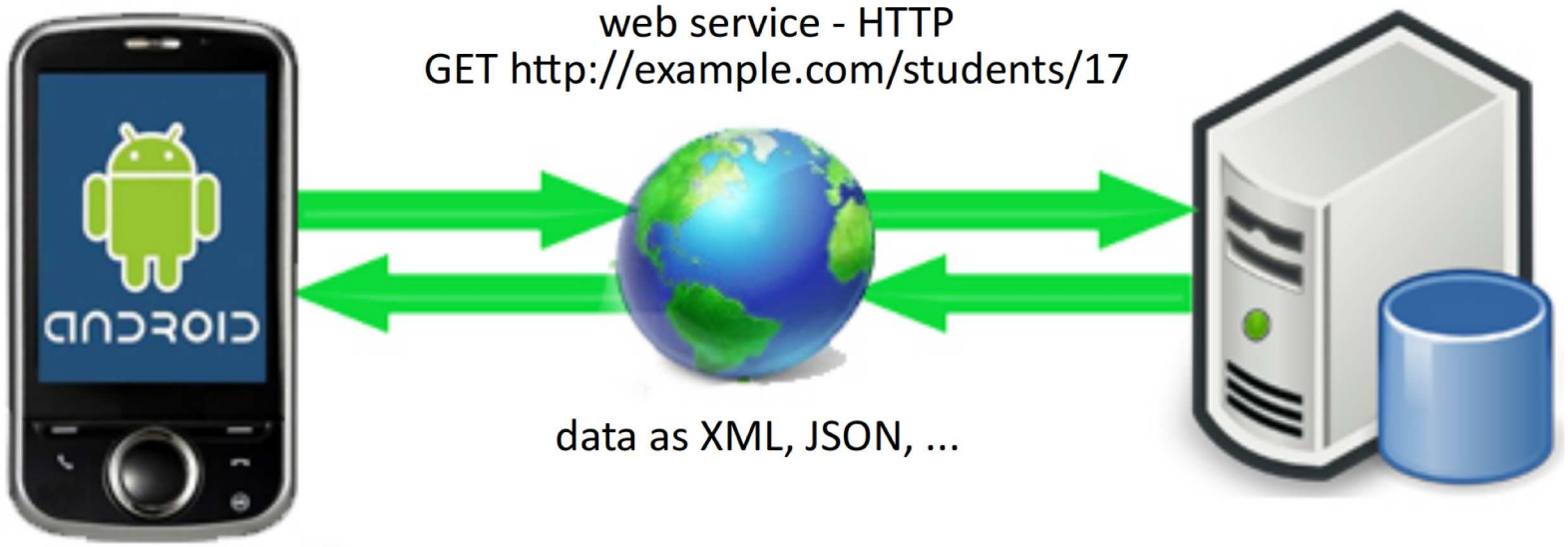
# 16. RESTful Web APIs

# Web services to access data

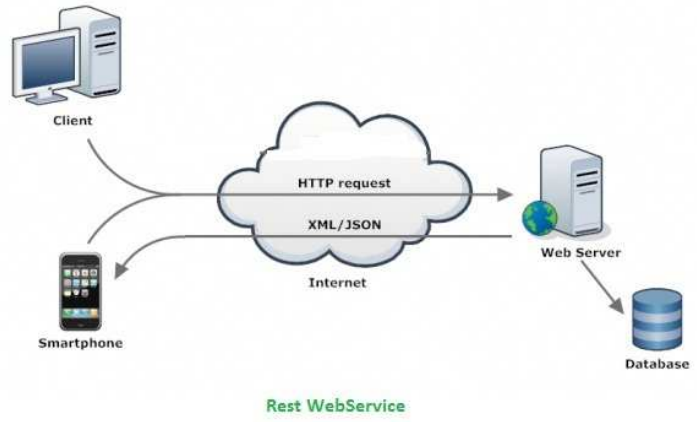
- Many apps access data through a web layer.
- **Client** (app) makes queries by contacting certain specific URLs.
- **Server** (web URL) sends the appropriate database data back.
- Client parses the data, displays it, etc.



