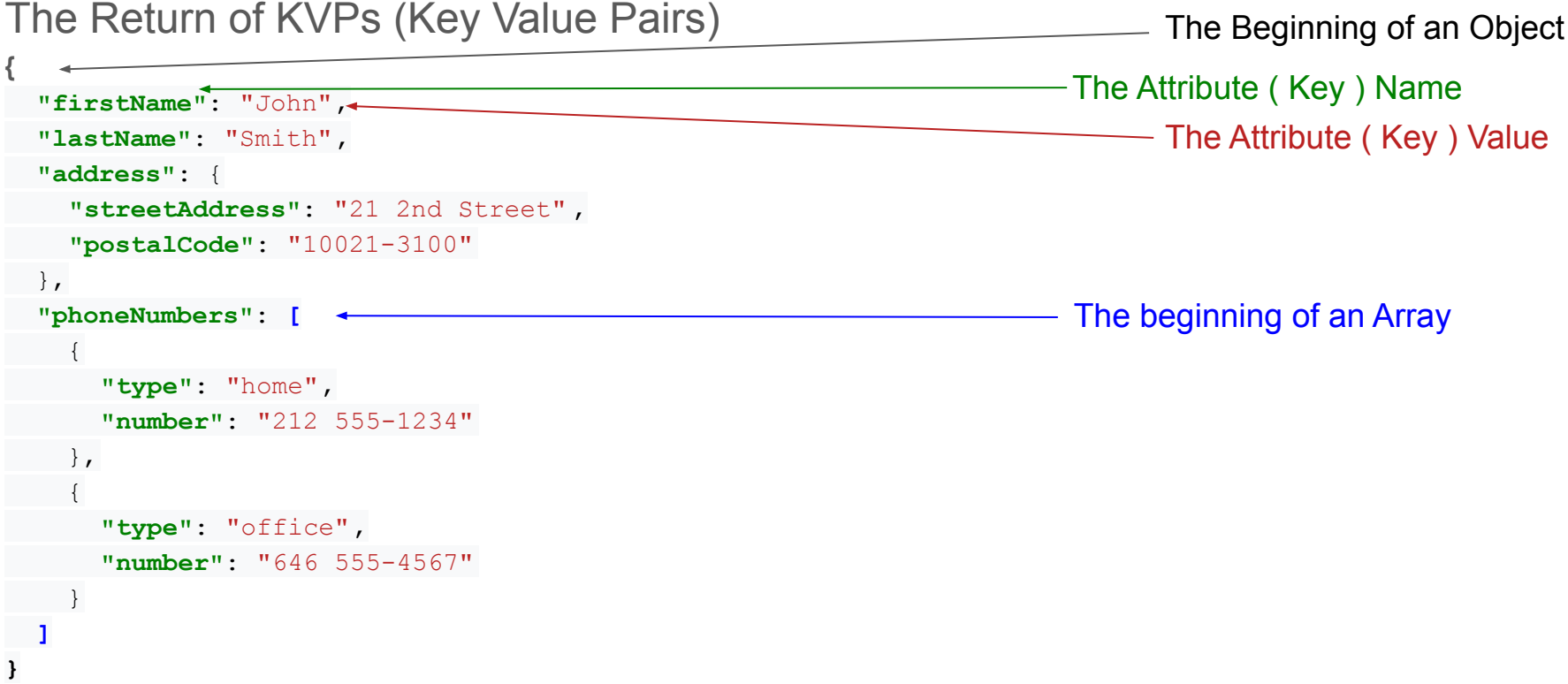
  GET (Today's Code)

- Converting data from an object into a JSON string is called **serializing**, which converts the object into a byte stream used to generate JSON text.

  POST
  PUT

JSON - Javascript Object Notation: RESTful data

The Return of KVPs (Key Value Pairs)




```
{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ]
}
```

XML - Extensible Markup Language

While not as common in web APIs you may encounter it, here is the same example in XML

```xml
<person>
   <address>
      <postalCode>10021-3100</postalCode>
      <streetAddress>21 2nd Street</streetAddress>
   </address>
   <firstName>John</firstName>
   <lastName>Smith</lastName>
   <phoneNumbers>
      <phone>
         <number>212 555-1234</number>
         <type>home</type>
      </phone>
      <phone>
         <number>646 555-4567</number>
         <type>office</type>
      </phone>
   </phoneNumbers>
</person>
```

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ]
}
```

Time to write some GET requests and try out the toolset and code!