# Web services

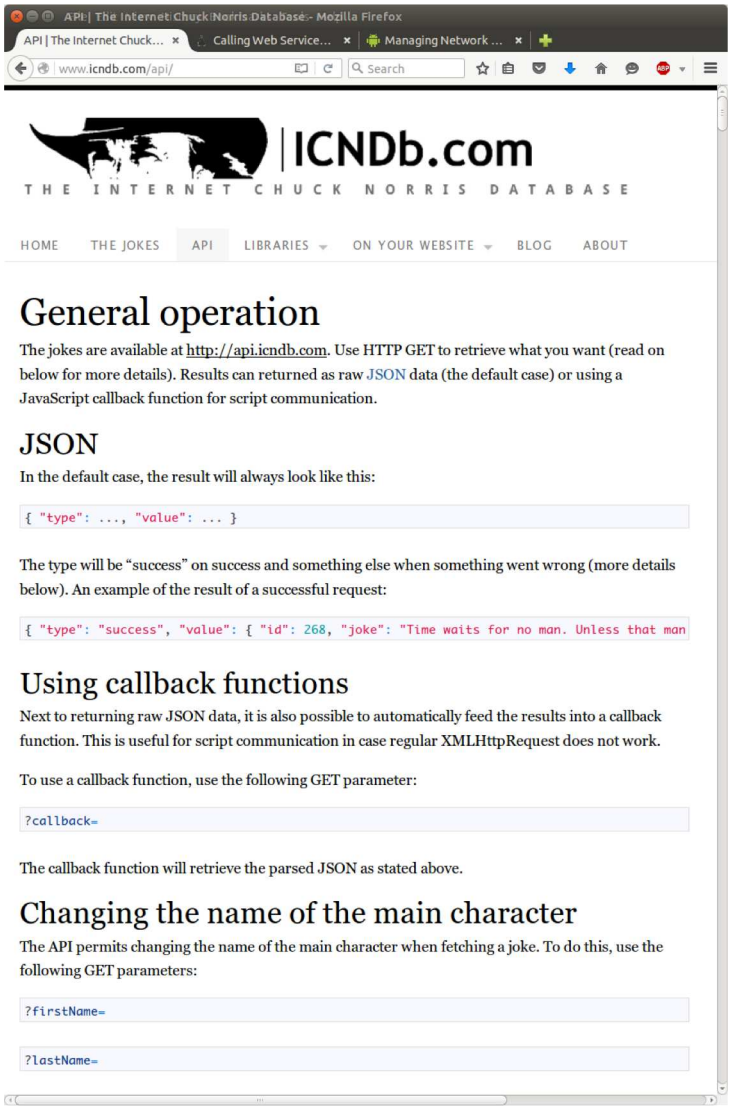
- **web service:** a set of functionality offered over a server using the web, but not web pages / HTMLUse the web's HTTP protocol to connect and transfer data.
- Client connects to specific URLs to request specific data, which is then sent back in some documented format such as XML or JSON.
- **REST:** Representational State Transfer. Common style of web services.
  - "RESTful web services" or "RESTful APIs"

- Web services are a bit like remote function calls where you can request data via URLs with parameters and get the data returned as a response.



# How to find and use web APIs

- **Locate** them online
  - Google for phrases like "*<company>* REST API" or "*<service>* free API"
- **Sign up** for an account
  - Many web APIs require a login or API key
  - Register to receive key or account
- Read the online **documentation** to find out how the API works
  - APIs are not standardized; each one is completely unique
  - Need documentation to learn the available services, parameters, etc.



# Data formats: JSON, XML

- Most web APIs return their data in one of these formats:

- JSON: JavaScript Object Notation
  - Data is a JavaScript object literal.
  - JS objects are basically maps from keys to values.
  - All values in the data are the fields of the object.
  - Object can contain sub-objects, lists, strings, numbers, etc.
  - Slightly less capable than XML, but simpler to read, write, parse.
  - Currently most popular web data interchange format for most apps.


- XML: Extensible Markup Language
  - Data is a nested tree of tags and attributes.
  - More structured, but bulkier/harder to parse.
  - Very popular 5-10 years ago but being superseded by JSON.


- Some web APIs use other data formats:
  - YAML: Yet Another Markup Language. Popular in Ruby/Rails community.
  - plain text

# JSON data example

```
{
        "private": "true",
        "from": "Alice Smith (alice@example.com)",
        "to": [
                "Robert Jones (roberto@example.com)",
                "Charles Dodd (cdodd@example.com)"
        ],
        "subject": "Tomorrow's \"Birthday Bash\" event!",

        "message": {
                "language": "english",
                "text": "Hey guys, don't forget to call me this weekend!"
                }
}
```

# JSON data annotated

↙ {...} = object document

{ ↓ key / ↓ value pairs

"private": "true", ← boolean

"from": "Alice Smith (alice@example.com)", ← string

"to": [ ← [] denotes an array

"Robert Jones (roberto@example.com)", ← array element 0

"Charles Dodd (cdodd@example.com)" ← array element 1

],

"subject": "Tomorrow's \"Birthday Bash\" event!",

"message": { ← {...} = a nested object

"language": "english",

"text": "Hey guys, don't forget to call me this weekend!"

}

}

# Chuck Norris REST API

- fetches random Chuck Norris quotes and "Facts" in JSON format
  - http://www.icndb.com/api/
  - login/key required? **NO**

- API: **http://api.icndb.com/ _____**
  - **/jokes/random** - fetch a random joke
    - { "type": "success", "value": { "id": 194, "joke": "Chuck Norris kicked cancer.", "categories": [] } }
  - **/jokes/random/***N* - fetch multiple random jokes
    - { "type": "success", "value": [ { "id": 417, "joke": "..." }, { "id": 505, "joke": "...", "categories": ["nerdy"] }, { "id": 291, "joke": "...", "categories": [] } ] }
  - **/jokes/random/limitTo=[***categories***]** - limit categories of joke
  - **/jokes/random/exclude=[***categories***]** - exclude categories of joke
  - **/jokes/***N* - fetch a specific joke with ID #N
    - { "type": "success", "value": { "id": 194, "joke": "Chuck Norris kicked cancer.", "categories": [] } }
  - **/jokes/count** - fetch total number of jokes
    - { "type": "success", "value": 549 }
  - **/categories** - fetch names of all categories of jokes
    - { "type": "success", "value": [ "nerdy", "explicit", "chuck norris", "bruce schneier" ] }

# Parsing JSON data

```
{ "private": "true",
        "from": "Alice (alice@ex.com)",
        "subject": "Today's event",
        "to": [
        "Robert (roberto@ex.com)",
        "Charles (cdodd@ex.com)"
        ],
        "message": {
        "lang": "english",
         "text": "Call this weekend!"
        }
}
```

```
private void processData(String data) {
try {
// extract the information from JSON data
JSONObject json = new JSONObject(data);
→ boolean private = json.getBoolean("private");
→ String from = json.getString("from");
→ String subject = json.getString("subject");
→ JSONArray a = json.getJSONArray("to");
→ String to1 = a.getString(0);
→ String to2 = a.getString(1);

→ JSONObject msg =
→ json.getJSONObject("message");
→ String lang = msg.getString("lang");
String text = msg.getString("text");
} catch (JSONException e)
        { Log.wtf("json", e);
}
}
```

# JSONObject methods

| Method | Description |
|---|---|
| *j*.get("*key*")<br>*j*.getBoolean("*key*")<br>*j*.getDouble("*key*")<br>*j*.getInt("*key*")<br>*j*.getJSONArray("*key*")<br>*j*.getJSONObject("*key*")<br>*j*.getLong("*key*")<br>*j*.getString("*key*") | retrieve value of the given key, or throw a JSONException if key is not found |
| *j*.has("*key*") | return true if given key maps to a value |
| *j*.isNull("*key*") | return true if given key maps to null |
| *j*.keys() | return iterator of all keys in object |
| *j*.opt("*key*", *default*)<br>*j*.optBoolean("*key*", *default*)<br>... | retrieve value of the given key, or return default if key is not found |
| *j*.put("*key*", *value*); | set the value for a given key |
| *j*.remove("*key*"); | removes key/value mapping if it exists |
| *j*.toString()<br>*j*.toString(*spaces*) | convert JSON object to a string, with optional indentation and spacing |
| JSONObject.quote(*str*) | encodes data as a JSON string |

# Fetching web data, Ion library

```
// fetch REST API data in background with Ion library
public void fetchData(String urlString) {
        Ion.with(context)
                .load("urlString")
                .asString()
                .setCallback(new FutureCallback<String>() {
                        public void onCompleted(Exception e,
                                                String data) {
                        // process the data or error
                                JSONObject json = new JSONObject(data);
                                processData(json); // you write this!
                        }
                });
}
```